commodore

High Speed Graphics for CBM 8032

CBM 4016 and CBM 4032 (with 12" Monitor)

I N S T R U C T I O N S

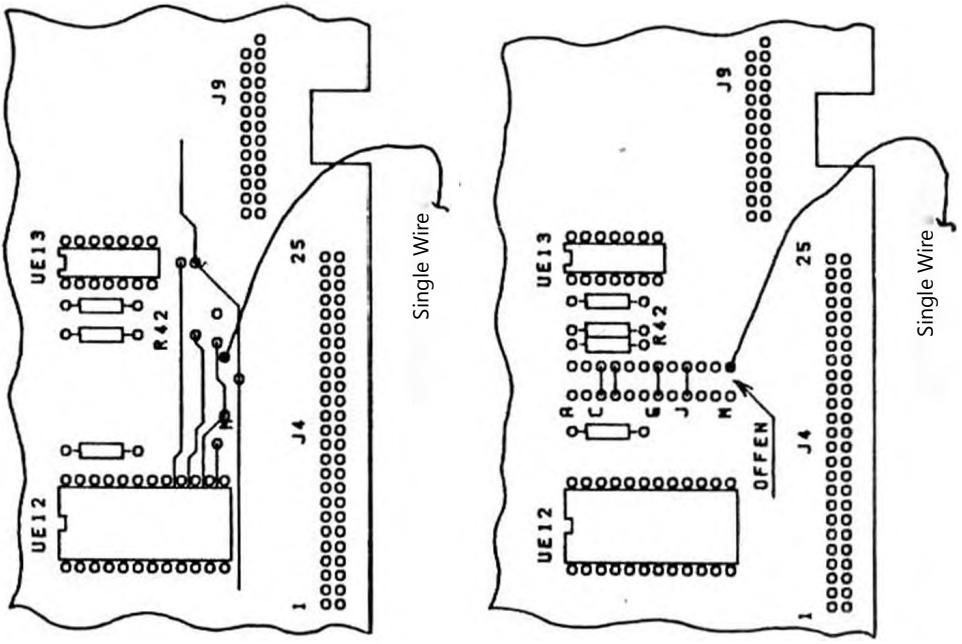
Table of Contents

commodore

The installation of the graphics board is described in the following section.

2. Installation Instructions

For the CBM computer to address the graphics board, the address range $A000-AFFF must be switched to external operation. Locate the wire jumper "M" on the CBM board and remove it. Solder the provided single-wire cable into the empty hole at "M" facing the rear (see Figure 1). No other soldering is required for the installation.



CBM BOARD VERSION: 1

CBM BOARD VERSION: 2

Figure 1. Connecting the single-wire cable to the CBM board

Socket UD11 on the CBM board must now remain empty. Its address range now controls the graphics board.

Now that the wire has been soldered into place, all connections can be made (see also Figure 2):

- The graphics board has two 50-pin female connectors that mount on top of the expansion headers (J4 and J9) of the CBM. Plug the board into the headers, taking care that the single-wire cable can still be passed freely up to its connector (J10) on the graphics board. The graphics board should be firmly seated on the headers. Check the alignment and ensure that all pins are inserted into the graphics board connectors.

- Plug the cable with the 2-pin connector into J10 on the graphics board. The other end has been soldered to the CBM board.

- Connect the power supply cables to the CBM as follows:
  J1 (Graphics) to J10 (CBM)
  J2 (Graphics) to J11 (CBM)

- Remove the internal monitor cable from J7 on the CBM and plug it into J6 on the graphics board.

- Connect a cable from the now-empty monitor port at J7 on the CBM to J5 on the graphics board.

- Mount the toggle switch to a convenient location on the CBM chassis, such as the lower right side of the bottom pan. A mounting bracket with double-sided tape has been provided. Ensure that the PET case can still be closed properly with the switch mounted.

- The last remaining cable has a 6-pin DIN socket on one end. Plug its other end into J8 on the graphics board and route the DIN socket to the outside of the CBM. This cable is used to drive an external monitor.

commodore

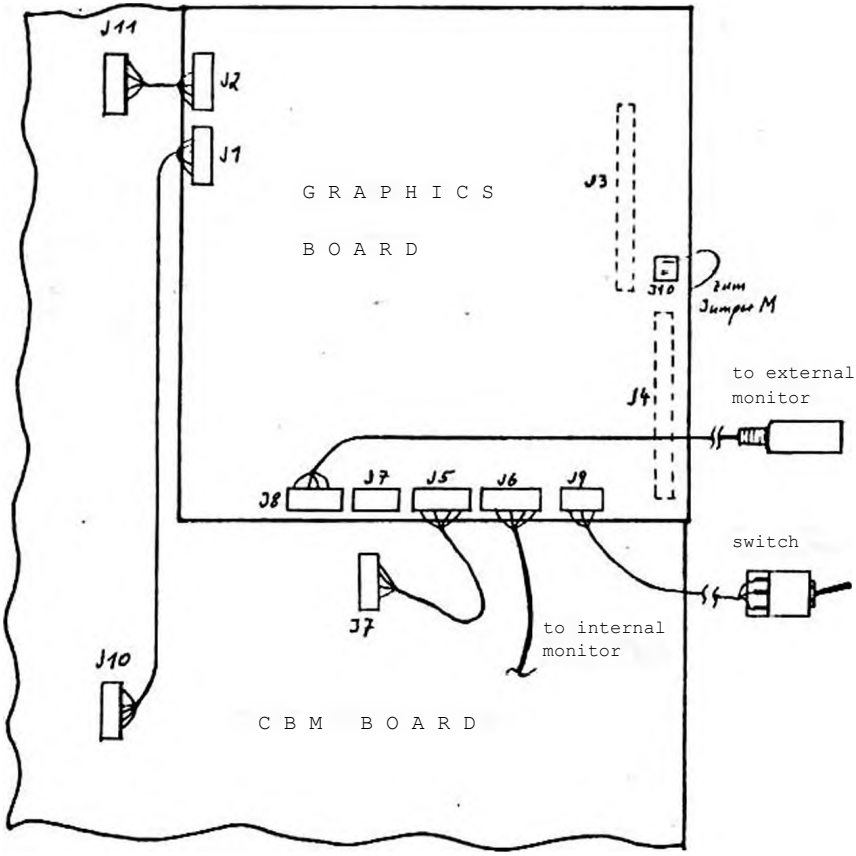


Figure 2: Installing the Graphics Board in the CBM

With installation of the graphics board finished, the graphics mode should now function.

