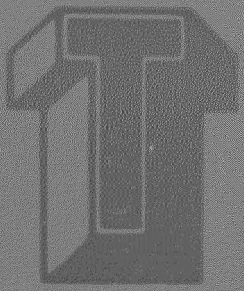
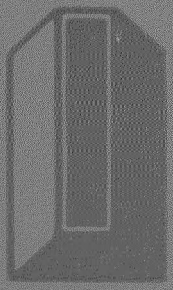
Kleinstrechner



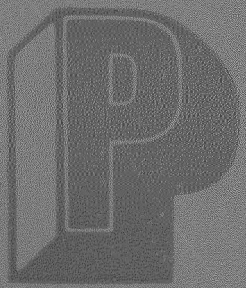
Tendenzen
und Theorien

Pseudographik



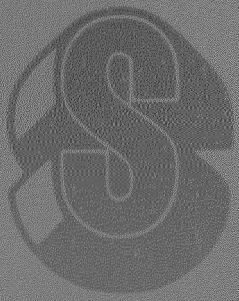
Informationen
und Ideen

Benchmarktests



Programme
und Projekte

Vollgraphik-
Bildschirm-
ansteuerung



Spaß
und Spiel

12

Kleinstrechner-TIPS

Heft 12

Mit 25 Bildern

Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Prof. Dr. sc. techn. Thomas Horn
Doz. Dr.-Ing. Wilhelm Leupold

VEB Fachbuchverlag Leipzig

Inhalt

ISBN 3-343-00581-9

Zum Inhalt

Beim Einsatz von Computern hat die Verwendung von Graphik sehr stark an Bedeutung zugenommen. Diese Tendenz beachtend, werden in diesem Heft mehrere Beiträge zur Computergraphik veröffentlicht. Des weiteren werden – wie in den bisherigen Heften – Anwendungen und Programme vorgestellt, die für Arbeitsgemeinschaften und Computerclubs, aber auch für den Einzelnutzer und Bastler, von Interesse sein dürften.

SCHÖNFELDER vergleicht in seinem Beitrag Pseudographik mit vollgraphischen Systemen und erläutert den Aufbau und die Bewegung von pseudographischen Bildern sowohl mittels statischer als auch dynamischer Pseudographik.

Ein von KÜHNEL entwickeltes Programm, das auf dem »Sieb des Eratosthenes« beruht, ermöglicht es dem Besitzer bzw. Nutzer eines Computers, dessen Leistungsfähigkeit bei Verwendung von BASIC, Turbo-PASCAL oder FORTH durch Benchmarktests zu ermitteln.

BOGATZ stellt eine Bildschirm-Ansteuerbaugruppe vor, die sowohl in industriellen Mikrorechnersystemen als auch in Eigenbaucomputern eingesetzt werden kann. Für den Elektronikbastler ist der Schaltplan als eingeheftetes Faltblatt beigefügt.

Im Beitrag von SCHÖNFELDER zum Einsatz graphischer Bildschirmsysteme werden die Möglichkeiten der Gestaltung von Computergraphik und Bildschirmsteuerung, der Textausgabe sowie ein BASIC-Interpreter zur Steuerung eines Graphikprozessors gezeigt. Die Befehle des Graphik-BASIC sind ebenfalls auf dem eingehefteten Faltblatt abgedruckt.

Wir möchten unseren Lesern für ihre Zuschriften danken. Wie aus diesen zu erkennen ist, hat sich ein ständiger Interessentenkreis herausgebildet. Beim Einsenden weiterer Manuskripte bitten wir, die auf der 3. Umschlagseite gegebenen Hinweise zu beachten.

Wilhelm Leupold

Möglichkeiten der Programmierung statischer und dynamischer Pseudographik

Bei der Nutzung von Rechnersystemen gewinnt die graphische und pseudographische Darstellung auf dem Bildschirm zunehmend an Bedeutung.
Die Pseudographik stellt in diesem Problemkreis eine Zwischenstufe dar, welche aus einem Kompromiß zwischen technischem Aufwand, Arbeitsgeschwindigkeit und graphischen Möglichkeiten resultiert. Es lassen sich aber bereits damit beeindruckende Ergebnisse erreichen, was sich beispielsweise mit dem Heimcomputer KC 87 demonstrieren läßt.

1. Anwendung von Pseudographik

Pseudographik besitzt gegenüber vollgraphischen Systemen einige Vorteile, die die Arbeit mit dieser für viele Anwendungsfälle interessant macht. Das sind unter anderem:

a) Pseudographische Bildschirme arbeiten nach den gleichen Prinzipien wie alphanumerische. Damit erreichen sie die gleichen zeitlichen Parameter beim Bildaufbau wie diese.
b) Die Be- und Verarbeitung von pseudographischen Bildern ist einfacher und schneller, da die rückgelesenen Symbole (genauer deren Codierung) für den Rechner eine eindeutige Verhaltensvorschrift enthalten können. So ist z. B. die Rückerkennung einer Linienkreuzung als pseudographisches Symbol äußerst einfach. In einem vollgraphischen System wäre dieser Sachverhalt aus dem Suchen, Einlesen und logischen Verarbeiten mehrerer Punkte zu rekonstruieren oder müßte durch Rechnung vorausbestimmt werden.
c) Der Speicherplatzbedarf für ein Bild ist wesentlich geringer.
d) Pseudographik kann in Form von Zeichenketten erzeugt und bearbeitet werden.

Dem gegenüber stehen bei der Vollgraphik natürlich:
a) eine feinere und flexiblere Linienführung und
b) die Realisierung von stark unsymmetrischen Figuren sowie Kurven aller Art.

2. Technische Voraussetzungen

Die Arbeit mit Pseudographik setzt bei den genutzten peripheren Baugruppen des Rechners einige spezielle Eigenschaften voraus. Diese Forderungen erstrecken sich vor allem auf Bildschirm, Tastatur und Drucker. Dabei kann bei der Tastatur notfalls auf eine Belegung mit den pseudographischen Symbolen noch verzichtet werden, da man diese zumeist aus dem Programm heraus nutzt. Im einfachsten Fall wird eine Steuerfunktion zur Umschaltung der gesamten Tastenbelegung ins System eingebracht.

2.1. Bildschirm

Das Bildschirmsystem hat die Hauptleistung bei der pseudographischen Arbeit zu stellen.

Zeichensatz des Bildschirms

Der normale Zeichensatz eines Bildschirms umfaßt den ASCII-Satz, womit 96 Zeichen belegt sind (einschließlich der Codierungen 00H bis 1FH). Es ergeben sich damit bei einem 1 KByte Zeichengenerator und Zeichen zu 8 × 8 Punkten noch 32 freie Zeichen bzw. bei 2 KByte noch 160 unbelegte Plätze. Bei Heimcomputern werden dann zumeist nur die 128 Zeichen ab der Codierung 80H genutzt, da die Nutzung der restlichen 32 Zeichen bei 00H ··· 1FH einen zusätzlichen Behandlungsaufwand erfordert.

Diese Zeichensätze bestehen aus flexibel nutzbaren Graphikelementen, wie Linien, Bogen, verschieden gefüllten Zeichen und Spielsymbolen. Die dabei verwendeten Symbole sind für die meisten Problemstellungen ausreichend, solange keine aufgabenspezifischen Zeichen benötigt werden.

Die Alternative dazu stellt der vollständig oder teilweise ladbare Zeichengenerator dar. Er arbeitet nach dem gleichen Wirkprinzip wie ein fester Zeichengenerator, ist jedoch vom Rechner aus nachladbar. Der Vorteil besteht im Einsatz von problemspezifischen Zeichen, wodurch auch eine Vielzahl technischer Aufgabenstellungen bearbeitet wird.

Zeichenformat

Heimcomputer sind allgemein so ausgelegt, daß die Bildpunkte in ihren Abmessungen gleichseitig sind, d. h., ein Quadrat wird auch optisch als solches dargestellt und nicht zum Rechteck verzerrt. Daraus läßt sich ableiten, daß die einzelnen Zeichen ebenfalls gleichseitig sein sollten, um die geometrischen Figuren nicht verzerrt wiederzugeben. Die meisten Rechner besitzen einen Zeichengenerator mit dem Format 8 × 8 Punkte. Dies ist auch dadurch begründet, daß man zur Darstellung eines ASCII-Zeichens 5 × 7 Punkte benötigt und sich somit Alphazeichen und Pseudographik in einem Format darstellen lassen. Die bei Schrift freien Punkte dienen zumeist gleich als Zwischenraum zwischen den einzelnen Zeilen und Spalten. Das Format 8 × 8 hat allerdings den Nachteil, daß man keine spiegelsymmetrischen Konturen mit einem Punkt Linienstärke darstellen kann, was jedoch nur in wenigen Fällen stört.

Eine andere Zeichengestaltung geht von 10 × 12 Punkten aus, wodurch verbesserte Möglichkeiten für die Darstellung von Schrägen unterschiedlicher Neigung gegeben sind. Eingesetzt wird dieses Format beispielsweise in Bildschirmtext-Systemen.

Bildschirmformat

Wenn man bei der Zeichenausgabe ein optisch unverzerrtes Bild erhalten möchte, ist eine Abstimmung zwischen Zeilenzahl und Punkttakt erforderlich. Im Regelfall werden identische Halbbilder mit je 256 sichtbaren Fernsehzeilen gezeichnet. Werden horizontal 256 Punkte angesteuert, ergibt sich eine Punktfrequenz von 7 MHz für ein quadratisches Bildfenster. Will man das Fernsehformat mit dem Seitenverhältnis 3:4 ausfüllen, so kann man horizontal auf 340 Punkte erhöhen. Dabei ist jedoch die Punktfrequenz beizubehalten und nur die dargestellte Bildbreite zu ändern.

Die Arbeit mit 512 Punkten in einer Zeile ist technisch zwar möglich, ergibt aber grobe optische Fehler. Beim Zeichnen eines Kreises wäre diese Verzerrung in der Programmierung zu berücksichtigen. Die Alternative liegt in

der Erhöhung der Zeilenzahl, was aber die Ausgabe von Vollbildern erfordert.

Lesbarkeit des Zeichengenerators

Für die Nutzung von Pseudographik, vor allem im Zusammenhang mit deren Druck, ist es erforderlich, den aktuellen Inhalt des Zeichengenerators vom System aus lesen zu können. Diese Rücklesbarkeit kann entweder hardwaretechnisch realisiert werden, oder der Zeichengenerator wird als Datenfeld im Systemspeicher zur Verfügung gestellt.

2.2. Drucker

Auch für die Ausgabe von pseudographischen Bildern wird ein vollgraphischer Drucker benötigt. Es gibt zwar international Drucker für Heimcomputer, welche einen pseudographischen Zeichensatz besitzen bzw. diesen laden können, jedoch sind diese meist an besondere Rechnertypen gebunden.
In der DDR stehen als graphische Drucker die Typen SD1157G, SD1154G und die Reihe K631x zur Verfügung. Für den Amateur kommt sicher nur die Reihe K631x in Frage. Bedingung an einen Drucker ist für diesen Zweck auf jeden Fall die Möglichkeit der Einzelnadelansteuerung.

3. Anwendung statischer Pseudographik

Unter statischer Pseudographik sollen alle Programme verstanden werden, in denen ein Zeichen (oder mehrere Zeichen) in einem ruhenden Bild bewegt werden. Dabei hängt die Zeichenbewegung nur vom Bediener ab. Die Bilder enthalten keine bedienerunabhängigen Bewegungsabläufe.
In diese Kategorie fallen Spiele, denen eine Wegoptimierung zugrunde liegt, wie z. B. Labyrinthe. Auch lassen sich Aufgabenstellungen hierzu zählen, wo die Pseudographik nur zur optischen Unterstützung von Spielen (z. B. »Turm von Hanoi«) bzw. zur Ausgestaltung von Texten Verwendung findet. Es sind auch anspruchsvolle Probleme wie die Digitalisierung von Leiterplatten mit statischer Pseudographik lösbar.

Aufbau von pseudographischen Bildern

Pseudographische Bilder können am einfachsten als hexadezimale Datenfolge definiert werden und durch einen Aufruf als »Text« auf den Bildschirm gebracht werden.
Arbeitet man in BASIC, so ist es meist möglich, die Zeichen von der Tastatur aus sofort in einer Zeichenkette einzutragen. Bei der Programmierung im Assemblerniveau wäre eine Definition der Zeichen als DB-Anweisung notwendig, was bei 1024 oder mehr Zeichen je Bild sehr mühsam ist. Man kann sich mit einem Unterprogramm zur Ausgabe von »Graphik-Texten« helfen, welches die Codierung der Buchstaben durch Addition einer Verschiebung in den Graphikbereich transformiert. Damit können die gewünschten Bilder als Textketten wesentlich bequemer erstellt werden. Dieser Weg schließt allerdings die gemischte Ausgabe von Bildern und Text durch einen Aufruf aus.
Ein wichtiger Punkt beim Aufbau von pseudographischen Bildern ist die Funktion der Zeichen für den Rechner. So ist bei Bewegungen im Bild für einen Rand zu sorgen, da das Bild während der Nutzung nicht verlassen werden darf. Dies ist mit Funktionszeichen meist einfacher zu realisieren als im Programm eine Auswertung der Bildschirmadressen vorzunehmen. Von diesem Problem ausgehend, sollen bei der Wahl der Zeichencodierung und Anordnung die folgenden Gesichtspunkte berücksichtigt werden.
a) Anordnung in aufsteigender Codierung nach den Funktionen

– überdeckbares Zeichen
– überdeckendes (bewegtes) Zeichen
– nicht überdeckbare Zeichen
– Bildrand bzw. Begrenzungszeichen.

Es ist damit möglich, durch einfache Vergleiche die Zulässigkeit einer Aktivität zu bestimmen. So gibt die Prüfung der Codierung z. B. Auskunft, ob eine Bewegung ausführbar ist oder ob damit ein Randsymbol »betreten« wird.
Es ist innerhalb der Gruppen, vor allem der überdeckenden Zeichen, auch die Einführung einer Priorität und damit optischer Ebenen möglich.

b) Wenn es notwendig ist, sollte die Zeichenanordnung arithmetischen und logischen Operationen entgegenkommen. Dies soll am Beispiel eines Spieles mit Autos verdeutlicht werden (Bilder 1 und 2). Es gibt ein Fahrzeuggrundsymbol und dieses mit Blinker rechts und links. Diese drei Zeichen werden in allen vier möglichen Bewegungsrichtungen benötigt, was 12 Symbole ergibt. Durch die Verteilung auf 16 Zeichen, also 4 Vierergruppen, läßt sich die Steuerung der Fahrzeuge arithmetisch ausführen. Das Setzen des Blinkers erfolgt unabhängig von der Bewegungsrichtung durch Setzen von Bit 0 oder 1, das Löschen durch ein »AND 0FC0H«. Der Wechsel der Bewegungsrichtung entspricht einer Addition oder Subtraktion von 4. Diese Operationen wären wesentlich aufwendiger, wenn das vierte, ungenutzte Zwischenzeichen entfallen würde.

Bewegung im pseudographischen Bild

Interessant wird erst die Bewegung von Objekten bzw. Zeichen im Bild. Dabei hat das jeweilige Programm eine Reihe von Aufgaben abzuarbeiten, die allen derartigen Aufgabenstellungen gemeinsam ist.

1. Es ist eine Bewegungsrichtung durch Eingabe oder mathematische Verfahren (z. B. Zufallszahlen) zu bestimmen.
2. Es erfolgt die Löschung des Zeichens am jetzigen Ort und die Herstellung des ursprünglichen Zustands (verdecktes Zeichen) bzw. das Eintragen eines der Aufgabe entsprechenden Symbols (beim Zeichnen einer Linie). Bei letzterem ist meist eine Berücksichtigung der Ankunfts- und Abgangsrichtung bei der Zeichenwahl nötig.
3. Die angegebene Bewegung wird rechnerisch ausgeführt.

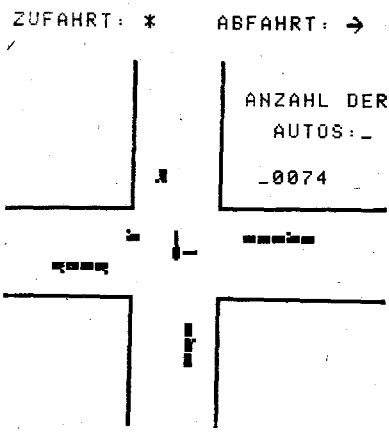


Bild 1. Spielbild zum Programm KREUZUNG

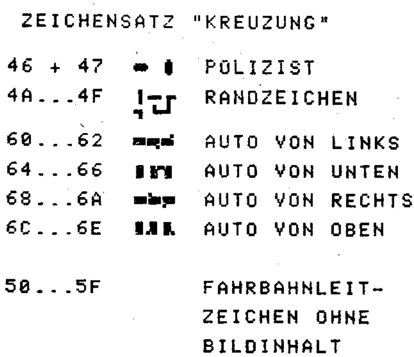


Bild 2. Zeichengenerator zum Programm KREUZUNG

4. Das Zeichen am neuen Ort wird geprüft auf:
- Betretbarkeit (Hindernisse, Rand, ...)
- notwendige Überdeckungen
- Überlagerungen von Zeichen zu neuen Zeichen (Codewechsel).

Im Fall der Nichtbetretbarkeit wird die Bewegung abgewiesen und der Zustand vor der Eingabe hergestellt.

5. Ausführung der Bewegung, d. h. Eintragen des Zeichens am neuen Ort und Retten des verdeckten Zeichens.

6. Auswertung der Bewegung bzw. Bewertung bei Spielen.

Bei der Programmierung von Spielen ist die Zeichen- und Positionsbehandlung relativ unkompliziert. Sie hängt meist nur von den Gegebenheiten der neu zu belegenden Position ab. Bei größeren Systemen kann sich bereits eine umfangreiche Auswertung erforderlich machen, da die aus der Bewegungsrichtung resultierenden Symbole außer vom Ort auch noch von den vorher gesetzten Zeichen abhängen.
Als Beispiele für diesen Abschnitt sollen zwei Spielprogramme vorgestellt werden, die zur Gruppe der statischen Pseudographik gehören. Es handelt sich dabei um ein Labyrinth (Bild 3) und ein Geländespiel (Bild 5).
Beim Labyrinth ist der kürzestmögliche Weg von links nach rechts zu finden (Bild 3). Dabei stellen die Wände gleichzeitig Begrenzungszeichen dar, welche nicht betretbar sind. Alle anderen Symbole sind betretbare Hindernisse, die mit Strafpunkten in Form erhöhter Schrittzahl belegt werden. Die Fußspuren markieren den gegangenen Weg, sind betretbar, aber nicht löschbar, und ergeben keine Strafpunkte. Diese Zuordnung hat zur Folge, daß ein falscher Weg sichtbar bleibt, auch wenn man ihn korrigiert hat. Wie in Bild 4 ersichtlich, trägt die Gestaltung des Zeichengenerators den oben aufgeführten Regeln Rechnung.
Das Geländespiel ist vom Spielgedanken (und vom Programm) her dem Labyrinth nahezu gleich – es stellt eine Wegoptimierung dar. Der Unterschied besteht im Fehlen der typischen Trennwände. Die Aufgabe ist, ein Rohr von links unten nach rechts oben zu verlegen. Dabei sind die Kosten zu minimieren, was nicht unbedingt der kürze-

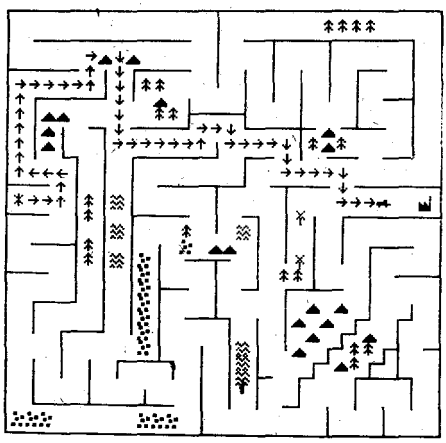


Bild 3. Spielbild zum Programm LABY

ZEICHENSATZ

46		LEERZEICHEN
47	⁘	SCHLECHTER WEG
48	⁑	BAHNUEBERGANG
49	↟	WALD
4A	≋	WASSERSTELLEN
4B		
4C		
4D		
4E		
4F	▲	BERGE
50 BIS 5A		TRENNWAENDE
5B		
5C	▬	FAHRZEUG
5D BIS 60		TRENNWAENDE

Bild 4. Zeichengenerator zum Programm LABY

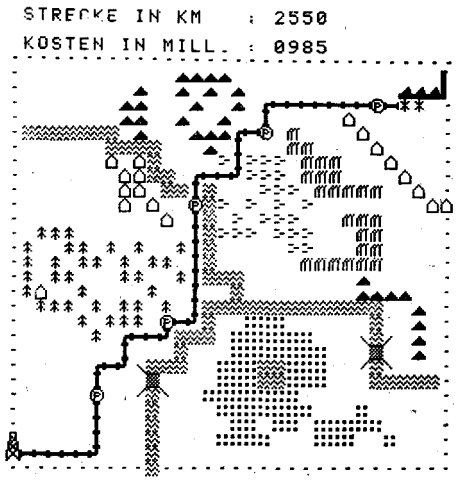


Bild 5. Spielbild zum Programm TRASSE

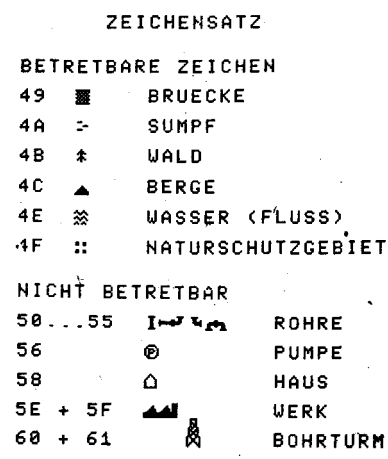


Bild 6. Zeichengenerator zum Programm TRASSE

sten Strecke entspricht. Die einzelnen Flächen werden wiederum mit einem Strafzuschlag (Kostenfaktor) belegt. Der Rechner fügt außerdem anstelle jedes zehnten Rohres eine wesentlich teurere Pumpstation ein, die man dann möglichst nicht in den Fluß setzen sollte. Man hat mit diesem Spiel bereits eine zweidimensionale Optimierung mit einigen Störgrößen durchzuführen.
Auch dieses Programm, dessen Zeichengenerator in Bild 6 zu sehen ist, hält sich an die Aufbauregeln.

4. Anwendung dynamischer Pseudographik

Unter dynamische Pseudographik sollen alle Anwendungen zählen, bei denen ein oder mehrere Zeichen bzw. Zeichengruppen auf stehendem Bild zeitgesteuert bewegt werden. Die Bewegung ergibt sich dabei aus einer zeitabhängigen Funktion und kann mittelbar durch den Bediener beeinflußt werden. Ein sehr typisches Beispiel für diese Gruppe wäre »TENNIS«, wo der Spielball sich zeitgesteuert bewegt und nur über den Schläger als zweites Objekt indirekt beeinflußt wird.

Einige der bei der Programmierung von dynamischer Pseudographik auftretenden Probleme sollen im folgenden angerissen werden.

4.1. Geschwindigkeit der Objekte

Die Bewegungsgeschwindigkeit der Objekte wird durch zwei Erscheinungen wesentlich beeinflußt. Zum einen nimmt das menschliche Auge Bildfolgen unter 16 Hz als getrennte Darstellungen wahr, und zum anderen gibt ein Fernsehgerät 50 Bilder je Sekunde aus. Damit ergibt sich, daß alle Objekte, die länger als 62 ms für einen Schritt benötigen, auf dem Bild springen, und Schritte unter 20 ms Dauer nicht unbedingt auf dem Bildschirm sichtbar werden. Im letzten Fall hieße das, daß dieser Schritt zwar rechentechnisch ausgeführt würde, auf dem Bildschirm jedoch einzelne Teilschritte fehlen bzw. eine Vergrößerung der Schrittweite eintritt. Besonders störend wirkt sich diese Erscheinung bei der Bewegung von Objekten aus mehreren Zeichen aus, wo sich optisch eine Überlagerung aus Bildfrequenz und Schrittfrequenz

ergibt und dadurch nur ein Teil der Figur sichtbar wird.
Man kann demzufolge eine Geschwindigkeit von 50 Zeichen je Sekunde für eine (zeichenweise) Bewegung als Grenze ansetzen. Dies ist für die Anwendung auch ausreichend, da das bei 40 Zeichen je Zeile einer Bildüberquerung in weniger als einer Sekunde entspricht. Da allgemein eine Reaktion des Bedieners benötigt wird, werden die Objekte in der praktischen Anwendung zumeist langsamer bewegt.

4.2. Bestimmung der maximalen Objektzahl

Aus den oben getroffenen Feststellungen kann man ableiten, daß sich eine Objektbewegung möglichst im Zeitraum von kleiner 20 ms ausführen lassen sollte. Werden mehrere Objekte bewegt, so gilt diese Zeit für die Bewegung aller Bildelemente. Betrachtet man das Problem noch genauer, so muß eine Teilung der Zeit in die eigentliche Ausgabezeit und die Dauer der rechnerischen Bearbeitung erfolgen. Dies ist erforderlich, da wie oben festgestellt das Bild innerhalb von 20 ms vom Fernsehgerät einmal vollständig aufgeblendet wird. Innerhalb dieser Darstellungszeit dürfen aber keine Zugriffe auf das Bild erfolgen, da das zum Flimmern führt und eine Verwaschung der Objekte bringen kann (altes Zeichen wird noch und neues schon dargestellt). Die eigentliche Ausgabe muß also im wesentlichen innerhalb der Bildsynchronlücke und Vertikalaustastung erfolgen, was aber nur ungefähr 3 ms Zeit läßt. In einer Formel ausgedrückt bedeutet das:

$$\text{Objektzahl} \leq \frac{20\ \text{ms}}{\text{mittlere Bearbeitungszeit eines Objektes} + \text{mittlere Ausgabezeit eines Objektes}}$$

mit der Bedingung: SUMME der Ausgabezeiten ≤ 3 ms.
Es ist unschwer zu erkennen, daß damit dieser Wert stark vom verwendeten System und der Programmierung abhängt. Die wichtigsten Parameter dabei sind:
- Dauer des Bildschirmzugriffs
- Wahl des Sprachniveaus (MC, BASIC, . . .)
- Umfang der Objekte
- Struktur und Darstellung der Objekte im Rechner.

