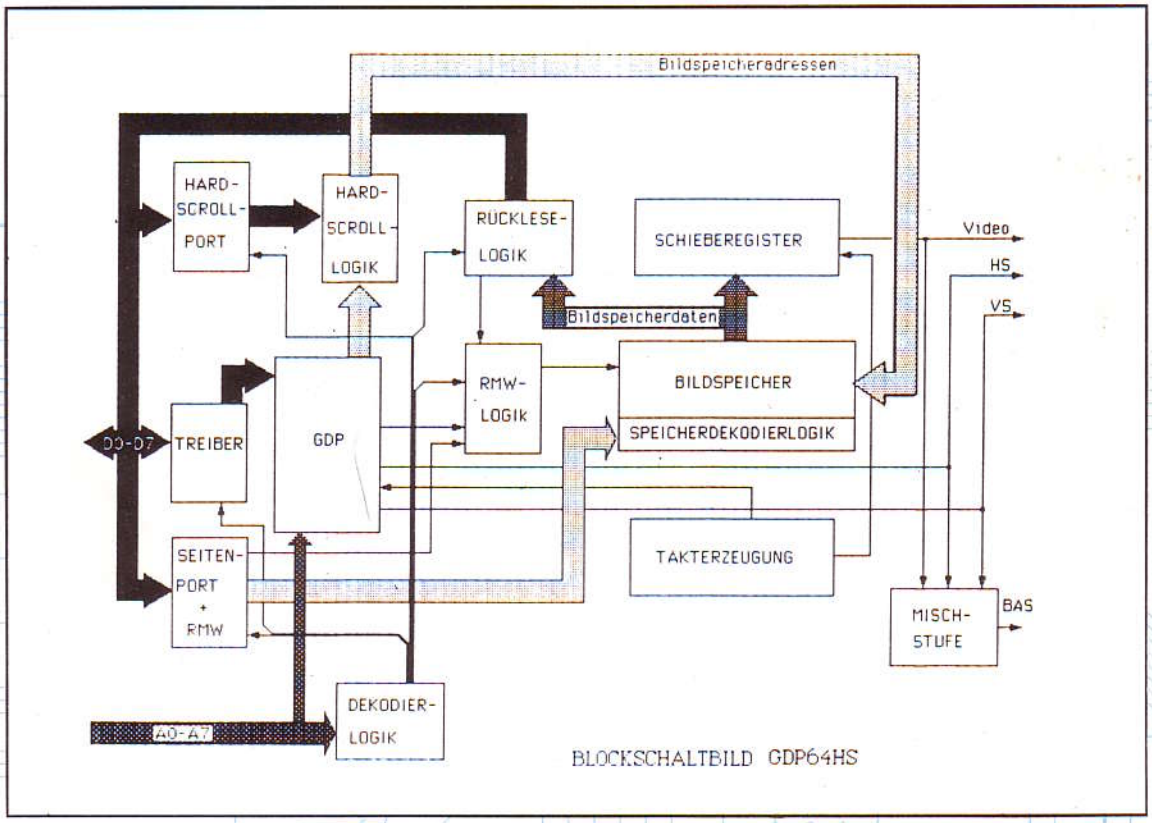
LOOP

Juli 1988

18

3. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter DM 3,–



BLOCKSCHALTBILD GDP64HS

Hardware:

GDP64HS - Schnittstelle zum Monitor (S. 18/4)

Test:

LOG 16 gegen LOGAN (S. 18/21)

CPU 680xx Grundprogramm:

Wie schreibt man relokative Programme ? (S. 18/23)

Leitartikel

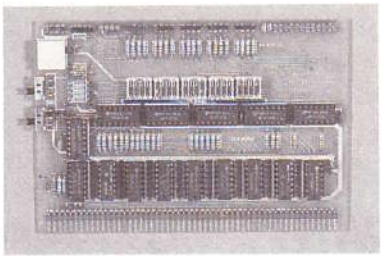
CPU Z80

CPU 680XX

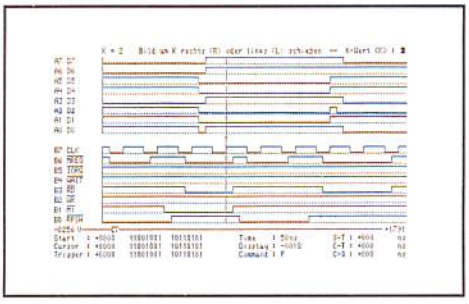
mc - modular - AT

LOOP 18 - Titelbild von GDP64HS

Neu: Tower-Gehäuse für den mc-modular-AT

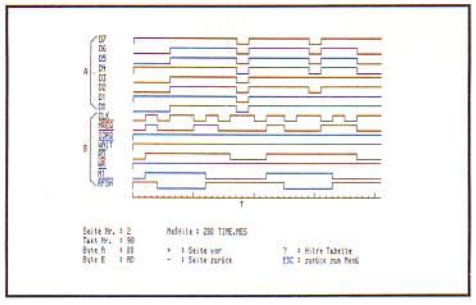


Neu: Bustest-Baugruppe



Der Vergleich:
LOG16 gegen LOGAN

LOGAN **LOG16**



Rubriken

Impressum:

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf Dieter Klein,Gerd Graf, Christoph Köhler

Gestaltung: Christoph Köhler

Druck: Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung:
GES GmbH
Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste 4/88

Liebe Leser,

Die Zeitschrift LOOP

kennen Sie bereits vermutlich seit 17 Ausgaben. Mit der hier nun vorliegenden 18. Ausgabe haben Sie es sicher schon bemerkt: Wir versuchen, aus der "LOOP" eine "richtige" Zeitung zu machen.

Das neue Layout

Dazu gehört auch etwas, das man gemeinhin mit "Layout" bezeichnet, und das sich, vergleichen Sie einmal die Ausgabe 0 bis 17, doch im Lauf der Zeit entwickelt hat.

Natürlich sind wir alle keine geborenen Journalisten, sondern mussten auch im Laufe der Zeit dazulernen. Uns stand - und steht - immer die Information unserer Kunden im Vordergrund, und nicht die Schönheit.

Viele Briefe aus Ihren Reihen haben uns aber auch erkennen lassen, daß sich eine Zeitschrift mit einem "richtigen" Layout einfach besser und leichter liest als eine Aneinanderreihung von Texten.
Natürlich soll durch die so gewonnene Schönheit nicht der Inhalt leiden - dafür stehen wir gerade.

LOOP und der mc-modular-AT

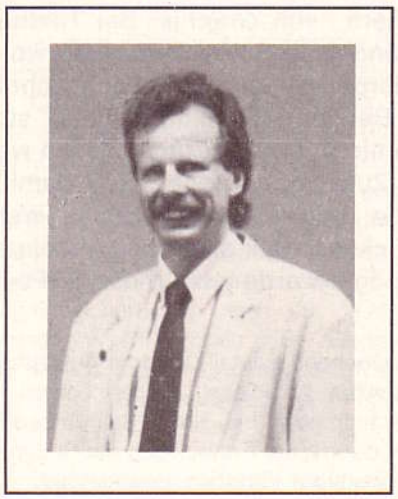
Die letzte Nummer hat's gezeigt: Auch der mc-modular-AT wird nun seinen festen Platz in der LOOP finden. Einige NDR-Computer-Anwender haben sich über diese Abwanderung der LOOP bitter beklagt: Hier soll all denen die Angst genommen werden, daß aus LOOP bald eine PC-, XT-, AT oder sonstige "Kompatible" Zeitung wird. Wir werden auch in LOOP nicht jeden AT oder PC testen. Wir wollen erreichen:

Daß unsere NDR-Kunden, die im Beruf oder auch Zuhause mit IBM zu tun haben, nicht allein gelassen werden

Daß unsere mc-modular-AT Kunden die gleiche Unterstützung auch nach dem Kauf erfahren wie unsere NDR-Kunden.

Daß unsere NDR-Kunden auch etwas über IBM erfahren, speziellauch was die Kopplung der beiden Systeme angeht.

Wir wollen nicht auf "Kosten" des NDR-Computers arbeiten - der NDR-Computer wird auch in Zukunft in der LOOP breit unterstützt werden. Bedenken Sie bitte, daß der Umfang der LOOP von ehemals 16 Seiten - davon 4 Werbung - auf generell 32 Seiten (diesmal 36) - bei gleichem Preis - gestiegen ist. Allerdings werden wir wohl bald gezwungen sein, den Preis der LOOP etwas anzuheben.



OS/9 oder nicht OS/9

Selten hat ein Hinweis in der LOOP so die Gemüter erregt: Der Verzicht auf OS/9 für den NDR-Computer. Man muß erst einmal mitteilen, daß man etwas NICHT macht, um dann festzustellen, wie viele unserer Kunden nun DOCH gerne OS/9 hätten!
Einer unser Software-Partner hat sogar im Rahmen einer "Selbsthilfe-Aktion" bereits während der Hannover Messe begonnen, potentielle Interessenten zu sammeln, um die 50 Lizenzen abnehmen zu können!

Nun ging es uns während der Messe fast wie dem Sterntaler-Mädchen: Ein NDR-Anwender kam zu uns auf den Stand, erzählte uns über OS/9, daß er auf eigene Kosten (!) sich einfach 50 Lizenzen gekauft und diese bereits an den NDR-Computer angepasst hätte! Wir waren schlicht stumm.

Mittlerweile hat auch schon die erste Vorführung stattgefunden; einige sehr kleine Änderungen an der Hardware müssen noch durchgeführt werden, und bereits im Spätsommer wollen wir OS/9 zunächst unseren Software-Partnern, dann unseren Kunden anbieten.

LOOP wird Sie informieren!

Dies jedoch nur als unverbindliche Vorab-Information. Wir haben noch einiges mit dem NDR-Computer, speziell auf der 68000-Seite vor. Dies als Hinweis an die NDR-User, die glauben, es "kommt nichts mehr".

Die neue GDP ist da.

Dank der Tätigkeit des Herrn Bulwien, der die Hardcopy-Aufsatzplatine entwickelte, haben wir nun daraus und aus einigen Verbesserungsvorschlägen die neue GD64HS entwickelt. Die alte GP64 Streitfrage: Geht nun der Hardware-Scroll und das Rücklesen oder nicht?" ist nunmehr entschieden: Es geht.

Grundsätzlich geht es wahrscheinlich: Es gab tatsächlich eine Maskenversion des GDP9366, bei dem das Timing so verschoben war, daß auch das Rücklesen bei der neuen GDP nicht funktioniert.
Alle neuen Baugruppen werden mit getesteten GDP´s geliefert - sollten Sie beim Umbauen Probleme haben, so KANN es am GDP liegen.

Für den Umbau haben wir uns einen Umbausatz einfallen lassen: Er enthält die Leiterplatte sowie alle auf der GDP eingelöteten Bauelemente. Wir haben den Preis sehr knapp kalkuliert, um möglichst allen Anwendern einen Umstieg zu erleichtern.

Ein Missverständnis sei vorab behoben: Die GDP64HS schafft nur die hardwaremässigen Vorraussetzungen für Hardscroll und Rücklesen: Natürlich müssen Programme dies auch unterstützen!
Ebenso natürlich arbeiten bereits viele Software-Partner daran, ihre Programme anzupassen.

Ihre Meinung ist uns wichtig

Schreiben Sie uns Ihre Meinung über die "neue" LOOP. Wir sind immer an allen Verbesserungsvorschlägen interessiert

Wir warten auf Ihre Beiträge

Die LOOP steht und fällt mit IHREN Beiträgen. Auch wenn Sie glauben, das sei doch gar nicht so toll: Schicken Sie uns Ihre Programme, Ihre Tips + Tricks und Hinweise! Wir danken Ihnen im Voraus.

Gerd Graf
und die LOOP Redaktion

Die neue GDP64 mit Hardscroll, Rücklesen und RMW

GDP64HS

Die GDP64k gehört seit Beginn der NDR-Fernsehserie zum NDR-Computer. Sie war die erste Standardgraphikkarte für diesen Rechner und war sowohl für Textdarstellung als auch für Graphikdarstellung geeignet. Allerdings hatte sie zwei Nachteile, sowohl in der Textdarstellung als auch beim Abspeichern von Graphik. Bei Textdarstellung war vor allem der langsame Scroll bemängelt worden, der jetzt durch eine Hardscrollogik erheblich schneller gemacht werden kann. Bei der Graphikdarstellung störte meistens, daß Graphiken nicht direkt zurückgelesen werden konnten (nur mit einer Zusatzbaugruppe) und damit auch nicht direkt auf den Drucker ausgegeben werden konnten. Auch diese Rückleselogik gekoppelt mit einer RMW (Read-Modify Write) Logik wurde jetzt in der GDP64HS mit aufgenommen.

Das Rücklesen wurde vorher von der Baugruppe HCOPY/MAUS erledigt, die dann entfällt. Der Anschluß der Maus wird dann auf der neuen KEY-Baugruppe (KEY3) enthalten sein.

Da die GDP64k weit verbreitet ist, haben wir uns entschlossen, außer dem Fertiggerät, dem Bausatz und der Leiterplatte noch einen Aufrüstsatz anzubieten. Dieser Aufrüstsatz soll es den bisherigen GDP64k Benutzern ermöglichen, die IC's ihrer alten GDP weiter zu benutzen, und nur die Leiterplatte, die passiven Bauelemente und natürlich die zusätzlichen Bauteile neu zu beziehen. Diese neu zu beziehenden Bauteile sind im Aufrüstsatz enthalten. An dieser Stelle möchten wir uns auch noch bei Herrn Bullwin bedanken, der bei der Entwicklung der Baugruppe wertvolle Vorarbeit geleistet hat.

In den folgenden Abschnitten wollen wir etwas näher auf die GDP64HS eingehen und anhand des Blockschaltbildes grob die Funktion der Baugruppe erläutern. Detailiertere Information erhalten Sie natürlich wie immer aus unseren Handbüchern, wobei das neue GDP64HS-Handbuch mit rund 100 Seiten besonders ausführlich geworden ist.

Das Herz der GDP64HS ist der Graphikprozessor 9366 von Thomson. Auf diesen Prozessor hier näher einzugehen wäre sinnlos und würde den Rahmen dieses Artikels sprengen. Deshalb wollen wir hier nur kurz einige Characteristica des Prozessors erläutern.

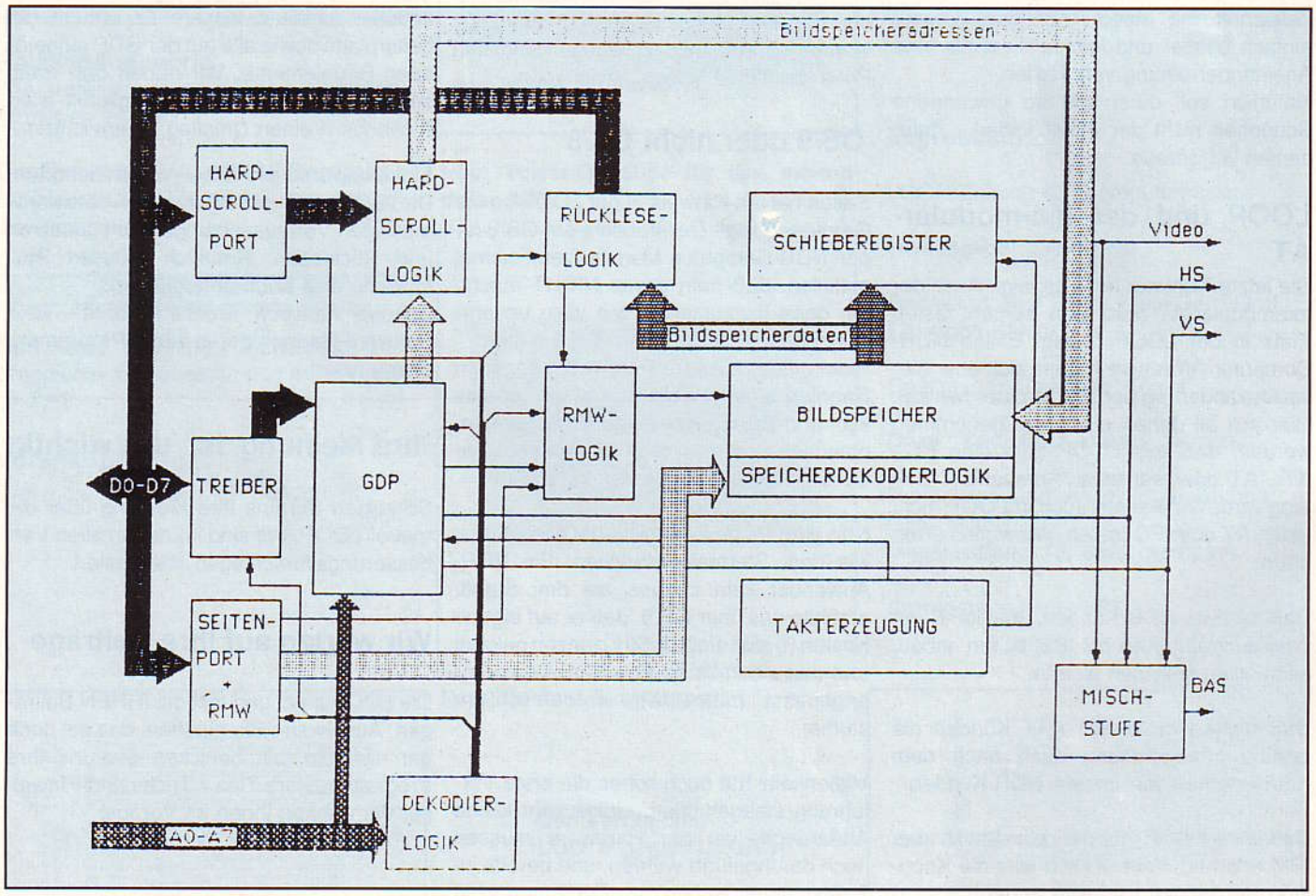


Bild 1: Blockschaltbild der GDP64HS

Der 9366 hat den kompletten ASCII-Satz integriert; diese Zeichen können mit Hilfe des sogenannten "CSIZE" Registers noch in verschiedenen Größen dargestellt werden. Außerdem besteht die Möglichkeit, Zeichen kursiv und senkrecht darzustellen. Anderseits besitzt der 9366 auch einige gute Graphikmöglichkeiten. Hier ist vor allem die Vektorgraphik zu erwähnen. Um extrem schnelle Graphiken zu erzeugen, kann mit sogenannten Kurzvektoren gearbeitet werden, die mit Hilfe nur eines Befehlsbytes an den Graphikprozessor übergeben werden. Es besteht auch die Möglichkeit mit Blockgraphik 8 x 5 oder 4 x 4 zu arbeiten.
Der 9366 verwaltet seinen Bildschirmspeicher selbst. Er kann direkt 16KByte adressieren, was bei einer Auflösung von 512 x 256 Bildpunkten einer Bildschirmseite entspricht. Will man mehr als eine Bildschirmseite verwalten, muß wie hier eine spezielle Logik bereitgestellt werden.

Von der Haupt- CPU (z.B. von der CPU68008) wird über die Adressleitungen A0...A3 eines der 16 Register des GDP (EF 9366) angewählt. Soll z.B. ein Vektor gezeichnet werden, so teilt man dem Graphik-Prozessor lediglich den Anfangs- und Endpunkt mit. Die Zwischenwerte werden von ihm selbst berechnet und dann in den Speicher abgelegt.

Der interne Aufbau des Speichers wird durch den Grafikprozessor und sekundär durch die Speicherdekodierlogik organisiert. Im Speicher steht dann die Information, die später auf dem Bildschirm erscheint. Um den Ablauf des "Displays" aus dem Speicher etwas zu verdeutlichen, haben wir ein kleines Beispiel angeführt.

Beispiel: Wir verfolgen das Auslesen eines Bytes vom Speicher zum Monitor (z.B. 10001110)

Das Byte steht am Ausgang des Speichers und wird beim Aktivieren des Signals SH/L (Shift Load, am Schieberegister) parallel in das Schieberegister eingelesen. Hier wird das Signal mit dem Punktetakt (14 MHz, CLK) verknüpft und seriell (in der Punktfolge 10001110) zum Monitor hinausgeschoben. Da jedes Bit einen Bildpunkt darstellt, werden jetzt 8 Punkte auf dem Bildschirm angezeigt. Ein dunkler Punkt entspricht einer 1 und ein heller einer 0. (ebenfalls in der Reihenfolge 10001110), siehe Bild 2.
Der Speicher ist für 4 Bildschirmseiten aufgebaut, die durch die Seitenumschaltung ausgewählt werden können. Es kann in eine Seite geschrieben und zugleich eine weitere gelesen werden. Das aus dem Schieberegister kommende Videosignal geht entweder direkt zum Monitor (TTL) oder zum Videomischer. In dieser Mischstufe wird das Video-Signal mit horizontalen und vertikalen Synchronisationssignalen (HS und VS) so aufbereitet, daß es danach als BAS-Signal (Bild Austast Synchron Signal) zur Verfügung steht.

Will man eine Hardcopy erstellen, muß man ähnlich der Datenübertragung zum Monitor den Bildspeicher auslesen. Das Auslesen des Bildschirmspeichers läuft parallel zum Display (Ausgeben der Bildschirminformation). Durch einen Befehl an den Graphikprozessor kann jeweils ein Byte des Bildschirmspeichers in die Rücklese Logik gelesen werden und dann über den Datenbus im Arbeitsspeicher abgelegt werden. Durch eine Drukkerroutine kann nun das Bild auf einen Drucker ausgegeben werden. Das Auslesendes Bildschirmspeichers ist auch für andere Aufgaben sehr nützlich (z.B. Vergleich von Bildschirmseiten, Abfrage des Maus-Zeigers für Graphikprogramme, Windowtechnik, usw.)

Die Hardware-Scroll-Logik dient zum Rollen (Scroll) des Bildschirms. Bei der bisherigen GDP64k wurde der Scroll softwaremäßig erzeugt. Sollte der Bildschirm bei Textdarstellung um eine Textzeile nach oben gescrollt werden, so mußte das gesamte Bild noch einmal aufgebaut werden (um eine Zeile versetzt). Dadurch war der Scroll natürlich sehr langsam und nur zeilenweise (Textzeile = 8 Bildschirmzeilen) möglich. Der Hardware-Scroll behebt diese beiden Mängel. Dabei wird der Bildschirmspeicher beim Scrollen nicht mehr verändert, sondern nur die Adressierung des Bildschirmspeichers. Dies wird durch zwei Addierer erledigt, die lediglich die Bildspeicheradressen, die vom Graphikprozessor kommen, mit einer Scrolladresse, die über Port 61 ausgegeben werden kann, aufaddiert und damit die neue Bildspeicheradresse erzeugt. Dadurch kann der Bildschirm zyklisch gescrollt werden, natürlich mit einer größeren Geschwindigkeit und "sanfter" (smooth-scroll).

Mit Hilfe des RMW-Modus kann man einfach bewegte Bilder erzeugen, da bei Schreibvorgängen in den Bildspeicher (=Monitor) alle Punkte komplementiert werden. Wenn z.B. ein Punkt gesetzt war, sowird er gelöscht, wenn er nicht gesetzt war, so wird er eingeschrieben. Der größte Vorteil des RMW-Modus ist, daß zerstörungsfrei gezeichnet werden kann. Dies ist z.B. nötig, um einen Maus-Zeiger oder ein Fadenkreuz auf einer Graphik oder einem Bild darzustellen, ohne das Hintergrundbild zu zerstören.
Die Dekodierlogik mit den Adressen A4...A7, den Signalen IORQ und M1 wird benötigt, um die GDP64HS bei I/O- Zugriff von 70...7F(Graphikprozessor) und 60...63h (Seitenport, RMW=Mode, Hardcopy und Hardscroll) anzusprechen.

