GRAF®
computer

GDP64HS

Die monochrome Standard-graphik- Karte für den

NDR- Computer

und

mc- Computer

Ausgabe 3

Graf Elektronik Systeme GmbH
8960 Kempten · Tel.: 08 31 - 62 11

Inhalt

1. Einführung

1.1 Zum NDR-Computer

Der NDR-Computer wird in der Fernsehserie "Mikroelektronik - Mikrocomputer selbstgebaut und programmiert " aufgebaut, erklärt und in Betrieb genommen. Diese Serie wird vom Norddeutschen Rundfunk und vom Bayerischen Fernsehen ausgestrahlt. Es werden bald auch die Regionalsender anderer Bundesländer die Sendung in ihr Programm aufnehmen.

Zur Serie gibt es einige Begleitmaterialien, es ist daher nicht unbedingt notwendig, die Fernsehserie gesehen zu haben, um den NDR-Computer zu bauen und zu begreifen:

- Bücher:

  Rolf-Dieter Klein,
  "Rechner modular"
  ISBN 3-7723-8721-7, DM 68,-
  erschienen im Franzis-Verlag, München
  Bestellnummer: 10991

  Rolf-Dieter Klein,
  "Die Prozessoren 68000 und 68008"
  Rechnerarchitektur und Sprache im NDR-KLEIN-Computer
  ISBN 3-7723-7651-7, DM 78.-
  erschienen im Franzis-Verlag, München

- Zeitschriften "mc" und "ELO" des Franzis-Verlages

- Zeitschrift "LOOP" der Firma Graf Elektronik Systeme

- Videocassetten:
  lizensierte Originalcassetten für den privaten Gebrauch. Auf diesen zwei Cassetten sind die 26 Folgen der Fernsehserie enthalten.
  Systeme: VHS, Beta, Video 2000
  Preise: siehe gültige Preisliste

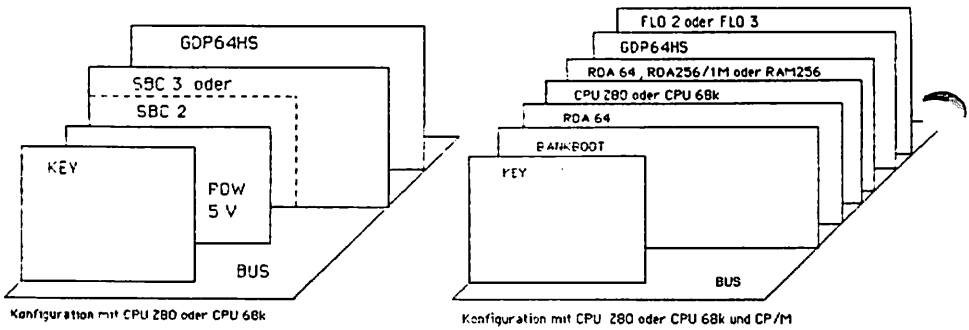
1.2 Wozu dient die Baugruppe

Die Baugruppe GDP64HS ist das Bindeglied zwischen dem Mikrocomputer (SBC 2, SBC 3, CPU 680xx oder CPU Z80) und einem Monitor. Sie ermöglicht es einen Monitor mit BAS - Anschluß oder einen TTL- Monitor anzuschliessen. Nun ist es möglich Arbeitsschritte, die der Computer durchführt, auf dem Monitor anzuzeigen, Graphiken darzustellen oder Einblick in das Innenleben des Computers zu bekommen (Speicherbelegung, Kontrolle der Eingaben). Die GDP64HS kann zudem noch sogenannte Hardcopys erstellen, d.h. der Bildschirm wird mit Hilfe eines Druckers auf ein Blatt Papier kopiert.

Da jeder Bildpunkt auf dem Monitor ansprechbar sein muß, wird bei einer Bildebene von 256 x 512 Bildpunkten ein eigener Speicher von 16 KByte benötigt, wenn jeder Bildpunkt ein Bit beansprucht. Da aber vier unabhängige Bildebenen aufgebaut werden können, braucht man demnach einen Speicherplatz von 64 KByte. Dieser ist in 8 x 64 KBit Speichern organisiert.

In diesem Speicher wird jeweils das gesamte Bild abgespeichert und seriell alle 20 ms abgerufen (50 mal in der Sekunde); dadurch entsteht ein stehendes Bild. Die Verwaltung des Speicherbereiches (Abruf des Bildes , Refresh...) übernimmmt der auf der GDP64HS befindliche Graphik-Prozessor EF 9366. Mit dem Mikrocomputer können per Datenbus Befehle übermittelt werden, z.B. Schreiben eines Zeichens, Größe des gewünschten Zeichens, Form des Zeichens, Lage und Position des Zeichens auf der Bildebene, Auswahl einer der vier Bildebenen . Dieser Prozessor ermöglicht es auch schnelle Graphik darzustellen (Blockgraphik und Vektoren). Durch Definition von verschiedenen Vektoren ist es möglich, Linien (Vektoren) in jede Richtung und in jeder Größe zu zeichnen .

Verschiedene Konfigurationen mit der GDP64HS (Bild 1):



2. Technische Daten

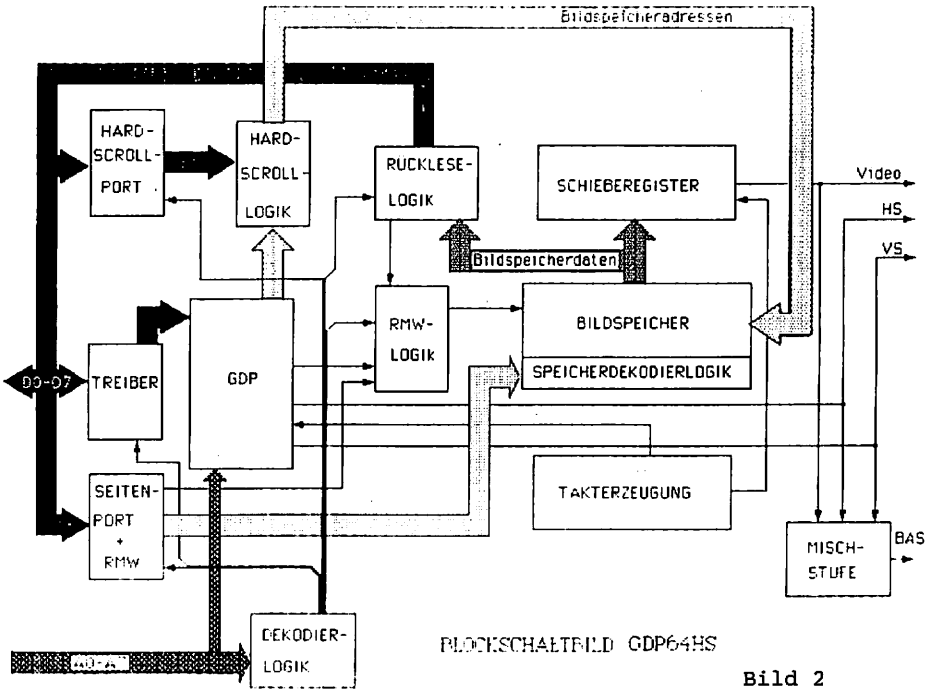
Spannungsversorgung:	+5V
Stromaufnahme:	500 mA
Busformat:	NDR - Bus 54-polig ECB - Bus 64-polig
Leiterplattenformat:	160mm x 100mm (Europakarte)
Ausgang:	1.BAS (beinhaltet HS,VS und VIDEO - Signal) und 2.TTL - Ausgang:HS,VS,VIDEO (invertierbar) z.B. IBM Monitor
Graphik - Controller:	EF 9366 (Thomson-CSF) -kann 4 Seiten bedienen, wobei in eine geschrieben und zugleich eine weitere gelesen werden kann. -integrierter ASCII- Zeichensatz -Graphikbefehle * Kurzvektoren * Vektoren * Blockgraphik 5x8 und 4x4
Speicher:	8 x 64k RAM (dynamisch)
sonstige Funktionen:	Read Modify Write (zerstörungsfreies Zeichnen auf dem Bildschirm)

- Hardcopy (Rücklesen des Bildschirminhaltes und Ausgabe auf einen Drucker)

- Hardscroll (Scrollen des Bildschirms mit Hilfe der Hardware, d.h. die Adressen des Bildschirmspeichers werden hardwaremäßig aufaddiert.

3. Prinzipbeschreibung

3.1 Blockschaltbild



BLOCKSCHALTBILD GDP64HS

Bild 2

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

Der interne Aufbau des Speichers wird durch den Grafikprozessor und sekundär durch die Speicherdekodierlogik organisiert. Im Speicher steht dann die Information, die später auf dem Bildschirm erscheint.

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

Das Byte steht am Ausgang des Speichers und wird bei aktivieren des Signals SH/L (Shift Load, am Schieberegister) parallel in das Schieberegister eingelesen. Hier wird das Signal mit dem Punktetakt (14 MHz, CLK) verknüpft und seriell (in der Punktfolge 10001110) an den Monitor hinausgeschoben. Da jedes Bit einen Bildpunkt darstellt, werden jetzt 8 Punkte auf dem Bildschirm angezeigt. Ein dunkler Punkt entspricht einer 1 und ein heller einer 0. (ebenfalls in der Reihenfolge 10001110).
Siehe Bild 3 auf der nächsten Seite.

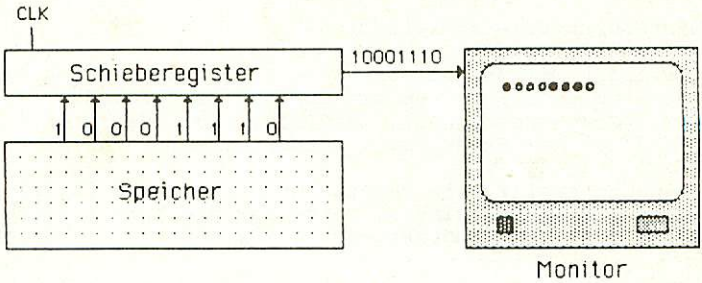


Bild 3

Der Speicher ist für 4 Seiten aufgebaut, die durch die Seitenumschaltung ausgewählt werden können. Es kann in eine Seite geschrieben und zugleich eine weitere gelesen werden. Das aus dem Schieberegister kommende Videosignal geht entweder direkt zum Monitor (TTL) oder zum Videomischer. In dieser Mischstufe wird das Video-Signal mit horizontalen und vertikalen Synchronisationssignalen (HS und VS) so aufbereitet, daß es danach als BAS-Signal (siehe 3.2.2) zur Verfügung steht.

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

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

Mit Hilfe des RMW-Modus kann man einfach bewegte Bilder erzeugen, da bei Schreibvorgängen in den Bildspeicher (=Monitor) alle Punkte komplementiert werden. Wenn z.B. ein Punkt gesetzt war, so wird er gelöscht, wenn er nicht gesetzt war, so wird er eingeschrieben. Der größte Vorteil des RMW-Modus ist, daß zerstörungsfrei gezeichnet werden kann. Dies ist z.B. nötig, um einen Maus-Zeiger oder ein Fadenkreuz auf einer Graphik oder einem Bild darzustellen,

ohne das Hintergrundbild zu zerstören.

Die Dekodierlogik mit den Adressen A4...A7, den Signalen IORQ und M1 wird benötigt, um die GDP64HS bei I/O- Zugriff von 70...7F (Graphikprozessor) und 60...6F (Seitenport, RMW=Mode, Hardcopy und Hardscroll) anzusprechen.

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

3.2 Datenübertragung zum Monitor:

3.2.1 Prinzip der Signale HS,VS und VIDEO.

HS- Signal: (Horizontal- Synchronisation) Dieses Signal ist für die Zeilensynchronisation zuständig. Der Bildschirm wird veranlaßt, mit dem Schreiben einer Zeile so lange zu warten, bis die zu Übertragende Information am VIDEO- Ausgang bereitgestellt ist. Wie der Name des Signales schon andeutet, wird das Übertragen der Daten und das Schreiben der Daten auf den Bildschirm synchronisiert.

VS- Signal: (Vertikal- Synchronisation) Dieses Synchronisationssignal veranlaßt einen neuen Bildschirmaufbau, der alle 20 ms stattfindet. Der Schreibstrahl des Bildschirms fährt also von der rechten unteren Ecke in die linke Obere und wird in dieser Zeit ausgeblendet. Während dieses Vorganges kann keine Information geschrieben (also nichts auf dem Bildschirm dargestellt) werden.

VIDEO- Signal: Es besteht aus High- und Low- Signalen die die Bildinformation wiederspiegeln. Will man dieses Signal ansehen, verwendet man am besten ein Oszilloskop.

3.2.2 Monitor mit BAS - Signal:

Das BAS - Signal wird bei normalen Monitoren über eine einzige Leitung übertragen und setzt sich aus den Einzelsignalen HS + VS + Video zusammen. Die Signale HS(bzw VS) und das Videosignal stehen im Verhältnis 1 zu 2.

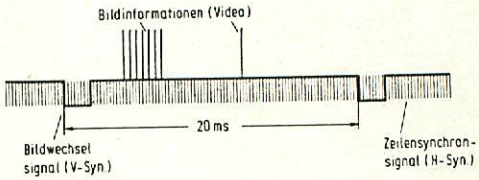


Bild 4

3.2.3 Monitor mit TTL-Signal z.B. IBM - Monitor:

Hier werden die Signale HS, VS und Video über verschiedene Leitungen zum Monitor übertragen. (siehe rechts, prinzipieller Verlauf der Signale ohne Angabe der Zeiten).

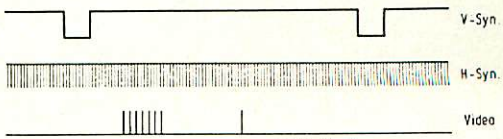


Bild 5

Beispiel: Eine Originalaufnahme der drei Signale HS, VS und Video mit den Logik-Analyser erstellt:

Im ersten Diagramm erkennt man das VS- Signal, darunter das Signal HS und das VIDEO-Signal(VI). Wie daraus zu ersehen ist, wird während des HS-Signales keine Bildinformation geschrieben. Wird der HS- Impuls aktiv, wird in der nächsten Zeile weitergeschrieben. Das VS-Signal leitet den Aufbau eines neuen Bildes ein.

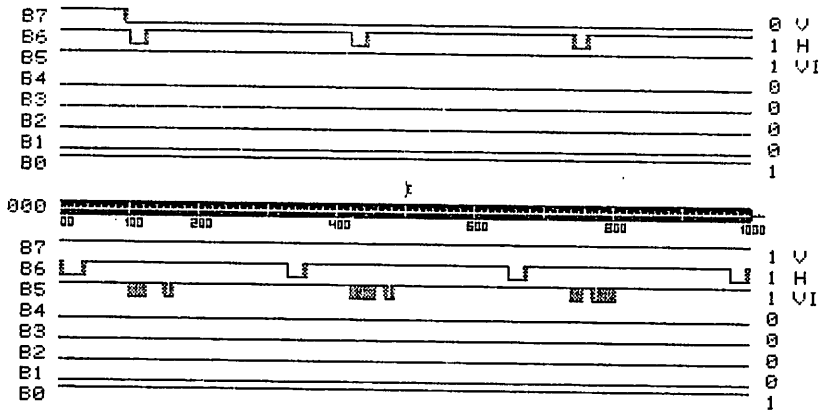


Bild 6

3.2.4 Wie funktioniert ein Monitor ?

Der Monitor (Brownsche Röhre) besteht aus einer Kathode einer Fokussiereinrichtung (Wehnelt- Zylinder und Anode), Ablenkplatten für horizontale und vertikale Ablenkung und einer auf der Röhreninnenseite aufgetragenen Leuchtschicht.

Von der Kathode werden Elektronen ausgesendet, die von der Fokussiereinrichtung gebündelt werden. Die Ablenkplatten sorgen in horizontaler sowie in vertikaler Richtung für die nötige Ablenkung des Elektronenstrahles, damit jeder Punkt des Bildschirms erreicht wird.

Trifft der Elektronenstrahl an der Frontseite des Bildschirms auf (auf die Leuchtschicht) so beginnt der angestrahlte Punkt zu Leuchten. Der ausgesandte Elektronenstrahl muß sehr scharf gebündelt sein, um eine hohe Auflösung zu erreichen.

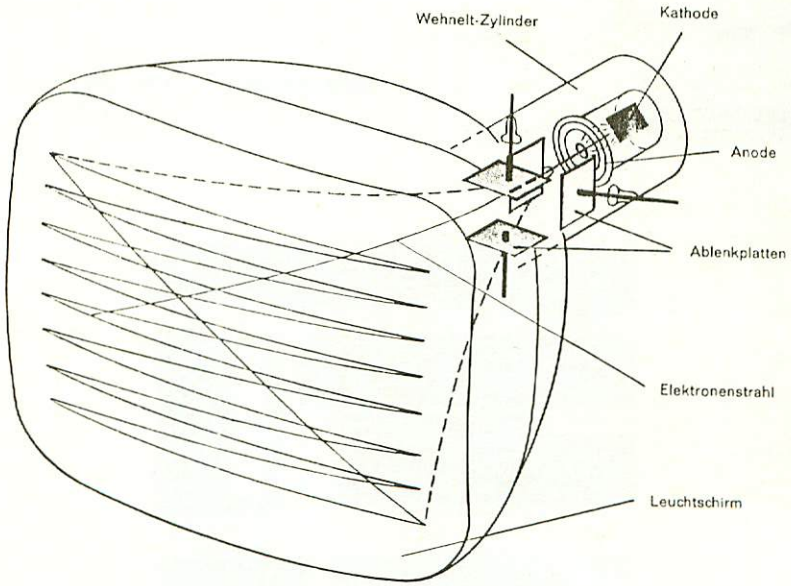


Bild 7

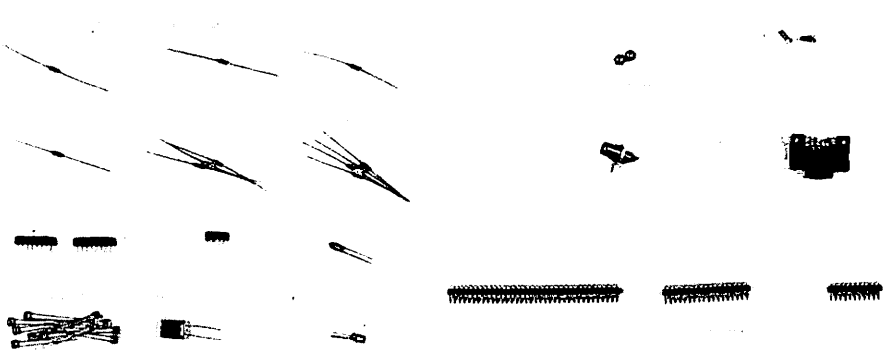
4. Aufbauanleitung

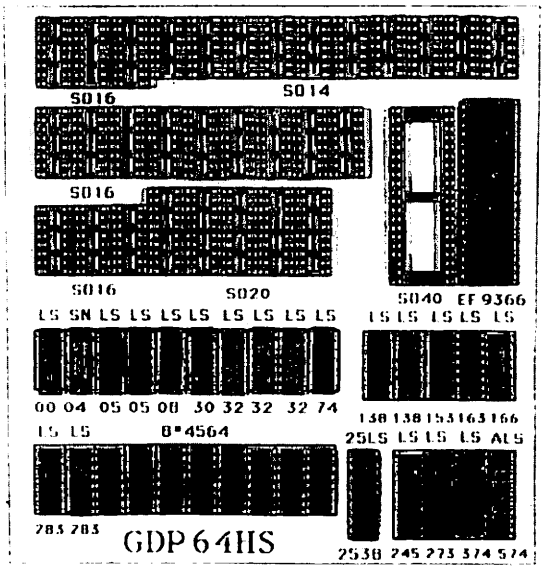
CMOS-Warnung:

CMOS-Bausteine sind hochempfindlich gegen elektrostatische Aufladung! Bewahren oder transportieren Sie CMOS-Bausteine nur auf dem leitenden Schaumstoff! Alle Pins müssen kurzgeschlossen sein.

Tip: Fassen Sie an ein geerdetes Teil (z.B. Heizung, Wasserleitung) bevor Sie einen Baustein berühren.

Bitte beachten Sie hierzu auch den Artikel "Schutzmaßnahmen für MOS-Schaltungen" in unserer Zeitschrift LOOP3.





4.1 Stückliste des Komplettbausatzes GDP64HS

An-zahl	Art.-Nr.	Position im Plan	Bezeichnung	Bemerkungen
1	11229			Leiterplatte GDP64HS
1	11232			Handbuch
1	60082	J5	74 LS 08	
2	60080	J1,J31	74 LS 05	Sechs Inverter
1	60104	J13	74 LS 166	8-Bit Schieberegister
1	60101	J7	74 LS 163	synch. 4-Bit Zähler
1	60075	J8	74 LS 00	4 NAND Gatter
1	60033	J11	7404	6 Inverter
1	60014	J16..J23	4164, 200ns	Dynam. RAM 64 kBit
1	60137	J6	74 LS 74	D-Flip Flop mit Preset
3	60121	J3,J4,J10	74 LS 32	Vier OR-Gatter
1	10806	J9	25 LS 2538	3 zu 8 Decoder
1	60098	J12	74 LS 153	4 zu 1 Multiplexer
2	60094	J29,J30	74 LS 138	3 Bit Binärdekoder
1	60115	J27	74 LS 245	8 fach Bus Transreceiver
1	10806	J28	EF 9366	Graphik-Processor
1	60118	J24	74 LS 273	8 Bit D-Register
1	60120	J2	74 LS 30	8-fach NAND
2	60119	J14,J15	74 LS 283	4 Bit Volladdierer
1	60126	J25	74 LS 374	8 Bit Datenlatch
1	61126	J26	74 ALS 574	8 Bit Datenlatch
10	60183	SO 14	14-polige IC-Fassung	
15	60185	SO 16	16-polige IC-Fassung	
5	60187	SO 20	20-polige IC-Fassung	
1	60193	SO 40	40-polige IC-Fassung	
1	60665	R1	75	Widerstand 75 Ohm
1	60621	R5	150	Widerstand 150 Ohm
1	60631	R10	220	Widerstand 220 Ohm
1	60643	R8	330	Widerstand 330 Ohm
3	60651	R3,R4..R11	470	Widerstand 470 Ohm
4	60626	R2,R6,R7,R9	1 K	Widerstand 1000 Ohm
2	60518	RN1,RN2	8X3,3,K	Netzwerkwiderstand
1	60958	RN3	4x470 Ohm	Netzwerkwiderstand
1	60248	C7	10 uF,Tantal ELKO	auf Polung achten !
11	60239	C1...C6, C8...C12	100 nF	Keramikkondensator
1	60590	T1	BC 107	Transistor
1	60166	Q1	Quarz 14.00 MHz	
1	60499	ST1	7-polige Stiftleiste gewinkelt	
1	60725	ST3	9-polige D-Sub-Buchse	
1	10787	ST2	ECB-Bus 64-polig oder	
1	10405	ST4	NDR - Bus 18-polig, gewinkelt und	
1	10406	ST4	NDR - Bus 36-polig, gewinkelt	
1	10097	BU1	Monitor Buchse Cinch einlötbar	

4.2 Stückliste des Aufbausatzes GDP64k-GDP64HS

An-zahl	Art.-Nr.	Position im Plan	Bezeichnung	Bemerkungen
1	11229			Leiterplatte GDP64HS
1	11232			Handbuch
1	60082	J5	74 LS 08	4 AND-Gatter
2	60121	J3,J4,J10	74 LS 32	Vier OR-Gatter
1	60094	J29,J30	74 LS 138	3 Bit Binärdekoder
1	60120	J2	74 LS 30	8-fach NAND
2	60119	J14,J15	74 LS 283	4 Bit Volladdierer
1	60126	J25	74 LS 374	8 Bit Datenlatch
1	61126	J26	74 ALS 574	8 Bit Datenlatch
10	60183	SO 14	14-polige IC-Fassung	
15	60185	SO 16	16-polige IC-Fassung	
5	60187	SO 20	20-polige IC-Fassung	
1	60193	SO 40	40-polige IC-Fassung	
1	60665	R1	75	Widerstand 75 Ohm
1	60621	R5	150	Widerstand 150 Ohm
1	60631	R10	220	Widerstand 220 Ohm
1	60643	R8	330	Widerstand 330 Ohm
3	60651	R3,R4..R11	470	Widerstand 470 Ohm
4	60626	R2,R6,R7,R9	1 K	Widerstand 1000 Ohm
2	60518	RN1,RN2	8X3,3,K	Netzwerkwiderstand
1	60958	RN3	4x470 Ohm	Netzwerkwiderstand
1	60248	C7	10 uF,Tantal ELKO	auf Polung achten !
11	60239	C1...C6, C8...C12	100 nF	Keramikkondensator
1	60590	T1	BC 107	Transistor
1	60166	Q1	Quarz 14.00 MHz	
1	60499	ST1	7-polige Stiftleiste gewinkelt	
1	60725	ST3	9-polige D-Sub-Buchse	
1	10787	ST2	ECB-Bus 64-polig oder	
1	10405	ST4	NDR - Bus 18-polig, gewinkelt und	
1	10406	ST4	NDR - Bus 36-polig, gewinkelt	
1	10097	BU1	Monitor Buchse Cinch einlötbar	

4.3 Aufbau Schritt für Schritt des GDP64HS Komplettbausatzes bzw. des Aufbausatzes

Auf einer Seite der Leiterplatte steht der Hinweis "Lötseite"; auf dieser Seite wird ausschließlich gelötet. Die Bauteile sind nur auf der anderen Seite (der Bestückungsseite) aufzustecken.

Beim Einlöten der Bauelemente beginnt man am besten mit den ganz flachen Bauelementen. Bevor Sie jedoch beginnen sollten Sie sich Ihren Bestückungsdruck genau ansehen und die Bauelemente mit der Stückliste vergleichen. Eventuell fehlende Bauteile sollten Sie sofort reklamieren.

