AKI- H8（HitachiH8/3048F-ONE）の使い方
マザーボードSWとLEDによるプログラミング

coskx

1.　はじめに

1．1　この文書の構成

この文書はH8/3048F-ONEのCプログラムをコンパイルする方法，AKI-H8 /3048foneにフラッシュメモリ書き込み（転送）する方法を修得した学習者が，AKI-H8/3048F-ONEのマザーボードを用いてLEDをつけたりスイッチを取り込んだりするプログラムを学習するための文書である。

TNCTマザーボード上のH8/3048FONE

テンプレートフォルダなどのダウンロード

DownLoad

1．2 AKI-H8（HitachiH8/3048F-ONE）

AKI-H8/3048F-ONE（図1）は日立製作所の製品であるマイクロコンピュータ H8製品群のH8/3048foneを用いて，秋月電子通商がCPUボード，マザーボードを製作販売している製品の商品名である。
マザーボードはTNCT特製基板である。

マザーボード上には次の要素が搭載されている
（1）　フラッシュメモリ(ROM)書き込み回路（CPUカードの裏側に隠れているので見えない）
（2）　LCD
（3）　8ビットスイッチ
（4）　プッシュスイッチ
（5）　LED（発光ダイオード）

AKI-H8/3048foneでのプログラミングはパソコン上で次のように行なう。
（1）PC上でプログラムを作成する。
（2）PC上でコンパイルして，実行形式プログラムをH8マイコンに転送する。
（3）H8マイコンを起動する。

もう少し丁寧に説明すると次のようになる。
（1）　PC上でC言語またはアッセンブリ言語でプログラムを開発する
（2）　PC上でコンパイル・リンクを行ない実行プログラムファイルを作る
（3）　PC上でロードモジュールに変換する
（4）　RS232C通信でロードモジュールをH8マイコンに転送（フラッシュメモリに書き込む）する。
（5）　H8マイコン上でプログラムを実行する。

図1　AKI-H8　TNCTマザーボード

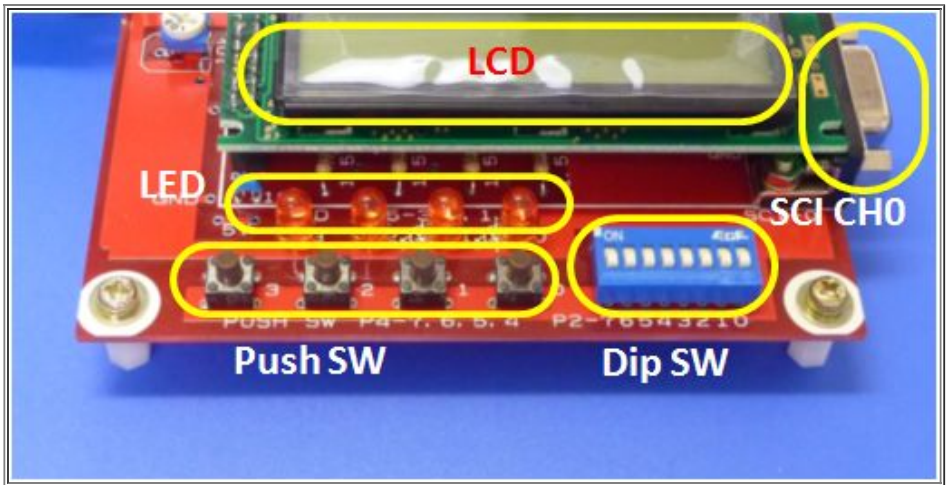
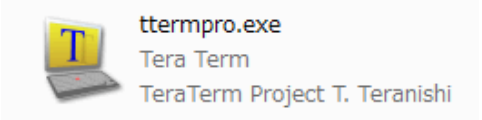


図2　AKI-H8　TNCTマザーボード下部拡大
LEDの実装は2個だけの場合もある

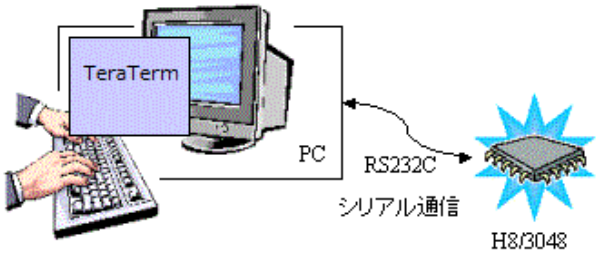
2. PCとマイコンのシリアル通信とビット演算

2．１　マイコンの足し算プログラム

プログラム書き込み用通信回線（シリアル通信ケーブル）はそのままプログラム実行時にも使える。



マイコンにはPCのような大きなディスプレイがついていないため，表示出力は，通信を使ってPCにあるターミナルに対して行う。
シリアル通信ユニットを使うと，マイコンが文字列を送信すると，PC側のターミナル上に表示され，PCのキーボードから与えた
文字列がマイコンに送られる。ターミナルソフトとしてTeraTerm（テラターム）を使用する。
（テラタームでは，通信の設定は，Async，8bit，NoParity，stop1 ，38400baud，(Backspace:Ctrl+H，Space，Ctrl+H)で設定されている。）



テラタームを見ている限りは，PCのみで動作しているように感ずるが，実際の作業はマイコンが行っており，PCはデータの出入り口にすぎない。最初のプログラムは，2つの整数値を与えて，その和を計算させるも

のである。

手順で気を付けるのは，コンパイル＋プログラム転送が終ってからテラタームを立ち上げ，マイコ ンをRUNモードにして電源を入れることである。再び，プログラムを転送する際は，テラタームは通信しないようにしてからプログラムの転送を行 う。（テラタームを起動したままでは，新しいプログラムの書き込みができない）
どうして，排他的に作業をするかというと，通信線は1本しかなく，プログラム転送プログラムとテラタームが1本の通信線を奪い合うからである。

addition.c 加算プログラム

```
/************************************************************
SCI1へ出力，WINDOWSのHyperTerminalなどで受信できる。
ただし，設定は
38400baud，Async，8bit ，NoParity，stop1
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    int x,y,z;
    initSCI1(); /*シリアル通信ユニット　SCI-ch1の初期化(SCI1_printfを使うために必要)*/
    SCI1_printf("Hello. How are you?¥n"); /*printfと同様*/
    SCI1_printf("Let's make an arithmetic addition¥n");
    while (1) {
        x=SCI1_getInt("the first  number = ");
        y=SCI1_getInt("the second number = ");
        z=x+y;
        SCI1_printf("x=%d y=%d z=x+y=%d¥n",x,y,z);
    }
}
```

実行結果（テラタームのターミナル画面）

```
Hello. How are you?
Let's make an arithmetic addition
the first  number = 12
the second number = 56
x=12 y=56 z=x+y=68
the first  number = 100
the second number = -50
x=100 y=-50 z=x+y=50
the first  number =
```

SCI通信の時PC側の「Enterキー」入力では，テンキー部ではなく，文字キーの右側にある大きなenter-keyを使うこと。
テンキーのすべてが使えるわけではないことに注意。

マイコンプログラムの説明

・シリアル通信を行う際は，マイコンプログラムの先頭で，シリアル通信ユニットを初期化しなければならない。
・SCI1_printfはシリアル通信ユニットch1に対するprintfである。PCと通信ができるようになっていて，PC上でターミナルソフトが動いていれば文字列がPCのターミナル上に表示される。
通常のCプログラムで使うprintfと異なる仕様が2つある。
（１）浮動小数点数の表示はできない（ソフトの軽量化のため）
（２）２進法表示ができる（書式で，%dではなく%bにすると2進法表示になる。）
・「while(1) {」というのは無限ループを表わす。
while文やif文でかっこの中の条件を書くところに，数値がある場合0は偽，それ以外は真を表わす。よって，「while(2) {」でも「while(100) {」でも無限ループになる。
・SCI1_getIntはScanfをint型専用にしたもので，シリアル通信から数値を表わす文字列を受け取り，整数型変数に保存する。
その際，引数として与えた文字列をターミナルに送信する。
・詳細はh8_3048fone.hの最初の部分をエディタで開いて調べておくこと。
また，この中にSCI1_printfの関数本体も入っているので，興味があったら見ておくこと。
・#include+ファイル名というのは，#include文の位置に，ファイル名で指定されたファイルの中身が書いてあるものとしてコンパイルせよという意味である。

２．２　ビット演算

マイコン特有のビット演算に慣れておこう。後にビット入出力ではよく使うことになる。

ビット演算は小坂の「Cプログラミング入門」の「B. 演算子」の「B. 2　C言語におけるビット演算子」に解説されている。
リンク
SCI_printfは2進法で表示することができるので助かる。

bitoperation.c bit演算プログラム

```
/************************************************************
ビット演算
SCI1へ出力, WINDOWSのHyperTerminalなどで受信できる。
ただし, 設定は
38400baud, Async, 8bit , NoParity, stop1
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    short int x,y,z;
    initSCI1(); /*シリアル通信ユニット　SCI-ch1の初期化*/
    SCI1_printf("Hello. How are you?¥n"); /*printfと同様*/
    SCI1_printf("Let's make an bit operation¥n");
    while (1) {
        x=SCI1_getInt("the first  number = ");
        y=SCI1_getInt("the second number = ");
        z=x&y;
        SCI1_printf("x=%b y=%b z=x&y=%b¥n",x,y,z);
        z=x|y;
        SCI1_printf("x=%b y=%b z=x|y=%b¥n",x,y,z);
        z=x^y;
        SCI1_printf("x=%b y=%b z=x^y=%b¥n",x,y,z);
        z=x>>3;
        SCI1_printf("x=%b z=x>>3=%b¥n",x,z);
        z=y<<3;
        SCI1_printf("y=%b z=y<<3=%b¥n",y,z);
    }
}
```

実行結果（テラタームのターミナル画面）

```
Hello. How are you?
Let's make an bit operation
the first  number = 12
the second number = 65
x=1100 y=1000001 z=x&y=0
x=1100 y=1000001 z=x|y=1001101
x=1100 y=1000001 z=x^y=1001101
x=1100 z=x>>3=1
y=1000001 z=y<<3=1000001000
the first  number = 54
the second number = 29
x=110110 y=11101 z=x&y=10100
x=110110 y=11101 z=x|y=111111
x=110110 y=11101 z=x^y=101011
x=110110 z=x>>3=110
y=11101 z=y<<3=11101000
the first  number =
```

次のプログラムは, 与えられたshort int型の値を2進法表示した時に, 有効な0の数を数える関数のテストである。ただし, 16ビット変数の上位に埋められている0は数えない。

countzeros.c bit演算プログラム例

```
/************************************************************
short int 型の変数を２進法表現で見たとき, 有効桁数内にある0
の個数を数える関数int countZeros(short int value)のテスト
SCI1へ出力, WINDOWSのHyperTerminalなどで受信できる。
ただし, 設定は
38400baud, Async, 8bit , NoParity, stop1
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

/* short int 型の変数を２進法表現で見たとき, /
```

```
/  有効桁数内にある0の個数を数える関数       */
int countZeros(short int value)
{
    int count=0;
    while (value!=0) {
        count+=1-(value&1);
        value=((unsigned short int)value)>>1;
        /*unsignedにみなさないと，valueが負の数のとき，
        最上位ビットが保たれてしまい，ループから抜け出
        せなくなる*/
    }
    return count;
}

main()
{
    short int x,y;
    initSCI1(); /*シリアル通信ユニット　SCI-ch1の初期化*/
    SCI1_printf("Hello. How are you?¥n"); /*printfと同様*/
    SCI1_printf("Let's count zeros¥n");
    while (1) {
        x=SCI1_getInt("the number = ");
        y=countZeros(x);
        SCI1_printf("x=%b y=%d¥n",x,y);
    }
}
```

実行結果（テラタームのターミナル画面）

```
Hello. How are you?
Let's count zeros
the number = 100
x=1100100 y=4
the number = 1000
x=1111101000 y=4
the number = 10000
x=10011100010000 y=9
the number = -10000
x=1101100011110000 y=8
the number =
```

練習問題　次の作業を行ないなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）「bitoperation.c」で演算結果は２進法表現だけであるが，同じ変数について2進法表現と10進法表現が
同時に見えるように改造しなさい。
負の整数を右シフトした時（例えばxの値を-1024などにしてz=x>>3;），どうなるか試して考察しなさい。
また，unsigned short int u;を導入するように改造し，uに無理やり-1024などを代入してz=u>>3にした時，
どうなるか試して考察しなさい。
(mp1ex01.txt)

（２）short int 型の変数を２進法表現で見たとき，1の個数を数える関数int countOnes(short int value)を作りなさい。
例えばこの関数に10進法表現の10を与えると，2進法表現では1010なので，2を返す。
例えばこの関数に10進法表現の13を与えると，2進法表現では1101なので，3を返す。
テストするmainもつくり，十分にテストしてから提出しなさい。どのように考えて十分なテストだと考えたのか，考察に書きなさい。
(mp1ex02.txt)

３．マイクロコンピュータのプログラムと，デジタルの0V，5Vの世界をつなぐ

マイクロコンピュータプログラムと入出力

PCでのCプログラムではメモリ上での操作が主であり，画面への表示やキーボードからの 入力においてはprintfやscanfなどの関数を呼び出して作業をしていた。原始的なマイクロコンピュータの実用プログラミングでは，関数を用いて IOポート入出力を簡単に記述できるような環境を用いる。しかし原始的なマイクロコンピュータの最初のプログラムでは，LEDを点滅させたり，ON- OFFスイッチの状態を

取り込むのもすべて自分でbit操作プログラムを記述しなければならない。

マイクロコンピュータマザーボードのピン

図２に示すようにマイクロコンピュータ（CPU）はLSIと呼ばれるディジタル回路の部品の１つである，１００本もの脚が付いており，それらの大部分はCPUカードからマザーボード上のピンに接続が伸びている。マザーボー ドとCPUカードをつないでいるのは，複数のコネクタであり，それぞれCN1，CN2，CN3のように名前が付いている。例えば図２ではCN3（コネクタ２）でピン番号が1から40まで付いている。 →参考１（こ のページ下部）
この表では，マザーボード上のCN3の31番ピンは，CPUの52番ピンに繋がっており， その名前はP5-0であることを示している。P5-0というのはポート５の第0ビットという意味である。CPU内部から見るとポート５というのは８ビット あり，P5-0,P5-1,P5-2,・・・,P5-7までである。
（P5はちょっと特別でCPUの外には４ビットしか見えていないのでP5-0～P5-3ま でしかこの表に出てこない）。またポートというのは出入り口のことである。

プログラムから見たポート操作

・ポートとは1バイトのメモリ
ポートはプログラム側から見ると，１つのアドレスをもつ８ビットのように扱える。H8CPUのハードウェアマ ニュアルによれば，P5は「0xfffca番地に置かれた1バイト」として扱える。これまでのCプログラムでは，あるアドレスに直接値を書き込んだり，あるアドレスのデータを読み取ったりすることはなかったが，C言語にはそのような記述ができる。

・あるアドレス（1バイト）への読み書き
例えば0xfffca番地に0x03を書き込むときには，「*(unsigned char *)0xfffca=0x03;」のように書く。これはポインタ表記を用いた記述である。キャストで変数の型を一時的に変更する方法を知っているだろう。例え ば，int型の変数xなどに対して(double)x,(double)100と記述すると，一時的にdouble型のxや100.0になる。だから， (unsigned char *)0xfffcaは，unsigned char を指すポインタとしての0xfffcaを意味し，*(unsigned char *)0xfffcaはそのポインタの指す値ということになるので，0xfffca番地に0x03を書き込むことになる。また例えば，0xfffca番地の値をunsigned char 型変数xxxに受け取るには，「xx=*(unsigned char *)0xfffca;」と書くことになる。

・ポートへの読み書き
それでは，「*(unsigned char *)0xfffca=0x03;」が実行されるとどうなるかというと，0x03は2進法で表すと00000011であり第0ビット（一番右側のビット）は 1，第1ビットも1，第2ビットは0，第3ビットも0なので，それらはP5の各ビットに対応するCN3のピンに出力される。この時0 はOVで，1は5Vで出力される。（P5-bit0:5V，P5-bit1:5V，P5-bit2:0V，P5-bit3:0V）プログラムの実行は一瞬で終わるが，これらの出力はそのまま出力され続け，別の値が出力されるまで変化しない。すなわちラッチ出力という ことになる。
また，「xx=*(unsigned char *)0xfffca;」が実行されると，どうなるかというと，P5の各ビットに対応するCN3のピンへの入力状態が変数xxに取得される。0Vが与えられ たピンは内部では0に，5Vが与えられたピンは内部では1に変換される。例えば，P5-0:5V，P5-1:5V，P5-2:0V，P5-3:0Vが与えられればxxの下位4ビットは0011となる。

