nascom journal

Zeitschrift für Anwender des NASCOM 1 oder NASCOM 2

3. Jahrgang | September 1982 | Ausgabe 9

Herausgeber:

MK-SYSTEMTECHNIK Michael Klein · Pater-Mayer-Straße 6 · 6728 Germersheim/Rhein
Telefon (0 72 74) 20 93 · Telex 453500 mks d

MK-SYSTEMTECHNIK Thomas Gräfenecker · Kriegsstraße 164 · 7500 Karlsruhe · Telefon (07 21) 2 92 43
MK-SYSTEMTECHNIK Michael von Keitz · Pfaffenberg 4 · 5650 Solingen 1 · Telefon (0 21 22) 4 72 67

Der Heftpreis beträgt DM 4,--. Ein Abonnement erhalten Sie für DM 48,— im Jahr. Dafür bekommen Sie 12 Hefte pro Jahr, bzw. 10 Hefte (zwei dicke Doppelausgaben).
Die Autoren sind für den Inhalt ihrer Beiträge selbst verantwortlich.

INHALT

nascom journal

INTERN

Liebe Leser,
Leider ging der Wechsel in der Redaktionsarbeit nicht so reibunslos vonstatten, wie wir es uns gewünscht hätten. Für die Arbeit brauchte ich einen NASCOM-2, der von MKV so verspätet geliefert wurde, daß Sie auch dieses Heft arg verspätet in Händen halten. Besonders ärgerlich ist wohl, daß wir einige Ausgaben brauchen werden, um zeitlich wieder Tritt zu fassen.
Erfreulich ist hingegen, daß sich Herr Ballarin bereit gefunden hat, das Cassetten-Kopieren zu übernehmen. JEDER, der einen Beitrag auf Cassette (auch Text!) schickt, erhält nach Fertigstellung des Heftes, in dem sein Beitrag erscheint, die Cassette mit allen Programmen dieses Heftes bespielt zurück - Format bitte immer angeben! Für alle anderen Leser steht die Cassette in einem Rundlauf zur Verfügung.
In diesem Heft beginnt ein Artikel über digitale Bildverarbeitung von Günter Böhm. Die Hardware läuft bereits mit einfachen Programmen. Da tun sich interessante Möglichkeiten auf: Lesende Computer? Die Mustererkennung ist dann ein Software-Problem. Demnächst vielleicht auch ein "hörender" Computer, denn es ist ein Beitrag (Hard- und Software) über Sprachverarbeitung angekündigt.
Der Rest des Heftes besteht durchweg aus Beiträgen, die man nur als "nützlich" beschreiben kann. Man kann fast alles irgendwie gebrauchen, das scheint mir überhaupt die Stärke unseres Journals zu sein.
Ihr Günter Kreidl

Aktuelles zur 80-Z.-Videoplatine
Von der Videokarte könnte eine fertig gebohrte und durchkontaktierte Platine hergestellt werden. Preis je nach Nachfrage 35,- bis 40,- DM.

Eine Folie ist bei Herrn Böhm für DM 10,- + Rückporto + kart. Rückumschlag erhältlich, die übrigen Folien kosten DM 5,-.
Bitte alle Zahlungen für Folien nur an Günter Böhm:
Kto.-Nr. 148029-751
Postscheck Karlsruhe

Das Platinen-Layout der Video-Karte enthält übrigens einen Fehler, wie uns Herr Rau in letzter Minute mitteilte: IC6 (ein 7405) Pin 4 sollte an Pin 11 und am Pullup-Widerstand R19 (1K) liegen, liegt aber stattdessen direkt an +5V. Man kann den Fehler leicht beheben, wenn man's erst mal weiß!

IMPRESSUM

REDAKTION: Günter Böhm, Günter Kreidl
Wolfgang Mayer-Gürr, Josef Zeller
RESSORTS:
MASCHINENPROGRAMME:
Günter Böhm, [redacted]
[redacted] Karlsruhe, Tel. [redacted]
Günter Kreidl, [redacted], [redacted] Straelen
Tel. [redacted]
BASIC und FLOPPY:
Wolfgang Mayer-Gürr, [redacted]
[redacted], [redacted] Recklinghausen
Tel. [redacted]
HARDWARE:
Josef Zeller, [redacted], [redacted] Neu-Ulm
VERLAG: NASCOM JOURNAL, c/o MK-Systemtechnik
Pater-Mayer-Str.6, 6728 Germersheim
Tel.07274/2756 Telex 453500 mksd
VERTRIEB: Direktvertrieb durch den Verlag
Erscheinungsweise: monatlich
Bezugspreis: Im In- und Ausland 48.- für ein Jahresabonnement. Abonnements können aus technischen Gründen immer nur für die Dauer eines Kalenderjahres, d.h. vom 1.1. bis 31.12. laufen. Bei Bestellung nach dem 1.1. werden die fehlenden Hefte mit der ersten Lieferung bis zum Bestellzeitpunkt automatisch mitgeliefert. Bei nicht fristgerechter Kündigung verlängert sich das Abonnement automatisch um ein Jahr. Die Kündigung für das Folgejahr muß bis spätestens sechs Wochen vor Jahresende erfolgen
Bezugsmöglichkeiten: Durch Bestellung bei MK Systemtechnik .
Bankverbindungen: Alle Zahlungen für das NASCOM JOURNAL unter Angabe der Rechnungsnummer an MK - Systemtechnik, Germersheim.
Zahlung: Nach Eingang Ihrer Bestellung erhalten Sie von uns die ausstehenden Hefte bis zur aktuellen Ausgabe sowie eine Rechnung. Bitte, zahlen Sie dann den Rechnungsbetrag .
Bitte keine Vorauszahlungen!
Bitte, Anfragen wegen Abonnements oder Lieferung nicht an die Redaktion sondern nur an den Verlag. Die Autoren tragen die Verantwortung für ihre Beiträge selbst. Für Fehler in Text, Bildern und sonstigen Angaben kann keine Haftung übernommen werden.

DMA - Teil 2

von Josef Zeller

Im NASCOM-Journal 9/81 wurde eine Applikation mit dem Z80-DMA-Baustein vorgestellt. Angeregt durch Anfragen soll hier dieser Baustein etwas näher beschrieben werden. Anhand von Programmbeispielen (bereits im NASCOM-Journal 7/8-82 abgedruckt! RED.) soll die Programmierung dieses komplexen Bausteins erläutert werden.
Der Beitrag ist die gekürzte Fassung eines Artikels, der in MARKT&TECHNIK 16/82 erschienen ist.

1) Z80-DMA - Funktionsbeschreibung

Der Z80-DMA-Controller soll hier nur soweit beschrieben werden, um einen ersten Eindruck über die Möglichkeiten dieses intelligenten Coprozessors zu gewinnen. Für genauere Details sei auf das Datenblatt verwiesen. Der DMA-Baustein verfügt, um mit den anderen Komponenten im Mikrocomputersystem in Verbindung zu treten, über eine interne Logik und Anschlüsse nach außen, die es erlauben, die Adreß-, Daten- und Kontrolleitungen wie die CPU zu steuern, Interrupts zur CPU zu senden und die Busverwaltung mit der CPU und anderen DMAs zu koordinieren. Zum Transfer von Daten sind dafür die Adreß-, Daten- und Kontrolleitungen (MREQ, IORQ, RD, WR) in ihrer Funktion identisch den entsprechenden Anschlüssen der CPU. Auch ist das Timing des DMA gleich dem der CPU. Die Logik der Interruptanforderung entspricht der aller anderen Z80-Peripheriebausteine (PIO, CTC, DART, SIO). Sie erlaubt damit über IEI und IEO eine hardwaremäßige Zuweisung der Priorität einer Interruptanforderung von einem I/O-Baustein und die Z80-spezifischen Vektorinterrupts. Anhand von Programmbeispiel 3 wird gezeigt, wie das Low Byte des Interruptvektors in den DMA geladen wird und bei welchen Ereignissen der DMA einen Interrupt auslösen kann. Der Zugriff auf den Bus wird gesteuert durch BUSREQ, BAI, und BAO. Ueber BUSREQ (Bus Request) teilt der DMA der CPU mit, daß er die Kontrolle über den Bus übernehmen will. Die CPU behandelt BUSREQ-Anforderungen noch vor dem nichtmaskierbaren Interrupt und dem normalen Interrupt. Ueber den BUSACK-Eingang der CPU, der mit dem BAI des höchstpriorisierten DMA verbunden ist, meldet die CPU, daß sie den Bus freigegeben hat und der DMA den Bus übernehmen kann. Durch BAI und BAO können Multi-DMA-Konfigurationen aufgebaut werden. Dazu wird immer BAO vom höherpriorisierten DMA mit BAI vom niederpriorisierten DMA verbunden (Daisy-Chain-Prinzip!). Der BUSREQ-Pin der CPU überwacht im inaktiven Zustand des DMA den Bus und stellt fest, ob ein anderer DMA aktiv ist. Dadurch wird der Zugriff auf den Bus erst erlaubt, wenn der andere DMA sich vom Bus zurückgezogen hat. Ein DMA mit hoher Priorität kann den aktiven DMA mit niedriger Priorität also nicht unterbrechen, sondern muß warten, bis dieser fertig ist. Den Pins CE und WAIT sind je zwei verschiedene Funktionen zugewiesen. Im passiven Zustand dient CE als Chip-Select und über INT wird das Interruptsignal ausgegeben. Durch Programmierung kann der CE als Wait-Eingang dienen und der INT-Pin das Puls-Signal aussenden, wenn ein Block von programmierbarer Länge übertragen worden ist. Durch den RDY-Eingang kann ein Peripherie-Baustein (SIO durch W/RDYA oder W/RDYB) den DMA zum Uebertragen auffordern.
Der DMA weist 3 Funktionsarten auf, die in den Programmbeispielen auch angewandt werden: Übertragung (Prg. 1), Suche (Prg. 2) und kombinierte Suche/Übertragung (Prg. 3) von Daten zwischen 2 Ports. Ein Port kann ein Memorybereich oder ein I/O-Baustein sein. Die Startadressen, ab denen die Übertragung beginnt, werden in die Port-Start-Register und die Übertragungslänge ins Block-Lenght-Register geladen. Es sind Blocklängen bis 64 K-Byte adressierbar. Die Counter können von der CPU ausgelesen werden und zeigen an, bis zu welcher Adresse und wieviel Bytes übertragen wurden (Programm Global). Der DMA kann einen Port nach einem bestimmten Byte oder einer Bitkombination absuchen. Beim Übertragen wird ein Byte von einem Port gelesen und dann zum anderen Port geschrieben. Beim Übertragen/Suchen kann so lange übertragen werden, bis Gleichheit auftritt. Die Optionen werden hier meistens anhand der Übertragung von Daten besprochen. Natürlich gilt das Gesagte auch für die Datensuche. lich gilt das Gesagte auch für die Datensuche.
Der Z80-DMA kann in drei Betriebsarten arbeiten. Im Byte-Mode wird pro Busanforderung nur ein Byte übertragen. Die CPU führt dann einen Befehl aus, bevor der DMA das nächste Byte überträgt. Um im Byte-Mode zu arbeiten, muß RDY aktiv sein. Im Burst-Mode werden Daten so lange übertragen, wie RDY aktiv ist, auch wenn die Operation noch nicht abgeschlossen ist. Im Continuous-Mode ist der DMA so lange aktiv, bis

der Datenblock vollständig übertragen wurde, unabhängig von RDY. Hier muß eine kleine Einschränkung beim Benutzen von dynamischen Rams gemacht werden. Da die Speicherzellen bei Drams alle 2 msec aufgefrischt werden müssen, darf die CPU nicht länger angehalten werden.

Eine Besonderheit des Z80-DMA stellt seine Eigenschaft dar, durch entsprechende Programmierung das Timing-Verhalten bei Zugriffen auf die Ports zu manipulieren. Das READ/WRITE-Timing kann den Zugriffszeiten der Speicher und I/O-Bausteine angepaßt werden. Eine Optimierung der Datenübertragung ist damit zu erreichen. Die CPU benötigt für Speicherzugriffe 3 und für I/O-Zugriffe 4 Taktzyklen. Nach dem RESET weist der DMA gleiches Timing-Verhalten auf. Die Schreib/Lese-Zyklen sind durch Programmierung des DMA auf 2 - 4 Taktzyklen einstellbar. Bei den heute üblichen Rams mit Zugriffszeiten von 200 nsec sind Zugriffe mit 2 Taktzyklen ausreichend. Bei Zugriffen auf Z80-I/O-Bausteine ist das Standard-Timing einzustellen, da sie für dieses Verhalten ausgelegt sind. Über WAIT kann natürlich der Zugriff auf extrem langsame Hardware beliebig verlängert werden. Der DMA kann maximale Datenübertragungsraten von 1 MHz und Suchraten von 2 MHz in der Z80A-Version erreichen.

Eine weitere Eigenschaft ist die Option "Auto Restart". Der DMA wird nach der Übertragung eines Blocks veranlaßt, die Port-Startadressen und den Blockcounter zurückzusetzen. Der DMA kann ohne Belastung der CPU seine Arbeit erneut aufnehmen. Dies ist für Operationen notwendig, die zyklisch wiederholt werden müssen, wie z.B. der Bildschirmspeicherrefresh. Beim Suchen wird während des Vergleichs bereits das nächste Byte gelesen (Pipeline-Architektur).

2) Programmbeispiele

Zum Schluß soll nun die Programmierung des DMA anhand von Programmbeispielen gezeigt werden. Das Programm 1 (COPY) zeigt, wie Speicherbereiche kopiert werden, Programm 2 (SEARCH) die Suche nach einem Byte und Programm 3 (TRASEA) wird genauer besprochen, da in diesem Programm fast alle Optionen, die der DMA anbietet, realisiert sind. Es sei folgende Problemstellung gegeben: Der DMA soll von einem Peripherie-Baustein (Input) Daten blockweise (256 Bytes) in den Hauptspeicher laden, bis die "End of Transmission" Marke erreicht wird. Der CPU wird über Interrupt mitgeteilt, wann ein Block gelesen oder die EOT-Marke erreicht wurde. Das Programm TRASEA wird vom Hauptprogramm aufgerufen und initialisiert den DMA durch Steuerworte. Der DMA wird wie ein I/O-Baustein über eine 8-Bit-Adresse angesprochen.

Der DMA verfügt über insgesamt 27 WRITE-Register, die in je 7 Hauptregister (WR0 - WR6) eingeteilt sind, denen insgesamt 14 weitere WRITE-Register untergeordnet sind. In TRASEA werden am Anfang die Register in den Stack gerettet, anschließend die Interruptvektoren geladen und über den OTIR-Befehl die Steuerworte in der Kommandotabelle 3 zum DMA ausgegeben. Die CPU kehrt dann ins aufrufende Programm zurück, währen der DMA simultan dazu seine Arbeit aufnimmt. In der Kommandotabelle sind nun sämtliche Steuerworte abgelegt, mit denen der DMA programmiert wird. Sie beginnt immer mit einem "RESET DMA", um den DMA in einen definierten Ausgangszustand zu setzen. In WR0 wird die Funktionsart festgelegt, hier: Übertragung/Suche von Port A nach Port B. Als nächstes folgen die Startadresse von Port A und der Blocklängenzähler. Im Beispiel steht für Port A die Input-Adresse 000BH und die Blocklänge 00FFH. Für die Adresse von Port A wäre ausreichend, nur 0BH anzugeben. Unter Umständen kann aber beim Auslesen vom Port-A-Counter das High Order Byte einen falschen Wert aufweisen. WR1 legt nun das Verhalten von Port A fest. Hier wird Port A durch eine 1 in Bit 3 als I/O definiert. Die Bits 4 und 5 legen fest, ob der Port-Counter incrementiert oder decrementiert wird oder einen festen Wert erhält (I/O: Adresse fest). Bit 6 auf 0 setzt das Timing von Port A in den Standardzustand. Analog wie WR1 bezieht sich WR2 auf Port B. Port B wird in WR2 definiert als Memory, der Port-Counter wird incrementiert, und durch Bit 6 auf 1 weiß der DMA, daß durch das folgende Byte das Timing von Port B programmiert wird. In diesem Byte wird festgelegt, daß Port B immer mit 2 Taktzyklen auf den Speicher zugreift. In WR3 folgt das "Match-Byte", nach dem gesucht werden soll. WR4 legt die Betriebsart fest (hier Byte Mode) und gibt an, ob die Startadresse von Port B, der Interruptvektor und die Interrupt-Kontrollregister folgen. Nach WR4 wird nun die Port-B-Adresse ausgegeben (hier 4000H, d.h. ab der Adresse 4000H werden die eingelesenen Daten abgespeichert), gefolgt vom Interrupt-Kontrollregister, das definiert, bei welchen Bedingungen ein Interrupt erzeugt wird. Hier wird

festgelegt, daß bei Ende des Blocks oder wenn das Match-Byte gefunden wurde, ein Interrupt erzeugt wird. Es besteht noch die Möglichkeit, einen Interrupt auszulösen, wenn beim DMA der RDY-Eingang aktiv wird. Eine Busanforderung wird dann noch nicht ausgegeben. Zum Schluß dieser Programmsequenz folgt die Ausgabe des Low Byte des Interruptvektors. Durch das "Status Affect Vector" Bit wird, je nach Interruptursache die entsprechende Routine angesprungen.

