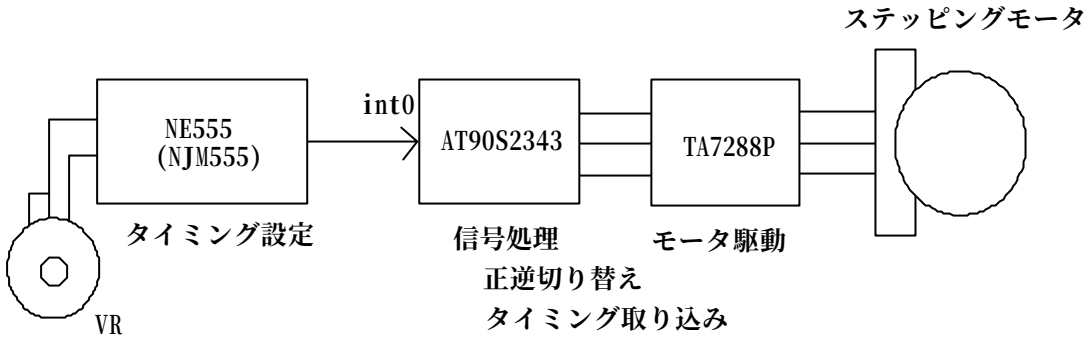
# ステッピングモータミニ基板の製作

ステッピングモータを駆動できる基板を１チップマイクロ CPU：AT90S2343（デジット 5/25 現在　店頭値　５５０円)とタイマIC　NJM555(デジット 5/25 現在　店頭値 50 円)、TA7288P を使って作ってみました。ボリュームを回すと速度が変わっていきます。正逆反転スイッチで回転方向も変えることができます。



