

Ansteuerung von Schrittmotoren

Gunther Zielosko

1. Grundlagen

Eine besonders reizvolle Aufgabe für das Arbeiten und Entwickeln mit Mikrorechnern ist die Steuerung von mechanischen Baugruppen. Wenn präzise, reproduzierbare Bewegungen mit geringem Sensoraufwand verlangt werden, sind dazu Schrittmotoren (Bild 1) erforderlich. Im Gegensatz zu allen anderen Motoren haben sie kein lastabhängiges Anlaufen und Abbremsen und bleiben auch beim „Abschalten“ fest auf ihrer gerade erreichten Position. Im folgenden werden wir verschiedene Arten von Schrittmotoren, ihre Eigenschaften sowie das Prinzip ihrer Ansteuerung kennenlernen.

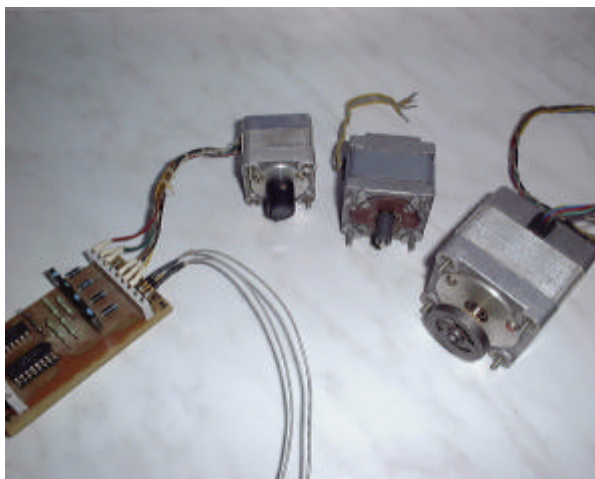


Bild 1 typische Schrittmotoren verschiedener Größe

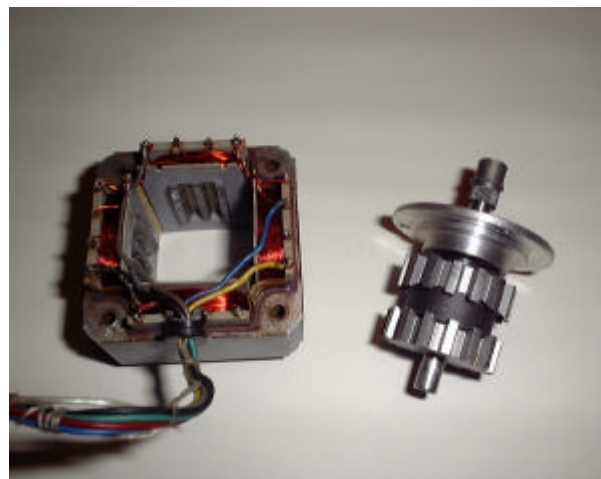


Bild 2 innen liegen die Statorspulen und der Rotor mit Permanentmagnet und Polschuhen

1.1. Aufbau und Wirkungsweise von Schrittmotoren

Prinzipiell besteht ein Schrittmotor aus einem festen Spulensystem und einem rotierenden Permanentmagneten, wie in Bild 2 zu erkennen. Die Spulen werden durch eine Ansteuerelektronik nacheinander eingeschaltet und der Rotor folgt dem dadurch erzeugten Magnetfeld. Die Wicklungen sind so auf den Umfang (360°) verteilt, daß es für jeden Schrittmotor eine festgelegte Schrittweite gibt. Bis zu diesem Punkt funktionieren alle Schrittmotoren gleich, Unterschiede gibt es dann bei der Spulenaufteilung und der Ansteuerung. Wir unterscheiden unipolare (die Spulen werden nur mit einer Polarität beschaltet) und bipolare (die Stromrichtung wird gewechselt) Schrittmotoren.

Unipolare Motoren erkennt man an der höheren Zahl von Anschlüssen (typisch 6), dagegen haben bipolare Motoren meist nur 4 Anschlüsse. Die Bilder 3 und 4 zeigen die beiden typischen Motorschaltungen.

