1. PIC の実習

1.1 回 路

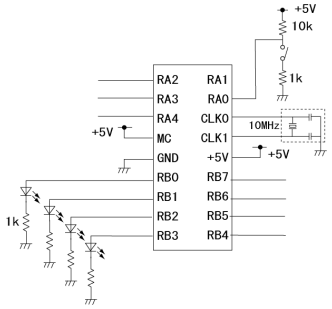
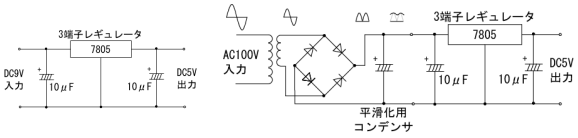


図 1.1 LED 点灯用 PIC 回路

(1)電 源



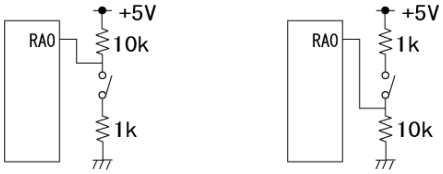
(a)DC9V→DC5V　　(b)AC100→DC5V 代表的な回路

(2) LED の点灯

(a) RB0 が High なら点灯
RB0 が Low なら消灯

(b) RB0 が High なら消灯
RB0 が Low なら点灯

(3) スイッチの回路



(a) SW が押されると RA0 は Low，開放なら High

(b) SW が押されると RA0 は High，開放なら Low

1.2 PIC について

PIC（Peripheral Interface Controller）は，コンピュータの周辺に接続される周辺機器との接続部分を制御するために開発された「マイクロコントローラ」と呼ばれる IC です．プログラムの書き換えが可能でプログラムについては 100 回，EEPROM の書き換えは 10 万回まで保証されている．

使用する PIC(16F84A) には，A ポートが 5 つ（ただし RA4 は特殊），B ポートが 8 つあります．それぞれ入出力を設定し，入力，出力することができます．**また，16F84A の場合 PIC に書き込めるプログラムメモリサイズ 1024words，EEPROM メモリサイズは 64byte と決められています．**

(1) **ポートの入出力を決定する**

(a) TRISA = 0x0F; //入力設定

A port				A4	A3	A2	A1	A0
値				0 ―	1 in	1 in	1 in	1 in

(b) TRISB = 0x00; //出力設定

B port	B7	B6	B5	B4	B3	B2	B1	B0
値	0 out	0 out	0 out	0 out	0 out	0 out	0 out	0 out

(2) **出力**

(a) RB0 だけ設定	B0=1;
(b) B ポートを設定	PORTB = 0x01;

(3) **入力**

図 1.1 の回路で RA0 のスイッチが押されるまで待機する場合．以下のようにプログラムできる

```
while(1){
  if( RA0==0 ){
    break;
  }
}

または while(RA0);
```

また，ポートをビット演算の論理で判定することもできる．

```
while(1){
  if((PORTA & 0x01)==0x01){
    break;
  }
}
```

1.3 プログラム例

【例 1】 回路 1 において B0 の LED を点灯する

```
#include <pic1684.h>

main(void){
  // ポート入出力初期設定
  TRISA = 0x0F; //入力設定
  TRISB = 0x00; //出力設定
  PORTB=0x00;
  RB0=1;
}
```