PC上のCプログラミングで，「printf(″%02x¥n″，*(unsigned char *)12300);」のような記述は可能であり，コンパイルもできる。しかし，実行時にエラーを発生することがある。これはOSがメモリを管理しており， そのプログラムでアクセスできるメモリのアドレス範囲を制限しているからである。

ポートの初期化

・ポートは初期化設定しないと使えない
H8CPUでは各ポートを初期化することで入力にも出力にも使えるようになっている。
しかし，実際のプログラムでは，P5を入力に使ったり，出力に使ったりというようにはできない。
マザーボー ドを作った段階でP5を入力に使うか出力に使うか決まっている。例えばLEDを接続するように設計すれば出力として使うことになるし，スイッチを繋げば入力として使うことになる。

プログラム起動時に，P5をどちらで使うかをソフトウェア上で設定しなければならない。
その設定はハードウェアマニュアルによれば，アドレス0xfffc8の1バイトを使う。0xfffc8の各ビットごとに0,1を設定 するが，0を設定すると，対応する0xfffcaのビットは入力になり，1を設定すると，対応する0xfffcaのビットは出力とし て使得ることになる。例えば，0xfffc8に0x0a(二進表記で00001010)を書き込むと，0xfffcaの第1，第3 ビットが出力で，残りは入力として使うように設定したことになる。
（なぜ初期化しないと使えないような設計になっているんだろう。CPUは製造段階ではポートを入出力どちらかに決めておかない方が汎用性が高くなる。同じCPUでは多くのニーズに合わせることができるため，このような設計になっている）

H8/3048foneハードウェアマニュアルの「９． ６ ポート5，モード7」参照
http://tnct20.tokyo-ct.ac.jp/~kosaka/for_students/H8/j602093_h83048.pdf

図３のマザーボードではP5の第0ビットと第１ビット（下位２ビットとも言う）にLEDが接続されている。

そうすると次のようなプログラムで，LEDの点滅ができることになる。

primitiveLED.c

```
void waitmsec(int msec)
/*msec間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<4190;j++); /*4190は実測によって求めた値 25MHz駆動*/
    }
}

main()
{
    /* P5の下位2ビットを出力に設定 */
    /* P5の入出力設定部の下位2ビットに1を与えるとこの設定になる*/
    *(unsigned char *)0xfffc8 = 0x3; /*0x3 は 00000011(二進法表記)*/

    while(1) {/*これは無限ループ*/
        /*P5の下位２ビットに01(二進法表記)を出力する*/
        *(unsigned char *)0xfffca = 0x01; /*0x01 は 00000001(二進法表記)*/

        waitmsec(1000);/*1000msecの間なにもしない*/

        /*P5の下位２ビットに10(二進法表記)を出力する*/
        *(unsigned char *)0xfffca = 0x02; /*0x02 は 00000010(二進法表記)*/

        waitmsec(1000);
    }
}
```

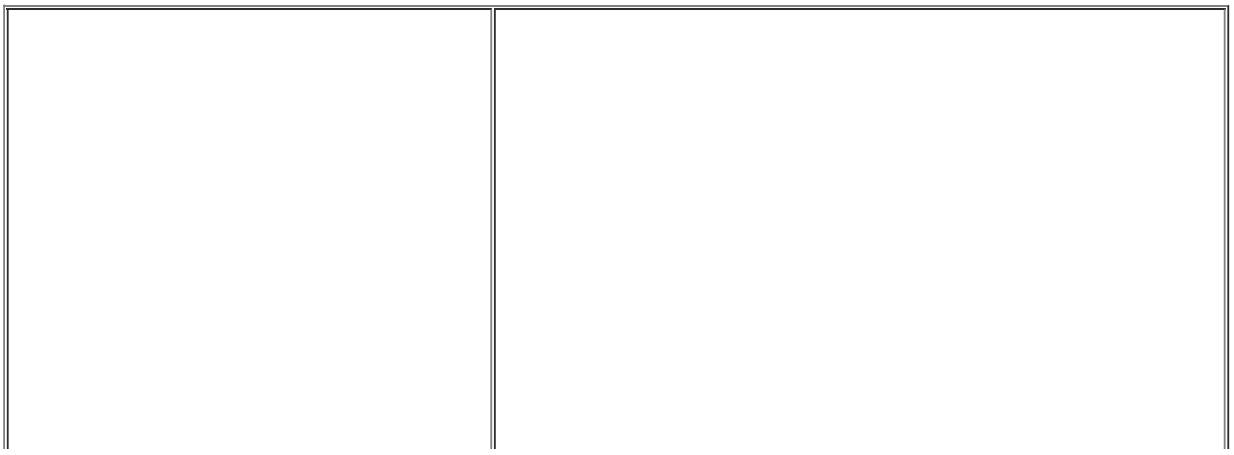
このプログラムは直接変更したいポートのアドレスを使って，値を書き込んで操作している。しかしポート操作に関しては。これから学ぶプロ グラムでは便利な関数群を与えるため，このようなプログラムは書かず，ポートのアドレスのことは意識しないことが多い。しかしこの部分はポート操作の本質であるため，課題の一部や試験では，ここで述べたことが必要になることがある。
ポート5を表わすのに，具体的な番地を記述すると，読みにくいしわかりにくいプログラムになってしまうため，記述方法に工夫をして，次の節で とりあげるような記述をするが，コンパイラにより機械語に翻訳されるとほぼ同じになる。

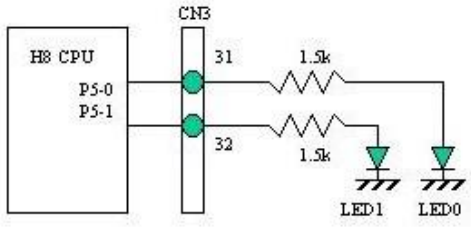
練習問題　次の作業を行ないなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）「primitiveLED.c」を実行して，その動作を記述しなさい。
その際，点滅周期とデューティ比を求めなさい。
また，プログラムのどこでその点滅周期とデューティ比が作られているのか考察しなさい。
ただし，周期とは点灯開始時刻から次の点灯開始時刻までの時間のことであり，
デューティ比とは1周期中の点灯時間の割合のことである。例えば周期8秒の点滅で，
点灯時間が2秒だった場合はデューティ比25%である。
(mp2ex01.txt)

４．LEDの点滅のプログラム

図3のLEDが対象となるLEDである。





（図1，2の写真では，LEDが4つついているが，
右側の2つが，P5-0，P5-1につながっている。
左側の2つはP5-2，P5-3につながっており，
ここでは使わない。
左側の2つは実装されていなくても問題ない。）

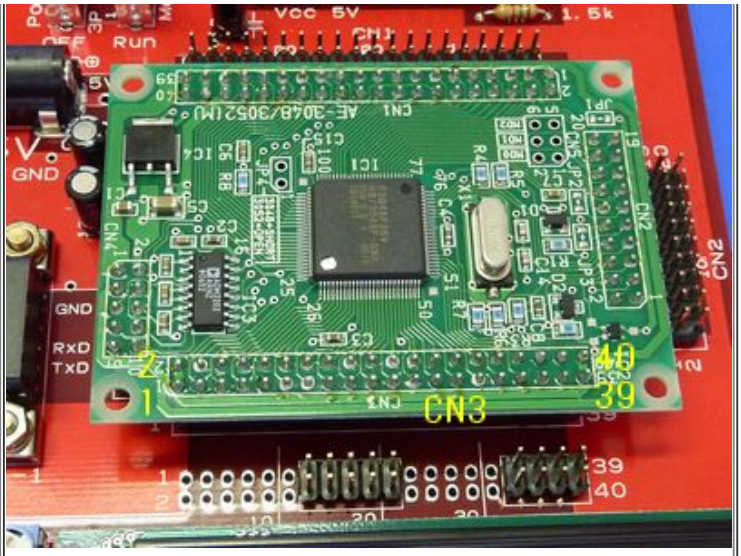


図3　2つのLED	CN3-31,32を探そう CN3の両側には「1,2」と「39,40」が書いてある。 2 4 6 ..... 38 40 1 3 5 ..... 37 39 のように並んでいる。31，32を探しなさい。

4．1　LED点滅プログラム

マザーボード上の2つのLEDを点滅させるプログラムを作る。直接アドレスを指定するのではなく，もうすこし洗練された表現でプログラムす る。
2つのLEDはH8/3048foneCPUのポート5の第0ビット，第1ビットの端子につけられている。
H8/3048foneCPUでポートというのは8ビットの出入り口のことである。ポート5には第0ビットから第7ビットまでの8ビットがあ る。
ポートがCPUのどのピンに接続されているかはCPU設計者によって決められ，CPUのマニュアルに書いてある。
またCPUのピンがCPUカードのどのピンに接続されているかは，CPUカードの設計者によって決められ，CPUカードのマニュアルに書いて ある。
このWebページの「参考１」に接続一覧が書いてある。

通常ポートは入出力兼用なので，この例のようにポート5の第0ビット，第1ビットの端子の先にLEDがついている場合は，CPUに対し，こ の2つのビットは出力に使用することを教える必要がある。（CPU起動後１回だけ設定しなければなりません。）

参考　電流制限

CPUの出力端子はレベルHの時5V，レベルLの時0Vであり，LED単体の電圧降下が1.5V程度であるため，保護抵抗1.5kΩに は3.5Vがかかっていることになる。この抵抗に流れる電流は3.5[V]/1500[Ω] = 0.0023[A] = 2.3[mA]となる。LEDは10mA駆動が標準であるが，1mA駆動でも光らないないわけではない。一方CPUの端子出力電流は最大 2.0mAとされている（H8/3048foneの仕様）ので，AKI-H8はわずかに仕様違反となっている。

4．2　LED点滅プログラムの考え方

（1）マイコン上での駆動の手順

(1)　ポート5の下位2ビットを出力に設定
　　0：入力　　1：出力
　　下位2ビット（第0ビットと第1ビット）を出力に設定するには
　　2進法表現で00000011，10進法表現（16進表現でも同じ）で3を与えればよい。
(2)　ポート5の下位2ビットに0または1を出力してLED点滅を行なう。
　　0：OFF　　1：ON
(3)　時間調整して1秒ごとに2つのLEDが交互に点滅するようにする。

（2）具体的なプログラミングの考え方

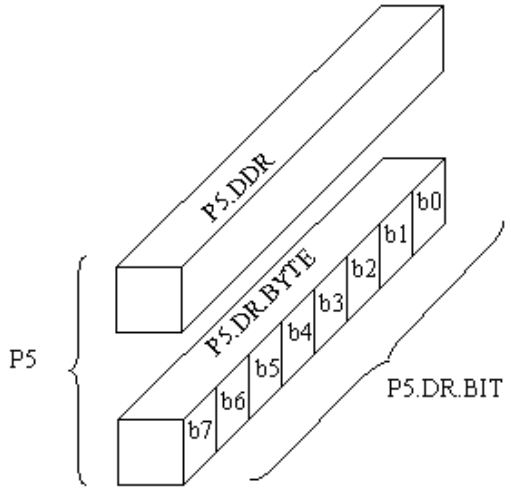
ポート5の下位2ビットを出力に設定
以下を無限ループ
　ポート5の第0ビットに1を出力

ポート5の第1ビットに0を出力
1秒間時間待ち
ポート5の第0ビットに0を出力
ポート5の第1ビットに1を出力
1秒間時間待ち

４．３ LED点滅プログラム

```
/*P5の下位2ビットを出力に設定*/
P5.DDR =  0x3;
while(1) {/*これは無限ループ*/
    /*LED0をONにする*/
    P5.DR.BIT.B0=1;
    /*LED1をOFFにする*/
    P5.DR.BIT.B1=0;
    msecwait(1000);/*1000msecの間なにもしない*/
    /*LED0をOFFにする*/
    P5.DR.BIT.B0=0;
    /*LED1をONにする*/
    P5.DR.BIT.B1=1;
    msecwait(1000);
}
```

参考
（１）ポート５はP5という構造体（構造体学習前なら，家の名前と考えればよい）で表されていると考えられる。
（２）P5の各ビットを入出力のどちらに使うかの設定はP5.DDRという8ビットのレジスタ（P5家のDDR部屋と考えればよ い）に設定する。ここでは下位2ビットのみを出力にするので2進数00000011に設定したいのでP5.DDRには3を設定す る。もし，第2，第3ビットのみを出力にしたいのなら，2進数00001100を設定すればよいので，P5.DDRには16進数 で0xc（10進数では8+4=12）を設定する。またもし第0ビットと第3ビットのみを出力にしたいのなら，2進数 00001001を設定すればよいので，P5.DDRには9を設定する。別のポートも同様な記述を行なうことができる。例えば ポート1の8ビットすべてを出力に設定したいのなら，2進数11111111に設定すればよいのでP1.DDR=0xffのよう に設定する。
（３）ポート5に出力する値はP5.DR（P5家のDR部屋）に出力する。P5.DRはさらに細かい小部屋に分かれてお り，P5.DR.BIT.B0はポート５の第0ビットを表す。同様にP5.DR.BIT.B1はポート５の第1ビットを，P5.DR.BIT.B2はポート５の第2ビットを，....のように使う。ポート５の第2ビットに0を出力したけれ ば，P5.DR.BIT.B2=0，ポート５の第2ビットに1を出力したければ，P5.DR.BIT.B2=1のように使う。こ の表現は1ビットのみを表すため，代入できる値は0または1のみである。またP5.DR.BYTEの表現では8ビットまとめて出 力が可能である。例えばポート５の第2ビットに1を，その他のビットには0を出力したければ，00000100を出力することに なるので，P5.DR.BYTE=4のように表現できる。（ポート1の第0から第3ビットに1を，その他のビットには0を出力し たければ，00001111を出力することになるので，P1.DR.BYTE=0xf）
（４）この構造体の定義はファイル「3048fone.h」にあるが，このファイルはH8CPU用コンパイラフォルダ中にある。
PCのファイル検索で見つけることができる。



P5はDDRとDRの２つの8ビットのレジスタで構成されている
さらにDRはBYTEとBITの２つの方法で書き込みが可能である。

参考図　レジスタP5の構造

構造体に関するさらに詳細な説明

勘どころ
ポート５のbit0とbit1にLEDをつけたのはマザーボードの 設計者が決めたことなので，回路がそのようにできている。 すでに取り付けられているLEDを別のポートを介して点滅させることは，ソフトウェアの変更だけではできない。 別のポートを介してLEDを点滅させるには，そのポートがどの端子とつながっているのか調べて，マザーボードの設計変更が必要になる。

ミニ知識
msec(ミリセカンド)はミリ秒のことである。ミリとは1/1000のことで，1mm(ミリメートル)は1/1000m(メートル)， 1mg(ミリグラム)は1/1000g，1ml(ミリリットル)は1/1000l(リットル)のように使われている。 本来ミリセカンドはmsと表記すべきものであるが，わかりやすさのため，msecと表記している。

４．４ 実際のプログラム

led1st.c

```
/*  waitmsec関数で1秒ごとのLEDのON-OFFを行う  */
#include <3048fone.h>

void waitmsec(int msec)
/*msec間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<4190;j++);     /*4190は実測によって求めた値 25MHz駆動*/
    }
}

main()
{
    /*P5の下位2ビットを出力に設定*/
    /* P5のDDRの下位2ビットに1を与えるとこの設定になる*/
    /*DDRとはDataDirectionRegister(データ方向設定レジスタ)*/
    P5.DDR =  0x3;      /*0x3 = 00000011(二進数)*/
    while(1) {/*これは無限ループ*/
        /*LED0をONにする  P5のDRの第0ビットを1にする*/
        /*DRとはDataRegister(データレジスタ)*/
        P5.DR.BIT.B0=1;
        /*LED1をOFFにする  P5のDRの第1ビットを0にする*/
        P5.DR.BIT.B1=0;
        waitmsec(1000);/*1000msecの間なにもしない*/
        /*LED0をOFFにする  P5のDRの第0ビットを0にする*/
        P5.DR.BIT.B0=0;
        /*LED1をONにする  P5のDRの第1ビットを1にする*/
        P5.DR.BIT.B1=1;
        waitmsec(1000);
    }
}
```

４．５ 実際のプログラム(「h8_3048fone.h」の利用)

実用プログラムでは，ポートの記述はできるだけ隠すようにしている。h8_3048fone.hはマザーボード上のハードウェア操作のプログラミングを関数呼び出しで行なえるようにしたものである。関数呼び 出しを利用すると，どこのポートのどのビットをどうするということを考えずに済み，専用コマンドを使うような感覚でプログラミングでき， 便利である。

もう１つ重要なことがある。将来別のCPUを使うことになって，このプログラムを移植することになった場合を想定しよう。全体の動作を記述するプロ グラムと，CPUのハードウェアを操作するプログラムを分離することにより，移植性が高くなることが想像できる。

