The VESA Based Graphics Package

Plot Library Kernel

R. Ratke
March 1995

Inhaltsverzeichnis

1 Overview

1.1 Preface

The IBM VGA (IBM and VGA are trademarks of International Business Machines Corporation) has become a de-facto standard in the PC graphics world. Many manufactureres offer Super VGAs, providing BIOS or register compatibility with the IBM VGA together with various extension of the VGA standard. These extensions range from higher resolutions and more colors to improved performance. Newest developements include fast graphics processing capabilities. To the benefit of the end user intense competition has dramatically improved the price/performance ratio.

Several serious problems face a software developer who intends to take advantage of these Super VGA environments. As the hardware implementation is not standardized, only very few software products can take advantage of the power and capabilities of Super VGA products directly. The VESA VGA BIOS Extension can be used to remedy this situation. This common software interface to Super VGA graphics products is supported by nearly all manufacturers and grants access to the wide range of features available in todays and future VGA extensions.

1.2 VESA Graphics Package

The graphics package described here provides an interface to 256 colour modes in resolutions of 320*200 and higher (up to 1280*1024 if implemented on your specific Super VGA). No use is made of graphic accelerator facilities. It may be used on every VGA, if a VESA driver is delivered with that card. The package consists of four parts:

- VSVGA.COM, a TSR program which inspects the graphics card and redefines INT 90h. Interrupt 90h, originally intended for use of ROM Basic, then is the interface from graphics primitiva to the hardware.
- A configuration file VGA.CFG residing in the current directory and / or the root of hard disk C:
- Graphics Primitiva is the first part of CALCOMP.LIB. A variety of basic graphics functions is implemented, callable from LAHEY Fortran.
- CALCOMP.LIB is completed by a subset of drawing routines, originally designed and described by CALIFORNIA COMPUTER PRODUCTS Inc. Naming and parameter conventions follow the CALCOMP standards, some extensions are added.

2 How to Use the VESA Graphics Package

To make graphics work on your computer, load:

- The appropriate VESA DRIVER for your VGA card. (Example: Insert the command TLIVESA.EXE into AUTOEXEC.BAT if you use a Tseng Labs ET4000 or compatible card.)
- VSVGA.COM Your screen then will look then similar to the following:

```
V************** Extended driver ***************V
E*      - Graphics Primitiva for SVGAs -      *E
S* with Lahey F77L3 rev 5 / RR, February 1995 *S
A******** VESA-Version, 64 KB Windows *********A
--> Tseng Labs ET4000 installed!

VESA driver version 1.2, cap.bytes: oooooooo oooooooo oooooooo oooooooo
 imode  VESA  Xres  Yres  ncol sup  winA  attA  winB  attB  granul winsize
     0   13h   320   200   256  N
     1 0100h   640   400   256  Y  A000h   05h A000h   03h    64KB    64KB
     2 0101h   640   480   256  Y  A000h   05h A000h   03h    64KB    64KB
     3 0103h   800   600   256  Y  A000h   05h A000h   03h    64KB    64KB
     4 0105h  1024   768   256  Y  A000h   05h A000h   03h    64KB    64KB
max. imode =     4
```

Explanations: `imode=0` will always be present and work as it is the IBM standard mode. All other `imode`s are present and supported by the VESA interface. Note: VSVGA in the present version will not work correctly, if winsize is less than 64 KB. Wether a mode is shown or not depends on your Super VGA and VESA driver only.

You may remove the TSR program VSVGA.COM at any time by the command

VSVGA U(nload)

- Copy the configuration file VGA.CFG to the current directory and / or the root of your hard disk C:

An application – written in Fortran, compiled with LAHEY F77L3 and linked together with CALCOMP.LIB – will be able to use all modes displayed by VSVGA.COM. Your program may call then

- Only the basic routines. All coordinates then are based on pixels. The origin always is in the lower left corner of the screen.
- Only the CALCOMP routines. Units of measure then are inches or cm or whatever you like. The image is scaled in a manner, that the screen is filled and nothing is cut off. The origin always is in the lower left corner of the screen.
- Basic and CALCOMP routines mixed. Colours, write modes etc. may be altered by calling a basic routine in a CALCOMP environment. All other cases must be handled carefully. If you define a window by a basic graphics call, CALCOMP routines have no access to pixels outside the window area.