In WR5 wird RDY auf aktiv bei High und der CE-Eingang als WAIT definiert. Durch Bit 6 kann der DMA veranlaßt werden, die Operation zu wiederholen (Auto Restart) oder abzubrechen. Zum Schluß der Programmierung folgen nun einige Steuerbefehle aus der WR6-Gruppe. Durch C7H wird Port A an das Standard-CPU-Timing angeglichen (ist aber hier nicht unbedingt nötig). Die folgenden Befehle bewirken, daß die Startadressen intern geladen und die Statusbits gelöscht werden. Nach "Interrupt Enable" und "Enable DMA" kann der DMA aktiv werden. Es sei angenommen, daß ein Block eingelesen wird. Dann löst der DMA einen Interrupt aus und die CPU springt die Interruptroutine IENDBL an. Es wird nun die Meldung ausgegeben, daß sich ein Interrupt durch Blockende ereignet hat. Anschliessend werden wieder einige Steuerworte zum DMA ausgegeben. Zuerst werden die Statusbits gelöscht, durch das Byte D3H die Blocklänge intern neu geladen, während die Adreßzähler aber unverändert bleiben. Die Übertragung wird also beim momemtanen Adreßstand fortgesetzt. Zum Schluß dieser Kommandosequenz muß dem DMA mitgeteilt werden, daß er erst nach dem RETI (Return from Interrupt) wieder aktiv werden darf, gefolgt vom "Enable DMA" Kommando. Diese letzten beiden Steuerworte müssen immer am Ende einer Interruptroutine ausgegeben werden, um eine ordnungsgemäße Funktion des DMA zu gewährleisten.

Der DMA überträgt die weiteren Blöcke, bis das Match Byte gefunden wird. In der Interruptroutine IMATCH wird die Meldung ausgegeben, daß die EOT-Marke gefunden und übertragen wurde. Der DMA wird nicht mehr initialisiert. Die Interruptroutine MATEND wird angesprungen, wenn ein Block übertragen und das letzte Byte gleichzeitig das Match Byte ist.

Das Programm COPY kopiert einen Speicherbereich in einen anderen Teil des verfügbaren Speichers. Dabei wird ab der Adresse D000H ein Block der Länge 01F3H ab 4000H im Continuous Mode abgespeichert. Da man bei Übertragungen, die nur auf den Speicher zugreifen, kein externes RDY-Signal benötigt, wird diese Bedingung durch das Byte B3H (Force RDY Signal) aktiv gesetzt. Das Programm SEARCH sucht ab D000H nach einem bestimmten Byte (D8H) und übergibt die Adresse, an der das Match Byte gefunden wurde, im Register HL dem aufrufenden Programm. Bei Suchoperationen wird nur ein Port benötigt. Beide Programme initialisieren den DMA im Continuous Mode und setzen das RDY-Signal intern aktiv. Der DMA beginnt nach dem "Enable DMA" Kommando sofort mit der Übertragung bzw. Suche. Lädt das aufrufende Programm die Speicherzellen, in denen die Portadressen und Blockcounter stehen, mit den aktuellen Parametern, so können diese beiden Programme die CPU-Befehle LDIR und CPIR ersetzen.

Die CPU kann vom DMA die 7 READ-Register RR0 - RR6 lesen, die die aktuellen Adressen für die beiden Port-Counter und den Block-Length-Counter enthalten. RR0 ist das Statusbyte, in dem der DMA anzeigt, ob RDY aktiv ist, der DMA aktiv war, Interrupts anstehen, Blockende übertragen und ein Byte gefunden wurde. Das Statusbyte wird ausgelesen vom Programm STATUS, das es in die Speicherzelle RR0 lädt, wo es zur weiteren Bearbeitung für die CPU zur Verfügung steht. Das Programm PRINT gibt den DMA-Status im Klartext zum Bildschirm aus. Das Programm GLOBAL liest sämtliche READ-Register aus und speichert die Werte für den Blockcounter nach RR1RR2, Port A Counter nach RR3RR4 und Port B Counter nach RR5RR6, wo die CPU diese Werte auslesen kann und im Programm LIST die aktuellen Werte der DMA-Counter zum Bildschirm ausgibt. Zum Lesen soll noch angemerkt werden, daß durch die Read-Maske definiert wird, welche READ-Register von der CPU ausgelesen werden sollen.

MDCR-Controller Karte

von H.G. Ingelaat und U. Forke

Die vorliegende Schaltung für MDCR-Benutzer und solche, die es noch werden wollen, basiert auf einer Idee von J. C. Lotter. Nachdem ein Programm zur MDCR-Steuerung in einer Journalausgabe des letzten Jahres vorgestellt wurde, folgt hier eine komplette Hardwareeinheit, die folgende Vorteile bietet:

- Eigene Pio auf der Karte; der NASCOM-Pio-Bus

bleibt frei für andere Anwendungen.
- Portadressen über Dil-Schalter individuell einstellbar.
- Durch Umstecken eines Jumpers auf der Karte läuft diese problemlos bei 2 oder 4 MHz.
- Voll softwarekompatibel mit UNICON 1.4.
- Aufbau auf einer doppelseitigen, durchkontaktierten Karte, voll NASBUS-kompatibel (77-polige Kontaktleiste).

Die Datenbusleitungen sind bidirektional gepuffert, so daß sich auch bei längeren Busleitungen keine Störspannungsprobleme ergeben. Die Schaltung läuft bei mir seit ca. einem Monat mit 70 cm langen Busleitungen. Sollten sich beim Betrieb, besonders bei 4 MHz, dennoch Probleme ergeben, dann sei an dieser Stelle schon auf eine Abhilfe hingewiesen, die voraussichtlich in der nächsten Journalausgabe vorgestellt wird:

Eine Mini-Pufferkarte für die CPU des NASCOM-1, die einfach an Stelle der CPU in den Sockel gesteckt wird; die Signalform auf dem NASBUS und die Störsicherheit werden erheblich verbessert.

Besonders beim Schalten elektrischer Verbraucher hat sich der NASCOM bei mir als echter "Aussteiger" erwiesen. In solchen Fällen kann ich jetzt schon für Abhilfe garantieren. Auch diese Karte ist bei mir in Gebrauch und ermöglicht den Betrieb der 32K-Ram-Karte bei 4 MHz, und zwar ohne Waitzyklen. Vorher war das nicht möglich; auch Tricks und Kniffe haben nichts genützt. Wer die MDCR-Karte zusammen mit anderen I/O-Bausteinen oder Geräten betreiben will, muß beachten, daß das I/O-EXT-Signal an der Grundplatine invertiert werden muß, da ansonsten die Ports 0 bis 3 nicht gesperrt werden (Konstruktionsfehler?). Bei Verwendung der Mini-Pufferkarte sind die Signale I/O-EXT und DBDR nicht erforderlich und IC15 auf der MDCR-Karte kann entfallen. So können auch Erweiterungskarten anderer Bussysteme problemlos angeschlossen werden.

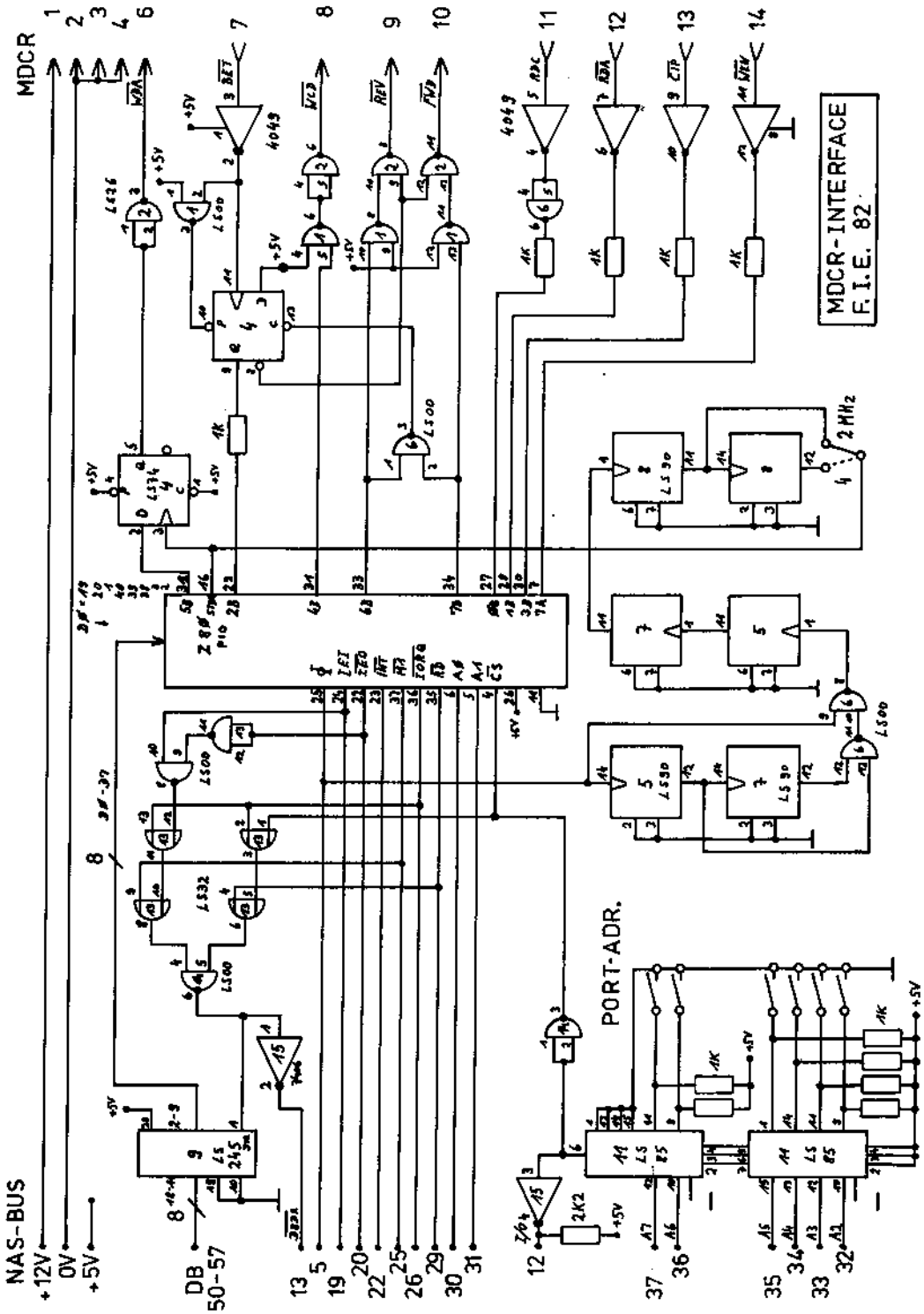
Anm. der Redaktion:
1) Hier liegt wohl ein Mißverständnis vor. Wenn externe I/O-Bausteine angeschlossen werden sollen, muß die Adreßdekodierung der internen Ports vervollständigt werden. Dazu muß LK1 auf EXT. umgelötet werden und auf der jeweiligen Zusatzkarte das Enable-Signal (active low!) für die internen Ports erzeugt und über I/O-EXT eingespeist werden.
2) Bei der MDCR-Karte würde ich die Ausgänge der LS26 mit Pullup-Widerständen (5.6K bei der Eltec-Karte) nach 12V und die Eingänge der 4049 mit Pulldown-Widerständen (68k bei Eltec) versehen. GK

Typenrad-Terminal
Teil 3
von Günter Kreidl

Halbschrittformatierung

Das im letzten Journal vorgestellte Drucker-Interface für die Praxis-30 von Olivetti ermöglicht auch die Ansteuerung fast aller Sonderfunktionen der Schreibmaschine. Eine der interessantesten Anwendungen für die Textverarbeitung ist die rechtsbündige Formatierung von Texten mit Hilfe der Halbschrittfunktion. Der MC-Texteditor verfügt (ebenso wie NASPEN) über eine rechtsbündige Formatierung, doch ist sie für anspruchsvolle Anwendungen nicht zu gebrauchen: Das Ding schiebt gnadenlos Spaces ein, bis die vorgegebene Stellenzahl erreicht ist gelegentlich besteht eine Zeile dann aus 2 (langen) Worten mit einem Dutzend Leerstellen dazwischen!

Das von Günter Böhm vor einiger Zeit im NASCOM-Journal vorgestellte Formatierprogramm vermeidet diese Schwächen: Wenn eine Formatierung mit einfacher Verdoppelung der Leerzeichen (also maximal 2 Leerstellen zwischen 2 Worten) nicht möglich ist, wird der Benutzer zu Hilfe gerufen er muß dann das letzte Wort trennen, um eine saubere Formatierung zu ermöglichen. Mit der Halbschrittfunktion der Praxis-30 (oder auch eines anderen Druckers bzw. einer anderen Schreibmaschine) ist aber noch eine viel elegantere Lösung möglich: Die Zwischenräume zwischen 2 Worten werden entweder auf die Hälfte verkürzt oder auf das Anderthalbfache verlängert. Dem in der letzten Ausgabe enthaltenen Druckerprogramm für die Praxis-30 hatte ich die entsprechenden Funktionen bereits eingebaut: Die Control-Codes 01 und 03 werden als halbe bzw. anderthalbfache Leerstelle ausgegeben. Bisher habe ich die eigentliche Formatierung (Einfügen der Control-Codes) von Hand besorgt. Das ist bei der Vielzahl von Texten, die ich jetzt zu bearbeiten habe, aber zu langwierig. Deshalb wollte ich zuerst Günter Böhms Formatierprogramm auf Halbschrittformatierung umschreiben. Dann wäre allerdings die Anbindung an verschiedene Textverarbeitungsprogramme nicht



VERKAUFE alle Teile Original NASCOM-1 mit Beschreibung:
NAS-SYS-ROM DM 30,-
Charakter-Generator-Rom DM 15,-
Disassembler auf Cass. (T2) DM 40,-
Set- und Schachgraphikkarte incl. Eprom DM 40,-
U. Forke
[redacted]

MDCR-CONTROLLER-KARTE
wie in diesem Heft beschrieben
Platine DM 80,-
Fertig aufgebaut und getestet DM 190,-
Bei etwas größerer Auflage entspr. billiger

U. Forke
[redacted]

möglich gewesen. Hier wird deshalb eine andere Lösung vorgestellt: das Formatierprogramm ist als Ausgabetreiber geschrieben, der mit dem "U"-Befehl des Betriebssystems der normalen Bildschirmausgabe zugeschaltet werden kann. Vorher muß die Startadresse des Programms HFORM bei C78H eingetragen werden. Damit kann das Formatierprogramm mit jedem Texteditor zusammenarbeiten.

Programmanlauf

Der auszugebende Text wird zunächst in einen Zeichenpuffer geschrieben, bis die vorgegebene Stellenzahl erreicht ist. Dann wird beim nächsten Leerzeichen (jeder Code kleiner als 21H wird als Leerzeichen interpretiert!) mit der Formatierung begonnen. Hat die Zeile genau die vorgeschriebene Länge, wird sie ausgedruckt. Sonst wird zunächst durch Einfügen von Halbschritten anstelle der ganzen Leerschritte eine Verkürzung versucht. Ist dies nicht möglich, wird das letzte Wort abgetrennt und eine Verlängerung der um dieses Wort verminderten Zeile mit anderthalbfachen Leerschritten versucht. Ist dies möglich, wird die Zeile ausgedruckt und das nicht berücksichtigte Wort an den Anfang des Zeilenpuffers geschoben, damit es bei der nächsten Zeile mitgedruckt wird. Dann kehrt die Ausgaberoutine ins Hauptprogramm zurück, das dann weitere Zeichen ausgeben kann, bis wieder die vorgegebene Stellenzahl erreicht ist. Ist eine Formatierung weder durch Verkürzung noch durch Verlängerung möglich, wird der Benutzer zu Hilfe gerufen: die nicht formatierbare Zeile wird auf dem Bildschirm angezeigt und kann nun mit den Editierfunktionen von NAS-SYS bearbeitet werden. In der Regel wird man das letzte Wort trennen und damit eine Formatierung ermöglichen. In manchen Fällen ist es auch notwendig, jeweils einen zusätzlichen Leerschritt von Hand einzusetzen, wenn eine Trennung des letzten Wortes nicht möglich ist. Das Editieren wird mit NEWLINE abgeschlossen. Dabei MUSS zwischen dem Cursor und dem letzten Zeichen der Zeile eine Leerstelle sein, sonst kann in der nächsten Zeile ein Leerzeichen verlorengehen!

Der Programmteil EDIT ruft HFORM auf; d.h. das ganze Programm ist rekursiv programmiert. Der Vorteil der Rekursion liegt in der Einfachheit und Kürze des Programms. Der Nachteil ergibt sich für den Benutzer beim Editieren: er darf keine Fehler machen! Wird z.B. das letzte Wort so getrennt, daß eine Formatierung immer noch nicht möglich ist (weil die nötige Anzahl von Leerstellen im Text nicht vorhanden ist), wird wieder die Editierfunktion aufgerufen. Dabei geht aber der bereits abgetrennte Wortteil verloren. Man kann dann nur noch mit RESET aussteigen und das Hauptprogramm neu starten. Man muß dann dafür sorgen, daß die Ausgabe genau an der Stelle wieder beginnt, wo der ausgedruckte Text endet. Nach meiner Erfahrung macht man diesen Fehler aber nur einmal!

