Combo AG
Tugginerweg 3
CH-4500 Solothurn
TEL. 065 232686

Graphic Booster 128 (tm)
(c) T. Giger 1986

All rights reserved.
(c)1986 T. Giger Solothurn Switzerland
COMBO AG represents all rights. It is forbidden to publish or copy the Software and Hardware.Place of jurisdiction is Solothurn Switzerland.

Dear customer of GRAPHIC BOOSTER,
with purchasing this product, you are to be promoted as a professional CAD User. The possibilitys that are given you with this product are unique in the sector of computergraphics. The system is that userfriendly and has such a number of professional routines that the programming of your 1st-quality software is an enjoyment. If you compare other computersystems with the same features, you'll see that other graphiccards with software (if available though) can be ten times the expense, with an also far too expensive hardware. Since 1986, many programmers sent us software, that is made with GRAPHIC BOOSTER. That made us collect all this software for a highresolution-software-pool. Every program that uses GRAPHIC BOOSTER's high resolution is awesome! For this reason, we decided to found a software-pool (like Public Domain) which works like this: You send us your GRAPHIC BOOSTER related software and we'll send you a disc full (around 20 programs) with other highresolution-software. There is no restriction in size nor in type of program. Here are some motivations: mathematical curves, single- or doublescreen programs, CAD, games, painting, symbols, digitized images, tools or subroutines, windows in textmode as described under "GRAPHIC BOOSTER and the textmode" ...and so on.

GRAPHIC BOOSTER (c) 1986/87/88/89 Combo AG
=====================================
GRAPHIC BOOSTER is a developers package with all rights by Combo AG, Solothurn, Switzerland.
Please follow the TradeMarks of GRAPHIC BOOSTER 128 and this manual.
The GRAPHIC BOOSTER is a product of Combo AG and will be exclusively distributed by Combo AG.
All rights reserved, also the photo-mechanical reproduction and saving in electronic media.
Commercial use is prohibited. GRAPHIC BOOSTER should not be used for education, nor demonstrated in the public.
GRAPHIC BOOSTER should not be published nor reviewed as a whole or in fragments.
By all means of spreading, publishing and review, it must be permitted in written form by Combo AG.
Especially the graphic-driver of GRAPHIC BOOSTER for resolutions higher than 640*200 and also the interlace-modus for the highresolution are worldwide copyrighted 1986/87/88/89 by Combo AG!
Adress: Combo AG, GRAPHIC BOOSTER Pool, Tugginerweg 3, CH-4500 Solothurn

GRAPHIC BOOSTER Commands:

You are programming in BASIC V7.0 so orientate yourself to the C-128 Manual.
new commands:

GRAPHIC1,1	switches graphic on and clears screen. Interlaced mode 640*720 or 720*700 pixel
GRAPHIC1,0	switches graphic on without screenclear
GRAPHIC2,1	switches graphic on and clears screen. normal mode 640*360 or 720*360 pixel
GRAPHIC2,0	switches graphic on without screenclear
GRAPHIC0	back to textmode (only with one monitor and in 80-chars mode)

GRAPHIC3,1	switches graphic on and clears screen. 720*400 or 640*400 (as choosen in Startprogram)
GRAPHIC3,0	switches graphic on without screenclear
GRAPHIC4	like GRAPHIC2, but 2nd screen
GRAPHIC4,2	switches on GRAPHIC4 and copies GRAPHIC2 into GRAPHIC4
GRAPHIC4,3	switches on GRAPHIC4 but draws in GRAPHIC2
GRAPHIC4,4	switches on GRAPHIC4 and copies GRAPHIC4 into GRAPHIC2
GRAPHIC4,5	switches on GRAPHIC4 and clears GRAPHIC2

With this commands you're able to draw hided and to pop up the whole screen.
If you use the 1750 REU, you are able to load hided from there.
With GRAPHIC2 the second value works like on GRAPHIC4.

On multicolor mode, only GRAPHIC1 is active.