So ist es i. allg. nicht ratsam, mit den im Rechner vorhandenen Bildschirmtreibern zu arbeiten, da sie weder die geforderten Geschwindigkeiten bringen noch den Zugriff in x, y-Koordinaten erlauben. Man sollte für den Bildschirm Treiber schreiben, die ein schnelles Lesen und Schreiben von Zeichen mit kartesischen Koordinatenangaben erlauben. Vor allem bei der Bewegung mehrerer Objekte sollten die Programme in Maschinensprache geschrieben werden, da höhere Programmiersprachen, vor allem BASIC-Interpreter, zu langsam sind. Man kann sich jedoch helfen, indem man sich in diesen Sprachen Sprachelemente schafft, die maschinennah und systembezogen komplexe Leistungen anbieten.

4.3. Realisierung der Bewegung

Die vorliegenden Erfahrungen beziehen sich vor allem auf die Arbeit mit linienförmigen Objekten, sogenannten Schlangen. Sie unterscheiden sich zu flächigen Objekten nur in der Kollisionsbehandlung, da dann nicht ein Frontelement, sondern mehrere Frontelemente existieren.

Begrenzte Bewegung

Eine Sonderform der Objektbewegung stellt die dynamische Veränderung eines Bilddetails dar, welches aber als Gesamtheit an einem festen Ort stehen

bleibt. Diese Elemente lassen sich wesentlich einfacher behandeln, da bei ihnen die Wegermittlung und die Kollisionserkennung entfällt. Es ist oft auch möglich, das Objekt nicht in Form einer Zeichenkette zu verwalten, sondern aus einer Bildungsvorschrift heraus aufzubauen. Ein Beispiel dafür stellen die in Bild 7 am oberen Bildrand befindlichen Bunker dar. In diesem konkreten Fall erfolgt die Darstellung des Füllstandes intern durch einen numerischen Wert, aus dem zyklisch das entsprechende Balkendiagramm (= Füllstand) errechnet wird. Diese Bewegungen erfolgen zeitlich parallel zur Bewegung und Entladung der Züge und stellen somit für den Rechner Objekte dar.

Bewegung von Einzelzeichen

Für die Bewegung einzelner Zeichen hat sich die Arbeit mit Beschreibungsvektoren als günstig erwiesen. Man hat dabei alle notwendigen Informationen konzentriert stehen und kann in den daraus resultierenden Tabellen gut arbeiten. So ist es möglich, ein allgemeines Behandlungsprogramm zu erstellen und vor Eintritt in dieses nur noch einen Zeiger auf den entsprechenden Vektor zu richten.

Der Beschreibungsvektor setzt sich aus mindestens vier Elementen zusammen, welche die Koordinaten x und y, den Wert des Objektzeichens und den Wert des am Standort verdeckten Zeichens enthalten. Er kann ergänzt werden durch Angaben wie:

- letzte Bewegungsrichtung
- Attribute, wie z. B. Farbe,
- Zähler zur Bewertung

Der Ablauf einer Bewegung vollzieht sich wie bei der statischen Pseudographik in den Phasen:

a) Bewegungsrichtung ermitteln
b) Zeichen am alten Ort löschen und verdecktes Zeichen eintragen
c) neuen Ort testen (Kollisionsbehandlung)
d) Zeichen am neuen Ort retten und Objekt setzen.

Probleme entstehen, wenn sich mehrere Zeichen bewegen und gegenseitig überdecken. So kann der Fall eintreten, daß die Objekte A und B auf einem Punkt stehen. Zuerst wird Objekt A gesetzt und erkennt als verdecktes Zeichen ein Leerzeichen, Objekt B erkennt jedoch Objekt A. Beim Verlassen der Position wird gewöhnlich Objekt A wieder als erstes bewegt und stellt ein Leerzeichen her. Objekt B setzt anschließend Objekt A an den alten Platz, so daß dieses als Duplikat zurückbleibt. Die Lösung des Problems erfordert, daß die Rekonstruktion der verdeckten Zeichen von der optisch vordersten Ebene nach hinten erfolgt und das Setzen der Objekte von hinten nach vorn durchgeführt wird. Damit lassen sich die oben aufgeführten Punkte a) bis d) nicht mehr geschlossen für jedes Objekt einzeln abarbeiten, sondern es ist eine parallele Ausführung dieser Schritte für jeweils alle Objekte notwendig. Dabei unterscheiden sich die Punkte b) und d) in der Reihenfolge der bearbeiteten Objekte. Eine andere Behandlungsmöglichkeit wäre der Test auf gleiche Koordinaten in den anderen Vektoren und ein Zeichenaustausch zwischen diesen auf direktem Weg.

Bewegung von Zeichengruppen

Die Bewegung von linienförmigen Objekten wird intern günstigerweise ebenfalls durch die Arbeit mit Beschreibungsvektoren unterstützt. Im Gegensatz zu Einzelzeichen werden zur Beschreibung von linienförmigen Objekten zweidimensionale Matrizen verwendet und bei flächigen Objekten drei Dimensionen benötigt. Die Matrix enthält für jedes Element des Objektes die Angabe der Koordinaten, Zeichencode

und verdecktes Zeichen. Auf die Einbringung von weiteren Informationen sollte hier verzichtet werden, da diese den Speicheraufwand unnötig erhöhen. Sie können besser in einem das Gesamtobjekt beschreibenden Vektor untergebracht werden.

Um mit einer variablen Objektgröße zu arbeiten, wird die Matrix durch eine nicht genutzte oder nicht existierende Koordinate begrenzt. Dabei bietet sich rechentechnisch die Koordinate 0,0 an. Da diese physisch vorhanden ist, müßte in den Bildschirmtreibern eine Korrektur erfolgen. Die Ausgabe der Objekte erfolgt damit vom jeweiligen Tabellenanfang bis zur »Koordinate 0,0«.

Aus rechentechnischen Gründen ist es notwendig, dem eigentlichen Objekt ein vorlaufendes und ein nachlaufendes Zeichen hinzuzufügen. Das nachlaufende Zeichen wird zur Herstellung des alten Zustands hinter dem Objekt benötigt und stellt die verlassene Position dar. Das vorlaufende Zeichen hat den gleichen Zweck für eine Umkehrung der Bewegungsrichtung. Das Objekt wird demzufolge durch eine Matrix mit folgender Struktur beschrieben:

X0	Y0	V0	B0
X1	Y1	V1	B1
.	.	.	.
Xm	Ym	Vm	Bm
.	.	.	.
Xn—1	Yn—1	Vn—1	Bn—1
Xn	Yn	Vn	Bn
0	0	0	0

X *x*-Koordinate
Y *y*-Koordinate
V verdecktes Zeichen
B Objektzeichen

Das optisch ausgegebene Objekt besteht aus den Elementen B1 bis Bn—1, wobei B1 das Frontzeichen bildet. Für eine Bewegung sind mit dieser Matrix die folgenden Operationen auszuführen:

a) Ausgehend von den Koordinaten X0, Y0, wird entsprechend der Bewegungsvorschrift die neue Position ermittelt.

b) Die Matrixelemente X, Y und V werden in Richtung des nächsthöheren Index verschoben, die alten Elemente Xn, Yn und Vn werden dabei überschrieben.

c) In X0, Y0 wird die neue Position eingetragen, das verdeckte Zeichen wird ermittelt und in V0 und B0 fixiert.

d) Das neue Element Vn wird nach Bn kopiert, wodurch die hinter dem Objekt befindliche Position gelöscht wird.

e) Die Objektzeichen B0 bis Bn werden auf dem Bildschirm ausgegeben.

Soll die Bewegungsrichtung umgekehrt werden, so ist die Matrix lediglich um die Elemente Xm · · · Bm zu spiegeln und eine Bewegung nach obiger Vorschrift auszuführen. Durch diese Maßnahme kann für beide Bewegungsrichtungen ein einheitliches Programm verwendet werden.

Bewegung mit unterschiedlicher Geschwindigkeit

Die Bewegung von mehreren Objekten mit unterschiedlichen Geschwindigkeiten läßt sich durch verschiedene Möglichkeiten erreichen. So kann man, vor allem bei Einzelzeichen, mit unterschiedlichen Schrittweiten bei der Bewegung arbeiten, wobei das Variationsspektrum begrenzt ist. Eine andere Methode ist das »Stehlen« von Schritten im Bewegungsprogramm, was auch programmtechnisch sehr übersichtlich ist.

Das Programm zur Realisierung der Objektbewegung wird im Normalfall im Polling die einzelnen Objekte bearbeiten. Wird dem jeweiligen Programm, z. B. in Form einer Zelle im Beschreibungsvektor, ein Zähler vorgeschaltet, so kann die Bewegung auf den

Fall des Zählernulldurchganges begrenzt werden. Es findet also nur nach jedem *n*-ten Aufruf des Objektes tatsächlich eine Bewegung statt. Auf diese Weise können die Objekte sich in sehr unterschiedlichen und fein stufbaren Geschwindigkeitsverhältnissen befinden. Bei dieser Methode muß allerdings das interne Steuerprogramm wesentlich schneller laufen als für die Grundgeschwindigkeit erforderlich wäre.
Nach dieser Methode wurde das Spiel »Modelleisenbahnanlage« aufgebaut, wobei man bei drei Objekten noch sehr hohe Bewegungsgeschwindigkeiten erreicht.

Objektbewegung durch Leitlinien

Ein weiterer Problemkreis ist die Ermittlung der Bewegungsrichtung des Objektes. Sehr häufig treten dabei als Lösungen für die Bahnbestimmung die folgenden auf:

- Tasteneingabe,
- Steuerung durch Zufallszahl,
- Steuerung durch mathematische Funktionen.

Eine weitere Möglichkeit ist die Verwendung von Leitzeichen auf dem Bildschirm. Diese Technik wurde bei

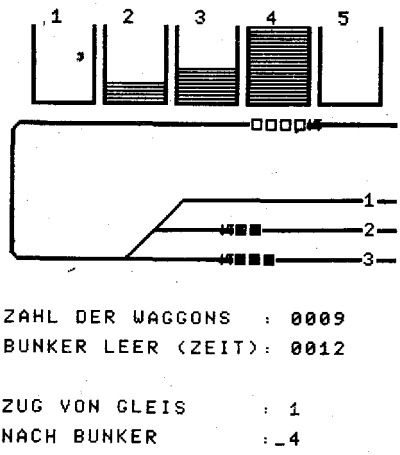


Bild 7. Spielbild zum Programm BAHN

den Bildern 1, 7 und 8 in verschiedenen Formen und Schwierigkeitsgraden verwirklicht. Der Grundgedanke besteht darin, daß dem optischen Zeichen, welches dargestellt wird, noch intern eine Bedeutung für die Bewegungsbestimmung unterlegt wird. Bei dem Spiel auf Bild 7 wurde das in einfachster Form genutzt. Es wird in diesem Fall nur ausgewertet, daß ein Gleissymbol vorhanden ist. Dazu wird vor der Lok, 45 Grad links neben dieser beginnend, im Uhrzeigersinn nach einem Gleisstück gesucht. Die Position des ersten gefundenen Zeichens ergibt dann das neue Bewegungsziel. Diese Form hat aber eine Reihe von Einschränkungen, da der Suchalgorithmus noch auf das Bild abgestimmt ist. Kritisch sind dabei die Weichen im Gleisbild, da hier (für den Rechner) zwei Abgangsrichtungen möglich sind. Er benötigt also zusätzliche Aussagen zur Ermittlung der logisch richtigen Bewegung, die hier durch die Wahl der Suchvorschrift abgefangen werden.
Bei dem Spiel in Bild 1 wird diese Leittechnik ohne sichtbare Zeichen angewendet. Hier wurde die »Straße« mit verschiedenen Leerzeichen »gepflastert«, welche die Informationen über mögliche Bewegungsrichtungen tragen. Dabei wird mit Hilfe des Zeichencodes eine Tabelle adressiert, die das zugehörige Richtungsbyte enthält. Jedes Fahrzeug verfügt mit Hilfe seines Zeichencodes über ein gleiches Byte, aus dem hervorgeht, ob es gerade, rechts oder links fahren will. Die effektive Fahrtrichtung wird ermittelt durch UND-Verknüpfung beider Bytes. Läßt das Straßenzeichen die Fahrtrichtung des Fahrzeuges zu, so wird diese ausgeführt. Ist das Ergebnis der Verknüpfung NULL, wird die Richtung, welche durch die Straße festgelegt wird, als bindend angesehen.
Die Arbeit mit diesen Leitzeichen wird

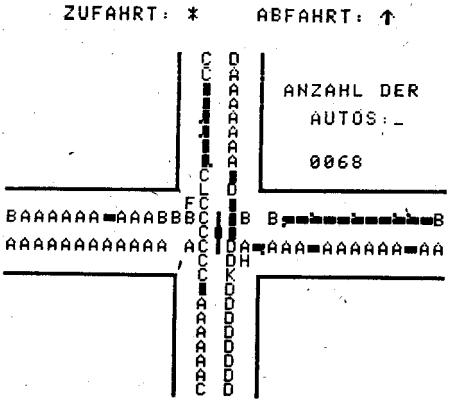


Bild 8. Spielbild KREUZUNG mit Leitsymbolen (1)

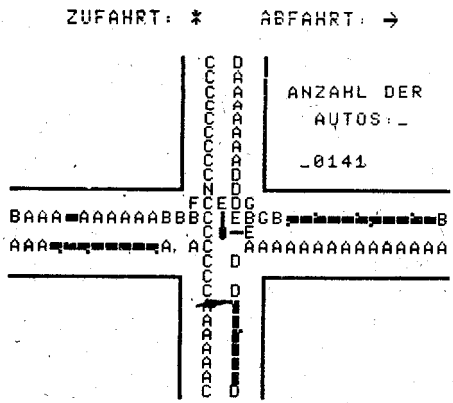


Bild 9. Spielbild KREUZUNG mit Leitsymbolen (2)

nochmals in den Bildern 8 und 9 sichtbar dargestellt. Es wird auch deutlich, daß der zentrale Bereich in Abhängigkeit vom Spielverlauf umprogrammiert wird. Mit dieser Variante kann ebenfalls sehr schnell gearbeitet werden. In dem Spiel »Kreuzung« werden programmtechnisch 12 Objektketten verwaltet, wobei die Zeichen der Objekte als Einzelzeichen bearbeitet werden. Dies hat zur Folge, daß nur die Bewegungsausgabe in Form einer Beschreibungsmatrix erfolgt, aber, z.B. durch abbiegende Fahrzeuge, das Objekt in sich keinen räumlichen Zusammenhalt besitzt.

Eine weitere Steigerung dieser Technik stellt das Spiel »Modelleisenbahnanlage« dar. Da hier das Gleisbild frei definierbar ist, müssen alle notwendigen Informationen über die Bewegungsrichtung an das Zeichen gebunden werden. Die Schwierigkeit liegt bei den Weichen und Kreuzungen. Wird auf eine Weiche spitz aufgefahren, so ergibt sich die zu wählende Richtung aus der Weichenstellung und ist eindeutig. Kommt man dagegen aus der abbiegenden Richtung und die Weiche steht auf Gerade, so hat der Rechner zwei Möglichkeiten, wovon eine eigentlich nicht gehen darf. Aus diesem Grund werden für jedes Zeichen drei Masken eingeführt, welche in 2 Bytes untergebracht sind.

Maske 1 Verhalten bei normaler Srecke bzw. freier Weiche

Maske 2 Verhalten bei gesperrter Strecke

Maske 3 verbotene Richtungen

Für die Berechnung der Bewegungsrichtung ergibt sich damit folgender Ablauf:

a) Bestimmung der Bewegungsrichtung des Zuges und damit der Richtung, aus der das Zeichen befahren wird

b) Bestimmung der gültigen Masken für das Symbol im Bild

c) Fallunterscheidung
- normale Strecke
- freie Weiche
- gesperrte Weiche.

mit Hilfe der Masken und der Information aus a)

d) Durchführung der Bewegungsrechnung.

Es ist unschwer zu erkennen, daß die Anzahl der notwendigen Parameter eines Leitsymbols mit dessen Freiheitsgraden wächst. Der daraus resul-

tierende Bearbeitungsaufwand schlägt sich in der möglichen Anzahl von Objekten bzw. in der maximalen Bewegungsgeschwindigkeit nieder.
Eine weitere Anwendung dieser Leittechnik stellt die Anwendung in einem Leiterplattenzeichenprogramm dar. Bei diesem wird die oben beschriebene Verfahrensweise noch durch den Parameter »Zeichenvergangenheit« ergänzt.

Beispielprogramme

An dieser Stelle sollen einige kurze Bemerkungen zu den in diesem Abschnitt als Anschauungsobjekte benutzten Programmen folgen.

Programm »Kreuzung« (Bild 1)

Ziel des Spieles ist, in einer vorgegebenen Zeit möglichst viele Fahrzeuge über die Kreuzung zu dirigieren. Der Rechner bewegt dabei die Fahrzeuge in Abhängigkeit von der Stellung des Polizisten auf der Kreuzung. Die zulaufenden Fahrzeuge werden durch vier Zufallsgeneratoren in Menge und Fahrtrichtung generiert. Der Spieler kann lediglich die Armstellung des Polizisten beeinflussen, wobei das StVO-gerechte Verhalten erzwungen wird (z.B. ist kein direkter Übergang von einer Dreiseitensperrung in eine andere möglich).

Programm »Bahn« (Bild 7)

Der Spieler hat als Dispatcher eines Bahnhofes für die kontinuierliche Versorgung der Kohlebunker (oben) eines Kraftwerkes zu sorgen. Ihm werden Züge mit 1 bis 4 Wagen bereitgestellt, wobei die Wagenzahl eine Zufallsgröße ist.
Als Spieleinschränkungen gelten:

a) Es kann auf dem eingleisigen Streckenteil nur ein Zug sein.

b) Wird an einem Bunker mehr angefahren als er momentan fassen kann, bleibt der Zug bis zu seiner völligen Entladung stehen (und sperrt durch a) damit das Füllen der anderen Bunker).

Die Entleerung der Bunker erfolgt kontinuierlich, aber mit individuellen Geschwindigkeiten.
Beide Spiele sind über Parametertabellen steuerbar, so daß sich der Schwierigkeitsgrad durch Änderung von Mengen, Geschwindigkeiten und anderem einstellen läßt.

Autor:
Dr.-Ing. Gert Schönfelder
Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden

Korrekturhinweis zu Heft 10

Beitrag
Polynomgrafik

```
20030 FORJ=0TO7
23030 IFJ<4THENPRINTSTRING$(J," ");:ELSEPRINT"<";J;">";
23080 IFA(J+1,I+1)>0ANDI<N-JTHENPRINT"+";
```

Wir bitten, diese Fehler zu entschuldigen.

Benchmarktests – eine Vergleichsmöglichkeit von Computerhardware und -software

Wie in jeder technischen Branche, so versucht man auch in der Computertechnik objektive Kriterien für die Bewertung der Leistungsfähigkeit entwickelter Hard- und Software zu suchen.

Naturgemäß treten hier bei den unterschiedlichen Anforderungen, die seitens der Anwender an einen Computer gestellt werden, eine Reihe gravierender Probleme und Widersprüche auf.

Häufig werden zur Charakterisierung bestimmter Eigenschaften eines Computers sogenannte »Benchmarktests« herangezogen, die vor allen Dingen Aussagen zu Programmlaufzeiten und damit zur Verarbeitungsgeschwindigkeit eines Computers liefern sollen. Bei höheren Programmiersprachen kommt aber auch solchen Größen, wie Compilier- und Linkzeit, der Länge des erzeugten Maschinencodes u.a.m. Bedeutung zu. Nun liegt es in der Natur der Benchmarktests, daß die Leistungsfähigkeit eines Computers in seiner Gesamtheit von Hard- und Software nur bezüglich einer konkreten Aufgabenstellung beschrieben werden kann. Deshalb gehen die Meinungen über Sinn und Unsinn derartiger Tests weit auseinander. Trotzdem werden sie genutzt, um wenigstens ein, wenn auch mängelbehaftetes Verfahren zum Vergleich zur Hand zu haben. Beispielsweise sind für eine Reihe von Programmiersprachen einer bestimmten Version (gleicher Dialekt) mitunter mehrere Compiler im Angebot, die sich mit Hilfe solcher Tests bei unverändertem Quelltext vergleichen lassen. Kleincomputer mit integriertem BASIC-Interpreter lassen bei gleichem Sprachumfang ebenfalls solche Tests zu.

Schwieriger wird es, wenn die Eignung unterschiedlicher Sprachen am Problem getestet wird. Hier gilt es, klar zu betrachten, welche Aussage an Hand des Tests formuliert werden kann. Vor vorschnellen Verallgemeinerungen muß an dieser Stelle gewarnt werden.

Mit den folgenden Ausführungen wird das Ziel verfolgt, eine Reihe solcher Benchmarktests einschließlich dem häufig anzutreffenden »Sieb des ERATOSTHENES« (Sieve of Eratosthenes) für die auf Kleincomputern verbreiteten höheren Programmiersprachen BASIC, Turbo-PASCAL und FORTH vorzustellen. Während BASIC leider sehr viele Dialekte vorweisen kann [1], sieht die Situation bei Turbo-PASCAL und FORTH (Standard FORTH-83) wesentlich besser aus. Diese Programme lassen sich für derartige Tests leicht auf einen anderen Rechner übernehmen, während sich bei BASIC u.U. Programmveränderungen erforderlich machen, die einer Verallgemeinerung der Aussage dann bereits wieder entgegenstehen.

Das Sieb des Eratosthenes

Der griechische Mathematiker ERATOSTHENES von Kyrene (etwa 275 bis 194 v.u.Z.) gab zum Auffinden der Primzahlen eines bestimmten Bereiches der natürlichen Zahlen das folgende Verfahren an:
»Man streicht aus einem Abschnitt der Folge natürlicher Zahlen nach der 2 jede durch 2 teilbare Zahl, dann nach der 3 jede durch 3 teilbare Zahl, dann nach der 5 jede durch 5 teilbare Zahl usw. Es bleiben gerade die Primzahlen des Bereiches stehen . . .« [2].

Dieses Verfahren ist die Grundlage für das Programm ERATOS, welches in BASIC, Turbo-PASCAL und FORTH in den Listings 1 bis 3 gegeben ist.

```
10 'programm eratos
20 DEFINT a-s
30 limit=600
40 DIM flags(limit)
50 t=TIME
60 PRINT"Start..."
70 PRINT 2;:counter=1
80 FOR i=0 TO limit:flags(i)=0:NEXT
90 FOR i=0 TO limit
100 IF flags(i)<>0 THEN 190
110  prime=i+i+3
120  PRINT prime;
130  k=i+prime
140  WHILE k<=limit
150    flags(k)=-1
160    k=k+prime
170  WEND
180  counter=counter+1
190 NEXT
200 PRINT
210 PRINT counter;" Primes"
220 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
```

Listing 1. Programm ERATOS. BAS

```
program eratos;


procedure uhrreset;
begin
inline($21/$00/$00/
       $11/$00/$00/
       $cd/$5a/$fc/$10/$bd);
end;

procedure laufzeit;

var itq:integer;
    laufzeit:real;

begin
inline($cd/$5a/$fc/$0d/$bd/
       $22/itq);
laufzeit:=itq/300;
writeln('Laufzeit =',laufzeit:8:3,' sec.');
end;
```

```
const limit = 600;

var flags: array [0..limit] of boolean;
    counter,prime,i,j,k: integer;

begin
uhrreset;
writeln('Start...');
prime:=2;
write(prime,' ');
counter:=1;
for i:=0 to limit do flags[i]:=true;
for i:=0 to limit do if flags[i] then
   begin
   prime:=i+i+3;
   write(prime,' ');
   k:=i+prime;
   while k<=limit do
      begin
      flags[k]:=false;
      k:=k+prime;
      end;
   counter:=counter+1;
end;
writeln;
writeln(counter,' Primes');
laufzeit;
end.
```

Listing 2. Programm ERATOS. PAS

```
Screen #    1
( UHR

: UHRRESET  ( -- )
    0. !TIME ;

: UHRLESEN  ( -- )
    @TIME DROP 1000 300 */ S>D
    <# # # # A9CII . HOLD #S #> TYPE ."  sec." ;

: LAUFZEIT  ( -- )
    ." Laufzeit = " UHRLESEN ;


Screen #    2
( ERATOS

600 CONSTANT LIMIT

  0 CONSTANT PRIMES

CREATE FLAGS LIMIT 2* ALLOT

: FLAG ( i -- addr )     2* FLAGS + ;

: ALLE (  -- n1 n2 )     LIMIT 0 ;

: NULL (  --  )          ALLE DO 0 I FLAG ! LOOP ;              -->


Screen #    3
( ERATOS
```

```
: ERATOS     (  --  )
   UHRRESET CR ." START..." CR
   2 . 1 NULL                          ( 1. Primzahl )
   ALLE DO I FLAG @ 0=
      IF I DUP + 3 + DUP TO PRIMES DUP . I +
         BEGIN
           DUP LIMIT < WHILE
           DUP FLAG -1 SWAP ! PRIMES +
         REPEAT
         DROP 1+
      THEN
   LOOP
   CR . ." Primes"
   CR LAUFZEIT ;
```

Listing 3. Programm ERATOS. 4TH

Tabelle 1

	BASIC	Turbo-PASCAL	FORTH	
Laufzeit/s	8,11	0,31	1,16	ohne Ausgabe
	10,22	2,45	3,53	mit Ausgabe
Speicherplatzbedarf/Byte	1742	8704	1426	

Der verwendete Rechner verfügt über eine eingebaute Uhr, die über die Funktion TIME in BASIC abgefragt werden kann. In Turbo-PASCAL wurde mittels INLINE die Firmware des Rechners direkt angesprochen. In der hier verwendeten FORTH-Version stehen über den Standard FORTH-83 hinaus die Worte !TIME und **TIME zur Arbeit mit der Uhr zur Verfügung. Kann man auf solche oder ähnliche Unterstützung zur Laufzeitmessung nicht zurückgreifen, dann kann man nur zur Stoppuhr greifen und ggf. die Zahl der Schleifendurchläufe erhöhen.

In Tabelle 1 sind die Laufzeiten des Programms ERATOS mit und ohne Ausgabe der einzelnen Primzahlen auf dem Bildschirm für einen Schneider CPC 6128 (CPU Z 80 A) angegeben.

