平成 24 年 6 月 13 日

実験 11．H8 マイコンを用いた LED 制御

１．はじめに

この実験では， H8 マイコンを用いて，LED を点滅させるプログラムを作成し，マイコンによる出力制御を学ぶ．既に前回の実験 10 において，LED を点滅させるプログラムは作成したが，今回は，プログラムの処理内容を理解し，複数のポートに接続した LED を点滅させることにする．

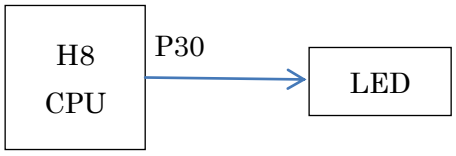
２．目的

H8 マイコンによる出力制御方法について， C プログラミングを使って学ぶ．また，マイコンに接続した LED の点滅プログラムを通して，出力制御の仕組みと方法について学ぶ．

３．実験

3.1　概要

H8CPU の P30（Port3 Bit0）の出力を制御するプログラムを作成し，ブレッドボード上に設置した LED を点滅（ON/OFF）させる．



実験の流れを以下に示す．

A：新規プロジェクトの作成

新規プロジェクトを作成し，プログラミング環境を準備する．

B：プログラムの作成

C 言語で LED 点滅プログラムをコーディングする．

C：ビルド（コンパイル作業）

作成したソースファイルをコンパイルし，H8 CPU が理解できる形式に変換し，「実行ファイル」を生成する．

D：実行ファイルをマイコンに書き込む

PC と H8 マイコンを接続し，生成した実行ファイルをマイコンに書き込む．

E：実験装置の制作

ブレッドボード上に，LED，抵抗などのパーツを回路図に従って配置する．

F：動作確認

電源を入れ，プログラム通りに LED が点滅することを確認する．

これらの手順のうち，A～D は実験 10 と同じである為，詳細は前回の実験資料を参考にせよ．

3.2　装置

（1）　LED

LED には極性（取り付ける向き）があり，接続の方向を間違えて電流を流すと LED や回路が壊れるため，注意が必要である．一般的には，リード線（足）の長い方がカソード（正極），短い方がアノード（負極）である（例外もある）．

LED の電流と光の関係は，下図に示すように LED に流す電流が多いほど発行する強度（cd という単位）も強くなるが，20mA 以上では，あまり変わらない．人が通常視認できる光度（100mcd）であれば，10mA の電流を流せば十分である．