Die Takterzeugung stellt alle benötigten Frequenzen (Pixelclock, Clock für Schieberegister, Takte für die Adressierung der Speicher) aus dem Grundtakt von 14 MHz her.

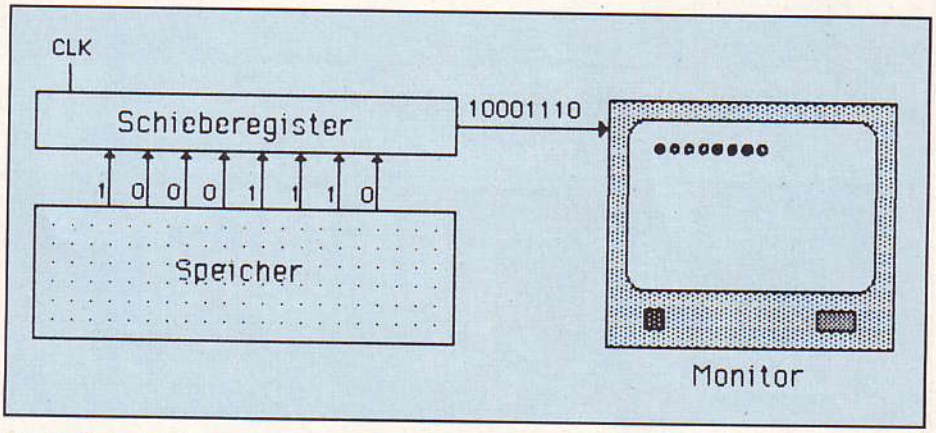


Bild 2: Prinzipielle Datenübertragung vom Speicher zum Monitor

Im neuen Grundprogramm 680xx wird die GDP64HS mit RMW, Hardscroll und Rücklesen bereits unterstützt. Beim Z80 ist ein neues FLOMON in Vorbereitung, das diese Optionen ebenfalls unterstützen wird. Sollten Sie die GDP mit der alten Software verwenden, ist diese natürlich absolut aufwärtskompatibel und verhält sich wie die alte GDP64k. Im Handbuch sind sowohl für Z80 als auch für 680xx Programme enthalten, die die neuen Funktionen der GDP64HS unterstützen.

Technische Daten

Spannungsverorgung	+5V
Stromaufnahme	500 mA
Busformat	NDR -Bus 54-polig ECB - Bus 64-polig
Leiterplattenformat	160 x 100 mm (Europakarte)
Ausgang	1: BAS: beinhaltet HS, VS und Video-Signal 2. TTL: beinhaltet HS, VS und Video-Signal (invertierbar)
Grafik-Controller	EF 9366 (Thomson-EFCIS) - kann 4 Seiten bedienen, wobei in eine geschrieben und zugleich eine weiter gelesen werden kann - integrierter ASCII-Zeichensatz
Speicher	64k RAM (dynamisch) - dadurch 4 Seiten
Sonstige Funktionen	Read Mody Write (zerstörfreies Zeichnen auf dem Bildschirm) Hardcopy (Rücklesen des Bildschirminhaltes und Ausgabe auf einen Drucker) Hardscroll (Scrollen des Bildschirmes mit Hilfe der Hardware, d.h. die Adressen des Bildschirmspeichers werden hardwaremäßig aufaddiert)

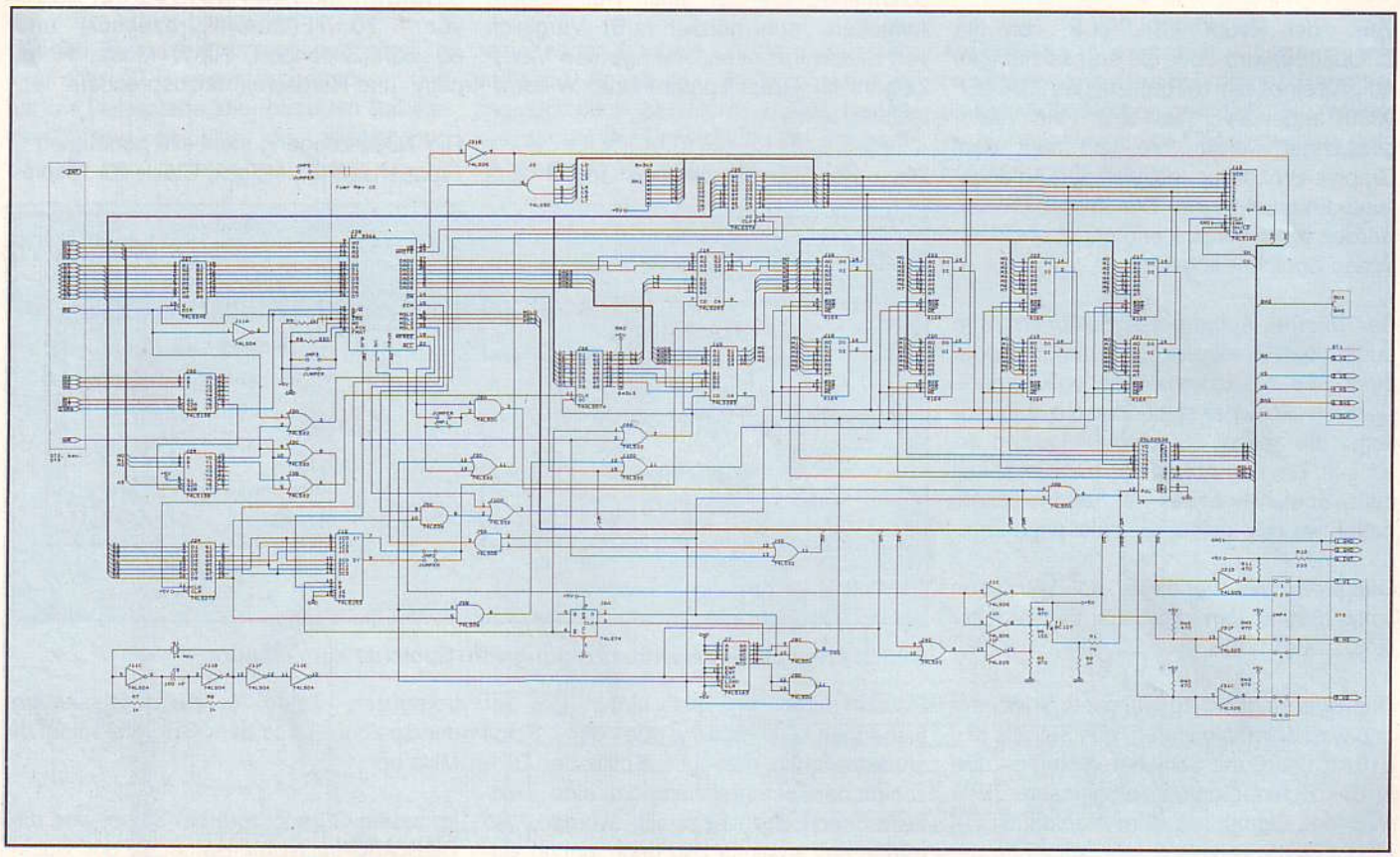


Bild 1: Schaltplan der GDP64HS

Spruch der Woche:
Bildung ist das, was übrig bleibt, wenn man alles vergessen hat, was man in der Schule gelernt hat

Albert Einstein

10 Jahre Graf - Computer

Geburtstagsfeier mit Tagen der offenen Türe bei GES

Am 15. und 16. April feierten mehr als 2000 Besucher aus Kempten und Umgebung mit uns Geburtstag und nahmen diese Gelegenheit war, sich die neuen Geschäftsräume in unserem Hause anzusehen. Ein besonderer Gag war das Torwandschießen, bei dem man einen BMW 320 i als Hauptpreis gewinnen konnte.
Bei strahlendem Sonnenschein und sommerlichen Temperaturen kam auch der gemütliche Teil nicht zu kurz. Alle Besucher zeigten sich von den neuen Räumen und den technischen Vorführungen beeindruckt.

Bild 1:Blick in die Entwicklungsabteilung

Bild 2: Besonderes starkes Interesse fanden die Führungen durch die Räumlichkeiten

Bild 3: Ein "schlagkräftiges" TEAM: GES zählt heute 40 Mitarbeiter

Neuer Katalog lieferbar

Endlich ist er da, die zweite Ausgabe des Farbkataloges mit noch mehr Information, mehr Produkten und Abbildungen. Er enthält statt bisher 168 jetzt 208 Seiten. Außerdem haben wir natürlich aus der ersten Ausgabe gelernt und auch für die Übersichtlichkeit bzw. Überblickbarkeit dieses "Nachschlagewerkes" etwas getan.

Die einzelnen Kapitel sind nicht mehr nach dem Einstiegsprinzip geordnet, sondern nach den einzelnen Funktionseinheiten eines Computers. So sind z.B. CPU´s in einem eigenen Kapitel, dann Speicherbaugruppen, usw. Vor diesen Kapiteln über die Funktionseinheiten gibt es einen erklärenden Block über die Einsetzbarkeit dieser Blöcke, Systemvorschläge bis hin zur Erläuterung von Datenbusbreiten und Taktfrequenzen. Danach wird die Software - wieder geordnet - und die Teachware (Bücher, Literatur) vorgestellt.

Neu in den Katalog aufgenommen wurden die Kapitel "mc-modular AT" und "Logiksimulator". Vor allem dem "mc-modular AT" wurden mehr als 10 Seiten gewidmet.

Ansonsten sind wie bisher sämtliche Baugruppen, Leiterplatten, Handbücher, Software und sonstige Dokumentation und Schulungssoftware zum NDR- und mc-Computer enthalten. Auch hier sind natürlich neue Baugruppen und vor allem viel neue Software enthalten.

Für die Anwender des NDR- und mc-Computer, sowie des mc-modular AT ist der neue Katalog das einzig übersichtliche Nachschlagwerk. Da der NDR-Computer und auch der Logiksimulator große Anteile an dem Mikrocomputer Ausbildungsmarkt hat, sollte dieser Katalog in keiner Schule oder Ausbildungsbetrieben, die nur irgendwie mit Mikroelektronik arbeiten, fehlen.

Bild 1: Ausschnitt aus dem neuen GES - Katalog

Softwarepartner stellen sich vor:

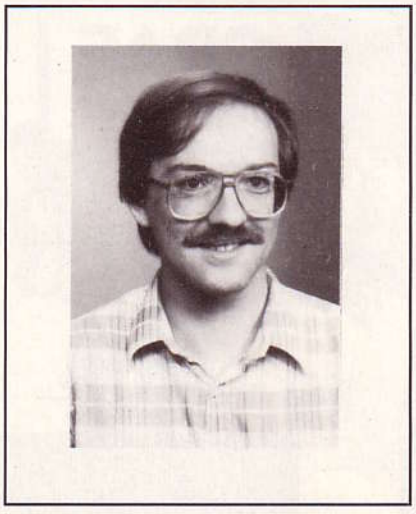
Persönliche Daten

Klaus Janßen
Hanninxweg 74
415 Krefeld 1

31 Jahre
verheiratet

Ausbildung und Beruf
1974 Abitur
1976-1982 Studium Elektrotechnik, Fachrichtung Nachrichtentechnik, an der Universität Duisburg 1982-1984 Wissenschaftlicher Mitarbeiter im Fachgebiet Meß- und Regelungstechnik seit 1984 Software-Entwicklungs-Ingegieur in der Industrie

Programmiere
seit circa 8 Jahren, zunächst in Fortran und Basic, später in Pascal, 8086-Assembler, 68000-Assembler, weniger in Modula 2, am liebsten in Pascal!



Mit dem NDR-Klein Computer (68008-Prozessor) begann ich - in meiner Freizeit - im Frühjahr 1985, weil es sich um ein offenes System mit einem leistungsfähigen Prozessor handelt, ich den 68000-Prozessor kennenlernen wollte und er ein breites Betätigungsfeld für die Softwareentwicklung bietet.

Mit JADOS

glaube ich, ein sehr leistungsstarkes Diskettenbetriebssystem für den NDR-Computer entwickelt zu haben. Meine weiteren Produkte sind der Disassembler DISASS, der Diskettendoktor INSPEC, das Strategiespiel REVERSI und zahlreiche kleinere Utilties.

Zur Zeit (Herbst 1987)

entwickle ich einen komfortablen Editor und nach einer Entscheidung über das Object-File-Format werde ich einen Linker und einen Assembler für das JADOS-System in Angriff nehmen. Ich hoffe, es wird sich jemand finden, der dann gleichzeitig an einem Compiler baut, denn meine Zeit ist leider begrenzt!

MeinTip für Anfänger:

Ein Programm immer erst auf dem Papier entwickeln!

NEU!
Die TURBO-TOOL-Diskette

Zum Betrieb der Diskette wird das Betriebssystem CP/M benötigt. Die Programme können über das jeweilige .PAS - File (=Pascal Quelltext) oder .COM - File gestartet werden.

Nach unserem "Aufruf an alle Software-Freaks !" in LOOP 15/3 können wir Ihnen jetzt als erstes Ergebnis dieser Aktion die TURBO-PASCAL-TOOL Diskette vorstellen. Die Diskette wurde ausschließlich aus Kundenroutinen erstellt. Für die vielen zugesandten Programme dürfen wir uns noch einmal ganz herzlich bedanken.

Die Programme dienen teilweise als Software-Werkzeuge im Umgang mit Turbo-Pascal, teilweise als Anwenderprogramme mit Lerneffekt oder spielerischem Charakter.

Vokabeltrainer
zum Einüben von bis zu 100 Vokabeln. Abfrage der Vokabeln erfolgt per Zufallsgenerator.

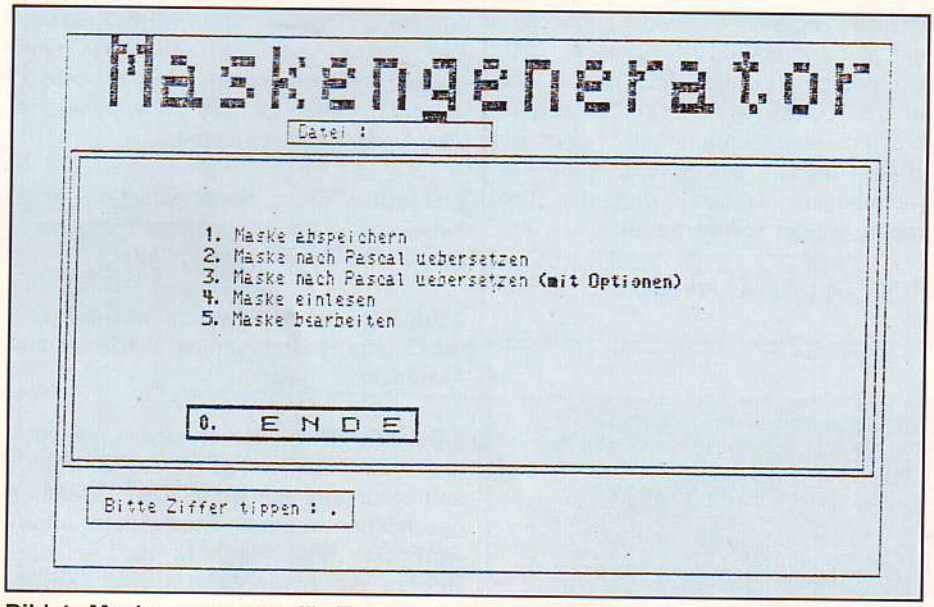


Bild 1: Maskengenerator für Turbo-Pascal-Programme

Eine kurze Beschreibung für einen Teil der Programme:

Maskenroutine
Das Programm Maskengenerator dient zum Erstellen von Bildschirmmasken unter Turbo-Pascal. Es verarbeitet den Text auf dem Bildschirm mit allen angegebenen Ein- und Ausgaben in Pascal Source-Code. Die Graphik des NDR-Computers wird hierbei voll berücksichtigt.
Die Copyright Meldung in Bild 1 wurde mit dieser Maskenroutine erstellt.

Vokabelprogramm
zum schnellen Nachschlagen von beliebigen Vokabel bzw. Übersetzungen als Vokabellexikon universell verwendbar

Diskettenverwaltung
Druckerausgabe im Diskettenformat
Erfassen und Sortieren der Disketten

Mathematische Funktionen
Berechnung mit 3 Real - Variablen

Puzzle Spiel
unterstützt von Mouse-Tools

Mühlespiel
Jeder kennt dieses Spiel. Die Vorlage zu diesem Spiel wurde CHIP-SPEZIAL entnommen und für die Grafischen Möglichkeiten des NDR-Computers umgeschrieben. (siehe Bild 2)
An dieser Stelle noch einmal vielen Dank für freundliche Genehmigung der CHIP-Redaktion für die Verwendung des Programms auf der Turbo-Tool-Diskette.

Virgula
"Würfelspiel" für 2-6 Personen

Routinen zur Ansteuerung der **COL256**
Abspeichern und Laden des Bildschirminhaltes
Ansteuerung der CLUT

Ansteuerroutine für die Grafik der GDP64K

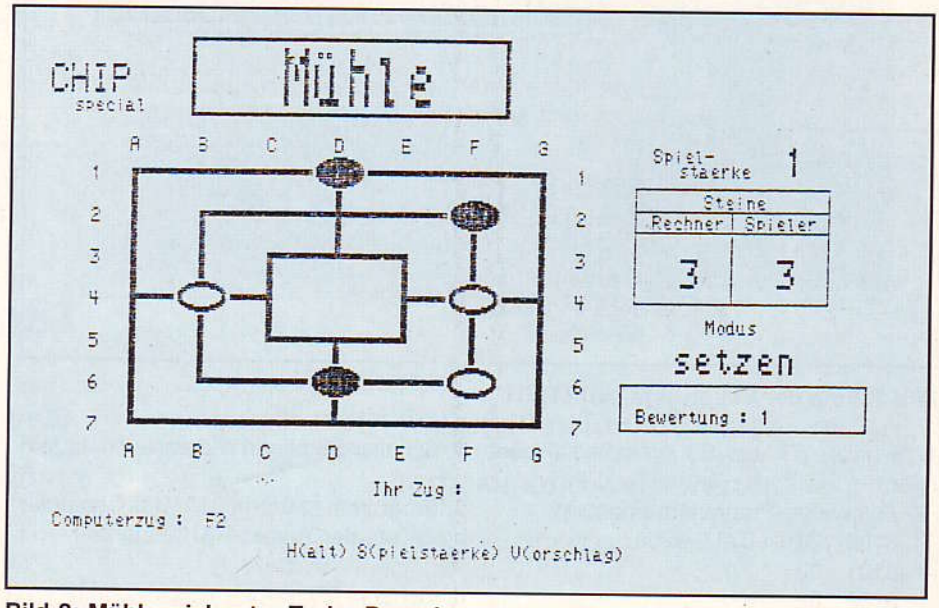


Bild 2: Mühlespiel unter Turbo-Pascal

Klaus Bischof

Verstehen von Rechnersignalen mit der neuen Baugruppe

BUSTEST

Jeder hat einmal klein angefangen. Dies gilt wohl auch in der Mikrocomputerei und natürlich auch für den NDR-Anwender. Dabei steht meistens am Anfang ein riesiges Loch bezüglich der Hardware und der Hardwareabläufe. Will man in die Hardware einsteigen, sieht man sich gezwungen, Stapel von Literatur, speziell Datenblätter, durchzuwühlen und kann das Ganze natürlich nur theoretisch (daher auch meist sehr trocken) bearbeiten.

Mit der Baugruppe BUSTEST kann man jetzt die Abläufe des Rechners sichtbar machen und somit sehen, was im Rechner wirklich vor sich geht. Diese Baugruppe hält den Rechner nach jedem Befehlszyklus an und stellt die entsprechenden Rechnersignale (die auf dem BUS liegen) mit Leuchtdioden dar. Mit dem folgenden kleinen Programm wollen wir Ihnen zeigen, wie Sie mit Hilfe des Einsteigerpaketes und der BUSTEST Rechnerabläufe darstellen können.

Anwendungsbeispiel mit dem Einsteigerpaket Z80

Mit Hilfe dieses Beispiels sind die Speicherzugriffe (Lese- und Schreib-Befehl) sehr gut zu erkennen. Dies kann durch schrittweises Abarbeiten der nachfolgenden Punkte geschehen.

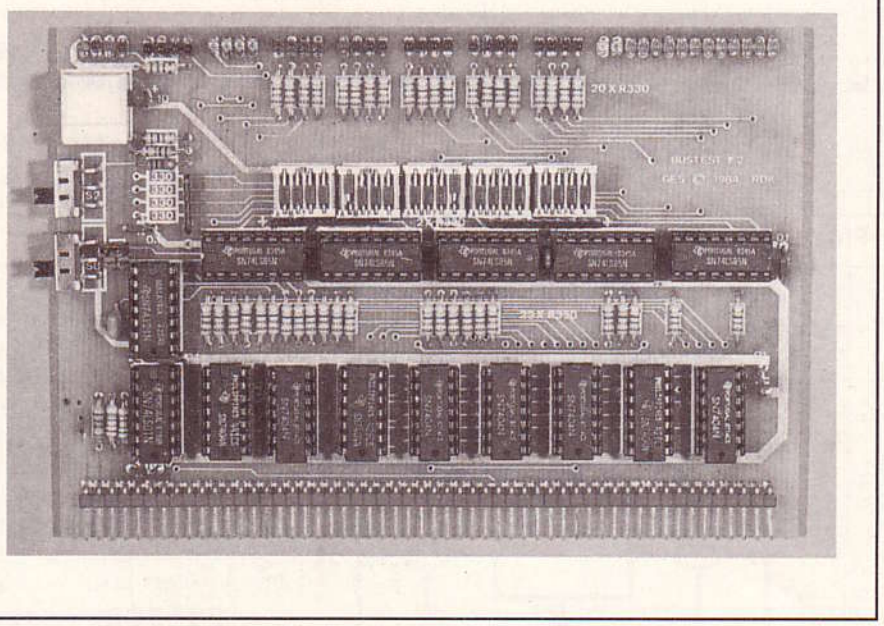


Bild 1: Foto der Baugruppe "BUSTEST"

1. Schalter S7 und S8 schließen (Hebel oben)
2. Folgendes Programm eingeben:

```
8100  CD   CALL 8200
8101  00
8102  82
8103  C3   JP 8100
8104  00
8105  81
8200  C9   RET
```

2.1 Das Programm verzweigt zunächst in ein Unterprogramm. In Adresse 8200 wird durch den Befehl "RET" sofort wieder ins Hauptprogramm zurückgesprungen. Im Hauptprogramm steht unter der Adresse 8103h ein Sprung zum Programmbeginn. Danach läuft das Programm wieder von vorne ab.

3. DIL - Schalter auf Adresse 8100 einstellen

4. Schalter S8 öffnen (Adresse einfangen)

5. Programm starten (Der Computer bleibt bei der Adresse 8100 stehen)

6. Schalter S7 öffnen (Einzelschritt)

7. Durch Taster das Programm Schritt für Schritt ablaufen lassen.
Dabei ergeben sich folgende Bitmuster:
(siehe Bild 2 --->)
Die oben abgebildeten Bitmuster entsprechen den Leuchtdioden auf der Bustest Baugruppe. Im folgenden Absatz werden diese Bitmuster kurz erläutert.

1.Schritt: Auf Adresse 8100 steht der Befehl CD (CALL). Die Steuersignale RD und MREQ werden aktiv. MREQ signalisiert, daß auf dem Adressbus eine gültige Adresse anliegt. Der Prozessor ist bereit, den Inhalt des Datenbusses in den Akkumulator zu laden.

2.Schritt: Die niederwertigen zwei Bytes der Unterprogrammadresse werden in den Akkumulator geladen.

3.Schritt: Die höherwertigen zwei Bytes der Unterprogrammadresse werden in den Akkumulator geladen.

