Tutorial4A

創造設計第二 TA：加藤 大地

2011 年 10 月 27 日

1. はじめに

Tutorial4A ではロータリエンコーダをセンサとしたモータの PID 制御を学ぶ．本試作検討で行う PID 制御の概要を Fig. 1 に示す．モータおよびロータリエンコーダのキットはこちらで配布するものを用いる．実際に取り組む内容は以下の通りである．

- モータの PID 制御
  - モータとエンコーダの動作確認
  - 目標軌道に対する PID 制御
- これまでの Tutorial で終わっていない課題に取り組む

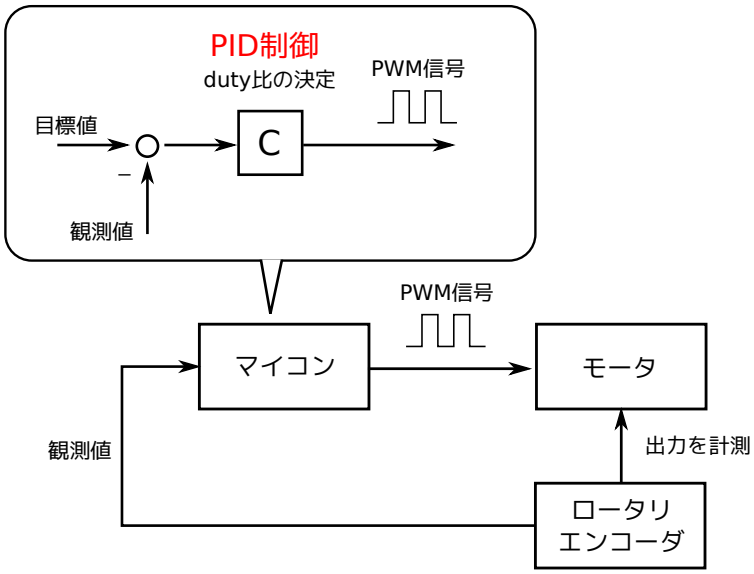

Fig. 1: Tutorial4A で扱う PID 制御概要

2. ロータリエンコーダ

本 Tutorial ではモーターの回転角を測定するのにロータリエンコーダを用いる. ロータリエンコーダは軸が一定角回転するとパルスを出力するため，このパルス数をマイコンのタイマでカウントすることでモーターの回転角を求めることができる．

今回用いるロータリエンコーダは Fig. 7 のような 2 相出力のものである. A 相，B 相の位相のずれた 2 つのパルスの立ち上がりおよび状態を計測することで回転方向と回転角を測定することができる．2 相式ロータリエンコーダの詳細はメカトロニクスラボ・マイコン編を参照する．また，エンコーダの詳しい仕様はデータシートで確認できる．

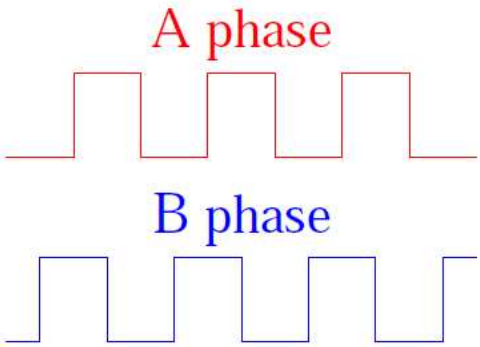

Fig. 2: 2 相出力信号

2.1 カウント方式

パルスのカウント方式には主に 1，2，4 逓倍カウントがあり，本 Tutorial では 4 逓倍カウントを用いる．4 逓倍カウントでは，A 相の立ち上がりエッジと立ち下がりエッジおよび B 相の立ち上がりエッジと立ち下がりエッジをカウントして回転角度を測定する．さらに各エッジを検知したときのもう一方の相の電位レベルを計測することで回転方向を判別する．本 Tutorial で用いる EC202A100A というエンコーダは 1 回転で 100 パルス出力するので，その分解能は 1 逓倍カウントの分解能 360/100=3.6 deg の 4 倍の 0.9 deg となる．Table 1 に 1，2，および 4 逓倍カウントとエッジの関係をまとめる．