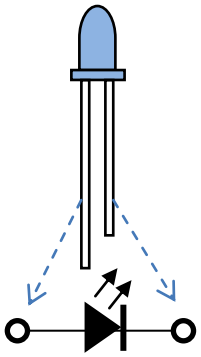


図　LED の極性の見方

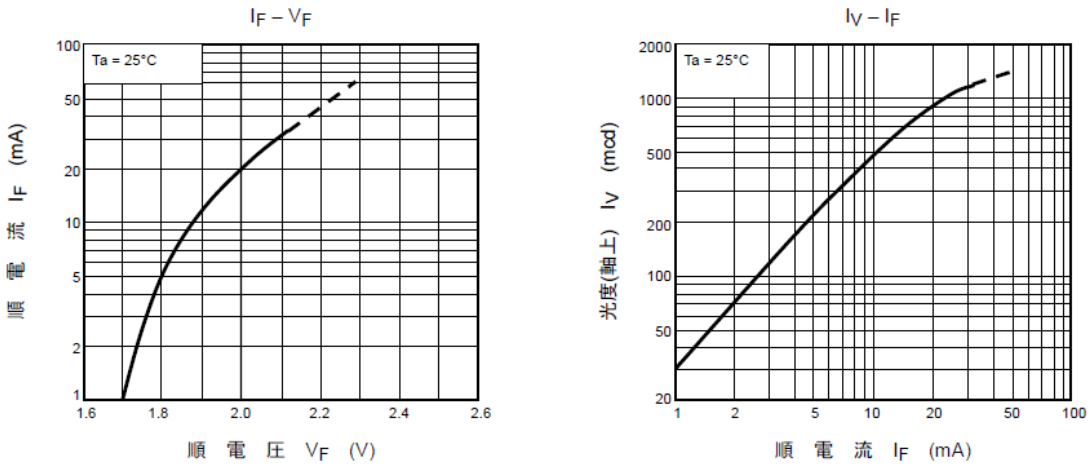


図　発光ダイオードの特性（TLOU11iP(F)）

http://www.alldatasheet.jp/datasheet-pdf/pdf/120679/TOSHIBA/TLOU113P.html

3.3　実験の手順

A. 新規プロジェクトの作成

デスクトップの「プログラミング数値処理関連」から「High-Performance Embedded Workshop」のショートカットをダブルクリックし、HEW を起動する。HEW の起動が完了したら，この実験用の新規プロジェクトを作成する.

- プロジェクトタイプ：Application
- ワークスペース名：LED_TEST2
- プロジェクト名：LED_TEST2
- ディレクトリ：z:¥LED_TEST2
- CPU 種別：H8S,H8/300（初期値のまま）
- ツールチェイン：Renesas H8S,H8/300 Standard（初期値のまま）

プロジェクト作成において，注意するところは以下の 2 か所である．それ以外は変更点はない.

（1）「1/9 CPU の選択」

CPU シリーズ　：　300H

CPU タイプ　　：　3687

（2）「4/9 標準ライブラリの選択」

「全て無効」ボタンをクリックして，ライブラリを外す.

プロジェクトが完了したら，第 10 回の実験同様にヘッダファイル「3687.h」を実験用のフォルダ（LED_TEST2.c と同じディレクトリ）にコピーする．3687.h は以下のディレクトリからコピーすること.

Y:¥HEW によるプログラミング開発¥3687.h

さらに、HEW の画面において、先にコピーしたヘッダファイル「3687.h」をプロジェクトに追加する作業も前回の実験と同様に行う（詳細は前回の資料を参照すること)。

B. プログラムの作成

HEW 上で、実験用のプログラムを作成する。大きく分けて、以下の処理を追加する。

（1） ヘッダファイル「3687.h」のインクルード

（2） ポートの設定（main 関数内）

（3） LED の点滅処理（main 関数内）

（4） 内部関数 wait()の宣言、定義

（１） インクルードファイルの設定

先ほどプロジェクトファイルに追加したヘッダーファイルを読み込む記述をソースファイルの先頭に記述する。

```
#include "3687.h"
```

（２） ポートの設定処理（main 関数内）

main 関数内において、まず最初に行う処理は、LED が接続されたポート(P30)を、「出力」の設定に切り替えることである。次の１行を記述する。

```
IO.PCR3 = 0x01;              //P30 (PORT3 BIT0 を出力に設定
```

（３） LED の点滅処理（main 関数内）

無限ループを作成し、その中で LED が接続されたポート(P30)を一定周期で、On/Off に切り替えることで、LED に流す電流の On/Off を行う。そのための処理を（２）の続きから記述せよ.

```
// メインループ（点滅の繰り返し）
while(1)
{
        //LED 点灯
        IO.PDR3.BIT.B0 = 1;
        sleep(100);

        //LED 消灯
        IO.PDR3.BIT.B0 = 0;
        sleep(100);
}
```

（４） sleep 関数の作成

LED を一定時間点灯または消灯させるために、一定時間何もしない処理（待ち時間）を作る必要がある。そこで、そのための関数 sleep を作成する。sleep は int 型の引数を取り、与えられた引数×0.01 秒だけ、待ち時間を作る処理を行う。つまり，引数に 100 を与えると，丁度 1 秒の待ち時間を作ることができる.

この関数のプロトタイプ宣言および定義を以下のように追加せよ。なお，プログラム中の（A）には，適切な文をいれよ（課題１）.