Natürlich soll nicht jede Zeile formatiert werden. Bei Überschriften, Absätzen oder am Textende muß die Formatierungsfunktion aufgehoben werden können. Dies geschieht mit dem Paragraphenzeichen (bzw. dem "Schnecken"-a - das kann ich deshalb auch nicht über den Drucker ausgeben!). Zwischen dem Paragraphenzeichen und dem letzten Textzeichen muß wiederum ein Leerschritt sein, sonst kann es in bestimmten Fällen zu Fehlfunktionen kommen. Folgt auf das Paragraphen-Zeichen ein Leerzeichen (oder irgendein Control-Code, z.B. OD= NEWLINE), so wird dies in die nächste Zeile übernommen. Das führt dazu, daß die nächste Zeile um eine ganze, halbe oder anderthalbfache Leerstelle (je nach Formatierung) eingerückt wird. Das ist in vielen Fällen erwünscht, kann aber auch vermieden werden, wenn hinter dem Paragraphenzeichen ohne Zwischenraum gleich wieder der Text folgt.

Das Programm hat noch 2 weitere Funktionen. Es ist ein Zeilenzähler eingebaut, der nach einer vorgegebenen Zeilenzahl anhält (Papierwechsel!) und nach Drücken von NEWLINE das Programm wieder startet. Der Programmteil EDIT wandelt 2 Tastencodes um (für Leute, die ihren NASCOM mit deutschem Zeichensatz versehen haben): "SHIFT O" = Ü und "CTRL O" = Ö. Auch ohne deutschen Zeichensatz ganz nützlich - schließlich kann die Schreibmaschine Deutsch.

Programmparameter

Das Programm habe ich nicht mit Kommentaren versehen. Ich hoffe aber, daß die Ablaufbeschreibung und die Namen der Labels und Unterprogramme aussagekräftig genug sind. Wichtig sind aber noch die Parameter des Programms, die vom Benutzer je nach Bedürfnissen zu ändern sind. COUNT enthält die vorgegebene Zeichenzahl pro Zeile; COUNT+1 enthält jeweils die aktuelle Anzahl der Zeichen im Puffer. BUFPTR zeigt jeweils auf die nächste freie Stelle im Zeichenpuffer. LCOUNT enthält die Anzahl der bereits gedruckten Zeilen; LCOUNT+1 die Zeilenzahl, die an einem Stück gedruckt werden soll. Das Druckerprogramm habe ich hier an die

Adresse E3OH gelegt (weil es sich da gerade an HFORM anschließt und noch genügend Platz für die Initialisierungsroutine des Texteditors bleibt). Wie das Programm arbeitet, kann man an dieser Ausgabe des NASCOM-Journals sehen, die damit erstellt wurde.

Tips, Tricks + Käfer

Grundsätzliches

Diese Kolumne soll ein regelmäßiger Bestandteil des Journals werden. In ihr sollen alle die nützlichen Hinweise enthalten sein, die keinen eigenen Artikel lohnen, aber dem Leser oft eine Menge Ärger ersparen. Wo (Zeitschriften, Bücher) findet man welche Programme und wie kriegt man sie auf dem NASCOM ans Laufen? Wie schließt man systemfremde Hardware an?

Und natürlich die lieben Käferchen!

Jeder Hinweis von den Lesern ist willkommen - Postkarte oder Anruf genügt (für ganz Eilige!).

Wenn sich die PIO aufhängt

Wenn man ein Programm testet, bei dem die PIO im Interrupt-Modus arbeitet, oder aus einem solchen Programm per RESET aussteigt, kann man eine böse Überraschung erleben: Die PIO gibt einfach keine Interrupt-Anforderung mehr aus. Hier hilft nur noch das Ausschalten des Computers, wenn man nicht den folgenden Trick kennt: Man ruft ein Unterprogramm auf, daß nur aus dem Befehl RETI besteht. Die PIO lauert nämlich auf diesen Befehl und ist erst dann wieder für einen Interrupt bereit, wenn er auf dem Bus erschienen ist. Es gibt auch eine Hardwarelösung. RESET wird mit M1 über ein Und-Gatter verknüpft auf den M1-Eingang der PIO gegeben. Dann wird die PIO durch RESET zurückgesetzt.

TOOLKIT

Soll das TOOLKIT im Ram laufen, müssen die folgenden Bytes auf OO gesetzt werden: BO2OH, BO21H, B23EH, B23FH, B247H, B248H.

Disassembler

Die Zeitschrift ELCOMP hat im Februar 81 einen schönen Z8O-Disassembler veröffentlicht (er wurde im Aprilheft noch entkäfert!), der auch die "inoffiziellen" Z8O-Befehle verarbeiten kann. Mir war das Programm jedenfalls sehr nützlich, aber einmal bin ich daran verzweifelt, als ich ein Maschinenprogramm rückübersetzen ließ und dann als Vorlage eines neuen Assemblerprogramms benutzte: Das Programm wollte einfach nicht laufen. Erst nach vielen Mühen kam ich dahinter, daß der Disassembler einen Fehler aufweist: die beiden Befehle RLD und RRD werden vertauscht! Im Originallisting steht bei 8368H der Wert CCH und bei 836AH der Wert D2H. Das muß vertauscht werden und dann läufts!

Hilferuf

Eine Reihe von NASCOM-Benutzern versucht bislang vergeblich, die 64K-Speicherkarte aus der Zeitschrift MC 4/81 am NASCOM ans Laufen zu bringen. Vor allem beim NASCOM-1 gibt es da Probleme (Ich gehöre selbst auch zu den Betroffenen!). Wer die Karte vielleicht sehon erfolgreich angeschlossen hat, der möchte sich bitte DRINGEND bei der Redaktion melden!

Tippgemeinschaft

Die Rechengenauigkeit des NASCOM-Basic reicht für viele Anwendungsfälle nicht aus. In seinem Buch "Basic Interpreter" hat Rolf-Dieter Klein das 12K-TDL-Basic veröffentlicht, das mit 11 Stellen rechnet und auch sonst einige Vorzüge aufweist. Aber 12K eintippen und dabei vollständig relozieren!? - Für einen allein ist das grausam! Wer macht bei einer "Tippgemeinschaft" mit? Interessenten sollen sich bitte bei der Redaktion melden.

G.K.

Verkaufe mein Schachprogramm (Originalkassette) Preis 70,- DM
Peter Urban
[redacted]
Tel.: [redacted] ab 18 Uhr

Selbstbau-PLOTTER (siehe Fotos im Journal 11/12-81)
-mit zusätzlichen Motoren und Getrieben
-Software im EPROM
-Netzteil (2x48V,5V) mit Trafo
-Software für Plotterschrift und Barcode
-weitere Programme für Grafik etc.
-Interface zur Motoransteuerung (Pio-Bus)
-ideal zum Experimentieren (auch für Schacharm etc.)
-Preis DM 900.- VB
Günter Böhm [redacted]

8-Kanal-Logiktester

von Günter Böhm

Folgendes Programm habe ich aus einer Notlage heraus geschrieben, als nämlich bei der Arbeit am Interface für die Video-Kamera mein Oszilloskop den Geist aufgab. Vorher hatte ich schon mit dem Gedanken gespielt, etwas ähnliches zu erstellen, denn mit meinem 1-Kanal-Gerät war es nie möglich, den Verlauf zweier TTL-Signale gleichzeitig zu erfassen.

Aus Zeitnot wurde das Programm schnell in Basic geschrieben; dies ist schon sein erster Nachteil, denn es dauert eine ganze Weile, bis sich das Bild aufgebaut hat. Der zweite Nachteil ist, daß der verwendete PIO-Port im Abstand von min. etwa 8 usec abgetastet werden kann (der Rechner schafft es bei 2 MHz. einfach nicht schneller), und so die dargestellte Länge der Signale verfälscht wird. Eine Verbesserung ergäbe der Einbau einer Möglichkeit, den Start der Abtastung zu triggern (vielleicht durch Interrupt). Möglicherweise wird ein Leser durch diesen Artikel angeregt, eine verbesserte Version in Maschinensprache zu schreiben.

Jedenfalls hat mir das Programm geholfen, einige Zeit ohne Oszilloskop auszukommen; und in Zukunft wird es wohl bei solchen Gelegenheiten zum Einsatz kommen, bei denen es nicht so sehr auf die exakte Darstellung der Impulslängen ankommt, sondern mehr auf die Anzeige des Pegels vieler TTL-Signale zu einem bestimmten Zeitpunkt.

In der abgedruckten Fassung ist die kleinste Zeiteinheit 10,5 usec. So wird zwischen 2 Teilstrichen eine Periode von 21 usec dargestellt. Die maximal meßbare Frequenz beträgt somit 47,6 KHz, die langsamste Frequenz, von der noch eine vollständige Periode abgebildet wird, beträgt 1 KHz.

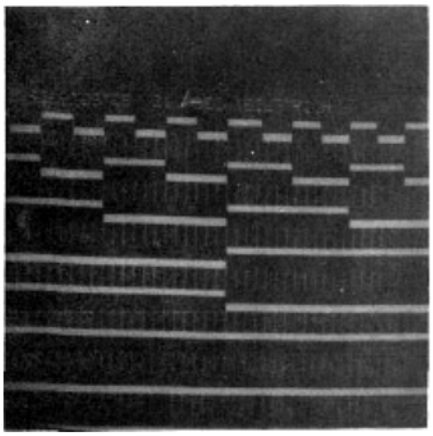
Die Anwendung ist einfach: an einem Port der PIO werden bis zu 8 Leitungen angeschlossen, die TTL-Pegel führen. Nach RUN startet die erste Messung. Wenn der Bildschirm vollgeschrieben ist, wartet das Programm auf das Drücken einer beliebigen Taste, um die nächste Messung durchzuführen.

Im Basic-Listing wird Port 3D/3F verwendet. Um Port B der Grundplatine verwenden zu können, müßten die zweite und dritte DATA-Anweisung in Zeile 40 durch 2003 und 1294 ersetzt werden.

Für die Benutzer anderer Systeme ist das Maschinenprogramm im Assembler aufgelistet. Dies kann durch POKE-Befehle (anstelle der NASCOM-eigenen DOKE) an einen beliebigen Speicherplatz geladen werden, der für Maschinenunterprogramme benutzt werden kann. Aufgerufen wird die Routine in Zeile 190 durch die USR-Funktion.

Zeile 60 gibt dem Programm an, wo das Unterprogramm abgelegt ist, Zeile 65 verhindert einen Zeilenvorschub, wenn das Programm am Ende der Anzeige auf eine Eingabe wartet.

```
1 REM LOGIKTESTER 8 KANAL
2 REM VER.1.Ø
3 REM GÜNTER BÖHM K'HE
4 REM Sept.1982
1Ø REM MASCH.PROGR. LADEN ******************
2Ø FOR I=32ØØ TO 3212 STEP 2
3Ø READJ:DOKEI,J : NEXTI
4Ø DATA 2Ø286,16339,1563Ø,33,1549,-4768,-13ØØ2
5Ø REM ************************************
6Ø DOKE 41ØØ,32ØØ :REM USERLOC
65 DOKE 4175,-6649 :REM KEIN ZEILENVORSCHUB
7Ø SP= 3328 : REM SPEICHER
8Ø CLS : SCREEN 1,16
9Ø PRINT"LOGIK-TESTER  21 uSEC/TEILSTRICH
1ØØ REM RASTER ZEICHNEN **********************
12Ø FOR I=2 TO 14 STEP 2 :REM ZEILEN
13Ø FOR J=1 TO 47 : REM SPALTEN
14Ø SCREEN J,I
15Ø PRINT"DEF"; : REM GRAPH-CONTR-T
16Ø NEXTJ:PRINT: NEXTI
17Ø REM ************************************
19Ø Z=USR(Ø) : REM WERTE ABSPEICHERN
2ØØ REM ************************************
21Ø REM WERTE ANZEIGEN
214 BIT=1
215 FOR Z=Ø TO 42 STEP 6 : REM ZEILEN
22Ø FOR I=Ø TO 95 : REM SPALTEN
23Ø A=PEEK(SP+I):B=A AND BIT
24Ø IF B=BIT  THEN SET(I,Z)
25Ø IF B=Ø THEN SET(I,Z+2)
26Ø NEXTI:BIT=BIT*2 : NEXTZ
27Ø SCREEN 45,15 :INPUT A$
28Ø GOTOØØ
Ok
```



ZEAP Z80 Assembler - Source Listing

```
                    0010 ;MASCHINENPROGRAMM
                    0020 ;ZUM ABSPEICHERN VON
                    0030 ;TTL-PEGELN PORT B
                    0040 ;G.BOEHM 26.8.82
                    0050 ;
215F 003F           0060 CPORT  EQU  #3F; PORT B CONTROL
215F 003D           0070 DPORT  EQU  #3D; PORT B DATA
215F 0D00           0080 BUFFER EQU  #D00; SPEICHER
                    0090 ;
0C80                0100        ORG  #C80
0C80 3E4F           0110 INIT   LD   A,#4F; INPUT MODE
0C82 D33F           0120        OUT  (CPORT),A
0C84 0E3D           0130        LD   C,DPORT
0C86 21000D         0140        LD   HL,BUFFER
0C89 0660           0150        LD   B,96 ; ANZAHL DER WERTE

0C8B EDB2           0160 GO     INIR ; WERTE ABSPEICHERN
0C8D C9             0170        RET
```

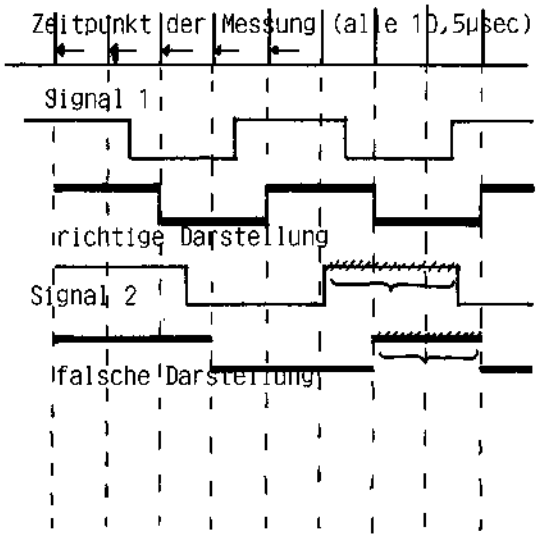
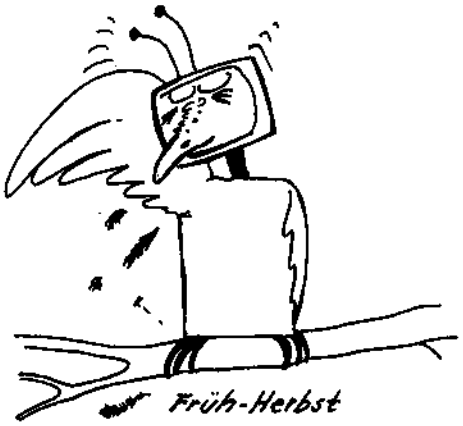


Abbildung der Signale eines Teilers

nascompl



Hallo liebe Leser,
Nehmen Sie einmal eine Lupe zur Hand und betrachten Sie die Tastatur Ihres NASCOM. Finden Sie auf den Tasten auch solche Fusseln und Körnchen? Kein Grund zur Aufregung: es ist mal wieder September, und das ist für unsere Rechner die alljährliche Zeit der Mauser. Ärgerlich für die Hausfrau, ein kleiner Makel für den bewundernden Besucher, aber nichts, was die Funktion beeinträchtigen würde. Wenn sich die Mauser allerdings über das ganze Jahr ausdehnt, wie es bei meinem Rechner der Fall ist, dann sollte man allerdings etwas dagegen tun!
Reiben Sie die befallenen Stellen (hauptsächlich Tastatur und Gehäusedeckel, manchmal aber auch sogar Peripheriegeräte und sogar Zuleitungskabel) sorgfältig mit etwas Franzbranntwein ab und wischen Sie mit einem desinfizierten Fensterleder nach. Sie könnten natürlich auch den ganzen Rechner in eine Wanne mit lauwarmem Wasser tauchen, aber dann funktioniert er vielleicht nicht mehr bei 4 MHz. Falls ein Leser bereits einen anderen bewährten Kniff kennt, bitte ans Journal einsenden unter dem Stichwort "Sadist".
In diesem Sinne Ihr NASCOMPL

Video-Kamera-Interface

von Günter Böhm

Begonnen hat alles vor einigen Jahren, als ich mit meinem kleinen Sohn einen Freizeitpark besuchte. Dort konnte man sich ein T-Shirt mit seinem Portrait bedrucken lassen, das zuvor mit einer Video-Kamera aufgenommen wurde. Diese Möglichkeit des Einsatzes eines Computers hat mich damals schon fasziniert. Vor einiger Zeit hat Peter Bentz in unserem Journal ein Beispiel gezeigt, wie er mit seinem Plotter und einer Fotodiode ein Bild abtastet und als Grauwerte vom Fernschreiber ausdrucken läßt. Diese Möglichkeit schien mir sehr umständlich, und ich erinnerte mich wieder an die Video-Kamera. So habe ich mir mal oberflächlich die Signale meiner Kamera (ein einfaches Modell, das zur Beobachtung von Parkplätzen diente) auf dem Oszilloskop angesehen, konnte aber mit dem Geflacker auf der Röhre nichts anfangen.