The toggle switch has the following positions:

Switch position 1 (front) : Normal CBM mode

Switch position 2 (middle) : Software changes mode

Switch position 3 (rear) : Graphics mode

To test whether the graphics add-on works properly, do the following:

Turn the computer on. If the screen does not show the BASIC banner, then the switch is in position 3 (graphics mode). Set the switch to position 1 (normal CBM mode) and wait about 3 seconds. The screen should change and show the BASIC banner. If not, turn the computer off and check all connections again.

Input: sys 40960

Message: graphic rev. .....
ready.

BASIC Program:

```
10   init                          (Initialize graphics)
20   display (1)                   (Change screen to make the
                                    graphics display active)
20   plot (1, 1, 1)                (Draw a diagonal line)
30   plot (0, 0.3, 0)              (New position)
40   chrsiz (4, 7)                 (Set character size)
50   chplot ("CBM Graphics", 1)    (Draw a string)
60   get a$: if a$ = "" then 60
70   display (0)                   (Change to CBM display)
```

Set the switch to the middle position. Changing to graphics mode and back to normal CBM mode can now be done under software control. Now start the program with "run".

On the screen should be a diagonal line from the bottom left to the top right. To the left of the line should be the text "CBM Graphics". If you press any key, the computer reverts to normal operation (it takes about 3 seconds for the image to appear).

The screen will also show:

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

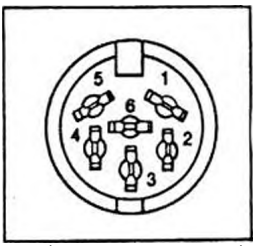
If everything happens as described above, then the graphics board is working correctly.

2.1. External Video Connection

The graphics can be displayed on the internal CBM monitor or an external monitor (or a TV with a 6-pin input jack). When a second monitor is connected, one can interact with the CBM using its standard character display on the internal CBM monitor while simultaneously displaying the new graphics mode on the external monitor.

It is not possible for the graphics board to reproduce the standard CBM display on the external monitor.

The external monitor port is standard. The monitor or TV is connected to the graphics board via a 6-pin A/V cable (like a video recorder). The pin assignments are as follows:



1: +12V (switching voltage)
2: Video Out
3: GND
4: GND
5: NC
6: NC

Figure 3: Pinout of the video jack

The video output signal (BAS, CCIR) provides 1 Vpp into 75 Ohms.

The switching voltage +12V (pin 1) is used to switch to video mode when connected to a TV. This voltage may not be used for powering other circuits.

Note: The graphics board configured for 512x512 (Version A) should be used with a picture tube that has slow phosphors or else the image will have noticeable flicker.

2.2. External ROM in the range $9000-9FFF

It is possible to use another expansion besides the graphics board, as long as the address range $A000-AFFF is not used. The expansion bus J4 and J9 of the Commodore computer is also available on the graphics board. There is also a row of soldering points next to each 50-pin header connection. These are also connected to the expansion bus. Wires or header connectors can be soldered to them.

In case the address range $9000-$9FFF is needed externally, the computer need to be configured for that. Without the graphics board, a wire connection has to be soldered onto jumper "M" on the CBM board. With the graphics board there must not be a connection in this place. The configuration must be done on the graphics board. To do this, a wire connection has to be soldered between IC U19 and the 50-pin header connector. This is also named "M". The ROM socket UD12 on the CBM board must be kept empty in this case.

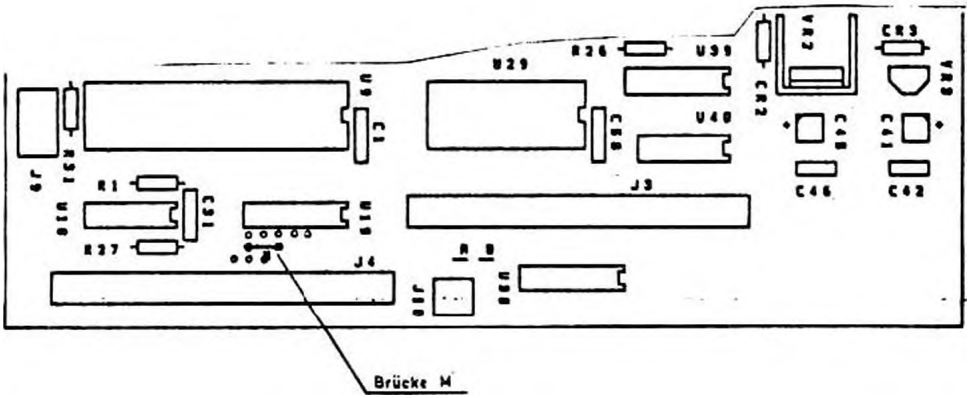


Figure 4: Jumper M on the Graphics Board

2.3. Differences between graphics versions A and B

The graphics board is available in two different versions. The difference is in the resolution.

Version A:

- 512 lines by 512 pixels/line (262,144 total pixels)
- Storage memory for one screen
- Display of 512 lines interlaced (25 Hz refresh rate)

Version B:

- 256 lines by 512 pixels/line (131,072 total pixels)
- Storage memory for two screens
- Display of 256 lines non-interlaced (50 Hz refresh rate)

In Version A, the refresh rate is only 25 Hz and will cause a noticeable flicker on a standard monitor. It should be used with a picture tube that has slow phosphors.

In Version B, the refresh rate is 50 Hz and does not flicker.

The graphics board is configured for either version by installing the required graphics processor and jumpering "A" or "B" on the board:

	Version A	Version B
Graphics Processor	EF9365	EF9366
Wire bridge (between J3 and J10)	A B	A B

The EPROM is the same for both versions.

3. New BASIC commands for graphics

The new BASIC commands are described in this chapter. They are implemented in the EPROM on the graphics board. The examples and demo programs given here always refer to the Version A with 512x512 pixel resolution. For Version B (512x256), the Y-coordinates must be changed accordingly. To initialize the new BASIC commands, enter following command on the CBM:

```
sys 40960
```

The CBM then responds with:

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

From now on, the CBM will understand the new graphics commands directly or when used in a program. As with the normal BASIC commands, the new commands may be abbreviated by shifting the second or third character. The abbreviated form is given with each. The new commands may be used freely in a BASIC program with one exception. When using an "IF THEN" statement, a graphics command may not immediately follow "THEN". It must be separated by a colon, e.g.:

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

Es ist unbedingt darauf zu achten, daß vor Eingabe eines Programms der BASIC-Befehlasatz 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 commands for display control

3.1.1. init 'inI'

This command clears the graphics screen, initializes the graphics processor, and sets some parameters as follows:

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

The starting position is at x=0, y=0. In addition, the "graphics rev . ...." message is displayed. When sys 40960 is run, the init command is automatically executed.