・プロトタイプ宣言（main 関数のプロトタイプ宣言の上に記述する）

```
void sleep( int wait_count );          //wait_count*0.01 秒だけ待つ
```

・関数の定義

```
void sleep( int wait_count )
{
        unsigned int i, j;

        for( i=0; i<wait_count ; i++) {
                for( j=0; j<  (A)  ;j++) {
                        ;          //何もしない
                }
        }
}
```

上記の（１）から（４）によって、完成した LED_TEST2.c を以下に示す。

```
#include "3687.h"

void sleep(int wait_count);
void main(void);

//一定時間の待ちを作る
void sleep(int wait_count)
{
        unsigned int i, j;
        for( i=0; i<wait_count; i++ ) {
                for( j=0; j<  (A)  ; j++ ) {
                        ;
                }
        }
}
```

```
//main 関数
void main(void)
{
        //初期環境設定
        IO.PCR3 = 0x01;// P30(PORT3 BIT0)を出力に設定

        //メインループ（点滅の繰り返し）
        while(1)
        {
                //LED 点灯
                IO.PDR3.BIT.B0 = 1;
                sleep (100);

                //LED 消灯
                IO.PDR3.BIT.B0 = 0;
                sleep(100);
        }
}
```

D.　コンパイル

メニュー中の「ビルド」から「全てをビルド」を選択し、ビルドを行う。ビルドが成功すると、下図のように HEW 下部のステータス画面に「Build Finished. 0 Errors, 0 Warnings」と表示される。ビルドに成功すると、Debug フォルダの中に「LED_TEST2.mot」という実行ファイルが生成されるので、確認せよ。

E　プログラムをマイコンに書き込む

H8 CPU 用のバイナリプログラム「LED_TEST2.mot」ファイルが準備できたら、PC と CPU 基盤を接続し、書き込み作業を行う。CPU 基盤と PC をシリアルケーブルで接続し、FDT を使って、バイナリプログラムを書き込め。

F.　実験装置の制作

コンパイルしたプログラムの書き込みが完了したら，いよいよ実験を開始する．まず，ベース基盤とブレッドボードを用いて，図のように実験回路を制作する．

【使用する部品】

H8CPU 基盤とベース基盤のセット	1 個
ブレッドボード	1 個
ジャンパワイヤ	必要数
LED	1 個
抵抗（560Ω ~~120Ω~~ ：　茶赤茶金）	1 個