Table 1: A 相・B 相と 1，2，4 逓倍カウントの関係

A 相	立ち上がり ↑		立ち下がり ↓		High(1)	Low(0)	High(1)	Low(0)
B 相	High(1)	Low(0)	High(1)	Low(0)	立ち上がり ↑		立ち下がり ↓	
回転方向	逆 (-)	正 (+)	正 (+)	逆 (-)	正 (+)	逆 (-)	逆 (-)	正 (+)
1 逓倍	○カウント		×なし		×なし		×なし	
2 逓倍	○カウント		○カウント		×なし		×なし	
4 逓倍	○カウント		○カウント		○カウント		○カウント	

2.2 マイコンとの接続

4 逓倍カウントを実装するためには，外部割り込みにより A 相 B 相それぞれに対して立ち上がり，立ち下がりエッジの検出が必須である．しかし，本 Tutorial で使用しているマイコンには立ち上がりおよび立ち下がりエッジ両方に対し割り込みを発生させるポートは存在しない．したがって A 相の立ち上がりエッジと立ち下がりエッジおよび B 相の立ち上がりエッジと立ち下がりエッジの検出を 4 つのポートで行う．本 Tutorial では Table 2 に示すポートでそれぞれのエッジを検出する．

Table 2: 4 逓倍カウントで用いるポート

A 相立ち上がり検出	P50/$\overline{\text{WKP0}}$
A 相立ち下がり検出	P51/$\overline{\text{WKP1}}$
B 相立ち上がり検出	P52/$\overline{\text{WKP2}}$
B 相立ち下がり検出	P53/$\overline{\text{WKP3}}$

接続の際には，エンコーダの A 相信号端子をマイコン上の P50/$\overline{\text{WKP0}}$ 端子および P51/$\overline{\text{WKP1}}$ 端子と繋ぎ，エンコーダの B 相信号端子をマイコン上の P50/$\overline{\text{WKP0}}$ 端子および P51/$\overline{\text{WKP1}}$ 端子と繋ぐ．ロータリエンコーダのピン配置を Fig. 3 に示す．また，マイコン側のピン配置を Fig. 4 に示す．A 相，B 相の信号がそれぞれ Table 2 に対応するポートへと繋がっていることを確認すること．

Fig. 3: ロータリエンコーダのピン配置

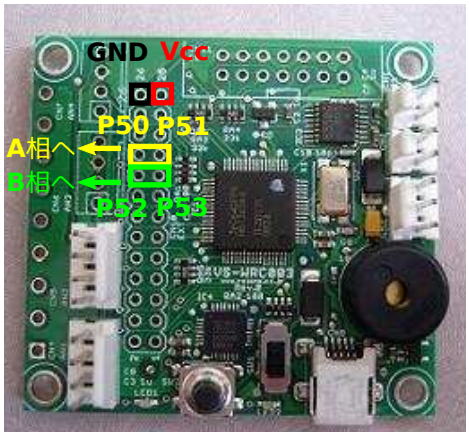

Fig. 4: マイコンのピン配置

2.3 割り込みによるパルスのカウント

前節のようにポートを割り当てた場合，エッジが検出できるよう各ポートを次のように設定する．

```
/*エンコーダ用の外部割り込みの設定*/
/*P50 で立ち上がりエッジを検出*/
IEGR2.BIT.WPEG0 = 1; //WKP0：立ち上がりエッジ検出（立ち下がりエッジ検出であれば 0）
IO.PMR5.BIT.WKP0 = 1; //P50 を WKP0 入力端子に設定
IWPR.BIT.IWPF0 = 0; //WKP0 の割り込み要求フラグをクリア
IENR1.BIT.IENWP = 1; //WKP0～WKP5 の割り込み許可
```

