Rolf-Dieter Klein

CRT-Controller unterstützt Grafikfunktionen

Bisherige CRT-Controller waren überwiegend textorientiert. Grafische Darstellungen mußte die Zentraleinheit aufbereiten. Dadurch ließen sich nur mäßige Geschwindigkeiten erzielen. Der sogenannte Grafik-Display-Prozessor (GDP) – ein kürzlich entwickelter LSI-Baustein – setzt hier neue Maßstäbe. Neben einem Zeichengenerator enthält er einen Vektorgenerator, der bis zu 1,3 Mio. Punkte/s erzeugen kann.

1 Interner Aufbau

Die *Bilder 1* und *2* zeigen Blockschaltung und Anschlußbelegung des Grafik-Display-Prozessors. Der Baustein besitzt ein Prozessor- und ein Display-Interface. Das Prozessor-Interface benötigt die Signale $\overline{E}$, R/$\overline{W}$, A 0...3 und D 0...7. Es läßt sich an praktisch alle gängigen Mikroprozessoren anschließen. Vom Prozessor-Interface aus können verschiedene interne Register beschrieben oder gelesen werden: CMD, STATUS, CTRL1, CTRL2, DELTAX, DELTAY, CSIZE, X, Y, XLP und YLP.

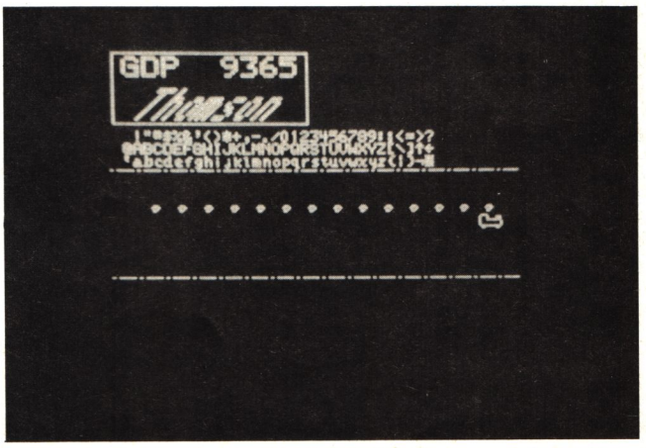
Der GDP übernimmt die komplette Steuerung des Bildwiederholspeichers. Als Speicher lassen sich dynamische 16-Kx1-RAMs verwenden, wobei sogar das Multiplexen der Adressen für die CAS-/RAS-Steuerung der Controller durchführt.

Der Typ 9365 erlaubt zwei Betriebsarten: eine mit 256 x 256 Punkten, eine weitere mit 512 x 512 Punkten (Zeilensprungverfahren). Die Auswahl der Betriebsart geschieht über den Anschluß FMAT. Liegt er auf L, werden 256 x 256 Punkte adressiert, sonst 512 x 512 Punkte im Zeilensprungverfahren. Ein weiterer Typ der GDP-Serie (EF 9366) erlaubt die Darstellung von 512 x 256 Punkten ohne Zeilensprung. Die gemultiplexten Adressen gelangen über die Ausgänge DAD 0...6 an die RAM-Bausteine. Die Selektion geschieht über die Ausgänge MSL 0...3. Bei 256 x 256 Punkten werden nur vier RAM-Bausteine benötigt. Die Aufteilung der Punkte von Zeichen und Vektoren wird vom GDP automatisch vorgenommen, und durch die Ausgänge MSL werden die nötigen Bausteine ausgewählt. Dabei steht an MSL ein decodiertes Signal zur Verfügung. Im Modus „512 x 512" werden 16 Bausteine benötigt, die in Gruppen zu acht zusammengefaßt werden. Die Selektion geschieht wieder über die Leitungen MSL 0...3, die diesmal aber ein undecodiertes Auswahlsignal beinhalten.

2 Minimalschaltung

Bild 3 zeigt ein praktisches Beispiel für den Anschluß des GDP an den Prozessor Z80. Es wurde der Modus „256 x 256" gewählt, um die Minimalkonfiguration zu zeigen. Mit B1 und B2 wird der Datenbus der CPU gepuffert. Selektiert wird der Baustein über den Eingang $\overline{E}$. Am Vergleicher V1 läßt sich die Port-Adresse einstellen. Wird der Baustein selektiert, wird durch die Verknüpfung mit $\overline{IORD}$ durch O1 erreicht, daß die Richtung des Bustreibers B1 vom Baustein 9365 zum CPU-Bus geschaltet wird. $\overline{IORD}$ ist die Verknüpfung von $\overline{RD}$ mit $\overline{IORQ}$, die bei den meisten Z80-Systemen schon vorhanden ist. $\overline{IORW}$ ($\overline{WR} \vee \overline{IORQ}$) wird über einen Puffer direkt an den Eingang R/$\overline{W}$ des GDP geführt.

Der GDP besitzt einen IRQ-Ausgang zur Ausgabe einer Interrupt-Anforderung, der hier aber der Einfachheit halber nicht verwendet wurde. Der Bildteil besteht aus den RAMs und einer Ablaufsteuerung. Das System wird von einer Quarzfrequenz mit 6,9888 MHz getaktet. Zähler Z1 teilt den Takt durch vier und



Einige Möglichkeiten des Grafik-Display-Prozessors demonstriert dieses Bild: Schon mit einem einfachen Programm lassen sich Texte, vergrößerte und schräggestellte Zeichen, ruhende und bewegte Objekte darstellen

zählt von 12 bis 15. Sein Ausgangssignal gelangt über zwei Invertierer an den Takteingang des GDP. Außerdem wird es auf die ODER-Gatter O3, O4, O5 und O6 geleitet und mit den Ausgängen MSL verknüpft. Es wird immer dann ein $\overline{\text{RAS}}$-Signal am Ausgang der ODER-Gatter erzeugt, wenn der Ausgang 16 und der entsprechende MSL-Ausgang des GDP auf L sind. Beim Auslesen eines Bildes werden alle $\overline{\text{RAS}}$-Eingänge der Speicher getaktet, beim Schreiben nur diejenigen, bei denen ein Punkt gesetzt oder gelöscht wird. Ob gesetzt oder gelöscht wird, wird durch den Ausgang DIN bestimmt. Der Schreibzyklus wird durch DW angezeigt. Da mit einem gemultiplexten Adreßbus gearbeitet wird, ist noch ein weiteres Auswahlsignal nötig: $\overline{\text{CAS}}$. Dieses Signal wird vom Ausgang des Invertierers I8 abgeleitet, der vom Übertragssignal des Zählers Z1 kommt. Das Signal bewirkt auch, daß die Ausgangsdaten der RAMs in das Schieberegister geladen werden. Die Daten werden mit dem Zentraltakt aus dem Register geschoben und gelangen über das ODER-Gatter O2, mit dem sie nochmals mit dem Takt verknüpft werden, zum BAS-Mischer. Der BAS-Mischer verknüpft das Videosignal und das Synchronsignal des GDP zu einem normgerechten Fernsehsignal, das direkt auf einen Monitor geführt werden kann.

Weitere Signale, die hier nicht verwendet wurden, sind: $\overline{\text{MFREE}}$, ein Ausgang des GDP, der dazu dient, um auch vom Prozessor aus Zugriff auf den Bildwiederholspeicher zu haben. Nach einer DMA-Anforderung über den Kommandokanal zeigt $\overline{\text{MFREE}}$ an, wann die Daten am Ausgang des RAMs in ein externes Register gespeichert werden können (danach kann sie der Prozessor auslesen). Die Adressen an DAD sind dabei die Schreibadressen.

$\overline{\text{WHITE}}$ und LPCK sind Eingänge für den Anschluß eines Lichtgriffels. Bei der steigenden Flanke an LPCK wird die augenblickliche Position in spezielle Register gespeichert. $\overline{\text{WHITE}}$ dient zur Steuerung des Videosignals, das z. B. auf „Hell" gesetzt werden kann, um ein helles Feld auf dem Bildschirm für den Lichtgriffel zu erzeugen.

Das System läßt sich leicht auf Farbe oder Graustufen erweitern, indem weitere RAM-Ebenen vorgesehen werden. Es wird dann ein Ausgabe-Port vorgesehen, in dem vom Prozessor die gewünschte Farbe oder der Grauwert geschrieben wird. Die Ausgänge der Multiplexer steuern dann beim Einschreiben in das RAM die Auswahl der einzelnen Ebenen. Es werden also einfach die $\overline{\text{RAS}}$-Eingänge gesteuert. Ausgangsseitig verwendet man mehrere Schieberegi-

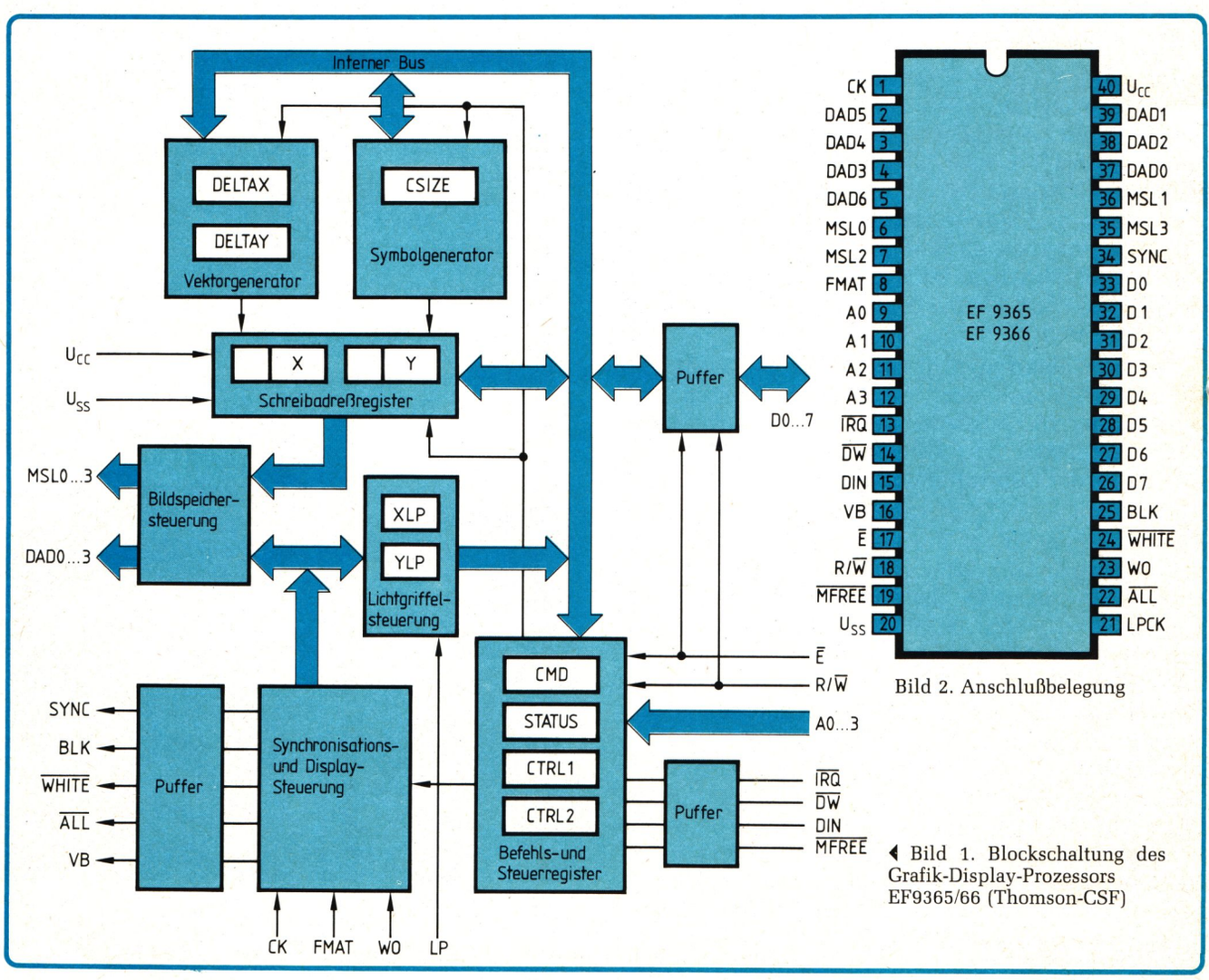


Bild 2. Anschlußbelegung

◀ Bild 1. Blockschaltung des Grafik-Display-Prozessors EF9365/66 (Thomson-CSF)

ster, deren Ausgänge die einzelnen Farbausgänge bilden. Ihre Signale kann man an RGB leiten oder mit einem Widerstandsnetzwerk dem BAS-Signal beimischen, um Graustufen zu erzeugen. Von der Software wird dann zunächst eine Farbe vorgewählt und dann ein normaler Zugriff auf den GDP ausgeführt.

Eine weitere Möglichkeit ist die Verwirklichung von Hintergrundspeichern. Die Schaltung ist praktisch die gleiche wie bei Grau und Farbe, nur das Videosignal wird gleichwertig gemischt. Beim Einschreiben kann dann z. B. der Hintergrund ausgewählt und der feste Bildanteil gespeichert werden. Bewegte Teile werden im Vordergrund gespeichert und können dort einfacher manipuliert werden, als wenn sie dauernd mit festen Bildteilen kollidieren. Allerdings ist diese Arbeitsweise beim GDP nicht nötig, da er gerade für bewegte Grafik sehr gut geeignet ist.

