試作検討 1

創造設計第二 TA：杉浦 元將, 岩淵 教郎

平成 20 年度 10 月 6 日,9 日

1. はじめに

今回の試作検討では開発環境に慣れることを主な目的とする．しかし，ここで使用するボードセットはメカトロニクスラボで使用したものと同一のものなので，今回の試作検討ではメカトロニクスラボの復習が主な内容となっている．実際に取り組む内容としては以下のようになっている．

- 開発環境の理解
- ポート入出力
- 割り込み（タイマ割り込みと外部割り込み）
- メカニカルスイッチの使い方（チャタリングの防止）

まず，開発環境に慣れるためにサンプルプログラムを実行し，動作を確認する．次に割り込みの方法（タイマ割り込みと外部割り込み）について確認する．最後にメカニカルスイッチのチャタリング防止回路を作成し，スイッチの信号を外部入力とする割り込みプログラムを作成する．

警告	各グループのメンバー全員が内容を理解できるように，確認し合って進めること．

警告	試作検討のプログラムや資料は必要に応じて創造設計第二のホームページ (http://www.cyb.mei.titech.ac.jp/2008SS2/main.htm) からダウンロードすること

2. 開発環境の理解

ここでは，サンプルプログラムを用いてプログラムを実行するまでの手順を確認する．以下では用意した「メカトロニクスラボ マイクロコントローラ編」資料を参照しつつ進めていく．

本マイコンボードでプログラムを実行するために，統合化開発環境 TM を用いる．TM はコンパイラ，アセンブラ，デバッガ，エディタなどのツールを GUI に統合して，開発効率を向上させるためのツールである．また，TM はソフトウェアの開発に必要な情報をプロジェクトとして管理しているので，サンプルプログラムを実行するにあたって，新規にプロジェクトを作成する．

サンプルプログラムの実行

1. メカトロニクスラボ資料の P7-13 を参照し，TM の設定を行う．なお，エディタは Notepad, Meadow, もしくはデバッガに内蔵されているエディタ (メカトロニクスラボ資料 P109 参照) がある．
2. 新規プロジェクトの雛形フォルダに必要なファイル (shisaku01/sys_test) があるのでこれを H ドライブの直下にコピーする．フォルダ src には c ソースファイルとヘッダファイルがあり，startup フォルダにはスタートアップファイル ncrt0_ss2.a30，sect30_ss2.inc が用意されている．これを元に新規プロジェクトを作成する．

注意	プロジェクトのワーキングディレクトリとして, 全角文字や空白を入れてはいけない

警告	T ドライブ以下のファイルは学生共通のフォルダなので絶対に消したり編集したりしないでください.

注意	新規プロジェクトの作成方法や，サンプルプログラムのコンパイル，デバッガの起動方法は「メカトロニクスラボ」の資料を参照すること．この資料の中に新規プロジェクトの作成方法などが記されている.

2.1 ポート入出力

I/O ポートは特定のアドレスからデータを読むことで入力ピンの電圧状態を調べ，また特定のアドレスにデータを書き込むことで出力ピンの電圧を決定できる．各ポートの入出力は方向レジスタによって 1 本ごとに決定でき，4 本ごとにプルアップを行うか選択できる（プルアップ制御レジスタ）．このプルアップ機能はセンサなどの入力を使用するときに，外部回路でプルアップする代わりに利用することできる.

プログラマブル入出力ポートは $P1_5$ ~ $P1_7$（3 本）, $P6$（8 本）, $P7$（8 本）, $P8_0$ ~ $P8_3$, $P8_5$ ~ $P8_7$（7 本）, $P9_0$ ~ $P9_3$（4 本）, $P10$（8 本）で合計 38 本ある．詳細な内容は「ハードウェアマニュアル」の P162 以後を参照．ハードウェアマニュアル（rjj09B0033_m16chm.pdf）は [スタート]-[プログラム（P）]-[RENESAS-TOOLS]-[Document]-[ハードウェアマニュアル] 内にある.（C:MTOOL/Document/HW）

