H8マイコンによるステッピングモータ制御

筒居 稔

平成 29 年 4 月 30 日

1　実験目的

産業用ロボットの関節, ハードディスクのヘッド移動、X-Y ステージの台座など、制度の要求される位置決めアクチュエータとして重要なステッピングモータについて理解する。ステッピングモータは、複数のチャンネルによるパルスの組み合わせにより制御するため、デジタル制御回路との相性がいい。そこで、C 言語によりプログラム開発を行い、H8 マイコンを用いてステッピングモータを制御する方法を習得する。

2　実験機器

表 1: 使用機器

機器名	メーカー	製品名
ステッピングモータ	日本電産コパル	SPG27-1601
H8 マイコン	ルネサスエレクトロニクス	H8/3052F
直流安定化電源	GW Instek	GPS-1805D
オシロスコープ	GW Instek	GDS-1052-U

3　実験原理

3.1　ステッピングモータの動作原理

ステッピングモータはパルスモータとも呼ばれ、パルス信号を与えることにより決められたステップ単位で回転する。

図 1 は今回使用した二相のユニポーラ型ステッピングモータの構成である。図 1 内のφ 1C とφ 2C はそれぞれ電源とつながり、それ以外はグランドとつながっている。φ 1C からφ 1、φ 2C からφ 2、φ 1C から $\overline{\phi 1}$、φ 2C から$\overline{\phi 2}$の順に電流を流す (励磁する) とモータが回転する。

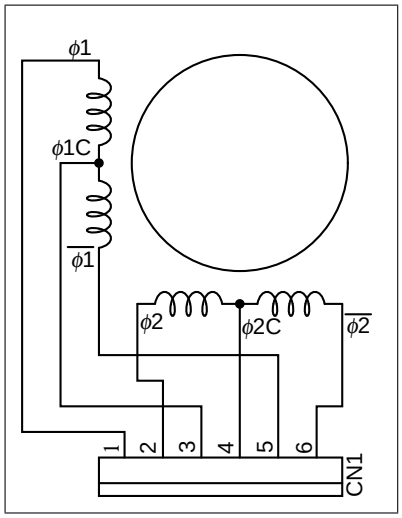

図 1: ステッピングモータの構成

3.2 1 相励磁と 2 相励磁

ステッピングモータにパルス信号を送る際に、1 相励磁の場合、φ 1、φ 2、$\overline{\phi\ 1}$、$\overline{\phi\ 2}$、と順に信号を送ればモータは回転する。2 相励磁の場合はパルスを送る順番は 1 相励磁と同じだかが、前に送った信号と次に送る信号に半分ずつ今回の信号をかぶせてモータの回転をなめらかにする。

4 実験方法

4.1 SCI 通信実験

自身の PC で GCC Developer Lite を管理者モードで起動し、設定をもう一度確認する。ソースコード 1 を記述し、メニューの「コンパイル」「ビルド」でコンパイルする。エラーが出た場合はミスを直し再びコンパイルする。

エラーが出ず、コンパイルに成功した場合以下の手順でプログラムの書き込みを行う。

1. USB シリアル変換ケーブルを PC に接続し、デバイスマネージャーで COM 番号を確認する。
2. シリアルケーブルを実習装置側の SCI コネクタに接続する。
3. 書き込みスイッチを ON にし、モード設定端子の MD2 をジャンパピンで短絡させる
4. マイコンボードの電源コネクタに AC アダプタを接続する。(緑色 LED が点灯)

5. H8 Writer を起動し、書き込み制御ファイルを「3052_F25M_P384.MOT」、Baud(B) を「19200」、Baund(P) を「38400」にした後、COM Port を自分のパソコンに合わせる。

6. 「書き込むファイル」をクリックし、書き込みたいファイル (作成したソースコードの拡張子を mot にしたもの) を指定する。最後に「書き込み開始」をクリックすると書き込みが始まる。

7. 書き込みが完了したら、AC アダプタを外してから、書き込みスイッチを OFF にし、モード設定端子のジャンパピンを MD2 から MD0 に付け替える。

そして「ツール」から SIMLE TERM を立ち上げ、「ファイル」「プロパティ」を開き、Propaty の Connect to を自分の設定に、Baudrate を「38400」に合わせ、「通信」「ポートオープン」を選択する。

マイコンボードに AC アダプタを接続すると、ターミナルに「@」が表示され、キー入力すると、その文字が表示されることを確認。確認後、AC アダプタを外す。

4.2 ステッピングモータの手動動作実験

4.2.1 未完成部分の接続

1. 図 1 を参考にモータドライバ回路と、ステッピングモータを接続。

2. モータ用電源に外部電源を接続。

3. マイコンボードに AC アダプタを接続する。

4. 外部電源を 5V に設定し電圧を印加する。

4.2.2 手動動作確認

1. 0 から 3 のスイッチを順番に押し、モータが回転することを確認する。

2. 3 から 0 のスイッチを順番に押し、モータが逆回転することを確認する。

3. スイッチの ON/OFF とトランジスタの状態、およびステッピングモータの励磁状態をまとめる。

4.3 ステッピングモータのマイコン制御実験

4.3.1 正回転動作の確認

1. program/stopmotor3052.c をコンパイルし、マイコンに書き込む。
2. モータ用電源に外部電源で 5V を印加する。
3. SIMPLE TERM を起動し、マイコンボードに AC アダプタを接続する。
4. 「g」キーを入力すると正回転することを確認する。
5. 「g」キー以外を入力すると回転が停止することを確認する。

4.3.2 モータ印加電圧による回転速度変化の確認

1. モータ用電源に外部電源で 5V を印加した状態で、1 回転に要する時間を測定する。
2. 印加電圧を 9V、12V に変化させ、同時に時間を測定する。

4.3.3 1 相励磁と 2 相励磁による回転速度とスムーズさの確認

1. プログラムを変更し、小文字の「g」で 1 相励磁、「j」で 2 相励磁となるようにする。
2. それぞれの相で回転速度を確認する。そのとき目視で動作の状態を確認する。
3. それぞれの相での波形状態をオシロスコープで確認する。

4.3.4 正回転、逆回転動作の確認

1. プログラムを変更し、「g」で正回転、「h」で逆回転動作となるようにする。
2. 動作を目視で確認し、オシロスコープで波形確認する。

5 実験結果

5.1 SCI 通信実験

ソースコードでコンパイル後手順に従い書き込み、通信を確認。

5.2　ステッピングモータの手動動作実験

5.2.1　未完部分の接続

図 1 に従いモータと回路を接続。

5.2.2　手動操作確認

スイッチを順番に押し回転、逆回転を確認。

5.3　ステッピングモータのマイコン制御実験

5.3.1　正回転動作の確認

ソースコード 2 を入力後正回転を確認。

5.3.2　モータ印加電圧による回転速度変化の確認

直流安定化電源を用いて、印加電圧を 5V、9V、12V と変化させモータの回転速度の変化を目視で確認。

表 2: 印加電圧によるモータの 1 回転に要する時間の変化

電圧 [V]	時間 [s]
4.99[V]	0.0[s]
9.00[V]	12.5[s]
12.00[V]	11.9[s]
15.00[v]	11.8[s]

5.3.3　1 相励磁と 2 相励磁による回転速度とスムーズさの確認

ソースコード 3 を入力後「g」で 1 相励磁「j」で 2 相励磁でステップモータを回転させ、回転状態を目視で確認。目視では違いを確認できなかった。オシロスコープで 1 相励磁と 2 相励磁の波形を取った

5.3.4　正回転、逆回転動作の確認

ソースコード 3 を入力後「g」で正回転、「h」で逆回転を確認。オシロスコープで波形を取った。

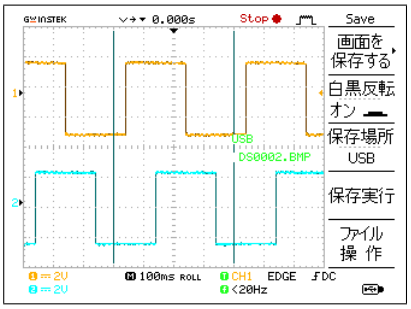

図 2: 2 相励磁

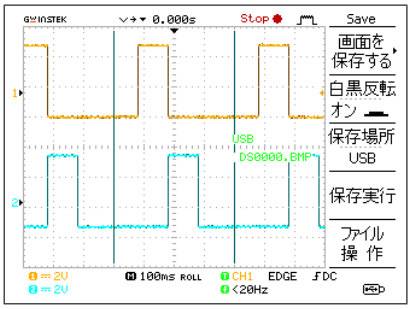

図 3: 1 相励磁

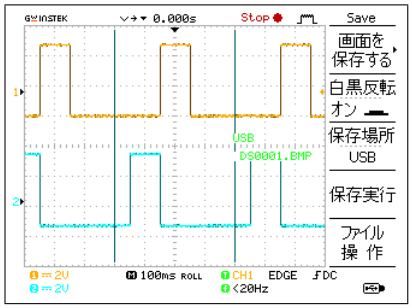

図 4: 逆回転

6 ソースコード

```
/* scitest3052.c --- SCI 通信確認プログラム */
#include <3052.h>
void timer_init(void);// タイマ初期設定
void wait(unsigned int);// 指定時間待機 (ms)
void sci_init(void);
char sci_get_char(void);
void sci_print_char(char);
int main(void) {
timer_init();
sci_init();
sci_print_char('@');
while(1) { sci_print_char(sci_get_char()); }
return 0;
}
// [SCI] ----------------------------------------
void sci_init(void) {
SCI1.SCR.BYTE = 0x00;
SCI1.SMR.BYTE = 0x00;
SCI1.BRR = 19; // 38400baud(25MHz)
wait(1);
SCI1.SCR.BYTE = 0x30; // scr = 0011 0000 (TE=1, RE=1)
SCI1.SSR.BYTE &= 0x80; // エラーフラグクリア
}
```

```
char sci_get_char(void) {
char c;
if(SCI1.SSR.BIT.RDRF == 1) {
c = SCI1.RDR;
SCI1.SSR.BIT.RDRF = 0;
return(c);
} else {
SCI1.SSR.BYTE &= 0xc7;
return('\0');
}
}
void sci_print_char(char c) {
if(c != '\0') {
while(SCI1.SSR.BIT.TDRE == 0); // wait
SCI1.TDR = c;
SCI1.SSR.BIT.TDRE = 0;
}
}
// [TIMER, wait] ------------------------------
void timer_init(void) {
ITU0.TCR.BYTE = 0x23; // timer 用レジスタ
ITU0.GRA = 0x0c35; // 1ms 単位にする
ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
void wait(unsigned int msec) {
int i;
ITU.TSTR.BIT.STR0 = 1; // timer スタート
for (i=0; i < msec; i++) {
while(ITU0.TSR.BIT.IMFA == 0) {} // wait
ITU0.TSR.BIT.IMFA = 0; // timer リセット
}
ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
```