【例 2】回路 1 において B0 から B4 のポートの LED を点灯します．ただし，スイッチ A0 が押されると speed が 2 倍になる．関数 DelayUS はμsec 待機する関数．DelayMs は msec 待機する関数．

```
#include <pic1684.h>

#define XTAL_FREQ 10MHZ /*Crystal frequency in MHz */
#define MHZ *1000

void DelayUs(unsigned char cnt){
  unsigned char i;
  i = (cnt)/(12MHZ/(XTAL_FREQ))|1;
  while(--i != 0)
    continue;
}

void DelayMs(unsigned int cnt){
  unsigned char i;
  do {
   i = 4;
   do {
       DelayUs(250);
    } while(--i);
 }while(--cnt);
}

main(void){
  unsigned int speed;

  // ポート入出力初期設定
  TRISA = 0x0F; //入力設定
  TRISB = 0x00; //出力設定

  PORTB =0x00;
  speed=1;
  while(1){
    while(1){
      if( RA0==0 ){
        speed=speed*2;
        if(speed>1000) speed=2;
        break;
```

```
    }
  }
  PORTB = 0xF1;
  DelayMs(1000/speed);
  PORTB = 0xF2;
  DelayMs(1000/speed);
  PORTB = 0xF4;
  DelayMs(1000/speed);
  PORTB = 0xF8;
  DelayMs(1000/speed);
  PORTB = 0xF8;
  DelayMs(1000/speed);
  PORTB = 0xF4;
  DelayMs(1000/speed);
  PORTB = 0xF2;
  DelayMs(1000/speed);
  PORTB = 0xF1;
  DelayMs(1000/speed);
  PORTB =0;
 }
}
```

1.4 その他

PIC16F84A には，EEPROM への書き込みや読み込み，割り込みなどの機能もある．

2. PIC へのプログラム書き込み方法

2.1 ソフトウェア

ここでは MPLAB（Microchip 社）,PICCLite（HiTech 社）,PIC programmer（秋月電子通商）のフリーソフトウェアを用いて，コンパイルおよびパソコンから PIC へのプログラムの転送を行う．ソフトウェアはそれぞれインターネットなどで入手することができる．PICCLite は，C 言語のコンパイルを行うソフトウェアで，DOS プロンプトからコマンドで操作するソフトである． MPLAB は，この DOS プロンプト上の面倒な操作を GUI(Graphical User Interface）により仲介する役割を担う．

2.2 MPLAB と PICCLite

(a) MPLAB を起動する．
(b) 新規プロジェクトの作成．「Project」の「New Project」を選択する．図 2.1 で保存先を選択し，File Name にファイル名を入力する（拡張子は.pjt）．図 2.2 で「Development Mode」→MPLAB SIM PIC16F84A を選択．「Language Tool Suite」→HI-TECH PICC Lite を選択．(「Project」→「Install Language Tool」を開き，Language Suite が HI-TECH PICCLite，Tool Name が PICC Lite Compiler，Executable が C:¥PICCLITE¥BIN ¥PICL.EXE になっていることを確認．)

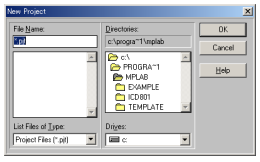


図 2.1 保存先設定

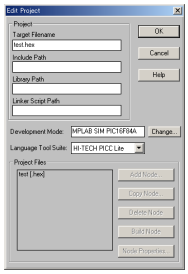


図 2.2 コンパイラ設定

2.3 ソースファイルの作成

「File」→「New」を選択し，ソースプログラムを書き保存する（拡張子は.c）．

2.4 プロジェクトにソースファイルを含める

「Project」の「Edit Project」を選択する．Project Files の一番上のプロジェクトを選択し，「Node Properties」を押す．図 2.3 のように「Floating Point for double」で 24-bit を選択．「ＯＫ」を押す．「Add Node...」を選択し，2.3 で作成したソースファイルを選ぶ．

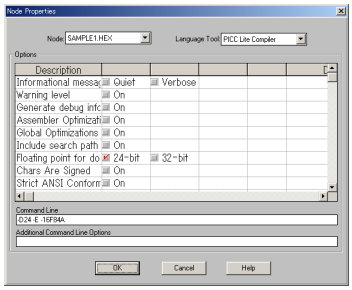


図 2.3 コンパイルオプション

2.5 コンパイル

「Project」→「Make Project」を選択．Build Results に Build completed successfully.が表示されればＯＫ．このとき，**「Total ROM used」の表示が 1024words（PIC16F84A を使う場合）より少なくなければならない**．