Die gewonnenen Ergebnisse stimmen mit den Erwartungen insofern überein, daß BASIC als Interpretersprache entsprechende Laufzeiten erwarten läßt, während durch die Compilierung bei Turbo-PASCAL direkter Maschinencode erzeugt wird, dessen Abarbeitung hinreichend schnell ist. FORTH liegt hinsichtlich seiner Laufzeit sehr günstig, kann aber die Ausführungszeiten des compilierten PASCAL-Programms nicht erreichen.

Numerische Tests

In den Listings 4 bis 6 sind einige Tests zusammengefaßt, die zur Ermittlung der Laufzeit einer Reihe von Integeroperationen sowie der Vergleichsoperation Verwendung finden können. Naturgemäß ergeben sich hier wieder Probleme hinsichtlich der Vergleichbarkeit an sich. In FORTH ist die zentrale Stelle zur Parameterübergabe der Parameterstack. Die Arbeit mit Variablen ist für die beiden anderen Programmiersprachen BASIC und Turbo-PASCAL kennzeichnend. In FORTH wurden die Tests deshalb sowohl mit Stackoperationen als auch unter Verwendung von Variablen durchgeführt.

```
10 ' benchmarktests
20 DEFINT a-s
30 limit=30000
40 CLS
50 '
60 PRINT"Leere Schleife"
70 t=TIME
80 FOR i=0 TO limit:NEXT
90 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
100 '
110 PRINT"Integeraddition"
120 t=TIME
130 a=1:b=1
140 FOR i=0 TO limit:a=a+b:NEXT
150 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
160 '
170 PRINT"Integersubtraktion"
180 t=TIME
190 a=32000:b=1
200 FOR i=0 TO limit:a=a-b:NEXT
210 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
220 '
230 PRINT"Integermultiplikation"
240 t=TIME
250 a=32000:b=1
260 FOR i=0 TO limit:a=a*b:NEXT
270 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
280 '
290 PRINT"Integerdivision"
300 t=TIME
310 a=32000:b=1
320 FOR i=0 TO limit:a=a/b:NEXT
330 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
340 '
350 PRINT"Vergleich"
360 t=TIME
370 a=32000:b=10000
380 FOR i=0 TO limit: IF a>b THEN:NEXT
390 PRINT"Laufzeit =";ABS(t-TIME)/300;" sec."
```

Listing 4. Programm BENCH. BAS

```
program benchmarktests;


procedure uhrreset;
begin
inline($21/$00/$00/
       $11/$00/$00/
       $cd/$5a/$fc/$10/$bd);
end;

procedure laufzeit;

var itg:integer;
    laufzeit:real;

begin
inline($cd/$5a/$fc/$0d/$bd/
       $22/itg);
laufzeit:=itg/300;
writeln('Laufzeit =',laufzeit:8:3,' sec.');
end;
```

```
const limit = 30000;

var i,a,b:integer;

{ leere Schleife }
procedure loop;
begin
writeln('Leere Schleife');
uhrreset;
for i:= 0 to limit do;
laufzeit;
end;

{ Integeraddition }
procedure addition;
begin
writeln('Integeraddition');
uhrreset;
a:=1;
b:=1;
for i:=0 to limit do a:=a+b;
laufzeit;
end;

{ Integersubtraktion }
procedure subtraction;
begin
writeln('Integersubtraktion');
uhrreset;
a:=32000;
b:=1;
for i:=0 to limit do a:=a-b;

laufzeit;
end;

{ Integermultiplikation }
procedure multiplication;
begin
writeln('Integermultiplikation');
uhrreset;
a:=32000;
b:=1;
for i:=0 to limit do a:=a*b;
laufzeit;
end;

{ Integerdivision }
procedure division;
begin
writeln('Integerdivision');
uhrreset;
a:=32000;
b:=1;
for i:=0 to limit do a:=a div b;
laufzeit;
end;

{ Vergleich }
procedure compare;
begin
writeln('Vergleichsoperation');
uhrreset;
a:=32000;
b:=10000;
for i:=0 to limit do if a<b then;
laufzeit;
end;


begin
loop;
addition;
subtraction;
multiplication;
division;
compare;
end.
```

Listing 5. Programm BENCH. PAS

```
Screen #   0


Screen #   1
( BENCHMARKTESTS

30000 CONSTANT LIMIT

VARIABLE A      1 A !
VARIABLE B      1 B !

: ALLE LIMIT 0 ;

: SCHLEIFE  CR ." Leere Schleife" CR
            UHRRESET
            ALLE DO LOOP
            LAUFZEIT ;                    -->


Screen #   2
( BENCHMARKTESTS
```

```
: ADDS  CR ." Integeraddition im Stack" CR
        UHRRESET
        ALLE DO 1 1 + DROP LOOP
        LAUFZEIT ;

: ADDV  CR ." Integeraddition mit Variablen" CR
        UHRRESET
        ALLE DO A @ B @ + A ! LOOP
        LAUFZEIT ;

: SUBS  CR ." Integersubtraktion im Stack" CR
        UHRRESET
        ALLE DO 10 9 - DROP LOOP
        LAUFZEIT ;                                          -->
```

Z-80 cpcForth 1.1 (Forth-83) by Forth-Systeme A. Flesch 1985

```
Screen #   3
( BENCHMARKTESTS

: SUBV  CR ." Integersubtraction mit Variablen" CR
        UHRRESET
        ALLE DO A @ B @ - A ! LOOP
        LAUFZEIT ;

: MULS  CR ." Multiplikation im Stack" CR
        UHRRESET
        ALLE DO 1 2 * DROP LOOP
        LAUFZEIT ;

: MULV  CR ." Multiplikation mit Variablen" CR
        UHRRESET
        ALLE DO A @ B @ * A ! LOOP
        LAUFZEIT ;                                          -->


Screen #   4
( BENCHMARKTESTS

: DIVS  CR ." Integerdivision im Stack" CR
        UHRRESET


        ALLE DO 5 2 / DROP LOOP
        LAUFZEIT ;

: DIVV  CR ." Integerdivision mit Variablen" CR
        UHRRESET
        ALLE DO A @ B @ / A ! LOOP
        LAUFZEIT ;

: COMP  CR ." Vergleich" CR
        UHRRESET
        ALLE DO 2 1 > IF THEN LOOP
        LAUFZEIT ;                                          -->


Screen #   5
( BENCHMARKTESTS

: BMT
        CLS
        SCHLEIFE
```

```
ADDS    ADDV
SUBS    SUBV
MULS    MULV
DIVS    DIVV
COMP

Z-80 cpcForth 1.1 (Forth-83)  by Forth-Systeme A. Flesch 1985
```

Listing 6. Programm BENCH. 4TH

Tabelle 2

Laufzeit/s @ Test	BASIC	Turbo-PASCAL	FORTH
Leere Schleife	13,127	1,043	1,303
Integer-addition	52,140	1,937	5,323 8,186
Integer-subtraktion	51,810	2,043	5,450 8,250
Integermulti-plikation	53,98	6,983	14,076 16,940
Integer-division	114,883	17,113	22,670 25,500
Vergleich	50,523	3,043	6,703
Speicher-platzbedarf in Byte	893	8960	918

Die Tabelle 2 zeigt wieder die gewonnenen Ergebnisse.
Auch für diese Tests ist das Ergebnis erwartungsgemäß ausgefallen. Für FORTH wurden an den betreffenden Stellen jeweils zwei Laufzeiten notiert. Die längere Laufzeit entsteht durch die Verwendung von Variablen und resultiert aus den erforderlichen Speicherzugriffen.
Der Speicherplatzbedarf wurde generell für die ausführbaren Programme angegeben. Das ist beim (Interpreter-) BASIC der Quellcode, beim Turbo-PASCAL die entstandene COM-Datei einschließlich der Laufzeitprozeduren und bei FORTH die durch Compilierung entstandene Dictionary-Erweiterung [3]. Diesen Randbedingungen ist bei einem Vergleich unbedingt Beachtung zu schenken.
Die vorgestellten Benchmarktests wurden auf einem Schneider CPC 6128 (mit der CPU Z 80 A) durchgeführt, für den beim Autor die verwendeten drei Programmiersprachen laufen.
Um die Aussagen auch auf Computer aus eigener Produktion zu erweitern, wurden die Turbo-PASCAL-Tests zusätzlich auf einem Bürocomputer A 5120 (mit CPU UB 880 D) vorgenommen.
Die Laufzeit für das Sieb des ERATOSTHENES ist trotz des langsamer getakteten Mikroprozessors mit einer Sekunde nur halb so lang wie im vorgestellten Beispiel. Diese Laufzeitverbesserung ist auf die wesentlich schnellere Ausgabe über den (alphanumerischen) Bildschirm des Bürocomputers zurückzuführen.
Bei den numerischen Tests läßt der Bürocomputer längere Laufzeiten erwarten, die aber mit einem Faktor von 1,24 auch unter dem Verhältnis der Taktfrequenzen von 1,6 liegen.
Die Ergebnisse der Tests unterstreichen die in der Einleitung gemachten Aussagen deutlich, weshalb an dieser Stelle nochmals vor unzulässigen Verallgemeinerungen von Ergebnissen solcher Tests gewarnt werden soll.

Autor:
Dr.-Ing. Claus Kühnel
Zschertnitzer Str. 52
Dresden
8020

(Literatur s. S, 56)

Vollgraphik-Bildschirmansteuerung

In diesem Beitrag soll eine Bildschirm-Ansteuerbaugruppe beschrieben werden, die es erlaubt, einen beliebigen Bildpunkt innerhalb einer Matrix von 512x256 Bildpunkten darzustellen. Diese Baugruppe läßt sich innerhalb industrieller Systeme, z.B. MC 80 oder K 1520, wie auch in Eigenbau- oder Amateurcomputern verwenden.

Nachdem in [1] auf einfache Lösungen des Kommunikationsproblems Rechner <--> Bediener bei der Konzipierung von Meß- und Steuergeräten auf Mikrorechnerbasis eingegangen wurde, soll hier nun eine Bildschirm-Ansteuerbaugruppe beschrieben werden, bei der auf den Einsatz eines integrierten Graphik- bzw. Video-Controllers verzichtet wurde.
Den Kern der zu beschreibenden Lösung bildet die bereits in [2] vorgestellte Video-Controller-Schaltung, mit deren Hilfe es möglich ist, auf einem normalen Fernsehbildschirm ein Bild im Raster von 512 x 256 Bildpunkten zu erzeugen. Weiterhin bietet diese Baugruppe die Möglichkeit, den Bildeindruck durch Anwendung des Zeilensprungverfahrens zu verbessern. Bild 1 zeigt dies am Beispiel des Buchstaben 'E'.
Voraussetzung dafür ist jedoch der Einsatz einer Monitorbildröhre mit gegen-

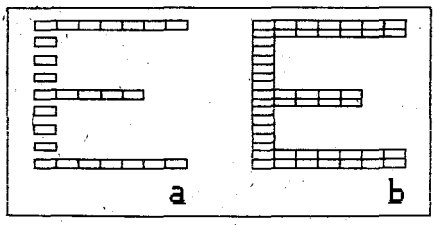


Bild 1: Zeichendarstellung a - ohne und b - mit Zeilensprung

über einer Fernsehbildröhre längeren Nachleuchtdauer.

Um die Anwendung vollgraphischer Bilddarstellungen in Amateurelektronikerkreisen zu fördern, soll neben der Hardware-Lösung auch ein entsprechendes Software-Paket angegeben und erläutert werden.

Hardware-Lösung

Bild 2 zeigt die Prinzipschaltung der Vollgraphik-Bildschirmansteuerung. Ein Video-Controller liefert neben dem fernsehspezifischen Austast- und Synchronsignal die Adressen zur Ansteuerung des 16 kByte-Bildpunktspeichers. Darüberhinaus liefert diese Taktzentrale der Baugruppe auch sämtliche Signale zur Steuerung des Bildpunkt-Schieberegisters, dessen Aufgabe in der Parallel-Serien-Wandlung der aus dem Bildpunktspeicher ausgelesenen Datenbytes besteht. Ein weiterer Teil der Schaltung realisiert die Synchronisation des Prozessorzugriffes (Bild lesen bzw. Bild schreiben) auf die internen Zeitabläufe der Bilderzeugung. Da es sich bei den verwendeten Speicherbauelementen um dynamische RAM handelt, kommt der Speicher-Ansteuerung über die RAS/CAS-Logik besondere Bedeutung zu.
Die Baugruppe enthält weiterhin einen Verwaltungsport, einerseits zur Bereitstellung der Helligkeit des Bildrandes (unbeschriebener Teil des Bildschirmes) und andererseits zur Verwaltung des Bildpunktspeichers innerhalb des Prozessoradreßraumes. Notwendig ist dies, da

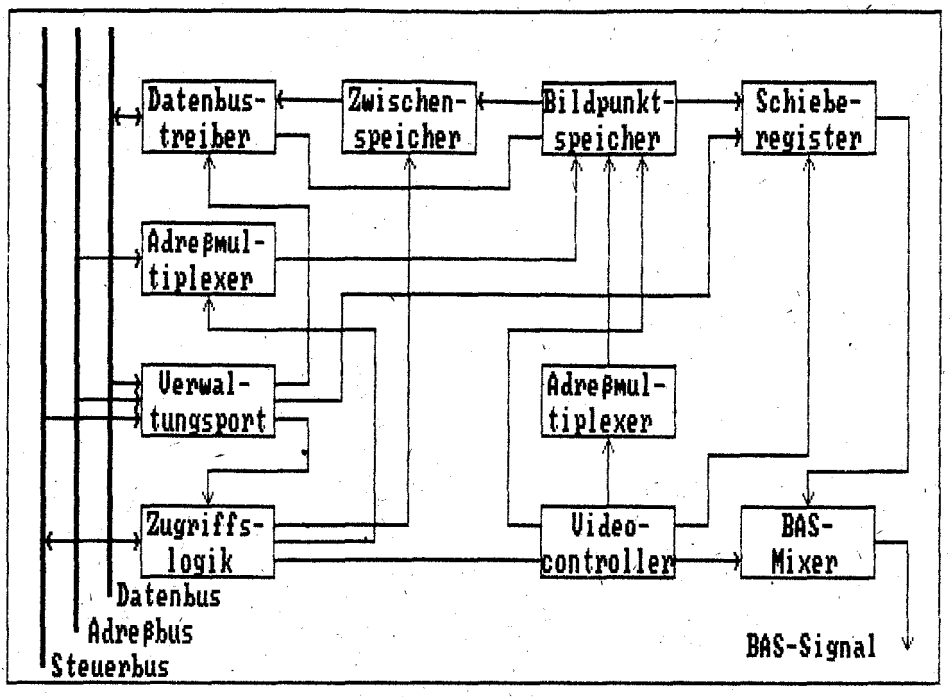


Bild 2: Prinzipschaltbild der Vollgraphik-Bildschirmansteuerung

der 16 kByte-Bildpunktspeicher nicht direkter Bestandteil des 64 kByte-Prozessoradreßraumes ist, sondern sich in einer Hintergrundebene befindet. Diese Arbeitsweise erscheint bei der Größe des Bildpunktspeichers angezeigt und befindet sich in Übereinstimmung mit der Verwaltung größerer RAM-Baugruppen (Speicherbank-Umschaltung bis zu 16x16 kByte) innerhalb des Mikrorechnersystems KMU 880, s. z.B. [3].
Letztlich liefert die vorzustellende Baugruppe alle für eine eventuelle Erweiterung um 32 kByte-Bildpunktspeicher notwendigen Signale, wodurch die Realisierung von insgesamt 3 Farbebenen oder 7 Graustufen ermöglicht wird. Dabei besitzen alle drei 16 kByte-Bildpunktspeicher identische Anfangsadressen und die Umschaltung der Speicherbänke erfolgt mit Hilfe des Verwaltungsport's.
In Bild 3 ist die Zuordnung der einzelnen Bildpunkte zu den Bit's innerhalb eines Bytes des Bildpunktspeichers dargestellt. Darüberhinaus kann dieser Darstellung auch die Adreßzuordnung der Bytes zu den Bildpositionen entnommen werden. Dabei bezeichnet 'RAM' die symbolische Anfangsadresse des 16 kByte-Bildpunktspeichers.
Eine Bildzeile ist in 64 Byte abgelegt, wobei die einzelnen Bildpunkte von links beginnend in den 8 Bit eines Bytes abgelegt werden. Alle 256 dieser Bildzeilen sind im 16 kByte-Bildpunktspeicher direkt hintereinander abgelegt, wodurch sich die Adreßrechnung bei Bildmanipulationen stark vereinfacht.

Praktische Realisierung

Die praktisch erprobte Schaltung der Vollgraphikansteuerung ist in Bild 4 dargestellt (Faltblatt im Anhang).
Um die Übersicht zu behalten ist es günstig, die Schaltung in 3 Teilschaltungsbereiche zu unterteilen: den Videocontroller, den Bildwiederholspeicher mit seinen Treibern und die Kartenverwaltungslogik.
Der Videocontroller ist in der linken unteren Hälfte des Schaltplanes dargestellt und besteht im wesentlichen aus den Schaltkreisen D31 bis D55.
Der Quarzgenerator (V1, V2, R23-R26, C3-

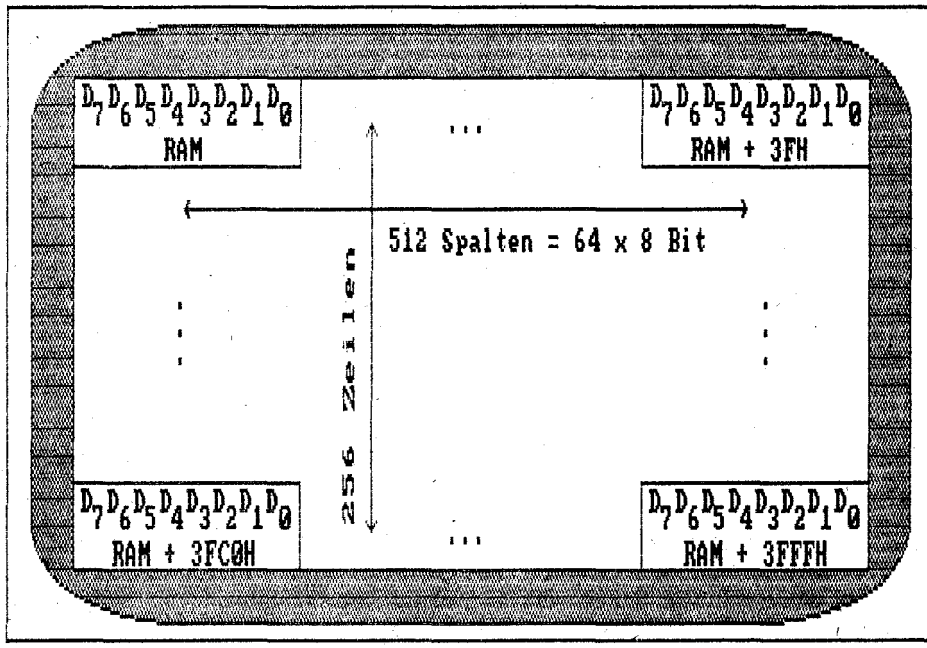


Bild 3: Bildorganisation und Adreßzuordnung des 16 kByte-Bildpunktspeichers

C5, Q1, L1) erzeugt eine Taktfrequenz von 24,5 MHz. Über einen Inverter (D55.1) gelangt diese an den Eingang des Binärteilers D54. An dessen Ausgang 1 steht nun der eigentliche Bildpunkttakt von 12,25 MHz zur Verfügung (Signal 21). Daraus ergibt sich eine Bildpunktdauer von 81,6 ns. Mit der genormten Zeilendauer von 64 µs liegt die Zeilenlänge mit 784 Bildpunkten fest. Da der Bildwiederholspeicher in einer Breite von 8 Bit organisiert ist, ist es sinnvoll, Zeichenplätze der gleichen Breite zu definieren. Von diesen gibt es nun 98 in jeder Zeile. Nach Abzug der in die Horizontalaustastlücke fallenden 18 Zeichenplätze und der je 8 Zeichenplätze für den linken und rechten Bildrand verbleiben noch 64 echte Zeichenpositionen zur Bilddarstellung.

Der Bildpunktzähler D54 (Signale 18-20) steuert über D53 und D51/52 die Abläufe beim Auslesen des Bildwiederholspeichers durch den Videocontroller und stellt den Zeichentakt (Signal 18) zur Verfügung. Über eine Torschaltung (D49.1, D55.6), die eine Verdopplung der Taktfrequenz ermöglicht, gelangt der Zeichentakt an den Eingang des Horizontalzählers (D50, D45, D46, D48). Um den Horizontalzähler auch gleichzeitig als Adresszähler für den Bildwiederholspeicher benutzen zu können, wird dieser am Ende der Austastlücke auf den Zahlenwert '-8' geladen (8 Zeichenplätze für den linken Rand). Der Dekoder D46 wertet die Zählerstände '72' (Ausgang 2 aktiv - Beginn der Horizontalaustastlücke), '82' (Ausgang 5 aktiv - Ende des Horizontalsynchronimpulses) und '90' (Ausgang 7 aktiv - Beginn der neuen Zeile - Laden des Horizontalzählers) aus.

Das Flipflop D41.1 realisiert den Horizontalaustastimpuls während D41.2 den Horizontalsynchronimpuls erzeugt.

Gleichzeitig wird mit der Vorderflanke des Synchronimpulses der Vertikalzähler (D33, D34, D38.2) weitergeschaltet. Um auch diesen für die Adressbildung des Bildwiederholspeichers direkt verwenden zu können, muß auch er zunächst auf einen negativen Zählerstand (-18) geladen werden. Da das komplette Fernsehbild aus zwei um eine Zeile versetzten Halbbildern besteht, beginnt am oberen Bildrand das darzustellende Bild erst nach 36 Fernsehzeilen. Danach kommmen die eigentlichen Bildzeilen (Zählerstände '0'

bis '255'), gefolgt vom unteren Bildrand (Zählerstände '256' bis '274'). Die Schaltkreise D35.2 und D40 dekodieren das Erreichen des Bildendes mit dem Zählerstand '275' und setzen das Flipflop D36.1, welches seinerseits den Vertikalaustastimpuls erzeugt. Durch das Kippen dieses Flipflops erfolgt eine Entaktivierung der Setzeingänge des Vertikalsynchronablaufzählers D37, D38.1 und damit dessen Freigabe. Je nach Lage des Flipflops D36.2, welches die Art des durchlaufenen Halbbildes (gerade oder ungerade Zeilen) festhält, ergibt sich ein unterschiedlicher Synchronimpulsablauf.

Für den Fall, daß das Flipflop D36.2 am Ausgang L-Pegel führt (Brücke X5 offen - Zeilensprung aktiviert), wird zunächst der Ausgang 0 des Dekoders D43 freigegeben (D37 ist auf '0' geladen). Dieser lädt den Horizontalzähler auf einen Zählerstand, der der Zeilenmitte entspricht. Der nächste Horizontalzyklus ist nun zeitmäßig nur eine halbe Zeile lang, wodurch der nötige Halbzeilenversatz erzeugt wird.

Mit der nächsten Horizontalsynchronimpulsflanke wird der Ablaufzähler D37 weitergeschalten und der Ausgang 1 des Dekoders D43 setzt das Doppelgeschwindigkeitsflipflop D44.2. Dadurch schaltet das Tor D49.1 um und der Horizontalzähler läuft mit doppeltem Zeichentakt. So wird ein auf die Hälfte der Zeit verkürzter Horizontalablauf realisiert, wie er für die Erzeugung der Vor- und Nachtrabanten sowie des Vertikalsynchronimpulses notwendig ist.

Nach fünf Horizontalsynchronimpulsen, also nach den fünf Vortrabanten, wird der Ausgang 6 von D43 aktiv und auch das zweite Torflipflop D44.1 gekippt. Mit Hilfe des Tores D49.2 wird die Rücksetzleitung des Synchronimpulsflipflops D41.2 an die Dekoderlogik D47/D55 geschaltet, die beim Zählerstand '55' aktiv wird. Auf diese Weise werden Synchronimpulse mit Überlänge erzeugt (von Horizontalzählerstand '72' bis '55' statt von '72' bis '82'), wie es für den Vertikalsynchronimpuls notwendig ist. Das Gatter D42.1 setzt das Torflipflop D44.1 nach fünf derartigen Impulsen in seine Ruhelage zurück. Nach weiteren fünf Synchronimpulsen (den Nachtrabanten) kippt das Ablaufzählerflipflop D38.1 am Ausgang Q nach H-Pegel. Da im betrachteten Fall das Halbbildflipflop D36.2 am Ausgang L-Pegel führt, wird über die Gatter D39.2 und D39.3 das Doppelgeschwindigkeitsflipflop D44.2 zurückgesetzt und nun wieder der ganz normale Horizontalzählerablauf realisiert. Die RC-Glieder R17/27 und C1/2 dienen nur der Unterdrückung von Störnadeln durch fehlerhafte Dekodierung infolge von Laufzeiteffekten (Bauelementestreuung der Gatter-Verzögrungszeiten).

Das Ende des Vertikalaustastimpulses wird durch das Gatter D42.2 erkannt. Der L-Pegel am Ausgang von D42.2 lädt den Vertikalzähler D33/D34/D38.2 und kippt das Vertikalaustastflipflop D36.1 in die Ruhelage zurück. Der L-Pegel am Ausgang Q von D36.1 setzt nun den Ablaufzähler D37/D38.1 in den Ausgangszustand zurück. Dadurch verschwindet der L-Pegel am Ausgang von D42.2, wodurch das Halbbild-Flipflop D36.2 gekippt wird. Im nun folgenden Halbbild wird ein ähnlicher Zyklus durchlaufen. Da der Ausgang des Halbbildflipflops D36.2 H-Pegel führt wird der Ablaufzähler D37 auf den Wert '1' geladen und der Ausgang 0 von D43 kann nicht aktiviert werden. Der Vertikalaustastzyklus beginnt deshalb gleich mit dem Senden der Vortrabanten. Die halbe Zeile wird hier dadurch eingeschoben, daß das Doppelgeschwindigkeitsflipflop D44.2 erst dann zurückkippt, wenn der Ablaufzähler D37 am Ausgang 1 wieder H-Pegel führt (über D39.1 dekodiert). Das bedeutet, daß vor Beginn des normalen Horizontalzählerablaufes noch ein Zyklus doppelter Geschwindigkeit und damit ein zusätzlicher, nun sechster Nachtrabant eingefügt wird. Um ein Erscheinen dieses Nachtrabanten im Synchronimpuls zu verhindern, wird über das Gatter D41.3 das Tor D32.1 für diese Zeit gesperrt. Der Zyklus dieses Halbbildes läuft bei gesperrtem Zeilensprung ständig ab (Brücke X5 geschlossen).