Demnach sollten Sie alle liegenden Widerstände zuerst bestücken. Dies sind die Widerstände R1 bis R11. Diese Widerstände sind durch Farbcodes zu indentifizieren:

Widerstand	Widerstandswert	Farbcode
R1	75 Ohm	violett-grün-schwarz
R2, R6, R7, R9	1 kOhm	braun-schwarz-rot
R3, R4, R11	470 Ohm	gelb-violett-braun
R5	150 Ohm	braun-grün-braun
R8	330 Ohm	orange-orange-braun
R10	220 Ohm	rot-rot-braun

Der vierte Strich kennzeichnet die Toleranz und bei diesen Widerständen immer "gold".

Gehen Sie beim Einlöten der Widerstände folgendermaßen vor: Anschlußdrähte der Widerstände rechtwinkelig nach unten biegen und in den entsprechenden Platz auf der Leiterplatte stecken. Achten Sie bitte darauf daß die Widerstände auf der Leiterplatte aufliegen. Auf der Lötseite sollten Sie die überstehenden Enden leicht abbiegen und dann mit einem Seitenschneider abschneiden und verlöten.

Wenn Sie die Baugruppe für den NDR-Computer aufbauen, sollten Sie jetzt die 54-polige abgewinkelte Stiftleiste bestücken. Die Stiftleiste wird als 18-polige und als 36-polige Stiftreihe geliefert. Beim Bestücken sollten Sie folgendermaßen vorgehen: Beide Stiftleisten mit dem abgewinkelten Ende in die vorgesehenen Bohrungen (ST4) stecken; dann die Leiterplatte umdrehen und 4 bis 5 Punkte (an den Enden und in der Mitte einige) verlöten. Jetzt sollten Sie erst auf der Bestückungsseite kontrollieren, ob die geraden Stifte der Steckerleiste parallel zur Leiterplatte liegen und ob sich zwischen den Lötpunkten "Bäuche" gebildet haben. Sollte einer dieser beiden Defekte vorliegen können Sie dies jetzt noch problemlos beheben und dann die restlichen Pins verlöten.

Ist die Steckerleiste bestückt, kommt die arbeitsintensivste Bestückung: Die IC-Sockel. Bei den IC-Sockel gibt es eigentlich nur eines zu berücksichtigen und das ist die richtige Polarisation der Sockel. Jeder Sockel ist mit einer Kerbe versehen, die Pin 1 markiert. Auf dem Bestückungsdruck ist ebenfalls eine Kerbe bei jedem IC-Sockel aufgedruckt. Diese beiden Kerben müssen übereinstimmen (siehe Abb.).

Stecken Sie nun alle IC-Sockel auf. Achten Sie darauf, daß Sie nicht einen 14-poligen in den Platz eines 16-poligen stecken usw. Sind alle Sockel aufgesteckt, ist es hilfreich wenn Sie nach folgender Beschreibung vorgehen:

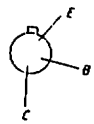
Legen Sie nun eine feste Pappe, ein Stück Holz oder etwas ähnliches auf alle Bauelemente und drehen dies unter festem andrücken der Pappe mit der Leiterplatte herum und löten von jedem IC-Sockel zwei diagonal gegenüberliegende Pins an. Bevor Sie die restlichen Pins verlöten, sollten Sie auf der Bestückungsseite noch einmal kontrollieren, ob alle Sockel richtig gesteckt worden sind und ob alle auf der Leiterplatte aufliegen.

Ein späteres Ändern macht meistens vielmehr Mühe und führt bei ungeübten Aufbauern oftmals zur Zerstörung der Leiterplatte (Leiterbahnen werden abgerissen oder Durchkontaktierungen zerstört usw.).

Die Keramikkondensatoren C1 bis C6 und C8 bis C12 sind ungepolt; sie brauchen hier also nicht auf die Polung zu achten.

Beim Einlöten des Tantalkondensators C7 achten Sie bitte auf richtige Polung. Das "+" auf dem Kondensator muß mit dem "+" auf dem Bestückungsdruck übereinstimmen.

Beim Transistor muß auf die Anschlüsse E,B,C geachtet werden. Der Transistor hat an seinem Umfang eine "Nase". Der PIN, der dieser Nase am nächsten kommt, ist der Emitter (siehe Abb.)



Transistor von unten gesehen Bestückungsdruck

Der Transistor sollte nicht sehr tief hineingesteckt werden, da sonst die Hitze des Lötkolbens ihn zerstören könnte.

Zum Schluß werden die Buchsen und der Quarz bestückt. Beim Einlöten der BAS-Buchse BU1 ist darauf zu achten, daß diese ganz auf der Leiterplatte aufliegt. Die Buchse sollte fest aufliegen, dann erst kann sie eingelötet werden.

Der Schwingquarz ist nicht gepolt und kann somit nicht falsch herum eingelötet werden. Der Quarz kann, wenn er stehend stört auch nach folgender Abb. gelegt werden. Allerdings müssen Sie dann beim Bestücken des Haltewinkels den Quarz so legen, daß die Schraube für diesen noch Platz findet. Das Gehäuse des Quarzes darf Kontakt mit der Schraube bzw. mit Masse haben.

Abb. Einlöten des Quarzes

Die Stecker ST1 und ST3, sowie die Jumper JMP1 bis JMP5 werden nicht bestückt. Die Jumper sind auf der Lötseite der Leiterplatte voreingestellt.

Sollten Sie die GDP64HS für den ECB-Bus aufgebaut haben, müssen Sie jetzt noch die 64-polige Messerleiste (ST1) bestücken, und die in folgender Abbildung durch Pfeil gekennzeichnete Brücken auf der Lötseite schließen.

Rückwandblech:

Wollen Sie diese Baugruppe in das Gehäuse GEH3 einbauen, so wird ein passendes Rückwandblech benötigt. (Bitte gesondert bestellen) #11211

5. Testanleitung

5.1 Erste Prüfung ohne IC's
Komplettbausatz bzw. Aufbausatz

Die Leiterplatte ist bis jetzt erst mit den Sockeln und mit den passiven Bauelementen bestückt. Mit diesem Aufbau wird der erste Test durchgeführt.

Zu diesem Test muß die Baugruppe in den Bus gesteckt werden. Achten Sie beim Einstecken in den Bus, daß Sie die Baugruppe richtig herum einsetzen. Ein falsches Einstecken, z.B. um ein Pin zu weit rechts kann zu Kurzschlüssen führen und kann Bauelemente zerstören.

Nach dem Einstecken der Leiterplatte muß der Rechner weiter problemlos funktionieren. Falls nein - weiter im Kapitel 6.

Man mißt, ob an allen IC-Sockeln die Versorgungsspannung von +5V ankommt. Dabei liegt bei Standard-TTL-Bausteinen jeweils am letzten Pin einer Fassung (z.B. bei 14-poligen an Pin 14, bei 16-poligen an Pin 16, bei 20-poligen an Pin 20), die Versorgungspannung von +5V. 0V bzw. Masse liegt jeweils auf dem letzten Pin der ersten Reihe (bei 14-poligen auf Pin 7, bei 16-poligen auf Pin 8, bei 20-poligen auf Pin 10). Achtung!: Bei den RAM-Bausteinen sind die Plus- und Masse-Anschlüsse genau andersrum (+5V liegt an Pin 8 und GND liegt an Pin 16). Beim Graphik-Prozessor EF 9366 liegt Masse auf Pin 20 und +5V auf Pin 40 (siehe auch Kapitel 9).

Liegt die Versorgungsspannung +5V und 0V (Masse,GND) an den richtigen Pins an, dann können die IC's eingesetzt werden. Dabei muß auf die Richtung der IC's geachtet werden. Die Markierung auf dem IC muß mit der Kerbe in der Fassung übereinstimmen.

5.2 Test der GDP64HS im System

5.2.1 Bestücken der ICs und erste Tests beim Komplettbausatz

Sie können jetzt alle ICs einstecken. Bitte achten Sie darauf, daß Sie die ICs richtig herum einstecken. Die Kerbe auf dem IC muß mit der Kerbe am Sockel bzw. auf dem Bestückungsdruck übereinstimmen. Kontrollieren Sie lieber doppelt, denn wenn ein IC falsch herum eingesteckt ist, ist es garantiert 'tot'.

Haben Sie ein Osszilloskop und wollen die Baugruppe Schritt für Schritt testen, so können Sie auch nach folgender Beschreibung vorgehen:

Zuerst wird nur das IC J11 eingesteckt (IC zur Takterzeugung). Wird die Leiterplatte nun auf den Bus gesteckt, muß an J11/8 eine Taktfrequenz von 14 MHz zu messen sein.
Wenn dieser Takt anliegt können die restlichen IC's bis auf den EF 9366 (J28) und die Speicherbausteine J16..J23 hineingesteckt werden; aber nicht bei angelegter Spannung !!!
Wird danach die Spannung wieder angelegt, muß an IC J28/1

ein 1,75 MHz Signal messbar sein.
An J7 müssen folgende Signale zu messen sein:

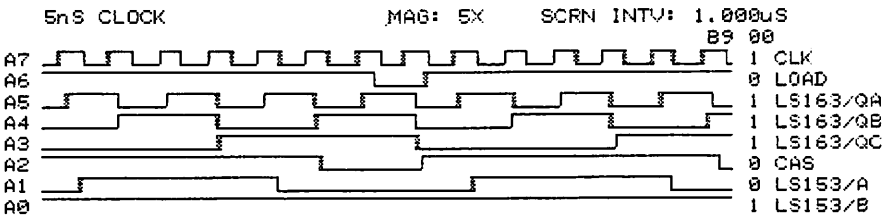


Bild 8

Anschließend muß die Versorgungsspannung wieder weggenommen und der EF 9366 eingesteckt werden. Beim Einschalten der Spannung muß auf dem Monitor ein abgegrenztes dunkles Bild erkennbar sein. Es ist noch keine Bildinformation erkennbar (auch nicht vorhanden). Nur das Synchronsignal, das der EF 9366 erzeugt, muß mit dem Oszilloskop am BAS - Ausgang zu messen sein. Wie das Signal aussieht, ist in Kapitel 3 unter 'Daten- übertragung zum Monitor' zu sehen.

Anschließend sollte die Spannungsversorgung der Speicherbausteine J16..J23 kontrolliert werden. Jeweils an Pin 8 müssen 5 Volt anliegen (Masse liegt an Pin 16). Nach Abschalten der Spannung können die Speicherbausteine eingesteckt werden.

5.2.2 Bestücken der ICs und erste Tests beim Aufbausatz

Da die Ausstattung des Aufbausatzes auf die Weiterverwendung von ICs der 'alten' GDP64k abgestimmt ist, sollten Sie diese nun bereithalten. Beim Entnehmen der benötigten Schaltkreise benützen Sie bitte einen Schraubendreher der Grösse zwei oder drei, bzw. ein Taschenmesser mit schmaler Klinge. Das Werkzeug wird nun vorsichtig von der Stirnseite der ICs her in den Spalt zwischen IC und Fassung eingeschoben. Durch die Keilform des Schraubendrehers wird der Schaltkreis jetzt im vorderen Bereich aus der Fassung angehoben. Durch leichte Kippbewegungen wird der Schaltkreis vollständig ausgehebelt. Eventuell verbogene IC-Beinchen werden mit einer Flachzange geradegerichtet. Bei Verwendung eines Taschenmessers können Sie ebenfalls Kippbewegungen durchführen, oder nachdem die Klinge ganz untergeschoben wurde das IC mit Drehbewegungen aus seiner Fassung lüften.
Beachten Sie hierbei jedoch die CMOS- Warnung!

Beim Umsetzen der ICs arbeiten Sie sich bitte IC für IC durch, d.h. hier Entnehmen und dort Einsetzen.
Besondere Erwähnung verdienen die ICs J1 und J31 (6-fach Inverter 7405): Im Schaltplan und im Bestückungsplan der GDP64HS werden hierfür LS- Typen vorgesehen. Alternativ können anstatt der LS- Typen aber auch Standardtypen eingesetzt werden, die auf der GDP64k bereits vorhanden sind. Aus diesem Grund sind dem Aufbausatz diese Bausteine nicht beigelegt worden.
Die sonstige Vorgehensweise entspricht der Testanleitung unter Kapitel 5.2.1.

5.2.3 Test im Z80-System

Sind nun alle Bausteine bestückt, kann der Test mit der Software beginnen. Setzen Sie in Ihrem Rechner nur das Grundprogramm ein, so reicht es, wenn Sie die SBC3 (bestückt mit EGRUND2), die KEY bzw. KEY2 und die GDP64HS im BUS stecken haben. Der Monitor wird mit dem Monitorkabel am Monitor und an BU1 der GDP64HS angeschlossen. Nach dem Einschalten der Spannung muß nach einer kurzen Copyright Meldung das Grundmenue des Grundprogrammes erscheinen. Bleibt das Bild dunkel, so sollten Sie erst mal am Kontrastregler des Monitors drehen. Es kann nämlich sein, daß dieser am Anschlag steht und somit kein Bild kommen erscheinen kann.

Haben Sie einen Z80-Rechner mit FLOMON, so genügt es für den ersten Test die SBC3 (mit FLOMON), die KEY bzw. KEY2, eine Speicherkarte mit mindestens 64k RAM (z.B. ROA64, RAM64, RAM256, ROA256/1M) und die GDP64HS einsetzen. Schließen Sie die Tastatur an die KEY und den Monitor an die GDP64HS an. Wenn Sie jetzt den Rechner einschalten, muß das FLOMON-Grundmenü erscheinen. Wenn dieses Menu nicht erscheint, können Sie natürlich noch kontrollieren, ob der Kontrastregler am Monitor richtig eingestellt ist. Bringt dies auch keinen Erfolg, sollten Sie mit Kapitel 6 fortfahren.

5.2.3 Test im 680xx System

5.2.3.1 Test im 68008 System

Zum Test der GDP64HS im 68008-System benötigen Sie als Mindestkonfiguration eine CPU68k, eine ROA64 (mit EASS 0-3 und mindestens einem RAM 8k; eingestellt auf Bank 0), eine KEY bzw. KEY2 und eine GDP64HS. Wenn Sie jetzt die Tastatur an der KEY und den Monitor an der GDP64HS anschließen, und die Spannung einschalten, erscheint nach einer kurzen Copyright Meldung das Grundmenue des Grundprogrammes 68k. Sollte es nicht erscheinen, können Sie wiederum versuchen den Kontrastregler des Monitors richtig einzustellen. Bringt dies keinen Erfolg sollten sie mit Kapitel 6 fortfahren.

5.2.3.2 Test im 68000-System

Zum Test der GDP64HS im 68000 System benötigen sie folgende Mindestkonfiguration: Eine CPU68000, zwei ROA64k (mit EG68000 ODD und EVEN; auf Bank 0 eingestellt), eine KEY bzw.KEY2 und die GDP64HS. Dabei ist zu beachten, daß eine ROA64 auf der "ODD"-Seite (ungerade) der CPU68000 steckt, und die andere auf ROA, die KEY und die GDP64HS auf der "EVEN" Seite gesteckt sein muß (siehe auch CPU68000 Handbuch Seite 5 unten). Nachdem Sie die Tastatur (an die KEY) und den Monitor (an die GDP64HS an BU1) angeschlossen haben, können Sie den Rechner einschalten. Es muß muß das Grundmenü des Grundprogrammes erscheinen. Bleibt der Bildschirm dunkel, sollten Sie den Kontrastregler des Monitors noch einstellen; dieser könnte am Anschlag sein und daher kein Bild zeigen. Bleibt auch dies ohne Erfolg, können Sie mit Kapitel 6 fortfahren.

5.2.3.3 Test im 68020-System

Zum Test der GDP64HS im 68020 System benötigen Sie folgende Mindestkonfiguration: CPU68020, vier ROA64k mit EG68020 auf Bank 0 eingestellt, eine KEY bzw. KEY2 und die GDP64HS. Auch bei dieser Konfiguration sollten Sie darauf achten, daß Sie die Baugruppen an die richtige Stelle im Bezug auf die CPU68020 stecken (siehe hierzu im CPU68020 Handbuch Seite 8). Wenn Sie jetzt die Tastatur (an die KEY) und den Monitor an die GDP64HS (an BU1) anschließen, muß das Grundmenu des Grundprogrammes 68k auf dem Bildschirm erscheinen. Sollte kein Bild erscheinen, sollten Sie den Kontrastregler des Monitor noch einstellen; ist dieser nämlich am Anschlag, kann kein Bild erscheinen. Bringt dies aber auch nicht den erhofften Erfolg sollten Sie mit Kapitel 6 fortfahren.

5.3 Beispiel und Testprogramme:

Beschreibung zu den Testprogrammen:

1. Beispiel:(Buchstaben) (lauffähig ohne Flomon)

Ist das Bit 2 des Registers 70h auf 0, so darf kein Kommando an die GDP gegeben werden, da diese dann beschäftigt ist. Man muß also vor jeder Befehlsausgabe oder jedem Umsetzen eines der anderen Register darauf warten, daß dieses Bit auf eins liegt. (Siehe auch 'Warteschleife' wail).
Im Programm: Aufruf der Warteschleife durch 'call wail'.

Wird der Wert 6 an die GDP geleitet, so erfolgt 'Bildschirm Löschen' und es kann die Ausgabe beginnen.
Zuerst wird der Buchstabe 'A' in gewohnter Größe geschrieben. 0fh an 73h bewirkt 'Groß -Schreibung'.Das Zeichen 'B' wird groß auf dem Bildschirm dargestellt. Danach wird wieder auf Kleinschreibung umgestellt (11h nach 73h), und es wird der Buchstabe 'C' geschrieben.

Die Verzögerung gestattet es, das Bild eine Weile anzusehen, bevor wieder ins Betriebssystem zurückgesprungen wird.

2. Beispiel:(Figurenzeichnen mit FLOMON)

Dieses Programm zeigt das Zeichnen verschiedener Figuren auf dem Bildschirm.

Mit dem Befehl 'ld de,A1' wird die unter 'A1' stehende Information eingelesen und mit 'call string' zur Ausführung gebracht.
Mit der Routine 'Eingabe' wird auf einen Tastendruck gewartet. Was die einzelnen Zeichen hinter db.. bedeuten ist dem Programm zu entnehmen.

3. Beispiel:(Vektoren mit FLOMON)

Das dritte Beispiel zeigt die Darstellung von Vektoren, die zu einem sternähnlichen Gebilde zusammengefügt werden.
Die Befehle 'clrall' und 'wait' sind FLOMON- Befehle ;ihre Sprungadressen werden am Programm Anfang definiert.
'clrall' löscht alle Bildschirmseiten und 'wait' führt genau das aus, was im vorigen Beispiel in der 'wail' Routine stand.
Zunächst werden die X und Y Register geladen (=Anfangspunkt Bildschirmmitte), der Schreibstift gesetzt (PEN down), und die Richtung, in der gezeichnet werden soll, festgelegt.
Das Ganze wird in der Schleife 'LOOP' abgearbeitet, die durch incrementieren von b die Zeichenrichtung ändert.

```
                                .z80
                                ;*********************************************
                                ;*     EF 9366    Testprogramm  RAKU 3'87     *
                                ;*********************************************

0070                            gdp     equ     70h     ;BASIS
0060                            seite   equ     60h     ;Seitenadr.
                                        org     8800h

8800'  3E F0                    start:  ld a,0F0h       ;Seite 3 verwenden
8802'  D3 60                            out (seite),a   ;warten bis GDP fertig
8804'  CD 8839'                         call wail       ;Warteschleife aufrufen
8807'  3E 06                            ld a,6          ;Bildschirm löschen
8809'  D3 70                            out (gdp),a     ;und ausführen

880B'  CD 8839'                         call wail       ;Warteschleife aufrufen
880E'  3E 03                            ld a,3          ;PEN down
8810'  D3 71                            out (gdp+1),a   ;PEN-Mode
8812'  3E 41                            ld a,41h        ;Zeichen A laden
8814'  D3 70                            out (gdp),a     ;und ausgeben

8816'  CD 8839'                         call wail       ;Warteschleife aufrufen
8819'  3E 0F                            ld a,15         ;* 15 (Zeichen gross)
881B'  D3 73                            out (gdp+3),a
881D'  3E 42                            ld a,42h        ;Zeichen B laden
881F'  D3 70                            out (gdp),a     ;und ausgeben

8821'  CD 8839'                         call wail       ;Warteschleife aufrufen
8824'  3E 11                            ld a,11H        ;* 1 (Zeichen klein)
8826'  D3 73                            out (gdp+3),a
8828'  3E 43                            ld a,43h        ;Zeichen C laden
882A'  D3 70                            out (gdp),a     ;und ausgeben

                        ; Verzögerung
882C'  01 2222                          ld bc,2222h     ;z.B. 2222 mal durchlaufen
882F'  0B                       wschl:  dec bc          ;Schleifenanfang
8830'  16 FF                            ld d,0ffh       ;innere Schleife
8832'  15                               er: dec d
8833'  20 FD                               jr nz,er
8835'  78                               ld a,b          ;durch diese Befehle wird
8836'  B1                               or c            ;das Flag Register beeinflusst
8837'  20 F6                            jr nz,wschl

                        ; Warteschleife
8839'  DB 70                    wail:   in a,(gdp)      ;Warten bis GDP fertig
883B'  E6 04                            and 4           ;(Maskierung)
883D'  28 FA                            jr z,wail       ;erst wenn fertig
883F'  C9                               ret             ;dann nächsten Befehl

                                END
```

```
                        .Z80
                        ;*****************************************************
                        ;*    ZEICHENPROGRAMM          raku 3'87             *
                        ;*****************************************************

000D                            cr      equ     0dh       ;Return-Taste
0060                            seite   equ     60h       ;Seitenadresse
0005                            system  equ     00005h    ;Systemadresse
0024                            stop    equ     '$'       ;Ende Eingabeliste
                                org     8800h

8800'  AF                   start: xor a                   ;Hier wird Seite 0
8801'  D3 60                       out (seite),a          ;verwendet

8803'  11 8820'                    ld de,al               ;Einlesen der Liste
8806'  CD 880C'                    call string            ;ausführen
8809'  C3 8812'                    jp eingabe             ;UP Eingabe rufen

880C'  0E 09                string: ld c,9h                ;Zeichenstring zur
880E'  CD 0005                      call system            ;Ausführung bringen
8811'  C9                           ret

8812'  3E FF                eingabe:ld a,0ffh              ;Warten bis
8814'  5F                           ld e,a                 ;eine Taste
8815'  0E 06                       ld c,6h                ;gedrückt wird
8817'  CD 0005                      call system
881A'  FE 00                        cp 0h
881C'  CA 8812'                     jp z,eingabe
881F'  C9                           ret

8820'  1B 1B 47 0D          al:    db 1bh,1bh,'G',cr      ;Zeichenliste (Grafikmodus)
8824'  5A 0D                       db 'Z',cr              ;Bildschirm löschen
8826'  4D 20 31 30                 db 'M 100 100',cr      ;Anfangspunkt setzen
882A'  30 20 31 30
882E'  30 0D
8830'  52 20 35 30                 db 'R 50 50',cr        ;Rechteck zeichnen
8834'  20 35 30 0D
8838'  4F 20 33 30                 db 'O 30 30 0 360',cr  ;Kreis zeichnen
883C'  20 33 30 20
8840'  30 20 33 36
8844'  30 0D
8846'  42 20 48 61                 db 'B Hallo',cr        ;'Hallo' schreiben
884A'  6C 6C 6F 0D
884E'  58 20 30 0D                 db 'X 0',cr            ;Rücksprung
8852'  41 24                       db 'A',stop            ;zum Alpha-Modus
                            END
```

```
                                .z80
                               ;*************************************************
                               ;*      Demoprogramm Vectoren      raku  3'87      *
                               ;*************************************************
0070                                   gdp     equ     70h     ;BASIS
0060                                   seite   equ     60h     ;Seitenadr.
F040                                   clrall  equ     0f040h  ;Bildschirmseiten löschen
F055                                   wait    equ     0f055h  ;Warten
0005                                   system  equ     00005h  ;Systemadresse
                                       org     0100h

0100'  CD F040                 start:  call clrall
0103'  CD F055                         call wait
0106'  3E 10                           ld a,010h               ;Hier wird Seite 1
0108'  D3 60                           out (seite),a           ;verwendet

010A'  06 17                           ld b,23
010C'  0E 20                           ld c,32
010E'                                  loop:
010E'  CD F055                                 call wait
0111'  3E 05                                   ld a,5
0113'  D3 70                                   out (gdp),a

0115'  CD F055                                 call wait       ;Warteschleife aufrufen
0118'  3E 7E                                   ld a,07eh       ;Y-Register low
011A'  D3 7B                                   out (gdp+11),a

011C'  CD F055                                 call wait       ;Warteschleife aufrufen
011F'  3E FF                                   ld a,0ffh       ;X-Register low
0121'  D3 79                                   out (gdp+09),a

0123'  CD F055                                 call wait
0126'  3E 03                                   ld a,3h         ;pen down
0128'  D3 71                                   out(gdp+1),a

012A'  CD F055                                 call wait       ;Warteschleife aufrufen
012D'  78                                      ld a,b          ;Richtung festlegen
012E'  D3 70                                   out (gdp),a

0130'  CD F055                                 call wait       ;Warteschleife aufrufen
0133'  3E 45                                   ld a,45h        ;laenge X-Richtung
0135'  D3 75                                   out (gdp+5),a

0137'  CD F055                                 call wait
013A'  3E 50                                   ld a,50h        ;laenge Y-Richtung
013C'  D3 77                                   out (gdp+7),a

013E'  04                                      inc b
013F'  78                                      ld a,b
0140'  91                                      sub c
0141'  20 CB                                   jr nz,loop
```

```
0143'  3E FF          eingabe:ld a,0ffh
0145'  5F                     ld e,a
0146'  0E 06                  ld c,6h
0148'  CD 0005                call system
014B'  FE 00                  cp 0h
014D'  CA 0143'               jp z,eingabe

                      END
```

Zum Schluß hier noch ein kleines Programm, welches auf 680xx-Systemen läuft:
In diesem Beispiel wird von den Befehlen 'hebe','senke','drehe' und 'schreite' Gebrauch gemacht.
Zuerst wird die Schildkröte bei x = 50, y = 50 positioniert, Richtung nach oben. Dann schreitet sie 50 mal schreibend nach oben, hebt an, schreitet 50 mal ohne zu schreiben, wird gedreht und schreibt weiter.

```
;*********************************************************
;*        Demo- Programm (Schildkroete)                  *
;*        fuer 680xx- Systeme                            *
;*********************************************************

move #50,d1             ;setzen bei 50/50
move #50,d2
move #90,d3             ;Richtung nach oben
jsr  §set

move #100,d0            ;50 * schreiten und zeichnen
jsr  §schreite

jsr  §hebe             ;Schreibstift anheben
move #50,d0            ;50 * schreiten und zeichnen
jsr  §schreite

move #-45,d0           ;um 45 Grad drehen
jsr  §drehe

jsr  §senke            ;Schreibstift absenken
move #50,d0            ;50 * schreiten und zeichnen
jsr  §schreite

move #-45,d0           ;um 45 Grad drehen
jsr  §drehe

move #50,d0            ;50 * schreiten und zeichnen
jsr  §schreite
rts
```

5.4 Jumperung der Baugruppe

Auf der Baugruppe sind die Jumper JMP1 bis JMP5, Alle diese Jumper sind voreingestellt und müssen bei Standardkonfigurationen auch nicht geändert werden. Hier nun die Beschreibung der einzelnen Jumper:

JMP1: Damit wird in Verbindung mit JMP2 und JMP3 auf den Betrieb mit dem EF 9367 umgestellt. Dabei ist jedoch der 14 MHz- Quarz gegen einen 12MHz- Quarz auszuwechseln.
In der voreingestellten Position von JMP1 wird der EF 9366 bedient.
Beim Einsatz des Videoprozessors EF 9367 ist eine Hardcopy- Funktion bis auf weiteres nicht möglich.

