

Hochauflösende Grafik für CBM 8032
CBM 4016 und CBM 4032 (mit 12" Monitor)

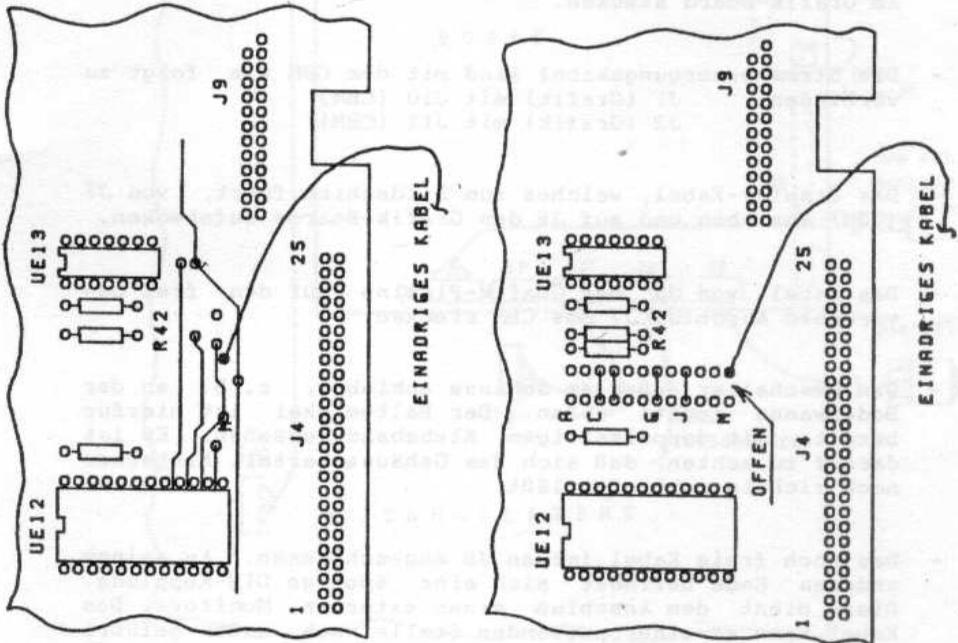
B E D I E N U N G S A N L E I T U N G

Inhaltsverzeichnis:	Seite
1. Einleitung	1
2. Einbauanleitung	2
2.1. Externer Video - Anschluß	5
2.2. Externer Rechner-Betrieb im Bereich \$ 9000 - 9FFF	6
2.3. Unterschied der Grafik-Version A und B	7
3. Neue BASIC-Befehle für die Grafik	8
3.1. BASIC-Befehle für die Bildschirmsteuerung	8
3.2. BASIC-Befehle für Liniendarstellung	13
3.3. BASIC-Befehle zur Character-Darstellung	18
3.4. BASIC-Befehle für Sonderfunktionen	20
3.5. Auflistung aller Grafik-BASIC-Befehle	22
4. Grafik Demo-Programm	23
5. Grafik Programmierung mit Maschinenprogrammen	25
5.1. Speicherbelegung	29
Anhang	
A Grafik ASCII Zeichensatz	30
B Datenblatt GDP EF9365/66 (Auszug)	31
C Light-Pen Anschluß	41
D Steckerbelegung	43

Der Einbau der Grafik-Platine ist sehr leicht und wird im folgenden Absatz beschrieben.

2. Einbauanleitung

Damit die Grafik vom CBM-Rechner angesprochen werden kann, muß der CBM im Adreßbereich von \$A000 bis \$AFFF auf ext. Betrieb geschaltet werden. Hierzu ist das mitgelieferte einadrige Kabel auf der CBM-Platine an den hinteren Lötpoint von Jumper M anzulöten (siehe Bild 1). Sollte der Jumper M gebrückt sein, so muß die Brücke entfernt werden (siehe auch Abschnitt 2.2.). Weitere Lötverbindungen sind für den Grafikeinbau nicht vorzunehmen.



CBM-PLATINE VERSION: I

CBM-PLATINE VERSION: II

Bild 1: Anschluß des einadrigen Kabels an die CBM-Platine

Der Sockel UD 11 im CBM muß frei bleiben, da in diesem Bereich jetzt die Grafik arbeitet.

Die weiteren Verbindungen werden gesteckt. Sie sollten folgendermaßen vorgenommen werden (siehe auch Bild 2):

- Die Grafik-Platine ist so über der CBM-Rechnerplatine zu plazieren, daß die beiden 50poligen Buchsenleisten genau über den beiden Stiftleisten J4 und J9 des CBM liegen. Es muß dabei darauf geachtet werden, daß das blaue einadrige Kabel frei zum Anschluß auf der Grafik-Platine geführt werden kann. Nun das Board ganz auf J4 und J9 aufstecken. Anschließend noch einmal kontrollieren, ob auch kein Pin außerhalb der Buchsenleiste liegt.
- Das blaue Einzelkabel auf die 2polige Stiftleiste (J10) am Grafik-Board stecken.
- Die Stromversorgungskabel sind mit dem CBM wie folgt zu verbinden:
J1 (Grafik) mit J10 (CBM)
J2 (Grafik) mit J11 (CBM)
- Das Display-Kabel, welches zum Bildschirm führt, von J7 (CBM) abziehen und auf J6 des Grafik-Boards aufstecken.
- Das Kabel von J5 der Grafik-Platine auf den frei gewordenen Anschluß J7 des CBM stecken.
- Den Umschalter außen am Gehäuse ankleben, z. B. an der Bodenwanne rechts außen. Der Haltewinkel ist hierfür bereits mit doppelseitigem Klebeband versehen. Es ist darauf zu achten, daß sich das Gehäuseoberteil hinterher noch richtig schließen läßt.
- Das noch freie Kabel ist an J8 angeschlossen. An seinem anderen Ende befindet sich eine 6polige DIN-Kupplung. Diese dient dem Anschluß eines externen Monitors. Das Kabel kann an einer passenden Stelle nach außen geführt werden.

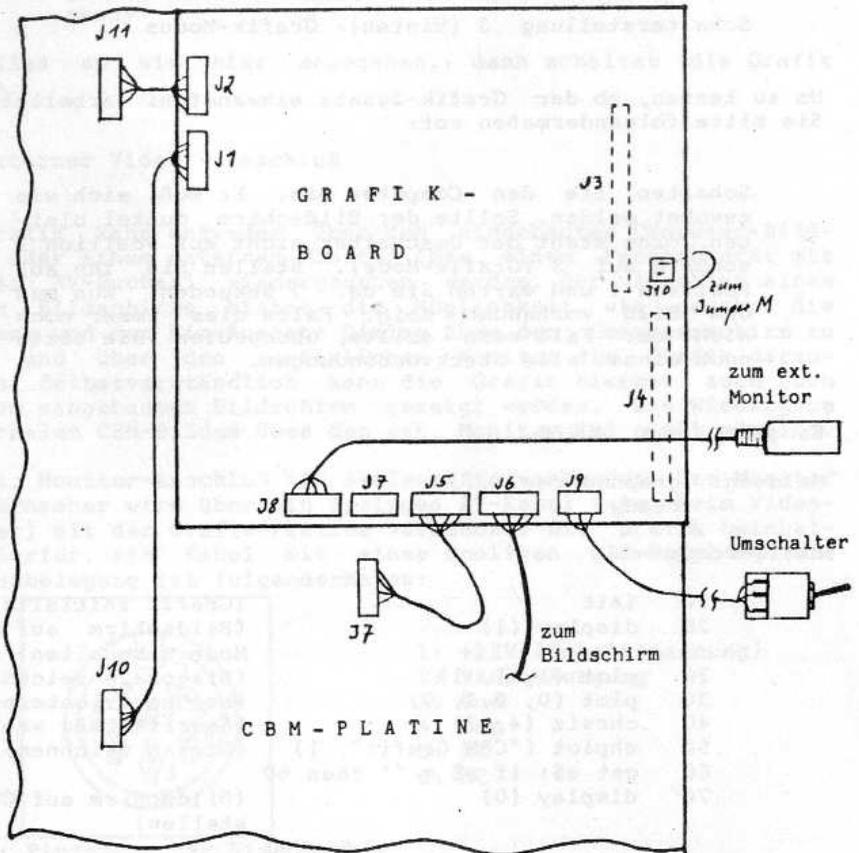


Bild 2 : Einbau der Grafik - Platine in den CBM

Damit ist der Einbau der Platine beendet und die Grafik funktionsfähig.

Der Umschalter hat folgende Funktion:

Schalterstellung 1 (Vorn) : CBM Normal-Modus

Schalterstellung 2 (Mitte) : Modus über Software steuerbar

Schalterstellung 3 (Hinten): Grafik-Modus

Um zu testen, ob der Grafik-Zusatz einwandfrei arbeitet, gehen Sie bitte folgendermaßen vor:

Schalten Sie den Computer ein. Er muß sich wie gewohnt melden. Sollte der Bildschirm dunkel bleiben, dann steht der Umschalter nicht auf Position 1 sondern auf 3 (Grafik-Mode). Stellen Sie ihn auf Position 1 und warten Sie ca. 3 Sekunden. Nun muß das Bild vorhanden sein. Falls dies immer noch nicht der Fall sein sollte, überprüfen Sie bitte noch einmal alle Steckverbindungen.

Eingabe: sys 40960

Meldung: graphic rev.
ready.

BASIC-Programm:

10	init	(Grafik initialisieren)
20	display (1)	(Bildschirm auf Grafik-Mode umschalten)
20	plot (1, 1, 1)	(Diagonale zeichnen)
30	plot (0, 0.3, 0)	(neu positionieren)
40	chrsiz (4, 7)	(Schriftgröße wählen)
50	chplot ("CBM Grafik", 1)	(String zeichnen)
60	get a\$: if a\$ = "" then 60	
70	display (0)	(Bildschirm auf CBM-Mode stellen)

Stellen Sie den Umschalter auf Mittelstellung. Die Umschaltung auf Grafik-Mode kann jetzt mit Software erfolgen. Starten Sie nun das Programm mit "run".

Auf dem Bildschirm muß eine Diagonale von links unten nach rechts oben sowie links von der Diagonale der Schriftzug "CBM Grafik" zu sehen sein. Wenn Sie jetzt eine beliebige Taste drücken, schaltet der Computer wieder in den Normalbetrieb (es dauert ca. 3 sec. bis das Bild erscheint). Auf dem Bildschirm steht zusätzlich:

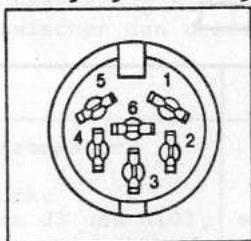
```
graphic rev. ....
ready
```

Ist alles so wie hier angegeben, dann arbeitet die Grafik korrekt.

2.1. Externer Video - Anschluß

Die Grafik kann entweder über den eingebauten Computer-Bildschirm oder einem externen Monitor (bzw. einem Fernsehgerät mit 6poliger AV-Buchse) wiedergegeben werden. Der Anschluß eines zweiten Bildschirms bietet die Möglichkeit, wie gewohnt die Programme und den Ein-Ausgabe Dialog über den Computerschirm zu führen und über den zusätzlichen Monitor die Grafik darzustellen. Selbstverständlich kann die Grafik hierbei auch noch über den eingebauten Bildschirm gezeigt werden. Die Wiedergabe des normalen CBM-Bildes über den ext. Monitor ist nicht möglich.

Der ext. Monitor-Anschluß ist serienmäßig vorhanden. Der Monitor oder Fernseher wird über ein 6poliges AV-Kabel (wie beim Video-Recorder) mit der Grafik-Platine verbunden. Die Grafik beinhaltet hierfür ein Kabel mit einer 6poligen DIN-Kupplung. Die Anschlußbelegung ist folgendermaßen:



- 1: +12V (Schaltspannung)
- 2: Video-Ausgang
- 3: Gnd
- 4: Gnd
- 5: NC
- 6: NC

Bild 3: Pinlegung der Videobuchse

Das Videoausgangssignal (BAS, CCIR-Norm) liefert 1 V_{ss} Pegel an 75 Ohm.