3.1.2. cscr 'cS'

This will clear the screen. No other parameters are changed.

In Version B (512x256) with memory for two screens, only the current screen selected for editing is cleared.

3.1.3. display (I) 'diS'

With this command, the screen mode of the computer is changed by software. To do this, the toggle switch must be in the middle (position 2) position. The change occurs with:

display (1) - Graphics rendering on the screen

display (0) - Normal text screen (CBM mode)

When switching from display (1) to display (0) it takes about 3 seconds until the other image appears. However, the external video output is not affected. It always displays graphics mode.

If you ever find that you are stuck in the graphics mode, such as you are developing a program but have not yet implemented the switch back to CBM mode, you do not need to type display (0) blindly. You only need to move the toggle switch to position 1 (CBM mode). When the switch is in positions 1 and 3, software control of the mode is not possible.

Here are the switch positions:

Position 1 (front) : Normal CBM Mode
Position 2 (middle) : Software changes mode
Position 3 (rear) : Graphics mode

3.1.4. mode (I) 'mO'

The mode command is only applicable to Version B (512x256 pixels). In this version, there is memory for 2 graphics screens. This means that there is one screen page that is displayed and a second page that is not displayed. The mode command selects which page should be displayed on the screen and also which page should be the target for graphics drawing.

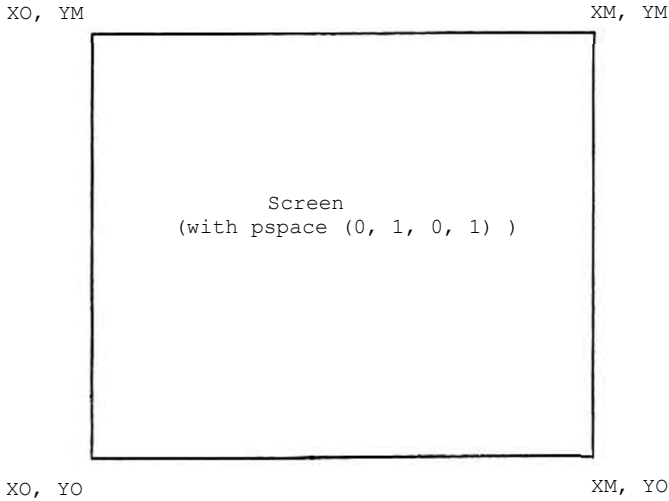
The mode command has the following options:

MODE (I)	Display Page	Edit Page
0	1	1
1	1	2
2	2	1
3	2	2

For example, with mode (1) the screen will display page 1 but drawing commands will modify page 2.

3.1.5. map (XO, XM, YO, YM) 'mA'

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 befir.det sich in der linken unteren Bildecke. Die Maximalwerte (rechte, obere Bildecke) werden über XM und YM angegeben.



Parameter:
XO - kleinste dargestellte X-Koordinate
XM — größt© dargestollt© X-Koordinate
YO - größt© dargestollt© X-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.

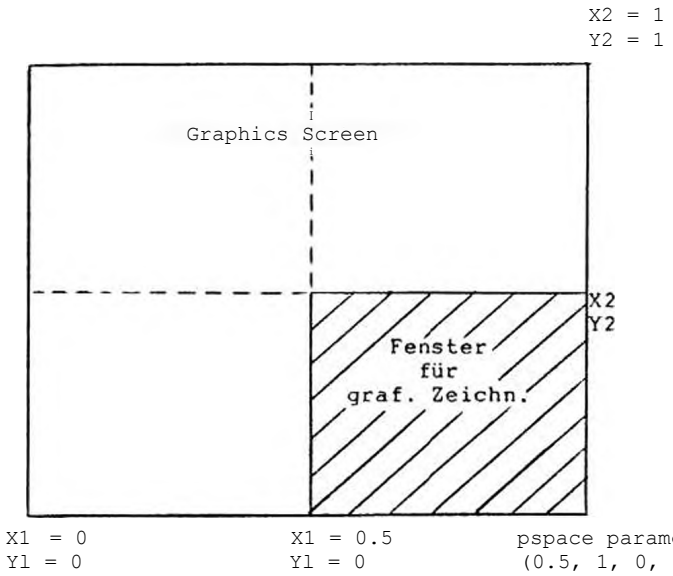
Beispielt map (-100, 1000, 0, 1000)

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

3.1.6. pspace (XI, X2, Yl, Y2) 'pS'

Uber 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: XI = 0, X2 « 1, Y1 =0, Y2 * 1. Diese Werte dürfen im Bereich von 0 bis 1 liegen. XI, 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 nich 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 übeir 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
Zeichnung 2

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 commands for line drawing

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

The plot command moves the "graphical pen" in a straight line from its current X,Y position to a new X,Y position. The X,Y points refer to the mathematical coordinate system (defined with the map command). The drawing mode of the line is specified by the IPEN parameter.

Parameters: X, Y - New Position (mathematical coordinates)
IPEN - Pen Control

0	- Move without drawing	(Pen up)
1	- Draws a line	(Pen down)
2	- Deletes a line	(Eraser)
3	- Inverts a line	

Each is drawn starting from the old position to the new X,Y position.

The parameter IPEN = 3 inverts all points on a line. Clear pixels will be set and set pixels will be cleared. If the command is executed a second time, the original line will be restored.

The type of line that is drawn or erased is specified by the line type command (lintyp).

Note: A line drawn from point A to point B can only be deleted (erased) in the same direction (A to B).

When you delete in the opposite direction from which it was drawn (B to A), the interpolated points may not match exactly the same. This means that some points may not be deleted.

Example for drawing a triangle:

```
10 init
20 map      (0,100,0,100)     define the mathematical
                              coordinate system
30 display  (1)
40 plot     (20,30,0)         set the pen to its
                              starting position
50 plot     (80,30,1)         draw 1 line
60 plot     (50,70,1)         draw 1 line
70 plot     (20,30,1)         draw 1 line
```

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.

