PC を用いたステッピングモータの制御とその特性に関する調査

情報学部コンピュータ科学科 3 年 野原健汰 (学籍番号:101730279)
メールアドレス:nohara.kenta@e.mbox.nagoya-u.ac.jp

2019 年 4 月 18 日実施

1　概要

本実験では、PC/AT 互換機とステッピングモータを接続する電子回路を組み、パラレルポートを介してステッピングモータの制御を行った。2 では、実験全体の目的とレポートの構成、3 では実験方法や実験結果・考察、4 では本実験のまとめを記し、5、6、7 ではステッピングモータの特性について調査した。

2　はじめに

2.1　実験の目的

本レポートでは、PC/AT 互換機とアクチュエータの電子回路の組み立てと、PC/AT 互換機を用いたアクチュエータの制御について実験的に検証し、工業・産業などあらゆる分野で用いられておりオープンループ制御可能なモータであるステッピングモータの特性を調査する。オープンループ制御とは、アクチュエータに動作信号を送るのみでアクチュエータからのフィードバックを必要としない制御方法である。フィードバック回路が不要であるため、他の制御方法に比べて制御が簡単である。
そこで本レポートの前半では、PC/AT 互換機とステッピングモータを接続する電子回路を組み、PC/AT 互換機からステッピングモータを制御することで、動作原理、制御方法を確認する (実験 1)。そして本レポートの後半では、ステッピングモータの立ち上がりの改善点に関する考察・調査 (調査課題 2-1) と、ステッピングモータの特性に関する調査 (調査課題 2-2)、ステッピングモータの台形速度制御方法についての実験検証 (調査課題 2-3) を行い、ステッピングモータの特性と制御についてまとめる。

3　[実験 1] ステッピングモータの運転

3.1　目的・概要

本実験では、PC/AT 互換機とステッピングモータをパラレルポートを介して接続し、ユニポーラ 1 相、2 相、1-2 相励磁でモータを制御 (正転、逆転、停止、速度制御) することにより、PC 制御の基礎と原理、その特性を確認し理解する。

3.2　実験対象

今回の実験で用いた機器は以下の通りである。

- ステッピングモータ (28BYJ-48 5V DC 81146122)
  28BYJ-48 は永久磁石を用いた PM 型 (調査課題 2-2 で後述する)4 相ステッピングモータである。中のコイルはバイファイラ巻き (逆方向に 2 つの線で巻く巻き方) で巻かれており、電流を流す線を切り替えることで磁極を切り替えることができる。4 相ステッピングモータであるため電流の流し方は 1 相励磁、2 相励磁、1-2 相励磁の 3 種類があり 1 相励磁は 1 相ずつ順に電流を流し、2 相励磁は 2 相ずつ、1-2 相励磁は 1 相→ 2 相を繰り返しながら電流を流してロータを回転させることができる。1 相励磁、2 相励磁の場合ロータは 32 ステップで 1 回転するためステップ角 (ステップ 1 回の回転角度) は 11.25°、 1-2 相励磁の場合は倍の 64 ステップで 1 回転するためステップ角は 5.625°である。
- 直流安定化電源 (KENWOOD PW18-1.8AQ)
  大きな電流を流すためステッピングモータを駆動させる電力を、コンピュータとは別の場所から得る必要があり、今回の実験では外部電源として用いた。

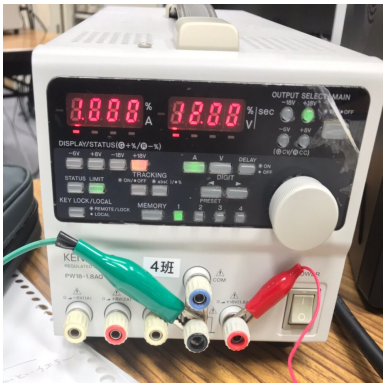

図 1　直流安定化電源

- ステッピングモータドライバーボード (トランジスタアレイ：TOSHIBA ULN2003APG)
  ステッピングモータは内部のコイルに電流を流し磁力を発生させることでロータを回転させるが、その回転には一定の大きさの電流が必要であり、コンピュータの出力では電流が不足であるためトランジスタなどにより電流の増幅が必要となる。今回の実験では電流の増幅と LED ランプの点灯によりピンの出力を確認するために用いた。