関数の定義はh8_3048fone.h中にある。本Webページの末尾に「h8_3048fone.h」があるが，その先頭部分には関数の説明がある。

led2nd.c

```
#include <3048fone.h>
#include "h8_3048fone.h"

void waitmsec(int msec)
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<4190;j++);    /*4190は実測によって求めた値 25MHz駆動*/
    }
}

main()
{
    initLed();  /*LED初期化*/
    while(1) {
        turnOnLed(0); /*LED0のON*/
        turnOffLed(1); /*LED1のOFF*/
        waitmsec(1000);
        turnOffLed(0); /*LED0のOFF*/
        turnOnLed(1); /*LED1のON*/
        waitmsec(1000);
    }
}
```

練習問題　led2nd.cを元にして次のプログラムを作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）周期1秒，デューティ比50%で両方のLEDが同時に点滅するプログラム（mp3ex01.txt）
　ただし，周期とは点灯開始時刻から次の点灯開始時刻までの時間のことであり，
　デューティ比とは1周期中の点灯時間の割合のことである。例えば周期8秒の点滅で，
　点灯時間が2秒だった場合はデューティ比25%である。
（２）周期0.5秒，デューティ比50%で左右のLEDが交互に点滅するプログラム
　（0.5秒ごとにではないことに注意）　（mp3ex02.txt）
（３）0.5秒ごとに点灯状態が変化し，次の点滅パターンを繰り返すプログラム
　左側2回点滅→右側2回点滅→左側1回点滅→右側1回点滅
　(mp3ex03.txt)
（４）10秒周期で両方のLEDが同時に点滅するプログラムをつくり動作させなさい。
　動作中にアナログテスタを使って，CN3の31，32の電圧を測定しなさい。
　測定に当たっては本文書参考１のピン配置よりGNDを基準としなさい。
　（実はシリアルケーブル・コネクタの金属部分もGNDである。ここが触りやすい。）
　（CN3の1,2がGND（グランド））
　(mp3ex04.txt)

電圧計の使い方

練習問題　次の作業を行ないなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）「led2nd.c」で「#include "h8_3048fone.h"」を消去し，h8_3048fone.h中にある関数initLed(),
turnOnLed(),
　turnOffLed()を自分のプログラム中にコピー＆ペーストし，他にも必要なものを記述して動作を確認し，
　提出しなさい。
　#include "ファイル名"とは，#includeが記述された位置に，そのファイルの内容が書かれていると解釈して
　コンパイルしなさいという意味である。
　考察に次のことを記述しなさい。
　「３．　マイクロコンピュータのプログラムと，デジタルの0V，5Vの世界をつなぐ」を読み直して
　h8_3048fone.h中にある関数 initLed()，turnOnLed()，turnOffLed()について説明しなさい。
　なおH8/3048foneハードウェアマニュアルの関連する説明を抜き出しなさい。（９．６ ポート5，モード７参照）
　http://tnct20.tokyo-ct.ac.jp/~kosaka/for_students/H8/j602093_h83048.pdf
　(mp3ex05.txt)
（２）ポート1のbit0,bit1,bit2,bit3に保護抵抗付きのLEDが接続されているとする。
　次のような設計仕様の場合のLED初期化関数initLed_P1()と turnOnLed_P1()，turnOffLed_P1()を作りなさい。
　void initLed_P1(void)　LED初期化関数
　void turnOnLed_P1(int number) LEDを点灯させる関数。
　　ただし，引数は0,1,2,3を取り，LED0,1,2,3をそれぞれ個別に点灯させる。

void turnOffLed_P1(int number) LEDを消灯させる関数。
ただし，引数は0,1,2,3を取り，LED0,1,2,3をそれぞれ個別に消灯させる。
mainプログラムではこれらの関数を用いて，４つのLEDについて2秒ごとに0,1,2,3番LEDが順に点灯し消灯する
ようにプログラム全体を作りなさい。LEDの点滅は見えないけれど，ポート1のbit0,bit1,bit2,bit3がどこのピン
に見えるか調べて，テスタにてチェックしなさい。
（ヒントP1-B0からP1-B3をこのページ後ろの「参考１ H8ピン配置」で探す）
(mp3ex06.txt)

４．６ LEDのPWM駆動

LEDを高速にON-OFFを繰返し，ONになっている時間と周期との比 （デューティ比）を変化させると，人間の目には点滅は見えず，デューティ比に応じて明るさが変化しているように見える。
このような高速ON-OFFスイッチングで出力を制御する方法は「パルス幅変調（ＰＷＭ）」「Pulse Width Modulation」と呼ばれる。
次のプログラムはＬＥＤ０を点灯状態に保ち，ＬＥＤ１をＰＷＭ駆動する。3秒ごとに，デューティ比を９０%，５０%，１０%と変化させている。

ledpwm.c

```
/************************************************************
LEDのPWM(PulseWidthModulation)駆動
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

void waitmsec(int msec)
/*msec間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<4190;j++);    /*4190は実測によって求めた値 25MHz駆動*/
    }
}

main()
{
    int i;
    initLed();  /*LED初期化*/
    turnOnLed(0); /*LED0のON*/
    while(1) {
        for (i=0;i<300;i++) { /*ループ3秒間ループ  デューティ比90%*/
            turnOnLed(1); /*LED1のON*/
            waitmsec(9);
            turnOffLed(1); /*LED1のOFF*/
            waitmsec(1);
        }
        for (i=0;i<300;i++) { /*ループ3秒間ループ  デューティ比50%*/
            turnOnLed(1); /*LED1のON*/
            waitmsec(5);
            turnOffLed(1); /*LED1のOFF*/
            waitmsec(5);
        }
        for (i=0;i<300;i++) { /*ループ3秒間ループ  デューティ比10%*/
            turnOnLed(1); /*LED1のON*/
            waitmsec(1);
            turnOffLed(1); /*LED1のOFF*/
            waitmsec(9);
        }
    }
}
```

次のプログラムは約1秒間隔でLED2が徐々に明るくなる動作を繰り返すプ ログラムである。
ｐは0から999まで変化するが，各ｐの値に対してｉの値が0から999まで変化する。
ｐが10の時は，ｉが0から9の時ＬＥＤ１はＯＮで10から999まではＯＦＦとなる。すなわちｐが10の時ＬＥＤ１がＯＮになっている 時間割合は1%程度である。
ｐが100の時は，ｉが0から99の時ＬＥＤ１はＯＮで100から999まではＯＦＦとなる。すなわちｐが100の時ＬＥＤ１がＯＮに なっている時間割合は10%程度である。

ｐが900の時は，ｉが0から899の時ＬＥＤ１はＯＮで900から999まではＯＦＦとなる。すなわちｐが900の時ＬＥＤ１がＯＮに なっている時間割合は90%程度である。
このように時間経過を考えると，LED1がONになっている時間とOFFになっている時間比が変化している。しかし大変高速にＬＥＤ１が点滅しているため，人間の目にはＬＥＤ１の明るさが変化しているように見える。

ledpwm1.c

```
/************************************************************
LEDのPWM(PulseWidthModulation)駆動
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    int p,i;
    initLed();  /*LED初期化*/
    turnOnLed(0); /*LED0のON*/
    while(1) {
        for (p=0;p<1000;p++) {
            for (i=0;i<1000;i++) {
                if (i<p) turnOnLed(1); /*LED1のON*/
                else turnOffLed(1); /*LED1のOFF*/
            }
        }
    }
}
```

練習問題　次の作業を行ないなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）「ledpwm.c」を参考にして，ＬＥＤ１をＰＷＭ駆動する。
　PWM周期を10msecに「保ったまま，４秒ごとに，デューティ比を２%，４%，８%，１６%，３２%，６４%と
　変化させるプログムを作りなさい。
　ヒント　次の関数の使用を検討する

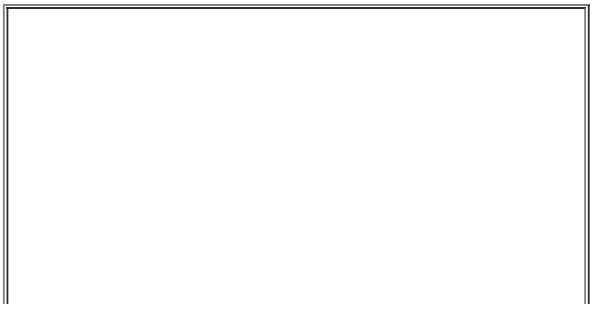
```
 void wait_m4sec(int m4_sec)
 /*10^-4secで指示する間なにもしない時間稼ぎ関数*/
 /*たとえば  wait_m4sec(15);を呼ぶと1.5msec後にこの関数から戻る*/
 {
     int i,j;
     for (i=0;i<m4_sec;i++) {
         for (j=0;j<419;j++); /*419の根拠を考えてみよう*/
     }
 }
```

　(mp3ex07.txt)

（２）「ledpwm.c」では高速点滅を眼で確認できなかったと思う。
　LEDをデューティ比50%で駆動し，1周期をどれくらいにすると点滅を眼で確認できてしまうのか，
　境界のを実験により求めなさい。そのためのプログラムを作って確認しなさい。
　(mp3ex08.txt)

５．　8ビットスイッチ

図3の8ビットスイッチの読み取りを行い，LEDを制御する。
8ビットスイッチはP2の8つのビットすべてにつながっている。



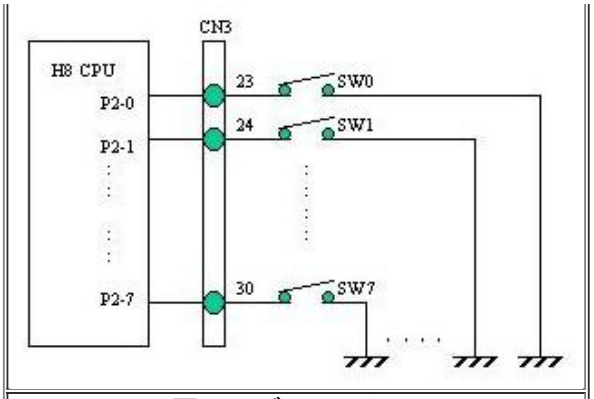


図3　8ビットスイッチ

参考
通常用いられるのは，図3．1の回路である。スイッチがOFFの時CPUの入力ピンには5Vが与えられ，スイッチがONの時CPUの 入力ピンには0Vが与えられる。この用途で用いられている抵抗のことをプルアップ抵抗と呼んでいる。通常この抵抗値は10kΩから 100kΩが用いられる。図3.2左はスイッチがOFFであるため，電流は流れず，抵抗で電圧降下が起こらないため，5Vが出力され ているところを示している。図3.2右はスイッチがONであるため，電流が流れ，抵抗で電圧降下が起こり，0Vが出力されているとこ ろを示している。
H8CPUのポート2，4，5では，スイッチのON-OFF状態の取得等に都合の良い仕掛けがある。図3．3に示すように，図3．1 のプルアップ抵抗をCPUユニットが内蔵しており，プルアップ抵抗を有効にするかどうかをソフトウェアで決めることができるように なっている。このプルアップ抵抗の有効無効を設定するのが，プルアップコントロールレジスタ(PCR)である。
AKI-H8のマザーボードでは，ディップスイッチ(8ビットスイッチ)がポート2の8つのビットに，プッシュスイッチ(4つ)が ポート4の上位4ビットにつながっており，P2.PCRとP4.PCRの対応するビットに1を書き込むことで，プルアップ抵抗を有効 にして使用することができる。そのため，マザーボード上のこれらのスイッチにはプルアップ抵抗がついていない。

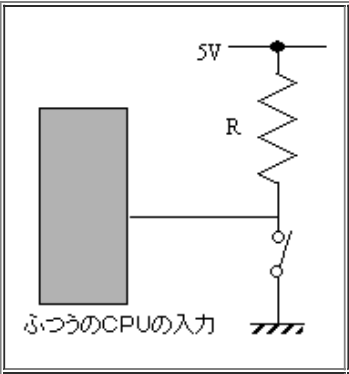


図3．1　スイッチ状態のCPUへの入力

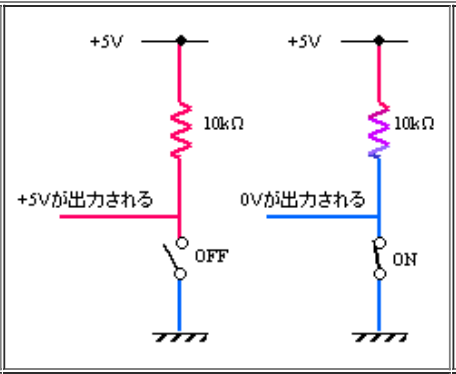


図3．2　プルアップ抵抗の役割

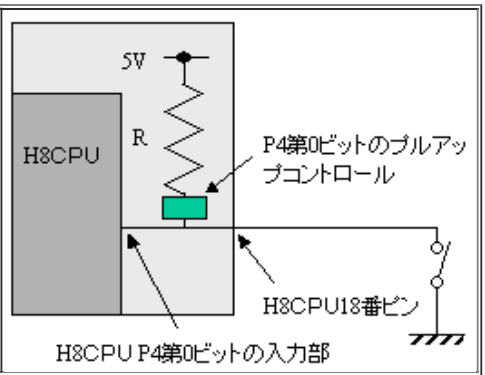


図3．3　H8のプルアップコントロール

外部からのポートのビット入力が5Vの時，CPU内部では1としてとらえ，0Vの時は0としてとらえ る。
その結果，ハードウェアの構成に依存して，スイッチの状態ONを0，状態OFFを1とし て，レジスタに取り込まれることになる。

スイッチの状態	ポートのビット入力端子の電圧	レジスタに取り込まれるビット状態
ON	0V	0
OFF	5V	1

5．1　8ビットSWでLED駆動

８ビットスイッチのON-OFFの状態によってLEDの ON-OFFを制御するプログラムを作成する。
ＬＥＤ駆動部のみh8_3048fone.hを用いてプログラムを作る。
８ビットスイッチの各端子はH8内部でプルアップされる設定なので，スイッチがONになるとポート２の対応するビットは0になる。 OFFになると１になる。
「8bitSWの０（ポート2の第0ビット）がONの時では」というのは 「if (P2.DR.BIT.B0==0) {」のようになり，
「8bitSWの１（ポート2の第0ビット）がOFFの時では」というのは「if (P2.DR.BIT.B0==1) {」のようになり，
通常の感覚と逆なので注意が必要である。

eightsw.c

```
/*************************************************************
8ビットスイッチによってLEDのON-OFFを行う
*************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    initLed();
    P2.DDR = 0x00;/*8bitSWのポートを入力に設定*/
    P2.PCR.BYTE = 0xff;/*8bitSWのプルアップ設定*/
    while(1) {
        if (P2.DR.BIT.B0==0) { /*8bitSWの0がONの時*/
            turnOnLed(0);
            turnOnLed(1);
        } else if (P2.DR.BIT.B1==0) { /*8bitSWの1がONの時*/
            turnOnLed(0);
            turnOffLed(1);
        } else if (P2.DR.BIT.B2==0) { /*8bitSWの2がONの時*/
            turnOffLed(0);
            turnOnLed(1);
        } else {
            turnOffLed(0);
            turnOffLed(1);
        }
    }
}
```

参考　「P2.DDR」は「P2の DDR」，「P2.PCR.BYTE」は「P2のPCRをバイト単位で見た時のバイトデータ」，「P2.DR.BIT.B0」は「P2のDRをビットごと指定した時の第0ビット」と読めばよい。　「P2.DDR」に指定できる値は0から255（0xff），「P2.PCR.BYTE」に入る値は0から 255（0xff），「P2.DR.BIT.B0」に入る値は0または1である。
この例ではでてこないが，P2.PCR.BIT.B0」は「P2のPCRをビットごと指定した時の第0ビット」 と読み，この場合は入る値は0または1となる。

参考　P2をアドレスでみると，P2.DDRは0xfffc1，P2.DRは 0xfffc3，P2.PCRは0xfffd8の1バイトということになる。
よって，P2表記との対応を見ると次のようになる。

意味	P2表記	アドレス直接表記
P2.DRを入力設定	P2.DDR=0;	*(unsigned char *)0xfffc1=0;
P2.DRの全ビットプルアップ設定	P2.RCR=0xff;	*(unsigned char *)0xfffd8=0xff;
P2.DR.BIT.B2のチェック	if (P2.DR.BIT.B2==0) {	if ((*(unsigned char *)0xfffc3 & 4)==0) {

５．２　8ビットSWでLED駆動(「h8_3048fone.h」の利用)

8ビットSWの操作も 「h8_3048fone.h」を利用すると都合がよい。
５．１のようなプログラムでは，スイッチのON-OFFの表現が直観とは異なる（ONが0，OFFが1）ので，関数check8BitSW()を使うのがよ い。