Example of positioning a symbol:

```
10 init
20 map (0,100,0,100)
30 display (1)
40 input "X, Y-Coordinates"; 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'

The iplot works like the plot command except for how the X,Y coordinates are interpreted. With plot, the coordinates are interpreted as mathematical coordinates in the system defined by the map command. With iplot, the coordinates are interpreted as physical screen coordinates that directly correspond to the pixels of the hardware screen.

The X,Y parameters therefore lie in the range of:

0 <= X, Y <= 511 (for Version B: 0 <= Y <= 255)

The iplot command achieves higher drawing speed by bypassing the rules defined by the map and pspace commands. Otherwise, it is like the plot command (e.g. 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).

For a better understanding, this table shows the differences between the four plot commands:

Addressing	Math Coordinates	Physical Coordinates
Absolute	plot	iplot
Relative	dplot	idplot

Figure: The Four Plot Commands

3.2.5. lintyp (ITYP) 'liN'

The lintyp (line type) command sets the style of the line that will be drawn by the plot command.

Parameter: ITYP		Line
0	-	Solid Line
1	-	Dotted Line
2	-	Dashed Line
3	-	Dash-Dot Line

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 werdeh muß.

3.2.6. icrcl (R,CPEN) 'iC'

With icrcl (circle), a circle can be drawn on the graphics screen with a radius of R. The center of the circle is the current character position. The radius value R always relates to the physical coordinate system.

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 mj^t Software durchgeführt wird. Lintyp muß für icrcl auf "0" gesetzt werden.

Parameter:		
	R	Radius
	IPEN	Pen Control
	1	draw
	2	delete

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

Der BASIC-Befehl arcus gestattet das Darstellen von Kreisen, Ellypsen und Sektoren. Mit RX wird der Radius in X-Richtung angegeben. Die Werte beziehen sich auf das mathematische Koordingatensystem. 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 Ellypse 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 icrcl

Mit dem folgenden Beispielprogramm kann der Befehl arcus untersucht werden:

```
 10   gr=6.284/360
 20   ini t
 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 4 gr : we = we * gr
100   plot ( mx,my,0)
110   arcus (rx,ry,wa,we,1)
120   goto 40
```

3.3. BASIC commands for text drawing

The graphics board has its own character generator and can draw text in horizontal and vertical directions. In addition, the height and width of the characters can be adjusted. The following section describes the BASIC commands for drawing text.

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) (Carrige 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 muG vorher gelöscht werden. Dies kann mit einem dem zu löschenden String indentischen, 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.

Example for plotting strings:

