5 HEWのヘッダファイルを利用する

前の章では、I/O レジスタに値を設定するために、

```
#define PCR8 (*((volatile unsigned char *)0xFFEB))
```

などを定義しました。このように自分で必要なレジスタを定義すればよいのですが、HEW では I/O レジスタをすべて定義した、iodefine.h というファイルが用意されています。

5.1 LED0 の点滅 (構造体を用いたレジスタの定義)

この章ではこのファイル (iodefine.h) の内容を理解するために、構造体、ビットフィールド、共用体などを復習しながら、読み解いていきたいと思います。

iodefine.h では、unsigned char 型のビットフィールドも定義しているために、他の環境では使えない可能性があります。移植性を考慮するのであれば、他の環境に移植する際に容易に修正ができるように注意するべきでしょう。

iodefine.h は比較的込み入った設定をしています。いきなりこれを使うサンプルを提示するのではなく、段階をおってやっていきたいと思います。

まずは構造体を用いてレジスタを定義してみましょう。

5.1.1 構造体の簡単な復習

構造体の機能は、既存のデータ型、例えば char 型や int 型などを集めて、新しい型を作りだすことでした。

LED の点灯消灯をプログラムしようとすると、ポートコントロールレジスタ 8(PCR8) とポートデータレジスタ 8(PDR8) の操作が必要でした。これらのアドレスは P.92 の表 4.2 で提示しています。

ここではこの二つのレジスタを一つの構造体にまとめることにします。

ポートコントロールレジスタ 8(PCR8) のアドレスは H'FFEB、ポートデータレジスタ 8(PDR8) のアドレスは H'FFDB で、いずれも 1 バイトのレジスタです。この間には 15 バイト分のレジスタがあること

になります。ハードウェアマニュアルを確認すると、この 15 バイトのレジスタの中には、他のポートのコントロールレジスタやデータレジスタがあることが分かります。ここではこれらのレジスタは利用しないので適当な名前をつけておくことにしましょう。

ポート 8 関連レジスタを、以下のような構造体を宣言して一つにまとめます。

ポート 8 関連レジスタの構造体

```
1    struct st_io {                 /* struct IO    */
2                 unsigned char     PDR8;                  /* PDR8         */
3                 char              wk[15];                /*              */
4                 unsigned char     PCR8;                  /* PCR8         */
5    };
```

st_io 型の構造体の解説

st_io 型の構造体の宣言です。

メンバ PDR8 は unsigned char 型、メンバ wk は要素数 15 の char 型の配列、メンバ PCR8 は unsigned char 型です。

図 5.1 にポート 8 関連レジスタの構造体の構成を図示してみました。

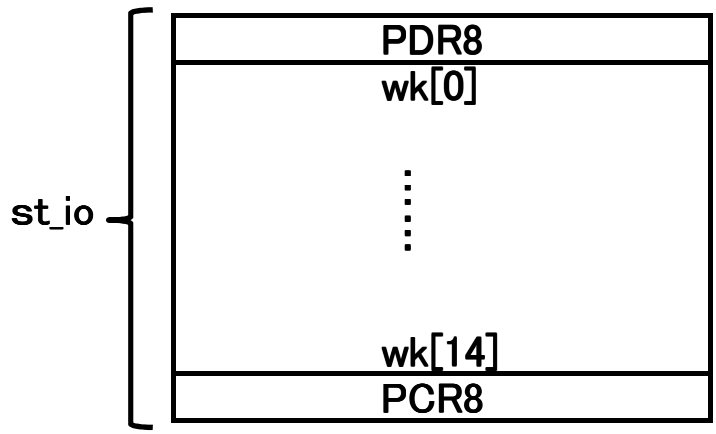

図 5.1　ポート 8 関連レジスタの構造体

この st_io 型を用いれば、ポートデータレジスタ 8(PDR8) は、

```
(*((volatile struct st_io *)0xFFDB)).PDR8
```

と書けます (図 5.2)。

同様に、ポートコントロールレジスタ 8(PCR8) は、

```
(*((volatile struct st_io *)0xFFDB)).PCR8
```

と書けます (図 5.2)。

```
#define IO (*((volatile struct st_io *)0xFFDB))
```

と定義しておけば、それぞれ

```
IO.PDR8
IO.PCR8
```

と書けることになります。

このように記述できると、これらが I/O ポートのレジスタであることがはっきりして、プログラムの見通しが良くなるのではないでしょうか。

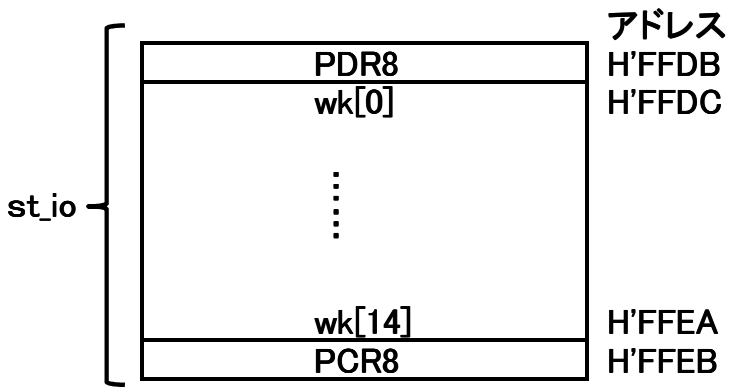

図 5.2　ポート 8 関連レジスタの構造体をレジスタ上に配置

01_LED10.c までで、LED 用の関数を作りファイルを分割しましたが、構造体などここで説明する機能が分かりやすいように、再度関数化を行わないプログラムから始めることにしましょう。

なお、LED の回路図は P.91 の図 4.2 と同じものを使うことにします。

01_LED11.c

```
 1  /************************************************************************/
 2  /*                                                                      */
 3  /*  FILE        :01_LED11.c                                             */
 4  /*  DESCRIPTION :LED0 の点滅 (構造体を用いたレジスタの定義)             */
 5  /*  CPU TYPE    :H8/3694F                                               */
 6  /*                                                                      */
 7  /*  LED0 P85                                                            */
 8  /*  LED1 P86                                                            */
 9  /*  LED2 P87                                                            */
10  /*                                                                      */
11  /*  This file is generated by Renesas Project Generator (Ver.4.9).      */
12  /*                                                                      */
13  /************************************************************************/
14
```

```
15  struct st_io {                  /* struct IO    */
16                unsigned char     PDR8;                   /* PDR8         */
17                char              wk[15];                 /*              */
18                unsigned char     PCR8;                   /* PCR8         */
19  };
20
21  #define IO  (*((volatile struct st_io *)0xFFDB)) /* P8    Address*/
22
23  #define LED0 0x01
24  #define LED1 0x02
25  #define LED2 0x04
26
27  #define LED_SHIFT 5
28  #define LED_MASK ((LED0 | LED1 | LED2) << LED_SHIFT)
29
30  #define WAIT_LOOP 0x3FFFFFL
31
32  void WaitLoop(void)
33  {
34    unsigned long count;
35
36    for(count=0; count<WAIT_LOOP; count++){
37      ;
38    }
39  }
40
41  void main(void)
42  {
43    /*********************************************
44      LED 接続端子の初期化
45    *********************************************/
46    IO.PCR8 |= LED_MASK;
47    IO.PDR8 |= LED_MASK;
48
49    while(1){
50      WaitLoop();
51
52      /*********************************************
53        LED0 を点滅
54      *********************************************/
55      IO.PDR8 ^= (0x01 << LED_SHIFT);
56
57    }
58  }
```

End Of List

実行結果

初期状態はすべての LED が消灯。
LED0 が点滅する。

プログラム解説 (01_LED11.c)

```
15    struct st_io {                  /* struct IO    */
16                  unsigned char     PDR8;                   /* PDR8         */
17                  char              wk[15];                 /*              */
18                  unsigned char     PCR8;                   /* PCR8         */
19    };
```

すでに説明したように、st_io 型の構造体の宣言です。

メンバ wk は、プログラムの中では利用しません。PDR8 と PCR8 の間の 15 バイト分のレジスタを定義するのに利用しています。

ここで行っているのは型宣言であって、変数の宣言ではありません。つまり、int 型や char 型のような新しい型を定義しただけです。利用に際しては、この型の変数を宣言するなどして、メモリ上に領域を確保しなくてはなりません。

今の場合には、アドレスをキャストして、メモリ上に領域を確保します。

```
21    #define IO  (*((volatile struct st_io *)0xFFDB)) /* IO  Address*/
```

すでに説明したように、このような定義を行っておくと、ポートデータレジスタ 8(PDR8) とポートコントロールレジスタ 8(PCR8) は、

```
IO.PDR8
IO.PCR8
```

と書けることになります。

```
46    IO.PCR8 |= LED_MASK;
47    IO.PDR8 |= LED_MASK;
```

ポートコントロールレジスタ 8(IO.PCR8) とポートデータレジスタ 8(IO.PDR8) を初期化しています。ともに上位 3 ビットのみを 1 に設定しています。

ポート 8 の LED がつながっている 3 ビットを出力に設定し、LED を消灯するように設定しています。

```
55    IO.PDR8 ^= (LED0 << LED_SHIFT);
```

LED0 が点滅するように、排他的論理和 (XOR) を用いてレジスタの値を反転させています。

LED0 を指定するのに LED0 をシフトしています。LED0 は 23 行目で定義されていて、値 0x01 のことです。

```
(LED0 << LED_SHIFT);
```

は毎回シフトしているので、これにあらたに名前をつけてもよいかもしれません。

5.2 LED1 の点滅 (ビットフィールドを用いたレジスタの定義)

第 2 章ではビット演算を説明しました。P.47 の 2.6 でおこなったビット演算の利用例では、必要なビットのみを設定し、他のビットは変えない演算について説明をしました。少々複雑で使いこなすには努力が必要かもしれません。

