Rolf-Dieter Klein, Joachim Arendt

Mikroelektronik im Fernsehen

Teil 4

mc begleitet die vom NDR produzierte Reihe „Mikroelektronik" monatlich. In dieser Folge geht es um den Aufbau einer Ampelsteuerung, um Roboter sowie um die Grafik-Karte mit dem Prozessor EF 9366. Im nächsten Heft widmen wir uns dem 68008-Computer.

Achtung, Schaltungsänderung!

Zur besseren Anpassung der Baugruppe GDP 64K an den Prozessor 68008 der Ausbaustufe des NDR-Klein-Computers ist es zweckmäßig, die Schaltung der Baugruppe GDP 64K zu verändern. Die veränderte Schaltung sichert selbstverständlich auch weiterhin die Verwendung der Baugruppe SBC 2 mit dem Z80A-Baustein. In einer späteren Auflage der Leiterplatte GDP 64K wird die Schaltungsänderung berücksichtigt werden. Wie man die vorliegende Leiterplatte beziehungsweise Baugruppe leicht selbst ändern kann, wird im folgenden gezeigt.

Der Eingang R/$\overline{W}$ des GDP 9366 ist nicht mehr mit dem $\overline{WR}$-Ausgang des Busses verbunden, sondern gelangt über einen Inverter zum $\overline{RD}$-Ausgang. Dadurch liegt das Signal früher an dem Baustein an. Den Inverter kann man vom IC 9 beziehen, z. B. Pin 13 als Eingang des Inverters und Pin 12 als Ausgang nehmen. Auf der Lötseite muß man dazu die Verbindung zum Grafikprozessor trennen und den Inverter dazwischenschalten. Diese Änderung sollten aber nur erfahrene Nachbauer durchführen.

Wie gesagt – diese Veränderung des Layouts der Leiterplatte GDP 64K gilt im wesentlichen der besseren Anpassung an den 68008-Baustein.

In diesem Zusammenhang sei darauf aufmerksam gemacht, daß der „große" NDR-Klein-Computer mit der CPU 68008 im Erwachsenenbildungsprogramm der Nordkette der III. Kanäle der ARD bereits in der letzten Aprilwoche eingeführt wird. mc wird die 68008-Konzeption des NDR-Klein-Computers in ihrem Maiheft ausführlich darstellen und auch in den folgenden Ausgaben dieser brandneuen Konzeption breiten Raum widmen.