Die Gatter D32 und D31 verknüpfen die Synchronimpulse (von D41.2) mit den Horizontal- (von D41.1) und Vertikalaustastimpulsen (von D36.1) und dem eigentlichen Bildinhalt (von D29) zu einem normgerechten BAS-Signal. Die Pegelverhältnisse werden über die Widerstände R18 bis R20 eingestellt.

Der Videocontroller neben den Austast- und Synchron-Signalen den 14-Bit-Adreßbus für den Bildwiederholspeicher (Signale 1 bis 6 und 8 bis 15 des internen Busses).

Der Bildwiederholspeicher besteht aus den Schaltkreisen D2 bis D5, D10 sowie D16 bis D30. Der eigentliche Bildwiederholspeicher (D16 - D23) ist zu 16 kByte Länge und 8 bit Breite organisiert.

Die 8-Bit Bustreiber vom Typ DS 8212 werden in der Schaltung als Adreßmultiplexer für die dynamischen RAM's eingesetzt, wobei je ein Multiplexerpaar

für die Zugriffe des Videocontrollers (D4, D5) und die des Prozessors (D2, D3) auf den Bildwiederholspeicher zuständig ist. Die Umschaltung zwischen höherwertigem und niederwertigem Adreßteil sowie zwischen externem und internem Bus wird über die Freigabe-Signale CS1, CS2 der Multiplexer realisiert. Eine Zwischenspeicherung der Adressen erfolgt nicht. Zur Vermeidung von Störungen sind in alle Adreß- und Steuerleitungen Dämpfungswiderstände (R5 - R14) eingefügt, jedes RAM besitzt eigene Abblockkondensatoren für alle drei Versorgungsspannungen.

Die sich aller 8 Bildpunkttakte wiederholenden Steuersignale zum Auslesen des RAM erzeugt der Dekoder D53 sowie die Flipflops D51/D52.1. Um das Entstehen von Störimpulsen an den Ausgängen des Dekoders wirksam zu verhindern, wird dieser durch den Ausgang 1 von D54 verknüpft. Der Zyklus beginnt mit dem Setzen der Flipflops D51, D52 durch den Ausgang 7 von D53. Damit sind sowohl RAS als auch CAS inaktiv und die niederwertigen Adressen (D5 oder D3 aktiv) liegen an den RAM's an. Zwei Bildpunkttakte später wird, mit dem Rücksetzen von D51.2 durch den Ausgang 5 des Dekoders (D54 zählt rückwärts !), das RAS-Signal aktiviert (L-Pegel). Mit Hilfe von Ausgang 4 schaltet einen Takt danach das Flipflop D51.1 um. Jetzt werden die höherwertigen Adressen (je nach Zugriff entweder D2 oder D4 aktiv) zum RAM durchgeschaltet. Der Ausgang 3 von D53 erzeugt wieder einen Takt später über D52.1 und D30.1 L-Pegel am CAS-Eingang der Speicherschaltkreise. Jetzt ist die vollständige Adresse in den RAM übernommen. Drei Bildpunkttakte später beginnt im Speicherlesezyklus mit der H-L-Flanke am Ausgang 0 die Weiterverarbeitung des RAM-Inhalts. Danach startet der Zyklus mit dem H-Pegel für RAS und CAS erneut. Unterbrochen wird dies nur in der Vertikalaustastlücke während des Schnelllaufs des Horizontalzählers (über E3 von D53), um eventuellem Datenverlust durch unvollständige (zeitlich zu kurze) Speicherlesezyklen vorzubeugen.

Zur Bilddarstellung wird der Inhalts des Bildwiederholspeichers während der Aktivzeit von Ausgang 0 des Dekoders D53 parallel in die Schieberegister D26 und D29 übernommen. Mit der L-H-Flanke am Takteingang von D52.2 übernimmt dieses Flipflop die Information der Dekoder D35.1/D55.2, ob ein Laden der Schieberegister erwünscht ist (H-Pegel am D-Eingang) oder statt dessen die Information des Border-Ports D16 seriell verschoben wird (L-Pegel am D-Eingang, d.h. Strahl außerhalb des Zeichendarstellungsbereiches). Einen Bildpunkttakt lang wird der Steuereingang MC von D26/D29 auf Parallelladen gelegt. Mit der nächsten H-L-Flanke des Signals 21 (durch D55.3/D55.4 getrieben) wird sowohl die Bildinformation übernommen als auch das Flipflop D52.2 rückgesetzt und die Schieberegister auf seriellen Betrieb umgestellt. In den folgenden 7 Takten wird der Bildinhalt aus den zwei Registern auf den Schirm geschoben. Gleichzeitig wandert die Randinformation über ES in D26 hinein, so daß beim Ausbleiben des Parallelladeimpulses an MC beim 8.Takt sofort die Rand-Farbe (border) ins Bild eingeblendet wird.

Sowohl das Auslesen als auch das Beschreiben des Bildwiederholspeichers durch den Prozessor kann nur in der Horizontalaustastlücke erfolgen (Reset-Eingang von D10.2 auf H-Pegel). Das Flipflop D10 synchronisiert die Prozessorzugriffe mit den internen Speicher-Zyklen, wie es anhand wichtiger Signale im Bild 5 als Taktdiagramm dargestellt ist. Wenn die Bildschirmkarte selektiert ist (D-Eingang von D10.1 H-Pegel), so kippt der Ausgang von D10.1 mit der fallenden Flanke von /MREQ ebenfalls auf H-Pegel und meldet einen Prozessorzugriff an. Solange sich D10.1 in diesem Zustand befindet, wird über R1 und V3 der Prozessor in den Wait-Zustand versetzt. Erst in der Austastlücke wird D10.2 freigegeben und kann mit dem Beginn des nächsten RAS/CAS-Zyklus (Ausgang 7 von D53 kippt den Ausgang von D10.2 auf H-Pegel) die Tore D30.4 und D15.4 öffnen. Durch L-Pegel am negierten Ausgang von D10.2 werden nun auch die externen Multiplexer D2 und D3 selektiert, über die die vom Prozessor gewünschte Adresse in den nächsten 6 Takten an die RAM's gelangt. Die H-L-Flanke an Ausgang 0 von D53 lädt den Zwischenspeicher D25/D28 mit dem jeweiligen Inhalt des Bildwiederholspeichers und setzt die Voranmeldung von D10.1 zurück. Nun kann der Prozessor wegen der Aufhebung des Wait-Zustandes über die Datenbustreiber D24 und D27 den Inhalt der Zwischenspeicher D25, D28 einlesen. Wenn der Ausgang 7 von D53 erneut aktiv wird, kippt auch D10.2 in seine Ausgangslage zurück.

Das Beschreiben des Bildwiederholspeichers erfolgt in analoger Weise. Der Prozessor verharrt nach dem Ausgeben der Bildadresse und des einzuschreibenden Datenworts so lange im Wait-Zustand bis die Austastlücke erreicht und der erste RAS/CAS-Zyklus, bei dessen Beginn (getort über D10.2 und D15.4/D14.4) auch

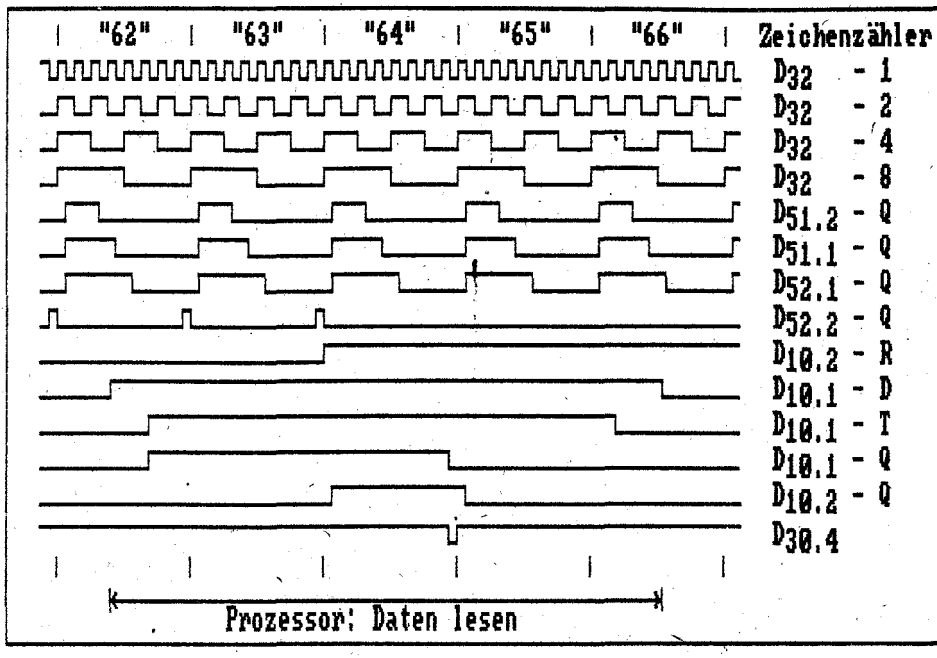


Bild 5: Impulsübersicht der Zugriffs-Steuersignale

/WE aktiviert wird, durchlaufen ist. Über die Datenbustreiber wird nun das Datenwort in den Bildwiederholspeicher eingeschrieben.

Die Kartenverwaltung erfolgt mit Hilfe von zwei Verwaltungsports. Die Gatterschaltkreise D7 bis D9 generieren die Adressen für den Kartenverwaltungsport D11 und den Border-Port D16 unter Einbeziehung der /IORQ- und /WR-Signale. Die OUT-Adressen liegen bei 90H für D11 und 91H für D16 unveränderlich fest. Die Ports sind für die Verwaltung von maximal drei 16 kByte-Bildwiederholspeichern ausgelegt.

Mit Hilfe von D16, der über die drei niederwertigen Ausgänge die Hell- oder Dunkel-Steuerung des Bildschirmrandes (ebenfalls für 3 x 16k projektiert) organisiert, ist über den vierten Ausgang 8 und die Gatter D31.1, D31.2 und D32.3 eine Invertierung des Bildes einschließlich der Ränder möglich. Dadurch ist es je nach Geschmack möglich, als "normale Darstellung" entweder schwarz auf weiß oder umgekehrt festzulegen.

Um bei einem RESET-Signal definierte Verhältnisse zu schaffen, sind beide Verwaltungsports an die Prozessor-RESET-Leitung angeschlossen. Die Karte ist nach dem RESET für den Prozessor deselektiert, der Rand ist hellgetastet und der Speicherinhalt wird nichtinvertiert zur Anzeige gebracht.

Wird über die Adresse 90H z.B. das Datenwort 03H in den Verwaltungsport eingeschrieben, so ist die Karte ab der über X3 und X4 festgesetzte Adresse in den Prozessoradressraum eingeblendet und der vorhandene Bildspeicher vorselektiert. Wenn jetzt ein Zugriff auf diesen Bereich erfolgt, wird durch Q4 des Dekoders D12 und D7.3 das Anmeldeflipflop D10.1 vorbereitet. Gleichzeitig erzeugt das mit MREQ-getorte Gatter D14.1 ein chip-select-Signal für die Datenbustreiber D24AD27. Nach Beendigung des Speicherlese- oder Speicherschreibzyklus kann die Karte durch das Einschreiben eines anderen Datenworts in den Verwaltungsport wieder aus dem Prozessoradreßbereich ausgeblendet werden.

Zur Sicherung eindeutiger Verhältnisse muß die jeweils aktive RAM-Konfiguration in einer Hilfszelle des Systems abgelegt sein. Dies ist vorallem dann nötig, falls Interruptroutinen auf spezielle RAM-Segmente wie z.B. den Bildpunktspeicher zugreifen. Diese Routinen müssen zum Abschluß stets die vorgefundene RAM-Konfiguration regenerieren.

Software-Lösung

Zur Benutzung des Vollgraphik-Bildschirmes wird ein Mindestmaß an programmtechnischer Unterstützung benötigt. Zunächst soll die Ausgabe alphanummerischer Symbole betrachtet werden. Hierbei ist es notwendig, eine Aufteilung der verfügbaren Punktmatrix von 512x256 Bildpunkten auf Zeichenplätze bestimmter Größe vorzunehmen. Dabei muß ein Kompromiß zwischen der Anzahl der Zeichenplätze und der Übersichtlichkeit der Darstellung sowohl eines einzelnen Zeichens als auch des Gesamtbildes geschlossen werden. Dazu einige Beispiele:
Legt man für die alphanummerischen Symbole ein Rastere von 5x7 Bildpunkten fest, so kann ein Bild bei einer Zeichenplatzgröße von 6x10 Punkten 25 Zeilen zu je 85 Zeichen, also insgesamt 2125 Zeichen, enthalten. Unter Berücksichtigung der Unterzüge von Kleinbuchstaben sind die einzelnen Symbole nur durch jeweils eine Punktzeile bzw. -spalte voneinander getrennt. Ein solches Bild wirkt unübersichtlich.
Vergrößert man die einzelnen Zeichenplätze auf 7x12 Punkte unter Beibehaltung der 5x7-Matrix, so erhält man ein Bild mit 21 Zeilen zu 73 Zeichen, also 1533 Zeichenplätze. Hierbei sind die einzelnen Symbole durch drei Punktzeilen und zwei Punktspalten getrennt, wodurch ein Gewinn an Übersichtlichkeit bezüglich des Gesamtbildes erreicht wird.
Verwendet man dagegen eine Zeichenplatzgröße von 8x16 Bildpunkten, so beträgt die Bildaufteilung 16 Zeilen zu 64 Zeichen, also insgesamt 1024 Zeichen. Dabei ist die Darstellung alphanummerischer Symbole in Form einer 7x9-Matrix möglich (7x12 bei Unterzügen).
Diese zuletzt genannte Möglichkeit soll im folgenden beschrieben werden, da sie auch wegen der byteweisen Bildpunktspeicherorganisation der beschriebenen Hardware einen vergleichsweise geringeren Software-Aufwand erfordert.

Praktische Realisierung

Mit der Festlegung einer Zeichenplatzgröße von 8x16 Bildpunkten und der Bildaufteilung in 16x64 solcher Zeichen sind die Randbedingungen für das Software-Modul klar abgesteckt. Zunächst soll anhand von Programmbeispielen die Organisation der Darstellung alphanummerischer Symbole beschrieben werden, daran anschließend folgt die Unterstützung graphischer Darstellungen.
Als Besonderheit sei erwähnt, daß der mögliche Schreibbereich für alpanummerische Zeichen auf dem Bild durch Festlegung eines Fensters (window) eingeschränkt werden kann. Alle bildbeeinflussenden Funktionen, z.B. das Unterprogramm "Bild löschen", beziehen sich dann nur auf diesen Schreibbereich.
Weiterhin erweist es sich als günstig, im Prozessoradreßraum einen zusätzlichen ASCII-Bildspeicher einzurichten. Dieser Speicher hat bei der hier gewählten Bildorganisation eine Größe von 1024 Byte. In diesem Speicher legt der Prozessor den Bildinhalt der ASCII-Ebene des Bildes ab, wodurch die Möglichkeit eines späteren Auslesens der Zeichen besteht (ein Auslesen aus dem Bildpunktspeicher liefe dagegen auf eine Bildanalyse hinaus). Neben diesem ASCII-Zeichenspeicher werden noch einige Hilfszellen benötigt, z.B. für die Cursor-Position, die Fenster-Koordinaten usw. So wird z.B. in der Cursor-Positionszelle die aktuelle Zeichenplatznummer innerhalb der 1024 Plätze abgelegt, d.h. der Inhalt dieser Zelle liegt zwischen 0 ... 3FFH. Die Cursor-Freigabezelle legt mit ihrem Inhalt fest, ob auf der Cursorposition ein blinkender Cursor erscheint oder nicht. Diese Zelle wird von einer hier nicht näher zu betrachtenden Interrupt-Routine etwa aller 600ms kontrolliert, wobei bei Cursor-Freigabe im gleichen Rythmus abwechselnd das Cursorbild oder das Zeichenbild des auf der Cursorposition befindlichen Zeichens auf dem Bildschirm erscheint.
Innerhalb der Modus-Zelle wird einerseits die Art der Bilddarstellung (schwarz auf weiß oder umgekehrt) abgelegt und andererseits die Art der Zeichendarstellung (mit oder ohne Unterstreichung).
Die RAM-Segmentmerkzelle wird für die Speicherverwaltung des segmentierten RAM benötigt und kann entfallen, falls der 16 kByte-Bildpunktspeicher direkter Bestandteil des 64 kByte-Adreßraumes ist.
Weiter benötigt der Prozessor für den alphanummerischen Bildaufbau einen Zeichengenerator, d.h. einen Speicher, in dem die Gestalt der darzustellenden Zeichen z.B. Zeile für Zeile abgelegt ist. Eine zeilenweise Organisation des Zeichengenerators bietet sich hier wegen der Zeichenplatzbreite von 8 Bit an. Da die Zeichen in einer 7x12-Matrix angelegt werden, benötigt man pro Zeichen 12 Byte im Zeichengenerator. Im Bild 6 ist die Einordnung der 12 Zeichenbytes innerhalb des Zeichenplatzes sowie die Lage der eventuellen Unterstreichung dargestellt.

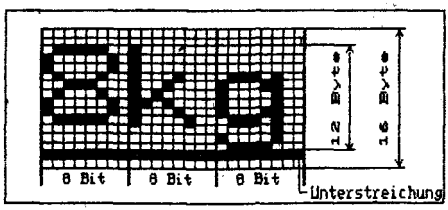


Bild 6: Lage der 12 Byte des Zeichengenerators innerhalb der 16 Byte des Zeichenplatzes

Tafel 1 zeigt den verwendeten Zeichensatz, der abweichend vom Standard-ASCII-Zeichensatz computerintern verwendet wird.
Da nur 16 Steuerzeichen 00H ... 0FH verwendet werden, konnten die deutschen Umlaute und einige besonders häufig benötigte Sonderzeichen mit berücksichtigt werden. Insgesamt umfaßt der Zeichensatz 112 Zeichen, wodurch der Zeichengenerator eine Länge von 1344 Byte belegt. 5 Zeichen sind noch unbelegt und daher nachträglich definierbar. Pseudographische Zeichen sind nicht Bestandteil des Zeichengenerators, da sie durch die Vollgraphik erzeugbar sind. Das Software-Modul zur Unterstützung der alphanummerischen Bildschirmarbeit besteht aus einer Vielzahl von Unterprogrammen mit definierten Schnittstellen. Dabei werden konsequent alle innerhalb der Unterprogramme verwendeten Register gerettet, wodurch eine einfache Hauptprogrammerstellung gewährleistet wird. Die Unterprogramme lassen sich sowohl direkt aufrufen, als auch mit Hilfe von Steuercodes über das Programm "Zeichen schreiben" oder über das Programm "Text aus Tabelle schreiben".

Tafel 3 (am Ende des Beitrages) zeigt die wichtigsten Teile des Software-Moduls, wobei deren Wirkung anhand der Kommentare erschlossen werden kann. Der Abdruck des vollständigen Programmes würde den hier gegebenen Rahmen sprengen. Um dennoch einen möglichst breiten Einblick in die Arbeitsweise des Software-Moduls geben zu können, werden die Programmlisten zweispaltig wiedergegeben.
Als Schnittstellen zu den Unterprogrammen werden einheitlich die Register A, DE, HL verwendet:

- A enthält den Zeichencode (zu schreibende oder gelesene Information)
- DE enthält die Bildposition, wobei D die Zeilennummer (0 ... 0FH) und E die Spaltennummer (0 ... 3FH) enthält
- HL dient als allgemeiner Zeiger.

Die Wirkung aller das Bild verändernden Programme, z.B. "Zeichen schreiben", "Byte schreiben", "Zeile löschen" und alle den Cursor steuernden Programme, beschränkt sich auf das Innere des aktuellen Bildschirmfensters. Die einzige Außnahme bildet das Programm "Cursor setzen", mit dem das momentane Bildschirmfenster verlassen werden kann.
Mit Hilfe dieses Programmes und unter Zuhilfenahme des Programmes "Fenster setzen" kann ein neues Bildschirmfenster eingestellt werden.
Zur Unterstützung graphischer Anwendungen sind Unterprogramme nötig, die es erlauben, einen beliebigen Bildpunkt innerhalb der 256x512-Matrix zu setzen bzw. rückzusetzen. Diese Forderung ist gleichbedeutend mit dem Setzen eines

Tafel 1: Zeichensatz im Mustergerät

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	frei	frei	frei	frei	frei	π		Σ	Ω	°	Ä	ö	Ü	ä	ö	ü
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	ß	]	^	_
60	§	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	µ	}	~	▒

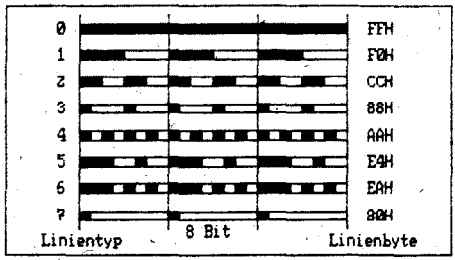


Bild 7: Übersicht zu den realisierten Linientypen

Punktes entsprechend dem momentanen Bildschirm-Modus (schwarzer Punkt entsteht bei Modus "schwarz auf weiß" und umgekehrt). Tafel 4 zeigt das entsprechende Programm. Das Zielkoordinatenpaar (x,y) wird als positives Wertepaar in den Registern HL und A übergeben, d.h. der Bildschirm bildet den ersten Quadranten eines Koordinatensystems und der Punkt (0,0) ist der linke untere Eckpunkt. Innerhalb des Programmes erfolgt für die x-Koordinate eine Begrenzung auf den Bildbereich. Außerdem wird das übergebene Koordinatenpaar als zuletzt bearbeitete Bildposition (Pixel-Cursor) in Merkzellen abgelegt.

Ein weiteres grundlegendes Unterprogramm realisiert das Zeichnen einer Geraden zwischen zwei Koordinatenpaaren.

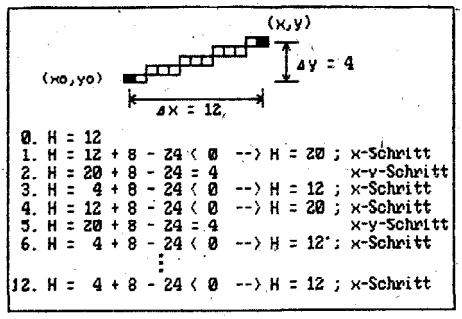
Den Startpunkt für das Zeichnen der Geraden bildet der zuletzt bearbeitete Bildpunkt (Pixel-Cursor) und das Zielkoordinatenpaar wird in den Registern HL und A übergeben. Im Sinne einer hohen Übersichtlichkeit der graphischen Darstellung sind für das Geradenzeichnen die in Bild 7 dargestellten 8 verschiedene Linientypen vorgesehen. Diese werden mit Hilfe eines Initialisierungsprogrammes angewählt, wobei der Linientyp als Nummer in A übergeben werden muß.

Ein einmal gewählter Linientyp bleibt solange aktiv, bis ein anderer gewählt wurde. Als Merkzelle für das Zeichenprogramm dient eine Rotationszelle in der das dem gewählten Linientyp entsprechende Bitmuster während des Zeichenvorgangs rotiert.

Das Programm zum Zeichnen einer Geraden

Tafel 2: Probesubtraktionsalgorithmus

dx := |x-xo|, dy := |y-yo|, sx := sign(x-xo), sy := sign(y-yo)
dx ≥ dy ?
ja: dX := dx, dY := dy, F := 0 / nein: dX := dy, dY := dx, F := 1
H := dX, Z := dX
H := H + 2 · dY − 2 · dX
H ≥ 0 ?
ja: xo := xo + sx; yo := yo + sy / nein: H := H + 2 · dX
F = 0 ?
ja: xo := xo + sx / nein: yo := yo + sy
Zeichne den Punkt (xo,yo)
Z := Z − 1
Wiederhole, solange Z > 0 ist
Return




```
0. H = 12
1. H = 12 + 8 - 24 < 0  --> H = 20 ; x-Schritt
2. H = 20 + 8 - 24 = 4             x-y-Schritt
3. H =  4 + 8 - 24 < 0  --> H = 12 ; x-Schritt
4. H = 12 + 8 - 24 < 0  --> H = 20 ; x-Schritt
5. H = 20 + 8 - 24 = 4             x-y-Schritt
6. H =  4 + 8 - 24 < 0  --> H = 12 ; x-Schritt
                :
12. H =  4 + 8 - 24 < 0  --> H = 12 ; x-Schritt
```

Bild 8: Beispiel zum Algorithmus nach Tafel 1

berechnet nicht direkt die Geradengleichung (wegen der dann notwendigen Division und Multiplikation), sondern benutzt einen Probesubtraktions-Algorithmus, dessen Ablauf in Tafel 2 veranschaulicht ist. Weiterhin ist zu beachten, daß stets die Koordinate mit der größten Änderung als unabhängige Koordinate für die Berechnung benutzt wird. Der Algorithmus nach Tafel 2 wird für die Beträge der Koordinatenänderungen unter den eben genannten Gesichtspunkten abgearbeitet und das Vorzeichen der Änderung, d.h. die Richtung der Geraden, wird für beide Koordinaten nach dem Durchlaufen des Algorithmus berücksichtigt. In Bild 8 ist die Wirkungsweise dieses Algorithmus anhand eines Beispiels verdeutlicht. Das entsprechende Programm ist ebenfalls in Tafel 4 enthalten.