Der Auslöser war nun das kleine Büchlein "KW-Amateurbildfunk SSTV und FAX" von H.J. Pietsch (RPB Taschenbuch 154, Franzis-Verlag), das mir ein Leser zuschickte. Einige Kapitel beschrieben sehr schön die Umsetzung eines Fernsehbildes in digitale Informationen. Das war genau, was ich brauchte. Noch fehlende Informationen holte ich mir aus einem weiteren Buch über Fernsehtechnik ("Fernsehtechnik ohne Ballast" von Otto Limann, Franzis-Verlag), und dann machte ich mich an die Planung für ein Video-Kamera-Interface.

Die folgenden Ausführungen vollziehen meine Überlegungen nach und sollen kein Lehrwerk für Laien sein. Mir helfen sie, die Schaltung und die Programme auch noch in einem Jahr zu verstehen, wenn ich wieder aus der "Materie herausgewachsen" bin. Für manche Leser können sie eine Hilfe sein, die Gedanken nachzuvollziehen und die Schaltung möglicherweise sogar zu verbessern. Experten, denen die Grundlagen geläufig sind, empfehle ich, diesen Text einfach zu überblättern.

Signale der Kamera

Ein Fernsehbild nach der CCIR- Norm ist aus 625 Zeilen aufgebaut, von denen jede 833 Bildpunkte enthält. Die Bildfrequenz beträgt 25 Bilder pro Sekunde. (Dadurch ergibt sich eine Zeilenfrequenz von 15625 Hz). Um das Flimmern der Bilder zu reduzieren, werden sie in zwei Teilbilder zerlegt, die mit einer Frequenz von 50 Hz aufeinander folgen. Dabei enthält das erste Teilbild die ungeraden Zeilenzahlen, und das zweite die geraden. Jedes Teilbild (Raster) enthält 312 (einhalb) Zeilen.

Um den Bildschirm und die Kamera synchron laufen zu lassen, gibt die Kamera für jeden Raster- und Zeilenwechsel einen Synchronisationsimpuls ab.(Dies ist etwas vereinfacht. Tatsächlich bestehen obengenannte Impulse jeweils aus mehreren Einzelimpulsen. Dies kann aber für Funktion und Verständnis der Schaltung vernachlässigt werden).

Der Ablauf der Signale aus der Kamera sieht nun folgendermaßen aus:

1. Rasterimpuls (50 Hz)
2. Zeilenimpuls (15625 Hz)
3. Bildinhalt (Analogwerte zwischen Min. und Max.)

Wenn Sie einmal die Geschwindigkeit durchrechnen, mit der die Kamera die einzelnen Bildpunkte ausgibt, kommen Sie auf ca 5 MHz. Das liegt weit oberhalb der Grenze, die es erlauben würde, die analoge Bildpunktinformation zu digitalisieren und im Rechner abzuspeichern.

Blockschaltung

Auch die Anzahl der Bildpunkte pro Fernsehbild übersteigt wohl die Kapazität unseres Rechners. Deshalb muß die Schaltung zwei Bedingungen erfüllen:

1. Reduzierung der Punktanzahl
2. Reduzierung der Auslesefrequenz

Punkt 1 wird leicht erfüllt, indem man die Geschwindigkeit beim Ein- und Auslesen der Bildinformation der Zeile unter 5 MHz legt. Es kann ebenso nur ein Teil der Zeile erfaßt werden (man speichert z.B. nur die ersten 256 Punkte). Eine weitere Reduzierung ergibt die Beschränkung auf nur ein Halbbild und hiervon jede 2. Zeile.

Punkt 2 läßt sich erreichen, indem man von einem Fernsehbild nur eine Zeile speichert und sich dann zur Verarbeitung Zeit läßt. Die nächste Zeile wird aus einem beliebigen späteren Teilbild ausgelesen. Dies setzt natürlich die Verarbeitung stehender Bilder voraus. Das scheint mir aber kein Nachteil; ich wüßte im Augenblick nicht, wie ich bewegte Bilder im Rechner verarbeiten sollte.

Diese Überlegungen führen zu folgendem Ablaufplan: 1. Zweites Teilbild erkennen

2. n-te Zeile erkennen (einstellbar von 1 bis 156)

3. Bildinformation hardwaremäßig speichern (bis maximal 5 MHz)

4. Bildinformation in Computergerechter Geschwindigkeit auslesen und verarbeiten

5. Wenn Halbbild noch nicht vollständig erfaßt, dann n=n+1 und zurück zu 1.

Dazu müssen als Hardware die Komponenten zur Verfügung stehen, die im folgenden Blockschaltbild wiedergegeben sind.

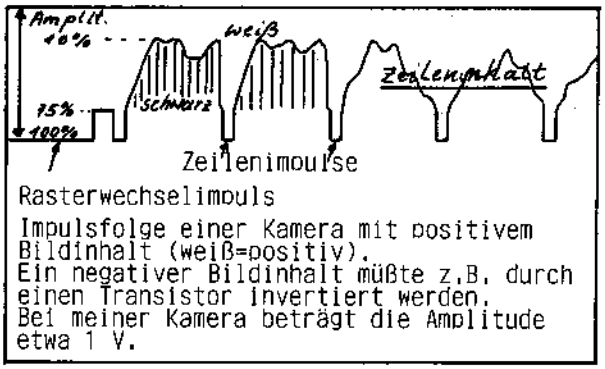
Der Computer stellt also über PIO die Zeilennummer ein (von 1 bis 156). Beim zweiten Halbbild wird der Zähler freigegeben und zählt mit jeder 2. Zeile rückwärts. Nach der eingestellten Anzahl wird ein Impuls ausgegeben (mit einem Monoflop auf 64 usec festgelegt, was etwa der Zeit entspricht, die man zum Einlesen einer Zeile benötigt). Dieser Impuls läßt einen Oszillator starten, der mithilfe eines Zählers ein RAM "hochzählt", das die digitalisierten Bildpunkte einliest. (Aus dem Blockschaltbild ist nicht ersichtlich, daß der RAM- Zähler nach 256 Takten automatisch stoppt).

Der Rechner schaltet dann den Multiplexer (aus Gattern bestehend) auf den Auslesetakt um, den er selbst liefert. Nach dem Auslesen und Abspeichern des Zeileninhalts wird die Zeilennummer erhöht und der Multiplexer wieder auf Einlesetakt umgeschaltet. Die nächste Zeile kann eingelesen werden.

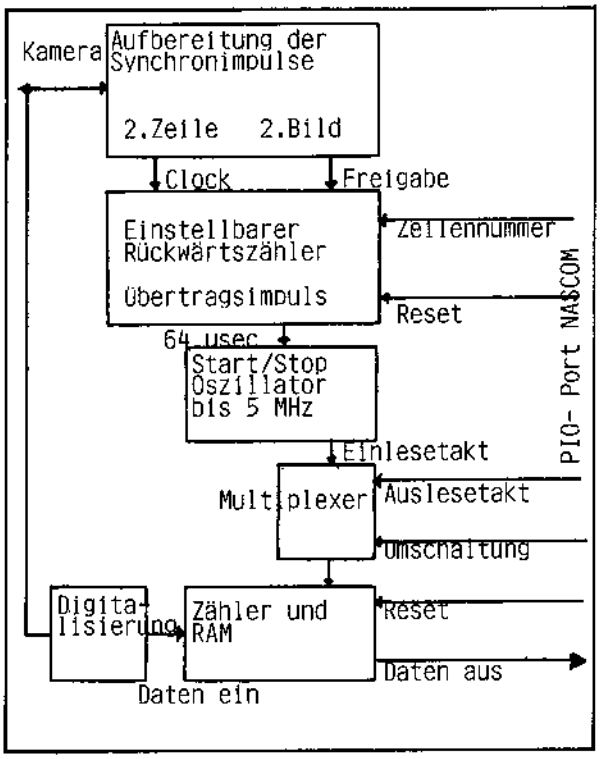
Wie bereits erwähnt, kann der Zeileninhalt in Grauwerten abgespeichert werden, wenn man die Analogspannung durch A/D-Wandler entsprechend umsetzt. Da mir aber mit einem Drucker bei entsprechender Auflösung und auch auf dem Bildschirm ohne dazu notwendige Hardwareerweiterung keine Möglichkeit gegeben ist, Grauwerte darzustellen, kann ich mir auch den Aufwand zur Speicherung sparen. Die Bildinformation soll also nur schwarz/weiß verarbeitet werden.

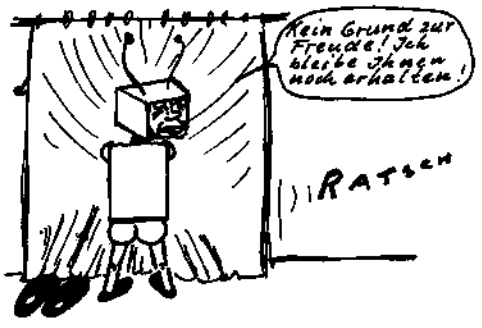
Die Digitalisierung der Zeileninhalte besteht also nur darin, daß mittels eines Komparators zwischen zwei Helligkeitswerten unterschieden wird. Ist der Punkt dunkler als der einstellbare Grenzwert, kippt der Komparator, und damit die gespeicherte Information, auf high. Ist er heller, wird ein low gespeichert. Dadurch entstehen sehr kontrastreiche Bilder, die vielleicht nicht nach jedermanns Geschmack sind. Wie weit man da Kompromisse eingehen muß, um Hardware- und Softwareaufwand in Grenzen zu halten, kann wohl nur im Versuch festgestellt werden.

wird fortgesetzt



Impulsfolge einer Kamera mit positivem Bildinhalt (weiß=positiv).
Ein negativer Bildinhalt müßte z.B. durch einen Transistor invertiert werden.
Bei meiner Kamera beträgt die Amplitude etwa 1 V.





Seite(n) für Einsteiger

von Günter Böhm

NASBUG/NASSYS

Immer noch geistern Unmengen von Programmen herum, die ursprünglich für das Betriebssystem NASBUG T2 oder T4 geschrieben wurden. Die meisten Leser haben aber inzwischen auf NASSYS umgestellt, und da verwundert es nicht, daß der Wunsch geäußert wird, doch einmal zu erklären, wie man ein Programm für das neue Betriebssystem umschreibt. Dies soll hiermit geschehen. Was sind eigentlich die grundsätzlichen Unterschiede zwischen den Betriebssystemen NASBUG und NASSYS?

1. Die NASBUG-Programme können bei C50 beginnen (und tun es auch meist, um die Kapazität des Speichers der Grundversion NASCOM1 voll auszunutzen). NASSYS benötigt aber selbst einen Speicher bis C7F, weshalb die Programme erst ab C80 beginnen können. Ein NASBUG Programm muß also zunächst verschoben werden. Wer einen Disassembler besitzt (NASDIS), kann den Maschinencode (das sind diese unendlichen Reihen von Ziffern und Buchstaben) disassemblieren und dadurch in eine lesbarere Form umwandeln. Der so entstandene Assemblertext kann mit Hilfe eines Assemblers (ZEAP oder ASM) durch einfaches Umschreiben der Ursprungsadresse (ORG=origin=Ursprung) auf eine beliebige Adresse gelegt werden. Noch schöner ist es selbstverständlich, wenn ein Programm schon im Assembler vorliegt. Die Möglichkeit, relativ einfach Veränderungen an einem Programm vorzunehmen, ist auch der Grund, weshalb wir Programme bevorzugt in dieser Form veröffentlichen. Was macht man aber, wenn man weder Disassembler noch Assembler besitzt? In ersterem Fall bleibt nichts anderes übrig, als in mühsamer Kleinarbeit unter Benutzung des Z80 Manuals den Maschinencode Stück für Stück in Assembler zu übersetzen. Die einfachste Art, das Programm dann per Hand zu verschieben, ohne unnötige Fehler zu machen, wäre dann wohl, vor jede Adresse eine 1 zu schreiben, Dadurch ist das Programm, was seine Lage im Speicher betrifft, mit NASSYS lauffähig.

Zum besseren Verständnis hier ein praktisches Beispiel:

Maschinencode (Hexdump)

```
0C50  3E FE
0C52  32 03 0F
0C55  3E 05
0C57  32 07 0F
0C5A  EF 1E 00
0C5D  21  5F 0F
0C60  11 CA 0B
0C63  01 0C 00
0C66  ED B0
```

0C68 CD 01 0D (Bis auf die letzte Zeile ist das Hexdump dem Morsetrainer von Herrn Toss entnommen. Die weiteren Beispiele stammen aus demselben Programm).

Disassemblierter Text:

```
0C50  LD A,-2
```

0C52 LD (F03),A; (Wer den Original-Quelltext besitzt liest hier LD (Klang+1),A. Durch den Namen, der der Adresse gegeben wurde (Label), wird das Programm verständlicher. Man weiß dann sofort, daß diese Speicheradresse etwas mit der Tonausgabe zu tun haben muß).

```
0C55  LD A,5
0C57  LD (F07),A
```

0C5A RST 40 ; (Die RESTART Befehle springen an eine bestimmte Adresse im Monitor. Zum Glück sind sie bei allen NASCOM Betriebssystemen gleich und bereiten keine Schwierigkeiten. RST 40 ruft die Stringausgabe auf den Bildschirm auf (PRS=Print String).

0C5B 1E (T4 Clear Screen; das lautet bei NASSYS 0C. Unterschiedlich ist auch 1F - New Line, das bei NASSYS 0D heißen muß. Alle anderen Bildschirmsteuerbefehle sind gleich).

0C5C 00 Ende des Strings. ZEAP würde beide Befehle als DEFW (Definiere Wort) 1E darstellen.

```
0C5D  LD HL,F5F (im Original LD HL,Titel)
0C60 LD DE,0BCA
0C63  LD BC,12
0C66 LDIR
0C68  Call 0D01 (rufe Unterprogramm)
```

Durch die Eingabe ORG L1000 würde der Assembler die Adressen automatisch verschieben:

```
0C50 = 1000
0C52 = 1002
0C55 = 1005 und so weiter.
```

Die vorgeschlagene Änderung per Hand ergäbe:

```
0C50 = 1C50
```

0C52 = 1C52 und so weiter (man denkt sich eben immer die EINS dazu).

So geschieht das auf dem Papier. Liegt das Hexlisting auf Cassette vor, so kann es eingelesen und durch einen "COPY"-Befehl leicht verlegt werden. Im Beispiel:

```
CC50 1C50 XXXX
```

wobei XXXX die Länge des Programms (in Hex) ist.

Das Programm hat nun wohl eine neue Startadresse, lauffähig ist es hier allerdings noch nicht, wenn es direkte Sprünge oder Unterprogrammaufrufe enthält. Das Programm muß nun also nach solchen Befehlen durchforscht werden. Die Adressen werden in oben beschriebener Weise abgewandelt.

In unserem Beispielprogramm liegt ein Aufruf bei C68 vor. Das Unterprogramm liegt innerhalb des eigentlichen Programm-Speicherbereichs und wird folgendermaßen geändert:

1C68 CD 01 1D

Liegt die aufgerufene Adresse außerhalb, so wird (hauptsächlich bei T2/T4-Programmen) mit größter Wahrscheinlichkeit ein Unterprogramm aus dem Betriebssystem vorliegen. Bei Verwendung des gleichen Systems, kann die Adresse stehen bleiben, beim Umschreiben auf einen anderen Monitor muß die Adresse geändert werden oder (wie bei NASSYS) ein indirekter Aufruf eingetragen werden. Mehr zu den monitoreigenen Unterprogrammen im nächsten Heft.

Die Adressen für direkte Sprünge werden als nächstes modifiziert. So wird aus JP D80 - JP 1D80 oder aus JP Z F00 - JP Z 1F00.

Wird eine Adresse mit einem Wert geladen, so ist auch meist eine Änderung notwendig. In unserem Beispiel:

1C52 : 0F03 liegt innerhalb des Programms und wird geändert (32 03 1F).

1C57 : 0F07 liegt ebenfalls innerhalb. Die Änderung lautet 32 07 1F.

Ebenso verfährt man bei 1C5D. Bei 1C63 liegt wahrscheinlich ein Zahlenwert vor, der nicht geändert werden darf, denn 000C liegt nicht innerhalb des Programms und auch nicht im Arbeitsspeicher des Monitors (hier könnten z.B. die Adressen für ARG1 bis ARG3 mit Werten geladen werden).

1C60 : 0BCA liegt ebenfalls außerhalb des Programms. Es ist eine Adresse aus dem Bildwiederholspeicher (Bildschirm 080A bis 0BF9) und wird natürlich beibehalten.

Man sieht, daß etwas Überlegung dazugehört, um festzustellen, wann es sich um Adressen aus dem Programmbereich handelt, die umgeändert werden müssen, und wann man es mit festen Werten, Adressen aus dem Monitor etc. zu tun hat. Dies ist auch der Grund, warum man kein Programm schreiben kann, das Ihnen die lästige Arbeit des Anpassens erspart. Der Rechner kann ein Problem eben nicht als Ganzes überblicken.