【回路図】

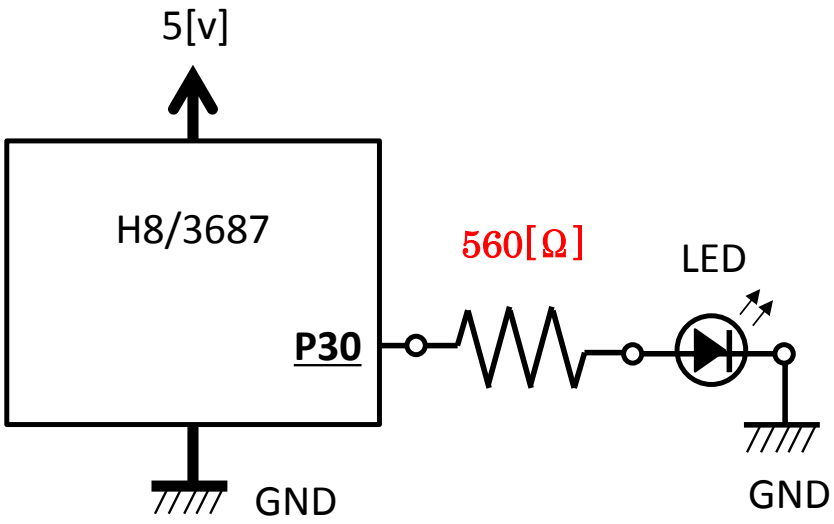


図　実験回路図

【配線図】

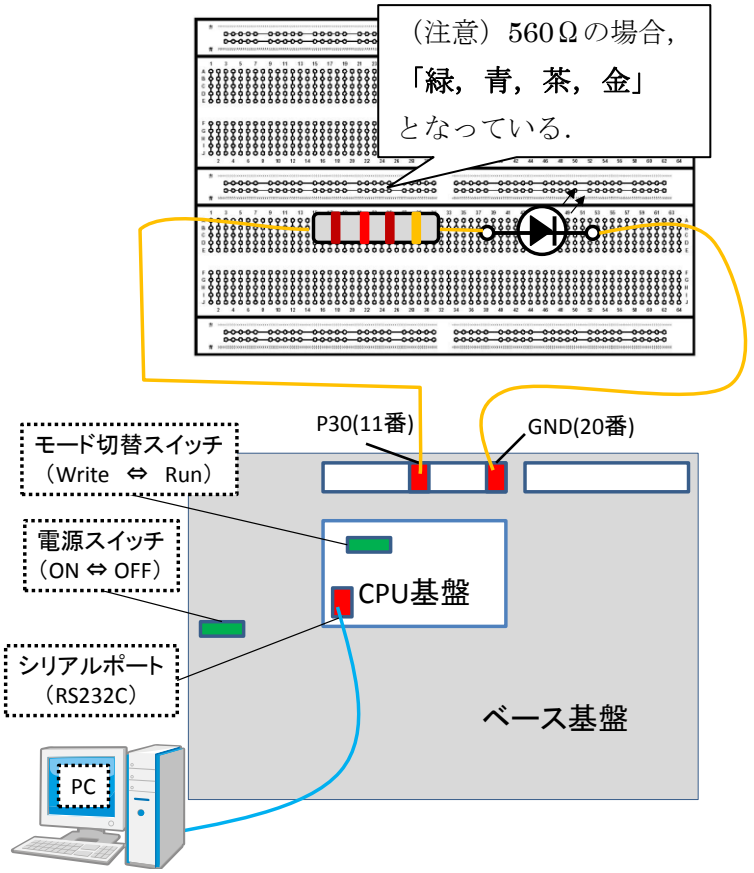


図　実験回路の配線図

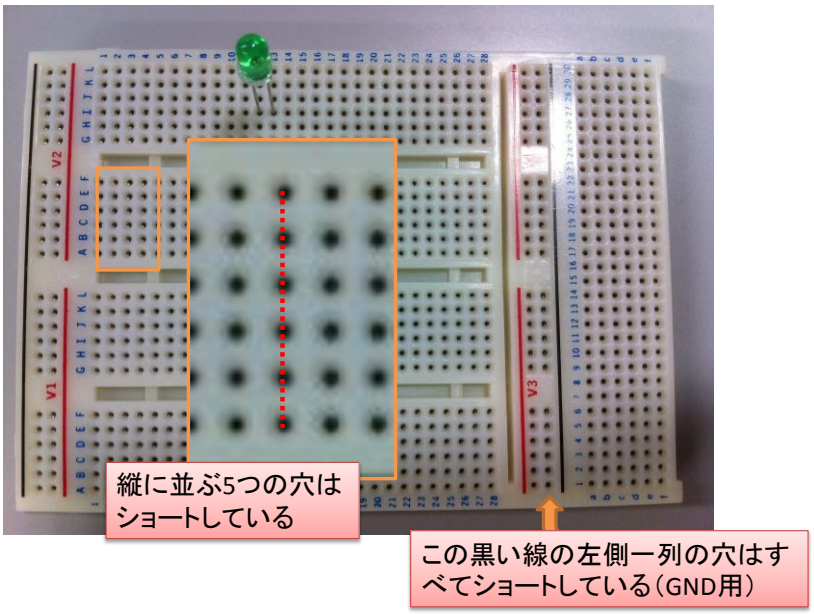


図　ブレッドボードについて

G.　実験

接続がすべて完了したら，次の手順に従って動作確認を行う．

Step 1　ベースボードの電源を一度 OFF にする．

Step 2　CPU 基盤の DIP スイッチを「実行モード（RUN)」に変更する．

Step 3　準備が出来たら，ベース基盤の電源スイッチを ON にする．

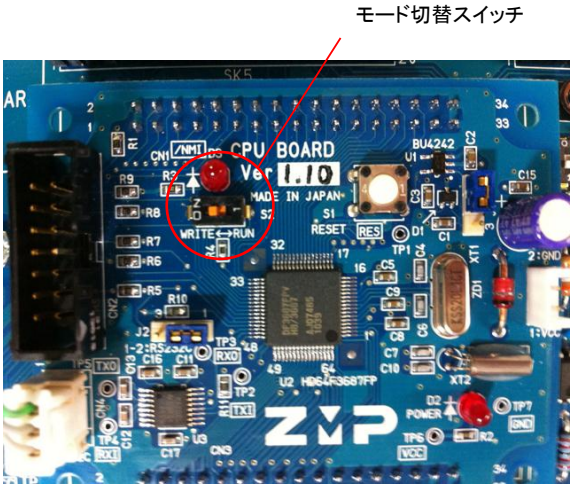


図　モード切替スイッチ（左：WRITE ⇔　右：RUN）

以上の手順の結果，ブレッドボード上の LED が点滅すれば成功である．

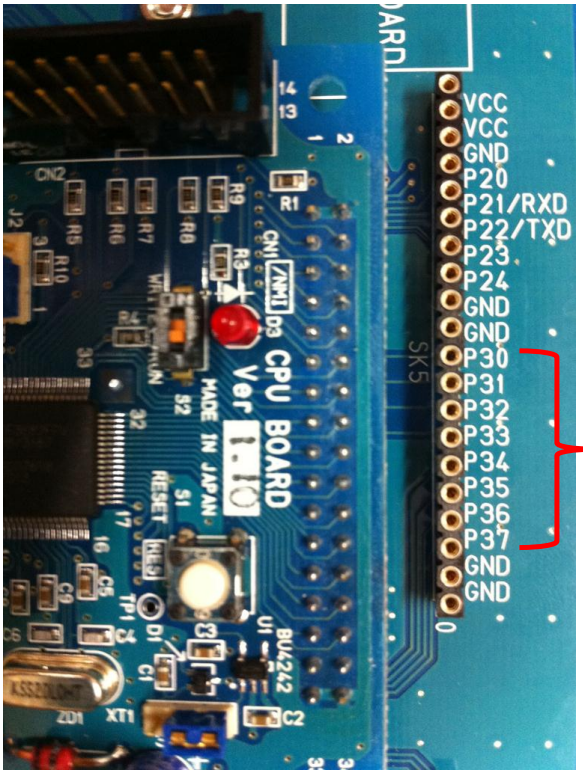
3.4　プログラムの解説

このプログラムでは，LED が接続された P30（PORT3 の BIT0）を操作するため，ポートコントロールレジスタ 3（PCR3）と，ポートデータレジスタ 3（PDR3）の 2 つを利用している．この 2 つのレジスタを使うことで，P30～P37 までのピンの ON／OFF を制御することができる．この仕組みについて詳しく見てみよう．

先ほどの実験で使用した回路図を見ると，コネクタの 11 番ピンが P30，20 番が GND に当たることが分かる．

GND は共通なので説明は割愛するが，今回のプログラムのポイントとなるのは P30 端子である．コネクタの P30 は H8 CPU の P30 端子に繋がっており，CPU から制御することが出来る．P30 を High = 5[V]に設定すると，LED に電流が流れ，LED が点灯する．また，Low = 0[V]に設定すると，電流が流れないため LED は消灯する．

P30 を操作するためには，ポートコントロールレジスタ 3（PCR3）と，ポートデータレジスタ（PDR3）を利用する．P30 は H8/3687 では，I/O ポートのポート 3 に接続されている．また，P30 のみではなく，P30 から P37 の 8 つの端子はポート 3 で一元管理される．



