■■　第１週目　Ｈ８マイコンと H8cc 開発環境に慣れる　■■

２．Ｉ／Ｏポートと時間の実験

学生諸君は，これまでの学習でマイコンボードの製作や H8cc などの開発ツールをインストールし，マイコンの開発環境を整えてきました．

ここでは，マイコンを利用するにあたって最も基本となるデジタル情報の入力や出力の実習を行ないます．他の講義で学習するプログラミング技術と共に，本章で演習するデジタル信号処理の実習により，マイコンに関わるハードウエアやそれを実行するプログラムの理解を深め，さらにこれらを演習しながら H8 マイコンの開発環境に慣れていただき，なおかつ，学生諸君が各々の力で独自にマイコンを利用できるための能力としてハードウエアマニュアルも見ることができる，その様な総合力を養うことを学習の目的とします．

１．基本動作の確認事項

・デジタル I/O ポート(input / output port)とＬＥＤとスイッチによるデータの受渡し
（port: 外部とのデータ受け渡し，外部との入出力端子）
P1-0～P1-7　ＬＥＤ８個　　点灯，消灯制御
P2-0～P2-3　タクトスイッチ　４個の信号取り込み

・ITU(integrated timer counter unit)と時間制御の基本動作
５つのＩＴＵから，ＩＴＵ０を使用した遅延時間の作り方

２．ポート回路と動作原理

デジタル IO ポートは，CPU と外部信号の中継を担います．ポートの入出力ピンにスイッチが入力として，LED が出力として接続された回路を図 1 に示します．スイッチは電源に対して抵抗と直列接続し，抵抗とスイッチの接続点は入力ピンに接続されています．これによりスイッチを押すと，ポートには０Ｖが入力され，放すと５Ｖが入力されます．また，LED はポートの出力が１ならば LED は点灯，０ならば消灯します．

機械的なスイッチを用いた論理回路を図２に示します. 入力が０Ｖなら sw1 がＯＮ, sw2 がＯＦＦで，５Ｖなら sw1 がＯＦＦ，sw2 がＯＮになるスイッチとすると，入力が０Ｖの時にこの回路は５Ｖ(Hi)を出力し，入力が５Ｖの時に０Ｖ(Low)を出力するスイッチ回路として働きます．この機械的なスイッチを２つの極性が逆で同一特性のトランジスタ（CMOS）に置き換えた回路を図３に示します．sw1 から手を放した状態では，トランジスタのゲート電圧に５Ｖ加わり，P ch がＯＦＦ，N ch がＯＮとなり,LED は消灯します．逆に，sw1 押した場合はゲート電圧が０Ｖとなり，Pch が ON，Nch が OFF となり LED は点灯します．また，この回路における抵抗 10kΩ は，sw1 を解放したとき，論理レベルが H となるように電位を持上げています．これをプルアップ(pull up)抵抗と呼んでいます．

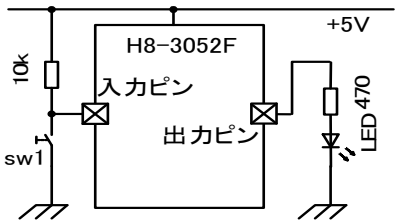

図１　ポートとスイッチ，LED 回路

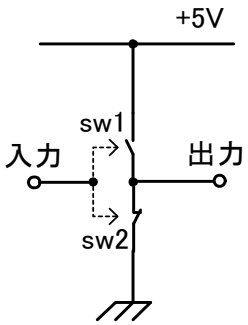

図２　スイッチによる
論理回路

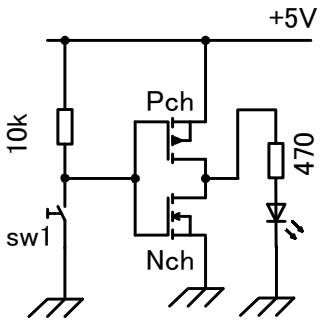

図３　ＣＭＯＳトランジスタ
によるＮＯＴ回路

一方，マイコンの各 I/O ポートは，入力と出力が同一端子になっています．この入出力を切り替えているのが，データ・ディレクション・レジスタ（Data Direction Register :DDR）です．図４は，データレジスタ（Data Register:DR）のビット５に１がセットされ，それにより P1-5 端子が出力端子となって LED が点灯している様子を示しています．

同様に，スイッチ入力として I/O ポートを利用した回路を図５に示します．特に，ポート２（Ｐ２）では，プルアップ抵抗と同じ働きをする回路の接続と切り離しがソフト的に設定できます．これを入力プルアップ MOS コントロールレジスタ（PCR）といいます．

各レジスタは８ビットの長さがあり，それぞれの I/O ポートの入出力設定を１バイト，あるいは１ビット単位で設定することができます．

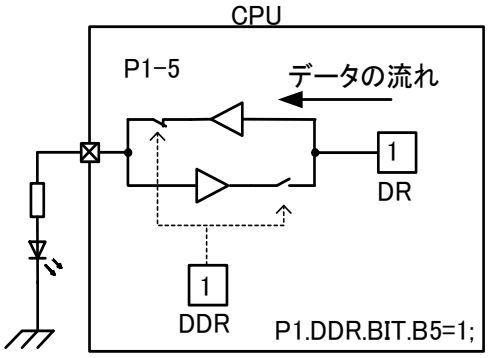

図４　出力設定時の I/O ポート

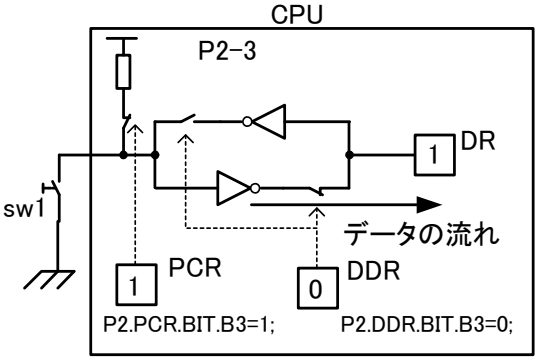

図５　スイッチ入力時の I/O ポート

３．マイコンを活用するための基本プログラム

この節での学習のねらいは,H8 マイコンと開発環境を利用したマイコンの利用法に慣れながら,出力ビットの制御に慣れることとします. 社会で活用されている家電, 産業機械, ロボットなどの全てがこれから学習する出力ビットでコントロールされています.つまり, 制御をつかさどる原点が出力ビット（ポート）操作であり，マイコン制御の正体です．マイコンを使いこなすということは，いかに出力を操作できるかということにほぼ等しいことになります.

【演習 1】LED の点灯制御（出力制御のウオーミングアップ）

ＬＥＤを８個点灯させるプログラム Prg1-1d を打ち込み実行します．なお，授業の進行上, **保存場所とファイル名は h8_toin¥user¥test¥Prg1_1d.c** と, 皆さん統一してください.