Das größte Problem dürfte dabei wohl das Erkennen von Tabellen innerhalb eines Programms sein, das heißt von Werten, die beim Verschieben nicht geändert werden dürfen.

In unserem nächsten Beispiel ist ab 1F5F der Text für einen Programmtitel abgespeichert. (in 1C5D wird darauf hingewiesen). Solche Tabellen erkennt man durch das dauernde Erscheinen von ASCII-Codes oder die Folge von Befehlen, die einfach keinen Sinn ergeben. So z.B. LD H,D - LD H,A; allein schon das Aufeinandertreffen dieser Codes, von denen der zweite die Funktion des ersten aufhebt, macht deutlich, daß es sich hier wohl um Werte einer Tabelle handelt. (Programmfehler sollten wir einmal ausschließen).

Vergleichen Sie diese Ausführungen mit den Artikeln im Journal 6/81 über den "Relocator" und versuchen Sie einmal, ein Programm anhand obiger Tips zu verschieben.

Nochmals zusammenfassend eine mögliche Verfahrensweise

1. Programmstart und Speicherbereich auf ähnliche Adresse verschieben (z.B. durch Addition von 1000)
2. CALL- und JUMP-Aufrufe untersuchen; wenn innerhalb des Programms, dann verändern.
3. LOAD-Befehle untersuchen; wenn wahrscheinlich Adressen des Programms geladen werden, dann verändern.
4. Bei obigen 3 Punkten darauf achten, daß keine Tabelle vorliegt.

Bis zum Erscheinen des nächsten Heftes sind Sie sicher schon ganz firm in der Anwendung dieser Kniffe. Dann soll es direkt an die Anpassung der Programme an die verschiedenen Betriebssysteme gehen.

Funktionstaste-Software Repeat

von Gustav W. Delius

Der Sinn dieses kleinen Programms ist es, dem Nascom-Benutzer möglichst viel Tipparbeit zu ersparen. Es bedient sich dazu zweier Methoden:

1. Funktionstaste -- Wenn GRAPH 1 getippt wird, erscheint nicht das übliche Grafikzeichen, sondern eine vorher festgelegte Zeichenkette. Diese Zeichenkette wird vom Nascom so behandelt, als würden die Zeichen einzeln auf der Tastatur gedrückt. Da die Zeichenkette jedes beliebige Zeichen enthalten darf, also auch

ÖENTERÜ, ist es zum Beispiel möglich, mit GRAPH 1 mehrere Befehle ausführen zu lassen. Zum Definieren der Zeichenkette springt man zur Routine FUNK mit E D20 und gibt die Zeichenkette ein, eingerahmt von einem beliebigen Begrenzungszeichen.

2. Software Repeat -- Ein gedrücktes Zeichen wird solange wiederholt, wie gleichzeitig die ÖCHÜ Taste gedrückt wird. Dieses Prinzip kennt man z.B. vom Apple II. Es funktioniert anders, als der sogenannte Auto-Repeat wie man ihn von Schreibmaschinen her gewohnt ist. Eine Routine für den letzteren ist von C. Rau im Nascom Journal 11/12 1981 veröffentlicht worden.

Das Programm wird mit E CC0 initialisiert. Dabei kreiert es einen neuen Inputtable. Dies ist schon in dem oben erwähnten Artikel von Christoph Rau erklärt.

Das Programm ist für NAS-SYS 1 geschrieben, läuft aber auch auf anderen Monitoren, die in soweit kompatibel sind, dass sie Arbeitsspeicheradressen und Routinennummern gleich benutzen. Es ist 76H Bytes lang und kann an beliebiger Stelle im Speicher laufen. Zusätzlich braucht es 3 Byte Arbeitsspeicher und Raum für die Zeichenkette. Zur Anpassung an sein System kann der Benutzer die im Assemblerlisting unter ***VARIABLEN*** stehenden Werte verändern.

Change
von Constantin Olbrich

Dieses Programm ersetzt eine beliebige Zeichenkette im Quellfile des ZEAP 2.0 durch eine andere Zeichenkette beliebiger Länge. Nach dem Aufruf des Programms mit EC80 wird eine Zeile mit folgendem Format eingegeben:

/String1/String2/ (CR)

wobei '/' ein beliebiges Zeichen (Delimiter) außer Space ist, sowie String1 die zu ersetzende Zeichenkette und String2 die einzusetzende Zeichenkette ist. Wer sein ASM ZEAP 2.0 mit dem ':' Befehl versehen hat, (beliebiges NAS-SYS Kommando ist nun gültig) arbeitet weiter im Assembler, was sicher wesentlich eleganter ist. Übrigens ist das Argument vom Execute Befehl 'OC80' nur beim Erstaufruf von CHANGE nötig, sonst genügt:

":" (CR) "E" (CR)

Falls das erste Zeichen der Eingabezeile ein Leerzeichen ist, erfolgt ein Rücksprung (Warmstart) in den ZEAP 2.0. Abschluß der Zeile ist CR. Es kann danach sofort eine weitere Eingabe (neuer CHANGE Befehl) erfolgen, oder CR beendet mit nun leerer Zeile den CHANGE-Modus.

Da sich die Länge des Quellfiles verändern kann, erfolgt automatisch eine Korrektur der EOF Marke 'Free' in der Überschriftzeile des ZEAP beim Rücksprung. Wichtig zu wissen ist, daß beliebig viele Ersetzungen pro Zeile möglich sind, z.B.:

/*/$/

würde alle im File vorkommenden '*' in Dollars umwandeln. Wird aus Versehen der dritte Delimiter vergessen, (was mir häufig passiert) geschieht in der Regel gar nichts, es sei denn der eingangs definierte Delimiter steht im Video Ram in den nächsten 45 Bytes, was allerdings äußerst unwahrscheinlich ist. Weiterhin ist es möglich, den String2 wegzulassen, um z.B. gezielte Löschungen vorzunehmen, z.B.:

/Name//

löscht alle "Name" im File. Anwendung findet das Programm z.B. beim disassemblierten Quellprogramm, um obskuren Labels wie L1EF9 einen sinnvollen Namen, z.B. "START", zu geben.

Rom Basic V. 4.7-
zwei Versionen
von Gerhard Wilharm

Die im NASCOM-Journal, Heft 1/82, von Günter Böhm diskutierten zwei ROM BASIC-Versionen unterscheiden sich dadurch, daß man der "anderen" Version den direkten Sprungbefehl C3 03 E0 (JP E003) vorangesetzt hat und sich somit die nachfolgenden Bytes um drei Adressen nach hinten verschieben. Warum diesen scheinbar unsinnigen "Hopser" am Anfang?

Die Lösung ist in der Hardware des NASCOM-2 zu suchen, und zwar in den Schaltungsteil, der die "Jump on Reset"-Funktion bewerkstelligt. Diese Funktion ergibt die Möglichkeit, daß nach einem Reset jede beliebige 4K-Grenze des Speicherbereichs den "Nullpunkt" des Systems darstellen kann (einstellbar mit 4 DIL-Schaltern), also auch unser heißgeliebtes ROM BASIC auf Adresse E000. Damit diese Geschichte aber richtig funktioniert, muß das Programm, das per Reset angesprochen werden soll, als ersten Befehl einen direkten Sprung (also C3 XX XX) aufweisen. Der Programmzähler kann sonst nicht richtig "einrasten" und stürzt nach dem ersten Befehl ab. Und wie funktioniert das Ganze ?

Nach einem Reset macht die CPU das einzig Richtige, sie gibt 0000 auf den Adreßbus.

Leider erzeugt sie unter anderem auch ein M1-Signal, welches mit Hilfe von zwei Flipflops (IC 16) den Adreßmultiplexer (IC 2) auf die 4 DIL-Schalter umsteuert und die obersten 4 Adreßleitungen A12 bis A15 von der CPU abtrennt.

Der Effekt: Die CPU wird arglistig getäuscht, denn statt dem Byte der Adresse 0000 muß sie jetzt das Byte der eingestellten 4K-Grenze schlucken, und dieses Byte heißt in jedem Fall C3. "Aha", sagt sich die CPU, "ich soll noch zwei Bytes holen und meinen Programmzähler damit füttern." Das macht sie auch prompt, und schon arbeitet sie in dem Programmteil, der mit den DIL-Schaltern eingestellt wurde. Rein zufällig erschein genau zu diesem Zeitpunkt das scheinheilige M1-Signal wieder und schaltet den Adreßmultiplexer auf die Leitungen A12 bis A15 der CPU zurück.

(Das Umschaltsignal erscheint wirklich nur zweimal. Es ist mit dem mono-getriggerten Reset-Impuls verknüpft. Siehe N2-Schaltplan IC 12a).

Wenn das so ist wie beschrieben, dann müßten ja alle uns vertrauten Firmware-Programme, die auf einer 4K-Grenze beginnen, diesen direkten Sprungbefehl enthalten. Schauen wir doch einfach mal nach:

Programm	Startadresse	1.Befehl
ROM BASIC	E000	C3 03 E0
(falls kein Minderheiten-BASIC)		
ZEAP 2.0	D000	C3 36 DB
DEBUG	C000	C3 03 C0
Tool-Kit	B000	C3 03 B0
UNICON 1.4	A000	C3 0B A4
(Herr Lotter, gewußt wie oder Zufall?)		
Menue-Progr.	9000	21 DF 6F
(Nanu? Lieber Herr Maurer)		

Hoffentlich sind alle Minderheite-BASIC-Besitzer nicht allzu traurig darüber, daß bei ihren Versionen nicht sofort nach Reset die Microsoft-Reklame erscheint, sondern daß sie erst noch die J-Taste drücken müssen.

Autostart von Rom-Basic Vers. 4.7

von Peter Urban

Für NASCOM2 MIT NAS-SYS1. Durch längere Untersuchungen mit Disassembler, Breakpoint und Debugger ist es mir gelungen eine Routine zu finden, die ein Basic-Programm läd und es startet. Da es nachweislich verschiedene Versionen des Rom-Basics gibt empfiehlt es sich mit dem entscheidenden Teil zu beginnen.

Als erstes scheibt man ein kleines Basic-Programm, macht einen Reset und führt das Maschinenprogramm ab 0D36 aus.

Diese Routine schreibt in den Textbuffer das Token Run und springt dann nach E816.

Prinzipiell lässt sich auf diese Art und Weise jedes Basic-Kommando ausführen aber (jetzt kommt der Pferdefuss) nur nach dem Befehl Run wird ein Warmstart ausgefürt, der den Stackpointer und den Workspace des Basic setzt.

Nach der Ausführung anderer Befehle stürzte das Programm früher oder später ab.

Wurde Ihr Programm ausgeführt dann brauchen Sie jetzt einen durch die Cass.-LED ferngesteuerten Kassettenrecorder.

Legen Sie eine Kassette mit Basic-Programm ein, machen sie einen Basic-Kaltstart und starten Sie die Maschine bei 0DBF. Hat es nicht funktioniert haben Sie noch eine Chance. Ändern Sie folgende Zeilen:

```
220 CRD2   CALL IN1
270        CALL IN1
290 kann entfallen
300        CALL IN2
340 kann entfallen
350        CALL IN4
460        CALL IN5
```

Die entspechenden Routinen finden Sie im Anhang. Jetzt muss noch der Basic-Kaltstart getestet werden. Dazu löschen Sie den Workspace von 1000 bis 1100 mit dem Copy-Befehl und legen die

Kassette mit dem Basic-Programm ein. Starten sie jetzt bei 0D44 die Maschine. Erscheint die übliche Meldung mit dem maximal zur Ferfügung stehenden Speicherraum und läuft das Programm nach korrektem Einlesen nicht, muss ich passen.

Die Praxis sieht folgendermassen aus:

Man initialisiert die USR-Funktion mit Doke 4100,3328 und kann dann mit A=USR(n) vom Basic aus jedes beliebige File starten. Wenn n=0 ist wird das nächste File geladen. Wenn n=65 ist wird File A geladen, ist n=66 wird File B geladen ect. Bei Label CRD0 steht der Hexwert des Filenamens dann in Register D.

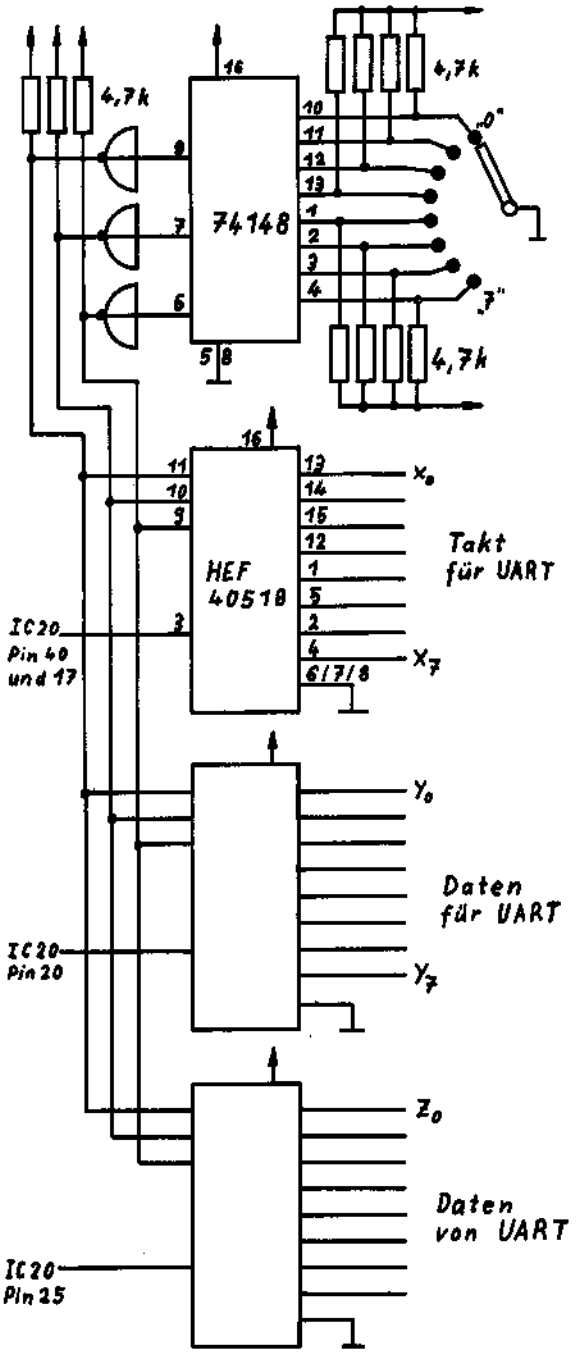
Eine weitere Möglichkeit ist das Programm mit G(enerate) D00 DC4 D44 am Anfang einer Kassette abzuspeichern, gefolgt von einem Basic-Programm. Wenn der Kassettenrecorder beim Abschalten der LED nicht sofort abstellt (Verzögerung), kann man nach dem Einschalten des

Computers mit dem Read-Kommando das komplette System hochfahren. Ist der Recorder voll fern= bedienbar, ist der Aufbau eines "NAS-KOS" in Basic nicht mehr schwer, was ich der Fantasie der Leser überlassen will.

Seriellport Erweiterung

von Peter Urban

Will man mehrere Geraete an der UART betreiben die auch noch unterschiedliche Pegel und Baud= raten haben, ist man gezwungen dauernd die Geraete umzustecken und Baudraten zu aendern. Die untenstehende Schaltung schafft hier Ab= hilfe. Der Prioiritaetsenkoder SN74148 liefert die Binaeradresse fuer die Multiplexer/Demul= tiplexer HEF 4051B. An den Eingaengen X0-X7 sollte der entsprechende Takt fuer die UART anliegen (z.B. IC31 Pin 1 fuer 2400 Baud ect.). Die aufbereiteten Daten zur UART werden an die Eingaenge Y0-Y7 angelegt (z.B. IC29b Pin 12 fuer den Kassettenrekorder). Die Ausgaenge Z0-Z7 leiten die Daten zu den entsprechenden Treibern (z.B. IC33b Pin 5 fuer den Kassetten= rekorder). Durch hinzufuegen weiterer Multi= plexer kann z.B. die Information Cass.-LED "ON" umgeschaltet werden, oder Geraete ein und aus= geschaltet werden. Auch kann man wenn an einer PIO noch drei Bit frei sind das ganze per Software steuern.