```
10    init
20    iplot (20, 350, 0)                    position
30    xs = 9 : ys = 8                       character size
40    a$ = "Commodore" : gosub 500
50    iplot (150, 200, 0)
60    xs = 6 : ys = 6
70    a$ = "Graphics" : gosub 500
100   end
500   chrsiz (xs, ys)                       Set character size
510   chplot (a$,1) : return                Plot string
```

3.3.2. chrori (IDIR) 'chrO'

The direction and style used to draw text is defined by chrori (Character-Orientation). Writing is possible in both horizontal and vertical directions as well as normal and italic styles.

Parameter:	IDIR	Direction	Style
	0	- horizontal	, normal
	1	- horizontal	, italic
	2	- vertical	, normal
	3	- vertical	, italic

3.3.3. chrsiz (XS, YS) 'chrS'

With chrsiz (character size), the size of the characters can be chosen in the X and Y directions independently. XS and YS can be chosen in the range of 1 to 15. This results in 225 different character sizes. Each character is generated from a 6x8 dot matrix. XS and YS are scaling factors. The size show will be (6*XS)x(8*YS). The 4x4 and 5x8 blocks described in chplot can be increased at the same scale.

3.4. BASIC commands for special functions

3.4.1. hcopy 'hC'

The hcopy command (hardcopy) creates a copy of the screen and prints it on a Commodore 8023P dot matrix printer. The screen is scanned pixel by pixel and then printed as such.

The hcopy command internally uses file numbers 124 through 127. Therefore, no other programs may use these file numbers at the same time.

If a printer other than the 8023P is to be used, then new routines must be written to support it. However, this can be doing by using the BASIC command 'tstb' supported by this software.

3.4.2. tstb (A) 'tS'

With tstb (test byte), the graphic screen is scanned byte by byte.

The screen is divided into 512 lines (256 lines for Version B) and 512 pixels per line. The 512 pixels are organized into 64 bytes (8 bits = 1 byte, 1 bit = 1 pixel). The leftmost pixel in the byte has the smallest value (Bit 0) while the rightmost pixels has the highest value (Bit 7). See Figure 5.

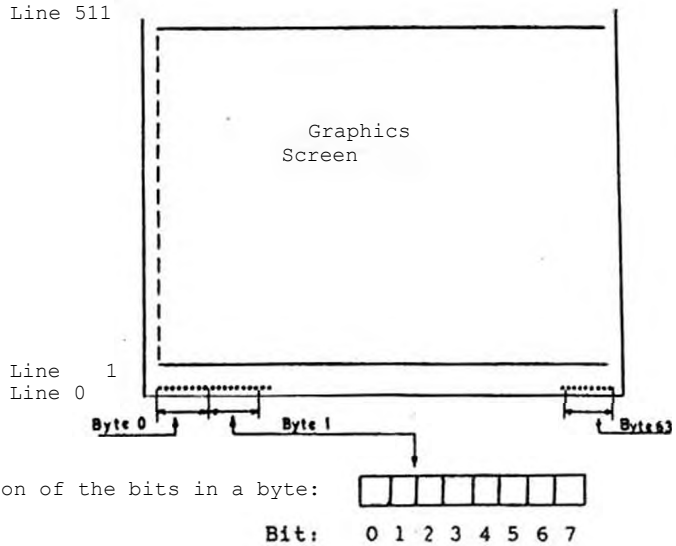


Figure 5: Organization of the Pixels

Bit Assignments:	$0 = 2^0 =$	1	(Pixel 1)
	$1 = 2^1 =$	2	(Pixel 2)
	$2 = 2^2 =$	4	(Pixel 3)
	$3 = 2^3 =$	8	(Pixel 4)
	$4 = 2^4 =$	16	(Pixel 5)
	$5 = 2^5 =$	32	(Pixel 6)
	$6 = 2^6 =$	64	(Pixel 7)
	$7 = 2^7 =$	128	(Pixel 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)                          position
130 tstb (a)                               read byte
140 print #4,a                             print byte
150 next y
160 print #4, chr$(13)                     next printer line
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'

The cursor command displays a cross at the current cursor position.

If the cursor command is called a second time and the cursor position has not changed, the cross is cleared. The actual drawing is not changed. The cross is always drawn by inverting the pixels under it (IPEN 3). The cursor command is used in the graphics demo program listed in section 4.

3.5. List of all BASIC graphics commands

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'
iplot	(X,Y,IPEN)	'iP'
idplot	(DX,DY,IPEN)	'iD'
lintyp	(ITYP)	'liN'
icrcl	(R,IPEN)	'iC'
arcus	(RX,RY,WA,WE,IPEN)	'aR'
chplot	(AS,IPEN)	'chP'
chrori	(IDIR)	'chrO'
chrsiz	(Xs,Ys)	'chrS'
hcopy		'hC'
tstb	(A)	'tS'
tstp	(P)	
cursor		'cU'

4. Grafik Demo Programm

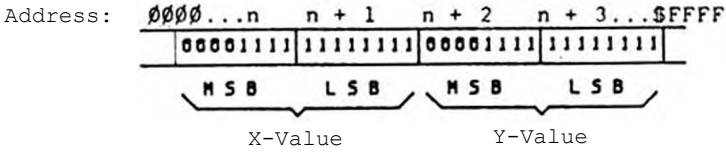
```
10 rem ***************************************************
20 rem ***        graphics demo program                ***
30 rem ***                                             ***
40 rem *** zeichnen unter benutzung der 10er zahlen-   ***
50 rem *** tastatur zur Cursor - Steuerung             ***
60 rem ***                                             ***
70 ran ***************************************************
80 rem
100 rem *** graphics initialization and command table
110 rem
120 print"" :sys40960:print" Set switch to middle position"
200 xs=3 :ys=3
210 iplot(0,485,0)
220 c$="COMMAND LIST": gosubl000
230 rem
240 xs=2:ys=2
250 c$="z - Zeichnen":gosubl000
260 c$="l - Loeschen"rgosubl000
270 c$="p - Positionieren":gosubl000
280 c$="e - Einzelschritt”: gosubl000
290 c$="d - Dauer  ":gosubl000
300 c$="a - Pos. Za ehl er an": gosubl000
310 c$="u - Pos. " aus":gosubl000
320 c$="c - clear Bild":gosubl000
330 c$="x - zurueck ins basic":gosubl000
335 rem
340 iplot(0,0,0)
350 c$=" x-Pos. :   y-Pos.:": gosubl000
360 rem
400 rem *** variable definieren und Cursor auf anfangs-
410 rem     positicn setzen
420 fori=lto4:cl$=cl$+chr$(10):next
430 display (1) :rem screen to graphics mode
440 xp=100 :yp=100: iplot(xp,yp, 0): Cursor
450 rem
500 rem *** poll keyboard
510 rem
520 geta8:ifa$<>""then a=val(aS):goto 540