2.2 I/O ポートの設定

I/O ポートの機能を設定するためには，まず設定したいポート番号に対応した SFR のアドレスを調べて，ソースプログラムに記述する.（これは sfr26.h というヘッダファイルに記述されている.）次にポート方向レジスタで入出力方向を指定する．ここで必要な場合はプルアップの設定も行う．後はポートレジスタの値を読み書きすることでポートの入出力を行う.

```
#include "sfr26.h" // OAKSmini 用定義ファイル
re
unsigned char LED_out;
unsigned int TGL_out;
pd1 = 0x00; //ポート 1 方向レジスタ：入力
pd10 = 0x00; //ポート 10 方向レジスタ：入力
pd7 = 0x3f; // P7-6,7: 入力．残りは出力.

// トグルスイッチの状態を読む
TGL_out = 0;
if ( ( p10 & 0x80 ) == 0 ) { TGL_out = TGL_out + 1000; }
if ( ( p10 & 0x40 ) == 0 ) { TGL_out = TGL_out +  100; }
if ( ( p10 & 0x20 ) == 0 ) { TGL_out = TGL_out +   10; }
if ( ( p10 & 0x10 ) == 0 ) { TGL_out = TGL_out +    1; }

// プッシュスイッチの状態を読んで，対応する LED を点灯する.
LED_out = 0;
if ( ( p1 & 0x80 ) == 0 ) { LED_out = LED_out | 0x20; }
if ( ( p1 & 0x40 ) == 0 ) { LED_out = LED_out | 0x08; }
if ( ( p1 & 0x20 ) == 0 ) { LED_out = LED_out | 0x02; }
p7 = ~LED_out;
```

ポートの入力には主にビット演算を用いる．ビット演算には主に&（and）と|（or）がある．and は両方のビットが 1 の場合のみ結果が 1 になり，特定のビットをマスクする（有効にする）のに利用する．or では少なくとも一つのビットが 1 なら結果が 1 になり，両方のビットが 0 の場合のみ結果が 0 になる．これは特定のビットを必ず 1 にしたいような場合に用いる．他にも変数のビット並びを右や左にシフトさせるシフト演算子などがある．以下にビット処理の例を示す．

a=0x5555 & 0x00ff

	0101	0101	0101	0101	(5555)
and)	0000	0000	1111	1111	(00ff)
a =	0000	0000	0101	0101	(0055)

a=0x5555 | 0x00ff

	0101	0101	0101	0101	(5555)
or)	0000	0000	1111	1111	(00ff)
a =	0101	0101	1111	1111	(0055)

short int a = 0xf00f;
a= a << 1

	1111	0000	0000	1111	(f00f)
a =	1110	0000	0001	1110	(e01e)

上の and 処理例は下位 8 ビットをマスクしている（有効にしている）．また，or 処理例は下位 8 ビットを強引に 1 にしている．最後の例は左に 1 ビットだけシフトしている．

2.3 LCD

MCU ボード上に設置してある LCD（Lyquid Crystal Display；液晶ディスプレイ）を用いてセンサの情報などを表示することができる．LCD の表示用関数は **LCDfunc.c** 内にあり，これらの関数による LCD の表示方法について説明する．

```
#include <stdio.h>// sprintf() を使うため
#include "LCDfunc.h" // LCD 表示関数をインクルード

char buff[64]; // バッファを確保

// LCD の初期化
LCD_init(); // LCD 初期設定
LCD_cls();  // 全て消去

// カーソル位置の決定
LCD_locate( 0, 0, 0, 0 ); // カーソル位置 (0,0), カーソル OFF, 点滅しない

// 表示
//sprintf を利用した場合
//sprintf はメモリを大量消費するので使ってはいけない
/*
  sprintf( buff, " %4d %4d %4d  %4d %4d %04d ",
            ad2_result, ad1_result, ad0_result,
            count4, count3, TGL_out);
  LCD_print_str( buff );  // 数値で表示
*/

//itoa を利用した場合 ---> nosprintf.c
//itoa(integer to ascii : 整数を 10 進数表記にして文字列に変換する)
/*
  itoa(buff, ad2_result,5);
  itoa_cat(buff, ad1_result,5);
  itoa_cat(buff, ad0_result,5);
  itoa_cat(buff, count4,6);
  itoa_cat(buff, count3,5);
  itoa_cat(buff, TGL_out,5);
  LCD_print_str( buff );  // 数値で表示
*/

//exc_itoa を利用した場合 ---> nosprintf.c
//exc_itoa : 整数を１０進表記にして文字列の指定したアドレスを位置基準
//として整数で置き換える.
//LCD に合わせた例 32 文字出力
strcpy(buff,"                                "); //枠を作る.
//5 番目の文字を基準にして最後の桁から置き換える -> -123
exc_itoa(&buff[4],ad2_result);
//10 番目の文字を基準として後ろから置き換える->1111
exc_itoa(&buff[9],ad1_result);
//以下同様
exc_itoa(&buff[14],ad0_result);
exc_itoa(&buff[20],count4);
exc_itoa(&buff[25],count3);
exc_itoa(&buff[30],TGL_out);
//  buff -> " -123 1111 2222  3333 4444 5555 "になります.
```

