Frequenzbogen-Steuerung für den ATmega

Die Fragestellung entstand bei dem Versuch, einen Schrittmotor bestimmte Positionen anfahren zu lassen, wobei die Beschleunigungs- und Bremsphase mit einer Tabelle für die Schrittfrequenzen gesteuert werden soll. Die Tabelle ist so aufgebaut, dass bei der Bewegung keine starken Beschleunigungsspitzen vorkommen. Deshalb soll die Rampe Priorität vor dem Erreichen der Sollpositiion haben. Das kann leicht zum „Nachruckeln" in unmittelbarer Nähe des Sollortes führen - was natürlich vermieden werden soll. Lieber soll der Motorüber die paar letzten Schritte in die Zielposition „hineinkriechen", als Hin- und Herruckeln oder schlagartig aus höherer Geschwindigkeit stehenbleiben - das ist die Vorgabe.

Frequenztabelle

Wiederkehrende Taktimpulse mit einstellbarer Frequenz können mit MCs nur mit Timern oder Zählerschleifen erzeugt werden. Dabei kann nur die Zeitdauer zwischen zwei Impulsen (die Periode) eingestellt werden, nicht die Frequenz. Man kann auch nicht jede beliebige Frequenz einstellen, sondern nur Bruchteile des MC-Taktes. Dadurch wird aus der Rampe eine Frequenzrampe. Jede Stufe dieser Treppe ist durch eine Periode gekennzeichnet und durch die Angabe, wie oft Takte mit dieser Periode ausgegeben werden soll; diese Werte heissen im folgenden Text „Periode" und „Wiederholungen". In diesem Beispiel werden die Perioden als Anzahl Timer-Interrupts zwischen zweit Taktimpulsen dargestellt.

Beispiel:

```
Timer-Interrupts       +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+
                       | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
                   ----+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-+ +-

Taktimpulse
                       +----+          +----+          +----+          +----+
                       |    |          |    |          |    |          |    |
                   ----+    +----------+    +----------+    +----------+    +-----------------

                       |<-  Periode  ->|
                        Periode = 4
                        Wiederholungen = 3
```

Die Taktimpulse werden mit der Prozedur „FREQBOGEN_TAKT" erzeugt. Sie muss bei jedem Timerinterrupt aufgerufen werden.

Die Perioden und Wiederholungen der einzelnen Stufen der „Frequenztreppe" sind in einer Tabelle zusammengefasst. Jeder Eintrag umfasst 4 Byte. Ein Tabellenzeiger zeigt jeweils auf den aktuellen Eintrag. Der Tabellenzeiger setzt sich aus dem Zeiger auf den Tabellenanfang und einem Index zusammen. Zum ersten Tabelleneintrag gehört der Index Null.

Beispiel:

Index	Periode (Anzahl Timer-interrupts)	Wiederholungen (Anzahl)
0	0x0200	0x0007
1	0x01D8	0x0007
2	0x01C0	0x0007
3	0x01A8	0x0007
4	0x0198	0x0007
...	...	...
52	0x0038	0x04D7

Eine Schwierigkeit besteht darin, rechtzeitig vor dem Erreichen der gewünschten Impulszahl mit dem Bremsen anzufangen. Nun kann man aber zu jedem Wert des Tabellenzeigers leicht zusammenzählen, wieviele Impulse er bisher ausgegeben hat. Logischerweise braucht er mindestens dieselbe Anzahl, um wieder zum Rampenanfang zurückzukehren. Diese Anzahl Impulse nenne ich die „Rampendistanz" (RAMPDIST). Natürlich könnte man sie durch einfache Addition laufend mitberechnen, es ist aber noch einfacher, sie vorher auszurechnen und in der Rampentabelle abzulegen. Dadurch hat jeder Eintrag in die Rampentabelle nun drei Werte: Die Periode, die Wiederholungen und die Rampendistanz.