530 ifc=0 then goto 520
540 ifa<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 *** show new cursor position, if z=1
710 rem
720 x$=str$(xp):y$=str$(yp)
730 iplot(80,0,0):chplot(cl$,2):iplot(80,0,0)
740 chplot(x$,1)
750 iplot(260,0,0):chplot(cl$,2):iplot(260,0,0)
760 chplot(y$,1)
770 iplot(xp,yp,0)
780 goto 500
790 rem
800 rem *** assign   x,y values for cursor direction
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 *** set command status
905 rem
910 if a$="d" then c=1
920 if a$="e" then c=0
930 if a$="p" then p=0
940 if a$="z" then p=1
950 if a$="1" then p=2
960 if a$="a" then z=1
970 if a$="u" then z=0
980 if a$="x" then : display(0):end:rem switch to cbm screen
985 rem                                 and end the program
990 goto500
995 rem
1000 rem *** char.-plot subroutine
1005 rem
1010 chrsiz(xs,ys)
1020 c$=c$+chr$(13)
1030 chplot(cS, 1)
1040 return
```

5. Graphics programming in machine language

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 hi« 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).

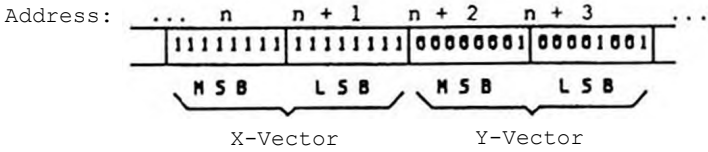
Jeder Koordinatenwert wird als vorzeichenlose ganze Zahl mit 12 Nutzbits dargestellt. Die nicht benutzten Bits stehen immer a\if 0. Die Anordnung von MSB und LSB dieser Zahl entspricht dem in BASIC verwendeten INTEGER-Zahlenformat.



Representation of Coordinates (4095,4095)

This storage format is used for passing absolute coordinates to the appropriate subroutines.

The relative representation of vectors uses all 16 bits of an INTEGER number format, with bit 15 containing the size.



Representation of Relative Vectors (-1,+265)

Even when using relative coordinate vectors, the usable windows cannot be exceeded.

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 Maschinenunterprogrammc mit ihren Einsprungadressen, Argumenten und vergleichbaren BASIC-Befehlen.

$AED0 Initialization (INIT)

Action: Wie der INIT-Befehl in BASIC, jedoch ohne Veränderung eventueller MAP- oder PSPACE-Parameter.

Arguments: keine

$AED3 Set Character Size (CHRSIZ)

Action: Setzen der Größe für Schriftzeichen in x- und y-Richtung.

Arguments: .a s Bit 0...3 - Zeichengröße in x-Richtung
Bit 4...7 - Zeichengröße in y-Richtung

$AED6 Set Character Orientation (CHRORI)

Action: Sets the direction (horizontal or vertical) and the character type (normal or italic).

Arguments: .a wie in CHRORI

$AED9 Set Line Type (LINTYP)

Action: Für alle folgenden Linienoperationen wird der Linientyp festgelegt.

Arguments: .a Linientyp wie in LINTYP

$AEDC Internal Display Mode (DISPLAY)

Action: The built-in screen is connected to either the CPU or the graphics system.

Arguments: .x , Werte wie DISPLAY in BASIC

$AEDF Command Transfer (-)

Action: Der Inhalt des Akkumulators .a wird in das Commandregister des Graphikprozessor geschrieben. Sollte der Prozessor noch mit der Ausführung des vorigen Kommandos beschäftigt sein, so wird zuerst gewartet, bis der Prozessor frei ist.

Arguments: .& - Kommandocode

$AEE2 Wait until the graphics processor is free

Action: This subroutine waits until the graphics processor finishes the last command. It should always be called before any register

on the graphics processor is modified, to ensure the execution of the last or current command is not disrupted. The registers a, x, and y are not changed.

Arguments: none

$AEE5 Character Plot

Das Textzeichen, dessen Commodore-ASCII-Code im Akkummulator .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

$AEE8 String Plot (CHRPLT)

Zeichnet einen String auf dem Bildschirm

Argumente: .y - Länge des Strings
.x - Pen code
Zeroe Page: $1F g20 c' Startadresse 'des String

$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

$AEEE DPLOT (IDPLOT)

Draws a relative vector. Functions similar to 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.

Arguments: none

$AEF4 Mode

Works like the MODE command in BASIC.

Arguments: .x - Mode

$AEF7 HCOPY (HCOPY)

Works like the HCOPY command for BASIC.

Arguments: none

$AEFA circle (ICRCL)

Works like the ICRCL command for BASIC.

Arguments: .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 * 7T)

Argumente: . x - Pen code

$0388	Radius x	msb
S0389	Radius x	lsb
S038A	Radius y	msb
S038B	Radius y	lsb
S038C	Startwinkel	msb
S038D	Startwinkel	lsb
S03QE	Startwinkel	msb
S038F	Startwinkel	lsb

5.1. Memory Map:

$033A - $0391	RAM area for graphics routines
$A000 - $AEFF	ROM area for graphics routines
$AF00	Mode Register, write only
	Bit 0 - Hardcopy bit, 0 for hcopy-mode 1 - Operating page select (only 512 * 256 version - Copies) 3 - Read-modify-write Bit, active=1 4 - display switch bit, 1=graphic 5 - display page select (only 512 * 256 version)
$0372	Copy of the Mode Register (for read)
$AF10	Bit 0: Light Pen Contact, Read only
$AF30	Hardcopy register, read only
