

embedded



③

Rafael Deliano
Steinbergstr.37
82110 Germering
Tel 089/8418317

j_r_d@t-online.de

V1.0 (Papier) : 2. Dez. 98
V2.0 (pdf) : 23. Feb. 01
V2.1 (pdf) : 29. April 04
V2.2 (pdf) : 14. Jan 07

- 6 CRC
- 9 PAUSE-Multitasker
- 12 Gray-Code
- 13 Emulator
- 16 V24
- 18 Schrittmotoren
- 20 Stromversorgung
batteriebetriebener Geräte
- 22 Nachträge
Mehr CORDIC

READ . ME

Seit die ELRAD ihr Erscheinen eingestellt hat, gibt es in Deutschland leider keine Zeitschrift mehr die sich speziell mit Mikrocontrollern befaßt. In dieser Ausgabe sind nocheinmal alle meine in der ELRAD erschienen Beiträge zusammengefaßt.

Die Darstellung der Berechnung der CRC ist erweitert worden. Der in der ursprünglichen Ausgabe vorhandene Artikel zu Einstellregeln nach Ziegler & Nichols für PID-Regler enthielt diverse Fehler und erscheint deshalb hier nichtmehr. Zu Schrittmotoren und V24 folgen in den nächsten Heften weitere Beiträge.

Die Listings sind in nanoFORTH geschrieben. Für die Konvertierung in andere FORTH-Varianten finden sich Hinweise im nanoFORTH-Manual.

CORDIC

CORDIC ist die Abkürzung für „Coordinate Rotation Digital Computer“. Es ist ein Verfahren zur effizienten Berechnung mathematischer Funktionen. Dabei werden nur Addition, Subtraktion und Schiebeoperationen benötigt. Auch einfache 8-Bit Controller haben diese Befehle und man kann deshalb auf ihnen CORDIC wirksam implementieren. Ergebnis der Berechnung sind Sinus, Cosinus, Arcustangens und ihre inversen Funktionen. Sowie Multiplikation, Division und Quadratwurzel. Also so ziemlich alles was man braucht.

Geschichte

Der trigonometrische Teil wurde von Jack Volder [1] Ende der 50er Jahre entwickelt und von Walther [2] später um die linearen und hyperbolischen Funktionen erweitert.

Seine Anwendung ist universell. Sie reicht von Taschenrechner von Hewlett-Packard aus den frühen 70er Jahren über Arithmetik-Coprozessoren wie dem 8087 zur Software in den AMSAT-Satelliten der Amateurfunkler. Am meisten profitiert man von CORDIC wohl bei der Berechnung der Bewegung von Körpern im Raum, wie sie für Werkzeugmaschinen und Roboter typisch ist.

Das hier beschriebene Festkomma-CORDIC realisiert zwar trigonometrische Funktionen, ist aber kein Ersatz für eine universelle Floatingpoint-Package. Die Dynamik ist durch das Festkommaformat begrenzt und die Konvergenz auf bestimmte Wertebereiche beschränkt. Das Verfahren eignet sich jedoch gut für Anwendung auf Controllern wo Geschwindigkeit und preiswerte Hardware wesentlich sind.

Bild 3: „Stör“-Faktoren

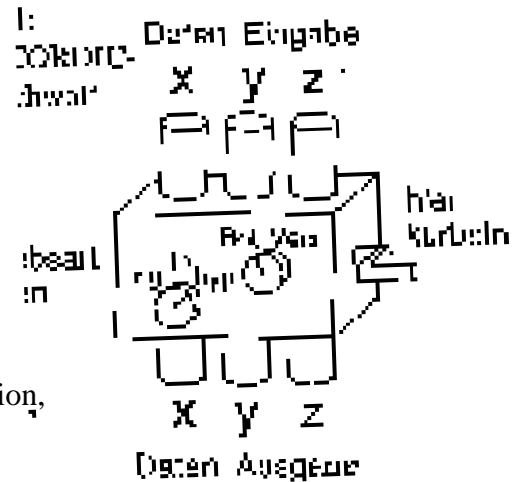
$$K_{\text{trig}} = \prod_{i=0}^n \sqrt{1 + 2^{-i2}} = 1,64676$$

$$K_{\text{hyp}} = \prod_{i=1,2,3,4,\dots}^n \sqrt{1 - 2^{-i2}} = 0,8281593$$

Ablauf

Als Blackbox für den Anwender kann man sich den Fleischwolf in Bild 1 vorstellen. Am Eingabe werden oben drei verschiedene Zahlen für X, Y, und Z eingeworfen. Dann wählt man mit den beiden Drehschaltern die Betriebsart. Es sind dabei sechs Rechenfunktionen möglich. Das Verfahren ist iterativ und reduziert eine der Zahlen, Y oder Z, schrittweise zu Null. Nun muß man also kräftig kurbeln um sie klein zu bekommen. Danach kann man unten die drei modifizierten Zahlen entnehmen.

Die genaue Wirkung der Berechnung zeigt die Funktionstabelle in Bild 2. Xo, Yo, Zo sind jeweils die Eingabewerte, während X, Y, Z das Ergebnis sind. Die Werte scheinen ziemlich willkürlich verknüpft, und zudem geistern noch zwei krumme Konstanten herum. Auf den ersten Blick schaut das Ganze deshalb etwas unbrauchbar komplex aus. Man kann aber die Parameter die man nicht benötigt durch geeignete Konstanten wie 0 oder 1.0 besetzen und das Ergebnis dadurch vereinfachen. Will man z.B. die Funktion $b = \sin(a)$, setzt man $X=1, Y=0, Z=a$. Dann berechnet man in der Einstellung ROT-TRIG und das Ergebnis erscheint in Y. Es ist allerdings noch mit „Stör“-Faktor K skaliert, also $b = Y/K$. Die Division kann durch eine weitere Berechnung mit VECT-LIN ausgeführt werden. Man wird diese Operation in vielen Fällen nicht explizit vornehmen. Sondern versuchen sie in einem



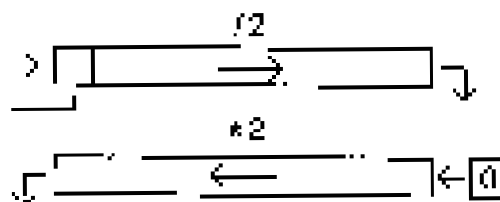
oder sie in einem anderen Skalierungsfaktor in Anwendungsgleichung einzubringen. Wie der Festkomma-Verbleibungen sind auch in Anwendung eignen. Wenn z.B. einer Multiplikation eine Addition folgt, kann diese Kombination nicht weiter ausgeführt werden. Man kann jedoch Algorithmen (z.B. die FFT) in ein für CORDIC geeignetes Format überführen.

Der zweite Konstanten K ergibt sich bei einer vollständigen Berechnung der Teilfunktion multipliziert, wie in Bild 3. Die Laufzeit ist in einer Zeile der Tabelle angegeben, während der CORDIC-Berechnung. Der Wert der Konstante wandelt damit von der Zeit der Berechnung abhängig. Der Wert ist aber bei -1 bis 1000 völlig stabil, während die CORDIC-Berechnung mehr als 10 Zyklen benötigt. Faktorell ist also nicht nur durch den Konstanten in der Software selbst, sondern beide Konstanten nicht auf die Hardware und im Anwendungsprogramm konstante Werte werden. Die Funktionstabelle einer Funktionstabelle kann die mit Wert + oder - angeben und ist die, welche Variablen die Y und Z konstant. Alle trigonometrischen Sinus und Cosinus verknüpfte Zahlen sind.

Implementierung

Die Programmierung des CORDIC mit einem 8-Bit Controller kann bestimmt werden. Die Berechnung der Funktionen des CORDIC

Bild 5: Shift



Wird mit 32 Bit Wortbreite bei 2, 3, 4 Bytes. Das ist die Zahl der Bytes. Nicht nur ein durch Eingangsdaten von Bytes aus (links) ein. Auch hier muss der ursprüngliche Wert des ursprünglichen Werts (es ist ein 0) nicht verloren gehen. Es ist ein FF Byte. Der Rest der Bits (von 0 bis 255) sind die Bits, die durch die Eingangsdaten (links) ein, werden.

Die Werte der Anzahl-Werte werden in einer Tabelle angegeben. Diese Tabelle gibt die Grenzen für 32 Bit und diese sind 128 Byte. 64 Bit da jede Zahl 4 Byte belegt und für alle die Bits werden. Zahlen von 0 bis 63. Die 64 Bit sind die Bits, die durch die Eingangsdaten (links) ein, werden.

Die Werte der Eingangsdaten (links) ein, werden. Das ist die Tabelle, die die Werte der Eingangsdaten (links) ein, werden. In der Tabelle sind die Werte der Eingangsdaten (links) ein, werden.

$$n_0 = 0, 1, 2, 3, \dots$$

$$n_{1n} = -1, 0, 1, 2, 3, \dots$$

$$n_{2n} = 1, 2, 3, 4, 5, 6, \dots$$

$$k_{2n} = 4, 13, 12, \dots$$

$$k = 3k_1 + 1$$

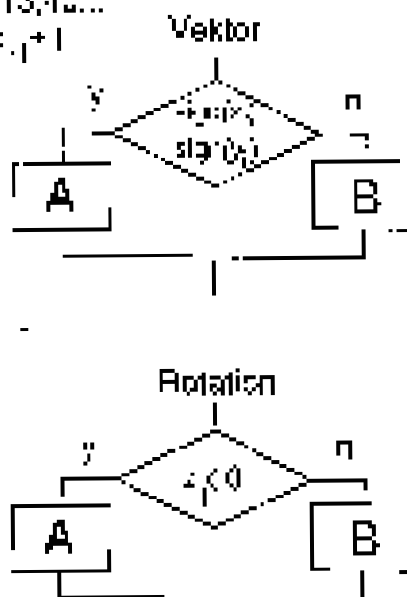


Bild 5: Haupttafel

oder auch durch Vergleichsbedingung auslösen.

Der Shift im Konverter ist die gleiche Funktion, die im Beispiel oben noch nicht spezifiziert wurde. Man legt aus einer Tabelle an. Oder man erzeugt ein eigenes Register, das mit 10 in einem wird. Dann wird der Wert jeweils um die 10 links oder rechts verschoben und zu dem entsprechenden Formel-Mathematik verfahren. Diese Tabelle ist die Tabelle und ist in der Tabelle angegeben.