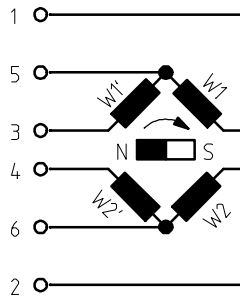


Bild 3 unipolare

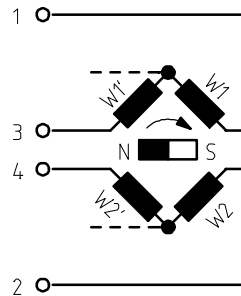


Bild 4 bipolare Schrittmotorschaltung

Man erkennt, daß ein unipolarer Schrittmotor einfach durch Nichtbeschaltung der Spulenmittelabgriffe als bipolarer Schrittmotor benutzt werden kann, nicht aber umgekehrt, wenn diese Mittelabgriffe fehlen (es also nur 2 Wicklungen gibt). Die Ansteuerung der Spulen erfolgt nun nach einem festgelegten Schema, das Tabelle 1 zeigt.

Beim unipolaren Schrittmotor liegen zwei Anschlüsse konstant auf einer Betriebsspannung, während zwei weitere zeitweilig an die andere gelegt werden. Die restlichen beiden bleiben in dieser Phase frei, was natürlich bedeutet, daß durch die entsprechenden Spulen kein Strom fließt. Beim bipolaren Schrittmotor fließt immer durch alle Spulen Strom, was ein höheres Drehmoment des Bipolarmotors zur Folge hat. Gegenüber diesem Vorteil ist die kompliziertere Ansteuerung (Brückenschaltung wegen der Stromumpolung) ein Nachteil. Der unipolare Schrittmotor kommt mit einfachen Schalttransistoren ohne Umkehrung der Stromrichtung aus. Deshalb werden wir uns in diesem Applikationsbericht nur mit der Ansteuerung von Unipolarmotoren beschäftigen.

Schritt	Unipolarschaltung Anschluß:						Bipolarschaltung Anschluß:			
	1	2	3	4	5	6	1	2	3	4
1	-	-	O	O	+	+	-	-	+	+
2	O	-	-	O	+	+	+	-	-	+
3	O	O	-	-	+	+	+	+	-	-
4	-	O	O	-	+	+	-	+	+	-

Erläuterungen:

- + positive Spannung
- negative Spannung (hier Masse)
- O ausgeschaltet

Tabelle 1 Ansteuerschema von unipolaren und bipolaren Schrittmotoren

Bei jedem Schritt dreht sich der Rotor nun um einen (für jeden Motor konstruktiv festgelegten) Winkel vor oder zurück. Typisch sind z.B. 36 oder 100 Schritte pro Umdrehung, was einem Schrittwinkel von 10° oder $3,6^\circ$ entspricht. Die in der Tabelle dargestellten Schritte werden als Vollschritte bezeichnet, neben dieser Betriebsart kann man einen Schrittmotor auch im sogenannten Halbschrittbetrieb betreiben. Dabei erreicht man zwar kleinere Schrittwinkel, der Motor hat aber dann ein noch kleineres Drehmoment, deshalb ist dieser Betrieb weniger interessant für uns.

1.2. Besonderheiten von Schrittmotorantrieben

Eigentlich scheint alles klar: Man koppelt einen Schrittmotor an eine Elektronik, die genau die in Tabelle 1 gezeigten Schrittfolgen erzeugt. Dies kann eine einfache Logikschaltung sein oder ein Mikrorechner wie der BASIC-Tiger®. Allerdings gibt es ein paar Besonderheiten, die in diesem Kapitel etwas näher beleuchtet werden sollen.

Zunächst ist zu beachten, daß wir mit den Spulen der Motoren kräftige Induktivitäten schalten müssen. Dies setzt eine leistungsfähige Stromversorgung voraus. Auch kleine Schrittmotoren brauchen bei Betriebsspannungen von 12V aufwärts locker einige 100mA pro Spule. Zusätzlich sind die erheblichen Induktionsspannungen (bis zu einigen 100V!) zu berücksichtigen, immerhin werden die Spulen impulsförmig ein- und ausgeschaltet. Eine Interface-Schaltung muß also solche Ströme und Spannungen sicher verkraften können.