TEXT(0-1),x,y,(writeyourtextinhere),(inverse) is like the charcommand from BASIC V7.0 but writes only on the textscreen

OLD	gets back a basicprogram, deleted with NEW
OFF	switches GRAPHIC BOOSTER off

80-columns-mode graphic-commands:
DRAW, BOX, CIRCLE, SSHAPE, GSHAPE (syntax like in BASIC V7.0 in 40-columns-mode)

40-columns-mode graphic-commands:
VGRAPHIC, VDRAW, VBOX, VCIRCLE, VPAINT, VCHAR, VSSHAPE, VGSHAPE, VCOLOR, VWIDTH, VSCNCLR, VSPRITEDEF (syntax like in Basic V7.0)

These commands can be mixed optionally. The same SHAPEs can be used on both screens. Sprites can be defined while the program runs (VSPRITEDEF) and can be transfered into the SHAPEs of the 80-columns-screen! Sprites together with the BYTE and FBYTE command lets you use both screens as one! If you want to move pixelwise also in x-direction, do like this: Get the matrix into the stringvariable with VSSHAPE or with SPRSAV, then move with GSHAPE 7 times (one pixel moved at a time) on the bottom save 80-columns and swap with FBYTE. Such tools always are useful and shouldn't be absent in any programming session. Like mentioned above build these tools like that: the first bunch of REMs explain the INPUT and OUTPUT values, then follows the function and confirm with RETURN. At a later MERGE the numbering don't must be changed anymore before loading, if the tools numbered high from the start and with the same numbering.

<u>GRAPHIC BOOSTER and the textmode</u>

(this isn't understandable for me, also, although I'm german! Original manual: site 3, right block, translated it anyway... maybe you get it?)
Load a resolution, twocolored and 640 width Height isn't relevant.
The graphic-commands are cool to use on textmode. BYTE, FBYTE and SSHAPE, GSHAPE give you possibilitys, that you dreamt of, before. Real filling of windows and screenparts, characterpokes, atributpokes, plot/noplot of single pixel (underline, blinking, reverse, color and intensity), manipulation of charset like change as whole, swap single characters. No problem, since GRAPHIC BOOSTER manages full 64kB!
In medias res: S/GSHAPE takes 3 textlines 80chars width OR 251/witdh=height OR 15width 15height.
The x-coordinate calculates like this: x * 8 (coordinate-system of SHAPE is 640 width)
Some examples:
POKE13480,PEEK(2606) equals the start of textscreen
if you want to change or copy the atribute of the char, write:
POKE13480,PEEK(2607) LOCATE0,0 equals the atribute left, top.
Define following parameters before start:
POKE2606,0:POKE2607,16:bank15:SYS52684,16,20:GRAPHIC4:GRAPHIC0

for example: text on the screen
POKE13480,0:PRINT"CLR HOME":FORI=0TO4:PRINT"GRAPHIC BOOSTER";:NEXT
FBYTE1,0,1,80:DRAW,0,0TO0,20
for example: underline from text on/off
POKE13480,PEEK(2607):WIDTH20,10:BYTE2,32:DRAW,10,10:SLEEP2:DRAW,10,10

Systemenhancements:
The 1750 REU is implemented for easyer usage. The commands STASH, FETCH, SWAP also work in fastmode. Bank1 is accessable from REU now. Simply write Bank1. Negative values doesn't give a ?ILLEGAL QUANTITY ERROR anymore. The char-command now works fine. It's possible to use CHAR also on 80-columns-screen.

The transfer of programs of GRAPHIC BOOSTER V1:
List the listing on screen; 15lines-wise and accept with RETURN. The COLOR command has to be corrected. Save the program and check it.

Printing:
With EPSON or compatible printers
Syntax:
EPSON(mode),(devicenumber),(number of lines * 8)

mode 0-13	Devicenumber 4-7 Number of lines 1-255
mode 0-6	printing over serial bus
mode 7-13	printing over centronics (on Userport with special cable) mode0=7 1=8 ...
mode0	480ppi (ESC"K")
mode1	960ppi (ESC"L")
mode2	960ppi (ESC"Y") fast
mode3	1920ppi (ESC"Z")
mode4	640ppi
mode5	576ppi
mode6	720ppi

Example: EPSON4,4,65

IBM(mode),(devicenumber),(number of lines * 8)

mode 0-8	
mode 0-3	serial
mode 4-8	parallel centronics 4=0 5=1 ...
mode0	480ppi
mode1	960ppi
mode2	960ppi fast
mode3	1920ppi