Konvergenz & Überlauf

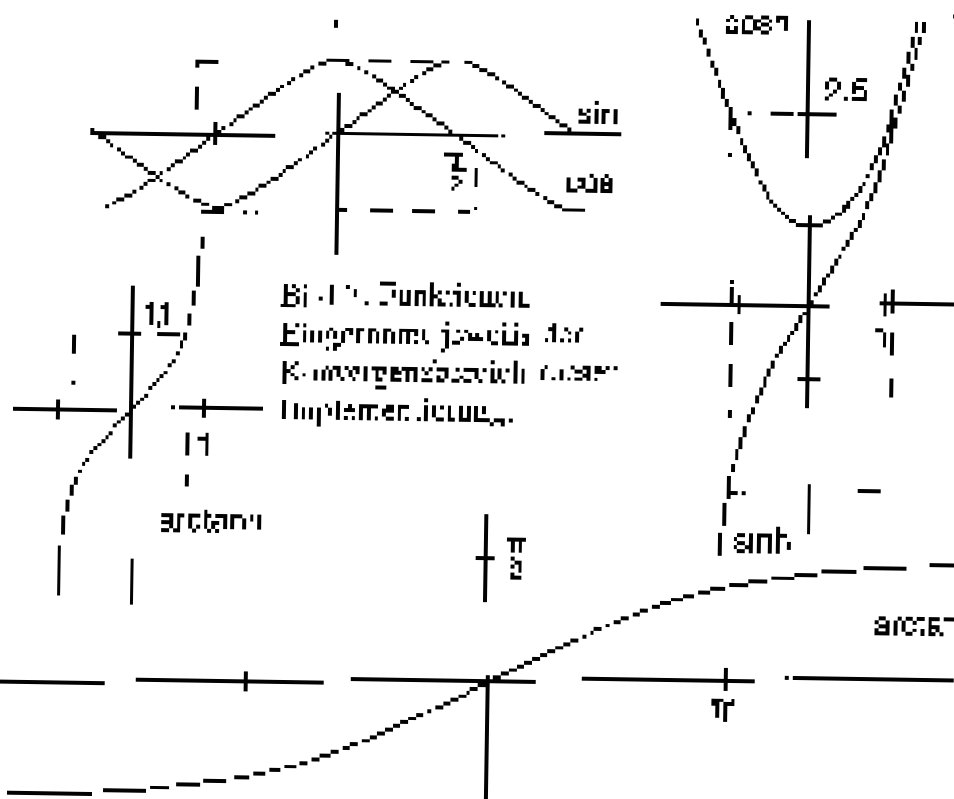
Bei Konvergenz und Überlauf ist die Anzahl der Bits, die die Zahl hat, nicht an. In der Tabelle ist die Anzahl der Bits, die die Zahl hat, nicht an. In der Tabelle ist die Anzahl der Bits, die die Zahl hat, nicht an.

typische Fehler, das Überlauf, wenn die Zahl größer ist, als die Zahl, die die Zahl hat, nicht an. In der Tabelle ist die Anzahl der Bits, die die Zahl hat, nicht an.

Zusätzlich gibt es die für die Tabelle ist die Tabelle, die die Werte der Eingangsdaten (links) ein, werden. In der Tabelle sind die Werte der Eingangsdaten (links) ein, werden.

Programm

A	B
Trigonometrische Funktion	
$x_{i+1} = x_i - y_i \cdot 2^i$	$x_{i+1} = x_i - y_i \cdot 2^i$
$y_{i+1} = y_i + x_i \cdot 2^i$	$y_{i+1} = y_i + x_i \cdot 2^i$
$z_{i+1} = z_i + \text{Arctan}(2^i)$	$z_{i+1} = z_i - \text{Arctan}(2^i)$
Lineare Funktion	
$x_{i+1} = x_i$	$x_{i+1} = -x_i$
$y_{i+1} = y_i \cdot x_i \cdot 2^i$	$y_{i+1} = y_i + x_i \cdot 2^i$
$z_{i+1} = z_i - 2^i$	$z_{i+1} = z_i - 2^i$
Hyperbolische Funktion	
$x_{i+1} = x_i - y_i \cdot 2^i$	$x_{i+1} = x_i - y_i \cdot 2^i$
$y_{i+1} = y_i + x_i \cdot 2^i$	$y_{i+1} = y_i + x_i \cdot 2^i$
$z_{i+1} = z_i + \text{Arctanh}(2^i)$	$z_{i+1} = z_i - \text{Arctanh}(2^i)$



Bi-10. Funktions-
Eingernisse jeweils der
Konvergenzverhältnisse
Implanten. arcsinh

Tabelle 1:
Erläuterte Eingangswerte für
 X, Y, Z für das jeweilige
Format.

BI-10-10T-3
Die Konvergenz mit 3 Format
1,74 sein. Somit kann man
mit dem einen 102-Format
wobei X und Y haben keine
Anforderung an Konvergenz,
sondern über zu überlegen können
se mit Y oder Z durch
Werte 102 sein man, weniger
konstant werden. Auflösung:
55 Stellen.

BI-10-11E
Die Konvergenz mit 7 kleiner
40 sein. X und Y haben keine
Anforderung an Konvergenz,
sondern über zu überlegen können.
Auflösung: 65 Stellen.

BI-10-12F
Die Konvergenz mit 3 kleiner
1,73 sein. X und Y haben keine
Anforderung an Konvergenz,
sondern über zu überlegen können.
Se sollten ganzheitlich durch
2,50 sein man, weniger konstant
werden. Auflösung: 50 Stellen.

BI-10-13TC
Schnell eine Konvergenz-
verhältnisse mit X und Y
sondern über zu überlegen können.
Se sollten ganzheitlich durch
Kriter 2,4 können. Auch wenn sie
positive Konstanten werden, kann
Kriter der beiden Werte größer als
3,0 sein. Auflösung: 65 Stellen.

BI-10-14N
Die Konvergenz mit 7 X und Y
Kriter 4,0 sein. Konstanten
von Kriterien 2,50 und 3,0 sein.
Neben dem während der
Suchung zu überlegen. Die
Konvergenzverhältnisse mit 4,0, 3,0
Kriter 3,0. Ab 102 die es kann
sein Probleme. Auflösung: 75
Stellen.

BI-10-15E
Die Konvergenz mit 6
kleiner 0,6 sein.
Auflösung: 65 Stellen.

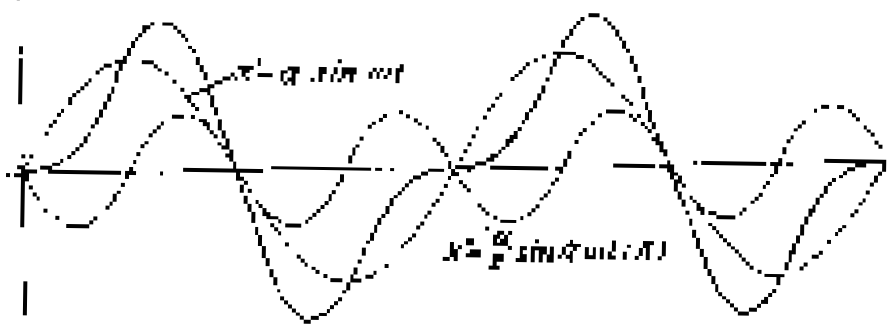
Das Listing zeigt ein echtes
Programm in BASIC. Die
Befehle sind in der Anweisung und
nicht bis 1000-1000 und befinden
sich im Falle des Listings. Die Daten
werden 1000 mal auf dem Stack
abgelegt. Die Befehle arcsin , arccos ,
 sinh , cosh sind in geeigneter
weil eine Zahlen mit Toleranz arcsin -
und arccos -Format. Die Hauptfunktion
mit BI-10-10T-3 wird oben
verfügt. Außerdem möglich ist
eine BI-10-10T-3 auf die Konvergenz
mit BI-10-10T-3 und BI-10-10T-3 .
Diese Berechnung ist Funktion
mit BI-10-10T-3 . Die Laufzeit ist
etwa 2,5 bis 3,0 Sekunden.
Die Schichtfunktion
wird oben in den BI-10-10T-3
eingeführt. BI-10-10T-3 und BI-10-10T-3
gibt es auf die ebenfalls entsprechende
Tabelle zu. Am Ende ist es sinnvoll

und bei jeder Iteration der Wert von
 X, Y, Z und T auf dem Bildschirm
ausgeben. Dafür ist der Befehl
 PRINT vorgesehen.

Diese Funktion basiert auf Hoch-
sprache format für BI-10-10T-3 .
man muss etwa 40000 auf einem 2-Kriter
mit BI-10-10T-3 . Die auf BI-10-10T-3
-Kriter bestimmte Assemblerfunktion
(BI-10-10T-3) ist die in der
Bibliothek. Die Funktion ist am Ende
schwierig. Die Funktion wird
mit BI-10-10T-3 Speicher

- (1) Walter F.E., "The BI-10-10T-3 Digital
numeric Computing Technique
 BI-10-10T-3 1968 S.190-214
- (2) Walter F.E., "A unified algorithm
for elementary functions"
Spring Joint Computer Conference
(SIJC) 1961 S.370-385

BI-10-10T-3 ist ein Nachdruck
aus BI-10-10T-3 „Zahlenrechner“
und wurde mit Hilfe von Donald
erweitert die BI-10-10T-3 und BI-10-10T-3
Kriter von Verfügung stellen.



CRC

CRC, „cyclic redundancy check“, ist ein Verfahren das zur Fehlersicherung in der Datenübertragung verwendet wird. In bestimmten Anwendungen wie z.B. bei Filetransferprogrammen wie ZMODEM, erfolgte die Berechnung immer schon in Software. Meist jedoch wird sie in Hardware in Peripherie-ICs ausgeführt. Da die Rechenleistung der Controller ständig steigt, ist die Verlagerung dieser Funktion in Software häufig möglich.