このような演算が必要な理由は、レジスタが 1 バイト単位でしかアクセスできないからです。もしも 1 ビット単位でアクセスできるのであれば、必要なビットだけ直接変更することが可能なわけです。

じつはビットフィールドを用ると 1 ビット単位でアクセスできるようになります。それならばビットフィールドを用れば第 2 章のビット演算は不要かというとそうでもありません。

後ほど説明するように、char 型や unsigned char 型に対してはビットフィールドを利用することはできません。HEW の場合には独自の拡張を行ってこれを利用できるようにしています。したがって、これを他のコンパイラにかけたときにエラーが出る可能性があります。移植性を考慮するのなら充分に配慮して使う必要があるでしょう。

また、必ずしも 1 ビット単位で扱えるほうが便利というわけでもありません。変更するビット数が多いと、かえって 1 バイト単位で扱ったほうが見やすくなることも少なくないのです。

5.2.1 ビットフィールドの簡単な復習

ビットフィールドの宣言は構造体と同様に行います。メンバはビット数の指定付きで宣言します。

ビットフィールドのメンバは以下のように記述します。

型 名前 : サイズ;

メンバのデータ型は int, signed int, unsigned int のいずれかです。

名前は必ずしもつける必要はありません。サイズはビット数を記述します。

ビットフィールドの宣言の例をあげてみましょう。

ビットフィールドの宣言例

```
struct bitflg{
    unsigned int HB  : 8;
    unsigned int B7  : 1;
    unsigned int B6  : 1;
    unsigned int B5  : 1;
    unsigned int B4  : 1;
    unsigned int B3  : 1;
    unsigned int B2  : 1;
    unsigned int B1  : 1;
    unsigned int B0  : 1;
};
```

ビットフィールドの解説

struct bitflg というビットフィールドを宣言しています。

上位 8 ビットには HB という名前をつけています。HB というメンバ名を用いると、8 ビットを扱うことができるわけです (図 5.3)。

同様に次のビットのメンバ名は B7 で、扱えるのは 1 ビットです。B6 から B0 もそれぞれ 1 ビットの大きさになります (図 5.3)。

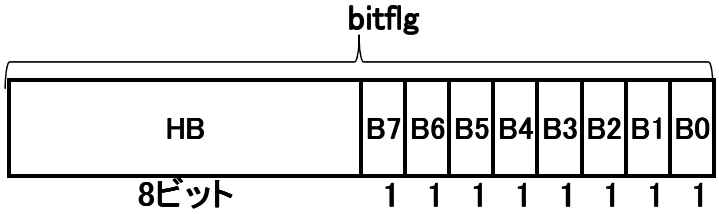

図 5.3　ビットフィールドの宣言

このように、ビットフィールドを用いると、1 ビットずつ扱えるようになるのですが、上記のように C 言語ではメンバのデータ型は int, signed int, unsigned int のいずれかしか使えません。HEW の場合には他の整数型 (例えば unsigned char 型) も許されています。したがって HEW で unsigned char 型のメンバを宣言した場合には、他のコンパイラでは利用できない可能性があります。そのことを理解したうえで利用してください。

なお、HEW では、ビットフィールドは上位ビットから割り付けられます。

では、ポートデータレジスタ 8(PDR8) を 1 ビット単位で利用できるように、ビットフィールドを使ってみましょう。

ポートデータレジスタ 8(PDR8) にビットフィールドを宣言

```
struct st_pdr8{
    unsigned char B7  : 1;
    unsigned char B6  : 1;
    unsigned char B5  : 1;
    unsigned char B4  : 1;
    unsigned char B3  : 1;
    unsigned char B2  : 1;
    unsigned char B1  : 1;
    unsigned char B0  : 1;
};

#define PDR8  (*((volatile struct st_p8cr *)0xFFDB)) /* PDR8  Address*/
```

このように宣言すると、ポートデータレジスタ 8(PDR8) は図 5.4 のように配置されます。このとき例えば最下位ビットは、PDR8.B0 で指定することができます。

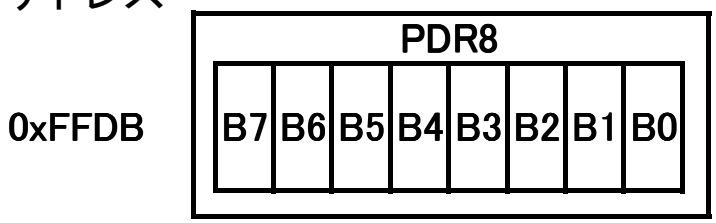

図 5.4　ビットフィールドの宣言

01_LED12.c

```
 1  /***********************************************************************/
 2  /*                                                                     */
 3  /*  FILE        :01_LED12.c                                            */
 4  /*  DESCRIPTION :LED1 の点滅 (ビットフィールドを用いたレジスタの定義)    */
 5  /*  CPU TYPE    :H8/3694F                                              */
 6  /*                                                                     */
 7  /*  LED0 P85                                                           */
 8  /*  LED1 P86                                                           */
 9  /*  LED2 P87                                                           */
10  /*                                                                     */
11  /*  This file is generated by Renesas Project Generator (Ver.4.9).     */
12  /*                                                                     */
13  /***********************************************************************/
14
15  #define PCR8 (*((volatile unsigned char *)0xFFEB)) /* ポートコントロールレジスタ 8(PCR8) */
16
17  struct st_p8dr{                /*  Bit  Access */
18          unsigned char B7:1;    /*    Bit 7     */
19          unsigned char B6:1;    /*    Bit 6     */
20          unsigned char B5:1;    /*    Bit 5     */
21          unsigned char B4:1;    /*    Bit 4     */
22          unsigned char B3:1;    /*    Bit 3     */
23          unsigned char B2:1;    /*    Bit 2     */
24          unsigned char B1:1;    /*    Bit 1     */
25          unsigned char B0:1;    /*    Bit 0     */
26  };
27
28  #define PDR8  (*((volatile struct st_p8dr *)0xFFDB)) /* PDR8  Address*/
29
30  #define LED0 0x01
31  #define LED1 0x02
32  #define LED2 0x04
33
34  #define LED_SHIFT 5
35  #define LED_MASK ((LED0 | LED1 | LED2) << LED_SHIFT)
36
37  #define WAIT_LOOP 0x3FFFFFL
38
39  void WaitLoop(void)
40  {
41    unsigned long count;
42
43    for(count=0; count<WAIT_LOOP; count++){
44      ;
45    }
46  }
47
48  void main(void)
49  {
50
51    /*********************************************
52      LED 接続端子の初期化
53    *********************************************/
54    PCR8 |= LED_MASK;
55    PDR8.B5 = 1; /* LED0 OFF */
56    PDR8.B6 = 0; /* LED1 ON  */
57    PDR8.B7 = 1; /* LED2 OFF */
58
59    while(1){
60      WaitLoop();
61
62      /*********************************************
63        LED1 を点滅
64      *********************************************/
65      PDR8.B6 = (! PDR8.B6);
```

```
66    }
67  }
```

End Of List

実行結果

初期状態は LED1 のみ点灯、LED0 と LED2 は消灯。
LED1 が点滅する。

プログラム解説 (01_LED12.c)

```
15    #define PCR8 (*((volatile unsigned char *)0xFFEB)) /* ポートコントロールレジスタ 8(PCR8) */
```

今回はポートコントロールレジスタ 8(PCR8) とポートデータレジスタ 8(PDR8) は別々の名前をつけることにします。ポートコントロールレジスタ 8(PCR8) は 1 バイト全体を PCR8 という名前をつけました。これは 01_LED10 までで定義していたものと同じです。

```
17    struct st_p8cr{                /*  Bit  Access */
18            unsigned char B7:1;    /*    Bit 7     */
19            unsigned char B6:1;    /*    Bit 6     */
20            unsigned char B5:1;    /*    Bit 5     */
21            unsigned char B4:1;    /*    Bit 4     */
22            unsigned char B3:1;    /*    Bit 3     */
23            unsigned char B2:1;    /*    Bit 2     */
24            unsigned char B1:1;    /*    Bit 1     */
25            unsigned char B0:1;    /*    Bit 0     */
26    };
27
28    #define PDR8  (*((volatile struct st_p8cr *)0xFFDB)) /* PDR8  Address*/
```

P.135 で説明したように、ポートデータレジスタ 8(PDR8) を 1 ビット単位で利用できるように、ビットフィールドを使用しています。

ここでは、8 ビットすべてに名前をつけていますが、LED が接続されているのは上位 3 ビットなので、それ以外は名前をつける必要はありません。

HEW は上位ビットから割り付けますから、

```
struct st_pdr8{
    unsigned char B7  : 1;
    unsigned char B6  : 1;
    unsigned char B5  : 1;
};
```

というように、上位 3 ビットだけに名前をつけても良いわけです。

```
struct st_pdr8{
    unsigned char B7  : 1;
    unsigned char B6  : 1;
    unsigned char B5  : 1;
    unsigned char     : 5;
};
```

のように、名前をつけずに宣言することも可能です。

また、上位 3 ビットに一つの名前をつけることも可能です。

```
struct st_pdr8{
    unsigned char LED  : 3;
};
```

```
55    PDR8.B5 = 1; /* LED0 OFF */
56    PDR8.B6 = 0; /* LED1 ON  */
57    PDR8.B7 = 1; /* LED2 OFF */
```

ポートデータレジスタ 8(PDR8) の上位 3 ビットに 1 ビットずつアクセスしています。

1 が消灯、0 が点灯ですから、LED1 のみが点灯し、LED0 と LED2 は消灯になります。

ビットごとにアクセスできるのはよいのですが、3 ビット分操作しようとすると 3 行プログラムを書かなくてはいけません。5 ビット、6 ビットと増えると、1 ビットずつ操作していたのではかえって面倒です。

状況に合わせてビット単位でもバイト単位でも扱えると更に便利です。

ビット単位でもバイト単位でも扱えるようにするには、共用体を用います。共用体については次で説明します。

```
65    PDR8.B6 = (! PDR8.B6);
```

