I.H.L.S.
(Intelligentes Hochregal Lager System)

Version 1.00

entwickelt und dokumentiert von Thomas Unmuth

Inhaltsverzeichnis:

1. Einleitung

2. Charakteristika des I.H.L.S.

3. Mechanik
   - 3.1. Führung der einzelnen Elemente
   - 3.2. Antrieb der Achsen
   - 3.3. Ausführung des Regals

4. Elektronik
   - 4.1. Eigenschaften der LPT Schnittstelle
   - 4.2. Schaltung der Motoren
   - 4.3. Die Sensorik

5. Software
   - 5.1. Motorensteuerung
   - 5.2. Regalsteuerung

6. Anhang
   - 8.1 Bilder
   - 8.2 Quellcode der Motorsteuerung

7. Credits

1. Einleitung

Bereits in der Schule war ich fasziniert von computergesteuerten Maschinen. Angefangen hat diese Begeisterung bei mehreren Führungen mit der Schule in Firmen wie BMW, SRI Siemens, Allgäuer Brauhaus etc. Das Besondere bestand für mich darin, dass praktisch ohne menschliche Hilfe neue Maschinen erschaffen werden, bzw. komplexe Arbeit von Maschinen in enormer Geschwindigkeit auf Genauigkeiten im Bereich von 10^{-5} m verrichtet werden.
Am meisten hatte mich die Karosserie-Schweißerei bei BMW beeindruckt, indem sich etwa hundert Roboter wie von Geisterhand bewegten, ohne dass Menschen nötig waren.
Während dieser Zeit entschied ich mich dann auch, „Automatisierungstechnik in der Produktion“ zu studieren, weil ich genau solche Maschinen bauen wollte.
1999 wurde dann auch die Idee geboren, privat im kleinen Rahmen eine computergesteuerte Maschine zu bauen.
Voraussetzung war, dass die Mechanik nicht allzu kompliziert ist, dass keine analogen Elemente vorhanden sind, die Software einfach zu programmieren ist und dass sich all dies in einem finanziell günstigen Rahmen hält.
Nach einigem Überlegen entschied ich dann, das Modell eines Hochregallagers zu bauen, das zusätzlich über eine gewisse Intelligenz verfügt.
Im selben Jahr begann ich mit dem der Bau der Mechanik.
Das Projekt wurde allerdings gestoppt, da mir zu diesem Zeitpunkt das Wissen zu der Elektronik und der Software fehlte.
2003 wurde die Entwicklung des I.H.L.S. wieder aufgenommen und Anfang 2004 fertiggestellt.

2. Charakteristika des I.H.L.S.

Das I.H.L.S. (Abb. 1) besteht aus einem 2D Ladeportal, das Paletten von einem bestimmten Punkt zu einem anderen befördern kann.
Das Regal besitzt 8 Fächer und kann im Rahmen der Mechanik beliebig erweitert werden. Die Koordinaten der neuen Fächer können dann in der Software übernommen werden. Außerdem ist ein variables Fach vorhanden, bei dem Paletten gestapelt werden können.
Beim Einlagern kann die Verfügbarkeit eingestellt werden. Man kann Auswählen zwischen „hoch“, „mittel“ und „niedrig“. Die Höhe der Paletten muss bei der Version v1.00 noch manuell eingegeben werden.

Wird die Verfügbarkeit „hoch“ eingegeben, lagert das Programm die Palette in das zur Ladestation nächste freie Regalfach ein. Dies ermöglicht die kürzeste Zugriffszeit auf die Palette und somit die höchste Verfügbarkeit.
Bei der Verfügbarkeit „mittel“ wird die Palette auf das von der Ladestation am weitesten entfernte Regalfach eingelagert.

Bei der Verfügbarkeit „niedrig“ wird die Palette auf das variable Fach gestellt. Steht dort bereits eine Palette, werden sie gestapelt. Dies bedeutet höchste Platzeffektivität, aber auch die längsten Zugriffszeiten, da, falls es sich nicht um die oberste Palette handelt, bei der Auslagerung umgeräumt werden muss.