【１０進数を扱って出力する場合】

Prg1_1d.c（保存ファイル名）

```
#include<3052f.h>
main()
{
  P1.DDR= 0xff;      /* ポート 1 を 8bit 出力モードに初期設定  */

  P1.DR.BYTE=255;    /* 全 LED 点灯する値を代入（これがプログラム本体）*/
}
```

【マイコンの停止の方法】実行するとＬＥＤ８個が点灯し，それ以上は何もしません．消灯させるには，赤色の reset スイッチを押します.

【１６進数を扱って出力する場合】

Prg1_1h.c（保存ファイル名）

```
#include<3052f.h>
main()
{
    P1.DDR= 0xff;        /* ポート 1 を 8bit 出力モードに初期設定 */

    P1.DR.BYTE=0xff;     /*全 LED 点灯の値を代入(これがプログラム本体)*/
}
```

【マイコンの停止の方法】　P1.DR.BYTE に代入する値を上記のように変更し,実行してもＬＥＤ８個が点灯するか確認してください．この例も，それ以上は何もしません．消灯させるには赤色の reset スイッチを押します.

課題１　上記のプログラムを実行すると，LED が 8 個点灯します．このとき，このプログラムはマイコンの中でどのような状態になっているのか簡潔（1 行～2 行以内）に答えなさい．

課題２　main のプログラム内の「P1.DDR」と「P1.DR.BYTE」に関する内容は，ハードウエアマニュアルの何ページに書かれているか，そのページ数を答えなさい．（＊＊ページのように）また，ハードウエアマニュアルと上記のように実際にコーディングするプログラムの記述とが等しいかどうか簡潔に答え，これらをどう対応付ければ良いか簡潔に述べなさい．

課題３　プログラム Prg1_1d.c で，P1.DR.BYTE=255 の行の 255 の部分に，１を代入して実行しなさい，同様に２，４，８，１６，３２，６４，１２８をそれぞれ代入して実行したとき，LED がどのように点灯するか報告しなさい．

課題４　この　１，２，４，８，１６，３２，６４，１２８　の 8 個の数で，自分の好みの数を選び（複数選ぶのも可），選んだ数を全て加算した値を出力した場合，LED はどのように点灯したか簡潔に答えなさい．また，これらの 8 個の数と LED の点灯箇所との関係を報告しなさい．

課題５　LED が接続されているポート１が出力 Low レベルの際，流し込める最大の電流が何 mA か答えなさい．また，それがハードウエアマニュアルの何ページに書かれているか答えなさい．

～　マイコンハードウエアに関するヒント　その１　～

マイコンは基本的に内部も外部との入出力部も電圧レベルで制御されています．そのため I/O ポートから大きな電流を流す能力がない場合が多々見受けられます．このような場合，つまり図４の回路で点灯させられない場合，図６のように５V 電源から LED と抵抗の直列に接続することで点灯させられます．マイコン側へ電流を流し込む回路の場合は，（限度はありますが）比較的電流を流し込むことができます．この電流の出力値や最大値はハードウエアマニュアルを参照することで確認できます．また,これを「電流を引っ張る」,「電流をマイコンに引き込む」というような言葉で表現されます．図６の回路では，レジスタに与える各ビットの値は，０であれば点灯，１であれば消灯になります．

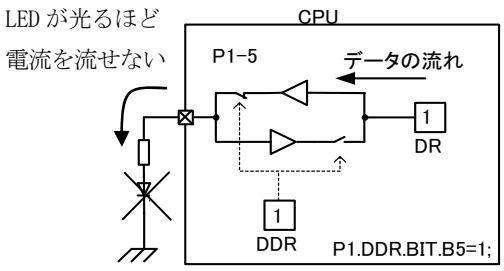

図４出力設定時の I0

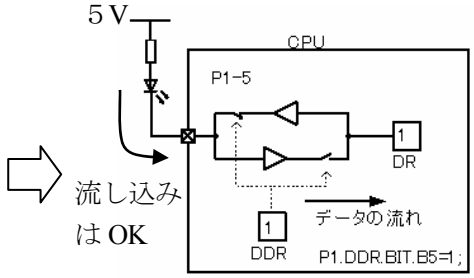

図６　負論理の LED 回路

～マイコンハードウエアを操るための基礎知識　その２　～

Ｉ／Ｏポートレジスタの種類と使用例

H8-3052F の I/O ポートは, P1～P6, P8～PB であるが, 各ポートごとに機能が異なるため, 詳細は H8-3052F ハードウエアマニュアル参照して下さい.

データディレクションレジスタ（DDR）

		使用例
P1．DDR．BYTE	8bit	P1.DDR.BYTE=0xFF;
P2．DDR．BIT．B2	1bit	P2.DDR.BYTE=0x00;

入力プルアップMOS コントロールレジスタ（PCR）

P2.PCR.BYTE	8bit	P2.PCR.BYTE=0x0F;
P2.PCR.BIT.B4	1bit P2,P4,P5のみ	

データレジスタ（DR）

P1．DR．BYTE	8bit	P1.DR.BYTE=230;
P2．DR．BYTE	8bit	P1.DR.BYTE=0xE3;
P4．DR．BIT．B3	1bit	P1.DR.BYTE=a;
P1．DR．BIT.B0	1bit	a1=P2.DR.BYTE;
P4．DR．BIT．B3	1bit	P1.DR.BYTE=P2.DR.BYTE; P1.DR.BYTE= ~P2.DR.BYTE; while(P2.DR.BIT.B0==1)

～実際のプログラムで利用する上記レジスタの意味と対応～

これはあくまでハードウエア（マイコン側）で規定されているコマンドです. ですから, 実際のプログラムで記述する際とは，若干異なります．マイコン側のコマンド記述とプログラムでの記述を対応付けているのが，先ほど演習したプログラム中にあった，#include<3052f.h>　です．みなさんが記述した 5 行のプログラム（main（）を含めた 5 行）だけでは，実行しても LED は点灯しません．実行する際，3052f.h がないと機械語を生成するコンパイラは main()内で書かれたコマンドが何であるかわからないのです．余裕のある人は，#include<3052f.h>　の 1 行を削除して試してください．この#include のという命令（プリプロセッサ）が，3052f.h というヘッダーファイルに記述してある沢山のプログラムとみなさんのプログラムとを統合することにより，ハードウエア側の各レジスタのアドレスとソフトウエアで記述したレジスタコマンドを対応することができるのです.

今後皆さんが簡単に入出力を行えるようになるのは，全て「3052f.h」というファイルのお陰というわけです.