JMP2: JMP2 dient zur Umstellung der Auflösung der GDP64HS. Die Standardauflösung liegt bei 256 x 512 Bildpunkten. Legt man diesen Jumper um, kann man auch mit einer Auflösung von 512 x 512 Bildpunkten im Interlace-Modus (Zeilensprungverfahren) arbeiten. Allerdings kann dies nur mit dem Graphikprozessor 9367 durchgeführt werden. Außerdem gibt es für diese Auflösung eigentlich keine Software, da das Flimmern des Interlace Modus wohl meistens als störend empfunden wird.

JMP3 JMP3 dient nur dazu einzustellen, welcher Graphikprozessor verwendet wird (9366 oder 9367). Voreingestellt ist der 9366.

JMP4: JMP4 ist in drei Abschnitte aufgeteilt. Die oberen 2 Brücken dienen zum Invertieren des Videosignales. Die Brücken 3 und 4 dienen zum Invertieren des HSync-Signales und die Brücken 6 und 7 zum Invertieren des Vsync-Signales.

JMP4

o 1 o
o 2 o
o 3 o
o 4 o
o 5 o
o 6 o

Videosignal invertiert:	1 gebrückt, 2 offen
Videosignal nicht invertiert:	2 gebrückt, 1 offen
HSYNC invertiert:	3 gebrückt, 4 offen
HSYNC nicht invertiert:	4 gebrückt, 3 offen
VSYNC invertiert:	5 gebrückt, 6 offen
VSYNC nicht invertiert:	6 gebrückt, 5 offen

JMP5: JMP5 dient lediglich dazu, das VSYNC-Signal auf den INT zu legen. Dieser JMP ist natürlich offen.
JMP5 wurde vorgesehen, um einen definierten Interrupt von 20 ms erzeugen zu können. Dies ist eventuell für Multitasking-Aufgaben interessant. Sonst bleibt dieser Jumper immer offen.

6. Fehlersuchanleitung

Sollte Ihre Baugruppe bei den in Kapitel 5 beschriebenen Tests nicht funktionieren, so heißt es jetzt systematisch auf Fehlersuche zu gehen.

Wir wollen Ihnen nun ein paar Vorschläge machen, wie eine systematische Fehlersuche mit und ohne Oszilloskop vor sich gehen kann:

6.1 Mögliche Fehler und ihre Behebung

6.1.1 Sind die bisher verwendeten Baugruppen in Ordnung? (Funktionierte das System ohne die Baugruppe GDP 64K?)

6.1.2 Sind die Jumper richtig gesteckt?

6.1.3 Machen Sie zuerst eine Sichtprobe. Können Sie irgendwo auf Leiterplatte unsaubere Lötstellen (zuviel Lötzinn, manch- mal zieht das Lötzinn Fäden) erkennen, die eventuell einen Kurzschluß verursachen könnten? Dann müssen Sie diese Lötstellen nachlöten und die unzulässige Verbindung beseitigen.

6.1.4 Haben Sie auch alle IC's richtig herum und am richtigen Platz eingesteckt? (Vergleiche mit Bestückungsplan)

6.1.5 Sind alle gepolten Bauteile (Elkos, Dioden, usw.) richtig herum eingelötet?

6.1.6 Haben Sie auch keine Lötstelle vergessen zu löten? (sehen Sie lieber noch einmal nach)

6.1.7 Sehen Sie irgendwo "kalte" Lötstellen?
Kalte Lötstellen erkennt man daran, daß sie nicht glänzen, sie sind im Vergleich mit richtig gelöteten Lötstellen trübe.

6.1.8 Haben Sie auch nicht zu heiß gelötet?
Wenn der Lötkolben zu heiß eingestellt ist und (oder) Sie zu lange auf der Lötstelle bleiben, dann kann es passieren, daß sich die Leiterbahnen von der Platine lösen und Unterbrechungen bilden. Ferner kann es auch passieren, daß Durchkontaktierungen unterbrochen werden, oder daß Bauteile durch zu heißes Löten zerstört werden.

6.1.9 Nehmen Sie alle IC's aus ihren Fassungen. Schauen Sie sich beim herausnehmen die IC-Füßchen genau an. Manchmal sind Füßchen umgeknickt oder steckten daneben. Nehmen Sie sich die Layouts zur Hand und kontrollieren Sie alle Leiterbahnen, mit einem Durchgangsprüfer oder mit einem Ohmmeter auf Durchgang. Bereits kontrollierte Leiterbahnen können Sie, der Übersicht wegen, auf dem Layout mit Bleistift durchstreichen oder mit Farbstiften nachziehen.

6.1.10 Prüfen Sie die Versorgungsspannung mit einem Digital-Voltmeter (am Bus +5V, nicht am Netzgerät, da am Kabel bei starker Belastung bis zu 0,5V abfallen können). Toleranzen von +/- 5% also von 4,75V bis 5,25V sind erlaubt. Falls die Spannung zu gering ist, prüfen Sie, ob die Verbindung vom Netzteil zum Bus mit ausreichend dickem (mind. 2 mm Quadrat) Kabel erfolgt ist. Gegebenenfalls müssen Sie Ihr Netzteil nachregeln. Vorsicht: nie über 5,1V nachregeln, da sich auf einigen Platinen 5,1V Zenerdioden befinden, die ab 5,1V durchschalten, was entweder zum Zusammenbruch Ihrer Versorgungsspannung führt oder die Zenerdiode bis zu Ihrer Zerstörung erhitzt.
Übrigens: Wir empfehlen 5,05V.

Wenn Sie alle Leiterbahnen kontrolliert und nichts gefunden haben, dann ist die Wahrscheinlichkeit groß, daß ein Bauteil defekt ist.

Wenn Sie einen Prüfstift oder ein Oszilloskop haben, dann können Sie jetzt überprüfen, ob an den jeweiligen Ausgängen die richtigen Signale anliegen. Welche Signale wo anliegen müssen, können Sie aus der Schaltungsbeschreibung und aus dem Schaltplan entnehmen.

Falls Sie keine Meßgeräte haben, dann müssen Sie alle Bauteile systematisch austauschen, bis Sie das Defekte gefunden haben. Verwenden Sie dazu eventuell eine zweite Baugruppe (die eines Freundes oder eines Bekannten).

Sollten Sie gar nicht zurande kommen, hilft Ihnen unser Pauschal-Reparatur-Service, dessen Bedingungen Sie der Preisliste entnehmen können.

7. Schaltungsbeschreibung

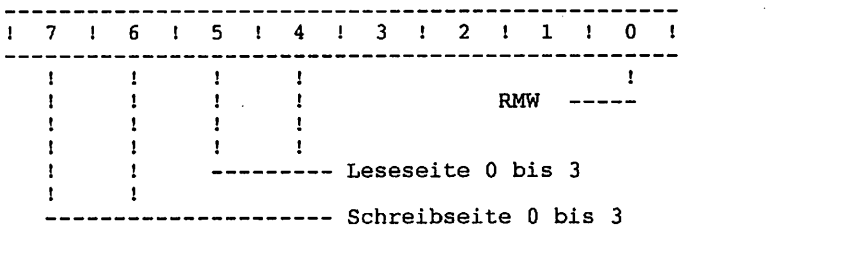
7.1 Wie funktioniert die Baugruppe ?

Das Herz der GDP64HS ist der Graphik Display Prozessor EF 9366 für 512 mal 256 Bildpunkte. Es könnte aber auch der EF 9365 oder der EF 9367 verwendet werden, dazu müssen aber die Brücken JMP1, JMP2 und JMP3 verändert werden. Standard ist die Bestückung mit dem EF 9366, auf den auch die gesamte Software abgestimmt ist.

Der GDP J28 ist an den Datenbus und an die Adressbits A0...A3 angeschlossen. Die vier Adressbits bewirken, daß eines der 16 GDP-Register ausgewählt wird. Über den Datenbus kann dann in das gewählte Register geschrieben bzw. vom gewählten Register gelesen werden. Der Datenbus des Graphikprozessors ist außerdem noch durch einen biderektionalen Treiber (J27) vom Systemdatenbus getrennt, der nur dann aktiviert wird, (J27/19) wenn auf ein Register des GDP zugegriffen wird (Portadressen 70h bis 7Fh). Die Richtung des Datentransfers wird durch das READ-Signal festgelegt (J27/1).

Die Ausdekodierung der Adressen 70h bis 7Fh übernimmt der 3 zu 8 Dekoder (J30). Wird eine Portadresse 70h bis 7Fh angesprochen, so wird der Ausgang 7 (J30/7) aktiviert (low). Dieser Ausgang steuert die Enable-Eingänge des Graphikprozessors (J28/17) und des bidirektionalen Treibers (J27/19).

Die Ausdekodierung der Adressen 60h und 61h übernimmt zum Teil auch der 3 zu 8 Dekoder 74LS138. Der Ausgang Y6 (J30/9) wird aktiviert, wenn eine Adresse von 60h bis 6Fh angesprochen wird. Die Selektion der Adressen 60h und 61h aus diesen 16 Portadressen übernimmt der zweite 3 zu 8 Dekoder (J29). An dessen Ausgang Y0 (J29/15) liegt dann ein Low-Signal an, wenn auf Port 60h zugegriffen wird. Der Ausgang Y1 (J29/14) wird aktiv (LOW), wenn auf Port 61 zugegriffen wird. Soll z.B auf Port 60 (Seiten- und RMW-Port) geschrieben werden, so wird der Ausgang J29/15 low, ebenso ist das Signal -WR LOW. Dadurch wird der Ausgang des ODER J3/8 low, was das Latch J24 dazu bewegt neue Daten aufzunehmen. Mit dem Inhalt dieses Latches kann die Schreib- und Leseseite eingestellt, sowie das Steuerbit für RMW gesetzt werden (siehe Abb.).



RMW: Bit 0 = LOW RMW nicht aktiviert
Bit 0 = HIGH RMW aktiviert
Die Schreib- und Leseseite kann universell eingestellt wer-

den. Man kann z.B. im Hintergrund schreiben, also Lese- und Schreibseite verschieden wählen, oder man kann die Seiten wechselseitig hin und herschalten, um zwei Seiten qasi gleichzeitig sichtbar zu haben. Die Seiten können synchron (nach einem VSYNC-Signal) oder asynchron (zu einem beliebigen Zeitpunkt umgeschalten werden.

Damit es hierbei nicht zu Kollisionen kommen kann, dient J12, ein 2 mal 4 zu 1 Multiplexer, als Umschalter für die Videospeicherseiten (je 16kB).
Zunächst sei die Brücke (Jumper) JMP2 auf der Stellung '9366' (Ist auf der Platine bereits realisiert). Damit ist nur der obere Teil des Multiplexers (J12) maßgebend. J12 erzeugt mit seinen Ausgang 1Y (J12/7) das jeweils höchstwertige Adressbit A15 bzw. A7 der gemultiplexten Adresse, deren Wertigkeiten von der gewählten Seite abhängig sind. Abhängig von der Information an den Select- Eingängen A und B (J12/14,2) wird einer der Eingänge 1C0...1C3 (J12/6,5,4,3) auf den Ausgang 1Y (J12/7) durchgeschaltet. Die logischen Pegel an den Eingängen 1C0 und 1C1 (J12/6,5) bestimmen die Seite, aus der gelesen werden soll, 1C2 und 1C3 (J6/4,3) definieren die Schreibseite (vgl. auch obige Abbildung).
Die Taktaufbereitungslogig (J8 und J7) sorgt dafür, daß die Adressbits im für die Speicher richtigen Timing erzeugt werden.
Bild 9 zeigt beispielsweise folgende Betriebsart: Auf Seite 1 einschreiben und die Seite 0 auslesen. Die Signale S153/1C0 und S153/1C1 liegen auf low (entspricht Seite 0). Das Signal S153/1C2 liegt auf high und S153/1C3 liegt auf low, somit ist Seite 1 zum Beschreiben angewählt.
In der ersten Hälfte des Timingdiagrammes ist BLK (Signal S153/B) aktiv, d.h. der EF9366 ist mit der Bildausgabe aus Speicherseite 0 beschäftigt. Das Signal S153/1Y repräsentiert die Adressleitungen A15 bzw. A7, jenachdem ob eine Spaltenadresse oder eine Zeilenadresse ausgegeben wird.
A15 bzw.A7 sind hierbei erwartungsgemäß low.
In der zweiten Hälfte des Timingdiagrammes (Bild 9) ist die Leitung BLK (Signal S153/B) unwahr geworden; es wird in die DRAMs eingeschrieben. Man erkennt, daß Signal S153/1Y seine Polarität wechselt: Wenn die fallende Flanke von RAS auftritt (geschieht ca. 150ns vor CAS, das sind rund 100% der Zeitdauer von CAS) ist A15 low und bei CAS ist A7 high. A15 und A7 sind aber ja ein- und- dasselbe Signal (hier S153/1Y). Signal RAS wurde hierbei nicht explizit aufgenommen, da es ja acht dieser Leitungen gibt.

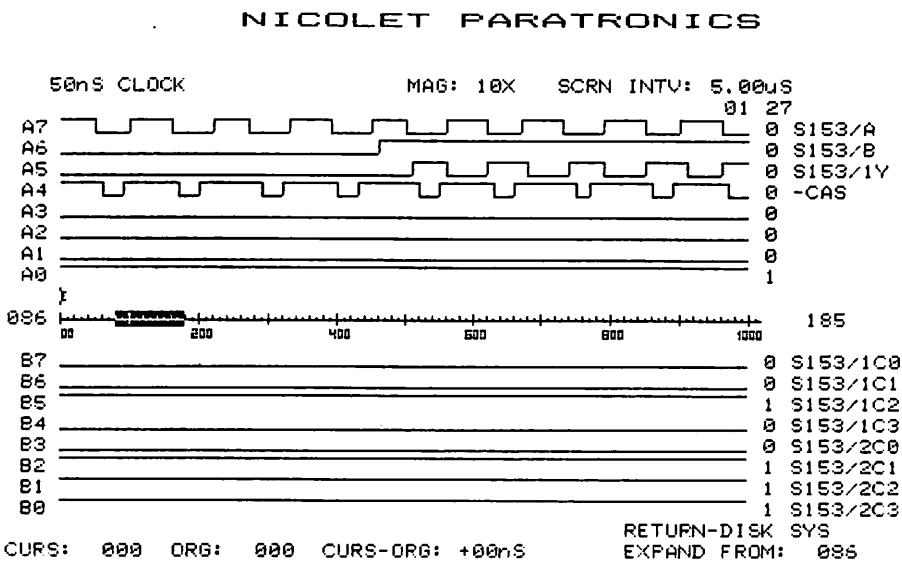


Bild 9

Seitenanwahl und Adressbiterzeugung.
Zur Zeitintervallabschätzung: Der Beobachtungszeitraum beträgt 5us.

Wenn die RMW-Logik aktiviert wird, bewirkt sie eigentlich nichts anderes, als daß sie Dateneintragungen vom GDP aus in den Videospeicher abfängt, wenn es erforderlich ist modifiziert und schließlich in die vorgesehene Speicherstelle einschreibt. Dieser Ablauf ist natürlich sehr zeitkritisch, da während der Zeitdauer eines normalen Schreibzyklus zuerst noch gelesen und dann geschrieben wird.
RMW verändert also den Videospeicherinhalt, ohne daß es der GDP mitbekommt.
Dies ist dann nützlich, wenn z.B. auf ein zum Teil weißes Bild noch gezeichnet werden soll. Ohne RMW wird dann Weiß auf weiß gezeichnet, was dann natürlich unsichtbar ist. Mit RMW wird der Inhalt des Speichers erst einmal "angeschaut" (weiß oder schwarz) und dann entsprechend mit dem neuen Speicherinhalt modifiziert, d.h. wird auf einen hellen Hintergrund ein helles Rechteck gezeichnet, so erscheint dies mit RMW schwarz und ist damit sichtbar. Sehr nützlich ist dies z.B. für einen Maus-Zeiger, da dieser durch zweimaliges Zeichnen wieder verschwindet, ohne daß das Hintergrundbild zerstört wird.

Der Read-Modify-Write Vorgang kann in drei Schritten erläutert werden:

1.READ

Will der GDP in eine Speicherzelle (1Bit) schreiben, wird die Leitung WE (Write Enable) am DRAM- Baustein kurzfristig "künstlich" auf high gehalten. Da jedoch die übrigen Signale wie Adressen, CAS und RAS stimmen, glaubt sich der betroffene DRAM- Baustein im Lese-Modus und gibt die gewünschte Bitinformation auf den internen Datenbus.

Im Timingdiagramm (Bild 10) ist zu erkennen:
RMW- Modus ist eingeschaltet (Signal RMWENABL ist high).
Der GDP 9366 leitet einen (vermeindlichen) WRITE-Zyklus ein (Signal 9366/-WR ist auf low und die Signale J16/-RAS und -CAS werden nacheinander low).
Der Baustein J16 befindet sich jedoch im Lesemodus (Signal 4164/-WR ist noch high) und gibt ein Bit aus (Signal J16/DO wird low). Die übrigen sieben DRAM-Bausteine sind nicht angesprochen, deren Datenausgänge sind hochohmig (Signale J20/DO, J22/DO, J23/DO, J17/DO und J18/DO sind high, da für Logik- Analysator tristate und high nicht unterscheidbar sind).
Die Wertigkeit (low) des Datenausgangsbit von J16 besagt, daß an dieser Stelle momentan ein heller Bildpunkt ist.

2.MODIFY

Ist die Speicherinformation ausgelesen, wird sie nach folgendem Schema mit dem neuzuschreibenden Speicherinhalt verknüpft:

altes Pixel	neues Pixel	tatsächliches Pixel
hell (Bit =0)	hell (Bit =0)	dunkel (Bit =1) !!
hell (Bit =0)	dunkel (Bit =1)	dunkel (Bit =1)
dunkel (Bit =1)	hell (Bit =0)	hell (Bit =1)
dunkel (Bit =1)	dunkel (Bit =1)	dunkel (Bit =1)

(Pixel kommt von "picture element" und heißt: Bildpunkt, Bildelement.)

In Bild 10 ist zu erkennen, wie die Datenleitung, die zu den DRAM- Bausteinen führt (Signal 4164/DI), High- Potential annimmt. Das heißt, das tatsächliche Bildelement (Pixel) wird dunkel gezeichnet, obgleich der Prozessor 9366 ein helles Bildelement schreiben wollte (Signal 9366/DIN ist während der Beobachtungszeit low).

3.WRITE

Das so behandelte Datenbit wird erst jetzt in die Speicherstelle eingetragen, indem die WR- Leitung auf low gezogen wird (Signal 4164/WR), noch bevor die Auswahl des Bausteins J16 beendet wird (-CAS und J16/-16 gehen wieder auf high).
Wenn die Signale CAS und RAS unwahr werden, ist ein RMW- Zyklus abgeschlossen.

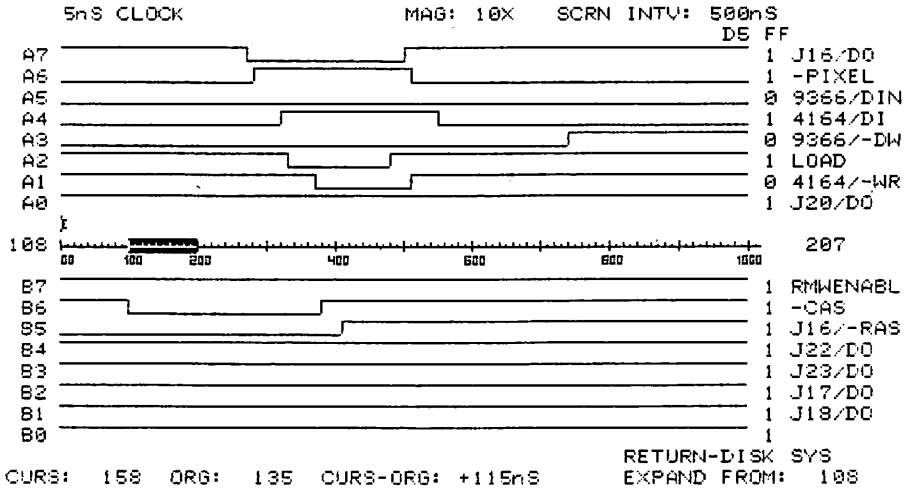


Bild 10

Timing eines RMW- Zyklus.

Zur Abschätzung der Zeitintervalle: Der gesamte dargestellte Zeitraum beträgt 500ns.

Schaltungstechnisch wurde dies mit nur wenigen Gattern realisiert. Da beim Schreiben auf der GDP64HS immer nur ein Bit geschrieben wird, ist auch immer nur einer der Speicherausgänge aktiv, die Ausgänge der anderen Speicher sind im hochohmigen Zustand. Dadurch können alle Speicherausgänge (siehe Bild 10, Signale J16/DO, J22/DO, J23/DO, J17/DO, J18/DO) auf ein Achtfach- NAND (J2/1,2,3,4,5,6,11,12) gelegt werden. Ist der Speicherausgang high (Bildpunkt dunkel), ist der Ausgang J2/8 invertiert, also auf high. Ist der Speicherausgang aber low, so ist der Ausgang J2/8 auch invertiert, also auf high (Bildpunkt hell) (siehe Bild 10, Signal -PIXEL). Das AND- Gatter (J5/9,10,8) dient lediglich dazu, den RMW zu sperren oder freizugeben. Ist das RMW-Bit gesetzt, geht das ausgelesene Bit weiter (zu J10/2), wenn nicht, wird es hier abgeblockt. Das ODER- Gatter (J10/1,2,3) vergleicht den ausgelesenen Speicherinhalt (durch J2 invertiert) mit dem einzuschreibenden Bit. War der ausgelesene

Punkt hell (Low-Signal), liegt wegen der Invertierung an J2 ein High Signal an J10/2. Soll jetzt wieder ein heller Punkt geschrieben werden (also low), so wird der Ausgang J10/3 high und es wird ein dunkler Punkt in den Speicher geschrieben (siehe Bild 10, Signal4164/-WR).

Die Verzögerung des Write-Signales für die DRAM- Speicher wird mit dem Load-Signal (J7/9) realisiert (siehe Bild 10, Signale LOAD und 4164/-WR). Mit J5/4,5,6 kann dieses Load-Signal mit Hilfe des RMW- Bits wieder gesperrt werden. Das OR-Gatter (J10/12,13,11) läßt ohne RMW das Schreibsignal DW (J28/14) original vom GDP passieren und mit RMW das Load-Signal .

Wird auf den Port 61h geschrieben, so wird J29/14 aktiviert und gleichzeitg der Ausgang J3/6. Dadurch wird auf den Scroll-Port (J26) ein Byte übergeben. Der Wert dieses Bytes (von 0 bis 255) wird als "Scroll-Wert" zur horizontalen Adresse aufaddiert.
Wird z.B. der Wert 40h auf Port 61h ausgegeben, so wird der Bildschirminhalt um 40h = 64 dezimal Punkte nach unten gescrollt. Der Rest, der unten vom Bildschirm verschwindet, wird oben wieder hineingeschoben (zyklisches Rotieren des Bildschirmspeichers). Der kleinste Wert, der beim Scrollen eine Wirkung zeigt, ist zwei, da die Datenleitung D0 nicht an den Scrollport 61h angeschlossen ist.
Die Steuerung des Hardscrolls wird von dem -BLK und dem Steuersignal für den Seitenport (J6/6) gesteuert. Diese beiden Signale werden ODER-verknüpft (J4/1/2/3) und ergeben ein vorgezogenes CAS- Signal für die Addierlogik. Die Addition der Scroll- Werte wird nur bei CAS durchgeführt, und außerdem nur im Display Modus, um den Schreibzugriff und den Refresh des Speichers nicht zu stören. In Bild 11 sind es die Signale LS74/6 und -BLK, die ODER- Verknüpft das Signal LS32/3 ergeben. Dieses verfrühte CAS- Signal aktiviert das Datenlatch ALS574 (J26/1); damit wird der Scrolloffset an die Addierer J14 und J15 angelegt. Mit dem Aktivieren des Latches J26 wird zugleich der Übertragseingang CI (Carry In) des niederwertigeren Addierers (J15/7) auf low gesetzt.
Die Addition des Scrolloffsets zur horizotalen Adresse ist in Bild 11 folgendermaßen nachzuvollziehen: Nachdem das Signal LS32/3 aktiv wird (=low), liegen am Addierer J15 der niederwertigere Teil der Horizontaladresse (=90h) (Signale DAD3...DAD6) und der Offsetwert 02h (Signale D4..D1) an.
DAD3 hat hierbei die höchste Wertigkeit und DAD6 die Niedrigste.
Die Signale M3...M6 präsentieren das Ergebnis der Addition, nämlich 92h. Auch hierbei ist M3 höherwertiger als M6.
Mit der fallenden Flanke von -CAS können nun die DRAM-Speicherbausteine die manipulierte Horizontaladresse übernehmen. Nachdem das Scrollen faktisch erledigt ist, wird der Latchbaustein J26 wieder hochohmig geschaltet (Signal LS32/3 wird high) und an den Eingängen der Addierer J14, J15 liegen High- Pegel an (Sicherheitshalber ist RN2 als Pull-Up- Widerstandsnetzwerk eingesetzt worden).
Damit aber nicht OFFh zu jeder folgenden Adresse hinzugezählt wird, wird gleichzeitig der CI- Eingang des niederwertigeren Addierers (J15/7) mit High- Pegel beschaltet

(0FFh + 01 = 00). Effektiv wird also zu den verikalen Adressen Null dazugezählt.