Eine weitere Besonderheit hängt mit einer eigentlich als Vorteil empfundenen Eigenschaft der Schrittmotoren zusammen. Sie bleiben in der jeweiligen Lage solange stehen, bis ein neuer Schritt ausgeführt wird. Das kann man zum stabilen Halt in der jeweiligen Position nutzen, anders als bei herkömmlichen Motoren gibt es keinen Leerlauf. Andererseits fließt aber auch beim Stillstand des Motors der Spulenstrom – ja dieser Strom ist sogar noch wesentlich größer als beim laufenden Motor. Wenn sich der Motor nämlich dreht, gibt es eine Gegeninduktion, die den fließenden Strom mit der Drehzahl verkleinert. Zusätzlich bietet die Induktivität der Spulen den Ansteuerimpulsen einen höheren Widerstand als ihr rein ohmscher Widerstand im Falle von Gleichstrom. Es ist verzwickt, der Motor braucht im Stillstand mehr Leistung als beim schnellen Drehen. Es gibt nun einige Maßnahmen, die man beim professionellen Einsatz von Schrittmotoren ergreifen kann, um dieses Problem zu entschärfen. Ein erster Schritt ist der Betrieb über eine Stromquelle, die so ausgelegt ist, daß sie gerade den typischen Strom liefert, den der Motor im Betrieb maximal braucht. Das begrenzt den „überhöhten“ Strom und damit auch die Erwärmung des Motors beim Stillstand. Eine weitere Möglichkeit ist, den Betriebsstrom des Motors zu „choppern“, d.h. ihn schnell ein- und auszuschalten. Das erzeugt einen zusätzlichen (induktiven) Widerstand der Spulen auch beim Stillstand des Motors.

Aber das ist noch nicht alles. Wie wir wissen, „springen“ die Rotoren der Schrittmotoren von Position zu Position. Vom Stand aus müssen sie die gesamte Masse einschließlich der anzutreibenden Last schlagartig beschleunigen. Kommt der nächste Schritt zu schnell, kann der Rotor nicht folgen und dreht nicht weit genug, er „verhaspelt“ sich und brummt nur noch. Will man Schrittmotoren richtig betreiben, muß man dafür sorgen, daß sie langsam anlaufen können. Ist erst einmal eine höhere Drehzahl erreicht, kann der Rotor auch höheren Schrittfrequenzen mühelos folgen. Das gleiche gilt für das Abbremsen. Bei hohen Drehzahlen plötzlich stehenbleibende Ansteuersignale bewirken, daß der Rotor infolge seiner Trägheit über die „befohlene“ Position hinausläuft und damit keine definierte Lage mehr hat. Eine

intelligente Schrittmotorsteuerung muß also dafür sorgen, daß Anlauf- und Bremsvorgänge nicht plötzlich, sondern unter allen Umständen „weich“ erfolgen.

Zuletzt noch ein allgemeiner Hinweis zu Schrittmotorantrieben. Wir haben gelernt, daß Schrittmotoren in der Lage sind, sich ihre Position zu „merken“, wenn die Ansteuerimpulse stehenbleiben. Leider haben sie aber nur ein „Kurzzeitgedächtnis“. Wird der Strom abgeschaltet, weiß der Antrieb nicht mehr, wo er sich gerade befindet. Deshalb gibt es in allen Systemen mit Schrittmotoren eine Initialisierungsphase. Sicher haben Sie auch schon Ihren Drucker beim Einschalten beobachtet. Überall rappelt es, ohne daß Sie einen Druckauftrag ausgelöst haben. Was hier passiert, ist einfach erklärt. Alle Motorantriebe, bei denen es auf die genaue Position ankommt, fahren erst einmal in eine Endlage. Dort befindet sich ein Schalter (meist ein normaler Kontakt, manchmal ein Sensor), der dem steuernden System meldet, daß sich der Antrieb in Ausgangsposition befindet. Hier wird der interne Schrittzähler auf 0 gesetzt. Ab jetzt kann sich das System bezogen auf diese Ausgangslage orientieren. Würde man diese Initialisierung vergessen, kann der Antrieb über die Endlagen hinausfahren, Seilzüge zerreißen, sich verklemmen oder anderen Schaden anrichten. Im übrigen sind Endschalter immer zu empfehlen, wenn Sie mit Schrittmotorantrieben experimentieren. Denken Sie nur an Programmierfehler, statt 100 Schritte haben Sie eine Null mehr eingegeben....!