Diese Funktionen können einfach von einer übersichtlichen Software gesteuert werden.

3. Mechanik

3.1. Führung der einzelnen Elemente

Die Mechanik musste aufgrund fehlenden Werkzeuges sehr einfach ausfallen. Daher sind viele Elemente aus Holz und nicht sonderlich hochwertig gefertigt.
Die X und Y-Achse sind beide mit jeweils 8 Kugellagern auf Aluschienen gelagert. Dies bedeutet eine sehr geringe Reibung, was die Grundlage für akzeptable Bewegungsgeschwindigkeiten ist. Gehalten werden die Kugellager allerdings von Holzstäben, was zu einem somit nicht vermeidbaren Spiel führt. Dies ist auch der größte Ungenauigkeitsfaktor bei dem Lager.
Bei der Z-Achse läuft ein Kunststoffschlitten auf 2 Alustäben. Aufgrund schlechten Werkzeuges ist auch hier das Spiel sehr groß, es fällt im Betrieb des IHLS aber nicht negativ auf. Die Reibung der Alustangen in dem Kunststoff ist verhältnismäßig klein, nur bei größerem Palettengewicht ist der Schrittmotor überfordert.

3.2. Antrieb der Achsen

Angetrieben werden alle 3 Achsen durch Schrittmotoren, die aus alten Druckern ausgebaut wurden.
Angelenkt wird die X-Achse durch eine 0,15mm geflochtene Angelschnur. Angelschnur deshalb, weil diese am wenigsten aufträgt und relativ stabil und unelastisch ist. Der Motor ist direkt an eine Welle angeschlossen, die gleichzeitig auf der einen Seite die Schnur aufrollt und auf der anderen Seite abrollt. Am hinteren Ende wird die Schnur umgelenkt und an dem beweglichen Schlitten durch eine Feder gespannt.
Die Y-Achse wird auch mit der selben Angelschnur angesteuert, nur dass diese den Schlitten nur nach oben zieht, die Abwärtsbewegung wird von der Schwerkraft verrichtet. Da der vorhandene Schrittmotor nicht stark genug war, musste ein einfacher Flaschenzug mit der Untersetzung 1:3 installiert werden, damit der Motor den Schlitten nach oben ziehen kann. Durch diese Untersetzung reduziert sich leider auch die Geschwindigkeit und damit ist die Y-Achse die am langsamste Achse.
Die Z-Achse wird von einem Zahnriemen angetrieben. Dieser kann an einer Seite gespannt werden, liefert somit also relativ präzise Positionen.

3.3. Ausführung des Regals

Das Regal wird auf einer in ca.100mm Höhe liegenden Holzplattform aufgebaut. Darauf steht ein einfaches Sperrholzregal, das an die Plattform geschraubt ist. Somit entsteht eine hohe Flexibilität und das Regal kann sowohl versetzt als auch vergrößert bzw. verändert werden.

4. Elektronik

4.1. Eigenschaften der LPT Schnittstelle

Die LPT Schnittstelle wurde ausgewählt, weil man sehr einfach auf diese zugreifen kann und sie relativ viele Aus- und Eingänge hat.
Die Belegung sieht folgendermaßen aus:

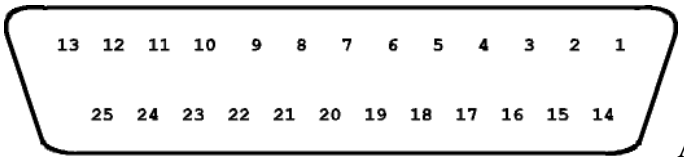

Abb. 1

Bezeichnung	Pin	Richtung	Bezeichnung	Pin	Richtung
Data Bit 0	2	Output	Error	15	Input/Output
Data Bit 1	3	Output	Select	13	Input/Output
Data Bit 2	4	Output	Paper End	12	Input/Output
Data Bit 3	5	Output	Acknowledge	10	Input/Output
Data Bit 4	6	Output	Busy	11	Input/Output
Data Bit 5	7	Output	Strobe	1	Input
Data Bit 6	8	Output	Auto Feed	14	Input
Data Bit 7	9	Output	Init	16	Input
Ground	18-25		Select In	17	Input

Per Software kann nun jeder einzelne Output Port auf high oder low gelegt werden (high bedeutet, dass gegenüber der Masse 5V anliegen, bei low liegen 0V an).
Bei den Eingangs Pins kann per Software ausgelesen werden, ob der Pin auf Masse geschlossen ist oder nicht. Somit können all diese Pins Ausgänge und Eingänge realisieren.