上の例では P50 の設定を行っているが，P51~P53 でも同じように設定を行う．また，割り込みによる処理を INT WKP(void) に記述する必要がある．なお本 Tutorial では，上記のポートの設定は encorder.c の EncInit()，割り込み処理については intprg.c に記述してある．また，ロータリーエンコーダによるエッジ検知に関しては，encoder.c の Four Count() に次のように記述されている．

```
/*gEncCnt：エッジ検出のカウント*/
/*A 相の立ち上がり処理*/
//立ち上がりエッジを検出すれば"1"
if( IWPR.BIT.IWPF0 == 1 ){
   IWPR.BIT.IWPF0 = 0}; //割り込みフラグクリア
   //B 相の状態が"Low"なら正転
   if( IO.PDR5.BIT.B2 == 0 ){
      /*正転時の処理*/
      gEncCnt++;
   }else{
      /*逆転時の処理*/
      gEncCnt--;
   }//if( IO.PDR5.BIT.B2 )}
}//if( IWPR.BIT.IWPF0 )

/*以下 A 相の立ち下がり，B 相の立ち上がりおよび B 相の立ち下がりに関しての記述*/
```

このようにして，gEncCnt の増減を見れば正転，逆転の判断ができる．さらに前述した通り，本 Tutorial で用意したエンコーダでは 1 回転で 100 パルス出力し，A 相・B 相の立ち上がり・立ち下がりエッジを検出する．そのため，1 周正転させた場合には gEncCnt は 400 増加することになる．これを用いれば，回転角を求めたり，回転の速度を求めたりすることができる．例えば 1 sec で gEncCnt の値が 100 増加していれば，その 1 sec での平均速度は $2\pi*(100/400)=\pi/2$

rad/s と算出できる．

3. PID 制御

PID 制御とは制御手法の 1 つであり，産業界では主な制御手法として広く用いられている．ここで”PID” とは，

- 比例：Proportional
- 積分：Integral
- 微分：Derivative

それぞれの頭文字をとったものであり，その名の通りこれら 3 つの動作を組み合わせた制御手法である．

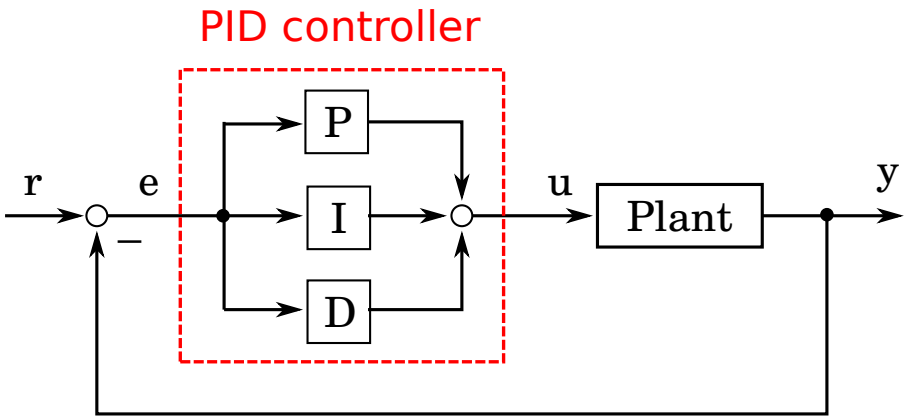

Fig. 5: PID 制御のブロック線図

具体的には Fig. 5 のように目標値 r と出力 y との偏差 e，その積分，そしてその微分に係数を掛けて足したものを制御入力とする. 式で表すと以下のようになる．

$$u = K_p + K_I \int edt + K_D \dot{e} \tag{1}$$

それぞれの係数 K_p，K_I，K_D は比例，積分，微分動作に対するゲインであり，これらの値を調節することで所望の出力を実現する．以下各節では，比例，積分，微分の項がそれぞれどのように出力に対し影響を与えるのかを復習し，次章で課題を通じマイコンを用いて実装する．

3.1 P 制御

P 制御は最も単純な制御方法である．入力は次式のように表される．

$$u = K_P e \tag{2}$$

すなわち，“出力が目標値からずれていれば入力を大きくして，目標値に近づいたら入力を小さくする” 制御手法である．しかし，これだけでは必ずしも所望の動作は得られない可能性がある．例えば，