Tafel 3: ALPHANUMMERISCHE GRUNDPROGRAMME

```
;----VEREINBARUNGEN / SYSTEMZELLEN----------------------


SBS: EXT 01800H       ;BILDSPEICHER (16x64 ASCII-ZEICHEN)
MOD: EXT 01C08H       ;MERKZELLE BILDSCHIRM-MODE
SEG: EXT 01C09H       ;RAM-SEGMENT-MERKZELLE
BLS: EXT 01C0AH       ;BLINKFREIGABE SCHIRM
SCP: EXT 01C0BH       ;CURSORPOSITION BILDSCHIRM
SFA: EXT 01C0CH       ;SCHIRM-FENSTER ANFANG
SFE: EXT 01C0EH       ;SCHIRM-FENSTER ENDE

RAM: EXT 08000H       ;ANFANGSADRESSE BILDPUNKTSPEICHER
                      ;UND SEGMENTIERTER RAM

ZEI: EXT 01000H       ;ANFANGSADRESSE ZEICHENGENERATOR
                      ;112x12 BYTE



;----UP SCHIRM-FENSTER SETZEN---------------------------

;    STELLT DAS SCHIRMFENSTER ENTSPRECHEND DER CURSOR-
;    POSITION UND DER ZEILEN- BZW. SPALTENANZAHL IN DE.
;    DIE CURSORPOSITION ENTSPRICHT DER LINKEN OBEREN ECKE
;    DES FENSTERS !

;    IN  : DE=FENSTERGROESSE, D=ZEILENZAHL E=SPALTENZAHL
;    OUT : WIE IN

SFS: PUSH HL
     PUSH AF
     EX DE,HL         ;HL=FENSTERGROESSE
     CALL SCL         ;SCHIRMCURSOR LESEN
     EX DE,HL
     LD (SFA),HL      ;ALS FENSTER-ANFANG ABLEGEN
     LD A,L           ;END-SPALTE BERECHNEN
     DEC A            ;OFFSET -1
     ADD E
     CMP 40H          ;AUF MAXIMALSPALTE BEGRENZEN
     JRC S00
     LD A,3FH
S00: LD L,A
     LD A,H           ;END-ZEILE BERECHNEN
     DEC A            ;OFFSET -1
     ADD D
     CMP 10H          ;AUF MAXIMALZEILE BEGRENZEN
     JRC S01
     LD A,0FH
S01: LD H,A
     LD (SFE),HL      ;FENSTER-ENDE ABLEGEN
     POP AF
     POP HL
     RET

;----UP SCHIRM-CURSORPOSITION LESEN----------------------

;    OUT : DE=POSITION, D=ZEILE UND E=SPALTE

SCL: PUSH HL
     LD HL,(SCP)      ;RELATIVADRESSE LESEN
     ADD HL,HL        ;UMRECHNEN
     ADD HL,HL
     SRL L
     SRL L
     EX DE,HL
     POP HL
     RET


;----UP SCHIRM-CURSORPOSITION STELLEN--------------------

;    IN  : DE=POSITION, D=ZEILE UND E=SPALTE
;    OUT : DE=POSITION

SCS: PUSH HL
     PUSH DE
```

```
    PUSH AF
    LD A,E
    AND 3FH          ;SPALTE BEGRENZEN
    LD E,A
    LD A,D
    AND 0FH          ;ZEILE BEGRENZEN
    LD D,A           ;DE=POSITION
    JR SC2


;----UP SCHIRM-CURSOR AN BILDANFANG STELLEN-------------

;   SETZT DEN CURSOR IN DIE LINKE OBERE ECKE DES
;   AKTUELLEN FENSTERS

SCH: PUSH HL
     PUSH DE
     PUSH AF
     LD HL,(SFA)     ;SCHIRM-FENSTER-ANFANG
     EX DE,HL        ;ALS NEUE CURSORPOSITION
     JR SC2


;----UP SCHIRM-CURSOR AN ZEILENANFANG-------------------

;   SETZT DEN CURSOR INNERHALB DES AKTUELLEN FENSTERS AN
;   DEN ANFANG DER ZEILE IN DER DER CURSOR STEHT

SC2: PUSH HL
     PUSH DE
     PUSH AF
     CALL SCL        ;CURSORPOSITION LESEN
     JR SC0


;----UP SCHIRM-CURSOR NACH RECHTS-----------------------

;   RUECKT DEN CURSOR UM EINE POSITION NACH LINKS. FALLS
;   DABEI DIE LETZTE SPALTE DES FENSTERS UEBERSCHRITTEN
;   WIRD, GELANGT ER AN DEN ANFANG DER NAECHSTEN ZEILE.
;   WIRD DABEI DIE LETZTE ZEILE DES FENSTERS UEBER-
;   SCHRITTEN, SO WIRD DER FENSTERINHALT NACH OBEN
;   GEROLLT UND DIE LETZTE ZEILE GELOESCHT (BEI EINZEI-
;   LIGEM FENSTER WIRD NICHT GEROLLT)

SC>: PUSH HL
     PUSH DE
     PUSH AF
     CALL SCL        ;SCHIRMCURSOR LESEN
     LD HL,SFE       ;SCHIRMFENSTER-ENDE-ZELLE
     LD A,E          ;CURSORSPALTE
     INC E           ;ERHOEHEN
     CMP (HL)        ;MIT ENDSPALTE VERGLEICHEN
     JRNZ SC2        ;SPRUNG,FALLS NICHT ENDSPALTE
     LD A,D          ;ZEILENNUMMER
     INC HL          ;ZEIGER AUF ENDZEILE
     CMP (HL)        ;AUCH ENDZEILE ?
     CAZ BND         ;JA: BILD ROLLEN
     JRZ SC0         ;JA: CURSOR BLEIBT AUF ENDPOSITION
     INC D           ;SONST: EINE ZEILE TIEFER
SC0: LD HL,SFA       ;FENSTER-ANFANG-ZELLE
     LD E,(HL)       ;FENSTER-ZEILEN-ANFANG
     JR SC2


;----UP SCHIRM-CURSOR NACH LINKS------------------------

;   RUECKT DEN CURSOR UM EINE POSITION NACH LINKS. FALLS
;   DABEI DIE ERSTE SPALTE DES FENSTERS UEBERSCHRITTEN
;   WIRD, GELANGT ER AN DAS ENDE DER VORIGEN ZEILE.
;   WIRD DABEI DIE ERSTE ZEILE DES FENSTERS UEBER-
;   SCHRITTEN, SO WIRD DER FENSTERINHALT NACH UNTEN
;   GEROLLT UND DIE ERSTE ZEILE GELOESCHT (BEI EINZEI-
;   LIGEM FENSTER WIRD NICHT GEROLLT)

SC<: PUSH HL
     PUSH DE
     PUSH AF
     CALL SCL        ;CURSORPOSITION LESEN
     LD HL,SFA       ;SCHIRM-FENSTER-ANFANG-ZELLE
     LD A,E          ;CURSORSPALTE
     DEC E           ;VERMINDERN
     CMP (HL)        ;MIT ANFANGSSPALTE VERGLEICHEN
     JRNZ SC2        ;SPRUNG,FALLS NICHT ANFANGSSPALTE
     LD A,D          ;ZEILE
     INC HL          ;ZEIGER AUF ANFANGSZEILE
     CMP (HL)        ;AUCH ANFANGSZEILE ?
     CAZ BNU         ;JA: BILD ROLLEN
     JRZ SC1         ;JA: CURSOR BLEIBT
     DEC D           ;SONST: EINE ZEILE HOEHER
SC1: LD HL,SFE       ;FENSTER-ENDE-ZELLE
     LD E,(HL)       ;ENDSPALTE
     JR SC2


;----UP SCHIRM-CURSOR NACH OBEN-------------------------

;   RUECKT DEN CURSOR UM EINE ZEILE NACH OBEN.
;   WIRD DABEI DIE ERSTE ZEILE DES FENSTERS UEBER-
;   SCHRITTEN, SO WIRD DER FENSTERINHALT NACH UNTEN
;   GEROLLT UND DIE ERSTE ZEILE GELOESCHT (BEI EINZEI-
;   LIGEM FENSTER WIRD NICHT GEROLLT)

SC^: PUSH HL
     PUSH DE
     PUSH AF
     CALL SCL        ;CURSOR LESEN
     LD A,(SFA+1)    ;FENSTERANF.ZEILE HOLEN
     CMP D           ;STEHT DORT CURSOR ?
     CAZ BNU         ;JA: BILD ROLLEN
     JRZ SC2         ;JA: SO BLEIBT ER
     DEC D           ;NEIN: ZEILE NACH OBEN
     JR SC2


;----UP SCHIRM-CURSOR NACH UNTEN------------------------

;   RUECKT DEN CURSOR UM EINE ZEILE NACH UNTEN.
;   WIRD DABEI DIE LETZTE ZEILE DES FENSTERS UEBER-
;   SCHRITTEN, SO WIRD DER FENSTERINHALT NACH OBEN
;   GEROLLT UND DIE LETZTE ZEILE GELOESCHT (BEI EINZEI-
;   LIGEM FENSTER WIRD NICHT GEROLLT)

SC\: PUSH HL
     PUSH DE
     PUSH AF
     CALL SCL        ;CURSOR LESEN
     LD A,(SFE+1)    ;SCHIRMFENSTER-ENDZEILE
```

```
    CMP D           ;STEHT DORT CURSOR ?
    CAZ BNO         ;JA: BILD ROLLEN
    JRZ SC2         ;JA: SO BLEIBT ER
    INC D           ;NEIN: ZEILE TIEFER
SC2: LD H,D         ;IN HL RELATIVADRESSE ERRECHNEN
    XOR A
    SRL H
    RRA
    SRL H
    RRA
    OR E
    LD L,A
    LD A,(BLS)      ;BLINKFREIGABEZELLE LESEN
    OR A            ;FREIGEGEBEN ?
    CANZ SCA        ;CURSOR ABSCHALTEN, FALLS AN
    LD (SCP),HL     ;ABLEGEN
    CANZ SCB        ;CURSOR ANSCHALTEN,FALLS VORHER AN
    POP AF
    POP DE
    POP HL
    RET

;----UP SCHIRM-CURSOR ABSCHALTEN-------------------------

SCA: PUSH HL
    PUSH DE
    PUSH AF
    LD HL,BLS       ;FREIGABEZELLE
    LD (HL),0       ;RUECKSETZEN
    CALL SZL        ;ZEICHEN LESEN
    JR SC3          ;UND SPRUNG


;----UP SCHIRM-CURSOR ANSCHALTEN-------------------------

SCB: PUSH HL
    PUSH DE
    PUSH AF
    LD HL,BLS       ;FREIGABEZELLE
    LD (HL),0FFH    ;SETZEN
    LD A,5FH        ;CURSORBILD
SC3: CALL SBE       ;BILD ERZEUGEN
    POP AF
    POP DE
    POP HL
    RET


;----UP ZEICHEN AUF SCHIRM-CURSORPOSITION LESEN----------

;   OUT : A=ZEICHENCODE

SZL: PUSH HL
    CALL SAD        ;SCHIRMADRESSE ERMITTELN
    LD A,(HL)       ;ZEICHEN HOLEN
    POP HL
    RET


;----UP ZEICHEN AUF SCHIRM-CURSORPOSITION SCHREIBEN------

;   SCHREIBT DAS ZEICHEN ENTSPRECHEND DES ZEICHENCODES
;   AUF DIE CURSORPOSITON UND RUECKT ANSCHLIESSEND DEN
;   CURSOR UM EINE STELLE NACH RECHTS ( SIEHE AUCH SC> )
;   ES WERDEN AUCH STEUERCODES LAUT TABELLE AUSGEFUEHRT

;   IN  : A=ZEICHENCODE
;   OUT : A=ZEICHENCODE

SZS: PUSH AF        ;FLAGS RETTEN
    CMP 80H         ;ZEICHEN GROESSER 80H
    JRNC SZ0        ;NICHT AUSSCHREIBEN
    CMP 10H         ;STEUERZEICHEN ?
    JRC SZ1         ;SPRUNG,FALLS JA
    CALL SZU        ;SONST:ZEICHEN SCHREIBEN
    CALL SC>        ;CURSOR RUECKEN
SZ0: POP AF
    RET
SZ1: EX (SP),HL     ;STACK UMSORTIEREN, AF ZUOBERST
    PUSH HL
    LD HL,SZT       ;SPRUNGTABELLENADRESSE
    CALL TSU        ;TABELLENSUCHE
    JRC SZ2         ;SPRUNG,FALLS NICHT GEFUNDEN
    POP AF          ;AF ZURUECK
    EX (SP),HL      ;HL ZURUECK, SPRUNGADRESSE ABLEGEN
    RET             ;UND SPRUNG
SZ2: POP AF
    POP HL
    RET


;----SPRUNGVERTEILERTABELLE SCHIRM-ZEICHEN SCHREIBEN-----

SZT: DB 00H         ;UNTERSTREICHUNG EIN
    DW UMO
    DB 01H          ;NORMALMODE
    DW NMO
    DB 02H          ;INVERSMODE
    DW IMO
    DB 04H          ;CLEAR
    DW CLL
    DB 08H          ;CURSOR NACH LINKS
    DW SC<
    DB 09H          ;CURSOR NACH RECHTS
    DW SC>
    DB 0AH          ;CURSOR NACH UNTEN
    DW SC\
    DB 0BH          ;CURSOR NACH OBEN
    DW SC^
    DB 0CH          ;BILD CLEAR
    DW CLS
    DB 0DH          ;CURSOR AN ZEILENANFANG
    DW SCZ
    DB 0EH          ;ZEILE LOESCHEN,CURSOR AN Z.-ANFANG
    DW CLZ
    DB 0FH          ;UNTERSTREICHUNG AUS
    DW UMO
    DB 0FFH         ;END


;----UP ZEICHEN AUF SCHIRM-CURSORPOSITION EINTRAGEN------

;   IN  : A=ZEICHEN (ACHTUNG: KEIN TEST AUF GUELTIGE
;         ZEICHENCODES)
;   OUT : A=ZEICHEN
```

```
SZU: PUSH HL
     CALL SAD          ;SCHIRMADRESSE ERMITTELN
     LD (HL),A         ;ZEICHEN ABLEGEN
     CALL SBE          ;BILD ERZEUGEN
     POP HL
     RET

;----UP TABELLENSUCHE-----------------------------------

;    DAS PROGRAMM ERMITTELT ZU EINEM EINGANGSCODE EINE
;    ENTSPRECHENDE SPRUNGZIELADRESSE AUS EINER TABELLE
;    DER GESTALT: DB SUCHBYTE--DW ZIELADRESSE--DB...--
;    DW...--...--DB OFFH=END

;    IN  : HL=TABELLENADRESSE, A=SUCHCODE
;    OUT : HL=SPRUNGADRESSE, CARRY=0, FALLS GEFUNDEN
;          HL=?,             CARRY=1, FALLS NICHT

TSU: PUSH DE
TS1: LD D,A            ;RETTEN SUCHBYTE
     LD A,0FFH         ;END-ZEICHEN
     CMP (HL)          ;ERREICHT ?
     SCF               ;SET CARRY
     LD A,D            ;A ZURUECK
     JRZ TS2           ;UND SPRUNG,FALLS ENDE ERREICHT
     CMP (HL)          ;SUCHBYTE = TABELLENBYTE ?
     INC HL            ;ZEIGER STELLEN
     LD E,(HL)         ;ADRESSE EINLESEN
     INC HL
     LD D,(HL)
     INC HL
     JRNZ TS1          ;SPRUNG,FALLS NICHT GEFUNDEN
     EX DE,HL          ;SONST: HL=ZIELADRESSE
TS2: POP DE
     RET

;----UP ZEICHEN VOR SCHIRMCURSORPOSITION LOESCHEN (CLEAR)

;    RUECKT DEN CURSOR UM EINE POSITION NACH LINKS UND
;    LOESCHT DAS AUF DIESER POSITION BEFINDLICHE ZEICHEN
;    VGL. AUCH SCC !

CLL: CALL SCC          ;CURSOR AUF ZU LOESCHENDES ZEICHEN
     PUSH AF
     LD A,20H          ;LEERZEICHEN
     CALL SZU          ;EINTRAGEN,CURSOR BLEIBT
     POP AF
     RET

;----UP SCHIRMADRESSE ERMITTELN-------------------------

;    ERMITTELT DIE ADRESSE INNERHALB DES ASCII-SPEICHERS
;    ENTSPRECHEND DER CURSOR-POSITION

;    OUT : HL=SCHIRMADRESSE ENTSPRECHEND CURSORPOSITION

SAD: PUSH DE
     PUSH AF           ;FLAGS RETTEN
     LD HL,(SCP)       ;RELATIVE ADRESSE HOLEN
     LD DE,SBS         ;ABSOLUTADRESSE BERECHNEN
     ADD HL,DE
     POP AF
     POP DE
     RET

;----UP TEXT AB CURSORPOSITION AUF SCHIRM SCHREIBEN-----

;    SCHREIBT EINEN BELIEBIGEN TEXT VOM SPEICHER AUF DEN
;    BILDSCHIRM AB CURSOR-POSITION UND RUECKT DEN CURSOR
;    ENTSPRECHEND, SIEHE AUCH SZS !

;    IN  : HL=TEXTZEIGER, TEXTENDE=0FFH
;    OUT : HL=ZEIGT AUF ERSTES ZEICHEN NACH TEXTENDE

TXS: PUSH AF
TX2: LD A,(HL)
     INC HL
     CMP 0FFH
     PUSH AF
     CANZ SZS
     POP AF
     JRNZ TX2
     POP AF
     RET


;----UP SCHIRM LOESCHEN---------------------------------

CLS: CALL SCH          ;CURSOR AN FENSTERANFANG


;----UP SCHIRM AB CURSORZEILE LOESCHEN------------------

CLR: PUSH HL
     PUSH DE
     PUSH BC
     PUSH AF
     CALL SCL          ;CURSOR LESEN
     LD A,(SFE+01H)    ;SCHIRM-FENSTER-ENDZEILE
     SUB D             ;ZEILENANZAHL ERMITTELN
     INC A
     JR C00            ;UND SPRUNG


;----UP SCHIRM-ZEILE LOESCHEN---------------------------

CL2: PUSH HL
     PUSH DE
     PUSH BC
     PUSH AF
     LD A,1            ;KENNZEICHEN EINE ZEILE
C00: CALL SCZ          ;CURSOR AN ZEILENANFANG
     CALL SCL          ;CURSOR LESEN
     PUSH AF           ;ZEILENZAHL MERKEN
     LD C,A            ;AUCH IN C
     LD A,(SFE)        ;FENSTER-END-SPALTE
     SUB E             ;ZEILENLAENGE BESTIMMEN
     INC A
     CALL SAD          ;SCHIRMADRESSE ERMITTELN
C05: LD B,A            ;ZAEHLER LADEN
C01: LD (HL),20H       ;LEERZEICHEN IN ASCII-SPEICHER
     INC HL            ;EINTRAGEN
     DJNZ C01
     DEC C             ;ZEILENZAEHLER ERNIEDRIGEN
     JRZ C07           ;SPRUNG,FALLS FERTIG
     LD DE,03FH        ;OFFSET
     ADD HL,DE         ;ZEIGER STELLEN
     LD B,A            ;ZAEHLER LADEN
```

```
C06: LD (HL),20H      ;NAECHSTE LEERZEILE EINTRAGEN
     DEC HL
     DJNZ C06
     DEC C            ;ZEILENZAEHLER VERMINDERN
     JRZ C07          ;SPRUNG,FALLS FERTIG
     LD DE,41H        ;OFFSET
     ADD HL,DE        ;ZEIGER STELLEN
     JR C05           ;UND WEITER
C07: LD B,A           ;ZAEHLER LADEN
     POP AF           ;ZEILENANZAHL ZURUECK
     RLCA             ;IN PUNKTDOPPELZEILEN UMRECHNEN
     RLCA
     RLCA
     LD C,A           ;C IST ZAEHLER
     CALL RAD         ;BILD-RAM-ADRESSE ERMITTELN
     LD E,0           ;ZUNAECHST HINTERGRUND DUNKEL
     LD A,(MOD)       ;MODUSZELLE
     BIT 0,A          ;HINTERGRUND DUNKEL ?
     JRZ C02          ;JA
     DEC E            ;NEIN: DANN HELL
C02: LD A,(SEG)       ;ALTES RAM-SEGMENT
     LD D,A           ;MERKEN
     LD A,3           ;BILD-RAM
     LD (SEG),A       ;ABLEGEN
     OUT 90H          ;EINSCHALTEN
C03: PUSH BC          ;ZAEHLER MERKEN
     LD A,B           ;AUCH IN A
C04: LD (HL),E        ;BILD-BYTE ABLEGEN
     INC HL
     DJNZ C04         ;EINE PUNKTZEILE LANG
     LD C,03FH        ;OFFSET
     ADD HL,BC        ;ZEIGER STELLEN
     LD B,A
C08: LD (HL),E        ;BILDBYTE ABLEGEN
     DEC HL
     DJNZ C08         ;ZWEITE PUNKTZEILE LANG
     LD C,041H        ;OFFSET
     ADD HL,BC        ;ZEIGER STELLEN
     POP BC           ;SCHLEIFENZAEHLER ZURUECK
     DEC C            ;ZEILENZAEHLER VERMINDERN
     JRNZ C03         ;WEITER,BIS ALLE ZEILEN GELOESCHT
     LD A,D           ;ALTES RAM-SEGMENT
     LD (SEG),A       ;ABLEGEN
     OUT 90H          ;UND EINSCHALTEN
     POP AF
     POP BC
     POP DE
     POP HL
     RET

;----UP SCHIRMBILDERZEUGUNG-------------------------------

;    TRANSPORTIERT DAS BILD DES UEBERGEBENEN ZEICHENS AUS
;    DEM ZEICHENGENERATOR IN DEN BILDPUNKTSPEICHER ENT-
;    SPRECHEND DER CURSOR-POSITION

;    IN  : A=ZEICHENCODE
;    OUT : A=ZEICHENCODE

SBE: PUSH HL
     PUSH DE
     PUSH BC
     PUSH AF
     SUB 10H          ;KORREKTUR ASCII (10H...7FH)
     LD L,A           ;ZEIGER IM ZEICHENGENERATOR STELLEN
     LD H,0
     ADD HL,HL
     ADD HL,HL
     LD D,H
     LD E,L
     ADD HL,HL
     ADD HL,DE
     LD DE,ZEI+08H
     ADD HL,DE
     LD A,(MOD)       ;MODUSZELLE AUSLESEN
     BIT 0,A          ;TEST AUF INVERSMODE
     SCF              ;SET CARRY FUER NORMALMODE
     LD C,0FFH        ;WEISSHINTERGRUND FUER NORMALMODE
     JRNZ SB0         ;SPRUNG,FALLS NORMALMODE
     CCF              ;RESET CARRY FUER INVERSMODE
     INC C            ;DUNKLER HINTERGRUND
SB0: BIT 1,A          ;UNDERLINE-MODE ?
     LD A,C           ;HINTERGRUND NACH A
     PUSH AF          ;UNTERSTE PUNKTZEILE EINKELLERN
     JRNZ SB1         ;SPRUNG,FALLS NICHT UNDERLINE-MODE
     CPL              ;SONST: UNTERSTREICHUNG ERZEUGEN
SB1: PUSH AF          ;ZWEITE ZEILE EINKELLERN
     LD B,0CH         ;SCHLEIFENZAEHLER AUF 12
SB2: LD A,(HL)        ;ZEILE AUS ZEICHENGENERATOR HOLEN
     JRC SB3          ;UND SPRUNG,FALLS NORMALMODE
     CPL              ;SONST: ZEICHENBILD INVERTIEREN
SB3: PUSH AF          ;UND EINKELLERN
     DEC HL           ;ZEIGER IM ZEICHENGENERATOR RUECKEN
     DJNZ SB2         ;UND WEITER
     LD A,C           ;HINTERGRUND ERNEUT HOLEN
     PUSH AF          ;UND ZWEI ZEILEN EINKELLERN
     PUSH AF
     CALL RAD         ;BILD-RAM-ADRESSE ERMITTELN
     LD A,(SEG)       ;ALTE SEGMENT-NUMMER
     LD C,A           ;IN C MERKEN
     LD A,3           ;BILD-RAM-SEGMENT
     LD (SEG),A       ;ABLEGEN
     OUT 90H          ;UND EINSCHALTEN
     LD DE,40H        ;OFFSET ZWISCHEN ZEICHENZEILEN
     LD B,010H        ;SCHLEIFENZAEHLER AUF 16 ZEILEN
SB4: POP AF           ;ZEILE AUSKELLERN
     LD (HL),A        ;UND IN'S BILD EINTRAGEN
     ADD HL,DE        ;RAM-ZEIGER STELLEN
     DJNZ SB4         ;UND WEITER
     LD A,C           ;URSPRUENGLICHES RAM-SEGMENT
     LD (SEG),A       ;ABLEGEN
     OUT 90H          ;UND EINSCHALTEN
     POP AF
     POP BC
     POP DE
     POP HL
     RET

;----UP SCHIRM-BILD NACH OBEN ROLLEN----------------------

;    ROLLT DEN INHALT DES BILDSCHIRMFENSTERS UM EINE
;    ZEILE NACH OBEN (NICHT BEI EINZEILIGEM FENSTER)

BNO: PUSH HL
     PUSH DE
```

```
        PUSH BC
        PUSH AF
        LD HL,(SFE)     ;SCHIRMFENSTER-ENDE
        EX DE,HL
        LD HL,(SFA)     ;SCHIRMFENSTER-ANFANG
        LD A,E          ;SPALTENANZAHL BESTIMMEN
        SUB L
        INC A
        LD B,0
        LD C,A          ;BC ALS SPALTENZAEHLER
        LD A,D          ;ZEILENANZAHL BESTIMMEN
        SUB H
        JRZ BN0         ;SPRUNG,FALLS NUR EINE ZEILE
        CALL SCH        ;CURSOR AN FENSTERANFANG
        CALL SAD        ;SCHIRMADRESSE BESTIMMEN
        EX DE,HL        ;ALS ZIELADRESSE
        LD HL,0040H     ;OFFSET
        ADD HL,DE       ;QUELLADRESSE
        CALL BM0        ;ASCII-SPEICHER UMLAGERN
        CALL RAD        ;BILD-RAM-ADRESSE ERMITTELN
        EX DE,HL        ;ZIELADRESSE
        LD HL,0400H     ;OFFSET 16 PUNKTZEILEN
        ADD HL,DE       ;QUELLADRESSE
        RLCA            ;ZEILENZAHL IN PUNKTZEILENZAHL
        RLCA            ;UMRECHNEN
        RLCA
        RLCA
        LD B,A          ;ZUNAECHST RETTEN
        LD A,(SEG)      ;AKTUELLES RAM-SEGMENT MERKEN
        PUSH AF
        LD A,3          ;BILD-RAM
        LD (SEG),A      ;ABLEGEN
        OUT 90H         ;UND EINSCHALTEN
        LD A,B          ;ZEILENZAEHLER
        LD B,0
        CALL BM0        ;BILDPUNKT-MATRIX UMLAGERN
        POP AF          ;ALTES RAM-SEGMENT
        LD (SEG),A      ;ABLEGEN
        OUT 90H         ;UND EINSCHALTEN
        CALL SCU        ;CURSOR AN ANFANG LETZTE ZEILE
        CALL CLZ        ;DIESE LOESCHEN
BN0:    POP AF
        POP BC
        POP DE
        POP HL
        RET

;----HILFSPROGRAMM BILDMATRIX NACH OBEN UMLAGERN---------

BM0:    PUSH AF         ;ZAEHLER BLEIBEN
BM0:    PUSH BC
        LDIR            ;EINE ZEILE UMLAGERN
        LD C,03FH       ;OFFSET
        ADD HL,BC       ;ZEIGER STELLEN
        EX DE,HL
        ADD HL,BC
        EX DE,HL
        POP BC
        DEC A           ;ZEILENZAEHLER VERMINDERN
        JRZ BM1         ;SPRUNG,FALLS FERTIG
        PUSH BC
        LDDR            ;EINE ZEILE UMLAGERN
        LD C,041H       ;OFFSET
        ADD HL,BC       ;ZEIGER STELLEN
        EX DE,HL
        ADD HL,BC
        EX DE,HL
        POP BC
        DEC A           ;ZEILENZAEHLER VERMINDERN
        JRNZ BM0        ;WEITER,FALLS NICHT NULL
BM1:    POP AF
        RET


;----UP SCHIRM-BILD NACH UNTEN ROLLEN--------------------

;       ROLLT DEN INHALT DES BILDSCHIRMFENSTERS UM EINE
;       ZEILE NACH UNTEN (NICHT BEI EINZEILIGEM FENSTER)

BNU:    PUSH HL
        PUSH DE
        PUSH BC
        PUSH AF
        LD HL,(SFE)     ;SCHIRMFENSTER-ENDE
        EX DE,HL
        LD HL,(SFA)     ;SCHIRMFENSTER-ANFANG
        LD A,E          ;SPALTENANZAHL BESTIMMEN
        SUB L
        INC A
        LD B,0
        LD C,A          ;BC ALS SPALTENZAEHLER
        LD A,D          ;ZEILENANZAHL BESTIMMEN
        SUB H
        JRZ BN1         ;SPRUNG,FALLS NUR EINE ZEILE
        CALL SCU        ;CURSOR AN ANF. LETZTE FENSTERZEILE
        CALL SAD        ;SCHIRMADRESSE ERMITTELN
        EX DE,HL        ;ALS ZIELADRESSE
        LD HL,0FFC0H    ;OFFSET
        ADD HL,DE       ;QUELLADRESSE
        CALL BMU        ;ASCII-SPEICHER UMLAGERN
        CALL RAD        ;BILD-RAM-ADRESSE ERMITTELN
        LD DE,03C0H     ;OFFSET FUER 16.PUNKTZEILE
        ADD HL,DE
        EX DE,HL        ;ZIELADRESSE
        LD HL,0FC00H    ;OFFSET -16 PUNKTZEILEN
        ADD HL,DE       ;QUELLADRESSE
        RLCA            ;ZEILENZAHL IN PUNKTZEILENZAHL
        RLCA            ;UMRECHNEN
        RLCA
        RLCA
        LD B,A          ;ZUNAECHST RETTEN
        LD A,(SEG)      ;AKTUELLES RAM-SEGMENT MERKEN
        PUSH AF
        LD A,3          ;BILD-RAM
        LD (SEG),A      ;ABLEGEN
        OUT 90H         ;UND EINSCHALTEN
        LD A,B          ;ZEILENZAEHLER
        LD B,0
        CALL BMU        ;BILDPUNKT-MATRIX UMLAGERN
        POP AF          ;ALTES RAM-SEGMENT
        LD (SEG),A      ;ABLEGEN
        OUT 90H         ;UND EINSCHALTEN
        CALL SCH        ;CURSOR AN BILDANFANG
        CALL CLZ        ;DIESE LOESCHEN
BN1:    POP AF
        POP BC
```

```
    POP DE
    POP HL
    RET


;----HILFSPROGRAMM BILDMATRIX NACH UNTEN UMLAGERN--------

BMU: PUSH AF          ;ZAEHLER BLEIBEN
BM2: PUSH BC
     LDIR             ;EINE ZEILE UMLAGERN
     LD BC,OFFBFH     ;OFFSET
     ADD HL,BC        ;ZEIGER STELLEN
     EX DE,HL
     ADD HL,BC
     EX DE,HL
     POP BC
     DEC A            ;ZEILENZAEHLER VERMINDERN
     JRZ BM3          ;SPRUNG,FALLS FERTIG
     PUSH BC
     LDDR             ;EINE ZEILE UMLAGERN
     LD BC,OFFC1H     ;OFFSET
     ADD HL,BC        ;ZEIGER STELLEN
     EX DE,HL
     ADD HL,BC
     EX DE,HL
     POP BC
     DEC A            ;ZEILENZAEHLER VERMINDERN
     JRNZ BM2         ;WEITER,FALLS NICHT NULL
BM3: POP AF
     RET


;----TEILPROGRAMM BILD-RAM-ADRESSE----------------------

;    ERMITTELT ZUR AKTUELLEN CURSORPOSITION DIE ADRESSE
;    DES ERSTEN ZUGEHOERIGEN BYTES IM BILD-RAM.

;    OUT : HL=ADRESSE

RAD: PUSH DE
     LD HL,(SCP)      ;RELATIVADRESSE ASCII-SPEICHER
     RLC L            ;IN RELATIVADRESSE BILD-RAM
     RL H             ;UMRECHNEN
     RLC L
     RL H
     SLA H
     SLA H
     SRL L
     SRL L
     LD DE,RAM        ;RAM-BEGINN
     ADD HL,DE        ;DAZUADDIEREN
     POP DE
     RET


;----UP UNDERLINE-MODE----------------------------------

UMO: PUSH AF
     LD A,(MOD)       ;KENNBYTE HOLEN
     RES 1,A          ;KENNBIT UNTERSTREICHUNG EIN
     JR MO1


;----UP UNDERLINE-MODE AUS------------------------------

OMO: PUSH AF
     LD A,(MOD)       ;KENNBYTE HOLEN
     SET 1,A          ;KENNBIT UNTERSTREICHUNG AUS
     JR MO1


;----UP NORMALMODE EINSCHALTEN--------------------------

NMO: PUSH AF
     LD A,(MOD)       ;KENNBYTE LADEN
     SET 0,A          ;KENNBIT NORMALMODE
     JR MO1


;----UP INVERSMODE EINSCHALTEN--------------------------

IMO: PUSH AF
     LD A,(MOD)       ;KENNBYTE HOLEN
     RES 0,A          ;KENNBIT INVERSMODE
MO1: LD (MOD),A       ;KENNBYTE EINTRAGEN
     POP AF
     RET


;----UP BYTE AUF SCHIRM-CURSORPOSITION SCHREIBEN--------

;    SCHREIBT DAS UEBERGEBENE BYTE ALS ZWEI ASCII-ZEICHEN
;    AB DER CURSOR-POSITION UND RUECKT DEN CURSOR
;    ENTSPRECHEND, VGL. AUCH SC> !

;    IN  : A=HEXAZAHL (BYTE)
;    OUT : A=HEXAZAHL (BYTE)

SHS: PUSH AF
     PUSH BC
     PUSH HL
     CALL SAD         ;SCHIRMADRESSE ERMITTELN
     CALL BAP         ;BYTE ALS ZWEI ASCII ABLEGEN
     DEC HL           ;ZEIGER AUF ERSTES ZEICHEN
     DEC HL
     LD B,2           ;ZWEI BYTE
SH2: LD A,(HL)        ;ZEICHEN HOLEN
     CALL SBE         ;UND AUSSCHREIBEN
     CALL SC>         ;CURSOR RUECKEN
     INC HL           ;GLEICHES MIT ZWEITEM ZEICHEN
     DJNZ SH2         ;UND WEITER BIS FERTIG
     POP HL
     POP BC
     POP AF
     RET


;----UP KONVERTIERUNG HEXA-HALBBYTE IN ASCII------------

;    IN  : A=HEXAHALBBYTE
;    OUT : A=ASCII-ZEICHEN

HHA: AND 0FH          ;WEGBLENDEN HOEHEREN TEIL
     ADD 90H          ;AUFBEREITUNG PSEUDOTETRADEN
     DAA
     ADC 40H          ;ZEICHEN ERZEUGEN
     DAA
     RET
```

```
;----UP DOPPELBYTE AUF SCHIRM-CURSORPOSITION SCHREIBEN---

;    SCHREIBT DAS UEBERGEBENE DOPPEL-BYTE ALS ZWEI ASCII-
;    ZEICHEN AB DER CURSOR-POSITION UND RUECKT DEN CURSOR
;    ENTSPRECHEND, VGL. AUCH SC> !

;    IN  : HL=DOPPELBYTE
;    OUT : HL=DOPPELBYTE

SDS: PUSH AF
     LD A,L
     PUSH AF
     LD A,H
     CALL SHS          ;ERSTES BYTE AUSSCHREIBEN
     POP AF
     CALL SHS          ;ZWEITES BYTE AUSSCHREIBEN
     POP AF
     RET
```

```
;----UP ABLEGEN EINES BYTES ALS ZWEI ASCII IN PUFFER----

;    IN  : HL=PUFFERADRESSE, A=HEXAZAHL
;    OUT : HL=PUFFERADRESSE + 2, A=HEXAZAHL

BAP: PUSH AF           ;AKKU BLEIBT
     RRCA              ;HALBBYTE ROTIEREN
     RRCA
     RRCA
     RRCA
     CALL HHA          ;HOEHERWERTIGES HALBBYTE
     LD (HL),A         ;ABLEGEN
     INC HL            ;ZEIGER RUECKEN
     POP AF            ;AKKU ZURUECK
     PUSH AF
     CALL HHA          ;NIEDERWERTIGES HALBBYTE
     LD (HL),A         ;ABLEGEN
     INC HL            ;ZEIGER STELLEN
     POP AF
     RET
```