警告	sprintf はメモリを大量消費するので使ってはいけません．数字を文字列に変換するときには nosprintf.c 内の関数を用いてください．

nosprintf.c 内の関数は変換したい数字が整数か小数かによって使い分けてください．

- 整数のとき
  (10 進数として表示したい場合) itoa, itoa_cat
  (16 進数として表示したい場合) htoa, htoa_cat
- 小数のとき
  ftoa, ftoa_cat

関数の後ろに cat がついている関数は既にある文字列の最後に引数の数字を連結する関数です．また，指定した位置の文字列を 10 進数の整数で置き換えたい場合には exc_itoa を用いてください．nosprintf.c 内の関数の具体的な使い方は nosprintf.c 内の関数のコメントアウトを参照してください．

また，LCD の出力に使用している P9 はタイマ入力ピンを兼ねているため，プログラムの暴走などでポート入出力が簡単に書き換えられないようにプロテクトレジスタ PRCR によって PD9 が保護されている．このプロテクトの解除は LCD_init() 内で行われています．

3. ポーリング

一定の間隔で繰り返しポートの値を読み込むことをポーリングという．入力ポートの電圧 (H/L) をプログラム中で変数の値 (0/1) として扱う[1]．前回の値を保持しておき今回の値と比較すれば，入力の変化 (エッジ) を検出することができる．次節でタイマ割り込みを用いた．ポーリングによるリアルタイム制御プログラムについて説明する．

4. 割り込み

割り込みとは，実行中の処理を一時的に中断して，他の処理（割り込み処理）を行うことをいう．また，割り込み処理が終了した後には元の実行していた処理を再開させることができる．割り込み処理を行う主な利点としては次のようなものがある．

- スイッチやセンサ入力などの外部入力によって行う処理を確実に実行できる．(外部割り込み)
- 定期的な処理を確実に実行できる．(タイマ割り込み)
- プログラム全体の処理効率が上がる．

4.1 タイマ割り込み

4.1.1 タイマ

本ボードには 16 ビットタイマ用が 8 本ある．この 8 本のタイマは機能によってタイマ A（5 本）とタイマ B（3 本）に分類できる．ここでタイマの機能について説明する．詳細については「ハードウェアマニュアル」を参照すること．

タイマの機能（タイマ A）

- タイマモード：内部カウントソース（CPU クロックやその分周）をカウントする．
- イベントカウンタモード：外部からのパルス，他のタイマのオーバーフロー，アンダーフローをカウントする．
- ワンショットタイマモード：カウント値が 0 になるまで，一度だけパルスを出力する．
- パルス幅変調（PWM）モード：任意の幅のパルスを連続して出力する．

タイマモードは次節で説明するタイマ割り込みを利用する際に使用する．イベントカウンタモードはエンコーダのような外部パルスをカウントする際に利用する．パルス幅変調（PWM）モードは D/A 変換で PWM 制御を用いるときなどに使用する．このようにタイマにはいろいろな機能があるので，用途に合った設定が必要である．

4.1.2 タイマ割り込みの設定

タイマによるカウント値を利用して，一定周期で割り込みを行うことをタイマ割り込みという．ここでは，タイマ割り込みを利用するにあたって必要な設定について説明する．詳細は「ハードウェアマニュアル」を参照．