Suche NASCOM-Journal Nr. 8-12/80, 9-10/81 und 1/82.
Rolf Kottke
[redacted]

Verkaufe V.24 Terminal
bedingt einsatzfähig, da spezielle Leitungs= prozedur. Beinhaltet schnellen S/W Monitor (12"), Keyboard mit Hall-Schaltern (kein ASCII-Code). Schaltbeispiel für Umcodierung und alle anderen Unterlagen sind vorhanden. Zudem Netzteil alle Spannungen (5V/10A!) und schönes Gehäuse. VB DM 350.-
Dieter Oberle [redacted]
Tel. [redacted]

2 Digitale Cassettenrecorder (Normalcassette)
Aufzeichnungsformat NRZ, digital steuerbar über TTL. DM 150.- pro Stück.
Dieter Oberle [redacted]
Tel. [redacted]

Suche Matrixdrucker (Normalpapier)
schnell, leise, grafikfähig u. preisgünstig
Günter Böhm [redacted]

Suche NASCOM2 Dokumentation auf Deutsch.
Georg Assmann [redacted]

```
                0010 ;HALBSCHRITT-FORMATIERUNG
                0015 ;FUER PRAXIS-30/35
                0020 ;V 1.1    G.K. 5.9.82
0C80            0040        ORG  £C80
0C80 E5         0050 HFORM  PUSH HL
0C81 D5         0060        PUSH DE
0C82 C5         0070        PUSH BC
0C83 F5         0080        PUSH AF
0C84 2AD70D     0090        LD   HL,(BUFPTR)
0C87 4F         0100        LD   C,A
0C88 ED5BD90D   0110        LD   DE,(COUNT)
0C8C 7A         0120        LD   A,D
0C8D BB         0130        CP   E
0C8E 79         0140        LD   A,C
0C8F 3804       0150        JR   C,HFORM1
0C91 FE21       0160        CP   £21
0C93 382E       0170        JR   C,FORMAT
0C95 FE40       0180 HFORM1 CP   "§
0C97 2815       0190        JR   Z,ABS
0C99 FE21       0200        CP   £21
0C9B 3002       0210        JR   NC,HFORM2
0C9D 3E20       0220        LD   A,£20
0C9F 77         0230 HFORM2 LD   (HL),A
0CA0 23         0240        INC  HL
0CA1 22D70D     0250        LD   (BUFPTR),HL
0CA4 14         0260        INC  D
0CA5 ED53D90D   0270        LD   (COUNT),DE
0CA9 F1         0280 FOREND POP  AF
0CAA C1         0290        POP  BC
0CAB D1         0300        POP  DE
0CAC E1         0310        POP  HL
0CAD C9         0320        RET
0CAE 7A         0330 ABS    LD   A,D
0CAF BB         0340        CP   E
0CB0 3011       0350        JR   NC,FORMAT
0CB2 42         0360 ABS1   LD   B,D
0CB3 CD850D     0370        CALL AUSG
0CB6 21DF0D     0380        LD   HL,BUFFER
0CB9 22D70D     0390        LD   (BUFPTR),HL
0CBC 3E00       0400        LD   A,0
0CBE 32DA0D     0410        LD   (COUNT+1),A
0CC1 18E6       0420        JR   FOREND
0CC3 7A         0430 FORMAT LD   A,D
0CC4 BB         0440        CP   E
0CC5 28EB       0450        JR   Z,ABS1
0CC7 43         0460        LD   B,E
0CC8 CDAD0D     0470        CALL COUNTS
0CCB 7A         0480        LD   A,D
0CCC 93         0490        SUB  E
0CCD CB27       0500        SLA  A
0CCF B9         0510        CP   C
0CD0 3802       0520        JR   C,VERK
0CD2 2008       0530        JR   NZ,VERL
0CD4 47         0540 VERK   LD   B,A
0CD5 0E01       0550        LD   C,1
0CD7 CDC50D     0560        CALL INSERT
0CDA 18D6       0570        JR   ABS1
0CDC 7B         0580 VERL   LD   A,E
0CDD 90         0590        SUB  B
0CDE 3C         0600        INC  A
0CDF CB27       0610        SLA  A
0CE1 0D         0620        DEC  C
0CE2 B9         0630        CP   C
0CE3 3802       0640        JR   C,VERL1
0CE5 2037       0650        JR   NZ,EDIT
0CE7 0C         0660 VERL1  INC  C
0CE8 C5         0670        PUSH BC
0CE9 47         0680        LD   B,A
0CEA 0E03       0690        LD   C,3
0CEC CDC50D     0700        CALL INSERT
0CEF C1         0710        POP  BC
0CF0 78         0720        LD   A,B
0CF1 C5         0730        PUSH BC
0CF2 3D         0740        DEC  A
0CF3 47         0750        LD   B,A
0CF4 CD850D     0760        CALL AUSG
0CF7 3E20       0770        LD   A,£20
0CF9 77         0780        LD   (HL),A
0CFA 23         0790        INC  HL
0CFB 14         0800        INC  D
0CFC C1         0810        POP  BC
0CFD 7A         0820        LD   A,D
0CFE 90         0830        SUB  B
0CFF 47         0840        LD   B,A
0D00 C5         0850        PUSH BC
0D01 4F         0860        LD   C,A
0D02 0600       0870        LD   B,0
0D04 AF         0880        XOR  A
0D05 ED42       0890        SBC  HL,BC
0D07 C1         0900        POP  BC
0D08 11DF0D     0910        LD   DE,BUFFER
0D0B 0E00       0920        LD   C,0
0D0D 7E         0930 VERL2  LD   A,(HL)
0D0E 12         0940        LD   (DE),A
0D0F 13         0950        INC  DE
0D10 23         0960        INC  HL
0D11 0C         0970        INC  C
0D12 10F9       0980        DJNZ VERL2
0D14 ED53D70D   0990        LD   (BUFPTR),DE
0D18 79         1000        LD   A,C
0D19 32DA0D     1010        LD   (COUNT+1),A
0D1C 188B       1020        JR   FOREND
0D1E DF         1030 EDIT   RST  £18
0D1F 4E         1040        DEFB "N
0D20 3E0D       1050        LD   A,£D
0D22 F7         1060        RST  £30
0D23 42         1070        LD   B,D
0D24 D5         1080        PUSH DE
0D25 ED5B290C   1090        LD   DE,(CURPOS)
0D29 21DF0D     1100        LD   HL,BUFFER
0D2C 7E         1110 ED1    LD   A,(HL)
0D2D F7         1120        RST  £30
```

```
OD2E 23          1130          INC  HL
OD2F 10FB        1140          DJNZ ED1
OD31 D5          1150          PUSH DE
OD32 DF          1160 ED2      RST  £18
OD33 7B          1170          DEFB £7B
OD34 FEOD        1180          CP   £D
OD36 2811        1190          JR   Z,ED3
OD38 FEOF        1200          CP   £F
OD3A 2004        1210          JR   NZ,ED21
OD3C 3E5C        1220          LD   A,£5C
OD3E 1806        1230          JR   ED22
OD40 FE5E        1240 ED21     CP   £5E
OD42 2002        1250          JR   NZ,ED22
OD44 3E5D        1260          LD   A,£5D
OD46 F7          1270 ED22     RST  £30
OD47 18E9        1280          JR   ED2
OD49 D1          1290 ED3      POP  DE
OD4A 2A290C      1300          LD   HL,(CURPOS)
OD4D C1          1310          POP  BC
OD4E E5          1320          PUSH HL
OD4F AF          1330          XOR  A
OD50 ED52        1340          SBC  HL,DE
OD52 7D          1350          LD   A,L
OD53 B8          1360          CP   B
OD54 3008        1370          JR   NC ED31
OD56 214000      1380          LD   HL,£40
OD59 EB          1390          EX   DE,HL
OD5A AF          1400          XOR  A
OD5B ED52        1410          SBC  HL,DE
OD5D EB          1420          EX   DE,HL
OD5E E1          1430 ED31     POP  HL
OD5F ED53290C    1440          LD   (CURPOS),DE
OD63 EB          1450          EX   DE,HL
OD64 E5          1460          PUSH HL
OD65 21DFOD      1470          LD   HL,BUFFER
OD68 22D70D      1480          LD   (BUFPTR),HL
OD6B AF          1490          XOR  A
OD6C 32DAOD      1500          LD   (COUNT+1),A
OD6F E1          1510          POP  HL
OD70 7E          1520 ED4      LD   A,(HL)
OD71 F7          1530          RST  £30
OD72 CD800C      1540          CALL HFORM
OD75 2A290C      1550          LD   HL,(CURPOS)
OD78 7D          1560          LD   A,L
OD79 BB          1570          CP   E
OD7A 20F4        1580          JR   NZ,ED4
OD7C 7C          1590          LD   A,H
OD7D BA          1600          CP   D
OD7E 20FO        1610          JR   NZ,ED4
OD80 DF          1620          RST  £18
OD81 55          1630          DEFB "U
OD82 C3A90C      1640          JP   FOREND
OD85 E5          1650 AUSG     PUSH HL
OD86 21DFOD      1660          LD   HL,BUFFER
OD89 7E          1670 AUSG1    LD   A,(HL)
OD8A CD300E      1680          CALL DRUCK
OD8D 23          1690          INC  HL
OD8E 10F9        1700          DJNZ AUSG1
OD90 3EOD        1710          LD   A,£D
OD92 CD300E      1720          CALL DRUCK
OD95 21DBOD      1730          LD   HL,LCOUNT
OD98 7E          1740          LD   A,(HL)
OD99 23          1750          INC  HL
OD9A 3C          1760          INC  A
OD9B BE          1770          CP   (HL)
OD9C 2B          1780          DEC  HL
OD9D 2803        1790          JR   Z,STOP
OD9F 77          1800 AUSG2    LD   (HL),A
ODAO E1          1810          POP  HL
ODA1 C9          1820          RET
ODA2 DF          1830 STOP     RST  £18
ODA3 62          1840          DEFB £62
ODA4 30FC        1850          JR   NC,STOP
ODA6 FEOD        1860          CP   £D
ODA8 20F8        1870          JR   NZ,STOP
ODAA AF          1880          XOR  A
ODAB 18F2        1890          JR   AUSG2
ODAD E5          1900 COUNTS   PUSH HL
ODAE 21DFOD      1910          LD   HL,BUFFER
ODB1 OE00        1920          LD   C,O
ODB3 D5          1930          PUSH DE
ODB4 1600        1940          LD   D,O
ODB6 14          1950 CS1      INC  D
ODB7 7E          1960          LD   A,(HL)
ODB8 FE20        1970          CP   £20
ODBA 2002        1980          JR   NZ,CS2
ODBC OC          1990          INC  C
ODBD 5A          2000          LD   E,D
ODBE 23          2010 CS2      INC  HL
ODBF 10F5        2020          DJNZ CS1
ODC1 43          2030          LD   B,E
ODC2 D1          2040          POP  DE
ODC3 E1          2050          POP  HL
ODC4 C9          2060          RET
ODC5 E5          2070 INSERT   PUSH HL
ODC6 21DFOD      2080          LD   HL,BUFFER
ODC9 7E          2090 INS1     LD   A,(HL)
ODCA FE20        2100          CP   £20
ODCC 2803        2110          JR   Z,INS2
ODCE 23          2120          INC  HL
ODCF 18F8        2130          JR   INS1
ODD1 71          2140 INS2     LD   (HL),C
ODD2 23          2150          INC  HL
ODD3 10F4        2160          DJNZ INS1
ODD5 E1          2170          POP  HL
ODD6 C9          2180          RET
ODD7 DFOD        2190 BUFPTR   DEFW BUFFER
ODD9 2COO        2200 COUNT    DEFB 44,O
ODDB 0036        2210 LCOUNT   DEFB 0,54
ODDD OE30        2220 DRUCK    EQU  £E30
ODDD OC29        2230 CURPOS   EQU  £C29
ODDD 0000        2240 SPSAVE   DEFW O
ODDF 0050        2250 BUFFER   DEFS 80
```

ZEAP Z80 Assembler - Symbol Table

```
OCAEH 0330 ABS          OCB2H 0360 ABS1
OD85H 1650 AUSG         OD89H 1670 AUSG1
OD9FH 1800 AUSG2        ODDFH 2250 BUFFER
ODD7H 2190 BUFPTR       ODD9H 2200 COUNT
ODADH 1900 COUNTS       ODB6H 1950 CS1
ODBEH 2010 CS2          OC29H 2230 CURPOS
OE30H 2220 DRUCK        OD2CH 1110 ED1
OD32H 1160 ED2          OD40H 1240 ED21
OD46H 1270 ED22         OD49H 1290 ED3
OD5EH 1430 ED31         OD70H 1520 ED4
OD1EH 1030 EDIT         OCA9H 0280 FOREND
OCC3H 0430 FORMAT       OC80H 0050 HFORM
OC95H 0180 HFORM1       OC9FH 0230 HFORM2
ODC9H 2090 INS1         ODD1H 2140 INS2
ODC5H 2070 INSERT       ODDBH 2210 LCOUNT
ODDDH 2240 SPSAVE       ODA2H 1830 STOP
OCD4H 0540 VERK         OCDCH 0580 VERL
OCE7H 0660 VERL1        ODODH 0930 VERL2
```

ZEAP Z80 Assembler - Source Listing

```
                0010 ;LADEN EINES BASIC-
                0020 ;PROGRAMMS MIT
                0030 ;AUTOSTART
                0040 ;505355820627/3.2

OD00            0060        ORG  £D00
OD00 E98B       0070 USRX   EQU  £E98B
OD00 FD56       0080 INP1   EQU  £FD56
OD00 F58E       0090 INP2   EQU  £F58E
OD00 F210       0100 INP3   EQU  £F210
OD00 F594       0110 INP4   EQU  £F594
OD00 FE91       0120 INP5   EQU  £FE91
OD00 1062       0130 TBUF   EQU  £1062
OD00 E816       0140 RUNB   EQU  £E816

OD00 CD8BE9     0160        CALL USRX
OD03 53         0170        LD   D,E
OD04 D5         0180 CRD0   PUSH DE
OD05 DF         0190        RST  £18
OD06 5F         0200        DEFB £5F
OD07 0603       0210 CRD1   LD   B,3
OD09 CD56FD     0220 CRD2   CALL INP1
OD0C D6D3       0230        SUB  £D3
OD0E 20F7       0240        JR   NZ,CRD1
OD10 05         0250        DEC  B
OD11 20F6       0260        JR   NZ,CRD2
OD13 CD56FD     0270        CALL INP1
OD16 F5         0280        PUSH AF
OD17 218EF5     0290        LD   HL,INP2
OD1A CD10F2     0300        CALL INP3
OD1D F1         0310        POP  AF
OD1E F5         0320        PUSH AF
OD1F F7         0330        RST  £30
OD20 2194F5     0340        LD   HL,INP4
OD23 CD10F2     0350        CALL INP3
OD26 C1         0360        POP  BC
OD27 F1         0370        POP  AF
OD28 FE00       0380        CP   £00
OD2A 2805       0390        JR   Z,CRD3
OD2C B8         0400        CP   B
OD2D F5         0410        PUSH AF
OD2E 20D7       0420        JR   NZ,CRD1
OD30 F1         0430        POP  AF
OD31 DF         0440 CRD3   RST  £18
OD32 5F         0450        DEFB £5F
OD33 CD91FE     0460        CALL INP5
OD36 216210     0470        LD   HL,TBUF
OD39 3600.      0480        LD   (HL),0
OD3B 2B         0490        DEC  HL
OD3C 3689       0500        LD   (HL),£89
OD3E 2B         0510        DEC  HL
OD3F 3620       0520        LD   (HL),£20
OD41 C316E8     0530        JP   RUNB

                0550 ;BASIC KALTSTART.
                0560 ;EINSPRUNG FUER
                0570 ;GENERATE.

OD44 F3         0590        DI
OD45 DD21FFFF   0600        LD   IX,£FFFF
OD49 210010     0610        LD   HL,£1000
OD4C F9         0620        LD   SP,HL
OD4D 3E00       0630        LD   A,0
OD4F 324D10     0640        LD   (£104D),A
OD52 11DFE2     0650        LD   DE,£E2DF
OD55 0663       0660        LD   B,£63
OD57 210010     0670        LD   HL,£1000
OD5A 1802       0680        JR   CST

                0700 ;HILFSSPRUNG
OD5C 18A6       0710 CRD4   JR   CRD0

OD5E 1A         0730 CST    LD   A,(DE)
OD5F 77         0740        LD   (HL),A
OD60 23         0750        INC  HL
OD61 13         0760        INC  DE
OD62 05         0770        DEC  B
OD63 20F9       0780        JR   NZ,CST
OD65 F9         0790        LD   SP,HL
OD66 CDDFE4     0800        CALL £E4DF
OD69 CD81EB     0810        CALL £EB81
OD6C 32AA10     0820        LD   (£10AA),A
OD6F 32F910     0830        LD   (£10F9),A
OD72 215D11     0840 CST1   LD   HL,£115D
OD75 23         0850 CST2   INC  HL
OD76 7C         0860        LD   A,H
OD77 B5         0870        OR   L
OD78 2808       0880        JR   Z,CST3
OD7A 7E         0890        LD   A,(HL)
```

```
OD7B 47        0900         LD   B,A
OD7C 2F        0910         CPL
OD7D 77        0920         LD   (HL),A
OD7E BE        0930         CP   (HL)
OD7F 70        0940         LD   (HL),B
OD80 28F3      0950         JR   Z,CST2
OD82 2B        0960 CST3    DEC  HL
OD83 115C11    0970         LD   DE,£115C
OD86 CD8AE6    0980         CALL £E68A
OD89 38E7      0990         JR   C,CST1
OD8B 11CEFF    1000         LD   DE,£FFCE
OD8E 22AF10    1010         LD   (£10AF),HL
OD91 19        1020         ADD  HL,DE
OD92 225A10    1030         LD   (£105A),HL
OD95 CDBAE4    1040         CALL £E4BA
OD98 2A5A10    1050         LD   HL,(£105A)
OD9B 11EFFF    1060         LD   DE,£FFEF
OD9E 19        1070         ADD  HL,DE
OD9F 11F910    1080         LD   DE,£10F9
ODA2 7D        1090         LD   A,L
ODA3 93        1100         SUB  E
ODA4 6F        1110         LD   L,A
ODA5 7C        1120         LD   A,H
ODA6 9A        1130         SBC  A,D
ODA7 67        1140         LD   H,A
ODA8 E5        1150         PUSH HL
ODA9 21C5E0    1160         LD   HL,£E0C5
ODAC CD10F2    1170         CALL £F210
ODAF E1        1180         POP  HL
ODB0 CDADF9    1190         CALL £F9AD
ODB3 21B7E0    1200         LD   HL,£E0B7
ODB6 CD10F2    1210         CALL £F210
ODB9 316610    1220         LD   SP,£1066
ODBC CDDFE4    1230         CALL £E4DF

               1250 ;LADEN DES ERSTEN
               1260 ;BASIC PROGRAMMS.

ODBF 110000    1280         LD   DE,0
ODC2 1898      1290         JR   CRD4

               1310 ;ANHANG

ODC4 DB02      1330 IN1     IN   A,(£02)
ODC6 17        1340         RLA
ODC7 30FB      1350         JR   NC,IN1
ODC9 DB01      1360         IN   A,(£01)
ODCB C9        1370         RET

ODCC EF        1390 IN2     RST  £28
ODCD 46696C65  1400         DEFM /File /
     20
ODD2 00        1410         DEFB 0
ODD3 C9        1420         RET
```



```
ODD4 EF        1440 IN4     RST  £28
ODD5 20466F75  1450         DEFM / Found/
     6E64
ODDB 0D0A00    1460         DEFB £0D,£0A,0
ODDE C9        1470         RET

ODDF 3E52      1490 IN5     LD   A,£52
ODE1 322B0C    1500         LD   (£0C2B),A
ODE4 DF        1510         RST  £18
ODE5 52        1520         DEFB £52
ODE6 C9        1530         RET
```