3 Demonstration of the VESA Graphics Package

After loading the specific VESA driver for your card and VSVGA.COM as described in the previous section, you may run the demonstation program GDEMO.EXE. Be sure, VGA.CFG is present either in the current or / and the root directory of disk C: ! First you will be asked for a mode number. Enter 0 or a higher number (up to max. mode) to select the desired resolution. The program will loop to this point, so each mode can be tested. Enter -1 and the program will finish.

GDEMO first shows the basic features, dots, lines, fill patterns, animation, write modes etc. Each of the parts run until a key on your keyboard is pressed. The demonstration is completed by some CALCOMP examples:

- Symbols available in the `SYMBOL` routine.
- Text drawn by the `SYMBOL` routine at different sizes, angles and slants. (Press ESC to finish this part, any other key to see another text example.)
- A wave, online computed by a Finite Difference scheme. Press any key to pause / continue or ESC to finish this part.)
- Grid and results of a Finite Element computation (groundwater flow).

Whenever a plot is finished, a flickering text Done. Press ENTER [PRTSC] is shown. If now you press

- Print Screen: A hardcopy request is issued and the message is shortened to indicate the pending request.
- ENTER: The screen is dumped byte by byte if a hardcopy request was issued before. The screen is cleared and the program continues.
- any other key: The flickering message disappears and will appear again on a second keystroke.

4 Calls to CALCOMP.LIB

Abbreviations: `I` and `R` denote integer or real parameters, eventually followed by `*2` or `*4` to specify the exact lenght. Reals are always of single precision. If a dimension in brackets follows, the parameter is an array.

Arg denotes a parameter used as argument only: Must have a value before the routine is called, value remains unchanged. Res denotes a result: The parameter transfers a value back to the calling program unit. A/R is a parameter used as argument and a result is obtained too.

After the keyword Remarks a short summary is given, accompanied by necessary explanations. To achieve maximum readibility and clarity, the descriptions are in short tabular form, more a quick reference for an experienced programmer than a tedious thick book for beginners.

4.1 Basic Routines: Primitiva

The subroutines and functions described in this subsection form the interface between LAHEY Fortran and the graphics card. They call interrupt INT 90h, passing and getting back all necessary information via 16-bit registers only. INT 90h is directed to the resident part of VSVGA.COM, where fast subfunctions manipulate the video memory directly and invoke the appropriate VESA functions.

General: All integer parameters are internally declared as `INTEGER*2`. Values passed to the routines however may be of type `INTEGER*4`, but only the low word will be processed. This will cause no error for arguments, as the low word is the first one. Non negative results are obtained correctly, if an `INTEGER*4` variable is set to zero before the call. In all other cases `INTEGER*2` variables must be used. `REAL` parameters are not used in this subsection.

Purpose: Set graphics mode SUBROUTINE: **G_MODE**

Call: `CALL G_MODE (IMODE)`

Parameter	Type	A/R	Description			
IMODE	I	Arg	Graphic resolution:	< 0	:	Return to text mode
				0	:	320*200
				1	:	640*400
				2	:	640*480
				3	:	800*600
				4	:	1024*768
				5	:	1280*1024

Remarks: `G_MODE` turns on the graphics environment in the selected mode. If a mode not supported by the specific VGA card or VESA driver is chosen, the nearest mode available with a smaller number will be used.

Purpose: Initialize graphics ENTRY: **G_INIT**

Call: `CALL G_INIT`

Parameter	Type	A/R	Description
			– none –

Remarks: All internal values are reset to their default values. The RGB palette remains unchanged. Window is set to full screen. The screen is blanked.
`G_INIT` is part of `G_MODE`.