[1]電圧とレジスタ値の関係は正論理である (つまりレジスタを 1 にすると電圧が出る)．ポート入力は負論理となっている

```
// プロトタイプ宣言
void ta0int( void ); // 割込み関数
#pragma INTERRUPT ta0int

//50msec 周期でカウント
#define cnt_ta0 31250-1 // タイマ A0 カウンタ値

void main( void ) {

// タイマ割り込み初期化
udf = 0x00; // ダウンカウント設定
ta0mr = 0x80; // タイマモード クロック：1/32
//CPU クロック（20MHz）の 32 分周は，20MHz/32=625KHz
ta0 = cnt_ta0; // タイマ値の初期化

ta0ic = 0x06; // 割り込みレベルの設定（7 が最も優先される）
tabsr = 0x01; // カウント開始
_asm( "\tFSET I"); // 割り込み許可

}
//割り込み関数の記述
void ta0int( void ) {
割り込み処理の内容
}
```

上記のように割り込みを使用する場合は，全ての割り込み関数に対してプロトタイプ宣言と割り込み関数の記述をする必要がある．**これは使用しない割り込み関数でも行う必要がある．（使用しない割り込み関数の中身は空で良い．）**これらの宣言などはプロジェクト作成時に必要だったスタートアップファイル sect30_ss2.inc 内で割り込み関数を定義してあるため，使用しない割り込み関数に関しても宣言しなければならない．もし宣言しなかった場合はエラーが発生するので注意が必要である．また，タイマ割り込み以外にも外部割り込みやシリアル送受信に使用する割り込み関数も存在するので注意する．

```
// プロトタイプ宣言
void ta0int( void ); // 割込み関数
#pragma INTERRUPT ta0int

//割り込み関数の記述
void ta0int( void ) {
//割り込み処理の内容
}
```

は必ず行わなければいけない．

課題：タイマ割り込み

1. フォルダ **kadai0101** 内にある割り込み用サンプルプログラム **kadai0101.c** を実行し，その動作を確認する．
2. タイマ割り込みを利用して，0.1sec 周期でフルカラー LED の緑が点滅するプログラムを作成する．なお，タイマの初期カウント値は変更しなくてもよい．また，LED のポートは，ポート割り当て表と，I/O ポートの設定方法（メカトロニクスラボの資料かハードウェアマニュアル（rjj09B0033_m16chm.pdf）P162 以後）を参照すること．

警告 割り込み関数内の処理が重すぎると割り込み処理からメインのループに戻ってこれない危険性が高くなるので，割り込み関数内の処理はできるだけ軽くする．

4.2 外部割り込み

外部割り込みとは入力ポートの電圧が変化したときに，実行中の処理を一旦中断し，特定の関数 (割り込みハンドラ) を実行する．割り込みハンドラの終了後，元の処理が再開される．例えばメカニカルスイッチなどの外部パルスの立ち上がり（または立ち下がり）を検出し，割り込み処理を実行することができる．外部割り込みが使用できるポートは INT0,1,3,4,5 の 5 本ある．それぞれ対応しているポートはポート割り当て表を参照すること．

4.2.1 外部割り込みの設定

外部割り込みの設定ではタイマ割り込みと同様に，プロトタイプ宣言とすべての割り込み関数の記述が必要である．また，立ち下がりか立ち上がりのどちらを検出するかと割り込み優先レベルを決定し，割り込みの許可を出す．

```
// プロトタイプ宣言
//使わない割り込みも記述する必要がある
void int3int( void ); //外部割込み 3 関数
#pragma INTERRUPT int3int


//外部割込み設定
int3ic = 0x06;  //bit2,1,0= 110 ：割り込み優先レベル 6
//bit4 = 0 ：立ち下りエッジ検出
_asm( "\tFSET I"); // 割り込み許可


void int3int( void ) { //割り込み処理の内容}
```

5. メカニカルスイッチ

マイクロスイッチなどの機械的なスイッチは，接触センサとして衝突検知や壁沿い走行に使用できる．

一般にメカニカルスイッチを電子回路に接続するときは Fig. 1 のように適当な抵抗でプルアップする．この場合，スイッチが ON のとき出力が 0 V，OFF のとき 5 V になる．