2.6 hex ファイルを PIC に転送

2.5 により作成された hex ファイルを PIC Programmer を使い PIC へ転送する．PIC をセットし転送する．（PIC の向きに注意すること）．

(a) PIC Programmer を起動する．
(b) Hex ロードを押し，ファイルを選択
(c) デバイス設定は PIC16F84A，FOSC は HS，WDTE は Disable，PWRTE は Enable にする．CP は Disable.
(d) プログラムボタンを押し書き込み．

■コラム（C 言語）

16 進数　0x‥‥　　0xFF =255

ビット演算について

論理積（&）	(0x01 & 0x0F)　=0x01
論理輪（｜）	(0x02 ｜ 0x07)　=0x07
排他的論理和（^）	(0x02 ^ 0x07)　=0x05
ビット反転（～）	(~0x02)　=0xFD
bit シフト右（>>）	(0x08 >> 2)　=0x02
bit シフト左（<<）	(0x03 << 1)　=0x06

3. ライントレーサコンテスト

3.1 PIC 回路

ライントレーサ用の PIC 回路を図 3.1 に示す．RB0～RB7 については 3.2 において説明する．

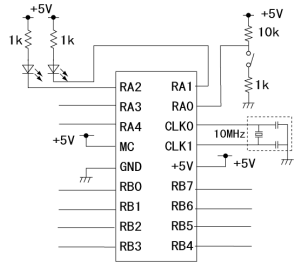


図 3.1 ライントレーサ用 PIC 回路

3.2 ライントレーサの制御

ライントレーサとの接続には 11 ピンコネクタを用いる．コネクタの仕様は表 3.1 のとおりである．

表 3.1　ライントレーサとの接続を行う 11 ピンコネクタの仕様

1	NC		
2	+5V		
3	GND		
4	RB0	右モータ（前進）	
5	RB1	右モータ（後進）	
6	RB2	左モータ（前進）	
7	RB3	左モータ（後進）	
8	RB4	右中 S	黒 1 白 0
9	RB5	左中 S	
10	RB6	右外 S	
11	RB7	左外 S	

3.3 競技内容

(1) ライントレーサを PIC により制御する．
(2) 競技コースを図 3.2 に示す．
(3) ライントレースし，スタートからゴールへ到達すること．
(4) スタートはポート A0 のスイッチにより動作を開始すること．
(5) ゴールに到達後は停止すること．
(6) ポート A1,A2 に接続されている LED を有効に活用すること．
(7) ライントレーサの電池の消耗を考慮する場合は，電池を持ち込んでも良い（単 3 電池 4 本）．

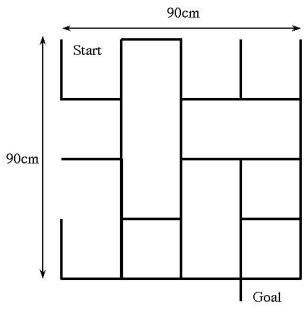


図 3.2 競技コース
（ラインは白，背景は黒とする．）

3.4 日　程

2 月 4 日	12:50～	競技会
2 月 18 日	8:50	レポート提出
	12:50 ～	プレゼンテーション

3.5 評価のポイント

技術点，プレゼンテーション点，提出レポート，取り組み姿勢により後期期末の評価を行う．

レポートについては，目的，実験の基礎，使用器具（使用ソフトウェアも含む），結果，考察，参考文献を記すこと．

実験の基礎については，「回路素子の解説」，「LED 点灯回路（図 1.1）の説明」「ライントレーサ回路（図 3.1）の説明」を詳細に記すこと．また，レポート末尾には，自作プログラムのリストを添付すること(リストについてはプリンタ出力でも認める)．

ライントレーサのレポートについては，通常の実験レポートの 3 倍以上を期待する．

PIC 役立つホームページ

■マイクロチップテクノロジー社
マイクロチップテクノロジー
http://www.microchip.com/
マイクロチップテクノロジージャパン
http://www.microchip.co.jp/