Purpose: Get graphics information ENTRY: **G_INFO**

Call: `CALL G_INFO (MAXMOD)`

Parameter	Type	A/R	Description
MAXMOD	I	Res	Maximum mode availabe for graphics

Remarks: `G_INFO` performs a query to the `VSVGA` kernel. The `COMMON` block `/SVGA_INFO` becomes actualized and may be interrogated by the user. The error flag `GERROR` is internally reset. A subsequent call will only report new errors.

Graphics information can be taken from `COMMON /SVGA_INFO/` declared as

```
 INTEGER*2
&   NPIXH, NPIXV,ACTCOL,ACTMOD,OLDMOD,PTHICK,NCOLTX,NROWTX,
&  X0CLIP,Y0CLIP,NXCLIP,NYCLIP,ACTFIL, ACTLS, ACTFS,ACTASP,
&  ACTWRI, HCREQ,GERROR
 COMMON/SVGA_INFO/
&   NPIXH, NPIXV,ACTCOL,ACTMOD,OLDMOD,PTHICK,NCOLTX,NROWTX,
&  X0CLIP,Y0CLIP,NXCLIP,NYCLIP,ACTFIL, ACTLS, ACTFS,ACTASP,
&  ACTWRI, HCREQ,GERROR
```

where

`NPIXH, NPIXV`	number of pixels horizontal, vertical
`ACTCOL`	colours in effect (low byte: draw colour, high: background)
`ACTMOD`	current graphics mode (0 to max. imode)
`OLDMOD`	previous (text) mode
`PTHICK`	pen thickness, not supported, always 1
`NCOLTX, NROWTX`	number of columnns / rows for text in current mode
`X0CLIP, Y0CLIP`	coordinates of the lower left corner of of the graphics area
`NXCLIP, NYCLIP`	size of the graphics window
`ACTFIL`	fill pattern number actually selected
`ACTLS`	line style number actually selected
`ACTASP`	aspect ratio (*10000) in current graphics mode
`ACTWRI`	write mode actually in effect
`HCREQ`	if > 0, hardcopy request is pending
`GERROR`	error flag (0: ok, 1: minor, 2: severe error(s) occured)

Note: No graphics information except `MAXMOD` and `GERROR` is returned in text mode. See also: **G_CONFIG**.

Purpose: Clear graphics screen ENTRY: **G_CLS**

Call: `CALL G_CLS`

Parameter	Type	A/R	Description
			– none –

Remarks: The entire screen is blanked.
`G_CLS` is part of `G_MODE` and `G_INIT`.

Purpose: Define graphics window ENTRY: **G_SETCLIP**

Call: `CALL G_SETCLIP (IX1,IY1,IX2,IY2)`

Parameter	Type	A/R	Description
`IX1`	`I`	Arg	left border of the window
`IY1`	`I`	Arg	lower border
`IX2`	`I`	Arg	right border
`IY2`	`I`	Arg	upper border

Remarks: Afterwards no pixel outside the borders can be accessed. The lower left corner `(IX1,IY1)` becomes the origin of the pixel coordinates.

Purpose: Move to point ENTRY: **G_MOVE**

Call: `CALL G_MOVE (IX,IY)`

Parameter	Type	A/R	Description
`IX`	`I`	Arg	x pixel coordinate
`IY`	`I`	Arg	y pixel coordinate

Remarks: Current pixel is defined at `(IX,IY)` and coloured according to selected coulours, write mode and line style. Entry is used to draw the first point of a polygonal line.

Purpose: Draw to point ENTRY: **G_DRAW**

Call: `CALL G_DRAW (IX,IY)`

Parameter	Type	A/R	Description
`IX`	`I`	Arg	x pixel coordinate
`IY`	`I`	Arg	y pixel coordinate

Remarks: A straight line is drawn from current pixel (starting pixel not included) to `(IX,IY)` and coloured according to selected coulours, write mode and line style. `(IX,IY)` becomes the new current pixel.