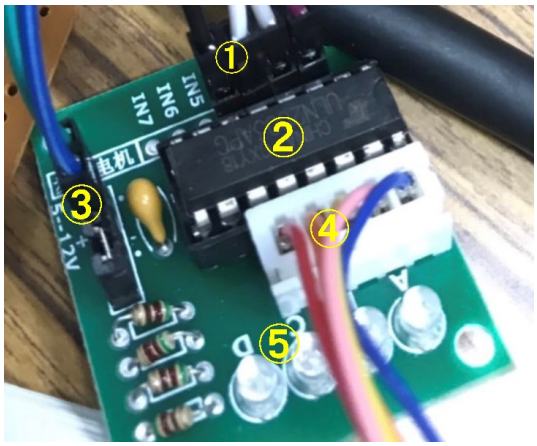

図 2　ステッピングモータドライバーボード

図 2 の 1 はパラレルポートからの入力、2 はトランジスタアレイ、3 は電源、4 はモーターへの出力、5 は出力中のピンを確認する LED である。

- ブレッドボード
- PC/AT 互換機
- 制御実験用ケーブル
- テスター

3.3　実験方法

3.3.1　パラレルポートとステッピングモータ駆動回路の接続

PC/AT 互換機のパラレルポートに制御実験用ケーブルを繋いだ。(図 3 参照)

図 3　PC/AT 互換機と制御実験用ケーブルの接続

次に、制御実験用ケーブルとブレッドボードを接続した。(図 4 参照)
制御実験用ケーブルは 2、3、4、5 ピンに繋ぎ、2 ピンから出るケーブルをブレッドボードの 13 に、4 ピンから出るケーブルをブレッドボードの 11、5 ピンから出るケーブルをブレッドボードの 9、6 ピンから出るケー

ブルをブレッドボードの 7 にそれぞれ接続した。

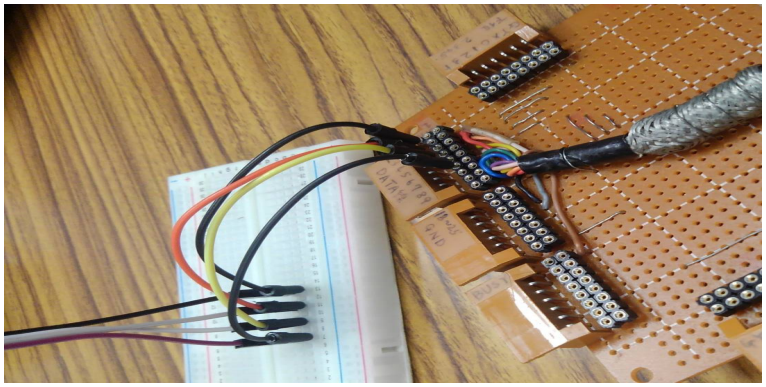

図 4　制御実験用ケーブルとブレッドボードの接続

3.3.2　直流安定化電源とブレッドボード、ステッピングモータの接続

直流安定化電源とステッピングモータドライバーボードを図 2 の 3 に±端子を間違えないように接続した。直流安定化電源は、-を GND に、＋ を +18V と書かれている端子に繋いだ。図 2 の 1 にはブレッドボードから出るジャンパー線を、4 にはステッピングモータを接続した。