チャタリングを防ぐには Fig. 2 のようなチャタリング防止回路を用いる方法とプログラム上で防止する方法がある．**この回路はコンデンサで波形をなめらかにし，シュミットトリガで波形をきれいな方形波に成形している．**マシンにメカニカルスイッチを使うときは，スイッチ 1 個ごとに必ずチャタリング防止回路を付加することを推奨する．

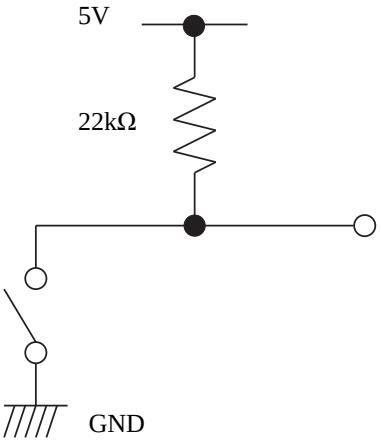

Fig. 1: 単純なスイッチ回路

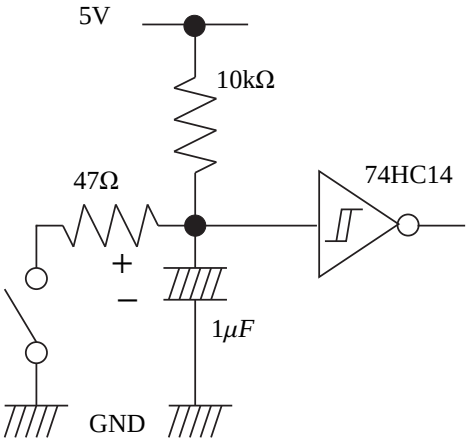

Fig. 2: チャタリング防止回路

回路試作の際の注意

ブレッドボードの使い方

ブレッドボードははんだ付けなしに回路が試作でき，素子も無駄にすることなく大変便利なものである．ブレッドボードでの試作は**正しく動作する回路を作成すること**を目的にする．適当に回路を作ることではないので注意．また，使用する前にどの穴同士が導通しているかはきちんと確認すること．

警告	付属のジャンパ線以外のもの（例:スズめっき線）を無理にさしこみ，穴をつまらせないように．

ブレッドボードの使用例を以下に示す．Fig. 3 はジャンパ線を無理に曲げたりして使わず，ちょうどよい長さを使っている．こうすると回路全体の見通しもよくなり，ミスが少なくなる．