LED1 が接続されているポートデータレジスタ 8(PDR8) の B6 の値を反転させています。

1 と排他的論理和を取るなど、他の方法を使うこともできますが、ここでは NOT を使ってみました。

✍ 課題 5.2.1　(提出) 3 ビット一度に設定する (e01_LED12)

LED のレジスタ 3 ビット分を一度に設定できるように、ビットフィールドを設定しましょう。

5.3　LED2 の点滅 (共用体を用いたレジスタの定義)

ビットフィールドを利用すると、ビットごとにアクセスできるのはよいのですが、プログラムがかえって煩雑になることもあります。たとえば、1 ビット単位で利用できるように宣言をした場合、6 ビット分操作しようとすると 6 行プログラムを書かなくてはいけません。

ビット単位で操作できるとともに、バイト単位でも操作できるとより便利です。

そのためには、メモリの同じ領域に二通りの名前をつける必要があります。

これを実現するのが共用体です。

共用体はメモリ上にメンバを重ねて確保します。一方、構造体はメモリ上にメンバを順番に並べて確保します。

具体的な例で両者の違いをみていきましょう。

構造体

```
struct st_pdr8{
    unsigned short WORD;
    unsigned char  BYTE;
};

#define PDR8  (*((volatile struct st_p8cr *)0xFFDB)) /* PDR8  Address*/
```

このような宣言を行ったとき、PDR8.WORD はアドレス 0xFFDB と 0xFFDC の 2 バイト分に対応し、PDR8.BYTE はその後の、アドレス 0xFFDD の 1 バイト分に対応します (図 5.5)。PDR8.BYTE に値を代入しても、PDR8.WORD は変更されません。

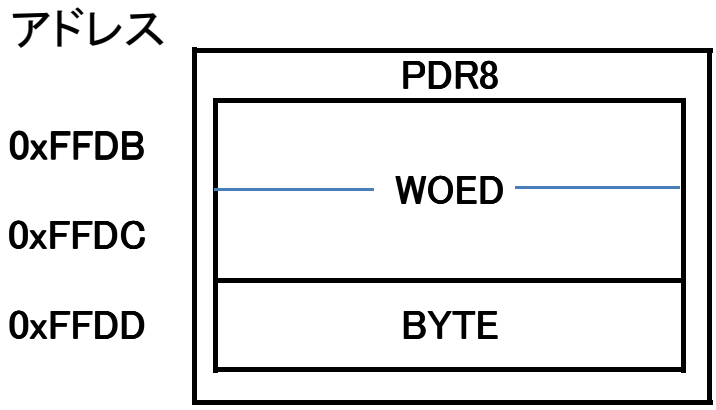

図 5.5　構造体の配置

共用体

```
union st_pdr8{
    unsigned short WORD;
    unsigned char  BYTE;
};

#define PDR8  (*((volatile union st_p8cr *)0xFFDB)) /* PDR8  Address*/
```

このような宣言を行ったとき、PDR8.WORD はアドレス 0xFFDB と 0xFFDC の 2 バイト分に対応し、PDR8.BYTE はアドレス 0xFFDB の 1 バイト分に対応します (図 5.6)。つまり、アドレス 0xFFDB の 1 バイトは PDR8.WORD と PDR8.BYTE の両方で指定することができるわけです。PDR8.BYTE に値を代入すると、PDR8.WORD は変更されます。

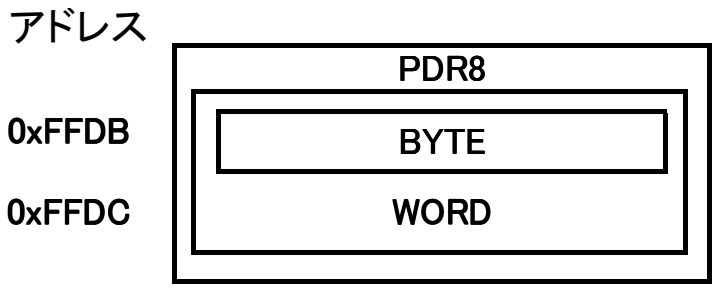

図 5.6　共用体の配置

たとえば、PDR8.WORD の値が 0x0000 だったときに、PDR8.BYTE に 0xFF を代入すると、PDR8.WORD の値は 0xFF00 になります (図 5.7)。これは、PDR8.WORD の上位バイトと、PDR8.BYTE が同じメモリを意味しているからです。

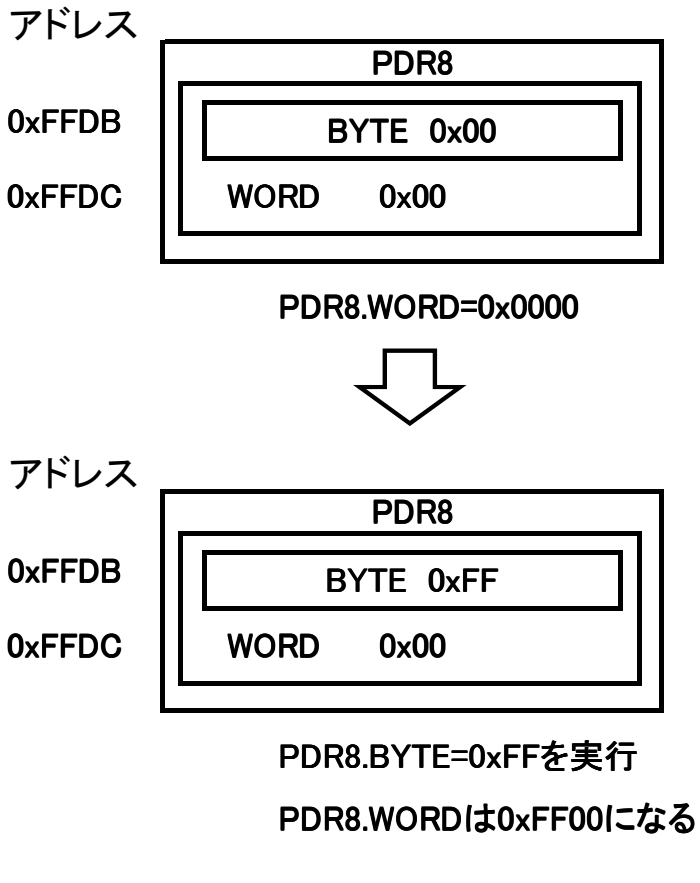

図 5.7　共用体にデータを入力

このことから、共用体とビットフィールドを用いると、1 バイト単位でも 1 ビット単位でもメモリを操作できるようになります。

ポートデータレジスタ 8(PDR8) に対してそのような宣言をしてみましょう。

01_LED13.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :01_LED13.c                                             */
/*  DESCRIPTION :LED2 の点滅 (共用体を用いたレジスタの定義)             */
/*  CPU TYPE    :H8/3694F                                               */
/*                                                                      */
/*  LED0 P85                                                            */
/*  LED1 P86                                                            */
/*  LED2 P87                                                            */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/

#define PCR8 (*((volatile unsigned char *)0xFFEB)) /* ポートコントロールレジスタ 8(PCR8) */

union uni_p8dr{              /* PDR8         */
    unsigned char BYTE;      /*  Byte Access */
    struct {                 /*  Bit  Access */
        unsigned char B7:1;  /*    Bit 7     */
        unsigned char B6:1;  /*    Bit 6     */
        unsigned char B5:1;  /*    Bit 5     */
        unsigned char B4:1;  /*    Bit 4     */
        unsigned char B3:1;  /*    Bit 3     */
        unsigned char B2:1;  /*    Bit 2     */
        unsigned char B1:1;  /*    Bit 1     */
        unsigned char B0:1;  /*    Bit 0     */
     }BIT;
};

#define PDR8  (*((volatile union uni_p8dr *)0xFFDB)) /* PDR8  Address*/

#define LED0 0x01
#define LED1 0x02
#define LED2 0x04

#define LED_SHIFT 5
#define LED_MASK ((LED0 | LED1 | LED2) <<LED_SHIFT)

#define WAIT_LOOP 0x3FFFFFL

void WaitLoop(void)
{
  unsigned long count;

  for(count=0; count<WAIT_LOOP; count++){
    ;
  }
}

void main(void)
{

  /*********************************************
    LED 接続端子の初期化
  *********************************************/

  PCR8 |= LED_MASK;
  PDR8.BYTE &= (~(LED2 << LED_SHIFT));
  PDR8.BYTE |= ((LED0 | LED1) << LED_SHIFT);

  while(1){
    WaitLoop();
    /*********************************************
      LED2 を点滅
    *********************************************/
    PDR8.BIT.B7 ^= 1;
  }
}
```

End Of List

実行結果

初期状態は LED2 のみ点灯。LED0 と LED1 は消灯。
LED2 が点滅する。

プログラム解説 (01_LED13.c)

```
17    union uni_p8dr{              /* PDR8         */
18        unsigned char BYTE;      /*  Byte Access */
19        struct {                 /*  Bit  Access */
20            unsigned char B7:1;  /*    Bit 7     */
21            unsigned char B6:1;  /*    Bit 6     */
22            unsigned char B5:1;  /*    Bit 5     */
23            unsigned char B4:1;  /*    Bit 4     */
24            unsigned char B3:1;  /*    Bit 3     */
25            unsigned char B2:1;  /*    Bit 2     */
26            unsigned char B1:1;  /*    Bit 1     */
27            unsigned char B0:1;  /*    Bit 0     */
28         }BIT;
29    };
30
31    #define PDR8  (*((volatile union uni_p8dr *)0xFFDB)) /* PDR8  Address*/
```

ポートデータレジスタ 8(PDR8) に対して、共用体を用いて 1 バイトのメンバ BYTE とビットフィールド BIT を配置しています (図 5.8)。

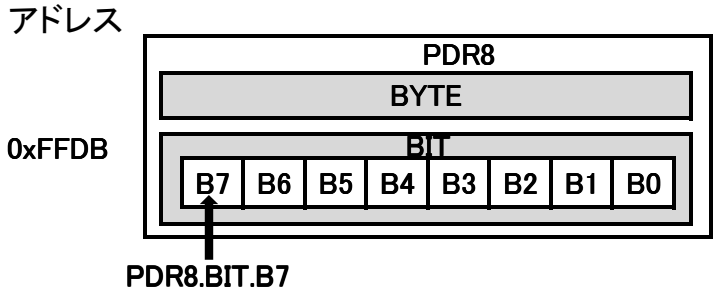

図 5.8　共用体 uni_p8dr のメンバと配置