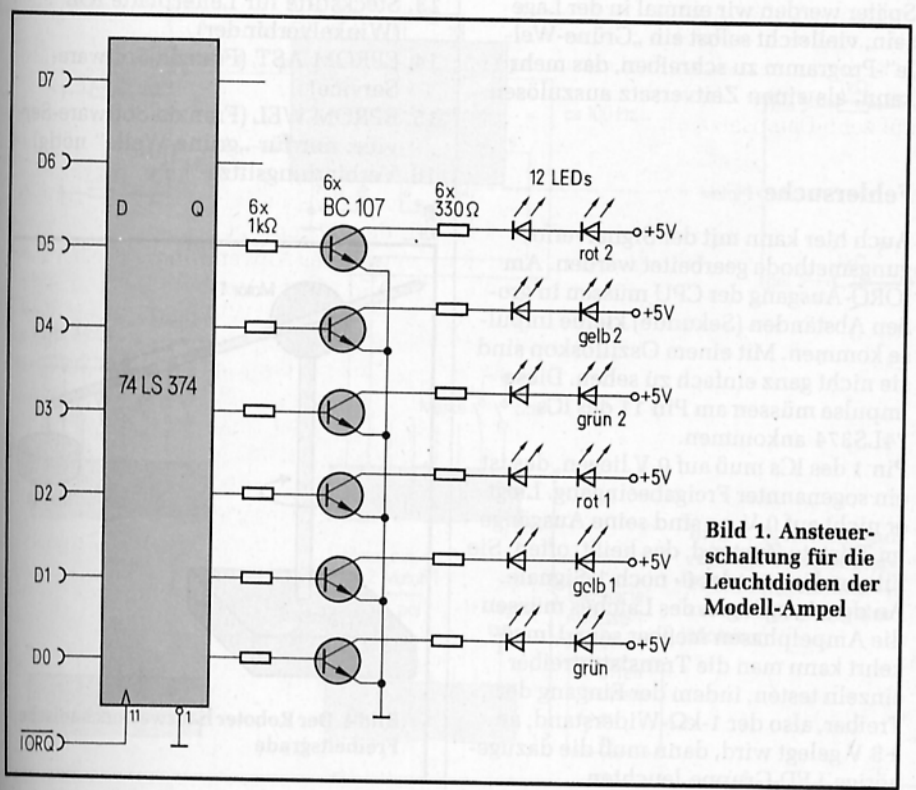

Bild 1. Ansteuerschaltung für die Leuchtdioden der Modell-Ampel

Alarmstufe Rot

In dieser Folge soll eine etwas umfangreichere Interface-Schaltung aufgebaut werden: eine Ampelanlage. Bisher wurde nur die $\overline{IORQ}$-Leitung als Ausgabeleitung eingesetzt. Sie ist aber eigentlich nicht dafür gedacht, direkt als Ausgang verwendet zu werden. Der $\overline{IORQ}$-Ausgang gibt nur an, wann am Datenbus Signale anliegen, die für die Außenwelt verwendet werden können. Da am Datenbus auch noch andere Signale anliegen, müssen diese Ausgabewerte festgehalten werden. Dies geschieht mit Hilfe eines Zwischenspeichers, auch Latch genannt. Er hat die Eigenschaft, sich den Signalzustand an seinen Eingängen merken zu können. Dazu besitzt er einen Eingang, der ihm sagt, wann dies der Fall sein soll. Wir verwenden für ihn das IC 74LS374, ein Latch mit acht einzelnen Zwischenspeicherelementen. Der Takteingang wird mit dem $\overline{IORQ}$-Ausgang der CPU verbunden. Bei einer positiven Flanke, das heißt, wenn der Pegel am Takteingang von 0 auf 1 springt, werden die Daten in den Speicher übernommen. Genau zu diesem Zeitpunkt, also wenn der Impuls vom $\overline{IORQ}$-Ausgang wieder auf 1 zurückgeht, sind aber auch die Daten auf dem Datenbus gültig und können damit in das Latch übernommen werden.

Bild 1 zeigt den Schaltplan der Ampelsteuerung. Die Schaltung wird auf einer IOE-Leiterplatte aufgebaut. Das Latch besitzt acht Daten-Ein- bzw. -Ausgänge. Der Datenbus ist von D0 bis D7 gekennzeichnet. Welches Zwischenspeicherelement welchem Datenbit zugeordnet wird, ist unwesentlich. Die Zuordnung zu den Ampelfarben ist jedoch fest vor-

gegeben: So ist D0 das Bit für die grüne LED der Ampel 1.
Die restliche Schaltung wird auf dem Lochrasterfeld aufgebaut. Die LEDs der Ampeln können über lange Leitungen mit der Karte verbunden werden.
Bild 2 zeigt die Anordnung der Ampeln auf der Kreuzung. Es gibt die beiden Ampelgruppen 1 und 2. Die IOE-Baugruppe wird dann über die Bus-Karte mit der SBC2-Karte und der POW5V-Karte verbunden.
Achtung: Beim Einstecken der Karten unbedingt darauf achten, daß die Versorgungsspannungsausgänge der POW5V-Karte auch an die Versorgungsspannungs-Eingänge der Bus-Karte gelangen und die anderen Karten entsprechend richtig angeschlossen werden!
Masse und 5 V sind paarweise verbunden; daran erkennt man sie auf der Bus-Karte und auf den Leiterplatten. Wer sichergehen will, kann sich Führungsschienen am Bus anbringen, die ein Verdrehen verhindern. Das EPROM AST wird in die Fassung 0 der SBC2-Karte eingesetzt und der Rechner gestartet; dann müssen die Ampelsequenzen sichtbar sein.
Wenn mehrere solcher Anlagen aufgebaut wurden, kann man sie auch miteinander koppeln und eine „grüne Welle" nachbilden. Dazu wird eine SBC2-Baugruppe mit dem EPROM WEL ausgerüstet (Fassung 0). Die Ausgänge 0...6 der IOE-Baugruppe, also die Ausgänge direkt am Kollektor des Transistors BC 107, werden mit den Reset-Eingängen der anderen SBC2-Baugruppen verbunden. Es wird ebenfalls eine Masseverbindung zu jeder Anlage hergestellt. Die Kollektoren werden dabei zu einem der Tasteneingänge der Reset-Taste parallel geschaltet.