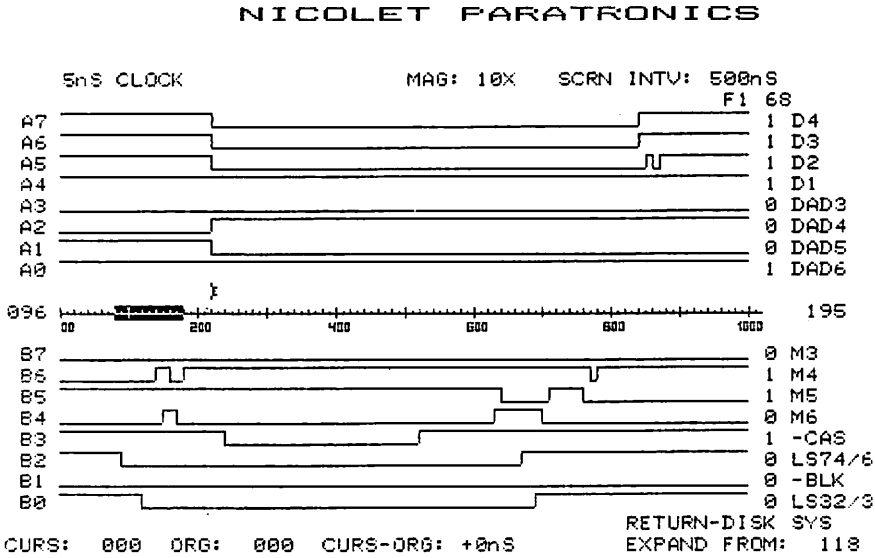


Bild 11

Timingdiagramm beim Hardscroll.

Dargestellt sind die Signale rund um den Addierer J15 (Manipulation des niederwertigeren Teils der Horizontaladresse).
Zum Abschätzen der Zeitintervalle: Der Beobachtungszeitraum erstreckt sich über 500ns.

Der Bildschirmspeicher kann über Port 60h rückgelesen werden. Dabei muß allerdings die Adresse des rückzulesenden Bytes im Bildschirmspeicher an das X- und Y-Register des Graphikprozessors übergeben werden und anschließend ins Kommandoregister der Befehl 0Fh eingetragen werden. Ist der Graphikprozessor im Display-Mode an der eingestellten Adresse angekommen, legt der Graphikprozessor das Signal MFREE auf low und beim nächsten Ladeimpuls (J7/9) wird der Datenwert ins Latch J25 übernommen. Dieses Byte muß nun über Port 60h eingelesen werden, bevor das Ganze wieder von neuem beginnt (Setzten der neuen Adresse, Befehl 0Fh ausgeben, warten bis Adresse im Display erreicht).

In Bild 12 wird beispielsweise der Wert FEh aus dem Videospeicher ausgelesen. Bevor dieses Bild aufgenommen werden konnte, wurde auf den dunklen Bildschirm an der Stelle x=256 und y=128 ein heller Punkt gezeichnet (Kurzvektor). Danach wurden die selben x- und y- Werte in die oben erwähnten xlsb- und ylsb- Register des EF9366 eingetragen. Normaler-

weise sind die Pixelkoordinaten mit den Speicheradressen nicht identsch, in diesem Fall funktioniert es aber. Nachdem der Wert 0Fh in das CMD- Register geschrieben ist, aktiviert der GDP beim Erreichen der vorgegebenen Speicheradresse die Leitung MFREE (Signal -MFREE wird low). Seitens des EF9366 wird nur ein Bildpunkt angesprochen, denn die Leitung -ALL ist schon am Anfang des Beobachtungszeitraumes unwahr (Signal -ALL ist high). Ist jedoch die Brücke JMP1 vor dem NAND- Gatter (J8/1,2,3) in der voreingestellten Position, wird der Demultiplexer 25LS2538 (J9) mit Hilfe der Gatter (J8/1,2,3) und (J8/4,5,6) dazu bewegt, alle -RAS- Leitungen auf low zu legen (siehe Signale -MFREE, -ALL, J8/3 und J7/QC).
J8/1,2,3 verknüpft die Ausgänge -ALL und -MFREE des GDP (J28), siehe Bild 12, Signale -MFREE, -ALL und J8/3. Eine Freigabe (Enable) des Demultiplexers J9 wird unterbunden, da die Enable- Eingänge -E1, -E2, E3, E4 zum Teil unwahre Information erhalten. Eingang -E1 wird von -DW des EF9366 angesteuert und ist zu diesem Zeitpunkt wahr, ebenso wie E3 (siehe Bild 12, Signal J7/QC). Die Eingänge -E2 und E4 des J9 sind hingegen unwahr beschaltet (siehe Bild 12, Signale J8/3, J8/6).
Der POL-Eingang des Multiplexers J9/12 schaltet alle Ausgänge in ihrer Polarität um. Ohne ein Lowsignal an POL zu diesem Zeitpunkt, wären alle Ausgänge von J9 auf '1'; durch POL=0 werden sie zu '0', d.h. alle Speicher erhalten ein identisches -RAS, das die Ausleseadresse definiert.
Damit geben alle acht DRAM- Bausteine ihre angewählte Bitinformation an den Latchbaustein (J25/3,4,7,8,13,14,17,18). In Bild 12 sind das die Signale D0, D1, D2, D3, D4, D5, D6 und D7. Die Signale D1....D7 repräsentieren hierbei Bildpunkte links vom Punkt x=256 und y=128 in der gleichen Bildzeile.
Mit dem Auftreten des nächsten LOAD- Impulses (negative Logik) erhält Latch J25 einen Clock- Impuls vom ODER-Baustein (J3/11) und speichert die Bildschirminformation (in Bild 12 die Signale -MFREE, LOAD und J25/CLK).
Befände sich JMP1 in seiner alternativen Position, würde der Demultiplexer J9 normal arbeiten können; nur ein Bit würde ausgelesen werden. Alle übrigen Datenausgänge währen hochohmig und der Hauptprozessor (Z80 oder 680xx) würde sie als high erkennen. JMP1 wird jedoch nur beim Betrieb des EF 9367 umgesteckt, um die übrigrn GDP- Funktionen zu ermöglichen, ein Rücklesen des Videospeicherinhaltes ist damit (noch) nicht möglich.

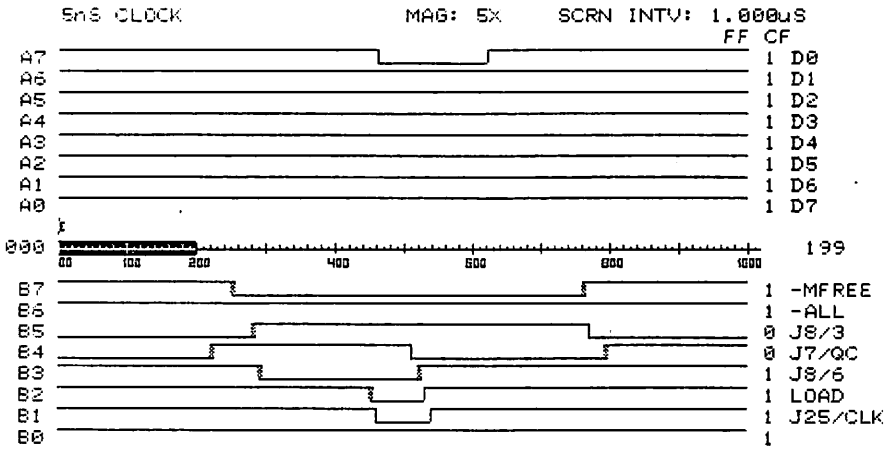


Bild 12

Auslesen des Videospeicherinhaltes (hier Datum FEh).
Zum Abschätzen der Zeitintervalle: Der Beobachtungszeitraum beträgt lus.

Der Arbeitstakt der GDP64HS wird über den Quarz Q1 und dem Taktgenerator J11 erzeugt. Es sollte darauf geachtet werden, daß auf Grund der hohen Taktrate ein 7404 ohne LS eingesetzt wird! Der Bildpunkttakt liegt am Punkt C der 7-poligen Stiftleiste am oberen Platinenrand.

Der 14MHz Bildpunkttakt führt direkt zum Schieberegister J13 (wird später erwähnt) und zum Zähler J7, der die sonstigen im System benötigten Signale erzeugt. J7 zählt von 8 bis F und lädt sich nach Überschreiten von 'F' wieder selbst über die Eingänge A, B, C und D (J7/3,4,5,6). Der Grundtakt steuert das Schieberegister J13 über J13/7 (Punkttakt der Graphik). Weiter führt er zum Videomischer J4/10. Dadurch ergibt sich ein Punkttakt von 71 ns. Der Grundtakt der mit dem Zähler J7 maximal durch acht geteilt wird, ergibt einen Systemtakt von 1,75 MHz, der am Ausgang J7/12 ansteht und als CK zum GDP geführt wird.
Über IC J8/8 wird das CAS - Signal für die Speicher erzeugt.

Um Störungen zu vermeiden, darf in ein Bild nur dann geschrieben werden, wenn der Strahl außerhalb des darstellbaren Bereiches liegt. Der GDP (J28) teilt dies durch seinen BLK- (Blank) Ausgang J28/25 mit, der an den Eingang B von J12 8J12/2) gelegt ist. Der Eingang A wird von einer Taktflanke, die mit J6 zwischen RAS und CAS liegt, belegt. Dadurch wird die Adresse einer aktuellen Seite nur zum erlaubten Zeitpunkt umgeschaltet, da die dynamischen Speicher die Adressinformation ja sequentiell angelegt bekommen.

In Bild 13 sind diese Signale nocheinmal abgebildet.

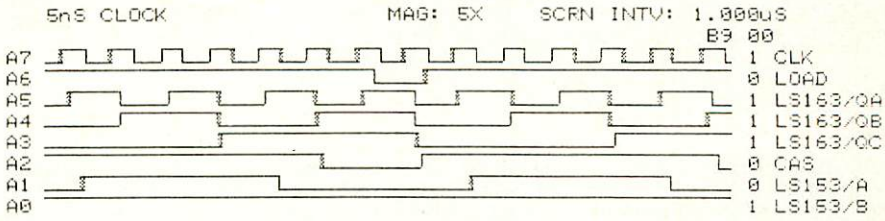


Bild 13

Takterzeugung.

Zur Abschätzung der Zeitintervalle: Der Beobachtungszeitraum erstreckt sich über lus.

Der EF 9366 übergibt eine 14 Bit große Adresse für acht Bildpunkte auf einmal (Byte) an seinen Ausgängen DAD0 (J28/37)...DAD6 (J28/59). Aufgrund einer Eigenart des EF9366 erscheint am Ausgang DAD0 das jeweils höchstwertigste Adressbit und an DAD6 das Niederwertigste. Die Byteadressen werden gemultiplext, d.h. da nur sieben Ausgänge vorhanden sind, wird die Adresse auf zweimal übertragen; zuerst die Niederwertigen dann die höherwertigen sieben Bit. Dieser Vorgang wird durch die Signale CAS und RAS gesteuert, und zwar jeweils zur abfallenden Flanke von RAS bzw. CAS wird ein Adressteil übernommen. Die Adressausgänge DAD führen über die Addierstufen J14 und J15 zu den DRAMs. In Wirklichkeit werden die 64kBit- DRAM- Bausteine aber mit zwei mal acht Adressbits versorgt, wobei das achte Bit durch die zusätzliche Hilfslogik mit J12 (Seitenlogik) erzeugt wird.
Mit den Ausgängen MSL0 (J28/6)...MSL2 (J28/7) kann der EF9366 darüberhinaus auch noch einzelne Pixel adressieren (Auswahl eines von acht DRAM- Bausteinen). Sie werden zum Baustein J9 geleitet, einem 3 zu 8 Decoder mit einigen Besonderheiten:
J9 decodiert die an seinen Eingängen (A,B,C) anstehende Information und schaltet jeweils einen Ausgang J9 /11,9,8 usw. auf low. Dadurch wird ein RAS erzeugt, und die Adressen dem jeweiligen Speicherchip übergeben. Die Besonderheit von J9 ist, daß sich die Polarität der Ausgänge J9/11,9,8,18,19,1,2,3 mittels des Einganges POL (J9/12) steuern läßt: Wenn POL auf high ist, herrscht am Ausgang negative Logik vor, wenn POL auf low ist, sind die Ausgänge auf positive Logik eingestellt. Dieser Effekt wird u.a. im Rücklesemodus ausgenutzt. Desweiteren können alle Ausgänge mit vier Enable- Eingängen gesperrt werden. Dabei nehmen alle Ausgänge gleichen Pegel an. Ob sie entweder high oder low werden, hängt von der Information am POL- Eingang ab.

Die Ausgabedaten der RAM's führen an die Eingänge des 8-Bit Schieberegister J13. Jeweils zu Beginn eines neuen Taktschrittes werden die 8 Punktdaten hier übernommen, gesteuert durch den LOAD- Impuls von J7. Während der Blank-Zeit, gesteuert durch BLK (J28/25), wird der LOAD- Impuls an ODER-Gatter J4/12,13,11 unterbunden. Im Schieberegister J13 wird

die feste '1' des 'Serial Inputs' (J13/1) übernommen. Eine '1' entspricht einem dunklen Bildpunkt.

Die aus dem Schiebregister hinausgeschobenen Daten (J13/13) werden am ODER- Glied (J4/9,10,8) mit dem Punkttakt (14 MHz) verknüpft und der Mischstufe zugeführt. Sie besteht aus den Invertern (J1/9,8), (J1/13,12), den Widerständen R1...R3 und dem Transistor T1. Hier wird noch ein Synchronisationssignal, das vom GDP generiert wird (J28/34), hinzuaddiert. Das Signalgemisch steht mit einem Quellwiderstand von 75 Ohm zur Verfügung. Am Ausgang BU1 liegt dann das BAS-Signal an. Am Ausgang ST3 liegen die Video-Signale mit TTL-Pegel an, wobei die Synchronsignale und das Video-Signal jeweils invertiert werden können (JMP4/x).

7.2 Hinweise zum Monitoranschluß :

VS - Signal: Eine logische Eins geht direkt an ST3/9 (nichtinvertiert) wenn JMP4/6 gesetzt ist. Ist JMP4/5 gesetzt, wird das VS - Signal negiert an ST3/9 geführt.

HS - Signal: Eine logische Eins geht direkt an ST3/8 wenn JMP4/4 gesetzt ist. Ist JMP4/3 gesetzt, wird das HS - Signal negiert an ST3/8 geführt.

VI - Signal: Eine logische Eins geht direkt an ST3/7 wenn JMP4/2 gesetzt ist. Ist JMP4/1 gesetzt, wird das VI - Signal negiert an ST3/7 geführt.

Bei Verwendung eines normalen Monitors wird die BAS- Buchse BU1 verwendet, soll ein TTL-Monitor z.B. ein IBM - Monitor angeschlossen werden, benötigt man das Viedeo-Signal (nur Bildinformation) und zusätzlich die Synchronisationssignale (HS und VS). Deshalb wird der Monitor an ST3 angeschlossen. JMP4 ist so eingestellt, daß ein IBM-Monitor funktioniert. Bei kompatiblen Monitoren kann allerdings, aufgrund der verschiedenen Horizontal Synchronfrequenzen auftreten, daß der Monitor nicht synchronisiert. Sollten Sie diese Frequenz am Monitor nicht per Drehpotentiometer verändern können, besteht keine Möglichkeit den Monitor anzuschließen.

ST1 verwendet man intern für Messzwecke oder zum Anschluß der alten HCOPY/MAUS Baugruppe.

8. Anwendungsbeispiele

8.1. Grafikzeichen direkt eingeben:

Nach dem Einschalten erscheint (mit CPU Z80) das Flomon-Grundmenü. Drückt man nun CTRL-C gefolgt von ESC ESC G, so wird auf den Grafik-Modus umgeschaltet. Nun ist der Grafikprozessor bereit, Grafikbefehle aufzunehmen. Die Eingaben sind jetzt nicht mehr sichtbar.

Beispiel: Drücken der Taste 'Z', der gesamte Bildschirm gelöscht, es ist alles dunkel und auch kein Cursor zu sehen. Durch Eingabe von M 100 100 wird der Anfangspunkt mit den Koordinaten x=100, y=100 festgelegt. Danach zeichnen wir ein Rechteck. Der Befehl hierzu lautet: R30 30 (Rechteck mit 30 x 30 Punkten, aber aufgepaßt, weil der Bildschirm 256 x 512 Punkte hat, wird dieses Rechteck kein Quadrat! Um ein Quadrat zu erhalten muß eine Seite dementsprechend geteilt werden : R30 15)

Beim Einschalten des Grafikmodus wird automatisch die Seite 0 angewählt. Nun ist es aber auch möglich, eine andere der 4 Seiten auszuwählen. Der Befehl P n cr macht dies möglich, wobei n der Parameter zur Festlegung der Schreib- und Leseseite ist: (n = Schreibseite x 4 + Leseseite).
P 10 cr (cr=RETURN) wählt also die Seite 2 als Schreib- und Leseseite aus. Will man verschiedene Seiten der Reihe nach anzeigen, verwendet man den X - Befehl. X n cr ist die Befehlssyntax, wobei n die Zeitdauer der Anzeige darstellt (n= Multiplikator * 20 ms).

Eine kleine 'Programm' - Sequenz:
Achten Sie bitte darauf, daß die Grafikbefehle in Großbuchstaben eingegeben und mit RETURN abgeschlossen werden müssen.

Nach dem Einschalten des Computers gehen Sie wie folgt vor:

```
                CTRL-C drücken,
          dann ESC ESC G  (Grafikmodus ein)
(Vorsicht, jetzt sehen Sie nicht mehr, was Sie eintippen.)
                Z          (Bildschirm löschen)
                M100 100   (Anfangskoordinaten)
                R 20 20    (Rechteck x=20, y=20 zeichnen)
                P 5        (Seite 1 auswählen)
                R 30 30    (Rechteck x=30, y=30 zeichnen)
                P 10       (Seite 2 auswählen)
                R 40 40    (Rechteck x=40, y=40 zeichnen)
                P 15       (Seite 3 auswählen)
                R 50 50    (Rechteck x=50, y=50 zeichnen)
                X 20       (Seiten 0...3 zyklisch anzeigen)
```

Programmbeschreibung:

Nachdem mit M100 100 ein Anfangspunkt festgelegt wurde, wird ein Rechteck auf die Seite 0 gezeichnet. Nun wird mit dem P - Befehl die nächste Seite angewählt und ein größeres Rechteck konstruiert. Zum Schluß wird mit X 20 ein zyklisches Anzeigen aller Seiten gestartet.
Gestoppt werden kann dieser Vorgang durch Eingabe von X 0. Es wird bei der augenblicklich angezeigten Seite angehalten.

8.2. Oben beschriebenes Beispiel als Basic-Programm:

Die Grafikbefehle werden hier durch BASIC übertragen. Mit CHR$(27) wird die Taste ESC dargestellt. Achten Sie auch hier darauf, daß Grafikbefehle nur in Großbuchstaben eingeben werden.

```
 1  Print CHR$(27);CHR$(27);"G"
 2  Print "Z;P0"
 3  Print "M100 100"
 4  Print "R20 20"
 5  Print "P 5"
 6  Print "R30 30"
 7  Print "P 10"
 8  Print "R40 40"
 9  Print "P 15"
10  Print "R50 50"
11  Print "X20"
```

8.3. Die gleiche Routine in TURBO PASCAL.

```
program test;
BEGIN
    writeln (#27,#27,'G');
    writeln ('Z;P0');
    writeln ('M100 100');
    writeln ('R20 20');
    writeln ('P 5');
    writeln ('R30 30');
    writeln ('P 10');
    writeln ('R40 40');
    writeln ('P 15');
    writeln ('R50 50');
    writeln ('X 20');
END.
```

Nach Starten des Programmes werden wie auch im BASIC - Programm alle vier Seiten der Reihe nach angezeigt. Da auf jeder Seite ein Rechteck verschiedener Größe abgebildet ist, bekommt man den Eindruck, ein wachsendes Rechteck vor sich zu sehen.
Weitere Befehle finden Sie im Buch 'Rechner modular' beschrieben, dessen Bestellnummer der Einführung zu entnehmen ist.