Secondary adress is 16248. Default is 4

Switch secondary adress to 1:	SCREENB:POKE 16248,1
Print 640ppi with MPS1000:	SCREENB:POKE14814,32:POKE14314,7
	IBM3,4,65
Reset:	SCREENB:POKE14814,76:POKE14314,2
Printing Mode2 default is:	GRAPHIC2 SCREENB:POKE14782,127
	GRAPHIC4 SCREENB:POKE14782,255
Linefeed off:	SCREENB:POKE12250,13
on:	SCREENB:POKE12250,10
switch to 80-columns (VDC):	SCREENB
switch to 40-columns (VIC):	SCREENV
activate mouse1531 port2:	MOUSE(0-3)
	MOUSE0 on

		MOUSE1,20,30	Mouse on x=20 Y=31
		MOUSE2,a,b	peeks mouseposition and saves in a,b
		MOUSE3	off
Mousebuttons:	left:	t=JOY(2)	t=128
	right:	t=JOY(2)	t=1
	both:	t=JOY(2)	t=129
	no:	t=JOY(2)	t=0

move 40-char bitmap to CharRAM and viceversa: TRANS(0-3)
Charsets on 40-char VIC can be build or whole
pictures can be set as Charset.
CharRAM is at $E000-$EFFF. (see CHAR)

TRANS0	transfers $2000-$2FFF to $E00-$EFFF (1st half of VIC)
TRANS1	transfers $E000-$EFFF to $2000-$2FFF (1st half of VIC)
TRANS2	transfers $E000-$EFFF to $3000-$3FFF (2nd half of VIC)
TRANS3	transfers $3000-$3FFF to $E000-$EFFF (2nd half of VIC)

Example:
```
BLOAD"charset",U8,ONB0,PDEC("E000")
TRANS1:VGRAPHIC1:VCIRCLE3,3,3:TRANS0:CHAR,30,30,1,1,1,"@"
```
with all commands the charset can be changed (see VGRAPHIC commands)

CHAR writes text in a GRAPHIC BOOSTER graphic.
CHAR(set),X,Y,(width),(height),(mode),(string or number)

set0:	set	
set1:	delete	
width/height:	zoomfactor 1-255	
mode0:	Capitals/graphic	(from charROM)
mode1:	lower/Capitals	(from charROM)
mode2:	Capitals/graphic	(from RAM)
mode3:	lower/Capitals	(from RAM)
mode4:	pokecodes	(from charROM)
mode5:	pokecodes	(from RAM)

Example:
```
CHAR,40,40,1,2,0,"GRAPHIC BOOSTER 128"
CHAR,40,60,1,2,1,"Graphic Booster 128"
```
mode4 and mode5 is a comfortable way to transfer charROM and RAM
```
   CHAR,40,80,1,2,4,0,1,2,3,4,5,6,7,8,9 etc.
or FORI=0TO20:CHAR,40,120+I,8,2,2,4,122,153,I,250,255:NEXT
or FORI=0TO20:CHAR,40+I,8,100,1,2,4,I:NEXT
```

Instead of strings you can insert values between 0 and 511, which is the pokecode of charROM or RAM.
The zoomfactor also works with own charset.40-columns-graphic can be zoomed easily on 80-columns-screen.
Reversed chars mode0-3 are printed with CTRL-RVSON "(RVS)text"

PAINTx,y,(mode) fill a closed area with or without pattern

x,y:	position in coordinatesystem
mode0:	fill, point set
mode1:	fill, point set, fine pattern
mode2:	fill new, test on point not set
mode3:	delete new, point set
mode4,(0-3),(0-3),(0-3):	tests if not set, fine pattern
mode5,(0-3),(0-3),(0-3):	tests if not set, big pattern
mode6,(0-3),(0-3),(0-3):	tests if set, fine pattern
mode7,(0-3),(0-3),(0-3):	tests if set, big pattern

Mode 0 and 1 works like PAINT on th 40-columns-screen
Mode 2 to 7 fills or deletes with or without pattern. Tests around point is the four quadrants. Filling will only be around these testpoints. Fillspeed is twice the speed of mode0/1. PAINT also changes PATTERN, so you have to change to full linethickness after PAINT with PATTERN0.

Example: CIRCLE,50,50,50:PAINT30,30,4,2,3,2:PATTERN0

PATTERN(mode0-2)	set Pattern
PATTERN0:	no pattern
PATTERN1,(0-3),(0-3),(0-3)	fine pattern (0-3 for different patterns)
PATTERN2,(0-3),(0-3),(0-3)	big pattern

Example:	PATTERN1,0,0,0	fine quadrants
	PATTERN1,1,1,1	fine quadrants
	PATTERN1,2,2,2	bigger quadrants
	PATTERN1,3,3,3	big quadrants

PATTERN is used in all drawing commands like DRAW. Big patterns can be seen after change of linethickness (WIDTH), els the line will be dashed.