```
59    PDR8.BYTE &= (~(LED2<<LED_SHIFT));
```

ポートデータレジスタ 8(PDR8) を 1 バイトで扱う場合には、このように PDR8.BYTE と記述します。

LED のつながっていないビットに値を代入しないようにビット演算を行っています。

```
67    PDR8.BIT.B7 ^= 1;
```

ポートデータレジスタ 8(PDR8) を 1 ビット単位であつかう場合には、このように PDR8.BIT.B7(あるいは B6,B5) と記述します。

1 ビット単位なので、他のビットを設定しないようにビット演算する必要はありません。直接代入することができます。

今回は値の反転に排他的論理和 (XOR) を用いました。

5.4 LED0 の点滅 (構造体を用いたレジスタの定義：完成形)

ここまで説明してきた構造体、ビットフィールド、共用体をすべて使うと、LED のつながったポート (ポート 8) の I/O レジスタをひとまとめにし、バイト単位でもビット単位でもアクセスできるような宣言が可能になります。

実際にやってみましょう。

01_LED14.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :01_LED14.c                                             */
/*  DESCRIPTION :LED0 の点滅 (構造体を用いたレジスタの定義：完成形)     */
/*  CPU TYPE    :H8/3694F                                               */
/*                                                                      */
/*  LED0 P85                                                            */
/*  LED1 P86                                                            */
/*  LED2 P87                                                            */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/

struct st_io {                  /* struct IO    */
  union uni_p8dr{               /* PDR8         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
        unsigned char B7:1;     /*    Bit 7     */
        unsigned char B6:1;     /*    Bit 6     */
        unsigned char B5:1;     /*    Bit 5     */
        unsigned char B4:1;     /*    Bit 4     */
        unsigned char B3:1;     /*    Bit 3     */
        unsigned char B2:1;     /*    Bit 2     */
        unsigned char B1:1;     /*    Bit 1     */
        unsigned char B0:1;     /*    Bit 0     */
     }BIT;
  }PDR8;
  char  wk5[15];                /*              */
  unsigned char       PCR8;     /* PCR8         */
};

#define IO  (*((volatile struct st_io *)0xFFDB)) /* P8 Address*/

#define LED0 0x01
#define LED1 0x02
#define LED2 0x04

#define LED_SHIFT 5
#define LED_MASK ((LED0 | LED1 | LED2) <<LED_SHIFT)

#define WAIT_LOOP 0x3FFFFFL

void WaitLoop(void)
{
  unsigned long count;

  for(count=0; count<WAIT_LOOP; count++){
    ;
  }
}

void main(void)
{

  /*******************************************
    LED 接続端子の初期化
  *******************************************/

  IO.PCR8 |= LED_MASK;
  IO.PDR8.BYTE &= (~((LED0 | LED2) <<LED_SHIFT));
  IO.PDR8.BYTE |= (LED1 <<LED_SHIFT);

  while(1){
    WaitLoop();
    /*******************************************
      LED0 を点滅
    *******************************************/
    IO.PDR8.BIT.B5 = (IO.PDR8.BIT.B5 ? 0 : 1);
  }
}
```

End Of List

実行結果

初期状態は LED0 と LED2 が点灯。LED1 は消灯。
LED0 が点滅する。LED1 は消灯、LED2 は点灯のまま。

プログラム解説 (01_LED14.c)

```
15    struct st_io {                /* struct IO    */
16      union uni_p8dr{             /* PDR8         */
17        unsigned char BYTE;       /*  Byte Access */
18        struct {                  /*  Bit  Access */
19            unsigned char B7:1;   /*    Bit 7     */
20            unsigned char B6:1;   /*    Bit 6     */
21            unsigned char B5:1;   /*    Bit 5     */
22            unsigned char B4:1;   /*    Bit 4     */
23            unsigned char B3:1;   /*    Bit 3     */
24            unsigned char B2:1;   /*    Bit 2     */
25            unsigned char B1:1;   /*    Bit 1     */
26            unsigned char B0:1;   /*    Bit 0     */
27         }BIT;
28      }PDR8;
29      char  wk5[15];              /*              */
30      unsigned char       PCR8;   /* PCR8         */
31    };
32
33    #define IO  (*((volatile struct st_io *)0xFFDB)) /* P8 Address*/
```

ポート 8 に対しての宣言です。

ポートコントロールレジスタ 8(PCR8) とポートデータレジスタ 8(PDR8) を構造体 st_io のメンバとして宣言しています。

さらに、ポートデータレジスタ 8(PDR8) に対して、共用体を用いて、1 バイトのメンバ BYTE と 1 ビットのメンバを持つビットフィールド BIT を宣言しています (図 5.9)。

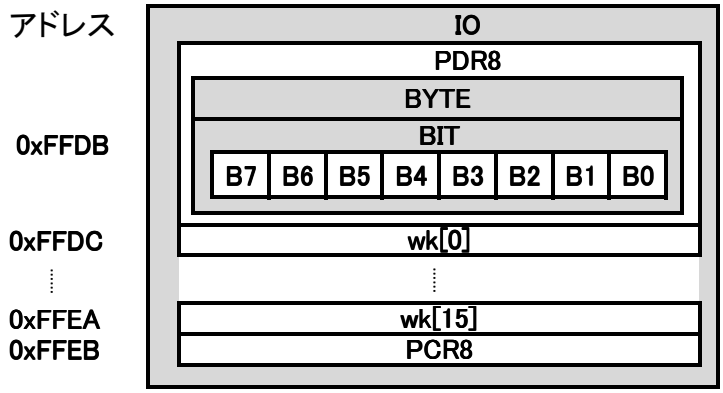

図 5.9　ポート 8 のレジスタ宣言

```
60    IO.PCR8 |= LED_MASK;
```

ポートコントロールレジスタ 8(PCR8) へのアクセスは、IO.PCR8 と書くことができます。

LED につながっているレジスタだけ 1(出力) に設定しています。

```
61    IO.PDR8.BYTE &= (~((LED0 | LED2) <<LED_SHIFT));
```

ポートデータレジスタ 8(PDR8) へのアクセスは、1 バイトでアクセスするときには IO.PDR8.BYTE と書くことができます。

ここでは、LED0 と LED2 に対して 0 を設定しています。LED0 と LED2 が点灯します。

```
62    IO.PDR8.BYTE |= (LED1 <<LED_SHIFT);
```

LED1 に対して 1 を設定しています。LED1 は消灯します。

```
69    IO.PDR8.BIT.B5 = (IO.PDR8.BIT.B5 ? 0 : 1);
```

ポートデータレジスタ 8(PDR8) へのアクセスは、1 ビットでアクセスするときには IO.PDR8.BIT.B5 のように書くことができます。

ここでは、値の反転に条件演算子 (?) を使ってみました。

```
IO.PDR8.BIT.B5 ? 0 : 1
```

は、IO.PDR8.BIT.B5 が真 (1) の場合には 0 になり、偽 (0) の場合には 1 になります。

5.5 LED0,2 の点滅 (HEW の iodefine.h を利用)

ここまでに HEW のレジスタ定義ファイル iodefine.h を利用するために必要な事項は説明してきました。さっそく iodefine.h を確認してみてください。01_LED14.c で宣言した構造体に似た宣言が見つかるでしょうか。

少々長くなりますが、関連する部分を抜き出してみましょう。なお、行番号は説明の都合上入れていますが、実際の iodefine.h の行番号とは一致していませんので注意してください。