3 Programmierung

Tabelle 1 zeigt eine Übersicht über alle Register. Von theoretisch 16 adressierbaren Registern sind nur zwölf belegt. Das wichtigste ist Register 0, über das beim Lesen die Statusinformation und beim Schreiben Befehle und ASCII-Zeichen übergeben werden. *Tabelle 2* zeigt eine Übersicht über alle Befehle. Die Codes 20H (H steht für hexadezimal bzw. sedezimal) bis 7FH sind mit dem normalen ASCII-Zeichensatz belegt; 0...0FH sind mit verschiedenen Befehlen wie „Löschen des Bildschirms", „Pen down", „Pen up", „löschender oder schreibender Zugriff" belegt. 10H...1FH sind Vektorbefehle. Bei Vektorzeichnun-

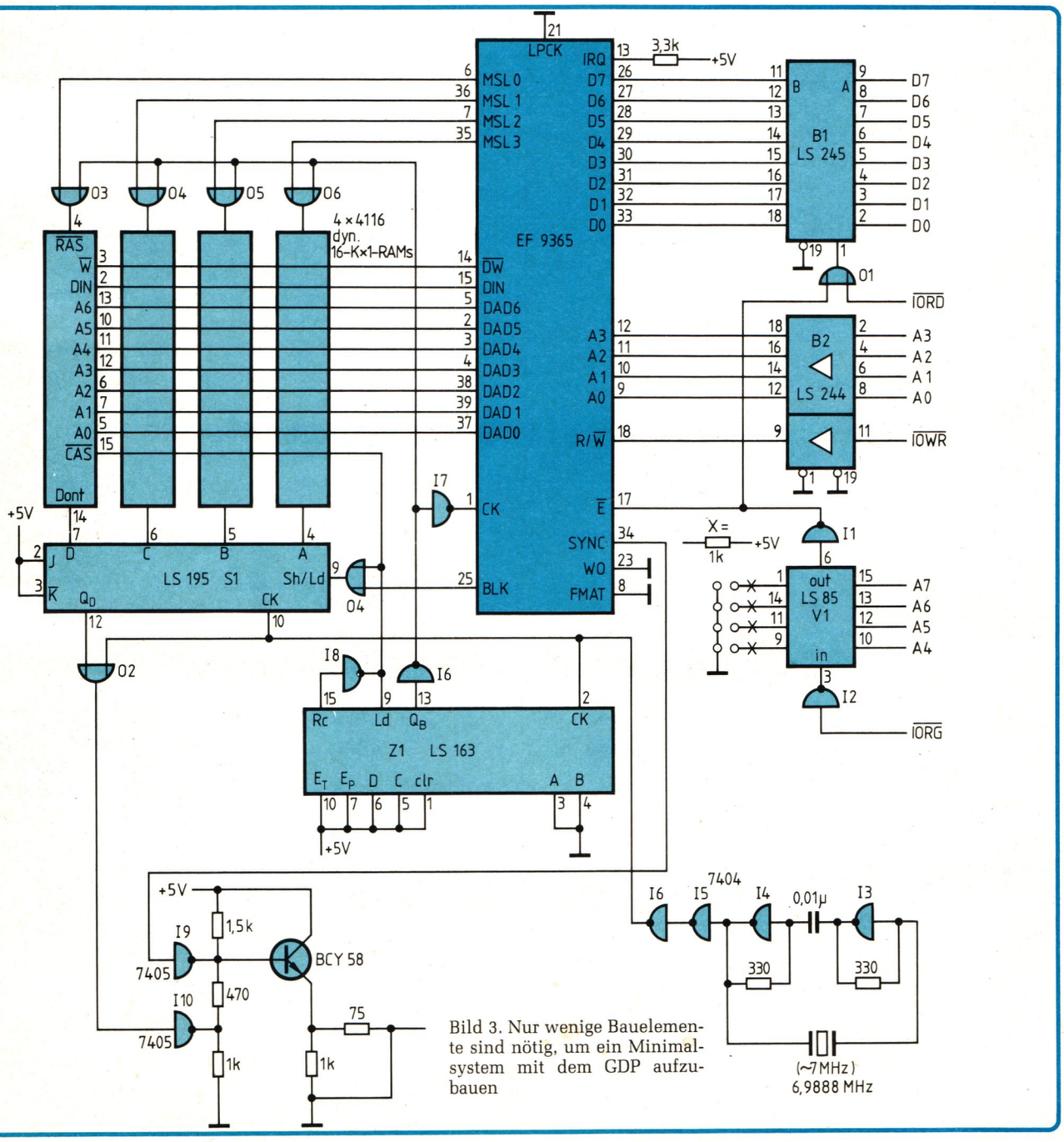


Bild 3. Nur wenige Bauelemente sind nötig, um ein Minimalsystem mit dem GDP aufzubauen

Tabelle 2. Befehlsübersicht ▶

Tabelle 1. Register des Grafik-Display-Prozessors

ADDRESS REGISTER					REGISTER FUNCTIONS		Number of bits
Binary				Hexa	Read R/$\overline{W}$ = 1	Write R/$\overline{W}$ = 0	
A3	A2	A1	A0				
0	0	0	0	0	STATUS	CMD (Command)	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	Not used		—
0	1	0	1	5	DELTAX		8
0	1	1	0	6	Not used		—
0	1	1	1	7	DELTAY		8
1	0	0	0	8	X Most significant bits		4
1	0	0	1	9	X Least significant bits		8
1	0	1	0	A	Y Most significant bits		4
1	0	1	1	B	Y Least significant bits		8
1	1	0	0	C	XLP (Light-pen)	Not used	7
1	1	0	1	D	YLP (Light-pen)	Not used	8
1	1	1	0	E	Not used		—
1	1	1	1	F	Not used		—

Auszug aus dem Originaldatenblatt

b3 b2 b1 b0		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0	Set bit 1 of CTRL1 : Pen select	Vector generation (for b2, b1, b0 see small vector definition)	SPACE	0	@	P	`	p								
0 0 0 1	1	Clear bit 1 of CTRL 1 : Eraser select		!	1	A	Q	a	q								
0 0 1 0	2	Set bit 0 of CTRL1 : Pen/Eraser down select		"	2	B	R	b	r								
0 0 1 1	3	Clear bit 0 of CTRL 1 : Pen/Eraser up select		#	3	C	S	c	s								
0 1 0 0	4	Clear screen		$	4	D	T	d	t								
0 1 0 1	5	X and Y registers reset to 0		%	5	E	U	e	u								
0 1 1 0	6	X and Y reset to 0 and clear screen		&	6	F	V	f	v								
0 1 1 1	7	Clear screen, set CSIZE to code "minsize" All other registers reset to 0 (except XLP, YLP)		'	7	G	W	g	w								
1 0 0 0	8	Light-pen initialization	Special direction vectors (for b2, b1, b0 see small vector definition)	(	8	H	X	h	x								
1 0 0 1	9	Light-pen initialization		)	9	I	Y	i	y								
1 0 1 0	A	5 x 8 block drawing (size according to CSIZE)		*	:	J	Z	j	z								
1 0 1 1	B	4 x 4 block drawing (size according to CSIZE)		+	;	K	[	k	{								
1 1 0 0	C	Screen scanning : Pen or Eraser as defined by CTRL1		,	<	L	\	l	¦								
1 1 0 1	D	X registers reset to 0		-	=	M	]	m	}								
1 1 1 0	E	Y registers reset to 0		.	>	N	↑	n	~								
1 1 1 1	F	Direct image memory access request for the next free cycle.		/	?	O	←	o	▒								

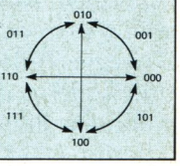
SMALL VECTOR DEFINITION :

b7	b6	b5	b4	b3	b2	b1	b0
1	|ΔX|		|ΔY|		Direction		

Dimension

ΔX or ΔY	Vector length
0 0	0 step
0 1	1 step
1 0	2 steps
1 1	3 steps

Direction



Auszug aus dem Originaldatenblatt

gen müssen zuvor immer die Register X und Y belegt sein. Sie sind 12 Bit lang. Bei 256 x 256 Punkten wären nur 8 Bit nötig und bei 512 x 512 nur 9 Bit, so daß 3 Bit eigentlich überflüssig sind. Doch damit hat es eine besondere Bewandtnis: Wird der Bildadreßraum überschritten, so werden Vektoren zwar nicht mehr gezeichnet, doch die Eckpunkte dennoch korrekt behandelt. Damit kann in einem virtuellen Raum von 4096 x 4096 Punkten gearbeitet werden. Durch ein besonderes Statusbit kann dieser Modus auch abgeschaltet und die Bildfläche zyklisch geschlossen behandelt werden.

Für die Grundvektorbefehle ist ferner die Belegung der Register DELTAX und DELTAY nötig, die den Betrag in der jeweiligen Richtung festlegen. Vorzeichen werden erst bei den Vektorbefehlen angegeben.

Bild 4a zeigt die Codierung der Grundvektorbefehle (im Register CMD). Die Register X und Y sind nach Ausführung des Befehls auf den Endpunkten, was für alle Befehle gilt. In *Bild 4b* sind Spezialvektorbefehle dargestellt, die es erlauben, auf einfache Art z. B. Rechtecke zu zeichnen, wobei in DELTAX und DELTAY nur die Diagonale bestimmt wird. *Bild 4c* schließlich zeigt weitere Vektorbefehle, die nun aber einen Code mit Diagonalrichtungen beinhalten.

Der Codebereich 80H...0FFH ist den Kurzvektorbefehlen vorbehalten, die es erlauben, ohne Belegung von DELTAX und DELTAY Zeichnungen auszuführen. Dabei können Vektoren von 0...3 Punkten in einer durch den Richtungscode bestimmten Schreibrichtung angegeben werden (*Bild 4d*). Die Befehle können z. B. dazu verwendet werden, um Umrißzeichnungen sehr schnell anzufertigen. Sie sind deshalb auch für bewegte Bilder sehr gut geeignet.

Ein weiteres wichtiges Register ist das Statusregister. *Bild 5* zeigt die Bedeutung der einzelnen Bits.

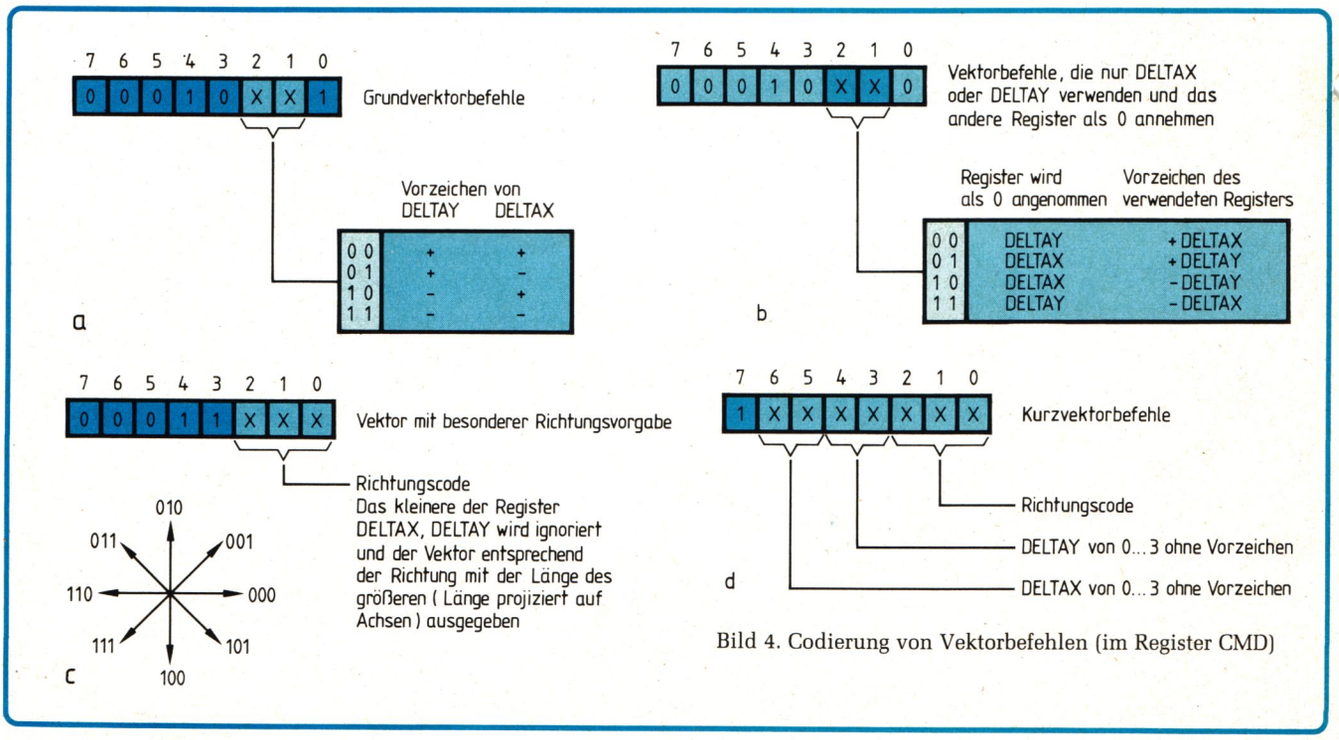
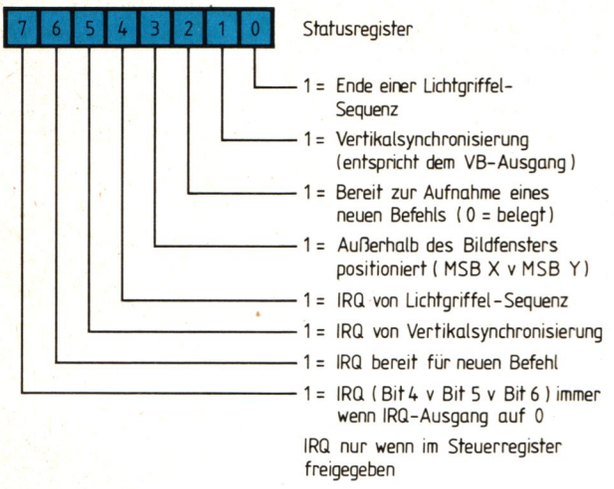


Bild 4. Codierung von Vektorbefehlen (im Register CMD)