4.Schritt: Das Steuersignal WR wird aktiv. Die höherwertigen Bytes der Rücksprungadresse werden in den Speicher geschrieben (auf dem Stackpointer abgelegt). Der Stack liegt ab der Adresse BFFFh; die Bytes BFFEh und BFFDh werden nun beschrieben.

5.Schritt: Die niederwertigen Bytes werden in den Speicher geschrieben.

6.Schritt: Das Programm ist auf die Adresse 8200 gesprungen. Hier wird der Befehl C9 (RTS) in den Prozessor gelesen.

7.Schritt: Die höherwertigen Bytes der Rücksprungadresse werden vom Stackpointer geholt.

8.Schritt:: Die niederwertigen Bytes der Rücksprungadresse werden vom Stackpointer geholt.

9.Schritt: Der Befehl C3 (JP) auf Adresse 8103 wird vom Prozessor gelesen.

10.Schritt: Die niederwertigen Bytes der Sprungadresse werden gelesen.

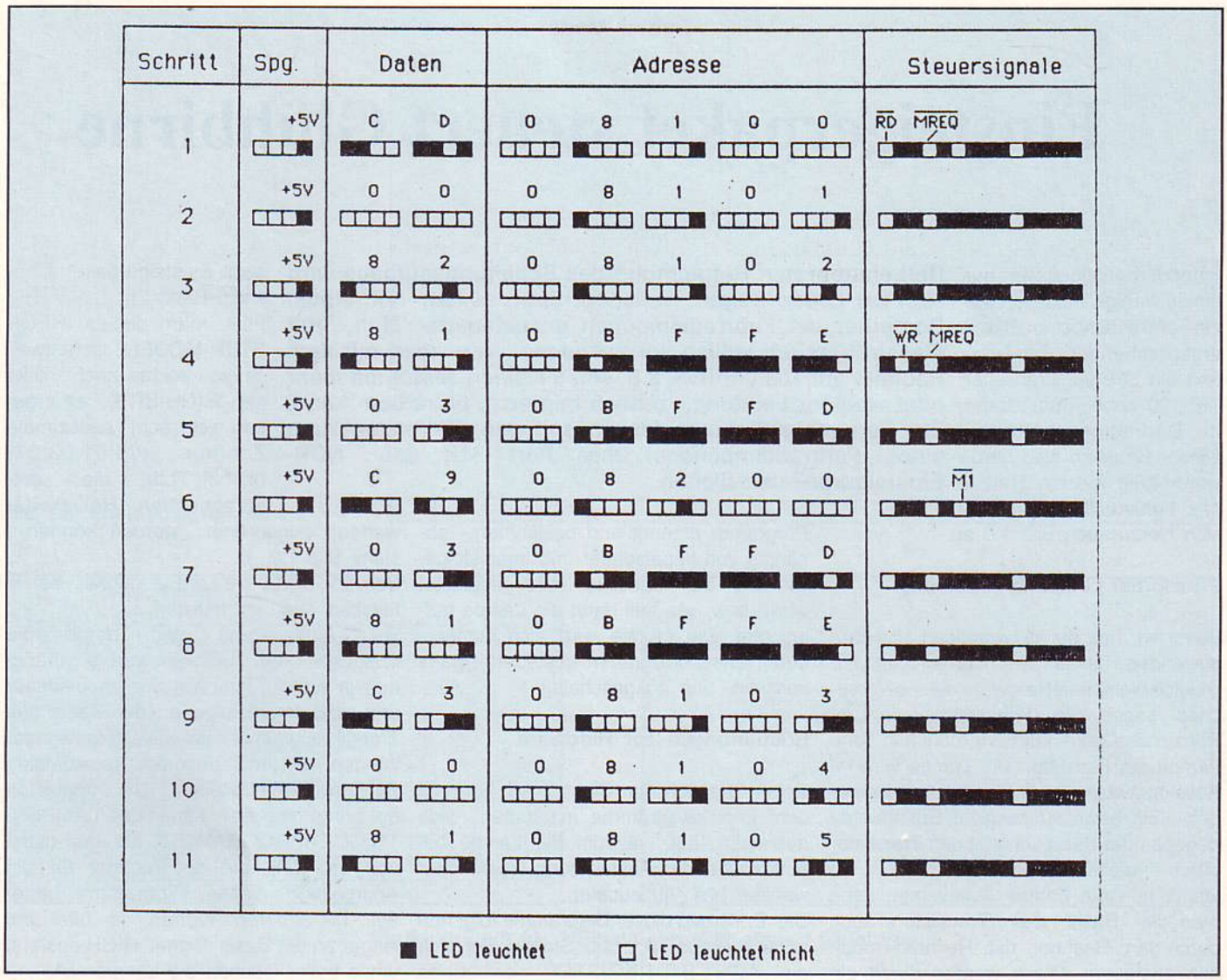


Bild 2: Bitmuster

11.Schritt: Die höherwertigen Bytes der Sprungadresse werden gelesen.

12.Schritt: Das Programm beginnt wieder von vorne.

An diesem Beispiel läßt sich der Ablauf des Rechners schön darstellen. Vor allem kann man erkennen, welche Signale wann aktiv sind.
Die Baugruppe BUSTEST ist vor allem für die Selbstausbildung gedacht.
Sollte sie dann als Ausbildungsbaugruppe ihren Dienst getan haben, kann sie aber immer noch sinnvoll bei der Fehlersuche - auch in Programmen - eingesetzt werden.

Nikolaus Bischof, GES

Theorie ist

wenn man alles weiß

und nichts funktioniert

Praxis ist

wenn alles funktioniert

und niemand weiß warum

Gunter Mader

Einsteigerpaket steuert Glühbirne

Bei einer ersten Betrachtung des Schaltungsaufbaus wird sich der Leser fragen, welchen Sinn es hat, mit einem Computer ein Fahrradlämpchen anzusteuern. Nun, mit diesem Versuch wollen wir aufzeigen, wie man mit dem Rechner auf Tastendruck z.B. einen kleinen Motor mit mehr oder weniger Leistung, zeitlich begrenzt, betreiben kann. Zur Veranschaulichung dessen soll uns die Ansteuerung eines Fahrradlämpchens über Port 31h des NDR-Einsteigerpakets dienen.

Hierzu benötigen wir nur einige wenige Bauteile, wie ein Fahrradlämpchen 6V mit entsprechender Fassung und ein Leistungstransistor TIP 120 (zwei Transistoren in Darlingtonschaltung). Dieser Baustein wird hardwaremäßig wie ein Transistor behandelt. Sehen Sie sich hierzu auch Bild 1.0 an.

Prinzip der Lampenansteuerung

Bevor wir uns der notwendigen Software zuwenden, lassen Sie mich einiges zur wirklich einfachen Hardware dieses Versuches sagen. Die Schaltung besteht im Prinzip aus dem Transistorbaustein und Lampe mit Fassung. Die Lampe wird im Kollektorzweig des Transistors betrieben. Ein gewisser positiver Spannungspegel an der Basis schaltet den Transistor durch - die Lampe leuchtet.
Damit nun die Lampe aufleuchtet, kann man die Basis des Transistors auch durch den Rechner mit Rechteckimpulsen ansteuern. Diese werden durch ein Programm erzeugt und bestimmen, abhängig von Impulsdauer und Impulsfolge, wie oft der Transistor durchgeschaltet wird, bzw. wie hell dann die Lampe aufleuchtet. Die Lampe wird also immer - durch unser Programm gesteuert - ganz kurz ein- und ausgeschaltet.

Erläuterungen zur Hardware

Sehen Sie sich nun Bild 1.2 an. Es zeigt acht Impulsdiagramme mit denen sich aufzeigen läßt, warum die Lampe bei entsprechender Ansteuerung mehr oder weniger hell aufleuchtet.
Die Erklärung dieser Diagramme folgt nun anhand eines Modells. Stellen Sie sich ein ACHT-BIT-REGISTER vor (siehe auch Einsteigerpaket, Christiani Band 2).
Füllt man dieses REGISTER-MODELL schrittweise von rechts nach links mit 'HIGH-BITS', so ergeben sich acht sedezimale Zah-len (00;01;03;07; 0f;1f;3f; 7f;ff), die entsprechenden Helligkeitswerten zugeordnet werden können (siehe Bild 1.2).
Bei '00' bleibt die Lampe dunkel, bei 'ff' leuchtet sie am hellsten.
Die Ansteuerung und damit die Ausgabe des Helligkeitswertes hängt nun in erster Linie von der Impulsdauer und der Impulsfolge an der Basis des Transitorelements ab. Die Impulsdauer können wir mit unseren sedezimalen Zahlenwerten einstellen. Die Impulsfolge hängt von der Länge des 'LAMPEN-TRIGGERPROGRAMMS' ab und damit von der Zeit, die der Rechner für die Abarbeitung dieses Programms benötigt. Mit anderen Worten, je öfter und länger an der Basis Signal HIGH ansteht, umso heller leuchtet die Lampe auf.

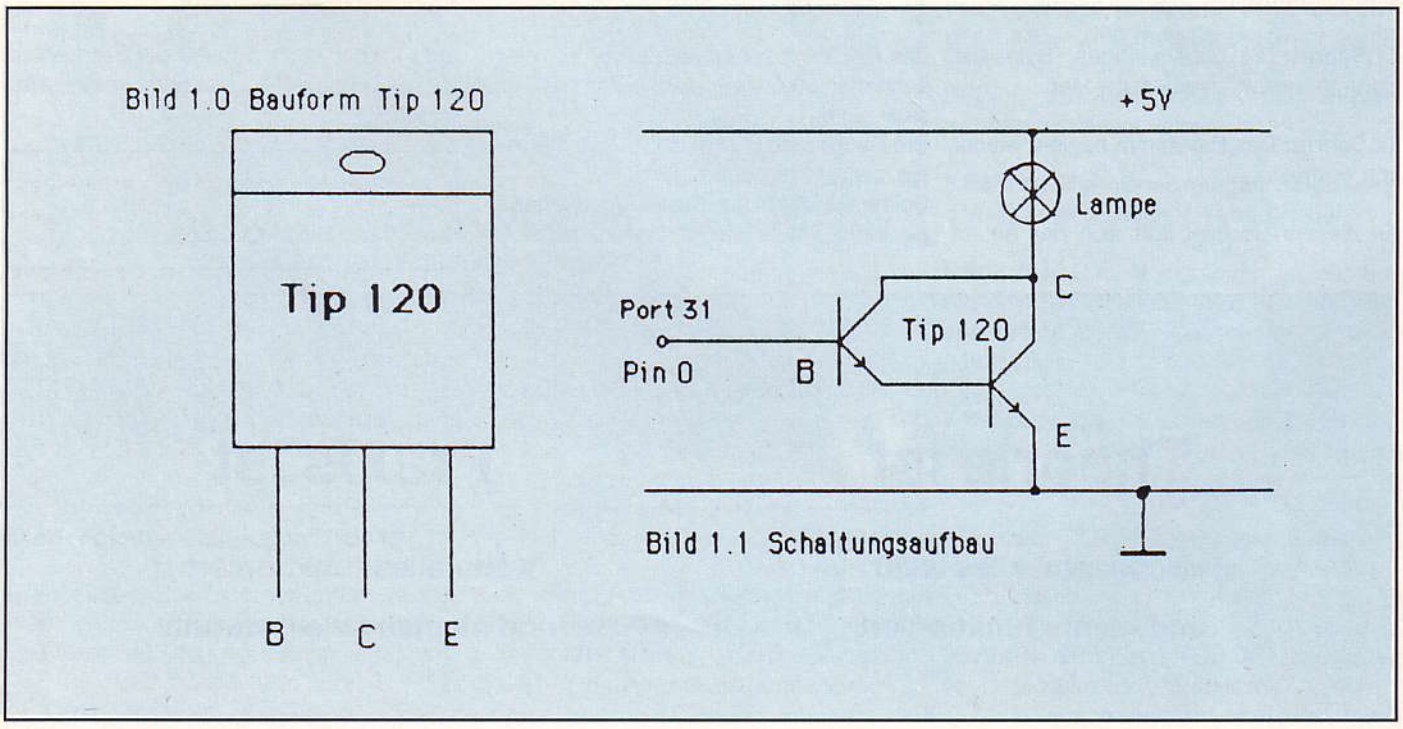
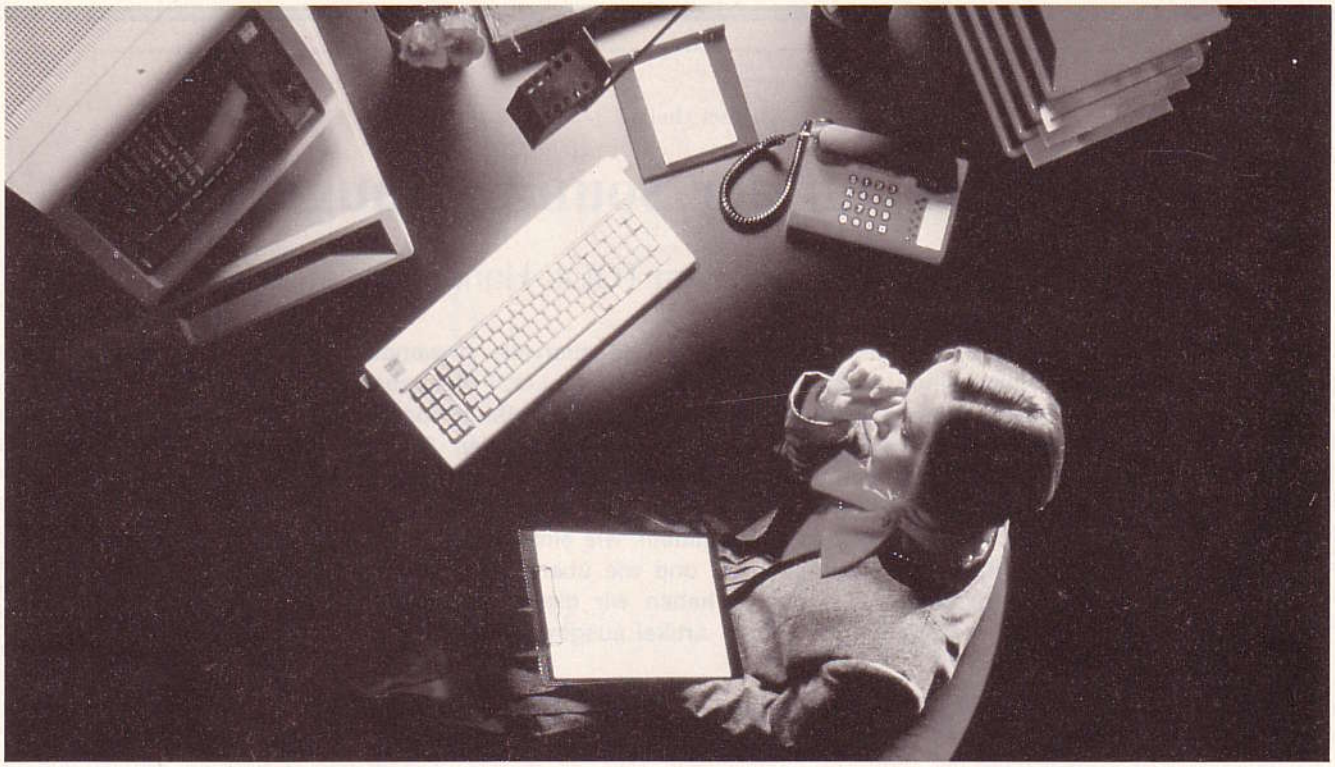


Bild 1.0 und 1.1

BBW BIERBÜSSE, FOTO: THE IMAGE BANK



Durch mc wissen Computer-Profis, was der Markt zu bieten hat:

In der Juli-Ausgabe zeigt mc u. a. 100 Matrixdrucker.

Mit mc sind Sie bestens informiert.
Denn mc wird für Profis gemacht, die mit und durch den Computer Geld verdienen.
Sie lesen u. a. in der Juli-Ausgabe:

Computer-Tuning
Wenn ein PC-XT oder AT zu langsam ist, kann man ihn mit Einsteckkarten schneller machen. Höchstgeschwindigkeiten erreicht man mit dem Inboard-386.

Künstliche Wesen
Wie sich biologische Schaltvorgänge in Modula 2 simulieren lassen.

Workshop
Im mc-Juli-Heft beginnt die neue Serie „C-Workshop". Fundiertes Wissen für die Praxis. Präprozessor-Kommandos für Turbo C.

mc-Report
Über Hacker und Computerviren…

mc gibt es jeden Monat neu an jeder größeren Zeitschriften-Verkaufsstelle zum Preis von DM 7,-.
Die Juli-Ausgabe ist ab 27. Juni im Handel

Die Mikrocomputer-Zeitschrift - bringt Profis weiter.

Sollte die neueste mc an Ihrer Zeitschriften-Verkaufsstelle vergriffen sein, senden wir Ihnen einmalig ein kostenloses Probeheft.
Frankierte Postkarte an Franzis-Verlag, Postfach 37 02 80 L18, 8000 München 37, mit Berufsangabe und Vermerk „Bitte Probeheft mc" oder Anruf zum Ortstarif (**Tel.: 01 30-51 17**) genügt.

Michael Giulini

Tonleiter auf der Sound - Baugruppe

Ein Auszug aus dem TOOL-Handbuch

Daß mit dem Einsteigerpaket bereits sinnvolle Anwendungen möglich sind, wurde bereits in der letzten LOOP-Ausgaben bewiesen. Alle unsere Standard EIN/AUSGABE-Baugruppen sind mit dem Einsteigerpaket programmierbar. Um den Anwendern das Programmieren zu erleichtern, haben wir, wie bereits in der letzten LOOP erwähnt, ein TOOL-EPROM mit Handbuch geschaffen. Um Ihnen ein Bild zu vermitteln, wie einfach die Baugruppe SOUND zu programmieren ist und wie übersichtlich das Handbuch dazu gestaltet wurde, haben wir diesen Ausschnitt aus diesem Handbuch für diesen Artikel ausgewählt.

Das Programm Sound1 belegt die 13 Tasten 0-C mit den Tönen einer Oktave von C-C' (einschließlich der Halbtöne). Zusätzlich kann mit der Plus-Taste die Lautstärke erhöht, mit der Minus-Taste erniedrigt werden.
Die Sound-Baugruppe wird über die Adressen E0h und E1h angesteuert. Dabei wird über die Adresse E0h ein internes Register der Karte ausgewählt, über die Adresse E1h werden dann die Daten an das angewählte Register ausgegeben.

Programmablauf

Zuerst wird das Register 7 angewählt und der Ton abgeschaltet. Dies ist wichtig, wenn am Programmende wieder zum Anfang gesprungen wird.

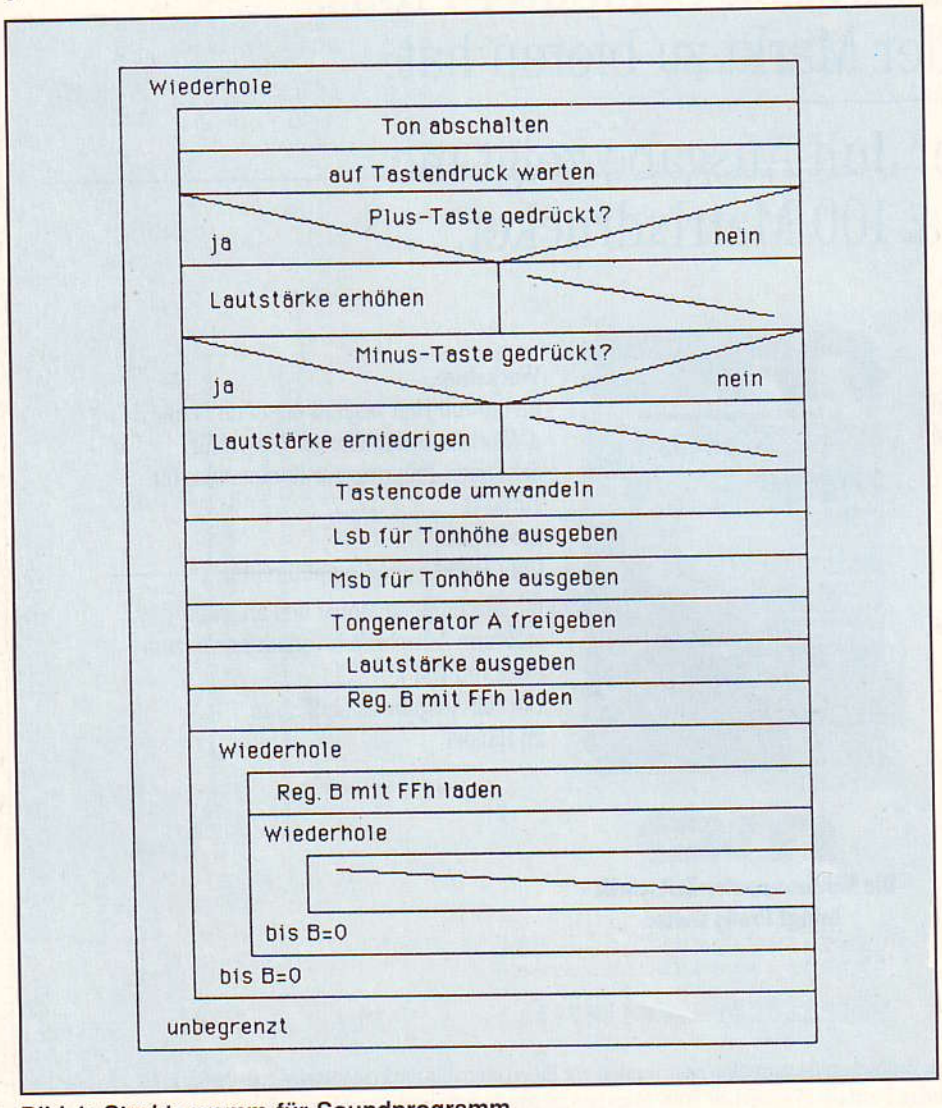


Bild 1: Struktogramm für Soundprogramm

Die Betriebssystemroutine HOLETASTE wartet auf einen Tastendruck und speichert den Tastencode dann im Akku ab. Durch den Befehl "cp" wird geprüft, ob entweder die Minus-Taste oder die Plus-Taste gedrückt wurde. Ist dies der Fall, so wird das D-Register (Lautstärke) erhöht (+) bzw. erniedrigt (-).
Anschließend wird der Tastencode durch die Routine TONUM umgewandelt. In das IX-Register wird die Anfangsadresse der Tabelle geladen, in der die Werte für die Noten stehen. Dazu wird der doppelte Wert der gedrückten Taste hinzuaddiert, damit die richtigen Notenwerte aus der Tabelle geholt werden können.
Jetzt wird der Kanal 0 der Sound-Karte angewählt und das LSB der Tonhöhe an die Baugruppe ausgegeben. Identisch wird das MSB derTonfrequenz dem Kanal 1 übermittelt.
Über den Kanal 7 wird das Byte 11111110b gesendet, das den Tongenerator A freigibt. Nun muß nur noch über das Register 8 die Lautstärke festgelegt werden, damit der Ton im angeschlossenen Lautsprecher gehört werden kann.
Um nicht nur ein kurzes Knacksen im Lautsprecher zu hören, muß durch eine Verzögerungsschleife das Ausschalten des Tones hinaus geschoben werden. Um eine respektable Tonlänge zu erreichen, werden zwei Schleifen, die von FFh bis 0h hinabzählen, ineinandergeschachtelt.
Abschließend wird zum Programmanfang gesprungen, wo der Ton ausgeschaltet wird und das Programm auf einen neuen Tastendruck wartet.