iodefine.h(一部)

```
1  struct st_io {                                          /* struct IO    */
2                union {                                   /* PUCR1        */
3                      unsigned char BYTE;                 /*  Byte Access */
4                      struct {                            /*  Bit  Access */
5                             unsigned char B7:1;          /*    Bit 7     */
6                             unsigned char B6:1;          /*    Bit 6     */
7                             unsigned char B5:1;          /*    Bit 5     */
8                             unsigned char B4:1;          /*    Bit 4     */
9                             unsigned char   :1;          /*    Bit 3     */
```

```
10                              unsigned char B2:1;             /*    Bit 2     */
11                              unsigned char B1:1;             /*    Bit 1     */
12                              unsigned char B0:1;             /*    Bit 0     */
13                              }       BIT;                    /*              */
14                      }               PUCR1;                  /*              */
15              union {                                         /* PUCR5        */
16                      unsigned char BYTE;                     /*  Byte Access */
17                      struct {                                /*  Bit  Access */
18                              unsigned char   :2;             /*    Bit 7,6   */
19                              unsigned char B5:1;             /*    Bit 5     */
20                              unsigned char B4:1;             /*    Bit 4     */
21                              unsigned char B3:1;             /*    Bit 3     */
22                              unsigned char B2:1;             /*    Bit 2     */
23                              unsigned char B1:1;             /*    Bit 1     */
24                              unsigned char B0:1;             /*    Bit 0     */
25                              }       BIT;                    /*              */
26                      }               PUCR5;                  /*              */
27              char                    wk1[2];                 /*              */
28              union {                                         /* PDR1         */
29                      unsigned char BYTE;                     /*  Byte Access */
30                      struct {                                /*  Bit  Access */
31                              unsigned char B7:1;             /*    Bit 7     */
32                              unsigned char B6:1;             /*    Bit 6     */
33                              unsigned char B5:1;             /*    Bit 5     */
34                              unsigned char B4:1;             /*    Bit 4     */
35                              unsigned char   :1;             /*    Bit 3     */
36                              unsigned char B2:1;             /*    Bit 2     */
37                              unsigned char B1:1;             /*    Bit 1     */
38                              unsigned char B0:1;             /*    Bit 0     */
39                              }       BIT;                    /*              */
40                      }               PDR1;                   /*              */
41              union {                                         /* PDR2         */
42                      unsigned char BYTE;                     /*  Byte Access */
43                      struct {                                /*  Bit  Access */
44                              unsigned char   :5;             /*    Bit 7-3   */
45                              unsigned char B2:1;             /*    Bit 2     */
46                              unsigned char B1:1;             /*    Bit 1     */
47                              unsigned char B0:1;             /*    Bit 0     */
48                              }       BIT;                    /*              */
49                      }               PDR2;                   /*              */
50              char                    wk2[2];                 /*              */
51              union {                                         /* PDR5         */
52                      unsigned char BYTE;                     /*  Byte Access */
53                      struct {                                /*  Bit  Access */
54                              unsigned char B7:1;             /*    Bit 7     */
55                              unsigned char B6:1;             /*    Bit 6     */
56                              unsigned char B5:1;             /*    Bit 5     */
57                              unsigned char B4:1;             /*    Bit 4     */
58                              unsigned char B3:1;             /*    Bit 3     */
59                              unsigned char B2:1;             /*    Bit 2     */
60                              unsigned char B1:1;             /*    Bit 1     */
61                              unsigned char B0:1;             /*    Bit 0     */
62                              }       BIT;                    /*              */
63                      }               PDR5;                   /*              */
64              char                    wk3;                    /*              */
65              union {                                         /* PDR7         */
66                      unsigned char BYTE;                     /*  Byte Access */
67                      struct {                                /*  Bit  Access */
68                              unsigned char   :1;             /*    Bit 7     */
69                              unsigned char B6:1;             /*    Bit 6     */
70                              unsigned char B5:1;             /*    Bit 5     */
71                              unsigned char B4:1;             /*    Bit 4     */
72                              }       BIT;                    /*              */
73                      }               PDR7;                   /*              */
74              union {                                         /* PDR8         */
75                      unsigned char BYTE;                     /*  Byte Access */
76                      struct {                                /*  Bit  Access */
77                              unsigned char B7:1;             /*    Bit 7     */
78                              unsigned char B6:1;             /*    Bit 6     */
79                              unsigned char B5:1;             /*    Bit 5     */
80                              unsigned char B4:1;             /*    Bit 4     */
81                              unsigned char B3:1;             /*    Bit 3     */
82                              unsigned char B2:1;             /*    Bit 2     */
83                              unsigned char B1:1;             /*    Bit 1     */
84                              unsigned char B0:1;             /*    Bit 0     */
85                              }       BIT;                    /*              */
86                      }               PDR8;                   /*              */
87              char                    wk4;                    /*              */
88              union {                                         /* PDRB         */
89                      unsigned char BYTE;                     /*  Byte Access */
90                      struct {                                /*  Bit  Access */
91                              unsigned char B7:1;             /*    Bit 7     */
92                              unsigned char B6:1;             /*    Bit 6     */
93                              unsigned char B5:1;             /*    Bit 5     */
94                              unsigned char B4:1;             /*    Bit 4     */
```

```
 95                              unsigned char B3:1;           /*    Bit 3     */
 96                              unsigned char B2:1;           /*    Bit 2     */
 97                              unsigned char B1:1;           /*    Bit 1     */
 98                              unsigned char B0:1;           /*    Bit 0     */
 99                              }      BIT;                   /*              */
100                        }            PDRB;                  /*              */
101                char                wk5[2];                /*              */
102                union {                                    /* PMR1         */
103                      unsigned char BYTE;                  /*  Byte Access */
104                      struct {                             /*  Bit  Access */
105                              unsigned char IRQ3:1;        /*    IRQ3      */
106                              unsigned char IRQ2:1;        /*    IRQ2      */
107                              unsigned char IRQ1:1;        /*    IRQ1      */
108                              unsigned char IRQ0:1;        /*    IRQ0      */
109                              unsigned char     :2;        /*              */
110                              unsigned char TXD :1;        /*    TXD       */
111                              unsigned char TMOW:1;        /*    TMOW      */
112                              }      BIT;                  /*              */
113                      }            PMR1;                  /*              */
114                union {                                    /* PMR5         */
115                      unsigned char BYTE;                  /*  Byte Access */
116                      struct {                             /*  Bit  Access */
117                              unsigned char     :2;        /*              */
118                              unsigned char WKP5:1;        /*    WKP5      */
119                              unsigned char WKP4:1;        /*    WKP4      */
120                              unsigned char WKP3:1;        /*    WKP3      */
121                              unsigned char WKP2:1;        /*    WKP2      */
122                              unsigned char WKP1:1;        /*    WKP1      */
123                              unsigned char WKP0:1;        /*    WKP0      */
124                              }      BIT;                  /*              */
125                      }            PMR5;                  /*              */
126                char                wk6[2];                /*              */
127                unsigned char       PCR1;                  /* PCR1         */
128                unsigned char       PCR2;                  /* PCR2         */
129                char                wk7[2];                /*              */
130                unsigned char       PCR5;                  /* PCR5         */
131                char                wk8;                   /*              */
132                unsigned char       PCR7;                  /* PCR7         */
133                unsigned char       PCR8;                  /* PCR8         */
134  };                                                       /*              */
135
136  #define IO      (*(volatile struct st_io    *)0xFFD0)   /* IO    Address*/
```

End Of List

I/O ポート全体に対する構造体を宣言しているためにメンバが多くなっています。

ポートデータレジスタ 8(PDR8) に関連する部分は 74 行目から 86 行目までです。

ポートコントロールレジスタ 8(PCR8) に関連する部分は 133 行目です。

この iodefine.h を利用して LED 点滅のプログラムを書いてみましょう。

01_LED15.c

```
 1  /***********************************************************************/
 2  /*                                                                     */
 3  /*  FILE        :01_LED15.c                                            */
 4  /*  DESCRIPTION :LED0,2 の点滅 (HEW の iodefine.h を利用)                 */
 5  /*  CPU TYPE    :H8/3694F                                              */
 6  /*                                                                     */
 7  /*  LED0 P85                                                           */
 8  /*  LED1 P86                                                           */
 9  /*  LED2 P87                                                           */
10  /*                                                                     */
11  /*  This file is generated by Renesas Project Generator (Ver.4.9).     */
12  /*                                                                     */
13  /***********************************************************************/
14
15  #include"iodefine.h"
16
17  #define LED0 0x01
18  #define LED1 0x02
19  #define LED2 0x04
20
21  #define LED_SHIFT 5
22  #define LED_MASK ((LED0 | LED1 | LED2) <<LED_SHIFT)
23
24  #define WAIT_LOOP 0x3FFFFFL
25
```

```
26  void WaitLoop(void)
27  {
28    unsigned long count;
29
30    for(count=0; count<WAIT_LOOP; count++){
31      ;
32    }
33  }
34
35  void main(void)
36  {
37
38    /*********************************************
39      LED 接続端子の初期化
40    *********************************************/
41
42    IO.PCR8 |= LED_MASK;
43    IO.PDR8.BIT.B5 = 0;
44    IO.PDR8.BIT.B6 = 1;
45    IO.PDR8.BIT.B7 = 1;
46
47    while(1){
48      WaitLoop();
49      /*********************************************
50        LED0 を点滅
51      *********************************************/
52      IO.PDR8.BYTE ^= ((LED0 | LED2) <<LED_SHIFT);
53    }
54  }
```

End Of List

実行結果

- LED0 のみが点灯 (○ ● ●)(○ : 点灯、● : 消灯)
- LED2 のみが点灯 (● ● ○)
- 以下繰り返し

プログラム解説 (01_LED15.c)

```
15    #include"iodefine.h"
```

iodefine.h を利用するために include しています。

```
42      IO.PCR8 |= LED_MASK;
```

ポートコントロールレジスタ 8(PCR8) は 1 バイトでメンバが宣言されています。

LED 部分のみを出力に変更しています。

```
43    IO.PDR8.BIT.B5 = 0;
44    IO.PDR8.BIT.B6 = 1;
45    IO.PDR8.BIT.B7 = 1;
```

ポートデータレジスタ 8(PDR8) に 1 ビット単位でアクセスし、値を設定しています。

LED0 は点灯、LED1 と LED2 は消灯です。

```
52    IO.PDR8.BYTE ^= ((LED0 | LED2) <<LED_SHIFT);
```

ポートデータレジスタ 8(PDR8) に 1 バイトでアクセスし、値を設定しています。

LED0(P85) と LED2(P87) を排他的論理和 (XOR) で反転させています。

6 ハードウェア依存性の分離

6.1 LED のカウントアップ (ハードウェア依存性の分離)

第 4 章においてはレジスタ定義ファイルを用いずに組込みプログラミングを行いました。自分でレジスタの定義をするので大変ですが、きちんとプログラムすれば移植性・汎用性の高いプログラムが書けるのではないかと思います。第 4 章では最終的に LED 関連関数を作り、別ファイルに分割して再利用性を高めました。

第 5 章においては HEW で自動生成されるレジスタ定義ファイル iodefine.h を利用する方法について解説しました。移植性は低くなりますが、すべてのレジスタを自分で定義するのは大変ですし、バグの原因にもなりかねません。そこで、このようなあらかじめ用意されているレジスタ定義ファイルを利用することを推奨したいと思います。

このテキストの方針としては、開発環境で用意されているレジスタ定義ファイルは利用し、その上で移植性をできるだけ高めたいと思ってます。

そのために、レジスタ定義ファイルを利用する部分 (ハードウェアに依存する部分) はできるだけ一箇所にまとめるように努力していきましょう。

たとえば、ファイルを分割する際に led.h というヘッダファイルと led.c というソースファイルを作りました。レジスタ定義ファイルを使うプログラム (ハードウェア依存部) は、できるだけヘッダファイルに限定しましょう。ただし移植性を考えると必ずしもヘッダファイルにハードウェア依存部をまとめれば良いというものでもありません。場合によってはソースファイルに記述する必要があるかもしれません。そのような場合でも、できるだけ一か所にまとめるようにしたいと思います。