～マイコンハードウエアを操るための基礎知識　その３　～

１０進数と２進数，１６進数の関係を改めて確認します．

私達の生活の中では，通常，算数や数学あるいは日常の買い物などにおいて，１０進数を利用しています．これは１０になると桁を上げる数です．しかし，演習１で体験したようにコンピュータを扱う上では，２進数や１６進数も理解しておく必要があります．それぞれ特徴や関係がありますので，ここで改めて実践を通して身に付けてください．

２進数：これは，０，１と数えて２になるときは２にならずに一桁繰り上げる数です．

１６進数：１６進数は，０，１，２・・と数え，１０になっても桁上げせずに，１６になったときに一桁上げます．ただし，１０以上の数字は標記する自体が２桁になってしまうので，便宜上，A, B, C, D, E, F と記し１桁に保たせています．

10 進数と 2 進数と 16 進数の関係

10 進数	2 進数	16 進数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
１０	１０１０	A
１１	１０１１	B
１２	１１００	C
１３	１１０１	D
１４	１１１０	E
１５	１１１１	F
１６	１００００	１０

ここで１桁上がっている

ここまでがひとまとまり
２進は 1111 が４つ
16 進は１桁目の最後

ここで１桁上げる

マイコンの出力端子は８本単位でひとまとまりなっているため，Hi が１，Low を０としたとき，２進数で表現すると都合がよくできています．それに対して，プログラム中で扱う数値は，やはり１０進や１６進が扱いやすいです．なぜなら,２進数では,桁数ばかりが増えてしまうので記述するには不便ですし,見にくいのです．ですから,プログラムの中の数と出力先の状態を対応するには，１０進と２進の関係を理解することが必要になります．一方，２進は桁数が大きくなってしまう欠点がありますが,２進数の４桁は，上表のように１６進数が１桁で対応しますので，２進数を１６進で表せれば，４桁分の２進数を１桁の１６進数で表現できることがでるのです．なお,８Bit（８桁）を１６進数で表現すれば，２桁の１６進数で表現します．

～マイコンハードウエアを操るための基礎知識　その４　～

■ 8bit マイコンと１０進数・２進数･１６進数の関係

下表の２進数の列に注目してください．演習１の LED の点灯と関連付けて，消灯している状態が「０」で点灯している状態は「１」と見てみましょう．すると **P1.DR.BYTE** に代入した数値（１０進数）と LED の点灯との関係が見えてきたでしょうか．そうです．**P1.DR.BYTE** に代入した１０進や１６進数は，２進数で LED に表示されていたのです．ここで演習１で使用した８bit の LED は，４桁の２進数のまとまりが“２つ”あると考えてください．また，換算方法は様々ありますが，ここではそれぞれの“１”には表の上段の重みの数が対応していると覚えてください．

２進数の重み付け

重み	128	６４	３２	１６	８	４	２	１
２進	1	1	1	1	1	1	1	1

⇦ **255**

F　　**F**

■ 8bit の LED 制御と１０進数・２進数･１６進数の数値換算

(1)　１０進数と２進数の換算

次のような関係になっています．例えば，（０００１　１０１０）という 2 進数があった場合，１の部分の重み数を加算すれば，その 2 進数に対応する 10 進数に変換できるのです．つまり，この例の（０００１　１０１０）場合は，上表より対応させれば，（０＋０＋０＋１６＋８＋０＋２＋０）＝**２６**となるのです．

従って，LED を　「消」「消」「消」「灯」「灯」「消」「灯」「消」と点灯させたい場合に，演習1のプログラムで **P1.DR.BYTE**＝26 と記述すると，意図する点灯の制御が実現します．

(2) ２進数と１６進数の換算

同様に，（０００１　１０１０）を 16 進数で表すにはどうしたらよいでしょうか．０と１は 4 桁のまとまりとして見れば良く，０００１と１０１０を前頁の表で対応させて考えると，０００１は１で，１０１０はＡなので，（０００１　１０１０）

１　　Ａ

は，“**１Ａ**”と２桁で表現することができます．

演習１の **P1.DR.BYTE** に１A を代入すると，LED は同じように点灯します．ただし，プログラムの中でこの数は 16 進ですよと，コンパイラに知らせるために 16 進数を代入する時は．

P1.DR.BYTE＝０ｘ１Ａ；

１A が 16 進であることを示しています

と記述します．時間に余裕のある者は試してください．

また，コンパイルによって生成される HEX ファイル（ヘキサは１６を表す）の中身は，この 16 進数で記述されています．いくら機械が０と１しか分からないといっても，2 進数で表示されたら桁数ばかり多くて読みにくいものです．しかし，8 桁(bit)の数が 2 桁で表現されれば，人は大変見やすくなる効果があります．HEX ファイルを見ればわかります．

【演習２】プログラムを継続的に動作させる手法

この演習課題が実際に**組み込みマイコンとして利用する場合の最も基本プログラム**になる大切な項目となりますので，しっかり理解し身に付けてください．

■ 無限ループ（永久ループ）によって，プログラムを囲み実行し続けさせます．
それでは演習しましょう．ポイントは，**“無限ループでプログラム本体を囲む”**のです．

Prg2_1.c（保存ファイル名）

```
#include<3052f.h>
main()
{
    P1.DDR= 0xff;  /* 8 bit 出力初期設定*/

    while(1){       /*ここから無限ループ*/

        P1.DR.BYTE=255; /*全 LED 点灯*/

    }               /*ここまで無限ループ*/
}
```

Prg1_1d（対比するﾌﾟﾛｸﾞﾗﾑ，4-20ﾍﾟｰｼﾞ）

```
#include<3052f.h>
main()
{
  P1.DDR= 0xff;  /*8 bit 初期設定*/
  P1.DR.BYTE=255; /* ﾌﾟﾛｸﾞﾗﾑ本体*/
}
```

■さまざまな無限ループの記述方法

Prg2_do_while.c（保存ファイル名）

```
#include<3052f.h>
main()
{
    P1.DDR= 0xff;       /* 8 bit 出力初期設定*/

    do{                 /*ここから無限ループ*/
        P1.DR.BYTE=255; /*全 LED 点灯*/
    }while(1);          /*ここまで無限ループ*/
}
```

Prg2_for.c（保存ファイル名）

```
#include<3052f.h>
main()
{
    P1.DDR= 0xff;/* 8 bit 出力初期設定*/

    for(;;){            /*ここから無限ループ*/
        P1.DR.BYTE=255; /*全 LED 点灯*/
    }                   /*ここまで無限ループ*/
}
```

課題６　上記のプログラムと Prg1_1d.c とで，動作の違いについて考察し，報告しなさい．
（**考え方のヒント**:見た目には演習１と同じようであるが,このプログラムにより,出力値は,I/O ポートにラッチされているのではなく，常に，P1.DR.BYTE の状態（代入されている値）を出力するようになっている）

【演習３】パソコンのキーボードによるマイコン操作（１０進入力２進出力）

パソコンと本ボードの通信ポート（ＲＳ２３２Ｃ）を接続し，パソコンのキーボードから入力した情報でＬＥＤの点灯を制御するプログラムを作成します（binary.c 参照）．
パソコンとの情報のやり取りを実現する関数（コマンド）は，次のサンプルプログラムで利用されている，printf や scanf などです．これらの関数は，プログラムの冒頭に書かれた #include によって「stdio.h」（スタンダード IO）というヘッダファイルを読み込む（記述する）ことで利用できるようになります．