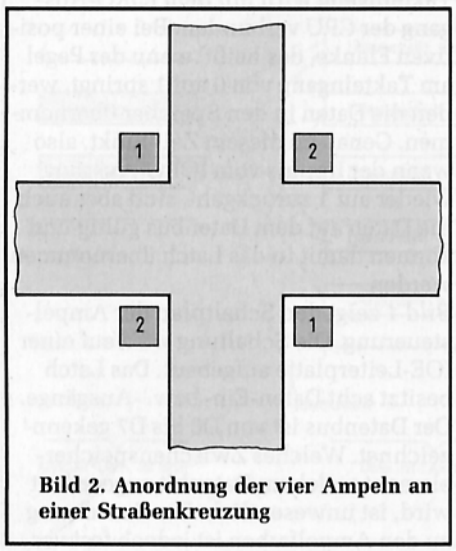

Bild 2. Anordnung der vier Ampeln an einer Straßenkreuzung

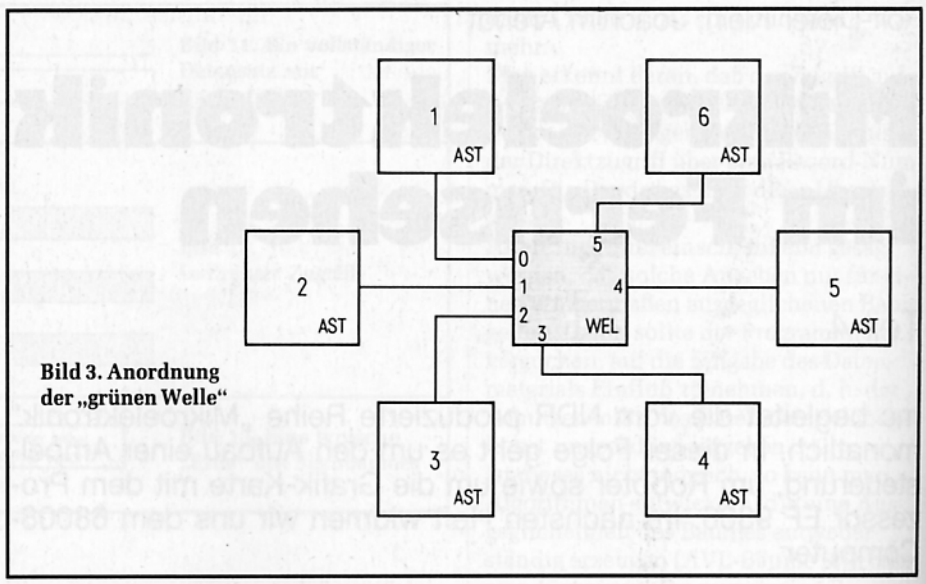

Bild 3. Anordnung der „grünen Welle"

Nach dem Einschalten der Versorgungsspannung der SBC2-Karte mit dem WEL-EPROM werden die anderen SBC-Karten zeitversetzt rückgesetzt und dadurch ergibt sich eine zeitversetzte grüne Welle. Da die SBC2-Karten mit einem Quarz arbeiten, ergibt sich danach auch keine merkliche Zeitverschiebung, denn der Zeitversatz wird quarzgenau eingehalten. Das Verdrahtungsschema zeigt *Bild 3*.

Man kann auch eine beliebige grüne Welle durch Rücksetzen per Hand durchführen.
Später werden wir einmal in der Lage sein, vielleicht selbst ein „Grüne-Welle"-Programm zu schreiben, das mehr kann, als einen Zeitversatz auszulösen.

Fehlersuche

Auch hier kann mit der Signalverfolgungsmethode gearbeitet werden. Am IORQ-Ausgang der CPU müssen in großen Abständen (Sekunde) kleine Impulse kommen. Mit einem Oszilloskop sind sie nicht ganz einfach zu sehen. Diese Impulse müssen am Pin 11 des ICs 74LS374 ankommen.
Pin 1 des ICs muß auf 0 V liegen, das ist ein sogenannter Freigabeeingang. Liegt er nicht auf 0 V, so sind seine Ausgänge im Tristate-Zustand, das heißt, offen: Sie führen dann weder 0- noch 1-Signale. An den Ausgängen des Latches müssen die Ampelphasen meßbar sein. Umgekehrt kann man die Transistortreiber einzeln testen, indem der Eingang der Treiber, also der 1-kΩ-Widerstand, an +5 V gelegt wird, dann muß die dazugehörige LED-Gruppe leuchten.

Materialliste zur Ampelsteuerung

1. POW5V-Bausatz
2. SBC2-Bausatz
3. Bus-Bausatz
4. Leiterplatte IOE (Leiterplatte genügt hierbei)
5. 1× 74LS374
6. 1× 20polige IC-Fassung
7. 6× Widerstand 1 kΩ, ⅛ W
8. 6× Widerstand 330 Ω, ¼ W
9. 6× Transistor BC107
10. 4× rote LEDs, 5 mm
11. 4× grüne LEDs, 5 mm
12. 4× gelbe LEDs, 5 mm
13. Steckstifte für Leiterplatte IOE (Winkelverbinder)
14. EPROM AST (Franzis-Software-Service)
15. EPROM WEL (Franzis-Software-Service; nur für „grüne Welle" nötig)
16. Verbindungslitze

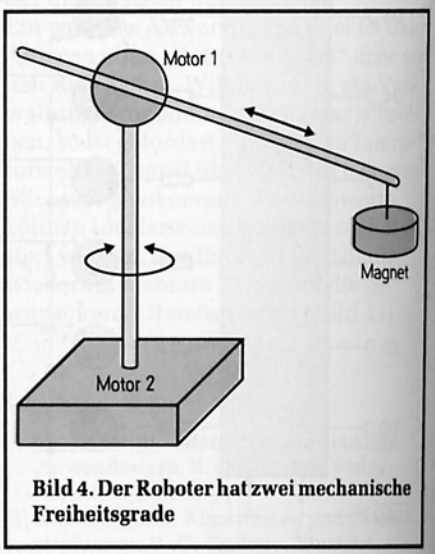

Bild 4. Der Roboter hat zwei mechanische Freiheitsgrade

Roboter steuern

In dieser Folge wird ein kleines Roboter-Modell angesteuert, das z. B. aus Fischer-Technik gebaut werden kann.