Beispiel:

Index	Periode (Anzahl Timer-interrupts)	Wiederholungen (Anzahl Takte)	Rampendistanz (Anzahl Takte)
0	0x0200	0x0007	0x0007

1	0x01D8	0x0007	0x000E
2	0x01C0	0x0007	0x0015
3	0x01A8	0x0007	0x001C
4	0x0198	0x0007	0x0023
...	...	...	...
52	0x0038	0x04D7	0x1420

Mit so einer Frequenztabelle (die nun pro Eintrag 6 Byte umfasst) kann man zwei Betriebsarten verwirklichen: Die Frequenzrampe und den Frequenzbogen. Die Frequenzrampe ist im Grunde ein Frequenzfolge-Regler, der den Änderungen der Sollfrequenz rampenartig folgt. Im Gegensatz dazu erzeugt der Frequenzbogen eine vorbestimmte Anzahl von Taktimpulsen. Als zusätzliche Anforderung soll dabei die Frequenz der Taktimpulse nicht konstant bleiben, sondern, wie ein Brückenbogen, allmählich auf- und zum Ende hin absteigen. Der Frequenzgang beginnt immer mit der niedrigsten Frequenz (Bogenfuss) und muss, möglichst gleichzeitig mit dem Erreichen der gewünschten Anzahl Takte, auch dorthin wieder zurückgekehrt sein. Aber das Wesentliche bleibt immer die Anzahl Takte die geliefert werden soll.

Die einzige Struktur, die für beide Betriebsarten nutzbar ist, ist die Frequenztabelle.

Im folgenden geht es ausschliesslich um den Frequenzbogen, weil er für die Anwendungen im Tarzanplotter und in CNC-Maschinen ein zentrales Element ist.

Frequenzbogen

Wenn man die Bewegungen mehrerer Schrittmotoren zu koordinieren hat, dann sieht die allgemeinste Konstruktion so aus: Es gibt den Frequenzbogen als Taktgenerator, der soviele Takte erzeugt, wie der Schrittmotor mit dem längsten Weg braucht. Aus der Taktfolge die er erzeugt, leitet man auch die Taktfolgen für die anderen Motoren ab, indem man die Takte mit Hilfe eines leicht abgewandelten Bresenham-Algorithmus genau so herunterteilt, dass am Ende des Frequenzbogens beide Motoren die gewünschte Anzahl Schritte zurückgelegt hat. Der Motor, der die meisten Schritte bis zum Ziel zurückzulegen hat, wird als Führungsmotor bezeichnet, alle anderen sind Folgemotoren.

Der Frequenzbogen ist also der zentrale Takterzeuger. Sein Frequenzprofil spiegelt sich in den Bewegungen aller Motoren wider. Er braucht über die Drehrichtung usw. der Schrittmotoren gar nichts zu wissen - seine einzigsten Parameter sind die vorbestimmte Anzahl Takte (Soll-Taktanzahl), die er liefern soll und die Tabelle der Frequenzen, die er dabei durchlaufen soll. Die Tabelle der Kriterien, die den Zustand des Automaten steuern, ist:

Fall	Name des Bits im Zustandswort	Nr. des Bits im Zustandswort
Bogen befindet sich zwischen minimaler und maximaler Frequenz/bei der minimalen Frequenz der Tabelle	FQBG_IDX_NULL	0
Bogen befindet sich zwischen minimaler und maximaler Frequenz/bei der maximalen Frequenz der Tabelle	FQBG_IDX_MAX	1
voraussichtliche Gesamt-Taktzahl ist ungleich/gleich der vorbestimmten Soll-Taktanzahl	FQBG_TBW_NULL	2
die Gesamt-Taktanzahl bis zur Rückkehr zum Bogenfuss ist gleich/grösser als die Soll-Taktanzahl	FQBG_TZL_PLUS	3
Taktgenerator soll weiterlaufen/zwangsweise anhalten	FQBG_HALT	4
die aktuelle Taktperiode muss noch nochmal/nicht mehr wiederholt werden	FQBG_WDH_NULL	5

In dieser Tabelle entspricht der Zustand links des Schrägstrichs immer dem Bitwert „0", der rechts davon dem Bitwert „1". Die Abkürzung „FQBG" steht für „Frequenzbogen"; das Kürzel „TBW" soll „Taktabweichung", „TZL" „Taktziel" bedeuten.

Das Bit FQBG_IDX_NULL enthält die „0", wenn der Tabellenzeiger, der die aktuell ausgegebene Taktfrequenz anzeigt, nicht den Wert „0" hat. Es enthält eine „1", wenn der Tabellenzeiger auf den ersten Eintrag der Tabelle zeigt (d.h. den Wert „0" hat). Dieser Eintrag enthält die minimale Frequenz, die ausgegeben werden kann (Bogenfuss). Das Bit FQBG_IDX_MAX ist gleich „0", wenn der Tabellenzeiger auf irgendeinen Eintrag vom Ersten bis zum Vorletzten der Tabelle zeigt. Es ist gleich „1", wenn er auf den letzten Eintrag zeigt. Dieser Eintrag enthält die maximale Frequenz, die überhaupt ausgegeben werden kann.