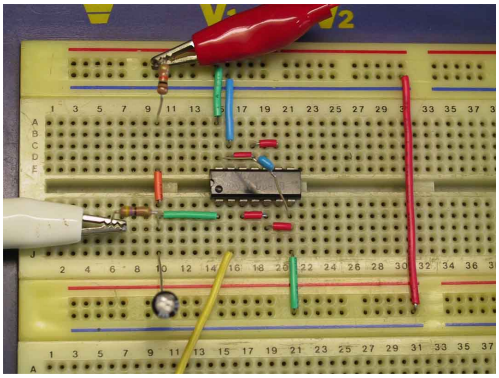

Fig. 3: よい例

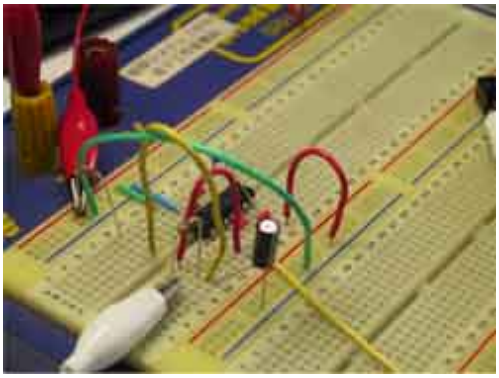

Fig. 4: 悪い例

悪い例に関しては，回路全体の見通しも悪く，後で最終的な回路の完成図を起こすときにも面倒になる．

⚠ 注意	ジャンパ線はきれいに使い，たくさんあるからといってずさんな管理しない．

ジャンパ線の数はスタッフが整理・確認している．

74HC14 使用時の留意点

Fig. 5: 74HC14 の外観

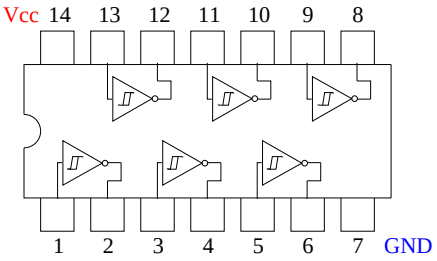

Fig. 6: 74HC14 のピン配置

Fig. 5 のようにおいたとき，ピン配置は Fig. 6 のようになります．また使わない入力は GND と接続しておくのがよい．また，積層セラミックコンデンサ 104（0.1μF）を Vcc と GND 間に接続し，動作を安定化させる．このような用途のコンデンサのことを「バイパスコンデンサ」という．

5.1　端子台への接続

MCU ボード上には端子台が設置されており，Fig. 7 のようなピン配置になっている．P1-5,6,7 は外部割り込み関数の INT3,4,5 にそれぞれ対応しており，P10-0〜7 は A/D 変換が使用できるポートに対応している．次のスイッチの課題ではこの端子台を用いるため，**接続は十分注意して行うこと．さらに，電解コンデンサ (1μF) には極性（＋ー）があるので接続を間違えないように注意すること．**積層セラミックコンデンサには極性は存在しない．

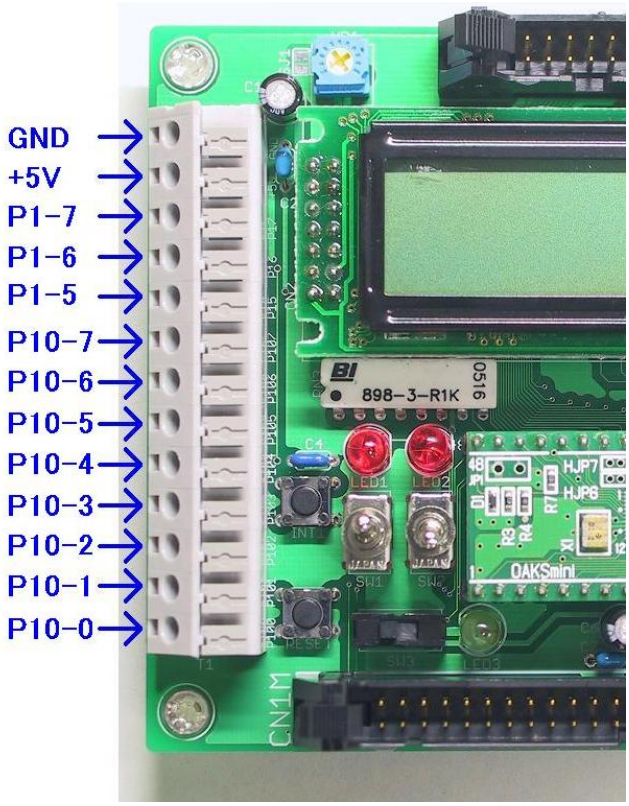

Fig. 7: MCU ボード（端子台）

課題：メカニカルスイッチ（チャタリング防止回路）

1. チャタリング防止回路 (Fig. 2) を作り，ON/OFF 時の波形を確認する．[a]
2. メカニカルスイッチによって外部割り込みが発生し，LCD ディスプレイにスイッチのカウントを表示するプログラムを作成する．

Hint P.8 のサンプルプログラムを参考にしてまずコントローラボードのプッシュスイッチ赤 (P1-5,int3int 関数を用いる) によって外部割り込みが発生するプログラムを作るとよい．その後，チャタリング防止回路を作成したら，MCU ボードの端子台の GND と+5V と P1-5 につなぎ，きちんと外部割り込みが入るかどうか試してみると良い．回路を MCU ボードにつなぐときはコントローラボードのコネクタを MCU ボードから外しておくこと．

[a] 参考のためにシュミットトリガに入る前の信号も確認すると良い．

注意 作成した回路をすぐにマイコンの入力ポートに接続しないで，まず導通チェックをする．今回は行わないが，オシロスコープで波形が正しく出力されていることを確認した後にマイコンの入力ポートに接続するとよい．