Die Schaltspannung +12V (PIN 1) dient zur Umschaltung auf Videobetrieb bei Anschluß eines Fernsehgerätes. Diese Spannung darf nicht zur Speisung weiterer Schaltungen benutzt werden.

Anmerkung: Bei der Grafik-Platine mit 512 x 512 Punkten Auslösung (Version A) sollte ein Monitor mit einer lang nachleuchtenden Bildröhre verwendet werden, da das Bild sonst flimmert.

2.2. Externer Rechner-Betrieb im Bereich \$ 9000 - 9FFF

Soll außer dem Grafik-Board noch eine andere Erweiterung vorgenommen werden, so ist dieses möglich, sofern der Adreßbereich \$ A000 - AFFF nicht benutzt wird. Der Expansion-BUS J4 und J9 des Commodore Computers ist auf der Grafik-Platine wieder vorhanden. Außerdem befindet sich auf dem Grafik-Board neben den 50poligen Steckerleisten eine freie Löttaugenleiste. Diese sind ebenfalls mit dem Expansion-BUS verbunden. Hier können Drähte oder Stiftleisten eingelötet werden.

Wird der Adreßbereich \$ 9000 - 9FFF extern benötigt, muß hierzu der Computer darauf umgestellt werden. Ohne Grafik-Platine ist eine Drahtbrücke bei Jumper "M" im CBM einzulöten. Mit Grafik-Platine darf an dieser Stelle keine Brücke vorhanden sein. Die Umstellung muß auf dem Grafik-Board vorgenommen werden. Hierzu ist eine Drahtbrücke zwischen dem IC U19 und der 50poligen Steckerleiste einzulöten. Sie ist ebenfalls mit "M" bezeichnet. Der ROM-Steckplatz UD 12 auf der CBM Platine muß dann frei bleiben.

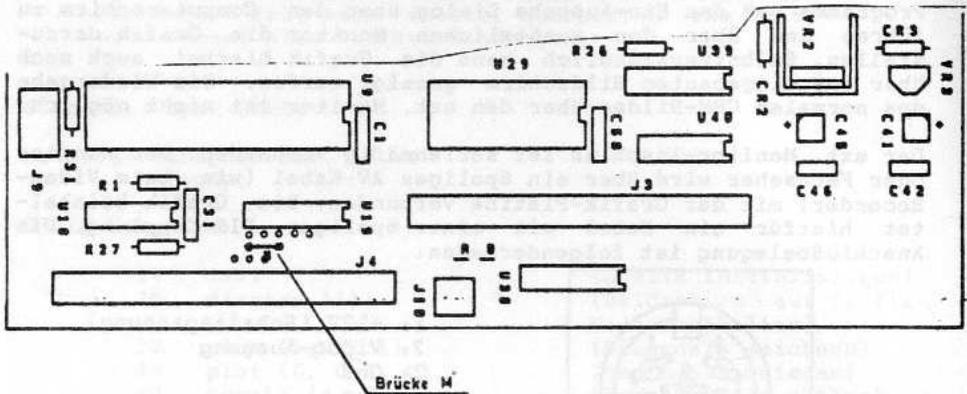


Bild 4: Drahtbrücke M auf dem Grafik-Board

2.3. Unterschied der Grafik-Version A und B

Die Grafik-Platine wird in zwei verschiedenen Ausführungen angeboten. Der Unterschied liegt in der Auflösung.

Version A:

- 512 Zeilen mit 512 Punkte/Zeile (Gesamt 262 144 Punkte)
- Speicher für 1 Bildschirmseite
- Darstellung der 512 Zeilen im Zeilensprungverfahren (Bildwiederholrate 25 Hz)

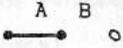
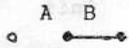
Version B:

- 256 Zeilen mit 512 Punkte/Zeile (Gesamt 131 072 Punkte)
- Speicher für 2 Bildschirmseiten
- Darstellung der 256 Zeilen ohne Zeilensprung (Bildwiederholrate 50 Hz)

Bei der Version A ist das Grafik-Bild bei der Wiedergabe über einen Bildschirm mit Standardphosphor nicht flimmerfrei, da die Bildwiederholrate nur 25 Hz beträgt. Es sollte eine Bildröhre mit lang-nachleuchtendem-Phosphor verwendet werden.

Bei der Version B ist das Bild durch die Bildwiederholrate von 50 Hz immer flimmerfrei.

Bei der Bestückung der Grafik-Platinen besteht folgender Unterschied zwischen den beiden Ausführungen:

	Version A	Version B
Grafik-Prozessor	EF9365	EF9366
Drahtbrücke (zwischen J3 und J10)		

Das EPROM ist für beide Versionen gleich.

3. Neue BASIC-Befehle für die Grafik

In diesem Kapitel sind die neuen BASIC-Befehle beschrieben. Sie sind in dem EPROM auf dem Grafik-Board implementiert. Angeführte Beispiele und Demo-Programme beziehen sich immer auf die Version A mit 512 x 512 Punkten Auflösung. Für die Version B (512 x 256 Punkte) müssen die y-Koordinaten entsprechend geändert werden. Zur Initialisierung der neuen BASIC-Befehle geben Sie bitte folgenden Befehl über den CBM ein:

```
sys 40960
```

der CBM antwortet dann mit:

```
graphic rev. ....  
ready
```

Von jetzt an versteht der CBM die neuen Grafik Befehle im Direkt-Mode sowie im Programm. Die BASIC-Befehle können wie beim normalen Commodore-BASIC an der 2.- bzw. 3.-Stelle durch die Eingabe mit Shift gekürzt werden. Die abgekürzte Form ist jeweils mit angeben. Sie können bis auf eine Ausnahme genauso angewandt werden wie die bisherigen Befehle. Die Ausnahme bildet der CBM-Befehl "IF THEN". Direkt hinter ihm darf kein Grafik Befehl stehen. Er ist z. B. mit einem Doppelpunkt zu trennen:

```
40 if a = 5 then : plot (20, 0, 1)
```

Es ist unbedingt darauf zu achten, daß vor Eingabe eines Programms der BASIC-Befehlssatz mit sys 40960 eingeschaltet wurde. Falls nicht, wird dies bei der Eingabe und beim Auflisten nicht bemerkt. Erst nach dem "run" meldet der Rechner "syntax error". Ein nachträgliches initialisieren der Grafik-Befehle ändert hieran nichts.

3.1. BASIC-Befehle für die Bildschirmsteuerung

3.1.1. init 'inI'

Dieser Befehl löscht den Grafik-Bildschirm, initialisiert den Grafik-Prozessor und setzt einige Parameter wie folgt:

map	(0, 1, 0, 1)
pspace	(0, 1, 0, 1)
chrsiz	(1, 1)
chrori	(0)
lintyp	(0)
display	(0)
mode	(0)

MODE	(I)	Dargestellte Seite	Bearbeitete Seite
	0	1	1
	1	1	2
	2	2	1
	3	2	2

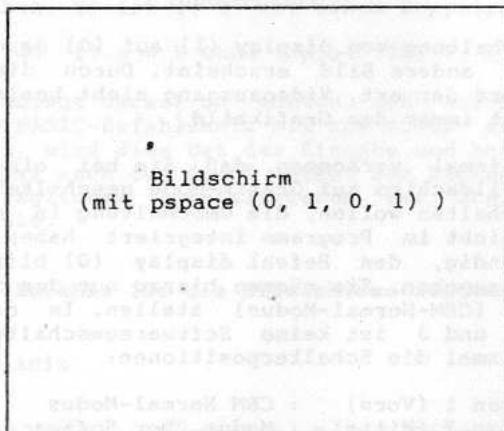
So kann z. B. mit mode (1) die Seite 1 auf dem Bildschirm betrachtet werden, während gleichzeitig eine neue Zeichnung auf der Seite 2 erstellt wird.

3.1.5. map (XO, XM, YO, YM) 'mÄ'

Mit map werden die Grenzen des mathematischen Koordinatensystems definiert, in dem die Zeichnung dargestellt werden soll. Die Parameter XO und YO geben dabei den kleinsten X- und Y-Wert an. Dieser Punkt befindet sich in der linken unteren Bildecke. Die Maximalwerte (rechte, obere Bildecke) werden über XM und YM angegeben.

XO, YO

XM, YM



XO, YO

XM, YO

Parameter: XO - kleinste dargestellte X-Koordinate
 XM - größte dargestellte X-Koordinate
 YO - kleinste dargestellte Y-Koordinate
 YM - größte dargestellte Y-Koordinate

Jeder Aufruf von plot oder dplot übernimmt die X, Y-Parameter des mit map definierten Koordinatensystems. Danach werden die in plot oder dplot angegebenen X, Y-Werte in die physikalischen Koordinaten des Bildschirms gewandelt und die entsprechenden Punkte modifiziert.

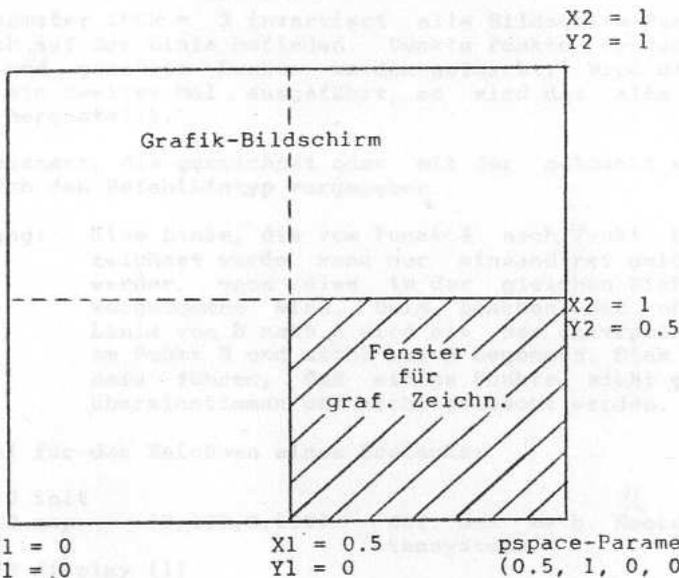
Beispiel: `map (-100, 1000, 0, 1000)`

Die X-Koordinate auf dem Bildschirm gehen von -100 bis +1000, die Y-Koordinate von 0 bis +1000.

3.1.6. `pspace (X1, X2, Y1, Y2)` 'ps'

- Über `pspace` kann für die grafische Zeichnung auf dem Bildschirm ein Fenster definiert werden. Im Normalfall wird der ganze Bildschirm als Zeichenebene genutzt. Hierbei sind die X, Y-Parameter von `pspace` wie folgt definiert: $X1 = 0$, $X2 = 1$, $Y1 = 0$, $Y2 = 1$. Diese Werte dürfen im Bereich von 0 bis 1 liegen. $X1$, $Y1$ geben die Lage der unteren linken Fensterecke auf dem Bildschirm an; $X2$, $Y2$ die obere rechte Fensterecke.

Eine Zeichnung, die vorher den ganzen Bildschirm belegt hat, läßt sich z. B. im rechten unteren Bildschirmviertel darstellen. Hierzu muß nur der Befehl `pspace (0.5, 1, 0, 0.5)` vor dem Erstellen der Zeichnung eingegeben werden.



Die gesamte Zeichnung, die von `map` definiert wurde, wird auf dem schraffierten Bereich von `pspace` abgebildet.

Die X, Y-Parameter sind ähnlich denen des `map`-Befehls, jedoch wird nicht das mathematische, sondern das physikalische Fenster definiert. Die X, Y-Koordinaten von `plot` und `dplot` werden entsprechend den Parametern von `map` und `pspace` in den physikalischen Bildschirmwertebereich umgerechnet.