ZEAP Z80 Assembler - Symbol Table

```
OD04H 0180 CRD0          OD07H 0210 CRD1
OD09H 0220 CRD2          OD31H 0440 CRD3
OD5CH 0710 CRD4          OD5EH 0730 CST
OD72H 0840 CST1          OD75H 0850 CST2
OD82H 0960 CST3          ODC4H 1330 IN1
ODCCH 1390 IN2           ODD4H 1440 IN4
ODDFH 1490 IN5           FD56H 0080 INP1
F58EH 0090 INP2          F210H 0100 INP3
F594H 0110 INP4          FE91H 0120 INP5
E816H 0140 RUNB          1062H 0130 TBUF
E98BH 0070 USRX
```

ZEAP Z80 Assembler - Source Listing

```
               0005 ;*****************************
               0010 ;*** CHANGE UNTERPROGRAMM ***
               0015 ;*****************************
               0020 ;
               0025 ; 4.7.82  Constantin Olbrich
               0030 ;
               0035 ;  V 2.0
0C80           0040         ORG  0C80H
0C80           0045         ENT

0C80 DF63      0055 START   SCAL INLIN ; Line Input
0C82 1A        0060         LD   A,(DE) ; DE zeigt auf Beginn 1. Z
0C83 FE20      0065         CP   SPACE ; Springe wenn erstes Zeich
0C85 CA03D0    0070         JP   Z,0D003H ; ist nach ASM Warmstart
               0075 ; DE ZEIGT AUF VDU ESTEN DELIMITER
0C88 EB        0080 CHANGE  EX   DE,HL ; HL := Pointer im VDU
0C89 23        0085         INC  HL
0C8A 4F        0090         LD   C,A ; C:= DELIMITER
               0095 ; STRING1 HERE
0C8B 224E0D    0100         LD   (STR1),HL ; (STR1):= START STRING
0C8E 11FFFF    0105         LD   DE,-1
0C91 7E        0110 LIN2    LD   A,(HL)
0C92 23        0115         INC  HL
0C93 14        0120         INC  D
0C94 B9        0125         CP   C
```

```
0C95 20FA      0130         JR   NZ,LIN2
               0135 ;D := LENGHT OF STRING1
0C97 E5        0140         PUSH HL
0C98 FDE1      0145         POP  IY ; IY:= START OF STRING2
0C9A 7E        0150 LIN3    LD   A,(HL)
0C9B 23        0155         INC  HL
0C9C 1C        0160         INC  E
0C9D F5        0165         PUSH AF
0C9E 7B        0170         LD   A,E
0C9F FE2D      0175         CP   MAXSTR ; if len(str2) maxstr then
0CA1 CA420D    0180         JP   Z,ERR ; print error then go to st
0CA4 F1        0185         POP  AF
0CA5 B9        0190         CP   C
0CA6 20F2      0195         JR   NZ,LIN3
               0200 ; E:= LENGHT OF STRING2
0CA8 DD21000F  0205 GTPOIT  LD   IX,WRAM ; ASM Workspace
0CAC DD6E00    0210         LD   L,(IX+0)
0CAF DD6601    0215         LD   H,(IX+1)
0CB2 DD4E02    0220         LD   C,(IX+2)
0CB5 DD4603    0225         LD   B,(IX+3)
               0230 ; HL:= EDITOR START,BC:= EDITOR Maxmem
0CB8 ED43480D  0235         LD   (END),BC ; RETTE Maxmem
0CBC E5        0240         PUSH HL ; PUT EDT START ON STACK
0CBD 010700    0245         LD   BC,7
0CC0 09        0250         ADD  HL,BC ; HL zeigt auf 1. Editorbyt
0CC1 E5        0255         PUSH HL ; PUT EDT START+7 ON STACK
0CC2 3EFF      0260 ENDTST  LD   A,-1 ; FLAG EOF
0CC4 BE        0265         CP   (HL) ; Test auf EOF
0CC5 2869      0270         JR   Z,ENDCHN
0CC7 7E        0275 FIND    LD   A,(HL)
0CC8 DD2A4E0D  0280         LD   IX,(STR1)
0CCC 23        0285         INC  HL
0CCD B7        0290         OR   A ; Zeilenende erreicht ?
0CCE 285C      0295         JR   Z,LINEND
0CD0 DDBE00    0300         CP   (IX)
0CD3 20F2      0305         JR   NZ,FIND
0CD5 DD7E01    0310         LD   A,(IX+1) ; if LEN(Str1)=1 then Ze
0CD8 DDBEFF    0315         CP   (IX-1)
0CDB 280D      0320         JR   Z,INSERT
0CDD 42        0325         LD   B,D ; Lenght OF String1
0CDE 05        0330         DEC  B ; Korrektur
0CDF 7E        0335 FIND1   LD   A,(HL) ; weiter suchen
0CE0 23        0340         INC  HL ; Pointer im File inkrementier
0CE1 DD23      0345         INC  IX ; Pointer im String1 inkrement
0CE3 DDBE00    0350         CP   (IX)
0CE6 20DF      0355         JR   NZ,FIND ; nicht gefunden, dann En
0CE8 10F5      0360         DJNZ FIND1 ; vergleichen bis Ende(Stri
0CEA E5        0365 INSERT  PUSH HL
0CEB E5        0370         PUSH HL
0CEC 2A480D    0375         LD   HL,(END)
0CEF B7        0380         OR   A ; clear Carry
0CF0 C1        0385         POP  BC
0CF1 ED42      0390         SBC  HL,BC ; HL:= Start of String1 imF
0CF3 E5        0395         PUSH HL
0CF4 C5        0400         PUSH BC
0CF5 E1        0405         POP  HL
0CF6 D5        0410         PUSH DE
0CF7 5A        0415         LD   E,D
0CF8 1600      0420         LD   D,0
0CFA B7        0425         OR   A ; clear Carry
0CFB ED52      0430         SBC  HL,DE ; HL=START(STR1)+LENGHT(STR
0CFD 224A0D    0435         LD   (STRT),HL
0D00 D1        0440         POP  DE
0D01 D5        0445         PUSH DE
0D02 1600      0450         LD   D,0
0D04 19        0455         ADD  HL,DE
0D05 D1        0460         POP  DE
0D06 ED534C0D  0465         LD   (ARG),DE
0D0A EB        0470         EX   DE,HL ; DE= Destination
0D0B C1        0475         POP  BC ; BC= Byte Counter
0D0C E1        0480         POP  HL ; HL= Quelle
0D0D DF49      0485         SCAL "I ; Verschieben : Intelligent Co
0D0F FDE5      0490         PUSH IY
0D11 E1        0495         POP  HL ; HL= Quelle
0D12 ED5B4A0D  0500         LD   DE,(STRT) ; DE= Destination
0D16 ED4B4C0D  0505         LD   BC,(ARG) ; BC= Byte Counter
0D1A 0600      0510         LD   B,0 ; Clear LENGHT(STR1)
0D1C AF        0515         XOR  A
0D1D B9        0520         CP   C
0D1E 2802      0525         JR   Z,NONCOP
0D20 DF49      0530         SCAL "I ; Einsetzen : Intelligent Copy
0D22 ED5B4C0D  0535 NONCOP  LD   DE,(ARG)
0D26 2A4A0D    0540         LD   HL,(STRT)
0D29 23        0545         INC  HL
0D2A 189B      0550         JR   FIND ; weitersuchen
0D2C 23        0555 LINEND  INC  HL ; SKIP LINE NUMBER
0D2D 23        0560         INC  HL
0D2E 1892      0565         JR   ENDTST
0D30 E1        0570 ENDCHN  POP  HL ; GET EDT START+7 FORM STACK
0D31 3EFF      0575         LD   A,-1 ; EOF Marke suchen
0D33 BE        0580 END1    CP   (HL)
0D34 23        0585         INC  HL
0D35 20FC      0590         JR   NZ,END1
0D37 B7        0595         OR   A ; clear Carry
0D38 D1        0600         POP  DE ; GET EDT START FROM STACK
0D39 ED52      0605         SBC  HL,DE
0D3B EB        0610         EX   DE,HL
0D3C 73        0615         LD   (HL),E ; LD DE,(HL)
0D3D 23        0620         INC  HL
0D3E 72        0625         LD   (HL),D ; INSERT NEW SOURCE LENGHT
0D3F C3800C    0630         JP   START

0D42 F1        0635 ERR     POP  AF
0D43 DF6B      0640         SCAL ERRM
0D45 C3800C    0645         JP   START

               0655 ; Konstanten
0D48 0020      0660 SPACE   EQU  20H
0D48 0063      0665 INLIN   EQU  63H ; Nas - sys LINE INPUT
0D48 006B      0670 ERRM    EQU  6BH
0D48 002D      0675 MAXSTR  EQU  45
0D48 0F00      0680 WRAM    EQU  0F00H ; Workspace des ZEAP- ASM
0D48 0002      0685 END     DEFS 2 ; Variablen
0D4A 0002      0690 STRT    DEFS 2 ;    "
```

```
0D4C 0002      0695 ARG    DEFS 2 ;   "
0D4E 0002      0700 STR1   DEFS 2 ;   "

ZEAP Z80 Assembler - Symbol Table

0D4CH 0695 ARG          0C88H 0080 CHANGE
0D48H 0685 END          0D33H 0580 END1
0D30H 0570 ENDCHN       0CC2H 0260 ENDTST
0D42H 0635 ERR          006BH 0670 ERRM
0CC7H 0275 FIND         0CDFH 0335 FIND1
0CA8H 0205 GTPOIT       0063H 0665 INLIN
0CEAH 0365 INSERT       0C91H 0110 LIN2
0C9AH 0150 LIN3         0D2CH 0555 LINEND
002DH 0675 MAXSTR       0D22H 0535 NONCOP
0020H 0660 SPACE        0C80H 0055 START
0D4EH 0700 STR1         0D4AH 0690 STRT
0F00H 0680 WRAM



ZEAP Z80 Assembler - Source Listing

              0010 ; FUNKTIONSTASTE/REPEAT
              0020 ; ====================
              0030 ;
              0040 ; Version :  5.9.1982
              0050 ; Gustav Delius
              0060 ;
              0070 ;
              0080 ; *** NAS-SYS KONSTANTEN ***
              0090 ;
19E7 0030     0100 ROUT   EQU  30H
19E7 0038     0110 RDEL   EQU  38H
19E7 005B     0120 MRET   EQU  5BH
19E7 005E     0130 FFLP   EQU  5EH
19E7 006A     0140 CRLF   EQU  6AH
19E7 007B     0150 BLINK  EQU  7BH
19E7 0072     0160 NIM    EQU  72H
19E7 0061     0170 KBD    EQU  61H
19E7 0076     0180 UIN    EQU  76H
19E7 0C7B     0190 $UIN   EQU  0C7BH
19E7 0C01     0200 KMAP   EQU  0C01H
19E7 0000     0210 KBDP   EQU  0
              0220 ;
              0230 ;
              0240 ; *** VARIABLEN ***
              0250 ;
              0260 ; 1 Byte Flag
19E7 0D38     0270 FFLAG  EQU  0D38H
              0280 ; 2 Byte Zeiger
19E7 0D39     0290 FUNP   EQU  0D39H
              0300 ; Anfangsadresse des Speichers
              0310 ; fuer den Funktionsstring
19E7 0D3B     0320 FSTR   EQU  0D3BH
              0330 ; Wiederholungsgeschwindigkeit
              0340 ; fuer Repeat
19E7 0010     0350 RRATE  EQU  10H
              0360 ; ASCII-Code der Funktionstaste
19E7 00B1     0370 FKEY   EQU  0B1H
              0380 ; ASCII-Code der Repeattaste
19E7 0017     0390 RKEY   EQU  17H
              0400 ; Startadresse des Programms
0CC0          0410 START  ORG  0CC0H
              0420 ;
              0430 ;
              0440 ; *** INITIALISIERUNG ***
              0450 ;
0CC0 21CD0C   0460 INIT   LD   HL,ITAB
0CC3 DF72     0470        SCAL NIM
0CC5 21CF0C   0480        LD   HL,INPUT
0CC8 227B0C   0490        LD   ($UIN),HL
0CCB DF5B     0500        SCAL MRET
              0510 ;
              0520 ;
              0530 ; *** NEUE INPUTTABELLE ***
              0540 ;
0CCD 76       0550 ITAB   DEFB UIN
0CCE 00       0560        DEFB 0
              0570 ;
              0580 ;
              0590 ; *** NEUE EINGABEROUTINE ***
              0600 ;
              0610 ; Wenn FFLAG=1 lese naechstes
              0620 ; Zeichen der FUNKTION.
0CCF 3A380D   0630 INPUT  LD   A,(FFLAG)
0CD2 B7       0640        OR   A
0CD3 2810     0650        JR   Z,REPEAT
              0660 ; Lese naechstes Zeichen
              0670 ; des Strings.
0CD5 2A390D   0680        LD   HL,(FUNP)
0CD8 7E       0690        LD   A,(HL)
              0700 ; Erhoehe Zeiger auf
              0710 ; neues Zeichen.
0CD9 23       0720        INC  HL
0CDA 22390D   0730        LD   (FUNP),HL
              0740 ; Return mit Zeichen
0CDD B7       0750        OR   A
0CDE 37       0760        SCF
0CDF C0       0770        RET  NZ
              0780 ; String ganz eingelesen:
              0790 ; Loesche FFLAG
0CE0 AF       0800        XOR  A
0CE1 32380D   0810        LD   (FFLAG),A
0CE4 C9       0820        RET
              0830 ;
              0840 ; SOFTWARE-REPEAT
              0850 ; beim druecken von LF
              0860 ;
              0870 ; LF gedrueckt?
0CE5 3E02     0880 REPEAT LD   A,2
0CE7 DF5E     0890        SCAL FFLP
0CE9 DB00     0900        IN   A,(KBDP)
0CEB 21010C   0910        LD   HL,KMAP
```

```
0CEE CB7?       0920        BIT  6,A
0CF0 2010       0930        JR   NZ,REND
                0940 ; Wenn LF gedrueckt:
0CF2 3E10       0950        LD   A,RRATE
0CF4 47         0960        LD   B,A
                0970 ; Warteschleife
0CF5 3EFF       0980 RDELAY LD   A,0FFH
0CF7 FF         0990        RST  RDEL
0CF8 10FB       1000        DJNZ RDELAY
                1010 ; Loesche KMAP, damit
                1020 ; altes Zeichen erneut
                1030 ; eingelesen werden kann
0CFA 0608       1040        LD   B,8
0CFC 77         1050 RLOOP  LD   (HL),A
0CFD 23         1060        INC  HL
0CFE 10FC       1070        DJNZ RLOOP
0D00 3680       1080        LD   (HL),80H
                1090 ; Lese Zeichen ein
0D02 DF61       1100 REND   SCAL KBD
0D04 F5         1110        PUSH AF
                1120 ; Funktionstaste?
0D05 FEB1       1130        CP   FKEY
0D07 2809       1140        JR   Z,FUN
                1150 ; Ignoriere RKEY
0D09 FE17       1160        CP   RKEY
0D0B 2003       1170        JR   NZ,RRET
0D0D F1         1180        POP  AF
0D0E AF         1190        XOR  A
0D0F C9         1200        RET
0D10 F1         1210 RRET   POP  AF
0D11 C9         1220        RET
                1230 ;
                1240 ; FUNKTIONSTASTE
                1250 ;
                1260 ; Setze FFLAG
0D12 3E01       1270 FUN    LD   A,1
0D14 32380D     1280        LD   (FFLAG),A
                1290 ; Anfang des Strings
0D17 213B0D     1300        LD   HL,FSTR
0D1A 22390D     1310        LD   (FUNP),HL
0D1D F1         1320        POP  AF
0D1E 18AF       1330        JR   INPUT
                1340 ;
                1350 ;
                1360 ; FUNKTIONSTASTENBELEGUNG
                1370 ;
                1380 ; Lese Begrenzungszeichen
0D20 DF7B       1390 FUNK   SCAL BLINK
0D22 F7         1400        RST  ROUT
0D23 47         1410        LD   B,A
                1420 ; Lade Anfang des Speichers
                1430 ; fuer den String
0D24 213B0D     1440        LD   HL,FSTR
0D27 E5         1450 FKL    PUSH HL
                1460 ; Lese Zeichen ein und
                1470 ; speichere es ab
0D28 DF7B       1480        SCAL BLINK
```

```
0D2A F7         1490        RST  ROUT
0D2B E1         1500        POP  HL
0D2C 77         1510        LD   (HL),A
0D2D 23         1520        INC  HL
                1530 ; Wiederhole bis zum
                1540 ; Begrenzungszeichen
0D2E B8         1550        CP   B
0D2F 20F6       1560        JR   NZ,FKL
0D31 2B         1570        DEC  HL
                1580 ; Beende String mit 0
0D32 3600       1590        LD   (HL),0
0D34 DF6A       1600        SCAL CRLF
0D36 DF5B       1610        SCAL MRET
```