組み込みマイコンのプログラムに慣れるために，最初の内はプログラムの構成も意識するようにしてください．

binary.c（保存ファイル名）

```
#include<3052f.h>
#include <stdio.h>     /*  printf ,scanf 用    */
main()
{
          int n1;

          P1.DDR= 0xFF;                /* P1－0～P1－7  8LED */

        while (1){                     /*無限ループ先頭*/
             printf(" Num : ");        /*メッセージ表示*/
             scanf("%d",&n1);          /*  １０進数値入力*/
             printf(" n=%d¥n: ",n1); /*  １０進表示  */
             P1.DR.BYTE=n1;            /*  ポート出力  */
        }                              /*無限ループ末尾*/
}
```

プリプロセッサの記述

初期設定の記述

メインプログラムの記述

このプログラムを実行すると，パソコン側のモニタ画面に「Num」と表示されるので，キーボードから０～２５５までの数を入力してください．それ以上入力しても，ポート 1 は 8ibt のポートのため，下位８ビット分だけの表示となってしまいます．

課題７　上記の binary.c を実行しなさい．その上で，このファイル名を binary00.c に変更して，無限ループの働きを確認するために，上記のプログラムの無限ループ部分を削除したプログラムを実行しなさい．また，その結果どうなったかを簡潔に答えなさい．

課題８　printf と scanf の働きについて述べなさい．また，この他の入出力関数としてどのようなものがあるか適宜，調べて述べなさい．（C:¥h8_toin¥cc¥ stdio.h のファイルを見ながら，C コンパイラユーザマニュアルを利用して調べること．また,このファイルを壊さないようにプロパティで読み取り専用にチェックしておくと良い）

【演習４】パソコンのキーボードによるマイコン操作と，ASCII コード学習

キーボードの文字や数字，記号には，８ビットの組み合わせによるアスキーコード(Ascii code: American Standard Code for Information Interchange)が割り当てられています．次のサンプルプログラムをコーディングし，１文字入力して，入力したパターンを 16 進数で表示するプログラムを実行してください．また，LED でも表示せよ（１は点灯，０は消灯）．

ascii.c（保存ファイル名）

```
#include<3052f.h>
#include <stdio.h>      /*  printf */

main(){
    int i;
    char n;
    P1.DDR= 0xFF;   /* P1-0-P1-7 8LED */

    P1.DR.BIT.B0=0;
    while (1){
        printf(" Char : ");
        scanf("%s",&n);
        printf(" n=%x¥n: ",n);
        P1.DR.BYTE=n;
    }
}
```

課題９　上記のプログラムを実行しなさい．また，printf と scanf の働きについて述べなさい．特に，printf や scanf の（　）で囲われている内部の違いについて調べて答えなさい．また，その他の変数の使い方（表現指示文字）利用方法などについて調べなさい．（C 言語関連の書籍等を参照すること）

課題10　パソコンモニタ画面のn＝の値，LED の点灯の状態，および ASCII コード表との関係について，2進数，10進数，16進数と任意の文字(例えば，A～Z)との対応を一覧表にまとめなさい．(下表にまとめ方の例を示す)

２進数	１０進数	１６進数	文字
100 0001	65	41	A
:	:	:	:

参考資料（記号に関するもの）

【ASCII コード表】

行＼列	00	01	02	03	04	05	06	07
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	″	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	IS4	,	<	L	¥	l	|
D	CR	IS3	−	=	M	]	m	}
E	SO	IS2	.	>	N	^	n	~
F	SI	IS1	/	?	O	_	o	DEL

表の上の数字は文字コードの上位３ビットを表し，左の数字は下位４ビットを表しています．また,この表の中の，SP は空白(半角スペース)，NUL は空または数値の 0 (Null 文字と呼ばれる¥0 もこれに当たります)，LF/NL は改行(New Line) ＵＮＩＸの改行，CR は復帰(Carrige Return) マッキントッシュ系の改行コードを示しています.

【記号の名称】

記号	名　称	記号	名　称
!	イクスラメーション	?	クエスチョンマーク
’	シングルクオーテーション，アポストロフィー	”	ダブルクオーテーション
#	シャープ	$	ダラー，ドル
%	パーセント	&	アンパサンド，ドル
(	左カッコ	)	右カッコ，閉じカッコ
{	左中カッコ，ブレース	}	右中カッコ，閉じ中カッコ，ブレース
[	左大カッコ，ブラケット	]	右大カッコ，閉じ大カッコ，ブラケット
「	左カギカッコ	」	右カギカッコ，閉じカギカッコ
−	ハイフン	^	山型，キャレット，ハット
¥	円	=	イコール
～	チルダ	|	縦棒，パイプ
@	アットマーク	;	セミコロン
:	コロン	＋	プラス
*	アスタリスク	,	カンマ，コンマ
.	ピリオド，ドット	・	中点
／	スラッシュ	＼	バックスラッシュ
<	小なり，ギュメ	>	大なり，ギュメ
＿	アンダーバー		

【演習５】パソコンのキーボードによるマイコン操作と条件分岐（if 文）を学ぶ

課題11　printf と scanf の関数を使ってキーボードからある数値を入力し，この数値が 128 未満なら下位 4 ビットの LED（左側４つ）が点灯し，128 以上ならば上位 4 ビットの LED（左側４つ）が点灯するプログラムを，下記を参照にしながらコーディングし実行しなさい.

Prg5_1.c(プログラム保存名)

```
#include<3052f.h>
#include <stdio.h>      /*  printf */
main(){
    int c;
    P1.DDR=0xff;

    while (1) {
        printf("128 未満なら右4つ点灯,128 以上なら左4つ点灯,999 で消灯¥n");
        scanf("%d",&c);
        if (c<128){
            P1.DR.BYTE=0x0f;
        }                                          (←IF  THEN  ELSE 型)
        else{
            P1.DR.BYTE=0xf0;
        }
        if (c==999){ P1.DR.BYTE=0x00; }          (←IF  THEN 型)
    }
}
```

【 条件分岐（if 文）について 】

if 文は，マイクロコンピュータを制御する上で非常に重要な項目です．特に，以下の２つの形を覚えることが大切です．この２つの流れを次のフローチャートで確認しておきます.