Beispiel: 4 Zeichnungen, die unabhängig voneinander sind, sollen gleichzeitig über den Bildschirm wiedergegeben werden. Hierzu wird der Befehl pspace angewandt:

init

pspace (0.5, 1, 0.5, 1) I. Quadrant
map ()
Zeichnung 1

pspace (0.5, 1, 0, 0.5) II. Quadrant
map () erneutes MAP, wenn anderes
math. Koordinatensystem

pspace (0, 0.5, 0, 0.5) III. Quadrant
Zeichnung 3

pspace (0, 0.5, 0.5, 1) IV. Quadrant
Zeichnung 4



3.2. BASIC-Befehle für Liniendarstellung

3.2.1. plot (X, Y, IPEN) 'pl'

Mit dem Befehl **plot** wird der "grafische Pen" von seiner Ausgangsposition geradlinig zur neuen Position X, Y bewegt. Die X, Y-Angaben beziehen sich hierbei auf das in **map** gewählte mathematische Koordinatensystem. Die Art der Ausführung (zeichnen, positionieren usw.) hängt vom Parameter **IPEN** ab.

Parameter:	X, Y	- neue Position (mathem. Koordinaten)
	IPEN	- Zeichenstiftkontrolle
	0	- bewegen ohne zu zeichnen (Pen up)
	1	- zeichnet eine Linie (Pen down)
	2	- löscht eine Linie (Eraser)
	3	- invertiert eine Linie

gezeichnet wird jeweils beginnend bei der alten Position bis zur neuen X, Y-Position.

Der Parameter **IPEN = 3** invertiert alle Bildschirm-Punkte, die sich auf der Linie befinden. Dunkle Punkte werden gesetzt und gesetzte Punkte werden gelöscht. Wird dieser Befehl ein zweites Mal ausgeführt, so wird das alte Bild wieder hergestellt.

Die Linienart, die gezeichnet oder mit der gelöscht wird, ist durch den Befehl **Lintyp** vorgegeben.

Anmerkung: Eine Linie, die vom Punkt A nach Punkt B gezeichnet wurde kann nur einwandfrei gelöscht werden, wenn dies in der gleichen Richtung vorgenommen wird. Beim Löschen der obigen Linie von B nach A wird mit der Interpolation am Punkt B und nicht bei A begonnen. Dies kann dazu führen, daß einige Punkte nicht genau übereinstimmen und nicht gelöscht werden.

Beispiel für das Zeichnen eines Dreiecks:

```

10 init
20 map      (0,100,0,100)  def. des math. Koordina-
                           tensystems
30 display (1)
40 plot    (20,30,0)      Pen auf Anfangsposition
                           setzen
50 plot    (80,30,1)      1 Linie zeichnen
60 plot    (50,70,1)      1 Linie zeichnen
70 plot    (20,30,1)      1 Linie zeichnen

```

3.2.2. dplot (DX, DY, IPEN) 'dp'

Die Funktion ist wie bei dem **plot**-Befehl, nur daß die Parameter **DX, DY** relativ auf die letzte aktuelle Zeichenposition bezogen sind. Hiermit lassen sich z. B. wiederkehrende

Figuren definieren und mit plot an eine bestimmte Bildschirmstelle positionieren.

Der IPEN-Parameter ist unter plot beschrieben.

Beispiel für das Positionieren eines Symbols:

```

10 init
20 map      (0,100,0,100)
30 display (1)
40 input "X, Y-Koordinate"; X, Y
50 plot    (X, Y, 0)
60 Gosub 500
70 goto 40

500 dplot (4,0,1) : dplot (-2,-2,1) : dplot (-2,2,1)
510 return

```

In Zeile 40 wird die X, Y-Position des darzustellenden Symbols abgefragt und in Zeile 50 darauf positioniert. Zeile 500 schreibt das Symbol auf den Bildschirm.

Ein einzelner Punkt auf dem Bildschirm kann mit dplot gesetzt, gelöscht oder invertiert werden. Hierzu wird mit plot (X,Y,0) auf den Punkt positioniert und danach mit dplot (0,0,IPEN) der Punkt entsprechend IPEN verändert.

3.2.3. iplot (X, Y, IPEN) 'ip'

Der Befehl iplot gleicht in seiner Wirkungsweise dem plot. Der Unterschied liegt in der Definition des Koordinatensystems. Während plot im math. System arbeitet, sind die X, Y-Parameter des Befehls iplot im physikalischen Koordinatensystem des Grafik-Bildschirms definiert. Dies bedeutet mit iplot werden direkt die entsprechenden Punkte auf dem Bildschirm ausgewählt. Die X, Y-Parameter liegen deshalb im Bereich von:

$0 \leq X, Y \leq 511$ (bei Version B: $0 \leq Y \leq 255$)

Durch die Umgehung der mit map und pspace gewählten Abbildungsvorschrift ergibt sich eine höhere Zeichengeschwindigkeit. Ansonsten gilt das in plot beschriebene (z. B. für IPEN).

3.2.4. idplot (DX, DY, IPEN) 'id'

Der Befehl idplot ist ähnlich dem iplot-Befehl. Die Parameter DX, DY des idplot sind relative Koordinaten, die sich auf die letzte aktuelle Zeichenposition beziehen. Die weitere Funktionsweise ist wie bei iplot.

Zum Bearbeiten eines einzelnen Punktes ist idplot (0,0,IPEN) zu verwenden. Vorher muß mit iplot (X,Y,0) auf diesen positioniert werden (siehe auch unter dplot).

3.2.5. **lintyp (ITYP)** 'liN'

Mit lintyp (Linientyp) wird die Art der gewünschten Linie in den Plot-Befehlen definiert.

Parameter:	ITYP	Linienart
	0	- durchgezogene Linie
	1	- punktierte Linie
	2	- gestrichelte Linie
	3	- Strich-Punkt-Linie

Die gewählte Linienart ist beim Zeichnen wie auch beim Löschen wirksam. Zum Löschen einer Linie empfiehlt es sich ITYP=0 zu verwenden, da hierbei nicht auf die Art der zu löschenden Linie geachtet werden muß.

3.2.6. **icrcl (R, CPEN)** 'iC'

Mit icrcl (Circle) lassen sich Kreise mit dem Radius R auf dem Grafik-Bildschirm darstellen. Der Kreismittelpunkt wird durch die letzte aktuelle Zeichenposition festgelegt. Der Radius R bezieht sich immer auf das physikalische Koordinatensystem.

Nach der Abbildung des Kreises befindet sich die Zeichenposition wieder auf dem Mittelpunkt. Die Zeichengeschwindigkeit ist bei icrcl und arcus geringer als bei plot, da die Interpolation mit Software durchgeführt wird. Lintyp muß für icrcl auf "0" gesetzt werden.

Parameter:	R	Radius
	IPEN	Zeichenstiftkontrolle
	1	zeichnen
	2	löschen

3.2.7. **arcus (RX,RY,WA,WE,CPEN)** 'aR'

Der BASIC-Befehl arcus gestattet das Darstellen von Kreisen, Ellipsen und Sektoren. Mit RX wird der Radius in X-Richtung angegeben. Die Werte beziehen sich auf das mathematische Koordinatensystem. Mit WA und WE muß der Anfangs- und Endwinkel im Bogenmaß definiert werden. Für IPEN und Lintyp gilt das in icrcl beschriebene. Soll z. B. eine liegende Ellipse gezeichnet werden, so kann das mit folgenden Parametern vorgenommen werden:

RX=100, Ry=50, WA=0, WE=6,28, CPEN=1

Der Mittelpunkt wird wie bei icrcl durch die letzte Zeichenposition bestimmt. Nach der Darstellung wird die Zeichenposition wieder auf den Mittelpunkt gesetzt (auch bei Sektoren).

Parameter: RX Radius in X-Richtung
 RY Radius in Y-Richtung
 WA Anfangswinkel im Bogenmaß
 WE Endwinkel im Bogenmaß
 IPEN siehe unter icrc1

Mit dem folgenden Beispielprogramm kann der Befehl **arcus** untersucht werden:

```
10  gr=6,284/360
20  init
30  map (0,500,0,500)
40  input "Mittelpunkt X,Y";mx,my
50  input "X-Radius,Y-Radius";rx,ry
60  input "Anfangswinkel,Endwinkel (in Grad)";wa,we
70  wa=wa * gr : we = we * gr
100 plot (mx,my,0)
110 arcus (rx,ry,wa,we,1)
120 goto 40
```

3.3. BASIC-Befehle zur Character-Darstellung

Die Grafik-Platine besitzt einen eigenen Character-Generator. Eine Beschriftung der Zeichnung kann in horizontaler und vertikaler Richtung vorgenommen werden. Außerdem können die Zeichen in der Höhe und Breite variiert werden. Die folgenden Abschnitte beschreiben die BASIC-Befehle hierfür.

3.3.1. `chplot (A$,IPEN)` 'chP'

Mit `chplot` (Character-Plot) wird der in `A$` enthaltene Character-String auf den Bildschirm gedruckt.

Der String beginnt an der letzten aktuellen Zeichenposition. Es können alle Zeichen des ASCII-Zeichensatzes verwendet werden (siehe hierzu Anhang A).

Soll der Text am Anfang der nächsten Zeile beginnen, so ist dies auch bei der Grafik mit `CHR$(13)` (Carriage Return) möglich. Die gewählte Schriftgröße wird automatisch berücksichtigt. Allerdings muß beachtet werden, daß ein `CR` nur bei horizontaler Schreibweise möglich ist. Der zu plottende String sollte einer Stringvariablen zugewiesen werden und nicht direkt im `chplot` angegeben werden, z. B.:

```
chplot ("Commodore",1)
```

```
besser:  *C$ = "Commodore"
         chplot (C$,1)
```

Jedes Zeichen wird aus einer 6x8 Punktmatrix zusammengesetzt. Mit dem `chplot`-Befehl lassen sich außer den ASCII-Zeichen auch Blöcke mit 4x4 und 5x8 Punktgröße wiedergeben. Ausgewählt wird dies mit:

```
CHR$(10) - 5x8 Block
CHR$(11) - 4x4 Block
```

Der Block-Plot-Befehl bietet außer dem Darstellen von quadratischen und rechteckigen Blöcken ein einfaches Löschen von Strings auf dem Bildschirm. Ein String wird nicht durch Überschreiben eines zweiten automatisch gelöscht, sondern muß vorher gelöscht werden. Dies kann mit einem dem zu löschenden String identischen, oder einem 5x8 Block geschehen. IPEN ist hierbei auf 2 zu setzen. Mit dem 5x8 Block kann jeder beliebige Buchstabe gelöscht werden. Es müssen genau soviel Blöcke aneinander gereiht werden, wie der String Zeichen enthält.

Der Parameter IPEN ist bei plot (Kapitel 3.2.) beschrieben und ist hier genauso anzuwenden.

Beispiel für das Plotten von Strings:

```

10  init
20  iplot (20, 350, 0)           positionieren
30  xs = 9 : ys = 8             Buchstabengröße
40  a$ = "Commodore" : gosub 500
50  iplot (150, 200, 0)
60  xs = 6 : ys = 6
70  a$ = "Grafik" : gosub 500
100 end
500 chrsiz (xs, ys)             Zeichengr. setzen
510 chplot (a$,1) : return      String a$ plotten

```

3.3.2. chrori (IDIR) 'chrO'

Die Richtung und Art in der die Buchstaben geschrieben werden sollen, wird mit chrori (Character-Orientatation) definiert. Schrift ist in horizontaler und vertikaler Richtung sowie aufrecht und kursiv möglich.

Parameter:	IDIR	Richtung	Art
	0	- horizontal	, aufrecht
	1	- horizontal	, kursiv
	2	- vertikal	, aufrecht
	3	- vertikal	, kursiv

3.3.3. chrsiz (XS, YS) 'chrS'

Mit chrsiz (Character-Size) kann die Größe der Buchstaben in X- und Y-Richtung unabhängig voneinander gewählt werden. XS und YS können im Bereich von 1 bis 15 gewählt werden. Hieraus ergeben sich 225 verschiedene Buchstabengrößen. Jeder Buchstabe wird aus einer 6x8 Punkt-Matrix generiert. XS und YS sind Vergrößerungsfaktoren. Die dargestellte Größe ist (6xXS) x (8xYS). Die in chplot beschriebenen 4x4 und 5x8 Blöcke werden im gleichen Maßstab vergrößert.

3.4. Basic-Befehle für Sonderfunktionen

3.4.1. hcopy

'hc'

Der hcopy-Befehl (Hardcopy) erstellt eine Kopie der Bildschirmzeichnung über einen Commodore 8023P Matrix-Drucker. Der Bildschirm wird hierzu Punkt für Punkt abgetastet und die Information über den Drucker zu Papier gebracht.