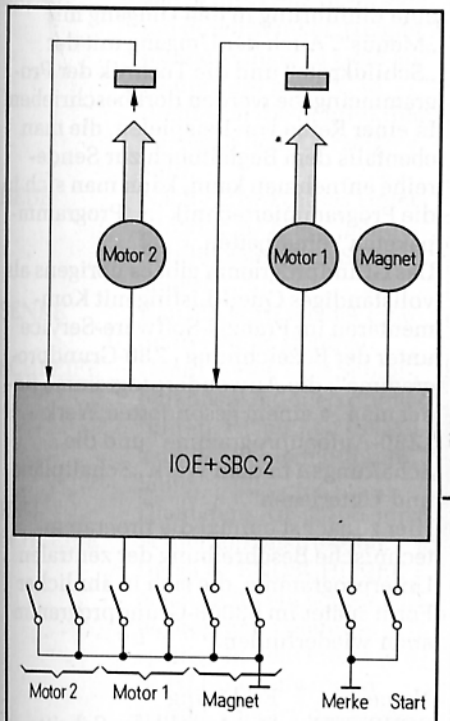

Bild 5. Verbindung der Roboter-Motoren und des Magneten mit dem Rechner

Der Roboter hat die Aufgabe, zwei Münzen zu vertauschen. *Bild 4* zeigt den schematischen Aufbau des Roboters. Er besitzt zwei Freiheitsgrade. Ein Arm kann sich um eine Grundachse drehen, und der Arm kann herauf- und heruntergeführt werden. Mit einem Magneten ist es möglich, Fünfer- oder Zehnerstücke aufzunehmen. Die Bewegung wird mit zwei Motoren gesteuert. Damit der Rechner weiß, wie gerade seine Position ist, sind an den beiden Achsen jeweils Potentiometer angebracht, die über die Schleiferstellung eine Rückmeldung an den Computer ermöglichen. Da die Potentiometer nur ein analoges Signal liefern, muß es noch zuvor in ein digitales Signal gewandelt werden. Dies geschieht mit einem Spannungs-/Frequenz-Umsetzer. Hier wird der Baustein 74LS627 verwendet, der besonders einfach anzusteuern und außerdem preiswert ist. Er enthält außerdem zwei Umsetzer, die wir brauchen, um zwei Achsen zu kontrollieren.

Die Schaltung ist nun schon erheblich umfangreicher als in bisherigen Aufbauten. Daher muß man beim Nachbau schrittweise vorgehen und nach jedem Schritt testen.

Der Aufbau erfolgt auf der IOE-Baugruppe; diese wird dazu voll bestückt. *Bild 5* zeigt einen Überblick über die Verdrahtung. Der Motor 1 wird von der IOE-Karte gesteuert, das Potentiometer liefert die Rückmeldung wieder an die IOE-Karte zurück; entsprechend ist es bei Motor 2.

Bild 6 zeigt den vollständigen Schaltplan. Beim Aufbau beginnt man mit den Spannungs-Frequenz-Umsetzern des Bausteins 74LS627. Er läßt sich auch