また、関数の仕様は P.113 のものとほぼ同じです。今回はファイルを分割したので、利用に際して led.h が必要であることを書き加えておきました。関数の仕様自体は変更していません。

```
LED 関連関数 (H8/3694F)

ヘッダファイル : led.h

LED0 P85
LED1 P86
LED2 P87

*****************************************************************
void LedInit(void);

    形式  : #include"led.h"
            void LedInit(void);
    機能  : LED で使用する I/O ポートを初期化する関数。
            初期状態は LED すべて消灯。
            LED を使用する前に実行すること。
    引数  : なし
    戻り値 : なし

*****************************************************************
void LedOn(unsigned char led_data);

    形式  : #include"led.h"
            void LedOn(unsigned char led_data);
    機能  : 1 で指定された LED を点灯する。
            ビットパターンで同時に複数の LED を指定できる。
            1 で指定された LED 以外 (0 で指定された LED) には変化が無い。
    引数  : unsigned char led_data
            点灯する LED の指定 (LED0 or LED1 or LED2)
            下位 3 ビット分のみ有効。
    戻り値 : なし

*****************************************************************
void LedOff(unsigned char led_data);

    形式  : #include"led.h"
            void LedOff(unsigned char led_data);
    機能  : 1 で指定された LED を消灯する。
            ビットパターンで同時に複数の LED を指定できる。
            1 で指定された LED 以外 (0 で指定された LED) には変化が無い。
    引数  : unsigned char led_data
            消灯する LED の指定 (LED0 or LED1 or LED2)
            下位 3 ビット分のみ有効。
    戻り値 : なし

*****************************************************************
void LedSet(unsigned char led_data);

    形式  : #include"led.h"
            void LedSet(unsigned char led_data);
    機能  : 1 で指定された LED を点灯し、
            0 で指定された LED を消灯する。
            ビットパターンで同時に 3 ビット分の LED を指定する。
    引数  : unsigned char led_data
            点灯する LED の指定 (LED0 or LED1 or LED2)
            下位 3 ビット分のみ有効。
    戻り値 : なし
```

6.1.1 ヘッダファイルと関数のファイルをひとまとめにする

LED の関数の利用に必要なプログラムは led.h と led.c に記述しました。LED を使う場合にはこれらのファイルを include すればよいわけです。

ビルドに必要なファイルはすべてプロジェクトのフォルダの中に入れていますが、LED を使うプロジェクトすべてに、これらのファイルをコピーしておくのは良いやり方とはいえません。ワークスペース全体の容量が増えますし、何よりも修正する場合に困ります。つまり、led.c のソースコードを修正したいと思ったときに、ワークスペース内のすべての led.c を修正しないといけなくなってしまいます。

そこで、このようなほかのプログラムでも利用するファイルについては、新たに lib という名前のフォルダを作って、その中に入れておきたいと思います。フォルダ lib は 01_LED01 などと同じ階層に制作してください。なお、iodefine.h も以下ではこの lib というフォルダに入れておきたいと思います。

HEW で生成される iodefine.h には、二重呼び出しを回避するような記述がありませんので (P.126 の led.h 解説参照)、これを付け加えて置いてください。

つまり、ファイルの最初の行に

```
#ifndef _IODEFINE_H_
#define _IODEFINE_H_
```

を記述し、最終行に

```
#endif
```

を記述しておいてください。

以後 iodefine.h はプロジェクト作成の際に自動生成しないようにするか (P.67 の図 3.12 参照)、生成されたものを削るようにしましょう。

重要：
iodefine.h はファイルの最初の行に

```
#ifndef _IODEFINE_H_
#define _IODEFINE_H_
```

を記述し、最終行に

```
#endif
```

を記述する。

led.h と led.c は、P.124 で入力したソースコードをコピーし、内容を変更することにします。

6.1.2 プロジェクトにソースファイルを追加 (既存)

Windows の「スタート」-「コンピュータ」をクリックし、エクスプローラから HEW のワークスペースを作ったフォルダを開きます (本実習の場合には `Z:\H8_3694F` です)。

フォルダ内で右クリックし、「新規作成」-「フォルダ」を選びます (図 6.1)。

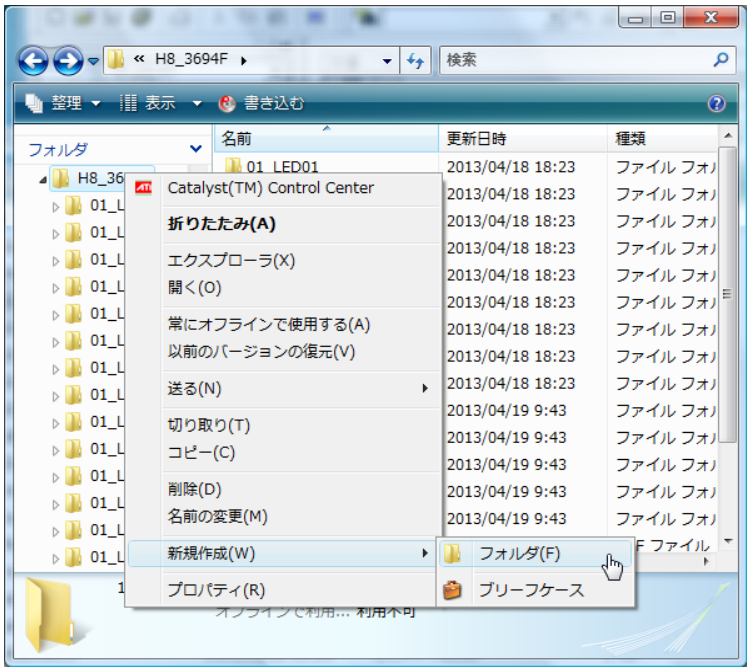

図 6.1　ワークスペースフォルダにフォルダを作る

「新しいフォルダ」の名前を「lib」に変更します (図 6.2)。

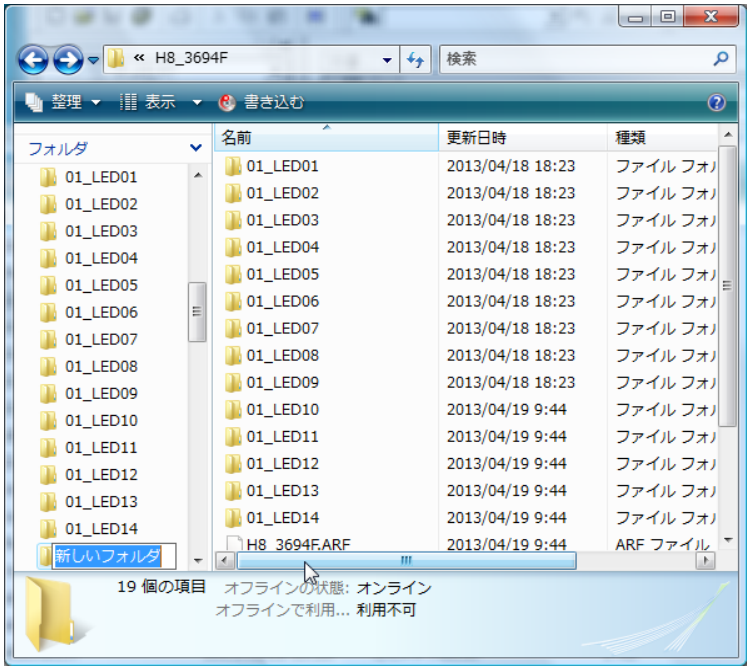

図 6.2　フォルダを作る

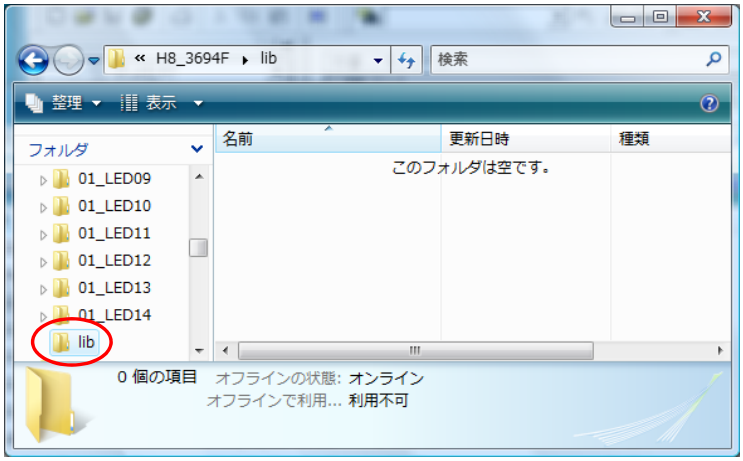

図 6.3　フォルダ lib を作る

フォルダ「01_LED10」の中にある led.h、led.c、さらにはレジスタ定義ファイル iodefine.h を選び、右クリックで表示されるメニューから「コピー」を選びます (図 6.4)。