◀ Bild 5. Bedeutung der Statusregisterbits

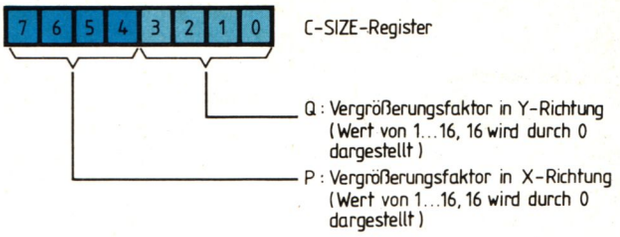
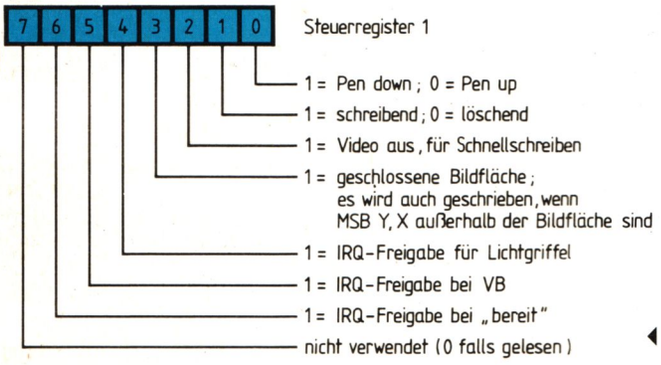


Bild 8. Die Abmessungen der dargestellten Zeichen bestimmt das CSIZE-Register



◀ Bild 6. Bedeutung der Bits im Steuerregister 1

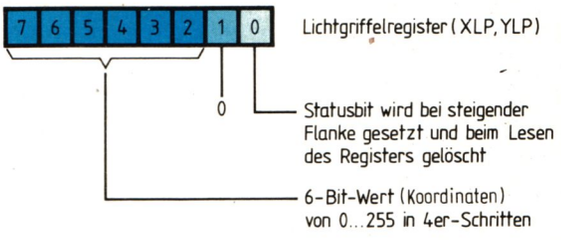


Bild 9. Bedeutung der Bits in den Lichtgriffelregistern XLP und YLP

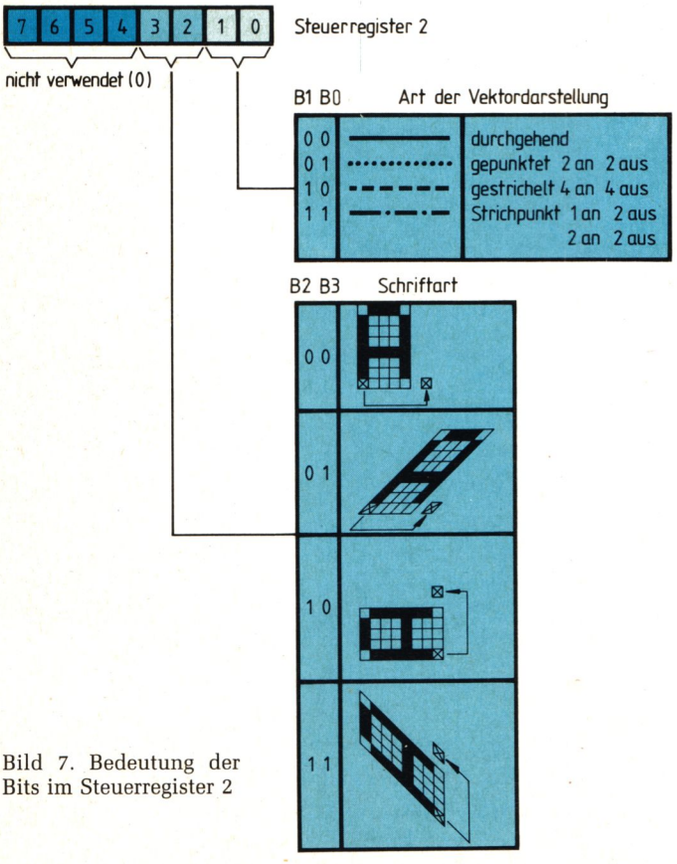
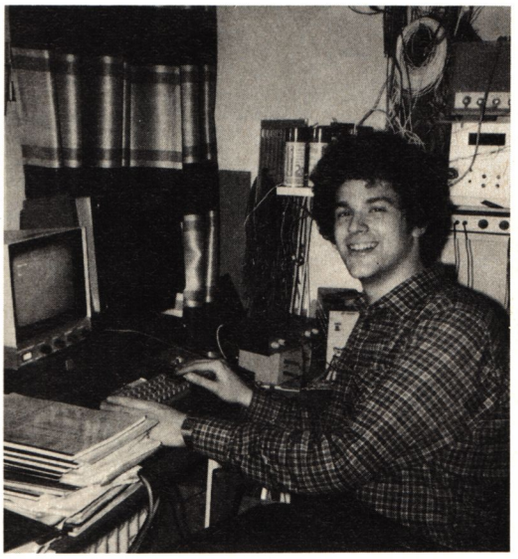


Bild 7. Bedeutung der Bits im Steuerregister 2



Rolf-Dieter Klein, geboren in Tettnang/Bodensee, legte 1978 das Vordiplom an der TU München ab und studiert jetzt im siebten Semester Elektrotechnik. Er beschäftigt sich seit 1975 mit Mikrocomputern und arbeitet als freier Mitarbeiter in der Hard- und Softwareentwicklung sowie in der Systemgenerierung.
Hobby: Filmen
Telefon: (0 89) 3 51 94 16

Vor der Ausgabe eines Befehls oder vor dem Lesen eines Registers muß die CPU Bit 2 dieses Registers abfragen und so lange warten, bis es angibt, daß der GDP nicht mehr beschäftigt ist. Aus diesem Register können auch Informationen wie VG (*Vertikal Blank*) oder Interrupt-Zustände abgefragt werden. Dabei ist VB besonders von Nutzen, wenn man bei bewegten Bildern Einschreibvorgänge synchron zum Bildwechsel durchführen will.

Mit dem Steuerregister 1 (*Control Register 1*) können Interrupts freigegeben und Schreibarten festgelegt werden (*Bild 6*). Die Bits 0 und 1 können dabei auch durch die Kommandos 0, 1, 2, 3 belegt werden. Bit 3 ermöglicht ein Abschalten der Bildfläche für besonders hohe Schreibgeschwindigkeiten.

Bild 7 zeigt das Steuerregister 2 (*Control Register 2*). In Bit 0 und 1 wird die Vektordarstellung codiert, die von „durchgehend" bis „strichpunktiert" reicht

```
                        .phex
                        .pabs
0100                    .loc 100h
                        ;* Beispielprogramm
                        ;* Rolf - Dieter Klein   25.1.81
0100    C3 017F         jmp start       ;hauptprogramm
0103    C3 F003         ci: jmp 0f003h  ;eingabe von Konsole nach a
0106    C3 F009         co: jmp 0f009h  ;ausgabe auf Konsole von c
                        .insert vidins.asm      ;insert videopackage
0070                    @port = 70h     ;16 Adressen belegt von da ab
0070                    @status=port
0070                    @cmd=port
0071                    @ctrl1=port+1
0072                    @ctrl2=port+2   ;befehls Kanaele
0073                    @csize=port+3   ;character size
0075                    @deltax=port+5  ;Abstand x
0077                    @deltay=port+7  ;Abstand y
0078                    @msbx=port+8    ;x register hoeherwertig
0079                    @lsbx=port+9    ;x register niederwertig
007A                    @msby=port+0ah  ;y register hoeherwertig
007B                    @lsby=port+0bh  ;y register niederwertig
007C                    @xlp=port+0ch   ;x lightpen
007D                    @ylp=port+0dh   ;y lightpen
                        @;
                        @;
0000                    @penset=0  ;CMD Code Definitionen
0001                    @eraset=1  ;eraser select
0002                    @pendow=2  ;eraser/pen down
0003                    @penup= 3  ;eraser/pen up
0004                    @clear= 4  ;clear screen
0005                    @resxy= 5  ;reset x,y to 0
0006                    @resclr=6  ;reset x,y to 0 and clear screen
0007                    @ctrall=7  ;all reg to 0,csize to min,clear screen
0008                    @lighini=8 ;light pen init
0009                    @lighin2=9 ;light pen init
000A                    @blank=0ah ;5 x 8 acc csize field
000B                    @squar=0bh ;4 x 4 field
000C                    @scan=0ch  ;screen scanning acc pen,eraser
000D                    @resx=0dh  ;reset x
000E                    @resy=0eh  ;reset y
000F                    @dma=0fh   ;dma req for next free cycle
                        @;
0109                @wait:              ;warten bis GDP fertig
0109    F5          @push psw
010A                @..lp:
010A    DB70        @in status
010C    E604        @ani 00000100b      ;busy flag
010E    CA 010A     @jz ..lp            ;low dann warten bis fertig
0111    F1          @pop psw            ;ok fertig
0112    C9          @ret
                    @;
0113                @cmdout:        ;in A Befehl der auf CMD ausgegeben
                    @               ;werden soll.
0113    CD 0109     @call wait      ;warten bis GDP fertig mit vorgehenden
                     ef
0116    D370        @out cmd        ;ausgabe A
0118    C9          @ret
0119                @init:
0119    3E07        @mvi a,ctrall
011B    CD 0113     @call cmdout
011E    3E03        @mvi a,3        ;pen down, pen
0120    CD 0109     @call wait
0123    D371        @out ctrl1
0125    C9          @ret
                    @;
                    @;
0126                @setosi:        ;set csize in AKK
0126    CD 0109     @call wait
0129    D373        @out csize
012B    C9          @ret
                    @;
                    @;
012C                @print:         ;ausgabe von texten und cmd sequenzen
                    @               ;ht zeigt auf den start des buffers
                    @               ;0fh (dma) ist ende zeichen, da nicht
                    @               ; gebraucht wird.
012C    7E          @mov a,m
012D    FE0F        @cpi dma
012F    C8          @rz             ;ende
0130    CD 0113     @call cmdout    ;ausgabe des befehls
0133    23          @inx h          ;next adress
0134    C3 012C     @jmp print      ;bis 0fh Kommt.
                    @;
                    @;
0137                @setxy:         ;x und y belegen
                    @               ;H=y und L=x msb auf 0 setzen
0137    CD 0109     @call wait
013A    AF          @xra a
013B    D378        @out msbx
013D    D37A        @out msby
013F    7D          @mov a,l
0140    D379        @out lsbx
0142    7C          @mov a,h
0143    D37B        @out lsby
0145    C9          @ret
                    @;
0146                @setdel:        ;delta x,y set H=y L=x
0146    CD 0109     @call wait
0149    7D          @mov a,l
014A    D375        @out deltax
014C    7C          @mov a,h
014D    D377        @out deltay
014F    C9          @ret
                    @;
                    @;
0150                @rechteck:      ;x,y auf linker unterer Ecke
                    @               ;h=hoehe l=breite
0150    CD 0146     @call setdel    ;deltax und deltay belegen
0153    3E10        @mvi a,00010000b ;x>0 y=0
0155    CD 0113     @call cmdout
0158    3E12        @mvi a,00010010b ;x=0 y>0
015A    CD 0113     @call cmdout
015D    3E16        @mvi a,00010110b ;x<0 y=0
015F    CD 0113     @call cmdout
0162    3E14        @mvi a,00010100b ;x=0 y<0
0164    C3 0113     @jmp cmdout     ;ende rechteck
                    @;
0167                @line:          ;zeile von links nach rechts
                    @               ;laenge in l . x,y bereits gesetzt
0167    CD 0109     @call wait
016A    7D          @mov a,l
016B    D375        @out deltax     ;belegen nur x
016D    3E10        @mvi a,00010000b ;nur deltax y=0 mit x>0
016F    C3 0113     @jmp cmdout
                    @;
                    @;
0172                @vektor:        ;in a vorzeichenmaske
                    @               ; 0 : x,y >0
                    @               ; 1 : x<0 ,y>0
                    @               ; 2 : x>0 ,y<0
                    @               ; 3 : x>0 ,y>0
                    @               ; in h y betrag
                    @               ; in l x betrag
0172    F5          @push psw
0173    CD 0146     @call setdel    ;richtung belegen
0176    F1          @pop psw
0177    07          @rlc
0178    E606        @ani 00110b             ;maske
017A    F611        @ori 00010001b  ;Basis Befehl
017C    C3 0113     @jmp cmdout
                    @;
                    @
                    @
                    @; ende insert package
                    @;
017F                 start:
017F    CD 0119      call init      ;init GDP
0182    3E22         mvi a,22h      ;CSIZE doppelt hohe u breite Sch
                     rift
0184    CD 0126      call setosi
0187    26E6         mvi h,230      ;y
0189    2E0A         mvi l,10       ;x
018B    CD 0137      call setxy
018E    21 028B      lxi h,heading
0191    CD 012C      call print     ;Ausgabe Ueberschrift
                     ;
                     ; schraeg Schrift
0194    3E23         mvi a,23h              ;zwei breit x und drei y
0196    CD 0126      call setosi
0199    3E64         mvi a,0100             ;mode
019B    CD 0109      call wait
019E    D372         out ctrl2
                     ;
01A0    26C8         mvi h,200
01A2    2E0A         mvi l,10
01A4    CD 0137      call setxy
01A7    21 0295      lxi h,subtitle
01AA    CD 012C      call print
                     ; umranden
01AD    26C8         mvi h,200      ;lxi h,  besser aber hier wegen
01AF    2E04         mvi l,4        ;  Verstaendniss mit mvi
01B1    CD 0137      call setxy
01B4    2630         mvi h,48               ;hoehe
01B6    2E76         mvi l,118              ;breite
01B8    CD 0150      call rechteck          ;umranden
                     ;
01BB    3E11         mvi a,11h              ;normale schrift
01BD    CD 0126      call setosi
01C0    3E00         mvi a,0000     ;mode schrift
01C2    CD 0109      call wait
01C5    D372         out ctrl2
                     ;
                     ;alle zeichen ausgeben
01C7    26BC         mvi h,188
01C9    2E0A         mvi l,10
01CB    CD 0137      call setxy
01CE    3E20         mvi a,20h              ;20h bis 3fh
01D0    0620         mvi b,32
01D2                 ..lp:
01D2    F5           push psw
```

und für alle Vektorbefehle gilt. Die Bits 2 und 3 bestimmen die Schriftart für den Zeichengenerator. Dabei kann zwischen „horizontal“ und „vertikal“ gewählt werden sowie zwischen „*gerade*“ und „*kursiv*“. Geschrieben wird immer bei der aktuellen X/Y-Position. Nach dem Schreiben rückt die Position auf die markierte Stelle vor.