2. Steuerung mit dem BASIC-Tiger®

Mit den bisherigen Kenntnissen könnten wir mit dem BASIC-Tiger® bereits einen Schrittmotor ansteuern. Wir wählen 4 Portleitungen aus, die die in Tabelle 1 gezeigten Signalkombinationen ausgeben. Vier Verstärkertransistoren und die Schrittmotoransteuerung für einen Unipolar-Schrittmotor wäre praktisch fertig. Damit der Tiger nicht immer das jeweilige Codewort für den Motor ausrechnen muss, wird in unserem Beispiel diese Aufgabe einer Hardware-Schaltung übertragen. Der BASIC-Tiger® liefert nun an zwei Ausgängen einfache Impulse, ein Eingang ist für Vorwärts-, der andere für Rückwärtsschritte zuständig. Der jeweils nicht benötigte Ausgang muß auf High-Pegel gelegt werden. Jeder Impuls bewirkt einen Schritt in die entsprechende Richtung. Dieses Verfahren vereinfacht die Positionierung des Antriebs erheblich, der BASIC-Tiger® gibt nur noch die gewünschte Anzahl von Impulsen aus, die Verteilung der Signale auf die Spulen erledigt die Zusatzschaltung. Eine realisierte Schaltung für Unipolarmotoren kleiner Leistung zeigt Bild 5.