Tafel 4: Graphik-Grundprogramme

```
;----VEREINBARUNGEN SYSTEMZELLEN-------------------------

PIY: EXT 01C00H        ;Y-KOORDINATE LETZTE PIXELPOSITION
PIX: EXT 01C01H        ;X-KOORDINATE LETZTE PIXELPOSITION
DIY: EXT 01C03H        ;HILFSZELLE ABHAENGIGE KOORDINATE
DIX: EXT 01C05H        ;HILFSZELLE UNABHAENGIGE KOORDINATE
LIN: EXT 01C07H        ;ROTATIONSZELLE LINIENBYTE
MOD: EXT 01C08H        ;MERKZELLE BILDSCHIRM-MODE
SEG: EXT 01C09H        ;RAM-SEGMENT-MERKZELLE

RAM: EXT 08000H        ;ANFANGSADRESSE BILDPUNKTSPEICHER
                       ;UND SEGMENTIERTER RAM


;----UP PIXEL ERZEUGEN-----------------------------------

;    DAS PROGRAMM DIENT ZUM DAZUMISCHEN EINES PUNKTES ZUM
;    BILDSCHIRMINHALT, ABHAENGIG VON DER GERADE HERSCHEN-
;    DEN VIDEO-MODE. UEBERGEBENE KOORDINATEN X (0...511)
;    UND Y (0...255) WERDEN IN MERKZELLEN ABGELEGT.
;    DIE KOORDINATE 0,0 IST DIE LINKE UNTERE BILDSCHIRM-
;    ECKE !

;    IN  : A =KOORDINATE Y DES ZU ERZEUGENDEN PUNKTES
;          HL=KOORDINATE X DES ZU ERZEUGENDEN PUNKTES
;    OUT : WIE IN

POI: PUSH HL           ;ALLES ERHALTEN
     PUSH DE
     PUSH BC
     PUSH AF
     LD (PIY),A        ;NEUE Y-KOORDINATE ABLEGEN
     NEG               ;FUER ADRESSRECHNUNG AUFBEREITEN
     LD C,A            ;UND MERKEN
     LD A,H            ;X-KOORDINATE
     AND 01H           ;AUF BILDBEREICH BEGRENZEN
     LD H,A
     LD (PIX),HL       ;NEUE X-KOORDINATE ABLEGEN
     LD A,L            ;NIEDRIGSTE 3 BIT DER X-KOORDINATE
     AND 07H           ;AUFHEBEN (POSITION IM BILDBYTE)
     LD B,3            ;HL 3 BIT RECHTSSCHIEBEN
P00: SRL H
     RR L
     DJNZ P00
     LD DE,RAM-40H     ;BASISADRESSE BILD-RAM - 40H
     ADD HL,DE         ;ADRESSRECHNUNG BILDSPALTE
     LD DE,0040H       ;OFFSET ADRESSRECHNUNG BILDZEILE
     LD B,C            ;Y-ADRESSE ALS SCHLEIFENZAEHLER
P01: ADD HL,DE         ;ADRESSRECHNUNG BILDZEILE
     DJNZ P01
     LD B,A            ;POSITION INNERHALB DES BYTES
     INC B             ;ALS ROTATIONSZAEHLER
     LD A,0FEH         ;ZUNAECHST BILDBYTE MIT BIT 0
P02: RRCA              ;ROTIEREN
     DJNZ P02          ;BIS ZIELBIT ERREICHT
     LD C,A            ;IN C MERKEN
     LD A,(MOD)        ;BILDSCHIRM-MODE
     RRCA              ;IM CARRY MERKEN
     LD A,(SEG)        ;AKTUELLE RAM-SEGMENTNUMMER
     LD B,A            ;MERKEN
     LD A,03H          ;BILD-SEGMENT
     LD (SEG),A        ;ABLEGEN
     OUT 90H           ;UND AUSGEBEN
     LD A,C            ;PIXELMASKE NACH A
     LD C,(HL)         ;AKTUELLES BILDBYTE HOLEN
     JRNC P03          ;SPRUNG,FALLS INVERS-MODE
     AND C             ;PIXEL DAZUBLENDEN
     JR P04
P03: CPL               ;MASKE UMKEHREN
     OR C              ;UND DAZUBLENDEN
P04: LD (HL),A         ;NEUES BILDBYTE ABLEGEN
     LD A,B            ;BISHERIGES RAM-SEGMENT ZURUECK
     LD (SEG),A        ;ABLEGEN
     OUT 90H           ;UND EINSCHALTEN
     POP AF
     POP BC
     POP DE
     POP HL
     RET
```

```
;----UP LINIENTYP INITIALISIEREN-------------------------

;    ES KOENNEN 8 VERSCHIEDENE LINIENTYPEN ANGEWAEHLT
;    WERDEN:

;    IN  : A=LINIENTYP-NUMMER
;    OUT : WIE IN

LTY: PUSH HL
     PUSH BC
     AND 07H          ;NUMMER BEGRENZEN
     LD B,0           ;BC=NUMMER
     LD C,A
     LD HL,LTT        ;ZEIGER IN LINIENTYPTABELLE STELLEN
     ADD HL,BC
     LD A,(HL)        ;LINIENBILD
     LD (LIN),A       ;IN ROTATIONSZELLE EINTRAGEN
     LD A,C           ;A REGENERIEREN
     POP BC
     POP HL
     RET


;----LINIENMUSTERTABELLE---------------------------------

LTT: DB 0FFH          ; ----------------
     DB 0F0H          ; ----    ----
     DB 0CCH          ; --  --  --  --
     DB 088H          ; -   -   -   -
     DB 0AAH          ; - - - - - - - -
     DB 0E4H          ; ---  -  ---  -
     DB 0EAH          ; --- - - --- - -
     DB 080H          ; -       -


;----UP LINIE ERZEUGEN-----------------------------------

;    DAS PROGRAMM ERZEUGT MIT HILFE DES UP PIXEL EINE
;    LINIE ENTSPRECHEND DEM MOMENTAN GUELTIGEN LINIENTYP
;    ZWISCHEN DER AKTUELLEN PIXELPOSITION UND EINEM
;    ZU UEBERGEBENDEN ZIELKOORDINATENPAAR. DIE PIXEL-
;    POSITION IST BEI AUSTRITT AUS DEM PROGRAMM GLEICH
;    DEN ZIELKOORDINATEN.

;    IN  : A=ZIELKOORDINATE Y
;          HL=ZIELKOORDINATE X
;    OUT : WIE IN

PL0: PUSH HL          ;ALLES BLEIBT
     PUSH DE
     PUSH BC
     PUSH AF
     LD C,A           ;C = Yneu
     LD A,H           ;X-KOORDINATE
     AND 01H          ;AUF BILDBEREICH BEGRENZEN
     LD H,A
     EX DE,HL         ;DE = Xneu
     LD HL,(PIX)      ;HL = Xalt
     OR A             ;RESET CARRY
     SBC HL,DE        ;Xalt - Xneu
     LD B,08H         ;MERKE: X UNABH. VAR., dX<0, dY<0
     JRNC PL1         ;SPR.,FALLS Xalt >= Xneu
     ADD HL,DE        ;SONST: SUBTRAKTION UMKEHREN
     EX DE,HL
     OR A
     SBC HL,DE
     SET 0,B          ;MERKE: dX>0
PL1: LD A,(PIY)       ;A = Yalt
     SUB C            ;Yalt - Yneu
     JRNC PL2         ;SPR.,FALLS Yalt >= Yneu
     NEG              ;SONST: ZWEIERKOMPLEMENT
     SET 1,B          ;MERKEN dY>0
PL2: LD D,0           ;DE = /dY/
     LD E,A
     LD A,H           ;TEST,OB dX>=dY
     OR A
     JRNZ PL3         ;SPR.,FALLS JA
     LD A,L
     CMP E
     JRNC PL3         ;SPR.,FALLS JA
     EX DE,HL         ;SONST: TAUSCHEN X <--> Y
     RES 3,B          ;MERKE: dY UNABHAENGIGE KOORDINATE
     SET 2,B
PL3: LD A,B           ;MERKMALE IN A
     LD B,H           ;BC = SCHLEIFENZAEHLER
     LD C,L
     ADD HL,HL
     LD (DIX),HL      ;DIX = 2 * dX
     EX DE,HL
     ADD HL,HL
     LD (DIY),HL      ;DIY = 2 * dY
     LD H,B           ;HL = dX ZUANFANGS
     LD L,C
     LD D,A           ;MERKMALE NACH D
PL4: LD A,B           ;BC = 0 ?
     OR C
     JRNZ PL5         ;SPRUNG,FALLS NICHT
     POP AF           ;SONST: FERTIG
     POP BC
     POP DE
     POP HL
     RET
PL5: DEC BC           ;ZAEHLER VERMINDERN
     LD A,D           ;MERKMALE NACH A
     LD DE,(DIY)
     ADD HL,DE        ;HL = HL + 2 * dY
     LD DE,(DIX)
     OR A
     SBC HL,DE        ;HL = HL + 2 * dY - 2 * dX
     JRNC PL6         ;SPR.,FALLS SCHRITT
     ADD HL,DE        ;SONST: HL = HL + 2 * dY
     SCF              ;SET CARYY
PL6: LD D,A           ;VORZEICHEN UND UNABH. VAR. MERKEN
     JRC PL7          ;SPRUNG,FALLS KEIN SCHRITT
     OR 0CH           ;SONST: dX UND dY FREIGEBEN
PL7: PUSH HL          ;RETTEN ADDITIONSZELLE
     LD E,A           ;MERKMALE IN E
     LD A,(PIY)       ;MOMENTANE KOORDINATEN HOLEN
     LD HL,(PIX)
     BIT 3,E          ;dX FREIGEGEBEN ?
     JRZ PL8          ;SPR.,FALLS NICHT
     DEC HL           ;SONST: dX = -1 ZUNAEC
     BIT 0,E          ;RICHTIG ?
     JRZ PL8          ;JA
     INC HL           ;SONST: dX = +1
     INC HL
```

```
PL8: BIT 2,E          ;dY FREIGEGEBEN ?
     JRZ PL9          ;NEIN
     DEC A            ;SONST: dY = -1
     BIT 1,E          ;RICHTIG ?
     JRZ PL9          ;JA
     INC A            ;SONST: dY = +1
     INC A
PL9: LD E,A           ;A RETTEN
     LD A,(LIN)       ;LINIENBYTE HOLEN
     RLCA             ;ROTIEREN
     LD (LIN),A       ;ABLEGEN
     LD A,E           ;A ZURUECK
     CAC POI          ;PIXEL ERZEUGEN,FALLS ERLAUBT
     LD (PIX),HL      ;KOORDINATEN STETS ABLEGEN
     LD (PIY),A
     POP HL           ;ADDITIONSZELLE
     JR PL+           ;UND WEITER
```

Hinweis:
Bitte auf S. 27, 14. Zeile von unten, hinter „Videocontroller" das Wort „liefert" ergänzen.

Literatur

[1] Bogatz, A.: Tastatur- und Anzeigebaugruppe für Mikrorechner. – In: radio fernsehen elektronik. – Berlin 34 (1985) 11. – S. 727—730

[2] Bogatz, A.: Videocontrollerschaltung. – In: radio fernsehen elektronik. – Berlin: 34 (1985) 8, S. 500—502

[3] Bogatz, A.; Lenzner, J.: Mikrorechner-Modulsystem. – In: radio fernsehen elektronik. Berlin: 33 (1984) 12. – S. 771 bis 772

Autor:
Dipl.-Phys. Andreas Bogatz
Würzburger Str. 12a
Leipzig
DDR – 7031

Korrekturhinweis zu Heft 11

Beitrag
Elektronische Schreibmaschine S 6005 als Drucker

Leider haben sich im Programmlisting ein paar Fehler eingeschlichen.

Folgende Programmzeilen lauten richtig:

```
1ØØ2  D3 Ø1      OUT PORT Ø1  PIO im Bit E/A-Mode
1ØØ6  D3 Ø1      OUT PORT Ø1  8 Ausgänge
1ØØ8  D3 ØØ      OUT PORT ØØ  keine Taste "betätigt"
1ØØB  D3 ØØ      OUT PORT ØØ  "Druck der Zeichentaste"
1ØØD  CD 1E 1Ø CALL ZEIT KURZ
1Ø14  CD 29 1Ø CALL ZEIT LANG
1Ø18  D3 ØØ      OUT PORT ØØ
1Ø1A  CD 1E 1Ø CALL ZEIT KURZ
1Ø1D  C9       RET
```

Durch das Einfügen des OUT-Befehls auf Adresse 1Ø18 verschieben sich die Adressen der beiden Unterprogramme ZEIT KURZ und ZEIT LANG. Die Adressen des Listings auf S. 64 sind also jeweils um zwei zu erhöhen.

Einsatz graphischer Bildschirmsysteme

Die Entwicklung der Bauelementebasis ermöglicht es dem Amateur, sich mit graphischen Bildschirmsystemen zu befassen. Es läßt sich sowohl im Amateurbereich als auch auf kommerziellem Gebiet eine stark zunehmende Verbreitung feststellen, welche mit einer Vielzahl technischer Lösungen einhergeht. Dabei werden die Leistungsparameter wesentlich von den verfügbaren Speicherbauelementen und der Steuerung bzw. dem Steuerschaltkreis bestimmt.

Das Hauptproblem für die Anwendung im Hobbybereich besteht im Mangel an sinnvollen Aufgaben zum Einsatz graphischer Systeme. Die Ursache dafür ist der gegenüber pseudographischen Systemen erheblich höhere technische Aufwand, welcher zudem mit steigender Auflösung der Bilder exponentiell anwächst. Gerade die interessanten technischen Anwendungen erfordern aber hohe Auflösungen, leistungsfähige E/A-Geräte und eine hohe Verarbeitungsleistung. Es ergeben sich dadurch für den Amateurbereich zwei wesentliche Anwendungsschwerpunkte:

- die graphische Gestaltung von Spielen und
- der Einsatz der Graphik für Lehr- und Lernzwecke in mathematisch-technischen Disziplinen.

Ergänzt werden kann dies beispielsweise mit Programmen zum Zeichnen von Leiterplatten und zur Anfertigung von kleinen Zeichnungen, was jedoch schon entsprechende Peripheriegeräte voraussetzt.

1. Technische Voraussetzungen

Da der Aufwand für graphische Bildschirmsysteme sehr hoch ist, sollte man sich vor dem Baubeginn ausführlich die Randparameter überlegen. Diese Dimensionierungen werden dabei sicher wesentlich vom Anwendungsfall abhängen. Es ist zwar nicht sinnvoll, den Bildschirm nur auf einen Anwendungsfall auszulegen, man muß aber vor allem an dieser Stelle vor dem Bau eines alle Probleme lösenden Universalsystems warnen. Letzteres hält man ökonomisch nicht durch oder scheitert am oft zu gering veranschlagten Programmieraufwand.

Im folgenden Teil sollen einige Fragestellungen aufgeworfen werden, die eine Dimensionierung beeinflussen.

1.1. Auflösung des Bildes

Diese Größe, die Einfluß auf verschiedene weitere Fragen hat, läßt sich vom zu lösenden Problem sehr schnell ableiten. Da die Vorstellungen sicher sehr differieren, an dieser Stelle einige Zahlenwerte, die man dazu wissen sollte. Diese Werte basieren dabei auf der Annahme, daß im Amateurbereich gewöhnlich Fernsehgeräte als Display eingesetzt werden.