$AF70 - $AF7F	Graphics Processor

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

EF9365
EF9366

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 :

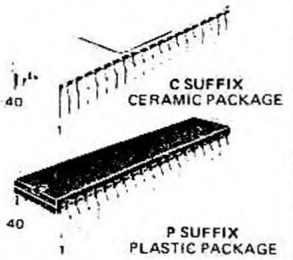
- Selectable resolutions in black and white or color :
  EF9365 : 512 x 512 (interlaced scan)
  256 x 256, 128 x 128, 64 x 64 (non interlaced scan)
  EF9366 : 512 x 256 (non interlaced scan)
- High speed vector plot well suited to animation (up to 1 500 000 dots/s. and an average value of 900 000 dots/s.) - 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.

MOS

(N - CHANNEL, SILICON - GATE)

GRAPHIC DISPLAY PROCESSOR (GDP)

CASE CB-182



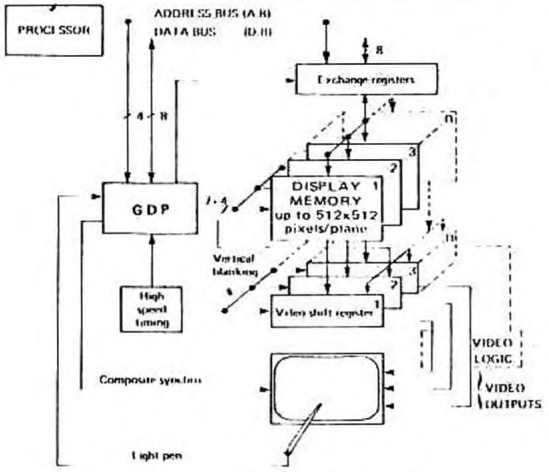
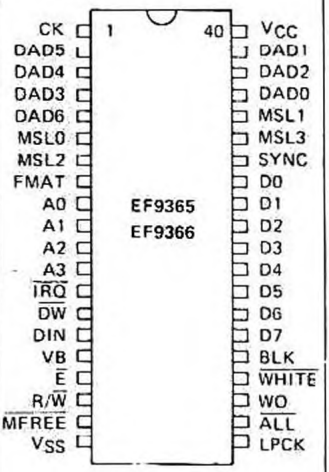


FIGURE 1 – TYPICAL APPLICATION

PIN ASSIGNMENT



Developped in conjunction with the Ecole Normale Superieure.

NT8132R1 A 1/27

THOMSON-EFCIS
Sales headquarters
45, av. de l'Europe 78140 VELIZY FRANCE
Tel. : (3) 946 97 19 / Telex 698 866 F

THOMSON-CSF

REGISTER DESCRIPTION

X AND Y REGISTERS (Addresses : 8_{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.

DELTAX AND DELTAY REGISTERS (Addresses : 5_{16}, 7_{16}).

The DELTAX and DELTAY 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.

MSB	P	Q	LSB

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_H)

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 recopy.

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_H and D_H)

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 LPCK 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.

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, $\overline{DW}$, $\overline{MFREE}$ and $\overline{IRQ}$ outputs.

These harwired 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 $\overline{DW}$ 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

→ DELTAX sign } 0 if positive
→ DELTAY sign } 1 if negative

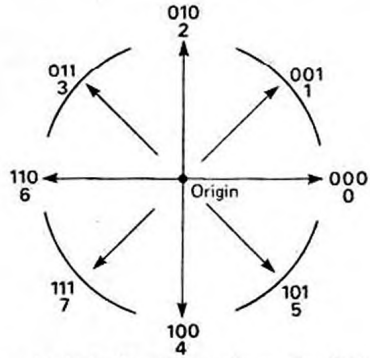
- Commands which allow ignoring the DELTAX or DELTAY registers by considering them as of zero value.

0	0	0	1	0	X	X	0

0 0 DELTAY ignored, DELTAX > 0
0 1 DELTAX ignored, DELTAY > 0
1 0 DELTAX ignored, DELTAY < 0
1 1 DELTAY ignored, DELTAX < 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 : $\begin{cases} X & 47_{10} \\ Y & 75_{10} \end{cases}$

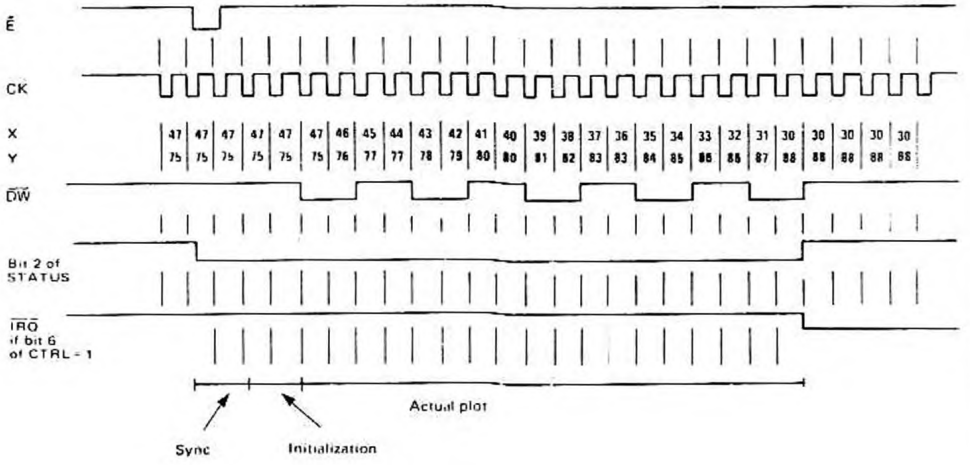
Projections: $\begin{cases} DELTAX = 17_{10} \\ DELTAY = 13_{10} \end{cases}$

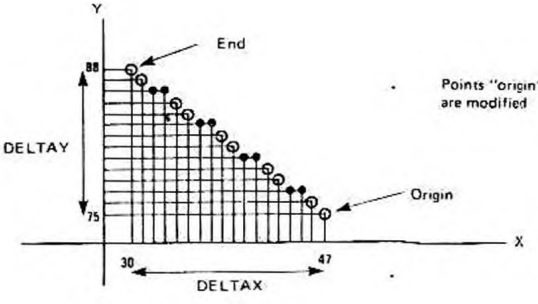
CMD = 13_{16} — Corresponding to
- Basic command,
- DELTAX < 0
- DELTAY > 0

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
o 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.

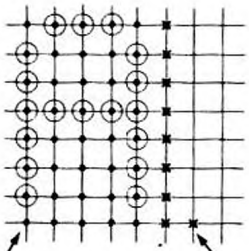
CHARACTER AND SYMBOL GENERATOR

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 $\overline{DW}$ 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.



Origin End

• Unchanged
⊙ Altered dots } if CMD = 41_{16} (in the ROM standard version)

× Computed dots, not defined into the ROM (not modifiable).

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

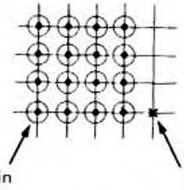
The character size becomes 5P x 8Q. Upon completion of the writing process, X is incremented by 6P. The CK clock cycle count required is 6P x 8Q.

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 0_{16}.

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 20_{16} to $7F_{16}$, and the 97th matrix to $0A_{16}$. 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 ($0B_{16}$) 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.



⊙ Modified dots
× Computed dot not defined in ROM (not modifiable)

Origin End

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 $0A_{16}$. 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	DELTAX, DELTAY	CSIZE, tilting
$\overline{DW}$ modulation	Type of line	Character code

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 08_{16} or 09_{16} code into the CMD register.

Here, the frame origin is counted starting with the VB falling edge. With code 08_{16}, the $\overline{WHITE}$ 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 09_{16}, the $\overline{WHITE}$ 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. Dots 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.

The rising edge first received (LPCK or VB) sets bit 0 in STATUS register high. An interrupt is initiated if bit 4 in CTRL1 is high.

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).

SCREEN BLANKING COMMANDS

Three commands (04_{16}, 05_{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.

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 (MFREE OUTPUT)

On writing code $0F_{16}$ into the CMD register, the $\overline{MFREE}$ 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 $\overline{E}$ 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 $\overline{MFREE}$ 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 $\overline{IRQ}$ is forced low.