Purpose: Set RGB values for a single colour ENTRY: **G_SETRGB**

Call: `CALL G_SETRGB (NO,IR,IG,IB)`

Parameter	Type	A/R	Description
NO	I	Arg	Colour number, 0 to 255
IR	I	Arg	Red value , 0 to 63
IG	I	Arg	Green value , 0 to 63
IB	I	Arg	Blue value , 0 to 63

Remarks: The lookup table for colour number `NO` is immediately set to the given RGB values. Used to define a colour and to animate pixels already drawn in this colour.
Note: Usually not all colours are predefined in 256 colour modes.

Purpose: Set current colours ENTRY: **G_COLORS**

Call: `CALL G_COLORS (IDRAW,IBACK)`

Parameter	Type	A/R	Description
IDRAW	I	Arg	Draw colour number, 0 to 255
IBACK	I	Arg	Background colour, 0 to 255

Remarks: Subsequent graphic calls will use the current colours.

Purpose: Set current line style ENTRY: **G_LINESTYLE**

Call: `CALL G_LINESTYLE (NO,USER)`

Parameter	Type	A/R	Description
NO	I	Arg	Line style number, 0 to 4
USER	I	Arg	User provided line style for NO=4

Remarks: Styles 0 to 3 are predefined, with style number 0 drawing a continous line. If style no 4 is selected, the bits of USER define a pattern: Pixels are coloured with draw colour if corresponding bits are set else with background colour.

Purpose: Fill rectangle ENTRY: **G_PATTERN**

Call: `CALL G_PATTERN (IX1,IY1,IX2,IY2,NO,USER)`

Parameter	Type	A/R	Description
IX1	I	Arg	x coordinate of 1st corner
IY1	I	Arg	y coordinate of 1st corner
IX2	I	Arg	x coordinate of 2nd corner
IY2	I	Arg	y coordinate of 2nd corner
NO	I	Arg	Fill style number, <0 to 12
USER	I(4)	Arg	User fill style

Remarks: The rectangle is filled with a pattern defined by NO. Number NO=0 fills it completely with draw colour, 1 with background colour. Different patterns are predefined by NO from 2 to 11. For number <0, the INTEGER*2 array USER must supply 8 bytes for a user pattern. The 8*8 bits map fore- and background colours for an 8*8 pixel area. The pattern is stored, used and can be referenced in later calls as pattern number 12.

Purpose: Text interface ENTRY: **G_TEXT**

Call: `CALL G_TEXT (IX,IY,IFACT,ISWITCH,TEXT)`

Parameter	Type	A/R	Description
IX	I	Arg	x coordinate of lower left corner for text
IY	I	Arg	y coordinate
IFACT	I	Arg	Enlargement factor: 1 to 8
ISWITCH	I	Arg	< 0: Store coordinates and factor, no text output. $= 0$: Write text at (IX,IY) enlarged with IFACT. > 0: Ignore (IX,IY) and IFACT. Write at current text position with current factor.
TEXT	CHAR	Arg	Text to write

Remarks: Text position and factor are stored and updated internally. The 8*8 ROM font is used and can be enlarged. Draw and background colours apply.
Note: Coordinates refer to the lower left corner of TEXT. On return, the current internal text x-position is updated to the right of the text for use in a following call with ISWITCH >0.

Purpose: Set pixel row ENTRY: **SET_PIX**

Call: `CALL SET_PIX (IX,IY,NPIX,ICOL)`

Parameter	Type	A/R	Description
IX	I	Arg	x coordinate of leftmost pixel
IY	I	Arg	y coordinate of row
NPIX	I	Arg	Number of pixels
ICOL	I	Arg	Colour for pixel row

Remarks: A horizontal line of `NPIX` pixels is coloured with colour `ICOL`.

Purpose: Get pixel colour ENTRY: **GET_PIX**

Call: `CALL GET_PIX (IX,IY,ICOL)`

Parameter	Type	A/R	Description
IX	I	Arg	x coordinate of pixel
IY	I	Arg	y coordinate
ICOL	I	Res	Colour of pixel

Remarks: On return, `ICOL` will have colour number of specified pixel.