4.2. Die Schaltung der Motoren

Da die Pins des LPT Port nur wenige mA liefern, müssen die Ströme entsprechend verstärkt werden, damit die Motoren laufen. Diese Schaltung sieht für alle 3 Motoren identisch aus, somit beschränke ich mich auf die Schaltung eines Motors. (Abb. 3)

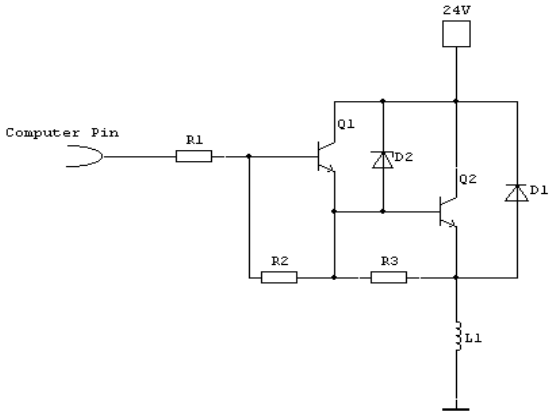

Abb. 2

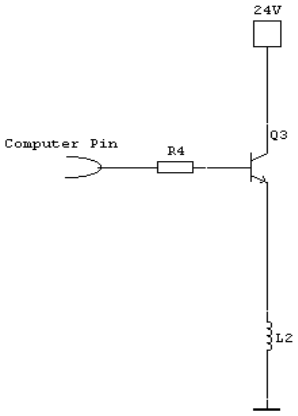

Abb. 3

Über einen Vorwiderstand wird der Computerpin an einen Transistor angeschlossen, der die 24 V der Spannungsquelle an die Motorspule L1 freischaltet. Die in Abb. 2 gezeigte Schaltung dient lediglich zur extremen Verstärkung des Stromes. Das Prinzip der Schaltung wird in der Ersatzschaltung Abb. 3 vereinfacht.
Im Wesentlichen muss also nur jede der vier Spulen über einen Transistor angesteuert werden. Dabei muss darauf geachtet werden, dass eine Freilaufdiode verwendet wird (D1 in Abb.2). Um eine ausreichende Verstärkung erreichen zu können sollte entweder ein Darlington Transistor oder ein Mosfet verwendet werden. Diese Bauteile befinden sich in der Regel auf der Platine und können bei alten Druckern einfach ausgelötet und wiederverwendet werden. In meinem Fall habe ich einfach die Darlington Transistoren von der Original Drucker Platine verwendet. Bei diesen war sogar bereits die Freilaufdiode integriert.

4.3. Die Sensorik

Da bei Beginn des Programms das Regal alle 3 Achsen Nullen muss, um einen festen Ausgangspunkt zu erreichen, werden als Sensoren hierfür Gabellichtschranken verwendet. Sie sind billig und einfach in der Handhabung. An den beiden Schlitten und der Gabel des Regals sind jeweils Gegenstände angebracht, die in der Nullposition durch die Gabellichtschranken gehen.