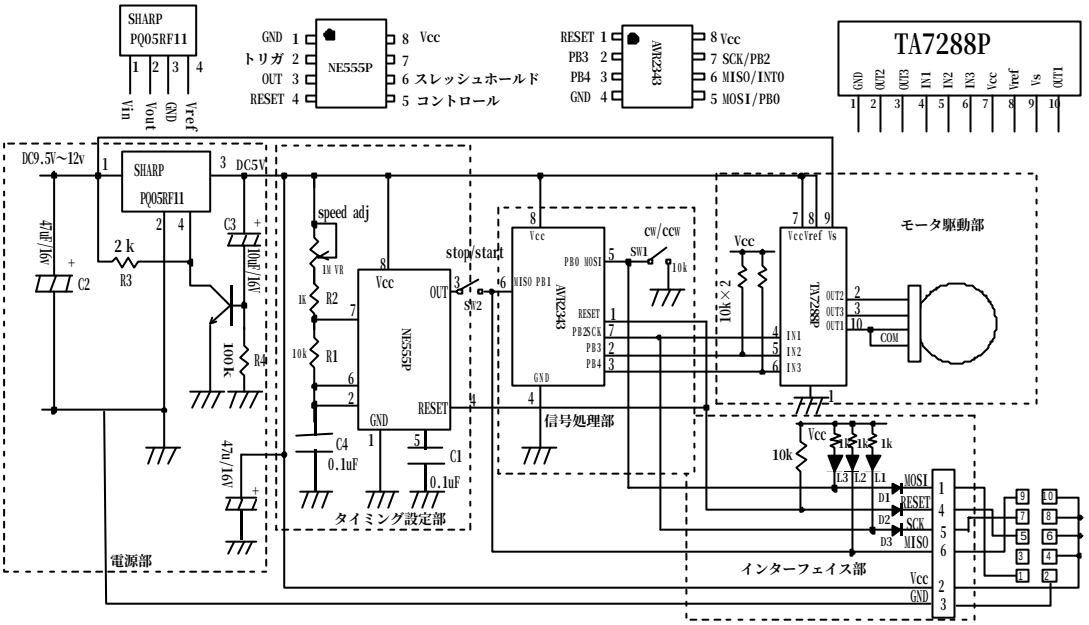

全体の構成は図のようになっています。タイミング用に NE555（デジット 5/25 現在　店頭値 50 円）（NJM555 相当品）を使っています。この５５５の立ち上がりパルスに反応（プログラムで他の設定も可能です）して、１チップマイクロ CPU 中で外部割りこみが起こります。プログラム動作中に、外部割りこみルーチンに入ると、EEPROM からステッピングモータを回転させるための正転パターンを読み出し、出力していきます。スイッチを切りかえると、逆転パターンの読み出しに変わって、モータも逆転することになります。NE555（デジット 5/25 現在　店頭値 50 円）のタイミングに合わせて出力信号が出てくるので、この信号のタイミングが早ければ、モータも速く、遅ければ、モータもゆっくりと回ることになります。また、AT90S2343（デジット 5/25 現在　店頭値　５５０円）では直接ステッピングモーターが駆動できないため（ 20mA までしか流せないのと、出力電圧も５ｖどまり)、別にステッピングモータを駆動するための IC が必要になります。ステッピングモータ専用 IC は割高になるため、IC１個で 2 個の DC モータを回せる DC モータ駆動用 IC TA7288P を代用して使うことにしました。（デジットでの 5/25 現在の店頭値 150 円）現在使っているステッピングモータ（デジット 5/25 現在店頭値　350 円）に 200mA くらい流して使用するので、TA7288P で十分余裕をもってステッピングモータを駆動できます。

使用する 1 チップマイクロ CPU について

Atmel　AVR シリーズ中、(これから、このワンチプマイクロ CPU のことを AVR と呼ぶことにします。）最もピン数が少なく、内部クロック（５V 電源で１MHz）動作可能で省スペース化にも都合のよい AT90S2343（デジット 5/25 現在　店頭値　５５０円）を使っています。当店で動作テストをしていたところ、朝、ブレーカを入れたのち、この CPU がうまく動いていない状態になりました。これは、AC アダプタの平滑コンデンサが完全に放電し

た状態で電源を入れると、電源電圧の立ち上がりの遅くなるのが原因で、正常に動作しないようです。この状態になるとリセットが効かず、もう一度電源を切ってから ON にすると動きました。この状態の本当の原因はわかりませんが、どちらにせよ電源をもう一度切ると確実に動くので、この ON-OFF 動作をする回路を電源部分に付け加えてあります。この辺の事情は別にコラム（作ったけど動かないときのための対策その 1「内部発振モードの時の誤動作」）で書いています。参照してください。

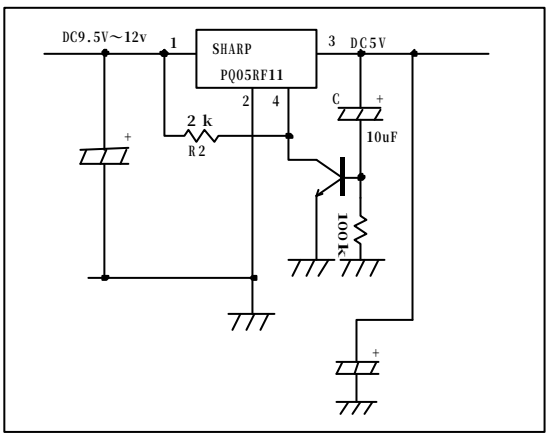
この基板の回路図(下図)を見てください。



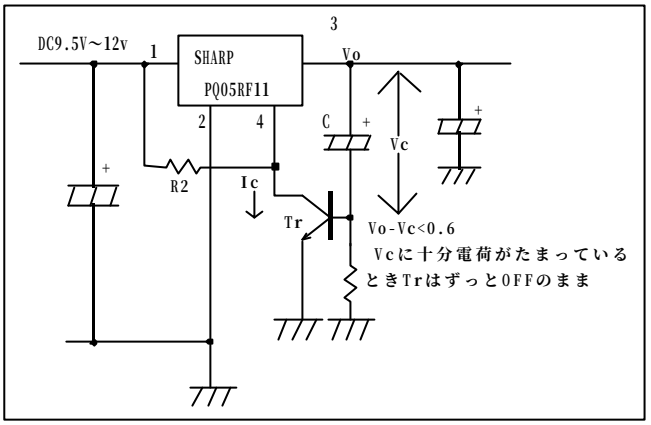

前ページの構成図に示してあるように、タイミングをとるための回路（NE555 周辺）、信号処理の部(AVR の INT0 ピンと出力ピン PB2,3,4)、インターフェイス部(コネクタ、MOSI、MISO、SCK、RESET と LED・抵抗からなるプルアップ回路)、駆動部分（ステッピングモータ、TA7288P）からなっています。電源部分は確実に CPU を動かすための回路がついています。電源から順番に見ていきましょう。

電源部

AT90S2343（デジット 5/25 現在　店頭値５５０円）を内部クロックモードで使うとき、電池などの立ち上がりの早い電源電圧を使う場合を除いて、この回路が必要です。立ち上がりが遅い電源を使うと、CPU が動かなくなるだけでなく、リセット自体が効かなくなるようです。しかし、基板のスイッチを入れなおすと、正常に動きますし、水晶を外付

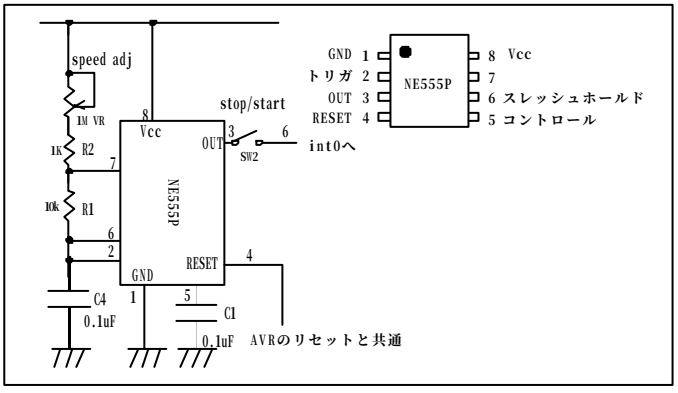


けしている場合は正常に動きます。立ち上がりの遅い電源（例えば、AC アダプタでも、中のコンデンサに電荷が入っていない場合）を使うと、CPU が外部クロック動作で動こうとするようです。そのため外付け水晶のない本基板では CPU が動かなくなります。これは、AC アダプタを一旦ショート（100V 電源から抜いていないとアダプタが壊れます。）させて、電源を入れなおすと、この動かなくなる状態が再現できます。根本的にこの問題を取り除くことはできませんでしたが、手動でスイッチを入れなおす操作をすれば正常に動かすことができることから、それを自動的にやってしまう回路を入れてみました。




レファレンス端子のついた５V　4 端子レギュレータ SHARP PQ05RF11（デジット 5/25 現在店頭値 150 円）を使っています。レギュレータへの入力電源がゆっくり立ち上がって、Vin （＝Vref）が２V 以上になったとき ON 状態になります。レギュレータが ON 状態になると、コンデンサを通ってトランジスタに電流が流れはじめ、Vref（＝Vin-R2*Ic）が下がって（R2 の電圧降下 R2*Ic が大きくなります。）いきます。Vref が２V をきると、コンデンサにいくらか電荷がたまった状態で、レギュレータの出力が OFF（＝０V）になります。このとき、トランジスタに逆バイアスがかかるため完全に OFF になります。トランジスタが OFF になると Vref（＝Vin）は立ち上がっている電源と等しくなって、４端子レギュレータの出力が出始めます。ここで、コンデンサ C に十分電荷がたまっている（上図）と、トランジスタが OFF のままになりトランジスタの ON-OFF 切り替えが起こらなく（図では Vo－Vc＜0.6 のとき）なります。トランジスタの切り替え動作はコンデンサにかかる電圧が（５－$V_{BE}$）V 近くになるまで続きコンデンサにかかる電圧が増えていきます。スイッチングが起こらなくなるころには、電源の電圧が十分上がっている状態で（Vin＞８ｖ）、最後の切り替えで出力からすばやく０から５V に電源が立ち上がらせることができます。

次にタイマ IC555（デジット 5/25 現在　店頭値 50 円）を見て見ましょう。
この回路図を見てください。



新日本無線のデータブック（バイポーラ IC　1993）によると発振周波数ｆは

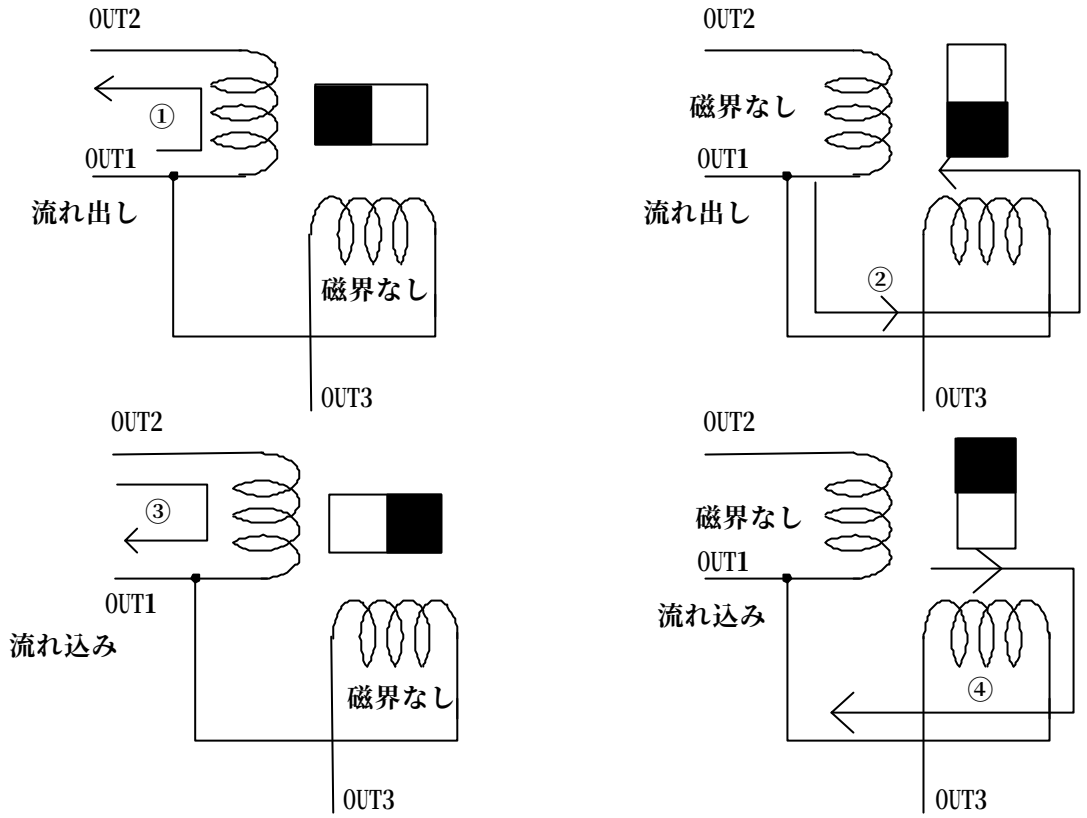
$$f=\frac{1.44}{(R1+2\times R2)\times C}$$

で表せます。１MΩのボリュームを使っているので R1=１ｋ～１．００１MΩ、R2＝１０ｋΩ、ｃ＝0.1μF で計算すると、振周波数は 14.3Hz～１３１Hz の間を変化することになります。これを可変パルス源として AVR の INT0 端子と連動させます。回転速度は発振周波数ｆをステッピングモータのパルス数（1 回転）で割った値になります。このパルス数と、モータのステップ数は異なります。実際、手で回して確かめてみたところ、現在使っているモータ（デジットでの 5/25 現在の店頭値 350 円）を使って 10 秒間での回転数を測ると、実測値が 0.7Hz となっています。1 回転あたりのパルス数を調べないと正確に計算することはできません。1 回転あたりのステップ数が大きいモータを使うとステッピングモータの軸の回転速度が小さくなり、それだけゆっくりモータがまわるように見えます。

駆動部分をみてみましょう。TA7288P は Vs はモータ駆動のための電源端子で５V に落とす前の直流電源から線を回してきます。今回は、9.5v の AC アダプタ（デジットでの 5/25 現在での店頭値 150 円）を使っています。Vcc と Vref は TA7288P の電源として使っており、4 端子の５V 出力につなげてください。TA7288P に入力する信号と出力の関係は下の表のようになります。

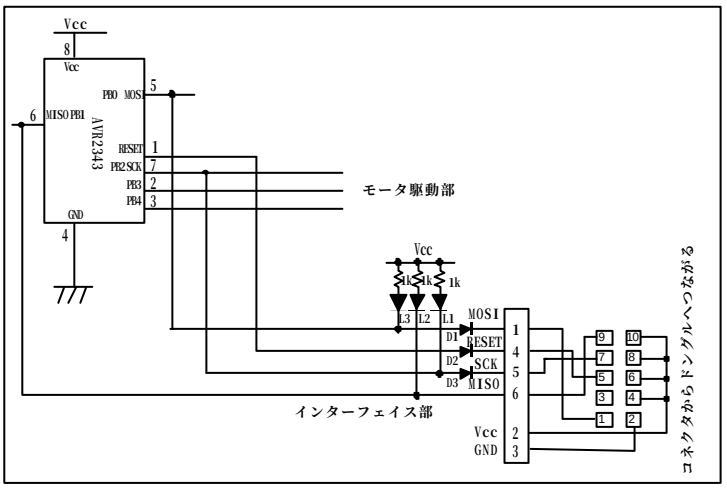
| | 対応する EEPROM データ | (PB2) IN1 | (PB3) IN2 | (PB4) 1IN3 | OUT1 | OUT2 | OUT3 |
|---|---|---|---|---|---|---|---|
| ① | $04 | 1 | 0 | 0 | H | L | ∞ |
| ② | $08 | 0 | 1 | 0 | H | ∞ | L |
| ③ | $14 | 1 | 0 | 1 | L | H | ∞ |
| ④ | $18 | 0 | 1 | 1 | L | ∞ | H |

∞はハイインピーダンス状態を表しており、この間電流はこの端子に流れ込むことはありません。




**駆動電流は上のモデル図のように流れ、OUT1 では電流が流れ出し、流れ込み、常に電流が流れています。AVR から①、②、③、④の順でパルスを送っていくと、磁石をひきつける極が順番にずれていきます。、モデル図の二つのコイルに交互に磁界が発生し、パルスが入るごとに、ある一定角度ずつモータの軸が動いていきます。（一方のコイルにしか電流が流れないため回転方向が決まる）**

**インターフェイス回路は、パソコンと通信するための回路で、LED と抵抗、ダイオードからなる回路です。コネクタとドングルのケーブルは自作される場合、逆接続になりますので注意してください。（本体基板コネクタ０－９に対し、ドングル側のコネクタ９－０の順に接続）**



次に、AVR 中のソースをみてみます。このソース（min2343s）はデジットのホームページでもダウンロードできます。

```
;*****************************************************************************
;*                          ステッピングモータ駆動ミニ基板                   *
;*この基板は、タイマIC  NE555（NJM555と同等）と８ピンのワンチップマイク     *
;*ロコントローラAT90S2343を使っており、小さな駆動基板に正転逆転、速度調     *
;*整(1MΩのボリューム、0.1μFのコンデンサで計算上は14.3Ｈｚ～1310Hzまで)が   *
;*できるようになっています。回転速度はタイマICの発振周波数を変えて、その     *
;*信号にAVRを同期させてステッピングモータを制御します。                     *
;*****************************************************************************
        .equ memo=$61           ;正転状態記憶メモリ
        .include "2343def.inc";インクルードファイルの呼び出し
        .org $000               ;メモリアドレス$000を次の命令に割り当てる
        rjmp ini        ;リセット条件が満たされるとrjmp iniが実行される
        .org $001               ;メモリアドレス$001を次の命令に割り当てる(*1)
        rjmp int0i      ;外部割りこみ条件が満たされるとrjmp int0iが実行される
        .org $003
ini:    ldi r16,$df             ;スタックポインタアドレスを$dfに設定(*2)
        out spl,r16
        ldi r16,$1c             ;PB0,PB1を入力設定。PB2,3,4はステッピングモータ出力
        out ddrb,r16
        ldi r16,$1f             :レジスタr16に$1f（意味は下のとおり）をいれる
        out portb,r16   ;入力プルアップ抵抗ON。モータをブレーキ状態に設定;
*******************モータの回転パターンを設定**************************
        ldi r16,$00             ;EEPROM書きこみアドレスを$00に設定
        ldi r17,$04             ;アドレス＄00にデータ$04を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$01             ;EEPROM書きこみアドレスを$01に設定
        ldi r17,$08             ;アドレス＄01にデータ$08を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$02             ;EEPROM書きこみアドレスを$02に設定
        ldi r17,$14             ;アドレス＄02にデータ$14を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$03             ;EEPROM書きこみアドレスを$03に設定
        ldi r17,$18             ;アドレス＄03にデータ$18を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
```

```
        ldi r16,$05             ;EEPROM書きこみアドレスを$05に設定
        ldi r17,$18             ;アドレス＄05にデータ$18を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$06             ;EEPROM書きこみアドレスを$06に設定
        ldi r17,$14             ;アドレス＄06にデータ$14を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$07             ;EEPROM書きこみアドレスを$07に設定
        ldi r17,$08             ;アドレス＄07にデータ$08を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$08             ;EEPROM書きこみアドレスを$08に設定
        ldi r17,$04             ;アドレス＄08にデータ$04を書きこむための設定
        rcall eew               ;eewルーチンで実際に書きこむ
        ldi r16,$00             ;EEPROM読み込み開始アドレスを$00設定
        out eear,r16
;***************************************************************
        ldi r16,$03
        out mcucr,r16    ;外部割りこみをパルスの「立ち上がり」で入るように設定(*3)
        ldi r16,$ff             ;正転逆転のどちらでもない状態
        sts memo,r16            ;これをmemoに記憶させておく。外部割りこみに入ると
                                ;すぐindrへ飛んで行く。（brne indrが実行される）
        ldi r16,$40             ;
        out gimsk,r16           ;外部割りこみ許可(*4)
        sei                     ;グローバル割りこみ許可。（*5）これで、外部割り
                                ;こみint0が入るためのすべての条件（*1,*2,*3,*4,*5）
                                ;がそろう
main:
        rjmp main               ;mainルーチンでは何もしないでぐるぐる回っている
;*************EEPROM書きこみルーチン*****************************
;*EEPROMにデータを書きこむためには、AVRのマニュアルに指定さ          *
;*れてある特別な手続きに従う必要があります。始めの二行で書きこ      *
;*み準備ができるまでeewとの間をぐるぐる回って待っています。          *
;***************************************************************
eew:    sbic eecr,1     ;EEWEビットが0（＝書きこみ準備完了）になったら1行スキップ
        rjmp eew                ;eewへもどる
        out eear,r16            ;書きこむデータのアドレスを指定
        out eedr,r17            ;そのアドレスに書きこむデータを指定
```

```
        sbi eecr,2              ;EEMWEビットに1を書きこむ
        sbi eecr,1              ;その後すぐに、EEWEビットに1を書きこむ(書きこみ完了)
        ret                     ;割りこみ元に戻る
;******************外部割りこみ0*************************************************
;*555からの立ち上がり信号をint0ピンで受けて、割りこみが入ります。この割りこみが      *
;*入ると現在のスイッチ状態（正転逆転）を読み込んで、それに応じて出力パターンを変    *
;*えて出力します。EEPROM内のデータは$01から$03までが正転パターン、$05から$08までは  *
;*逆転パターンのデータが入っています。PB0のスイッチが切り替わるまでは外部割りこ      *
;*みが入るたびに、一つずつ読み込むアドレスをずらしながら同じパターンで出力していき  *
;*ます。モータはそのパルスに応じて回転し（$03,$08）までいくと始め($00,$05)のデータ  *
;*から出力されます。このようにして、このルーチンで正転逆転、速度調整(速度＝タイマIC *
;*の周波数/n(パルス/秒)　n：ステッピングモータが一回転するためのパルス数)を行って  *
;*います。                                                                        *
;*********************************************************************************
int0i:  in r16,eear             ;EEPROMアドレスをr16におとす
        cpi r16,$04             ;それが正転の端を過ぎたところ（$03+1）か？
        breq intr               ;intrへ分岐する
        cpi r16,$09             ;それが正転の端を過ぎたところ（$08+1）か？
        breq intr               ;intrへ分岐する
        in r16,pinb             ;ポートＢから
        andi r16,$01            ;スイッチ部分だけを残して（このときPB0以外はすべて0）
        lds r17,memo            ;前回の「正逆」を読み出す。正転＝$01
        cp r16,r17              ;前回と同じ回転状態か？
        brne indr               ;切り替わっていた場合indrへ
        in r16,eear
        inc r16         ;現在のEEPROMアドレス＋１　ここのデータをint00で出力
;*********************************************************************************
;*EEPROMの読み込みにも特別な手続きがあります。１：アドレスの設定　２：読み込みスト  *
;*ロボON ３：データ取り込み　の順に読み込んでいってください。                      *
;*********************************************************************************
int00:  out eear,r16            ;そのアドレスを読み込みアドレスとして、設定。
        ldi r16,$01             ;EEREビット（EECRの0ビット目）の設定（＝１）
        out eecr,r16
        in r16,eedr             ;EEPROMデータからの取り込み
        andi r16,$1c            ;出力部分（PB2,3,4）以外はすべて0
        out portb,r16           ;ポートに実際に出力
```

```
        reti                    ;外部割りこみ終了で割りこみ元（main）に戻る
intr:   subi r16,$04            ;アドレスをはじめに戻す（$04または$09）-$04
                                                        ;＝（$00または$05）
        out eear,r16            ;開始点（$00または$05）をEEPROMアドレスに設定
        rjmp int00              ;int00へ：データ出力ルーチン
indr:                           ;memo=PINB0でない場合にここにやってくる
        sts memo,r16            ;新しいスイッチ状態をmemoに記憶
        cpi r16,$01             ;逆転　->　正転状態か？
        breq cw         ;r16＝$01ならcwへ
                ;正転ー＞逆転:逆転コードへ読み込みアドレスを移す
ccw:    in r16,eear             ;アドレス読み込み
        cpi r16,$00             ;アドレス$00か？
        breq cc8        ;$00ならcc8へ
        cpi r16,$01             ;アドレス$01か？
        breq cc7        ;$01ならcc7へ
        cpi r16,$02             ;アドレス$02か？
        breq cc6        ;$02ならcc6へ
        ldi r16,$05             ;EEPROM読み込みアドレスを$05に設定
                                ;(EEAR＝$03のばあい)
        out eear,r16
ccw2:   reti                    ;割りこみ元へ戻る
cc6:    ldi r16,$06             ;EEPROM読み込みアドレスを$06に設定
        out eear,r16
        rjmp ccw2               ;ccw2へ
cc7:    ldi r16,$07             ;EEPROM読み込みアドレスを$07に設定
        out eear,r16
        rjmp ccw2               ;ccw2へ
cc8:    ldi r16,$08             ;EEPROM読み込みアドレスを$08に設定
        out eear,r16
        rjmp ccw2               ;ccw2へ
                ;逆転ー＞正転:正転コードへ読み込みアドレスを移す
cw:     in r16,eear             ;アドレス読み込み
        cpi r16,$05             ;アドレス$05か？
        breq c3         ;$05ならc3へ
        cpi r16,$06             ;アドレス$06か？
        breq c2         ;$06ならc2へ
```

```
        cpi r16,$07             ;アドレス$07か？
        breq c1         ;$07ならc1へ
        ldi r16,$00             ;EEPROM読み込みアドレスを$00に設定
                                ;(EEAR=$08のばあい)
        out eear,r16
cw2:    reti                    ;割りこみ元へ戻る
c1:     ldi r16,$01             ;EEPROM読み込みアドレスを$01に設定
        out eear,r16
        rjmp cw2        ;cw2へ
c2:     ldi r16,$02             ;EEPROM読み込みアドレスを$02に設定
        out eear,r16
        rjmp cw2        ;cw2へ
c3:     ldi r16,$03             ;EEPROM読み込みアドレスを$03に設定
        out eear,r16
        rjmp cw2        ;cw2へ
```