Das hcopy-Command benutzt intern die Filenummern 124 ... 127. Daher darf auf diesen Commands kein File geöffnet sein.

Soll ein anderer grafikfähiger Matrix-Drucker verwendet werden, so muß die Anpassung der Software hierzu selbst erstellt werden. Dies kann allerdings in BASIC vorgenommen werden und wird durch den BASIC-Befehl tstb von der Grafik-Software unterstützt.

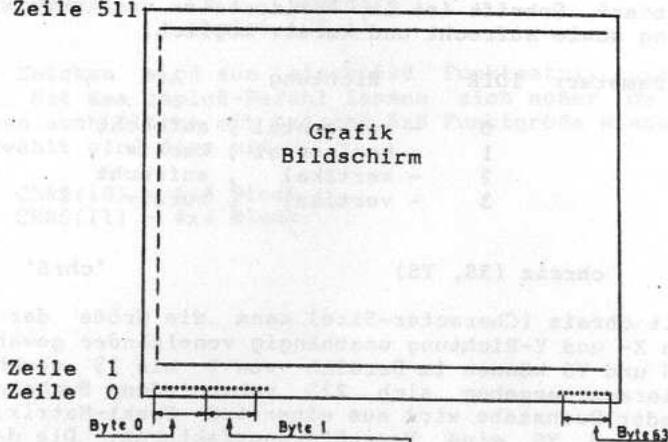
3.4.2. tstb (A)

'ts'

Mit tstb (Test-Byte) kann der Grafik-Bildschirm Byte für Byte abgetastet werden.

Der Bildschirm ist aufgeteilt in 512 Zeilen (256 Zeilen bei Version B) und 512 Punkte je Zeile. Die 512 Punkte sind organisiert zu 64 Byte (1Byte = 8Bit, 1Bit = 1 Punkt). Der linke Punkt in dem Byte hat die kleinste Wertigkeit = Bit0, der achte Bildpunkt in jedem Byte die Höchste = Bit7 (siehe Bild 5).

Zeile 511



Lage der Bits in einem Byte:



Bit: 0 1 2 3 4 5 6 7

Bild 5: Organisation der Bildpunkte

Wertigkeit der Bits:	$0 = 2^0 = 1$	(Punkt 1)
	$1 = 2^1 = 2$	(Punkt 2)
	$2 = 2^2 = 4$	(Punkt 3)
	$3 = 2^3 = 8$	(Punkt 4)
	$4 = 2^4 = 16$	(Punkt 5)
	$5 = 2^5 = 32$	(Punkt 6)
	$6 = 2^6 = 64$	(Punkt 7)
	$7 = 2^7 = 128$	(Punkt 8)

Wird **tstb** aufgerufen, so wird der angegebenen Variablen A der entsprechende Wert des Bytes zugewiesen, indem sich die aktuelle Zeichenposition befindet. Es ist hierbei egal, auf welchem Bit des Bytes positioniert wird, da immer das ganze Byte gelesen wird. Hat die Variable hinterher z. B. den Wert 73, so sind die Punkte 1,4 und 7 gesetzt ($73 = 1+8+64$).

Dieser Befehl kann z. B. zur Ausgabe des Bildschirminhalts über einen anderen Drucker benutzt werden. Hier ein Beispiel-Programm dazu:

```

100 for x = 0 to 511 step 8
110 for y = 0 to 511
120 iplot (x,y,0)           positionieren
130 tstb (a)                lese Byte
140 print #4,a             drucke Byte
150 next y
160 print #4, chr$(13)     nächste Drucker-
                           zeile
170 next x

```

Dieses Programm geht davon aus, daß die oberste Nadel des Matrix-Druckers dem Bit 0 entspricht. Das Grafik-Bild wird um 90° gedreht gedruckt. Soll dies nicht der Fall sein, so ist das Programm entsprechend zu ändern.

3.4.3. tstp (P)

Der Befehl **tstp** (Test-Punkt) arbeitet wie **tstb**, nur daß nicht das kpl. Byte sondern nur der eine Punkt, auf dem sich die Zeichenposition befindet, gelesen wird. Mit **iplot (x,y,0)** kann auf den Bildschirmpunkt positioniert werden. Mit **tstp (P)** wird abgefragt, ob der Punkt gesetzt ist oder nicht. Ist der Punkt gesetzt, also hell, so erhält die Variable P den Wert -1, andernfalls ist $P=0$.

3.4.4. cursor

'CU'

Durch den **cursor**-Befehl wird an der aktuellen Zeichenposition ein Cursorkreuz dargestellt.

Wird das **cursor**-Command ein zweites Mal aufgerufen, ohne daß die Zeichenposition geändert wurde, wird das Kreuz wieder gelöscht. Die eigentliche Zeichnung wird dabei nicht verändert. Das Kreuz wird immer im Invert-mode (IPEN=3) gezeichnet, d. h. dunkle Punkte werden hell gesetzt und gesetzte Punkte werden gelöscht. Der **cursor**-Befehl wird auch in dem Grafik-Demo-Programm, daß in Kapitel 3.6. beschrieben ist, benutzt.

3.5. Auflistung aller Grafik-BASIC-Befehle

init		'inI'
cscr		'cS'
display	(I)	'dis'
mode	(I)	'mO'
map	(Xo, XM, YO, YM)	'mA'
pspace	(X1, X2, Y1, Y2)	'pS'
plot	(X, Y, IPEN)	'pL'
dplot	(DX, DY, IPEN)	'dP'
lplot	(X, Y, IPEN)	'lP'
idplot	(DX, DY, IPEN)	'iD'
lintyp	(ITYP)	'lin'
icrcl	(R, IPEN)	'iC'
arcus	(RX, RY, WA, WE, IPEN)	'aR'
chplot	(A\$, IPEN)	'chP'
chrcl	(IDIR)	'chrO'
chrsz	(Xs, Ys)	'chrS'
hcopy		'hC'
tstb	(A)	'tS'
tstp	(P)	
cursor		'cU'

4. Grafik Demo Programm

```

10 rem *****
20 rem ***          grafik demo programm          ***
30 rem ***
40 rem ***    zeichnen unter benutzung der 10er zahlen-    ***
50 rem ***    tastatur zur cursor - steuerung          ***
60 rem ***
70 rem *****
80 rem
100 rem ***    grafik-initialisierung und befehlstabelle
110 rem
120 print"":sys40960:print" Schalter auf Mittelstellung"
200 xs=3:ys=3
210 iplot(0,485,0)
220 c$="BEFEHLSLISTE":gosub1000
230 rem
240 xs=2:ys=2
250 c$="z - Zeichnen":gosub1000
260 c$="l - Loeschen":gosub1000
270 c$="p - Positionieren":gosub1000
280 c$="e - Einzelschritt":gosub1000
290 c$="d - Dauer ' ' ":gosub1000
300 c$="a - Pos. Zaehler an":gosub1000
310 c$="u - Pos. ' ' aus":gosub1000
320 c$="c - clear Bild":gosub1000
330 c$="x - zurueck ins basic":gosub1000
335 rem
340 iplot(0,0,0)
350 c$="x-Pos.:          y-Pos.:" :gosub1000
360 rem
400 rem ***    variable definieren und cursor auf anfangs-
410 rem          position setzen
420 for i=1 to 4:cl$=cl$+chr$(10):next
430 display(1):rem          bildschirm auf grafik-mode schalten
440 xp=100:yp=100:iplot(xp,yp,0):cursor
450 rem
500 rem ***    tastaturabfrage
510 rem
520 geta$:ifa$<>" " then a=val(a$):goto 540
530 if c=0 then goto 520
540 if a<l then goto 900
550 rem
600 rem ***    cursor neu positionieren
610 rem
620 on a gosub 810,820,830,840,850,860,870,880,890
630 cursor:idplot(x,y,p):cursor:rem cursor neu positionieren
640 xp=xp+x:yp=yp+y:rem          absolute cursor-position
650 if z=0 then 500
660 rem
    
```

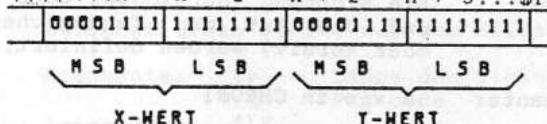
```
700 rem ***   neue cursor-position anzeigen,wenn z=1
710 rem
720 x$=str$(xp):y$=str$(yp)
730 iplot(80,0,0):chplot(c1$,2):iplot(80,0,0)
740 chplot(x$,1)
750 iplot(260,0,0):chplot(c1$,2):iplot(260,0,0)
760 chplot(y$,1)
770 iplot(xp,yp,0)
780 goto 500
790 rem
800 rem ***   werte fuer x,y-cursor-richtung zuweisen
805 rem
810 x=-2:y=-2:return
820 x= 0:y=-2:return
830 x= 2:y=-2:return
840 x=-2:y= 0:return
850 x= 0:y= 0:return
860 x= 2:y= 0:return
870 x=-2:y= 2:return
880 x= 0:y= 2:return
890 x= 2:y= 2:return
895 rem
900 rem ***   befehlsstatus setzen
905 rem
910 ifa$="d"then c=1
920 ifa$="e"then c=0
930 ifa$="p"then p=0
940 ifa$="z"then p=1
950 ifa$="l"then p=2
960 ifa$="a"then z=1
970 ifa$="u"then z=0
980 ifa$="x"then :display(0):end:rem   bildschirm auf cbm-mode
985 rem                               schalten und programmende
990 goto500
995 rem
1000 rem ***   char.-plot unterprogramm
1005 rem
1010 chrsiz(xs,ys)
1020 c$=c$+chr$(13)
1030 chplot(c$,1)
1040 return
```

5. Grafik-Programmierung mit Maschinenprogrammen

Alle BASIC-Kommandos, die im physikalischen Koordinatensystem arbeiten, sind auch als Unterprogramme auf Assembler-Ebene verfügbar. Dabei liegen alle absoluten Koordinaten in einem Wertebereich zwischen $(0,0)$ und $(4095,4095)$. Der Nullpunkt liegt in der unteren linken Ecke des Bildschirms. Aus diesem Wertebereich wird in x-Richtung der Bereich 0 bis 511 auf dem Bildschirm dargestellt. In y-Richtung sind in Abhängigkeit vom verwendeten Graphikprozessor entweder der Bereich 0 bis 511 (Version A) oder der Bereich 0 bis 255 (Version B) sichtbar. Die unsichtbaren Bereiche des gesamten Koordinatenfensters erlauben eine einfache Maskierung von Linien oder Textzeichen, die das sichtbare Fenster überschreiten (Hardware-Clipping).

Jeder Koordinatenwert wird als vorzeichenlose ganze Zahl mit 12 Nutzbits dargestellt. Die nicht benutzten Bits stehen immer auf 0. Die Anordnung von MSB und LSB dieser Zahl entspricht dem in BASIC verwendeten INTEGER-Zahlenformat.

Adresse: $0000 \dots n \quad n+1 \quad n+2 \quad n+3 \dots \$FFFF$

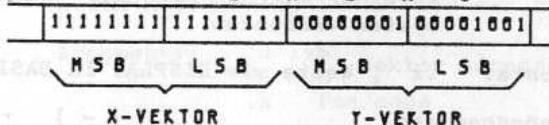


Darstellung der Koordinate $(4095,4095)$

Dieses Speicherformat wird für die Übergabe absoluter Koordinaten an die entsprechenden Unterprogramme benutzt.

Die Darstellung relativer Vektoren benutzt alle 16 Bits des INTEGER-Zahlenformats, wobei Bit 15 das Vorzeichen enthält.

Adresse: $\dots n \quad n+1 \quad n+2 \quad n+3 \quad \dots$



Darstellung des relativen Vektors $(-1,+265)$

Auch bei Benutzung relativer Vektoren sollte das nutzbare Koordinatenfenster nicht überschritten werden.

Einige Unterprogramme benötigen einen PEN-Parameter, der stets im .x-Register übergeben wird und wie der IPEN-Parameter der entsprechenden BASIC-Befehle arbeitet.