Purpose: Hardcopy: Screen dump ENTRY: **G_DUMP**

Call: `CALL G_DUMP (LFN)`

Parameter	Type	A/R	Description
LFN	I	Res	number of dump file (<0 if an error occured)

Remarks: The screen is dumped row by row and pixel by pixel. File names are built of a hexadecimal number from 0000 to 7FFF with the extension .RAW. The routine searches for a free filename. The number is returned in `LFN` if the dump completed successfully. Together with the file COLOR.MAP the complete video information is saved. (There are a lot of programs available on the market for creating any kind of graphics metafile from the raw information – not provided here.)

Purpose: Set write mode ENTRY: **SET_WRMOD**

Call: `CALL SET_WRMOD (MODE)`

Parameter	Type	A/R	Description
MODE	I	Arg	Write mode 0 to 3

Remarks: Write mode has the effects:

0	Write absolute:	Set pixel to current colour
1	XOR mode:	Pixels colour is XORed with current colour
2	OR mode:	Pixels colour is ORed with current colour
3	Transparent:	Write absolute if foreground leave pixel unchanged if background

Purpose: Save or restore VGA state ENTRY: **VGA_STATE**

Call: `CALL VGA_STATE (MODE)`

Parameter	Type	A/R	Description
`MODE`	`I`	A/R	on entry: 0: saves, 1: stores on return: 0: errors, 1: ok

Remarks: Video mode and RGB palette should be saved. If there is not enough room in the internal buffer, return from the routine gives `MODE` < 1.

Purpose: Read keyboard INTEGER*2 FUNCTION: **INKEY**

Call: `CALL INKEY (WAIT)`

Parameter	Type	A/R	Description
`WAIT`	`I`	Arg	0: return immediately if nothing is in the keyboard buffer else: Wait for key and read it

Remarks: `INKEY` result:
- 0: nothing in buffer (only if `WAIT` was 0)
- > 0: `ASCII` code of the key
- < 0: negated scan code of an extended key

Purpose: Return configuration information SUBROUTINE: **G_CONFIG**

Call: `CALL G_CONFIG (PENS,MXMOD)`

Parameter	Type	A/R	Description
PENS	I (0:15)	Res	colours numbers of the first 16 pens used in NEWPEN
MXMOD	I	A/R	if < 0 on entry: only the maximum mode is returned in MXMOD, no other action result: maximum mode allowed in VGA.CFG, -1 means no restriction is given.

Remarks: The routine reads the configuration file which is first searched in the current directory. If not present there VGA.CFG must be in the root directory of hard disk C:

If only the maximum mode is wanted, the graphics system may be in text mode. Other actions require graphics mode in effect. Then the pen lookup table will be returned and all RGB registers are set according to the configuration. A map file COLOR.MAP is created to save the RGB values for later use.

The first call stores all values internally. The second and later calls are faster, as they use the stored values instead of reading the configuration file again.

Note: Differentiate between MXMOD here and MAXMOD returned by the G_INFO routine! The later tells the maximum mode supported by VGA card and VESA driver while MXMOD should be set in VGA.CFG to protect a monitor not capable of the higher modes.

Purpose: Draw an arc SUBROUTINE: **G_ARC**

Call: `CALL G_ARC (MX,MY,IR1,IR2,IA1,IA2)`

Parameter	Type	A/R	Description
MX	I	Arg	x coordinate of center point
MY	I	Arg	y coordinate of center point
IR1	I	Arg	Distance from center point to start of the arc
IR2	I	Arg	Distance from center point to end of the arc
IA1	I	Arg	Angle in degrees from center to start of the arc
IA2	I	Arg	Angle in degrees from center to end of the arc

Remarks: Coordinates and distances must be given in pixels. Angles are in mathematical positive sense (counterclockwise), starting at the x-axis. Write mode and linestyles apply. If the aspect ratio is not 1.0, only x direction will have true lenght, while y distances are modified then.

`CALL G_ARC (MX,MX,25,25,0,360)` will result in a full circle with diameter 50 around `(MX,MY)`. It should always look like a true circle!