8.4 Kleines Hardcopyprogramm unter CP/M 2.2

Die nachfolgende Hardcopyroutine ist für den Einsatz unter CP/M 2.2 gedacht, als nachträglich zu ladende Hilfsroutine für EPSON FX80-Drucker mit paralleler Schnittstelle.
Sie belegt den RAM- Speicher oberhalb von CP/M ab der Adresse F900h. Das Ende des Programmes inclusive Bufferbereich ist bei FC42h. Der FLOMON- Zeiger FREEMEM (F031h) wird nicht beachtet bzw. aktualisiert.
Die SER- Hilfsroutine kann dazugeladen werden, wenn sie im verbleibenden RAM- Bereich zwischen FLOMON- Ende (dorthin zeigt der Inhalt von FREEMEM) und F8FFh Platz findet.

```
                                .Z80
0000'                           cseg

                                ;
                        ;*****************************************************
                        ;* Programm für Hardcopy                             *
                        ;* Starten mit GDPHCOP                               *
                        ;* A C H T U N G : Bei ältrere GDP 64k als r8 sind Änderungen *
                        ;*                 auf der GDP erforderlich.         *
                        ;* Ralf Röttgen              22 Dezember 1987        *
                        ;*****************************************************

0000'  11 001A'                 ld de,Text      ; Statuszeile ausgeben
0003'  0E 09                    ld c,09h        ; Kennung fuer "Print String"
0005'  CD 0005                  call 0005h
0008'  21 007B'                 ld hl,lo        ; Startadresse
000B'  11 F900                  ld de,0f900h    ; Zieladresse
000E'  01 0342                  ld bc,hi-lo     ; Zahl der zu kopierenden Bytes
0011'  ED B0                    ldir            ; Programm kopieren
0013'  21 F900                  ld hl,Check     ; Adresse der Check-Routine
0016'  22 EA0A                  ld (0ea0ah),hl  ; Consol-Input umlenken (60 k CP/M)
0019'  C9                       ret

001A'  1B 1B 47         Text:   db 1bh,1bh,'G'
001D'  50 30 0D 4C              db 'P0',0dh,'L0 0 511 0 511 12 0 12',0dh
0021'  30 20 30 20
0025'  35 31 31 20
0029'  30 20 35 31
002D'  31 20 31 32
0031'  20 30 20 31
0035'  32 0D
0037'  4D 34 33 30              db 'M430 2',0dh,'BHardcopy ^§',0dh
003B'  20 32 0D 42
003F'  48 61 72 64
0043'  63 6F 70 79
0047'  20 5E 40 0D
```

```
004B'  50 35 0D 4C                db 'P5',0dh,'LO 0 511 0 511 12 0 12',0dh
004F'  30 20 30 20
0053'  35 31 31 20
0057'  30 20 35 31
005B'  31 20 31 32
005F'  20 30 20 31
0063'  32 0D
0065'  4D 34 33 30                db 'M430 2',0dh,'BHardcopy ^S',0dh
0069'  20 32 0D 42
006D'  48 61 72 64
0071'  63 6F 70 79
0075'  20 5E 40 0D
0079'  41 24                      db 'A$'

007B'                       lo:

                                  .Phase 0f900h

F900   CD F003              Check:  call 0f003h       ; eingegebenes Zeichen laden
F903   FE 00                       cp 0             ; Control 0 als Startzeichen
F905   CA F90B                     jp z,Hcopy       ; Hardcopy ausfuehren
F908   E6 7F                       and 7fh          ; Bit 7 loeschen
F90A   C9                          ret

F90B   21 0000              HCopy:  ld hl,0          ; H+L-Register loeschen
F90E   39                           add hl,sp        ; Stackpointer nach H+L kopiern
F90F   22 FA0F                      ld (OldStack),hl ; alten Stackpointer retten
F912   31 FC41                      ld sp,Stack      ; Stackpointer laden
F915   CD F937                      Call InitRX80    ; Initialisierung des Druckers
F918   3E 20                        ld a,32          ; Zahl der Druckzeilen
                                                     ; (Punkte pro Zeile / 8)

F91A   C5                   Loop1:  push bc
F91B   4F                           ld c,a           ; Zeilenanzahl übernehmen
F91C   06 00                        ld b,0           ; GDP-Zeile =0
F91E   F5                           push af          ; Widerholungsfaktor sichern
F91F   E5                           push hl
F920   CD F96B                      Call GetLint     ; eine Druckzeile abtasten
F923   CD F93D                      Call InitLine    ; Drucker fuer die Ausgabe
                                                     ; dieser Zeile initialisieren
F926   CD F9E7                      Call PrtLine     ; Zeilenpuffer ausgeben
F929   E1                           pop hl
F92A   F1                           pop af           ; Wiederholungsfaktor laden
F92B   C1                           pop bc
F92C   3D                           dec a            ; Druckzeile a um 1 erhöhen
F92D   C2 F91A                      jp nz,Loop1

F930   2A FA0F                      ld hl,(OldStack) ; H+L mit altem Stackpointer laden
F933   F9                           ld sp,hl         ; Stackpointer restaurieren
F934   C3 F900                      jp Check         ; Eingabe des naechsten Zeichens
```

```
                                ; ***************************************
                                ; Initialisierung des Druckers
                                ; ***************************************
                                ;
F937   21 F94E          InitRX80: ld hl,InitTab     ; Adresse der Tabelle mit den
                                                    ; Daten zur Initialisierung
F93A   C3 F940                    jp Loop
F93D   21 F959          InitLine: ld hl,InitlTab    ; Initialisierung fuer eine Zeile
F940   4E               Loop:     ld c,(hl)         ; Byte laden
F941   E5                         push hl           ; Register sichern
F942   CD F00F                    call 0f00fh       ; Byte ausgeben
F945   E1                         pop hl            ; Register restaurieren
F946   23                         inc hl            ; Adresse erhoehen
F947   3E FF                      ld a,0ffh         ; Akku loeschen
F949   BE                         cp (hl)           ; Ende-Marke ?
F94A   C2 F940                    jp nz,Loop        ; naechstes Byte ausgeben
F94D   C9                         ret               ; Ende der Initialisierung

F94E   1B 40            InitTab:  db 1bh,'§'        ; Drucker normieren
F950   0D 0A 0A 0A                db 0dh,0ah,0ah,0ah,0ah ; oberer Rand
F954   0A
F955   1B 33 17                   db 1bh,'3',23     ; Zeilabstand auf 24/216 Zoll setzen
F958   FF                         db 0ffh           ; Offh als Ende-Marke
F959   20 20 20 20       InitlTab: db '            ' ; linker Rand
F95D   20 20 20 20
F961   20 20 20 20
F965   1B 2A 01 00                db 1bh,'*',1,0,2  ; 8 Punkt-Bitmuster mit doppelter Dichte
F969   02
F96A   FF                         db 0ffh           ; Offh als Ende-Marke

                                ; ***************************************
                                ; 8 Spalten des Bildschirms abtasten
                                ; ***************************************
                                ;
F96B   78               GetLint:  ld a,b            ; GDP-Zeile b = c * 8
F96C   C6 08                      add a,8           ; a um 8 erhöhen
F96E   47                         ld b,a
F96F   0D                         dec c             ; Zähler um 1 erniedrigen
F970   79                         ld a,c            ; wenn c = 0 Schleife c-mal
F971   D6 00                      sub 0
F973   CA F979                    jp z,Getsub       ; durchlaufen springe nach
F976   C3 F96B                    jp getlint
F979   21 FA11          Getsub:   ld hl,Buffer      ; erste Speicherzelle nach HL
F97C   0E 00                      ld c,0            ; Zeilenzähler auf 0
F97E   11 0000          Getline:  ld de,0           ; Bytezähler auf 0
F981   05                         dec b             ; Zeile um 1 erniedrigen
F982   CD F9E0          NZEILE:   call warten       ; prüfen ob GDP bereit
F985   78                         ld a,b
F986   D3 7B                      out (7bh),a       ; Zeilenzahl nach Port $7b
F988   CD F9E0          NBYTE:    call warten       ; prüfen ob GDP fertig
F988   7B                         ld a,e
F98C   D3 79                      out (79h),a       ; Bytezahl niederwertig
F98E   CD F9E0                    call warten
F991   7A                         ld a,d
F992   D3 78                      out (78h),a       ; Bytezahl höherwertig
F994   CD F9E0                    call warten
F997   3E 0F                      ld a,0fh
F999   D3 70                      out (70h),a       ; beim nächsten Zugr. auslesen
F99B   CD F9E0                    call warten
```

```
F99E  DB 60                   in a,(60h)       ; Speicher auslesen
F9A0  D5                      push de
F9A1  C5                      push bc
F9A2  CD F9BC                 call Sortiere    ; Speicher sortieren
F9A5  C1                      pop bc
F9A6  D1                      pop de
F9A7  E5                      push hl
F9A8  EB                      ex de,hl         ; vertausche DE mit HL
F9A9  11 0008                 ld de,8          ; lade 8 nach DE
F9AC  19                      add hl,de        ; adddiere HL mit DE
F9AD  EB                      ex de,hl         ; vertausche HL mit DE
F9AE  E1                      pop hl
F9AF  7A                      ld a,d           ; wenn 512 Bit = eine Zeile
F9B0  D6 02                   sub 2h           ; noch nicht ausgelesen
F9B2  20 D4                   jr nz,NBYTE      ; springe nach
F9B4  79                      ld a,c           ; wenn 8 Zeilen ausgelesen
F9B5  D6 08                   sub 8            ; springe zurück ins
F9B7  C8                      ret z            ; Hauptprogramm
F9B8  0C                      inc c            ; erhöhe Anzahl ausgelesener Zeilen
F9B9  C3 F97E                 jp Getline
F9BC  5F              Sortiere: ld e,a         ; ausgelesener Wert nach e
F9BD  16 00                   ld d,0           ; Zähler auf 0
F9BF  14              Sort:   inc d            ; Zähler um 1 erhöhen
F9C0  7B                      ld a,e           ; aktueller Wert nach a
F9C1  07                      rlca             ; rotiere Akku links cyclisch
F9C2  5F                      ld e,a           ; rotiertes Byte nach e
F9C3  7E                      ld a,(hl)        ; Speicherinhalt nach a
F9C4  17                      rla              ; rotiere Akku links durch Übertragb.
F9C5  77                      ld (hl),a        ; neuer Speicherinhalt nach HL
F9C6  23                      inc hl           ; HL = Speicherzelle incrementieren
F9C7  3E 08                   ld a,8
F9C9  92                      sub d            ; wenn d = 8

F9CA  C2 F9BF                 jp nz,Sort       ; neues a von GDP lesen
F9CD  01 FA11                 ld bc,Buffer     ; HL = Buffer + 512 (200h)
F9D0  7C                      ld a,h           ; wenn 512-ten Speicher erreicht
F9D1  D6 02                   sub 2            ; neues GDP-Byte laden und
F9D3  90                      sub b            ; Speicherzelle auf anfangswert
F9D4  C2 F9DF                 jp nz,Sortrset
F9D7  7D                      ld a,l           ; wenn niederwertige Bytes gleich
F9D8  91                      sub c            ; dann springe nicht nach
F9D9  C2 F9DF                 jp nz,Sortrset
F9DC  21 FA11                 ld hl,Buffer     ; Speicherzelle auf Anfangswert
F9DF  C9              Sortrset: ret
F9E0  DB 70           warten: in a,(70h)       ; Port 70h auslesen
F9E2  E6 04                   and 4            ; wenn 4 Bit = 1 dann
F9E4  C0                      ret nz           ; springe zurück
F9E5  18 F9                   jr warten
```

```
                                ; ************************************
                                ; Ausgabe des Zeilenpuffers
                                ; ************************************
                                ;
F9E7   E5              PrtLine: push hl
F9E8   C5                       push bc
F9E9   D5                       push de
F9EA   21 FA11         Sortiere2: ld hl,Buffer      ; erste Speicherzelle nach HL
F9ED   11 0200                  ld de,512           ; erste Speicherzelle nach DE
F9F0   7E              sort2:   ld a,(hl)           ; Speicher nach a
F9F1   2F                       cpl                 ; Akku negieren
F9F2   E5                       push hl             ; Register sichern
F9F3   C5                       push bc
F9F4   4F                       ld c,a              ; auszudruckender Wert nach c
F9F5   CD F00F                  call 0f00fh         ; Byte ausgeben
F9F8   C1                       pop bc              ; Register restaurieren
F9F9   E1                       pop hl
F9FA   23                       inc hl              ; Adresse um 1 erhöhen
F9FB   1B                       dec de              ; Zähler um 1 erniedrigen
F9FC   7B                       ld a,e              ; wenn e <> 0 und d <> 0 dann
F9FD   B2                       or d                ; springe nach
F9FE   C2 F9F0                  jp nz,sort2         ; sort2
FA01   0E 0D                    ld c,0dh            ; Carriage Return ausgeben
FA03   CD F00F                  call 0f00fh         ; Byte ausgeben
FA06   0E 0A                    ld c,0ah            ; Line Feed ausgeben
FA08   CD F00F                  call 0f00fh         ; Byte ausgeben
FA0B   D1                       pop de
FA0C   C1                       pop bc
FA0D   E1                       pop hl
FA0E   C9                       ret
FA0F   0000            OldStack: dw 0               ; Wert des Stackpointers
FA11                   Buffer:  ds 540              ; Puffer fuer eine Zeile
FC2D                            ds 20               ; Platz fuer Stack
FC41   00              Stack:   db 0

                                .Dephase

03BD'                  hi:

                                end
```

Hardcopyprogramm für 680xx

Diese Hardcopyroutine für die Prozessoren 680xx läuft unter JADOS, sowie im RDK- Grundprogramm. Unter CP/M68k können damit keine Hardcopys angefertigt werden.
Zum Betrieb mit der CPU68k (Prozessor 68008) müssen beispielsweise folgende Voraussetzungen erfüllt werden:

- Jumper JMP2 auf der CPU-Platine muß gesteckt werden, da INT und NMI gleichzeitig aktiviert werden müssen.
- Die Rechnerkonfiguration muss aus BANK/BOOT, mind. 256kB RAM ab Adr.00000 und dem Grundprogramm auf der Bank E bestehen.
- Die Ausgabe erfolgt über die parallele Schnittstelle (Port 40h/41h) an einen EPSON- Drucker.
- Auf der Bank E müssen 32k RAM (4 * 6264) eingesteckt werden.
- Das Programm benötigt als Buffer 16kByte RAM ab der Adresse $2A000.