Die folgende Übersicht beschreibt alle Maschinenunterprogramme mit ihren Einsprungadressen, Argumenten und vergleichbaren BASIC-Befehlen.

ausgeführt hat. Es sollte stets aufgerufen werden, bevor irgendein Register des Graphikprozessors modifiziert wird, um die ungestörte Abarbeitung des laufenden oder letzten Kommandos sicherzustellen. Die Register a, x und y werden nicht verändert.

Argumente: keine

\$AEES Zeichenplot

Das Textzeichen, dessen Commodore-ASCII-Code im Akkumulator .a steht, wird in der vorher mit chrsiz definierten Größe an der aktuellen Zeichenposition auf den Bildschirm geschrieben.

Argumente: .a - Character code
.x - Pen code

\$AEES Stringplot (CHRPLT)

Zeichnet einen String auf dem Bildschirm

Argumente: .y - Länge des Strings
.x - Pen code
Zeroe Page: \$1F Startadresse des String
\$20

\$AEEB PLOT (IPLOT)

Von der aktuellen Zeichenposition wird eine Linie zu dem Punkt gezogen, der durch eine neue Koordinate definiert ist. Das Wertepaar für die neue Koordinate wurde am Anfang beschrieben und steht an einer beliebigen Stelle im Speicher. Bei Programmaufruf enthalten die Register .a und .y die Startadresse der neuen Koordinate. In .x wird der Pen code übergeben.

Argumente: .a lsb Vektor Adresse
.y msb
.x Pen code

\$AEED DPLOT (IDPLOT)

Zeichnen eines relativen Vektors. Arbeitsweise analog zu PLOT (\$AEEB).

\$AEF1 Cursor

An der aktuellen Zeichenposition wird durch Invertieren ein Kreuz erzeugt. Durch einen erneuten Aufruf dieses Unterprogramms wird das Kreuz wieder gelöscht, ohne Spuren zu hinterlassen.

Argumente: keine

\$AEF4 Mode
Arbeitsweise wie MODE-Befehl in BASIC

Argument: .x - Modus

\$AEF7 HCOFY (HCOFY)

Wie Hcopy in BASIC.

Argumente: keine

\$AEFA circle (ICRCL)

wie ICRCL in BASIC.

Argumente: .a lsb Radius
.y msb Radius
.x Pen code

\$AEFD ARCUS (ARCUS)

Wie ARCUS in BASIC, jedoch werden die Argumente RX, RY im physikalischen KOORDINATENSYSTEM übergeben. Die Winkel-Parameter sind Integer-Werte zwischen 0 (=0) und 1023 (= 360° oder 2 * π)

Argumente: .x - Pen code

\$0388	Radius x	msb
\$0389	Radius x	lsb
\$038A	Radius y	msb
\$038B	Radius y	lsb
\$038C	Startwinkel	msb
\$038D	Startwinkel	lsb
\$038E	Startwinkel	msb
\$038F	Startwinkel	lsb

5.1. Speicherbelegung:

\$033A - \$0391

RAM-Bereich für Graphik-Routinen

\$A000 - \$SAEFF

ROM-Bereich für Graphik-Routinen

\$AF00

MODE-Register, write only

- Bit 0 - Hardcopy bit, 0 für hcopy-mode
- 1 - Operating page select (nur 512 * 256 - Kopien)
- 3 - Read-modify-write Bit, aktive=1
- 4 - display switch bit, 1=graphic
- 5 - display page select (nur 512 * 256-Version)

\$0372

Kopie des Mode-Registers (für Read)

\$AF10

Bit 0: Light Pen Contact, Read only

\$AF30

Hardcopy-Register, read only

\$AF70 - \$AF7F

Graphik-Controller

Abschalten der Graphikbefehle : SYS 40991 / \$ A01F

ASCII CHARACTER GENERATOR (5 x 8 matrix)

b7	0	0	0	0	0	0
b6	0	0	1	1	1	1
b5	1	1	0	0	1	1
b4	0	1	0	1	0	1

b3	b2	b1	b0						
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						

THOMSON-EFCIS

Integrated Circuits

GRAPHIC DISPLAY PROCESSOR (GDP)

The GDP is a true high resolution graphic display processor, which contains all the functions required to process vector generation at a very high speed and to generate all the timing signals required for interfacing interlaced or non interlaced video data on a raster scan CRT display compatible with the CCIR 625 line 50 Hz standard.

The GDP flexibility results from its direct interfacing with any 8-bit MPU bus and its 11 internal registers.

The GDP's main features are :

- Selectable resolutions in black and white or color :
EF9365 : 512 x 512 (interlaced scan)
 256 x 256, 128 x 128, 64 x 64 (non interlaced scan)
EF9366 : 512 x 256 (non interlaced scan)
- High speed vector plot well suited to animation (up to 1 500 000 dots/s. and an average value of 900 000 dots/s.) - 4 types of lines.
- Multiplexed address and refresh for 16K or 4K dynamic RAMs
- On-chip full ASCII character generator (96) - maximum alphanumeric screen density : 85 x 57 - programmable sizes and orientations
- Direct interfacing with the monitor through the composite synchro and blanking signals
- Automatic allocation of display memory in refresh, write, dump, and display cycles
- Light pen registers and control signals
- Three types of interrupt requests
- Fully static design
- TTL compatible I/O
- Single + 5 volt supply.

EF9365

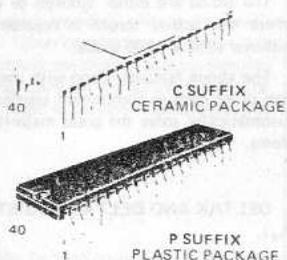
EF9366

MOS

(N - CHANNEL SILICON - GATE)

GRAPHIC DISPLAY PROCESSOR (GDP)

CASE CB-182



PIN ASSIGNMENT

CK	1	VCC	40
DAD5		DAD1	
DAD4		DAD2	
DAD3		DAD0	
DAD6		MSL1	
MSL0		MSL3	
MSL2		SYNC	
FMAT		D0	
A0		D1	
A1		D2	
A2		D3	
A3		D4	
TRQ		D5	
DW		D6	
DIN		D7	
VB		BLK	
E		WHITE	
R/W		WO	
MFREE		ALL	
VSS		LPCK	

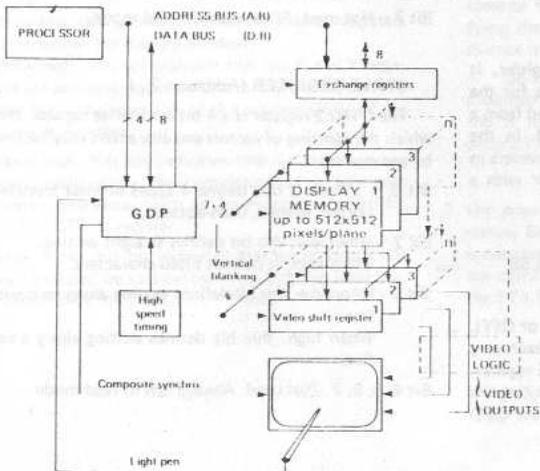


FIGURE 1 - TYPICAL APPLICATION

Developed in conjunction with the Ecole Normale Supérieure.

NT8132R1 A 1/27

THOMSON-EFCIS

Sales headquarters
 45, av. de l'Europe 78140 VÉLIZY, FRANCE
 Tel. (3) 946 97 19 / Telex : 69R R66 F

THOMSON-CSF

REGISTER DESCRIPTION

X AND Y REGISTERS (Addresses : B_{16} , 9_{16} , A_{16} , B_{16})

The X and Y registers are 12-bit read-write registers. They indicate the position of the next dot to be written into the display memory. They have no connection at all with the video signal generating scan, but they point the write address, in the same way as the pen address on a plotter.

These 2 registers are incremented or decremented, prior to each write operation into the display memory, by the internal vector and character generators, or they may be directly positioned by the microprocessor.

This 2 x 12 bit write address covers a 4096 x 4096 point addressing space. Only the LSBs are used here, since the maximum definition of the picture actually stored is 512 x 512 pixels (picture elements).

The MSBs are either ignored or used to inhibit writing where the actual screen is regarded as being a window within a 4096 x 4096 space.

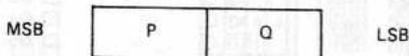
The above features along with the relative mode description of all picture component elements make it possible to automatically solve the great majority of edge cut-off problems.

DELTA X AND DELTA Y REGISTERS (Addresses : 5_{16} , 7_{16}).

The DELTA X and DELTA Y registers are 8-bit read-write registers. They indicate to the vector generator the projections of the next vector to be plotted, on the X and Y axes respectively. Such values are unsigned integers. The plotting of a vector is initiated by a write operation in the command register (CMD).

CSIZE REGISTER (Address : 3_{16})

The CSIZE register is an 8-bit read-write register. It indicates the scaling factors of X and Y registers for the symbols and characters. 98 characters are generated from a 5 x 8 pixel matrix defined by an internal ROM. In the standard version, it contains the alphanumeric characters in the ASCII code which may be printed, together with a number of special symbols.



Each symbol can be increased by a factor P(X) or Q(Y). These factors are independent integers which may each vary from 1 to 16 and which are defined by the CSIZE register. The symbol generation sequence is started after writing the ASCII code of the symbol to be represented in the CMD register.

CTRL1 REGISTER (Address : 1_{16}).

The CTRL1 register is a 7-bit read-write register, through which the general circuit operation may be fed with the required parameters.

- Bit 0 : When low, this bit inhibits writing in display memory (equivalent to pen or eraser up).
When high, this bit enables writing in display memory (pen or eraser down).
This bit controls the \overline{DW} output.
- Bit 1 : When low, this bit selects the eraser.
When high, this bit selects the pen.
This bit controls the DIN output.
- Bit 2 : When low, this bit selects normal writing mode (writing apart from the display and refresh periods, which are a requirement for the dynamic storages) in display memory.
When high, this bit selects the high speed writing mode ; the display periods are deleted. Only the dynamic storage refresh periods are retained.
- Bit 3 : When low, this bit indicates that the 4096 x 4096 space is being used (the 12 X and Y bits are significant)
When high, this bit selects the cyclic screen operating mode.
- Bit 4 : When low, this bit inhibits the interrupt triggered by the light pen sequence completion.
When high, this bit enables the interrupt.
- Bit 5 : When low, this bit inhibits the interrupt release by vertical blanking.
When high, this bit enables the interrupt.
- Bit 6 : When low, this bit inhibits the interrupt indicating that the system is ready for a new command.
When high, this bit enables the interrupt.
- Bit 7 : Not used. Always low in read mode.

CTRL2 REGISTER (Address : 2_{16})

The CTRL2 register is a 4-bit read/write register, through which the plotting of vectors and characters may be denoted by parameters.

- Bit 0, 1 : These 2 bits define 4 types of lines (continuous, dotted, dashed, dash-dotted).
- Bit 2 : When low, this bit defines straight writing.
When high, it defines tilted characters.
- Bit 3 : When low, this bit defines writing along an horizontal line.
When high, this bit defines writing along a vertical line.
- Bit 4, 5, 6, 7 : Not used. Always low in read mode.

CMD COMMAND REGISTER (Address 0_{16})

The CMD register is an 8 bit write-only register. Each write operation in this register causes a command to be executed, upon completion of the time necessary for synchronizing the microprocessor access and the GDP's CK clock.

Several types of command are available :

- vector plotting
- character plotting
- screen erase
- light pen circuitry setting
- access to the display memory through an external circuitry.
- indirect modification of the other registers (commands that make it possible for the X, Y, DELTAX, DELTAY, CTRL1, CTRL2 and CSIZE registers to be amended or scratched).

STATUS REGISTER (Address 0_{16})

The STATUS register is an 8-bit read-only register. It is used to monitor the status of the executing statements entered into the circuit, and more specifically to avoid the need for modifying a register that is already used for the command currently executing.