コネクタのP30からP37の8個の端子は，それぞれCPU側のP30～P37に繋がっている．

このP30～P37をまとめてPort3と呼ぶ

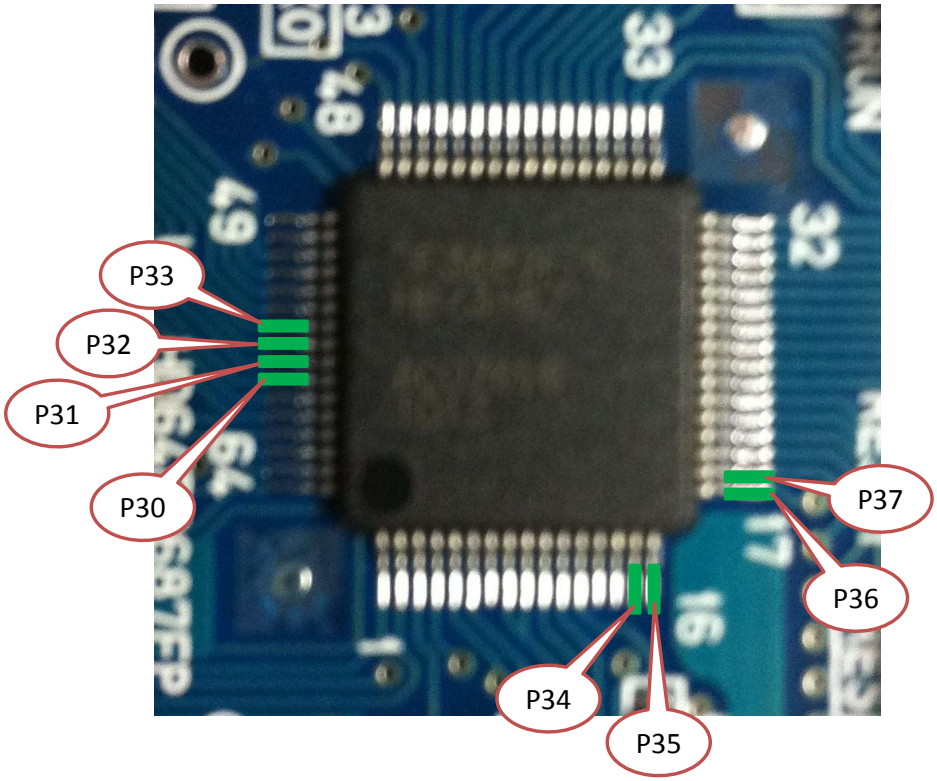


図　Port3（P30～P37）の場所（上図：ベース基盤上コネクタ，下図：CPU）

表　ポート３の端子構成

ポート３							
ビット７	ビット６	ビット５	ビット４	ビット３	ビット２	ビット１	ビット０
P37	P36	P35	P34	P33	P32	P31	P30

ポートコントロールレジスタ３（PCR3）では，ポート３に接続されている端子を，入力端子として使用するか，出力端子として利用するかを指定する．下表は，ポートコントロールレジスタ３の機能についての仕様である．表の説明にあるように，ビット単位で対応する端子の入力／出力を選択することが出来る．ただし，ポートコントロールレジスタは，ポート単位で指定するため，ビット単位での直接指定はできない．

ここで， P33 と P36 を入力端子に設定し，それ以外は出力端子に設定する場合を考えてみる．この場合， PCR3 に，２進数で 10110111 すなわち 16 進数で 0xB7 を設定すれば良いことが分かる（なお，「IO」は入出力ポートを制御するための構造体を表している．）．

```
IO.PCR3 = 0xB7;
```

表　ポートコントロールレジスタ３（PCR3）の仕様

ビット	ビット名	初期値	入力(R)／出力(W)	説明
7	PCR37	0	W	このビットを１にセットすると，対応する端子は「出力」となり，０にセットすると「入力」となる．
6	PCR36	0	W	
5	PCR35	0	W	
4	PCR34	0	W	
3	PCR33	0	W	
2	PCR32	0	W	
1	PCR31	0	W	
0	PCR30	0	W	

次に，ポートデータレジスタ３（PDR3）について見てみる．PDR3 では，実際に P30 に High=5[V]を出力するのか，Low=0[V]を出力するのか，を設定する．PDR3 の各ビットに対応する端子に出力する値（High ならば 1，Low ならば 0）をビット単位で指定することができる（ポートコントロールレジスタはビット単位では指定できなかった）．

例えば，既にすべてのポート３すべての端子が出力に設定されている場合，P32 と P37 のみ High にし，それ以外は Low に設定する場合は，PDR3 の bit2 と bit7 を１にし，それ以外は 0 に設定すればよい．ビットを直接指定する場合は，入出力ポートを制御する構造体メンバ*「BIT.B△」（△にはビット番号 0～7 が入る）を使って，「IO.PDR3.BIT.B2 = 1」のように指定すればよい（「BIT」の部分は，正確には構造体メンバではなく，共用体名である．詳細はインクルードファイル 3687.h と C 言語の解説図書で調べよ）．この方法を使えば，次のようなプ