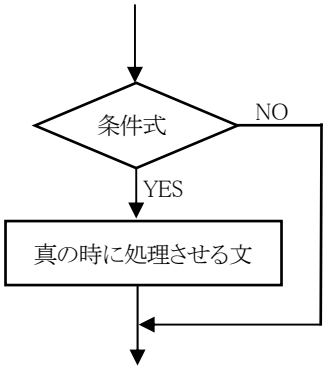

図７　IF　THEN 型

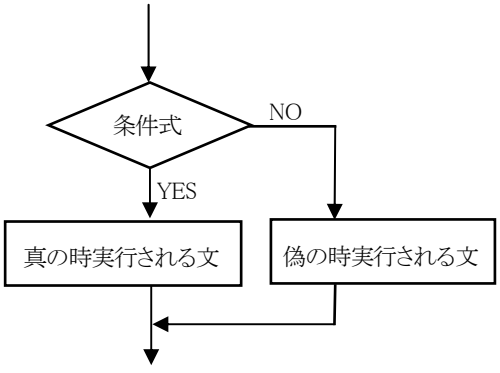

図８　IF　THEN　ELSE 型

【 条件分岐（if 文）記述について 】

ある条件によって複数の情報処理を区別し使い分ける制御は，if 文を使います．if 文は次の書式で示すように，条件式で演算した結果が成り立っていれば真（1），不成立である時は偽（0）として，その後のプログラムの流れ（処理）を変えることができます．

if 文の書式（書き方）

基本書式（IF　THEN　ELSE 型）	IF　THEN 型（Else 省略型）

if（条件式）
{
　実行文；
　実行文；
}　→ 条件式が真の時，処理される範囲
else
{
　実行文；
　実行文；
}　→ 条件式が偽の時，処理される範囲
	if（条件式）
	{
	　実行文；
	　実行文；
	}　→ 条件式が真の時，処理される範囲
	
	条件式が偽の時は｛　｝内は処理されずに次にジャンプする．

if 文の条件式で一般的に用いられる比較演算子と論理演算子（これだけではないので注意）

演算子		演算子の意味	具体例
比較演算子	＜	a が b より小さい（時に真）	if(a ＜ b){　文；　}
	＞	a が b より大きい(時に真)	if(a ＞ b){　文；　}
	＜＝	a が b より小さいか等しい(時に真)	if(a <= b){　文；　}
	＞＝	a が b より大きいか等しい(時に真)	if(a >= b){　文；　}
	＝＝	a と b は等しい(時に真)	if(a == b){　文；　}
	！＝	a と b は等しくない(時に真)	if(a ！= b){　文；　}
論理演算子	＆＆	論理積(AND)で全て条件を満たす	if((a＜b)&&(a＜c)){文;} a<b と a<c を満たせば真
	｜｜	論理和(OR)でいづれかを満たす	if((a＜b)||(a＜c)){文;} a<b か a<c を一方満たせば真
	！	否定(NOT)で，条件結果を否定	if(！(a＜b)){文;} a<b でなければ真

【演習６】タクトスイッチからの信号をマイコンに入力する基本プログラム

４個のタクトスイッチ S0～S3 を押すと，それらの上部の対応した位置に配置してある LED の L0～L3 が，点灯または消灯する作成し実行しなさい.

keyin.c（プログラム保存名）

```
#include<3052f.h>
main()
{
  int sw;
  P1.DDR= 0xff;       /* ポート１（8 個 LED）は出力設定  */
  P2.DDR= 0x00;       /* ポート２（4 switch）は 入力設定  */
  P2.PCR.BYTE=0xff;  /* ポート２をさらに pull up する設定*/

  while (1) {
          sw= P2.DR.BYTE;    /*ポート２に入力された情報を sw に代入 */
          P1.DR.BYTE=sw;     /* sw の値をポート１に出力*/
  }
}
```

上記のプログラムでは，変数 sw を介さず，P1.DR.BYTE= P2.DR.BYTE;でも可能です.

課題 12　以下の説明に従ってプログラムを修正して実行し，そのプログラムと動作の様子を報告しなさい．なお，修正したプログラム名は keyin_12.c で保存しなさい.

スイッチからの入力される各ビットを反転させるには，１の補数演算子（~）を次の様に，上記のプログラムへ書き加えます.

```
sw= ~P2.DR.BYTE;
```

タクトスイッチの情報を１６進数表示でモニタ画面に表示するには，上記の行の次に以下の文を書き加えます.

```
printf(" sw=%X: ¥n",sw);
```

注意：　printf 文を使うときは #include <stdio.h> を忘れないようにします.

何も押さなければパソコンのモニタ画面に，次のように表示されます.

```
sw=FF00:
```

タクトスイッチがない（接続されていない）上位４ビットは，プログラムによってプルアップされているので上位４ビットは常に１となります．つまり，FF のままです.

【演習７】タクトスイッチからの信号をマイコンに入力する基本プログラム２

演習６を体験し，スイッチを押した時，消灯するのは感覚的に逆！と思った諸君も多いと思います．実際，情報処理をする段階においてもこの状態ではとても厄介なのです．
では，なぜこのようになってしまうか考えてみましょう．

ここで改めて図1を見直してください，デジタル信号を普通のスイッチ（A接点）で作り出す回路は，押したら導通状態になりLow（０）になり，離したらHi（１）の信号になってしまう宿命なのです．これを負論理と言ったり，アクティブローといって，論理が逆になっている回路を指します．そのためこれをプログラムで逆にすると，LED点灯の感覚だけでなく，内部に取り込まれている情報も扱いやすい情報になるのです．
これは学生の諸君は今後のためにも，是非身に付けていただきたい大切な事柄なのです．

【タクトスイッチを利用した情報処理プログラム】

上記の理由から,タクトスイッチの情報を扱う場合,大別すると２つの扱い方があります．

- **Bit 単位で扱う場合のプログラミング方法（1つ1つの入力毎に情報を処理する）**
- **Byte 単位で扱う場合のプログラミング方法（複数の入力情報をまとめて情報を処理する）**

課題13　Bit 単位で扱う場合のプログラミング方法の具体例を学習をします．

タクトスイッチS0を押すとLED0～3が点灯し，S1を押すとLDE2～5が点灯し，S2を押すとLED4～7が点灯し，S3でLEDが消灯するプログラムを作成しなさい．

Prg7_bit_in.c（プログラム保存名）

```
#include<3052f.h>
main()
{
    P1.DDR=0xff;
    P2.DDR=0;
    P2.PCR.BYTE=0xff;

    while (1){
        if (P2.DR.BIT.B0==0) P1.DR.BYTE=0x0f;
        if (P2.DR.BIT.B1==0) P1.DR.BYTE=0x3c;
        if (P2.DR.BIT.B2==0) P1.DR.BYTE=0xf0;
        if (P2.DR.BIT.B3==0) P1.DR.BYTE=0x00;
   }
}
```

課題１４　例題のプログラム（Prg7_bit_in.c）の「P2.DR.BIT.B0==0」の意味を調べて報告しなさい．また，P1.DR.BYTE=0x0f についてもその意味を報告しなさい．

課題１５　例題は１６進数を代入して出力しているが，１０進数を代入したプログラムを作成しなさい．作成したプログラムは，prg1_bit_in_15.c で保存し報告する．

2）Byte 単位で扱う場合のプログラミング方法の具体例

Prg7_byte_in.c（プログラム保存名）

```
#include<3052f.h>
main()
{
    int n;

    P1.DDR=0xff;
    P2.DDR=0;
    P2.PCR.BYTE=0xff;

    while (1){
        n=~P2.DR.BYTE;
        n=n & 15;          /*マスキング*/

        if (n==1) P1.DR.BYTE=0x0f;
        if (n==2) P1.DR.BYTE=0x3c;
        if (n==4) P1.DR.BYTE=0xf0;
        if (n==8) P1.DR.BYTE=0x00;
    }
}
```