Ein Register, das auch für den Zeichengenerator Bedeutung hat, ist in *Bild 8* dargestellt. Es bestimmt die Abmessungen der Zeichen. Für jede Richtung, X und Y, kann ein eigener Vergrößerungsfaktor (1...16) programmiert werden. X- und Y-Vorschub werden immer entsprechend ausgeführt. Die Belegung der Lichtgriffelregister gibt *Bild 9* wieder. Hier werden die Koordinaten in 4er-Schritten abgelegt. Ein Beispielprogramm zeigt *Bild 10*.

Literatur

[1] Datenblätter der Firma Thomson-CSF/EFCIS

```
0103    C5              push b
01D4    CD 0113         call cmdout
01D7    C1              pop b
01D8    F1              pop psw
01D9    3C              inr a
01DA    05              dcr b
01DB    C2 0102         jnz ..lp
                        ;
01DE    26B4            mvi h,180
01E0    2E0A            mvi l,10
01E2    CD 0137         call setxy
01E5    3E40            mvi a,40h
01E7    0620            mvi b,32
01E9                    ..lp1:
01E9    F5              push psw
01EA    C5              push b
01EB    CD 0113         call cmdout
01EE    C1              pop b
01EF    F1              pop psw
01F0    3C              inr a
01F1    05              dcr b
01F2    C2 01E9         jnz ..lp1
                        ;
01F5    26AC            mvi h,172
01F7    2E0A            mvi l,10
01F9    CD 0137         call setxy
01FC    3E60            mvi a,60h
01FE    0620            mvi b,32
0200                    ..lp2:
0200    F5              push psw
0201    C5              push b
0202    CD 0113         call cmdout
0205    C1              pop b
0206    F1              pop psw
0207    3C              inr a
0208    05              dcr b
0209    C2 0200         jnz ..lp2
                        ;
                        ; gestrichelte linie ausgeben
020C    3E03            mvi a,0011b     ;strichpunkt - . - .
020E    CD 0109         call wait
0211    D372            out ctrl2
0213    26AA            mvi h,170       ;y
0215    2E00            mvi l,0         ;ganzen bereich
0217    CD 0137         call setxy
021A    2EFF            mvi l,255       ;breite
021C    CD 0167         call line       ;
                        ;
021F    2664            mvi h,100       ;noch eine
0221    2E00            mvi l,0
0223    CD 0137         call setxy
0226    2EFF            mvi l,255
0228    CD 0167         call line
                        ;
                        ; animated picture
                        ;
022B    3E00            mvi a,0000b     ;linie normal
022D    CD 0109         call wait
0230    D372            out ctrl2
                        ;
0232    2687            mvi h,135       ;start hoehe
0234    2E00            mvi l,0         ;links
                        ;
0236    06E6            mvi b,230       ;anzahl bewegeungen
                        ;
0238                    ..lpm:
0238    CD 0137         call setxy
023B    3E01            mvi a,erasel
023D    CD 0113         call cmdout     ;loeschmode
0240    E5              push h
0241    C5              push b
0242    21 029D         lxi h,fig       ;erst loeschen
0245    CD 012C         call print      ;damit am schluss stehen bleibt
0248    C1              pop b
0249    E1              pop h
                        ;
                        ; verschieben um eins nach rechts
024A    2C              inr l
024B    CD 0137         call setxy
                        ;
024E    3E00            mvi a,pensel    ;setzen
0250    CD 0113         call cmdout
0253    E5              push h
0254    C5              push b
0255    21 029D         lxi h,fig
0258    CD 012C         call print
025B    C1              pop b
025C    E1              pop h
                        ;
025D    E5              push h
025E    C5              push b
025F    7D              mov a,l         ;x
0260    E60F            ani 0fh
0262    2011            jrnz ..skp
                                        ;alle 16 mal
```

```
0264    7D              mov a,l
0265    C608            adi 11
0267    6F              mov l,a
0268    7C              mov a,h
0269    C609            adi 9
026B    67              mov h,a
026C    CD 0137         call setxy
026F    21 0283         lxi h,fig1
0272    CD 012C         call print
0275                    ..skp:
0275    C1              pop b
0276    E1              pop h
                        ;
                        ;
0277    11 0320         lxi d,800
027A    CD 0284         call warte      ;warten
027D    05              dcr b
027E    C2 0238         jnz ..lpm
                        ;
                        ;
                        ;
                        ;
0281    CD F81E         call 0f81eh             ;monitor
                        ;
0284                    warte:
0284    1B              dcx d
0285    7B              mov a,e
0286    B2              ora d
0287    C2 0284         jnz warte
028A    C9              ret
                        ;
                        ;
                        ; Texte und Daten
                        ;
028B                    heading:
028B    474456202039    .ascii 'GDP  93\
0291    333635          \65'
0294    0F              .byte dma               ;ende zeichen
0295                    subtitle:
0295    54686F6D736F    .ascii 'Thomson'\
029B    6E              \
029C    0F              .byte dma
                        ;
                        ; Figure fuer animated picture
                        ;
029D                    fig:
                        ;1 dx dx dy dy dir dir dir
                        ;
029D    FA              .byte 11111010b
029E    D2              .byte 11010010b
029F    A9              .byte 10101001b
02A0    F8              .byte 11111000b
02A1    AD              .byte 10101101b
02A2    D4              .byte 11010100b
02A3    F8              .byte 11111000b
02A4    F8              .byte 11111000b
02A5    D2              .byte 11010010b
02A6    D0              .byte 11010000b
02A7    D4              .byte 11010100b
02A8    AD              .byte 10101101b
02A9    D4              .byte 11010100b
02AA    AF              .byte 10101111b
02AB    AE              .byte 10101110b
02AC    AC              .byte 10101100b
02AD    AE              .byte 10101110b
02AE    AA              .byte 10101010b
02AF    FE              .byte 11111110b
02B0    D6              .byte 11010110b
02B1    FE              .byte 11111110b
02B2    AC              .byte 10101100b
02B3    AE              .byte 10101110b
02B4    AA              .byte 10101010b
02B5    AE              .byte 10101110b
02B6    AB              .byte 10101011b
02B7    0F              .byte dma               ;ende figure
                        ;
02B8                    fig1:
02B8    B3              .byte 10110011b
02B9    A9              .byte 10101001b
02BA    D0              .byte 11010000b
02BB    AD              .byte 10101101b
02BC    D7              .byte 11010111b
02BD    AE              .byte 10101110b
02BE    0F              .byte dma
                        ;
                        ;
                        .end
```

Bild 10. Beispielprogramm für die Programmierung des Grafikprozessors, es beinhaltet einige nützliche Moduln zur Ansteuerung. Das Hauptprogramm erzeugt das Foto auf Seite 63. Im Listing sind die allgemein verwendbaren Moduln mit dem Zeichen @ versehen

11. *Abb. 5.8* zeigt das Oszillogramm einer einzelnen Zeile. Dabei müssen dort Zeichen handen sein. Dazu wurde hier mit einem verzögerten Triggersignal gearbeitet und gegert wurde mit Hilfe des gesamten BAS-Signals aber auf dem vertikalen Abschnitt. d nun auf das Zeilensignal ohne Verzögerung getriggert, so ist die Zeile im Prinzip h so erkennbar, aber der Informationsinhalt nicht so klar zu sehen. Erscheint das Sigwie angegeben, so muß auf dem Monitor bereits ein Bild sichtbar sein.

12. Ist die Tastatur nach dem vorherigen Abschnitt aufgebaut und stimmt die Schalstellung, so müssen sich nun bei Vorhandensein des Monitorprogramms Befehle einen lassen und auf dem Bildschirm erscheinen. Nach dem RESET muß der Bildschirm ächst gelöscht werden und dann die Monitormeldung in der oberen Hälfte erschei-. Wird eine Taste der Tastatur betätigt, so muß der Befehl auf dem Bildschirm ausgeen werden. Test z. B. Eingabe der Sequenz:

D100 200cr

steht dabei für die Betätigung der Taste carriage return (Wagenrücklauf).
Auf dem Bildschirm muß der Inhalt des Speicherbereichs 100 bis 200 in hexadezimaler reibweise ausgegeben werden, siehe auch den Softwareabschnitt zum Monitorprogramm.

graphisches Datensichtgerät

sentlich mehr Möglichkeiten als das alphanumerische Sichtgerät bietet eines mit grahischen Fähigkeiten. Zum Glück gibt es heute sehr hoch integrierte Bausteine, die einen fbau sehr erleichtern. Wir verwenden einen sogenannten Graphik-Prozessor, den 9366, der neben den Bildsignalen auch Vektoren und Buchstaben in den Bildwiederlspeicher einschreiben kann und dies sehr schnell.

Bei dem alphanumerischen Datensichtgerät wurde jedem Code in Speicher ein Zeichen dem Bildschirm zugeordnet. Ein Zeichen bestand aus mehreren Bildpunkten. Wir llen nun jedem Bit des Bildwiederholspeichers genau einen Punkt auf dem Bildschirm rdnen.

Dazu benötigt man natürlich einen viel größeren Bildwiederholspeicher. Sollen 512 mal 6 Punkte dargestellt werden, so werden 131072 Speicherzellen benötigt, also 128K Bit K = 1024).

Wenn man diesen Speicher in 8-er Gruppen aufteilt, so ergibt sich eine Kapazität von K Byte.

Bei unserem Datensichtgerät wollen wir noch einen Schritt weiter gehen: wir wollen unabhängige Bildebenen darstellen. *Abb. 5.3.21* zeigt das Schema. Dann kann man nlich in einer Bildebene ein Bild einschreiben und gleichzeitig eine andere betrachten. benötigen somit 64K Byte Bildwiederholspeicher. Von den vier Bildebenen kann aber mer nur eine sichtbar sein. Sollen zwei scheinbar gleichzeitig abgebildet werden, so muß n sie schnell, am besten im 20ms-Takt hin- und herschalten.
Nun aber zum Aufbau des Graphik-Datensichtgerätes.

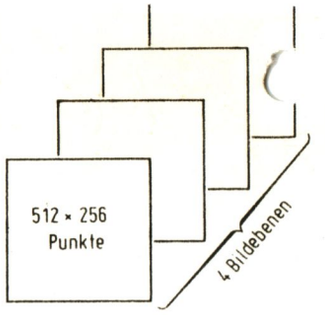


Abb. 5.3.21 Bildebenen

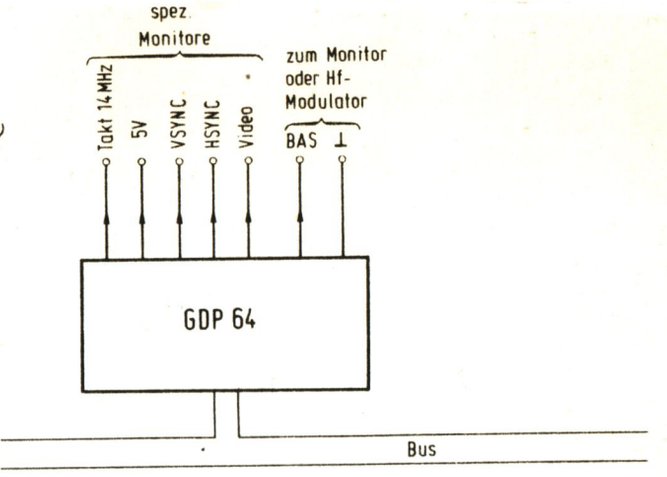


Abb. 5.3.22 Die GDP64-Baugruppe

Abb. 5.3.22 zeigt den Anschluß an die Außenwelt. Die Leiterplatte GDP64 liefert mehrere Ausgangssignale. Zum einen das schon bekannte BAS-Signal, das an einem Monitor oder HF-Modulator geleitet werden kann und die komplette Bildinformation beinhaltet. Dann sind aber noch die Signale Video, -HSYNC, -VSYNC, +5V und Takt herausgeführt. Diese Leitungen können für spezielle Monitore oder HF-Modulatoren verwendet werden, die das BAS-Signal nicht direkt verarbeiten können.

Abb. 5.3.23 zeigt die Gesamtschaltung. Der Graphikprozessor steht im Mittelpunkt der Schaltung, links neben ihm befindet sich die Ansteuerschaltung zur Anpassung an den CPU-Bus, rechts der Bildwiederholspeicher und die Takterzeugung.

Die Schaltung arbeitet mit einem 14 MHz Takt, aus dem alle wichtigen Signale abgeleitet werden. Der 74LS163 erzeugt daraus unter anderem einen 1.75 MHz Takt, der den EF 9366 versorgt. Der Bildwiederholspeicher besteht aus dynamischen Speichern des Typs 4164; hier können alle gängigen 64K x 1 Speicher verwendet werden, die einen 128 Refresh-Zyklus haben (im Datenblatt des Herstellers angegebenen). Nicht verwenden kann man dynamische Speicher und einem 256-Refresh-Zyklus. Da auf dem freien Markt beide Speichertypen vorkommen, ist Vorsicht geboten. Die Speicher NEC 4164-2, Hitachi HM 4864, Mitsubishi MK 4164 sind z. B. für uns einsetzbar.