Nachdem das Programm eingegeben ist, wird es assembliert und als Bibliothekeintrag aufgerufen.

```
*-------------------------------------------------------------------------------*
*  ( G D P - Speicher auslesen Vers. SCC )                                      *
*    Quelltext assemblieren und als Bibliothek im Speicher aufrufen.            *
*    Bildschirmspeicher der GDP mit dem MREE-Signal vom Register 74LS374        *
*    mit Port-Adresse $FFFFFF60 auslesen in Speicher mit 16 KByte               *
*    ab Adresse $2A000 ablegen.                                                 *
*    Beim Invertieren des Speichers oder Ausgabe auf den Bildschirm,            *
*    werden Langworte vom Speicher eingelesen und Vektoren ausgegeben.          *
*    Alle verwendeten IO-Adressen werden indirekt adressiert,                   *
*    damit ist das Programm auf allen CPU'S lauffaehig.                         *
*-------------------------------------------------------------------------------*
 Org $0
 OFFSET $ED800
*-------------------------------------------------------------------------------*
*:              B I B L I O T H E K S K O P F                                  *
*-------------------------------------------------------------------------------*
HEADINT:
 DC.L    $55AA0180          *` Kennung Bibliothek
 DC.B    'HCOP-INT'         *` Name
 DC.L    INITTAST-HEADINT   *` Programmstart
 DC.L    HCENDE-HEADINT     *` Laenge
 DC.B    1
 DC.B    0,0,0
 DC.L    0,0

*-------------------------------------------------------------------------------*
*:              P O R T A D R E S S E N / B U F F E R                          *
*-------------------------------------------------------------------------------*
GDPPRD    EQU $FFFFFF60    *` GDP - PORT 74LS374 lesen
GDP       EQU $FFFFFF70    *` GDP - PORT
INTVEC    EQU $0C          *` INT - LVL.5 Grundprogr. (A5)
GDPREGBF  EQU $188         *` GDP0 ... GDP15 save (A5)
BUFHC     EQU $E0          *` Speicher fuer Hardcopy auf Drucker
BUF       EQU $EA          *` Hilfsspeicher fuer Sortierung 8 Byte
CI        EQU 12           *` RDK-GDP Zeichen einlesen
CLRSCREEN EQU 20           *` RDK-GDP Bildschirm loeschen + Cursor
LO        EQU 22           *` RDK-GDP Zeichen an Drucker
CMD       EQU 26           *` RDK-GDP Befehl an GDP
NEWPAGE   EQU 27           *` RDK-GDP Schreib- und leseseite setzen
CO2       EQU 33           *` RDK-GDP Zeichen Ausgabe
CRT       EQU 49           *` RDK-GDP Ausgabe auf Bildschirm
GETBASIS  EQU 89           *` RDK-GDP Startadresse des Grundprogramms
SETA5     EQU 91           *` RDK-GDP RAM-Zeiger A5
*          >> Verwendeter Speicherbereich, evtl. an eigenes RAM anpassen  <<
BILDSP    EQU $2A000       *` Start Bildspeicher 16 kByte gross

*-------------------------------------------------------------------------------*
*:              I N T E R R U P T   I N I T I A L I S I E R E N                *
*-------------------------------------------------------------------------------*
INITTAST:
 MOVEQ    #SETA5,D7              *` RAM-Adresse RDK-Grundprogr. in A5
 TRAP     #1
 BSR.S    HCHILFI                *` Hilfstexte ausgeben
 LEA      TASTINT(PC),A0         *` Programmstart nach A0
 MOVE.W   #$4EF9,D0              *` <4EF9> = Maschinencode <JMP>
 MOVE.W   D0,INTVEC(A5)          *' auf "INTLV7:" (Ltg. INT + NMI)
 MOVE.L   A0,INTVEC+2(A5)        *` Sprungadr. Programmstart
 RTS
```

```
HCHILFI:
 MOVEQ      #CLRSCREEN,D7          *` Bildschirm loeschen
 TRAP       #1
 MOVEQ      #CRT,D7                *` Ausgabe auf Bildschirm
 TRAP       #1
 LEA        HILFTEXT(PC),A0        *` tEXTSTART NACH A0
HILFAUS:    .
 MOVE.B     (A0)+,D0
 BEQ.S      HILFEND
 MOVEQ      #C02,D7                *` und ausgeben
 TRAP       #1
 BRA.S      HILFAUS
HILFEND:
 RTS

HILFTEXT:
 DC.B  '>>>  HCOP-INT ist nun initialisiert, nach jedem Reset bitte neu <<<'
 DC.B  $A,$D,$A,'Start mit LOW-Impuls auf INT-Leitung, BUS PIN 32',$A,$D,$A
 DC.B  '> B  < auf Tastatur = Ausgabe Speicher auf Bildschirm',$A,$D
 DC.B  '> I  < auf Tastatur = Speicherinhalt invertieren',$A,$D
 DC.B  '> L  < auf Tastatur = Bilschirm loeschen',$A,$D
 DC.B  '> H  < auf Tastatur = Hcopy auf Drucker',$A,$D
 DC.B  '> CR < auf Tastatur = Ruecksprung in Programm',0
 DS       0

*-------------------------------------------------------------------------*
*:          H A U P T P R O G R A M M  -  R E G I S T E R  R E T T E N    *
*-------------------------------------------------------------------------*
TASTINT:            *` >>> S T A R T , wird durch "INTLV5:" aufgerufen
 MOVEM.L D0-D7/A0-A6,-(A7)      *` Alle Register auf Stack
 MOVEQ   #SETA5,D7              *` RAM-Adresse RDK-Grundprogramm in A5
 TRAP    #1
 MOVEQ   #GETBASIS,D7           *` Grundprogramm-Start nach A0
 TRAP    #1
 MOVE    $416(A0),D0            *` CPU - Kennung in Wortgroesse
 MOVE    D0,D6                  *` D6 = Kennung der CPU
 MULS    #GDPPRD,D0             *` IO-Adresse multiplizieren
 EXG     D0,A1                  *` A1 = Adresse 74LS374 auslesen
 MOVE    D6,D0
 MULS    #GDP,D0
 EXG     D0,A2                  *` A2 = Basisadresse GDP

 MOVEA.L A2,A0                  *` GDP-Register retten,
 ADDA    D6,A0                  *` aber erstes Register nicht
 LEA     GDPREGBF(A5),A3
 MOVEQ   #15-4-1,D3
 GDPSAVE:
   MOVE.B     (A0)+,(A3)+
   DBRA.S     D3,GDPSAVE

 BSR     BSPAUSL            ***` Unterprogramm GDP-Bildspeicher auslesen
 BRA.S   TPXXX              ***` Unterprogramm Tastatur abfragen
TPEEE:

 MOVEQ   #GETBASIS,D7           *` Grundprogramm-Start nach A0
 TRAP    #1
 MOVE    $416(A0),D0            *` CPU - Kennung in Wortgroesse
 MOVE    D0,D6                  *` D6 = Kennung der CPU
 MULS    #GDPPRD,D0             *` IO-Adresse multiplizieren
 EXG     D0,A1
 MOVE    D6,D0
```

```
 MULS     #GDP,D0
 EXG      D0,A2                *` A2 = Basisadresse GDP
 MOVEA.L  A2,A0                *` GDP-Register zurueck,
 ADDA     D6,A0                *` aber erstes Register nicht
 LEA      GDPREGBF(A5),A3
 MOVEQ    #15-4-1,D3
GDPLOAD:
   MOVE.B    (A3)+,(A0)+
   DBRA.S    D3,GDPLOAD
 MOVEM.L  (A7)+,D0-D7/A0-A6    *` Register zurueck
 RTE                           *` und wieder ins Programm

*-------------------------------------------------------------------------------*
*:                  T A S T A T U R   A B F R A G E N                          *
*-------------------------------------------------------------------------------*
TPXXX:
 MOVEQ      #C1,D7          *` Neu einlesen
 TRAP       #1
 CMP.B      #$0D,D0         *` Falls <CR>, dann Ende
 BEQ.S      TPEEE
ZKLGR:
 CMPI.B     #'a',D0
 BHS.S      ZKLEIN
 BRA.S      ZEINGR
ZKLEIN:
 CMPI.B     #'z',D0
 BLS.S      ZKORIG
 BRA.S      ZEINGR
ZKORIG:
 SUBI.B     #$20,D0         *` Wenn klein, -$20, dann gross
ZEINGR:
 CMP.B      #'H',D0         *` H = Hardcopy auf Drucker
 BNE.S      TPB
 BSR        HCOPYD
 BRA.S      TPXXX
TPB:
 CMP.B      #'B',D0         *` B = Hcopy auf Bildschirm
 BNE.S      TPI
 BSR        CRTAUS
 BRA.S      TPXXX
TPI:
 CMP.B      #'I',D0         *` I = Hcopy im Speicher invertieren
 BNE.S      TPL
 BSR        HCINVERS
 BRA.S      TPXXX
TPL:
 CMP.B      #'L',D0         *` L = Bildschirm loeschen
 BNE.S      TPXXX
 CLR        D0              *` Schreib-
 CLR        D1              *` und Leseseite = 0
 MOVEq      #NEWPAGE,D7     *` SETZEN
 TRAP       #1
 MOVE.B     #$4,D0          *` Befehl GDP loeschen
 MOVEQ      #CMD,D7
 TRAP       #1
 BRA.S      TPXXX
```

```
*------------------------------------------------------------------------*
*:              G D P  -  B I L D S P E I C H E R   A U S L E S E N     *
*------------------------------------------------------------------------*
*` X, Y - Register des GDP EF9366 (Lesen und Schreiben) ADR. = 68008
*` +-------------------------------+       Y = Zeilen   X = Punkte wagerecht
*` : 7 : 6 : 5 : 4 : 3 : 2 : 1 : 0 :  LSB  $FFFFFF7B    $FFFFFF79 ($000 - $0FF)
*` :-----------:-----------------------:
*` : X : X : X : X : X : X : X : X :  MSB  $FFFFFF7A    $FFFFFF78
*` +-------------------------------+       nicht        2.Bildhaelfte
*`                                        benutzt      Bit 0 = 1  ($100 - $0FF)
BSPAUSL:
 LEA     BILDSP,A3              *` Bildspeicherstart
 MOVE    #256-1,D2              *` Y-Koordinate (Zeilen)
         *`  >>> Routine mehr als 3 mal schneller als "MOVETO" und "CMP" <<<
 MOVE    D6,D7          *` CPU-Kennung nach D7 und       (MOVETO)
 MULS    #$B,D7         *` multiplizieren mit GDP Y-LSB Reg. :
 MOVEA.L A2,A4          *` IO-Adresse GDP nach A4            :
 ADDA.W  D7, A4         *` und Y-LSB Register addieren ($7B) :
NZEILE:
   BTST.B  #2,(A2)      *` Bit 2 pruefen, wenn 1,            :
   BEQ.S   NZEILE       *` dann GDP fertig, sonst warten     :
   MOVE.B  D2,(A4)      *` LSB Port Y ($7B)                  :
   SUBA.W  D6,A4        *` Zweimal CPU-Kennung               :
   SUBA.W  D6,A4        *` sbtrahieren, Port dann = ($79)    :
   CLR     D1           *` X-Koordinate
   MOVEQ   #64-1,D3     *` Bytezaehler (64 *8 = 512)
NBYTE:
     BTST.B  #2,(A2)    *` Bit 2 pruefen, wenn 1,        (MOVETO)
     BEQ.S   NBYTE      *` dann GDP fertig, sonst warten     :
     MOVE.B  D1,(A4)    *` LSB Port X ($79) (Positionieren)  :
     SUBA.W  D6,A4      *` CPU-Kennung subtrahieren = ($79) :
     ROR.W   #8,D1      *` Rotiere rechts 1 Byte             :
     MOVE.B  D1,(A4)    *` MSB Port X   ($78)                :
     ROL.W   #8,D1      *` Rotiere links wieder zurueck      :
     ADDA.W  D6,A4      *` CPU-Kennung addieren              :
     MOVE.B  #$F,(A2)   *` GDP - DMA Speicher auslesen     (CMD)
   GDPWO:
     BTST.B  #2,(A2)    *` Bit 2 pruefen wenn 1,             :
     BEQ.S   GDPWO      *` dann GDP fertig, sonst warten     :
     MOVE.B  (A1),D4    *` Register 74LS374 auslesen (D4)
     MOVE.B  D4,(A3)+   *` In Bildspeicher ablegen
     ADDQ    #8,D1      *` Naechstes Byte in Zeile
     DBRA.S  D3,NBYTE   *` Bis 64 Stueck (64 * ( = 512)
   ADDA.W  D6,A4      *` Zwei mal CPU-Kennung addieren,      :
   ADDA.W  D6,A4      *` Zeilen- Port dann wieder  = ($7B)   :
   DBRA.S  D2,NZEILE  *` Bis alle 256 Zeilen
RTS

*------------------------------------------------------------------------*
*:                S P E I C H E R   A U F   B I L D S C H I R M         *
*------------------------------------------------------------------------*
CRTAUS:
 MOVEQ     #GETBASIS,D7     *` Grundprogramm-Start nach A0
 TRAP      #1
 MOVE      $416(A0),D0      *` CPU-Kennung in Wortgroesse
 MOVE      D0,D6            *` D6 = Kennung der CPU
 MULS      #GDP,D0          *` IO-Adresse multiplizieren
 EXG       D0,A2            *` A2 = Bsisadresse GDP
 LEA       BILDSP,A3        *` Bildspeicherstart
 MOVE      #256-1,D2        *` Y-Koordinate (Zeilen)
```

```
        *` >>> Routine mehr als 3 mal schneller als "MOVETO" und "CMP"  <<<
 MOVE     D6,D7          *` CPU-Kennung nach D7 und          (MOVETO)
 MULS     #$B,D7         *` multiplizieren mit GDP Y-LSB Reg.
 MOVEA.L  A2,A4          *` IO-Adresse GDP nach A4
 ADDA.W   D7,A4          *` und y-lsb register addieren ($7B)
CRTZEILE:
 CLR      D1             *` X-Koordinate  (Zeilenanfang)
GDPW1:
   BTST.B  #2,(A2)       *` Bit 2 pruefen, wenn 1,
   BEQ.S   GDPW1         *` dann GDP fertig, sonst warten
   MOVE.B  D2,(A4)       *` LSB-Port Y ($7B)  (Zeile-Pos.)
   SUBA.W  D6,A4         *` Port minus 3 mal
   SUBA.W  D6,A4         *` CPU-Kennung = $7B -> $78
   SUBA.W  D6,A4
   ROR.W   #8,D1         *` Rotiere rechts ein Byte
   MOVE.B  D1,(A4)       *` MSB Port X  ($78)  (1.Bildhaelfte)
   ROL.W   #8,D1         *` Rotiere links wieder zurueck
   ADDA.W  D6,A4         *` LSB Port X  ($79) nach A4
   MOVEQ   #16-1,D3      *` Langwortzaehler (16 * 32 = 512)
CRTBYTE:
    MOVE.L    (A3)+,D4   *` Langwort (32 Bit) nach D4
   MOVEQ      #32-1,D5     *` 32 Punkte  (Langwort)
CRTBYT1:
       BTST    D5,D4        *` Bit-Test, Bit: D5, in D4
       BNE.S   CRTBYT2      *` Nicht gesetzt = kein Punkt
    GDPW2:
       BTST.B  #2,(A2)      *` Bit 2 pruefen wenn 1,            (MOVETO)
       BEQ.S   GDPW2        *` dann GPD fertig, sonst warten       :
       MOVE.B  D1,(A4)      *` LSB-Port X ($79) (Positionieren)    :
       MOVE.B  #$80,D0      *` Kurzvektor mit einem Punkt laden
       BRA.S   CRTBITPR
     CRTBITOA:
       BTST.B  D5,D4        *` Naechstes Bit testen
       BNE.S   CRTBITNO     *` Sprung wenn kein Punkt,
       ADDQ    #1,D1        *` sonst X Achse + 1 und
       ADDI    #$28,D0      *` Kurzvektor + 1 Punkt
       CMPI.B  #$F8,D0      *` bis max. 4 Punkte,
       BEQ.S   CRTBITKV     *` dann unbedingt Ausgabe
     CRTBITPR:
       DBRA.S  D5,CRTBITOA
     CRTBITNO:
       ADDQ    #1,D5        *` Bitzaehler korrigieren
     CRTBITKV:
       MOVE.B  D0,(A2)      *` 1 - 1 Punkte an GPD ausgeben
     CRTBYT2:
       ADDQ    #1,D1        *` Bit + 1
       DBRA.S  D5,CRTBYT1   *` 32 mal (Langwort)
     CMPI.W  #256,D1      *` zweite Bildhaelfte erreicht ?
     BNE.S   CRTNOMSB     *` dann MSB Port X setzen
   GDPW3:
       BTST.B  #2,(A2)      *` Bit 2 pruefen wenn 1,            (MOVETO)
       BEQ.S   GDPW3        *` dann GDP fertig,sonst warten        :
       SUBA.W  D6,A4        *` MSB Port X ($78) nach A4            :
       ROR.W   #8,D1        *` Rotiere rechts 1 Byte               :
       MOVE.B  D1,(A4)      *` MSB Port X (47B) (Positionieren)    :
       ROL.W   #8,D1        *` Rotiere links wieder zurueck        :
       ADDA.W  D6,A4        *` LSB Port X ($79) wieder nach A4     :
```

```
   CRTNOMSB:
     DBRA.S   D3,CRTBYTE   *` Naechstes Langwort in Zeile          :
    ADDA.W   D6,A4        *` Zwei mal CPU-Kennung addieren,       :
    ADDA.W   D6,A4         *` Zeilen-Port dann wieder  = ($78)     :
    DBRA.S   D2,CRTZEILE    *` Naechste Zeile
 RTS

*-------------------------------------------------------------------------*
*:          B I L D S P E I C H E R   I N V E R T I E R E N              *
*-------------------------------------------------------------------------*
HCINVERS:
 LEA     BILDSP,A0       *` Startadresse
 MOVE    #32*128-1,D7    *` Zaehler
HCINV1:
   MOVE.L    (A0),D0      *` Langwort in d0
   NOT.L     D0           *` invertieren
   MOVE.L    D0,(A0)+     *` und zurueck, A0 erhoehen
   DBRA.S    D7,HCINV1    *` 16 k MAL
 RTS

*-------------------------------------------------------------------------*
*:                  H C O P Y   A U F   D R U C K E R                    *
*-------------------------------------------------------------------------*
HCOPYD:                    *` D1,D2,D3,D4 nicht veraendern
 LEA    DZABST(PC),A1      *` Drucker Zeilenabstand
 BSR    DRSETZ             *` und ausfuehren

 CLR    D1                 *` D1 = Zaehler fuer Bytes in Bildspeicher
 MOVE   #32-1,D3           *` 32 * 8 = 256 Zeilen
KOPSP:
   LEA     DBITIM(PC),A1   *` Bit Image Graphik laden
   BSR     DRSETZ          *` und ausfuehren
   MOVE    #64-1,D2        *` 64 * 8 = 512 Punkte / Zeile
 KOPSP0:
    LEA   BUFHC+8(A5),A4   *` A4 = Hilfsbuffer fuer Hardcopy
    LEA   BUF (A5),A2      *` A2 = Hilfsbuffer
    LEA   BILDSP,A3        *` A3 = Bildspeicher Start
    ADDA  D1,A3            *` gelesene Bildpunkte addieren
    MOVEQ #8-1,D0          *` 8 Bytes fuer Block einlesen
  KOPSP1:
      MOVE.B   (A3),(A2)+  *` BAILDSPEICHERBYTE NACH A2
      ADDA     #64,A3       *` A3 + 64 = Bildpunkt naechste Zeile
      DBRA.S   D0,KOPSP1
    ADDQ  #1,D1            *` Bildpunktzaehler erhoehen
    BSR.S Sortieren        *` ruechwaerts abgelegt wird
      MOVEQ   #8-1,D5      *` 8 Bbyte an Drucker
  HCOPY:
     MOVE.B   (A4)+,D0     *` Hilfsspeicher nach do
     MOVEQ    #LO,D7       *` und an Drucker ausgeben
     TRAP     #1
     DBRA.S   D5,HCOPY
   DBRA.S  D2,KOPSP0
     ADDI     #448,D1      *` 448 Bildpunkte addieren = 7 Zeilen
     DBRA.S   D3,KOPSP
   LEA     DINORM(PC),A1   *` Drucker wieder in normalen
   BSR     DRSETZ          *` Modus setzen
   RTS
```

```
*`                BITS SORTIEREN UND ABLEGEN
SORTIEREN:
 CLR.L     D5          *` Block mit 8 * 8 Bit
 CLR.L     D6          *` Spaltenzaehler
SORT:
   CLR.L     D0        *` Register fuer HCOPY-BYTE
   LEA  BUF(A5),A2     *` Buffer nach A2
   CLR.L     D7        *` Zeilenzaehler
 LOAD:
     CLR.L   D4
     MOVE.B  (A2),D4   *` Byte nach D4
     MOVE.B  D4,D5
     ROR.L   #1,D4     *` Aufschieben und
     MOVE.B  D4,(A2)+  *` zurueckspeichern
     ANDI.B  #1,D5     *` Bit 1 markieren
     ROL.B   #1,D0     *` Platz fuer Bit 1
     ADD.B   D5,D0     *` Und Verknuepfen
     ADDQ    #1,D7     *` Bitzaehler
     CMPI.B  #8,D7     *` Byte fertig ?,
     BLT.S   LOAD
   NOT.B   D0          *` Negieren fuer Drucker
   MOVE.B  D0,-(A4)    *` In Speicher ablegen
   ADDQ    #1,D6       *` Spaltenzaehler
   CMPI.B  #8,D6       *` Alle Spalten fertig ?
   BLT.S   SORT
RTS

*:              STEUERZEICHEN AN DRUCKER
DRSETZ:                         *` Schicke eine Zeichenserie an den
 CLR            D4
 MOVE.B         (A1)+,D4        *` Drucker, Adresse in A1, erstes Zeichen
 SUBQ.B         #1,D4           *` = Anzahl der Zeichen in der Serie -1
DRSETZ1:
   MOVE.B       (A1)+,D0        *` Nach D0 und
   MOVEQ        #LO,D7          *` ausgeben
   TRAP         #1
   DBRA.S       D4,DRSETZ1      *` bis alle
 RTS
*`       >>  Je nach Drucker veraendern <<
DZABST:   DC.B 4,$D,$1B, '3',23                    *` <CR>, Vorschub 23/216
DBITIM:   DC.B 9,$A,$D,$1B, 'l',8,$1B,'L',0,2      *` Rand 8, Doppelte Dichte
DINORM:   DC.B 10,$D,$A,$A,$A,$1B,'§',$7,$7,$7,$A  *` Vorschub, Init, Bell

HCENDE:
 END
```

```
*------------------------------------------------------------------------*
*' ( S C R O L L  Vor-Rück  8-Zeilen  V.2.1 )                           *
*                                                                        *
*' Test-Programm für Hardware - Scroll.                                 *
*' Es wird ein Text der auf Adresse $10000 ( od. Textstart auf "STXTXT" ) *
*' steht, mit der Hardware-Scoll-Erweiterung (BU8702) der GDP-Karte auf  *
*' dem Bildschirm ausgegeben. Der Zeilenabstand beträgt 8 Linien.        *
*                                                                        *
*' Mit der Variablen "SPEED" ist die Scroll-Geschwindigkeit einstellbar, *
*' wenn die Variable "ZEILEN" auf 8 Linien Vorschub steht.               *
*' Steht die Variable "ZEILEN" auf 2 oder 4 Linien Vorschub, sollte      *
*' "SPEED" auf 0 stehen, es wird dann ein Softscroll ausgeführt.         *
*                                                                        *
*' <CTRL+E> (Pfeil oben) = Rückwärts, <CTRL+X> (Pfeil unten) = Vorwärts, *
*' <SPACE> = Stop, Start und <ESC> = Abbruch werden ausgewertet.         *
*' Deutsche Sonderzeichen werden berücksichtigt.                         *
*                                                                        *
*' (C) Hans-Dieter Bulwien     6.08.87    V.2.1                         *
*------------------------------------------------------------------------*

ORG      $2A000

*------------------------------------------------------------------------*
*ö            B U F F E R  /  P O R T A D R E S S E N                    *
*------------------------------------------------------------------------*
TEXT     EQU $10000          *' Text - Start - Adresse
STXTXT   EQU $36             *' RDK Zeiger auf aktuellen Textstart
SPEED    EQU 0               *' Scroll-Geschwindigkeit   < 0,    4 - 30 >
ZEILEN   EQU 2               *' Zeilenvorschub / Scroll  < 2, 4,   8 >
CPU      EQU $1              *' 68008 = 1, 68000/10 = 2, 68020 = 4
SCROLL   EQU $FFFFFF61*CPU   *' GDP - Port für Scroll-Addition
KEYDAT   EQU $FFFFFF68*CPU   *' KEY - Port Daten
KEYRES   EQU $FFFFFF69*CPU   *' KEY - Port Reset
GDP      EQU $FFFFFF70*CPU   *' GDP - Port

*------------------------------------------------------------------------*
*ö            H A R D W A R E  -  S C R O L L  -  R O U T I N E          *
*------------------------------------------------------------------------*
*' ------------ Voreinstellungen:
HS:                          * >>>   S T A R T   <<<
 MOVEM.L D0-D7/A0-A4,-(A7)  *' Register auf Stack retten
 LEA     GDP,A3             *' A3 = I/O Adresse des GDP-Port
* LEA     TEXT,A4            *' A4 = Text-Start  -- oder --
 MOVEA.L STXTXT(A5),A4      *' A4 = Zeiger auf aktuellen Textstart
 MOVEA.L A4,A2              *' A2 = Pointer für Startadresse
HWSCR:
 BTST.B  #2,(A3)           *' Bit 2 prüfen, wenn 1,
 BEQ.S   HWSCR             *' dann GDP bereit, sonst warten
 MOVE.B  #$04,(A3)         *' GDP - Bildschirm löschen
```

```
HWSCR1:
 BTST.B  #2,(A3)          *' Bit 2 prüfen, wenn 1,
 BEQ.S   HWSCR1           *' dann GDP bereit, sonst warten
 MOVE.B  #$11,CPU*$3(A3)  *' GDP - Schriftgröße
 MOVE.B  #$00,SCROLL      *' Scroll-Register auf 0
 CLR     D6
 CLR     D5               *' Wenn 0, dann noch nicht scrollen
 MOVE    #248,D2          *' Startpunkt Y-Achse
 BSR     HSSETY           *' und Position an GDP
 BSR     HSSETX0          *' X-Register GDP auf 0
*' ------------- Zeichen ausgeben:
HSAUS:
 TST.B   D5               *' Wenn noch 1. Seite,
 BEQ.S   HSNOWAIT         *' dann nicht warten
 MOVE    #SPEED*100,D4    *' Warteschleife zur Beinflussung
HSAUSW1:                  *' der Ausgabe-Geschwindigkeit
 NOP
 DBRA.S  D4,HSAUSW1
 HSNOWAIT:
 MOVE.B  (A4)+,D0         *' Zeichen aus Text-Buffer nach D0
 TST.B   D0               *' Wenn 0-Byte,
 BEQ.S   HSENDV           *' dann Textende erreicht
 CMPI.B  #$0D,D0          *' Wenn <CR>, dann Zeit
 BEQ.S   HS85Z            *' für 85 Zeichen / Zeile abwarten
 BPL.S   HSAUSW           *' Daten-Byte <$7F ?, sonst
 BSR     HSGERMAN         *' Deutsche Sonderzeichen
 SUBQ    #1,D7            *' Zähler Zeichen / Zeile -1
 BRA.S   HSAUS
HSAUSW:
 BTST.B  #2,(A3)          *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSAUSW           *' dann GDP bereit, sonst warten
 MOVE.B  D0,(A3)          *' ASCII - Zeichen an GDP
 SUBQ    #1,D7            *' Zähler Zeichen / Zeile -1
 BRA.S   HSAUS            *' und nächstes Zeichen holen
*' ------------- Evtl. Zeit für 85 Zeichen / Zeile abwarten:
HS85Z:                    *' ( Kann bei Soft-Scroll entfallen )
 TST.B   D5               *' Wenn noch 1. Seite, dann sofort
 BEQ.S   HSSCR1           *' nächste Zeile positionieren
 TST.B   D7               *' Wenn 0, dann Zeile voll
 BEQ.S   HSSCR1
HS85Z1:
 MOVE    #SPEED*100,D4    *' Warteschleife zur Beinflussung
HS85Z2:                   *' der Ausgabe-Geschwindigkeit
 NOP
 DBRA.S  D4,HS85Z2
 DBRA.S  D7,HS85Z1
 BRA.S   HSSCR1           *' Nächste Zeile positionieren

*' ------------- Ausgabe beenden:
HSENDV:
 MOVE.B  #2,D5            *' Kennung für Textende ereicht
 BRA.S   HSEND
HSENDR:
 MOVE.B  #3,D5            *' Kennung wieder Textanfang
HSEND:
 MOVE.B  KEYDAT,D0        *' Tastatur-Port einlesen, wenn 0,
 TST.B   D0               *' dann kein Zeichen eingegeben
 BMI.S   HSEND            *' Zurück und warten
 CMPI.B  #3,D5            *' Wenn Textanfang, kein rückwärtsscrollen
 BEQ.S   HSNURV
 CMPI.B  #$05,D0          *' <CTRL+E> rückwärts
 BEQ     HSRW
```

```
HSNURV:
 CMPI.B  #2,D5          *' Wenn Textende, kein vorwärtsscrollen
 BEQ.S   HSNURR
 CMPI.S  #$18,D0        *' <CTRL+X> vorwärts
 BEQ.S   HSNOSTOP


HSNURR:
 MOVE.B  #$00,SCROLL        *' Scroll-Register auf 0 zurück
 MOVEM.L (A7)+,D0-D7/A0-A4  *' Register vom Stack zurück
 RTS

*' ------------- Auswertung <ESC>, <CTRL+E>, <CTRL+X> und <SPACE>:
HSSCR1:
 MOVE.B  KEYDAT,D0      *' Tastatur-Port einlesen, wenn 0,
 TST.B   D0             *' dann kein Zeichen eingegeben
 BMI.S   HSNOSTOP       *' In Textausgabe nicht anhalten
 CMPI.B  #$1B,D0        *' Wenn <ESC>, dann Abbruch
 BEQ.S   HSEND
 CMPI.B  #$20,D0        *' Wenn <SPACE>, dann Ausgabe stoppen
 BEQ.S   HSWAITK0
 TST.B   D5             *' 1. Seite, kein Richtungswechsel
 BEQ.S   HSVR2
 CMPI.B  #$05,D0        *' <CTRL+E> rückwärts
 BEQ     HSRW
 BRA.S   HSNOSTOP
HSWAITK0:
 CLR.B   KEYRES         *' Tastatur-Port löschen
HSWAITK1:
 MOVE.B  KEYDAT,D0      *' Tastatur-Port einlesen, wenn 0,
 TST.B   D0             *' dann kein neues Zeichen eingegeben
 BMI.S   HSWAITK1       *' Zurück und warten
 CMPI.B  #$20,D0        *' Wenn <SPACE>, dann Ausgabe fortsetzen
 BNE.S   HSWAITK0       *' Sonst weiter warten

*' ------------- Scroll-Richtung auswerten:
HSNOSTOP:
 CMPI.B  #$18,D0        *' <CTRL+X> Vorwärts-Scrollen gesetzt ?
 BEQ.S   HSVR1
 CMPI.B  #$05,D6        *' <CTRL+E> Rückwärts-Scrollen gesetzt ?
 BEQ     RSSCR3
HSVR1:
 CMPI.B  #$05,D6        *' War zuletzt Rückwärts-Scrollen gesetzt ?
 BNE.S   HSVR2
 MOVEQ   #32-1,D4       *' Dann im Text um 32 Zeilen vorrücken
HSRW1B:                 *' an unteren Bildrand
 MOVE.B  (A4)+,D0
 CMPI.B  #$0A,D0        *' Zeilen-Ende
 BNE.S   HSRW1B
 DBRA.S  D4,HSRW1B
 SUBQ    #1,A4          *' Text-Position -1
```

```
*' ------------ Ausgabe bis 1.Seite voll:
HSVR2:
 CLR.B   D6             *' Scroll-Richtungserkennung löschen
 MOVEQ   #85-1,D7       *' Zeichen / Zeile Zähler
 CLR.B   KEYRES         *' Tastatur-Port löschen
 TST.B   D5             *' Wenn 0, dann noch 1.Seite
 BNE.S   HSSCR3         *' sonst scrollen
 TST.B   D2             *' Y-Achse = 0, dann 1.Seite voll
 BEQ.S   HSSCR2         *' und ab nun scrollen
 MOVE    #3000,D3       *' Zähler für Ausgabe-Verzögerung
HSWAIT1:                *' Warten am Ende jeder Zeile
   NOP
   DBRA.S  D3,HSWAIT1
 SUBQ.B  #8,D2          *' Y-Achse -8 = nächste Zeile
 BSR     HSSETY         *' und Position an GDP
 BSR     HSSETX0        *' X-Register GDP auf 0

 ADDQ    #1,A4          *' <LF> im Textbuffer überspringen
 BRA     HSAUS          *' und dann nächste Zeile ausgeben

*' ------------ Ausgabe scrollen vorwärts:
HSSCR2:
 MOVEQ   #1,D5          *' Kennung 1.Seite voll
HSSCR3:
 BTST.B  #2,(A3)        *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSSCR3         *' dann GDP bereit, sonst warten
 MOVE.B  #$01,(A3)      *' GDP - Löschmode
 SUBQ.B  #8,D2          *' Y-Achse -8 = nächste Zeile
 BSR     HSSETY
 BSR     HSSETX0
 MOVEQ   #80-1,D3       *' 80 Zeichen = Eine Zeile löschen
HSBLK2:
   BTST.B  #2,(A3)      *' Bit 2 prüfen, wenn 1,
   BEQ.S   HSBLK2       *' dann GDP bereit, sonst warten
   MOVE.B  #$0A,(A3)    *' GDP - 5 * 8 Block zeichnen
   DBRA.S  D3,HSBLK2
HSSCR4:
 BTST.B  #2,(A3)        *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSSCR4         *' dann GDP bereit, sonst warten
 MOVE.B  #$00,(A3)      *' GDP - Schreibmode
 BSR     HSSETX0        *' X-Register GDP auf 0
 ADDQ.B  #8,D2          *' Y-Achse wieder +8

*' ------------ Scroll-Routine vorwärts:
 MOVE.B  #ZEILEN,D4     *' Zeilenvorschub nach D4
 CMP1.B  #2,D4          *' 2 Zeilen ?
 BEQ.S   HSZEIL4
 CMP1.B  #4,D4          *' 4 Zeilen ?
 BEQ.S   HSZEIL2
 MOVEQ   #1-1,D3        *' Ein Scroll-Umlauf mit 8 Zeilen
 BRA.S   HSWAIT2
HSZEIL4:
 MOVEQ   #4-1,D3        *' Zwei Scroll-Umläufe je 4 Zeilen
 BRA.S   HSWAIT2
HSZEIL2:
 MOVEQ   #2-1,D3        *' Vier Scroll-Umläufe je 2 Zeilen
```

```
HSWAIT2:
   MOVE    #$FFFF,D0     *' Voreinstellung D0
HSWAIT3:                 *' Warten auf VSYNC der GDP (BIT = 1)
   BTST.B  #1,(A3)       *' Abfragen des GDP-Status-Register
   BEQ.S   HSSYN1        *' Wenn 0, D0 löschen und warten
   TST     D0            *' Wenn 1, D0 testen, wenn noch $FF, dann
   BNE.S   HSSYN2        *' war VSYNC sofort da, warten bis
   BRA.S   HSWAIT4       *' nächstes VSYNC-Signal anliegt (20 msec)
HSSYN1:
   CLR     D0
HSSYN2:
   BRA.S   HSWAIT3
HSWAIT4:
   SUBQ.B  #ZEILEN,D2    *' Verschiebung Linien / Umlauf  ( 8   4   2)
   MOVE.B  D2,SCROLL     *' und in Scroll-Register laden
   DBRA.S  D3,HSWAIT2
 ADDQ    #1,A4           *' <LF> im Textbuffer überspringen
 BRA     HSAUS           *' und nächste Zeile ausgeben

*' ============  Rückwärts - Scrollen:
HSRW:                    *' Nur bei Richtungswechsel
 CMPI.B  #$05,D6         *' Wenn schon rückwärts,

 BEQ.S   RSSCR3          *' dann Taste ignorieren
 MOVE.B  D0,D6           *' Scroll-Richtung nach D6 retten
 MOVEQ   #31-1,D4        *' Zunächst im Text 31 Zeilen zurück
HSRW1:                   *' auf Bildanfang
 MOVE.B  -(A4),D0
 CMPI.B  #$0A,D0         *' Zeilenende
 BNE.S   HSRW1
 DBRA.S  D4,HSRW1

*' ------------- Ausgabe scrollen rückwärts:
RSSCR3:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   RSSCR3          *' dann GDP bereit, sonst warten
 MOVE.B  #$01,(A3)       *' GDP - Löschmode
 MOVEQ   #85-1,D7        *' Zeichen / Zeile Zähler
 CLR.B   KEYRES          *' Tastatur-Port löschen
 BSR     HSSETY
 BSR     HSSETX0
 MOVEQ   #80-1,D3        *' 80 Zeichen = Eine Zeile löschen
RSBLK2:
   BTST.B  #2,(A3)       *' Bit 2 prüfen, wenn 1,
   BEQ.S   RSBLK2        *' dann GDP bereit, sonst warten
   MOVE.B  #$0A,(A3)     *' GDP - 5 * 8 Block zeichnen
   DBRA.S  D3,RSBLK2
RSSCR4:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   RSSCR4          *' dann GDP bereit, sonst warten
 MOVE.B  #$00,(A3)       *' GDP - Schreibmode
 BSR     HSSETX0         *' X-Register GDP auf 0
```

```
*' ------------ Scroll-Routine rückwärts
 MOVE.B  #ZEILEN,D4      *' Zeilenvorschub nach D4
 CMPI.B  #2,D4           *' 2 Zeilen ?
 BEQ.S   RSZEIL4
 CMPI.B  #4,D4           *' 4 Zeilen ?
 BEQ.S   RSZEIL2
 MOVEQ   #1-1,D3         *' Ein Scroll-Umlauf mit 8 Zeilen
 BRA.S   RSWAIT2
RSZEIL4:
 MOVEQ   #4-1,D3         *' Zwei Scroll-Umläufe je 4 Zeilen
 BRA.S   RSWAIT2
RSZEIL2:
 MOVEQ   #2-1,D3         *' Vier Scroll-Umläufe je 2 Zeilen
RSWAIT2:
   MOVE    #$FFFF,D0     *' Voreinstellung D0
RSWAIT3:                 *' Warten auf VSYNC der GDP (BIT = 1)
   BTST.B  #1,(A3)       *' Abfragen des GDP-Status-Register
   BEQ.S   RSSYN1        *' Wenn 0, D0 löschen und warten
   TST     D0            *' Wenn 1, D0 testen, wenn noch $FF, dann
   BNE.S   RSSYN2        *' war VSYNC sofort da, warten bis
   BRA.S   RSWAIT4       *' nächstes VSYNC-Signal anliegt (20 msec)
RSSYN1:
   CLR     D0
RSSYN2:
   BRA.S   RSWAIT3
RSWAIT4:
   ADDQ.B  #ZEILEN,D2    *' Verschiebung Linien / Umlauf  ( 8  4  2)
   MOVE.B  D2,SCROLL     *' und in Scroll-Register laden
   DBRA.S  D3,RSWAIT2

*' ------------ Im Text nun zwei Zeilen zurück:

 MOVEQ   #2-1,D4
RSRW1:
 MOVE.B  -(A4),D0
 CMPA.L  A4,A2           *' Textanfang erreicht ?
 BEQ.S   RSANF
 CMPI.B  #$0A,D0         *' Zeilenende
 BNE.S   RSRW1
 DBRA.S  D4,RSRW1
 ADDQ    #1,A4           *' Um <LF> im Textbuffer vorrücken
 BRA     HSAUS           *' und nächste Zeile ausgeben

*' ------------ Wenn Text-Anfang wieder erreicht wurde:
RSANF:
 MOVE.B  (A4)+,D0        *' Zeichen aus Text-Buffer nach D0
 CMPI.B  #$0D,D0         *' Wenn <CR>, dann Ende 1. Zeile
 BEQ     HSENDR          *' und fragen ob wieder vorwärts
 BPL.S   RSAUSW          *' Daten-Byte <$7F ?, sonst
 BSR     HSGERMAN        *' Deutsche Sonderzeichen
 BRA.S   RSANF
RSAUSW:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   RSAUSW          *' dann GDP bereit, sonst warten
 MOVE.B  D0,(A3)         *' ASCII - Zeichen an GDP
 BRA.S   RSANF           *' und nächstes Zeichen holen
```

```
*-------------------------------------------------------------------------*
*ö            X, Y-POSITION GDP - REGISTER                               *
*-------------------------------------------------------------------------*
*'------------- Y-Achse in GDP-Register laden:
HSSETY:                  *' A3 = GDP I/O Port
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSSETY          *' dann GDP bereit, sonst warten
 MOVE.B  D2,CPU*$B(A3)   *' LSB Port Y
 ROR.W   #8,D2           *' Rotiere rechts 1 Byte
 MOVE.B  D2,CPU*$A(A3)   *' MSB Port Y
 ROL.W   #8,D2           *' Rotiere links, wieder zurück
 RTS

*'------------- X-Achse in GDP-Register laden:
HSSETX:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSSETX          *' dann GDP bereit, sonst warten
 MOVE.B  D1,CPU*$9(A3)   *' LSB Port X
 ROR.W   #8,D1           *' Rotiere rechts 1 Byte
 MOVE.B  D1,CPU*$8(A3)   *' MSB Port X
 ROL.W   #8,D1           *' Rotiere links, wieder zurück
 RTS
*' ------------ X-Register der GDP auf 0 setzen:
HSSETX0:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSSETX0         *' dann GDP bereit, sonst warten
 MOVE.B  #$0D,(A3)       *' GDP - X-Register auf 0 setzen
 RTS

*' ------------ X, Y-Register von GDP auslesen:
HSGETXY:
 BTST.B  #2,(A3)         *' Bit 2 prüfen, wenn 1,
 BEQ.S   HSGETXY         *' dann GDP bereit, sonst warten
 MOVE.B  CPU*$8(A3),D1   *' MSB Port X nach D1
 ROL.W   #8,D1           *' Byte nach links schieben
 MOVE.B  CPU*$9(A3),D1   *' dann LSB Port X
 MOVE.B  CPU*$A(A3),D2   *' MSB Port Y nach D2
 ROL.W   #8,D2           *' Byte nach links schieben
 MOVE.B  CPU*$B(A3),D2   *' dann LSB Port Y
 RTS

*-------------------------------------------------------------------------*
*ö            DEUTSCHE SONDERZEICHEN                                     *
*-------------------------------------------------------------------------*
*' ------------ Sonderzeichen in Tabelle suchen:
HSGERMAN:                *' A3 = GDP I/O Port
 MOVEM.L D0-D4/A1-A2,-(A7) *' Register auf Stack retten
 LEA     HSTAB1(PC),A1   *' A1 = Start Zeichen-Tabelle
 LEA     HSTAB2(PC),A2   *' A2 = Start Matrix-Tabelle
 BCLR.B  #7,D0           *' Bit 7 auf 0 setzen
 MOVEQ   #7-1,D1         *' Anzahl der Zeichen
HSGER1A:
   CMP.B   (A1)+,D0      *' Zeichen mit Tabellenwert vergleichen,
   BEQ.S   HSGER1B       *' wenn gefunden, ausgeben
   ADDQ    #5,A2         *' Sonst Matrix +5 und
   DBRA.S  D1,HSGER1A    *' nächstes Zeichen vergleichen
 BRA.S   HSGER1C         *' Zeichen nicht gefunden
HSGER1B:
 BSR.S   HSGER2          *' Zeichen nun ausgeben
HSGER1C:
 MOVEM.L (A7)+,D0-D4/A1-A2 *' Register vom Stack zurück
 RTS
```

```
*' ------------ Sonderzeichen als Matrix an GDP
HSGER2:
 BSR     HSGETXY            *' Hole Aktuelle X, Y-Position der GDP
 MOVEQ   #5-1,D3            *' Anzahl Spalten
HSGER2A:
   MOVE.B  (A2)+,D0         *' Byte der Matrix laden
   MOVEQ   #8-1,D4          *' Anzahl der Zeilen
  HSGER2B:
     BTST.B  D4,D0          *' Wenn kein Punkt, dann
     BEQ.S   HSGER2C        *' nächste Y-Position laden
     MOVE.B  #$80,(A3)      *' Punkt zeichen
  HSGER2C:
     BTST.B  #2,(A3)        *' Bit 2 prüfen, wenn 1,
     BEQ.S   HSGER2C        *' dann GDP bereit, sonst warten
     ADDQ.B  #1,CPU*$B(A3)  *' Nur LSB Port Y +1 an GDP
     DBRA.S  D4,HSGER2B
 HSGER2D:
   ADDQ    #1,D1            *' X - Position +1
BSR     HSSETX             *' X - Position an GDP
   BSR     HSSETY           *' Y - Position an GDP
   DBRA.S  D3,HSGER2A
 ADDQ    #1,D1              *' X - Position +1 ( 6. Spalte )
 BSR     HSSETX             *' X - Position an GDP
 RTS

*' ------------ Tabellen der Sonderzeichen:
HSTAB1: DC.B $5B,$5C,$5D,$7B,$7C,$7D,$7E
HSTAB2:                                                    DC.B
%01111101,%00001010,%00001001,%00001010,%01111101   *' $5B =
nÄ



DC.B %00111101,%01000010,%01000010,%01000010,%00111101   *' $5C = Ö
DC.B %01111101,%01000000,%01000000,%01000000,%01111101   *' $5D = Ü
DC.B %01110001,%01010100,%01010100,%01111000,%01000001   *' $7B = ä
DC.B %00000000,%00111001,%01000100,%01000100,%00111001   *' $7C = ö
DC.B %00111101,%01000000,%01000000,%01111101,%01000000   *' $7D = ü
DC.B %00000000,%01111111,%00000001,%01001101,%00110010   *' $7E = ß
END          >>>  (C) H.-D.BULWIEN 1987  <<<
```