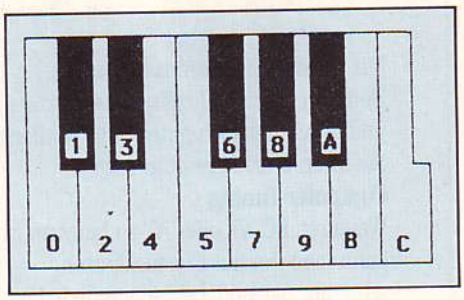


Bild 2: Tastenbelegung

Bild 3: Listing für Programm Sound1

```
MACRO-80 3.43  27-Jul-81      PAGE  1


                        .z80
                        ;****************************************
                        ;*   Tonleiter auf Sound-Baugruppe      *
                        ;*    Sound1 V 1.0                      *
                        ;*   Michael Giulini  29. Sept. 1987    *
                        ;****************************************


                              org 8100h          ; Speicheradresse für Programmanfang
00E0                          sound equ 0e0h     ; Adresse der SOUND-Baugruppe
000C                          holetaste equ 000ch ; Sprungadresse für Betriebssystemsroutine HOLETASTE
000F                          tonum equ 000fh    ; Sprungadresse für Betriebssystemsroutine TONUM

8100'  3E 07            start:  ld a,7h          ; Adresse 7 von
8102'  D3 E0                    out (sound),a    ; Sound-Baugruppe anwählen
8104'  3E FF                    ld a,0ffh
8106'  D3 E1                    out (sound+1),a  ; Ton abschalten
8108'  06 00                    ld b,0h
810A'  CD 000C                  call holetaste   ; auf Tastendruck warten
810D'  FE 5D                    cp 5dh           ; wenn "+"-Taste gedrückt ist,
810F'  C2 8113'                 jp nz, dec
8112'  14                       inc d            ; Lautstärke erhöhen
8113'  FE 5E             dec:  cp 5eh           ; wenn "-"-Taste gedrückt ist,
8115'  C2 8119'                 jp nz, ton
8118'  15                       dec d            ; Lautstärke erniedrigen
8119'  CD 000F             ton:  call tonum       ; Tastencode umwandeln
811C'  DD 21 8154'              ld ix,noten      ; ix-Reg. auf Notentabelle stellen
8120'  4F                       ld c,a           ; Tastennummer nach Reg. C
8121'  DD 09                    add ix,bc        ; Tastennummer *2 zu Startadresse der
8123'  DD 09                    add ix,bc        ; Notentabelle addieren
8125'  3E 00                    ld a,0h
8127'  D3 E0                    out (sound),a    ; Kanal 0 von Sound-Karte anwählen
8129'  DD 7E 00                 ld a,(ix)        ; Lsb von Ton aus Tabelle holen
812C'  D3 E1                    out (sound+1),a  ; und an Sound-Karte übergeben
812E'  3E 01                    ld a,1h
8130'  D3 E0                    out (sound),a    ; Kanal 1 anwählen
8132'  DD 7E 01                 ld a,(ix+1)      ; Msb von Ton aus Tabelle holen
8135'  D3 E1                    out (sound+1),a  ; und an Sound-Karte übergeben
8137'  3E 07                    ld a,7h
8139'  D3 E0                    out (sound),a    ; Kanal 7 anwählen
813B'  3E FE                    ld a,11111110b
813D'  D3 E1                    out (sound+1),a  ; Tongenerator A freigeben
813F'  3E 08                    ld a,8h
8141'  D3 E0                    out (sound),a    ; Kanal 8 anwählen
8143'  7A                       ld a,d           ; Lautstärke nach Reg. A
8144'  D3 E1                    out (sound+1),a  ; Lautstärke an Sound ausgeben
8146'  06 FF                    ld b,0ffh        ; Verzögerungsschleife:
8148'  48               halten:  ld c,b           ; zwei ineinandergeschachtelte
8149'  06 FF                    ld b,0ffh        ; Schleifen, in denen jeweils von
814B'  00               schleife: nop            ; FFh hinabgezählt wird
814C'  10 FD                    djnz schleife
```

```
MACRO-80 3.43  27-Jul-81      PAGE  1-1

814E'  41                       ld b,c
814F'  10 F7                    djnz halten
8151'  C3 8100'                 jp start         ; Sprung zum Programmstart

8154'  DE 01 C3 01        noten:  db  0deh,01h, 0c3h,01h, 0aah,01h, 92h,01h, 7bh,01h, 66h,01h, 52h,01h
8158'  AA 01 92 01
815C'  7B 01 66 01
8160'  52 01
8162'  3F 01 2D 01              db  3fh,01h, 2dh,01h, 1ch,01h, 0ch,01h, 0fdh,00h, 0efh,00h
8166'  1C 01 0C 01
816A'  FD 00 EF 00



                        ; ACHTUNG: Bei Sprungadressen muß erst das LSB und dann das MSB
                        ;          der Adresse auf der HEXIO eingegeben werden!

                        ; Beispiel: Listing:  810F'   C2 8113'         jp nz, dec
                        ;           eingeben:    C2 13 81

                          end
```

```
MACRO-80 3.43  27-Jul-81      PAGE  S

Macros:

Symbols:
8113'  DEC              8148'  HALTEN          000C   HOLETASTE
8154'  NOTEN            814B'  SCHLEIFE        00E0   SOUND
8100'  START            8119'  TON             000F   TONUM


No Fatal error(s)
```

Christoph Köhler

Wunschkonzert

Mit diesem Artikel beginnt eine Serie, um die Zusammenarbeit zwischen den NDR-Computer-Anwendern auszubauen. Wir nennen die mit diesem Artikel begonnene Aktion schlicht "Wunschkonzert".
Der Ablauf des neuen Wunschkonzertes ist folgendermaßen:

Spaßhalber kann man die Aktion statt einer genauen Erklärung auch durch ein Struktogramm erläutern. Versuchen Sie doch ausnahmsweise erst das "Struktogramm" (Bild 1) zu verstehen, bevor Sie das "Pflichtenheft" lesen.

Das Pflichtenheft

Das Wunschkonzert beginnt damit, daß Sie (wir) sich Programme oder Routinen wünschen, die Sie als eine Bereicherung für Ihren Rechner betrachten. Dabei kommen alle Ausbaustufen des NDR-Computers in Frage. Anschließend werden wir Ihre Anregung in der LOOP veröffentlichen. Falls sich jemand in der Lage fühlt, einen der Wünsche kurz- oder langfristig zu erfüllen, melde er sich bei der LOOP-Redaktion.
Um Parallelentwicklung zu vermeiden, können wir noch gegebenfalls Kontakt mit Ihnen aufnehmen. Dann beginnt die "Programmiermühle" zu mahlen.

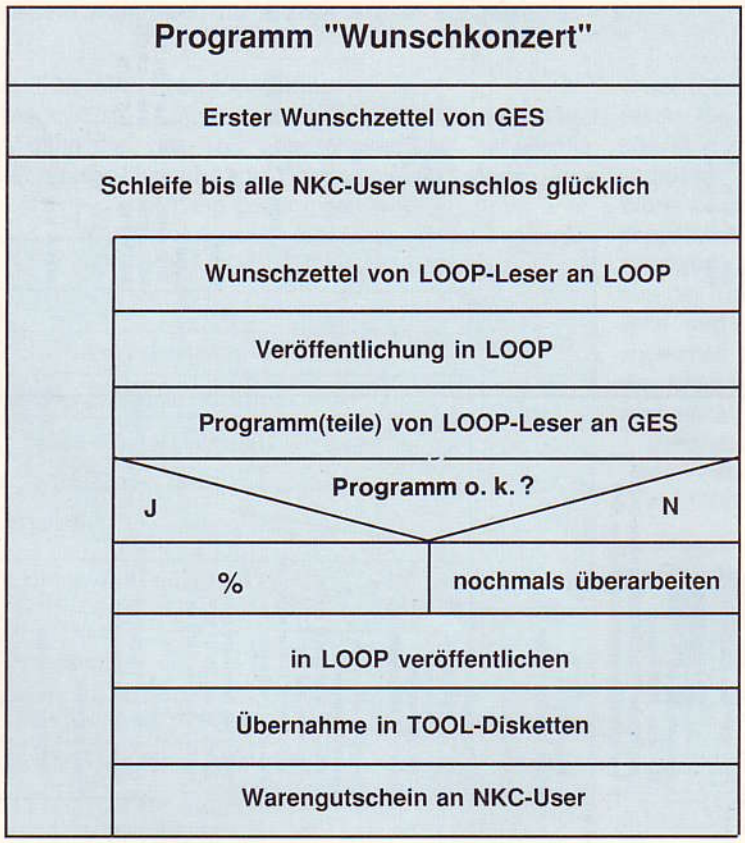


Bild 1. Das Struktogramm der Aktion Wunschkonzert

Selbstverständlich soll das nicht umsonst sein.
Es werden alle Beiträge, die auf TOOL-Disketten übernommen werden, mit Warengutscheinen belohnt und deren Autoren entsprechend auf der Diskette und gegebenfalls in der LOOP namentlich veröffentlicht. Unvollständige oder fehlerhafte Programme können eventuell in "Teamwork" vervollständigt werden.

Es ist also ein interessanter Versuch !

Neuentwicklungen von Programmen können von Anwender für Anwender mitgestaltet und vor allen Dingen auch mitverfolgt werden.

Als Auftakt stellen wir den ersten Wunschzettel.

An unserem Wunschzettel können Sie ersehen, in welcher Form Ihre Wunschzettel eingegeben werden könnten. Am besten teilen Sie es auf in:
- Hard- und Softwarevoraussetzungen
- Leistungsbeschreibung
- Skizzen
- Tips bzw. eigene Routinen

Nun aber zu unserem Wunschzettel:

Programm: "Ringpuffer"
CPU: Z80 oder 680xx
Betr. Sys.: CP/M xx oder JADOS
Sprache: unerheblich

Haben Sie sich nicht schon mal geärgert, wenn Sie nach einer Kommando-Eingabe auf Ihrem Betriebssystem wegen einem falschen Buchstaben die ganze Zeile neu eingeben müssen. Logischerweise stellt sich in diesem Fall Ihr Betriebssystem auf "stur".
Abhilfe dazu wäre das Programm "Ringpuffer", das sich am besten gleich die letzten 20 Eingaben merkt. Diese jeweiligen 20 letzten Kommandos könnten mit den Pfeiltasten durchgeblättert werden. Jedes Kommando kann von Ihnen editiert werden (Cursor-Steuerung mit CTRL S,D,A,F,T,Y ..) und dann von jeder Cursor-Stellung aus übernommen werden. Die geblätterten Kommandos bleiben immer in der aktuellen Zeile nach dem Prompt. (z.B. A>)

Nun aber "RUN" !

Hiermit starten wir das "Programm" Wunschkonzert und werden es betreiben, solange Sie Spaß daran haben.
Schreiben Sie Ihre Vorstellungen an die LOOP-Redaktion mit dem Stichwort "Wunschkonzert".
Wir freuen uns auf eine gute Zusammenarbeit mit Ihnen.

Die LOOP - Redaktion

Leo Hepperger

Für Z80-Grundprogramm:

Symbole am Drucker ausgeben

Durch dieses Programm erspart man sich das Abschreiben von Symbole vom Bildschirm. Auch während dem Programmieren leistet das Programm "Symboltabelle" gute Dienste. Das Programm druckt die Symboltabelle so aus, wie sie am Bildschirm erscheint. Die Symbole werden absichtlich untereinander gedruckt, um Bemerkungen anhängen zu können.

Geben Sie "TESTPRINT := 0613H" ein, sonst kann dieses interne Grundprogramm nicht angesprungen werden. Start mit ST oder 0F333H. Noch ein Tip: Wird zu einem Programm (im Speicher) dazugeladen, werden dessen Symbole geloescht (bzw. Zeiger neu gesetzt). Daher SYMBOLTABELLE immer zuerst laden, oder resident im Speicher halten. (Bei mir der Fall). Wird jetzt ein Programm dazugeladen, werden die Symbole von SYMBOLTABELLE geloescht, daher mit 0F333H starten.

Leo Hepperger
Am Schlag 7
8031 Eichenau

```
Start-Adresse:0F333H

End-Adresse:0F447H
F333 CDD5F3   ST: CALL TEXT6         ;ST (Symbol Tabelle) =Programmstart
F336 3E00      LD A,00H              ;loescht A
F338 CD2700   SLOOP: CALL CSTS       ;ist Taste gedrueckt? (aus GRUND)
F33B CB7F      BIT 7,A               ;dann A=FF
F33D 2007      JR NZ,START           ;dann Sprung nach START
F33F 00        NOP
F340 18F6      JR SLOOP              ;sonst zurueck nach SLOOP
F342 00        NOP
F343 CD3000    CALL CLR              ; Bildschirm loeschen (aus GRUND)
F346 21C981   START: LD HL,SS        ;START der Ausgabe,SS=1.Byte v.Sym.Speich.
F349 4E       NEXT: LD C,(HL)        ;Inhalt von Speicher nach C
F34A 3E00      LD A,00H              ;loescht A
F34C B9        CP C                 ;A-C,wenn C 00=Symb.Speich.leer oder Ende
F34D CA0000    JP Z,0000H            ;dann Sprung ins GRUND-Menue
F350 110000    LD DE,0000H           ;loescht DE
F353 7E       ANZAHL: LD A,(HL)      ;Anzahl der auszugebenden Zeichen
F354 13        INC DE                ;zaehlt DE
F355 23        INC HL                ;fuer naechstesmal erhoehen
F356 CB7F      BIT 7,A               ;letztes Zeich.v.Symb.? (=ASCII Wert+80H)
F358 28F9      JR Z,ANZAHL           ;Sprung wenn nicht letztes Zeichen
F35A 0602      LD B,02H              ;Schleifenzaehler
F35C 23        INC HL                ;fuer H-Byte von Adresse
F35D CD8AF3    CALL SPACE            ;fuer linken Rand
F360 7E       OUTBYTE: LD A,(HL)     ;Ausgabe der Adresse
F361 CDB8F3    CALL AUSBYTE2         ;zur Umwandlung dann ausdrucken
F364 2B        DEC HL                ;fuer L-Byte von Adresse
F365 10F9      DJNZ OUTBYTE          ;nocheinmal fuer L-Byte
F367 CD8AF3    CALL SPACE            ;Abstand zwischen Adresse und Text
F36A CD8AF3    CALL SPACE            ;    "        "        "
F36D 23        INC HL
F36E A7        AND A                 ;loescht Uebertrag
F36F ED52      SBC HL,DE             ;HL-DE,um 1.Zeich.eines Symb.zu finden
F371 7E       OUTSYM1: LD A,(HL)     ;Ausgabe Symbol
F372 CB7F      BIT 7,A               ;letztes Zeich.v.Symb.=ASCII Wert + 80H
F374 280E      JR Z,OUTSYM2          ;Sprung bei < 80H
F376 CBBF      RES 7,A               ;Abzug (-80H) beim letzten Zeichen
F378 CD9BF3    CALL DRUCKER2         ;Ausgabe Symbol - letztes Zeichen
F37B CD90F3    CALL CRLF2            ;neue Zeile
F37E 23        INC HL                ;bereits ausgegebene Adr.ueberspringen
F37F 23        INC HL                ;    "        "        "
F380 23        INC HL                ;    "        "        "
F381 CD49F3    CALL NEXT             ;naechstes Symbol
F384 CD9BF3   OUTSYM2: CALL DRUCKER2 ;Ausgabe der Symb.(ohne letztes Zeichen)
F387 23        INC HL                ;fuer naechste Ausgabe
F388 18E7      JR OUTSYM1            ;Ende Hauptprogramm
F38A 3E20     SPACE: LD A,20H        ;fuer Abstand
F38C CD9BF3    CALL DRUCKER2
F38F C9        RET

F390 3E0D     CRLF2: LD A,0DH        ;fuer Wagenruecklauf (CR)
F392 CD9BF3    CALL DRUCKER2
F395 3E0A      LD A,0AH              ;fuer neue Zeile    (LF)
F397 CD9BF3    CALL DRUCKER2
F39A C9        RET

F39B C5       DRUCKER2: PUSH BC      ;entspricht CENTRONICS aus LOOP 5
F39C D5        PUSH DE
F39D E5        PUSH HL
F39E 0E49      LD C,49H
F3A0 ED40     DJUMP2: IN B,(C)
F3A2 CB40      BIT 0,B
F3A4 20FA      JR NZ,DJUMP2
F3A6 D348      OUT (48H),A
F3A8 06FF      LD B,0FFH
F3AA ED41      OUT (C),B
F3AC 0600      LD B,00H
F3AE ED41      OUT (C),B
F3B0 06FF      LD B,0FFH
F3B2 ED41      OUT (C),B
F3B4 E1        POP HL
F3B5 D1        POP DE
F3B6 C1        POP BC
F3B7 C9        RET

F3B8 F5       AUSBYTE2: PUSH AF      ;Umwand.fuer Adr.macht aus 1-31H,A-41H usw.
F3B9 F5        PUSH AF               ;linkes Nibble wird nach rechts geschoben
F3BA 1F        RRA
F3BB 1F        RRA
F3BC 1F        RRA
F3BD 1F        RRA
F3BE CDC7F3    CALL AUSDIGIT2
F3C1 F1        POP AF
F3C2 CDC7F3    CALL AUSDIGIT2        ;ohne schieben
F3C5 F1        POP AF
F3C6 C9        RET

F3C7 E60F     AUSDIGIT2: AND 0FH     ;loescht linke Seite
F3C9 C630      ADD A,30H             ;fuer Ziffern + Buchstaben
F3CB FE3A      CP 3AH                ;A minus 3A
F3CD 3802      JR C,ADJUMP2          ;springen bei Ziffern
F3CF C607      ADD A,07H             ;fuer Buchstaben
F3D1 CD9BF3   ADJUMP2: CALL DRUCKER2 ;Ausgabe Drucker
F3D4 C9        RET

F3D5 DD21E5F3 TEXT6: LD IX,INHALT6   ;fuer Erklaerung am Bildschirm
F3D9 212500    LD HL,0025H           ;X-Koordinaten
F3DC 11A000    LD DE,00A0H           ;Y-Koordinaten
F3DF 3E22      LD A,22H              ;Groesse
F3E1 CD1306    CALL TEXTPRINT        ;v.GRUND intern,ermoeglicht mehrere Zeilen
F3E4 C9        RET                   ;im Gegensatz zu WRITE

F3E5 3E3E     INHALT6: LD A,3EH      ;Textanfang
F3E7 3E20      LD A,20H
F3E9 53        LD D,E                ;  >>> SYMBOL-TABELLE ausdrucken <<<
F3EA 59        LD E,C
F3EB 4D        LD C,L                ;        DRUCKER einschalten !!
F3EC 42        LD B,D
F3ED 4F        LD C,A                ;  dann beliebige Taste druecken
F3EE 4C        LD C,H
F3EF 2D        DEC L
F3F0 54        LD D,H
F3F1 41        LD B,C
```

Bild 1: Listing des Programmes "Symboltabelle ausgeben"

```
F3F2 42          ;nur Text fuer Bildschirm
F3F3 45          LD B,D
F3F4 4C          LD B,L
F3F5 4C          LD C,H
F3F6 65          LD B,L
F3F7 2020        JR NZ,0F419H
F3F9 61          LD H,C
F3FA 75          LD (HL),L
F3FB 73          LD (HL),E
F3FC 64          LD H,H
F3FD 72          LD (HL),D
F3FE 75          LD (HL),L
F3FF 63          LD H,E
F400 68          LD L,E
F401 65          LD H,L
F402 6E          LD L,(HL)
F403 203C        JR NZ,0F441H
F405 3C          INC A
F406 3C          INC A
F407 0A          LD A,(BC)
F408 0A          LD A,(BC)
F409 2020        JR NZ,0F42BH
F40B 2020        JR NZ,0F42DH
F40D 2020        JR NZ,0F42FH
F40F 44          LD B,H
F410 52          LD D,D
F411 55          LD D,L
F412 43          LD B,E
F413 48          LD C,E
F414 45          LD B,L
F415 52          LD D,D
F416 2020        JR NZ,0F438H
F418 65          LD H,L
F419 69          LD L,C
F41A 6E          LD L,(HL)
F41B 73          LD (HL),E
F41C 63          LD H,E
F41D 68          LD L,B
F41E 61          LD H,C
F41F 6C          LD L,H
F420 74          LD (HL),H
F421 65          LD H,L
F422 6E          LD L,(HL)
F423 2021        JR NZ,0F444H
F425 210A0A      LD HL,0A0AH
F428 2020        JR NZ,0F44AH
F42A 64          LD H,H
F42B 61          LD H,C
F42C 6E          LD L,(HL)
F42D 6E          LD L,(HL)
F42E 2062        JR NZ,0F492H
F430 65          LD H,L
F431 6C          LD L,H
F432 69          LD L,C
F433 65          LD H,L
F434 62          LD H,D
F435 69          LD L,C
F436 67          LD H,A
F437 65          LD H,L           ;nur Text
F438 2054        JR NZ,0F48EH
F43A 61          LD H,C
F43B 73          LD (HL),E
F43C 74          LD (HL),H
F43D 65          LD H,L
F43E 2064        JR NZ,0F4A4H
F440 72          LD (HL),D
F441 75          LD (HL),L
F442 65          LD H,L
F443 63          LD H,E
F444 68          LD L,E
F445 65          LD H,L
F446 6E          LD L,(HL)
F447 00          NOP              ;Ende

Das Programm >>> SYMBOLTABELLE <<< druckt seine eigenen Symbole !

0613  TEXTPRINT
F333  ST
F338  SLOOP
F346  START
F349  NEXT
F353  ANZAHL
F360  OUTBYTE
F371  OUTSYM1
F384  OUTSYM2
F38A  SPACE
F390  CRLF2
F39B  DRUCKER2
F3A0  DJUMP2
F3B8  AUSBYTE2
F3C7  AUSDIGIT2
F3D1  ADJUMP2
F3D5  TEXT6
F3E5  INHALT6
B1C9  SS
```

```
Hexdump von >>> SYMBOLTABELLE <<< , mit >>> CHECKSUMMEN <<< ausgedruckt:

F333  CD D5 F3 3E 00 CD 27 00 CB 7F 20 07 00 18 F6 00   +=  0646
F343  CD 30 00 21 C9 81 4E 3E 00 69 CA 00 00 11 00 00  +=  0486
F353  7E 13 23 CB 7F 28 F9 06 02 23 CD BA F3 7E CD BB  +=  0797
F363  F3 2B 10 F9 CD B4 F3 CD B4 F3 23 A7 ED 52 7E CB  +=  0A0D
F373  7F 28 0E CB 8F CD 9B F3 CD 90 F3 23 23 23 CD 49  +=  0B69
F383  F3 CD 9B F3 23 18 E7 3E 20 CD 9B F3 C9 3E 0D CD  +=  090A
F393  9B F3 3E 0A CD 9B F3 C9 C5 D5 E5 0E 49 ED 40 CB  +=  09CB
F3A3  40 20 FA D3 48 06 FF ED 41 05 00 ED 41 06 FF E0  +=  07CE
F3B3  41 E1 D1 C1 C9 F5 F5 1F 1F 1F 1F CD C7 F3 F1 CD  +=  0A2B
F3C3  C7 F3 F1 C9 E6 0F C6 30 FE 3A 38 02 C6 07 CD 9B  +=  0906
F3D3  F3 C9 DD 21 E5 F3 21 25 00 11 A0 00 3E 22 CD 13  +=  06C9
F3E3  06 C9 3E 3E 20 53 59 4D 42 4F 4C 20 54 41 42     +=  0483
F3F3  45 4C 4C 45 20 20 61 75 73 64 72 75 63 6B 65 6E  +=  0597
F403  20 3C 3C 3C 0A 0A 20 20 20 20 20 20 44 52 55 43  +=  02D6
F413  48 45 52 20 20 65 69 6E 73 63 68 61 6C 74 65 6E  +=  0580
F423  20 21 21 0A 0A 20 20 64 61 6E 6E 20 62 65 6D 69  +=  0413
F433  65 62 69 67 65 20 54 61 73 74 65 20 64 72 75 65  +=  05E0
F443  63 68 65 6E 00
```

Bild 2: listing und Hexdump des Programmes "Symboltabelle ausgeben

Markus Haberstock

Logikanalyzer: LOG16 gegen LOGAN

Wer sich schon einmal intensiver mit digitalen Schaltungen beschäftigt hat, wird sicher schon einmal den Wunsch gehabt haben, den Signalverlauf sichtbar vor sich zu haben. Dies ist mit einem Logikanalyzer möglich. Denn damit kann man exakt den zeitlichen Verlauf und das Zusammenspiel von digitalen Signalen darstellen. Während bei Oszilloskopen die Kurvenform möglichst genau wiedergegeben werden soll, ist hier nur der Zeitverlauf entscheidend.