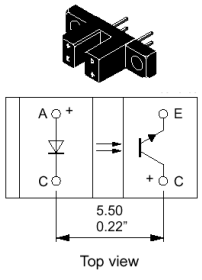

Abb.4

Gabellichtschranken bestehen aus einer Infrarotdiode und einem Phototransistor. Wird ein lichtundurchlässiges Objekt zwischen die Lichtquelle und dessen Sensor gehalten, verringert sich der Widerstand der Photodiode. Nun muss eine kleine Schaltung gemacht werden, die den Datenpin auf Masse schließt, solange sich nichts in der Lichtschranke befindet. Dazu wird folgende Schaltung verwendet:

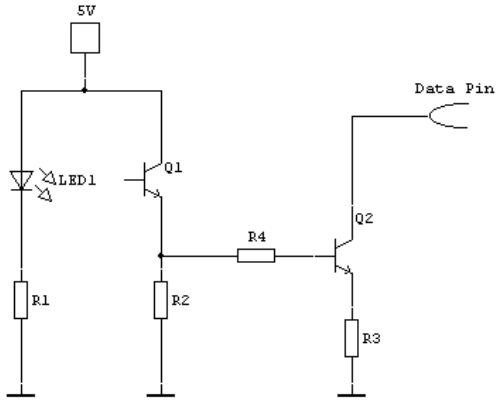

Abb.5

Diese Schaltung (Abb. 5) wird auch hier wieder bei allen 3 Achsen angewendet und somit kann die korrekte Nullposition der entsprechenden Achsen ermittelt werden.

5. Software

5.1. Motorensteuerung

Die Software für die Motorensteuerung hat sich als die größte Hürde des Projektes I.H.L.S. herausgestellt. Es galt eine Software zu entwickeln, die folgende Eigenschaften besitzt:

1. Beide Motoren können unterschiedlich weit fahren.
2. Es müssen unterschiedliche Geschwindigkeiten möglich sein.
3. Jeder Motor darf eine andere Beschleunigung haben.
4. Es muss eine sinusförmige Beschleunigungskurve besitzen.

Alle diese Eigenschaften sind in einem Graphen dargestellt:

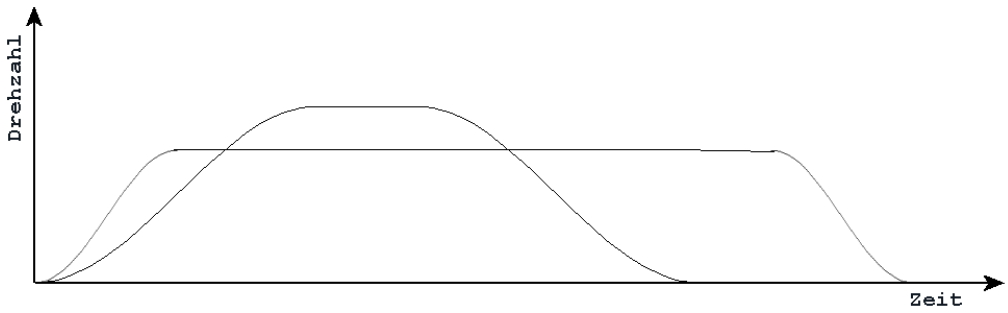

Abb. 6

Das Modul mit dem kommentierten Quellkode befindet sich im Anhang.
Diese Lösung funktioniert nach folgendem Prinzip:
Als erstes wird mit Hilfe einer Zählschleife ein künstlicher Takt im Hz Bereich erstellt. Dazu wird eine Zählschleife verwendet, da ein Zugriff auf die Systemzeit nicht mit der benötigten Geschwindigkeit möglich ist. Mit Hilfe dieses Taktes kann sich nun der Motor mit konstanter Geschwindigkeit drehen. Gibt man z.B. die Geschwindigkeit 80 ein, wird alle 80 Takte der Motor um einen Halbschritt weiter bewegt.
Befindet sich der Motor noch in dem beschleunigten Bereich, wird er mit einer Sinus Funktion multipliziert. Dadurch entsteht eine sinusförmige Beschleunigung. Durch diese Methode ist es möglich, mehrere Motoren mit unterschiedlichen Geschwindigkeiten und Beschleunigungen zu betreiben.

5.2. Regalsteuerung

Die Regalsteuerungssoftware ist eine übersichtlich gestaltete Software, mit der die einzelnen Paletten verwaltet werden.