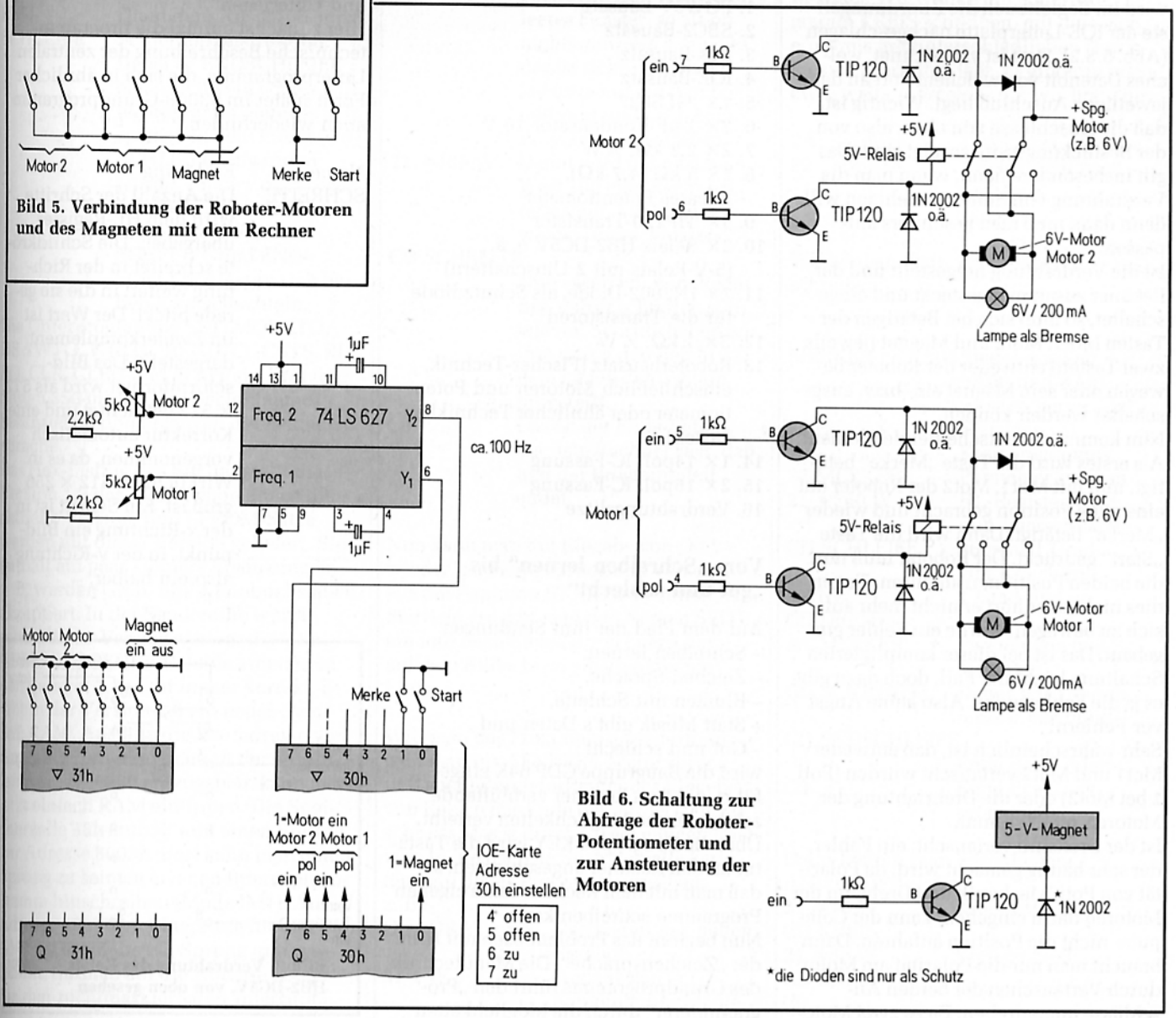

Bild 6. Schaltung zur Abfrage der Roboter-Potentiometer und zur Ansteuerung der Motoren

schon ohne Computer testen: An den Ausgängen Pin 6 und Pin 8 müssen jeweils zwei Frequenzen anliegen, die sich in Abhängigkeit von der Potentiometerstellung ändern lassen.
Die Transistortreiber kann man auch ohne Rechner testen: über den Eingang „ein" muß der Motor gestartet werden können, und wenn ein +5-V-Pegel an „pol" gelegt wird, so muß sich die Drehrichtung ändern. *Bild 7* zeigt die Beschaltung des hier verwendeten Relais, andere Bauformen können jedoch unterschiedlich belegt sein.
Mit dem EPROM ROB wird die Fassung 0 der SBC2-Karte ausgerüstet. Dann kann's losgehen. Waren die Einzeltests erfolgreich und funktionierten der Spannungs-/Frequenz-Umsetzer und die Treiberschaltung, so können die Aus- und Eingänge mit den Anschlüssen auf der IOE-Karte verbunden werden. Dazu wird im Begleitbuch die Zuordnungsliste der IOE-Leiterplatte nachgeschlagen (Abb. 5.5.2). Dort ist verzeichnet, welches Datenbit von welchem Port an dem jeweiligen Anschluß liegt. Wichtig ist, daß die Anschlüsse von oben, also von der Bestückungsseite gezeigt sind. Das gilt insbesondere dann, wenn man die Verdrahtung von unten vornehmen will, denn dann muß man besonders aufpassen.
Ist die Verdrahtung hergestellt und der Rechner zusammengesteckt und eingeschaltet, so muß sich bei Betätigen der Tasten Mot1, Mot2 und Magnet (jeweils zwei Tasten) entweder der Roboter bewegen oder sein Magnet ein- bzw. ausgeschaltet werden können.
Nun kommt der entscheidende Moment: Als erstes wird die Taste „Merke" betätigt, nun mit Mot1, Mot2 der Roboter auf eine neue Position gebracht und wieder „Merke" betätigt. Dann wird die Taste „Start" gedrückt. Der Roboter muß nun die beiden Positionen anfahren. Tut er dies nicht oder hört er nicht mehr auf, sich zu bewegen, wurde ein Fehler eingebaut. Das ist bei dieser komplizierten Schaltung leicht der Fall, doch dazu gibt es ja die Fehlersuche. Also keine Angst vor Fehlern!
Sehr wahrscheinlich ist, daß entweder Mot1 und Mot2 vertauscht wurden (Poti 1 bei Mot2) oder die Drehrichtung der Motoren nicht stimmt.
Ist der Drehsinn vertauscht, ein Fehler, der sehr häufig gemacht wird, da Polarität von Poti, Mechanik und Drehsinn der Motoren darin eingehen, kann der Computer nicht die Position anfahren. Dann braucht man nur die Polarität am Motor durch Vertauschen der beiden Anschlüsse umzudrehen. Da es zwei Motoren gibt, kann dieser Fehler zweimal vorhanden sein. Ist Poti 1 dem Motor 2 zugeordnet und umgekehrt, gibt es ebenfalls Probleme. Das ist aber durch einen Verdrahtungsvergleich zu finden.
Ob überhaupt eine Reaktion stattfindet, läßt sich feststellen, indem man die Motoren mechanisch löst, so daß sie frei laufen. Nach Betätigen der „Start"-Taste können die Achsen von Hand hin- und herbewegt werden. Die Motoren müssen dann eine Reaktion zeigen; tun sie das nicht, liegt ein Fehler in der Spannungs-Frequenz-Umsetzerschaltung vor: Die Frequenz muß im Bereich von etwa 100 Hz liegen, und die Verbindungen zum 74LS245 sind zu prüfen.