RLOAD(bank),(0-2) loads a picture from 1750 REU
(bank0-7 for 1750, bank 0-1 for 1700)
0: graphic1 interlace picture
1: graphic2
2: graphic2

RSAVE(bank),(0-2) save a picture onto the REU
(parameters like RLOAD)

GLOADa$,(devicenumber) loads a picture from disc, device 8-15
Example: GLOAD"picture",8 or A$="picture".GLOADA$,8

GSAVEa$,8 saves a picture on disc 8
(works with GRAPHIC1 and GRAPHIC2 pictures)
Oversize picture2 is saved with GRAPHIC1:GSAVEa$,8 or GRAPHIC4

MERGE(A$),U8 merges a basicprogram with a program in memory
Example: MERGE"tool" or A$="tool":MERGE(A$),U8
With MERGE you can merge your GRAPHIC BOOSTER subroutines and make them executable with RENUMBER

BWINDOW(0-12) calls two different windows, one on the right side (80 pixel width) and one in the middle (320width *174(GRAPHIC1) / *87(GRAPHIC4) height)
Startpoint from left border is 160 or 200 pixel (differs from choosen resolution) and from top border 143 (GRAPHIC1) / 72 (GRAPHIC4) pixel.
BOX,160,143,479,316
these parameters relate to the right window

BWINDOW0	Swap changes data with buffer
BWINDOW1	Fetch gets data from buffer
BWINDOW2	Stash writes data into buffer
BWINDOW3	deletes window (on multicolor the color has to be set separately)

the parameter 4-7 relate to window in the middle on GRAPHIC4

BWINDOW4	Swap
BWINDOW5	Fetch
BWINDOW6	Stash

BWINDOW7	delete

the parameters 8-12 relate to the window in the middle on GRAPHIC1

BWINDOW8	Swap
BWINDOW9	Fetch
BWINDOW10	Stash
BWINDOW11	delete
BWINDOW12	sets color, which is defined in ECOL

(range of adress see BUFFER)

BUFFER(0-4),(0-2)	copys adressrange from 80-columns-screen into a buffer
BUFFER0	relates to atributeRAM in multicolor mode
BUFFER1	relates to the first 16Kbyte in VDC
BUFFER2	relates to the second 16Kbyte in VDC
BUFFER3	relates to the third 16Kbyte in VDC
BUFFER4	relates to the fourth 16Kbyte in VDC

BUFFER0,0	Swap	changes atributeRAM with buffer
BUFFERx,1	Fetch	gets data from buffer and writes them into VDC RAM
BUFFERx,2	Stash	writes data from VDC into buffer

BUFFER 1-4 can be used as alternative to GLOAD/GSAVE. It's also useful in textmode with full 64Kbyte. The buffer is used also with BWINDOW, so the buffer is deleted then.
With SCREENV the buffer is now at $2000-$5FFF (dec. 8192-24575 in bank1). This adressarea can now be saved/loaded from disc/1750 REU).
You may have asked yourself, how to save from bank1 to REU. Thats easy, as GRAPHIC BOOSTER gets the data out of the last defined bank!
Also you may use STASH, FETCH and SWAP in FASTmode with REU (without systemcrash anymore)!
The buffer from the right window is at BANK1:SCREENV (dec. 8192-16383), the buffer from the window in the middle also with BANK1:SCREENV
at adress (dec. 16384-25856).
Example: Save screen with buffer

Special copy and fill commands:
The following commands change the actual MODE. With TMODEa the actual value is saved in variable "a", to get it back MODEa.