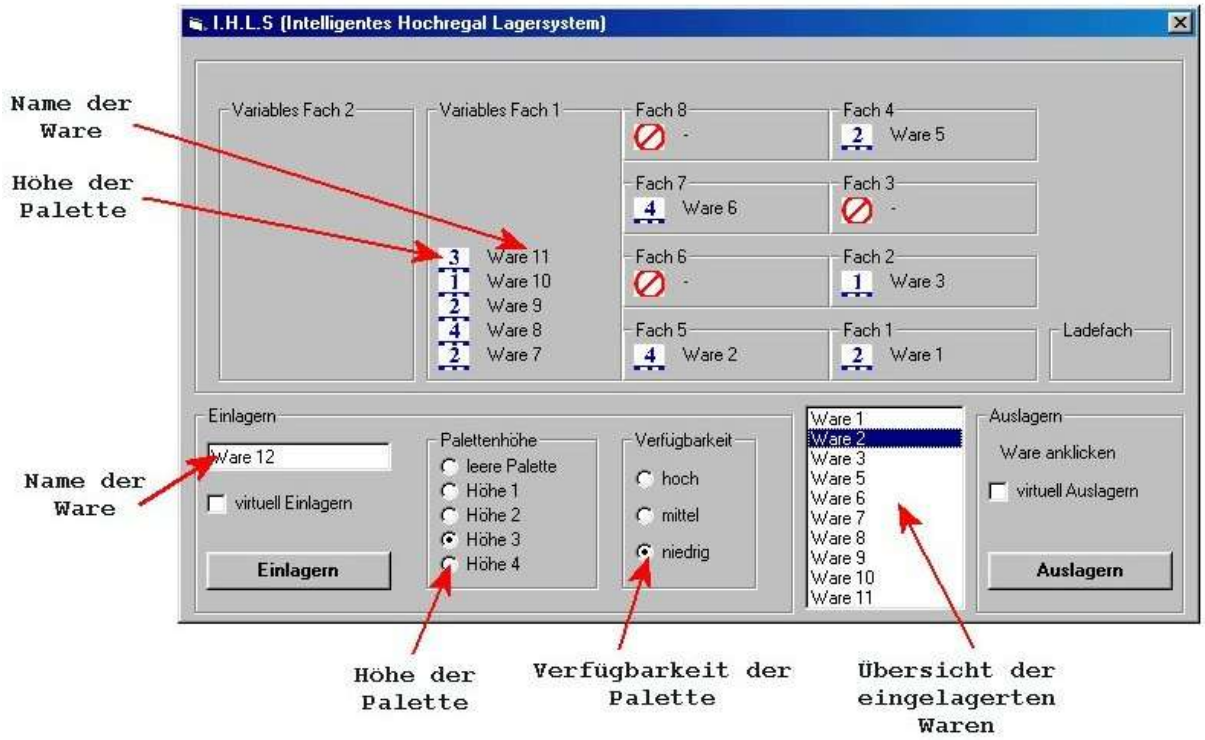

Abb. 7

Die Oberfläche besteht aus der Übersicht der Regale, in welchem die Palette stehen und Ein- bzw. Auslagereinheit.

Wird eine neue Palette eingelagert, kann der Name eingegeben werden, es muss die Höhe eingegeben werden und die Verfügbarkeit muss eingestellt werden. Wird dann auf „Einlagern“ geklickt, wird die Palette in das entsprechende Fach gelagert.

Beim Auslagern kann die gewünschte Palette angeklickt werden und dann ausgelagert werden.
Da das Programm die Palettenplätze noch nicht speichern kann, wenn das Programm beendet wird, besteht die Möglichkeit, Palette virtuell ein- und auszulagern. Dies bedeutet, dass die Daten vom Computer voll verwaltet werden, nur dass das Regal nicht real agiert.