ログラムによって，指定することができる．

```
IO.PDR3.BIT.B0 = 0;
IO.PDR3.BIT.B1 = 0;
IO.PDR3.BIT.B2 = 1;
IO.PDR3.BIT.B3 = 0;
IO.PDR3.BIT.B4 = 0;
IO.PDR3.BIT.B5 = 0;
IO.PDR3.BIT.B6 = 0;
IO.PDR3.BIT.B7 = 1;
```

また，上記の処理は，BYTE 単位にすれば，下記のように一括しておこなうこともできる．

```
IO.PDR3.BYTE = 0x84;
```

表　ポートデータレジスタ３（PDR3）の仕様

ビット	ビット名	初期値	入力(R)／出力(W)	説明
7	P37	0	R/W	ポート３の出力値を設定する
6	P36	0	R/W	
5	P35	0	R/W	
4	P34	0	R/W	
3	P33	0	R/W	
2	P32	0	R/W	
1	P31	0	R/W	
0	P30	0	R/W	

４　課題

先のプログラムを参考に，次の４つの課題を行え．

課題１：実験テーマ 10 で計測した結果をもとに，sleep(100)が丁度 1 秒の待ち時間となるように，プログラム中の（A）に適切な数値を入れて完成させよ．また，レポートには，なぜその値になったのか，導出の過程を説明すること（ヒント：テーマ 10 のときのプログラムでは，（A）には 10000 が入っていた．このとき，引数を 200 にすると，1 分間に点滅を約 50 回おこなった）

課題２：上記の点滅させるプログラムのフローチャートを書き，各処理について説明せよ．

課題３：ポート３について，P30，P31，P35 を出力，それ以外の端子を入力として利用する場合，PCR3 にはどのような値を設定すれば良いか．

課題４：３つの LED（緑と黄と赤）を用意して、以下のようなパターンで点滅するプログラムを作れ。ただし，緑の LED は P30 を使用し，黄の LED は P33，赤の LED は P34 を利用せよ)．また，レポートには，各プログラムの動作について詳細な説明を加えること．なお，各パターンの詳細は配布した動画で確認せよ．

（１）信号機パターン：

緑色，黄色，赤色の順に１つずつ信号機のように点灯する．緑，黄，赤の表示時間はそれぞれ 10 秒，３秒，10 秒とする．赤色の点灯時間が終われば，再び緑色の点灯に戻ること．

（２）イルミネーションパターン：

次のような順序で各色を点灯させる．

緑のみ点灯（2 秒）　→　緑と黄のみ点灯（2 秒）　→　緑と黄と赤（2 秒）　→　全消灯（４秒）　→　最初に戻る

（３）2 進数カウントパターン：

LED の点灯（ON）／消灯（OFF）を，2 進数 1 ビットの 1（High）／0（Low）に見立てて，緑・黄・赤の 3 色で 3 ビットの 2 進数をカウントアップするように LED を点灯させよ．具体的には，緑を 1 ビット目，黄を 2 ビット目，赤を 3 ビット目とし，下図のように 3 色を横にならべて，0 から 7 までの数字を順に増やすようにせよ．なお，7 の次は 0 に戻るものとする．また，このプログラムでは，PDR は BYTE 単位で制御せよ．また，各状態の点灯時間は 2 秒程度にせよ．

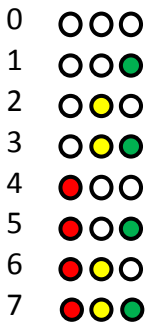


図　3 ビットの 2 進数を LED で表現

実験を行う前に確認すべき注意事項

１．ベース基盤上の余計なスイッチには手を触れない（組み合わせによっては回路がショートする）

２．CPU 基盤・ベース基盤を手でベタベタ触らない.

３．コネクタの抜き差しは慎重におこない，無理な力が掛からないように注意する.

４．説明書をよく読んで，順番に操作する（勝手に操作することによって，装置を壊す危険性があります）.

５．電源を ON にするときは，回路の接続が正しいかを再度確認すること.

６．作業に自信がないときは，教員に相談し，確認してもらう.

７．細かい部品は，無くさないように注意をする．それでも，無くしてしまった場合はすぐに教員に報告しておくこと（次に使う学生が困ります）.

８．装置や部品は，勝手に実験室から持ち出さないこと.

９．実験終了後は，必ず使った部品の個数を確認して，元の通りに戻すこと.