Das Bit FQBG_TBW_NULL hat den Wert „0", wenn die gelieferte Taktanzahl ungleich der vorbestimmten Soll-Taktanzahl ist und den Wert „1", wenn die Soll-Taktanzahl erreicht ist. Unter „Taktziel" verstehe ich die Summe aus der bereits gelieferten Taktanzahl und der Anzahl, die bis zur Rückkehr an den Tabellenanfang (Bogenfuss) noch geliefert werden wird. Die Anzahl der Takte, die bis zum Tabellenanfang anfällt ist natürlich immer gleich der Summe der Takte, die seit dem Aufbruch vom Bogenfuss bereits ausgegeben wurden. Man könnte also leicht nebenher aufsummieren. Es ist aber einfacher und weniger fehlerträchtig, den Wert einfach in den Tabelleneintrag mit aufzunehmen. Er wird im Folgenden als „Taktsumme" bezeichnet.

Das Bit FQBG_TZL_PLUS hat den Wert „0", wenn das Taktziel kleiner oder gleich der Soll-Taktanzahl ist. Es hat den Wert „1", wenn es die Soll-Taktanzahl überschreitet.

Das Bit FQBG_WDH_NULL hat den Wert „0", wenn noch Wiederholungen der aktuellen Taktperiode ausstehen. Sie hat den Wert „1", wenn alle Wiederholungen abgearbeiet sind, d.h. der Wiederholungszähler steht auf Null.

Die Zustandstabelle kommt also mit 64 Zuständen aus, von denen aber nicht alle vorkommen: Die Zustände, bei denen das Bit FQBG_IDX_MAX und FQBG_IDX_NULL gleichzeitig gesetzt sind, gibt es in Wirklichkeit nicht. Sie tauchen in der Tabelle trotzdem auf, weil der Index alle Zahlen von 0 bis IDX_MAX durchlaufen können muss. Da müssen in der Tabelle auch für die Indexwerte Einträge reserviert werden, die es gar nicht geben kann.

Im Grunde könnte man die einzelnen Bits auch per „IF..THEN..ELSE"-Abfragen auswerten. Vermutlich würde man damit sogar ein paar Zyklen Rechenzeit sparen können. Ich bevorzuge die Tabellenmethode, in der jeder Kombination von Bits (auch denen die gar nicht vorkommen können) eine Reaktion zugeordnet ist (es ist eigentlich die Karnaugh-Tabelle). Erstens kann man sie sehr einfach in einer Tabellenkalkulation erzeugen und die Bits in der gewünschten Reihenfolge den Binärstellen zuordnen. Zweitens kann man dort auch gleich den fertigen Tabellentext erzeugen und einfach in das Programm kopieren. Und drittens braucht man bei Änderungen nicht jedesmal mühsam durch den „IF..THEN..ELSE"-Baum klettern, um die Stelle zu finden, an der man die Änderung anbringen muss. Es gibt also erhebliche Vorteile beim Vermeiden von Fehlern. Interessant ist auch die Tatsache, dass man bei dieser Art der Auswertung immer sicher sein kann, dass alle Fälle berücksichtigt sind. Bei der „IF..THEN..ELSE"-Lösung übersieht man leicht einen.

Es kann aber immer noch sein, dass man wegen der kurzen Zeitspanne, die zwischen zwei Timer-Interrupts zur Verfügung steht, doch zu der schnelleren „IF..THEN..ELSE"-Lösung übergehen muss. Dann sollte man aber durch die verschiedenen Bitkombinationen gehen und die Ergebnisse jeweils mit der Reaktion vergleichen, die die Tabelle für diesen Fall vorgesehen hätte. Als Anhaltspunkt: Auf einem ATmega16 und mit 32-Bit-Werten für Soll- und Istwert beansprucht die Tabellenvariante maximal 230 MC-Takte.

Die Tabelle ist hier wegen ihrer Länge in zwei Hälften geteilt: Die erste ist die langweiligere, weil sie nur die Fälle umfasst, in denen der Wiederholungszähler noch nicht abgelaufen ist. In der zweiten, die die Fälle auflistet, in denen der Wiederholgungszähler abgelaufen ist, geht es lebhafter zu.

Die Tabelle ordnet jeder Bit-Kombination eine Reaktion zu.

Zustandswort							Reaktion	
Zst. Nr.	FQBG_ WDH_NULL	FQBG_ HALT	FQBG_ IDX_MAX	FQBG_ IDX_NULL	FQBG_ TBW_NULL	FQBG_ TZL_PLUS	(Freq.- Änderg.)	
0	0	0	0	0	0	0	WDHOL	
1	0	0	0	0	0	1	WDHOL	
2	0	0	0	0	1	0	HALT	
3	0	0	0	0	1	1	HALT	
4	0	0	0	1	0	0	START	
5	0	0	0	1	0	1	WDHOL	
6	0	0	0	1	1	0	HALT	
7	0	0	0	1	1	1	HALT	
8	0	0	1	0	0	0	WDHOL	
9	0	0	1	0	0	1	WDHOL	
10	0	0	1	0	1	0	HALT	
11	0	0	1	0	1	1	HALT	
12 -	0	0	1	1	0	0	-	gibt's nicht
13 -	0	0	1	1	0	1	-	gibt's nicht
14 -	0	0	1	1	1	0	-	gibt's nicht
15 -	0	0	1	1	1	1	-	gibt's nicht
16	0	1	0	0	0	0	WDHOL	
17	0	1	0	0	0	1	WDHOL	
18	0	1	0	0	1	0	HALT	
19	0	1	0	0	1	1	HALT	
20	0	1	0	1	0	0	HALT	
21	0	1	0	1	0	1	HALT	
22	0	1	0	1	1	0	HALT	
23	0	1	0	1	1	1	WDHOL	
24	0	1	1	0	0	0	WDHOL	
25	0	1	1	0	0	1	HALT	
26	0	1	1	0	1	0	HALT	
27	0	1	1	0	1	1	HALT	
28 -	0	1	1	1	0	0	-	gibt's nicht
29 -	0	1	1	1	0	1	-	gibt's nicht
30 -	0	1	1	1	1	0	-	gibt's nicht
31 -	0	1	1	1	1	1	-	gibt's nicht

Zustandswort							Reaktion	
Zst. Nr.	FQBG_ WDH_NULL	FQBG_ HALT	FQBG_ IDX_MAX	FQBG_ IDX_NULL	FQBG_ TBW_NULL	FQBG_ TZL_PLUS	(Freq.- Änderg.)	
32	1	0	0	0	0	0	AUFW	
33	1	0	0	0	0	1	ABW	
34	1	0	0	0	1	0	HALT	
35	1	0	0	0	1	1	HALT	
36	1	0	0	1	0	0	START	
37	1	0	0	1	0	1	AUFW	
38	1	0	0	1	1	0	HALT	
39	1	0	0	1	1	1	HALT	
40	1	0	1	0	0	0	KONST	
41	1	0	1	0	0	1	ABW	
42	1	0	1	0	1	0	HALT	
43	1	0	1	0	1	1	HALT	
44 -	1	0	1	1	0	0	-	gibt's nicht
45 -	1	0	1	1	0	1	-	gibt's nicht
46 -	1	0	1	1	1	0	-	gibt's nicht
47 -	1	0	1	1	1	1	-	gibt's nicht
48	1	1	0	0	0	0	ABW	
49	1	1	0	0	0	1	ABW	
50	1	1	0	0	1	0	HALT	
51	1	1	0	0	1	1	HALT	
52	1	1	0	1	0	0	HALT	
53	1	1	0	1	0	1	HALT	
54	1	1	0	1	1	0	HALT	
55	1	1	0	1	1	1	HALT	
56	1	1	1	0	0	0	ABW	
57	1	1	1	0	0	1	ABW	
58	1	1	1	0	1	0	HALT	
59	1	1	1	0	1	1	HALT	
60 -	1	1	1	1	0	0	-	gibt's nicht
61 -	1	1	1	1	0	1	-	gibt's nicht
62 -	1	1	1	1	1	0	-	gibt's nicht
63 -	1	1	1	1	1	1	-	gibt's nicht

Die Reaktionen sind jeweils kleine Prozeduren. Die Tabelle enthält jeweils den Zeiger auf Prozeduranfang. Diese Prozeduren führen folgende Tätigkeiten aus:

Prozedur HALT:
1. Tabellenzeiger auf Null setzen
2. den Wert für Periode der Taktfrequenz aus der Tabelle einlesen,
   den Periodenzähler mit neuem Wert anfangsetzen
3. den Wiederholungszähler auf Null setzen
4. Taktsumme auf Null setzen
5. Taktzähler unverändert lassen
6. kein Taktsignal ausgeben
7. Haltsignal ausgeben

Prozedur START:
1. Tabellenzeiger auf Eins setzen
2. den Wert für Periode der Taktfrequenz aus der Tabelle einlesen,
   den Periodenzähler mit neuem Wert anfangsetzen
3. die Anzahl der Wiederholungen aus der Tabelle einlesen,
   den Wiederholungszähler auf Null setzen
4. die Taktsumme aus der Tabelle einlesen und speichern
5. Taktzähler mit Eins anfangsetzen
6. Taktsignal ausgeben
7. kein Haltsignal ausgeben

Prozedur AUFW:
1. Tabellenzeiger um Eins aufzählen
2. den Wert für Periode der Taktfrequenz aus der Tabelle einlesen,
   den Periodenzähler mit neuem Wert anfangsetzen
3. die Anzahl der Wiederholungen aus der Tabelle einlesen,
   den Wiederholungszähler auf Null setzen
4. die Taktsumme aus der Tabelle einlesen und speichern

5. Taktzähler um Eins aufzählen
6. Taktsignal ausgeben
7. kein Haltsignal ausgeben

Prozedur ABW:
1. Tabellenzeiger um Eins abzählen
2. den Wert für Periode der Taktfrequenz aus der Tabelle einlesen,
   den Periodenzähler mit neuem Wert anfangsetzen
3. die Anzahl der Wiederholungen aus der Tabelle einlesen,
   den Wiederholungszähler auf Null setzen
4. die Taktsumme aus der Tabelle einlesen und speichern
5. Taktzähler um Eins aufzählen
6. Taktsignal ausgeben
7. kein Haltsignal ausgeben

Prozedur KONST:
1. Tabellenzeiger unverändert lassen
2. den Wert für Periode der Taktfrequenz aus der Tabelle einlesen,
   den Periodenzähler mit neuem Wert anfangsetzen
3. die Anzahl der Wiederholungen aus der Tabelle einlesen,
   den Wiederholungszähler auf Null setzen
4. die Taktsumme aus der Tabelle einlesen und speichern
5. Taktzähler um Eins aufzählen
6. Taktsignal ausgeben
7. kein Haltsignal ausgeben

Prozedur WDHOL:
1. Tabellenzeiger unverändert lassen
2. den Periodenzähler nochmals mit dem aktuellen Wert für Periode
   der Taktfrequenz anfangsetzen
3. den Wiederholungszähler um Eins abzählen
4. die Taktsumme bleibt unverändert
5. Taktzähler um Eins aufzählen
6. Taktsignal ausgeben
7. kein Haltsignal ausgeben

Es fällt auf, dass die Reaktionsprozeduren die Bits, die in die Berechnung der Reaktion eingehen, nur indirekt (z.B. über den Wert des Tabellenzeigers) beeinflussen. Sie beschäftigen sich ausschliesslich mit der Frequenzerzeugung und dem Mitzählen der ausgebenen Takte. Diese Trennung von Tabellen- und Ausgabeverwaltung macht diese Programmkonstruktion ziemlich robust und wenig fehleranfällig.

Um den Frequenzbogen vom Hauptprogramm aus bedienen zu können, ohne in seinen inneren Strukturen herumfummeln zu müssen, empfiehlt es sich, eine Reihe von zusätzlichen Prozeduren bereitzustellen, die die notwendigen Manipulationen ausführen. Dies sind:

Prozedur CREATE:
Sie richtet in einem reservierten RAM-Bereich die Variablen den Frequenzbogen ein. Dabei wird der Tabellenzeiger mit Null anfanggesetzt und der Frequenzgenerator desaktiviert und das HALT-Flag gesetzt. Der Periodenzähler wird mit dem Wert der Periode aus dem Eintrag Nr. 0 der Frequenztabelle anfanggesetzt.

Diese Prozedur dient zum Einrichten des Moduls und muss ganz am Anfang des Hauptprogramms aufgerufen werden. Die Funktionen des Frequenzbogens/Frequenzgenerators stehen erst danach zur Verfügung.

Prozedur SET_AKTIV:
Sie aktiviert den Frequenzgenerator, so dass ab jetzt Taktsignale ausgegeben werden können; das HALT-Flag wird gelöscht. Der Taktzähler und der Tabellenzeiger werden mit Null anfanggesetzt.

Mit dieser Prozedur lässt sich der Frequenzgenerator einschalten. Man braucht sie eigentlich nicht, weil es praktischer ist, den Frequenzgenerator gleichzeitig mit dem Einstellen der Soll-Taktanzahl einzuschalten.

Prozedur SET_INAKTIV:
Sie desaktiviert den Frequenzgenerator, so dass keine Taktsignale mehr ausgegeben werden; das HALT-Flag wird gesetzt.

Der Aufruf dieser Funktion ist so etwas wie ein NOT-AUS. Wenn der Frequenzgenerator in Betrieb war, dann fällt die Frequenz schlagartig auf Null.

Funktion ISTAKTIV:
Diese Funktion gibt ein Bit zurück, das anzeigt, ob der Frequenzbogen aktiv ist oder nicht.

Es ist manchmal praktisch, wenn das Hauptprogramm nachfragen kann, ob der Frequenzgenerator noch läuft.

Prozedur SET_HALT:
Sie setzt das HALT-Flag, was bewirkt, dass die Frequenz sofort entlang des Frequenzbogens heruntergefahren wird. Dabei wird keine Rücksicht auf das Erreichen der Soll-Taktanzahl genommen.

Dies gibt dem Hauptprogramm die Möglichkeit, die Frequenz sachte, aber bestimmt auf Null zu fahren.

Prozedur SET_SOLL:
Diese Prozedur muss die Soll-Taktanzahl als Parameter mitbringen. Die Soll-Taktanzahl wird im RAM gespeichert und die Prozedur SET_AKTIV aufgerufen. Dadurch beginnt der Frequenzgenerator sofort nach diesem Aufruf zu arbeiten.

Der Aufruf dieser Prozedur schaltet den Frequenzgenerator auch ein.

Prozedur GET_SOLL:
Diese Funktion liefert die Soll-Taktanzahl an das aufrufenden Programm zurück.

Dasu aufrufende Programm braucht manchmal die aktuelle Soll-Taktanzahl, z.B. um angeschlossene Bresenham-Frequenzgeneratoren zu bedienen (Aufruf von BRESFREQ_STEP).

Funktion GET_TAKTANZAHL:
Diese Funktion gibt den aktuellen Stand des Taktzählers an das aufrufende Programm zurück.

Eigentlich braucht man diese Funktion nicht, denn die intern mitgeführte Taktanzahl ist immer positiv. Im Hauptprogramm, wo der Frequenzgenerator beispielsweise einen Schrittmotor steuert, der vorwärts und rückwärts laufen kann, ist es viel praktischer, selbst die Takte (=Schritte) mitzuzählen; und zwar in positivem wie auch mit negativem Vorzeichen. Es kann aber auch Spezialfälle geben, wo die positive Taktanzahl ausreicht. Für diese Fälle ist diese Prozedur gedacht.

Mit diesen zusätzlichen Prozeduren bildet der Frequenzbogen nun ein eigenständiges Modul, das zweckmässigerweise als separate Datei ins Hauptprogramm eingebunden wird.

Bresenham-Frequenzgenerator

Beim Tarzanplotter, aber auch bei jeder CNC-Maschine und jedem RepRap-Drucker, müssen Schrittmotoren koordiniert gesteuert werden. „Koordiniert" bedeutet dabei, dass alle Motoren beim Erreichen der Zielkoordinaten gleichzeitig jeder eine andere, genau festgelegte Anzahl Schritte zurückgelegt haben muss. Wie macht man sowas?

Im Grunde ist es dasselbe Problem, wie man es beim Plotten einer Geraden auch hat: Während der Cursor die Pixel entlang der x-Achse durchläuft, muss er gleichzeitig auch die Pixel in Richtung der y-Achse durchgehen. Der Bresenham-Algorithmus besorgt genau das, wobei das Besondere ist, dass er, obwohl er nur ganze Zahlen (Integer) verwendet, jede beliebige positive Steigung (ausgedrückt als Anzahl y-Pixel dividiert durch die Anzahl x-Pixel) darstellen kann, solange sie nicht grösser als Eins ist: Die Anzahl y-Pixel pro x-Pixel muss nicht ganzzahlig sein.

Wenn man in Gedanken, die Pixel durch Schrittmotor-Schritte ersetzt, dann hat man die Lösung des Problems koordinierter Bewegungen schon gelöst.

Die Bedingung „Steigung mindestens gleich Null, aber nicht grösser als Eins" kann man einfach so ausdrücken: Die die Anzahl Schritte entlang der x-Achse darf nicht kleiner sein, als die der Schritte entlang der y-Achse. Wenn das doch der Fall sein sollte, dann wird einfach die x-Achse zur y-Achse erklärt und die y-Achse zur x-Achse. Es ist deshalb einfacher, nicht nach x- und y-Achse zu unterscheiden, sondern nach dem Motor mit der grösseren Schritt-Anzahl und dem mit der kleineren. Der Motor mit der grössten Anzahl Schritte nenne ich deshalb den „Führungsmotor", der andere ist ein „Folgemotor". Der Führungsmotor braucht also die meisten Takte, der Folgemotor maximal genausoviele.

Der Einfachheit halber tun wir jetzt so, als bräuchte ein Schrittmotor für jeden Schritt genau einen Taktimpuls (in Wirklichkeit sind es, je nach Betriebsart Voll-, Halb- oder Mikroschritt mehr als einer). Der Taktzähler jeden Motors soll zu Anfang der Bewegung auf Null stehen. Die gewünschte Anzahl Schritte jedes Motors heisst „Spanne"; sie ist gleichgross wie die gewünschte Anzahl Takte. Die Soll-Taktzahl des Führungsmotors heisst „Führungsspanne".

Der Bresenham-Algorithmus wird nun folgendermassen eingesetzt:

```
      Taktzähler := 0
      Kriterium := 0
      Entscheidung := falsch
       |
 +---->+
 |     |
 |    Taktzähler < Spanne?
 |    ja   nein
```

```
|     |     |
|     |     +---------------- Ziel erreicht, keine Takte mehr ausgeben
|     |
|    Kriterium := Kriterium + 2*Spanne
|    Entscheidung = wahr?
|    ja   nein
|     |     |
|     |     +---------------- Kriterium := Kriterium - 2*Führungsspanne
|     |                       |
|     +<-----------------------+
|     |
|    Kriterium > 0 ?
|    ja    nein
|     |     |
|     |     +---------------- Entscheidung := falsch
|     |                       |
|    Entscheidung := wahr     |
|    Takt ausgeben            |
|     |                       |
+-----+<-----------------------+
```

Jedesmal, wenn die „Entscheidung wahr" ist, wird ein Takt ausgegeben. Beim Führungsmotor ist die Spanne gleich der Führungsspanne; deshalb liefert er eine ununterbrochene Folge von Takten, bis die Taktzahl die Führungsspanne erreicht hat. Beim Folgemotor werden, je nach Wert der Folgespanne, weniger (maximal gleichviele) Takte ausgegeben. Der Algorithmus sorgt aber dafür, dass die Intervalle zwischen den Takten möglichst wenig schwanken.

Ausgeführt wird dieser Algorithmus bei jedem Takt eines Taktgebers. Er bestimmt die Reisedrehzahl der Motoren. Man kann dafür z.B. einen normalen Timer-Interrupt nehmen. Dann springt der Führungsmotor mit der vollen Reisedrehzahl an und bleibt am Ende der Bewegung auch aus voller Fahrt stehen. Die Drehzahl des Folgemotors bleibt entsprechend geringer. Aber auch er springt sofort mit Reisedrehzahl an und hält auch schlagartig an.

Um die ruckartigen Bewegungen zu vermeiden, setzt man statt der festen Frequenz eines Timer-Interrupts die des Frequenzbogens ein. Als Soll-Taktanzahl des Frequenzbogens wird die Führungsspanne eingestellt. Bei jedem Takt, den der Frequenzbogen ausgibt, werden für Führungs- und Folgemotor je ein Schritt des Bresenham-Algorithmus ausgeführt. Die Motoren werden mit dem Takt gespeist, den der jeweils zugeordnete Bresenham-Algorithmus liefert. So führt die auf- und absteigende Frequenz des Frequenzbogens zu einem sanften Anfahren und Auslaufen beider Motoren.

Die Programmstruktur sieht dann so aus (das Diagramm zeigt den Fluss der Taktimpulse)

```
     Timer-Interrupt
      |
      +--> Frequenzbogen
           (Soll-Taktanzahl = Führungsspanne)
            |
            +--> Bresenham-Frequenzgenerator --> Führungsmotor
            |    (Spanne = Führungsspanne)
            |
            +--> Bresenham-Frequenzgenerator --> Folgemotor
                 (Spanne = Folgespanne)
```

Auf einem ATmega16 braucht ein Durchlauf durch diese Variante des Bresenham-Algorithmus mit 32-Bit Integer Zahlen für das Kriterium maximal 80 MC-Zyklen. Dabei ist die Abfrage nach dem Erreichen der Spanne (die oben im Ablaufdiagramm noch dargestellt ist) weggelassen: Der Frequenzbogen hört ja auf, Takte zu liefern, wenn er seine Soll-Taktanzahl (sie ist gleich der Führungsspanne) erreicht hat.

Es ist klar, dass sich die Anzahl der Folgemotoren erweitern lässt. Die Gesamtrechenzeit für den Frequenzbogen und die Bresenham-Algorithmen darf nicht länger sein, als die Zeit zwischen zwei Timer-Interrupts; vorausgesetzt, der Rechner hat nichts anderes zu tun, als die Motoren zu steuern.

Angenommen, es sind zwei Motoren zu steuern. Sie sollen sich in einer Stellung befinden, die der Punkt (x0|y0) entspricht. Um sie einen Punkt (x|y) anzufahren zu lassen, geht man folgendermassen vor:

```
      Distanz_X := x-x0
      Distanz_X < 0?
      ja   nein
       |     |
       |     |
       |     +---------------- Drehrichtung(X_Motor) := Uhrzeigersinn
       |                                             |
      Drehrichtung(X_Motor) := Gegenuhrzeigersinn    |
       |                                             |
       +<---------------------------------------------+
```

```
 |
Spanne(X_Motor) := abs(Distanz_X)
Spanne(X_Motor) am Bresenham-Frequenzgenerator von X_Motor einstellen
 |
 |
Distanz_Y := y-y0
Distanz_Y < 0?
ja   nein
 |     |
 |     |
 |     +---------------- Drehrichtung(Y_Motor) := Uhrzeigersinn
 |                                              |
Drehrichtung(Y_Motor) := Gegenuhrzeigersinn     |
 |                                              |
 +<---------------------------------------------+
 |
Spanne(Y_Motor) := abs(Distanz_Y)
Spanne(Y_Motor) am Bresenham-Frequenzgenerator von Y_Motor einstellen
 |
 |
Spanne(Y_Motor) > Spanne(X_Motor)
ja   nein
 |     |
 |     |
 |     +---------------- Spanne(X_Motor) am Frequenzbogen als Soll-Taktanzahl einstellen
 |                                                                  |
Spanne(X_Motor) am Frequenzbogen als Soll-Taktanzahl einstellen  |
 |                                                                  |
 +<-----------------------------------------------------------------+
 |
 |
Frequenzbogen starten.
```

Es gibt noch eine Variante der Betriebsart, die mit dieser Struktur verwirklichen kann: Man kann als Soll-Taktanzahl für den Frequenzbogen auch einen Wert nehmen, der grösser ist als die Führungsspanne. Die Motoren erreichen ihre Zielpositionen dann erst, wenn der Frequenzbogen seine Soll-Taktanzahl geliefert hat. D.h. nun bestimmt die Soll-Taktanzahl des Frequenzbogens die Bewegungsgeschwindigkeit und nicht die Führungsspanne. Diese Betriebsart erlaubt es, die maximale Verfahrgeschwindigkeit beider Motoren einzustellen. So etwas ist bei CNC-Maschinen aber auch beim RepRap wichtig.

Das Ganze sieht beim ersten Durchgang ziemlich verwirrend aus. Ich hoffe aber, dass beim zweiten oder dritten Lesen klar wird, wie's geht. Das Prototypprogramm läuft jedenfalls auf einem ATmega16 mit 16MHz und liefert eine maximale Schrittfrequenz von ca. 1kHz. Es kann neben der Motorsteuerung auch noch über die UART kommunizieren und die aktuellen Schrittzahlen beider Motoren auf einem LCD anzeigen (nicht vergessen, dass dabei zwei 32-Bit Integerzahlen in Dezimal-Strings umgewandelt werden!). Vom Flashspeicher bleiben noch 40% verfügbar.