6. Anhang

6.1. Bilder

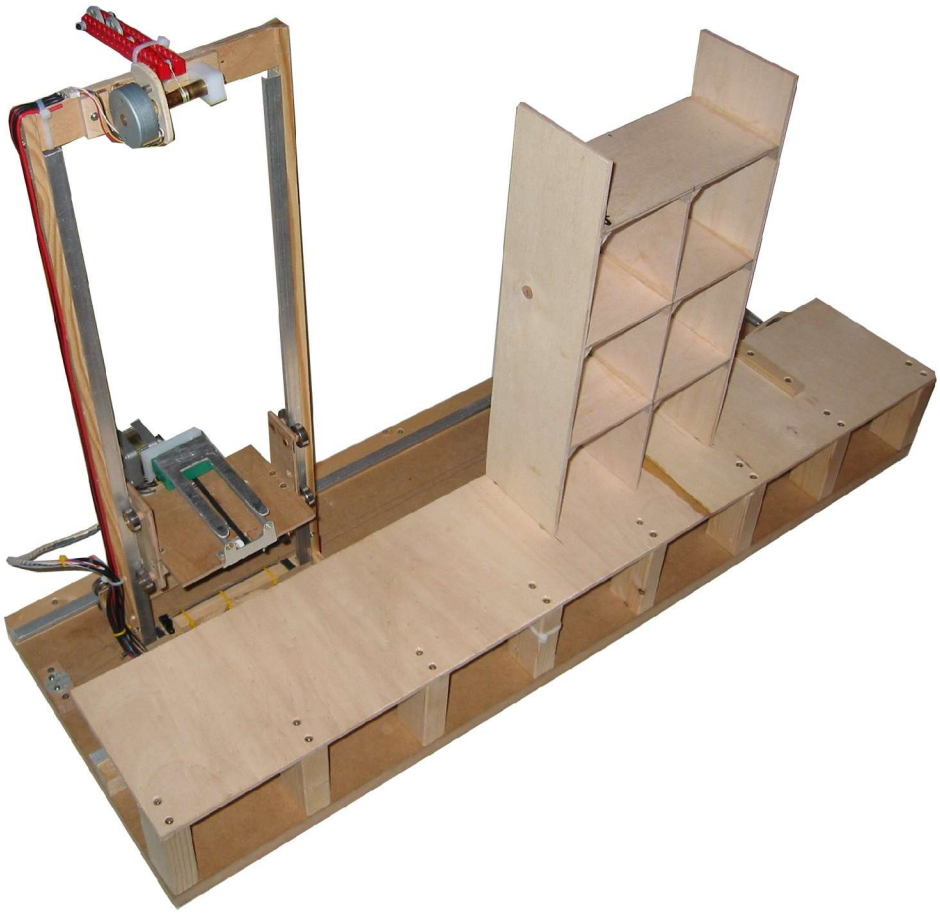

Bild. 1

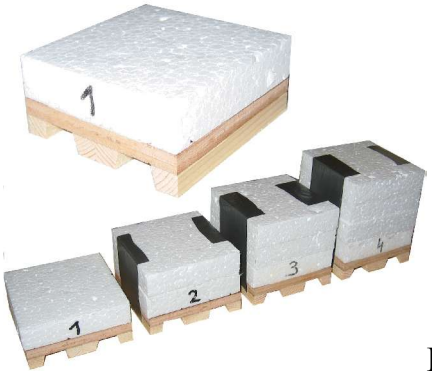

Bild 2

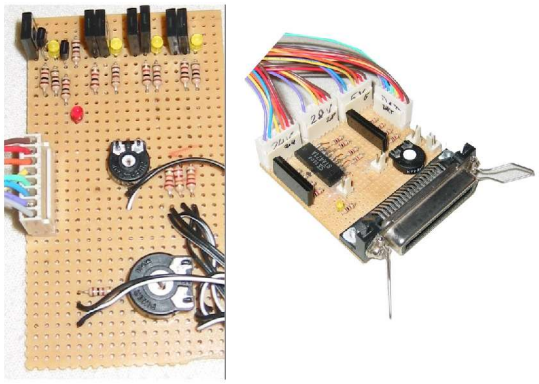

Bild 3

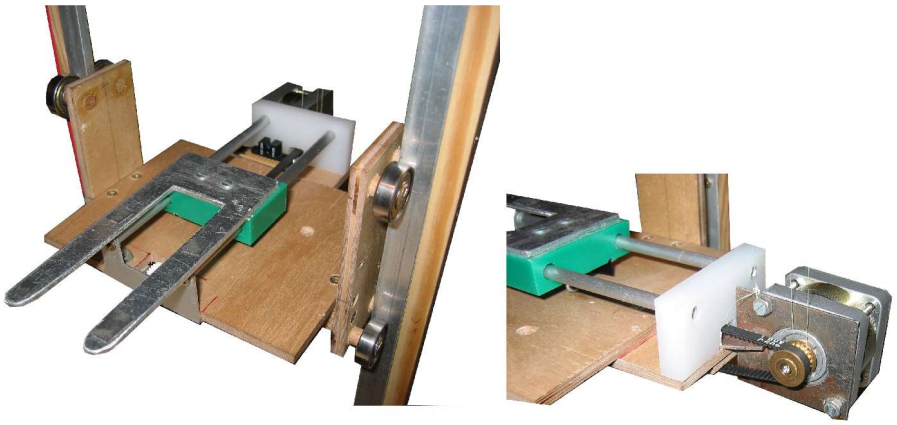

Bild 4

6.2. Quellcode der Motorsteuerung

```
Private Sub step_move11(steps1 As Integer, steps2 As Integer, direction1 As
Boolean, direction2 As Boolean, time1 As Integer, time2 As Integer, accel1 As
Variant, accel2 As Variant)

    Dim ausgabe_steps As Integer
    Dim counter As Integer
    Dim next1 As Integer, next2 As Integer
    Dim pos_accel1 As Integer, pos_accel2 As Integer
    Dim status_accel1 As Boolean, status_accel2 As Integer
    Dim veraendert As Boolean
    Dim t1 As Integer
    Dim t2 As Integer
    Dim pi As Double
    Dim wait_at_end As Boolean

    pi = 3.1415

    'Damit die Motoren nicht zu heiss werden, werden sie ausgeschaltet, wenn sie
      nicht benötigt werden.
    'An dieser Stelle werden sie dann wieder eingeschaltet.
    OutPort 888, speicher_steps

    'Coounter ist mein künstlich erzeugter Takt.
    counter = 0

    'Next bedeutet dass in Next schritten der Motor um eine Halbdrehung bewegt
      wird.
    next1 = 0
    next2 = 0


    'Falls Beschleunigung=0 wird sie hier auf 1 gesetzt damit in der Sin Formel
      kein Fehler auftritt.
    If accel1 = 0 Then
        pos_accel1 = 1
        wait_at_end = False
    Else:
        pos_accel1 = accel1
        wait_at_end = True
    End If

    If accel2 = 0 Then
        pos_accel2 = 1
    Else: pos_accel2 = accel2
    End If

    'status_accel gibt an, ob sich die Motoren im beschleunigten Zustand befinden
    status_accel1 = True
    status_accel2 = True

    'Wenn zu einem Takt sich ein Motor um einen Schritt bewegt, wird verändert auf
      true gesetzt,
    'so muss nicht bei jedem Takt das Ausgangssignal generiert werden.
    veraendert = False

    t1 = 1
    t2 = 1


    Do
        'wenn next mit dem counter übereinstimmt, kann sich der Motor um einen
             Halbschritt bewegen.
        If next1 = counter And direction1 = True And steps1 > 0 Then

            'hier wird die viruelle Position um ein weiter gesetzt