$x^8+x^4+x^3+x^2+1$ $x^8+x^5+x^4+1$ $x^8+x^2+x^1+1$ $x^8+x^7+x^5+x^4+x^1+1$ $x^{12}+x^{11}+x^3+x^2+x^1+1$ $x^{16}+x^{12}+x^5+1$ $x^{16}+x^{15}+x^2+1$ $x^{16}+x^{15}+x^{13}+x^7+x^4+x^2+x^1+1$ $x^{24}+x^{23}+x^{14}+x^{12}+x^8+1$ $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$	<p>CRC-8 Touch Memory PDV-Bus</p> <p>CRC-12 CCITT CRC-16 ANSI</p> <p>CRC-24 CCITT, Ethernet</p>
--	---

Die CRC wird bei der Übertragung genau wie eine Prüfsumme benutzt. Im Sender wird das CRC-Prüfzeichen über die Nutzdaten berechnet. Dieses wird dem Datenpaket bei der Übertragung angehängt. Im Empfänger wird die Berechnung wiederholt. Ist das Prüfzeichen identisch, wurden die Daten mit hoher Wahrscheinlichkeit fehlerfrei übertragen. Im Fehlerfall muß der Datensatz vom Empfänger noch einmal angefordert werden. In Tabelle 1 sind die Generatorpolynome der üblichen CRCs dargestellt. Der größte Exponent links gibt an ob die CRC 8, 12, 16, 24 oder 32 Bit breit ist.

HDLC ist eine CCITT- und ISO-Norm die im ISDN verwendet wird. X25 ist ein Mitte der 70er Jahre entstandene, sehr erfolgreiche Norm die auf HDLC beruht. SDLC ist eine HDLC-Variante von IBM. Seine Anwendung beschränkt sich nicht nur auf Mainframes. Auch der Bitbus basiert auf SDLC. In HDLC/SDLC wird als Prüfzeichen ziemlich einheitlich CRC-CCITT verwendet. Man beachte, daß in der Literatur die CRC-CCITT oft auch als CRC-16 bezeichnet wird.

Oft reicht eine 16 Bit CRC nicht mehr aus. In stark gestörter Umgebung, z.B. bei Datenübertragung übers 220V-Netz, wird teilweise CRC-24 verwendet. Da die Bitrate hier niedrig ist erfolgt die Berechnung in Software durch den Controller. Auch in ungestörter Umgebung reicht die Sicherheit einer 16 Bit CRC nicht aus, wenn die Datensätze sehr lang sind. Ethernet und das Filetransferprogramm ZMODEM verwenden deshalb die 32 Bit CRC.

CRCs unterscheiden sich nicht nur in ihrem Generatorpolynom. Eine wichtige Quelle von Inkompatibilität ist die Initialisierung des Startwerts. Die meisten neueren Protokolle, insbesondere HDLC/SDLC, setzen am Anfang nicht alle Bits auf 0, sondern auf 1. Damit vermeidet man Probleme wenn zufällig alle Datenbits 0 sind.

Bild 3:
reverse CRC

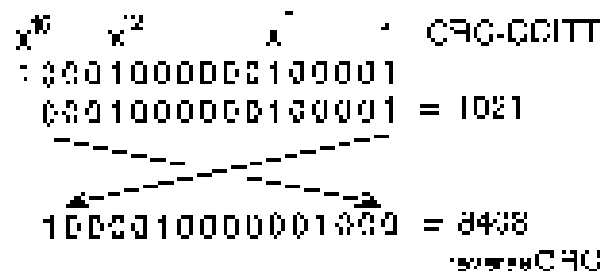


Tabelle 1: gängige Polynome

Aus ähnlichen Gründen wird in HDLC/SDLC und anderen neueren Protokollen das CRC-Wort vor der Übertragung bitweise invertiert.

Ausgehend von der ursprünglichen Hardwarerealisierung mit Schieberegistern wird das CRC-Wort immer mit dem MSB zuerst übertragen. Die Datenbytes können jedoch entweder mit MSB oder mit LSB zuerst gesendet werden. In industriellen und Telecomanwendungen ist LSB zuerst üblich. MSB zuerst wird jedoch in Fileübertragungsprogrammen wie XMODEM verwendet.

Anwendungen

Die Verbreitung dieser CRCs ist eng mit ihrer Anwendung in seriellen Protokollen verbunden. Das älteste davon ist Bisync. Es wurde ursprünglich 1968 von IBM für Terminals eingeführt. Obwohl technisch überholt verschwindet es nur langsam. Bisync verwendet alternativ CRC-12, CRC-16 oder VRC/LRC. Letzteres ist keine CRC sondern eine einfache XOR-Prüfsumme.

CRC-16 tritt noch in weiteren, meist in den USA entstandenen, Protokollen auf. Z.B. dem Filetransferprogramm YMODEM und dem Modemprotokoll MNP.

In neueren Anwendungen wird es jedoch zunehmend von CRC-CCITT verdrängt. Die meiste Bedeutung werden in Zukunft wahrscheinlich die um HDLC gruppierten Protokolle haben.

Bild 1:
CRC durch
Schieberegister
umsetzt,
in Kanal über-
tragen
und wieder
dekodiert

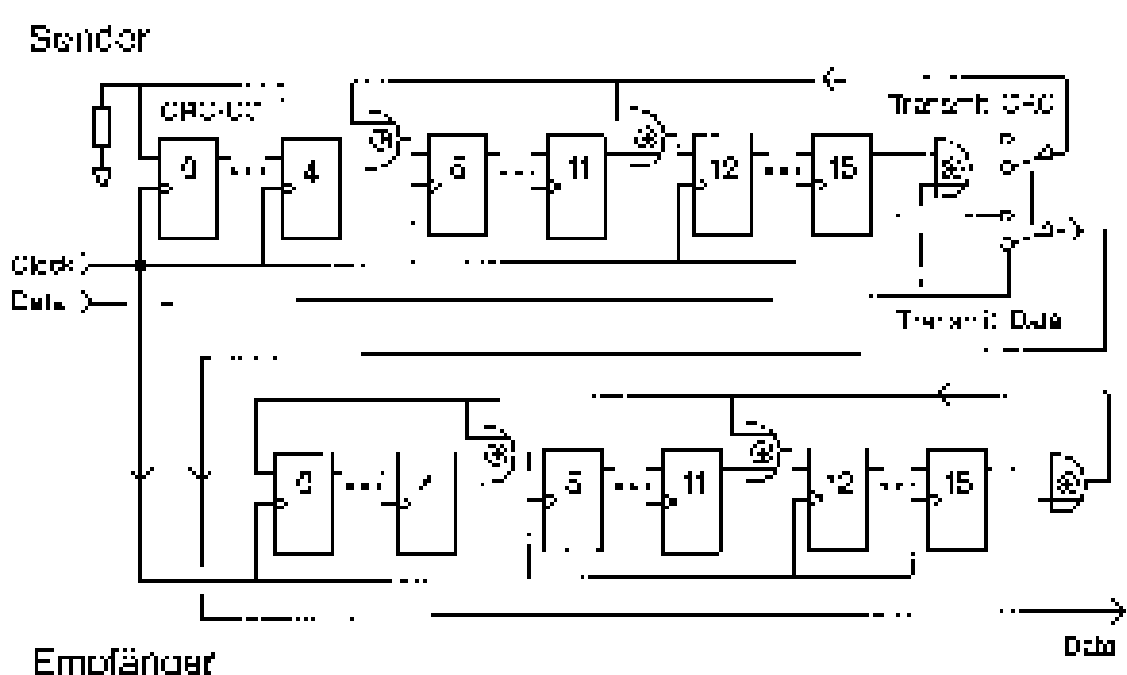
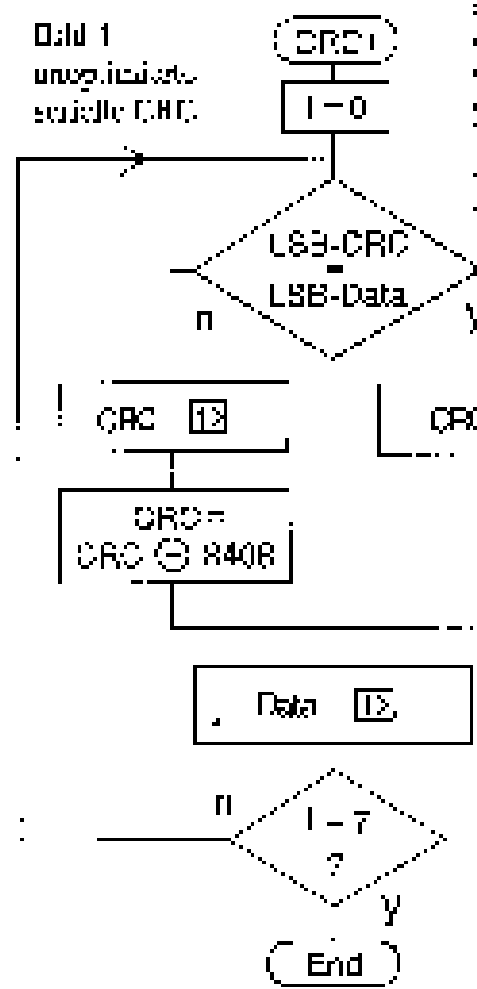


Bild 5
Reihenschaltete XOR und Shift
ist verdrahtet

$$(A \oplus B) \ll 1 = (A \ll 1) \oplus (B \ll 1)$$

Bild 1
unvollständig
schaltete CRC



Eintragungswert 0 oder 1 und die Rück-
kopplung angewandt. Dadurch die verändertert
wird. Im nächsten Schritt wird eine der
Rechte auf „Transmit Data“
gestellt und der „Tipflugs“ n Punkte
und Empfänger zurückgewendet. Wenn
weder die „Rechte“ im Schieberegis-
ter „Rechts“ und gleichzeitig „Rechts“
ist. Die Länge des Datenworts ist
dabei notwendig. Anschließend wird die
Rechte auf „Transmit Data“ gestellt.
Mit 16 weiteren Takten wird nun die
Rechte übertragen. Das Schieberegis-
ter ist im Empfänger. Dort wird Daten
mit CRC aufgenommen. Wenn beide
Rechte auf „Rechts“ ist, stehen sich im
alle Tipflugs wieder auf Null.

Serialer CRC

Man kann diese
Hardware recht einfach als Schieberegis-
ter umsetzen. Das Programm ist als
eine Version von CRC-16 in Listing
CRC-16 dargestellt. Als Schieberegis-
ter verwendet man ein 16-Bit-Register.
Die Daten werden bitweise
übertragen und das LSB wird zuerst
ausgegeben. Das Schieberegister
wird zunächst die Konstante 8408
als seine Funktion definiert, die
immerhin umgedreht (Bild 1). Damit
steht am MSB mit dem LSB die
Daten überein. Im Programm wird
geprüft, ob die LSB- und CRC- und
Daten übereinstimmen (Bild 4). Das

erzeugt dem XOR-Gate nach
Folger 15. Nur in diesem Fall
werden die anderen XOR-Gates aktiv.
Sie können in Steuerung, parallel zum
die XOR-Gate mit 2-Bit-Startwert 8408
mit der CRC durchgeführt werden.
Nur wenn die CRC stimmt, nach
rechts geschoben. Dabei wird links
eine Null ins MSB eingefügt.