Ein Logikanalyzer sollte mindestens 16 Kanäle besitzen, um auch den Adressbus eines 8-Bit Prozessors abbilden zu können. Logikanalyzer mit einem wesentlich höheren Preis (30000 DM und teurer) besitzen in der Regel mehr Kanäle. Von weiterer Bedeutung ist die Abtastrate des Logikanalyzers. Sie sollte deutlich über der des Eingangssignals liegen, damit die Impulse korrekt bewertet werden können.

Dies verdeutlicht Bild 1 .

Die Triggerung eines Logikanalyzers, d.h. wenn er damit beginnt, die Eingangsdaten in den Speicher zu schreiben, ist der dritte wichtige Punkt. Was nützt es einem schließlich, wenn die Messung wegen des vollen Speichers beendet ist, noch bevor der interessante Bereich begonnen hat.

Eine preiswerte Alternative zu den professionellen Logikanalyzern bieten die Logikanalyzer-Baugruppen LOG16 und LOGAN, die hier verglichen werden sollen.

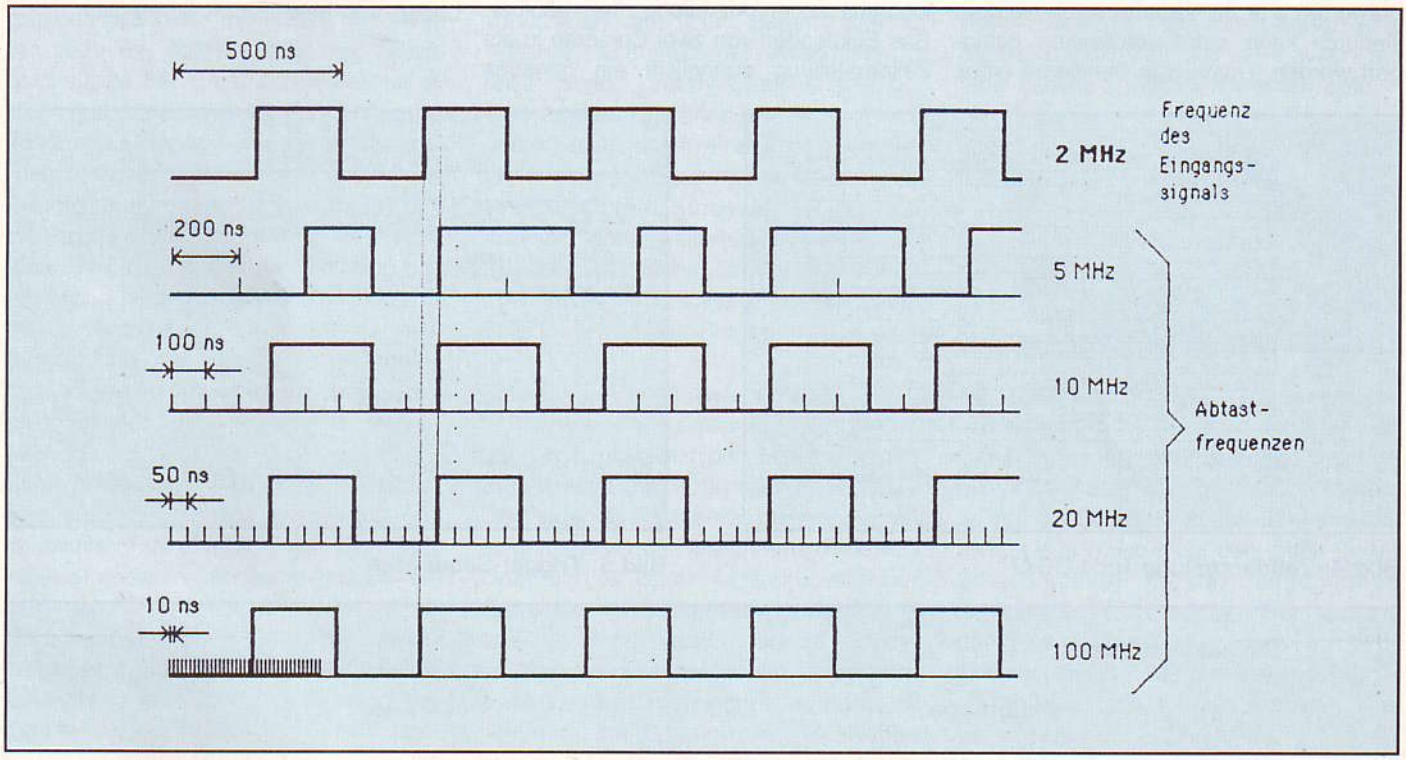


Bild 1: Abtastfrequenz

Die Baugruppe LOG16

Das Menue des Hauptprogrammes enthält eine große Anzahl von Funktionen. Mit Hilfe des Startkommandos werden die Signale in den Meßpuffer eingelesen. Dabei wird entweder intern oder extern getriggert. Bei externer Triggerung kann sowohl durch den Anschluß 'ext. Trigger' als auch durch ein Triggerbyte (Kanalgruppe B) die Messung begonnen werden. Durch Laden der Daten vom Meßpuffer in den Anzeigepuffer ist es möglich, die Messung binär und zeitlich darzustellen. Im Anzeigemodus können durch 25 Befehle komfortable Funktionen ausgeführt werden. Jede Messung kann auf Diskette abgespeichert und jederzeit wieder geladen werden. Da die Baugruppe sowohl einen Meßpuffer als auch einen Anzeigepuffer besitzt, können weitere interessante Abläufe durchgeführt werden. So ist es möglich, eine alte Messung in den Anzeigepuffer zu laden und diese mit der aktuellen Messung im Meßpuffer zu vergleichen. Danach kann die aktuelle Messung wieder auf dem Bildschirm dargestellt werden. Zwei weitere Funktionen der Software sind das Erstellen eines Z80-Disassembler und eines Hexdump- Listings. Andere Unterprogramme ermöglichen, das Bezugslaufwerk zu wechseln und das Disketteinhaltsverzeichnis oder eine genaue Beschreibung der einzelnen Menüfunktionen auf dem Bildschirm aufzulisten. Hochinteressant ist die neue Möglichkeit, zwei Baugruppen zu koppeln und damit 32 Kanäle darzustellen (LOOP 17).

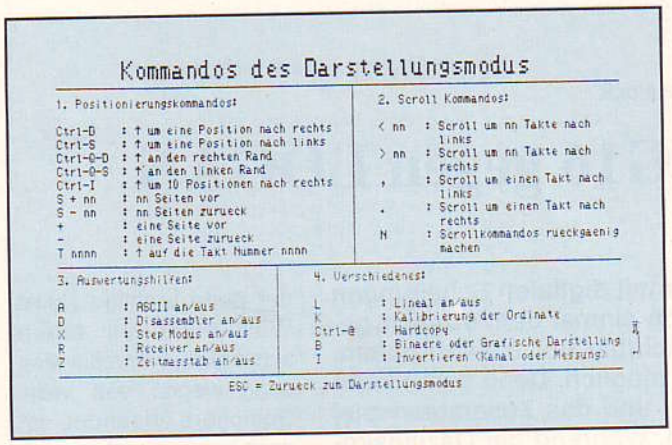


Bild 2: Befehlsübersicht für LOG 16

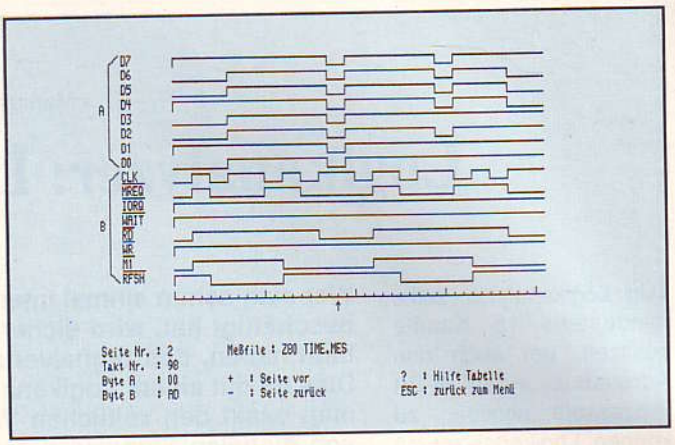


Bild 3: Zeitdarstellung für LOG 16

Die Baugruppe LOGAN

Mit dieser Baugruppe werden die Eingangssignale ebenfalls binär oder zeitlich dargestellt. Die externe Triggerung kann durch zwei Triggerebenen zu je zwei Triggerworten a' 8 Bit vorgenommen werden. Dadurch kann auf Signalflanken getriggert werden. Durch das Definieren eines Pre-Trigger - Parameters ist es möglich, den Signalverlauf auch vor der Triggerung zu untersuchen. Die Frequenz, mit der die Eingangssignale abgetastet werden, ist in logarithmischer Abstufung frei wählbar. Das Einblenden von zwei Cursoren in die Zeitdarstellung ermöglicht ein genaues Abmessen der Dauer der Signale. Weiterhin kann man den Bildausschnitt um einen beliebigen Wert nach rechts oder links verschieben. Das Abspeichern und Laden von Messungen ist möglich.

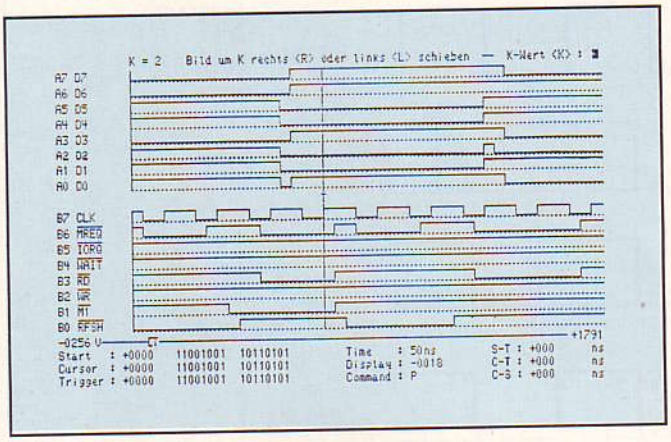


Bild 4: Zeitdarstellung für LOGAN

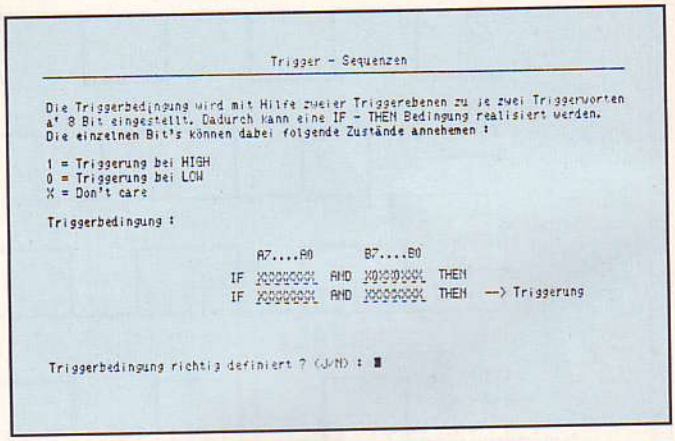


Bild 5: Trigger-Sequenzen

Technische Daten:

Baugruppe	LOG16	LOGAN
Anzahl der Kanäle	16	16
Max. Abtastfrequenz	10 MHz	20MHz
Triggermöglichkeit	intern/extern	intern/extern
Stromaufnahme	0.4 A	1.3 A
Rechnertyp	NDR-Computer	NDR- und mc-Computer

Gegenüber dem LOG16 hat die Baugruppe LOGAN zwei wesentliche Vorteile. Durch die doppelte Abtastfrequenz von 20 MHz lassen sich die Signale genauer darstellen (man vergleiche dazu nur das CLK-Signal der beiden Zeitdarstellungen). Der zweite Punkt ist die Definitionsmöglichkeit der Triggerbedingung. Durch das Einstellen eines jeden einzelnen Bits in zwei Ebenen sind hier keine Grenzen gesetzt. Dagegen sehen die Befehle der Software gegenüber der Fülle der Befehle des LOG16 eher etwas dürftig aus. Denn genau darin liegt die Stärke der Baugruppe LOG16. Doch eines haben beide gemeinsam: Für relativ wenig Geld erhalten die Käufer einen brauchbaren Logikanalyzer.

Folkert Hallenga

680XX - Programme laufen überall

Vielleicht haben sich einige von Ihnen schon einmal gewundert, wie es möglich ist, ein Programm in einem Eprom unterzubringen, das dann irgendwo im Speicher hineingesteckt werden kann und dann sogar läuft.
Eng mit solchen Programmen verbunden tauchen auch Begriffe wie "frei verschieblich" und "relokativ" auf.
Was es damit auf sich hat und welche Möglichkeiten wir dazu haben, soll in diesem Artikel erklärt werden.

Die relative Adressierung

Damit die Überschrift zu diesem Artikel nicht völlig zusammenhanglos erscheint, möchte ich an dieser Stelle einmal den Begriff der relativen Adressierung einführen.
Dies sei an einem Beispiel erklärt:
Nehmen wir einmal an, Sie wollten jemandem erklären, wo die Familie Hoppenstedt wohnt (Name aus Sicherheitsgründen geändert - der Autor). Sie könnten dazu die Adresse angeben: Schlaglochstraße 88. Da Sie sich absolut sicher sind, daß diese Adresse richtig ist, können wir hier von absoluter Adressierung sprechen.
Sie hätten aber auch sagen können, daß die Familie Hoppenstedt in der übernächsten Straße auf der linken Seite und dort im dritten Haus rechts wohnt. Da mit dieser Antwort (bei geeignetem Ausgangspunkt) die Familie relativ genau ausfindig gemacht werden kann, sprechen wir hier von einer relativen Adressierung.
Spaß beiseite: Im ersten Fall haben wir eine absolute Adresse angegeben und im zweiten Fall eine Adresse, die relativ zu einer anderen (der Ausgangsposition) war.
Da uns die Familie Hoppenstedt weniger interessiert, transferieren wir dieses Problem in ein zu schreibendes Programm. Um mir jetzt die Arbeit nicht unnötig schwer zu machen, gehe ich davon aus, daß Sie sich schon ein wenig mit dem 68000/8 Assembler vertraut gemacht haben. Ein Programm mit absoluten Adressen könnte z.B. folgendermaßen aussehen:

```
Start:      * irgendwelche Kommandos
            JMP Adresse
            * irgendwelche Kommandos
Adresse:    * irgendwelche Kommandos
            RTS
```

Wenn wir statt "irgendwelche Kommandos" tatsächlich ausführbare Befehle eingesetzt hätten, könnten wir das Programm übersetzen. Der Assembler ersetzt die symbolischen Befehle und Sprungadressen in eine computergerechte Form und legt sie ab einer bestimmten Adresse (eventuell mit ORG vorwählbar) ab. Den Befehl "JMP - Adresse" übersetzt der Assembler in etwa: "Springe auf die Speicherzelle, die hier Adresse genannt wurde". Im Programm steht somit eine feste Adresse. Wenn dieses Programm nun im Speicher verschoben wird, würde der Sprung nach Adresse aber weiterhin zu der alten Speicherzelle führen, in der inzwischen aber etwas völlig anderes stehen kann. Folglich würde das Programm nicht laufen.
Ersetzen wir den Befehl "JMP" durch "BRA" haben wir unser Problem gelöst, denn nun übersetzt der Assembler den Befehl etwa mit: "Überspringe von hier bis zu der Speicherzelle, die hier Adresse genannt wurde , alle anderen Befehle". Oder um es klarer auszudrücken: Der Assembler berechnet die Sprungdistanz von der momentanen Adresse zu der angegebenen Adresse. Im Programm steht also nicht mehr eine feste Adresse, sondern eine Sprungweite. Wenn nun dieses Programm verschoben wird, bleibt die Sprungweite die gleiche und das Programm würde auch auf der neuen Adresse laufen.
Ein ähnliches Problem taucht bei den Unterprogrammaufrufen mit JSR auf. Beim Übersetzen von Programmtexten, die ein "JSR Unterprogramm" enthalten, schreibt der Assembler wieder feste Speicherzellen als Startadresse für den Unterprogrammaufruf in das Programm. Auch hier heißt die Lösung; Ersetze "JSR" durch "BSR".
Wenn die Sprünge nur sehr kurz sind (max: +/- 128 Byte), dann kann hinter das "BRA" oder "BSR" noch ein ".S" gesetzt werden. Der Assembler benutzt dann eine Kurzform dieser Sprünge, was jeweils 2 Byte spart und somit in der Ausführung auch schneller ist. Der CP/M 68K Assembler generiert auch ohne das ".S" jeweils den optimalen Code.
Ein weiteres Problem taucht auf, wenn über Register auf bestimmte Adressen innerhalb des Programms zugegriffen werden soll. Solche Fälle treten auf, wenn das Programm zum Beispiel eine Meldung ausgeben soll oder wenn auf eine Tabelle zugegriffen werden soll.

```
Start:      * irgenwelche Kommandos
            LEA Text(PC),A0
            BSR Print
            * irgendeine Text-
              ausgaberoutine
            * irgendwelche Kommandos
Text:       DC.B 'Meldung',0
```

Der Befehl "LEA" lädt - wie schon der Name sagt - die effektive Adresse von "Text" in das Adressregister A0. Dazu ermittelt der Assembler die Speicheradresse, auf der die Marke "Text" steht und lädt diesen Wert (also nicht den Inhalt dieser Speicherzelle) in das angegebene Adressregister. Nach der Übersetzung durch den Assembler steht im Programmcode also wieder eine feste Speicherzelle als zu ladende Adresse, was unserer Absicht widerspricht.

Raffinierte Adressierungsarten

Der 6800X Prozessor besitzt einige recht raffinierte Adressierungsarten. Im Rahmen dieses Artikels sollen hier aber nur zwei dieser besonderen Adressierungsarten angesprochen werden: Die Adressierung über Indexregister und die PC relative Adressierung. Wer allerdings viel mit Tabellen oder Feldern arbeitet, dem sei dringend ein gutes Fachbuch empfohlen, da durch geschickte Ausnutzung der enormen Adressierungsmöglichkeiten die Programme erheblich kürzer und somit übersichtlicher und auch schneller werden.
PC (program counter) ist die Bezeich-

nung für das Register, wo der Prozessor seine momentane Programmadresse ablegt. Bei der PC relativen Adressierung ermittelt der Assembler eine Sprungweite von dem momentanen Stand des Programmzählers (PC) bis zu der angegebenen Adresse. Im eigentlichen Programmcode steht also wieder nur eine Sprungweite (die aber nun auf den PC bezogen ist) statt einer festen Adresse. Wird das Programm später auf einer anderen Adresse als ursprünglich gestartet, steht auch der Programmzähler auf einer entsprechend anderen Adresse beim Erreichen des Registerladebefehls. Da die Sprungweite aber beim Verschieben des Programms nicht verändert wurde, wird auch die gewünschte Adresse wieder korrekt geladen. Für das obige Beispiel sähe das Programm dann wie folgt aus:

```
Start:    *irgendwelche Kommandos
          LEA Text(PC),A0
          BSR Print
           * irgendeine Text-
            ausgaberoutine
Text:     DC.B 'Meldung',0
```

Verarbeiten von Variablen

Werden innerhalb eines Programms Variablen benötigt, dann gibt es erneut Probleme. Hier müssen wir aber unterscheiden, ob die Variablen noch innerhalb des frei verschieblichen Programmteils angelegt werden sollen oder außerhalb, wie es bei Programmen im Eprom sogar unumgänglich ist. Können die Variablen noch innerhalb des Programmcodes angelegt werden, ist prinzipiell das gleiche Verfahren wie im vorherigen Beispiel möglich.

```
Start:    * irgendwelche Kommandos
          LEA Variable(PC),A0
          * arbeiten mit der Variablen
          * irgendwelche Kommandos
Variable: DC.L 0
```

Nachdem die Adresse der Variablen auf diese Weise ermittelt wurde, kann auf die Variable über die einfache indirekte Adressierung "(A0)" zugegriffen werden. Bei der einfachsten Möglichkeit, Variablen außerhalb des eigentlichen Programmkörpers einzurichten, wird lediglich mit ORG eine neue Adresse definiert, ab der die Variablen dann abgelegt werden. Da dieser Bereich unabhängig vom eigentlichen Programmkörper ist, kann auf diese Variablen nicht über relative, sondern nur über absolute Adressierungsarten zugegriffen werden, wodurch das Programm selbst aber frei verschieblich bleibt.

```
Start:    * irgendwelche Kommandos
          MOVE.L D0,Variable
          * arbeiten mit der Variablen
          MOVE.L Variable,D0
          * irgendwelche Kommandos
          RTS

ORG       A000
          *neu definierter Variablenbe-
          reich
OFFSET    0
Variable: DC.L 0
```

Diese Methode hat leider den Nachteil, daß die Variablen auf eine bestimmte Stelle im RAM festgelegt sind. Andere Programme dürfen nicht auf diesen Bereich zugreifen, wenn der Inhalt der Variablen nicht zerstört werden soll.