Möchte man auf dem Bildschirm ein gleichseitiges, quadratisches Bild erzeugen, so stehen von den 64 Mikrosekunden Zeilenlänge etwa 36 Mikrosekunden zur Verfügung, wobei 256 Linien vertikal als Bezugsgröße gelten. Möchte man mehr formatfüllend arbeiten, kann man etwa 43 Mikrosekunden der Zeile nutzen, wobei sich eine Auflösung von 320 Punkten ergibt. Diese Punktzahl stellt nach [1] auch bereits die Grenze bei einer Farbfernsehbildröhre dar. Der nächste Schritt in der Auflösung bringt bei optisch unverzerrtem Bild eine Verdopplung der Punkt- und Zeilenzahl, also 640 × 512 Punkte. Diese kann man mit handelsüblichen Geräten bereits nicht mehr in Farbe darstellen. Die Wiedergabe mit einem umgebauten (!!!) JUNOST ist noch möglich gewesen.
Zusammengefaßt ergibt sich somit das folgende Bild:

Punktzahl	Speicherbedarf (Bit) s/w	8 Farben	Punkttakt bei (36) 43 Mikrosekunden
256 × 256	64 K	192 K	(7,1 MHz) 5,9 MHz
320 × 256	80 K	240 K	4,7 MHz
512 × 512	256 K	768 K	(14,2 MHz) 11,9 MHz
640 × 512	320 K	960 K	14,9 MHz
1024 × 1024	1024 K	3072 K	(28,4 MHz)

1.2. Möglichkeiten der Bildschirmsteuerung

Zur Erzeugung eines graphischen Bildes und Ansteuerung eines Fernsehgerätes kann man drei prinzipielle Wege beschreiten, die sich vor allem im technischen Niveau unterscheiden. Die einfachste Möglichkeit ist ein diskreter Aufbau der Steuerung. Diese wird unter Umständen einfacher als bei einem alphanumerischen Bildschirm, da gegenüber diesem der Zeichengenerator fehlt. Es gibt auch weniger Probleme bei Bildpunktzahlen, welche keine glatte Zweierpotenz darstellen, da man hier das gesamte Bild über einen linear arbeitenden Start-Stop-Zähler adressieren kann. Ein Beispiel für eine derartige Ansteuerung ist in [2] vorgestellt.
Die technisch nächste Stufe wäre der Einsatz eines Bildschirmcontrollers. Diese Bauelemente übernehmen allgemein den gesamten Aufwand der Adressierung des Bildwiederholspeichers und der zeitlichen Aufbereitung aller Steuersignale einschließlich Fernsehbildansteuerung. Sie bringen ebenfalls noch keine den Bildinhalt betreffenden Verarbeitungsleistungen. Ein typischer Vertreter dafür ist der 6845 von Motorola, welcher in [3] vorgestellt wird. Dieser Schaltkreis läßt sich sowohl für alphanumerische als auch für graphische Systeme einsetzen. Er erlaubt eine umfangreiche Programmierung von Funktionen des Bildaufbaus (Punktzahl, Zeichenzahl, Synchronimpulse, . . .) sowie die Funktionen Cursor und Blinken. Er kann außerdem durch seine 14 Adreßleitungen in geringem Umfang eine Ausschnittdarstellung aus einem größeren Bildwiederholspeicher realisieren. Ein weiterer Vertreter dieser Gruppe ist der I8275 von Intel, der weit verbreitet ist.
Beide bisher genannten Möglichkeiten bringen keine Leistungen zur Gestaltung des Bildinhaltes. Alle Funktionen zur Darstellung sind über entsprechende Treibersoftware zu realisieren.
Den dritten Weg stellt der Einsatz von Graphikprozessoren dar, wovon der EF9365 · · · 9367, der µPD7220 und der I82720 typische Vertreter sind. Es gibt

außerdem eine große Anzahl weiterer, die im Rahmen von Heimcomputern in großer Vielfalt für diese speziell entwickelt wurden. Das Hauptmerkmal für alle ist, daß sie außer der Steuerung des Bildaufbaus auch inhaltliche Leistungen übernehmen. So bietet der EF9365 [4], [5] ein noch überschaubares Spektrum an Leistungen an:

Formatsteuerung 256*256 oder 512*512	
interner Zeichengenerator (ASCII)	
Bild löschen	
Wahl der Zeichengröße	(256 Kombinationen)
Wahl der Schriftart	(4 Arten)
Wahl des Linientyps	(4 Arten)
Zeichnen von Vektoren	
Cursorverwaltung	
Lichtstiftbearbeitung	
Realisierung von	Refresh, Adressenmultiplex, Ansteuerung des Bildwiederholspeichers

Mit dem Einsatz von Graphikprozessoren sinkt zwar der Hardwareaufwand gegenüber Controllern nur gering, dafür steigert man aber die Darstellungsgeschwindigkeit erheblich. Auch ist die damit verbundene Softwareeinsparung nicht zu unterschätzen.
Ein zu nennender Nachteil ist das Fehlen der Rollbildfunktion. Es ist auch mühselig, diese nachzugestalten, da der Bildspeicher im E/A-Bereich liegt, wodurch bereits das Einlesen des Bildes etwa 262000 E/A-Operationen darstellt, was allein ungefähr 1,1 Sekunde dauert.

1.3. Gestaltung der Bildschirmsteuerung

Nachdem einige Bemerkungen zu den Aufbaumöglichkeiten gemacht wurden, noch einige Gedanken zur technischen Gestaltung der Bildschirmsteuerung. Die in der Literatur angegebenen Schaltungen stellen allgemein eine Minimalvariante oder eine Durchschnittskonfiguration dar. Es ist also jedem selbst überlassen, diese Schaltungen anwendungsspezifisch zu erweitern. Als Modifikationsmöglichkeiten kommen dabei prinzipiell in Frage:

a) Erweiterung auf Farbe durch n parallel laufende Bildspeicher und eine Steuerung zum Setzen und Löschen von Vordergrund- und Hintergrundfarbe,

b) Einrichtung eines vom Prozessor bedienten Bildfensters in einem größeren Bildspeicher durch Manipulation der Speicheradressen,

c) Einsatz mehrerer Bildspeicher mit getrennter Lese-Schreib-Steuerung zur Realisierung optischer Ebenen,

d) Möglichkeit der Rücklesbarkeit des Bildspeichers (diese ist bei Graphikprozessoren nicht immer gegeben).

Besonders hervorgehoben werden soll die Arbeit mit mehreren getrennt steuerbaren Bildspeichern. Dies bringt den Vorteil, daß man verschiedene Betriebsweisen fahren kann, welche sich bei Programmiersprachen wie LOGO als günstig erweisen. In diesem Fall zum Beispiel konnte ein getrennter Bildteil und Textteil geschrieben werden, welche sich im Bedarfsfall gleichzeitig (übereinander) darstellen lassen. Auch die Realisierung von Bewegungen vor einem »nur-lese«-Hintergrund wäre eine denkbare Anwendung. Allein die Trennung von Kommandoteil und Graphik würde diesen Aufwand rechtfertigen, wenn man dafür nicht ein zweites, alphanumerisches Display einsetzen will.

2. Beschreibung des verwendeten Systems

Alle hier beschriebenen Experimente und Untersuchungen beruhen auf einer Graphiksteuerung, welche aus einer Schaltung nach [4] abgeleitet wurde. In diesem System bildet der Graphikprozessor (GDP) EF9365 das Kernstück. Er wurde für das Bildformat 256 × 256 aufgebaut. Als Besonderheiten gegenüber der Originalschaltung wurden folgende wesentliche Änderungen ausgeführt:

a) Der Speicher wurde mit 8 Bit Aufrufbreite realisiert, wobei jeweils 4 Bit auf ein Schieberegister führen.
Dadurch entstehen zwei parallel arbeitende Bildspeicher.

b) Die WR-Signale wurden direkt am jeweiligen Speicherblock getort, so daß eine getrennte oder gemeinsame Schreibfreigabe erfolgen kann.

c) Die RD-Selektion erfolgt durch Trennung der Ausgangssignale der Schieberegister. Der Mischer wurde auf weißes Bild mit schwarzer Schrift umgestellt und läßt eine Überlagerung von beiden Bildern zu.

d) Einbau einer DMA-Schnittstelle für den Bildspeicher 1 zum Rücklesen des Bildinhaltes.

3. Textausgabe auf Graphikbildschirmen

Steht im Rechnersystem nur ein Display zur Verfügung, so ist man gezwungen, auf diesem auch die Textkommunikation durchzuführen. Durch die Arbeitsweise eines Graphikdisplays ergibt das eine Reihe von Problemen, die man von normalen alphanumerischen Bildschirmen her nicht kennt.
So wird die Abhängigkeit von Bildpunktzahl und Zeilenzahl wesentlich deutlicher klar, da man bei 256 Punkten nur 42 Zeichen unterbringt. Für eine Darstellung von 64 Zeichen je Zeile ist also bereits ein System mit 512 horizontalen Bildpunkten erforderlich. Hält man das Bildformat auf Grund der oben angestellten Betrachtungen quadratisch, bekommt man zwar 85 × 56 Zeichen unter, kann die Schrift aber nicht längere Zeit lesen, da sie extrem klein wird.
Ein weiteres, schon erwähntes Problem ist das fehlende Rollbild. Dieses ist zwar allgemein technisch realisierbar, ergibt aber Ausführungszeiten, welche sich bei der Programmabarbeitung störend bemerkbar machen. Es ist auch die Frage, ob eine Rollbildfunktion auf einem Graphikdisplay in jedem Fall sinnvoll ist. Für den alphanumerischen Betrieb wurde das Problem durch einen umlaufenden »Blattrand« gelöst, der in Form einer Trennlinie unter der Cursorzeile gezeichnet wird. Außerdem wird bei einer Zeilenschaltung die neue Cursorzeile gelöscht.
Ein drittes Problem, welches sich hier unmittelbar anschließt, ist die Überschreibung von Zeichen. Beschriftet man eine Graphik, so ist es gut, wenn ein Zeichen eine Linie so schneidet, daß diese einfach überschrieben und nicht ein Zeichenfeld herausgeschnitten wird. Dieser Effekt wirkt sich beim Schreiben von Texten genau umgekehrt aus, vor allem wenn man mit direktpositionierenden Ausgaben (z. B. PRINT.AT . . .) arbeitet. Ein alphanumerisches Display löscht das alte Zeichen an einer Position durch Austausch gegen ein neues, womit z. B. alle 8 × 8 Punkte neu belegt werden. Der Graphikprozessor setzt nur die schwarzen Punkte neu, ohne den Untergrund zu beachten, wodurch man nach einer Weile einen Klecks an dieser Stelle hat. Der Alphatreiber muß vor dem Setzen eines Zeichens den dafür notwendigen Ort erst löschen.
Um die zwei zuletzt genannten Probleme weitestgehend zu beseitigen, hat

es sich als günstig erwiesen, zwei getrennte Bildschirmtreiber zu verwenden, die die Anwendungsfälle »nur Text« und »Text mit Graphik« trennen. Der erstere realisiert den »Blattrand« und löscht die Zeichenpositionen, wogegen der zweite diese Funktionen unterdrückt.

4. Drucken von Vollgraphik

Der Druck von Vollgraphik ist programmtechnisch einfacher als der Druck von Pseudographik, da die Punkte allgemein wahlfrei gelesen werden können.

Da der Bildwiederholspeicher gewöhnlich punktweise gelesen werden kann, holt man sich die Punkte in der benötigten Reihenfolge. Es sind weder Drehungen von Punktsätzen noch die Pufferung der Daten notwendig. Die Druckerausgabe erfolgt selbstverständlich als geschlossener Bildschirmabzug, da auf Grund der mechanischen Konstruktion des Druckers eine vertikale Vorwärts-Rückwärts-Bewegung nicht möglich ist.

Das Druckerprogramm (s. Beilage) besteht aus zwei ineinander laufenden Schleifen und einem Unterprogramm zum Lesen der Bildpunkte. Die innere Schleife druckt eine Zeile aus sechs Bildschirmlinien. Diese Zeile wird mit einem geschlossenen Graphikaufruf an den Drucker übergeben, um die Zeit für die Sichtautomatik zu sparen. Die äußere Schleife realisiert die sich ergebenden 43 Druckerzeilen. Da der Bildschirm sich koordinatenmäßig im ersten Quadranten befindet, erfolgt die Ausgabe mit abnehmenden y-Werten und steigenden x-Werten. Der Startpunkt liegt demzufolge bei $x = 0$ und $y = 255$.

5. Ein BASIC-Interpreter zur Steuerung eines Graphikprozessors

Um den im System verwendeten Graphikprozessor EF9365 einfach nutzen zu können, wurde ein BASIC-Interpreter als Arbeitsmittel entwickelt und implementiert. An und mit diesem wurden Untersuchungen zu den Möglichkeiten des Prozessors und der Leistungsfähigkeit der gewählten BASIC-Befehle durchgeführt.

5.1. Der BASIC-Interpreter

Als Basis für den Interpreter diente ein TINY-BASIC mit Integerarithmetik. Dieses Programm wurde in seiner Struktur beherrscht und stand als Quelle zur Verfügung. Die Arbeit mit ganzen Zahlen reicht für den Graphikbildschirm vom Prinzip her auch aus, da nur ganze Punkte ansteuerbar sind. Probleme entstehen durch die notwendigen Winkelfunktionen und das Runden bei arithmetischen Operationen. Es waren deshalb einige spezielle Untersuchungen notwendig, um diese Aufgabe zu lösen. Als Betrachtungsbasis dafür wurde ein Bildschirm mit 256×256 Punkten angenommen.

Winkelfunktionen

Da die Winkelfunktionen Ergebnisse kleiner gleich eins bringen, muß deren Berechnung mit einem Vergrößerungsfaktor erfolgen. Da die Arithmetik abzüglich Vorzeichen 15 Bit bewältigt, ergibt sich der maximale Faktor im Fall eines bildschirmfüllenden Viertelkreises, also bei einem Radius von 256 Punkten. Da der Kreis mit der Formel

$$X = Xm + r \times \cos W$$

berechnet werden kann, ergeben sich bei 8 Bit für den Radius noch sieben Bit für den maximalen Wert der Winkelfunktion. Aus diesem Grund wurde der Wertebereich für SIN und COS auf $-128 \cdots +128$ festgelegt. Programmtechnisch erfolgt die Berechnung des cos durch Umrechnung in eine wertentsprechende sin-Funktion. Durch die

Integerarithmetik erfolgt die Berechnung in Grad. Das Argument wird dabei mittels einer Modulo-360-Rechnung auf den Grundkreis zurückgeführt und auf den 90-Grad-Bereich gefaltet. Die Funktionswerte für diesen Winkelbereich wurden in einer Tabelle abgelegt, womit außerdem eine hohe Bearbeitungsgeschwindigkeit erreicht wird.
Zur schnellen Korrektur des Dehnungsfaktors wurde die Funktion DIV(X) eingeführt, welche eine Festdivision durch 128 realisiert.

Kreisberechnung

Der Interpreter sollte mit einer komplexen Funktion zum Zeichnen von Kreisen und Ellipsen ausgestattet werden. Dabei war als erstes zu untersuchen, in welcher Art und Weise der Kreis zu zeichnen ist.
Der Kreis kann prinzipiell nach zwei Verfahren gezeichnet werden:

a) Es werden an den errechneten Koordinaten Punkte gezeichnet, oder
b) es wird von Koordinate zu Koordinate ein Vektor gezeichnet.

Die Entscheidung läßt sich mit einer Proberechnung herbeiführen. Die größte Koordinatenänderung auf der Y-Achse entsteht bei sehr kleinen Winkeln. Da nur Winkel im ganzen Gradschritt nutzbar sind, ergibt sich für einen Radius von 256 Punkten folgendes Bild:

Winkel in Grad	Y-Koordinate
0	0
1	4
2	9
3	13
4	18
usw.	

Damit steht fest, daß Vektoren gezeichnet werden müssen, wenn auch größere Kreise als geschlossene Linien dargestellt werden sollen.

Die zweite Problemstellung ist die Auswahl der Parameter der Kreisfunktion. Als Forderung standen die folgenden Funktionen:
– Zeichnen von Kreisen
– Zeichnen von Ellipsen
– Zeichnen beliebiger Bahnabschnitte
– Verdrehung im Koordinatensystem
Für das Zeichnen von Kreisen und Ellipsen bietet es sich an, die allgemeine Ellipsenformel zu verwenden. Es werden somit die Mittelpunktkoordinaten und beide Radien benötigt. Für die Bahnabschnitte ist der Anfangs- und Endwinkel erforderlich und für die Verdrehung der entsprechende Winkel. Es ergeben sich damit für die Gesamtfunktion sieben Parameter. Um die Rechengeschwindigkeit zu erhöhen, wurde weiterhin die Größe »Winkelschritt« eingeführt. Dies ist bei der vektoriellen Arbeitsweise günstig, da bei kleinen Radien nicht unbedingt in 1-Grad-Schritten gerechnet werden muß. So kann zum Beispiel bei Radien kleiner 60 Punkte in 4-Grad-Schritten gearbeitet werden, wodurch die Rechenzeit auf 25 % sinkt. Durch diesen Parameter ergeben sich im Zusammenhang mit der vektoriellen Arbeitsweise sehr interessante Effekte. So kann z.B. mit der Kreisfunktion durch Angabe des Winkelschrittes mit 60 Grad ein Sechseck gezeichnet werden.
Als neunter Parameter wurde der Winkeloffset eingeführt, welcher intern zum Anfangs- und Endwinkel addiert wird. Er dient lediglich der Vereinfachung der Programmierung, wenn beispielsweise mit Kreisabschnitten konstanter Länge über eine Laufvariable gearbeitet werden soll. Die Kreisfunktion wurde so implementiert, daß die ersten vier Parameter unbedingt angegeben werden müssen. Für alle anderen werden Standardwerte genutzt, wenn diese Angaben fehlen. Der Aufruf als Gesamtheit lautet:

CIRCLE Xm, Ym, a, b
[, Tanf, Tend, FO, WS, SW]
mit

Xm, Ym	Mittelpunktkoordinate
a, b	Radien
Tanf	Anfangswinkel
Tend	Endwinkel
FO	Systemverdrehung
WS	Winkelschritt (Rechenauflösung)
SW	Startwinkel (Offset zu Tanf, Tend)

Der Aufruf zum Zeichnen eines Kreises mit verringerter Rechenzeit hat dann z.B. das folgende Aussehen:

CIRCLE 100, 100, 50, 50, , , , 5

Zur Realisierung der Funktion war eine zusätzliche 32-Bit-Multiplikation erforderlich, die aber nur intern für die Kreisfunktion genutzt wird. Der eigentlichen Kreisberechnung geht ein wesentlich größerer Programmteil zur Parameteraufnahme und Berechnung von Arbeitskonstanten voraus. Der Kreis wird mit der folgenden Formel berechnet:

$$X = Xm + a \times COS(T) \times COS(FO) - b \times SIN(T) \times SIN(FO)$$

$$Y = Ym + a \times COS(T) \times SIN(FO) + b \times SIN(T) \times COS(FO)$$

Objektverbindungen

Ein weiteres zu lösendes Problem ist die Verbindung von Figuren untereinander, ohne alle Punkte neu zu berechnen. Will man z.B. zwei Kreise zu einem Zylinder verbinden, so müßten die Endpunkte der Verbindungslinien unabhängig von der Kreisberechnung nochmals bestimmt werden. Es wäre also zweckmäßig, aus der Kreisberechnung Werte für die weitere Verwendung herauszuziehen.

Zu diesem Zweck muß die Arbeitsweise des Systems bekannt sein. Der wichtige Kern ist hierbei das Linienprogramm. Es verfügt über einen Koordinatenstapel, dessen unterstes Datenpaar nach der Funktionsausführung den aktuellen Cursorstand enthält. Da auch der Kreis mit Hilfe des Linienprogramms gezeichnet wird, stehen nach jedem Kreissegment die Endkoordinaten in diesem Stapel zur Verfügung. Da diese Daten aber nicht vom BASIC aus direkt zugänglich sind, wurden zwei Wege zur Nutzung geschaffen. Das ist zum einen der Befehl LINETO, welcher vom letzten Endpunkt eine Linie zu einem neuen (End-)Punkt zeichnet. Zum anderen kann man sich den Wert mit Hilfe des Befehls ENTER in einen Variablenstack übertragen, wo er zur späteren Verwendung bereit liegt.

Für die Zwischenwertgewinnung bei Geraden mußte ein gesondertes Programm geschrieben werden, da die eigentliche Linienfunktion im Graphikprozessor realisiert wird. Diesem Sonderprogramm wird mit dem Befehl PUNKT die Anzahl der Zwischenwerte vorgegeben, welche er ebenfalls im Variablenstack ablegt.

5.2. Der Befehlssatz des BASIC-Interpreters

Ein Graphikbildschirm läßt sich zwar bereits mit zwei Befehlen, dem Punkt-Setzen und -Löschen, bedienen, erfordert aber damit immer noch eine aufwendige Programmierung. Ein Graphikprozessor, der selbst bereits höhere Leistungen anbietet, wäre bei Beschränkung auf diese elementaren Zugriffe sinnlos.

Es wurden Überlegungen angestellt, wie man zu einem Kompromiß zwischen einer optimalen Ausnutzung der Leistungen des Prozessors und einer noch weitestgehend prozessorunabhängigen Schnittstelle kommt.

Als Schnittstelle wurden die folgenden Leistungen herausgearbeitet, die sich in jedem Fall implementieren lassen müssen:

- Setzen Punkt auf Koordinate X, Y
- Löschen Punkt auf Koordinate X, Y
- Setzen/Löschen einer Linie von X1, Y1 nach X2, Y2
- Lesen eines Punktes von der Koordinate X, Y
- Umschalten des Zeichenstiftes auf aktiv/passiv (PENUP/PENDOWN)
- Löschen des Graphikbildschirmes

Aufbauend auf diese Schnittstelle, wurde ein Befehlssatz für den BASIC-Interpreter entworfen. Dabei wurde zusätzlich versucht, möglichst viele Aufgabenstellungen durch leistungsfähige Befehle abzufangen, welche auf Maschinenebene arbeiten (s. Beilage).
Der gesamte Befehlssatz des Interpreters läßt sich in mehrere funktionelle Gruppen einteilen, wobei hier auf eine Gliederung der Standardbefehle verzichtet werden soll.
Diese Gruppen sind:

a) Standard-BASIC-Befehle
b) Generierung und Verwaltung von Graphikdaten
c) Steuerung der Zeichenfunktion
d) Zeichenfunktionen
e) prozessorspezifische Funktionen

Standardbefehle

Der Interpreter hat sich in seinem Kern über mehrere Jahre hinweg aus einem 3K-TINY-BASIC entwickelt. Die Graphikversion wurde aus dem bereits beschriebenen Interpreter für Pseudographikarbeit abgeleitet. Der Befehlsvorrat spiegelt dabei das Fehlen von Stringoperationen und Gleitkommaarithmetik wieder. Da keine Trennung zwischen Daten- und Adreßrechnung erfolgt, läßt er eine Vielzahl von Operationen zu, welche eine effektive Programmierung erlauben, aber das Programm sehr unübersichtlich gestalten können. So kann als Beispiel in einer GOTO-Anweisung ein arithmetischer Ausdruck verwendet werden, wie

50 GOTO ((50*X) + 1000)*(X < 5)

der zu den Adressen 1000, 1050, 1100, 1150, 1200 verzweigt. Die Gesamtheit aller Befehle ist im Anlagenteil aufgelistet.

Generierung und Verwaltung von Graphikdaten

Wie schon erwähnt, wurde zur Sicherung der Verbindung zwischen Graphikblock und Systemvariablen ein Variablenstack eingeführt. Zu dessen Bedienung existieren fünf Befehle. So kann er sequentiell geschrieben (PUSH) oder gelesen (POP) werden. Der Stapelzeiger kann über zwei weitere Anweisungen gesetzt und gelesen werden. Die fünfte Anweisung ist ENTER, mit welcher der Transport von Wertpaaren aus dem Graphikblock in den Variablenstack erfolgt. Die Abspeicherreihenfolge ist dabei Y, X, so daß mit dem POP-Befehl zuerst X gelesen wird.
Als einzeln stehender Befehl in dieser Gruppe ist noch die Pseudovariable DMA zu nennen, welche selbst zwei Parameter besitzt. Mit ihr kann der Bildschirminhalt punktweise gelesen werden.

Steuerbefehle

Diese Gruppe besteht aus fünf Befehlen. Sie enthält die Schreibstiftsteuerung und Schreibpositionssteuerung. Das Kommando SCAN entspricht im Normalfall dem CLR, löscht also den Bildschirm. Die Besonderheit besteht darin, daß er durch Inversschaltung das gesamte Bild auch auf Schwarz setzen kann.
Der Befehl INV wurde so programmiert, daß er bei verschiedenen Befehlen als Vorsilbe verwendet werden kann und eine Umschaltung von Schreiben (PEN) auf Löschen (ERASE) bewirkt.

Zeichenfunktionen

Die Zeichenfunktionen sind die Gruppe, welche den eigentlichen Kern des Graphiksystems ausmacht. In ihr sind als

Basis Befehle zum Zeichnen eines Punktes, einer Linie und einer Fläche enthalten.

* PSET / PRES

Mit diesen beiden Befehlen kann ein Bildpunkt gesetzt oder gelöscht werden.

* LINE

Zum Zeichnen einer Geraden zwischen zwei Punkten wird der Befehl LINE verwendet. Diese Linie kann mit INVLINE wieder gelöscht werden.

* LINETO

Bei diesem Befehl kann man sich bei fortlaufenden Linien die Angabe des Anfangspunktes ersparen. Dieser ergibt sich automatisch aus dem Endpunkt der letzten Linie, was der Cursorposition entspricht.

* AREA

Der Befehl AREA kann mit drei oder vier Koordinatenpaaren genutzt werden. Im Ergebnis entsteht ein Dreieck oder Viereck, wobei die Form völlig frei wählbar ist.

* CIRCLE / CIRCLESEG

Mit dem bereits ausführlich beschriebenen Befehl CIRCLE können sowohl Kreise und Ellipsen als auch Teile davon gezeichnet werden. Der Befehl CIRCLESEG unterscheidet sich nur dadurch, daß er die Radien mitzeichnet, so daß Flächensegmente entstehen.

* POLYSET / POLYMOVE

Zum Zeichnen nichtgeschlossener Vielecke wurden Befehle zum Aufbau und zur Bewegung von Polygonzügen implementiert. Sie benötigen zwar für die Daten den Umweg über den Speicher, sind damit aber schneller in der Abarbeitung. Sie sind vor allem zur Bewegung von Objekten im Bild gedacht.

* Sonderfunktionen

Dieser Teil umfaßt nur drei Befehle, die sich aus den konkreten Eigenschaften der verwendeten Hardware und dem Graphikprozessor ergeben. Das ist die Speichersteuerung des Graphiksystems, welches über zwei Bildwiederholspeicher verfügt. Sie wird mit dem Befehl BILD realisiert. Der Befehl TYPE wählt eine der vier möglichen Linientypen aus, und die Pseudovariable GDP stellt die implementierte Basisadresse des Graphikprozessors bereit.