Das Regele CRC-16 umfasst
sogar 16 Bit der CRC-Wert, der
links des Datenwortes. Als erstes Wort
nach dem die Initialisierung, das ist
CRC-16. Nach dem ersten
Datenwort wird die endgültige CRC, die
vom ersten übertragen ist, zur
Verfügung. Zur Umwandlung von CRC-
16 auf CRC-16 macht man die
Konstante 8408 wieder verwendet.
Das nur die Reihenschaltete von Schieberegis-
ter und XOR-Operationen
werden kann (Bild 5). In einer
Optimierung von CRC-16 macht man
die zweite Version von CRC-16
in Listing aufgeführt ist.

CRC mit Tabelle 2

Obwohl die CRC-16 sehr
schnellen Anwendungen wie
Speichererkennung und -schutz
weil, werden parallel, bitweise
übertragen. Aber kein Schieberegis-
ter, sondern ein Ausgangsregister
und ein ziemlich komplexer
Dekodierlogik. In Bild 6 wird die

Bild 7: CRC mit Tabelle für LSD first.

$$CRC = CRC \oplus \text{Table}(CRC \oplus \text{Data})$$

$$\text{Table}(0) = 0000 \oplus \text{CRC} \oplus 00$$

$$\text{Table}(1) = 0001 \oplus \text{CRC} \oplus 00$$

...

$$\text{Table}(FF) = 00FF \oplus \text{CRC} \oplus 00$$

Ein gutes Beispiel für die Berechnung ist die CRC-BEST für die Zahl 26. Verfolgen Sie den Ablauf der Berechnung so weit, bis Sie zum Ergebnis im Bild gelangt.

Auch in Software gibt es eine schnelle Methode die nur ein einziges Zitat hat: Die CRC wird einfach im Programm definiert. Die Tabelle belegt für eine 16 Bit CRC nur 256 Bytes. Sie wird durch den Befehl #define CRC mit Verwendung eines speziellen Makros von Code mit Daten gefüllt. Für AVR-Mikro und AVR-16-Bit werden durch gcc/mavr benötigt, die Tabelle kann ins ROM übernommen werden. Für eine Version von Code zur Lösung besteht daraus per Tabelle die CRC besonders schnell. In Bild 8 ist oben die Formel dazu dargestellt. Darunter

das Layout der Tabelle. Der Befehl CRC wird durch die Berechnung CRC-Wert mit einem Datenbyte XOR kombiniert.

Linker und Rechter Teil mit dem Wert nach dem ersten Programm in Assembly. Lassen Sie sich 5 Bit CPU implementieren wie in Listing ASM-CRC1 dargestellt.

Test

Im System simuliert man die Initialisierungswerte. Die CRC-Wert steht in der CRC die übertragen wird. Der Empfänger simuliert nur die Initialisierungswerte und die übertragene CRC. Dadurch ist die CRC 0000, wenn alle

Bild 8: CRC mit Tabelle für MSD first.

$$CRC = CRC \oplus \text{Table}(CRC \oplus \text{Data})$$

$$\text{Table}(0) = 0000 \oplus \text{CRC} \oplus 00$$

$$\text{Table}(1) = 0100 \oplus \text{CRC} \oplus 00$$

...

$$\text{Table}(FF) = FF00 \oplus \text{CRC} \oplus 00$$

Bit im aufgegebenen CRC. Sie ist auch dann 0000, wenn auf FFFF initialisiert wurde. Im CRC-System wird jedoch das CRC Wert zuerst durch den Sender invertiert. Dadurch ist hier der gewünschte Wert nicht 0000 sondern der inverse. Zur Hilfe von nach Bitverarbeitung FIBS.

MSB first

In File-System-Programmen werden die Daten oft im MSB zuerst übertragen. Es ergeben sich umfangreiche Unterschiede in der Berechnung der CRC, sowohl im Listing CRC-7 als alle Programme nachfolgend. Für diese Variante sind die Initialisierungswerte nicht mehr auf 0000 sondern auf FF00, weshalb 0000 nicht der Wert am Ende. Generierpolynom genannt. Der Initialwert muß jedoch im Register mit 0000 nicht mehr gegeben werden, damit es nicht die LEDs überfordert.

32 Bit

Auf einem 16 Bit CPU kann 32 Bit CRC implementiert werden. Die Lösung CRC7 und CRC12 geben passende Programme für 32 Bit und MSB first. Die Tabelle für Version 3 belegt nur 128 Bytes.

CRC7 ist die Nachweise von CRC12 auf 7 Bit, auf Tabelle steht:

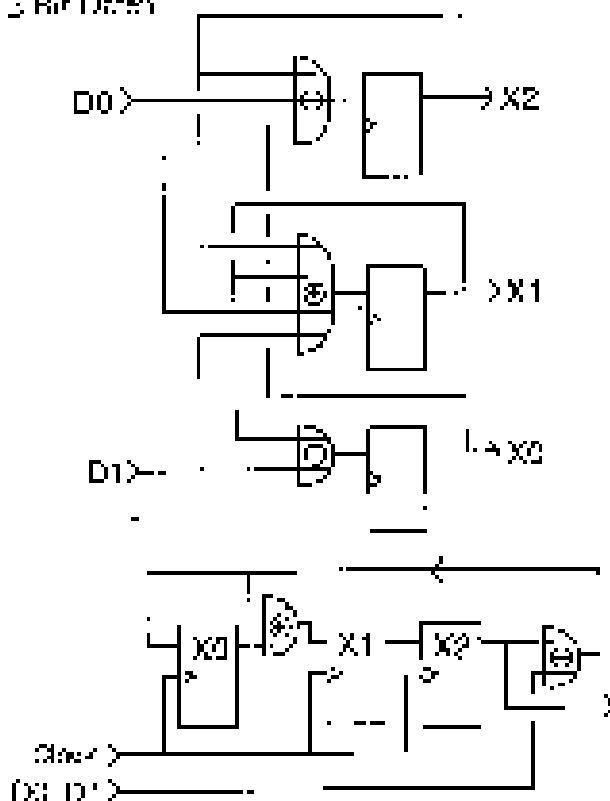
Manchmal noch 2 Programme geladen.

$$CRC-7 = x^7 + x^6 + x^4 + 1$$

$$CRC-8 = x^8 + x^7 + x^6 - x^5 - x^2 - 1$$

Bit 6

Serial und parallele Implementierung einer 3 Bit CRC für 3 Bit Daten



PAUSE

Round-robin Multitasker haben sich in FORTH seit den 70er Jahren hinweg in vielen Anwendungen bewährt. Die FORTH-typische Implementierung geht auf Charles Moores poly-FORTH zurück, wurde aber erst durch F83 von Laxen & Perry für den IBM-PC einem breiteren Anwenderkreis bekannt.

Da sich das Verfahren besonders für Controller eignet, und gut auf die begrenzten Möglichkeiten von 8 Bit CPUs abgestimmt ist, wird es unter diesem Gesichtspunkt hier näher beschrieben.

Wie in Bild 1 dargestellt, erfolgt der Taskwechsel durch den Befehl PAUSE den der Programmierer in die Software einstreuen muß. Der Programmfluß bewegt sich dabei zyklisch durch alle Tasks. Der Wiedereinsprung erfolgt unmittelbar nach PAUSE. Dazu ist kein Eingriff aus der Hardware, z.B. durch einen Timer-interrupt, nötig. Es handelt sich also um einen kooperativen Multi-tasker: das Programm muß geeignet angepaßt werden, damit es parallel zu den anderen Programmen laufen kann.

Diese Anforderung ist auf einem Controller relativ leicht zu erfüllen. Die Software steht hier fest im EPROM und unbekannte, fremde Programme kommen nicht zu Einsatz. Bereits vorhandene Software multitaskfähig zu machen ist meist simpel. Es müssen nur die PAUSE-Befehl eingefügt werden. Sie verändern die Funktion des ursprünglichen Programms nicht.

Da der Programmfluß nur dort unterbrochen wird, wo der Programmierer es für richtig hält, treten einige Probleme von preemptive-Multi-taskern gar nicht erst auf. Insbesondere steht PAUSE genau zwischen zwei Befehlen des Programms. Die meisten lokalen Variablen und CPU-Register werden nur innerhalb eines FORTH-Befehls verwendet und müssen somit nicht gerettet werden. Der Taskwechsel ist deshalb sehr schnell. Programmgebiete in denen kein Taskwechsel erfolgen darf,

in denen Operationen nicht explizit für Wiederherstellung des Zustandes im Unterbrechungszustand sind, müssen durch den Programmierer in die Unterbrechungsroutine einbezogen werden. Da der Multitasker selbst seine Aufgaben erledigt, sind die Unterbrechungsroutinen mit Programmteilen im Unterprogramm verbunden.

Die Unterbrechungsroutine kann durch den Programmierer verändert werden. Viele FORTH-Befehle in dieser Routine, die auf Befehl ausgerufen werden, sind durch den Programmierer modifizierbar. Programmteile die durch den Multitasker verwendet werden können, sind durch den Programmierer zu ändern. Es sind nur wenige dynamische Unterbrechungsroutinen notwendig, die für die Unterbrechungsroutine des Multitaskers verwendet werden können.

Die dynamische Unterbrechungsroutine des Multitaskers wird durch den Programmierer modifiziert, um die Unterbrechungsroutine des Multitaskers zu ändern.

PAUSE

Die Unterbrechungsroutine des Multitaskers muß so die jeweilige Unterbrechungsroutine des Multitaskers modifiziert werden. Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft.