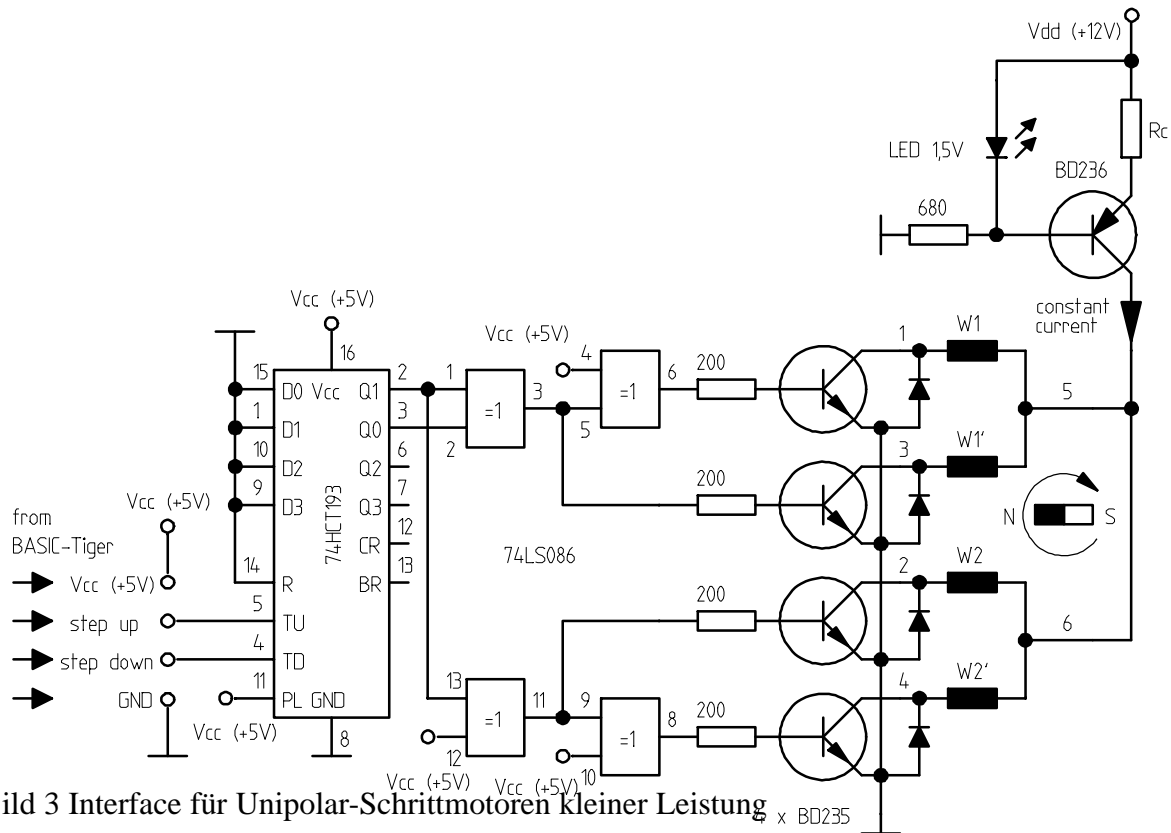


Bild 3 Interface für Unipolar-Schrittmotoren kleiner Leistung

Zur Funktion dieser Schaltung:

Der erste Baustein, ein binärer Vorwärts-/Rückwärtszähler 74HCT193, zählt die eingehenden Impulse an einem der beiden Takteingänge TU oder TD. Voraussetzung für eine ordentliche Funktion ist, daß der jeweils nicht getaktete Eingang auf H-Potential liegt. An seinen Ausgängen Q0 bis Q3 erscheinen die typischen Binärdaten. Die Schaltung wertet nur die beiden niederwertigen Bits Q0 und Q1 (blau) aus:

- 0. Takt 0000
 - 1. Takt 0001
 - 2. Takt 0010
 - 3. Takt 0011
 - 4. Takt 0100
- usw.

Wer sich mit Binärzahlen auskennt weiß, daß die beiden niederwertigen Bits sich ab dem 4. Takt genauso verhalten wie ab dem 0. Takt, d.h. beim Zählen wiederholt sich die Ausgabe

alle 4 Takte. Nach dem Zähler folgen Logik-Gatter vom Typ XOR (74LS086 oder 74HCT086), d.h. an deren Ausgang erscheint nur dann eine logische 1, wenn ein Eingang allein 1 ist. Sind beide 1 oder beide 0, ist das Ergebnis 0. Die Ausgänge der Gatter in der obigen Schaltung treiben über einen Widerstand bereits die npn-Schalttransistoren, die dann die Wicklungen des Schrittmotors an Masse legen (Betriebsspannung kommt aus der Stromquelle mit dem pnp-Leistungstransistor). Betrachtet man die logischen Signale an den Ausgängen der Gatter, ergeben sich folgende Zustände.

Pin 74LS086	6	11	3	8		
Schrittmotor-Anschluß	1 W1	2 W2	3 W1'	4 W2'	5	6
Zählerstand						
00	1	1	0	0	+	+
01	0	1	1	0	+	+
10	0	0	1	1	+	+
11	1	0	0	1	+	+

Tabelle 2 Logische Zustände des Schrittmotor-Interface

Eine logische 1 am Gatterausgang bedeutet, der npn-Schalttransistor schaltet durch und legt den betreffenden Motoranschluß an Masse. Eine logische 0 macht den Transistor hochohmig. Das ist genau das, was wir gemäß Tabelle 1 für unseren Schrittmotor brauchen!

Noch ein Wort zum Leistungsteil der Schaltung. Der Motor muß aus verschiedenen Gründen (es sind meist 12 V und größere Ströme nötig, Störungen und gefährliche Spannungsspitzen für den BASIC-Tiger® können auftreten usw.) aus einer separaten Spannungsquelle versorgt werden. Die 4 npn-Transistoren sollten im Schaltbetrieb kaum warm werden und brauchen wahrscheinlich keine Kühlmaßnahmen. Etwas kritischer ist die Stromquelle mit dem pnp-Transistor. In der Erprobungsphase verbinden Sie einfach den Pluspol der 12V – Quelle mit den Anschlüssen 5 und 6 des Motors (Betrieb also ohne Stromquelle). Messen Sie den Strom Ihres Motors im normalen Betrieb mit einem Strommesser. Der Widerstand R_c sollte nun so dimensioniert werden, daß beim Kurzschluß des Kollektors gegen Masse ein Strom fließt, der etwas größer ist als der, der sich im normalen Betrieb des Motors einstellt. Ein Richtwert für den Anfang sind vielleicht 2 bis 5 Ω / 2 Watt. Kontrollieren Sie dann speziell bei langem Stillstand des Motors die Temperatur des Transistors und legen Sie dementsprechend dessen Kühlkörper aus (wenn nötig). Die Schutzdioden an den Wicklungen sollen deren Selbstinduktionsspannungen ableiten und so die Transistoren schützen. Verwenden Sie hier entsprechend schnelle und leistungsfähige Typen!