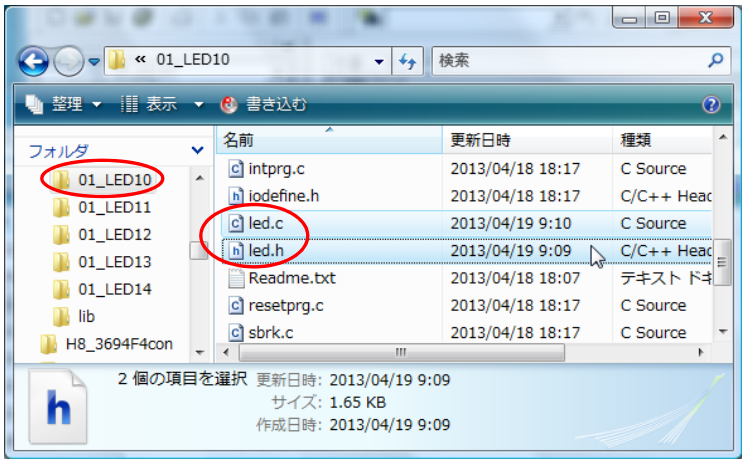

図 6.4　01_LED10 のファイルをコピー

「lib」で右クリックし、「貼り付け」を選びます (図 6.5)。

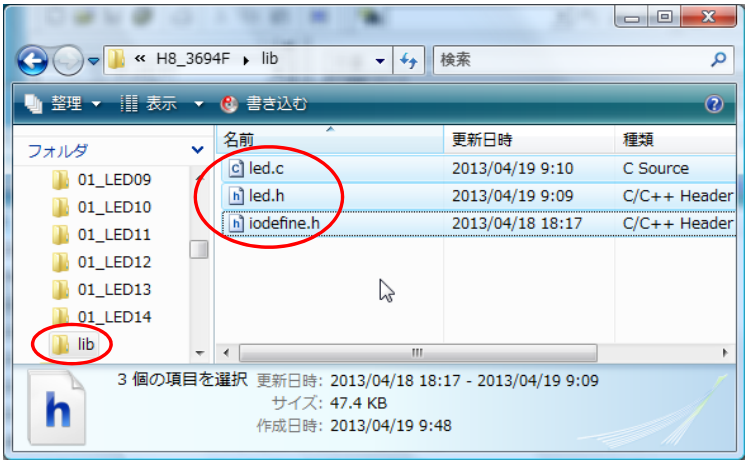

図 6.5　01_LED10 のファイルを貼り付け

01_LED16 に iodefine.h があるようでしたら削除しておきましょう (図 6.6)。

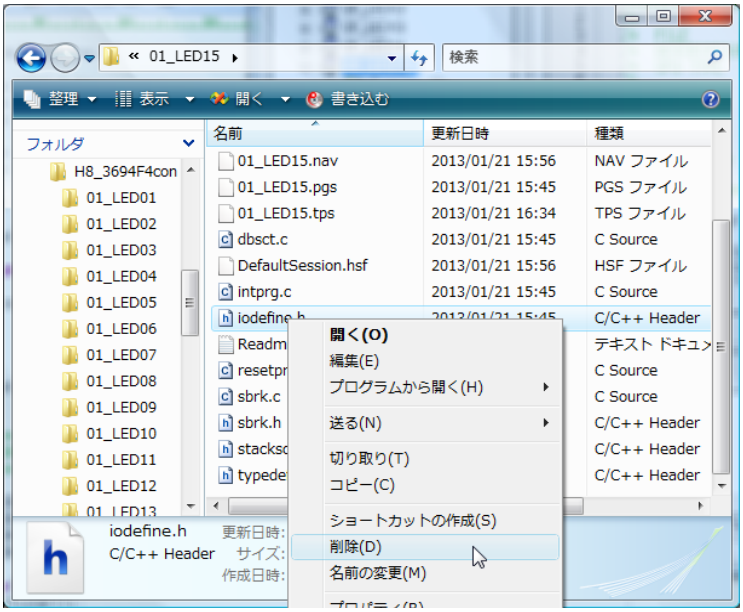

図 6.6　01_LED16 の iodefine.h を削除

HEW のワークスペースウィンドウ上で、プロジェクト名 01_LED16 を右クリックし、表示されたメニューから「ファイルの追加……」を選びます (図 6.7)。

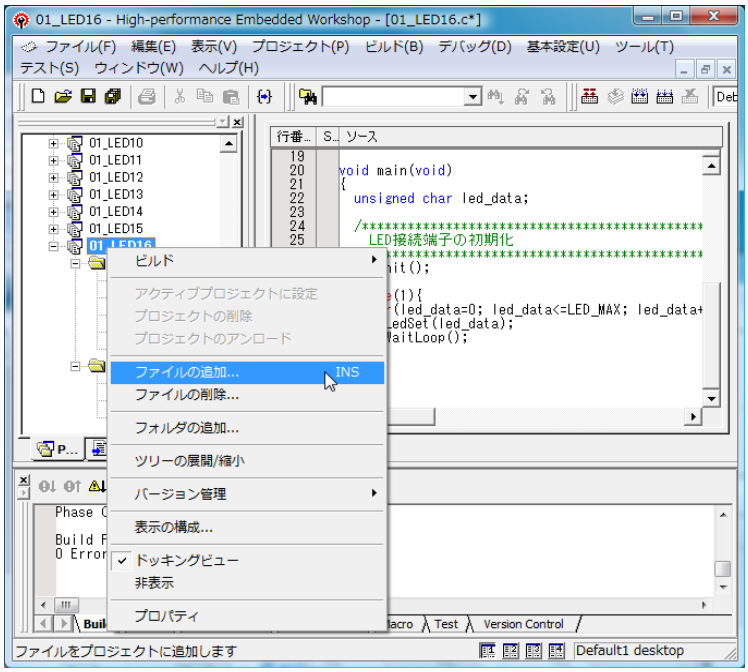

図 6.7　ファイルの追加

「ファイルを追加」ウィンドウが表示されるので、「lib」の中の led.c を選んで追加します (図 6.8)。ワークスペースウィンドウのプロジェクト名 01_LED16 に led.c が追加されたことを確認しておいてください。その際に、led.h も追加されていることを確認しましょう。

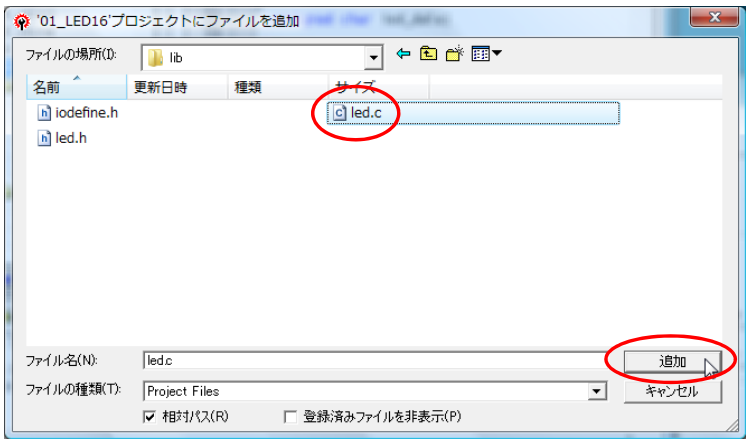

図 6.8　led.c の追加

HEW のメニューから「ビルド」-「H8S, H8/300 Standard Toolchain」を選びます (図 6.9)。

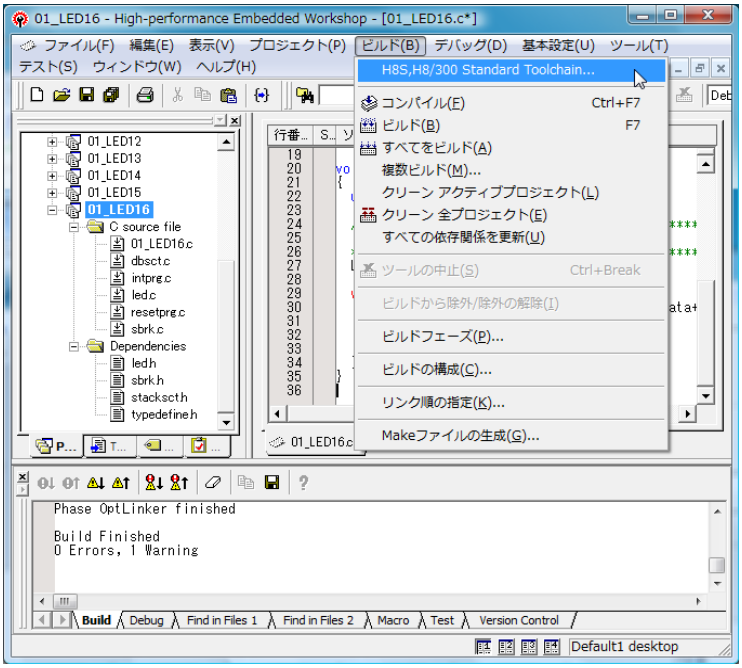

図 6.9　Standard Toolchain の設定

「コンパイラ」タブの、カテゴリを「ソース」に、オプション項目を「インクルードファイルディレクトリ」にし、「追加」ボタンをクリックします (図 6.10)。

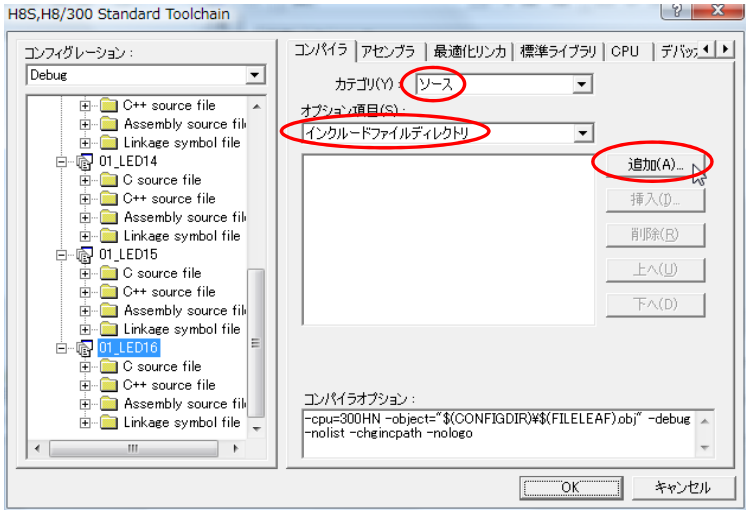

図 6.10　インクルードファイルディレクトリの追加

「Add include file directory」ウィンドウで、相対パスのプルダウンメニューを「Custum directory」にし、「ブラウズ」ボタンをクリックします (図 6.11)。