プログラム中でcheck8BitSWはh8_3048fone.h中に
short int check8BitSW(short int number)
で定義されている8bitsw 0,1,2,3,4,5,6,7の状態を調べる関数で，引数numberは0,1,2,3,4,5,6,or 7をとり，
numberで示されたスイッチがONなら1、そうでなかったら0を関数の戻り値として返してくる。
（ポートへの入力を0-1反転して考えている）
この関数のおかげで，「0:OFF，1:ON」という普通の感覚でプログラミングできる。

eightswh.c

```
/*************************************************************
8ビットスイッチによってLEDのON-OFFを行う
*************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
```

```
    initLed();
    init8BitSW();/*8bitSWの初期化*/
    while(1) {
        if (check8BitSW(0)) { /*8bitSWの0がONの時*/
            turnOnLed(0);
            turnOnLed(1);
        } else if (check8BitSW(1)) { /*8bitSWの1がONの時*/
            turnOnLed(0);
            turnOffLed(1);
        } else if (check8BitSW(2)) { /*8bitSWの2がONの時*/
            turnOffLed(0);
            turnOnLed(1);
        } else {
            turnOffLed(0);
            turnOffLed(1);
        }
    }
}
```

C言語では，if文のかっこの中に値があるときは，0なら偽，それ以外の値なら真と判断される。
check8BitSW()は対象のスイッチがONのとき1，OFFのとき0なので，その値で真偽判断すればよい。
当然　if(check8BitSW(0)==1)と　if(check8BitSW(0)!=0)，if(check8BitSW(0))は同じ働きとなる。

5．3　8ビットSWが入力されているポートの各ビットを見てみよう(「h8_3048fone.h」の利用)

直接P2の各ビットを見てみよう。またh8_3048fone.hに定義されているget8BitSW()では，ビッ トパタンでスイッチの様子を
見ることができるが，ONのスイッチは1，OFFのスイッチは0になるように工夫されている。

print8bitswstatus.c

```
/*************************************************************
マザーボードの8ビットスイッチがつながっているP2の各ビットを
直接見てみよう。
また，関数get8BitSW()が得る値を見てみよう。
*************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    unsigned char sw1,sw2;
    initLed();
    initSCI1();
    init8BitSW();/*8bitSWの初期化*/
    while(1) {
        sw1=P2.DR.BYTE;
        sw2=get8BitSW();
        SCI1_printf("P2.DR.BYTE=%08b get8BitSW()=%08b¥n",sw1,sw2);
    }
}
```

練習問題　eightsw.cを元にして次のプログラム を作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しな さい）

（１）8bitSWの1がOFFの時，0.5秒周期で２つのLEDを同時点滅，
　　8bitSWの1がONの時，1秒周期で２つのLEDを同時点滅
　　(mp4ex01.txt)
（２）8bitSWの0がONのとき，0.5秒周期で２つのLEDを同時点滅，
　　8bitSWの0がOFFで，8bitSWの1がONのとき，1秒周期で２つのLEDを同時点滅，
　　8bitSWの0と1がOFFで，8bitSWの2がONのとき，２秒周期で２つのLEDを同時点滅
　　(mp4ex02.txt)

練習問題

（１）8bitスイッチの回路図を見ると，CPUの端子からスイッチを経てGNDに接続されているだけである。
　　通常はこの回路ではスイッチの状態を読み取ることができない。
　　スイッチを読み取るプログラムの初期化部分の意味（8bitSWのプルアップ設定）とあわせてどうして可能なのか検討し，説明しなさい。
　　なおH8/3048foneハードウェアマニュアルの関連する説明を抜き出しなさい。（９．３ ポート2参照）
　　また小坂のweb文書「h8CPU_Input.html」も参考にしなさい。

(mp4ex03.txt)
（２）8bitスイッチで，4つのスイッチをON，残りの４つのスイッチをOFFにしなさい。
5.3のプログラムを動かして，テスタとターミナルの画面を使って，
8ビットスイッチのON-OFFの状態が電圧として見え，マイコン内部ではどのように見えているか
調べなさい。
（ヒントCN3のどこか）
本ページ下部の「参考１ H8ピン配置」を参照しなさい。
どのように検証したか（テスタの赤黒を何処に触れたのかを含む），どのような結果が得られたか
表にして報告しなさい。
(mp4ex04.txt)

8bit SW	ON/OFF	赤ピンで触れたところ	黒ピンでふれたところ	電圧	port番号	bit番号	ポートを直接見ると0，1のどちらが見えるか	関数get8BitSW()では0，1のどちらに見えるか
0		CN3-	CN3-					
1								
2								
3								
4								
5								
6								
7								

（３）次のポートのビットがCN1,CN2,CN3のどこのピンに見えるのかを表にして答えなさい。
後述「参考１ H8ピン配置」を参照しなさい。
１）ポート5のbit0からbit3
２）ポート2のbit0からbit7
３）ポート1のbit0からbit7
４）ポート3のbit0からbit7
５）ポートAのbit0からbit7
６）ポートBのbit0からbit7
(mp4ex05.txt)

６． プッシュスイッチ

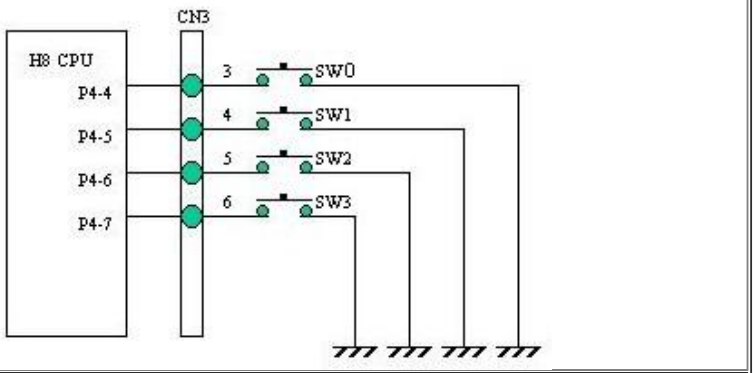


図４ プッシュスイッチ

プッシュスイッチはポート4の第4ビットから第7ビットにつながっている。

PushSWでLED駆動(「h8_3048fone.h」の利用)

図４の プッシュスイッチの読み取りを行い，ＬＥＤを制御する。
プッシュスイッチ0を押すとＬＥＤ0が点灯し，プッシュスイッチ1を押すとＬＥＤ1が点灯するプログラムとす

る。

pushsw.c

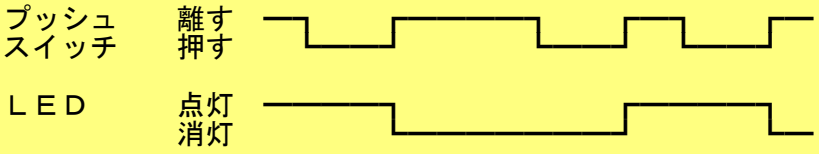
```
/*************************************************************
プッシュスイッチによってLEDのON-OFFを行う
*************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    initLed();
    initPushSW();/*PushSWの初期化*/
    while(1) {
        if (checkPushSW(0)==1) { /*PushSWの0がONの時*/
            turnOnLed(0);
            turnOffLed(1);
        } else if (checkPushSW(1)==1) { /*PushSWの1がONの時*/
            turnOffLed(0);
            turnOnLed(1);
        } else {
            turnOffLed(0);
            turnOffLed(1);
        }
    }
}
```

練習問題　pushsw.cを元にして次のプログラムを作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）５．３のプログラムを参考に，4つのpushSWの状態を取得してターミナル画面に表示するプログラムを作成しなさい。(mp5ex01.txt)

（２）プッシュスイッチの0のみがONの時，0.5秒周期で２つのLED を同時点滅，
プッシュスイッチの1のみがONの時，1秒周期で２つのLEDを交互点滅，
プッシュスイッチの2のみがONの時，２秒周期で２つのLEDを交互点滅
それ以外の状態ではすべて消灯
(mp5ex02.txt)

（３）プッシュスイッチ０を1回押すごとに，LEDの点灯，消灯の切り替えが起こり，何回でも繰り返すことができるプログラム。
正確に言うとプッシュスイッチを押したときには現在の点灯状態はまだ変化せず
プッシュスイッチから指を離した瞬間に点灯状態が反転するように作りなさい。




```
void pushreleaseSW(void)
/*プッシュスイッチが押されるまで待って，*/
/*その後離されるまで待つ関数             */
{
    while (checkPushSW(0)==0);
    while (checkPushSW(0)==1);
}

void pushreleaseSW(void)
/*プッシュスイッチが押されるまで待って，*/
/*その後完全に離されるまで待つ関数       */
{
    int status=1;
    const int Nantichatter=5000;
    int statuscnt=0;
    while (checkPushSW(0)==0);
    while (status==1) {
        if (checkPushSW(0)==1) {
            statuscnt=0;
        } else {
```

```
                statuscnt++;
                if (Nantichatter<statuscnt) {
                    status=0;
                }
            }
        }
    }
```

注意深く100回ほどプッシュスイッチを押して動作を観察しなさい。
点灯消灯の交互リズムが崩れ，不安定な動作が起こるかもしれない。「チャタリング」について調べること。
２つのvoid pushreleaseSW(void)の性能を比較しなさい。
(mp5ex03.txt)

7． インターバルタイマ割り込み

通常の関数は，関数がプログラム中の他の関数から呼び出されたときに作業を行なう。これに対して，割り込み関数は何らかの割り込み要因に よって呼び出される関数である。
インターバルタイマ割り込み関数は，タイマ割り込み初期設定によって設定された時間間隔で起動する割り込み関数である。
「７．１」のプログラムでは，500μｓ（1秒の1/1000の1/1000）のタイマ割り込み初期設定が行なわれ，CPUの割り込み許可がなされ，タイマがスタートした後，プロ グラムの流れは
while(1);
となり，何もしない無限ループに突入する。しかし，500ms（0.5秒）ごとにタイマ割り込み関数「interrupt_cfunc()」が起動し，LEDのON-OFFが継続して行なわれる。変数tickはstatic修飾されているので，関数が呼 び出されたときに，前回呼び出しの時の値が残っている。かつ0が代入されるのははじめの1回だけである。tick=1- tickの演算により，tickの値は0，1を繰り返す。
ロボットの制御には一定時間ごとに起動する定時間割り込み（タイマー割り込み）が良く用いられる。またPWMの生成にもこの定時間割り込み が用いられる場合がある。
関数void msecwait(int msec)を用いた時間管理よりはるかに正確な時間管理ができる。（CPUクロックの精度依存）

7．1 割り込みでLED駆動(インターバルタイマによるタイマ割り込み)

プログラム中main()側で10000μsec（0.01秒）間隔でタイマ割り込みを設定する。
main()関数内でなにもしないループ動作をしている最中に，割り込み関数interrupt_cfunc()は0.01秒ごとに起動 し，
割り込み関数が起動して，100回に99回は何もせず，残りの1回はLEDのON-OFFを行なう。
このタイマ割り込みはITUのch1が使われている。

割り込みを実現するには次のことが必要になる。
（１）割り込みベクトルの設定（割り込みが発生した時に起動すべき関数のアドレスを所定の領域に書いておく）
（２）割り込み要因となる機能の初期化
（３）CPUの割り込み許可
（４）割り込み関数は割り込み関数として定義され，レジスタの退避復帰，割り込みリターンなどの特殊な作り方が必要

これらの必要事項は次のように記述される。
（１）アセンブラでしか書くことが出来ないのでインラインアセンブリで記述
（インラインアセンブリ ： C言語プログラム中にアセンブリ言語を記述すること）

タイマ割り込みの関数のアドレスは0x70に書くことになっている。（割り込みベクタテーブル領域）
アセンブラから見るとCの関数「TimerIntFunc」の名前は「_TimerIntFunc」に見える。
関数の名前は関数の先頭アドレスを意味する。

```
#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm
```

ハードウェアマニュアル
http://tnct20.tokyo-ct.ac.jp/~kosaka/for students/H8/j602093 h83048.pdf
において，割り込みベクタテーブルについての記述は4.1.3に記述されている。
ここで用いているのは，ITUｃｈ1を使用したIMIA1という割り込み要因である。

（２）main関数内で記述 （関数の定義はh8_3048fone.hにある）

```
initTimer1Int(10000); /*時間割り込み10000μsec=10msec ITUch1使用*/
startTimer1();  /*時間割り込みタイマスタートch1*/
```

（３）main関数内で記述

```
E_INT();        /*CPU割り込み許可*/
```

この関数の本体はアセンブリ言語でないと書けないので，　スタートアップルーチンのソースの後ろについている。次の説明を参照のこと
h8startup2.html

（４）割り込み関数独特の宣言を行なう　TimerIntFuncは割り込み関数名

```
#pragma interrupt (TimerIntFunc)
```

この宣言が行われた関数は，プログラム内から関数呼び出 ししてはいけない。

int1st.c

```
/************************************************************
時間割り込みによってLEDのON-OFFを行う
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    initLed();
    initTimer1Int(10000); /*時間割り込み10000μsec=10msec ITUch1使用*/
    E_INT();        /*CPU割り込み許可*/
    startTimer1();  /*時間割り込みタイマスタートch1*/
    while(1);       /*なにもしないループ*/
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*タイマ割り込みルーチン*/
{
    static int tick=0;
    static int count=0;
    clearTimer1Flag();  /*タイマステータスフラグのクリア 忘れないこと*/
    count++;
    if (count==100) {
        count=0;
        if (tick==1) {
            turnOnLed(0);
            turnOffLed(1);
        } else {
            turnOffLed(0);
            turnOnLed(1);
        }
        tick=1-tick;
    }
}
```

```
ここではアセンブリ言語には踏み込まないが簡単に解説
（深入りはしない。後にアセンブリ言語を学んだ後に読み返せばよい）

#pragma asm
    ここの間はアセンブリ言語による記述ですという意味
#pragma endasm

.SECTION    MYVEC, DATA, LOCATE=H'000070
      MYVECという名前のセクションを0ｘ70番地から始める

.ORG        H'000070  ;IMIA1
      0x70からこれ以降の内容を配置

.DATA.L     _TimerIntFunc
```

```
        関数TimerIntFuncの先頭アドレスをここに書く
        DATA.L  というのは，32ビット幅のデータの意味

.SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
        これ以降はCODEというセクションの続きになるの意味

アセンブリ言語の記述スタイルについて
プログラムの内容は行の先頭にいくつかのスペースまたはタブの後ろに書く。
「;」以降はコメントとみなされる。
```

勘どころ
ITU1の割り込みベクタ(割り込み関数の先頭番地)が0x70であるこ とは，CPUの設計者が決めたことなので，プログラム作成者が変更することはできない。 割り込み関数の名前は，プログラム作成者が決めることができる。

練習問題　int1st.cを元にして次のプログラムを作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）int1st.cは2秒周期のLED点滅であった。
　　int1st.cを元にして割り込み周期を変更せずに，１秒周期のLED点滅を行なうプログラムに変更しなさい。
　　initTimer1Int(10000);は変更しない。
　ヒント：間引き（割り込み関数が起動しても何回かに一回しかLEDを操作しない）間隔を変更する。
　　initTimer1Int()関数の定義はh8_3048fone.hにある。
　　initTimer1Int()，startTimer1()はμsec単位の割り込み間隔動作，最長でも約29msecのタイマ割り込みしかできない。
　(mp6ex01.txt)
（２）int1st.cにおいてinitTimer1Int(1000);に変更して１秒周期のLED点滅を行なうプログラムを作りなさい。
　（引数がマイクロ秒であること，引数にはshort int型が用いられていることに注意）
　ヒント：間引き（割り込み関数が起動しても何回かに一回しかLEDを操作しないようにすればよい）
　　initTimer1Int()関数の定義はh8_3048fone.hにある。
　　initTimer1Int()，startTimer1()はμsec単位の割り込み間隔動作，最長でも約29msecのタイマ割り込みしかできない。
　(mp6ex02.txt)

７．２　割り込みでLED駆動

割り込みを用いたPWM駆動でLEDの制御を行なう。
このプログラムではITUのch1のみを用いる。
PushSW 0 → LED 0 明るく点灯（10/10）
PushSW 1 → LED 0 暗く点灯（1/10）
PushSW 2 → LED 1 明るく点灯（10/10）
PushSW 3 → LED 1 暗く点灯（1/10）

volatile修飾子は，割り込み関数と通常の関数とで同じグローバル 変数を用いるときに使う。
コンパイラは通常コードの最適化を行なう。例えば，変数は通常はメモリ上にある。ある変数を一度レジスタに読み込んで，その変数の値を変化させていない場 合は，次にその変数の値が必要になったときには，メモリをもう一度読みに行くようなことはせず，レジスタの値を用いる。しかし，その間に割り込み 関数がその変数の値を変化させることもある。そこでvolatile修飾子をつけた変数にしておくと必ずメモリにある変数を使うコードになる。この例の場 合は，特に必要ないが，おまじないとしてつけておくようにする。

このプログラムではLEDの暗い点灯時では1/10の時間割合で点灯している。LEDの暗い点灯時に5 /10の時間割合で点灯するように変更しなさい。この点灯時間の割合は「デューティ比」と呼ばれる。