Purpose: Draw an ellipsis SUBROUTINE: **G_ELLIPS**

Call: `CALL G_ELLIPS (XM,YM,DX,DY,B,FILL)`

Parameter	Type	A/R	Description
XM	I	Arg	x coordinate of the center
YM	I	Arg	y coordinate of the center
DX	I	Arg	Diameter, x direction
DY	I	Arg	Diameter, y direction
B	I	Arg	Border thickness
FILL	I	Arg	Fill style, if < 0 :unfilled

Remarks: Coordinates and diameters must be given in pixel units. Y distances are modified by the aspect ratio. The border is drawn with draw color, `B` pixels wide. If `FILL` is not negative, the interior is filled with the selected fill style.

4.2 CALCOMP compatible routines

The subroutines described in this section are the basis of a CALCOMP style plot libray. They never use the `VESA` driver directly, only calls to the graphics primitiva are performed.

Sequence off calls: Calls to the library should be organized as follows:

- `CALL PLOTS (...)` to define the resolution used. If omitted, the maximum resolution available will be selected.
- `CALL NEWPLOT (...,XL,YL,...)` to define the size of a plot.
  `CALL PLOT (XL,YL,1)` has the same effects. Both call turn on turns on graphics mode.
- Call other graphics routines .....
- `CALL PLOT (...,...,0)` or `CALL PLOT (...,...,999)` ends the plot. The user will be prompted for an optional hardcopy and the ENTER key to return to text mode.
  `CALL PLOT (XL,YL,1)` or `CALL NEWPLOT (...,XL,YL,...)` do the same but initiate a new drawing instead of returning to text mode.
- `PLOTS` and `NEWPLOT` may be called as often as needed.

Parameter types: All real parameters are `REAL*4`. Most integer parameter are `INTEGER*4`, but some auxiliary routines use `INTEGER*2` for faster transfer. These paremeters are marked with `I*2` in the descriptions, `INTEGER*4` values may be used then as actual **arguments** only in calls!

Purpose: Main plot routine SUBROUTINE: **PLOT**

Call: `CALL PLOT (XPAGE,YPAGE,IPEN)`

Parameter	Type	A/R	Description
XPAGE	R	Arg	x coordinate or lenght
YPAGE	R	Arg	y coordinate or lenght
IPEN	I	Arg	Pen code

Remarks: Parameter `IPEN` defines the action to happen:

IPEN	action
0	Closes graphic
1	Turns on graphics with a virtual sheet of size `XPAGE * YPAGE`. The virtual sheet is scaled. The screen is filled in at least one direction. Coordinate origin is in the lower left corner of the screen. All internal variables are normalized.
2	Draw absolute: A line is drawn from last point to `(XPAGE, YPAGE)`.
3	Move absolute: Last point is updated only to `(XPAGE, YPAGE)`.
-2, -3	Same as 2, 3. The origin is moved to `(XPAGE, YPAGE)`.
12, 13	Plot / move relative: `XPAGE, YPAGE` are increments.
-12, -13	Same as 12, 13. The origin is moved to the new point.
4	`XPAGE, YPAGE` ignored. Write mode is set to absolute.
5	`XPAGE, YPAGE` ignored. Write mode is set to XOR mode.
6	`XPAGE, YPAGE` ignored. Write mode is set to OR mode.
7	`XPAGE, YPAGE` ignored. Write mode is set to transparent.
999	Same as 0

Notes:

a) A value of 999.0 for `XPAGE` or `YPAGE` means no change of the corresponding coordinate if `IPEN` is -2, -3, 2, 3, -12, -13, 12 or 13. In all other cases these values must be < 999.0!

b) `PLOT` will correctly clip all parts of a drawing exceeding the graphics window. Origin and actual pen position however will always be maintained, even if outside the window.

Purpose: Select resolution ENTRY: **PLOTS**

Call: `CALL PLOTS (I1,I2,MODE)`

Parameter	Type	A/R	Description
`I1`	`I`	Arg	unused
`I2`	`I`	Arg	unused
`MODE`	`I`	Arg	Mode number (0 to 5) to use in next plots

Remarks: See `G_MODE` also.