Der Begriff Refresh-Zyklus fiel soeben, was ist das. Wir hatten bisher nur sogenannte statische Speicher verwendet. Bei diesen Speichern geschieht die Informationsspeicherung mit Hilfe von Flip-Flops. Diese Flip-Flops können z. B. aus zwei Transistoren bestehen, von denen immer einer leitet. Je nach dem, welcher der beiden Transistoren bietet, ist eine 0 oder eine 1 gespeichert. Bei den dynamischen Speichern wird die Information in einem Kondensator gespeichert, dem ein Transistor zur Ansteuerung zugeordnet ist. Der Kondensator trägt entweder Ladung oder nicht und kann damit eine Information speichern. Leider entlädt sich der Kondensator in einigen Milisekunden, und die Ladung des Kon-

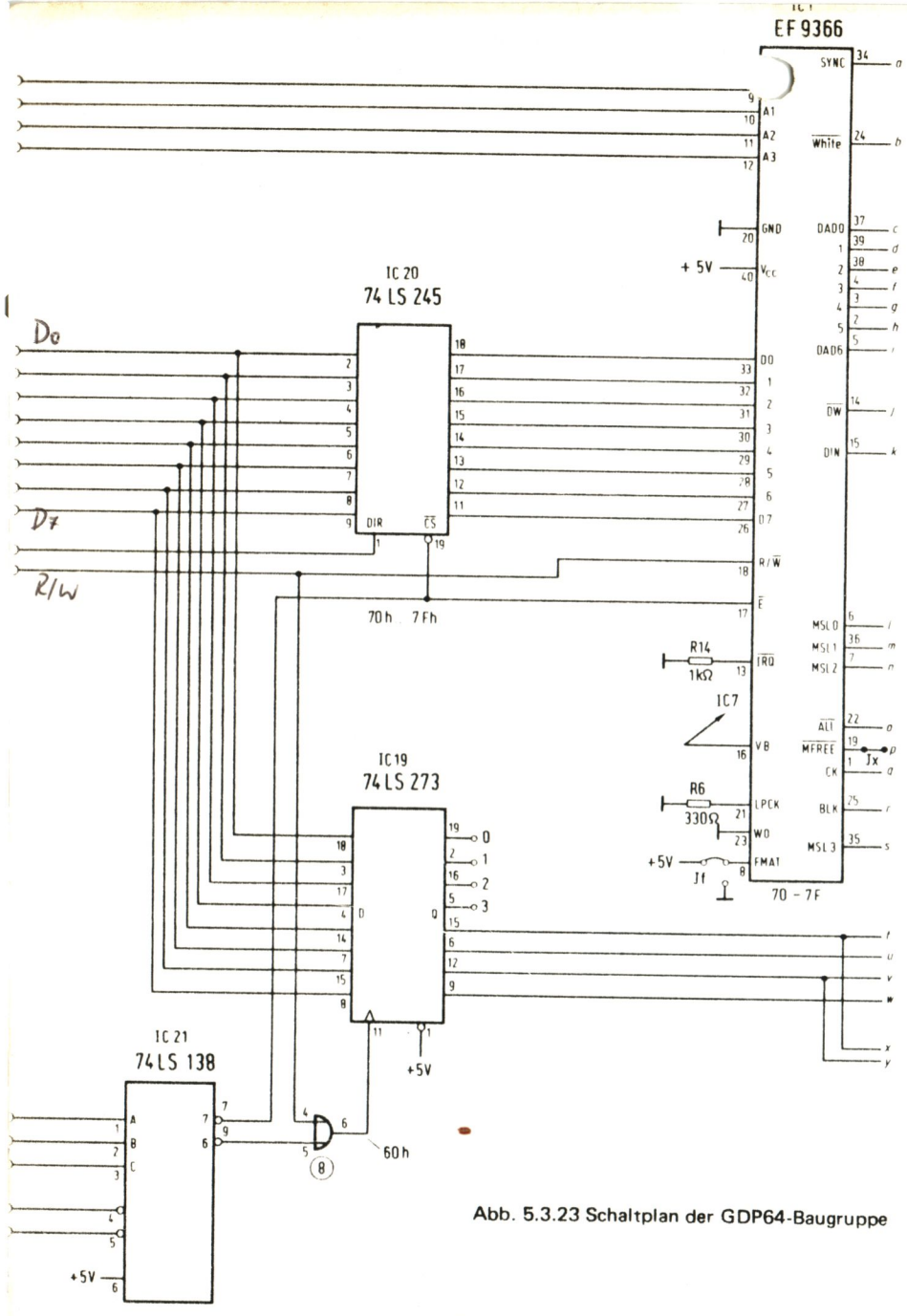
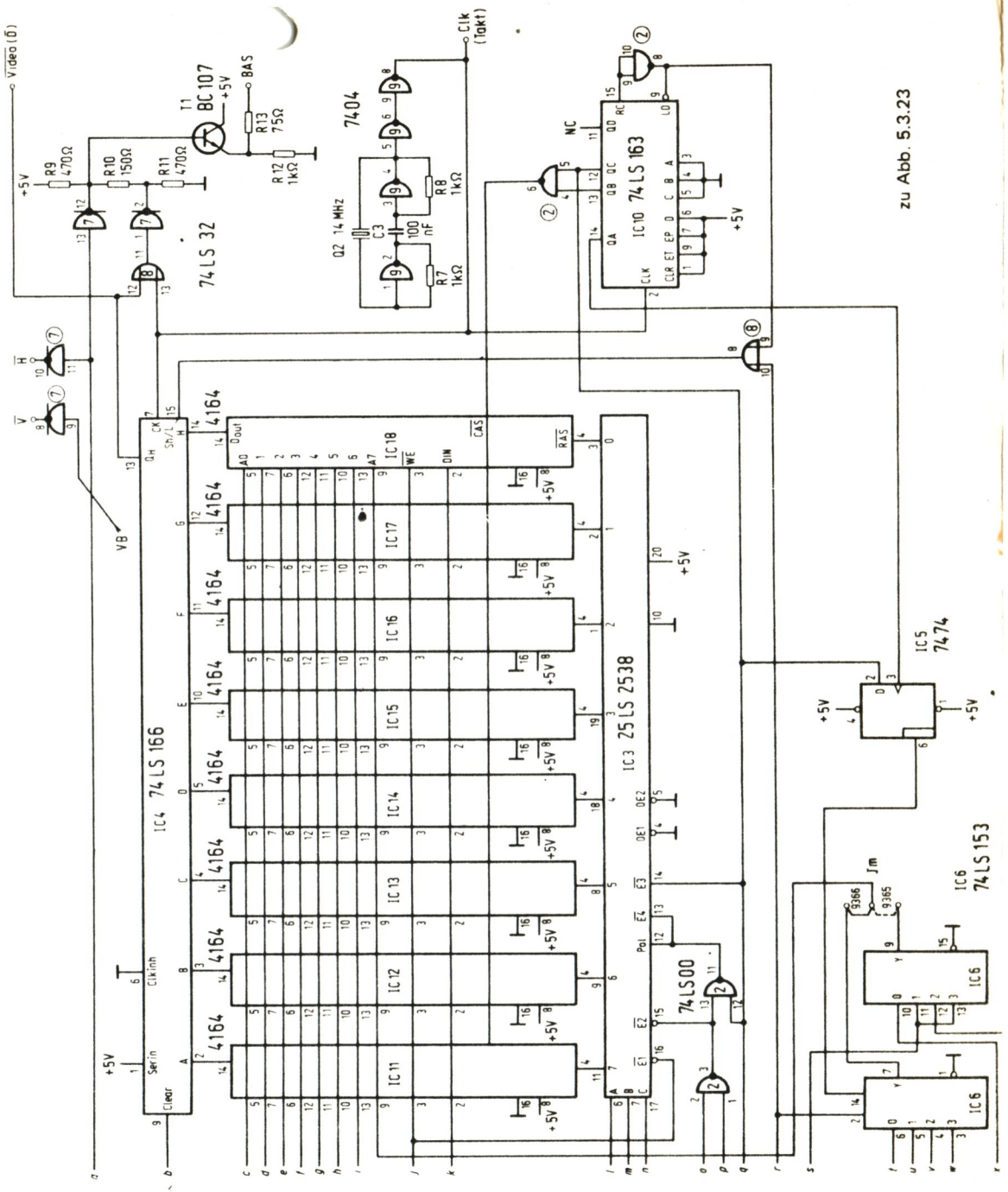


Abb. 5.3.23 Schaltplan der GDP64-Baugruppe



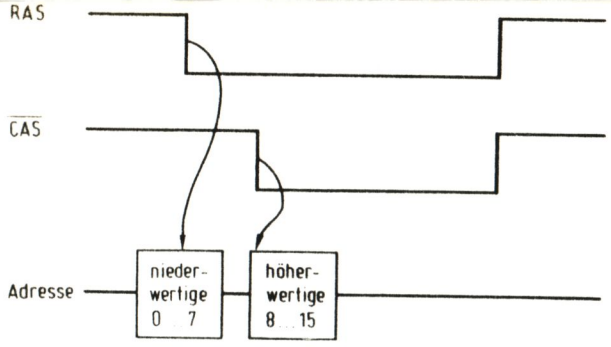


Abb. 5.3.24 Multiplex der Adressen

densators muß periodisch wiederaufgefrischt werden, daher der Begriff REFRESH. Der Refresh geschieht durch Anwahl einer Speicherreihe, wobei je nach Speichertyp 128 Reihen oder 256 Reihen angesprochen werden müssen, und damit automatisch 512 oder 256 Speicherzellen auf einmal regeneriert werden. Die Speicher besitzten nur 8 Adreßeingänge jedoch braucht man 16 Adreßleitungen, um 64K Bit zu adressieren. Die Adressen werden daher zeitlich nacheinander übertragen, und um das zu steuern, gibt es zwei Eingänge, CAS und RAS genannt. Mit RAS wird zuerst die niederwertige Adresse (Row) in den Speicher eingegeben und dort gemerkt; mit CAS wird dann der höherwertige Teil eingespeichert. *Abb. 5.3.24* zeigt den Vorgang. Man sagt auch, die Adressen werden „gemultiplext".

Diesen Vorgang übernimmt bei uns der EF9366, er liefert bereits das fertige Adreßsignal. Der EF9366 ist aber eigentlich nur für eine Bildebene zuständig, daher besitzt er auch nur 6 Adreßausgänge für 16K Adreßraum. Wir wollen aber vier Bildseiten ansprechen. Die Adreßerzeugung übernimmt bei uns das IC6 und ein Teil von IC 5. An das IC5 gelangt eine 4Bit-Seitenadresse vom Zwischenspeicher 74LS273. Die beiden Bits 4 und 5 bestimmen die Leseseite und die Bits 6 und 7 die Schreibseite. Der Multiplexer erhält die Information, ob gelesen oder geschrieben wird, über das Signal BLK. Das Signal DW, das eigentlich den Schreibvorgang kennzeichnet, ist leider zu zeitkritisch. BLK gibt den Rahmen des Bildes an, also ob sich der Schreibstrahl in sichtbaren oder unsichtbaren Teil des Bildes befindet. Da immer nur im unsichtbaren Teil geschrieben wird, ist er auch als Information über einen möglichen Schreibvorgang verwendbar. Beim schnellen Bildschirmlöschen, einem Befehl an den GDP (den Graphic Display Processor), ist jedoch dies nicht der Fall und muß in der Software berücksichtigt werden. Das GRUNDPROGRAMM, wie es im Softwareabschnitt dieses Buches vorgestellt wird, berücksichtigt alle diese Punkte.

Abb. 5.3.25 zeigt ein paar wichtige Signalformen, die auch zur Fehlersuche dienen können.

Der 74LS166 ist ein Schieberegister und dient dazu, die Bildinformation, die dort parallel ankommt, nacheinander herauszuschieben, um das Videosignal zu bilden. Es wird

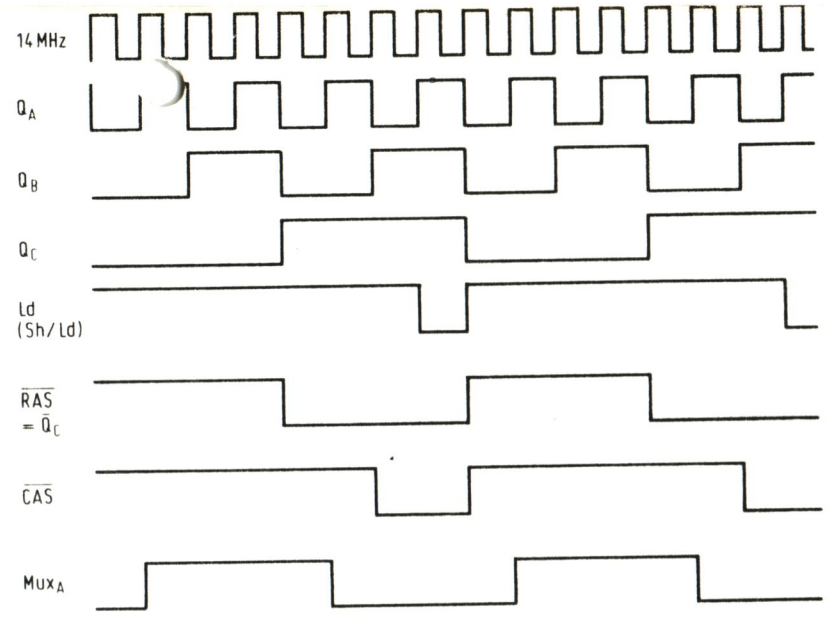


Abb. 5.3.25 Das Timing der GDP-Steuerung

dann zusammen mit den Synchronsignalen zu einem BAS-Signal gemischt, wie wir es schon von der alphanumerischen Anzeige her kannten. Der BAS-Mischer besteht aus den Widerständen R9, R10 und R11. R11 kann auch weggelassen werden; dann ergibt sich eine höhere Ausgangsspannung am BAS-Ausgang, was bei manchen Monitoren günstiger ist und ein helleres Bild ergibt.

Beginnen wir mit dem Aufbau der Baugruppe GDP64. *Abb. 5.3.26* zeigt die Lötseite der Leiterplatte, *Abb. 5.3.27* die Besücktungsseite und *Abb. 5.3.28* den Bestückungsplan, Abb. *5.3.29* zeigt ein Foto der Leiterplatte.

1. Begonnen wird mit dem Einlöten aller IC-Sockel und aller passiven Bauteile (Widerstände, Kondensatoren und Quarz).
2. Einsetzen des IC9 (7404). Nun kann die Spannungsversorgung eingeschaltet werden. Am Ausgang des Oszillators, Pin 8 des ICs 7404, muß ein Takt von 14MHz erscheinen.
3. Wenn ja, so können alle restlichen Bauteile, außer EF9366 und der Speicherbausteine 4164 eingesetzt werden.
4. Einschalten der Spannungsversorgung. An Pin 1 des EF9366 muß ein 1.75 MHz Signal erscheinen.
5. Pin 9 des IC 10 muß kleine, nach DV gehende Impulse tragen, wie in Abb. 5.3.25 (Ld) gezeigt.
6. Nun kann der Graphikprozessor EF 9366 eingesetzt werden. Zuvor unbedingt die Spannungsversorgung abschalten.