「h8_3048fone.h」にある関数initTimer1Int()，startTimer1()はITU1 を使ったタイマ割り込みを可能にしている。
ハードウェアマニュアルのITUの基本動作（TCNT（カウンタ）がカウントアップ）を見なさい。
カウンタ（TCNT）がカウントアップしてあらかじめ設定した値（GRA）とコンペアマッチ（比較して一致）で割り 込みが起こり，カウンタがクリアされ，この動作が継続される。カウンタは内部クロック（25MHｚ）の1/8で動作 している。
このことにより，タイマ割り込みが行われている。

「１」で取り上げた関数msecwait()を使 う方法と割り込みを使う方法とでは，時間精度が違う。高精度な時間管理には関数msecwait()を使うべきでは ない。タイマユニットが使えない場合に使用を検討するのがよい。

int2nd.c

```
/************************************************************
```

```
プッシュスイッチと時間割り込みによってLEDのPWM制御を行う
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

volatile int led0,led1;
const int period=10; /*周期10msec*/
const int low=1;

main()
{
    initLed();
    initPushSW();
    led0=led1=0;
    initTimer1Int(1000); /*タイマ割り込み1msec */
                         /*単位はμsec ITUch1のみ使用*/
    E_INT();         /*CPU割り込み許可*/
    startTimer1();  /*時間割り込みタイマスタート*/
    while(1){
        if (checkPushSW(0)==1) { /*PushSWの0がONの時*/
            led0=period;
        } else if (checkPushSW(1)==1) { /*PushSWの1がONの時*/
            led0=low;
        } else {
            led0=0;
        }
        if (checkPushSW(2)==1) { /*PushSWの2がONの時*/
            led1=period;
        } else if (checkPushSW(3)==1) { /*PushSWの3がONの時*/
            led1=low;
        } else {
            led1=0;
        }
    }
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*タイマ割り込みルーチン*/
{
    static int tick=0;
    clearTimer1Flag();  /*タイマステータスフラグのクリア 忘れないこと*/
    if (tick<led0) {
        turnOnLed(0);
    } else {
        turnOffLed(0);
    }
    if (tick<led1) {
        turnOnLed(1);
    } else {
        turnOffLed(1);
    }
    tick++;
    if (tick==period) tick=0;
}
```

練習問題　int2nd.cを元にして次のプログラムを作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

PushSWはどれも押されていない　→　2つのLEDは暗く点灯（1/10）
PushSW 0 を押す → LED 0 中間点灯（2/10）　　LED 1消灯
PushSW 1 を押す → LED 0 中間点灯（5/10）　　LED 1消灯
PushSW 0とPushSW 1 を押す → LED 0 明るく点灯（10/10）　　LED 1消灯
PushSW 2 を押す → LED 1 中間点灯（2/10）　　LED 0消灯

PushSW 3 を押す → LED 1 中間点灯（5/10）　　LED 0消灯
PushSW 2とPushSW 3 を押す → LED 1 明るく点灯（10/10）　　LED 0消灯
それ以外の状態ではすべて消灯
点灯周期は10msとする。

unsigned char getPushSW(void)　を使うと便利
　押しボタンスイッチの取得　ビット反転し，ONは1，OFFは0で取得される。
　押しボタンスイッチの状況は第0-第3ビットに現れる。
(mp6ex03.txt)

8.　パソコンとのシリアル通信

シリアル通信ユニットSCIを使って，マイコンの状態を表示するプログラムで，SCI関連の関数を使ってみよう。

sciout.c

```
/************************************************************
SCI1へ出力，WINDOWSのHyperTerminalなどで受信できる。
ただし，設定は　38400baud，Async，8bit ，NoParity，stop1
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

void func1(void)
{
    unsigned char sw,previous;
    SCI1_printf("Printing 8-bitSW status...¥n");
    SCI1_printf("Change 8-bitSW and new status will appear.¥n");
    SCI1_printf("If any key on the keyboard, this test will quit.¥n");
    previous=sw=get8BitSW();
    SCI1_printf("8-bitSW status= %2x[%08b]¥n",sw,sw);
    do {
        sw=get8BitSW();
        if (sw!=previous) {
            SCI1_printf("8-bitSW status= %2x[%08b]¥n",sw,sw);
            previous=sw;
        }
    } while (SCI1_chkgetChar()<0);
}

void func2(void)
{
    unsigned char sw,previous;
    SCI1_printf("Printing PushSW status...¥n");
    SCI1_printf("Change PushSW and new status will appear.¥n");
    SCI1_printf("If any key on the keyboard, this test will quit.¥n");
    previous=sw=getPushSW();
    SCI1_printf("PushSW status= %2x[%08b]¥n",sw,sw);
    do {
        sw=getPushSW();
        if (sw!=previous) {
            SCI1_printf("PushSW status= %2x[%08b]¥n",sw,sw);
            previous=sw;
        }
    } while (SCI1_chkgetChar()<0);
}

void func3(void)
{
    short int x;
    long int y;
    x=SCI1_getInt("Key in a decimal number >>>");
    SCI1_printf("The number you keyed in is %d %x¥n",x,x);
    x=SCI1_getInt("Key in a hexdecimal number (ex. 0x23ff) >>>");
    SCI1_printf("The number you keyed in is %d %x¥n",x,x);
    y=SCI1_getInt4("Key in a decimal big number (ex. 12345678)>>>");
    SCI1_printf("The number you keyed in is %ld %lx¥n",y,y);
    y=SCI1_getInt4("Key in a hexdecimal big number (ex. 0x23ff0000) >>>");
    SCI1_printf("The number you keyed in is %ld %lx¥n",y,y);
}
```

```
main()
{
    int menu;
    initSCI1(); /*SCI-ch1の初期化*/
    initPushSW(); /*押しボタンスイッチの初期化*/
    init8BitSW(); /*8ビットスイッチの初期化*/
    SCI1_printf("Hello. How are you?¥n");
    while (1) {
        SCI1_printf("***************menu**********¥n");
        SCI1_printf("1:   get 8-bit SW and print ¥n");
        SCI1_printf("2:   get Push SW and print ¥n");
        SCI1_printf("3:   get integer from SCI1 and print ¥n");
        do {
            menu=SCI1_getChar(); /*menuには'1','2','3'が入るはず*/
        } while (menu<'1'||'3'<menu);
        SCI1_printf("¥n");
        switch (menu) {
        case '1':
            func1();
            break;
        case '2':
            func2();
            break;
        case '3':
            func3();
            break;
        default:
            break;
        }
    }
}
```

特別な関数の説明 これらの関数はh8_3048fone.h中に定義されている。

short int SCI1_getChar()
SCI-ch1から1byte入力コードを得て関数の値として返す関数。通信エラーがあると-2が戻 る。
SIC-ch1入力バッファを検査し，データがあれば持ち帰るが，データがない場合はデータが来るまで待ち続け，
データが来たら，それを持ち帰ってくる
PCのキーボードが押され，SCI-ch1経由でデータが転送されてくる まで，永久に待つ関数である。
ANSIの関数getchar()と同じ動作をする

short int SCI1_chkgetChar()
SCI-ch1入力バッファを検査し，受信データがあれば1byte入力コードを得て関数の値として返す関数。
受信データガなければ-1が，通信エラーがあると-2が戻る。
SICバッファを検査し，データがあれば持ち帰るが，データがなくても -1をもってすぐに帰ってくる関数である。

int get8BitSW()
8bitスイッチの状態をそのまま1バイトの値として読み込む関数。状態（00～ff）を関数の値として返す。
ただし，値は反転しており，スイッチの状態がONのビットは1，OFFのビットは0で読み込む。

int getPushSW()
プッシュスイッチの状態をそのまま1バイトの値として読み込む関数。4つのプッシュスイッチは1バイト中で下位4ビットとして取り込まれ，状態（00～0f）を関数の値として返す。
ただし，値は反転しており，スイッチの状態がONのビットは1，OFFのビットは0で読み込む。

short int SCI1_getInt(char prompt[])
SCI-ch1からプロンプト付で，short intの値を受け取り，関数の値として返す。
（引数で与えた文字列を表示してから，short intの値を受け取る。）
正負の10進数または16進数を受け付ける。「0x」で始めまる文字列は16進数として受け取る。

long int SCI1_getInt4(char prompt[])
SCI-ch1からプロンプト付で，long intの値を受け取り，関数の値として返す。
（引数で与えた文字列を表示してから，long intの値を受け取る。）
正負の10進数または16進数を受け付ける。「0x」で始めまる文字列は16進数として受け取る。

●TeraTermの画面への表示をそのままファイル化する方法(テキストキャプチャ)
TeraTermのファイルメニューから「ログ」コマンドで，ファイル名を指定すると，その後では
TeraTermの画面への表示がそのままファイルに書きだされる。レポート作成に便利！

SCI通信の時PC側の「Enterキー」入力では，テンキー部ではなく，文字キーの右側 にある大きなenter-keyを使うこと。

```
enter-keyについて
キーボードには２つのenter-keyがあるが，実は文字コードが異なる。
      文字キーの右側にある大きなenter-key       ¥r¥n  (0x0d,0x0a)
      数字キーの右側にある小さなenter-key       ¥r  (0x0d)
である。
¥r(0x0d) はコンソール上で文字ポインタを左端に戻すコード
¥n(0x0a) はコンソール上で文字ポインタを次の行に進めるコード
```

練習課題
（実行の 様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）sciout.cを実行してキーボードから1を入力したところ次のような画面になった。
止まっているように見えるが，マイコンの内部ではプログラムが動作している。
どうして止まっているようにみえるのか説明しなさい。
(mp7ex01.txt)

```
Hello. How are you?
**********menu**********
1:   get 8-bit SW and print
2:   get Push SW and print
3:   get integer from SCI1 and print
1
Printing 8-bitSW status...
Change 8-bitSW and new status will appear.
If any key on the keyboard, this test will quit.
8-bitSW status=  0[00000000]
```

練習問題 次のプログラムを作りなさい。
（実行の 様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の中ファイル名で提出しなさい）

（１）マイコンのプログラム実行開始後，４つのプッシュスイッチがそれぞれ 何回押されたかを表示するプログラムを作成しなさい。
このプログラムは電源がOFFされるまでカウントを続け，絶えず最新のカウントを表示し続けるものとする。
通信して文字を表示するのはマイコンにとっては遅い作業である。（1msec間に3文字しか送れない）
そこで，SCI1_printはmainがわで行い，スイッチの状態観測は1/1000秒間隔の割り込み関数内で行うものとする。カウント値はグローバル変数に格納すると良い。
(mp7ex02.txt)

参考 SW0のみを数えるプログラム(チャタリング対策有り) 要点のみ 表示 隠す

（２）アドレス0x00000から0x0001FFまでのメモリを十六進ダンプするプログラムを作りなさい。
ヒント long int型変数ptrに0x1200が入っている時，アドレス0x1200の内容を表示するには次のように書くと良い。
SCI1_printf(" %02x",*(unsigned char *)ptr);
また，メモリ空間は0から0xfffffまであるのでアドレスは16進表現で5ケタ必要となる。
アドレスを表わす変数はunsigned long int型を使う。 表示のときの書式は%ld(%lx)になることに注意。
以下の実行例と同じ実行結果が得られたら正解。
提出ファイルには実行結果を貼り付けること。 メモリについての補足
(mp7ex03.txt)

実行例

```
memory dump 0x00000-0x001ff
       +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00000 00 00 01 00 ff ff ff ff ff ff ff ff ff ff ff ff
00010 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

```
00020 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00030 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00040 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00050 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00060 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00070 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00080 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00090 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000a0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000d0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000f0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00100 7a 07 00 0f ff 10 01 00 6b 20 00 00 0e 20 01 00
00110 6b 21 00 00 0e 24 01 00 6b 22 00 00 0e 28 01 f0
00120 64 22 5a 00 01 2e 6c 03 68 93 0b 71 1b 72 46 f6
00130 01 00 6b 21 00 00 0e 2c 01 00 6b 22 00 00 0e 30
00140 f3 00 01 f0 64 22 5a 00 01 50 68 93 0b 71 1b 72
00150 46 f8 5e 00 0c f4 40 fe 06 7f 54 70 04 80 54 70
00160 18 88 38 ba 38 b8 f8 13 38 b9 19 00 0b 50 79 20
00170 03 e8 4d f8 f8 30 38 ba 54 70 29 bc 17 51 0d 10
00180 79 60 00 38 47 16 7f bc 72 60 7f bc 72 50 7f bc
00190 72 40 7f bc 72 30 79 00 ff fe 54 70 73 69 47 da
001a0 7f bc 72 60 29 bd 17 51 0d 10 54 70 29 bc 17 51
001b0 0d 10 79 60 00 38 47 16 7f bc 72 60 7f bc 72 50
001c0 7f bc 72 40 7f bc 72 30 79 00 ff fe 54 70 73 69
001d0 47 0c 29 bd 17 51 7f bc 72 60 0d 10 54 70 79 00
001e0 ff ff 54 70 5e 00 0d e2 0f 86 0d 1d 19 55 1b 5d
001f0 0d d4 40 26 55 84 0d 0d 68 e8 a8 0d 47 20 0d dd
```

（３）指定したアドレスから 0x100byteをメモリダンプするプログラムを作りなさい。
ただし指定アドレスは十六進の１の位は0とする。
アドレスを表わす変数はunsigned long int型にし，キーボードからの先頭アドレスの入力は
long int SCI1_getInt4(char prompt[])
を使う。
表示は，何回でも続けるものとする。
提出ファイルには実行結果を貼り付けること。
(mp7ex04.txt)

実行例

```
start address (0xnnnnnn) =0x400
memory dump 0x00400-0x004ff
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00400 00 10 0f 86 5e 00 02 46 79 01 00 10 0f f0 5e 00
00410 01 e4 0f f0 5e 00 03 28 7a 17 00 00 00 10 01 00
00420 6d 76 54 70 01 00 6d f6 7a 37 00 00 00 10 0f 86
00430 55 38 79 01 00 10 0f f0 5e 00 02 e2 0f f0 5e 00
00440 03 28 7a 17 00 00 00 10 01 00 6d 76 54 70 6d f6
00450 0c 86 a6 0a 46 04 f8 0d 55 f4 28 b4 73 78 47 fa
00460 36 b3 7f b4 72 70 6d 76 54 70 01 00 6d f6 0f 86
00470 40 06 68 68 55 d8 0b 76 68 68 46 f6 01 00 6d 76
00480 54 70 19 11 40 0a 79 08 10 68 1b 58 46 fc 0b 51
00490 1d 01 45 f2 54 70 19 11 40 0a 79 08 00 04 1b 58
004a0 46 fc 0b 51 1d 01 45 f2 54 70 0c 80 28 c6 e8 f0
004b0 14 08 38 c6 7f c6 70 50 79 00 00 05 55 d8 7f c6
004c0 72 50 79 00 00 28 40 ce 6d f6 0c 8e 11 88 11 88
004d0 11 88 11 88 e8 0f 55 d2 ee 0f 0c e8 55 cc 6d 76
004e0 54 70 6d f6 0c 8e 7f c6 70 40 11 88 11 88 11 88
004f0 11 88 e8 0f 55 b4 ee 0f 0c e8 55 ae 7f c6 72 40
start address (0xnnnnnn) =0xffe00
memory dump 0xffe00-0xffeff
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
ffe00 df ff ff ff fb ff bb ff ff bf ff ef ff ff ff ff
ffe10 ff ff cf ef ff f7 fd ff ff f8 fd ff ff d7 be fb
ffe20 fb bb ff ff ff ff ff ff dd 7f ff ff ff ff ff ff
ffe30 fd bb ff bf fb f7 ff ff fd fd df fb ff ff 7f ff
ffe40 ff ff ff fe ff ff ff 7f cf ff ff ff ff e7 ff ff
ffe50 ff ff ff ff df ff df ef ff ae ff fb ff fb f7 fa
ffe60 ff ff ff fa ff fd df ff ff ff ff ff ff fd ff ff
ffe70 ff fd ff ff fe fb ff 27 ff ff ff f8 7b 7f ff ff
ffe80 ff ff ff ff ff ef fd 7e ff ff fe bc 0a 0c fe c0
ffe90 00 c2 00 00 02 52 00 0f ff 0c 00 00 0a b4 00 0f
ffea0 fe c1 ff df ff fd ff ff fe fd db df ff ff f7 ff
ffeb0 ff fb fb ff 5e ff ff ff 66 66 65 66 66 66 65 62
ffec0 30 33 33 00 00 0f ff 0a 00 0f fe c3 00 00 0f 0d
ffed0 00 00 00 00 02 46 00 2b 00 10 00 00 02 2a 00 00
ffee0 00 ee 00 00 00 00 00 00 0f 02 00 0f ff 00 00 0f
```