PIXEL(mode),w,z		copies pixelwise
	w:	x-offset to drawpoint (LOCATE DRAW)
	z:	y-offset to drawpoint
0	:	Swap exchange
1	:	copy
2	:	copy inverse
3	:	EOR and copy inverse
4	:	copy only EOR
5	:	OR copy
6	:	AND copy

```
Example:    copy area 150,100    200,250 to 0,0
            PIXEL1,-150,-100
            WIDTH50,1
            DRAW,150,100TO150,250
      or    WIDTH1,1
            PIXEL1,-150,-100
            FORI=150TO200:DRAW,I,100TOI,250:NEXT
      or    WIDTH1,1
            PIXEL1,-150,-100
```

```
                    BOX,150,100,200,250
```

With pixelwise copy you can zoom in and out, if you write PIXEL in the FOR-NEXT-routine and change the factor in DRAW. Overlap of target and source give special effects.
If the target is inside the source, you have to follow this instructions:
If startpoint of target is smaller than the startpoint of the source: DRAW [small value to high value] for X and Y
If startpoint of target is bigger than the startpoint of the source: DRAW [highvalue to small value] for X and Y

For SWAP, the first coordinates have to be set with LOCATEx,y because DRAW sets the first pixel twice and therefore changes them twice.
Effect: The first pixel wouldn't be changed. LOCATE10,10:DRAWto10,50 instead of DRAW,10,10to10,50 this has to be used for all following SWAP-commands.

BYTE(0-9)	Copy and Swapcommands bytewise with 640 pixel width these are 80 bytes.
BYTE0,value	sets a byte, value 0-255
BYTE1,value	sets a byte, OR linked
BYTE2,value	sets a byte, EOR
BYTE3,value	sets a byte, AND
BYTE4,value1,value2	only sets byte with value1, if same as value2
BYTE5,value1,value2	only sets byte with value1, if not same as value2
BYTE6,x,y	like PIXEL, copies EOR linked
BYTE7,x,y	OR
BYTE8,x,y	AND
BYTE9,x,y	SWAP swaps ;-)

What phantastic possibylities this command has in the textmode, you only can imagine here!
(check GRAPHIC BOOSTER and the textmode)
for example:

```
GRAPHIC1,1
WIDTH3,8
BYTE0,170
CIRCLE,40,300,20,160
WIDTH1,1
BYTE1,112
BOX,0,0,40,200
BYTE2,20
BOX,0,0,20,100
SLEEP1
BOX,0,0,20,100
WIDTH15,0
BYTE6,30,150
LOCATE0,0:DRAWTO0,100
SLEEP2
LOCATE0,100:DRAWTO0,0
WIDTH10,0
BYTE9,50,300
LOCATE0,0:DRAWTO0,100
SLEEP1
LOCATE0,100:DRAWTO0,0
```

If you want to display or print a picture reverse:

```
WIDTH80,1
BYTE2,255
DRAW,0,0TO0,500
```

FBYTE(mode),x,y,width	fast COPY and SWAP mode 0-7
FBYTE0,x,y,width	COPY with offset x y and width byte
FBYTE1,x,y,width	SWAP
FBYTE2,x,y,width	COPY attribute multicolor to pixelbitmap
FBYTE3,x,y,width	COPY pixelbitmap to attribute
FBYTE4,x,y,width	SWAP attribute with pixel
FBYTE5,x,y,width	SWAP pixel with attribute
FBYTE6,x,y,width	COPY attribute with attribute
FBYTE7,x,y,width	SWAP attribute with attribute

On twocolorbooster only Mode 0 and 1
Why Mode2-5? If you only use blockgraphic, you can use the bitmapmemory as hidden extramemory.
ECOL2,2,2,2 set fore-and backgroundcolor to same value
examples:

FILL0,width,value	fills area with Byte value 0-255
FILL1,width,value	fills area with color from ECOL
TBYTEa	read Byte at actual cursorposition LOCATE20,200:TBYTEA:PRINTA
USER	for own machinelanguage programs, jumps over Vector $1BEB $1BEC (7149 7150) POKE7149,0 POKE7150,12 USER runs program from $0C00

General informations:
=================
The standard commands can be combined with each other. For example with COPY you can copy any area with a circle or a line.
PIXEL, BYTE, FBYTE, FILL work together with DRAW, CIRCLE, BOX ...that means, instead of setting a pixel, one of these specialcommands will be used.
GRAPHIC BOOSTER is programmed to get the most out of it with simple commands.
For beginners: Try to work with draw-commands in Mode0,1 or 2 in the 2-color GRAPHIC BOOSTER
Important: Always define MODE, PATTERN, WIDTH and ECOL at the first lines of your program.
The multicolor commands couldn't be used in the 2-color mode. CTEST:ECOL:RECOL:CHANGE:SCLR
On the end of the directory on disc are some tools, you can use.
With the userprogram "Grafik Tablet" you can load the file "Shuttle" instead of "Körper".
This are the datas for the shuttle. With these, you may extend the graphic tablet: make your own program for zooming, moving or rotating and use the data again.
You also could send this program in or extend the other programs on the disc.
The data of the "Shuttle" only may used together with GRAPHIC BOOSTER, copyright Combo AG.