PIC16F84 データシート
http://www.microchip.co.jp/30430c-j2.pdf
（※124 ページあるので印刷しないこと）

MPLAB のページ
http://www.microchip.com/1010/pline/tools/picmicro/devenv/mplabi/mplab6/index.htm

■HI-Tech 社
http://www.htsoft.com/products/piclite/piclite.html
（PICC Lite は HI-Tech 社から提供されてます．）

■秋月電子通商
http://akizukidenshi.com/

■その他
電子工作の実験室
http://www.picfun.com/

PIC 工作
http://www.gulf.or.jp/~mosaku/make_top.htm

羽渕先生のページ
http://www.gifu-nct.ac.jp/elec/habuchi/habuchi.html

補　足

1. 割り込み

あるプログラムが実行している最中に，あるきっかけで他のプログラムを実行させることを，割り込みといいます.

1.1 タイマーの利用

図１の回路において, 一定時間(1sec)ごとに割り込みをいれLEDを点灯する.１秒間点灯し１秒間消灯.

タイマーの設定

タイマーTMR0 がオーバーフローするまでの時間は，初期値 TMR0=0x00 が 0xFF になるまでにはクロック周波数（10MHz）なので，以下のように求められる．割り込み周期=(1/10MHz)×4×256=0.1024ms．ここでプリスケーラ（倍率）を 256 と設定すると，割り込み周期は，0.1024ms×256=26.21ms となる.（つまり，interrupt 関数が呼び出されるのは 26.21ms ごと.）26.21ms を 38 回カウントすると約 1000ms=１秒.

```
#include "pic1684.h"

int cnt;                          割り込みが何回発生したか数える

void interrupt isr(void){         割り込みの関数(26.2msec ごと)
   if(T0IF==1){                   もし，割り込みがあったなら
      T0IF=0;                       割り込み状態フラグを 0 にする
      cnt--;                        カウントダウン
   }
   if(cnt==0){                    カウントが 0 になったら
      RB1=1-RB1;                    RB1 を反転　点灯⇔消灯
      RB2=1-RB2;                    RB2 を反転　点灯⇔消灯
      cnt=38;                       カウントを 38 にリセット
   }
}

main(){
   TRISA=0xFF;                    入出力設定Ａポート
   TRISB=0x00;                    入出力設定Ｂポート
   PORTA=0xFF;                    ポートＡの初期化
   PORTB=0x00;                    ポートＢの初期化

   OPTION=0x87;                   プリスケーラ（倍率）を 1:256
   TMR0=0x00;                     タイマーの時間設定
   T0IF=0;                        割り込み状態フラグを０にする
   T0IE=1;                        タイマー0 を有効にする
   GIE=1;                         割り込みを有効にする
   cnt=38;                        カウントを 38 にリセット
   while(1);
}
```

1.2 RB4～RB7 の値の変化で割り込みをする

RB4～RB7 の値の変化で割り込みを発生する場合は,
　RBIE　RB の割り込みが有効／無効
　RBIF　RB の割り込み状態フラグ
を用いる.

2. EEPROM

EEPROM とはデータを格納しておく領域（倉庫みたいなもの)のことで, この領域は電源の ON,OFF などでも消えません．不揮発性メモリといいます.（電源が消えても覚えておきたいパラメータの保存に利用されます．PIC programmer で新しいプログラムを書き込むと消えてしまいます.）PIC16F84 では EEPROM の大きさは 64byte（char 型で 64 文字分）と決められていて，アドレス（倉庫の住所）は 10 番地（10 進数の 10）からです．書き込み回数が制限されているため，EPPROM をプログラムの変数として使うことは避けましょう.

番地	10	11	12	13	14	・・	73
内容							

10 番地に 128 を保存

```
EEPROM_WRITE(10,128);
```

10 番地にあるデータを読み出す

```
char i;

i=EEPROM_READ(10);
```