```
ffef0 00 00 00 00 00 04 00 00 0b 6a 00 0f ff 0a 00 0f
start address (0xnnnnnn) =0xfff00
memory dump 0xfff00-0xfffff
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
fff00 00 00 00 00 0d 7a 00 00 0f 08 00 00 00 00 01 56
fff10 ff ff ff ff ff ff ff ff ff ff ff ff 7f 00 00 00
fff20 ff ff ff ff 21 ff ff 00 ff ff ff ff 77 ff ff 00
fff30 ff ff ff ff ef ff ff 00 ff ff ff ff fb ff ff 00
fff40 00 00 00 00 00 00 ff f0 00 ff ff 00 00 00 00 00
fff50 ff 00 ff ff ff ff ff ff ff ff ff ff fe fc 40 0f
fff60 e0 e0 80 c0 80 88 f8 f8 00 00 ff ff ff ff 80 88
fff70 f8 f8 00 00 ff ff ff ff 80 88 f8 f8 00 00 ff ff
fff80 ff ff 80 88 f8 f8 00 00 ff ff ff ff ff ff ff ff
fff90 ff ff 80 88 f8 f8 00 00 ff ff ff ff ff ff ff ff
fffa0 f0 ff 00 00 00 00 ff ff 18 00 ff 3f 02 07 00 ff
fffb0 00 ff 00 ff 84 00 f2 fb 00 13 30 30 40 0a f2 fc
fffc0 ff ff ff ff ff ff ff ff ff ff f0 fb ff ff ff ff
fffd0 ff ff ff ff ff ff ff ff 00 ff 00 f0 00 00 1f 1f
fffe0 00 00 00 00 00 00 00 00 00 7e ff ff ff ff f3 ff
ffff0 ff c7 0b fe 00 00 0f ff 00 00 ff ff ff ff ff ff
start address (0xnnnnnn) =
```

9. 「パソコンとのシリアル通信」,「タイマー割り込み」を用いた時計

「パソコンとのシリアル通信」,「タイマー割り込み」を用いた時計を作る。

timer.c

```
/************************************************************
割り込みを用いた時計　起動時からの経過時間［秒］を
SCI1へ出力, WINDOWSのHyperTerminalなどで受信できる。
ただし, 設定は
38400baud, Async, 8bit , NoParity, stop1
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

volatile unsigned int counter;/*0.1s単位でカウント*/

main()
{
    unsigned int counter1;
    unsigned int t1,t2;
    initSCI1(); /*SCI-ch1の初期化*/
    initTimer1Int(10000); /*ITUch1のインターバルタイマ初期化設定*/
                          /*単位μsなので, 時間割り込み10msec*/
    E_INT();         /*CPU割り込み許可*/
    startTimer1();  /*ITUch1タイマスタート*/
    counter=0;
    while(1) {
        counter1=counter;
        t1=counter1/10;
        t2=counter1%10;
        SCI1_printf("%10u.%1u¥r",t1,t2);
    }
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*インターバルタイマ割り込みルーチン*/
{
    static int cnt=0;
    if (++cnt==10) {
        /*この割り込みルーチンは0.01秒ごとに起動するので /
        / 10回に1回の割合でcounter++をすれは, counterは  /
        / 0.1秒にごとに1ずつ増えることにな る             */
        counter++;
        cnt=0;
```

```
    }
    clearTimer1Flag();  /*ITUch1タイマフラグのクリア 忘れないこと*/
}
```

練習問題　int2nd.c, sciout.c, timer.cを元にして次 のプログラムを作りなさい。
（実行の様子を検察して，その結果，気付いた事を含め，別途指定された書式で括弧の
中ファイル名で提出しなさい）

（１）1msec（1000μsec）ごとの割り込みを用いて，ストップウォッチを作成せよ。
スタート・ストップボタンなどはキーボードのキーに割り当てる。
また割り込み関数内で表示を行なってはいけない。グローバル変数のカウンタ変数を
用いればよいだろう。
キーの割り当て
A：スタート（再開，ストップしても再開できる）　B：ストップ　C：クリア
SCI1_getChar()ではなくSCI1_chkgetChar()の使用を考えなさい。
ヒント
どうして割り込み関数内で表示していけないか考えなさい。
H8CPUからPCへのデータ転送は38400bit/secである。1バイト送信するのに１０ビット
の転送が必要である。
割り込み関数は，次の割り込み要求が生ずる前に作業を終了しなければならない。
1msecごとの割り込みなら，割り込み関数内で文字をH8からPCへ転送するとしたら，
何文字程度送信できるか考えなさい。
(mp8ex01.txt)

（２）シリアル通信速度は"h8_3048fone.hの中の関数 initSCI1で設定されている。
この関数の中では通信速度が38400bit/secになるよう設定されている。
（テラタームも38400bit/secに設定されているので通信ができている。）
この部分を自分のファイルに取り込み，19200bit/secの通信速度にした例を示す。
sciout通信速度設定
このプログラムを改造し通信速度を 1200bit/secに設定し，通信を検証しなさい。
ハードウェアマニュアルを読まないとできません。
(mp8ex02.txt)
「テラターム」側の通信速度を設定するには
「設定」メニューから「シリアルポート...」を選び，ダイアログを開く
「ボー・レート」項目の値を変更する
「OK」でもどる。

１０．独立した複数の作業の書き方

リアルタイムOSを使うと，もっと別な書き方になるが，ここではリアルタイムOSを使わない方法でプログラミングする

例題

プッシュスイッチ0を押しした瞬間から２秒間LED0が点灯し、消灯する。
プッシュスイッチ1を押しした瞬間から２秒間LED1が点灯し、消灯する。
ただし、LED0が点灯中にプッシュスイッチ0が押されたら、残り点灯時間には無関係に、その時点から新たに２秒間点灯するものとする。
同様に、LED1が点灯中にプッシュスイッチ1が押されたら、残り点灯時間には無関係に、その時点から新たに２秒間点灯するものとする。
また、LED0が点灯中にプッシュスイッチ1が押されたら、LED0の動作には影響を与えず、LED1は、その時点から２秒間点灯するも のとする。
同様に、LED1が点灯中にプッシュスイッチ0が押されたら、LED1の動作には影響を与えず、LED2は、その時点から２秒間点灯する ものとする。
このような２つの動作を独立動作と呼ぶ。

独立した2つの作業を行う
失敗作（プログラムはわかりやすいが失敗する）

LED0が動作中に，LED1の動作をさ せようと思っても動作を開始しない。
LED0の動作が終了すれば，LED1の動作を開始できる。

```
/************************************************************
PushSW1でLED0をPushSW2でLED0を1秒間点灯させる
この２つの作業は独立に行われるはず？？
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

volatile int count1=-1;/*LED0用カウンタ　-1の時は休止中*/
volatile int count2=-1;/*LED1用カウンタ　-1の時は休止中*/
main()
```

```
{
    initLed();
    initPushSW();/*PushSWの初期化*/
    initTimer1Int(10000); /*時間割り込み10000μsec=10msec ch1使用*/
    E_INT();        /*CPU割り込み許可*/
    startTimer1();  /*時間割り込みタイマスタートch1*/
    while(1) {
        if (count1==-1 && checkPushSW(0)==1) {
            count1=0;
            turnOnLed(0);
            while (count1<200); /*200カウントで2秒経過*/
            count1=-1;
            turnOffLed(0);
        }
        if (count2==-1 && checkPushSW(1)==1) {
            count2=0;
            turnOnLed(1);
            while (count2<200); /*200カウントで2秒経過*/
            count2=-1;
            turnOffLed(1);
        }
    }
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*タイマ割り込みルーチン*/
{
    static int tick=0;
    clearTimer1Flag();  /*タイマステータスフラグのクリア 忘れないこと*/
    if (count1!=-1) count1++;
    if (count2!=-1) count2++;
}
```

独立した2つの作業を行う

```
/************************************************************
PushSW1でLED0をPushSW2でLED0を2秒間点灯させる
この２つの作業は独立に行われる
************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

volatile int count1=-1;/*LED0用カウンタ  -1の時は休止中*/
volatile int count2=-1;/*LED1用カウンタ  -1の時は休止中*/
main()
{
    initLed();
    initPushSW();/*PushSWの初期化*/
    initTimer1Int(10000); /*時間割り込み10000μsec=10msec ch1使用*/
    E_INT();        /*CPU割り込み許可*/
    startTimer1();  /*時間割り込みタイマスタートch1*/
    while(1) {
        if ((count1==-1 || 50<count1) && checkPushSW(0)==1) {
            count1=0;
            turnOnLed(0);
        } else if (200<count1) { /*200カウントで2秒経過*/
            count1=-1;
            turnOffLed(0);
        }
        if ((count2==-1 || 50<count2) && checkPushSW(1)==1) {
            count2=0;
            turnOnLed(1);
        } else if (200<count2) { /*200カウントで2秒経過*/
            count2=-1;
            turnOffLed(1);
```

```
        }
    }
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*タイマ割り込みルーチン*/
{
    static int tick=0;
    clearTimer1Flag();  /*タイマステータスフラグのクリア 忘れないこと*/
    if (count1!=-1) count1++;
    if (count2!=-1) count2++;
}
```

練習問題(1)　(mp9ex01.txt)

プッシュスイッチ0を押した瞬間からLED0は１秒間点灯し、１秒間消灯 し、
１秒間点灯して動作を終了する。（ON-OFF-ON-OFF）　このLED0の動作をLED0定型動作とする。
プッシュスイッチ1を押した瞬間からLED1は１秒間点灯し、１秒間消灯し、
１秒間点灯して動作を終了する。（ON-OFF-ON-OFF）このLED1の動作をLED1定型動作とする。

ただし、LED0が定型動作中にプッシュスイッチ0が押されたら、残り定 型動作時間には無関係に、その時点か
ら新たに定型動作を開始するものとする。
同様に、LED1が定型動作中にプッシュスイッチ1が押されたら残り定型動作時間には無関係に、その時点から
新たに定型動作を開始するものとする。
また、LED0が定型動作中にプッシュスイッチ1が押されたら、LED0の定型動作には影響を与えず、LED1は、
定型動作を開始するものとする。
同様に、LED1が定型動作中にプッシュスイッチ0が押されたら、LED1の定型動作には影響を与えず、LED0
は、
定型動作を開始するものとする。
この動作も前例題と同様に独立動作である。
なお，例題では失敗作と成功例があるが，どのように失敗していて，どのようにして成功させたかを考察で
述べなさい。

練習問題(2)　(mp9ex02.txt)

前課題の２つプッシュスイッチにより起動する独立動作に加え、プッシュスイッチ２によって起動するもう１つの独立動作を行うようにする。
シリアル通信とテラタームを用いて次の動作をおこなう。
テラタームで電光掲示板のように「Hello, everyone!」を右から左に流れるように表示する。表示中に再度プッシュスイッチが押されたら、表示動作は最初から再表示になる。
また、LED0、LED1とこの表示はそれぞれ独立動作である。
電光掲示板のように表示させるには次の文字列を0.1秒ごとに表示すればよい。
最後まで表示したら、この定型動作は終了である。

```
"                H¥r"
"               He¥r"
"              Hel¥r"
"             Hell¥r"
"            Hello¥r"
"           Hello,¥r"
        :
"Hello, everyone!¥r"
"ello, everyone! ¥r"
"llo, everyone!  ¥r"
"lo, everyone!   ¥r"
        :
"e!              ¥r"
"!               ¥r"
"                ¥r"
```

WindowsCプログラミングで，サンプルを作ると次のようになるので参考にするとよい。

```
#include <stdio.h>
```

```
#include <windows.h>

char hello[]="                Hello, everyone!                ";
/*          01234567890123456789012345678901234567890123456789*/

main()
{
    int i,j;
    i=0;
    while (1) {
        Sleep(300);//300msお休み windowsの関数
        printf("<");
        for (j=0; j<16; j++) putchar(hello[i+j]);//i番目から16個表示
        printf(">¥r");
        i++;
        if (i==34) i=0;
    }
}
```

もしHello表示だけなら，次のようなプログラムになるであろう。

```
volatile int count=-1;/*カウンタ  -1の時は休止中*/
volatile int cmdstertptr=-1;/*文字列表示開始位置指令*/

const char hello[]="                Hello, everyone!                ";

main()
{
    int i, j;
    i = 0;

    initSCI1();
    initLed();
    initPushSW();/*PushSWの初期化*/
    initTimer1Int(10000); /*時間割り込み10000μsec=10msec ch1使用*/
    E_INT();        /*CPU割り込み許可*/
    startTimer1();  /*時間割り込みタイマスタートch1*/

    while(1) {
        if ( (count<0 || 10<count) &&checkPushSW(2)==1) {
            count = 0;
            cmdstertptr = 0;
        }
        if(0<=cmdstertptr) {
            SCI1_printf("<");
            for(j = 0; j < 16; j++) {
                SCI1_printf("%c", hello[cmdstertptr+j]);
            }
            SCI1_printf(">¥r");
            if (cmdstertptr==34) {
                count=-1;
            }
            cmdstertptr=-1;
        }
    }
}

#pragma asm
    .SECTION    MYVEC, DATA, LOCATE=H'000070
    .ORG        H'000070  ;IMIA1
    .DATA.L     _TimerIntFunc
    .SECTION    P,CODE,ALIGN=2 ;これを忘れてはいけない
#pragma endasm

#pragma interrupt (TimerIntFunc)
void TimerIntFunc() /*タイマ割り込みルーチン*/
{
    static int tick=0;
    clearTimer1Flag();  /*タイマステータスフラグのクリア 忘れないこと*/
    if (count!=-1) count++;
    if (count%10==0) {
        cmdstertptr=count/10;
    }
}
```

11. LCDへの出力

h8_3048fone.hに定義されている関数を使って，LCDにpushSWの状態を表示する。
initLCD( ) LCDの初期化
LCD_gotoxy(y,x) 表示位置の設定 y:行，x:カラム

LCD_printf( ) LCD版printf

LCDout.c

```
/*************************************************************
LCDへ出力
*************************************************************/
#include <3048fone.h>
#include "h8_3048fone.h"

main()
{
    unsigned char sw,previous;
    initPushSW();
    initLCD(); /*LCDの初期化*/
    LCD_gotoxy(0,0);
    LCD_printf("pushSW status");
    previous=sw=getPushSW();
    LCD_gotoxy(0,1);
    LCD_printf("%02x[%08b]",sw,sw);
    while(1) {
        sw=getPushSW();
        if (sw!=previous) {
            LCD_gotoxy(0,1);
            LCD_printf("%02x[%08b]",sw,sw);
            previous=sw;
        }
    }
}
```

練習問題　(mp9ex03.txt)

10の練習問題(2) (mp9ex02.txt) で提出した課題において，電光掲示板表示をLCDに表示するように変更しなさい。
必要なら表示速度を遅くしなさい。
　(実行の様子を検察して，その結果，気付いた事を含め，提出しなさい)

参考1 H8ピン配置とポート割当

TNCT3048FONE・3052用 マザーボード回路図(pdf)

コネクタ1(CN1)ピン割り当て

CN1 pin	H8 pin	name
1	GND	GND
2	GND	GND
3	87	P8-0
4	88	P8-1
5	89	P8-2
6	90	P8-3
7	91	P8-4
8	93	PA-0
9	94	PA-1
10	95	PA-2
11	96	PA-3
12	97	PA-4
13	98	PA-5
14	99	PA-6

コネクタ2(CN2)ピン割り当て

CN2 pin	H8 pin	name
1	GND	GND
2	GND	GND
3	62	STBY
4	63	RES
5	64	NMI
6	69	P6-3
7	70	P6-4
8	71	P6-5
9	72	P6-6
10	76	AVCC
11	77	AREF
12	78	P7-0
13	79	P7-1
14	80	P7-2

コネクタ3(CN3)ピン割り当て

CN3 pin	H8 pin	name
1	GND	GND
2	GND	GND
3	23	P4-4
4	24	P4-5
5	25	P4-6
6	26	P4-7
7	27	P3-0
8	28	P3-1
9	29	P3-2
10	30	P3-3
11	31	P3-4
12	32	P3-5
13	33	P3-6
14	34	P3-7

コネクタ4(CN4)ピン割り当て

CN4 pin	H8 pin	name
1	–	–
2	GND	GND
3	5V	5V
4	U2-8	RXD1
5	U2-13	RXD0
6	U2-7	TXD1
7	U2-14	TXD0
8	10	FWE
9	75	MD2
10	63	RES

TNCTマザーボードでは1は
使用していない。

コネクタ5(CN5)ピン割り当て

CN5 pin	H8 pin	name
1	GND	GND
2	73	MD-0
3	GND	GND
4	74	MD-1
5	GND	GND
6	75	MD-2

15	100	PA-7
16	2	PB-0
17	3	PB-1
18	4	PB-2
19	5	PB-3
20	6	PB-4
21	7	PB-5
22	8	PB-6
23	9	PB-7
24	10	FWE
25	12	P9-0
26	13	P9-1
27	14	P9-2
28	15	P9-3
29	16	P9-4
30	17	P9-5
31	18	P4-0
32	19	P4-1
33	20	P4-2
34	21	P4-3
35	5V	5V
36	5V	5V
37	GND	GND
38	GND	GND
39	-	-
40	-	-

TNCTマザーボードでは
39，40は
使用していない。

15	81	P7-3
16	82	P7-4
17	83	P7-5
18	84	P7-6
19	85	P7-7
20	86	AVSS

15	36	P1-0
16	37	P1-1
17	38	P1-2
18	39	P1-3
19	40	P1-4
20	41	P1-5
21	42	P1-6
22	43	P1-7
23	45	P2-0
24	46	P2-1
25	47	P2-2
26	48	P2-3
27	49	P2-4
28	50	P2-5
29	51	P2-6
30	52	P2-7
31	53	P5-0
32	54	P5-1
33	55	P5-2
34	56	P5-3
35	58	P6-0
36	59	P6-1
37	60	P6-2
38	61	CK
39	GND	GND
40	GND	GND

ポート割当

TNCT仕様基板で，新しくIOを設置したい場合，ポートを割り当てる必要がある。
どのポートが使えるのかまとめておく

- P1：8ビット入力，出力ポートとして利用可能
- P2：8bitDIPSWが設置されているので利用不可能だが，この8bitDIPSWを外せば8ビット入力，出力ポートとして利用 可能
  プルアップコントロールがついている
- P3：LCD（液晶表示器）が設置されているので利用不可能だが，このLCDを外せば8ビット入力，出力ポートとして利用可能
- P4：上位4ビットにpushSWがついている。もともとは8ビット入力，出力ポートとして利用可能
  プルアップコントロールがついている
- P5：下位2ビット（B0,B1）にLEDがついている。B2,B3にもLEDが付けられるようになっている。
  もともとは下位4ビットのみ入力，出力ポートとして利用可能
  プルアップコントロールがついている
- P6：8ビット入力，出力ポートとして利用可能
- P7：AD変換入力なのでその用途につかう
- P8：IRQ用なのでその用途につかう
- P9：SCI通信用なのでその用途につかう
- PA：PWM波形出力，周期・デューティ測定用入力用(ITU)なのでその用途につかう
  8ビット入力，出力ポートとしても利用可能
- PB：PWM波形出力，周期・デューティ測定用入力用(ITU)なのでその用途につかう
  8ビット入力，出力ポートとしても利用可能

参考２　h8_3048fone.h

h8_3048fone.h