Index - Register

Eine Möglichkeit, bei der die Variablen nicht an einen festen (absolut adressierten) Bereich im RAM des Rechners gebunden sind, kann unter Zuhilfenahme eines Indexregisters realisiert werden.
Ein solches Indexregister ist nun nicht etwa ein besonderes Register im Prozessor, sondern irgend ein beliebiges Adressregister, das für diesen Zweck mißbraucht wird. So benutzt z.B. das Grundprogramm bei einigen Aufrufen das Adressregister A5 als Indexregister, um bei beliebiger Lage des Grundprogramms auf den RAM-Bereich direkt hinter dem Grundprogramm zuzugreifen. Dazu wird irgendwann vorher einmal die Startadresse des RAMs ermittelt und in diesem Adressregister abgelegt. Aufrufe an eine bestimmte Adresse in diesem Bereich werden dann praktisch nur noch "als die soundsovielte Adresse ab der Adresse in A5" angegeben. Etwas ähnliches fanden wir auch schon bei der PC relativen Adressierung, nur wird normalerweise der Wert im Indexregister konstant gehalten und braucht sich nicht unbedingt in Abhängigkeit der Lage des Programms mitzuändern, wenn der angesprochene Variablenbereich sich nicht mitändern soll.
Um die Komplexität dieser Adressierungsform auch richtig hervorzuheben hat sie einen besonders komplizierten Namen: "Address register indirect with displacement". Auf Deutsch heißt das etwa, daß zu dem Inhalt des Adressregisters ein sogenannter Offset dazuaddiert wird, ohne daß der Inhalt des Registers selbst dabei verändert wird. Mit dem so erhaltenen Wert wird dann weitergearbeitet. Der Offset ist dabei auf -32768 und +32767 begrenzt. Für unser Problem können wir diese Adressierungsart etwa so einsetzen:

```
Start:    * definiere (berechne)
          *  Startadresse für
          *  die Variablen und
          * lege sie in A4 ab
           * irgendwelche Kommandos
          MOVE.L D0.Var1(A4)
          * arbeiten mit der Variablen
           MOVE.L Var1(A4),D0
          * irgendwelche Kommandos
          RTS

ORG 0

Var1      DS.L 1    * nur DS, kein DC !!!
Var2:     DS.W 1
Var3:     DS.W 1
```

Mit der Definition der Variablen hinter dem ORG 0 erreichen wir auf eine recht bequeme Art einen sogenannte Adressoffset. Prinzipiell wären die Variablen (oder genauer deren Adressoffsets) auch folgendermaßen definierbar:

```
Var1      EQU  0
Var2      EQU  Var1+4
           * da Var2  Langwortgröße
Var3      EQU  Var3+2
           * da Var2 Wortgröße
```

Werden die Variablen wie im oberen Programmbeispiel definiert, dürfen die Variablen nicht mit "DC" , sondern müssen mit "DS" definiert werden, weil der Assembler beim Befehl "DC" feste Werte an die Stelle schreibt, wobei die Vektortabelle des Prozessors kaputtgeschrieben würde. Beim Befehl "DS" wird lediglich ein Bereich an dieser Stelle reserviert. Im Programm selbst wird dieser Bereich in einen anderen Bereich transferiert (durch das Indexregister angegeben), so daß die Vektortabelle dann unberührt bleibt. Sollen bei Programmstart definierte Werte in den Variablen stehen, müssen sie bei Programmbeginn erst einmal initialisiert werden, indem ihnen ein Wert zugewiesen wird.
Das oben aufgeführte Beispiel läßt sich jetzt noch weiter verfeinern: Mit einer kleinen Abänderung dieses Programms können wir erreichen, daß die Variablen auf dem Stack abgelegt werden. Der

Vorteil ist offensichtlich: Speicherkonflikte mit anderen Programmen können nicht auftreten und der zur Verfügung stehende Speicherraum wird optimal genutzt. Dazu reservieren wir uns gleich als erstes im Programm einen Variablenbereich auf dem Stack, der vor Verlassen des Programms wieder freigegeben wird. Die Adresse, auf der dieser so geschaffene Freiraum beginnt, merken wir uns in einem Adressregister. Wir haben dann praktisch die gleiche Situation wie im Beispiel zuvor.

```
Start:   SUBA.L VarEnd-VarAnf,A7
         MOVEA A7,A5
          * irgendwelche Kommandos
         MOVE.L D0,Var1(A5)
         MOVE.W D1,Var2(A5)
          * arbeiten mit den Variablen
         MOVE.L Var1(A5),D0
         MOVE.W Var2(A5),D1
          * irgendwelche Kommandos
        ADDA.L VarEnd-VarAnf,A7
         RTS

ORG 0
VarAnf:
Var1:    DS,L 1        * nur DS, kein DC !
Var2:    DS.W 1
 .
 .
 .       DS 0          * damit Stack auf
                         Wortbreite
VarEnd:
```

Werden im Programm auch bytegroße Variablen definiert, müssen Sie dafür sorgen, daß der Stack nach der Platzreservierung wieder auf einer geraden Adresse zu stehen kommt, da es sonst zu Adressfehlern kommen kann, wenn der Prozessor auf den Stack zugreifen möchte. Die oben vorgestellte Möglichkeit der Platzreservierung mit den symbolischen Namen erfüllt die Bedingung automatisch, da mit dem Kommando "DS 0" die Marke (Label) "VarEnd" auf eine gerade Adresse gezwungen wird.
Zu guter Letzt wollen wir das Problem noch von einer anderen Seite betrachten. Wir wollen mit unserem Programm ein anderes Programm oder Teile daraus aufrufen, die sich in ihrer Adresslage verändern können.
Wenn wir in unserem Programm die Möglichkeit haben, die Startadresse zu ermitteln, können wir mit Hilfe des indirekten Sprungs diese Adresse anspringen. Beim NDR- Computer durchsucht z.B. das Bibliotheksmenü den Speicherraum nach einer festen Kennung und springt auf Wunsch ein solcherart gefundenes Programm mittels der indirekten Adressierung an. Folgendes Beispiel zeigt das Prinzip:

```
Start:   * berechne Startadresse für
           das(den)
         * Programm(teil) und lege sie
           in A0 ab
         * irgendwelche Kommandos
         JSR   (A0)
         * irgendwelche Kommandos
         RTS
```

Der Trap

Nun wäre es sicher angenehmer, wenn es irgendwo eine feste Speicher"zelle" gäbe, in der jeweils steht, wo dieses Programm oder diese Routine zu finden ist. Allerdings müßte immer gewährleistet sein, daß dieses Programm seine jeweilige Lage der Speicherzelle mitteilt. Solche festen Speicherzellen gibt es sogar und außerdem auch gleich ein Kommando, das uns die Sache (Arbeit) damit erleichtert - der TRAP. Und da insgesamt 16 derartige Speicher"zellen" für solche Fälle vorhanden sind, gibt es 16 TRAPs (TRAP 0 .. TRAP15). Trifft der Prozessor auf einen Befehl "TRAP", springt er auf die dem jeweiligen TRAP zugeordnete Adresse in seiner Vektortabelle. Dort steht ein Sprung zu der Adresse auf der dann das eigentliche Programm steht. Um den Sinn solcher TRAPs zu verstehen, betrachten wir einmal unser Grundprogramm mit den insgesamt 120 Unterprogrammen, die für den User aufrufbar sind, genauer. Wird das Grundprogramm verschoben, verschieben sich sämtliche 120 Einsprungadressen für die Unterprogramme ebenfalls. Das Problem läßt sich lösen, indem ein Programm geschrieben wird, das in einem bestimmten Register einen Kennwert für das aufzurufende Unterprogramm erwartet und in Abhängigkeit von diesem Kennwert einen relativen Sprung zu dem entsprechenden Unterprogramm durchführt. Wenn dieses Programm durch einen TRAP aufgerufen werden kann, muß beim Verschieben des Gesamtprogramms nur noch in der Vektortabelle die Einsprungadresse zu dem Programm geändert werden. In unserem Grundprogramm gibt es auch so ein Programm, das abhängig vom Wert im Datenregister D7 eines der 120 Unterprogramme relativ anspringt. Beim Einschalten des Rechners ermittelt das Grundprogramm die richtige Einsprungadresse für dieses Programm und trägt in der Vektortabelle für den TRAP 1 einen entsprechenden Sprung dorthin ein.
Als Beispiel soll hier einmal die Routine "SCHREITE" herhalten. Wollen wir diese Routine von einem Programm aus aufrufen, könnte dies z.B. so aussehen:

```
Start:   * irgendwelche Kommandos
          MOVE #100,DO
         JSR $SCHREITE
          * irgendwelche Kommandos
          RTS
```

Wenn hier wieder sinnvolle Befehle für "irgendwelche Kommandos" eingesetzt werden, können wir das Programm mit dem im Grundprogramm eingebundenen Assembler übersetzen. Dieser ermittelt für "$SCHREITE" die Adresse im Grundprogramm, wo das Unterprogramm SCHREITE beginnt und setzt diese Adresse in den Programmcode ein. Es steht somit wieder eine feste (absolute) Adresse im Programmcode. Wird das Grundprogramm nun aber verschoben oder stimmen bei einer anderen Grundprogrammversion die Einsprungadressen nicht mit der Version überein, mit der das Programm übersetzt wurde, stürzt dieses Programm ab.
Benutzen wir dagegen den TRAP 1 des Grundprogramms, so sieht das Programm dann so aus:

```
Start:   * irgendwelche Kommandos
          MOVE #100,D0
          MOVE #!SCHREITE, D7
           TRAP #1
         * irgendwelche Kommandos
         RTS
```

Beim Übersetzen dieses Programms mit dem Assembler des Grundprogramms ermittelt dieser nur die Nummer des Unterprogramms und trägt sie in das Datenregister D7 ein. Danach erfolgt ein TRAP 1 Aufruf, der unabhängig von der Lage des Grundprogramms das Unterprogramm startet, dessen Nummer in D7 steht. Somit erfolgt praktisch wieder nur ein relativer Sprung zu dem Unterprogramm und wir haben unseren Willen bekommen.
Ich möchte doch unbedingt alle Assemblerprogrammierer bitten, diese Möglichkeit auch zu nutzen. Es ist traurig, immer wieder in der LOOP zu lesen, daß bei der und der Rechnerkonfiguration dieses oder jenes Programm nicht mehr läuft. Da es im Grundprogramm sogar eine feste Speicherzelle (bei $414) gibt, auf der eingetragen ist, für welchen Prozes-

sor es geschrieben ist, ist es prinzipiell sogar möglich, auch bei direkten Hardwarezugriffen Programme zu schreiben, die sowohl mit dem 68008 als auch mit dem 68000 laufen. Beim Zugriff auf eine feste IO Adresse muß dann dieser Wert mit dem Wert in der Speicherzelle multipliziert werden. Als Beispiel sei die Abfrage des Tastaturports hier einmal aufgeführt:

```
Tast    EQU  $FF68

Start:  MOVE    #!GETBASIS,D7
        TRAP    #1
       * in A0    Grundprogrammstart
        MOVE.W  $416(A0),D0
       * dort steht CPU Größe/laden
        MULS    #TAST,D0
       * mit Tastaturport multiplizieren
        MOVEA.L DO,A0
       * für Abfrage in Adressregister
        MOVE.B  (A0),D0
       * in D0.B steht Tastaturwert
       * weitere Kommandos
        RTS
```

Dem Symbol "Tast" wurde hier nur eine Wortgröße zugeordnet, aber das reicht hier vollkommen aus, da wir bei der Multiplikation mit dem Befehl MULS eine automatische Vorzeichenerweiterung erhalten, die dafür sorgt, daß die obere Hälfte des Datenregisters mit $FFFF vollgeschrieben wird.

Zu guter Letzt noch eine Bemerkung für diejenigen unter Euch, die relativ (um beim Thema zu bleiben) unabhängig vom Assembler (z.B. CP/M Assembler) programmieren wollen. Der CP/M Assembler kennt keine Ausdrücke wie z.B. !SCHREITE oder $SCHREITE. Sollen die Programme aber mit Assembler übersetzbar sein, empfiehlt sich eine universellere Schreibweise, die sowohl auf dem RDK Assembler als auch auf dem CP/M Assembler lauffähig ist:

```
SCHREITE  EQU  1

START:   * irgendwelche Kommandos
         MOVEQ  #100,D0
         MOVEQ  #SCHREITE, D7
         TRAP   #1
         * irgendwelche Kommandos
         RTS
```

Die Nummern der jeweiligen Programme stehen vorne im Buch zum Grundprogramm aufgeführt. Es ist jetzt auch eine Kurzform des Befehls (MOVEQ) möglich, die vorher - wohl durch einen Fehler im RDK Assembler - nicht möglich war.

Leider sind Programme mit Grundprogrammaufrufen unter CP/M 68K nicht so ohne Weiteres lauffähig, da dort der TRAP 1 nicht initialisiert wird. Es ist allerdings möglich, mittels einer kleinen Routine die Adresse des TRAP-Handlers im Grundprogramm zu ermitteln und diese dann in die für den TRAP 1 reservierte Speicherzelle zu laden. Beim Elektronikladen in Detmold gibt es auch eine Utilitydiskette mit einer Reihe von Hilfsprogrammen zum CP/M 68K, wo unter anderem auch ein Programm mit darauf ist, welches nach einmaligem Aufruf bis zum Abschalten des Rechners den TRAP 1 wieder initialisiert. Auf dieser Diskette befinden sich außerdem noch Programme zum Laden und Abspeichern von Daten, Programme zum Laden und Abspeichern von Texten auf beliebige Adressen, wobei beim Laden Tabulatorzeichen expandiert und Steuerzeichen gelöscht werden (können sonst zum Absturz des RDK Editors führen). Beim Abspeichern werden überflüssige Leerzeichen am Zeilenende gelöscht. Im Editoraufruf unter CP/M sind diese Verbesserungen ebenfalls eingebaut, zusätzlich zum Anlegen einer Sicherungsdatei (.BAK Datei) und der Möglichkeit einen Text zu verwerfen (nicht abzuspeichern). Des Weiteren gibt es auf dieser Diskette Programme zum Aufruf des Grundprogramms mit definierter Rücksprungmöglichkeit und ein Programm zum Starten von Programmen mit Bibliothekseintrag (z.B. PASCAL). Eine umfassende Dokumentation der Programme und der sich daraus ergebenden Möglichkeiten ist ebenfalls vorhanden.

Kleiner Test zur Lernkontrolle:
Wenn Sie jemandem erzählen, daß die Familie Hoppenstedt in der 3. Straße hinter der Hubertusgasse wohnt, welche Form der Adressierung war das dann?
Indirekte Adressierung mit Offset - im Indexregister steht "Hubertusgasse" und der Adressoffset lautet "3 Straßen weiter".
Und wenn Sie nun zum Einwohnermeldeamt gelaufen wären und sich erkundigt hätten, weil die Familie nun mal so oft umzieht?...

Folkert Hallenga
Hufellandstr.16
3000 Hannover 91
Tel.:(0511) 2133239

Merktafel für 680xx - Programmierer

Grundsätzlich: Nur TRAP - Befehle verwenden

Feststellen, welche CPU im System steckt::

Dies kann dadurch geschehen, indem man die Adresse Grundprogrammstart + $00000414 abfragt

Auswertung:

00000001 ---> CPU 68008
00000002 ---> CPU 68000
00000004 ---> CPU 68020

Tyko Strassen

NDR - Computer <-----> Großrechner

Lochstreifenleser selbstgebaut

Ein großes Problem für Rechner unter CP/M 68K ist, daß kaum Software verfügbar ist. Es gibt sicher viele Hobby-Programmierer, die versuchen, von anderen Rechnern die Software zu kopieren und dann, wenn nötig, dem NDR-Computer anzupassen. Zum Beispiel gibt es gerade auf UNIX-Systemen viele interessante C-Programme, die man im Prinzip direkt übernehmen könnte.

Als gemeinsames Übertragungsmedium zu einem anderen Rechnersystem gibt es prinzipiell mehrere Möglichkeiten. Die wohl am häufigsten verwendete Methode ist die Übertragung mit einem Magnetband. Allerdings kommt diese für Hobby-Benutzer nicht in Frage, weil eine Magnetbandstation für die meisten viel zu teuer ist.
Eine Übertragung via Telephon mit einem Modem ist auch ziemlich gebräuchlich. Dafür habe ich zwei sehr nützliche Programme entwickelt, für die man allerdings auch erst einmal tief in die Tasche greifen muß. Denn ein solches Telephonmodem darf man nicht selber bauen, sondern es muß von der Post gekauft werden.
Weiter werden Daten auch via Lochkarten übertragen. Diese Form nützt uns aber auch nicht viel.
Zuletzt bleibt noch das Übertragungsmedium "Papierlochstreifen", das zwar nicht gerade modern ist (die neuen Großrechner haben meistens keinen Lochstreifenstanzer mehr), aber den großen Vorteil hat, daß man sich mit geringem Aufwand an Material und Meßgeräten leicht einen eigenen Lochstreifenleser bauen kann. Dies möchte ich, natürlich mit der zugehörigen Software, ebenfalls beschreiben.

Papierlochstreifenleser

Technische Daten:
- Liest 8-Bit Papierlochstreifen
- Optische Abtastung. Das Papierband wird von Hand mit max. 2m/sec durch gezogen.
- Keine beweglichen Teile.
- RS-232 Ausgang.
  Das Gerät läßt sich also auch problemlos an einen beliebigen anderen Rechner anschließen, der über eine serielle Schnittstelle verfügt.
- Übertragungsformat: 9600 Baud, 8 Bit, 1 Stop, keine Parität. (Das Bit 7 des Lochstreifens hat normalerweise die Bedeutung einer geraden Parität.)
- Stromversorgung: +5V, -12V oder -5V (AY-5-1013: -12V notwendig!)
- Zum Aufbau und Abgleich ist nur ein Voltmeter nötig.
- Kostengünstig: Bauteile ca. 60.-DM

Dieser Papierlochstreifenleser funktioniert nach dem optischen Abtastungsprinzip. Auf der einen Seite des Streifens strahlen Leuchtdioden (aus Kostengründen vorzugsweise Infrarot) und auf der anderen Seite registrieren dann (Infrarot)- Photodioden, ob Löcher vorhanden sind. Diese Signale werden durch Schmitt-Trigger sauber aufgearbeitet und man erhält die acht parallelen Datenbits und den Strobe.
In dieser Form könnte man die Daten schon zum Tastatureingang (auch parallel) des NDR-Rechners führen. Sie werden jedoch durch einen parallel-seriell-Wandler (UART), der seinen Takt von einem 555-Oszillator erhält, in ein serielles Signal umgewandelt. Das serielle TTL-Signal wird dann anschließend durch einen Treiber auf den RS-232-Stecker geführt (25-polig).
Zur Schaltung:
Bei den Photo- und Leuchtdioden kann man entweder im sichtbaren oder im infraroten Spektrum arbeiten. Infrarot ist zu empfehlen, weil die Bauteile dann billiger werden. Probleme, weil man das Licht nicht sieht, wird es kaum geben, da das optische System in der Art völlig unkritisch ist.
Als Sender nimmt man 4 handelsübliche 5mm-Leuchtdioden, die jeweils vom Widerstand R' im Strom begrenzt werden. R' wird mit der Formel U=R*I berechnet. Für I nimmt man den maximal zulässigen Leuchtdiodenstrom. Man kann R' ruhig etwas höher wählen, denn es ist wirklich alles völlig unkritisch.

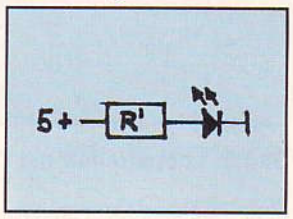


Bild 2: Leuchtdiode

(Für die Photodiode muß man einen Typ nehmen, den man im 2,45mm-(1/10")-Raster nebeneinander setzen kann, weil die Löcher im Lochstreifen diesen Abstand haben. Der große Vorteil dieses Rasters ist außerdem, daß zur Lochstreifenführung und für die "Optik" eine Standard-Lochrasterplatine verwendet werden kann.
Die Leselöcher, hinter denen die Photodioden liegen , werden auf ca. 1.8mm aufgebohrt, damit sie so groß sind wie die des Lochstreifens. Nur das Loch für den Strobe bleibt so klein wie es ist, damit dieser beim Lesen erst aktiviert wird, wenn die Daten schon stabil sind.

Die Widerstände R der Photodioden müssen so bemessen sein, daß der nachfolgende Schmitt-Trigger klar schalten kann.
Das heißt, bei einem Loch muß die Spannung unter 0.5V, bei den anderen über 1.9V liegen. Als Richtwert für R sei hier 1kOhm genannt. Der genaue Wert muß aber unter Berücksichtigung des

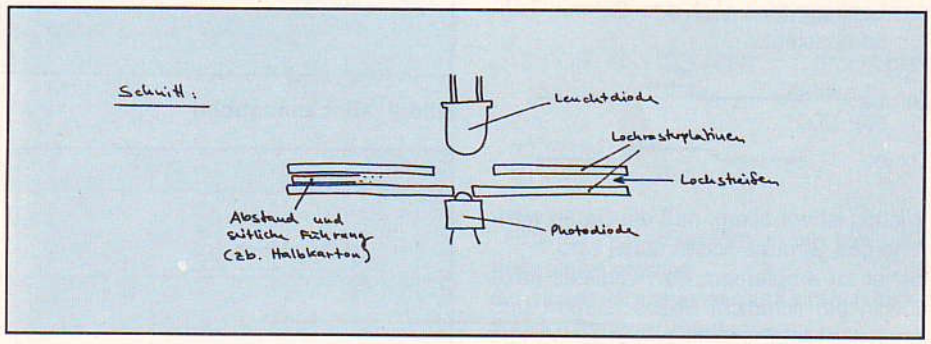


Bild 1: Anbringen der Dioden

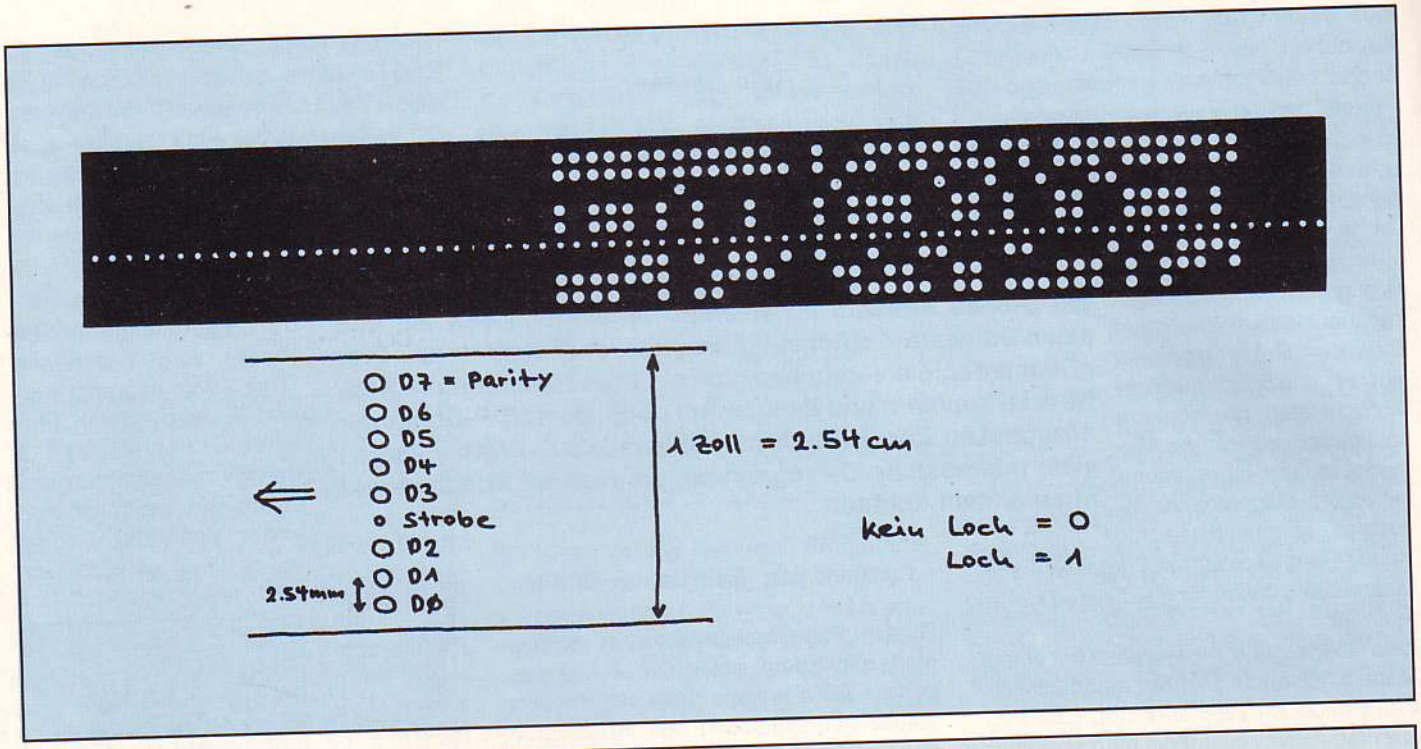


Bild 3: Lochstreifen mit Codierung

max. zulässigen Diodenstroms experimentell ermittelt werden. Er ist auch Bei der Photodiode für den Strobe muß man etwas genauer sein. Deshalb wird hier ein Trimmpoti mit ca. R"= 5*R verwendet. Der Trimmer wird so eingestellt, daß das Zeitdiagramm beim Durchziehen eines Lochstreifens nach dem Schmitt-Trigger so aussieht:

Strobe:

Dx je: 

Wichtig ist vor allem, daß die Daten während des Strobes schon stabil sind.
Es ist zu empfehlen, den Optikteil nach außen hin lichtdicht abzuschließen und innen alle Metallteile mit einem schwarzen Filzstift anzumalen, um Reflektionen zu vermeiden.