これらの接続を行い、以下の図 5 のような回路図を作成した後 C 言語によるプログラムでモータ制御を行った。

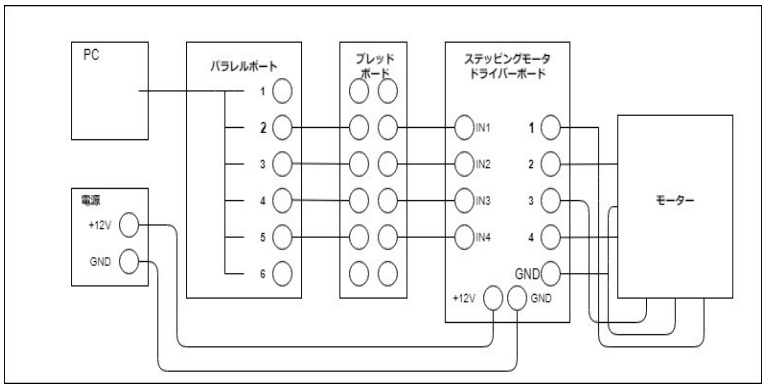

図 5　全体の回路図

3.3.3　1 相、2 相、1-2 相励磁でのモータ制御

以下の制御はすべて C 言語によるプログラムを用いた。

まず初めに以下は 1 相、2 相、1-2 相励磁の正転、逆転についてのソースコードである。コマンドライン引数によって値を入れるとそれぞれに送るパルス信号の周波数または正転・逆転を変えることができるようにした。この際、モータにパルス信号がうまく送られない場合にはテスターを用いてジャンパー線の不具合がないかど

うか確認した。

- 1 相励磁の正転、逆転

```
#include <stdio.h>
#include <unistd.h>
#include <sys/io.h>
#include <stdlib.h>

#define PPA_0 ((unsigned long)(0x378)) // パラレルポート出力アドレス
#define PBS 2                          // パラレルポート byte size
#define TRUE 1

int main(int argc, char* argv[]) {
  int f = atoi(argv[1]); //パルス信号の周波数 (Hz)
  int reverse = atoi(argv[2]); //正転逆転の切り替え
  int i;
  double T = 1.0e6/f; //usleep の引数、単位はマイクロ秒
  printf("%lf\n",T);
  int bi[8]; //パラレルポートのピンに対応するビットの配列
  for (int j = 0; j < 8; j++){
    bi[j] = 1<<j;
  }

  if ((i = ioperm(PPA_0, PBS, TRUE)) != 0) { //動作チェック
    printf("1\n");
    exit(-1);
  }
  if (reverse == 0){ //正転
    while (1) { //1 相励磁
      outb(bi[0],PPA_0); //1 番ピン
      usleep(T);
      outb(bi[1],PPA_0); //2 番ピン
      usleep(T);
      outb(bi[2],PPA_0); //3 番ピン
      usleep(T);
      outb(bi[3],PPA_0); //4 番ピン
      usleep(T);
    }
  }
```

```
  else { //逆転
     while (1) { //1 相励磁
      outb(bi[3],PPA_0); //4 番ピン
      usleep(T);
      outb(bi[2],PPA_0); //3 番ピン
      usleep(T);
      outb(bi[1],PPA_0); //2 番ピン
      usleep(T);
      outb(bi[0],PPA_0); //1 番ピン
      usleep(T);
    }
  }
}
```

- 2 相励磁の正転、逆転

```
#include <stdio.h>
#include <unistd.h>
#include <sys/io.h>
#include <stdlib.h>

#define PPA_0 ((unsigned long)(0x378)) // パラレルポート出力アドレス
#define PBS 2                          // パラレルポート byte size
#define TRUE 1

int main(int argc, char* argv[]) {
  int f = atoi(argv[1]); //パルス信号の周波数 (Hz)
  int reverse = atoi(argv[2]); //正転逆転の切り替え
  int i;
  double T = 1.0e6/f; //usleep の引数、単位はマイクロ秒
  printf("%lf\n",T);
  int bi[8]; //パラレルポートのピンに対応するビットの配列
  for (int j = 0; j < 8; j++){
    bi[j] = 1<<j;
  }

  if ((i = ioperm(PPA_0, PBS, TRUE)) != 0) { //動作チェック
    printf("1\n");
    exit(-1);
  }
```

```
  if (reverse == 0){ //正転
    while (1) { //2 相励磁
      outb(bi[0]+bi[1],PPA_0); //1,2 番ピン
      usleep(T);
      outb(bi[1]+bi[2],PPA_0); //2,3 番ピン
      usleep(T);
      outb(bi[2]+bi[3],PPA_0); //3,4 番ピン
      usleep(T);
      outb(bi[3]+bi[0],PPA_0); //4,1 番ピン
      usleep(T);
    }
  }
  else { //逆転
     while (1) { //2 相励磁
      outb(bi[3]+bi[0],PPA_0); //4,1 番ピン
      usleep(T);
      outb(bi[2]+bi[3],PPA_0); //3,4 番ピン
      usleep(T);
      outb(bi[1]+bi[2],PPA_0); //2,3 番ピン
      usleep(T);
      outb(bi[0]+bi[1],PPA_0); //1,2 番ピン
      usleep(T);
    }
  }
}
```