9. Diverses

9.1 Ausblick

Korrekturen für dieses Handbuch werden in der Zeitschrift LOOP bekanntgegeben. Man sollte dann die fehlerhaften Stellen von Hand korrigieren.

9.2 Kritik

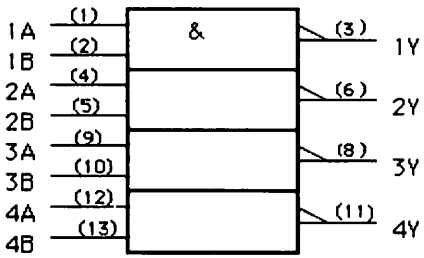
Bitte senden Sie uns die ausgefüllte Kritikkarte, die dem Bausatz beiliegt, zurück. Sie helfen uns, unsere Produkte und unseren Service noch besser zu gestalten.
Für Fehlermeldungen und Verbesserungen, die dieses Handbuch betreffen, sind wir immer dankbar!

10. Unterlagen zu den verwendeten IC's

10.1 TTL-IC's

74LS00

4 NAND-Gatter mit je 2 Eingängen

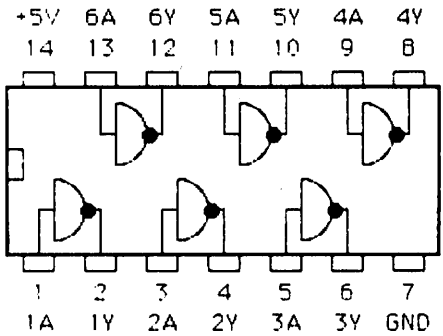


Typ. Impulsverzögerungszeit: 9,5 ns

Typ. Versorgunsstrom: 13 mA

7404

6 Inverter



Logiktabelle

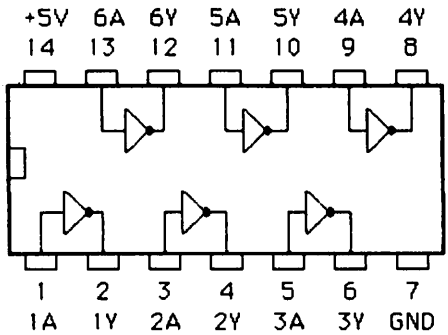
A	Y
0	1
1	0

Typ Impuls-Verzögerungszeit 10 ns

Typ Versorgungsstrom 4 mA

74LS 05

Sechs Inverter (open Collector)

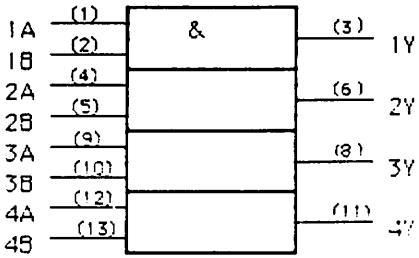


Positive Logik $Y = \overline{A}$

Typ. Impuls-Verzögerungszeit:	22 ns
Typ. Leistungs-aufnahme:	60 mW

74LS 08

4 AND-Gatter mit je 2 Eingangen

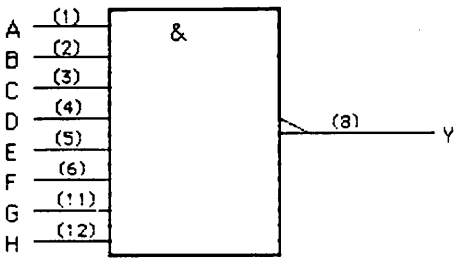


Typ Impulsverzogerungszeit 12 ns

Typ Versorgunsstrom 3,5 mA

74LS30

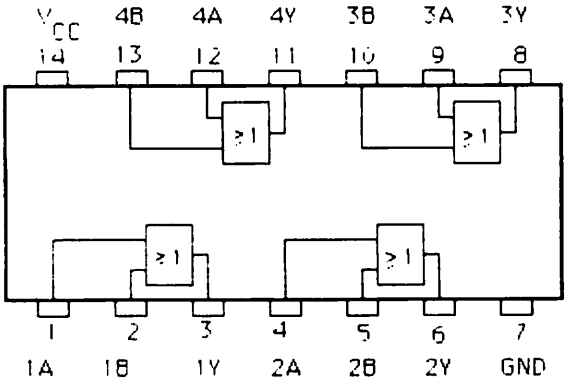
NAND-Gatter mit 8 Eingängen



Typ. Impulsverzögerungszeit	12 ns
Typ. Versorgungsstrom	0,4 mA

74LS32

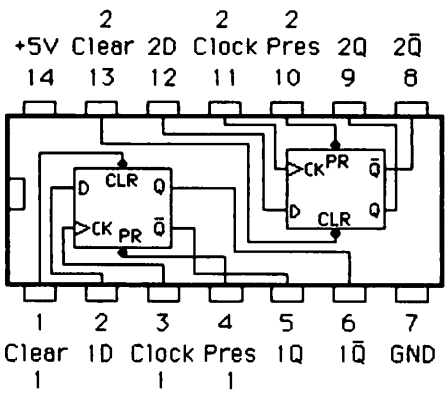
Vier Or-Gatter mit je 2 Eingängen



Typ Impulsverzögerungszeit	12 ns
Typ Versorgungsstrom	4 mA

74LS74

Zwei D-Flipflops mit Preset und Clear



Wahrheitstabelle:

Inputs				Outputs	
Preset	Clear	Clock	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

Positive Logik

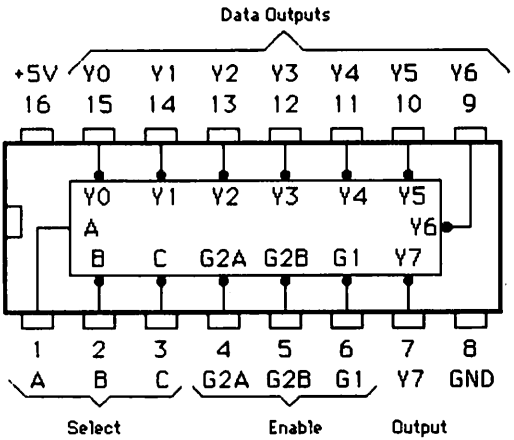
*Dieser Zustand ist nicht stabil; d.h. er bleibt nicht erhalten, wenn Preset und/oder Clear inaktiv (High) werden

Typ. Impulsverzögerungszeit : 19 ns

Typ. Versorgungsstrom . 4 mA

74LS138

3-Bit Binärdekoder/Demultiplexer (3 zu 8)



Logiktabelle

Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

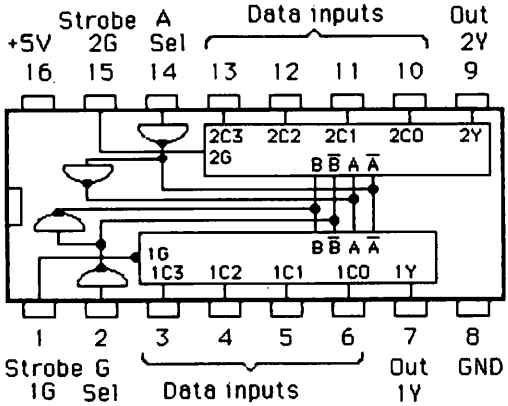
Positive Logik *G2 = G2A + G2B

Typ Impulsverzogerungszeit : 22 ns

Typ Versorgungsstrom . 7 mA

74LS 153

Zwei 4 zu 1 Datenselektor / Multiplexer



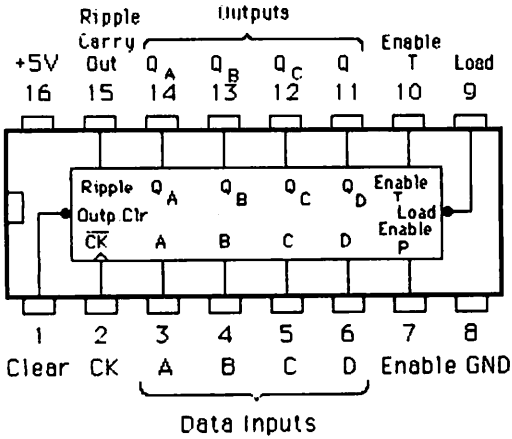
Wahrheitstabelle:

Select Inputs		Data Inputs				Strobe	Output
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

positive Logik

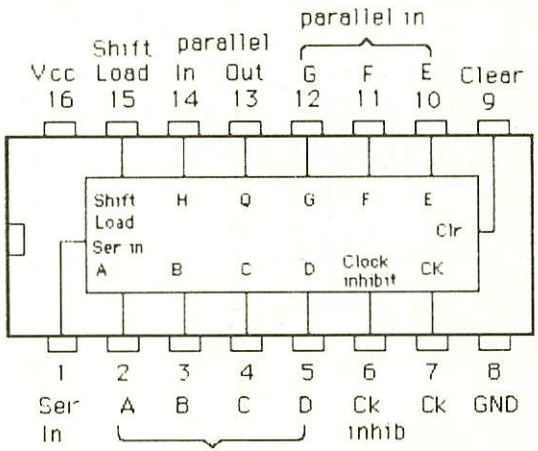
74LS 163

Synchroner programmierbarer 4 Bit Binärzähler
mit synchronem Clear



74LS166

Acht Bit Schieberegister mit Paralleleingabe und Clear



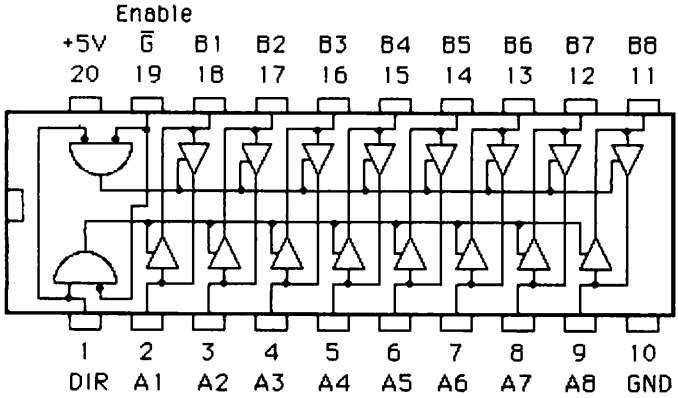
Function Table:

Inputs						Internal Outputs		Output
Clear	Shift/ Load	Clock Inhibit	Clock	Serial	Parallel A...H	Q_A	Q_B	Q_H
L	X	X	X	X	X	L	L	L
H	X	L	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	L	L	↑	X	a...h	a	b	h
H	H	L	↑	H	X	H	Q_{An}	Q_{Gn}
H	H	L	↑	L	X	L	Q_{An}	Q_{Gn}
H	X	H	↑	X	X	Q_{A0}	Q_{B0}	Q_{H0}

Positive Logik

74LS245

8-fach Bus-Transceiver mit 3-state Ausgängen



Function Table

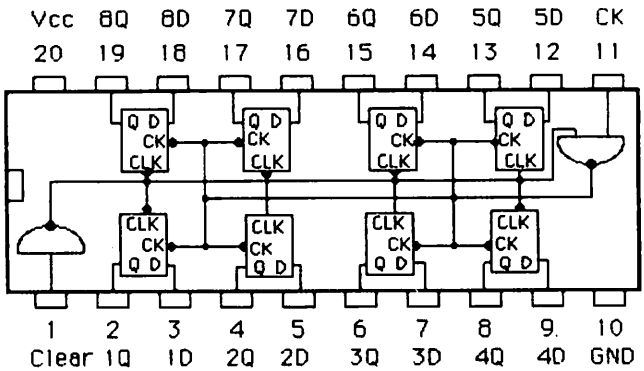
ENABLE $\overline{G}$	DIRECTION CONTROL DIR	OPERATION
L	L	B data to A bus
L	H	A data to B bus
H	x	Isolation

Typ. Impuls-Verzögerungszeit: 20 ns

Typ. Versorgungsstrom 75 mA

74 LS 273

8 - Bit D Register mit Clear



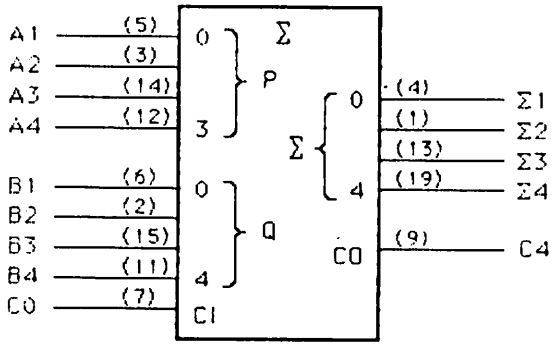
Function Table:

Inputs			Output
Clear	Clock	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

Typ. Impuls-Verzögerungszeit 17,5 ns

Typ Leistungs-aufnahme: 85 mW

74LS283 4-Bit Volladdierer



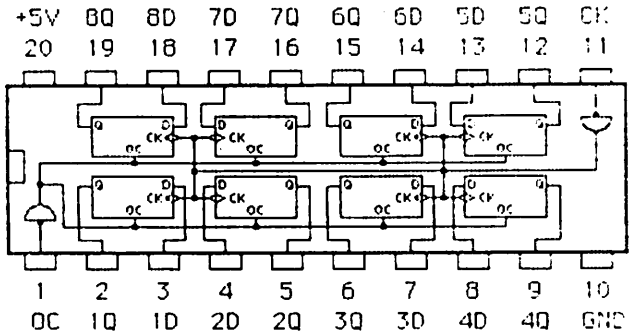
Wahrheitstabelle (positive Logik)

Eingänge				Ausgänge					
				CO=L bzw C2=L			CO=H bzw C2=H		
A1 bzw A3	B1 bzw B3	A2 bzw A4	B2 bzw. B4	Σ1 bzw. Σ3	Σ2 bzw. Σ4	C2 bzw. C4	Σ1 bzw Σ3	Σ2 bzw Σ4	C2 bzw C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

Die Zustände von Σ1, Σ2 und internem Carry C2 werden durch die Zustände der Eingänge A1, B1, A2, B2 und C0 bestimmt. Die Ausgänge Σ3, Σ4 und C4 werden von C2, A3, B3, A4, B4 best

74LS374

8-Bit D Register mit 3-state-Ausgängen



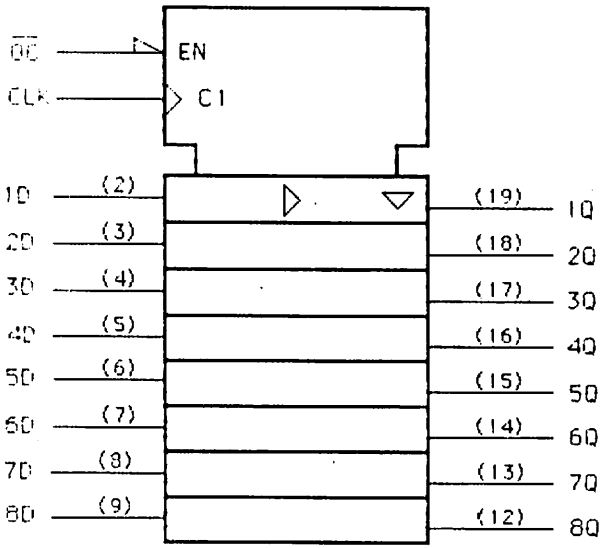
Logiktabelle

OC	CK	D	Q
0	↑	1	1
0	↑	0	0
0	0	X	Q_0
1	X	X	Z

Typ. Impuls-Verzögerungszeit	16 ns
Typ. Versorgungsstrom.	2.6 mA
positive Logik:	ja

74ALS574

8-Bit D-Flip-Flop mit nichtinvertierenden Ausgängen



Logiktabelle:

$\overline{OC}$	CLK	D	Q
0	↑	1	1
0	↑	0	0
0	0	X	Q_0
1	X	X	Z

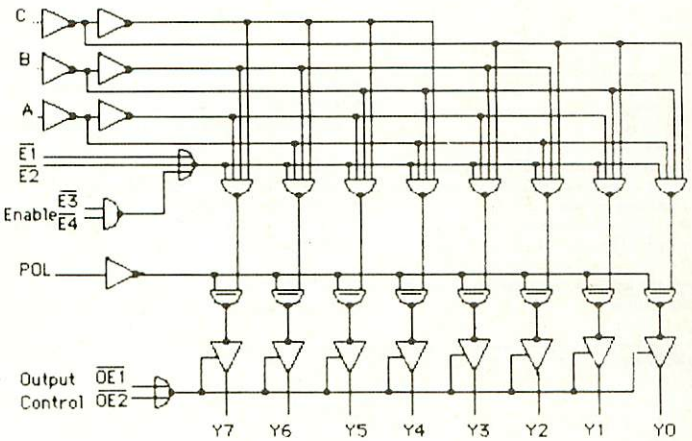
Typ. Impuls-Verzögerungszeit: 9 ns

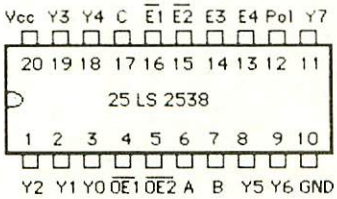
Typ. Verlustleistung: 75mW

positive Logik: ja

25 LS 2538

3 zu 8 Dekoder





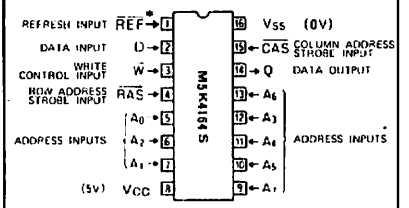
M5K4164S-15, S-20

65 536-BIT (65 536-WORD BY 1-BIT) DYNAMIC RAM

DESCRIPTION

This is a family of 65 536-word by 1-bit dynamic RAMs, fabricated with the high performance N-channel silicon-gate MOS process, and is ideal for large-capacity memory systems where high speed, low power dissipation, and low costs are essential. The use of double-layer polysilicon process technology and a single-transistor dynamic storage cell provide high circuit density at reduced costs, and the use of dynamic circuitry including sense amplifiers assures low power dissipation. Multiplexed address inputs permit both a reduction in pins to the standard 16-pin package configuration and an increase in system densities. The M5K4164S operates on a 5V power supply using the on-chip substrate bias generator.

PIN CONFIGURATION (TOP VIEW)



Outline 16S1

* If the pin 1 ($\overline{REF}$) function is not used, pin 1 may be left open (not connect).

FEATURES

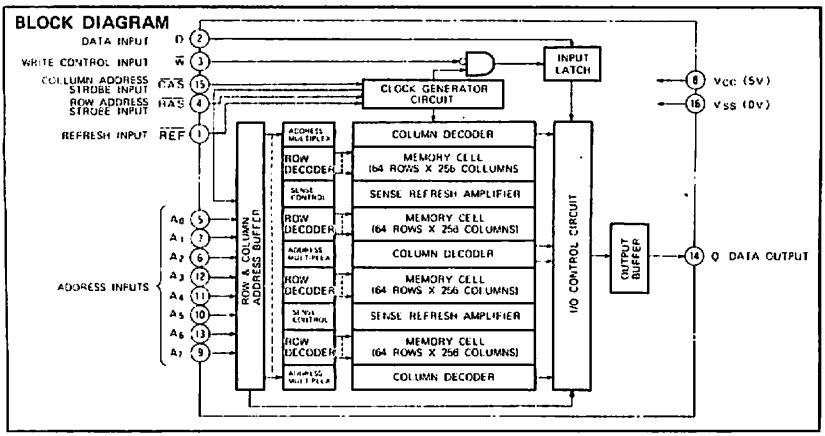
Type name	Access time (max) (ns)	Cycle time (min) (ns)	Power dissipation (typ) (mW)
M5K4164S-15	150	260	200
M5K4164S-20	200	330	170

- Standard 16-pin package
- Single 5V ±10% supply
- Low standby power dissipation: 28.0mW (max)
- Low operating power dissipation: 275mW (max)
- Unlatched output enables two-dimensional chip selection and extended page boundary
- Early-write operation gives common I/O capability
- Read-modify-write, $\overline{RAS}$-only refresh, and page-mode capabilities
- All input terminals have low input capacitance and are directly TTL-compatible
- Output is three-state and directly TTL-compatible
- 128 refresh cycles every 2ms (16K dynamic RAMs M5K4116P, S compatible)
- Pin 1 controls automatic- and self-refresh mode.
- $\overline{CAS}$ controlled output allows hidden refresh, hidden automatic refresh and hidden self-refresh.
- Output data can be held infinitely by $\overline{CAS}$.
- Interchangeable with Mostek's MK4164 and Motorola's MCM 6664 in pin configuration.

APPLICATION

- Main memory unit for computers.

BLOCK DIAGRAM



10.2 Graphik-Prozessor 9366

Register:

Über die Adressen A0 bis A3 können die 16 Register von J28 aufgerufen werden. Tabelle 1 zeigt die Register und die dazugehörigen Adressen.