            position1 = position1 + 1

            'Die verbleibenden Schritte werden um 1 heruntergesetzt
            steps1 = steps1 - 1
```

```
    'hier wird generiert wann der nächste Schritt geschehen soll
     next1 = counter + Round((1 / (Sin((pi * t1) / (pos_accel1 * 2))))
           * time1)
    veraendert = True

    If status_accel1 = True And t1 < pos_accel1 Then
        t1 = t1 + 1
    ElseIf status_accel1 = False And t1 > 1 Then
        t1 = t1 - 1
    End If

    If steps1 <= accel1 Then
        status_accel1 = False
    End If


'dies hier ist analog, nur für die andere drehrichtung
ElseIf next1 = counter And direction1 = False And steps1 > 0 Then

    position1 = position1 - 1
    steps1 = steps1 - 1
    next1 = counter + Round((1 / (Sin((pi * t1) / (pos_accel1 * 2)))) *
      time1)
    veraendert = True

    If status_accel1 = True And t1 < pos_accel1 Then
        t1 = t1 + 1
    ElseIf status_accel1 = False And t1 > 1 Then
        t1 = t1 - 1
    End If

    If steps1 <= accel1 Then
        status_accel1 = False
    End If

End If



'dies hier ist analog, für den 2. Motor
If next2 = counter And direction2 = True And steps2 > 0 Then

    position2 = position2 + 1
    steps2 = steps2 - 1
    next2 = counter + Round((1 / (Sin((pi * t2) / (pos_accel2 * 2)))) *
      time2)
    veraendert = True

    If status_accel2 = True And t2 < pos_accel2 Then
        t2 = t2 + 1
    ElseIf status_accel2 = False And t2 > 1 Then
        t2 = t2 - 1
    End If

    If steps2 <= accel2 Then
        status_accel2 = False
    End If


 ElseIf next2 = counter And direction2 = False And steps2 > 0 Then

    position2 = position2 - 1
    steps2 = steps2 - 1
    next2 = counter + Round((1 / (Sin((pi * t2) / (pos_accel2 * 2)))) *
     time2)
    veraendert = True

    If status_accel2 = True And t2 < pos_accel2 Then
        t2 = t2 + 1
    ElseIf status_accel2 = False And t2 > 1 Then
        t2 = t2 - 1
    End If
```

```
        If steps2 <= accel2 Then
            status_accel2 = False
        End If

     End If


    'Hier wird die richige Ausgabe Variable zusammengestellt

    If veraendert = True Then

        ausgabe_steps = 0