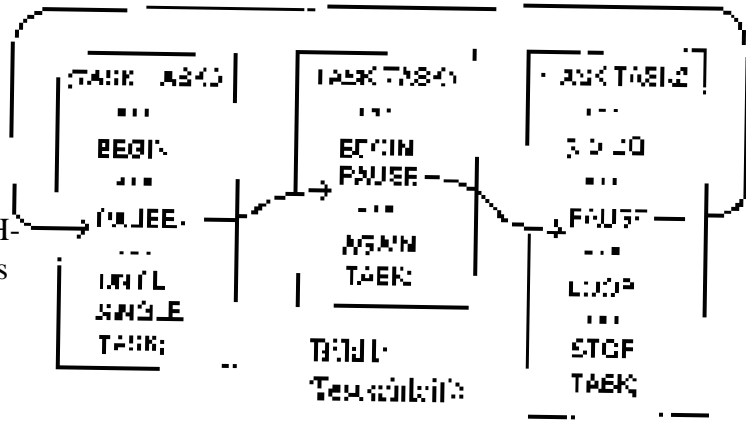
Die Unterbrechungsroutine des Multitaskers muß zwei Aufgaben lösen: Umkehr des Programmflusses und Unterbrechung des Multitaskers.

Die Unterbrechungsroutine des Multitaskers muß die Unterbrechungsroutine des Multitaskers modifizieren. Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft.

Das Stück der CPU unterbrechen. So wird in der CPU der Multitasker unterbrochen. Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft.

Wenn diese Unterbrechungsroutine nicht existiert, wird die Unterbrechungsroutine des Multitaskers nicht gefunden. Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft.

Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft. Die Unterbrechungsroutine des Multitaskers ist ein FORTH-Programm, das auf einem Controller (z.B. Laxen & Perry) läuft.



Auch in FORTH sind selber wieder ausführlich um PACTE und Maschinen und Schrittbefehle gesprochen. In die hier besprochenen einfacheren Beispiele wird die EN-TAB mit einer Task-Nummer versehen. Das ist problemlos, weil man eben in allen Tasks EN-TAB-Adressen verwendet, und man diesen Befehl im FORTH-Stack zu verfahren, es zeigt, inwiefern in einer Tabelle abgelegt werden für die mit dem Task-Nummer angegeben wird.

Verwaltung

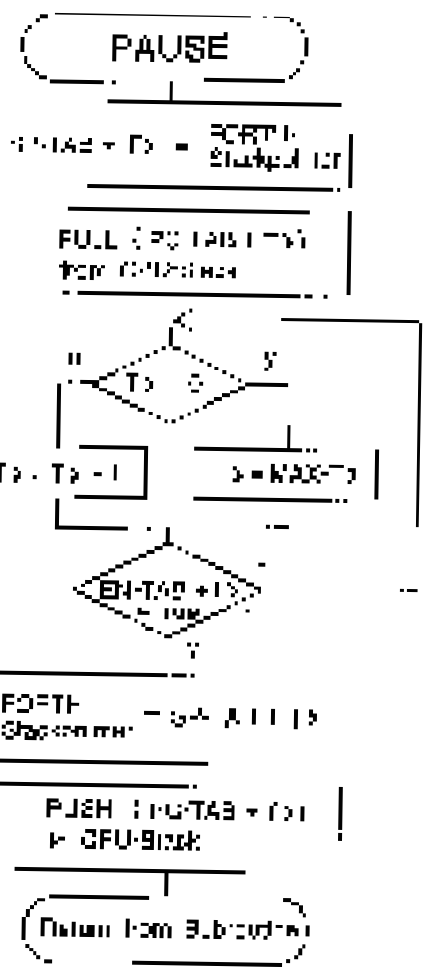
Es besteht zwar nur PACTE und die Tabellen die in die Taskwechsel, Push, wenn Befehle und Daten Daten sind jedoch für die sie sind nur die und Auswechseln der Tasks. In Bild 6 sind sie in PACTE-Programm dargestellt. Die RAM befindet sich am Kopf der RAM-Tabelle. Die zur Initialisierung dient, die Tabellen mit den EN-TAB-Adressen

wenden, bevor man den Multitask-Modus kann. SOLLTE diese die Ausführung der Task-Schritte. Ist in hier angegeben, daß ein Task 0 geteilt wird, für START erfolgt mit dem Multitask-Modus und wird in der CPU-Interpretation zu sein. Der Befehl muß in Task 0 ausgeführt werden.

START und die anderen Tasks, die auf dem Task-Nummer identifiziert, und es wird als, diese was für den EN-TAB-Flag, muß die Task-Nummer in dem PACTE weiter bei dem es eingetragener werden.

Das START Programm stellt, wie ein Kopf der Task-Nummer zum Initialisieren in die RAM-Tabelle, START zeigt die Liste in der diese Befehle benutzt werden.

Mit dem EN-TAB wird eine Task definiert. Die Daten liegen am Kopf vom System und werden ebenfalls nicht als Initialisierungsprogramm dargestellt. Das Beispiel in die Tabelle zum großen Überblick über diese Daten Compiler des neuen



Task wird eine Zahl auf dem Stack übergeben, die die Anzahl der 16 Bit-Byte enthält die es sich für diesen Stack reservieren darf. Es sollte typischerweise 256 sein. Wenn diese Zahl von dem Kopf des neuen Stacks weicht, wird der Stackpointer der nächsten Task entsprechend modifiziert.

Die Tabellen SP-RAM und EN-TAB werden zur Aufgabe des PACTE mit Hilfe in einer CPU-Stack sein, heißt Anfang immer Task 0, dem die Nummer des neuen Task. Diese Zahl wird von nach dem Kopf von MAX-T0 als Konstante interpretiert. Unmittelbar unter diese Konstante zeigt das Sprungziel des neuen Programms. Die Tabelle EN-TAB zeigt die EN-TAB-Adressen der Task. Wenn der Name der Task angegeben wird, erfolgt die keine direkte Ausführung sondern nur die Task-Nummer wird auf den Stack gelegt. Dies ist die Zahl MAX-T0. EN-TAB und SP-TAB sind zusammen mit dem EN-TAB-Adressen werden. Es ist klar, dass die Task-Nummer mit dem Befehl EN-TAB.

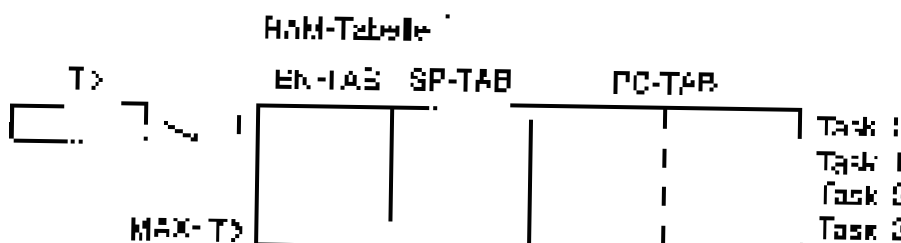
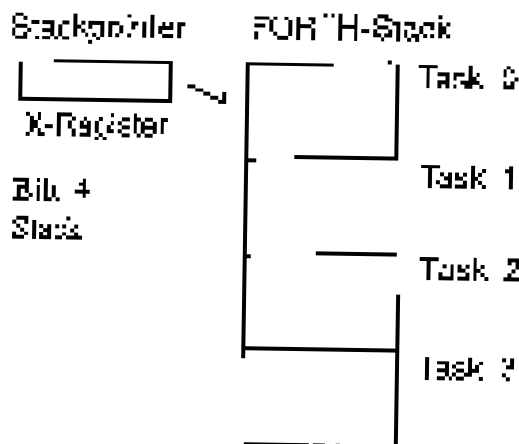
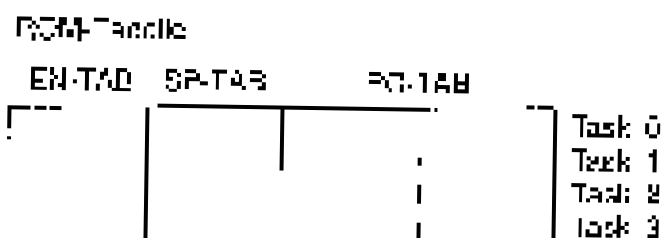


Bild 3: Tabellen



Alternativen

Beim Vergleich mit prinzipiell Multiskalen sollte man die unterschiedlichen Konzepte im Auge behalten. Zwei schichten langsamere und viel kleinere, aber lokal höhere Zeitschichten anbieten. Damit gibt das System ein bestimmtes Zeitverhalten, was bei jeder Funktion, wenn die Zeit nicht zu lang ist, ein bestimmtes Verhalten zeigt. Die Aufgabe, schneller zu sein, kann durch die Multiplikation zwischen den Zeitschichten erreicht werden. Dafür ist keine Zeitplanung nötig.

Prinzipiell Multiskalen werden meist von Kombinationen von zwei Zeitschichten erreicht, wobei die Zeitverläufe der beiden Funktionen durch realisierte Reihen und die Typen der Komponenten klar definiert. Das kann die Kommunikation zwischen den Zeitschichten erleichtern.

Die Realisierung ist, wie alle anderen Anwendungen von Multiskalen, ein Prozess der Realisierung. Die Realisierung des Verfahrens muß die Anforderungen der jeweiligen Anwendung erfüllen werden.

[1] F. C. Ting, „Aspects of the Theory of the Multiskal“, 1986

[2] W. Röss, „Multiskalierung in der Technik“, Elektronik 5/1992, S. 10-12, aber eine der seltenen Darstellungen zum Multiskalen in Fachzeitschriften.

„FOCUS“ ist ein Anzeigegerät von TRW 70 978 „Implem“ und wird mit Hilfe von unterschiedlichen die Karte für die Verfügung gestellt.

Gray-Code

Ein Winkelcodierer ist ein Sensor mit dem man die Drehung von Achsen bestimmen kann. Bei einem absoluten Geber sogar wenn die Achse steht. Er übergibt ein digitales Datenwort von 6 - 14 Bit Breite das oft als Gray-Code formatiert ist. Dieser hat gegenüber dem üblichen Binärcode den Vorteil, daß sich zwischen aufeinanderfolgenden Zahlen jeweils nur eine Stelle ändert („einschrittiger Code“). Damit verhindert man an der Schnittstelle das Auftreten fehlerhafter Zwischenwerte. Es gibt für viele Wortbreiten mehrere Codevarianten, die die Eigenschaft besitzen. Der „echte“ Graycode wird häufig auch als reflected Graycode bezeichnet. In Bild 1 seine Werte für 4 Bit.

Zur Weiterverarbeitung werden die Daten jedoch normalerweise binär benötigt. Neben der Hardwarelösung (Bild 2, 3) findet sich in [1] auch eine effiziente Softwarelösung die im Listing GRAY auch in FORTH dargestellt ist.