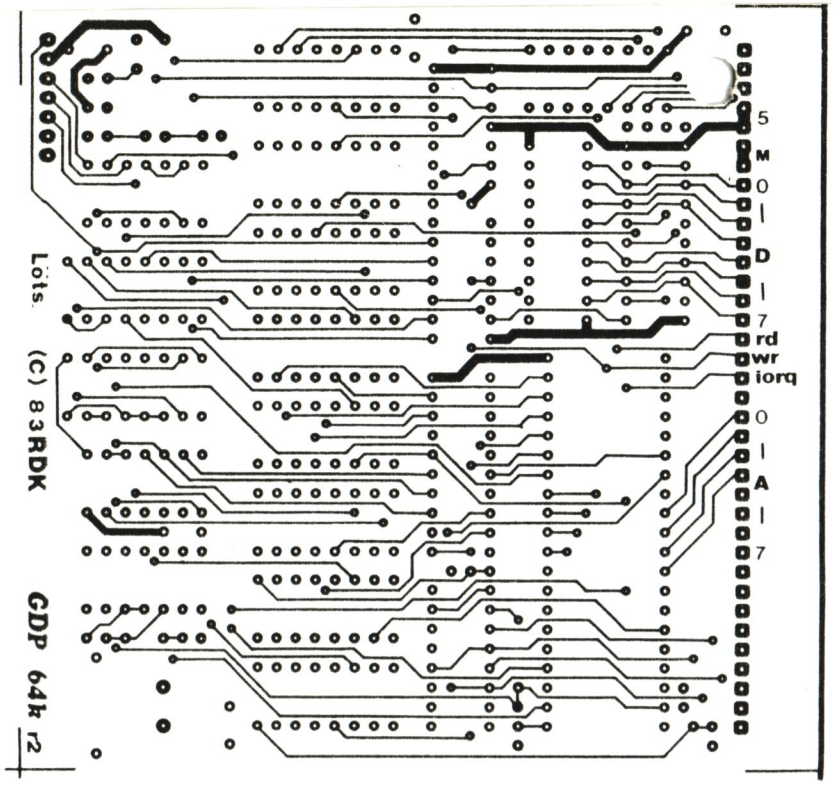


Abb. 5.3.26 Lötseite der Leiterplatte GDP64

7. Nach Abschluß des BAS-Signals an einen Monitor muß nach dem Einschalten der Spannungsversorgung auf dem Bildschirm ein dunkles, leicht abgegrenztes Feld erscheinen. Nicht alle Bildschirme lassen es erkennen, jedoch muß in jedem Fall ein stehendes Bild erzeugt werden. Mit dem Oszilloscop müssen am BAS-Ausgang die Synchronimpulse erkennbar sein, jedoch noch ohne Bildinformation. Wird nun z. B. an einem der ICs 11 bis 17 (Speicherbausteine) PIN 14 auf Masse geschaltet, so müssen senkrechte Linien auf dem Bildschirm erscheinen. Wird Pin 14, eines der Speicherbausteine, auf eine der Adressen DAD0 bis DAD6 geschaltet, so muß sich ein abgegrenztes Bildmuster ergeben. Darin können auch kleine Störungen sichtbar sein, die vom Multiplexvorgang des GDPs herrühren.

8. Messen an Pin 8 der Speicher-ICs. Dort liegt +5V. Die Speicher-ICs haben nicht die gängige IC-Beschaltung; die Polarität ist gerade umgekehrt.

9. Einsetzen der Speicher-ICs. Erneutes Einschalten der Versorgungsspannung.

10. Ist bereits das GRUNDPROGRAMM aus dem Softwarekapitel mit der SBC-2-Karte vorhanden, so muß auf dem Bildschirm nach einer Copyright-Meldung ein blinken-

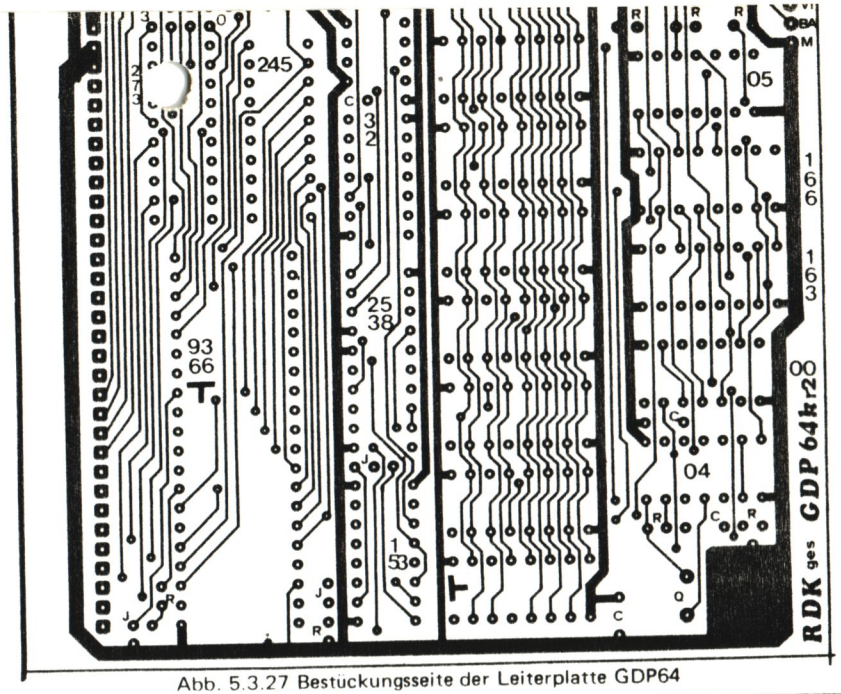


Abb. 5.3.27 Bestückungsseite der Leiterplatte GDP64

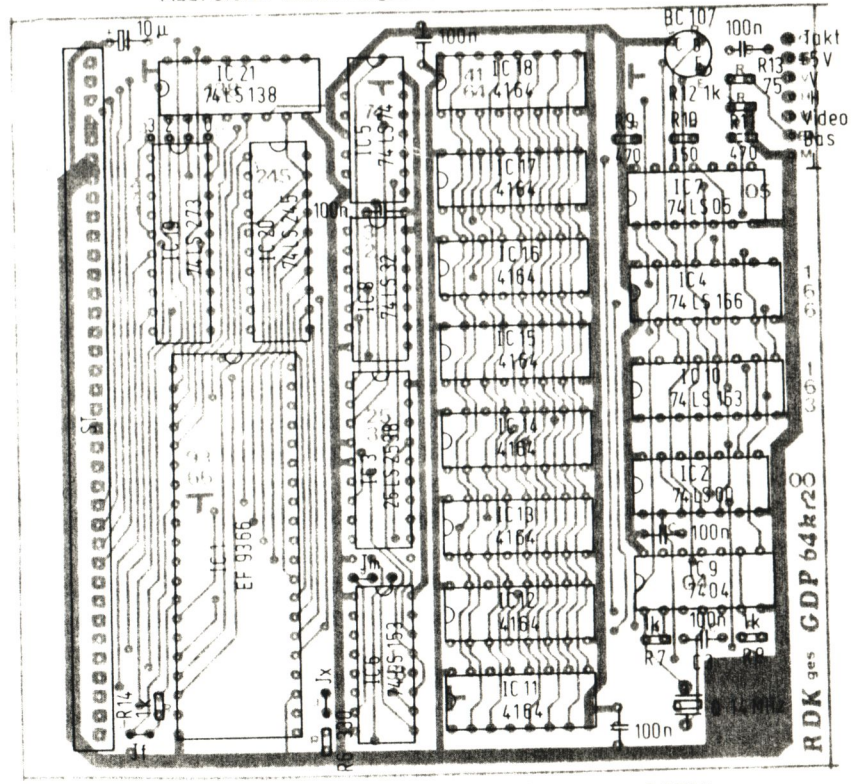


Abb. 5.3.28 Bestückungsplan der Leiterplatte GDP64

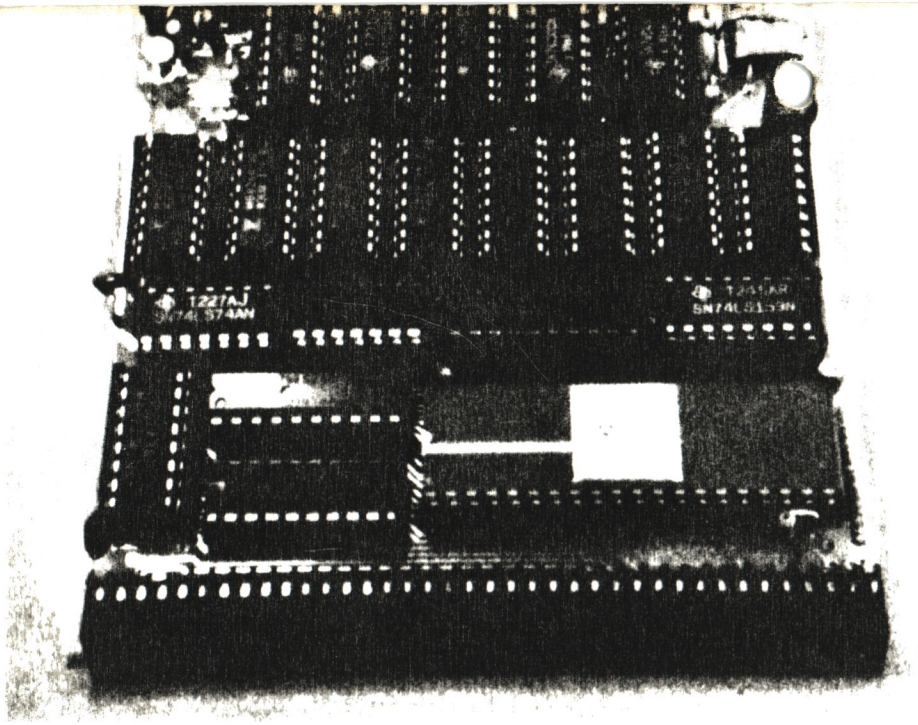


Abb. 5.3.29 Die Baugruppe GDP64

der großer Cursor mit dem Grund-Menü erscheinen, über die Tastatur, die an die Baugruppe KEY angeschlossen wurde, lassen sich dann Befehle eingeben.

11. Wird die Karte nicht mit dem Grundprogramm zusammen verwenden, so ergibt sich auch eine einfache Testmöglichkeit. Die Ausgabe des Wertes 0 an den Port 60H setzt die Schreib- und Leseseite auf 0.

12. Ausgabe des Wertes 6 an den Port 70h löscht den Bildschirm und setzt die Schreibposition auf 0,0.

13. Ausgabe des Wertes 3 an den Port 71h setzt den GDP in den Schreibmode.

14. Ausgabe des Wertes 41h an den Port 70h läßt das Zeichen „A" auf dem Bildschirm links unten erscheinen. Damit ist die Karte ganz gut getestet.

Diese Testausgaben lassen sich z. B. mit dem Monitor, der mit dem CRT64 zusammenarbeitet, als Q0-Befehle geben. Werden sie in einem EPROM festgehalten, so muß zuvor das Status-Register abgefragt werden.

Abb. 5.3.30 zeigt das vollständige Testprogramm, das in einem EPROM abgelegt werden kann. Es muß ein kleines A und ein großes B auf dem Bildschirm erscheinen. Das Programm ist an jeder Speicherstelle ablauffähig.

Abb. 5.3.31 zeigt die Belegung des Seitenauswahl-Ports. Er wird über die Adresse 50h angesprochen. Die unteren vier Bits sind nicht verwendet. Auf der Leiterplatte befinden

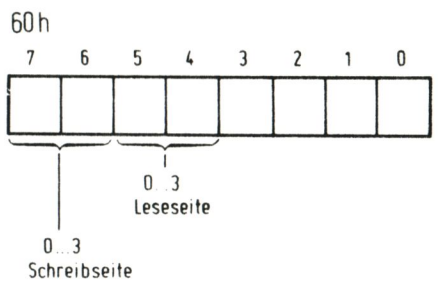

```
                        ;*************************
                        ;* EF9366 Testprogramm   *
                        ;*************************

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

0000'  AF               start:  xor a
0001'  D3 60                    out (seite),a   ;Seite 0 verwenden
0003'  DB 70            wai1:   in a,(gdp)      ;warten bis GDP
0005'  E6 04                    and 4           ;fertig
0007'  28 FA                    jr z,wai1       ;dann Befehl
0009'  3E 06                    ld a,6          ;Loeschen
000B'  D3 70                    out (gdp),a     ;dann warten
000D'  DB 70            wai2:   in a,(gdp)
000F'  E6 04                    and 4
0011'  28 FA                    jr z,wai2       ;bis geloescht
0013'  3E 03                    ld a,3          ;PEN down
0015'  D3 71                    out (gdp+1),a   ;PEN -Mode
0017'  3E 41                    ld a,41h        ;Zeichen A
0019'  D3 70                    out (gdp),a     ;ausgeben
001B'                   wai3:                   ;Gross auch
001B'  DB 70                    in a,(gdp)
001D'  E6 04                    and 4
001F'  28 FA                    jr z,wai3       ;erst wenn fertig
0021'  3E 00                    ld a,0          ;* 16
0023'  D3 73                    out (gdp+3),a
0025'  3E 42                    ld a,42h        ;B
0027'  D3 70                    out (gdp),a     ;und ausgeben
0029'  76                       halt            ;FERTIG
```

Abb. 5.3.30 Testprogramm für die GDP64-Baugruppe

Abb. 5.3.31 Belegung des Seitenports



sich 4 Lötaugen, an denen diese vier Bits für eigene Erweiterungen zur Verfügung stehen. Wird die Leseseite unterschiedlich zur Schreibseite ausgewählt, so erfolgt ein unsichtbarer Schreibvorgang. Damit ist es möglich, ein Bild aufzubauen, ohne daß der Aufbauvorgang sichtbar wird. Nach dem Bildaufbau wird dann einfach die Lese-Seite gleich der Schreibseite gesetzt und das Bild wird sichtbar.