- Bit 0 :** When low, this bit indicates that a light pen sequence is currently executing.
When high, it indicates that no light pen sequence is currently executing.
- Bit 1 :** This bit is high during vertical blanking. It is the VB signal copy.
- Bit 2 :** When low, this bit indicates that a command is currently executing.
When high, this bit indicates that the circuit is ready for a new command.
- Bit 3 :** When low, this bit indicates that the X and Y registers point within the display window.
When high, this bit indicates that the X and Y registers are pointing outside the memory display.
This bit is the logic OR of the unused MSBs of the X and Y registers.
- Bit 4 :** When high, this bit indicates that an interrupt has been initiated by the completion of a light pen running sequence. Such an interrupt is enabled by bit 4 in CTRL1 register.
- Bit 5 :** When high, this bit indicates that an interrupt has been initiated by vertical blanking. Such an interrupt is enabled by bit 5 in CTRL1 register.

Bit 6 : When high, this bit indicates that an interrupt has been initiated by the completion of execution of a command. Such an interrupt is enabled by bit 6 in CTRL1 register.

Bit 7 : When high, this bit indicates that an interrupt has been initiated. It is the logic OR of bits 4, 5 and 6 in STATUS register. The \overline{IRQ} output state is always the opposite of the status of this bit.

Note : Bits 4, 5, 6 and 7 are reset low by a read of the STATUS register.

XLP AND YLP REGISTERS (Addresses C_{16} and D_{16})

The XLP and YLP registers are read-only registers, with 7 and 8 bits respectively. Upon completion of a light pen running sequence, they contain the display address sampled by the first edge appearing rising on the LPKK input. The use of such registers is discussed in section : Use of light pen circuitry.

NOTES :

1. All internal registers may be read or written at any time by the microprocessor. However, the precautions outlined below should be observed :
 - Do not write into the CMD register if execution of the previous command is not completed (bit 2 of STATUS register).
 - Do not alter any register if it is used as an input parameter for the internal hardwired systems (e.g. : modifying the DELTAX register while a vector plotting sequence is in progress).
 - Do not read a register that is being asynchronously modified by the internal hardwired systems (e.g. : reading the X register while a vector plotting sequence is in progress may be erroneous if CK and \overline{E} are asynchronous).
2. On powering up, the writing devices may have any status. Before entering a command for the first time, it is necessary to wait until all functions currently underway are completed, which information can be derived from the STATUS register.

THOMSON-EGCIS

HARDWIRED WRITE PROCESSOR OPERATION IN DISPLAY MEMORY

The hardwired write processors are sequenced by the master clock CK. They receive their parameters from the microprocessor bus. They control the X, Y write address, and the DIN, \overline{DW} , $\overline{M\overline{FREE}}$ and \overline{IRQ} outputs.

These hardwired processors operate in continuous mode. In the event of conflicting access to the display memory, the display and refresh processors have priority.

Since command decoding is synchronous with the CK master clock, any write operation into the (CMD) command register triggers a synchronizing mechanism which engages the circuit for a maximum of 2 CK cycles when the \overline{E} input returns high. The circuit remains engaged throughout command execution.

No further command should be entered as long as bit 2 in STATUS register is low.

VECTOR PLOTTING

The internal vector generator makes it possible to modify, within the display memory, all the dots which form the approximation of a straight line segment. All vectors plotted are described by the origin dot and the projections on the axes.

The starting point co-ordinates are defined by the X, Y register value, prior to the plotting operation. Projections onto the axes are defined as absolute values by the DELTAX and DELTAY registers, with the sign in the command byte that initiates the vector plotting process.

The vector approximation achieved here is that established by J. F. BRESENHAM ("Algorithm for computer control of a digital plotter"). This algorithm is executed by a hardwired processor which allows for a further vector component dot to be written in each CK clock cycle.

During plotting, the display memory is addressed by the X, Y registers, which are incremented or decremented.

On completion of vector plotting, they point to the end of this vector.

All vectors may be plotted using any of the following line patterns: continuous, dotted, dashed, dash-dotted, according to the 2 LSBs in register CTRL2.

Irrespective of such patterns, the plotting speed remains unchanged. The "pen down-pen up" statement required for plotting non-continuous lines is controlled by the DW output.

For a specified non-continuous line plotted vector, defined by DELTAX, DELTAY, CTRL2, CMD, the \overline{DW} sequencing during the plotting process is always the same, irrespective of vector origin and of the nature of previous plots. This feature guarantees that a specified vector can be deleted by plotting it again after moving X and Y to the starting point, and complementing bit 1 in register CTRL1.

Since the vector plotting initiation command defines the sign of the projections onto the axes, all vectors may be plotted using 4 different commands.

For increased programming flexibility, the system incorporates 16 different commands, supplemented by a set of 128 commands which make it possible to plot small size vectors by ignoring the DELTAX and DELTAY registers.

Such commands are as follows:

Basic commands

0 0 0 1 0 X X 1

DELTA sign } 0 if positive
DELTA sign } 1 if negative

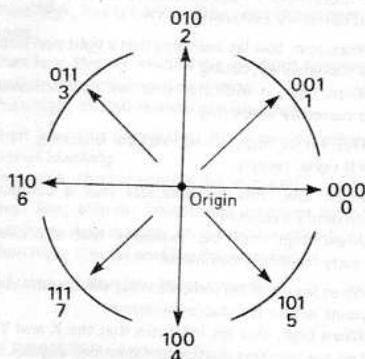
- Commands which allow ignoring the DELTAX or DELTAY registers by considering them as of zero value.

0 0 0 1 0 X X 0

0 0 DELTAX ignored, DELTAY > 0
0 1 DELTAX ignored, DELTAY > 0
1 0 DELTAX ignored, DELTAY < 0
1 1 DELTAX ignored, DELTAY < 0

Note: Bits 1 and 2 always have the same sign meaning.

These 8 codes may be summarized by the following diagram:



- Commands which allow ignoring the smaller of the two DELTAX and DELTAY registers, by considering it as being equal to the larger one, which is the same as plotting vectors parallel to the axes or diagonals, using a single DELTA register.

0 0 0 1 1 X X X

Same direction codes as above.

- Commands in which the two registers DELTAX and DELTAY may be ignored by specifying the projections through the CMD register (0 to 3 steps for each projection).

1 X X X X X X X

ΔX ΔY
(Unsigned integer values) Same direction code as previously

EXAMPLE : PLOTTING A DOTTED VECTOR

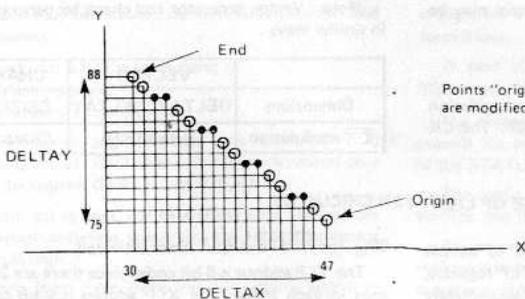
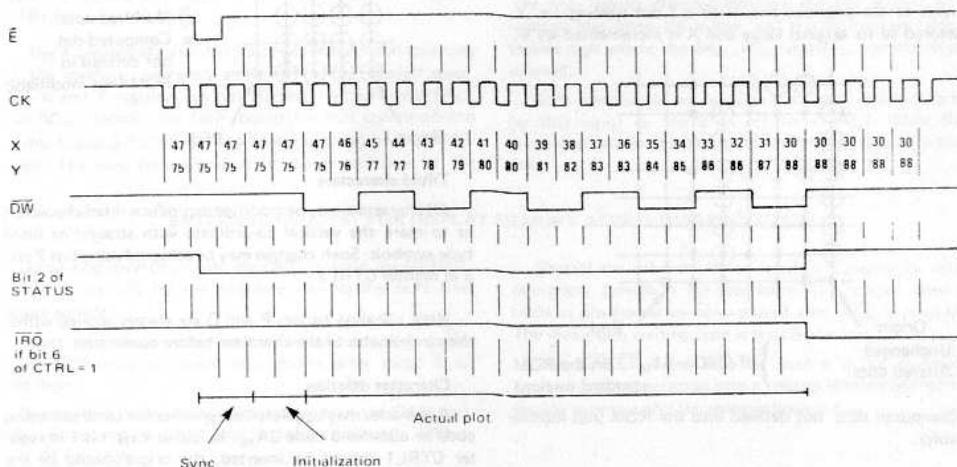
Origin : $\begin{cases} X = 47_{10} \\ Y = 75_{10} \end{cases}$

CMD = 13_{16} Corresponding to
 - Basic command,
 - DELTAX < 0
 - DELTAY > 0

Projections : $\begin{cases} DELTAX = 17_{16} \\ DELTAY = 13_{10} \end{cases}$

CTRL1 = 03_{16} Pen down
 CTRL2 = 1_{16} Dotted vector :
 2 dots on,
 2 dots off.

Plotting cycle sequence : (It is assumed that the vector generator is not interrupted by the display or refresh cycle).



Points "origin" and "end" are modified

Display shows : ● unmodified dot
 ○ dot on

Note :

Plotting a vector with DELTAX = DELTAY = 0 writes the point X, Y in memory. It occupies the vector generator for synchronization, initialization and one write cycle.

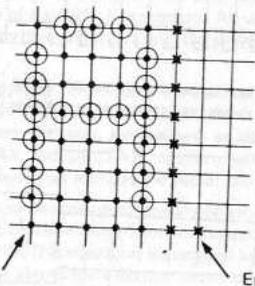
CHARACTER AND SYMBOL GENERATOR

The character generator operates in the same way as the vector generator, i.e. through incrementing or decrementing the X, Y registers, in conjunction with a DW output control.

It receives parameters from the CSIZE, CTRL2 and CMD registers. The characters plotted are selected, according to the CMD value, out of 98 matrices (97 8-dot high x 5-dot wide rectangular matrices, and one 4 dot x 4 dot matrix) defined in an internal ROM. Two scaling factors may be applied to the characters plotted using X and Y defined by the CSIZE register. The characters may be tilted, according to the content of register CTRL2.

Basic matrix

Upon completion of a character writing process, the X and Y registers are positioned for writing a further character next to the previous one, with a 1 dot spacing, i.e. Y is restored to its original value and X is incremented by 6.



- Unchanged
 - ⊙ Altered dots
 - ⊗ Computed dots, not defined into the ROM (not modifiable).
- } if CMD = 41₁₆ (in the ROM standard version)

Scaling factors

Each individual dot in the 5 x 8 basic matrix may be replaced by a P x Q size block.

P : X co-ordinate scaling factor

Q : Y co-ordinate scaling factor

The character size becomes 5P x 8Q. Upon completion of the writing process, X is incremented by 6P. The CK clock cycle count required is 6P x 8Q.

USE OF LIGHT PEN CIRCUITRY

A rising edge on the LPCK input is used to sample the current display address in the XLP and YLP registers, provided that this edge is present in the frame immediately following loading of the 08₁₆ or 09₁₆ code into the CMD register.

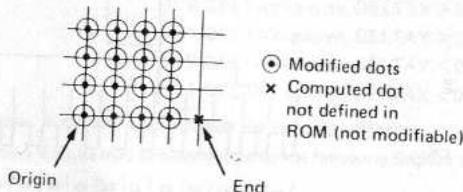
Here, the frame origin is counted starting with the VB falling edge: With code 08₁₆, the WHITE output recopies the BLK signal from the frame origin up to the rising edge on the LPCK input, or when VB starts rising again, if the LPCK input remains low for the entire frame. With code

P and Q may each take values from 1 through 16. They are defined by the CSIZE register. Each value is encoded on 4 bits, value 16 being encoded as 0₁₆.

In register CSIZE, P is encoded on the 4 MSBs and Q on the 4 LSBs.

Among the 97 rectangular matrices available in the standard ROM, 96 correspond to CMD values ranging from 20₁₆ to 7F₁₆, and the 97th matrix to 0A₁₆. In the standard version, these values correspond to the 96 printable characters in the ASCII set. The 97th character is a 5P x 8Q block which may be used for deleting the other characters.

The 98th code (0B₁₆) is used to plot a 4P x 4Q graphic block. It locates X, Y, without spacing for the next symbol. Such a block makes it possible to pad uniform areas on the screen.