        Select Case (position1 Mod 8)
            Case Is = 0
                ausgabe_steps = ausgabe_steps + 5
            Case Is = 1
                ausgabe_steps = ausgabe_steps + 1
            Case Is = 2
                ausgabe_steps = ausgabe_steps + 9
            Case Is = 3
                ausgabe_steps = ausgabe_steps + 8
            Case Is = 4
                ausgabe_steps = ausgabe_steps + 10
            Case Is = 5
                ausgabe_steps = ausgabe_steps + 2
            Case Is = 6
                ausgabe_steps = ausgabe_steps + 6
            Case Is = 7
                ausgabe_steps = ausgabe_steps + 4

        End Select

        Select Case (position2 Mod 8)
            Case Is = 0
                ausgabe_steps = ausgabe_steps + 80
            Case Is = 1
                ausgabe_steps = ausgabe_steps + 16
            Case Is = 2
                ausgabe_steps = ausgabe_steps + 144
            Case Is = 3
                ausgabe_steps = ausgabe_steps + 128
            Case Is = 4
                ausgabe_steps = ausgabe_steps + 160
            Case Is = 5
                ausgabe_steps = ausgabe_steps + 32
            Case Is = 6
                ausgabe_steps = ausgabe_steps + 96
            Case Is = 7
                ausgabe_steps = ausgabe_steps + 64

            End Select

        'Und hier wird der Port genullt und dann das Richtige ausgegeben

        vbOut 888, ausgabe_steps
        veraendert = False
    End If



    wait (1)
    'hier wird der virtuelle Takt um 1 hochgesetzt
    counter = counter + 1



Loop While steps1 > 0 Or steps2 > 0


speicher_steps = ausgabe_steps
If wait_at_end = True Then
    wait (100)
```

```
    Else
        wait (30)
    End If
    OutPort 888, 0

End Sub
```

7. Credits

Zum Schluss noch ein Dankeschön an alle, die mir geholfen haben. Besonders an:

- Rainer Haseitl mit dem die Idee geboren wurde,
- dem Spotlight Forum (www.spotlight.de) für die Beantwortung all der vielen Fragen,
- Christian Fuchs (http://mitglied.tripod.de/ChFuchs/index.html) von dem ich all mein Wissen über Schrittmotoren habe.
- und zuletzt meiner Oma, die mir diese Dokumentation korrigiert hat.