ZEAP Z80 Assembler - Symbol Table

```
0C7BH 0190 $UIN         007BH 0150 BLINK
006AH 0140 CRLF         0D38H 0270 FFLAG
005EH 0130 FFLP         00B1H 0370 FKEY
0D27H 1450 FKL          0D3BH 0320 FSTR
0D12H 1270 FUN          0D20H 1390 FUNK
0D39H 0290 FUNP         0CC0H 0460 INIT
0CCFH 0630 INPUT        0CCDH 0550 ITAB
0061H 0170 KBD          0000H 0210 KBDP
0C01H 0200 KMAP         005BH 0120 MRET
0072H 0160 NIM          0038H 0110 RDEL
0CF5H 0980 RDELAY       0D02H 1100 REND
0CE5H 0880 REPEAT       0017H 0390 RKEY
0CFCH 1050 RLOOP        0030H 0100 ROUT
0010H 0350 RRATE        0D10H 1210 RRET
0CC0H 0410 START        0076H 0180 UIN
```

ZEAP Z80 Assembler - Source Listing

```
                0010 ;
                0020 ; ********************
                0030 ; * PROGRAM  H E L P *
                0040 ; *  FOR GENERATING  *
                0050 ; *    A CHARACTER   *
                0060 ; *     GENERATOR    *
                0070 ; ********************
                0080 ;
                0090 ;BY: GREGOR  BIRNFELD
                0100 ;    [redacted]
                0110 ;    [redacted]
                0120 ;    ---------------
                0130 ;    TEL.  [redacted]
                0140 ;
                0150 ;
                0160 ;COMPATIBLE TO NAS-SYS 1
                0170 ;
                0180 ;ALL INP.CHARACTERS EXEPT " " AND "." WILL
                0190 ;PRODUCE A POINT OF THE CHARACTER.
                0200 ;
```

```
                    0210 ;STORE THE FIELD BY "ESC" (SHIFT + ENTER)
                    0220 ;
8000                0230        ORG  £8000; CHANGE IF YOU WANT.
8000 6000           0240 DATAM  EQU  £6000;WHERE TO STORE DATA
                    0250 ;DATA ARE STORED AT DATAM TO DATAM+800h
                    0260 ;(CHANGE DATAM IF YOU WANT)
8000                0270        ENT
8000 0028           0280 PRS    EQU  28H
8000 0018           0290 SCAL   EQU  18H
8000 0030           0300 ROUT   EQU  30H
8000 000C           0310 CLS    EQU  0CH
8000 0C29           0320 CURSOR EQU  £0C29
8000 001B           0330 ESC    EQU  1BH
8000 007B           0340 BLINK  EQU  7BH
8000 0C80           0350 MLOC   EQU  £0C80
8000 0895           0360 SST    EQU  £0895;START OF FIELD ON SCREEN
                    0370 ;
                    0380 ;
8000 210060         0390 START  LD   HL,DATAM;FIRST MEMORY LOCATION
8003 22800C         0400        LD   (MLOC),HL
8006 0604           0410        LD   B,4
8008 C5             0420 ST1    PUSH BC
8009 0600           0430        LD   B,0
800B C5             0440 ST2    PUSH BC
800C D70B           0450        RCAL PRMASK
800E D758           0460        RCAL SETMSK
8010 D768           0470        RCAL MEMSK
8012 C1             0480        POP  BC
8013 10F6           0490        DJNZ ST2
8015 C1             0500        POP  BC
8016 10F0           0510        DJNZ ST1
                    0520 ;
8018 C7             0530        RST  0;END, RETURN TO NAS-SYS
                    0540 ;
                    0550 ; S U B R O U T I N E S :
                    0560 ;
                    0570 ;PRMASK SETS A MASK 8*10 FOR THE CHARACTER
8019 3E0C           0580 PRMASK LD   A,CLS
801B F7             0590        RST  ROUT; CLEAR SCREEN
801C EF             0600        RST  PRS
801D 2020456E       0610        DEFM /  Enter by ESC !/
     74657220
     62792045
     53432021
802D 20204C4F       0620        DEFM /  LOCATION: /
     43415449
     4F4E3A20
8039 00             0630        DEFB 00
803A E5             0640        PUSH HL
803B 2A800C         0650        LD   HL,(MLOC)
803E DF             0660        RST  SCAL
803F 66             0670        DEFB £66; ROUTINE TBCD3
8040 E1             0680        POP  HL
8041 EF             0690        RST  PRS
8042 0D0D00         0700        DEFB 0DH,0DH,0
                    0710 ;
8045 060A           0720        LD   B,0AH

8047 EF             0730 PR1    RST  PRS
8048 20202020       0740        DEFM /           /
     20202020
     202020
8053 2E202E20       0750        DEFM /. . . . . . . . /
     2E202E20
     2E202E20
     2E202E20
8063 0D             0760        DEFB 0DH
8064 00             0770        DEFB 00
8065 10E0           0780        DJNZ PR1
8067 C9             0790        RET
                    0800 ;
                    0810 ;
                    0820 ;IN SETMSK YOU SET THE POINTS
8068 E5             0830 SETMSK PUSH HL
8069 219508         0840        LD   HL,SST
806C 22290C         0850        LD   (CURSOR),HL
806F DF             0860 SE1    RST  SCAL
8070 7B             0870        DEFB BLINK
8071 FE1B           0880        CP   ESC
8073 2803           0890        JR   Z,SEND
8075 F7             0900        RST  ROUT
8076 18F7           0910        JR   SE1
                    0920 ;
8078 E1             0930 SEND   POP  HL
8079 C9             0940        RET
                    0950 ;
                    0960 ;MEMSK LOADS THE MASK INTO MEMORY
                    0970 ;  HERE: FROM THE RIGHT TO THE LEFT!
807A 2A800C         0980 MEMSK  LD   HL,(MLOC)
807D D9             0990        EXX
807E 219508         1000        LD   HL,SST
8081 110E00         1010        LD   DE,0EH; ADD 14 FOR LEFT/RIGHT INV
8084 19             1020        ADD  HL,DE; IF NORMAL, DO NOT ADD.
8085 0608           1030        LD   B,8;8 LINES
8087 C5             1040 ME3    PUSH BC
8088 E5             1050        PUSH HL
8089 0608           1060        LD   B,8;8 CHARS PER LINE
808B 1600           1070        LD   D,0
808D 7E             1080 ME2    LD   A,(HL)
808E CB22           1090        SLA  D
8090 FE2E           1100        CP   ". ;IS IT A POINT?
8092 2806           1110        JR   Z,MEMO
8094 FE20           1120        CP   " ;OR IS IT A BLANK?
                    1130 ;      (SAME AS A POINT)
8096 2802           1140        JR   Z,MEMO
                    1150 ;IF NOT, SET THAT BIT:
8098 CBC2           1160        SET  0,D
                    1170 ;
809A 2B             1180 MEMO   DEC  HL; FOR INVERTED; IF NORMAL
809B 2B             1190        DEC  HL;  DO INCREMENT!
809C 10EF           1200        DJNZ ME2;TIL END OF LINE
809E 7A             1210        LD   A,D
809F D9             1220        EXX  ;HL'=(MLOC)
80A0 77             1230        LD   (HL),A
80A1 23             1240        INC  HL
```

```
80A2 D9        1250        EXX
80A3 E1        1260        POP  HL;START OF OLD LINE
80A4 114000    1270        LD   DE,40H;1 LINE
80A7 19        1280        ADD  HL,DE
80A8 C1        1290        POP  BC
80A9 10DC      1300        DJNZ ME3;TIL 8 LINES OK
               1310
80AB 0602      1320        LD   B,2;ANOTHER 2 LINES
80AD C5        1330 ME5    PUSH BC
80AE E5        1340        PUSH HL
80AF 0608      1350        LD   B,8;8 CHARACTERS
80B1 1600      1360        LD   D,0
80B3 7E        1370 ME6    LD   A,(HL)
80B4 CB22      1380        SLA  D
80B6 FE2E      1390        CP   ".;IS IT A POINT?
80B8 2804      1400        JR   Z,MEM1
80BA FE20      1410        CP   " ;OR A BLANK?
               1420 ;IF NOT, SET THAT BIT;
80BC CBC2      1430        SET  0,D
               1440 ;
80BE 2B        1450 MEM1   DEC  HL; IF NOT INVERTED, DO
80BF 2B        1460        DEC  HL;  INCREMENT!
80C0 10F1      1470        DJNZ ME6
80C2 7A        1480        LD   A,D
80C3 D9        1490        EXX
80C4 77        1500        LD   (HL),A
80C5 23        1510        INC  HL
80C6 D9        1520        EXX
80C7 E1        1530        POP  HL;START OF OLD LINE
80C8 114000    1540        LD   DE,40H;1 LINE
80CB 19        1550        ADD  HL,DE
80CC C1        1560        POP  BC
80CD 10DE      1570        DJNZ ME5; TIL LAST LINE
               1580 ;
               1590 ;
               1600 ;NOW FILL 6x00h INTO MEMORY
80CF D9        1610        EXX
80D0 0606      1620        LD   B,6
80D2 AF        1630        XOR  A
80D3 77        1640 ME8    LD   (HL),A
80D4 23        1650        INC  HL
80D5 10FC      1660        DJNZ ME8
               1670 ;
               1680 ;NOW GIVE NEW MEMORY LOCATION
               1690 ;TO MLOC
80D7 22800C    1700        LD   (MLOC),HL
80DA D9        1710        EXX
               1720 ;
80DB C9        1730        RET
               1740 ;
               1750 ;
```


ZEAP Z80 Assembler - Symbol Table

```
007BH 0340 BLINK        000CH 0310 CLS
0C29H 0320 CURSOR       6000H 0240 DATAM
001BH 0330 ESC          808DH 1080 ME2
8087H 1040 ME3          80ADH 1330 ME5
80B3H 1370 ME6          80D3H 1640 ME8
809AH 1180 MEM0         80BEH 1450 MEM1
807AH 0980 MEMSK        0C80H 0350 MLOC
8047H 0730 PR1          8019H 0580 PRMASK
0028H 0280 PRS          0030H 0300 ROUT
0018H 0290 SCAL         806FH 0860 SE1
8078H 0930 SEND         8068H 0830 SETMSK
0895H 0360 SST          8008H 0420 ST1
800BH 0440 ST2          8000H 0390 START
```

nascom

Die Alternative!

Kein »langweiliger Computer«
NASCOM 1 und NASCOM 2 sind Computer für Selbermacher, Tüftler, erfolgreiche Do-it-yourself-Freunde. NASCOM-Computer werden niemals langweilig! Die Systeme 1 und 2 sind keine fertigen »Kästen« ohne Erweiterungsmöglichkeit. Der hochwertige Platinensatz Computer und Keyboard kann so aufgebaut, erweitert und »verpackt« werden, wie **Sie** es wünschen.
Für Vollpreis gibt die NASCOMs auch als Bausatz. Aber aufgepaßt: Das ist eine Sache nur für wirkliche Könner! Und damit es auch nach dem Aufbauen nicht langweilig wird, gibt es das monatlich erscheinende NASCOM-JOURNAL. Eine Zeitschrift speziell für NASCOM-Freaks vollgestopft mit Hardware- und Software-Ideen, Kleinanzeigen, den neuesten Infos, und, und, und. . .
Die NASCOMs sind keine »Spielcomputer«. Mehr als 60% aller NASCOM-Systeme werden als sogenannte »OEM-Baugruppen« von professionellen Anwendern in eigene Systeme eingebaut. Ingenieurbüros verwenden den NASCOM als Entwicklungssystem. Die Anwendungsmöglichkeiten sind mehr durch Ihre Phantasie begrenzt. Ein NASCOM-System kann fast alle gängigen Probleme lösen.

Mit NASCOM wachsen!

NASCOM-Systeme sind aufwärtskompatibel. Das kleinste, preisgünstigste NASCOM 1-System kann bis auf NASCOM 3-Level mit Floppy-Laufwerken und CP/M* ausgebaut werden. Bildschirm-Ansteuerung, Tastatur Betriebsystem und Systemsoftware sind durchweg kompatibel. Ohne faule Kompromisse!
Mit NASCOM-Systemen gehen Sie kein Risiko ein. Ihr NASCOM wächst mit!

NASCOM 1
Spezifikationen:
- QWERTY-Tastatur, aufgebaut mit hochwertigen Magnettasten
- NAS-SYS Betriebsystem (2k Byte)
- 16 I/O-Leitungen
- Video (BAS) und TV-Ausgang
- 1k RAM, ausbaubar auf 192k RAM
- Display 48 Zeichen in 16 Zeilen

ab DM 935,—

NASCOM 2
Spezifikationen:
Wie NASCOM 1, jedoch zusätzl.:
- 8k Mikrosoft-BASIC u. 8k Stat. RAM
- Z80A-Mikroprozessor, 4 MHz
- Erweiterte Tastatur 57 Tasten
- Integrierte Bus-Pufferung
- Bis 192k Byte RAM
- Grafik-Möglichkeiten: 48 x 96 Punkte
- Serielle Schnittstelle; Baudrate wählbar, RS232C/20mA
- 16 parallele Ein/Ausgabeleitungen (Z80APIO)

ab DM 1950,—

nascom 3 — der Profi

Spezifikationen: Wie NASCOM 2, jedoch zusätzl.: **ab DM 2735,—**
- 0.35 Megabyte pro 5,25-Zoll Laufwerk
- Betriebsystem CP/M*2.2 oder NAS-DOS
- Bildschirmausgabe erweiterbar auf 80 x 25 Zeichen

Die dritte NASCOM-Generation

NASCOM 1 und 2 haben OEM-Board, Schulungscomputer, Kompaktrechner etc. ca. 20 000 mal ihren Partner gefunden. Der NASCOM 3 möchte Ihr persönlicher Computer werden! Er möchte Ihnen helfen, sich selbst fortzubilden, im Beruf weiter zu kommen, auch mal in die Computertechnik »rein zu riechen«. Ingenieurbüros und Softwareingenieuren dient der NASCOM 3 als preisgünstiges Entwicklungssystem.

Universelle Betriebssoftware

Der NASCOM 3 kennt zwei Betriebssysteme: Das CP/M* (Version 2.2) — inzwischen Standard — und sein eigenes NAS-DOS. Die 5-Zoll Floppys bieten eine Speicherkapazität von 0.35 Megabyte pro Laufwerk (single sided, double density, double tracked). Damit wird das Spektrum universeller CP/M*-Software verfügbar!

Wir informieren Sie unverbindlich:
Fordern Sie Ihr NASCOM-INFO-PAKET an!**

Unsere Händler:

Heinz Vogel Verlag GmbH & Co.
Lehrmittelzentrum, Herr Seloff
Innsbrucker Straße 96
2800 Bremen-Findorff
☎ (0421) 35 10 69

MK-SYSTEMTECHNIK
Michael von Keitz
Pfaffenberg 4
5650 Solingen
☎ (02122) 47267

Radio Zinburg
Herr Zinburg, Jr.
Röhrstraße 10
5760 Arnsberg
☎ (02932) 3 15 10

Christian Lampson
W.-Leuschner-Straße 4
6085 Nauheim
☎ (06152) 56730

MK-SYSTEMTECHNIK
Kriegsstraße 164
7500 Karlsruhe
☎ (0721) 29243

Graf Elektronik Systeme GmbH
Postfach 1610
8791 Kempten
☎ (0831) 61930

Autorisierter Distributor:

MK
MK-SYSTEMTECHNIK
Pater-Mayer-Straße 6
6728 Germersheim
☎ (07274) 2093
Telex 453500 mks d

CP/M* ist ein eingetragenes Warenzeichen der DIGITAL RESEARCH
** NASCOM-INFO-PAKET gegen DM 2,— in Briefmarken (wird bei Kauf angerechnet)