Tilted characters

All characters may be modified to produce tilted characters or to mark the vertical co-ordinate with straight or tilted type symbols. Such changes may be achieved using bits 2 and 3 in register CTRL2.

Note : Scaling factors P and Q are always applied within the co-ordinates of the character before conversion.

Character deletion

A character may be deleted using either the same command code or command code 0A₁₆. In either case, bit 1 in register CTRL1 should be inverted, the origin should be the same as prior to a character plotting operation, as should the scaling factors.

Note : Vector generator and character generator operate in similar ways :

	VECTOR	CHARACTER
Dimensions	DELTA X, DELTA Y	CSIZE, tilting
DW modulation	Type of line	Character code

09₁₆, the WHITE output is not activated.

The YLP address is 8-bit coded since there are 256 display lines in each frame. The XLP address is 6-bit coded since there are 64 display cycles in each line.

These 6 bits are left justified in the XLP register. XLP and YLP register contents match the write address if FMAT is low (or for the EF9366), but should be multiplied by 2 if FMAT is high, so as to be able to match the write address.

The address sampled into XLP corresponds to the current memory cycle. Data detected by the light pen were addressed in the memory during the previous cycle. Hence, 1 should be subtracted from bit 2 in XLP register where the light pen electronic circuitry does not produce any additional delay.

If the rising edge on input LPCK occurs while VB is low, then the LSB in XLP is set high. This bit acts as a status signal which is reset to the low state by reading register XLP or YLP.

SCREEN BLANKING COMMANDS

Three commands (04_{16} , 06_{16} , 07_{16}) will set the whole display memory to a status corresponding to a "black display screen" condition. Another command ($0C_{16}$) may be used to set the whole memory to a status other than black (this condition being determined by bit 1 in register CTRL1).

The 4 commands outlined above use the planned scanning of the memory addresses achieved by the display stage. The X and Y registers are not affected by commands 04_{16} , and $0C_{16}$. Hence, the time required is that corresponding to one frame (EF9366 or FMAT low) or two frames (FMAT high). The time corresponding to the completion of the

frame currently executing when the CMD register is loaded, should be added to the above time.

When commands 08_{16} , or 09_{16} , have been decoded, bit 2 of the status register goes high (circuit ready for any further command) and bit 0 goes low (light pen operating sequence underway).

For the screen blanking process, the frame origin is counted starting with the VB falling edge.

The only signals affected here are the \overline{DW} output, which remains low when VB is low, and the DIN output which is forced high where the 04_{16} , 06_{16} and 07_{16} commands are entered.

Such commands are activated without requiring action by WO input or bit 2 in register CTRL1. While these commands are executing, bit 2 in STATUS register remains low.

EXTERNAL REQUEST FOR DISPLAY MEMORY ACCESS (MFRÉE OUTPUT)

On writing code $0F_{16}$ into the CMD register, the MFRÉE output is set low by the circuitry, during the next free memory cycle.

Apart from the display and refresh periods, this cycle is the first complete cycle that occurs after input \bar{E} is reset high.

During this cycle, those addresses output on DAD and MSL correspond to the X and Y register contents: \overline{DW} is high, \overline{ALL} is high.

Should the memory be engaged in a display or refresh operation, (which is the case when \overline{ALL} is low), then this cycle is postponed to be executed after \overline{ALL} is reset high. The maximum waiting time is thus 64 cycles.

The MFRÉE signal may be used e. g. for performing a read or write operation into a register located between the display memory and the microprocessor bus.

INTERRUPTS OPERATION

An interrupt may be initiated by three situations denoted by internal signals:

- Circuit ready for a further command
- Vertical blanking signal
- Light pen sequence completed.

These three signals appear in real time in the STATUS register (bits 0, 1, 2). Each signal is cross-referenced to a mask bit in the register CTRL1 (bits 4, 5, 6).

If the mask bit is high, the first rising edge that occurs on the interrupt initiating signal sets the related interrupt flip-flop circuit high.

The outputs from these three flip-flop circuits appear in the STATUS register (bits 4, 5, 6). If one flip-flop circuit

is high, bit 7 in the STATUS register is high, and pin \overline{IRO} is forced low.

A read operation in the STATUS register resets its 4 MSBs low, after input \bar{E} is reset high.

The three interrupt control flip-flops are duplicated to prevent the loss of an interrupt coming during a read cycle of the STATUS register.

The status of bits 4, 5 and 6 corresponds to the interrupt control flip-flop circuit output, before input \bar{E} goes low.

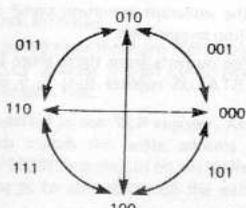
An interrupt coming during a read cycle of the STATUS register does not appear in bits 4, 5 and 6 during this read sequence, but during the following one. However, it may appear in bits 0, 1, 2 or on pin \overline{IRO} .

TABLE 1 - REGISTER ADDRESS

ADDRESS REGISTER					REGISTER FUNCTIONS		Number of bits
Binary				Hexa	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	DELTA X		8
0	1	1	0	6	Reserved		—
0	1	1	1	7	DELTA Y		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		—

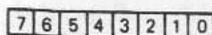
Reserved : These addresses are reserved for future versions of the circuit. In read mode, output buffers D0-D7 force a high state on the data bus.

TABLE 2 - COMMAND REGISTER

b7 b6 b5 b4	b3 b2 b1 b0	Command Register Data																																									
		0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1																									
		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)	SPACE	0	@	P	˘	p	SMALL VECTOR DEFINITION : <table border="1" style="margin: 5px;"> <tr> <th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> <tr> <td>1</td><td> ΔX </td><td> ΔY </td><td colspan="5">Direction</td> </tr> </table> Dimension <table border="1" style="margin: 5px;"> <tr> <th>ΔX or ΔY</th><th>Vector length</th> </tr> <tr> <td>0 0</td><td>0 step</td> </tr> <tr> <td>0 1</td><td>1 step</td> </tr> <tr> <td>1 0</td><td>2 steps</td> </tr> <tr> <td>1 1</td><td>3 steps</td> </tr> </table> Direction 								b7	b6	b5	b4	b3	b2	b1	b0	1	ΔX	ΔY	Direction					ΔX or ΔY	Vector length	0 0	0 step	0 1	1 step	1 0	2 steps	1 1	3 steps
b7	b6	b5		b4	b3	b2	b1	b0																																			
1	ΔX	ΔY		Direction																																							
ΔX or ΔY	Vector length																																										
0 0	0 step																																										
0 1	1 step																																										
1 0	2 steps																																										
1 1	3 steps																																										
0 0 0 1	1	Clear bit 1 of CTRL1 : Eraser selection		1	A	Q	a	q																																			
0 0 1 0	2	Set bit 0 of CTRL1 : Pen/Eraser down selection	"	2	B	R	b	r																																			
0 0 1 1	3	Clear bit 0 of CTRL1 : 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	☒																																			

OTHER REGISTERS

STATUS REGISTER (Read only)

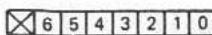


- 7 → HIGH = light-pen sequence ended
- 6 → HIGH = vertical blanking (idem on pin VB)
- 5 → HIGH = ready for a new command ; LOW = busy
- 4 → HIGH = pen out of display window (logical OR of the 6 MSBs of the X and Y registers)
- 3 → HIGH = light-pen sequence ended IRQ (if enabled)
- 2 → HIGH = vertical blanking IRQ (if enabled)
- 1 → HIGH = ready for a new command IRQ (if enabled)
- 0 → IRQ : logical OR of bits 4,5,6 ; HIGH when \overline{IRQ} output is low.

} These 3 bits are not latched and not masked

} These 3 bits are reset after a read cycle of the status register.

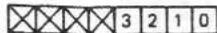
CONTROL REGISTER 1 (Read/Write)



- 6 → HIGH = pen down ; LOW = pen up (control \overline{DW} output)
- 5 → HIGH = pen ; LOW = eraser (control DIN output)
- 4 → HIGH = high speed write : no video (BLK output is high, mini. of memory refresh cycles)
- 3 → HIGH = cyclic screen (memory display write even if bit 3 of the status register is high)
- 2 → HIGH = enable end of the light pen sequence IRQ
- 1 → HIGH = enable VB IRQ
- 0 → HIGH = enable ready for a new command IRQ
- Not used (0 for reading)

} Interrupt masks

CONTROL REGISTER 2 (Read/Write)



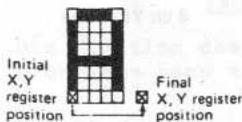
Not used (always read as 0)

- 3 → Type of vectors
- 2 → HIGH = tilted character
- 1 → HIGH = character on vertical axis
- 0 →

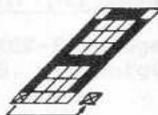
b1	b0	Type of vectors
0	0	continuous
0	1	dotted
1	0	dashed
1	1	dotted-dashed

2 dots on, 2 dots off
4 dots on, 4 dots off
10 dots on, 2 dots off, 2 dots on, 2 dots off.

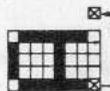
Types of character orientations



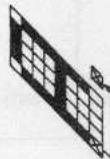
$b_3 = 0, b_2 = 0$



$b_3 = 0, b_2 = 1$

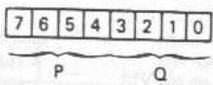


$b_3 = 1, b_2 = 0$



$b_3 = 1, b_2 = 1$

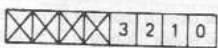
C-SIZE REGISTER (Read/Write)



P : Scaling factor on X axis
 Q : Scaling factor on Y axis

P and Q may take any value between 1 and 16. This value is given by the leftmost or rightmost 4 bits for P and Q respectively. Binary value (0) means 16.

X AND Y REGISTERS (Read/Write)



MSBs



LSBs

The 4 leftmost MSBs are always 0.

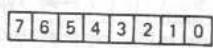
XLP and YLP REGISTERS



Status bit indicating if a rising edge has been applied on LPCK during the first complete frame following light-pen initialization. This bit is reset by a read on XLP or YLP.

always 0

6 bit XLP value



8 bit YLP value

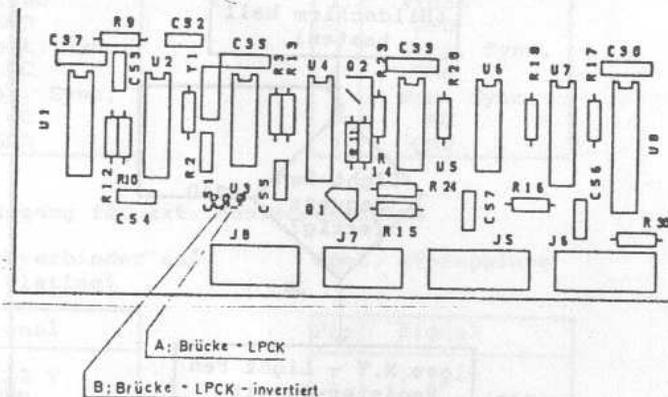
Anhang C: Light-Pen Anschluß

Auf der Grafik-Platine ist ein Anschluß (J8) für einen Light-Pen vorgesehen. Es stehen folgende Signale zur Verfügung:

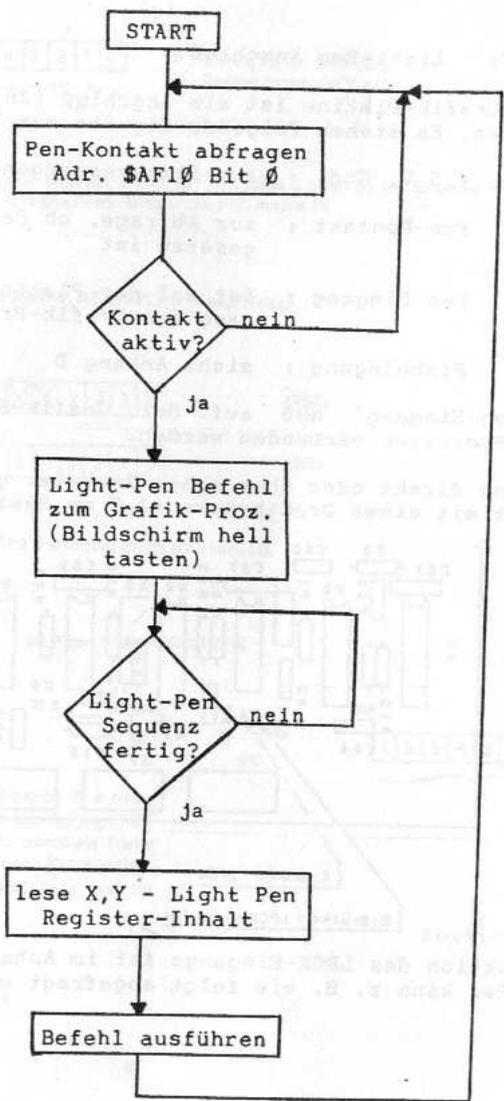
- + 5 V, Gnd : zur Stromversorgung des Light-Pen
- Pen-Kontakt : zur Abfrage, ob Pen auf Bildschirm aufgesetzt ist
- Pen-Eingang : ist auf der Platine mit dem LPCK Eingang des Grafik-Prozessors zu verbinden
- Pinbelegung : siehe Anhang D

Der "Pen-Eingang" muß auf der Grafik-Platine noch mit dem Grafik-Prozessor verbunden werden.