Materialliste für „Roboter steuern"

1. POW5V-Bausatz
2. SBC2-Bausatz
3. Bus-Bausatz
4. IOE-Bausatz
5. 1× 74LS627
6. 2× 1-µF-Kondensator 10 V
7. 2× 2,2 kΩ, ⅛ W
8. 2× 5 kΩ (4,7 kΩ), lineares Potentiometer
9. 5× TIP120-Transistor
10. 2× Relais HB2-DC5V o. ä. (5-V-Relais mit 2 Umschaltern)
11. 7× 1N2002-Diode, als Schutzdiode für die Transistoren
12. 5× 1 kΩ, ⅛ W
13. Roboterbausatz (Fischer-Technik, einschließlich Motoren und Potentiometer oder ähnlicher Technikbaukasten)
14. 1× 14pol. IC-Fassung
15. 2× 16pol. IC-Fassung
16. Verdrahtungslitze

Vom „Schreiben lernen" bis „gut und schlecht"

Auf dem Pfad der fünf Sendungen
– Schreiben lernen,
– Zeichen-Sprache,
– Blumen mit Schleife,
– Statt Musik gibt's Daten und
– Gut und schlecht
wird die Baugruppe GDP 64K eingeführt, die dem Rechner verblüffende zeichnerische Möglichkeiten verleiht. Über die Baugruppe KEY wird die Tastatur an den Rechner angeschlossen, so daß man mit dem Rechner seine eigenen Programme schreiben kann.
Nun beginnt das Problemlösen mit Hilfe der „Zeichensprache". Die Menütechnik des Grundprogramms führt den „Programmierer" durch die Möglichkeiten des NDR-Klein-Computers. Im Begleitbuch des Kurses (Rolf-Dieter Klein: „Mikrocomputer selbstgebaut und programmiert", erschienen in zweiter Auflage im Franzis-Verlag) findet man ab Seite 340 eine Einführung in den Umgang mit „Menüs". Auch der Umgang mit der „Schildkröte" und die Technik der Programmeingabe werden dort beschrieben. In einer Reihe von Beispielen, die man ebenfalls dem Begleitbuch zur Sendereihe entnehmen kann, kann man sich in die Programmiertechnik in „Programmpaketen" einarbeiten.
Das Grundprogramm gibt es übrigens als vollständiges Quell-Listing mit Kommentaren im Franzis-Software-Service unter der Bezeichnung „Z80-Grundprogramme"; die Anwenderprogramme findet man in einem gesonderten Werk „Z80-Aufbauprogramme" und die Schaltungen in dem Werk „Schaltpläne und Unterlagen".
Hier zunächst einmal die programmtechnische Beschreibung der zentralen Unterprogramme, die sich in ähnlicher Form später im 68008-Grundprogramm auch wiederfinden.

Name	*Erklärung*
SCHREITE	Die Anzahl der Schritte wird dem HL-Register übergeben. Die Schildkröte schreitet in der Richtung weiter, in die sie gerade blickt. Der Wert ist im Zweierkomplement dargestellt. Das Bildschirmformat wird als 512 × 512 betrachtet und eine Korrektur automatisch vorgenommen, da es in Wirklichkeit 512 × 256 groß ist. Ein Schritt ist in der x-Richtung ein Bildpunkt, in der y-Richtung also ein halber.

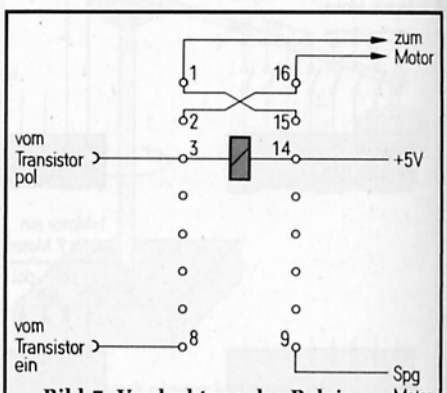

Bild 7. Verdrahtung des Relais HB2-DC5V, von oben gesehen

SCHR16TEL	Hier wird in der x-Richtung 1/16 Bildpunkt geschritten und in der y-Richtung 1/32. HL ist wieder das Übergaberegister.
DREHE	Die Schildkröte dreht sich um einen Winkel. HL gibt diesen Winkel in Grad an. Ist die Zahl positiv, so dreht sich die Schildkröte entgegen dem Uhrzeigersinn.
HEBE	Nach Aufruf dieses Unterprogramms hinterläßt die Schildkröte keine Schreibspur.
SENKE	Danach wird wieder eine Schreibspur sichtbar.
SCHLEIFE	Damit lassen sich Schleifenkonstruktionen durchführen. HL enthält die Anzahl der Schleifendurchläufe.
END-SCHLEIFE	Dient dem Anschluß einer Schleifenkonstruktion. Dabei ist ENDSCHLEIFE eigentlich kein echtes Unterprogramm, denn es springt zu dem Punkt, wo SCHLEIFE aufgerufen wurde, solange der vorgegebene Schleifenzähler nicht abgelaufen ist. SCHLEIFE und ENDSCHLEIFE lassen sich beliebig verschachteln.