ADDRESS REGISTER					REGISTER FUNCTIONS		Number of bits
Binary				Hex	Read R/W = 1	Write R/W = 0	
A3	A2	A1	A0				
0	0	0	0	0	STATUS	CMD	8
0	0	0	1	1	CTRL 1 (Write control and interrupt control)		7
0	0	1	0	2	CTRL 2 (Vector and symbol type control)		4
0	0	1	1	3	CSIZE (Character size)		8
0	1	0	0	4	Reserved		-
0	1	0	1	5	DELTAX		8
0	1	1	0	6	Reserved		-
0	1	1	1	7	DELTAY		8
1	0	0	0	8	X MSBs		4
1	0	0	1	9	X LSBs		8
1	0	1	0	A	Y MSBs		4
1	0	1	1	B	Y LSBs		8
1	1	0	0	C	XLP (Light pen)	Reserved	7
1	1	0	1	D	YLP (Light pen)	Reserved	8
1	1	1	0	E	Reserved		-
1	1	1	1	F	Reserved		-

Reserved: These addresses are reserved for future versions of the circuit. In read mode, output buffers D0-D7 force a high state on the data bus.

Bild 14

Beschreibung der einzelnen Register:

Status und Kommandoregister (Adresse 0):
Diese Register sind die Schlüsselregister zum Baustein. Das Statusregister kann vom Prozessor gelesen werden - hier meldet der EF 9366 seinen Status. Das Kommandoregister kann beschrieben werden - hier übergibt man ein Kommando zum Baustein.

Status-Register (lesen):
Bedeutung der einzelnen Bits

Bit 0	High	Ende einer Lichtgriffelsequenz
Bit 1	High	Vertikal Blank
Bit 2	High	Bereit für ein neues Kommando
	Low	BUSY (ist beschäftigt)
Bit 3	High	PEN ausserhalb des Anzeigebereiches
Bit 4	High	Lichtgriffel verursachte IRQ (falls freigeg.)
Bit 5	High	Vertikal Blank verursachte IRQ (falls frei.)
Bit 6	High	Bereit für neues Kommando verursachte IRQ
Bit 7	High	Bit 5,6,7 mit ODER verknüpft:

Kommandoregister (Schreiben):
Dieses Register hat 5 Funktionen.

Wert	Bedeutung
00h - 0Fh	Kommandos wie Bildschirm, Register löschen usw.
10h - 17h	Grundvektorbefehle Ein Vektor wird definiert durch den Betrag in den Registern DELTAX und DELTAY sowie durch das hier vorgebene Vorzeichen.
18h - 1Fh	Vektoren mit Richtungsvorgabe Ein Richtungscode (8 Richtungen) wird übergeben, das kleinere der Register DELTAX oder DELTAY ignoriert und der Vektor mit der Länge des größe ren DELTA - Registers gezeichnet.
20h - 7Fh	ASCII - Zeichen werden ausgegeben, Code - Zuordung siehe folgende Tabelle 2.
80h - FFh	Kurzvektoren In einem Byte ist ein Vektor vollständig definiert.

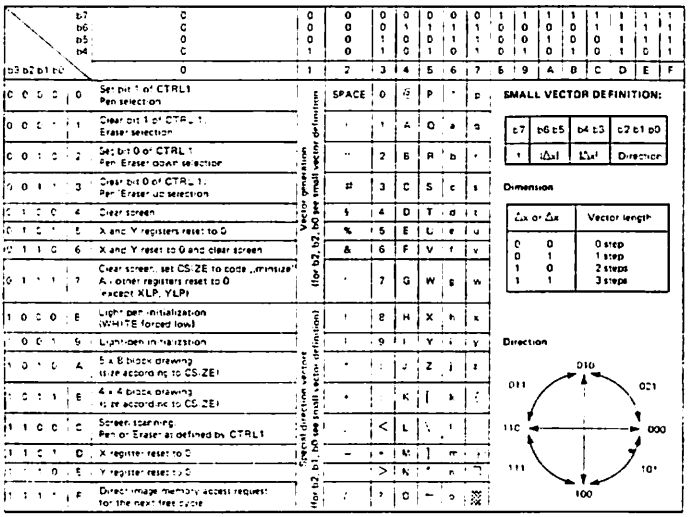


Abb. 5.3.34 Befehle 0 bis FF

Bild 15

Befehle von 00h bis 0Fh:

Bit	Bedeutung
0	Im Register CTRL 1 wird das Bit 1 gesetzt; der Schreibstift (=PEN) wird angewählt. Dieses Bit bzw.-Kommando ist vor jedem Schreiben zu geben.
1	Im Register CTRL 1 wird Bit 1 auf 0 gesetzt; das bedeutet , daß der Radiergummi (ERASER) angewählt wurde.
2	Im Register CTRL 1 wird Bit 0 gesetzt - der PEN oder ERASER wird abgesenkt.
3	Im CTRL 1 wird das Bit 0 gelöscht - PEN oder ERASER werden angehoben.
4	Bildschirm löschen.
5	Register X und Y werden rückgesetzt auf 0.
6	Bildschirm löschen und rücksetzen der Register X und Y.
7	Alle Register (außer XLP,YLP) werden zu 0 gesetzt, Bildschirm löschen.
8	Lichtgriffel initialisieren (GDP - Ausgang WHITE aktivieren, Bildschirm blinkt einen Zyklus weiß).
9	Lichtgriffel initialisieren.
A	5x8-Block zeichnen. Die Größe des Blockes ist von Register CSIZE abhängig.
B	4x4-Block zeichnen
C	Bildschirm scannen - PEN oder ERASER wie in CTRL 1 definiert.
D	Register X auf 0 rücksetzen.
E	Register Y auf 0 rücksetzen.
F	Direkter Bildzugriff im nächsten freien Zyklus.

Darstellung von ASCII-Zeichen:

Falls das Bit 7 (das höchstwertige) '0' ist und B6 bis B4 ungleich Null sind, so wird über das Kommandoregister ein ASCII-Zeichen übergeben. Dieses Zeichen wird an der Stelle X,Y mit der im CSIZE-Register angegebenen Größe und der in CTRL 2 definierten Richtung angezeigt. Diese Zeichen werden in einer 5x8-Matrix ausgegeben. Nach der Ausgabe eines Zeichens wird x um 6 Bildpunkte erhöht. Dies verdeutlicht nebenstehendes Bild.
Jeder der ausgegebenen Bildpunkte kann durch einen Block, der PxQ groß ist, ersetzt werden. P und Q können von 1 bis 15 reichen und werden im CSIZE-Register festgelegt. Die Zeichen liegen von 20h bis 7Fh und entsprechen den 96 Standard (USA) ASCII-Zeichen. Ein 97stes Zeichen (0Ah) erzeugt einen 5P x 8Q- Block und kann dazu verwendet werden, andere Zeichen zu löschen. Das 98ste Zeichen erzeugt ein 4Px4Q-Feld ohne Zwischenraum zum nächsten Zeichen. Mit diesem Zeichen können grobe graphische Zeichnungen (z.B. Balkendiagramme) erzeugt werden. Ein Zeichen kann auf zweierlei Arten gelöscht werden: Entweder mit dem Zeichen 0Ah, oder indem man das gleiche Zeichen (mit gleichem Startpunkt X,Y) und eingeschaltetem ERASER überschreibt.
Hinweis: Das Blank (20h) löscht nicht, sondern positioniert nur den X-Wert ein Zeichen weiter.

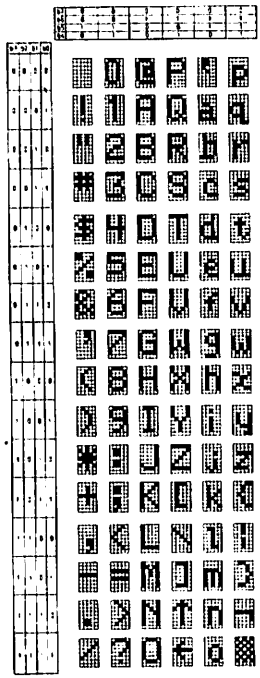


Bild 16

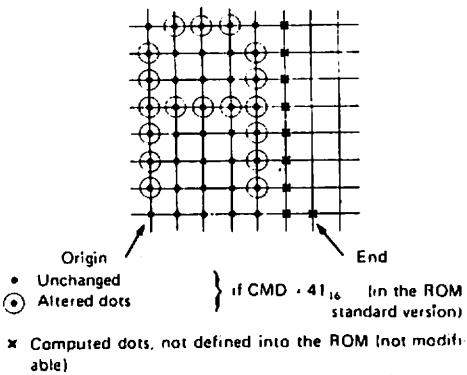


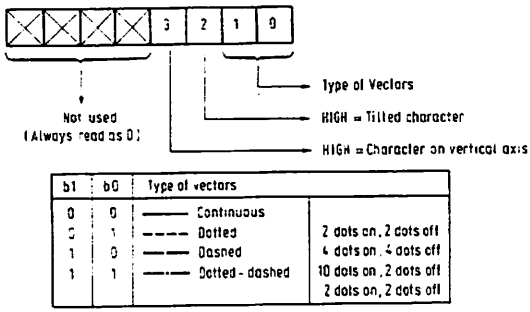
Bild 17

Andere Register:

CTRL 1 Steuerregister 1 (Adresse 1)
Einige Bits des Registers sind redundant,d.h. sie können auch über das Kontrollregister gesetzt bzw. gelöscht werden. Es sind dies die beiden (wichtigsten) niederwertigen Bits B0 und B1.

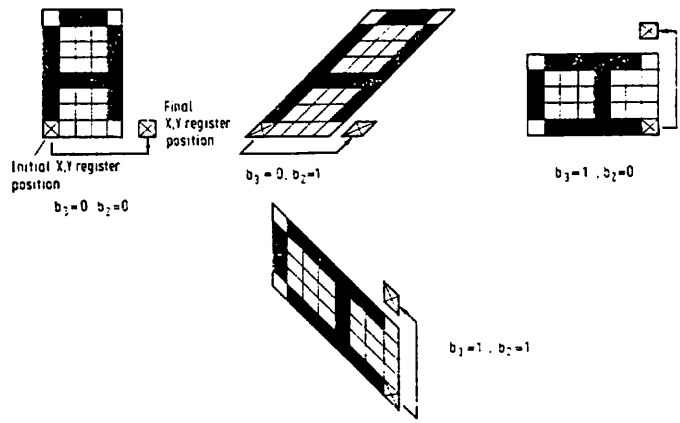
Bit	Bedeutung
0	HIGH = PEN unten LOW = PEN oben
1	HIGH = PEN LOW = ERASER
2	HIGH = schnell schreiben ohne Ausgangssignal
3	HIGH = geschlossene Bildfläche,d.h. es wird auch geschrieben, wenn MSB X,Y außerhalb der Bildfläche sind.
4	IRQ - Freigabe für Lichtgriffel
5	IRQ - Freigaben bei VB
6	IRQ - Freigabe bei 'bereit'
7	Nicht verwendet (0)

CTRL 2 Steuerregister 2 (Adresse 2)
Steuerregister 2 steuert die Art der gezeichneten Vektoren (durchgezogen, gepunktet, gestrichelt oder strichpunktiert), sowie die Art der Zeichendarstellung.

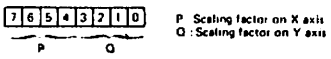


b1	b0	Type of vectors	
0	0	—— Continuous	
0	1	- - - - Dotted	2 dots on, 2 dots off
1	0	—— Dashed	4 dots on, 4 dots off
1	1	—·— Dotted - dashed	10 dots on, 2 dots off 2 dots on, 2 dots off

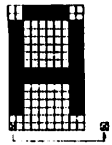
Types of character orientations



CSIZE (Register zur Festlegung der Zeichengröße) (Adresse 3):
In diesem Register wird die Größe der darzustellenden Zeichen übergeben. Die Größe ist in X und Y-Richtung in 16 Schritten wählbar.



CSIZE = 11_{16}

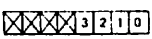


CSIZE = 22_{16}

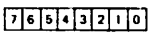
DELTAX und DELTAY (Adressen 5 und 7):
Diese Register werden bei den Grundvektorbefehlen verwendet und bedeuten die Projektion der Vektorlänge auf die jeweilige Achse. Ihre Bedeutung erhalten diese Register erst bei der Ausgabe von Grundvektoren. Im Befehl wird dann auch das Vorzeichen übergeben. Mit DELTAX und DELTAY = 0 können einzelne diskrete Punkte, deren Lage durch die Register X und Y definiert sind ausgegeben werden.

Lichtgriffel (Adressen C und D):
Hier wird vom Baustein der aktuelle Stand des Lichtgriffels übergeben. Diese Funktion ist allerdings bei der GDP 64k nicht vor- gesehen.

Register X und Y:
Diese Register beinhalten den Standpunkt für jede Operation (Vektor zeichnen oder Schriftausgabe). Die Übergabe erfolgt in 12 Bit; damit ergibt sich ein virtueller Raum von 4096 x 4096 Bit, der vom Baustein auch verwaltet wird. 512 (X) x 256 (Y) Bit werden angezeigt.



MSBs



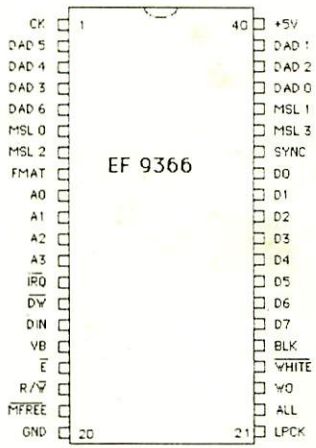
LSBs

Vektor-Befehle: Gruppe A

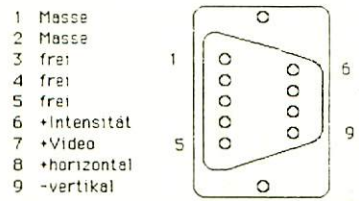
Vorzeichen		Befehl		
Delta-x	Delta-y	dez.	sedez.	binär
+	+	17	11	0001 0001
-	+	19	13	0001 0011
+	-	21	15	0001 0101
-	-	23	17	0001 0111
>0	-	16	10	0001 0000
-	>0	18	12	0001 0010
-	<0	20	14	0001 0100
<0	-	22	16	0001 0110

Richtung	Befehl dez.	sedez.	binär
3 Uhr	24	18	0001 1000
1 Uhr 30'	25	19	0001 1001
12 Uhr	26	1A	0001 1010
10 Uhr 30'	27	1B	0001 1011
6 Uhr	28	1C	0001 1100
4 Uhr 30'	29	1D	0001 1101
9 Uhr	30	1E	0001 1110
7 Uhr 30'	31	1F	0001 1111

Pinbelegung des EF 9366:



Pinbelegung IBM- Monitorkabel



Anmerkung: Bei niedrigem Pegel liegen die Signal-Spannungen zwischen 0,0 und 0,6 V DC, bei hohem Pegel liegen sie zwischen + 2,4 und +3,5 V DC

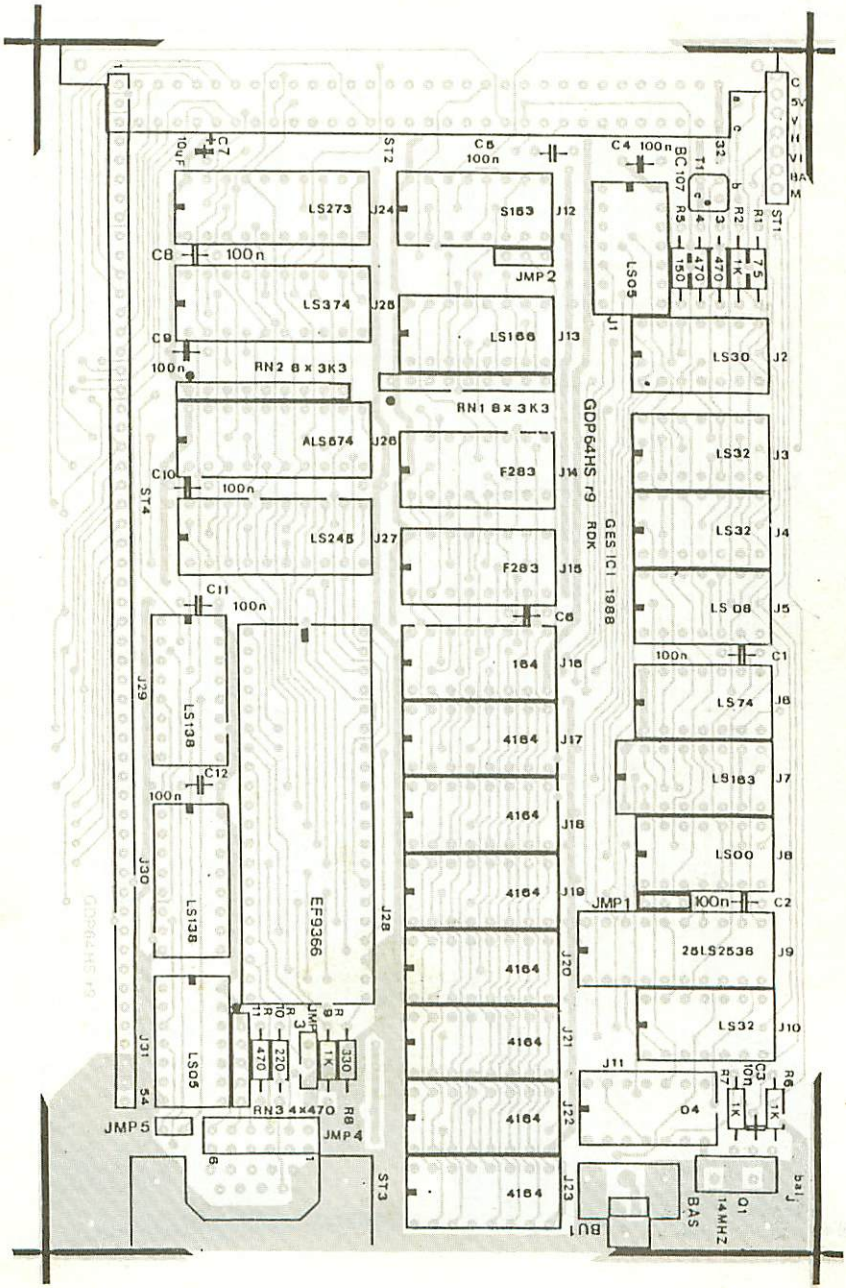
11. Literatur

In unserer Zeitschrift LOOP wird regelmäßig über neue Produkte und Änderungen bzw. Verbesserungen berichtet. Es ist für Sie von großem Vorteil, LOOP zu abonnieren, denn dadurch ist sichergestellt, daß Sie auch immer über die neuesten Informationen verfügen.

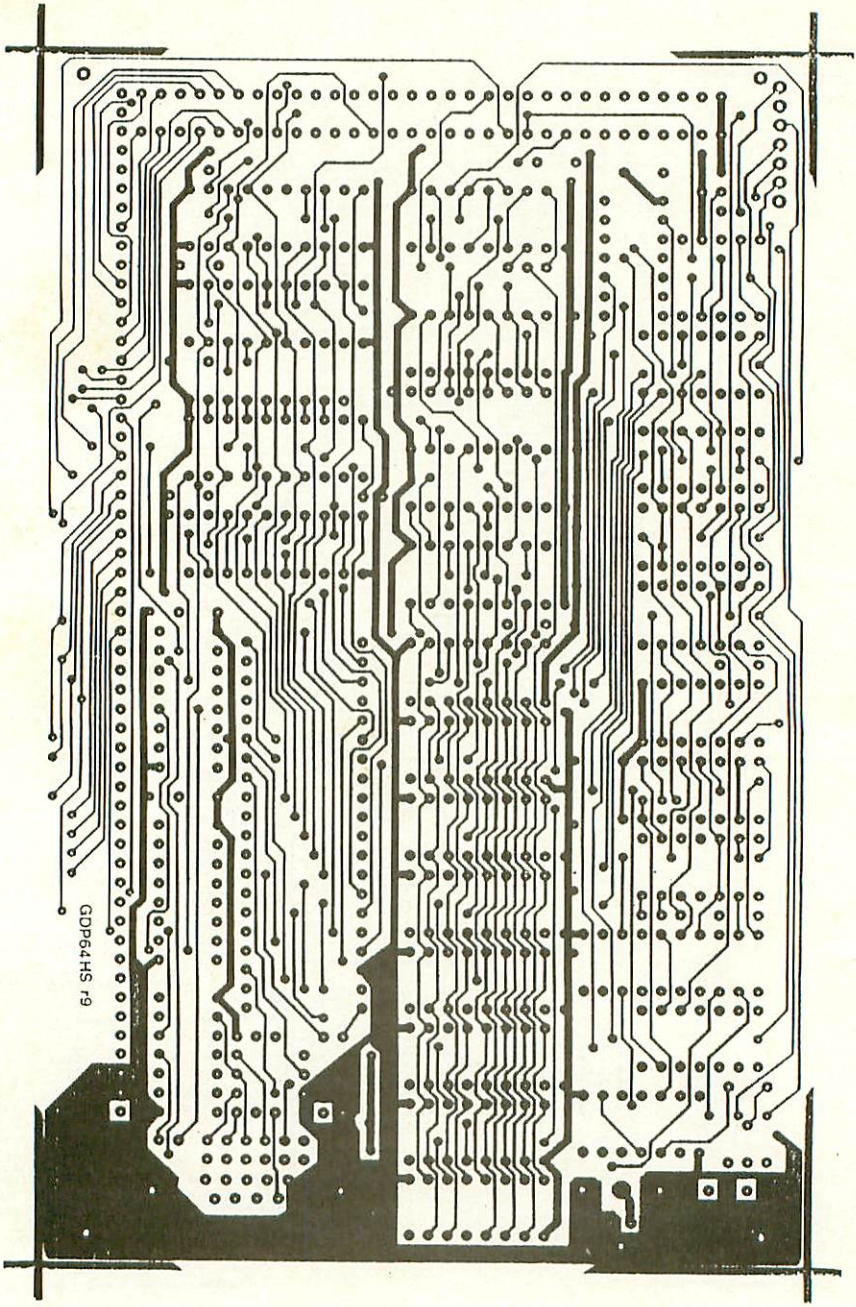
Ein LOOP-Abo können Sie bei jeder Bestellung einfach mitbestellen.

Auch auf der Kritikkarte können Sie ein LOOP-Abo ganz einfach bestellen.

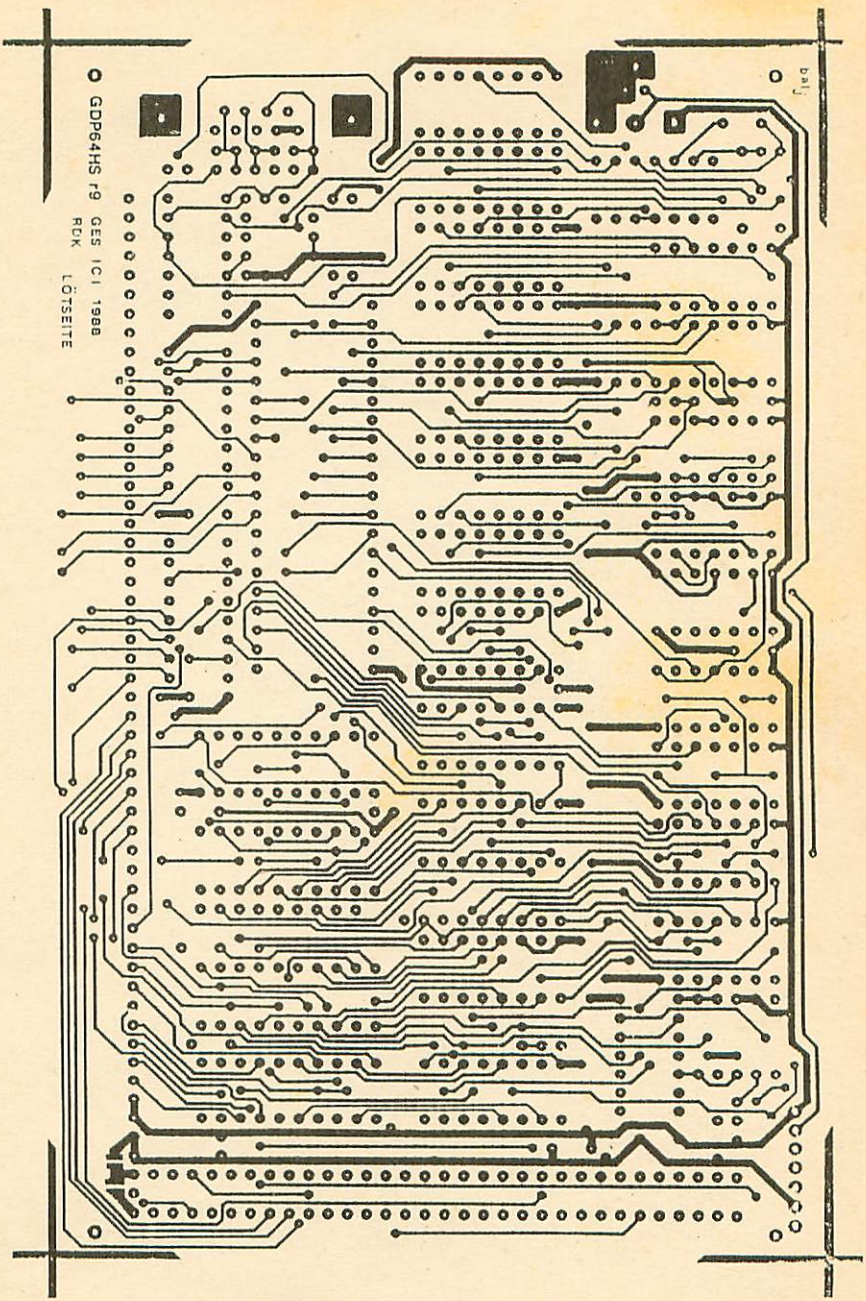
Anhang C: Bestückungsplan mit Layout Bestückungsseite

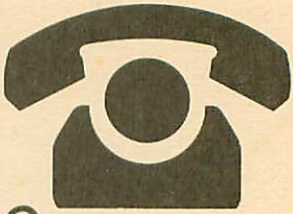


Anhang D: Layout Bestückungsseite



Anhang E: Layout Lötseite





Telefonservice
08 31- 62 11
jeden Mittwochabend
bis 20.00 Uhr

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr