

# H8 マイコンを使ったステッピングモーター駆動例

## はじめに

H8 マイコンを使ったステッピングモーター（パルスモーター）の駆動例を紹介します。サンプルでは H8/3664 を使用し、タイマ W のトグル出力機能を利用してパルスを生成しています。同様の機能は他の H8 マイコンや SH マイコンにもありますから同様に使用できます。タイマ機能でパルスを生成することから、モーター駆動時は CPU 負荷が 0 であることが特徴です。

## ステッピングモーターについて

ステッピングモーターはパルスモーターとも呼ばれ、入力したパルスの数に応じて回転角が決まりますので、非常に正確な制御ができることが特徴です。普通の 2 相ステッピングモーターの場合、1 パルスで 1.8 度回転します。したがって与えるパルスの個数で回転角が決まるのでデジタル向きといえるでしょう。

しかし、DC モーターなどと比べてトルクが劣るのが難点です。

## ハードウェア解説

回路図を見ながら参照して下さい。

### 1 電源

電源は 6V2A の AC アダプタを利用することにしました。今回使用したモーターの定格電圧が 2.66V なのでこのくらいの電圧にしました。モーターの定格電圧に合わせて電源を選定して下さい。

### 2 マイコン電源

6V の電源を 3 端子レギュレータ TA48M033F によって 3.3V まで落としてマイコンに供給します。このレギュレータは秋月電子さんでコンデンサとセットになって販売されています。

### 3 モーター電源

6V の電源を、やはり秋月電子さんの安定化電源キット(LM338T 使用)を使用して 2.66V に落として使用します。モーターの電流が最大 2A 流れるので、相当な発熱量です。今回は実験なので放熱板を付けませんでした。実際に使用する場合は、放熱板を必ず付けて下さい。

また、発熱量が気になる場合はスイッチング電源を使用すると良いでしょう。スイッチング電源はエネルギー効率が良いのであまり発熱しません。

また、今回 2.66V という定格電圧の低いモーターを使用しましたが、電圧がもっと高い場合(5V とか 12V) の場合は AC アダプタがスイッチングタイプのものを利用すれば、AC アダプタから直接供給しても構いません。モーターと電源はセットで考えて、モーターに合わせて電源を選定して下さい。

### 4 モーター

モーターは今回、定格電圧 2.66V で、1/36 の減速ギアのものを利用しました。しかし電源を変えるなら他のものでももちろん構いません。電源はモーターの定格電流以上のもにして下さい。

また、今回の例では 2 相励磁の制御をしますので 2 相ステッピングモーターを使用します。

## 5 H8 マイコン

弊社の H8/3664CPU ボード YHN64-1 とインターフェースボード YHN64-2 を利用しました。インターフェースボードは ROM に書き込む時だけ利用し使用時ははずします。

## 6 モータードライバ

マイコンからのパルス出力では、モーター駆動電流を流せませんので、ダーリントン接続されたトランジスタで増幅します。このダーリントントランジスタが 4 個入った IC MP4024 を使用しました。バイパス抵抗やサージ吸収用のダイオードも内蔵されており、外付け部品なしでそのまま使用できます。この IC も秋月電子さんで購入できます。

## 7 スイッチ

マイコンの外部割り込み端子(IRQ0、IRQ1)にスイッチを 2 個付けて逆転スイッチ、倍速スイッチとしました。

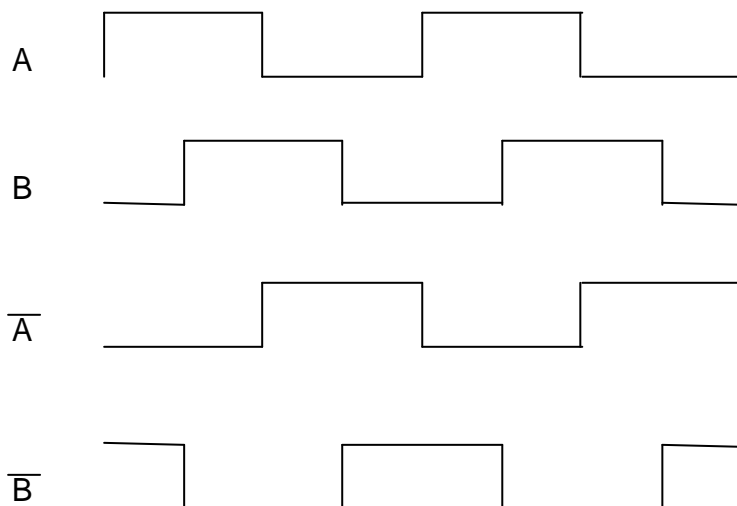
## ステッピングモーターの制御方法

ステッピングモーターの制御方法には以下のものがあります。

- 1 相励磁方式
- 2 相励磁方式
- 1-2 相励磁方式

このうちトルクが大きく、ポピュラーな 2 相励磁方式を採用します。

以下のようなパルスをモーターに送ってやると回転します。



## マイコンでの制御

それでは、このパルスをマイコンで生成するにはどうすれば良いでしょうか。一番簡単な方法は汎用ポートを4ビット使い、

1001  
1100  
0110  
0011  
。  
。  
。

と順に出力してやれば、同じ波形が得られます。出力するときは、インターバルタイマ割り込みを利用すれば正確な速度が得られます。

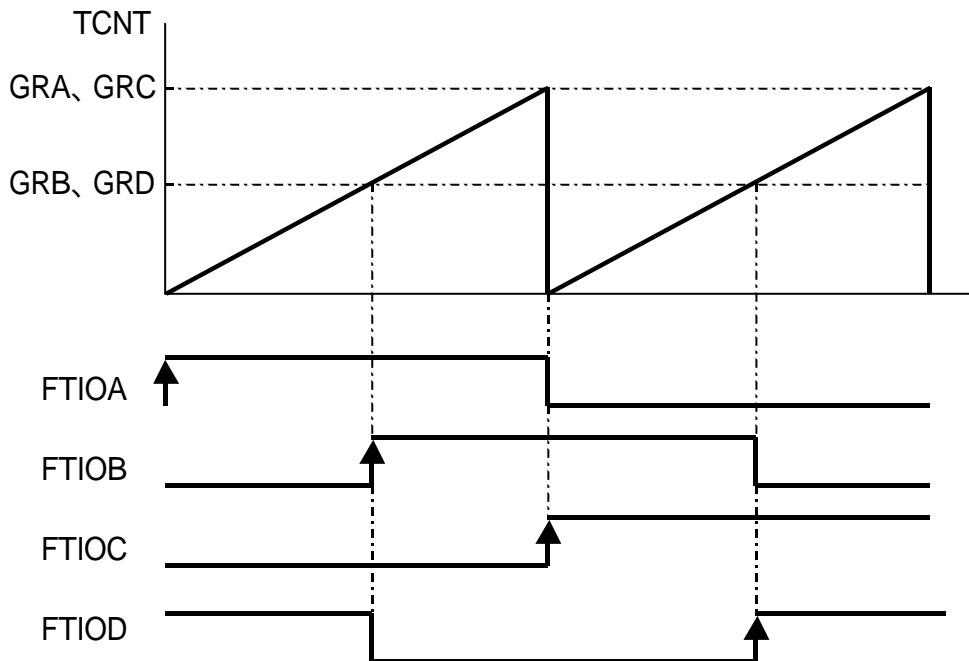
マイコンの仕事がモーターを回すためだけならこの方法でも良いのですが、インターバルタイマ割り込みが常に入りますので、高速回転の場合は頻繁に割り込みが入り、マイコンは他の仕事できません。

そこで内蔵タイマであるタイマWのトグル出力機能を使います。タイマのカウントレジスタ TCNT と GRA レジスタの値が比較され一致した場合に FTIOA 端子の出力が反転されます。

同様に GRB レジスタとの一致により FTIOB、GRC レジスタとの一致により FTIOC、GRD レジスタとの一致により FTIOD、が反転されます。

この機能を利用すれば、最初に初期設定をするだけで自動的にパルスが出力され続けます。割り込みは一切かかりません。

次にこの機能を利用した場合の TCNT、GRA ~ GRD、FTIOA ~ FTIOD の関係を図示します。



この図について説明します。

のこぎり波形は TCNT の値です。クロックに応じてインクリメントされていきます。TCNT が GRA に一致したときに 0 クリアしておくように設定しておけば、TCNT は上記のような、のこぎり波形になります。

GRB と GRD は GRA、GRB のちょうど半分の値に設定します。そうすると FTIOB と FTIOD は GRB、GRD と一致したときに反転します。

図を見れば分かると思いますが、FTIOA ~ FTIOD の初期値出力(1or0)が重要であることが分かります。初期出力違うとまったく違うパターンになります。

この図を見るとパルスの立ち上がり（矢印）が 4 つあることが分かります。したがって、この期間に 4 ステップ=1.8 度×4=7.2 度回転します。

## 値の設定

それでは実際に GRA ~ GRD レジスタに代入する値を決めて見ましょう。

条件は、

ベースクロックは、YHN64-1 のクロックが 10MHz で 4 分周するとして 2.5MHz です。

目標は

10 分で 432 回転とします。

（なぜ 432 回転かと言うと、今回使用したモーターは 1/36 の減速ギアがありますので 10 分で 12 回転になります。このくらいだと回転速度を測りやすいからです。）

まず、のこぎり波形の周期を決める GRA レジスタの値を決めます。

- 1 1 秒あたり 0.72 回転になります。(432 ÷ 600=0.72)
- 2 モーターは 200 パルスで一回転します。(360 ÷ 1.8=200)
- 3 したがって 0.72 回転させるには 144 パルスを与えます。(200 × 0.72=144)
- 4 したがって 1 秒間に 144 パルスを与えます。
- 5 のこぎり一山でパルスが 2 個ですから、1 秒間にのこぎり 72 個を作ります。(144 ÷ 2=72)
- 6 つまり周波数 72Hz ののこぎりクロックです。
- 7 TCNT のベースクロックは 2.5MHz ですから
- 8  $2500000 \div 72=34722$  カウントで TCNT をクリアすれば 72Hz になります。
- 9 したがって、GRA = 34722 になります。

さて、ここまでの計算で、GRA が TCNT の最大値 65535 を超えた場合ですが、その場合はベースクロックを 8 分周します。回転が低速の場合は 8 分周しても 65535 を超える可能性があります。16 分周はできないので、その場合の対処として、

CPU のクロックを 10MHz からもっと遅いものに変える。(計算も遅くなってしまいます)

クロックを外部入力する。(これがもっとも現実的、クロックの外部入力端子があります。)

減速ギア付きのモーターにする。

以上が考えられます。しかし低速の場合は、最初に説明したインターバルタイマ割込みによるパルスの出力をしても CPU にそれほど負荷がかかりませんからそれでも良いでしょう。

GRA の値が決まったので GRB ~ GRD の値も決まります。

GRB = (1/2)GRA = 17361

GRC = GRA = 34722

GRD = (1/2)GRA = 17361

## プログラミング

GRA の値が決まったので、これでプログラムを記述します。今まで分かったことを整理すると以下のようになります。

- 1 FTIOA ~ FTIOD をトグル出力として使う
- 2 TCNT のベースクロックはシステムクロックの 4 分周とする
- 2 TCNT を GRA とのコンペアマッチで 0 にクリアする
- 3 GRA=GRC=34722、GRB=GRC=17361
- 4 FTIOA ~ FTIOD の初期出力は FTIOA=1 FTIOB=0 FTIOC=0 FTIOD=1

以上の設定をタイマ W 関連のレジスタで設定すれば OK です。

ここからは H8/3664 ハードウェアマニュアルのタイマ W の章と合わせて読んで下さい。

### TCRW

このレジスタは GRA とのコンペアマッチで TCNT をクリアするかどうか、クロックの分周、および FTIOA ~ FTIOD の初期出力値を設定します。マニュアル通り設定すると

TCRW = 0xA9

になります。

### TIOR0/TIOR1

このレジスタは FTIOA ~ FTIOD のトグル出力の設定をします。マニュアル通り設定すると

TIOR0 = 0x33;

TIOR1 = 0x33;

になります。

### TCNT

TCNT は最初 0 にしておきます。

TCNT = 0;

### GRA ~ GRD

計算で求めた値を代入します。

GRA = 34722

GRB = 17361

GRC = 34722

GRD = 17361

## TMRW

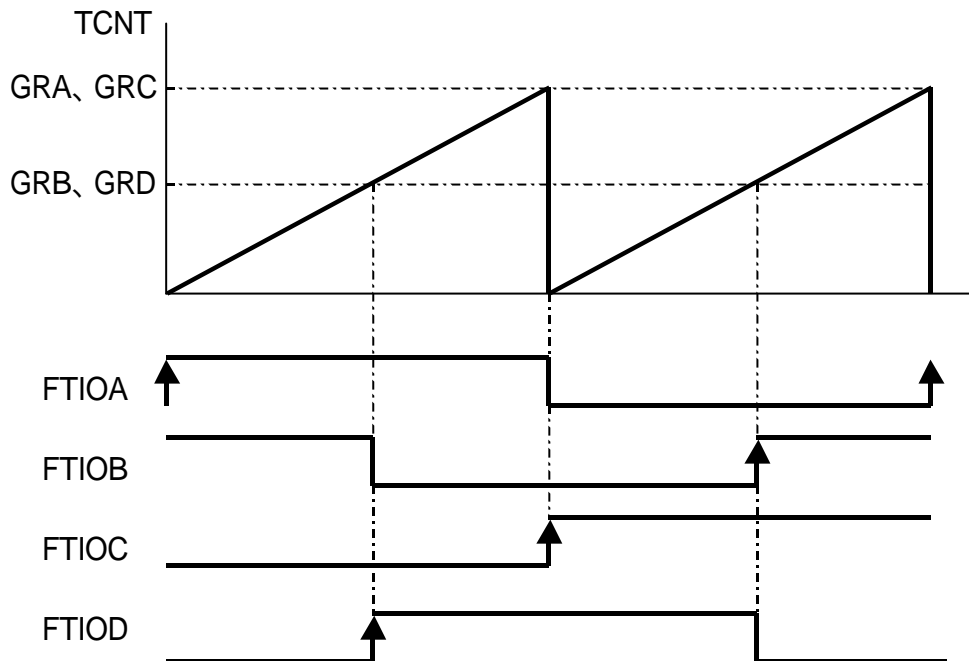
このレジスタの最上位ビットをセットすると、カウンタがスタートします。

```
TMRW |= 0x80;
```

以上をまとめて記述したものが関数 Motor です。

## 逆転

モーターの回転を逆にする波形は以下のようになります。



最初の図と比べると、矢印が左下にずれているのが分かります。そしてこの波形は最初の図で FTIOB、FTIOD の出力を反転させたものと同じです。

最初の図との違いは FTIOB と FTIOD の初期出力が違います。したがって、初期出力を以下のようにすればモーターは逆転します。

```
FTIOA = 1  FTIOB=1  FTIOC = 0  FTIOD = 0
```

プログラムで記述すると

```
TCRW = 0x9C;
```

になります。

## 倍速

モーターの回転を倍にするのは簡単です。分周比を 1/4 から 1/2 にすれば倍になります。また GRA ~ GRD の値を半分にしても倍になります。

## サンプルプログラムの使用上の注意

サンプルプログラムでは、割り込みを使いますので動作させる場合はプロジェクトの設定の「割り込み」画面で割り込みの設定を必ず行って下さい。

ベクタ番号	関数名	言語
14	irq0_int	C
15	irq1_int	C

ベクタ番号は H8/3664 の場合です。

## サンプルプログラムの説明

### 関数 Motor

この関数については今まで説明した通りです。

### 関数 ReInit

モーターを逆転、倍速にするとき呼び出します。

まず、カウンタを停止させて、TCNT を 0 にします。次に flag の値に応じて TCRW の値を再設定します。最後にカウンタをスタートさせます。

### 割り込み関数 irq0\_int

逆転スイッチが押された時の処理です。

逆転スイッチは外部割り込み入力 IRQ0 に接続されています。最初に for ループ文で時間をかせいでいます。これはスイッチのチャタリングを除去するためです。

flag のビット 0 をセットして関数 ReInit を呼び出します。

最後に割り込み要求フラグをクリアします。チャタリングによって 2 度 3 度と割り込みが入っても、この割り込み要求フラグのクリアのため、すべてがキャンセルされます。したがって、割り込み要求フラグのクリアは for 文によるウエイトの後に行うことが重要です。

### 割り込み関数 irq1\_int

倍速スイッチが押された時の処理です。

逆転スイッチの場合と同じです。

### main 関数

最初に for 文を入れて、周辺機器が安定するまで待ちます。なくても動きますが、安全のため入れておきました。

次に PMR1 によって IRQ0 と IRQ1 端子を使う設定をし、IENR1 によって割り込みを許可します。  
\_ei()の後に無限ループに入ります。