5.3. Anwendungsbeispiele

Programm 1 (Bild 1)

In diesem Programm wird vor allem die Funktion AREA genutzt. Das Netz erreicht man durch ein gegensinniges Laufen der Endpunkte auf den beiden Achsen. Das umspannende Quadrat wurde ebenfalls mit der AREA-Funktion gezeichnet. Es hätte genausogut durch Einzellinien oder mit der Kreisfunktion erzeugt werden können. Letztere hätte dann das folgende Aussehen: CIRCLE 125, 125, 120, 120, , , , 90

```
10 INPUT K
20 PENDOWN:BILD9:CLR
30 FOR X=0 TO 120 STEP K
40 AREA 5+X,125,125,125+X,245-X,125,125,125-X
50 NEXT X
60 CIRCLE 125,125,120,120
70 AREA 5,125,125,5,245,125,125,245
```

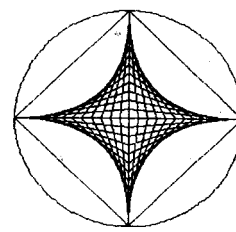


Bild 1. Beispielprogramm 1

Programm 2 (Bild 2 und Bild 3)

Dieses geometrische Gebilde wird aus miteinander verbundenen Kreisbögen (oben) bzw. Kreissegmenten (unten)

gebildet. Bei der gezeigten Abbildung wurde eine Schrittweite von 15° gewählt. Zuerst wird das untere Segment gezeichnet (Zeile 50) und der Endpunkt gerettet (Zeile 60). Anschließend erfolgt das Zeichnen des oberen, um 180° versetzt beginnenden Kreisbogens, dessen Endpunkt auf Zeile 80 mit dem des unteren verbunden wird. Bild 3 unterscheidet sich darin, daß die Ellipsen im Raum verdreht und die Laufrichtung der oberen Ellipse geändert wurden.

```
 5 PENDOWN
10 INPUT K
20 BILD 9
30 CLR
40 FOR X=0 TO 360 STEP K
50 CIRCLE 120,60,50,30,X,X+K,30
60 ENTER
70 CIRCLE 120,180,50,30,180-X,180-X+K,-30
80 LINETO 0,POP,POP
90 NEXT X
```

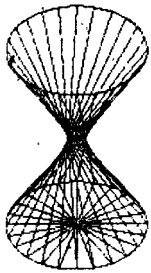


Bild 2
Beispielprogramm 2 (a)

```
 5 PENDOWN
10 INPUT K
20 BILD 9
30 CLR
40 FOR X=0 TO 360 STEP K
50 CIRCLE 120,60,50,30,X,X+K,30
60 ENTER
70 CIRCLE 120,180,50,30,180-X,180-X+K,-30
80 LINETO 0,POP,POP
90 NEXT X
```

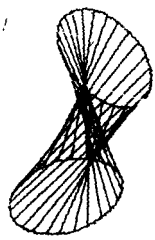


Bild 3
Beispielprogramm 2 (b)

Programm 3 (Bild 4)

In diesem Fall wird der Umstand ausgenutzt, daß die Kreisfunktion auch mit Endwinkeln größer als 360 Grad arbeitet. Es läßt sich dadurch mit einem Befehl und einer Schleifenanweisung eine Spirale zeichnen. Sie setzt sich aus Kreisabschnitten zusammen, welche von Stück zu Stück gedehnt werden.

```
 5 CLR
10 INPUT K
20 FOR X=0 TO 3600 STEP K
30 CIRCLE 128,128,(X/40)+20,(X/40)+20,X,X+K
40 NEXT X
```

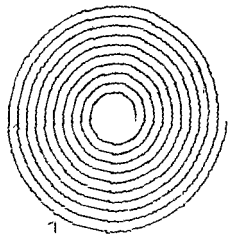


Bild 4
Beispielprogramm 3

Programm 4 (Bild 5)

Mit diesem Programm wurde versucht, einen räumlichen Eindruck zu erreichen. Das Bild setzt sich dabei aus ständig größer werdenden Kreisen zusammen, welche sich in einer Sinuskurve der Mittelpunkte aneinander reihen.

```
10 PENDOWN:CLR
15 K=5
20 FOR X=20 TO 200 STEP K
30 Y=(SIN(2*X)/2)+128
35 Z=X/5
40 CIRCLE X,Y,16+Z,16+Z,,,.5
50 NEXT X
```

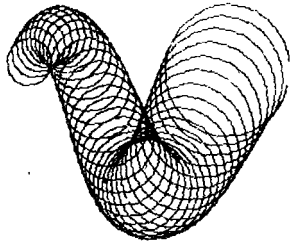


Bild 5
Beispielprogramm 4

Programm 5 (Bilder 6 bis 8)

Dieses Programm nutzt die Möglichkeiten der Überdrehung des Kreises und die Erhöhung der Rechenschrittweite aus. Ausgangspunkt war das Zeichnen eines siebenzackigen Sterns.

```
 2 K=7
 5 BILD 9;CLR
10 Z=360/K
15 Y=((K/2)*Z)+1
20 X=K*Y
30 CIRCLE 128,128,100,100,0,X,,Y
```

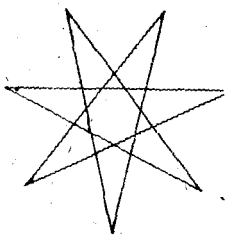


Bild 6. Beispielprogramm 5 (Grundversion)

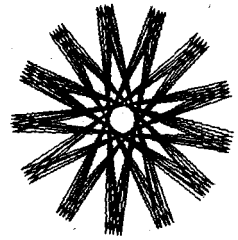
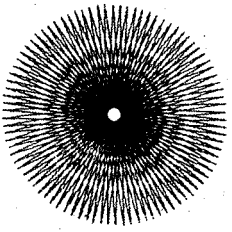


Bild 7. Beispielprogramm 5 (a)



```
 2 INPUT K
 5 BILD 9;CLR
10 Z=360/K
15 Y=((K/2)*Z)+1
20 X=K*Y
30 CIRCLE 128,128,100,100,0,X,,Y
36 CURSOR 0
38 K3=K/100;L=100*K3
40 K2=(K-L)/10;M=10*K2
42 K1=K-L-M
44 A=K3+48;B=K2+48;C=K1+48
46 OUT(GDP)=A;OUT(GDP)=B;OUT(GDP)=C
50 K=K+2
60 L=INCHAR
70 GOTO 5
```

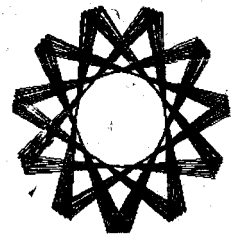


Bild 8. Beispielprogramm 5 (b)

```
;
;     COPY n
;                        n = 1    MASZSTAB 1:1
;                        n = 2         "=    2:1
;
S500:    LD       A,4       ; HIGHSPEED EIN
         CALL     WCMD1
         CALL     WERT
         LD       A,L
         CMP      2
         JPZ      S501      ; 2:1
;
;  COPY  1:1
;     (DE) = X-KOORDINATE
;     (HL) = Y-KOORDINATE
;
         LD       B,43
         LD       HL,255    ; Y-START
```

```
S500B:  LD      DE,0    ; X
        PUSH    HL
        LD      HL,STB1
        CALL    SDTEX   ; EROEFFNEN GRAPHICZEILE
        POP     HL
;
        EXX
        LD      B,0     ; B' = PUNKTPUFFER
        EXX
        LD      C,0     ; 256 mal
S500D:  CALL    S500C   ; PUNKT LESEN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        INC     HL
        INC     HL
        INC     HL
        INC     HL
        INC     HL
;
        EXX
        LD      A,B
        SLA     A       ; NADEL 1...6 NUTZEN
        CALL    SDOUT   ; ZEICHEN RAUS
        LD      B,0
        EXX
;
        INC     DE
        DEC     C
        JRNZ    S500D-# ; ZEILE NICHT ZU ENDE
;
        DEC     HL
        DEC     HL
        DEC     HL
        DEC     HL
        DEC     HL
        DEC     HL      ; 6 LINIEN TIEFER
        DJNZ    S500B-# ; BILD NICHT ZU ENDE
;
        JMP     S500G
;
;.....................................................
;
;
;  COPY  2:1
;     (DE) = X-KOORDINATE
;     (HL) = Y-KOORDINATE
;
S501:   LD      B,86
        LD      HL,255  ; Y-START
S501B:  LD      DE,0    ; X
        PUSH    HL
        LD      HL,STB2
        CALL    SDTEX   ; EROEFFNEN GRAPHICZEILE
        POP     HL
;
        EXX
        LD      B,0     ; B' = PUNKTPUFFER
        EXX
        LD      C,0     ; 256 mal
```

```
S501D:  CALL    S500C   ; PUNKT LESEN
        CALL    S500E   ; PUNKT DOPPELN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        CALL    S500E   ; PUNKT DOPPELN
        DEC     HL
        CALL    S500C   ; PUNKT LESEN
        CALL    S500E   ; PUNKT DOPPELN
        INC     HL
        INC     HL
;
        EXX
        LD      A,B
        SLA     A       ; NADEL 2...6 NUTZEN
        CALL    SDOUT   ; ZEICHEN RAUS
        CALL    SDOUT
        LD      B,0
        EXX
;
        INC     DE
        DEC     C
        JRNZ    S501D-# ; ZEILE NICHT ZU ENDE
;
        DEC     HL
        DEC     HL
        DEC     HL      ; 3 LINIEN TIEFER
        DJNZ    S501B-# ; BILD NICHT ZU ENDE
;
S500G:  LD      A,0CH
        CALL    SDOUT
        LD      A,3     ; HIGH-SPEED AUS
        CALL    WCMD1
        LD      A,1EH
        CALL    SD
        JMP     START
;
;.....................................................
;
;  TEXT AN DRUCKER (STEUERTEXTE)  ENDE = 03H
;
SDTEX:  LD      A,M
        CMP     3
        RZ
        INC     HL
        CALL    SDOUT
        JR      SDTEX-#
;
; STEUERUNG 1:1
;
STB1:   DB      0,0,0,0DH
        DB      1BH,5BH,31H,65H ; 1/2 ZEILE +
        DB      '               '
        DB      1BH,4BH,0,1     ; OPEN GRAPHIC
        DB      3
;
;  STEUERUNG 2:1
;
STB2:   DB      0,0,0,0DH
        DB      1BH,5BH,31H,65H ; + 1/2 ZEILE
        DB      '       '
        DB      1BH,4BH,0,2     ; OPEN GRAPHIC
        DB      3
;
;
        END
```

Literatur

[1] Bergmann, H.: Displayröhren erhöhter Auflösung. – In: radio fernsehen elektronik 35 (1986) 7. – S. 434
[2] Schmidt, W.: 8K-Vollgrafik für BASIC-Heimcomputer. – In: radio fernsehen elektronik 35 (1986) 9. – S. 563–565
[3] Klein, R. D.: Mikrocomputer Hard- und Softwarepraxis. – München, 1981
[4] Klein, R. D.: CRT-Controller unterstützt Graphikfunktionen. – In: Elektronik 8 (1981). – S. 63
[5] Klein, R. D.: Das mc-Grafic-Terminal. – In: mc 8 (1983). – S. 68

Autor:
Dr.-Ing. Gert Schönfelder
Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden

Literatur (Beitrag „Benchmarktests")

[1] Luther, W. D.: Die große BASIC-Referenztabelle. – Sprendlingen: Luther-Verlag, 1984
[2] Kleine Enzyklopädie Mathematik. – Leipzig: Enzyklopädie, 1967
[3] Kühnel, C.: FORTH – ein Softwarekonzept für Mikro- und Minicomputer. – In: Kleinstrechner-TIPS, Heft 10. – Leipzig: Fachbuchverlag, 1989

ISBN 3-343-00581-9

1. Auflage
Lizenznummer 114/210/3/90
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III/18-328
Redaktionsschluß: 15. 10. 1989
Bestellnummer: 547 6504
00780

Kleinstrechner-TIPS/Hrsg. von
Hans Kreul u. a. – Leipzig, Fachbuchverl.,
H. 12. – 1. Aufl. – 1990. – 64 S.: 25 Bild.

Anschrift des Verlages:
VEB Fachbuchverlag Leipzig
PSF 67
Leipzig
DDR – 7031

Herausgeber:
Prof. Dr.-Ing. *Hans Kreul*
Bruno-Schröter-Str. 1
Zittau
DDR – 8800
Prof. Dr. sc. techn. *Thomas Horn*,
Doz. Dr.-Ing. *Wilhelm Leupold*
Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden
Mommsenstr. 13
Dresden
DDR – 8027

Hinweise für Autoren

Herausgeber und Verlag danken den Lesern für das Interesse an den »Kleinstrechner-TIPS«, das sich in zahlreichen Zuschriften und Veröffentlichungsangeboten äußert. Beim Einsenden von Artikeln bitten sie folgende Hinweise zu beachten:

- In den »Kleinstrechner-TIPS« werden Artikel aus den auf der 4. Umschlagseite angegebenen Gebieten veröffentlicht.
- Manuskripte sind zweizeilig mit schwarzem Farbband mit Schreibmaschine zu schreiben, ä, ö und ü dürfen nicht durch ae, oe und ue ersetzt werden.
- Bilder sollten auf getrennten Blättern gezeichnet sein. Eine Bildunterschriftenliste ist beizufügen.
- Rechnerausdrucke (z.B. Programme) sollen tiefschwarz mit guter Ausnutzung des Formats (engzeilig, kein zu breites Kommentarfeld usw.) gedruckt sein.
- Die Autorenangabe soll enthalten: akadem. Grad, Vorname, Name, Tätigkeit, Arbeitsstelle, Privatanschrift.

Manuskripte mit einem Umfang von nicht mehr als 15 Schreibmaschinenseiten (einschließlich Bilder und Programme) sind in zwei Exemplaren an einen der Herausgeber zu senden (Anschrift s. S. 56).

Vorschau auf die nächsten Hefte

Hamm: LDIR – ein Weg zur Einstimmung auf die Programmierung in Maschinensprache

Finck/Berndt: Nutzung von Kleincomputern in der Psychologieausbildung

Pöschel: Römische Zahlen mit dem KC 85/1

Sonnemann: Schnell bewegte Grafik – Computeranimation

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00581-9

Beilage zu

Kleinstrechner-TIPS, Heft 12

Bitte ausheften und aufklappen!

Innenseite: *Bogatz*, Vollgraphik-Bildschirmansteuerung (Bild 4)

Außenseiten (4, 5 und 8): *Schönfelder*, Einsatz graphischer Bildschirmsysteme (Befehle des Graphic-BASIC)

********** BEFEHLE DES GRAPHIC-BASIC **********

VERSION 1.x 06.01.87

A U S Z U G

PUSH X Ablegen des Variablenwertes von X im Stack

POP = Pseudovariable
bringt naechsten Wert vom Stack

STK aktueller Stackpointerstand (relativ)

STACK X setzen des Stackpointer auf die relative Position X
(X = 0...127)

SIN(X) 128*SIN(X) X in Grad

COS(X) 128*COS(X) X in Grad

GDP Basisadresse des Graphicprozessors

DMA X,Y bringt Wert des Punktes der Position X,Y

---- Befehle fuer Graphic-Prozessor ----------

ENTER abspeichern des Endpunktes einer Linie im
Stack: (Reihenfolge --> Y --> X)

PSET X,Y Setzen des Punktes X,Y

PRES X,Y Loeschen des Punktes X,Y

LINE X1,Y1,X2,Y2
Linie von P1 (X1,Y1) nach P2 (X2,Y2)

LINETO Z,X1,Y1
Linie vom letzten Endpunkt nach P1
Z = Za + Zb
Za = 0 Pen
= 1 Eraser
Zb = 0,2,4-6 Linientyp

LINEON Z,X1,Y1
wie LINETO **,aber** bis zur ersten unterwegs
schneidenden Linie

```
CLR          Loescht aktuelles Bild

TYPE  X      Linientyp  0...3
              ( Achtung : geloescht durch LINETO und LINEON )

BILD  X      Bildauswahl
             Es existieren zwei unabhaengige Bilder, welche
             getrennt in RD und WR gesteuert werden koennen
              X  = 0...15
                              Bild 1       Bild 2

                   RD         Bit 0        Bit 1
                   WR         Bit 3        Bit 2

CURSOR        Cursorfunktion des GDP
                CURSOR 0        CU auf 0,0 setzen
                CURSOR 1        akt. CU nach PFILE (letzter
                                Endpunkt einer Linie )
                CURSOR 2,X,Y    CU und PFILE auf X,Y setzen

AREA X1,Y1,X2,Y2,X3,Y3 [,X4,Y4]
             Zeichnen eines Dreieck [Viereck] mit den
             Eckpunkten P1...P3 [P4]

PENUP        Cursor zeichnet nicht

PENDOWN      Cursor zeichnet

POLYSET  X,Y,Adr
             Setzen des Polygonzuges aus dem File (Adr) auf
             den Startpunkt X,Y
             Fileaufbau (im Speicher)
               WP1,WP2,WP3.......WPn      WP = Wertpaare
                WP1 =  80H       Ende des File
                WP1 >  80H       Kurzvektoren
                WP1 <  80H       dann folgen WP1b und WP1c mit
                                 der Bedeutung :
                                 WP1 = x-Laenge   (0...7FH)
                                 WP1b= y-Laenge   (0...FFH)
                                 WP1c= Richtung   (0...7  )

POLYMOVE  X,Y,Adr,dX,dY
             Verschieben eines Polygonzuges vom Ort X,Y um
             dX und dY

CIRCLE   X0,Y0,a,b [,Tanf,Tend,Fo,WS,SW]
             Zeichnen einer Ellipse (alle Winkel in Grad)
               X0,Y0          Mittelpunktkoordinaten
               a ,b           Radien
               Tanf           Anfangswinkel          (Standard 0 )
               Tend           Endwinkel              ( ="=    360)
               Fo             Verdrehungswinkel des Koordinaten-
                              systems der Ellipse zum Bildschirm
                                                     (Standard 0 )
               WS             Winkelschrittweite bei Rechnung
                                                     (Standadr 1 )
               SW             Startwinkel (Offset zu Tanf)
                                                     (Standard 0 )
```

Fehlende Parameter vom Ende her koennen entfallen.
Fuer nicht genutzte Werte muss nur das Komma
gesetzt werden.
Bsp.: CIRCLE 50,50,30,20,,,30

CIRCLESEG ...(wie CIRCLE).....
Zeichnen eines Ellipsensegmentes

FILL X,Y
Fuellen eines umrandeten Gebietes ausgehend von
Punkt X,Y

SCAN Loeschen Bildschirm
(Im Gegensatz zu CLR kann das Bild mit INV SCAN
auf schwarz gestellt werden)

INV Vorsatzwort fuer Befehle (= invers)
Es erfolgt eine Umschaltung von schreiben (PEN)
auf loeschen (ERASER).Auf schwarzem Grund entstehen

weisse Linien.
Einsetzbar fuer folgende Kommandos :

LINE	AREA
POLYSET	SCAN
FILL	LIGHTPEN
CIRCLE	CIRCLESEG

INV kann mit oder ohne Leerzeichen dem Schluesselwort vorangestellt werden .

LIGHTPEN X,Y
Ruft Lichtstift und gibt Koordinate in Variable
X und Y zurueck
INV LIGHTPEN bringt den gleichen Effekt , arbeitet
aber im Polling ohne auf den Abschluss der Sequenz
zu warten (Lichtstift nicht aktiv : X=Y= -1)

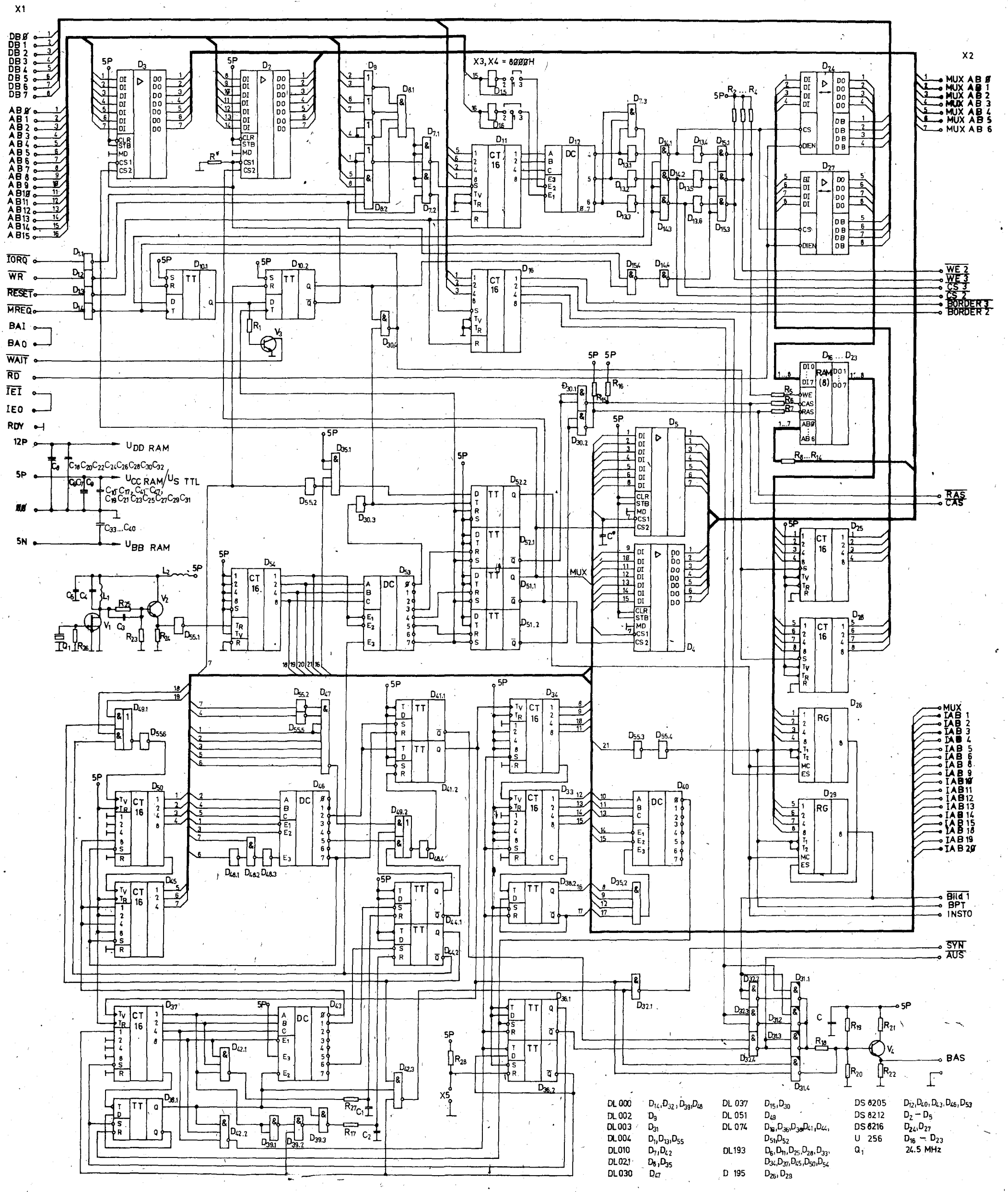
WRITE wie PRINT auf Graphic-Bild

LWRITE wie LPRINT auf Graphic-Bild

PUNKT X Zahl der Zwischenpunkte einer Linie (im Stack)
X = 0...16

ASGN X1,X2,X3.....
Geraetezuweisung

Xn = 0	Init Drucker
1	Drucker ein (am Zeilenende aus)
2	Formfeed bei COPY ein
3	Formfeed bei COPY aus
4	WRITE = PRINT
5	WRITE => Graphic-Bildschirm
6	Highspeed GDP ein
7	="= aus
8	Init GDP (X=Y=0 ; CLR ; PEN ; DOWN)



DL 000	$D_{14}, D_{32}, D_{39}, D_{48}$	DL 037	D_{15}, D_{30}	DS 8205	$D_{12}, D_{40}, D_{43}, D_{46}, D_{53}$
DL 002	D_9	DL 051	D_{49}	DS 8212	$D_2 - D_5$
DL 003	D_{31}	DL 074	$D_{18}, D_{36}, D_{38}, D_{41}, D_{44}, D_{51}, D_{52}$	DS 8216	D_{24}, D_{27}
DL 004	D_1, D_{13}, D_{55}			U 256	$D_{16} - D_{23}$
DL 010	D_7, D_{42}	DL 193	$D_6, D_{11}, D_{25}, D_{28}, D_{33}, D_{34}, D_{37}, D_{45}, D_{50}, D_{54}$	Q_1	24.5 MHz
DL 021	D_8, D_{35}				
DL 030	D_{47}	D 195	D_{26}, D_{29}		

Bogatz: Vollgraphik-Bildschirmansteuerung

Bild 4. Praktische Realisierung der Vollgraphik-Bildschirmansteuerung, bestehend aus den Teilbaugruppen:

- Video-Controller mit Quarzgenerator (V1, V2, Q1, L1), Bildpunktzähler (D54), Horizontalzähler (D50, D45, D46, D48), Vertikalzähler (D33, D34, D38.2), Horizontalaustast- und -synchronimpulsbereitstellung (D41.1, D41.2), Vertikalaustast- und -synchronimpulsbereitstellung (D37, D38.1, D36.1), Halbbildzähler für Zeilensprung (D36.2) und BAS-Mischer (D31, D32)
- Bildwiederholspeicher mit 16k × 8 Bit RAM (D16 bis D23), Adreßmultiplexer (D2 ··· D5), RAS/MUX/CAS-Steuerung (D51.2, D51.1, D52.1), Bildpunktschieberegister (D26, D29), Prozessor-Zugriffssteuerung (D10, V3) und Datenpuffer/Datentreiber (D25, D28, D24, D27)
- Kartenverwaltung mit Adreßdekoder (D7 ··· D9), Kartenverwaltung (D11, X3, X4) und Randverwaltung (D16).

Die gesamte hier dargestellte Schaltung findet auf einer Zweiebenen-Leiterplatte der Größe 215 × 170 mm² Platz und ist anschlußkompatibel zum System K1520.