ソースコード 1:SPI 通信確認プログラム

```
#include <3052.h>

void timer_init(void);// タイマ初期設定
void wait(unsigned int);// 指定時間待機 (ms)
void sci_init(void);
char sci_get_char(void);
void sci_print_char(char);
void sci_print(char *);
int drive_stpmotor(char, int, int);

int main(void) {
  volatile int count = -1, fst = 0, inputp = 0, startp = 0;
  volatile char data = '\0', pdata = '\0';

  timer_init();
  sci_init();
```

```
  sci_print("\n* Start\n");

  PA.DDR = 0xf0; // PA: Switch (0-3, 1:OFF / 0:ON)
  PB.DDR = 0xff; // PB: LED (0-3, 0:OFF / 1:ON)

  while(1) {
    wait(100);
    pdata = data;
    data = sci_get_char();
    inputp = data != '\0' ? 1 : 0;// 入力があったか？
    startp = count != -1 ? 1 : 0 ;// モータが回っているか？
    if(!inputp && !startp) continue;
    if( inputp && !startp) {
      fst = 1; count = 0;
    }
    if(!inputp &&  startp) { fst = 0; data = pdata; }
    if( inputp &&  startp && data != pdata) { fst = 1; }
    count = drive_stpmotor(data, count, fst);
    if(count == -1) {
      data = '\0';
      if(fst == 1 && pdata != '\0') {
        fst = 0;
        sci_print("  Stop\n");
      }
    }
  }
  return 0;
}


// [Motor] --------------------------------------
int drive_stpmotor(char c, int cnt, int fst) {
  switch(c) {
  case 'g':
    if(fst) { sci_print("  g\n"); fst = 0; }
    switch(cnt) {
    case 0: PB.DR.BYTE = 0b0001; break;
    case 1: PB.DR.BYTE = 0b0010; break;
    case 2: PB.DR.BYTE = 0b0100; break;
    case 3: PB.DR.BYTE = 0b1000; break;
    }
    return (cnt+1) & 0x3; break;
  }
  return -1;
}
// [SCI] ----------------------------------------
void sci_init(void) {
  SCI1.SCR.BYTE = 0x00;
  SCI1.SMR.BYTE = 0x00;
  SCI1.BRR = 19; // 38400baud(25MHz)
  wait(1);
  SCI1.SCR.BYTE = 0x30; // scr = 0011 0000 (TE=1, RE=1)
  SCI1.SSR.BYTE &= 0x80; // エラーフラグクリア
}
```

```
char sci_get_char(void) {
  char c;
//  while((SCI1.SSR.BYTE & 0x78) == 0);
  if(SCI1.SSR.BIT.RDRF == 1) {
    c = SCI1.RDR;
    SCI1.SSR.BIT.RDRF = 0;
    return(c);
  } else {
    SCI1.SSR.BYTE &= 0xc7;
    return('\0');
  }
}
void sci_print_char(char c) {
  if(c != '\0') {
    while(SCI1.SSR.BIT.TDRE == 0); // wait
    SCI1.TDR = c;
    SCI1.SSR.BIT.TDRE = 0;
  }
}
void sci_print(char *str) {
  while(*str != '\0') sci_print_char(*str++);
}

// [TIMER, wait] --------------------------------
void timer_init(void) {
  ITU0.TCR.BYTE = 0x23; // timer 用レジスタ
  ITU0.GRA = 0x0c35; // 1ms 単位にする
  ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
void wait(unsigned int msec) {
  int i;
  ITU.TSTR.BIT.STR0 = 1; // timer スタート
  for (i=0; i < msec; i++) {
    while(ITU0.TSR.BIT.IMFA == 0) {} // wait
    ITU0.TSR.BIT.IMFA = 0; // timer リセット
  }
  ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
// ---------------------------------------------
```