- 定常偏差が残る．
- 収束を早くするためにゲインを大きくすると振動的になる．

という問題がある．そこで，P 制御を基本として積分動作や微分動作を加えて所望の動作の実現を目指す．

3.2 PI 制御

P 制御に積分動作を加えたものである. 入力は次式のように表される．

$$u = K_P e + K_I \int edt \tag{3}$$

P 制御だけでは，今回扱うモータのように摩擦があるような場合には偏差が小さくなり入力が小さくなると入力が摩擦力より小さくなってしまう．そのため，モータが動かなくなり偏差が残ってしまう（定常偏差）．このとき，偏差を積分することで時間の経過と共に徐々に入力が大きくなり偏差をなくすことができる．このように積分動作には定常偏差を取り除く効果がある．

3.3 PD 制御

P 制御に微分動作を加えたものである．入力は次式のように表される．

$$u = K_P e + K_D \dot{e} \tag{4}$$

P 制御に速度の偏差の項が加わっているため，より速く動かそうとする．その結果，応答性が高まり外乱に強くなる．ただし，微分項のゲインを大きくしすぎるとオーバーシュートが発生しやすくなるので注意が必要である．

3.4 PID 制御

P 制御に積分動作，微分動作を加えたものである．積分動作，微分動作両方の特徴を活かすことができるが，それぞれの項に対するゲインのチューニングをしっかりと行うことが必須である．これらパラメータの設計方法としては限界感度法やステップ応答法などが主な方法として広く使われている．