Es gibt zwei Typen UARTs (AY-5-1013, AY-3-1015), von denen einer (AY-5-1013) eine Versorgungsspannung von -12V benötigt. Sonst sind beide identisch. Hier wird nur ein Teil des UARTs benötigt, deshalb sind viele Pins nicht belegt. Das Datenformat kann anhand der Liste leicht abgeändert werden. (Siehe Bild 8)

Zur Takterzeugung wird ein einfacher 555 verwendet, der mit der 16-fachen Baudrate, also mit 153.6kHz schwingt. Man muß nur darauf achten, daß der Kondensator (3900pF) einigermaßen temperaturstabil ist.
Vor dem Frequenzabgleich wird mit dem

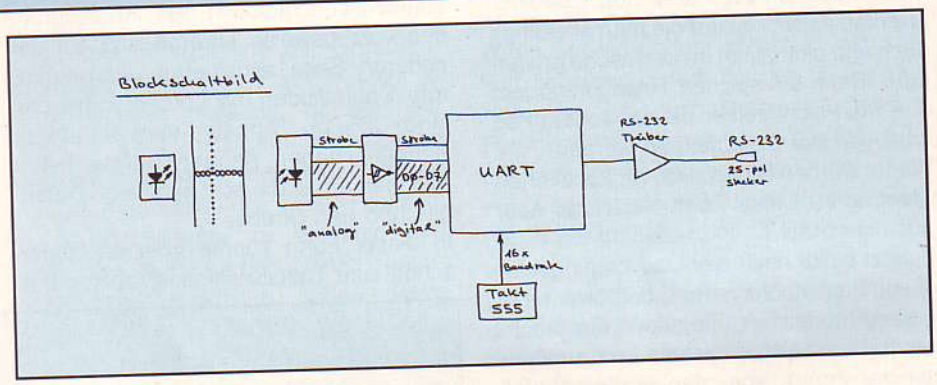
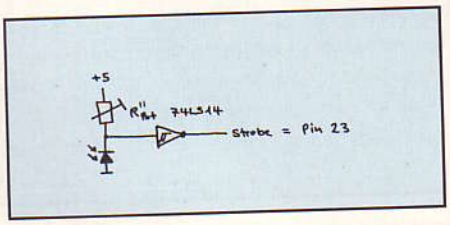


Bild 4: Blockschaltbild



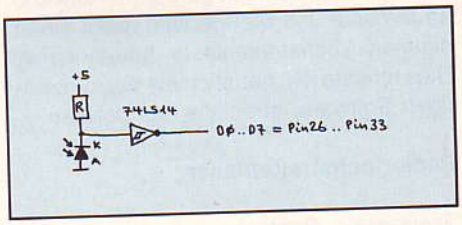


Bild 5 und 6: Photodioden

Trimmer ausprobiert, in welchem Bereich der Rechner noch alle Zeichen korrekt empfangen kann. Dann stellt man den Trimmer einfach in die Mitte dieses Bereiches.
Beim Treiber (Typ 1489) kann man als negative Betriebsspannung wahlweise -12V oder -5V nehmen.
Die Steckerbelegung ist angegeben.
Das zugehörige Einleseprogramm ist aufgelistet.

Nun also zur Software zum Übertragen von Daten mit Telephonmodem und Papierlochstreifenleser via serielle Schnittstelle:

Hierfür ist sicher das C-Programm <siinit.c>, mit dem man die serielle Schnittstelle initialisieren kann, nützlich. Man hat die Möglichkeit, die Baudrate, die Anzahl der Daten und Stopbits, normalen oder Echo-Modus und das Paritätsbit einstellen.
Das Programm demonstriert auch, wie man direkt in C, ohne Assemblerteil, abhängig von der CPU I/O-Adressen setzen kann. I
m Prinzip lassen sich mit diesem Programm und dem Dienstprogramm <pip> alle angesprochenen Übertragungsprobleme lösen.

Hierzu Beispiele:

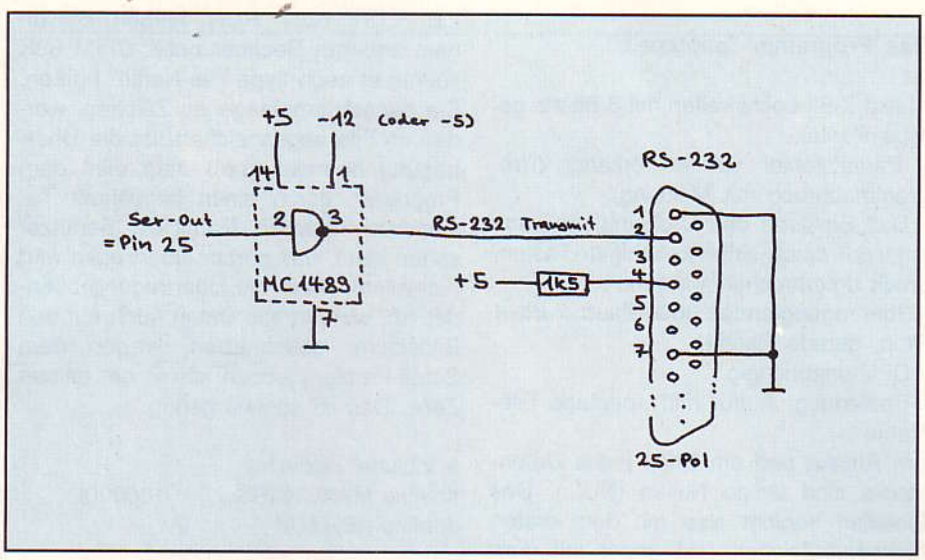


Bild 7: Steckerbelegung des AY-5-1013

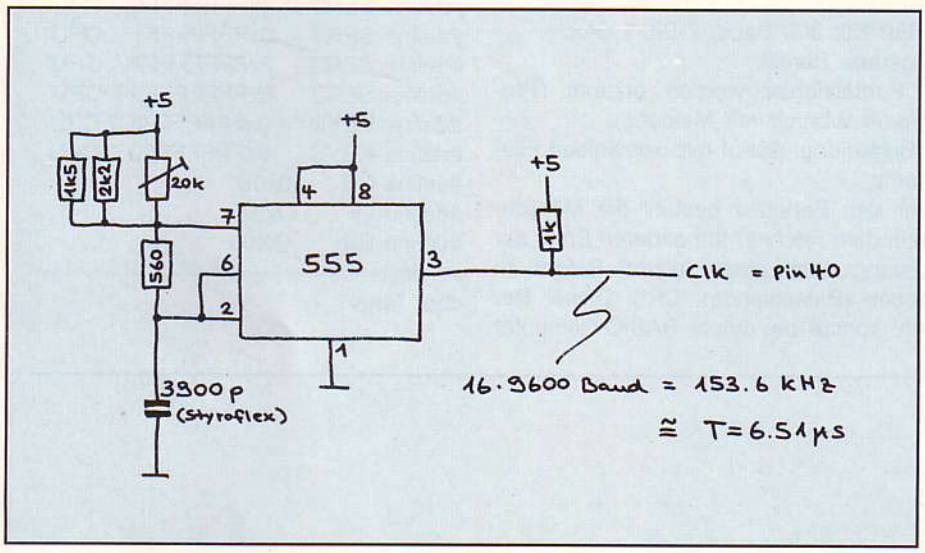


Bild 8: Takterzeugung

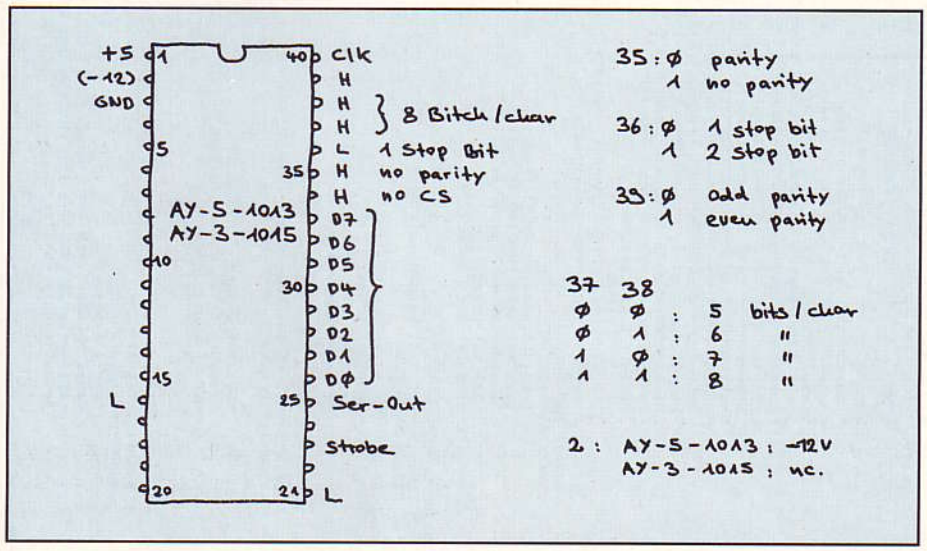


Bild 9: Serieller Treiber

1. Allgemeine Erklärung des <pip>-Befehls:

PIP <Ziel> := <Quelle>

Für Ziel:
- CON: Bildschirm
- AXO: Serieller Ausgang
- File-Name: File

Für Quelle:
- CON: Tastatur
- AXI: Serieller Eingang
- File-Name: File

2. Übertragung eines Files vom NDR-Computer auf einen UNIX-Rechner:

```
PIP AXO:= CON:
cat >UnixFile CCR     ;Zeichen in Unix
PIP AXO:= CON
CTRL Z
PIP AXO:= NDRFILE
PIP AXO: CON
CTRL D
CTRL Z                ; Ende der Übertragung
usw.
```

3. Basicprogramm

Am Seriellen Eingang ist ein BASIC-Einplatinenrechner angeschlossen und auf dem NDR-Rechner ist folgendes kleines Programm unter <BASPROG> abgespeichert:

```
10 FOR  N=1 TO 1000     ; druckt die
                          Zahlen 1-
                          1000 aus.
20 PRINT N
30 NEXT N
```

RUN <— Das Programm soll direkt ausgeführt werden.
So wird das Programm ganz einfach in den BASIC-Rechner eingelesen:

```
>PIP AXO: <- BASPROG
```

4. Wenn man den Output des Programms gleich danach auf dem Bildschirm sehen will:

```
>PIP CON: <- AXI
```

Hier treten zwei Probleme auf:
Einmal sind natürlich die ersten Zahlen schon vorbei und außerdem gehen teilweise Zeichen verloren, weil der NDR-Rechner beim Scrollen sehr langsam ist.

5. Die empfangenen Zeichen sollen nicht auf dem Bildschirm erscheinen, sondern in einem File abgespeichert werden:

```
>PIP OUTFILE <- AXI:
```

Die Probleme hier sind ähnlich. Der NDR-Computer braucht ab und zu Zeit, um die schon erhaltenen Daten (blockweise) im File abzuspeichern und dabei gehen einige Zeichen verloren.

Betrachtet man die Beispiele, wird deutlich, daß das Hauptproblem darin besteht, daß der NDR-Rechner im Gegensatz zu den meisten Großrechnern keine Datenpufferung besitzt.
Während ein Großrechner in einer beliebigen Geschwindigkeit Daten schicken kann (UPLOAD), die er alle registriert, übersieht der NDR-Computer in der umgekehrten Richtung (DOWNLOAD) einige Zeichen, weil er eben keinen schnellen Zwischenspeicher hat.

Genau diesen Zwischenspeicher simulieren die folgenden beiden Programme <download.c> und <paptape.c> (letzteres speziell zum Einlesen von Papierlochstreifen = papertape). Beide Programme sind in C geschrieben. Als Zwischenspeicher dient ein genügend großer ARRAY. Die empfangenen Daten werden bei beiden Programmen erst dann im File abgespeichert, wenn die eigentliche Übertragung beendet ist.

Zu den Programmen:

Das Programm "paptape.c"

- Liest 7-Bit Lochstreifen mit 8.Bit als gerade Parität.
- Paritätsfehler werden erkannt (Programmabbruch mit Meldung).
- Das Einlesen des Lochstreifens kann jederzeit durch einen beliebigen Tastendruck unterbrochen werden.
- Übertragungformat: 9600 Baud, 7 Bit, 1 Stop, gerade Parität.
- CPU-unabhängig.
- Bedienung: Aufruf mit: >paptape File-Name

Am Anfang und am Ende jedes Datenblocks sind einige Nullen (NUL:). Das Einlesen beginnt also mit dem ersten Nicht-Null-Zeichen und endet mit dem letzten Null-Zeichen.

Das Programm "download.c"

- Format: 300 Baud, 7 Bit, 1 Stop
, gerade Parität.
- Paritätsfehler werden erkannt (Programmabbruch mit Meldung).
- Bedienung: Aufruf mit: >download Filename

Für den Benutzer besteht die Möglichkeit, dem Rechner am anderen Ende der Leitung noch einen letzten Befehl zu geben (Befehlsende: "CR"). Dieser Befehl könnte bei einem BASIC-Computer z.B. "LIST" oder "RUN" heißen. Bei einem anderen Rechner unter CP/M 68K könnte er auch "type File-Name" heißen. Die danach empfangenen Zeichen, werden im File abgespeichert. Ist die Übertragung beendet, muß man dies dem Programm durch einen beliebigen Tastendruck mitteilen. Damit der Benutzer sehen kann, was gerade übertragen wird (vor allem, wann die Übertragung beendet ist), werden alle Daten auch auf den Bildschirm geschrieben, wegen dem Scroll-Problem jedoch alle in der selben Zeile. Das ist schnell genug.

```
# include <stdio.h>
#define MAXCHARS        100000L
#define ABHACK          2

BOOLEAN  ParityError;
int CPU;
/* 68008:1 68000:2 68020:4 */
#define SER0     0xFFFFFFF0 * CPU
#define SER1     0xFFFFFFF1 * CPU
#define SER2     0xFFFFFFF2 * CPU
#define SER3     0xFFFFFFF3 * CPU
#define KEYD     0xFFFFFF68 * CPU
#define KEYS     0xFFFFFF69 * CPU
#define CR       0x0D
#define LF       Ox0A
#define BS       OX08
int argc;
char *argv
```

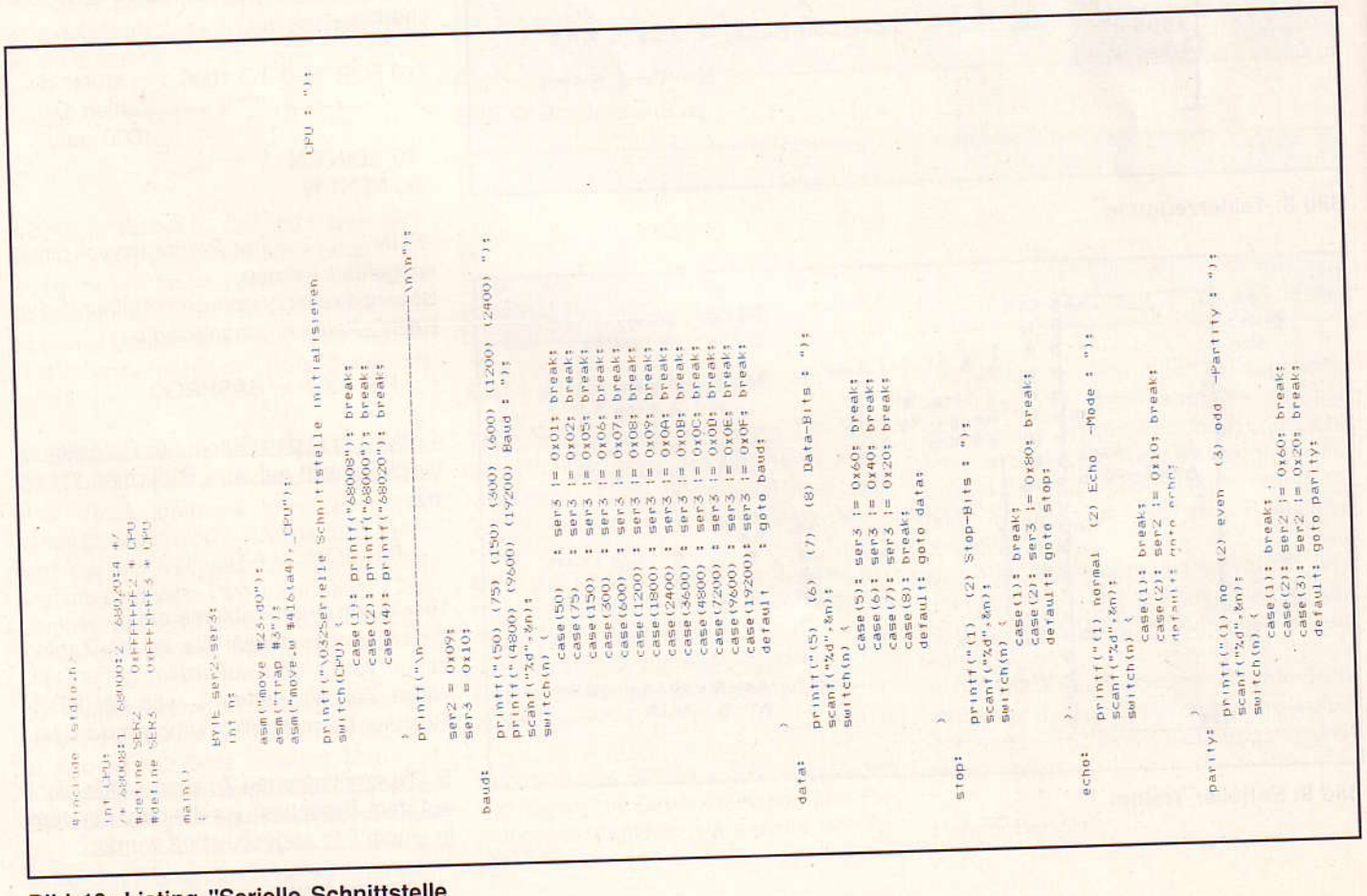


Bild 10: Listing "Serielle Schnittstelle

```
}
printf("\nSetze Adresse $%lx auf $%x",SER2,ser2 & 0xFF);
printf("\nSetze Adresse $%lx auf $%x\n\n",SER3,ser3 & 0xFF);
*((char *) SER2) = ser2;
*((char *) SER3) = ser3;

#include <stdio.h>

#define MAXCHARS        10000L
#define DUMMY           4

BOOLEAN ParityCheck;
int CPU;
/* 68008:1  68000:2  68020:4 */
#define SER0    0xFFFFFFF0 * CPU
#define SER1    0xFFFFFFF1 * CPU
#define SER2    0xFFFFFFF2 * CPU
#define SER3    0xFFFFFFF3 * CPU

#define KEYU    0xFFFFFF68 * CPU
#define KEYS    0xFFFFFF69 * CPU


main(argc,argv)
int argc;
char *argv[];
{
        FILE *fopen(),*fp;
        static char text[MAXCHARS],c;
        int n;
        long anz_chr,pos;

        if (argc != 2) {
                fprintf(stderr,"Usage:  paptape filename.ext\n");
                exit(1);
        }

        printf("\033P A P E R T A P E  -  R E A D E R    ");
        printf("                    Jan\'88  T.Strassen");
        printf("\nFormat : 9600 Baud  8 Bit  1 Stop  No Parity");
        printf("\nInterner Speicherplatz (BYTES) : %ld",MAXCHARS);
        printf("\n-----------------------------------------");
        printf("-----------------------------------------\n\n");

        if ( (fp = fopen(argv[1],"w")) == NULL ) {
                fprintf(stderr,"Cannot create %s\n",argv[1]);
                exit(1);
        }

        init();

        printf("    Lochstreifen einlesen !\n\n");

        /* Erst den Anfangs-Muell einlesen (Dummy) */
        ParityCheck = FALSE;
        for (n=0;n<DUMMY;n++) si();
        ParityCheck = TRUE;

        /* Zero's einlesen */
        do c = si(); while (c == 0);

        /* Tape einlesen */
        anz_chr = 0L;
        while (c != 0) {
                if (anz_chr == MAXCHARS) {
                        printf("\nMaximale Seicherfaehigkeit erreicht !!!\n");
                        break;
                }
                text[anz_chr] = c;
                ++anz_chr;
                c = si();
        }
        printf("\n%ld Zeichen eingelesen (max %ld).\n",anz_chr,MAXCHARS);

        for (pos = 0L;pos < anz_chr;pos++)
                putc(text[pos],fp);
        fclose(fp);
        printf("\nDaten im File <%s> abgespeichert.\n",argv[1]);
}
```

```
VOID
init()  /* Initialisiert Variable <CPU> und serielle Schnittstelle */
{
        asm("move #23,d0");
        asm("trap #3");
        asm("move.w $416(a4),_CPU");
        *((char *) SER3) = 0x3E;  /* %00111110 : 7 Bit  1 Stop  9600 Baud */
        *((char *) SER2) = 0x69;  /* %01101001 : even Parity, no Echo     */
}


BOOLEAN
si_empt()  /* Wurde ein Zeichen empfangen ? */
{
        register char c;
        c = *((char *) SER1);
        return( (c & 0x08) != 0x00 );
}


int
si()  /* Liest ein Zeichen von serieller Schnittstelle ein */
{
        while (!si_empt()) if (keypressed()) exit(0);
        if (((*((char *) SER1) & 0x01) != 0) && ParityCheck) {
                printf("\n\n*** Parity-Fehler\n");
                exit(1);
        }
        return(*((char *) SER0) & 0x7F);
}


BOOLEAN
keypressed()  /* Wurde eine Taste gedrueckt ? */
{
        register char d;
        d = *((char *) KEYU);
        if ((d & 0x80) == 0x00) {
                if (*((char *) KEYS)) {
                return(TRUE);
        return(FALSE);
```

Bild 11: Listing "Papertape - Reader"

```c
#include <stdio.h>

#define MAXCHARS        100000L
#define ABHACK          2

BOOLEAN ParityError;
int CPU;
/* 68000:1 68000:2 68020:4 */
#define SER0    0xFFFFFFF0 * CPU
#define SER1    0xFFFFFFF1 * CPU
#define SER2    0xFFFFFFF2 * CPU
#define SER3    0xFFFFFFF3 * CPU

#define KEYD    0xFFFFFF68 * CPU
#define KEYS    0xFFFFFF69 * CPU

#define CR      0x0D
#define LF      0x0A
#define BS      0x08


main(argc,argv)
int argc;
char *argv[];
{
        FILE *fopen(),*fp;
        static char text[MAXCHARS];
        char c;
        long anz_chr,pos;

        if (argc != 2) {
                fprintf(stderr,"Usage:  download filename.ext\n");
                exit(1);
        }

        printf("\032D O W N L O A D       HOST -> CP/M 68K");
        printf("                  Jan\'88  T.Strassen");
        printf("\nFormat :  300 Baud  / Bit  1 Stop  Even Parity");
        printf("\ninterner Speicherplatz (BYTES) : %ld",MAXCHARS);
        printf("\n---------------------------------------------");
        printf("---------------------------------------------\n\n");

        if ( (fp = fopen(argv[1],"w")) == NULL ) {
                fprintf(stderr,"Cannot create %s\n",argv[1]);
                exit(1);
        }

        init();

        printf("Letzter Befehl an HOST : ");
        c = getchar();
        while ((c != CR) & (c != LF)) {
                so(c);
                c = getchar();
        }
        so(CR); so(LF);
        printf("\nKommando ist eingegeben.\n");
        do c = si(); while ((c != CR) & (c != LF));

        printf("\nUebertragung beginnt ...\n\n");
        anz_chr = 0L;
        while (!keypressed()) {
                if (!si_empf()) continue;
                c = si();
                if (anz_chr == MAXCHARS) {
                        printf("\n\nMaximale Seichertaehigkeit erreicht !\n");
                        break;
                }
                text[anz_chr] = c;
                ++anz_chr;
                wrt_c(c);
        }
        printf("\n\n%ld Zeichen eingelesen (max %ld).\n",anz_chr,MAXCHARS);
        if (ParityError) printf("\n*** Parity - Fehler ***\n");

        anz_chr -= ABHACK;
        for (pos = 0L;pos < anz_chr;pos++)
                putc(text[pos],fp);
        fclose(fp);
        printf("\nDaten im File <%s> abgespeichert.\n\n",argv[1]);

VOID
wrt_c(c)   /* Schnelle Zeichenausgabe: kein Scroll */
char c;
{
        static int cur_pos;
        if (c > 0x7F) return;
        if (c == CR) {
                putchar(CR); putchar(' '); putchar(BS);
                cur_pos = 1;
        }
        if (c < 0x20) return;
        if (cur_pos > 77) return; /* kein Platz mehr */
        /* Normalfall */
        putchar(c); putchar(' '); putchar(BS);
        ++cur_pos;
}

VOID
init()  /* Initialisiert Variable <CPU> und serielle Schnittstelle */
{
        asm("move #23,d0");
        asm("trap #3");
        asm("move.w $416(a4),_CPU");
        *((char *) SER3) = 0x36; /* %00110110 : 7 Bit  1 Stop  300 Baud */
        *((char *) SER2) = 0x69; /* %01101001 : even Parity, no Echo    */
        ParityError = FALSE;
}


BOOLEAN
si_empf()   /* Wurde ein Zeichen empfangen ? */
{
        register char c;
        c = *((char *) SER1);
        return( (c & 0x08) != 0x00 );
}


int
si()   /* Liest ein Zeichen von serieller Schnittstelle ein */
{
        while (!si_empf()) if (keypressed()) exit(0);
        if (((*((char *) SER1) & 0x01) != 0)) ParityError = TRUE;
        return(*((char *) SER0));
}


VOID
so(c)   /* Gibt ein Zeichen auf serielle Schnittstelle aus */
char c;
{
        register char d;
        do d = *((char *) SER1); while ((d & 0x40) != 0x00);
        do d = *((char *) SER1); while ((d & 0x10) == 0x00);
        *((char *) SER0) = c;
}


BOOLEAN
keypressed()   /* Wurde eine Taste gedrueckt ? */
{
        register char d;
        d = *((char *) KEYD);
        if ((d & 0x80) == 0x00) {
                if (*((char *) KEYS)) ;
                return(TRUE);
        }
        return(FALSE);
}
```

Bild 12: Listing "Download HOST -> CP/ M 68K"

Erfahrungsbericht aus der NDR/mc- Reparaturabteilung

von Ulrich Kracker

FRUST? Sie haben mühevoll eine NDR- bzw. eine mc- Baugruppe aufgebaut- Ins System gesteckt- Einschalten- Keine Reaktion- Frust! ABER NICHT DOCH! Jetzt heißt es kühlen Kopf bewahren. Nachdem die erste Enttäuschung überwunden ist, sollte man (möglichst) systematisch den Fehler suchen gehen. Hier möchte ich versuchen ein paar Tips zu geben, wie Sie in dieser Situation weiterkommen könnten, ohne die Baugruppe gleich an GES zurückschicken zu müssen...