ソースコード 2:モータ正転プログラム

```
#include <3052.h>

void timer_init(void);// タイマ初期設定
void wait(unsigned int);// 指定時間待機 (ms)
void sci_init(void);
char sci_get_char(void);
void sci_print_char(char);
void sci_print(char *);
int drive_stpmotor(char, int, int);

int main(void) {
  volatile int count = -1, fst = 0, inputp = 0, startp = 0;
```

```
  volatile char data = '\0', pdata = '\0';

  timer_init();
  sci_init();

  sci_print("\n* Start\n");

  PA.DDR = 0xf0; // PA: Switch (0-3, 1:OFF / 0:ON)
  PB.DDR = 0xff; // PB: LED (0-3, 0:OFF / 1:ON)

  while(1) {
    wait(100);
    pdata = data;
    data = sci_get_char();
    inputp = data != '\0' ? 1 : 0;// 入力があったか？
    startp = count != -1 ? 1 : 0 ;// モータが回っているか？
    if(!inputp && !startp) continue;
    if( inputp && !startp) {
      fst = 1; count = 0;
    }
    if(!inputp &&  startp) { fst = 0; data = pdata; }
    if( inputp &&  startp && data != pdata) { fst = 1; }
    count = drive_stpmotor(data, count, fst);
    if(count == -1) {
      data = '\0';
      if(fst == 1 && pdata != '\0') {
        fst = 0;
        sci_print("  Stop\n");
      }
    }
  }
  return 0;
}


// [Motor] ---------------------------------------
int drive_stpmotor(char c, int cnt, int fst) {
  switch(c) {
  case 'g':
    if(fst) { sci_print("  g\n"); fst = 0; }
    switch(cnt) {
    case 0: PB.DR.BYTE = 0b0001; break;
    case 1: PB.DR.BYTE = 0b0010; break;
    case 2: PB.DR.BYTE = 0b0100; break;
    case 3: PB.DR.BYTE = 0b1000; break;
    }
    return (cnt+1) & 0x3; break;
  case 'h':
    if(fst) { sci_print("  h\n"); fst = 0; }
    switch(cnt) {
    case 0: PB.DR.BYTE = 0b1000; break;
    case 1: PB.DR.BYTE = 0b0100; break;
    case 2: PB.DR.BYTE = 0b0010; break;
    case 3: PB.DR.BYTE = 0b0001; break;
```

```
    }
    return (cnt+1) & 0x3; break;
  case 'j':
    switch(cnt){
    case 0: PB.DR.BYTE = 0b1001; break;
    case 1: PB.DR.BYTE = 0b0011; break;
    case 2: PB.DR.BYTE = 0b0110; break;
    case 3: PB.DR.BYTE = 0b1100; break;
    }
    return (cnt+1) & 0x3; break;
  }
  return -1;
}
// [SCI] ----------------------------------------
void sci_init(void) {
  SCI1.SCR.BYTE = 0x00;
  SCI1.SMR.BYTE = 0x00;
  SCI1.BRR = 19; // 38400baud(25MHz)
  wait(1);
  SCI1.SCR.BYTE = 0x30; // scr = 0011 0000 (TE=1, RE=1)
  SCI1.SSR.BYTE &= 0x80; // エラーフラグクリア
}
char sci_get_char(void) {
  char c;
//  while((SCI1.SSR.BYTE & 0x78) == 0);
  if(SCI1.SSR.BIT.RDRF == 1) {
    c = SCI1.RDR;
    SCI1.SSR.BIT.RDRF = 0;
    return(c);
  } else {
    SCI1.SSR.BYTE &= 0xc7;
    return('\0');
  }
}
void sci_print_char(char c) {
  if(c != '\0') {
    while(SCI1.SSR.BIT.TDRE == 0); // wait
    SCI1.TDR = c;
    SCI1.SSR.BIT.TDRE = 0;
  }
}
void sci_print(char *str) {
  while(*str != '\0') sci_print_char(*str++);
}

// [TIMER, wait] --------------------------------
void timer_init(void) {
  ITU0.TCR.BYTE = 0x23; // timer 用レジスタ
  ITU0.GRA = 0x0c35; // 1ms 単位にする
  ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
void wait(unsigned int msec) {
  int i;
  ITU.TSTR.BIT.STR0 = 1; // timer スタート
```

```
  for (i=0; i < msec; i++) {
    while(ITU0.TSR.BIT.IMFA == 0) {} // wait
    ITU0.TSR.BIT.IMFA = 0; // timer リセット
  }
  ITU.TSTR.BIT.STR0 = 0; // timer 停止
}
// ------------------------------------------
```

ソースコード 3:モータ正転、逆回転、2 相励磁プログラム

7　考察

印加電圧によるモータの 1 回転に要する時間の変化では、5V では回転しなかったが、9〜15V では時間に差はあったものの、小さかった。計測時間の差は誤差であり、印加電圧の差によりモータの 1 回転に要する時間は変化しない、なぜならステッピングモータの回転速度は与えられたパルス信号により変化するからである。しかし印加電圧が上がればモータのトルクは上がる。

ステッピングモータ制御のプログラムでは、マイコンとの通信により、キーボドから入力された値を検出し、

```
  case 'g':
    if(fst) { sci_print("  g\n"); fst = 0; }
    switch(cnt) {
    case 0: PB.DR.BYTE = 0b0001; break;
    case 1: PB.DR.BYTE = 0b0010; break;
    case 2: PB.DR.BYTE = 0b0100; break;
    case 3: PB.DR.BYTE = 0b1000; break;
    }
```

をループさせることにより制御している。この際、マイコンのポートに信号を送っているのは

0bxxxx(x は 0 か 1)

の部分であり、x が 0 か 1 かでステッピングモータのϕ 1、ϕ 2、$\overline{\phi\ 1}$、$\overline{\phi\ 2}$にパルス信号を送るか決める。そのため逆回転では正回転のパルス信号を反転させた信号を送信した。2 相励磁では前回と次回のパルス信号に自身の信号が半分かぶれば良いため前後の相と 1 パルス文ずらしながら同時に励磁させた。1 相励磁と 2 相励磁の違いは目視では確認できなかったが、実際には 2 相励磁では、パルス幅が 1 相励磁の 2 倍となり、1 相励磁に比べて回転が安定し、大きなトルクが得られる、ただし消費電力も二倍となる。