WIDTHx,y sets the thickness of the line, value 1-255, 0 equals 1
Set the thickness to WIDTH1,1, if a new mode is tried out. If a drawfunction doesn't end, maybe the thickness is set too big. But function will be executed.
example: MODE2:WIDTH16,16:CIRCLE,150,150,100
MODE0:WIDTH16,16:CIRCLE,350,150,100

MODE(0-17)	sets different drawingmodes
0	sets a pixel
1	sets a pixel EOR
2	sets a pixel EOR for CIRCLE
3	Fore-and backgroundcolor and pixel
4	fore-and backgroundcolor without pixel
5	only foregroundcolor and pixel
6	only backgroundcolor and pixel

7	only foregroundcolor without pixel	
8	only backgroundcolor without pixel	
9	fore-and backgroundcolor	Colorblock
10	only foregroundcolor	
11	only backgroundcolor	
12	CHANGE only foregroundcolor	
13	CHANGE only backgroundcolor	
14	CHANGE only foregroundcolor	Colorblock
15	CHANGE only backgroundcolor	Colorblock
16	CHANGE fore-and backgroundcolor	
17	CHANGE fore-and backgroundcolor	Colorblock

On the 2color-mode only Mode 0,1 and 2 are active.
Colorblock means, that only the colorattributes will be set; in a coordinate of 80 x 96 pixel.

TESTx,y,(var) tests if pixel is set or not. The result will be found then in the variable. 0=set 1=not set
example: TEST10,59,a:PRINTa

CTESTa,b,c,d reads the colorattribute at the actual cursorposition and saves them in a,b,c,d

ECOLa,b,c,d sets colorattributes for pixelcolor value 0-15
ECOL2,2,8,8
ECOLa,b also is allowed

RECOLa,b,c,d reads the actual pixelcolor from ECOL

the first two parameters of CTEST ECOL RECOL set the foregroundcolor, the second two the backgroundcolor. With the corresponding PATTERN1,0,0,0 you can display over 65000 colors. If you exchange a and b, you may get other colors again. With even and odd y-coordinates, you can set other effects. If you add the fading of the color-command (0-15), you could get a total of 980.000 colors!
The colorblocks 8x6 pixel can be set independent from each other. 256 foreground and 256 backgroundcolors. If this matrix is filled with PATTERN1,0,0,0 you can display 65000 colors in this matrix. With a total of 7200 of such blocks 80x94, the possibilitys are endless. Also you are not limited to use a 8x6 raster, because these matrices can be cut out and used with mode0 without dependency of colors. Not all colors are useful for direct working on the screen. But for pictures every combination is possible. A screenfilter or a monitor with long afterglow like the Commodore 2020 will help then. The CHANGE command can be used optimal then. First you drwa with odd colors, then CHANGE it.

CHANGEa,b,c,d changes the predefined colors with that in ECOL defined ones. Use Mode12-17 for that.
Use DRAW like you alsways do and so on. So only the colors will be changed, that equal with CHANGE.
example:
MODE0:ECOL2,2,8,8:GRAPHIC1,1:SCLR:CIRCLE,50,50,50:PAINT50,50,2:LOCATE50,50:CTESTa,b,c,d
:CHANGEa,b,c,d:ECOL5,5:MODE14:BOX,0,0,14,20,,1
This is a very simple example. If in this pixelarea 100x100 pixel are other colors, then these wouldn't be touched.

COLOR(backgroundcolor),(foregroundcolor),(fading) values 0-15 each
COLOR2,8,5 changes bgcol to 2, fgcol to 8 and fading to 5

SCLR sets the colors defined in ECOL to the whole screen. In multicolormode, the colorattributes have to be set with SCLR after GRAPHIC1,1

SWITCH switches from multicolormode to 2-color mode and vice versa. Loading from REU1750 is more than twice the speed.