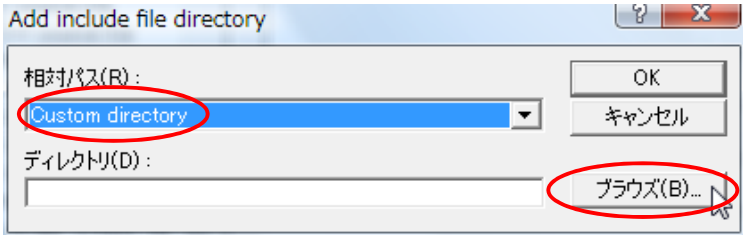

図 6.11　インクルードファイルディレクトリの追加

ディレクトリ「lib」を選びます (図 6.12)。

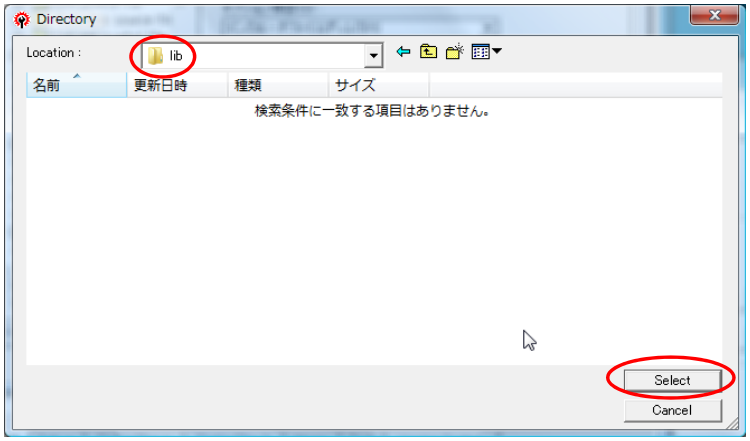

図 6.12　インクルードファイルディレクトリの追加

ディレクトリに選択したディレクトリ (今の場合「`Z:\H8_3694F\lib`」) が表示されます (図 6.13)。

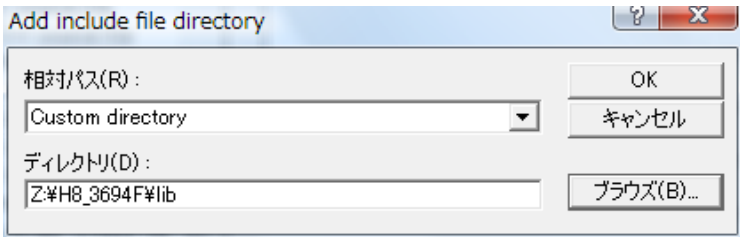

図 6.13　インクルードファイルディレクトリの追加

「OK」ボタンをクリックします (図 6.14)。これで設定は完了です。あとは通常通りビルドしてください。

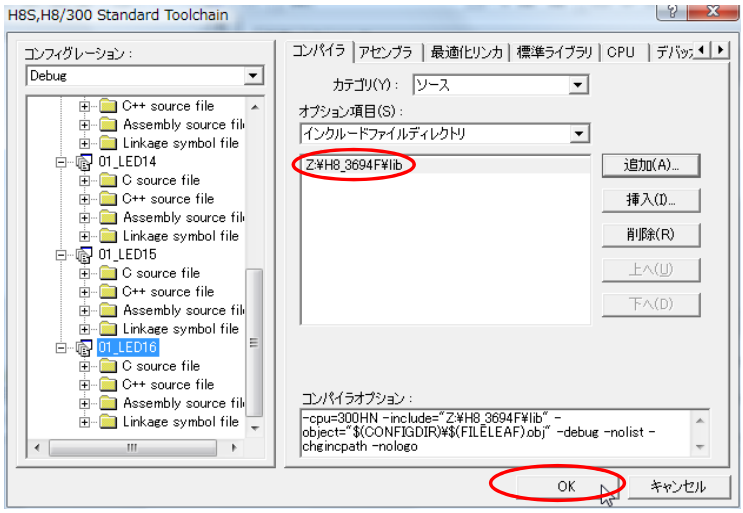

図 6.14　インクルードファイルディレクトリの追加

さて、プログラム led.h、led.c の内容は以下のように変更してください。

LED 点灯プログラムでは、以下のように記述してみました。プログラムは P.108 第 4.9 章で提示したサンプルプログラム 01_LED07.c を書き直したものです。

led.h

```
 1  #ifndef _LED_H_
 2  #define _LED_H_
 3
 4  #include"iodefine.h"
 5
 6  #define LEDDAT IO.PDR8.BYTE
 7  #define LEDIO  IO.PCR8
 8
 9  #define LED0 0x01
10  #define LED1 0x02
11  #define LED2 0x04
12
13  #define LED_SHIFT 5
14  #define LED_MASK ((LED0 | LED1 | LED2) << LED_SHIFT)
15
16  #define WAIT_LOOP 0x3FFFFFL
17
18  void LedInit(void);
19  void LedOn(unsigned char led_data);
20  void LedOff(unsigned char led_data);
21  void LedSet(unsigned char led_data);
22  void WaitLoop(void);
23
24  #endif
```

End Of List

led.c

```
 1  /*********************************************************************/
 2  /*                                                                   */
 3  /*  FILE        :led.c                                               */
 4  /*  DESCRIPTION :LED 用関数                                          */
 5  /*  CPU TYPE    :H8/3694F                                            */
 6  /*                                                                   */
 7  /*  LED0 P85                                                         */
 8  /*  LED1 P86                                                         */
 9  /*  LED2 P87                                                         */
10  /*                                                                   */
11  /*********************************************************************/
```

```
12
13  #include"led.h"
14
15  #define WAIT_LOOP 0x3FFFFFL
16
17  void WaitLoop(void)
18  {
19    unsigned long count;
20
21    for(count=0; count<WAIT_LOOP; count++){
22      ;
23    }
24  }
25
26  /***************************************
27    LED 接続端子の初期化 (LED はすべて消灯)
28  ***************************************/
29  void LedInit(void)
30  {
31    LEDIO  |= LED_MASK;
32    LEDDAT |= LED_MASK;
33  }
34
35  /*******************************
36    1 を指定した LED を点灯
37  *******************************/
38  void LedOn(unsigned char led_data)
39  {
40    LEDDAT &= (~((led_data << LED_SHIFT) & LED_MASK)); /* 0 をセット */
41  }
42
43  /*******************************
44    1 を指定した LED を消灯
45  *******************************/
46  void LedOff(unsigned char led_data)
47  {
48    LEDDAT |= ((led_data << LED_SHIFT) & LED_MASK); /* 1 をセット */
49  }
50
51  /*******************************
52    指定した LED を点灯
53    それ以外は消灯
54  *******************************/
55  void LedSet(unsigned char led_data)
56  {
57    LedOn(led_data);
58    LedOff(~led_data);
59  }
```

End Of List

01_LED16.c

```
 1  /*********************************************************************/
 2  /*                                                                   */
 3  /*  FILE        :01_LED16.c                                          */
 4  /*  DESCRIPTION :LED のカウントアップ                                */
 5  /*               (ファイルの分割、ハードウェア依存性の分離)          */
 6  /*  CPU TYPE    :H8/3694F                                            */
 7  /*                                                                   */
 8  /*  LED0 P85                                                         */
 9  /*  LED1 P86                                                         */
10  /*  LED2 P87                                                         */
11  /*                                                                   */
12  /*  This file is generated by Renesas Project Generator (Ver.4.9).   */
13  /*                                                                   */
14  /*********************************************************************/
15
16  #include "led.h"
17
18  #define LED_MAX (LED0 | LED1 | LED2)
19
20  void main(void)
21  {
22    unsigned char led_data;
23
24    /*********************************************
25      LED 接続端子の初期化
26    *********************************************/
27    LedInit();
28
29    while(1){
30      for(led_data=0; led_data<=LED_MAX; led_data++){
31        LedSet(led_data);
32        WaitLoop();
33      }
34    }
35  }
```

End Of List

実行結果

最初はすべて消灯。
LED の表示がカウントアップしていく。
すべて点灯までいったら、すべて消灯に戻る。

プログラム解説 (led.h)

```
4    #include"iodefine.h"
```

レジスタ定義ファイル iodefine.h を利用するために include しています。

```
6    #define LEDDAT IO.PDR8.BYTE
7    #define LEDIO  IO.PCR8
```

レジスタ定義ファイル iodefine.h の LED に関するものに、名前をつけなおしました。

6 行目でポートデータレジスタ 8(PDR8) は 1 バイトの定義を利用し、LEDDAT という名前をつけました。名前を付け替えることによって、他のレジスタ定義ファイルを用いる場合に、この定義だけ変更すれば良くなります。

7 行目も同様に、ポートコントロールレジスタ 8(PCR8) に LEDIO という名前をつけました。入出力を設定するレジスタの名前は必ずしもコントロールレジスタとつけられているわけでもないので、ここではこのような名前にしておきました。入出力を設定するレジスタという意味です。

プログラム解説 (led.c)

P.114 の 01_LED09.c の関数を、led.h の定義に沿って書き直しました。レジスタ定義の名前を変更しただけです。

プログラム解説 (01_LED16.c)

```
16    #include "led.h"
```

LED 用関数のプロトタイプ宣言などは led.h に宣言されています。利用する際には必ず include してください。

```
18    #define LED_MAX (LED0 | LED1 | LED2)
```

01_LED07.c では、

```
#define LED 0x07
```

と定義していたものです。

今回は最大値の意味でこのように定義しました。これは led.h には含めませんでした。

```
30    for(led_data=0; led_data<=LED_MAX; led_data++){
31      LedSet(led_data);
32      WaitLoop();
33    }
```

for ループを使って、led_data が 0 のときから、LED_MAX(7) になるまで値を 1 ずつ増やしていきます。31 行目でその値を関数 LedSet に渡して LED に反映させています。32 行目は待ちループです。