Purpose: Select colour(s) ENTRY: **NEWPEN**

Call: `CALL NEWPEN (NPEN)`

Parameter	Type	A/R	Description
`NPEN`	`I`	Arg	Pen number to use in next calls

Remarks: `NPEN` is background pen *256 + draw pen (or simply draw pen, if background color is 0). Pen numbers < 16 are converted by the pen-colour lookup table, otherwise they are colour numbers.

Purpose: Colour(s) of a pen ENTRY: **CHKPEN**

Call: `CALL CHKPEN (NPEN,ICOLOR)`

Parameter	Type	A/R	Description
NPEN	I*2	Arg	Pen number
ICOLOR	I*2	A/R	Colour number(s)

Remarks: `NPEN` is translated to the colour numbers, if back- or foreground bytes are > 15. If the `ICOLOR` argument is zero, the result will only contain draw color.

Purpose: Get transformation constants ENTRY: **KOORZZ**

Call: `CALL KOORZZ (AX,BX,AY,BY,LX1,LX2,LY1,LY2)`

Parameter	Type	A/R	Description
AX	R	Res	See below
BX	R	Res	See below
AY	R	Res	See below
BY	R	Res	See below
LX1	I*2	Res	Minimum x coordinate, always 0
LX2	I*2	Res	Maximum x coordinate
LY1	I*2	Res	Minimum y coordinate, always 0
LY2	I*2	Res	Maximum y coordinate

Remarks: Screen coordinates `(IX,IY)` of the pixel nearest to `(X,Y)` are obtained by:

```
IX = AX + BX * X
IY = AY + BY * Y
```

Purpose: Get pen position ENTRY: **WHERE**

Call: `CALL WHERE (XPAGE,YPAGE,FACT)`

Parameter	Type	A/R	Description
XPAGE	R	Res	Actual x coordinate
YPAGE	R	Res	Actual y coordinate
FACT	R	Res	Actual factor, dummy, always 1.0

Remarks: Coordinates are relative to the lower left corner of the virtual sheet!

Purpose: Scale a drawing ENTRY: **FACTOR**

Call: `CALL FACTOR (FACT)`

Parameter	Type	A/R	Description
FACT	R	Arg	Factor

Remarks: Dummy routine. Here `FACT` will have no effect. See `NEWPLOT`.
Included for compatibility with pen plotter versions.

Purpose: Convert coordinates ENTRY: **CHK999**

Call: `CALL CHK999 (XPAGE,YPAGE,IPEN,XP,YP)`

Parameter	Type	A/R	Description
XPAGE	R	Arg	x coordinate or 999.0
YPAGE	R	Arg	y coordinate or 999.0
IPEN	I	Arg	Pen code
XP	R	Res	x coordinate
YP	R	Res	y coordinate

Remarks: Coordinate values 999.0 or greater are converted to the actual pen position or zero, depending on whether `IPEN` indicates absolute or relative plotting.

Purpose: Initiate a new drawing SUBROUTINE: **NEWPLOT**

Call: `CALL NEWPLOT (IUNIT,IDEV,XL,YL,IERR)`

Parameter	Type	A/R	Description
IUNIT	I	Arg	Dummy: (Logical device number of plotter)
IDEV	I	Arg	Dummy: (Type of plotter)
XL	R	Arg	x length of the new plot
YL	R	Arg	y length of the new plot
IANZ	I	Arg	Dummy: (Plotting stops after `IANZ` errors)

Remarks: Included for compatibility with pen plotter versions.
`CALL PLOT (XL,YL,1)` is performed.

Purpose: Close graphics SUBROUTINE: **PL_END**

Call: `CALL PL_END`

Parameter	Type	A/R	Description
			– none –

Remarks: Prompts for ENTER, dumps screen to file if requested.