Nun zur Beschreibung des Graphik-Prozessors. Den inneren Aufbau zeigt *Abb. 5.3.32* in einem Blockschema.

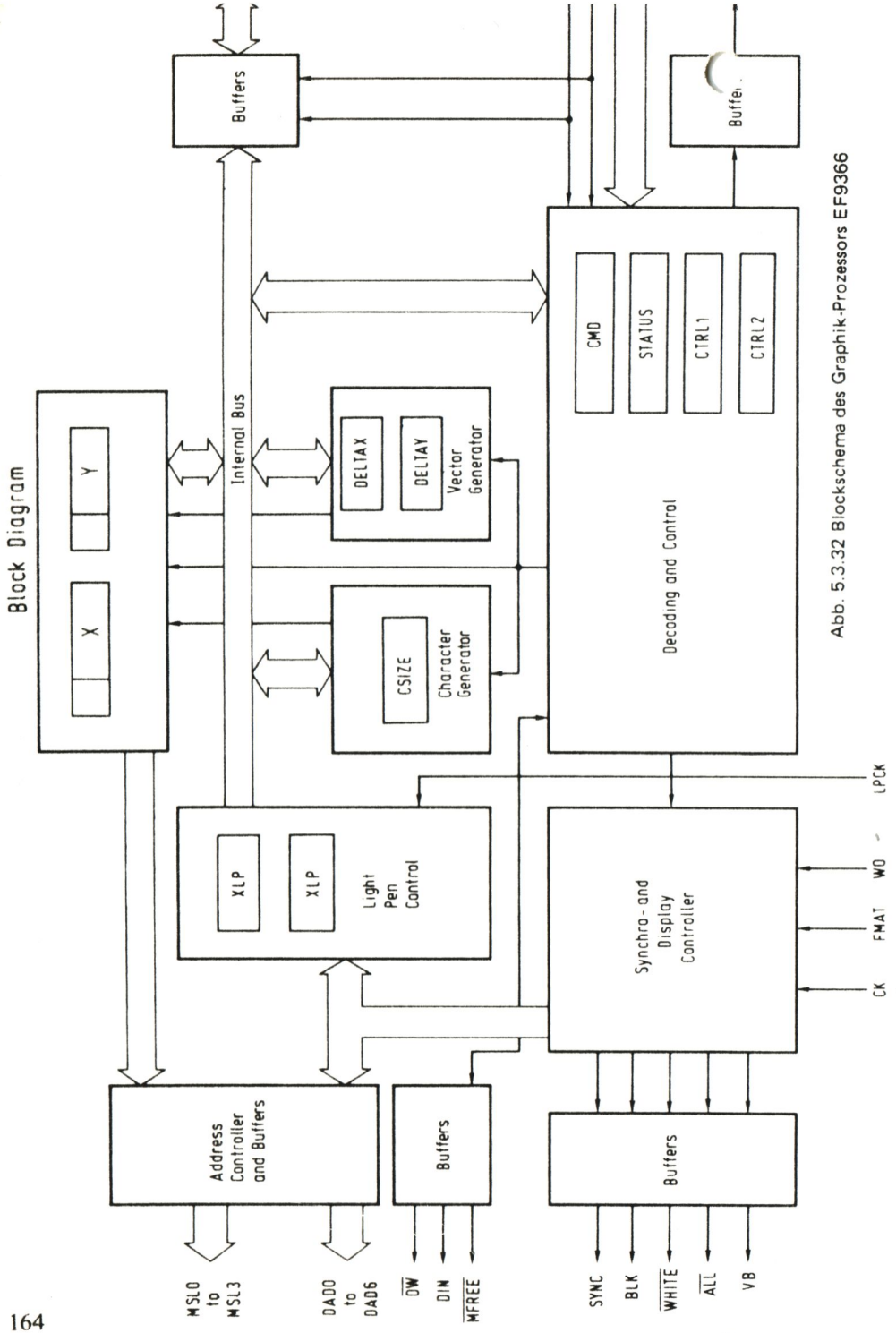


Abb. 5.3.32 Blockschema des Graphik-Prozessors EF9366

Links sind die Steuerleitungen zur Bildschirmsteuerung eingezeichnet und rechts der Teil für die CPU-Anpassung. Unten sind noch ein paar Steuerleitungen eingezeichnet. An den Eingang LPCK kann ein sogenannter Lichtgriffel angeschlosen werden. Das ist ein schreibstiftförmiger Aufnehmer, der ein photoempfindliches Bauteil enthält. Wird damit der Bildschirm berührt, ergibt sich am Ausgang ein Impuls, sobald der Strahl der Bildröhre am Aufnehmer vorbeikommt. Dadurch ist die Position des Lichtgriffels bestimmt und kann vom GDP weiter verwendet werden.

FMAT dient zur Formatauswahl; beim EF 9366 jedoch muß der Eingang auf +5V liegen. Es gibt noch ein paar andere GDPs der gleichen Serie. Mit dem EF 9365 läßt sich ein Bildfenster von 512 mal 512 Bildpunkten darstellen; auch dazu ist unsere Schaltung fähig, es muß nur die Brücke Jm umgestellt werden. Dann hat man zwei Bildseiten mit 512 mal 512 Bildpunkten. Die Software im GRUNDPROGRAMM unterstützt diesen Prozessor jedoch nicht. Dann gibt es noch den neueren EF 9367. Er arbeitet mit 12MHz anstelle des 14MHz Quarzes und besitzt ein breiteres Bildfenster, das ein 4 zu 3 anstelle des 1 zu 1 Sseitenverhältnisses hat. Er kann auch mit unserer Leiterplatte verwendet

ADDRESS REGISTER					REGISTER FUNCTIONS		Number of bits
Binary				Hexa	Read $R/\overline{W} = 1$	Write $R/\overline{W} = 0$	
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 buffer DO-D7 force a high state on the data bus.

Abb. 5.3.33 Register des EF 9366

des 1 zu 1 Seitenverhältnisses hat. Er kann auch mit unserer Leiterplatte verwendet werden, jedoch sind die Brücken Jx und Jf zu trennen. Wird Jf mit Mass verbunden, wird es mit +5V verbunden (Voreinstellung) und Jm umgesetzt, so ergeben sich 2 Bildseiten mit 512 mal 512 Bildpunkten.

Ich empfehle jedoch den EF 9366 einzusetzen, da er sowohl für Graphik als auch für Text sehr gut brauchbar ist.

Nun zum internen Aufbau des EF9366 (gilt auch für die anderen Prozessoren der Serie). *Abb. 5.3.33* zeigt die 16 Adreßplätze, die der GDP belegt. Bei uns sind sie auf den Adreßbereich 70h bis 7Fh gelegt.

Das Status-Register bei der Leseadresse 70h gibt Aufschluß über wichtige Zustände des GDPs.

Bit 0:
Ist es auf 1, so wurde eine Lichtgriffelsequenz beendet.

Bit 1:
Es entspricht dem VB-Anschluß am GDP. Damit kann also das vertikale Synchronsignal abgefragt werden. Es tritt alle 20ms auf und ist sehr nützlich für bewegte Graphiken, die synchron und flimmerfrei bleiben sollen. Das VB-Signal liegt an, wenn es auf 1 ist.

Bit 2:
Ist das Bit auf 0, so ist der GDP beschäftigt, und es darf kein Kommando an ihm gegeben werden. Man muß also vor jeder Befehlsausgabe oder jedem Umsetzen eines der anderen Register darauf warten, das dieses Bit auf 1 liegt.

Bit 3:
Die Schreibposition befindet sich außerhalb des sichtbaren Berichs, wenn das Bit auf 1 ist.

Bit 4:
Lichtgriffelsequenz zu Ende – IRQ-Signal (falls freigegeben).

Bit 5:
Vertikales Synchron-Signal – IRQ.

Bit 6:
Bereit für ein neues Kommando – IRQ.

Bit 7:
IRQ-Signal, Oder-Verknüpfung aus Bit 4 bis Bit 6.

Dann gibt es noch das Kommandoregister. Wird auf Port 70h geschrieben, so erfolgt eine Befehlsaufführung.

Abb. 5.3.34 zeigt die möglichen Befehle. Wird ein Code im Bereich 20h bis 7Fh ausgegeben, so wird er als ASCII-Code (siehe Abschnitt der Baugruppe KEY) betrachtet und das Zeichen auf dem Bildschirm ausgegeben. *Abb. 5.3.35* zeigt den eingebauten Zeichensatz. Deutsche Buchstaben wie Ä Ö Ü müssen durch Plotten von Punkten erzeugt werden.

b3 b2 b1 b0 \ b7 b6 b5 b4	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0	Set bit 1 of CTRL1 Pen selection	Vector generation (for b2, b1, b0 see small vector definition)						Special direction vectors (for b2, b1, b0 see small vector definition)							
0 0 0 1	1	Clear bit 1 of CTRL 1: Eraser selection	!	1	A	Q	a	q								
0 0 1 0	2	Set bit 0 of CTRL 1 Pen/Eraser down selection	"	2	B	R	b	r								
0 0 1 1	3	Clear bit 0 of CTRL 1: Pen/Eraser up selection	#	3	C	S	c	s								
0 1 0 0	4	Clear screen	$	4	D	T	d	t								
0 1 0 1	5	X and Y registers reset to 0	%	5	E	U	e	u								
0 1 1 0	6	X and Y reset to 0 and clear screen	&	6	F	V	f	v								
0 1 1 1	7	Clear screen, set CSIZE to code „minsize" All other registers reset to 0 (except XLP, YLP)	'	7	G	W	g	w								
1 0 0 0	8	Light-pen initialization (WHITE forced low)	(	8	H	X	h	x								
1 0 0 1	9	Light-pen initialization	)	9	I	Y	i	y								
1 0 1 0	A	5 x 8 block drawing (size according to CSIZE)	*	:	J	Z	j	z								
1 0 1 1	B	4 x 4 block drawing (size according to CSIZE)	+	;	K	[	k	{								
1 1 0 0	C	Screen scanning: Pen or Eraser as defined by CTRL1	,	<	L	\	l	|							
1 1 0 1	D	X register reset to 0	–	=	M	]	m	}								
1 1 1 0	E	Y register reset to 0	.	>	N	↑	n	~								
1 1 1 1	F	Direct image memory access request for the next free cycle.	/	?	O	←	o	▩								

SMALL VECTOR DEFINITION:

b7	b6 b5	b4 b3	b2 b1 b0
1	|Δx|	|Δx|	Direction

Dimension

Δx or Δx	Vector length
0 0	0 step
0 1	1 step
1 0	2 steps
1 1	3 steps

Direction

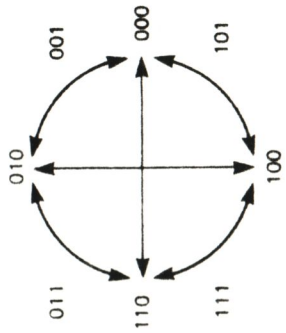


Abb. 5.3.34 Befehle 0 bis FF

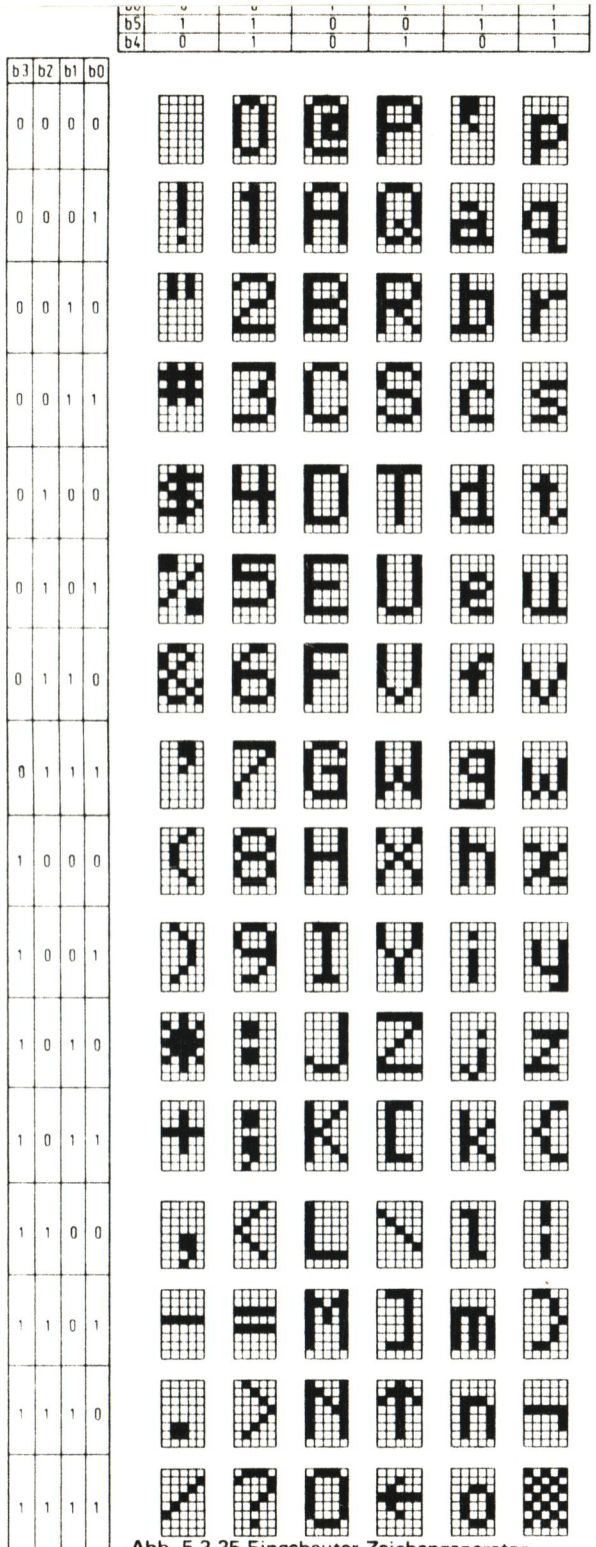
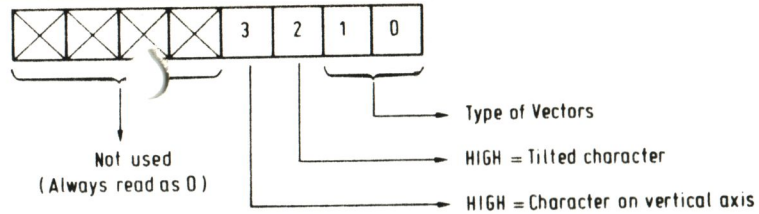
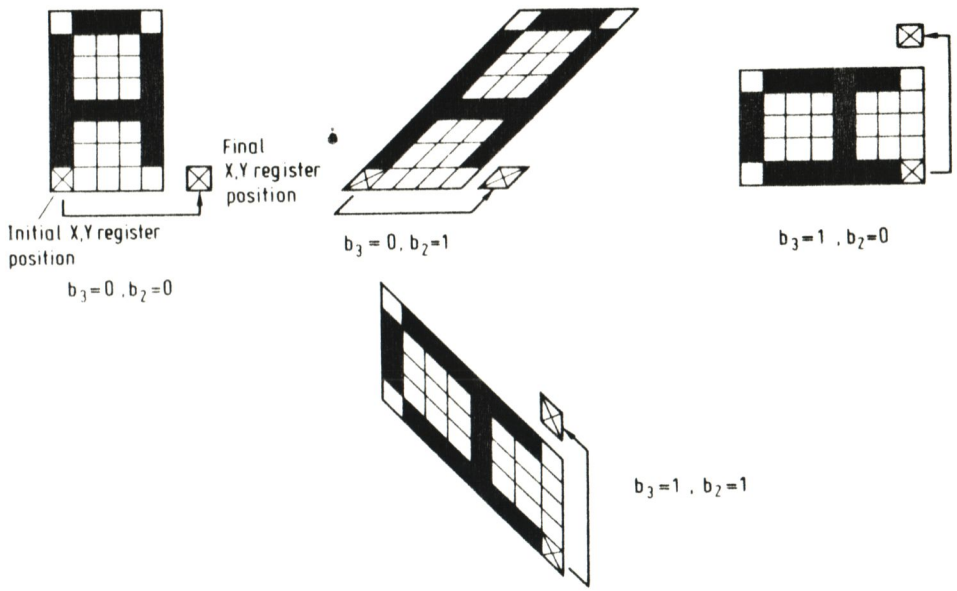