- 1-2 相励磁の正転、逆転

```
#include <stdio.h>
#include <unistd.h>
#include <sys/io.h>
#include <stdlib.h>

#define PPA_0 ((unsigned long)(0x378)) // パラレルポート出力アドレス
#define PBS 2                          // パラレルポート byte size
#define TRUE 1

int main(int argc, char* argv[]) {
  int f = atoi(argv[1]); //パルス信号の周波数 (Hz)
  int reverse = atoi(argv[2]); //正転逆転の切り替え
```

```
int i;
double T = 1.0e6/f; //usleep の引数、単位はマイクロ秒
printf("%lf\n",T);
int bi[8]; //パラレルポートのピンに対応するビットの配列
for (int j = 0; j < 8; j++){
  bi[j] = 1<<j;
}

if ((i = ioperm(PPA_0, PBS, TRUE)) != 0) { //動作チェック
  printf("1\n");
  exit(-1);
}
if (reverse == 0){ //正転
  while (1) { //1-2 相励磁
    outb(bi[0]+bi[1],PPA_0); //1,2 番ピン
    usleep(T);
    outb(bi[1],PPA_0); //2 番ピン
    usleep(T);
    outb(bi[1]+bi[2],PPA_0); //2,3 番ピン
    usleep(T);
    outb(bi[2],PPA_0); //3 番ピン
    usleep(T);
    outb(bi[2]+bi[3],PPA_0); //3,4 番ピン
    usleep(T);
    outb(bi[3],PPA_0); //4 番ピン
    usleep(T);
    outb(bi[3]+bi[0],PPA_0); //4,1 番ピン
    usleep(T);
    outb(bi[0],PPA_0); //1 番ピン
    usleep(T);
  }
}
else { //逆転
   while (1) { //1-2 相励磁
    outb(bi[3]+bi[0],PPA_0); //4,1 番ピン
    usleep(T);
    outb(bi[3],PPA_0); //4 番ピン
    usleep(T);
    outb(bi[2]+bi[3],PPA_0); //3,4 番ピン
    usleep(T);
```

```
      outb(bi[2],PPA_0); //3 番ピン
      usleep(T);
      outb(bi[1]+bi[2],PPA_0); //2,3 番ピン
      usleep(T);
      outb(bi[1],PPA_0); //2 番ピン
      usleep(T);
      outb(bi[0]+bi[1],PPA_0); //1,2 番ピン
      usleep(T);
      outb(bi[0],PPA_0); //1 番ピン
      usleep(T);
    }
  }
}
```

- 実行コマンド
  sudo ./ファイル名 [周波数の値 (任意)] [0 かその他の整数]

次に、モータの速度制御、停止についてのソースコードであるが、これは調査課題 2-3 で台形速度制御として行ったため、後述する。

最後に、1 相、2 相、1-2 相励磁の半周の回転にかかる時間を測定した。回転子にテープでシャーペンの芯をつけ (図 6 参照)、モータを固定して上から動画をとり目視によって 10 回、半周にかかる時間を測定した。

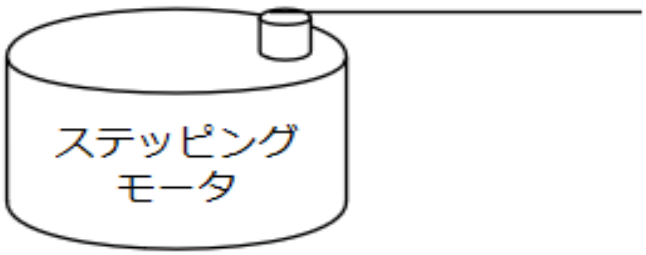

図 6　半周にかかる時間の測定

3.4　実験結果

先述したプログラムによって 1 相、2 相、1-2 相励磁でモータが回転 (正転、逆転) していることが確認できた。

以下は実行した際の直流安定化電源の示す値である。

図 7　1 相励磁の示す値

図 8　2 相励磁の示す値

図 9　1-2 相励磁の示す値

以下は 1 相、2 相、1-2 相励磁の半周にかかる時間の統計である。

```
標本数   : 10
標本     : 10.219 10.258 10.358 10.531 10.252 10.216 10.162 10.287 10.013 10.112
平均     : 10.240800000000002
不偏分散 : 0.019587733333333385
標準偏差 : 0.13995618361949352
誤差     : 0.0442580312862348
測定値   : 10.240800000000002±0.0442580312862348
```

図 10　1 相励磁

```
標本数   : 10
標本     : 10.326 10.449 10.151 10.04 10.037 10.591 10.23 10.134 10.042 10.198
平均     : 10.219800000000001
不偏分散 : 0.03462573333333328
標準偏差 : 0.1860799111493051
誤差     : 0.05884363460335645
測定値   : 10.219800000000001±0.05884363460335645
```

図 11　2 相励磁

```
標本数   : 10
標本     : 20.523 20.432 20.738 20.657 20.415 20.528 20.626 20.653 20.426 20.212
平均     : 20.520999999999994
不偏分散 : 0.02419444444444451
標準偏差 : 0.15554563460426818
誤差     : 0.04918784854457908
測定値   : 20.520999999999994±0.04918784854457908
```

図 12　1-2 相励磁

3.5　考察

4　まとめ

5　[調査課題 2-1]

駆動電圧が 3.5V、相電流が 1 相あたり 1.75A である、巻線抵抗 2 Ω のユニポーラステッピングモータの電流の立ち上がりを良くするために外付け抵抗を用いた場合について考察する。まず、ユニポーラ駆動のモータ 1 相分を取り出した等価回路は図 13 のようになる。ここで R は抵抗値、L はコイルのインダクタンス、i は電流を表す。

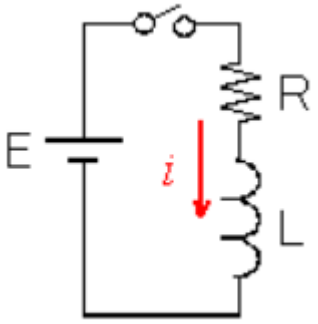

図 13　ユニポーラ駆動 1 相分の回路 (文献 [3] より引用)

この回路でスイッチを ON にしたあと流れる電流は、$I(t) = \frac{V}{R}\{1 - exp(-\frac{R}{L}t)\}$ である。

6　[調査課題 2-2]

ステッピングモータは電流を流すだけで回転する DC モータとは異なって電流を流すだけでは回転せず、コントローラによってパルス信号 (矩形状の電気信号) を送る必要がある。1 つのパルス信号に対して回転する角度 (ステップ角度) がモータの磁極数によって決められており、パルス信号の周波数変化に比例してモータの回転速度を変えることができる。さらに、停止精度は± 0.05°以内で誤差が累積しない。つまり、ステッピングモータはパルス信号によって簡単に正確な制御 (オープンループ制御) が可能であるため、エアコンやデジカメなど多くの製品に利用されている。ステッピングモータは励磁によって回転しており、ロータを囲むコイルに電流を流したり止めたりすることによってロータ内の磁石を回転させている。また、ステッピングモータは接続方法、ロータの種類などによって分類される。
まず、接続方法であるが、ユニポーラ駆動とバイポーラ駆動の 2 種類がある。

- ユニポーラ駆動
  ユニポーラ駆動とは、モータコイルに対し常に一定方向に電流を流す方式である。コイルの中央にタップ端子があり、どちらに電流を流すかでコイルの極性を切り替える。制御回路はシンプルであるが、バ