SORGFÄLTIG GEARBEITET?

Natürlich haben Sie bisher die Schritt-Für-Schritt- Aufbauanleitung, die in jedem Hardwarehandbuch erläutert wird, beachtet. So können wir also z.B. davon ausgehen, daß die Spannungsversorgung aller Bausteine in Ordnug ist und keine Fehlbestückung vorliegt.

SICHER DOCH!

A'propos Fehlbestückung: Halten Sie sich beim Einsetzen der Bauteile genau an den Bestückungsplan bzw. den Bestückungsaufdruck. Achten Sie darauf, daß Sie keine Bauteile versehentlich in eventuell in der Nähe liegende Durchkontaktierungen
eingesetzt haben. Wenn Sie lediglich eine Platine von GES bezogen haben und die Bausteine selbst beschafft haben, achten Sie darauf, daß die TTL- ICs auch vom Typ her mit den Angaben in der Stückliste übereinstimmen! Häufig sieht man nämlich LS- Typen anstatt TTL- Standardtypen eingesetzt. Timingfehler können dann damit erklärt werden, daß LS- Typen nicht die Schaltgeschwindigkeit besitzen, wie sie von Standardtypen beherrscht wird.

ERST 'MAL PUTZEN

Als nächsten Schritt in Richtung gezielter Fehlersuche waschen Sie bitte auf der Lötseite die vom Löten zurückgebliebenen Flussmittelrückstände ab. Bei frischen Lötrückständen genügt dazu Spiritus, bei alten Rückständen werden Sie zu 'härteren' Mitteln wie Waschbenzin oder Universalverdünner greifen müssen. **Aber Achtung:** Beachten Sie dabei die Warnhinweise auf der Flache!
Diese Maßnahme empfiehlt sich aus zweierlei Gründen, auch wenn kein Fehler vorliegt:-Zum einen verhindern Lötmittelrückstäde die 'ungetrübte' Sicht auf Leiterbahnen und Lötstellen.-Zum anderen können im Flussmittel enthaltene Säureanteile zu Langzeitkorrosionen führen. Zwar behaupten viele Hersteller von Elektronik-Loten nur säurefreies Kollophonium zu verwenden, aber sicher ist sicher.
Darüberhinaus bilden gealterte Lötmittelrückstäde beim Wiedererhitzen (Schaltungsänderung, Behebung eines später auftretenden Fehlers, usw.) schlechtriechende Dämpfe.
Nachdem die Platine gereinigt und getrocknet ist, kann bei einer funktionierenden Baugruppe die Lötseite mit durchlötbarem Schutzlack beschichtet werden. Dazu eignet sich z.B. Plastik 70 von Kontakt- Chemie. Bei defekter Baugruppe warten wir mit dem Schutzlack, bis sie endgültig funktioniert, da ja noch die eine oder andere Nachlötung erforderlich sein kann (auch diese Lötungen werden selbstverständlich nocheinmal abgewaschen).

SCHAU, SCHAU!

Jetzt beginnt die Suche nach sichtbaren Lötbrücken oder Ätzbrücken. Besonders anfällig für Lötbrücken sind jene Stellen auf der Lötseite, wo eine Leiterbahn derart dicht am Lötauge vorbeifährt, daß sie teilweise in die lötstopplackfreie Insel um das Lötauge herum hineinreicht.
Auch ungünstig abgewinkelte oder zu lange Drahtenden können Kurzschlüsse mit benachbarten Lötaugen oder Durchkontaktierungen bilden.Mühsam zu suchen sind (zum Teil feinste) Ätzbrücken, da sie in der Regel mit 'tarnendem' Lötstopplack überzogen sind und man in diesem Stadium noch nicht konkret weiß, in welchem Schaltungsblock man besonders suchen müßte.

GEHT'S AUCH OHNE?

Wenn es sich um eine 'Erweiterungskarte' in dem Sinne handelt, daß Sie Ihr System auch ohne die betreffende Karte betreiben können, haben Sie noch eine weitere Chance, den Fehler genauer zu lokalisieren:
Nutzen Sie die in allen NDR- Rechnergrundprogrammen vorhandenen Funktionen zum Setzen und Lesen von I/O-Ports! Viele Baugruppen werden über I/O-Adressen vom Prozessor angesprochen und dahinter stehen in der Regel Datenregister, die Werte aufnehmen und ausgeben können.
Unter welcher Adresse die Register zu erreichen sind, wird in den Handbüchern angegeben.

Tips

Tip der Redaktion

Ein komfortabler C Compiler wird von der Firma Rose unter CP/M z80 angeboten und ist somit auch für den NDR Computer
interessant.

Features des Compilers:

-Vollständige Version mit 13 stelliger BCD Arithmetik.
-Ausgabe in Z80,8080,8086 Assemblercode.
-Kompatibel zu M80/L80.
(siehe MS-Paket).
-Fehlerverfolgung mittels Trace möglich.
-Deutsche oder englische Version lieferbar.
-8", 5 1/4, 3 1/2, 3" Disk + dt. Handbuch

Vertriebsanschrift
.Herbert Rose
Bogenstraße 32
4390 Gladbeck

Korrekturen u. Ergänzungen

Probleme bei der Systemgenerierung

Häufige Probleme traten bei der Systemgenerierung des Betriebsystems CP/M68K auf. Dieser Umstand ist auf das unvollständige Submitfile der jeweiligen Speichergröße zurückzuführen. Irrtümlicherweise bestehen die Submitfiles auf der Systemdiskette 0 nur aus der Arbeitsanweisung die das einfügen einer entsprechenden Basisadresse ausführt.
Dieser Vorgang ist jedoch nicht ausreichend um die Datei CPM.SYS neu zu übersetzen.
Das Bios des NDR Computers muß aus diesem Grund integriert und neu gebunden (gelinkt) werden.
Vollständig würde eine solche Übersetzung lauten:

```
L068 -r -UCPM -o CPMREL CPMLIB
NDRBIOS.0
RELOC -BXXXXX CPM.REL
CPM.SYS
```

SCHREIBEN UND LESEN...

Geben Sie also mit den Grundprogrammfuntionen einen Wert in eines der Register und lesen Sie diesen Wert zurück. Bei Baugruppen, wo dies systembedingt nicht funktioniert, lesen Sie ein Register zurück, von dem Sie wissen, was aufgrund der vorhergehenden Ausgabe darinstehen müsste. Wenn nun eine Unkorrektheit auftritt, gilt es diese möglichst intelligent zu interpretieren:

...UND INTERPRETIEREN

-Beim Wert 'FF' könnte vorliegen:
Ist Adressdekodierung arbeit nicht richtig?!
Ist die Adressierung der Bauguppe korrekt? Überprüfen Sie die Jumperstellungen auch mit Prüfstift, Voltmeter oder Ohmmeter/ Durchgangsprüfer (vorher Spannung abschalten).
Ist die Chip- Select- Leitung von der Adressdekodierungslogik zum Betreffenden Register- Baustein in Ordnung? Überprüfung mit Ohmmeter, bzw. Durchgangsprüfer.
Ist die Adressdekodierlogik selbst in Ordnug (Auswechseln der Bausteine)
Ist der Bustreiber (meist ein 74LS245) in Ordnung?
Ist immer nur ein Bit gesetzt oder gelöscht? Schreiben Sie z.B. die Bitkombinationen 55h und AAh in die Register. Der Fehler liegt dann in den Datenwegen. Untersuchen Sie den betrffenden Datensignalpfad auf Unterbrechungen, wenn das Bit auf konstant 1 liegt und auf Kurzschlüsse wenn es auf 0 liegt.
-Bei nicht reproduzierbaren Ergebnissen liegt der Fehler vermutlich 'hinter' den Registern, soll sagen, eine eventuell vorhandene Zusatzschaltung gibt unerlaubte Signalpegel an die Register (fehlender oder unwirksamer Pull- Up- Widerstand etc.) weiter.

NUR SCHREIBEN

Wenn ein Rücklesen von Registerinhalten nicht möglich ist, hilft oft nur ein Oszilloskop, in manchen Fällen auch ein Logik-Prüfstift weiter.
Dazu wird sinnvollerweise eine kurze Endlosschleife für den jeweils eingesetzten Prozessor geschrieben, die nur einen Wert an die Baugruppe ausgibt. Damit können auch Speichererweiterungen untersucht werden. Mit dem Messgerät werden nun Adressdekodierung, Daten- und Steuerleitungen überprüft.
Dabei ist oft nur wichtig zu sehen, daß 'Aktivitäten' mit TTL- Pegeln vorliegen.
Um diese Tests durchzuführen ist es aber leider unumgänglich, den Schaltplan und die Schaltungsbeschreibung zu studieren.
Sollte es zum Auslöten von Bauteilen kommen, sollte immer die Methode des 'zerstörten Bauteiles' wählen: Mit einem Seitenschneider wird das Bauelement derart bearbeitet, daß jeder Drahtanschluß unabhängig von den anderen herausgezogen werden kann. Die Platine wird günstigerweise senkrecht positioniert (kleiner Schraubstock oder eine 'Dritte Hand') und die Lötstelle von der Lötseite her erwärmt.
Der freiliegende Drahtanschluss wird auf der Bestückungsseite mit einer Pinzette gefaßt und beim Schmelzen der Lotes herausgezogen. Wenn alle Drahtanschlüsse derart entfernt worden sind, werden die Lötaugen mittels Lötsaugpumpe oder Entlötlitze vom übrigen Zinn befreit.

WENN GAR NICHTS MEHR GEHT

Wenn das System beim Einstecken der defekten Bauguppe nicht mehr korrekt arbeitet, obwohl es müßte, ist ebenso die Addressierung (Jumper) zu überprüfen, oder (/und) der Bustreiber 74 LS 245 ist defekt.
Handelt es sich um eine Baugruppe, die bereits zum Mindestumfang eines Systemes notwendig ist, sind die Möglichkeiten zur 'Selbstdiagnose' sehr beschränkt: Da der 'Wurm' nicht unbedingt auf der gerade bearbeiteten Baugruppe konzentriert sein muß, wird zuerst der Bus mit der Spannungsversorgung nochmals überprüft. Wenn ausser sichtbaren Fehlern nichts zu finden ist, könnte das systematische Auswechseln gesamter Baugruppen innerhalb von zwei Sytemen zur Lokalisierung des Fehlers beitragen.

Ulrich Kracker , GES

Kontakte

Suche: NKC - CP/M 2.2 User im Raum Hamburg oder Hannover/Hameln. Wer hat unter CP/M 2.2 schon eine Festplatte installiert und gibt Tips? Rolf Ahrens, Lütt Kollau 1, 2000 Hamburg 61, Tel.:040/588285

Verkaufe: NDR-Computer POW5V, GDP64K, KEY, CPUZ80, CPU68K, BUS3, TAST m.G, 5*ROA64, 2*IOE, SBC2, 2*CAS, 2*DRAM128, DRAM64/256, MON1, ZASS, GOSI, BASIC, MON68K, PASCAL68008, DEMO;Gesamtbausatzwert: ca. DM2800.- aufgebaut, für nur DM1200.- VB.Hans May, Wöllsteiner Str. 8, 6551 Siefersheim, Tel.:06703/3189

Verkaufe: NE2, POW5V, POW22/26V, BUS2, BUS3, SBC2, CPUZ80, KEY, Preh Tastatur, IOE, CENT, GDP64K, BANKBOOT, FLO2, RAM256, 2 Teac 5 1/4" Floppy Disk, PROMER, Flomon 1.5, CP/M 2.2, Nevada Fortran, Z80 Tools usw. nur komplett für DM2500.- VB.Uwe Sänger, Hagenring 11, 2819 Thedinghausen, Tel.:04204/7977

Zu verkaufen weit unter Neupreis, NDR-Klein-Computer mit:
GEH3, BUS3, Netzteil, CPUZ80, KEY, CENT, BANKBOOT, CENT2, FLO2, IOE, RAM256, GDP64K, SER, ROA64, große Tastatur, 2 Floppy 5 1/4", alles in Betrieb, mit CP/M und Zeat, Preisvorstellung DM2000.-Anton Lanzl, Dieberg 69, 8411 Walderbach, Tel.:09464/346 (ab 18 Uhr)

68008: **Verkaufe** ROA64 komplett,geprüft, mit Original EASS0-3 V4.3 und Original EPASCAL V3.1 alles zusammen für DM130.-T. Strassen, Zürich, Tel.:(0041 1)555/246

In Berlin Schöneberg, Crellestraße 21, Verein Crelle 21 e.V., findet sich ab Anfang Mai regelmäßig ein **Computerclub** zusammen.Wir arbeiten mit dem NDR-Klein-Computer oder auch mit anderen Rechnern (wenn vorhanden) und werden uns mit Steuern, Messen und Regeln befassen.
Kontakte bei Interesse: Peter Ulrich 65 73 23 Crelle 21 e.V. 782 32 64

Druckausgabe ohne Verlassen des Editors !
Sparen Sie Papier und Zeit, indem Sie Dateien nur teilweise ausdrucken! RSX druckt markierten Block aus dem Editor von Turbo-Pascal unter CP/M-Plus. Programm mit ausf. Dokumentation geg. 10 DM (Scheck oder Schein) von K.P.Hasch, Postfach 250364, 4630 Bochum 25

Ab Juni 1988

PC-AT Sonderheft

Nachdem die Artikelserie der Zeitschrift "mc" zum mc-modular AT "voll eingeschlagen" hat, hat sich die Redaktion der "mc" entschieden, ein Sonderheft zum mc-modular-AT herauszubringen.

Dieses Sonderheft wurde bereits im Januar 1988 angekündigt und wird jetzt im Juni 1988 erscheinen.
Für die Anwender des mc-modular AT bildet dieses Sonderheft die ideale Ergänzung zum mc-modular AT Handbuch, das in erster Linie nur den Zusammenbau und die Inbetriebnahme der AT's beschreibt.

Wer tiefer in die Hard- und Software einsteigen will, sollte auf die detailierten technischen Beschreibungen im PC-AT Sonderheft nicht verzichten.

Eine grobe Inhaltübersicht:

- Vorstellung eines modernen Konzepts für Rechner der AT-Klasse am Beispiel des mc-modular ATs
- Detailierte Schilderung des in den meisten ATs verwendeten Chipsatzes von Chips & Technology
- Testberichte (Rein 300 LLC, VEGA VGA, MASM 5.0, Sanyo MBC-17-plus
- Aufbau und Funktion der weitverbreiteten Herculeskarte
- Filemanager für MS-DOS
- PC-/XT-/AT-Referenzliste
- Arbeitshilfen für Assembler-Programmierer
- Arbeiten mit dem 80287 (Coprozessor)

Tower-Gehäuse für den mc-modular-AT

Mit der Veröffentlichung dieses Artikels wollen wir (Graf Elektronik Systeme GmbH) unser erstes Untertisch-Gehäuse für den mc-modular-AT vorstellen.

Sie haben sich schon immer über Platzmangel auf Ihrem Arbeitstisch geärgert? Und jetzt soll auch noch ein Computer darauf stehen. Auf den Boden stellen wollen Sie den Rechner auch nicht, weil das kostbare Gehäuse zu leicht verkratzt wird. Dann stellt das neue Tower-Gehäuse eine willkommene Alternative zum Standardgehäuse für Sie dar.

Nicht nur den Vorteil der Plazierung des Gehäuses unter Ihrem Arbeitstisch beinhaltet das Tower-Gehäuse, auch die Geräumigkeit ist wesentlich vergrößert. In diesem Gehäuse finden Sie Laufwerkschächte für zwei 3 1/2"-Laufwerke, sowie zwei 5 1/4"-Laufwerken oder Festplatten. Möchten Sie zwei 5 1/4"-Laufwerke und zusätzlich noch eine Festplatte, so ist diese Möglichkeit ebenfalls berücksichtigt, denn im Gehäuse finden Sie für diesen Fall noch eine Installationsvorrichtung (hinter der Frontanzeige).

Die Frontanzeige beinhaltet den Netzschalter, den Schlüsselschalter, Turbo- und Resetschalter und eine Taktfrequenzanzeige, die die jeweils gewählte Arbeitsfrequenz der CPU-Baugruppe mit zwei Siebensegmentanzeigen darstellt. Selbstverständlich ist diese Anzeige mit dem Turboschalter gekoppelt, sodaß bei einer Betätigung auch die entsprechende Taktfrequenz angezeigt wird.

Die Außenmaße:

Breite: 165 mm
Höhe: 605 mm
Tiefe: 490 mm.

Günther Waschilewski, GES

Neue Produkte - Neue Preise!

Das neue Grundprogramm für 680xx-Systeme

Best.-Nr.	Beschreibung	Preis DM
11218	68008 (ab Juli '88)	95,-
11219	68000 (ab August '88)	95,-
11220	Quelle auf Diskette, 5 1/4" 80 Spuren	60,-
11221	3 1/2" 80 Spuren	60,-

LOG 16 - der Logik-Analysator,nun auch für 32 Kanäle

Best.-Nr.	Beschreibung	Preis DM
10906	Fertiggerät mit Software	598,-
10902	Bausatz mit SW	498,-
10898	Leiterplatte mit SW	298,-

Die mitgelieferte Software beinhaltet bereits die 32-Kanal-Lösung. Es werden dazu zwei Baugruppen benötigt.

LOGAN

Best.-Nr.	Beschreibung	Preis DM
10886	Platinen	159,60
10884	Bausatz	939,-
10885	Fertiggerät	1039,-
10887	Handbuch	20,-

BUSTEST

Best.-Nr.	Beschreibung	Preis DM
11206	Handbuch	20,-
11205	Platine	48,-
11204	Bausatz	148,-
11203	Fertiggerät	198,-

Restposten

10422	TEAC 3 1/2" Laufwerk für NDR-Computer	320,-

GDPHS - Die neue GDP64K

Best.-Nr.	Beschreibung	Preis DM
11229	GDPHS Platine	30,-
11230	GDPHS Bausatz	269,-
11231	GDPHS Fertiggerät	348,-
11232	GDPHS Handbuch	20,-
11238	Aufbausatz GDP64K - GDPHS	70,-

mc-modular-AT

Best.-Nr.	Beschreibung	Preis DM
11247	Tower-Gehäuse mit 200W-Netzteil	799,-
11118	VEGA-VGA-Karte, kann VEGA und VGA 800x400	1198,-
11120	NEC MultiSync II	1798,-
11227	mc-modular-AT Sonderheft	30,-

neu

61150	/286 CPU-Baugruppe ohne RAMs	848,-
61190	/386 CPU-Baugruppe ohne RAMs	3698,-

Speziell für LOOP-LESER

Matrix-Printer BX 100

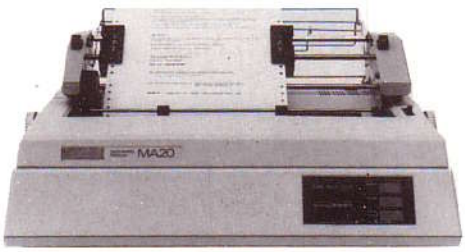
Technische Daten:

Punktmatrix-Drucker mit 100 Zeichen/sec. · Friktions- u. Traktorantrieb
Bidirektionaldruck mit Druckwegoptimierung · Inkremental-Druck
Proportional-Schrift · Unidirektionaldruck bei Graphik
Papiertransport vorwärts/rückwärts · Kursiv-Schrift
8 Nationale Zeichensätze · Programmierbarer Zeilenabstand
Text-Matrix: 9 x 9 Dots · Bit Image Mode (1 : 1 Graphik!)
Graphic-Matrix: 8 x 480 bzw. 576 Dots · 8 x 640 bzw. 720 Dots
8 x 960 bzw. 1920 Dots · 9 x 480 / 960 Dots
Programmierbare Seitenlänge · Skip over perforation
Programmierbarer linker Rand / rechter Rand
Vertikaler Tabulator · Horizontaler Tabulator
Papierbreite: 4" bis 10" (114,3 mm – 254 mm)
1 Original + 2 Durchschläge · Leistung ca. 80 Watt
Lebensdauer Druckkopf ca. 100 000 000 Zeichen
Lebensdauer Farbband ca. 2 500 000 Zeichen

LOOP-Leser-Aktion
inkl. 14 % MWSt.
Versand per UPS-NN – frei Haus DM 698,–

Typenraddrucker PETAL-MA 20

Korrespondenzqualität!
Jetzt zum erschwinglichen Preis.
Anschließbar an alle Mikrocomputer!



LOOP-Leser-Aktion:
inkl. 14 % MWSt.
Versand per UPS-NN – Frei Haus DM 698,–
Optional: Traktor DM 349,– inkl. MWSt.

mirwald electronic
Computer+Peripherie
Fasanenstraße 8b · 8025 Unterhaching/München
Tel. (089) 6111224 und 6112040 · Telex 5213476
Büro Frankfurt:
Adalbertstr. 15 · 6000 Frankfurt 90 · Tel. (069) 703538