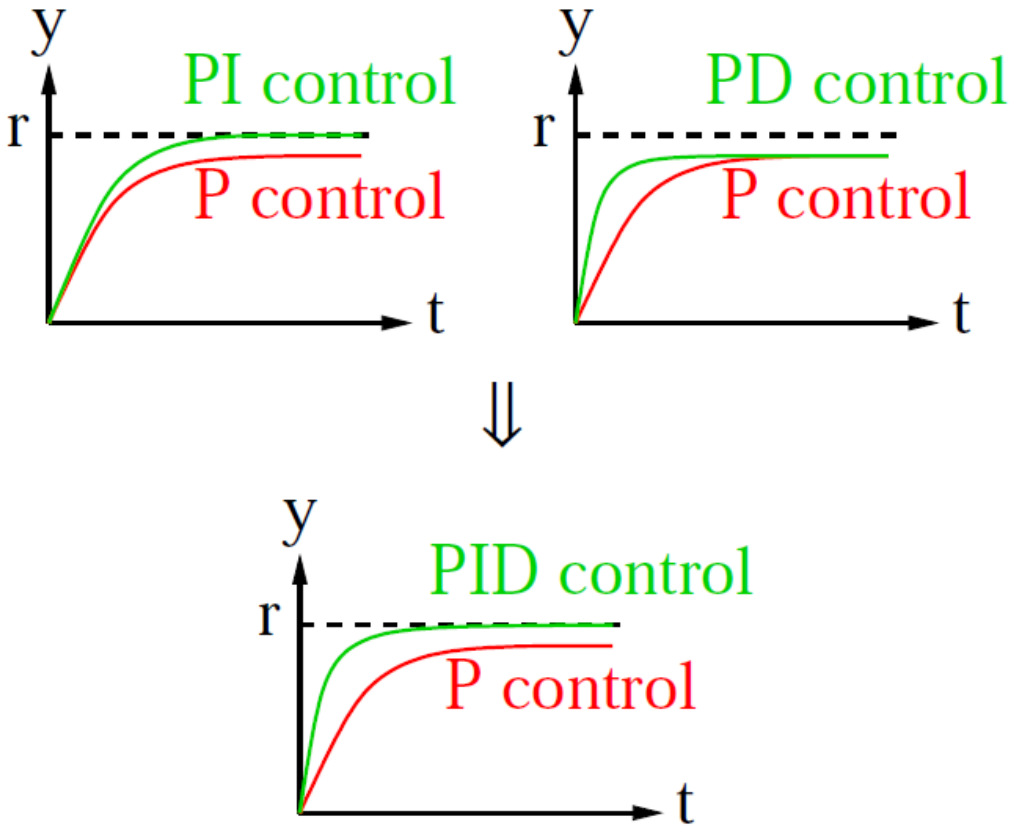

Fig. 6: PID 制御の特徴

3.5 目標軌道

PID 制御はある目標値に追従させるための制御方法である．原点に零点を持たない安定なシステムに対しては，PID 制御によって目標値に定常偏差なく追従させることが可能である．ここで，目標値をある値からある値へ切り替えること場合を考える．このときステップ状に切り替えると，定常応答を考えればやはり偏差なく追従できる．しかし過渡応答を考えると，収束性や応答性は制御器の各ゲインをチューニングすることである程度調整することはできるが，その時間応答を所望の軌道にすることは困難である．例えば，車両の位置や速度を目標値に追従させる場合，ステップ入力では急な加減速を引き起こしてしまうことになるが，これを（ゲインチューニングによって）緩やかな加減速にすることは難しい．そこでここでは，目標値間をある目標軌道で繋ぐことを考える．これにより目標値間の遷移を所望の軌道に近づけることができると考えられる．例えば速度の目標軌道 $r = \sin t$ などと設定し，PID 制御により速度をこれに追従させれば，滑らかに加減速を繰り返す動きが実現できると考えられる．また，状態遷移に伴い目標軌道を変化させれば，A 区間では加速，B 区間では減速といった動きが可能になる．

4. 課題

課題 1：モータとエンコーダの動作確認

1. 付録を参考にして，SS2shisaku04.c 内の課題 4.1.1 の部分に “Hello, World!” を Hterm に表示させる処理を記述せよ．
2. SS2shisaku04.c 内の課題 4.1.2 内の?の部分を変更することで，3.0 sec ごとに正転，逆転を繰り返すプログラムを作成し，モータが駆動することを確認せよ．
3. SS2shisaku04.c 内の課題 4.1.3 のコメントを外し，0.25 sec ごとに gEncCnt の値を Hterm または Excel のシートに表示させよ．duty の値は適当に定める（例えば duty=50000)．時刻，回転方向（dir），カウント数（gEncCnt）をタブ区切りで送信すると良い．なお，Excel シートは SS2shisaku04 フォルダ内の serial_com フォルダ内にある serial_com.xls を用いること．

課題 2：目標軌道に対する PID 制御

目標位置に到達するまで滑らかな加減速を行う目標軌道を生成し，出力がその軌道に追従するよう PID 制御器の設計を行う．

1. 課題 4.1.1 および 4.1.2 の部分をコメントアウトする．
2. Fig. 7 のように三次関数の極小値から極大値までの曲線を用いた目標位置軌道を考える．目標位置 x_d，目標到達時間を T_d と置いて条件を満たす三次関数を求め，時刻 t における目標値を求める関数 Caluc target x に求めた三次関数を反映させよ．なおデフォルトでは，$x_d = 600$，$T_d = 15$ となっている．
   （a）SS2shisaku04.c 内の課題 4.2.1 (1/2) のコメントアウトを外す．
   （b）PID.c 内の課題 4.2.1 (2/2) の?の部分を変更する．
3. PID 制御に必要な e，e_i，e_d（偏差，偏差の和，偏差の差）を求め，制御入力を算出するプログラムを完成させよ．
   （a）PID.c 内の課題 4.2.2 (1/2) の?の部分を変更する．
   （b）SS2shisaku04.c 内の課題 4.2.2 (2/2) のコメントアウトを外す．各ゲインの値は適宜調整して良い．
4. 設計した入力でモータを制御し，目標軌跡と観測軌跡を比較せよ．
   （a）SS2shisaku04.c 内の課題 4.2.3 のコメントアウトを外し，目標値データを Hterm に表示するようにする．
   （b）付録を参考に，観測出力と目標値出力のグラフを Excel シートに表示させる．
5. SS2shisaku04.c 内の課題 4.2.2 (2/2) の各ゲインの値を変更することで，ゲインのチューニングをせよ．