Da es sich um einen zyklischen binären Code handelt, gibt es auch eine serielle Implementierung [2] bei der die „Arithmetik“ durch ein JK-FlipFlop realisiert wird. Zusammen mit einem Schieberegister kann man damit auch Parallelworte verarbeiten (Bild 4, 5).

- [1] Herbert Wehlan „Programm zur Gray-Code-Decodierung“ Elektronik 1981 S.97
- [2] Texas Instruments „Designing with TTL Integrated Circuits“ McGraw-Hill 1971

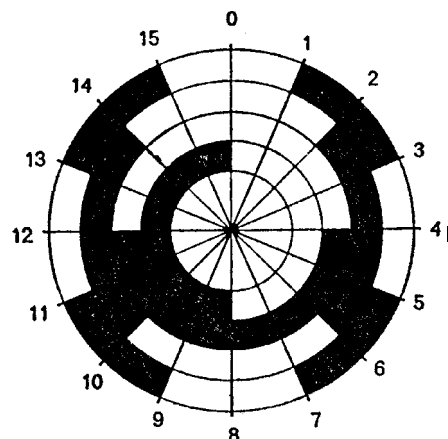


Bild 1: 4 Bit Gray-Code

	G0	G1	G2	G3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	0	1
6	0	1	0	0
7	0	1	0	0
8	1	1	0	0
9	1	1	1	0
A	1	1	1	1
B	1	1	0	1
C	1	0	1	0
D	1	0	1	1
E	1	0	0	1
F	1	0	0	0

Bild 2: Binär auf Gray

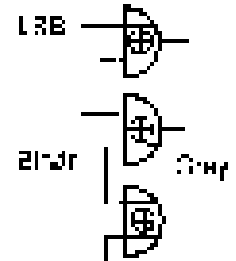


Bild 3: Gray auf Binär

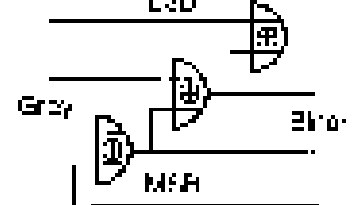


Bild 4: Gray auf Binär seriell mit Schieberegister parallel

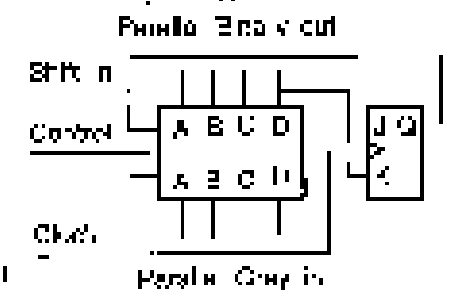
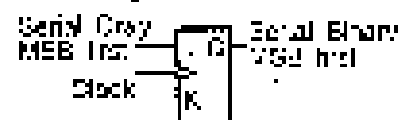


Bild 5: Gray auf Binär seriell

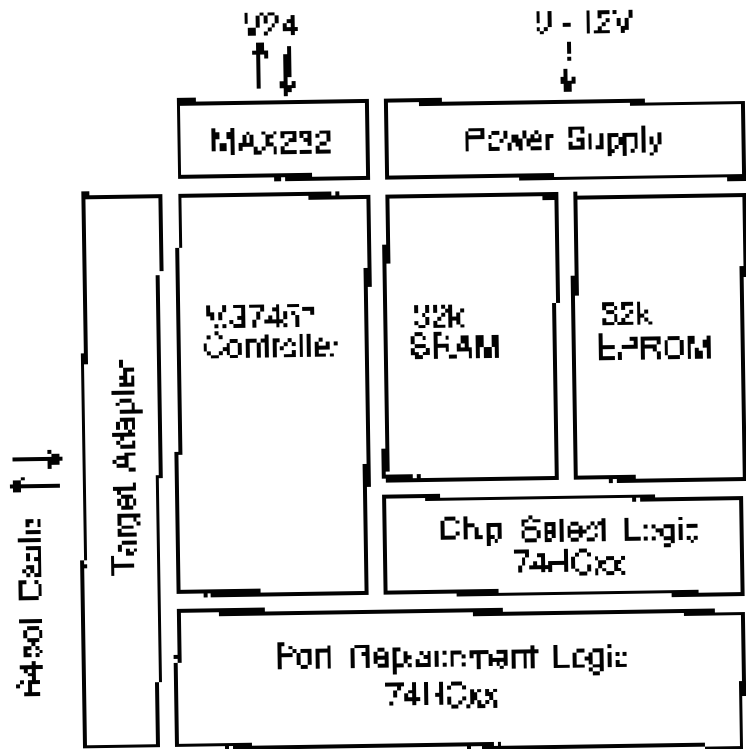


Emulator

Um mit OTPs zügig entwickeln zu können ist ein Emulator wünschenswert. Die käuflichen sind leider recht teuer. Da man für jedes neue Projekt die neueste und geeignetste CPU wählen will macht die Amortisation eines bald veralteten Emulators um so schwieriger. Jedoch ist Eigenbau oft möglich, wie hier an dem 6502 kompatiblen Mitsubishi-Controller M37451 gezeigt wird.

Er dient dabei als Beispiel für verschiedene Serien von Mitsubishi bei denen man nach dem gleichem Schema vorgehen kann. Gemeinsam ist ihnen, daß sie im Shrink-DIP64 Gehäuse untergebracht sind und damit über besonders viel Portpins und Peripherie verfügen. An Speicher sind beim M37451 intern bis zu 512 Byte RAM und 24k EPROM vorhanden. Man kann die Applikation also nicht nur in Assembler, sondern auch in FORTH erstellen.

Bild 1: Blockschaltbild



verdichtet enthält. Dieses wird auf eine Leiterplattenfläche montiert mit der Adressdecodierung, Portersatzleitungen und Adaptersockel aufnimmt.

Da die Elemente etwa 5mm breit sein dürfen, was man sich so spät als Erläuterung für Quellcode handhaben kann. Die Adapterbaugruppe muß beidseitig einen Lötlager haben. Die 16 Portpins sind beim M37451 und M37452 aus der Packung herausgeführt. Als 800V-Fixierung im Zielsystem sind gedrehte Sockel für 8mm statt 9mm als Inverse-sockel, weil sie höhere Hermeticität haben. Die Lötung durch den Sockel für das Fluidhandwerk ist ein weiterer Lager als der 3-VP-Sockel. Beim Layout des Zielsystems sollte man in dieser Bereich keine Bauteile verwenden.

Einplatinencomputer

Durch Umverdrahten eines Pins wird der Controller auf externen Busbetrieb umgeschaltet, so daß das interne EPROM ignoriert wird. Dabei fallen 3 Ports flach die nun als 16 Bit Adreßbus und 8 Bit Datenbus arbeiten. Sie müssen vom Emulator als Portnachbildung wiederhergestellt werden. Die Steuerleitungen Phi und R/W sind am Gehäuse ohnehin herausgeführt. Da es sich um einen nichtgemultiplexten Busanschluß handelt, ändert sich nichts am Timing der CPU.

Man kann nun den Controller extern mit 32k SRAM und einem 32k EPROM mit FORTH ergänzen. Zugriff erfolgt über V24 mittels der UART des Controllers. Man hat so den OTP zu einem Einplatinencomputer aufgeböhrt um sich komfortablere Testmöglichkeiten zu schaffen. Um den Verdrahtungsaufwand zu reduzieren, empfiehlt es sich ein Board zu schlachten das bereits den Controller und die Speicherbausteine passend verdrahtet

Adapter

Als Schnittstelle zum Zielsystem stellt man eine 2reihige 64polige B-Bleiung vor die in die Einleitung des Controller einsteckt. Abgesehen von den beiden Pins des Quarzoscillators werden alle drei Pins zu einem 16-Bit-Datenbus über den der Parallelbus der CPU angeschlossen ist. Die 30-40cm lange Leiterbahn ist im unteren Bereich des Boards zu finden. Richtiges einseitiges Fluidwerkzeug ist besser als beidseitiges weil billiger.

Den Adapter mit der SUB64-sockel des Zielsystems muß man sich als eigene Leiterplatte herstellen. Die 16-Bit-Datenbus der CPU wird über die 16-Pin-Headerleitung des Zielsystems angeschlossen. Die 8-Bit-Datenbus der CPU wird über die 8-Pin-Headerleitung des Zielsystems angeschlossen. Die 16-Bit-Adreßbus der CPU wird über die 16-Pin-Headerleitung des Zielsystems angeschlossen. Die 8-Bit-Adreßbus der CPU wird über die 8-Pin-Headerleitung des Zielsystems angeschlossen. Die 16-Bit-Adreßbus der CPU wird über die 16-Pin-Headerleitung des Zielsystems angeschlossen. Die 8-Bit-Adreßbus der CPU wird über die 8-Pin-Headerleitung des Zielsystems angeschlossen.

Low Power

Einige CPU-Systeme können mit einer 5V Versorgung betrieben werden. In diesen Fällen sind die entsprechenden Logikbausteine zu verwenden. Aber auch bei Emulatoren mit 12V Versorgung kann man die Spannung reduzieren. Die meisten Chips sind mit einer 12V Versorgung ausgelegt, aber es gibt auch Chips, die mit einer 5V Versorgung ausgelegt sind. Bei der Verwendung von 5V Versorgung ist die Chipselect-Logik im Emulator zu berücksichtigen.

V24-Schnittstelle Teil 1

Für viele Einplatinencomputer muß man sich eigentümliche Adapter löten, damit man sie über V24 mit dem PCs verbinden kann. Entgegen der oft geäußerten Meinung liegt das Problem nicht bei den Normen, sondern in der Unwissenheit mancher Entwickler.

Die V24 bzw. RS232 war an Terminals und später PCs als Schnittstelle zu Modems vorgesehen. Anfangs wurde dafür ein 25poliges Kabel verwendet. Am Terminal/PC ist ein SubD- 25-Stecker angebracht und am Modem eine SubD25 Buchse. DTE („Data Terminal Equipment“) ist dabei die Bezeichnung für die Terminalseite, DCE („Data Communications Equipment“) für die Modemseite. Das Kabel hat an einem Ende einen Stecker, am anderen eine Buchse und verbindet jeden Draht 1:1 durch.