A read operation in the STATUS register resets its 4 MSBs low, after input $\overline{E}$ 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 $\overline{E}$ 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 $\overline{IRQ}$.

TABLE 1 – REGISTER ADDRESS

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 buffers D0-D7 force a high state on the data bus.

TABLE 2 – COMMAND REGISTER

b7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b6	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1
b5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
b4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
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 selection		SPACE	0	@	P	`	p								
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 CTRL1 : 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 ($\overline{WHITE}$ 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	▒								

Vector generation (for b2, b1, b0 see small vector definition)

Special direction vectors (for b2, b1, b0 see small vector definition)

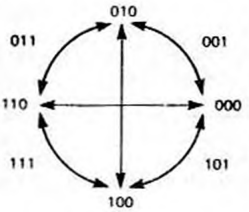
SMALL VECTOR DEFINITION :

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

Dimension

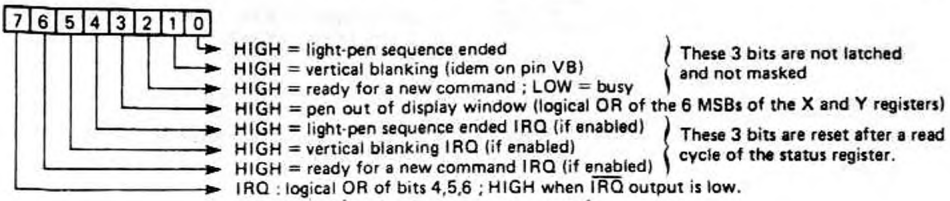
ΔX or ΔY	Vector length
0 0	0 step
0 1	1 step
1 0	2 steps
1 1	3 steps

Direction

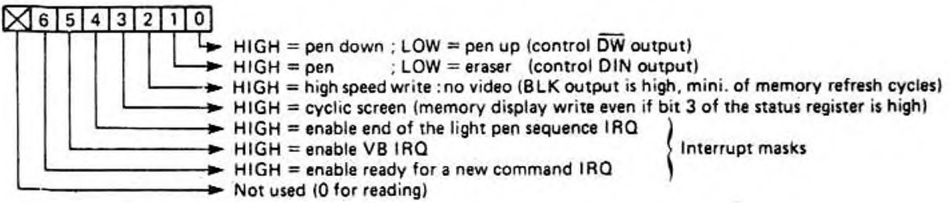


OTHER REGISTERS

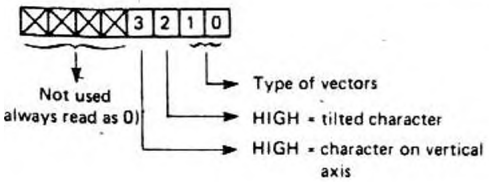
STATUS REGISTER (Read only)



CONTROL REGISTER 1 (Read/Write)

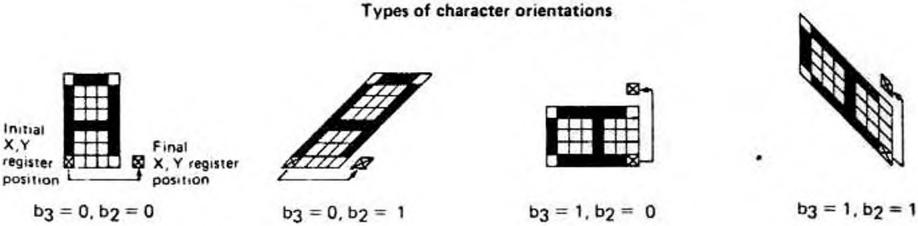


CONTROL REGISTER 2 (Read/Write)

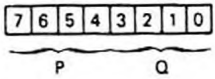


b1	b0	Type of vectors	
0	0	——— continuous	
0	1	------- dotted	2 dots on, 2 dots off
1	0	— — — dashed	4 dots on, 4 dots off
1	1	—·—· dotted-dashed	10 dots on, 2 dots off, 2 dots on, 2 dots off.

Types of character orientations



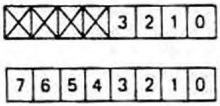
C-SIZE REGISTER (Read/Write)



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)

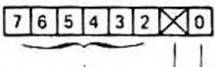


MSBs

LSBs

The 4 leftmost MSBs are always 0.

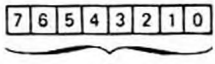
XLP and YLP REGISTERS



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

Appendix C: Light Pen Connection

The graphics board provides a connector (J8) for a light pen. It has the following signals:

+ 5 V, Gnd : to power the light pen

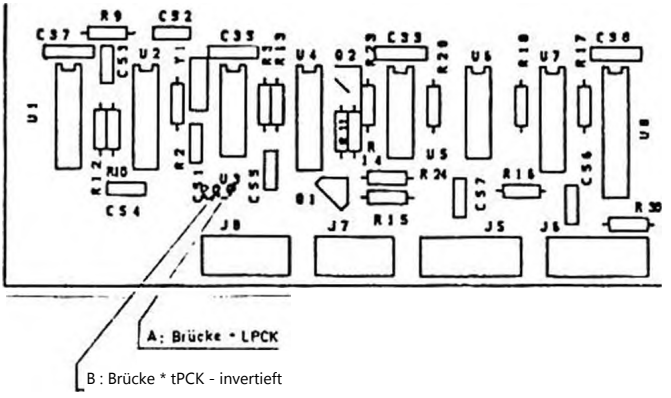
Pen Contact : to query whether the pen is placed on the screen

Pen Input : ist auf der Platine mit dem LPCK Eingang des Grafik-Prozessors zu verbinden

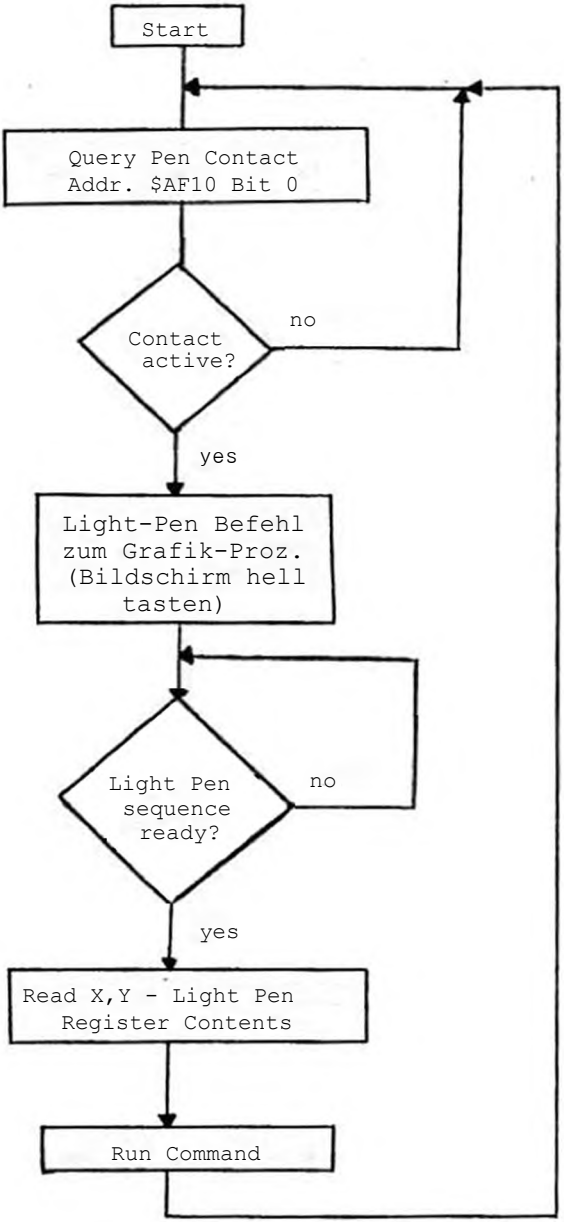
Pinout : see Appendix D

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 Draht.bcücke (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:


Start
Query Pen Contact
Addr. $AF10 Bit 0
Contact
active?
no
yes
Light-Pen Befehl
zum Grafik-Proz.
(Bildschirm hell
tasten)
Light Pen
sequence
ready?
no
yes
Read X,Y - Light Pen
Register Contents
Run Command

Appendix D: Pin Assignments

Power

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

Video for Internal CBM Monitor

J5 Video Input

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

J6 Video Output

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

Video for External Monitor Connector

J7 (Connector on the graphics board)

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

6-pin A/V Socket

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

Light Pen Port
J8

Pin	Signal
1	+ 5 V
2	NC
3	Pen Contact
4	GND
5	Pen Input
6	GND

Switch for Screen Mode
J9

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

commodore

Connection to the CBM Expansion Bus

The graphics board mounts to the CBM on the headers at J3 and J4 All signals from the headers are passed through and are available as two rows of unpopulated holes on the top of the graphics board. This table lists all of the expansion bus signals and which are used by the graphics board.

Pin	J3 Signal from CBM	Graphics	J4 Signal from CBM	Graphics
1	GND	x	GND	x
2	BA Ø	x	BD Ø	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	$\overline{\text{SEL 4}}$	
13	BA 11	x	$\overline{\text{SEL 5}}$	
14	BA 12		$\overline{\text{SEL 6}}$	
15	BA 13		$\overline{\text{SEL 7}}$	
16	BA 14		$\overline{\text{SEL 8}}$	
17	BA 15		$\overline{\text{SEL 9}}$	x 1)
18	SYNC		$\overline{\text{SEL A}}$	x
19	$\overline{\text{IRQ}}$		$\overline{\text{SEL B}}$	
20	DIAG		$\overline{\text{NO ROM}}$	
21	CØ2	x	$\overline{\text{PEN STR}}$	
22	BR/W	x	$\overline{\text{RES}}$	
23	$\overline{\text{BR/W}}$		RDY	
24	NC		$\overline{\text{NMI}}$	
25	GND	x	GND	x

1) Leads only to Jumper M on the graphics board

Translated to English by Mike Naberezny in February 2012

commodore

Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Fleschgasse 2
A-1130 Wien

Änderungen vorbehalten