課題１６　上記のプログラムを実行しなさい．その上で，if 文の中で，n==1 や n==2 等で，なぜ制御ができるのか，理由を報告しなさい．また，このプログラムを修正して，任意のタクトスイッチを使用して（複数利用可），LED の点灯パターンを自由に行うプログラムを製作し報告しなさい．プログラム保存名は，prg7_byte_in_16.c とする．

このプログラムで大切なテクニックは，**マスキング**です．この処理によって，自分の欲しい bit だけを切り出すことができます．逆に考えれば，必要ないビットを消すことができるのです．つまり，必要な bit は入力通りの状態で，不要な bit は常に０にできます．

マスキングを実現しているのが，「 n=n & 15; 」の行です．&は，AND 演算を行っています．たった１行のプログラムですが，ビットの反転同様，入力データを正しく扱う上で，最も基本となる大切な手法なので，もう少し詳しく説明します．こんなときにも活躍します．例えば，ロボットの製作中，センサが入力としてあるポートに接続されているとします．8bit 分全部接続されていれば良いのですが，多くの場合は，一部の bit しか接続されていません．この場合，接続されていない bit は不安定でどんなレベルになるかわかりません．これでは正確な情報処理ができずに誤動作の連続です．これを解決するのがこの手法です．必要な bit だけを入力情報として扱い，それ以外を常に０とします．

次に，byte で処理をするおもしろさとこの処理のメカニズムについて説明します．情報の授業等で AND という論理を以下のように習いますが，これだけではあまり使い物になりません．なぜならマイコン内の情報は byte 単位で情報の伝達（バスといいます）が行われているので，これを生かすには Byte での処理が有効です．

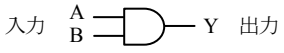

図９　AND ゲートの図記号

AND の真理値表

入力		出力
A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

a = a & 12;　というプログラムの場合のマスキング処理（演算）では，Byte 単位で情報を処理する場合，上記の真理値表の演算が“**縦に行われる**”のです．従って，１２と設定した場合は，３，４bit 以外はすべて０という並びの２進数なので，この０の部分は，どんな入力情報が入力されても０になった演算結果が得られるのです．

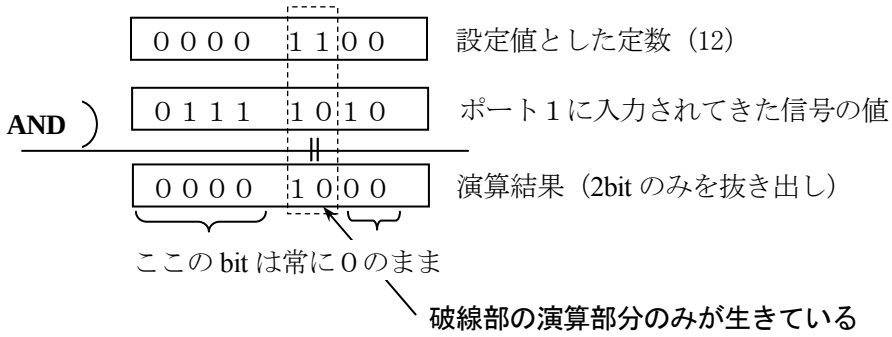

図 10　AND で実現するマスキングの仕組み

この考え方は，AND だけではなく，OR や XOR にも適用できます．OR の場合は変数と変数の合成や結合が実現します．また，XOR は，指定するビットの反転処理（先ほど学んだ, ～は全てのビット反転）やビットの変化を調査する演算が可能になります．論理演算を知るだけで効率的なプログラミングだけでなく，bit 操作の幅が広がります．これについては，あえて報告書の課題にしませんが，意欲のある諸君は，是非，Byte 演算において, OR や XOR の活用したプログラムを試して, その結果を報告してください．

余談ですが，人工知能の一つと分類される，遺伝的アルゴリズムなどもこのように Byte 単位の 2 進数の並びを，部分的に bit 操作することで，新しい世界を繰り広げているのです．物事を色々な側面から見られるようになると，諸君もきっと新しい技術を自ら生み出すようになることでしょう．

■■　第２周目　マイコンで時間を管理する ■■

４　ＩＴＵと時間の制御

H8 マイコンには，ITU(integrated timer counter unit)という，５つのユニットがあります.これらを組み合わせることによって,様々な目的に対応できるようになっています.ここでは，遅延動作を行うプログラミング技術とタイマユニットの活用方法を学習することを学習の目的としています．なお，詳細については，PDF資料のH8/3052F-ZTAT™ ハードウェアマニュアルp271～356 を参照して下さい.

４－１．　ループによる遅延時間

【演習８】　for 文利用で遅延時間を作成

ループによる遅延時間はforやwhile文等の繰返し文で,ある区間を何度も繰り返すと,１サイクルのわずかな処理時間が積み重なって，長い時間を作り出すことができます．また,ループを２段,３段と重ねるとループの積に比例した処理時間を得ることができます.

次のサンプルプログラムは，１つの LED を点滅させるプログラムです.

Prg8_1.c（プログラム保存名）

```
#include<3052f.h>
main()
{
    long int tim;
    P1.DDR=0xff;

    while (1){
        P1.DR.BIT.B3=0;
        for (tim=0 ; tim<2000000;tim++){}
            P1.DR.BIT.B3=1;
        for (tim=0 ; tim<2000000;tim++){}
    }
}
```

このプログラムを実行すると LED の点滅は１分間に約 60 回おこないます.この２つのループの合計は 400 万回ですから，１ループあたりの所要時間は 250ns になります．マイコンのクロックが 25MHz(40ns)ですから１ループに 6 クロック程度で処理していることが分かります．このようなやり方は，マイコンのハードウエア，言語の翻訳方法(コンパイラ)などにより遅延時間が変化します．そこで，上記のようなプログラムを作って，あらかじめ処理時間を計測しておくのも良いでしょう.

課題１７　上記のサンプルプログラムを修正して，上記以外の任意の LED を点滅させるプログラムを作成し報告しなさい．プログラム保存名は，Prg8_17.c とする.