Die Unterprogramme SET, MOVETO, DRAWTO, WRITE, READ, CI, CSTS, RI, PO, CLR, CLRG und WAIT sind auf den Seiten 350 bis 351 des Begleitbuches beschrieben und ergänzen die Zeichensprache.
Das Hex-Listing des Buches stellt die Version 1.2 dar. Die Version 2.0 ist die aktuelle Version, und EPROM-Sätze werden mit dieser Version geliefert. Sie enthält ein paar kleinere Ergänzungen, z. B. werden Groß- und Kleinbuchstaben akzeptiert. In der Sendereihe wurde schon diese Version verwendet. Die HEBE- und SENKE-Befehle arbeiten in der Version 1.2 nicht immer korrekt. Es gibt in der Version 2.0 ein neues Symbol: RAM. Anstelle die Startadresse von Anwenderprogrammen mit der absoluten Adresse 8800 einzugeben, kann man jetzt einfach RAM eintippen. Die Speicherzelle 38h enthält nun einen Sprung zur Adresse 8003h, dort kann man einen Sprung zu seinem eigenen Interruptprogramm hinschreiben (Mode IM1). Auf Adresse 66h, dem Einsprung für den NMI, ist ebenfalls ein Sprung untergebracht, diesmal auf die Adresse 8000h für den nichtmaskierbaren Interrupt.

Programmbeispiele

Bei der Zeichensprache programmiert man zunächst einmal in Maschinensprache, besitzt aber die Möglichkeiten, mit Symbolen zu arbeiten und hat daher die Gelegenheit, seine Unterprogramme und Sprungziele zu benennen. Ein einfaches Beispiel:
Man wählt das Menü AENDERN, dann gibt man als Startadresse RAM ein. Nun kann man das Programm eingeben.
Mit der Sequenz

SEITE := $	wird das Symbol SEITE definiert. Dem Namen Seite wird durch das Zeichen „$" die aktuelle Adresse zugeordnet. Man könnte auch schreiben SEITE := 8800. Das Bildfenster zeigt nach der Eingabe, die durch cr (Carriage Return) beendet wird, wieder ein leeres Fenster, so als ob nichts gewesen wäre und das ist richtig, denn wir haben ja nur ein Symbol definiert und noch kein Programm eingegeben.
21 #100.W	Damit wird der dezimale Wert 100 dem Rechner mitgeteilt. Mit cr wird wieder abgeschlossen.
CD SCHREITE	Nun sagen wir dem Rechner, daß er schreiten soll, wieviel Schritte haben wir ihm durch den Befehl 21, dem Ladebefehl, schon mitgeteilt.
C9	Das ist der Ende-Befehl, der dem Rechner sagt, daß hier der Programmteil endet.

Nun kann man mit Eingabe von „M" wieder ins Menü zurück und von dort aus die Funktion STARTE aufrufen. Als Startadresse gibt man den Namen SEITE ein und nach Eingabe von cr erscheint auf dem Bildschirm eine senkrechte Linie.

Will man das Programm erweitern, so geht man ins AENDERE-Menü zurück und sucht durch fortlaufendes Eingeben von Return das Ende des vorherigen Abschnitts, das ja durch C9 gekennzeichnet war. In die nächste Speicherzelle kann man ein neues Programm eingeben, z. B.:

QUADRAT := $	Definieren des neuen Symbols
21 4.W	Anzahl der Schleifendurchläufe
CD SCHLEIFE	Schleifenstart
CD SEITE	Unterprogramm Seite aufrufen
21 #90.w	den dezimalen Wert 90 laden
CD DREHE	Schildkröte um 90 Grad drehen
CD ENDSCHLEIFE	Ende der Schleife
C9	Ende des Programms

Man kann sich Programmaufgaben dieser Art aus Büchern über die Programmiersprache Logo heraussuchen und sie mit der Zeichensprache lösen.

Ein wichtiger Punkt ist noch die Übergabe von Variablen an Unterprogramme. Dazu benötigt man Speicherzellen. Auch hierfür kann man Symbole definieren.

Beispiel: Wir wollen ein Unterprogramm KREIS schreiben, mit dem sich ein Kreis mit unterschiedlichen Durchmessern gestalten läßt. Es gibt nun noch zwei weitere Befehle, die wir brauchen:

2A adresse	LADE von „adresse"
22 adresse	SPEICHERE nach „adresse"

Wir können dann das Programm so schreiben:

MERKER := 8A00	Speicherzellen 8A00 und 8A01 werden verwendet
KREIS :=$	Start des Kreisprogramms
22 MERKER	Parameter merken
21 #360.W	HL-Register wird nun überschrieben
CD SCHLEIFE	360mal 1 Grad gibt Kreisform
21 1.W	1 Grad drehen
CD DREHE	
2A MERKER	Parameter n mal 1/16 schreiten.
CD SCHR16TEL	Damit wird Durchmesser bestimmt
CD ENDSCHLEIFE	Ende der Schleife
C9	Ende des Kreisprogramms

Dann könnte man anschließen:

ANWENDER :=$	
21 #16.W	Kreisgröße 1
CD KREIS	
21 #50.W	Kreisgröße 2
CD KREIS	
C9	

Will man die Parameter verändern, so kann man nach und nach weitere Befehle des Z80 hinzuziehen.