```
/*****************************************************************
```

```
h8_3048fone.h
Copyright (c) Kosaka Lab CS TNCT

このインクルードファイルは小坂研究室の代々の研究生が開発した
h8/3048用の有用な関数群を改良して小坂がまとめたものである。
06 Mar 2013 h8_3048fone.h  小坂 教材用にリファイン
01 Jun 2009 h8-3048.h  小坂 教材用にリファイン
28 Jun 2006 h8-3048.h  小坂 chkgetSCI1のタイミング修正
 4 Dec 2003 h8-3048.h  小坂 printf更新,initLed更新,initDDR削除
08 Oct 2003 h8-3048.h  小坂 stopTimer追加,getIntSCI1でBS使用可
 6 Jan 2003 h8_3048.h  小坂 getIntSCIバックスペイスに対応。
17 Apr 2002 h8-01.h  小坂 %uの使い方をansiにあわせた。
14 Dec 2001 h8-01.h  小坂, 越智
15 Jly 2000 h8-00.h  小坂, 藤原
22 Dec 1999 h8-99.h  小坂, 高沢
29 Oct 1999 h8-99.h  小坂
05 Feb 1999 lib.h    笠井


【１】SCI
【１．１】 ch1 関係
void initSCI1()
 SCI-ch1の初期化  38400baud, Async, 8bit , NoParity, stop1

short int SCI1_getChar()
 SCI-ch1から1byte入力コード。エラーがあると-2が戻る。
short int SCI1_chkgetChar()
 SCI-ch1を検査し, 受信データがあれば1byte入力コード。なければ-1が, 失敗すると -2が戻る。
int SCI1_getString(char *buff,int max)
 SCI1から最大max-1文字の文字列を受け取る。（buffのサイズはmaxでよい）
short int SCI1_getInt(char prompt[])
 SCI-ch1からプロンプト付で, short intの値を受け取る。
 正負の10進数または16進数を受け付ける。16進数は0xで始まる
long int SCI1_getInt4(char prompt[])
 SCI-ch1からプロンプト付で, long intの値を受け取る。
 正負の10進数または16進数を受け付ける。16進数は0xで始まる

void SCI1_putChar(char c)
 SCI-ch1に1バイト出力する。
void SCI1_putString(char *str)
 SCI-ch1に文字列を出力する。
void SCI1_printf(char *format,...)
  関数printfのSCI1版
  軽量化のためエラー処理はないので桁数指定の場合は注意
  対応書式
  %d   : [int] integer with sign. '%d','%4d','%-4d', and '%04d' are available
  %ld  : explicit [long int]  '%ld','%9ld','%-9ld', and '%09ld' are available
  %u   : [unsigbed int] unsigned integer.
           '%u','%4u','%-4u', and '%04u' are available
  %lu  : explicit [unsigned long int]
           '%lu','%9lu','%-9lu', and '%09lu' are available
  %x   : [unsigned int] in Hex  '%x','%4x','%-4x', and '%04x' are available
  %lx  : explicit [unsigned long int] in Hex
           '%lx','%8lx','%-8lx', and '%08lx' are available
  %o   : [unsigned int] in Oct  '%o','%4o','%-4o', and '%04o' are available
  %lo  : explicit [unsigned long int] in Oct
           '%lo','%8lo','%-8lo', and '%08lo' are available
  %b   : [unsigned int] in Bin  '%b','%8b','%-8b', and '%08b' are available
  %lb  : explicit [unsigned long int] in Bin
           '%lb','%8lb','%-8lb', and '%08lb' are available
  %c   : char
  %s   : string %20s %-20s are available

【１．２】SCI ch0 関係
void initSCI0()
 SCI-ch0の初期化  38400baud, Async, 8bit , NoParity, stop1

short int SCI0_getChar()
 SCI-ch0から1byte入力コード。エラーがあると-2が戻る。
short int SCI0_chkgetChar()
 SCI-ch0を検査し, 受信データがあれば1byte入力コード。なければ-1が, 失敗すると -2が戻る。
int SCI0_getString(char *buff,int max)
```

 SCIOから最大max-1文字の文字列を受け取る。（buffのサイズはmaxでよい）
short int SCIO_getInt(char prompt[])
 SCI-ch0からプロンプト付で，short intの値を受け取る。
 正負の10進数または16進数を受け付ける。16進数は0xで始まる
long int SCIO_getInt4(char prompt[])
 SCI-ch0からプロンプト付で，long intの値を受け取る。
 正負の10進数または16進数を受け付ける。16進数は0xで始まる

void SCIO_putChar(char c)
 SCI-ch0に1バイト出力する。
void SCIO_putString(char *str)
 SCI-ch0に文字列を出力する。
void SCIO_printf(char *format,...)
  関数printfのSCIO版
  仕様はvoid SCI1_printf(char *format,...)参照

【２】LCD関係
void LCD_putchar(char data)
  LCDに向けた putchar()
void LCD_puts(char *str)
  LCDに向けた puts()
void initLCD(void)
  LCDの初期化
void LCD_gotoxy(unsigned x,unsigned y)
  LCDに向けた gotoxy()
void LCD_clrscr(void)
  LCDに向けた clrscr()   clear screen
void LCD_printf(char *format,...)
  関数printfのLCD版
  仕様はvoid SCI1_printf(char *format,...)参照

【３】文字列操作
void sprintf(char *buff,char *format,...)
  仕様はvoid SCI1_printf(char *format,...)参照
  buffのあふれは呼び出し側で起こらないようにしておく必要がある
long int atoi(char *buff)
  文字列を整数型の値に変換する
  正負の10進数または16進数の文字列を受け付ける。16進数は0xまたは-0xで始まる
 例 "123"
    "-123"
    "0x1a"
    "-0x100"

【４】AKI-H8マザーボード関係
void initLed()
 LEDの初期化
void turnOnLed(short int number)
 LEDの点灯 numberはLED番号で0または1を指定する
void turnOffLed(short int number)
 LEDの消灯 numberはLED番号で0または1を指定する

void initPushSW(void)
 押しボタンスイッチの初期化
unsigned char getPushSW(void)
 押しボタンスイッチの取得 ビット反転し，ONは1，OFFは0で取得される。
 押しボタンスイッチの状況は第0-第3ビットに現れる。
 これはマクロ定義で実現されている
short int checkPushSW(short int number)
 push sw 0,1,2,3の状態を調べる number:0,1,2,or 3
 押されていたら1、そうでなかったら0を返す

void init8BitSW(void)
 8ビットスイッチの初期化
unsigned char get8BitSW(void)
 8ビットスイッチの取得 ビット反転し，ONは1，OFFは0で取得される。
 8ビットスイッチの状況は第0-第7ビットに現れる。
 これはマクロ定義で実現されている
short int check8BitSW(short int number)
 8bitsw 0,1,2,3,4,5,6,7の状態を調べる number:0,1,2,3,4,5,6,or 7
 ONなら1、そうでなかったら0を返す

【５】インターバルタイマ割り込み

```
void initTimer1Int(unsigned short int period)
 ITU1による割り込みタイマーの設定
 割り込み間隔は引数peiodで単位はμsecである
 値は20971以下でなければならない。20.971msecまで設定可能

void startTimer1(void)
 Timer CH1 スタート
 これはマクロ定義で実現されている
void stopTimer1(void)
 Timer CH1 ストップ
 これはマクロ定義で実現されている
void clearTimer1Flag(void)
 Timer CH1 割り込みフラグのクリア
 これはマクロ定義で実現されている


*****************************************************************/

#include<stdarg.h>

extern void E_INT();
extern void D_INT();

/*次の1行はMicrosoftHyperTerminal使用時に有効にする*/
/*有効な場合H8からシリアル送信時に¥nを¥r¥nに自動変換する*/
#define HYPERTERMINAL

/*システムクロック関係*/
#define SYS_CLOCK 25 /*MHz 16,20,25 程度を想定*/

const int SCI0device=0;
const int SCI1device=1;
const int LCDdevice=2;
const int STRdevice= 3;

/*SCI関係の基本部分は笠井君(1998年度)藤原君(2000)の開発です*/
/* ------------------------------------------------ */
/* SCI1 INITIALIZATION fixed baud at 38400          */
/* ------------------------------------------------ */
void initSCI1()
{
    short int i;
    SCI1.SCR.BYTE = 0;       /* clear all flags */
                    /* 2400-38400baud are available at n=0(cks1=0,cks2=0) */
    SCI1.SMR.BYTE = 0;       /* Async, 8bit , NoParity, stop1, 1/1 */
    SCI1.BRR = (unsigned char)((8138L*SYS_CLOCK+5000L)/10000L-1);
    for(i=0;i<1000;i++);      /* wait more than 1bit time */
    SCI1.SCR.BYTE = 0x30;    /* scr = 0011 0000 (TE=1,RE=1) */
    return;
}

/* ------------------------------------------------ */
/* GET BYTE FROM SCI1 */
/* ------------------------------------------------ */
short int SCI1_getChar()
/* return value 0x00-0xFF:received data */
/*              -2(0xFFFE):error */
{
    short int flags,recdata;
    do {
        flags = SCI1.SSR.BYTE;
        if (flags&0x38) {/* error */
            SCI1.SSR.BIT.RDRF = 0;
            SCI1.SSR.BIT.ORER = 0;
            SCI1.SSR.BIT.FER = 0;
            SCI1.SSR.BIT.PER = 0;
            return -2;
        }
        if (flags&0x40) {/* normally received one data */
            SCI1.SSR.BIT.RDRF = 0;
            recdata=SCI1.RDR;
            return recdata;
```

```
        }
    } while (1);
}

/* ------------------------------------------------ */
/* CHECK SCI BUFFER AND GET DATA */
/* ------------------------------------------------ */
short int SCI1_chkgetChar()
/* return value -1(0xFFFF):no received data  */
/*              0x00-0xFF:received data */
/*              -2(0xFFFE):error */
{
    short int flags,recdata;
    flags = SCI1.SSR.BYTE;
    if (flags&0x38) {/* error */
        SCI1.SSR.BIT.RDRF = 0;
        SCI1.SSR.BIT.ORER = 0;
        SCI1.SSR.BIT.FER = 0;
        SCI1.SSR.BIT.PER = 0;
        return -2;
    }
    if (flags&0x40) {/* normally received one data */
        recdata=SCI1.RDR;
        SCI1.SSR.BIT.RDRF = 0;
        return recdata;
    } else {
        return -1;
    }
}

void SCI1_putString(char *str);

/*SCI1より文字列入力[return]が終端だが, '¥n'は取得されない*/
/*^Hでバックスペイス*/
int SCI1_getString(char *buff,int max)
{
    int i,ch;
    for (i=0;i<max-1;i++) {
        ch=SCI1_getChar(); /*1文字取得*/
        *buff=(char)ch; /*1文字取得*/
        if (*buff=='¥r'||ch<0) {
            *buff=0;
            return i+1;
        }
        if (*buff==0x8) {
            buff-=2;
            i-=2;
        }
        if (*buff!='¥n') buff++;
        else i--;
    }
    *buff=0;
    return i+1;
}

/* ------------------------------------------------ */
/* PUT BYTE TO SCI1 */
/* ------------------------------------------------ */
void SCI1_putChar(char c)
{
    unsigned char tmp;
#ifdef HYPERTERMINAL
    if (c=='¥n') SCI1_putChar('¥r');
#endif
    do{
        tmp = SCI1.SSR.BYTE;
    } while((tmp & 0x80)==0);
    SCI1.TDR = c;
    SCI1.SSR.BIT.TDRE = 0;
    return;
}
```

```
void SCI1_putString(char *str)
{
    while(*str){
        SCI1_putChar(*str);
        str++;
    }
}


/* ------------------------------------------------ */
/* SCI0 INITIALIZATION fixed baud at 38400          */
/* ------------------------------------------------ */
void initSCI0()
{
    short int i;
    SCI0.SCR.BYTE = 0;       /* clear all flags */
                    /* 2400-38400baud are available at n=0(cks1=0,cks2=0) */
    SCI0.SMR.BYTE = 0;       /* Async, 8bit , NoParity, stop1, 1/1 */
    SCI0.BRR = (unsigned char)((8138L*SYS_CLOCK+5000L)/10000L-1);
    for(i=0;i<1000;i++);      /* wait more than 1bit time */
    SCI0.SCR.BYTE = 0x30;    /* scr = 0011 0000 (TE=1,RE=1) */
    return;
}

/* ------------------------------------------------ */
/* GET BYTE FROM SCI0 */
/* ------------------------------------------------ */
short int SCI0_getChar()
/* return value 0x00-0xFF:received data */
/*              -2(0xFFFE):error */
{
    short int flags,recdata;
    do {
        flags = SCI0.SSR.BYTE;
        if (flags&0x38) {/* error */
            SCI0.SSR.BIT.RDRF = 0;
            SCI0.SSR.BIT.ORER = 0;
            SCI0.SSR.BIT.FER = 0;
            SCI0.SSR.BIT.PER = 0;
            return -2;
        }
        if (flags&0x40) {/* normally received one data */
            SCI0.SSR.BIT.RDRF = 0;
            recdata=SCI0.RDR;
            return recdata;
        }
    } while (1);
}

/* ------------------------------------------------ */
/* CHECK SCI BUFFER AND GET DATA */
/* ------------------------------------------------ */
short int SCI0_chkgetChar()
/* return value -1(0xFFFF):no received data  */
/*              0x00-0xFF:received data */
/*              -2(0xFFFE):error */
{
    short int flags,recdata;
    flags = SCI0.SSR.BYTE;
    if (flags&0x38) {/* error */
        SCI0.SSR.BIT.RDRF = 0;
        SCI0.SSR.BIT.ORER = 0;
        SCI0.SSR.BIT.FER = 0;
        SCI0.SSR.BIT.PER = 0;
        return -2;
    }
    if (flags&0x40) {/* normally received one data */
        recdata=SCI0.RDR;
        SCI0.SSR.BIT.RDRF = 0;
        return recdata;
    } else {
        return -1;
```

```
    }
}

void SCIO_putString(char *str);
/*SCIOより文字列入力[return]が終端だが, '¥n'は取得されない*/
/*^Hでバックスペイス*/
int SCIO_getString(char *buff,int max)
{
    int i,ch;
    for (i=0;i<max-1;i++) {
        ch=SCIO_getChar(); /*1文字取得*/
        *buff=(char)ch; /*1文字取得*/
        if (*buff=='¥r'||ch<0) {
            *buff=0;
            return i+1;
        }
        if (*buff==0x8) {
            buff-=2;
            i-=2;
        }
        if (*buff!='¥n') buff++;
        else i--;
    }
    *buff=0;
    return i+1;
}