４－２．　繰り返し文

繰り返し制御も非常に重要なプログラミング手法なので，本書においても改めて解説しておきます．これは，ある命令やプログラムを繰り返して処理したい場合，「For 」文を用います．繰り返しの書式は，次のように記述しますが，要するに，式２の部分が成り立っている間（**真**の状態の間）実行文を指定した回数繰り返すと理解してください．

```
for(式１; 式２ ; 式３ ){

        実行文（情報処理）;
}
```

式２に入る評価関数式の例

A ＜ i	A が i より小さい時，真
A ＞ i	A が i より大きい時，真
A >= i	A と i が同じか大きい時，真
A <= i	A と i が同じか小さい時，真
A == i	A と i が同じ時，真

■ 具体的な記述例

```
for( i = 0 ; i < 256 ; i++ ){
    P1.DR.BYTE = i;
}
```

for 文の書式は,
for(i = 初期値 ; 評価関数(繰り返しの最終値); 変数 i の変化分){　処理文 ; } となります．また，**i++** をインクリメントといい，繰り返すごとに i が１ずつ増加し，評価関数が成り立っている間（“真”の間）を繰り返します．例では，ｉが２５６以上になったとき，条件が偽になるので繰返しを終了し，次のプロセスへとジャンプします．
なお，**i- -** と記述した場合はディクリメントといい，繰り返すごとに逆に i を１ずつ減少させることも可能です．上記の例文では「i が 0 から１ずつ増えながら，0 から 255 までのループ，つまり 0 から数えるので 256 回繰り返す」制御文になります．

また，ｉは繰り返している間，その回数を数えるカウンタとして働いているので，ｉは繰り返しの回数（変数）としてプログラム中でも利用されることがしばしばあります．

【演習９】for 文中のカウンタを利用して動きのある LED 制御

Prg9_1.c（で保存します）

```
#include<3052f.h>
main()
{
    int ct;
    long int tim;

    P1.DDR=0xff;

    while (1){
        for (ct=0 ; ct<256;ct++){      /* ct が 0～255 まで繰り返し*/
            P1.DR.BYTE=ct;              /*for 文のカウンタ ct を出力*/
            for (tim=0 ; tim<1000000;tim++){;} /*点灯時間をかぜぐ*/
        }
    }
}
```

このプログラムには，２つの for 文がありますが，それぞれ役割は，全く異なりますので注意して見てください.

・１つめは，（純粋な繰り返し制御）

```
for (ct=0 ; ct<256;ct++){
        :
}
```

の部分は，本来の繰り返しの範囲で，{}で囲われた中のプログラムを，設定された繰り返しの回数繰り返しています（前ページで解説した使い方）.

・2 つ目の for 文　はウエイト，つまり待ち時間をつくるプログラムです.

```
for (tim=0 ; tim<1000000;tim++){;}
```

これはプログラムが処理を行う中で，ある時間，プログラムを止めておくための遅延用のウエイト処理(待ち時間)を行っています．ただし，この待ちのプログラムは完全にプログラムを止めてしまうので，あまり良い方法ではないことを承知して利用してください.

課題１８　これまで学んだ方法を用いながら，一つ LED を点灯させ，順次，一つずつ移動（シフト）させるプログラムを作成し，動作の様子と対象のプログラムを報告書にまとめなさい．プログラム保存名は，prg9_18.c とする．ヒント:LED を横にシフトさせるということはどういうことか考える．または，シフト演算子の利用を考える．（シフト演算子例：　a=a<<1;）

４－３．　ITU の仕組みと遅延プログラミング

ITU を使用する場合，マイコンのベースクロック 20MHz を図 11 に示されているプリスケーラの分周レートを，１，２，４，８分の１から選択します．ここで２分の１の比率を選択するとプリスケーラで分周されたベースクロックは 10MHz となります．

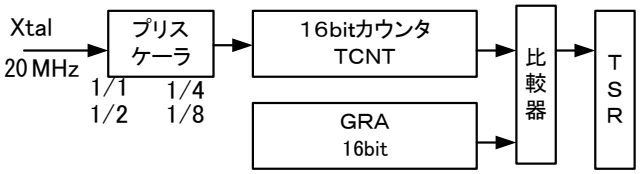

図 11　ITU の基本構成

このクロックを１００００になるまでカウントした時間は１ｍｓとなります．つまり，プリスケーラ直後の分周された周期は，$t = \frac{1}{10\ [MHz]} = 100ns$ となり，これを 10000 カウントすると,100ns×10000=1ms になります．

タイマスタートレジスタ(TSTR) に１をセットすると，タイマカウンタ(TCNT)が動作を開始します．これがジェネラルレジスタ A（GRA）にセットした１００００に一致すると，タイマステータスレジスタ(TSR)の IMFA ビットに１がセットされ，これをチェックすることにより 1ms の遅延時間をプログラムできます．このとき同時に，自動的に TCNT のカウンタはクリアされます．ここで，IMFA をリセットして，次のサイクルに備えます．

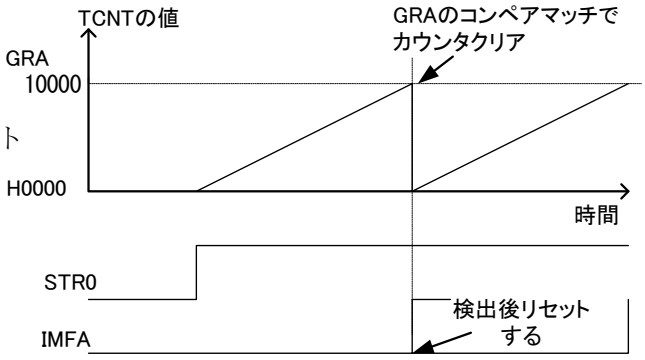

図 12　タイマカウンタのタイムチャート

このプロセスをプログラムで表現するには，次のように記述します．

初期設定

```
ITU0.TCR.BYTE=0x21;    /* GRA compare ,    clock= 1/2 */
ITU0.GRA=10000;        /* GRA 設定 20MHz/2 →1ms */
```

３秒の遅延時間を作るには

```
ITU.TSTR.BIT.STR0=1;                    /*ITU0 TCNT Start */
for (tim=0;tim<3000;tim++){             /*    1000= 1s */
  while (ITU0.TSR.BIT.IMFA==0){}        /* 1ms 毎のフラグ待ち  */
  ITU0.TSR.BIT.IMFA=0; }                /* フラグリセット  */
```

各レジスタのH8/3052F-ZTAT™ ハードウェアマニュアル記載ページを下記に示す.

タイマスタートレジスタ(TSTR)　p282 ジェネラルレジスタ(GRA)　p294 タイマコントロールレジスタ(TCR)　p295 タイマステータスレジスタ(TSR)　p299

なお，フラグ待ちを行う代わりに，IMFA ビットで割り込みがかかるように ITU を設定する事もできますので調べてみるのも良いでしょう.

【演習１０】ITU による時間の管理

ITU タイマを用い 0.5 秒間隔で LED0 の点滅させるプログラムを作成し,実行しなさい.

Prg10_1.c（プログラム保存名）

```
#include<3052f.h>
main()
{
    int tim;

    P1.DDR=0xff;
    /*- time unit set ---*/
    ITU0.TCR.BYTE=0x21;   /* GRA compare , clock= 1/2 分周 */
    ITU0.GRA=12500;       /* GRA 設定 1ms~25MHz/2/12500    */

    P1.DR.BYTE=0;
    while (1) {
        /*- timer program ----------------------*/
        ITU.TSTR.BIT.STR0=1;   /*ITU0 TCNT Start */

        for (tim=0 ; tim<500;tim++){        /* 0.5秒のウエイト  */
            while (ITU0.TSR.BIT.IMFA==0){}  /* IMFA フラグチェック*/
            ITU0.TSR.BIT.IMFA=0;
        }

        P1.DR.BIT.B0 = !P1.DR.BIT.B0;      /* LED0 の点滅の反転 */
    }
}
```

課題１９　上記タイマプログラムを利用し,課題１５を参考にしてＬＥＤが流れるように順次点灯させなさい．なお,完成プログラムは，Prg10_19.c で保存すること.