イポーラ駆動と比べて出力トルクが低い。電流をオン、オフにする際にコイルに逆起電力が発生し、高電圧がかかるため高耐圧の半導体が必要である。

- バイポーラ駆動
  バイポーラ駆動とは、モータコイルに対して双方向に電流を流す方式である。コイルに流す電流の向きを変えてコイルの極性を切り替える。ユニポーラ駆動に比べて制御回路は複雑になるが、電流の制御が簡単であり、高い出力トルクが得られる。また、ユニポーラ駆動とは異なりコイルに発生する逆起電力を低減できるため低耐圧の半導体で充分である。

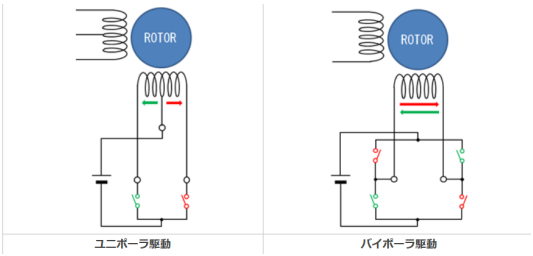

図 14　ユニポーラ駆動とバイポーラ駆動 (文献 [1] より引用)

次にロータの種類による分類について、ロータには VR 型、PM 型、HB 型の 3 種類が存在する。

- VR 型

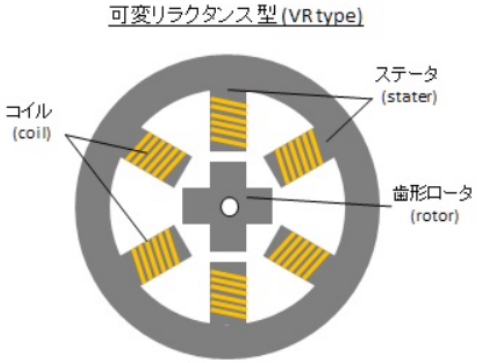

図 15　VR 型 (文献 [2] より引用)

図 15 は VR 型を示す。VR 型は永久磁石用いず、ステータに巻いたコイルに流れる電流のみによって励磁されるステッピングモータである。ロータの外面と、ステータの内面にはそれぞれ歯が付けられている。ロータ外面およびステータ内面は磁性材料になっており、ステータのコイルに電流が流れると励磁されロータの歯がステータの歯に引き寄せられて回転する。

- PM 型

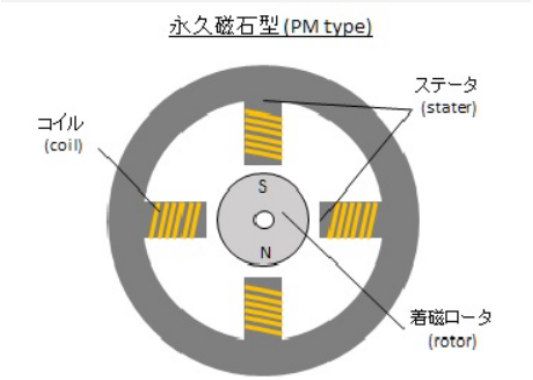

図 16　PM 型 (文献 [2] より引用)

図 16 は PM 型を表す。PM 型は 2 つのコイルと永久磁石によって回転するステッピングモータである。一般的に 2 相構造が多い。ロータは機械歯を持たないが、S 極と N 極に磁化されている。磁化されたロータ極によって磁束密度が増大し、強いトルクを発生する。パルス信号の数に比例した回転をするモータであり、パルス信号の周波数を変えることで回転速度を調節できる。

- HB 型

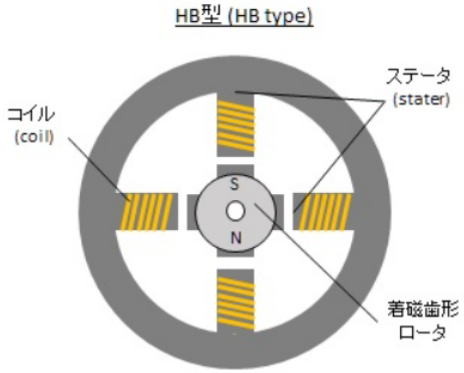

図 17　HB 型 (文献 [2] より引用)