Man kann nun Symbole nicht nur für Adressen definieren, sondern auch für Befehle:

```
RUFE := CD
LADE := 21
ENDE := C9
```

Man kann nun schreiben:

```
DREIECK :=$
LADE #3.W
RUFE SCHLEIFE
LADE #50.W
RUFE SCHREITE
LADE #120.W
RUFE DREHE
RUFE ENDSCHLEIFE
ENDE
```

Und der Rechner übersetzt es in Maschinensprache. Damit hat man einen ersten Übergang zu höheren Programmiersprachen geschaffen. Beispiel für die Anwendung einer Textausgabe auf dem Bildschirm:

```
TEXT :=$
20.W 30.W 44.B 0.B  Px=20,
                    y=30,
                    Höhe=4,
                    Breite=4
„HALLO GEHT“        beliebiger Text
0.B                 Endekennung

AUSGABE :=$
21 TEXT.W           Adresse laden
CD WRITE            Ausgabeprogramm
C9
```

Nach Start des Programms AUSGABE muß auf dem Bildschirm der Text „HALLO GEHT“ erscheinen.

Hier nochmals der Hinweis auf das „Arbeitsmaterial zur Serie Mikroelektronik“, die beim Franzis-Software-Service für etwa 12 DM je Heft erhältlich ist, in der alle Schaltpläne (z. B. die Waagenschaltung aus der Folge „Gut und schlecht“) sowie alle Programme erschienen sind.

Dort bekommt man auch alle Programme auf EPROMs. Übrigens werden alle Anwenderschaltungen wie ROBOTER und AMPEL von der Firma GES und Elektronikladen auch auf geätzten Leiterplatten angeboten, so daß der Verdrahtungsaufwand gespart werden kann.

Fortsetzung folgt

GETSTRING in Basic

Der Beitrag „VARPTR umgekehrt“ in mc 10/1983, S. 53, stellte eine GETSTRING-Funktion in Maschinensprache vor, die es gestattet, einen String von einer beliebigen Speicheradresse einzulesen. Das läßt sich auch in Basic realisieren, sofern der VARPTR-Befehl zur Verfügung steht. *Bild 1* verwendet die Variablen ST$ als einzulesenden String, LN als Länge der Zeichenkette und AD als ihre Anfangsadresse. *Bild 2* zeigt, wie beim TRS-80-Disk-Basic damit eine Bildschirmkopie auf dem Drucker ausgegeben werden kann. *Bild 3* liest einen String von einer Bildschirm-Maske ein, und *Bild 4* ermöglicht eine Stringspace-Zuordnung ohne zusätzlichen String-Speicherbedarf. Zum Vergleich enthält *Bild 5* die herkömmliche Lösung des Problems aus Bild 4.

Gerd Kluge

```
200 ST$=""
210 REM Laenge der Zeichenkette in den Eintrag schreiben
220 POKE VARPTR(ST$), LN
230 REM Adresse der Zeichenkette in den Eintrag schreiben
240 POKE VARPTR(ST$)+1, AD -256*INT(AD/256)
250 POKE VARPTR(ST$)+2, INT(AD/256)
```

Bild 1. Einlesen eines Strings ab einer bestimmten Anfangsadresse vom TRS-80-Basic aus

```
330 FOR I=15360 TO 16383 STEP 64
340 LP$="":POKE VARPTR(LP$), 64:
    POKE VARPTR(LP$)+1, I- 256*INT(I/256):
    POKE VARPTR(LP$)+2, INT(I/256)
350 LPRINT LP$
360 NEXT I
370 END
```

Bild 2. Ausgabe einer Bildschirm-Kopie auf dem Drucker

```
440 PRINT §IP,STRING$(IN,".");
450 FOR I=0 TO IN-1
460 PRINT §IP+I,CHR$(143)".";
470 IN$=INKEY$: IF LEN(IN$)=0 THEN 470
480 AS=ASC(IN$)
490 IF AS>31 AND AS<96 THEN 530
500 IF AS>63 AND AS<128 THEN 530
510 IF AS=8 AND I>0 THEN I=I-1
520 GOTO 460
530 PRINT §IP+I,IN$;
540 NEXT I
550 BA$="":POKE VARPTR(BA$), I:
    POKE VARPTR(BA$)+1, 15360+IP -256*INT((15360+IP)/256):
    POKE VARPTR(BA$)+2, INT((15360+IP)/256)
560 BA$=BA$
570 END
```

Bild 3. Einlesen aus einer Bildschirm-Maske; IP gibt die Einleseposition an (0...1023), IN die Anzahl der einzugebenden Zeichen. Das Ergebnis steht in BA$

```
610 CLEAR 355
620 A$=STRING$(255,"a"): V=VARPTR(A$)
630 B$="":POKE VARPTR(B$),100:
    POKE VARPTR(B$)+1, PEEK(V+1):
    POKE VARPTR(B$)+2, PEEK(V+2)
640 PRINT FRE(X$)
```

Bild 4. String-Zuweisung ohne unnötige Speicherplatz-Verschwendung

```
680 CLEAR 355
690 A$=STRING$(255,"a")
700 B$=LEFT$(A$,100)
710 PRINT FRE(X$)
```

Bild 5. Gleiches Problem wie Bild 4 – aber herkömmliche Lösung