課題 3：これまでの Tutorial で終わっていない課題

1. これまでの Tutorial の課題の中で，終わっていないものがあればそれを終わらせる．
2. 「センサ入力 → 状態遷移 → 状態に対応した出力」の一連の流れを確認する．その後，実際にレールの上を走らせてみる．

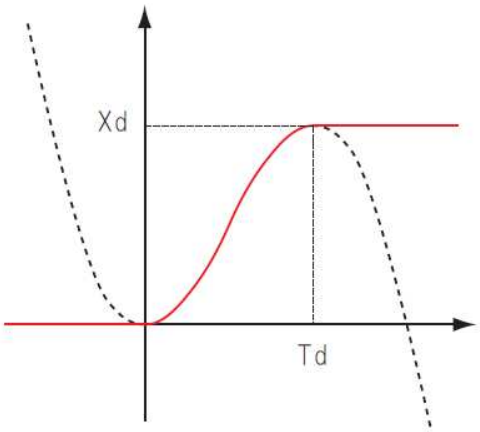

Fig. 7: 滑らかな加減速を行う目標軌道

付録 A　シリアル通信を用いたデバッグの方法

A.1　はじめに

効率良く適切なシステム設計を行うためにはデバッグを行うことが効果的である. そこで, 本付録では Vstone 社製マイコンボード（VS-WRC003）に関して，USB によるシリアル通信を用いたデバッグ方法を紹介する.

注意	以下で紹介するデバッグ方法は ROM に書き込まないと使用できないので注意すること.

A.2　準備

- 電源
  デバッグをする際には USB から電源供給を行う. そのため, 電池からの電源供給ケーブルは外しておく.

注意	USB からの供給電圧は 5 V, 電流は 500 mA である．これらの値を超えると USB ポートが壊れてしまったり, 最悪の場合 PC が破壊されてしまうので十分に注意すること.

- ソースコード
  ダウンロードしたフォルダ内の”sci.c, sci.h, nosprintf.c, nosprintf.h” を開発中のソースフォルダ内にコピーし, 利用する. なお, コピーしたファイルは HEW で「ファイルの追加」を行うことで利用可能になる.

A.3　ソースの内容

- sci.c
  sci.c にはシリアル通信に関する内容が記述されている. 基本的には次の二つの関数を用いる.
  1. sciinit()
     シリアル通信のための初期設定を行う. シリアル通信によるデバッグを行う際は必ずこの関数を記述する.
  2. sciputs(char *str)
     シリアル通信によって str を PC へ送信する.
- nosprintf.c
  nosprintf.c には数値を変換し, sciputs(char *str) によって Hterm に表示可能な形にする関数が記述されている. 詳しい内容は nosprintf.h を参照すること.

A.4　使用方法

以下に使用例を示す. これを適宜応用しデバッグを行う. なお, HEW, FDT, Hterm はあらかじめ立ち上げておく.

1. プログラムの作成
   ボタンを押すと Hterm に次の内容を表示させるプログラムを作成する.

```
Hello, World!
1234
```

   まず，”sci.h, nosprintf.h” を include し, 次に以下の内容を main 関数内に記述する.

```
volatile char buf[32];
/*シリアル通信のための初期化*/
sciinit();

/*ボタンが押されるまで待機．ボタンが押されたら文字列表示処理を開始する．*/
while (IO.PDR5.BIT.B5);

/*"Hello, World!"の表示*/
sciputs("Hello, World!\r\n");

/*"1234"の表示*/
itoa(1234, buf);
sciputs(buf);

/*改行をいれる*/
sciputs("\r\n");
```

2. ROM への書き込み
   書き込んだら 1 度マイコンの電源を切って, 再度付ける. このとき, USB コネクタは接続したままにしておく.
3. Hterm の接続
   Hterm で「通信」→「接続」を行う. 接続の状態になったらマイコンのボタンを押す. すると「Hello, World・・・」が表示される.
4. Hterm の切断
   表示されたことを確認したら Hterm で「通信」→「切断」を行い, マイコンの電源を落とす.
5. グラフの作成他
   Hterm に表示された文字列をコピーし, エクセルのシート等に張り付ければ数値の処理やグラフの作成ができる.