Abb. 5.3.35 Eingebauter Zeichengenerator



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



Mit dem Codebereich 0 bis 0F können Spezialanweisungen, wie Bildschirmlöschen etc. gegeben werden. Zum Schreiben heller Zeichen auf dunklem Grund muß z. B. mit Code 2 der Schreibstift gesenkt werden und mit Code 0 der setzende Schreibstift ausgewählt werden. Die Codes 80h..FFh sind sogenannte Kurzvektorbefehle. Ein Richtungscode und eine Längenangabe sind möglich. Dann wird ein kleiner Vektor gezeichnet.

Der Bereich 10h bis 1Fh arbeitet mit den DELTAX und DELTAY-Registern zusammen. Der Code sieht wie folgt aus:

Grundbefehl:
00010yx1 bedeutet Vektor zeichnen, mit DELTAX und DELTAY als Vektorgröße;
das Vorzeichen wird mit yx codiert.
x=0 bedeutet DELTAX ist positiv, y=0 bedeutet DELTAY ist positiv.

Spezial-Befehl 1:
00010nn0 hier wird eines der beiden Register DELTAX oder DELTAY als 0 betrachtet.

nn = 00 DELTAY ignoriert, DELTAX > 0
nn = 01 DELTAX ignoriert, DELTAY > 0
nn = 10 DELTAX ignoriert, DELTAY < 0
nn = 11 DELTAY ignoriert, DELTAX < 0

Spezial-Befehl 2:
00011rrr zeichnet einen Vektor mit dem größeren der beiden Register DELTAX und DELTAY, in die Richtung, die durch rrr angegeben ist. Für rrr gilt der gleiche Richtungscode, wie bei den Kurzvektoren in Abb. 5.3.34.

Nun gibt es noch weitere Register:
Port 71h: Control-Register 1 – Lesen und Schreiben möglich.

Bit 0:
Schreibstift unten = 1

Bit 1:
Schreibend = ; löschend = 0

Bit 2:
Schreiben auch im Bildfenster; BLK ist immer auf 1.
Es wird kein Videosignal ausgegeben.

Bit 3:
Das Bildfenster ist geschlossen = 1. Das bedeutet, die Schreibposition kann nicht aus dem Bildfenster wandern. Wenn das Bit auf 0 gesetzt wird, ist ein Bildfenster von 4096 mal 4096 vorhanden.

Bit 4:
Freigabe des Interrupts nach einer Lichtgriffelsequenz.

Bit 5:
Freigabe eines Interrupts bei jedem Vertikal-Synchronsignal.

Bit 6:
Freigabe eines Interrupts, wenn der GDP für einen neuen Befehl bereit ist.

Bit 7:
nicht verwendet.

Port 72h: Control-Register 2 – Lesen und Schreiben möglich. Damit läßt sich die Schreibart einstellen

Abb. 5.3.36 zeigt die Port-Belegung. Es kann zum einen die Linienart für Vektoren eingestellt werden, zum anderen die Schreibrichtung für Zeichen.

Port 73h: Vergrößerungsfaktor – Lesen und Schreiben möglich. Die Bits 3 bis 0 bestimmen die Vergrößerung in X-Richtung, die Bits 7 bis 4 die Vergrößerung in Y-Richtung. Dabei ist 1 der Faktor 1 und 0 der Faktor 16 (1,2,3. . .15,0).

Es wird dadurch nur der Einschreibevorgang für ASCII-Zeichen beeinflußt, nicht für Vektoren.

Stückliste zur Baugruppe GDP64

1x	EF0366 Firma Thomson
8x	NEC 4164 -2 64K Bit x 1 Speicher 200ns NEC o.ä.
1x	AMD 25LS2538 Firma Advance-Micro-Devices
1x	74LS138
1x	74LS245
1x	74LS273
1x	74LS74
1x	74LS32
1x	74LS153
1x	74LS05
1x	74LS166
1x	74LS163
1x	74LS00
1x	7404 kein LS-Baustein
1x	Quarz 14MHz
1x	75 Ohm
1x	150 Ohm 1/8W
1x	330 Ohm 1/8W
2x	470 Ohm 1/8W
4x	1kOhm 1/8W
1x	1.5 kOhm 1/8W
1x	Transistor BC 107 o.ä.
6x	100 nF keramische Scheibe
1x	Tantal 10µF
1x	40-pol Sockel
3x	20-pol Sockel
12x	16-pol Sockel
5x	14-pol Sockel

Abb. 5.3.37

Nun gibt es noch die schon erwähnten DELTAX (Port 75h) und DELTAY (port 77h)-Register, in denen die Vektorlänge gespeichert wird. Sie kann demnach maximal 255 sein, und daher muß bei längeren Vektoren eine Aufspaltung vorgenommen werden, wie das in den Unterprogrammen des GRUNDPROGRAMMs der Fall ist. Die Ports 78h und 79h bestimmen die X-Position des nächsten Schreibvorgangs und 7Ah und 7Bh die Y-Position. Dabei ist der punkt x=0, y=0 links unten in der Abbildung. Der Punkt x=511 und y=255 ist rechts oben.

Die Lichtgriffelregister kann man lesen, sie geben die Position des Lichtgriffels an, wenn man einen solchen passend an den GDP anschließen könnte, was wir jedoch nicht tun.

Hier noch der Hinweis auf die Datenblätter der Firma Thomson, in denen weitere Details stehen, die wir jedoch nicht brauchen. In dem GRUNDPROGRAMM des Softwarekapitals sind alle wichtigen Unterprogramme zum Betrieb des GDPs untergebracht.
Abb. 5.3.37 zeigt schließlich die Stückliste zur Baugruppe GDP 64.

5.4 Aufbau eines EPROM-Programmierers

Um nun endlich dem Problem der mühsamen Eingabe von Programmieren mit dem Dil-Schaltern zu entgehen, wird als nächstes eine EPROM-Programmier-Karte aufgebaut. Im Prinzip arbeitet diese Schaltung genauso wie die in Abb. 4.2.5, nur daß hier der Prozessor die Steuerung übernimmt. Mit der Karte können sowohl 2716, 2732 und 2764 der Intel-Familie (und kompatibler) programmiert werden. Das Steuerprogramm befindet sich bereits in unserem Monitor. Da uns nun die CPU zum Test mit einem Datensichtgerät oder der CRT-Karte + Tastatur zur Verfügung steht, wird es wesentlich einfacher sein, neue Karten wie diese aufzubauen. Testprogramme die evtl. benötigt werden, können im RAM abgelegt werden.

Abb. 5.4.1 zeigt die Schaltung des EPROM-Programmierers. Die Ausgabe der Daten erfolgt über das Latch LT1. Da es auch möglich sein muß, vom EPROM Daten zurückzulegen, gibt es den Buffer B1. Ferner muß der Ausgang des Zwischenspeichers LT1 beim Einlesen in den TRI-State-Zustand versetzt werden können. Dies geschieht über den Freigabe-Eingang -OE. Der Freigabe erfolgt per Software über Latch LT3 mit dem auch andere Steuersignale und ein Teil der höherwertigen Adressen ausgegeben werden. Mit dem Treiber B2 kann der Zustand des Monoflops abgefragt werden. Das Monoflop wird zur Erzeugung des Programmierpulses verwendet. Aus Sicherheitsgründen wird dies nicht durch Software ausgeführt, da sonst bei Fehlbedienungen eine Zerstörung des zu programmierenden EPROMs erfolgen könnte. Das Monoflop wird mit dem Trimmer Tr1 auf eine Zeitkonsante von 1ms eingestellt. Für die Programmierung eines Bytes sind aber 50ms nötig.

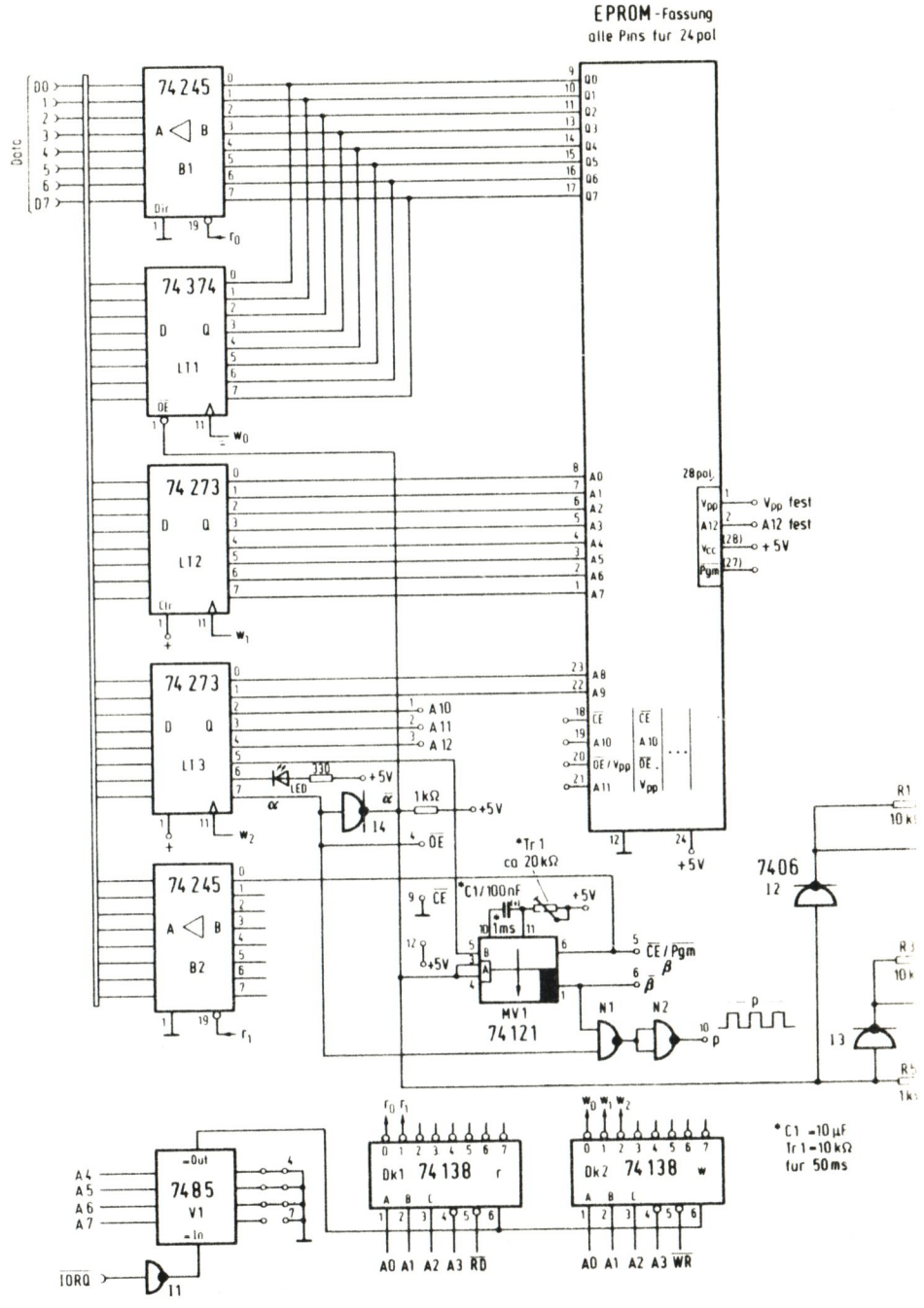


Abb. 5.4.1 Schaltung der EPROM-Programmier-Karte