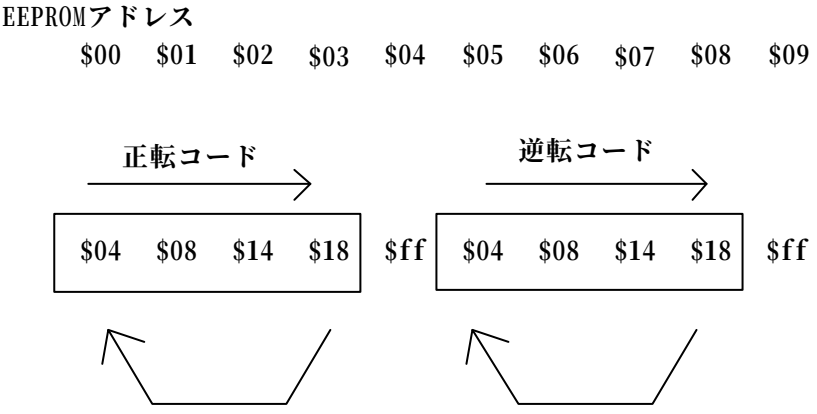
ソースの主な機能は、回転速度制御部分、速度制御部分から成ります。

1. 回転速度制御部分

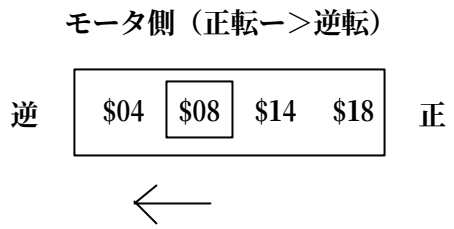
555 から入ってくる信号を基準にしてデータ出力を行っているため、555（デジット 5/25 現在　店頭値 50 円）のパルス周波数によって、データ出力の間隔が変わります。このように、外部からの信号によってステッピングモータの速度制御を行っています。この外部信号を AVR でうけて、プログラム中で外部割りこみが発生します。この外部割りこみルーチンに入るための設定（MCUCR）は、現在、立ち上がりパルス（$03）にしてありますが、他に立ち下がり（$20)、L レベル割りこみ（$00）の２種類も選択できるようになります。データは EEPROM と呼ばれる別のメモリ中に入っており、そこからデータが読み出され出力されます。外部割りこみルーチンで読み出された EEPROM アドレスは、出力時(int00)に 1 つ進めた(int0i 中の inc r16)アドレスのデータが出力されます。

2.方向制御部分

まず、正転時は下の図のように 4 つの EEPROM メモリ内のデータを循環して読み出しています。アドレス$04 のデータ$ff が読み出された場合は、読み出しアドレスが 4 つ戻ってアドレス$00 のデータ$04 を出力します。



これが、例えば$08 で逆転スイッチに切り替えたとすると、次に$04 を出力できれば、モータが逆方向に動きます。ここで、次の外部割りこみでデータ$04 が出力されるようにするためには図のようにデータ$08 が外部割りこみの始めに、読み出されるように、してやればよいことになります。このソースの int0i のルーチンを見てください。




```
int0i:  in r16,eear             ;EEPROMアドレスをr16におとす
        cpi r16,$04             ;それが正転の端を過ぎたところ（$03+1）か？
        breq intr               ;intrへ分岐する
        cpi r16,$09             ;それが正転の端を過ぎたところ（$08+1）か？
        breq intr               ;intrへ分岐する
        in r16,pinb             ;ポートＢから
        andi r16,$01            ;スイッチ部分だけを残して（このときPB0以外はすべて0）
        lds r17,memo            ;前回の「正逆」を読み出す。正転＝$01
        cp r16,r17              ;前回と同じ回転状態か？
        brne indr               ;切り替わっていた場合indrへ
        in r16,eear
        inc r16         ;現在のEEPROMアドレス＋１　ここのデータをint00で出力
```

読み出したアドレスと出力するアドレスが最後で一つずれます。）

あとがき

ワンチップマイクロＣＰＵといえば、ＰＩＣのイメージが強いこのごろですが、今回はＡｔｍｅｌのＡＶＲを使ってみました。今回のステッピングモータ駆動回路は、ワンチップマイコン中のＥＥＰＲＯＭを読み込んで、駆動デバイス（TA7288P）に合わせたデータを出力していきます。また、ピン数の節約のため、外部割りこみを使って int0 端子１つで速度調整ができるようにしました。基準パルス源としてタイマ IC　NE555（デジット 5/25 現在店頭値 50 円）（NJM555 相当品）を使ったので、ボリュームでアナログ的に速度を変えていくことができます。いまだに、AVR のマニュアルが英語版だけなど、とっつきにくい面もあるかもしれませんが、今回の製作をとおして、少しでも AVR の使い方になれてもらえたらいいなと思っています。

AVR,PIC,メカトロ、電子部品の
ことなら…

〒556-0005　大阪市浪速区日本橋 4-6-7

デジット営業所（電子部品店）

TEL 06-6644-4555

FAX 06-6644-1744

In need of AVR, PIC, Mechatoronics, Erectronic Parts…….

Digit sales department

(an Erectonic parts shop)