Dem Anschluß an den BASIC-Tiger® steht nun nichts mehr im Wege. Eine Portleitung steuert den TU-Anschluß (Pin 5) des 74HCT193, eine andere den TD-Anschluß an. Der Anschluß an einen der erweiterten Ausgangsports des Plug-and-Play-Labs (hier Portadresse 10h, Pin 0 und Pin 1) brachte das erwartete Resultat. Diese Ausgänge werden von normalen Logikbausteinen getrieben. Dies ist auch der Grund, weshalb das nachfolgende Programm für diese Ports geschrieben wurde.

Wenn die hier vorgestellte Eigenbau-Hardware nichts für Sie oder Ihren Antrieb ist – es gibt natürlich auch fertige Lösungen. Neben ähnlichen Interfaceschaltungen gibt es auch spezielle IC's die Ihnen praktisch alle Arbeit abnehmen. Allerdings ist es für einen Entwickler immer wieder reizvoll, eine selbst aufgebaute Schaltung vollständig zu verstehen und auszuprobieren.

3. Software

Die Anwendungsmöglichkeiten von Schrittmotoren sind vielfältig. Es hat daher wenig Zweck, ein universelles Programm schreiben zu wollen. Das vorgestellte Programm STEPPER1.TIG besteht in der Hauptsache aus zwei gleichartigen Unterprogrammen, die beim Unterprogrammaufruf die in den Variablen VSchritte bzw. RSchritte übergebene Schrittzahl realisieren. Die genannten Variablen werden hier mit 36*100 (das sind 100 Umdrehungen mit jeweils 36 Schritten pro Umdrehung) vorgegeben. Natürlich können Sie in einem eigenen Programm diese Variablen beliebig einsetzen oder ausrechnen lassen. Mit einer fest eingestellten Variablen können Sie allerdings Ihren Motor ausgezeichnet testen, wenn Sie dessen Schrittzahl pro Umdrehung kennen. Nach Ausführung von vielen Umdrehungen sollte er dann immer an der gleichen Position stehenbleiben.

Sehr wichtig sind die Teile des Programmes, die den weichen Start und das langsame Abbremsen des Motors gewährleisten. Hier können Sie ein wenig experimentieren. Die Variable M in den Unterprogrammen sorgt dafür, daß zwischen den Impulsen Pausenzeiten eingebaut werden (hier anfänglich 12 ms). Diese werden dann bei den ersten Impulsen von Schritt zu Schritt weiter verkürzt, bis sich die Maximaldrehzahl einstellt. Ähnliches gilt für das Abbremsen. Hier wird bei den letzten Schritten eine immer größer werdende Pausenzeit eingefügt. Also, probieren Sie alles aus – viel Spaß!

```
-----
' Name: STEPPER1.TIG
' Zweck: Steuert einen kleinen unipolaren Steppermotor mit einer
' zustzlichen Hardware an.
' Mit den Variablen "VSchritte" und "RSchritte" wird die Anzahl der
' Schritte vorwaerts und rueckwaerts vorgegeben. Ein langsamer Anlauf
' und ein weiches Abbremsen wird durch entsprechend ausgerechnete
' Wartezeiten erreicht. Die fuer den Anlauf notwendigen Wartezeiten
' sowie die maximale Schrittfrequenz richten sich nach dem verwendeten
' Motor und muessen ggf. ausprobiert werden!
'
' Anschlu der Hardware:
' Eingang vorwaerts an erweiteren Port 10h Bit 0
' Eingang rueckwaerts an erweiteren Port 10h Bit 1
' Anschluss Vcc an Vcc des Plug-and-Play-Lab
' Anschluss GND an GND des Plug-and-Play-Lab
' GND(0V) und Vdd(+12V) an ein externes Netzteil 12V/1A
-----
TASK MAIN                                     ' Beginn Task MAIN
INSTALL DEVICE #1, "LCD1.TDD"                 ' LC-Display installieren
PRINT #1, "<1>Schrittmotor-Demo";           ' Text ausgeben
WAIT_DURATION 3000                            ' 3 Sekunden warten

        OUT 10h,00000010b,255                 ' die erweiterten Ausgaenge
        OUT 10h,00000001b,255                 ' 10h Bits 0 und 1 auf 1 setzen

VSchritte = 36*100                            ' Schritte vorwaerts und rueck-
RSchritte = 36*100                            ' waerts vorbelegen (hier 100
                                                ' Umdrehungen zu 36 Schritten)

Anfang:                                       ' Marke Anfang
CALL V (VSchritte)                           ' Unterprogramm Vorwaertsschritte
WAIT_DURATION 1000                           ' danach 1 Sekunde warten

CALL R (RSchritte)                           ' Unterprogramm Rueckwaertsschritte
WAIT_DURATION 1000                           ' danach 1 Sekunde warten

GOTO Anfang                                  ' alles wiederholen

END                                           ' Ende Task MAIN

-----
' Unterprogramm Schritte vorwaerts ausfuehren
-----
SUB V (VSchritte)
Print #1, "<1Bh>A";CHR$(0);CHR$(1)&          ' Ausgeben der Schrittzahl
;"<F0h>"; "Schritte rechts:",VSchritte     ' vorwaerts (rechts) auf LC-Display
M = 12                                       ' Wartezeit fuer langsamen Anlauf
FOR N = 1 TO VSchritte                       ' alle Vorwaertsschritte ausgeben
    OUT 10h,00000001b,0                       ' (Bit 0 macht L-Impuls, Bit 1 auf H)
    IF N < 10 THEN                            ' bei den ersten 10 Impulsen abneh-
        M = M - 1                              ' mende Wartezeit einstellen
    ENDIF                                     ' (langsamer Anlauf)
    IF N > (VSchritte - 10) THEN              ' bei den letzten Impulsen wieder
        M = M + 1                              ' zunehmende Wartezeiten einstellen
    ENDIF                                     ' (bremsen)
    WAIT_DURATION M                           ' entsprechende Wartezeit
    OUT 10h,00000001b,255                     ' Bit 0 geht wieder auf H
    WAIT_DURATION 1                           ' etwas warten
NEXT
END

-----
' Unterprogramm Schritte rueckwaerts ausfuehren
-----
SUB R (RSchritte)
Print #1, "<1Bh>A";CHR$(0);CHR$(1)&          ' Ausgeben der Schrittzahl
```

```
;"<F0h>"; "Schritte links:",RSchritte      ' rueckwaerts (links) auf LC-Display
M = 12                                       ' Wartezeit fuer langsamen Anlauf
FOR N = 1 TO RSchritte                       ' alle Rueckwaertsschritte ausgeben
  OUT 10h,00000010b,0                         ' (Bit 1 macht L-Impuls, Bit 0 auf H)
  IF N < 10 THEN                             ' bei den ersten 10 Impulsen abneh-
    M = M - 1                                 ' mende Wartezeit einstellen
  ENDIF                                       ' (langsamer Anlauf)
  IF N > (RSchritte - 10) THEN               ' bei den letzten Impulsen wieder
    M = M + 1                                 ' zunehmende Wartezeiten einstellen
  ENDIF                                       ' (bremsen)
  WAIT_DURATION M                             ' entsprechende Wartezeit
  OUT 10h,00000010b,255                       ' Bit 1 geht wieder auf H
  WAIT_DURATION 1                             ' etwas warten
NEXT                                          '
END                                          ' Ende Unterprogramm
'-----
```