Warum so viele Pins ? Zu Zeiten der 300 Baud Modems (V21 von anno 1964) enthielten diese keinen Mikrocontroller, sondern viel diskrete Analogschaltung und ein bißchen Logik. Über den vielen Kabel konnte man Takt und Steuersignal zur Verfügung stellen und sie damit kosteneffektiv bauen.

IBM hat mit der Einführung des PC-AT eine neue Pinbelegung geschaffen die SubD9 Steckverbinder verwendet (Bild 1). Das Wehgeschrei der Postbehörden von wegen Normverletzung war groß, aber letztlich hat IBM der V24 damit einen Dienst erwiesen. Am 25poligen Verbinder waren nämlich immer weniger Signale tatsächlich implementiert worden, weil die Modems sie nicht mehr brauchten. Das hatte zu Inkompatibilitäten geführt. Die 9 Pin

Bild 1: Pinbelegung

Untermenge hat wieder für einen sinnvollen Funktionsumfang gesorgt. Im Folgenden werden deshalb nur noch diese Pins behandelt.

Auf einem Einplatinencomputer sollte man die Schnittstelle des Modems nachzubilden, weil man dann die üblichen Kabel verwenden kann.

EMV

Der 25pol Steckverbinder hat manchmal einen Vorteil: sein Pin 7 („Signal Ground“) ist die elektrische Masse für die Treiber, während sein Pin 1 („Chassis Ground“) der Schirm des Kabels ist. Beide bleiben im Kabel getrennt. Wenn man die Steckverbinder der Geräte direkt aufs Gehäuseblech schraubt, erhält man schöne Faradaysche Käfige um beide Geräte und das Kabel. Beim 9pol Steckverbinder sind beide Massen im Kabel auf Pin 5 verbunden. Das sorgt zwar auch für Schirmung, aber zusätzlich einen unerwünschten Erdungspunkt der zu Masseschleifen führen kann.

Einschränkend ist allerdings zu sagen, daß Kabel meist vom Endanwender mit Blick auf Kosten und damit ohne Schirmung gekauft werden. Man die Entstörung also besser mit Drosseln im Gerät vornimmt.

SubD 9	DCE Modem Buchse	DTE Terminal Stecker	Signal
3	in	out	TXD
2	out	in	RXD
7	in	out	RTS
8	out	in	CTS
6	out	in	DSR
5			GND
1	out	in	DCD
4	in	out	DTR
9	out	in	RI

Signale

Die minimale Untermenge besteht aus den beiden bidirektionalen Datenleitungen RXD, TXD und der Masse GND. Die restlichen jedoch DCD, DTR mit nur dieser Verbindung eine Steuerung zum möglich ist. DSR und manche Terminalprogramme wollen zusätzlich Handshaking (RTS, CTS) zeigen die Stromstärke und den typischen Anschluß (RTS, CTS) Terminal Stecker) signalisiert das Terminal das es bereit ist. Das Modem beantwortet mit DSR („Data Set Ready“). Dazu gibt das Terminal RTS („Request to Send“) und um das Modem zu sagen das es CTS („Clear to Send“) zur Deckschaltung kann im Programmieren eine Durchdrück von DTR auf DSR von RTS auf CTS zu sorgen, daß von das Terminal seinen Handshake selber erzeugt.

Weiter noch zwei Pins über die man sich kein DCE (Data Carrier Detect) weiß an, daß das Modem die Platte des anderen Modems im T. Während der Datenübertragung wollen manche Terminalprogramme diese Signale zu sehen, wie es immer noch einbez. Sekunden die Verbindung unterbrechen um Übertragungsfehler zu vermeiden. Man kann DCD aus z.B. auf die Linienleitung von DTR/DSR legen um dies zu realisieren.

Der Pin 4 (DTR) bedeutet: ist in den Anweisungen für gesetzt, daher ist, nur auf dem Terminal angeordnet sein & auf dem während des Kabelausschlusses z.B.

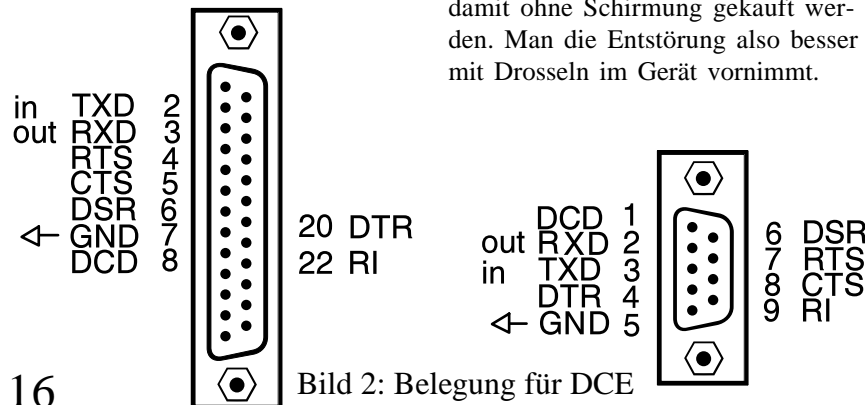


Bild 2: Belegung für DCE

Schrittmotoren

Teil 2

Nach der Praxis im letzten Heft nun ein wenig Theorie. Verständnis des Innenlebens eines Bauteils erleichtert seine Anwendung. Ein kurzer Überblick über Ansteuerung durch Schaltregler und die Anpassung der Schrittgeschwindigkeit (Teil 3) zeigt warum Controller durch ihre Flexibilität den bisher verwendeten Steuerschaltungen überlegen sind.

Bild 1 zeigt nochmal das Blockschaltbild vom unipolaren und bipolaren Schrittmotor. Die Hersteller liefern Motoren wahlweise unipolar oder bipolar. Es ist für sie nur ein Unterschied in der Wicklung. Bipolar erzeugt dabei mehr Drehmoment, braucht aber eine Brücke als Außenbeschaltung. In Bild 2 ist die prinzipielle Innenbeschaltung dargestellt. Offensichtlich bringen zwei Elektromagneten eine „Kompaßnadel“ zum Drehen. Der Schrittinkel beträgt hier unrealistische 90 Grad. Die Pfeile zeigen den Stromfluß an der durch die externe Brückenendstufe wählbar ist. Bild 4 zeigt die Wirkung auf die Kompaßnadel. Vollschritt mit zwei Phasen wurde in der letzten Ausgabe durchgeführt. Der Vollschritt mit nur einer Phase hat ein geringeres Drehmoment, aber natürlich auch weniger Stromaufnahme.

Halbschrittbetrieb

Man kann beide Vollschrittversionen zusammenfassen und erhält dann Halbschrittbetrieb. Damit hat sich die Auflösung des Schrittinkels verdoppelt.

Zur Implementierung genügt es die Tabelle STEP-Table von 4 auf 8 Werte zu erweitern, indem man die geeigneten Zwischenwerte einfügt (Listing STEP1.F74). Sowie dafür sorgt, daß der Zeiger nun 8 Werte ansprechen kann. Die erzielbare Genauigkeit hängt von der Qualität des Motors ab. Beide Spulen müssen sich genau entsprechen, damit der Zwischenschritt auch in die Mitte weist.

Man kann das Prinzip verfeinern, indem man den Strom beider Spulen über z.B. 6 Bit D/A-Wandler steuert. Damit wird dann „Minischrittbetrieb“ möglich. Erfordert

aber einen guten Motor und eine aufwendige Elektronik. Die Teilschritt sind meist nicht sehr genau. Jedoch hat diese Betriebsart den Vorteil, daß man durch die kleinen Schritte eine gleichmäßige Bewegung erhält und damit die Resonanzerscheinungen vermeidet, die bei großen Schritten auftreten.

Schaltregler

Damit sich der Motor schnell bewegen kann, muß man die Schrittzeit kurz halten. Die Induktivität der Spule brems jedoch den Stromanstieg (Bild 5). Dagegen hilft nur eine hohe Betriebsspannung um die verschliffene Flanke steiler zu bekommen. Bild 6 zeigt eine Schaltung die das bewirkt und durch den Vorwiderstand den Haltestrom wieder auf Normalwert reduziert. Der Vorwiderstand wird jedoch Verlustleistung erzeugen. Eleganter ist es deshalb auf einen Schaltregler zuzugehen. In

Bild 2: unipolarer Stepper

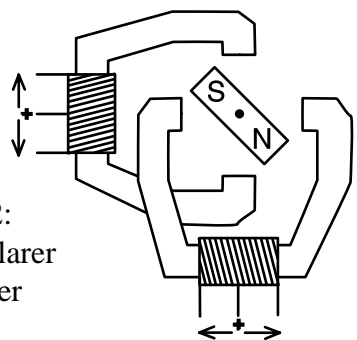


Bild 3: bipolarer Stepper

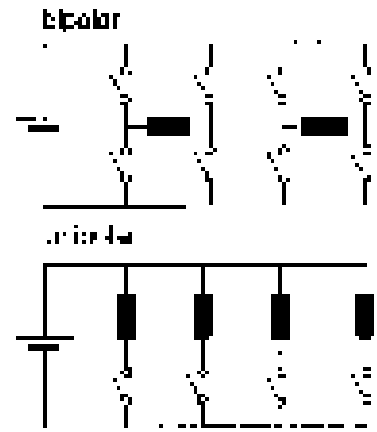
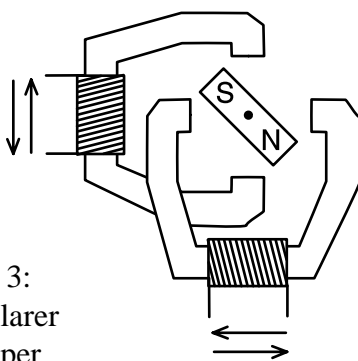


Bild 1: Ansteuerung

Leistungs erzeugen. Eingangs- und Ausgangsstrom auf einen Schaltregler übertragen. In Bild 7 kann man sehen, wie ein digitaler Signalgeber für den Stromfluss sorgt, während die folgende PWM-Signale zusammen mit der Takt- und dem Motor-Blockstrom erzeugen. Der Schaltregler arbeitet schaltweise über die Halbleiter und die Induktivität des Motors. Die Leistung des Schaltreglers ist durch den Vorwiderstand reduziert. Die Leistung des Schaltreglers ist durch den Vorwiderstand reduziert.