図 17 は HB 型を表す。 VR 型、PM 型のメリットを生かしたのが HB 型である。ステータの内面とロータの外面に歯をもち、ステータには励磁コイルをもつ複数の極がある。また、ロータ内は円筒形の磁石が入っており磁化されている。円筒形の磁石を囲む 2 枚のロータの歯は 1/2 ピッチずつずれており、ステータの励磁コイルに通電することによって磁化しているロータが引き合わされて回転する。

7 [調査課題 2-3]

以下に C 言語による台形速度制御のプログラムを示す。

```
#include <stdio.h>
#include <unistd.h>
#include <sys/io.h>
#include <stdlib.h>
#include <time.h>

#define PPA_0 ((unsigned long)(0x378)) // パラレルポート出力アドレス
#define PBS 2                          // パラレルポート byte size
#define TRUE 1

int main(int argc, char* argv[]) {
  int f_max = atoi(argv[1]); //パルス信号の最大周波数 (Hz)
  int t = atoi(argv[2]); //加減速にかける時間 (/20ms)
  int i;
  double f; //パルス信号の周波数 (Hz)
  double T = 1; //usleep の引数、単位はマイクロ秒
  clock_t x;
  int bi[8]; //パラレルポートのピンに対応するビットの配列
  for (int j = 0; j < 8; j++){
    bi[j] = 1<<j;
  }

  if ((i = ioperm(PPA_0, PBS, TRUE)) != 0) { //動作チェック
    printf("1\n");
    exit(-1);
  }
  while (1) { //1 相励磁, 台形速度制御, 約 t*5*10^(-5) 秒かけて f_max まで加速, その後約 5 秒間
最大周波数で駆動したのち, 約 t*5*10^(-5) 秒かけて減速, 停止
    if (T >= 0){
      x = clock();
      if (x <= t){
 f = (double)f_max/(double)t*x;
 T = 1.0e6/f;
      }
```

```
     else if (x < 100000+t){
T = T;
     }
     else if (x < 100000+2*t){
f = -(double)f_max/(double)t*(x-2*t-100000);
T = 1.0e6/f;
     }
     else{
T = -1;
     }
     outb(bi[0],PPA_0); //1 番ピン
     usleep(T);
     x = clock();
     if (x <= t){
f = (double)f_max/(double)t*x;
T = 1.0e6/f;
     }
     else if (x < 100000+t){
T = T;
     }
     else if (x < 100000+2*t){
f = -(double)f_max/(double)t*(x-2*t-100000);
T = 1.0e6/f;
     }
     else{
T = -1;
     }
     outb(bi[1],PPA_0); //2 番ピン
     usleep(T);
     x = clock();
     if (x <= t){
f = (double)f_max/(double)t*x;
T = 1.0e6/f;
     }
     else if (x < 100000+t){
T = T;
     }
     else if (x < 100000+2*t){
f = -(double)f_max/(double)t*(x-2*t-100000);
T = 1.0e6/f;
```

```
      }
      else{
 T = -1;
      }
      outb(bi[2],PPA_0); //3 番ピン
      usleep(T);
      x = clock();
      if (x <= t){
 f = (double)f_max/(double)t*x;
 T = 1.0e6/f;
      }
      else if (x < 100000+t){
 T = T;
      }
      else if (x < 100000+2*t){
 f = -(double)f_max/(double)t*(x-2*t-100000);
 T = 1.0e6/f;
      }
      else{
 T = -1;
      }
      printf("%lf\n",f);
      outb(bi[3],PPA_0); //4 番ピン
      usleep(T);
    }
  }
}
```

- 実行コマンド
  sudo ./trapezoid f_max t(f_max:最大周波数、t:加減速にかける時間を示す)

参考文献

[1] 日本パルスモーター株式会社
http://www.pulsemotor.com/products/feature/steppingmotordrive1.html(2019 年 4 月 20 日参照)

[2] index Pro
https://www.indexpro.co.jp/article/detail/2/6(2019 年 4 月 20 日参照)

[3] エフテック株式会社

http://www.ftech-net.co.jp/robot/howto/motor02.html(2019 年 4 月 23 日参照)