TMODEa	saves actual drawingmode to 'a' (see the example 'Maus' on disc) TMODEa:PRINTa
UPa	scrolls the screen up pixelwise, but only in 2-color mode. value 1-255 UP20 scrolls the screen up 20 pixels
DOWNa	scrolls the screen down pixelwise, but only in 2-color mode. value 1-255 DOWN20 scrolls the screen down 20 pixels
UP	same like UPa, but without value for multicolor mode
DOWN	same like DOWNa, but without value for multicolor mode
BLEFT(0-9)	moves the screen to the left. For adjusting the monitor and for effects. BLEFT0 = GRAPHIC1
BRIGHT(0-9)	moves the screen to the right. BRIGHT4

Installation of the hardware:
=====================
Remove from mains plug first !! Remove then all peripherie.

Commodore 128

Open the computer with a screwdriver on the bottom, where the screws are located. Push gently on the bottom-part housing, above the Joystickport1, while lifting the top of the housing. Remove the keyboardcable and Power-LED-cable. Remove the metalcasing, that covers the whole C128-board. Also remove the little metal-cover which covers the graphic-IC. Now you can remove the 8563-IC with the help of two screwdrivers, don't bend any of the pins! Remember the orientation of the nose of the IC; you have to plug it with the same orientation later. Now install the GraphicBooster Hardware to where the IC was and bend the transistor and other components to another position, to install it without shortcircuit. The board, which is connected to the GraphicBoosterHardware you have to install in the U36 socket. The socket has to be installed with the connected wire bottom-left, near the U4 (see drawing in original Manual, if needed). Now reinstall the 8563 into the GB-Hardware nose oriented to userport, like it was installed before. Doublecheck your installation! For a first test, reconnect the keyboard and press the 80-columns key. reconnect the power-unit and peripherie and switch on the C128. The known screen should show up now. If the screen remains black, turn of after 4 seconds and check the installation, again. If the test passed, you can reinstall the metalcasings and housing again. Don't mind, if the metalcasing doesn't fit properly. You have to lay the wire to U36, so the casing can't be closed. After closing the housing with the screwdriver, installation is finished.

Commodore 128D
----
1. housing bottom-side: remove 4 screws with screwdriver Phillips PH2
2. move carrying handle to the side
3. lift housing
4. remove floppy-board with removing the three screws
5a. Important!: mark the connections with a marker, so that you know later, how to reassamble the connections!
5b. remove the connections from the floppy-board
6. move the floppy-board out of its place
7. remove the mains-connector from the board
8.-17. remove screws from metalcasing of floppy

18. remove disc-shutter
19. lift floppy and remove
20. remove the metalcasing that covers the whole C128-board (7 screws)
21. remove MainsPowerUnit
22. now the metalcasing can be removed.
23. prepare the casing so that the place over U36 is liftet for 1cm, so the socket at U36 will hold the hardware for GB128 without shortcircuit.
24. (see drawing in orig. manual)
25. installation is done like explained before in C128-installation
26. remount all connections and cables (check the right orientation)

Installation is done at own risk!
Guarantee will expire, if you open the Computer!
If you install the GB128 hardware careful, no defect to the C128 hardware will occur.
If you don't feel experienced enough, let the local hardware-shop do it for you!

Commodore 128 DCR (with metal-housing)

Installation: Remove the 5 screws on the back and side,
slide the housing to the back and lift to remove.
Remove the EPROM on U34 with a screwdriver,
install the Eprom, delivered with GB128.
Check same orientation as removed Eprom.
Remount housing and close. Ready.

Technical improvements reserved.

Userprograms for the GRAPHIC BOOSTER 128
===================================
1. Boxes 1 screen: self explanatory
2. Boxes 2 screen: self explanatory, for 2 monitors
3. Circle 1 screen: self explanatory
4. Circle 2 screen: self explanatory, for 2 monitors
5. Grafiktablet:
   the use of the "-" before the x-coordinate chooses a new startpoint
   for example: x=-200 y=-300 :line begins on 200/300. without "-" draws from point to point.
   end the program with "END"
6. Artillerie: self explanatory
7. 3-D Konstruktionen:
   after setting the first pixel with the firebutton, wait for a short while.
   arrow-left then calculates 3-D data.
   "W" stores on Ram-expansion 1750
   "R" loads from Ram-expansion 1750
   "E" restarts

Translated by Thunder.Bird on 5. March 2007 (V.1)
Original supply: (c)1986 T. Giger Solothurn Switzerland / COMBO AG