課題２０　５秒タイマを作成しなさい（タクトスイッチを１個スタートに使用，ＬＥＤで指示）．プログラムは，Prg10_20.c というファイル名で保存すること.

課題２１　100 分の１秒単位のストップウォッチを作成せよ（タクトスイッチを２個使用，printf で表示する）．完成プログラムは，Prg10_21.c で保存すること.

課題２２　演習１０で学習した Prg10_1.c のプログラム下記のプログラムのように, ITU タイマに関する部分を関数化し,その動作を確認しなさい．また，C 言語プログラミングでの“関数”について調べ，知り得た知見を報告しなさい.

Prg10_22.c（プログラム保存名）

```
#include<3052f.h>
init_wait()
{
     ITU0.TCR.BYTE=0x21;   /* GRA compare ,   clock= 1/2 */
     ITU0.GRA=10000;       /* GRA 設定 1ms~25MHz/2/12500 */
}
wait(int t)
{
    int tim;
    ITU.TSTR.BIT.STR0=1;   /*ITU0 TCNT Start */
    for (tim=0 ; tim<t;tim++){
        while (ITU0.TSR.BIT.IMFA==0){}
        ITU0.TSR.BIT.IMFA=0;
    }/* 0.5 秒のウエイト  */
}
main()
{
    P1.DDR=0xff;
    /*- time unit set ---*/
    init_wait();              /*← 関数を呼び出す*/

    P1.DR.BYTE=0;
    while (1) {
        /*- timer-----*/
        wait( 500 );          /* ← 関数を呼び出し，引数を渡す*/
        P1.DR.BIT.B0=! P1.DR.BIT.B0;/* LED0 の点滅の反転 */
    }
}
```

課題２３　課題２１の網掛け部分を，テキストファイルにコピーし，wait.h というファイル名で保存し，このファイルを **h8_toin¥cc の cc フォルダ内に置きなさい**．その上で，上記のプログラムの網掛け部分を消し，#include<wait.h> を 2 行目に書き加えて実行し動作を確認しなさい.

wait.h と保存し h8_toin¥cc へ置く

```
init_wait()
{
     ITU0.TCR.BYTE=0x21;
     ITU0.GRA=10000;       }
wait(int t)
{
    int tim;
    ITU.TSTR.BIT.STR0=1;
    for (tim=0 ; tim<t;tim++){
        while (ITU0.TSR.BIT.IMFA==0){}
        ITU0.TSR.BIT.IMFA=0;
    }
}
```

Prg10_23.c(保存名)

```
#include<3052f.h>
#include<wait.h>          /* ヘッダファイル読込 */
main()
{
    P1.DDR=0xff;
    /*- time unit set ---*/
    init_wait();          /* ← 関数を呼び出す*/

    P1.DR.BYTE=0;
    while (1) {
        /*- timer-----*/
        wait(500);        /* ← 関数を呼び出し*/
        P1.DR.BIT.B0=! P1.DR.BIT.B0;
    }
}
```

課題２１の回答例（これは基礎的な記述でコーディングした例です．プログラムは自由な発想で様々な考え方やテクニックを盛り込めます．みなさんのアイデアでプログラムを考えてください）

```
#include<3052f.h>                                          プリプロセッサの部分
#include <stdio.h>
main()
{
    int tim,msec,sec,min,hour;
                                                           各種初期設定の部分
    P1.DDR=0xff;
    P2.DDR=0;
    P2.PCR.BYTE=0xff;

    /*- time unit set ---*/
    ITU0.TCR.BYTE=0x21;     /* GRA compare , clock= 1/2 */
    ITU0.GRA=10000;         /* GRA 設定 1ms~20MHz/2/10000*/

    P1.DR.BYTE=0;

    while (1) {
        while (P2.DR.BIT.B0==1){}
        hour=0,min=0,sec=0,msec=0;

        while (1) {
            if(P2.DR.BIT.B1==0) break;

            msec=msec+1;        /* 1/100 秒の変数のｶｳﾝﾄ*/
            if( msec >=100 ){
                msec=0;
                sec =sec + 1;   /*秒の変数のカウント*/
            }
            if (sec >= 60) {                                   60 進数（時間）を
                sec=0;                                         カウントしている部分．
                min = min + 1;  /*分の変数のカウント*/          この課題のポイントは
            }                                                  課題16，17に，この
            if (min >= 60) {                                   部分を適切に書き加
                min = 0;                                       えられるか，ということ
                hour = hour + 1;                               です．
            }
            if (hour >= 24){        /*時の変数のカウント*/
                hour=0;
            }

            printf(" %d: %d '%d ¥"%d ¥n",hour,min,sec,msec);

            /*- timer ------------------------------------*/
            ITU.TSTR.BIT.STR0=1;            /*ITU0 TCNT Start */
            for (tim=0 ; tim<10;tim++){     /* 0.01 秒のウエイト  */
            while (ITU0.TSR.BIT.IMFA==0){}
                ITU0.TSR.BIT.IMFA=0;
            }
            P1.DR.BIT.B7=! P1.DR.BIT.B7;    /*  動作モニタ用 LED7 点滅  */
        }
    }
}
```

実験報告書について（第 1 週，第 2 週分）

・実験報告書は第 1 週と 2 週をまとめて，2 週間分で提出してください.

・実験報告書のタイトル（実験テーマ）は,
　　１・２週目：**「マイコン基本プログラミングと ITU の実験」**とします.

・報告書の内容は，次の項目について記載すること.
　１）実験の目的
　２）実験の原理
　３）実験で使用した機器
　４）実験内容
　５）それぞれの課題の回答や結果報告などを記載する(課題1から２３までに対して)
　６）実験結果の考察，または実験全体で得られた知見と考察する.
　　（最後にまとめて記載する）
　７）実験の感想
　８）報告書をまとめるにあたって調べた文献や図書があれば，参考図書として書籍名，出版社，著者名等を全て記載する.

・報告書のコピーなどの不正行為は，写した者は無論の事，写された側も同罪として処断する（試験における不正行為と同等の扱い）.

・**２週目の授業終了後から 1 週間以内（第 3 週の授業の前まで）に S105 実験室前の各担当教員のメールボックスに提出**すること.

注意事項

- 実験の共同実験者の記入は不要とします.
- 実験は，全て個人のマイコンボード（自分のボードのもの）で行うこと.
- 本実験は，マイコンの仕組み，制御方法や制御対象の仕組み等の理解を深めること，プログラムの意味を理解することが主な目的であり，プログラム作成が目的ではありません．その点を留意して報告書を作成して下してください.