Dies kann direkt oder über einen Inverter geschehen. Die Verbindung ist mit einer Drahtbrücke (2,5 mm Raster) vorzunehmen:



Die Funktion des LPCK-Eingangs ist im Anhang B beschrieben. Der Light-Pen kann z. B. wie folgt abgefragt werden:



Anhang D: Steckerbelegung
Stromversorgung
J1

Pin	Signal
1	- 9 V
2	- 9 V
3	NC
4	+ 16 V
5	+ 16 V
6	GND
7	GND

J2

Pin	Signal
1	+ 9 V
2	NC
3	NC
4	+ 9 V
5	GND
6	+ 9 V
7	GND

Videoanschluß für CBM-Bildschirm
J5 Video-Eingang

Pin	Signal
1	Video
2	GND
3	Vert. Sync.
4	NC
5	Hor. Sync.
6	NC
7	GND

J6 Video-Ausgang

Pin	Signal
1	Video
2	GND
3	Vert. Sync.
4	GND
5	Hor. Sync.
6	NC
7	GND

Video-Ausgang für ext. Monitoranschluß
J7 (Steckverbinder auf der Platine)

Pin	Signal
1	+ 12 V
2	GND
3	Video (BAS)
4	NC
5	GND

6pol. AV-Kupplung

Pin	Signal
1	+ 12 V
2	Video (BAS)
3	GND
4	GND
5	NC
6	NC

**Light-Pen Anschluß
J8**

Pin	Signal
1	+ 5 V
2	NC
3	Pen-Kontakt
4	GND
5	Pen-Eingang
6	GND

**Schalter für Bildschirmmodus
J9**

Pin	Signal
1	+ 5 V
2	Mode-Eingang
3	NC
4	GND

Anschluß an den Computer Expansion-Bus

Über die Steckverbinder J3 und J4 ist die Grafik-Platine mit dem CBM-Computer verbunden. Alle Signale des Memory-Expansion-Bus vom CBM sind auf der Grafik-Platine zu der äußeren freien Lötungen-Reihe durchgeschleift. In der Tabelle sind alle Signale des Expansion-Bus aufgeführt und angegeben welche von der Grafik (x) benutzt werden.

Pin	J3 Signal von CBM	Grafik	J4 Signal von CBM	Grafik
1	GND	x	GND	x
2	BA 0	x	BD 0	x
3	BA 1	x	BD 1	x
4	BA 2	x	BD 2	x
5	BA 3	x	BD 3	x
6	BA 4	x	BD 4	x
7	BA 5	x	BD 5	x
8	BA 6	x	BD 6	x
9	BA 7	x	BD 7	x
10	BA 8	x	NC	
11	BA 9	x	NC	
12	BA 10	x	<u>SEL 4</u>	
13	BA 11	x	<u>SEL 5</u>	
14	BA 12		<u>SEL 6</u>	
15	BA 13		<u>SEL 7</u>	
16	BA 14		<u>SEL 8</u>	
17	BA 15		<u>SEL 9</u>	x 1)
18	SYNC		<u>SEL A</u>	x
19	<u>IRQ</u>		<u>SEL B</u>	
20	DIAG		<u>NO ROM</u>	
21	C02	x	<u>PEN STR</u>	
22	<u>BR/W</u>	x	<u>RES</u>	
23	<u>BR/W</u>		<u>RDY</u>	
24	NC		<u>NMI</u>	
25	GND	x	<u>GND</u>	x

1) führt nur zum Jumper M auf der Grafik-Platine

Handbuch des Commodore Expansion-Bus

Das die Erweiterungen 11 und 14 ist die Geräte-Liste mit den Adressen, welche die Module des Memory-Expansion-Bus von 11 bis auf den Geräte-Bus in den anderen Modulen angeschlossen werden dürfen. In der Tabelle sind alle Adressen angegeben, die für die Erweiterung 11 und 14 verwendet werden können.

Erweiterung	Signal von CPU	Geräte	Signal von CPU	Geräte
11	00	00	00	00
11	01	01	01	01
11	02	02	02	02
11	03	03	03	03
11	04	04	04	04
11	05	05	05	05
11	06	06	06	06
11	07	07	07	07
11	08	08	08	08
11	09	09	09	09
11	0A	0A	0A	0A
11	0B	0B	0B	0B
11	0C	0C	0C	0C
11	0D	0D	0D	0D
11	0E	0E	0E	0E
11	0F	0F	0F	0F
11	10	10	10	10
11	11	11	11	11
11	12	12	12	12
11	13	13	13	13
11	14	14	14	14
11	15	15	15	15
11	16	16	16	16
11	17	17	17	17
11	18	18	18	18
11	19	19	19	19
11	1A	1A	1A	1A
11	1B	1B	1B	1B
11	1C	1C	1C	1C
11	1D	1D	1D	1D
11	1E	1E	1E	1E
11	1F	1F	1F	1F
11	20	20	20	20
11	21	21	21	21
11	22	22	22	22
11	23	23	23	23
11	24	24	24	24
11	25	25	25	25
11	26	26	26	26
11	27	27	27	27
11	28	28	28	28
11	29	29	29	29
11	2A	2A	2A	2A
11	2B	2B	2B	2B
11	2C	2C	2C	2C
11	2D	2D	2D	2D
11	2E	2E	2E	2E
11	2F	2F	2F	2F
11	30	30	30	30
11	31	31	31	31
11	32	32	32	32
11	33	33	33	33
11	34	34	34	34
11	35	35	35	35
11	36	36	36	36
11	37	37	37	37
11	38	38	38	38
11	39	39	39	39
11	3A	3A	3A	3A
11	3B	3B	3B	3B
11	3C	3C	3C	3C
11	3D	3D	3D	3D
11	3E	3E	3E	3E
11	3F	3F	3F	3F
11	40	40	40	40
11	41	41	41	41
11	42	42	42	42
11	43	43	43	43
11	44	44	44	44
11	45	45	45	45
11	46	46	46	46
11	47	47	47	47
11	48	48	48	48
11	49	49	49	49
11	4A	4A	4A	4A
11	4B	4B	4B	4B
11	4C	4C	4C	4C
11	4D	4D	4D	4D
11	4E	4E	4E	4E
11	4F	4F	4F	4F
11	50	50	50	50
11	51	51	51	51
11	52	52	52	52
11	53	53	53	53
11	54	54	54	54
11	55	55	55	55
11	56	56	56	56
11	57	57	57	57
11	58	58	58	58
11	59	59	59	59
11	5A	5A	5A	5A
11	5B	5B	5B	5B
11	5C	5C	5C	5C
11	5D	5D	5D	5D
11	5E	5E	5E	5E
11	5F	5F	5F	5F
11	60	60	60	60
11	61	61	61	61
11	62	62	62	62
11	63	63	63	63
11	64	64	64	64
11	65	65	65	65
11	66	66	66	66
11	67	67	67	67
11	68	68	68	68
11	69	69	69	69
11	6A	6A	6A	6A
11	6B	6B	6B	6B
11	6C	6C	6C	6C
11	6D	6D	6D	6D
11	6E	6E	6E	6E
11	6F	6F	6F	6F
11	70	70	70	70
11	71	71	71	71
11	72	72	72	72
11	73	73	73	73
11	74	74	74	74
11	75	75	75	75
11	76	76	76	76
11	77	77	77	77
11	78	78	78	78
11	79	79	79	79
11	7A	7A	7A	7A
11	7B	7B	7B	7B
11	7C	7C	7C	7C
11	7D	7D	7D	7D
11	7E	7E	7E	7E
11	7F	7F	7F	7F
11	80	80	80	80
11	81	81	81	81
11	82	82	82	82
11	83	83	83	83
11	84	84	84	84
11	85	85	85	85
11	86	86	86	86
11	87	87	87	87
11	88	88	88	88
11	89	89	89	89
11	8A	8A	8A	8A
11	8B	8B	8B	8B
11	8C	8C	8C	8C
11	8D	8D	8D	8D
11	8E	8E	8E	8E
11	8F	8F	8F	8F
11	90	90	90	90
11	91	91	91	91
11	92	92	92	92
11	93	93	93	93
11	94	94	94	94
11	95	95	95	95
11	96	96	96	96
11	97	97	97	97
11	98	98	98	98
11	99	99	99	99
11	9A	9A	9A	9A
11	9B	9B	9B	9B
11	9C	9C	9C	9C
11	9D	9D	9D	9D
11	9E	9E	9E	9E
11	9F	9F	9F	9F
11	100	100	100	100
11	101	101	101	101
11	102	102	102	102
11	103	103	103	103
11	104	104	104	104
11	105	105	105	105
11	106	106	106	106
11	107	107	107	107
11	108	108	108	108
11	109	109	109	109
11	10A	10A	10A	10A
11	10B	10B	10B	10B
11	10C	10C	10C	10C
11	10D	10D	10D	10D
11	10E	10E	10E	10E
11	10F	10F	10F	10F
11	110	110	110	110
11	111	111	111	111
11	112	112	112	112
11	113	113	113	113
11	114	114	114	114
11	115	115	115	115
11	116	116	116	116
11	117	117	117	117
11	118	118	118	118
11	119	119	119	119
11	11A	11A	11A	11A
11	11B	11B	11B	11B
11	11C	11C	11C	11C
11	11D	11D	11D	11D
11	11E	11E	11E	11E
11	11F	11F	11F	11F
11	120	120	120	120
11	121	121	121	121
11	122	122	122	122
11	123	123	123	123
11	124	124	124	124
11	125	125	125	125
11	126	126	126	126
11	127	127	127	127
11	128	128	128	128
11	129	129	129	129
11	12A	12A	12A	12A
11	12B	12B	12B	12B
11	12C	12C	12C	12C
11	12D	12D	12D	12D
11	12E	12E	12E	12E
11	12F	12F	12F	12F
11	130	130	130	130
11	131	131	131	131
11	132	132	132	132
11	133	133	133	133
11	134	134	134	134
11	135	135	135	135
11	136	136	136	136
11	137	137	137	137
11	138	138	138	138
11	139	139	139	139
11	13A	13A	13A	13A
11	13B	13B	13B	13B
11	13C	13C	13C	13C
11	13D	13D	13D	13D
11	13E	13E	13E	13E
11	13F	13F	13F	13F
11	140	140	140	140
11	141	141	141	141
11	142	142	142	142
11	143	143	143	143
11	144	144	144	144
11	145	145	145	145
11	146	146	146	146
11	147	147	147	147
11	148	148	148	148
11	149	149	149	149
11	14A	14A	14A	14A
11	14B	14B	14B	14B
11	14C	14C	14C	14C
11	14D	14D	14D	14D
11	14E	14E	14E	14E
11	14F	14F	14F	14F
11	150	150	150	150
11	151	151	151	151
11	152	152	152	152
11	153	153	153	153
11	154	154	154	154
11	155	155	155	155
11	156	156	156	156
11	157	157	157	157
11	158	158	158	158
11	159	159	159	159
11	15A			