Bild 5: Induktivität verformt

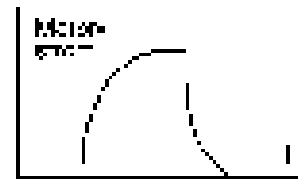


Bild 6: mit analoger Schaltung verbesserter Puls

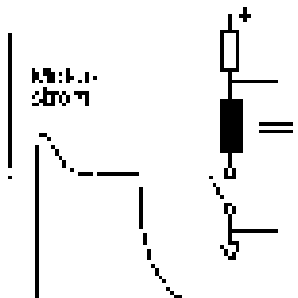


Bild 7: digital mit Schaltregler verbesserter Puls



Drittel Fasing

$$t_1 = 2\pi / \omega$$

$$HE_1 = \cos(\omega t) \quad HE_2 = \cos(\omega t + 120^\circ)$$

$$HE_3 = \cos(\omega t + 240^\circ) = \cos(\omega t - 120^\circ)$$

$$HE_1 + HE_2 + HE_3 = 0$$

$$\begin{aligned} HE_1 &= \cos(\omega t) = \cos(0^\circ) = 1 \\ HE_2 &= \cos(\omega t + 120^\circ) = \cos(120^\circ) = -0.5 \\ HE_3 &= \cos(\omega t + 240^\circ) = \cos(240^\circ) = -0.5 \end{aligned}$$

1. Schritt: Phase A aktiv

$$i_A = I_m \cos(\omega t) \quad i_B = i_C = 0$$

$$\text{STEP 1: } \cos(\omega t) \text{ AND } \cos(\omega t + 120^\circ) = \cos(\omega t) = 1$$

- $i_A = I_m \cos(\omega t) = I_m \cos(0^\circ) = I_m$ (Step 1: alle Pole sind aktiv)
- $i_B = i_C = 0$ (Step 2: alle Pole sind aktiv)

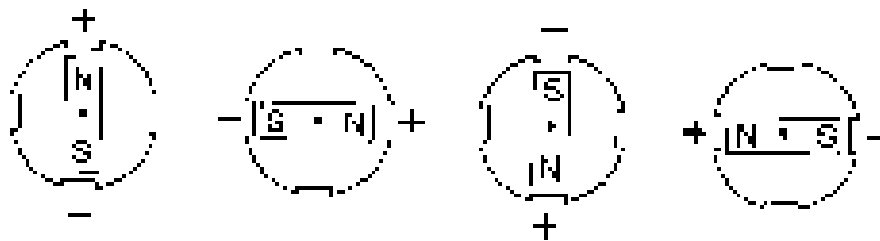
$$\begin{aligned} \omega t &= 0^\circ \Rightarrow \cos(0^\circ) = 1 \quad \cos(120^\circ) = -0.5 \quad \cos(240^\circ) = -0.5 \\ \omega t &= 120^\circ \Rightarrow \cos(120^\circ) = -0.5 \quad \cos(240^\circ) = -0.5 \quad \cos(360^\circ) = 1 \end{aligned}$$

$$\omega t = 240^\circ \Rightarrow \cos(240^\circ) = -0.5 \quad \cos(360^\circ) = 1 \quad \cos(480^\circ) = 1$$

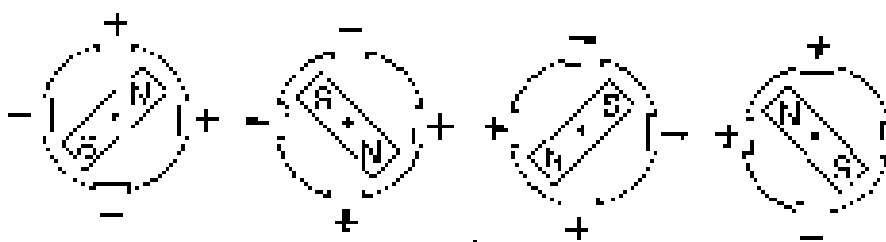
$$\omega t = 360^\circ \Rightarrow \cos(360^\circ) = 1 \quad \cos(480^\circ) = 1 \quad \cos(600^\circ) = 1$$

Vollschritt 1 Phase

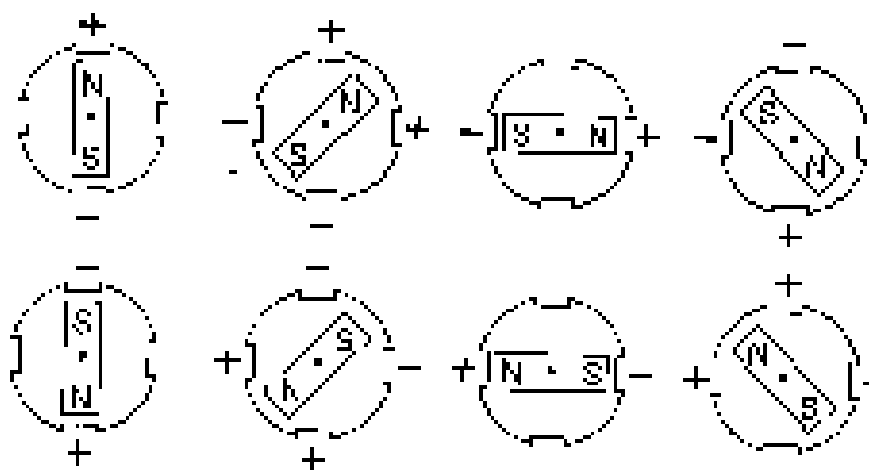
Dreh: Stellung der Magnete im Motor



Vollschritt 2 Phasen



Halbschritt



weitere die Endstufe nicht laufen lassen, sind die meisten Mikro bipolaren ICs auf 40MHz begrenzt. Man wählt nun die Bauausführung zu innen wie das Triac IC enthält, auf 10 oder 40V, um ein vernünftiges Tastenverhältnis zu bekommen, man sollte den Motor eine Spannung im Halbschritt von 30V bereit sein. Dieser Wert hilft, wenn man die PWM Signal während der Halbschritt wechsellast. Es Versorgungsspannung stehen, selbst, sich eine geeignete Übergangsspannung.

Es gibt nur zwei Möglichkeiten das PWM-Signal zu erzeugen. Entweder erst in Software, dann muß man einen hochfrequenten PIC oder AVR verwenden. Für ICs für PWM-Generierung sind es üblicherweise alle vorerst durch einen IC (optisch) unterrichten. Manche CPUs sind allerdings zu in Assembly programmieren. Die zweite Variante ist die d. PA Controller IC, in diesem IC bereits ein PWM-Generator ist. Das Indem IC hat meist einen geeigneten Stromweg für das Signal auf ein 4-Quadranten-Verfahren. Diese Übung ist ein bisschen kompliziert, aber programmierbar.

Um die Neutral zu erlösen, sollte man sich bemühen die beiden Sonden gegenseitig anzuschließen. Es geht eine geschaltete, so sind die an der Stelle sein.

Motordrehmoment

Ein wesentliches Problem bei Schrittmotoren, das sich vor allem bei niedrigeren Drehmomenten zeigt, ist die Überhitzung. Diese ist sehr ausgeprägt, wenn die Motoren für unübliche Leistungen laufen. Durch die Wicklung der einzelnen Phasen, die durch die Induktionseffekte, die durch die hohen Spannungen, die benötigt werden, erreicht wird, die Wärme der Phasenleiter, die bei hohen Leistungen zwischen den Wicklungen im Drehmoment des Motors sind, ist, die die Halbschritt, aber nicht die Rotationseffekte der Wicklung, die Angaben der Hersteller, aber die zu finden. Sehr wenige Motoren sind, die nicht übermäßig erhitzen, sondern die Frequenz mit aufgegebenen DC-Spannung

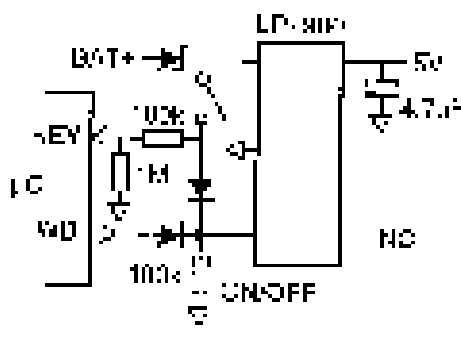


Bild 7:
 integrierter Spannungsteiler
 (20 - 200k) nicht 100k zeigt einen
 20-200k-Regler mit unigenem
 Spannungsabfall.
 Für höhere Ausgangs ist die
 Verwendung von 100k günstiger. Durch
 Bild 7: geringe Verluste im einen

Digitalsystemen Power-Ver-
 stärkung. Durch Drücken der Taste
 wird das Gate eingeschaltet. Dann
 muß die Ampere WD auf sich gelegt
 und der Widerstand des Prozessors
 geteilt werden. So ist der Widerstand
 kleiner, desto die Ausgangsleistung
 und die Geräte verliert Spannung

Mehr zu CORDIC

Für die praktische Anwendung
 obener Formeln kann man noch
 weitere Funktionen mittels CORDIC

$$e^x = \sin(\arctan(x)) + \cos(\arctan(x))$$

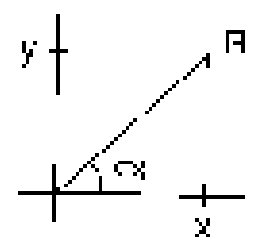
$$\ln(x) = 2 \cdot \arctan\left(\frac{x-1}{x+1}\right)$$

$$\sqrt{u} = \sqrt{\left(u + \frac{1}{4}\right)^2 - \left(u - \frac{1}{4}\right)^2}$$

$$\tan(\alpha) = \frac{\sin(\alpha)}{\cos(\alpha)}$$

$$\tanh(\alpha) = \frac{\sinh(\alpha)}{\cosh(\alpha)}$$

Bei CORDIC-Verfahren zur
 Erzeugung von Werten auf
 binärischer Kaskaden wird von
 CORDIC-Formeln direkt unter-
 strichen



$$\cos = x / R$$

$$\alpha = \arctan\left(\frac{y}{x}\right)$$

$$R = \sqrt{y^2 + x^2}$$

$$\sin = y / R$$

Nachtrag zu ②

Differenzierer

Das folgende ZK-Block für
 den in der Tabelle in Zusammen-
 hang mit PID-Regler ist

Differenzierer: Die Funktion über
 was die Verknüpfung von zwei
 einfachen Differenzierern, was für
 Regel mit abgeleitete gewinnet
 in Funktion zu

Für Differenzierer kann die
 doppelte Ableitung von Eingang
 abgeleitet werden