/*拡張atoi*/
/* 123,-123,0x1a,-0x1a の形の文字列をlong intに変換する*/
long int atoi(char *buff)
{
    long int x=0;
    int y,m=0,n=0,v=0,i=0;
    y=buff[i];
    while(y!=0){
        if(y=='-') m=1;
        if('a'<=y&&y<='z') y=y-'a'+'A';
        if(y=='0') n=1;

        if(v==1){
            if('0'<=y&&y<='9'){
                y=y-'0';
            }
            else if('A'<=y&&y<='F'){
                y=y-'A'+10;
            }
            x=16*x+y;
        }

        if(n==1&&y=='X'){
            v=1;
        }

        if(v==0&&'0'<=y&&y<='9'){
            y=y-'0';
            x=10*x+y;
        }

        y=buff[++i];
    }
    if(m==1) x=-x;
    return x;
}

#define SCI1_getInt(prompt) ((short int)SCI1_getInt4(prompt))

/*SCI1へプロンプトを表示して, SCI1より整数値を入力*/
long int SCI1_getInt4(char prompt[])
/*getting integer from serial port*/
/* format 123[ret] */
/*        -123[ret] */
/*        0x1a[ret] */
```

```
/*        -0x100[ret] */
{
    char buff[16];
    SCI1_putString(prompt);
    SCI1_getString(buff,16);
    return atoi(buff);
}

#define SCI0_getInt(prompt) ((short int)SCI0_getInt4(prompt))

/*SCI0へプロンプトを表示して，SCI0より整数値を入力*/
long int SCI0_getInt4(char prompt[])
/*getting integer from serial port*/
/* format 123[ret] */
/*        -123[ret] */
/*        0x1a[ret] */
/*        -0x100[ret] */
{
    char buff[16];
    SCI0_putString(prompt);
    SCI0_getString(buff,16);
    return atoi(buff);
}


/* ------------------------------------------------ */
/* PUT BYTE TO SCI0 */
/* ------------------------------------------------ */
void SCI0_putChar(char c)
{
    unsigned char tmp;
#ifdef HYPERTERMINAL
    if (c=='¥n') SCI0_putChar('¥r');
#endif
    do{
        tmp = SCI0.SSR.BYTE;
    } while((tmp & 0x80)==0);
    SCI0.TDR = c;
    SCI0.SSR.BIT.TDRE = 0;
    return;
}

void SCI0_putString(char *str)
{
    while(*str){
        SCI0_putChar(*str);
        str++;
    }
}



/* Port3 -> LCD */
/* 7    6    5    4    3    2    1    0  */
/*         ES   RS  DB7  DB6  DB5  DB4  */

/* i/o */
#define init_LCD_Port()     P3.DDR = 0x3f
#define LCD_Port            P3.DR.BYTE
#define LCD_RegisterSelect  P3.DR.BIT.B4
#define LCD_EnableSignal    P3.DR.BIT.B5

static void LCD_waitmsec(unsigned int msec)
/*mesc間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<168*SYS_CLOCK;j++);
    }
}

static void LCD_waitmicrosec(unsigned int microsec)
```

```
{/*だいたいmicrosec間なにもしない時間稼ぎ関数*/
    int i,j;
    for (i=0;i<microsec;i++) {
        for (j=0;j<168*SYS_CLOCK/1000;j++);
    }
}

static void LCD_putCommand(char command) /*command width must be 4bits*/
{
    LCD_Port=(LCD_Port&0xf0)|command;
    LCD_EnableSignal=1;
    LCD_waitmicrosec(5);
    LCD_EnableSignal=0;
    LCD_waitmicrosec(40);
}

static void LCD_putCommand2(char command2)
{
    LCD_putCommand((command2>>4)&0xf);
    LCD_putCommand(command2&0xf);
}

/*************************************************************
LCDに向けた putchar()
*************************************************************/
void LCD_putchar(char data)
{
    LCD_RegisterSelect=1;
    LCD_putCommand((data>>4)&0xf);
    LCD_putCommand(data&0xf);
    LCD_RegisterSelect=0;
}

/*************************************************************
LCDに向けた puts()
*************************************************************/
void LCD_puts(char *str)
{
    while(*str) LCD_putchar(*str++);
}

/*************************************************************
LCDの初期化
*************************************************************/
void initLCD( void )
{
    init_LCD_Port();        /* output */
    LCD_Port&=0xc0;
    LCD_waitmsec(30);           /* wait 30ms */
    LCD_putCommand(0x3);     /* function set */
    LCD_waitmicrosec(4100);
    LCD_putCommand(0x3);     /* function set */
    LCD_waitmicrosec(100);
    LCD_putCommand(0x3);     /* function set */
    LCD_waitmicrosec(100);
    LCD_putCommand(0x2);     /* function set data width=4bit*/
    LCD_putCommand2(0x28);   /* function set 4bit duty:1/16,size:5*7 */
    LCD_putCommand2(0x0c);   /* display on,cursor off,blink off */
    LCD_putCommand2(0x06);   /* address:auto increment,cursor shift:right */
    LCD_putCommand2(0x01);   /* clear display */
    LCD_waitmicrosec(1640);
}

/*************************************************************
LCDに向けた gotoxy()
*************************************************************/

void LCD_gotoxy(unsigned x,unsigned y)
{
    unsigned char point;
    point=0x80+x+0x40*y;
    LCD_putCommand2(point);
```

```
}

/*************************************************************
LCDに向けた clrscr()   clear screen
*************************************************************/
void LCD_clrscr(void)
{
    LCD_putCommand2(0x01);    /* clear display */
    LCD_waitmicrosec(1640);
}

static char *currentstrptr;
static void STR_putChar(char ch)
{
    *currentstrptr++=ch;
    *currentstrptr=0;
}

static void STR_putString(char *ptr)
{
    while(*ptr) STR_putChar(*ptr++);
}


const char hexstring[]="0123456789abcdef0123456789ABCDEF";
#define MAXDIGIT 34

void Device_printf(int device, char *format, va_list arg_ptr)
{
    void (*device_putchar)(char ch);
    void (*device_puts)(char *ptr);
    char buf[MAXDIGIT];
    unsigned char flag=0;  /*%d:bit2 l:bit1 %:bit0 */
    unsigned char digit=0; /* 桁数 */
    unsigned char minus=0;
    char fill=' ',format1=' ';
    unsigned char radix=10; /*N進基数*/
    char sign=' ';
    char *ptr=buf; /*出力文字ポインタ*/
    unsigned char cntr=0; /*出力文字数カウンタ*/
    unsigned char shift=0; /*16進シフト 0 or 6*/
    unsigned char i;
    unsigned long int value=0;
    if (device==SCI0device) {
        device_putchar=SCI0_putChar;
        device_puts=SCI0_putString;
    } else if (device==SCI1device) {
        device_putchar=SCI1_putChar;
        device_puts=SCI1_putString;
    } else if (device==LCDdevice) {
        device_putchar=LCD_putchar;
        device_puts=LCD_puts;
    } else { /*device==STRdevice*/
        device_putchar=STR_putChar;
        device_puts=STR_putString;
    }
    /*va_start(arg_ptr,format);*/
    while (*format) {
        format1=*format;
        if (flag==0) {
            if (format1=='%') {
                flag=1;
                digit=0;
                fill=' ';
                minus=0;
                radix=0;
                ptr=&buf[MAXDIGIT-1];
                *ptr--='¥0';
                cntr=0;
                shift=0;
                sign='+';
            } else {
```

```
            device_putchar(format1);
        }
    } else {
        if (format1=='l') {
            flag|=2;
        } else if ('0'<=(format1)&&(format1)<='9') {
            if (digit==0 && format1=='0') {
                fill='0';
            } else {
                digit=digit*10+((format1)-'0');
                if (MAXDIGIT-2<digit) digit=MAXDIGIT-2;
            }
        } else if (format1=='-') {
            minus=1;
        } else if (format1=='d') {
            flag|=4;
            radix=10;
        } else if (format1=='u') {
            radix=10;
        } else if (format1=='x') {
            radix=16;
        } else if (format1=='X') {
            radix=16;shift=16;
        } else if (format1=='o') {
            radix=8;
        } else if (format1=='b') {
            radix=2;
        } else if (format1=='p') {
            radix=16;shift=16;digit=8;fill='0';flag|=2;
        } else if (format1=='c') {
            device_putchar((unsigned char)(va_arg(arg_ptr,int)));
            flag=0;
        } else if (format1=='s') {
            if (digit) {
                cntr=0;ptr=va_arg(arg_ptr,char *);
                while (ptr[cntr]) cntr++; /*cntrは文字数*/
                if (!minus) for (i=cntr;i<digit;i++) device_putchar(' ');
                device_puts(ptr);
                if (minus) for (i=cntr;i<digit;i++) device_putchar(' ');
            } else {
                device_puts(va_arg(arg_ptr,char *));
            }
            flag=0;
        } else {
            device_putchar(format1);
            flag=0;
        }
        if (radix) {
            switch (flag&6) {
            case 0: /* unsig int */
                value=(unsigned int)va_arg(arg_ptr,int);
                break;
            case 2: /* unsig long int */
                value=va_arg(arg_ptr,long int);
                break;
            case 4: /* sig int */
                value=(long int)va_arg(arg_ptr,int);
                if ((long int)value<0) {
                    value=-(long int)value;
                    sign='-';
                }
                break;
            case 6: /* sig long int */
                value=va_arg(arg_ptr,long int);
                if ((long int)value<0) {
                    value=-(long int)value;
                    sign='-';
                }
                break;
            default:
                break;
            }
```

```
                while (value) {
                    *ptr--=hexstring[value%radix+shift];
                    cntr++;
                    value/=radix;
                }
                if (cntr==0) {
                    *ptr--='0';
                    cntr++;
                }
                if (fill==' ') {
                    if (sign=='-') {
                        *ptr--='-';
                        cntr++;
                    }
                    if (!minus) for (i=cntr;i<digit;i++) device_putchar(' ');
                    device_puts(++ptr);
                    if (minus) for (i=cntr;i<digit;i++) device_putchar(' ');
                } else {
                    for (;cntr<digit-1;cntr++) *ptr--='0';
                    if (sign!='-'&&cntr<digit) *ptr--='0';
                    else if (sign=='-') *ptr--='-';
                    device_puts(++ptr);
                }
                flag=0;
            }
        }
        format++;
    }
    /*va_end(arg_ptr);*/
}

void SCI0_printf(char *format,...)
{
    va_list arg;
    va_start(arg, format);
    Device_printf(SCI0device, format, arg);
    va_end(arg);
}

void SCI1_printf(char *format,...)
{
    va_list arg;
    va_start(arg, format);
    Device_printf(SCI1device, format, arg);
    va_end(arg);
}

void LCD_printf(char *format,...)
{
    va_list arg;
    va_start(arg, format);
    Device_printf(LCDdevice, format, arg);
    va_end(arg);
}

void sprintf(char *buff,char *format,...)
{
    va_list arg;
    va_start(arg, format);
    currentstrptr=buff;
    Device_printf(STRdevice, format, arg);
    va_end(arg);
}

/* ------------------------------------------------ */
/* LED INITIALIZATION */
/* ------------------------------------------------ */
/*********************************************************
LED 0:P5-0
LED 1:P5-1
LED 2:P5-2
LED 3:P5-3
```

```
下位4ビットを出力にする
*********************************************************/
void initLed()
{
    P5.DDR = 0xf;
}

/* ----------------------------------------------- */
/* LET LED ON */
/* ----------------------------------------------- */
/*numberは0または1*/
void turnOnLed(short int number)
{
    static const unsigned char mask[]={1,2,4,8};
    P5.DR.BYTE |= mask[number];
}

/* ----------------------------------------------- */
/* LET LED OFF */
/* ----------------------------------------------- */
/*numberは0または1*/
void turnOffLed(short int number)
{
    static const unsigned char mask[]={0xfe,0xfd,0xfb,0xf7};
    P5.DR.BYTE &= mask[number];
}

/* ----------------------------------------------- */
/* PUSH SW INITIALIZATION */
/* ----------------------------------------------- */
/*********************************************************
押しボタンスイッチS0：P4-4
押しボタンスイッチS1：P4-5
押しボタンスイッチS2：P4-6
押しボタンスイッチS3：P4-7
上位4ビットを入力にする. 下位4ビットも一応入力にする
上位4ビットのプルアップコントロールをONにする
*********************************************************/
void initPushSW(void)
{
    P4.DDR = 0;
    P4.PCR.BYTE = 0xf0; /*P4-4,5,6,7はプルアップ */
}

/* ----------------------------------------------- */
/* GET PUSH SW  */
/* ----------------------------------------------- */
/*push swのポートを取得し, 反転し, 右4シフトして返す
unsigned char getPushSW(void)
{
    return ((((unsigned char)(~P4.DR.BYTE))&0xf0)>>4);
}
*/
#define getPushSW() ((((unsigned char)(~P4.DR.BYTE))&0xf0)>>4)

short int checkPushSW(short int number)
/*push sw 0,1,2,3の状態を調べる  number:0,1,2,or 3*/
/*押されていたら1、そうでなかったら0を返す*/
{
    short int ret;
    static const unsigned char mask[]={0x10,0x20,0x40,0x80};
    if (P4.DR.BYTE&mask[number]) ret=0;
    else ret=1;
    return ret;
}

/* ----------------------------------------------- */
/* PUSH 8 BIT SW INITIALIZATION */
/* ----------------------------------------------- */
/*********************************************************
8ビットスイッチS0：P2-0
8ビットスイッチS1：P2-1
```

```
8ビットスイッチS2：P2-2
8ビットスイッチS3：P2-3
8ビットスイッチS4：P2-4
8ビットスイッチS5：P2-5
8ビットスイッチS6：P2-6
8ビットスイッチS7：P2-7
全8ビットを入力にする.
全8ビットのプルアップコントロールをONにする
*********************************************************/
void init8BitSW(void)
{
    P2.DDR = 0;
    P2.PCR.BYTE = 0xff;/*8bitSWのプルアップ設定*/
}

/*8bitswのポートを取得し，反転して返す
unsigned char get8BitSW(void)
{
    return (unsigned char)(~P2.DR.BYTE);
}
*/
#define get8BitSW() (unsigned char)(~P2.DR.BYTE)

short int check8BitSW(short int number)
/*8bitsw 0,1,2,3,4,5,6,7の状態を調べる  number:0,1,2,3,4,5,6,or 7*/
/*ONなら1、そうでなかったら0を返す*/
{
    short int ret;
    static const unsigned char mask[]={1,2,4,8,0x10,0x20,0x40,0x80};
    if (P2.DR.BYTE&mask[number]) ret=0;
    else ret=1;
    return ret;
}

/*インターバルタイマ割り込みオリジナルは笠井君(1998)，越智君(2001)による開発です*/
/* ------------------------------------------------ */
/* TIMER INITIALIZATION */
/* ------------------------------------------------ */
void initTimer1Int(unsigned short int period)
/*ITU1による割り込みタイマーの設定(越智君2001による)*/
/*割り込み間隔は引数peiodで単位はμsecである*/
/*値は20971以下でなければならない*/
/*20.971msecまで*/
/*割り込みベクトル  H'000070  ;IMIA1*/
{
    unsigned int periodGRA=(unsigned int)((SYS_CLOCK*(long int)period+4)>>3);
    ITU1.TCR.BIT.CCLR=1;  /*GRAのコンペアマッチでTCNTをクリア*/
    ITU1.TCR.BIT.CKEG=0;  /*立ち上がりエッジでカウント*/
    ITU1.TCR.BIT.TPSC=3;  /*内部クロックφ/8でカウント*/
    ITU1.GRA=periodGRA-1;  /*割り込みの周期をperiod[μs]に指定*/
    ITU1.TIER.BIT.IMIEA=1; /*TCNT=GRAとなったときの割り込み要求を許可*/
    ITU1.TIER.BIT.OVIE=0;  /*オーバー・アンダーフロー発生時の割り込みを禁止*/
    ITU1.TIER.BIT.IMIEB=0;  /*TCNT=GRBとなったときの割り込みを禁止*/
}

/* ------------------------------------------------ */
/* TIMER START */
/* ------------------------------------------------ */
/*
Timer CH1 スタート
void startTimer1(void)
{
    ITU.TSTR.BYTE |= 0x02;
}

Timer CH1 ストップ
void stopTimer1(void)
{
    ITU.TSTR.BYTE &= ~0x02;
}

Timer CH1 割り込みフラグのクリア
```

```
void clearTimer1Flag(void)
{
    ITU1.TSR.BIT.IMFA=0;
}

*/
#define startTimer1()   (ITU.TSTR.BYTE |= 0x02)  /* Timer CH1 スタート    */
#define stopTimer1()    (ITU.TSTR.BYTE &= ~0x02) /* Timer CH1 ストップ    */
#define clearTimer1Flag() (ITU1.TSR.BIT.IMFA=0) /* Timer CH1 flagクリア */
```