ROM への書き込み回数には限度があるので注意すること.

A.5 Excel を用いた処理

前章では, シリアル通信を用いてマイコンから PC へ情報を送信し, Hterm 上で表示する方法を紹介した. これによりマイコンが持つ情報を出力できるようになったわけだが, Hterm 上ではその情報に対し処理を加えることができない. すなわち, 得られた情報を効率良く解析できないということである. 一方で, 制御工学では得られた情報を解析し, システムのモデルを同定したり, コントローラーを設計したりする機会が多い. 例えば, PID 制御ではステップ応答の出力結果を解析し, 各ゲイン K_p, K_I, K_D を求め, コントローラーを設計する方法がある. したがって, 得られた情報を Hterm 上で表示するだけではなく, 得られた情報を効率良く解析できる環境が必要である. そこで, Microsoft が提供する Excel のシート上にマイコンから送られた情報を表示する環境を用意した. 本章ではその環境の使用方法を紹介する.

A.5.1 使用方法

以下に使用手順を示す. HEW, FDT は事前に立ちあげておくこと.

1. プログラムの作成
   Hterm を利用したときと同様.
2. ROM への書き込み
   Hterm を利用したときと同様.
3. “serial_com.xls” を開く
   このファイルではマクロが使用されている. そのため, ファイルを開いたときは必ず「マクロを有効」に設定する.
4. 「実験手順」のシートのフローに従って作業を行う
   VBA のソースコードを変更しない限り, ファイルを開くと同時に「実験手順」のシートがアクティブになる. このシートには Fig. A.1 のように実験手順が記述されている. その工程は Hterm を利用したときとほぼ同じであ

る. まずマイコンと PC が USB コネクタで繋がれていることを確認し, マイコンの電源をいれる. 次に「通信スタート」のボタンを押し, 通信を開始する. 「通信開始」のメッセージが表示されれば, マイコンから送られた情報を PC で受信する準備が整った状態なので,「OK」を押し, マイコンでの処理を開始する. 前章で紹介した通り,「マイコンのボタンを押すと処理を開始する」ように設定すると良い. 十分に処理が行えたら,「通信ストップ」のボタンを押してからマイコンの電源を切る.

5. データの解析をする

   ここまでの手順が終えたら「データ*」というシートが作成されているので, それをアクティブにする. このシートにマイコンからタブ区切りで送信したデータが各列に分かれ, 表示される.

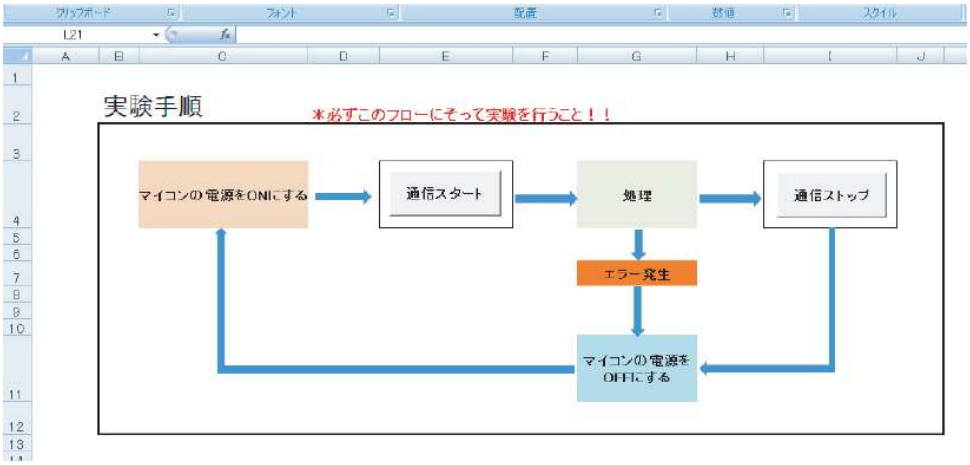

Fig. A.1: 実験手順