Purpose: Symbol routine, extended SUBROUTINE: **SYMBOL**

Call: `CALL SYMBOL (XPAGE,YPAGE,WIDTH,ASCII,IORD,ANGLE,NCHAR)`

Parameter	Type	A/R	Description
XPAGE	R	Arg	x coordinate or 999.0
YPAGE	R	Arg	y coordinate or 999.0
WIDTH	R	Arg	Symbol size
ASCII	CHAR	Arg	Text to draw if NCHAR > 0
IORD	I	Arg	Centered symbol number / ASCII code
ANGLE	R	Arg	Angle in degrees from x axis
NCHAR	I	Arg	Number of characters in ASCII or code

Remarks: There are three different calls to SYMBOL:

- NCHAR > 0: Text ASCII is plotted. If the first character is an underline, it will be omitted but the rest of the text is underlined. The routine uses proportional spacing. WIDTH is the average width and the height of the characters. XPAGE, YPAGE denote the lower left end of the text box. Pen position is updateted to the right end after plotting.
- NCHAR $= 0$: One single Character, whose ASCII code is given by IORD, is drawn. Pen position is updated.
- NCHAR -2 or -1: A centered Symbol is drawn. IORD is the number of the symbol. If NCHAR equals -2, a line is drawn from old pen position to the center of the symbol else the line is omitted. Pen position is XPAGE, YPAGE after the call.

The IBM PC character set is implemented. ASCII Codes < 32 are not displayable. A special greek technical set overlays the graphic symbols. Use of unimplemented symbols result in the number sign beeing plotted.
No part of a character is drawn more than once, SYMBOL will work well in XOR and other write modes also.

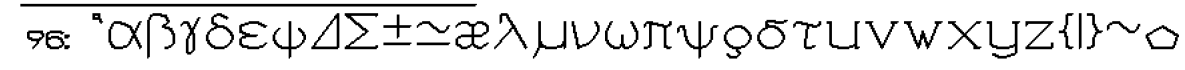
(↪ see next page)

continued: SYMBOL

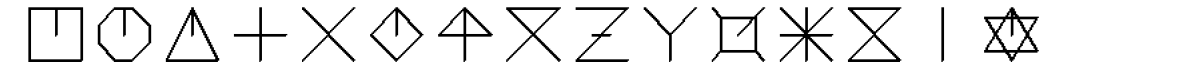
Character set in routine `SYMBOL`

ASCII–Set

GREEK–TECHNICAL Set



CENTERED SYMBOLS 0 – 14



Purpose: Define character slant ENTRY: **SYMBSL**

Call: `CALL SYMBSL (SLANT)`

Parameter	Type	A/R	Description
SLANT	R	Arg	Slant in degrees

Remarks: Next `SYMBOL` calls will show slanted characters. `NEWPLOT` presets `SLANT` with 0.0.

Purpose: Define character height ENTRY: **SYMBHB**

Call: `CALL SYMBHB (HB)`

Parameter	Type	A/R	Description
HB	R	Arg	Height to width ratio

Remarks: Next characters plotted by `SYMBOL` will have the given height to width ratio. `NEWPLOT` presets `HB` with 1.0.

Purpose: Select character set ENTRY: **SYMBGR**

Call: `CALL SYMBGR (GREEK)`

Parameter	Type	A/R	Description
GREEK	I	Arg	Code, see remarks

Remarks: `GREEK` $>$ 0 overlays characters a-t with the greek technical set. `GREEK` $<$ 1 resets the character set to its original state. `NEWPLOT` presets `GREEK` with 0.

Purpose: Save settings ENTRY: **SYMSAV**

Call: `CALL SYMSAV`

Parameter	Type	A/R	Description
			– none –

Remarks: The settings `SLANT, HB, GREEK` of `SYMBOL` are saved. Two levels can be pushed. Called from `NUMBER`.

Purpose: Restore settings ENTRY: **SYMRES**

Call: `CALL SYMRES`

Parameter	Type	A/R	Description
			– none –

Remarks: The settings `SLANT, HB, GREEK` of `SYMBOL` are restored. Information can be popped from two save levels. Called from `NUMBER`.

Purpose: Draw a number, extended SUBROUTINE: **NUMBER**

Call: `CALL NUMBER (XPAGE,YPAGE,WIDTH,FPN,ANGLE,NDEC)`

Parameter	Type	A/R	Description
XPAGE	R	Arg	x coordinate or 999.0
YPAGE	R	Arg	y coordinate or 999.0
WIDTH	R	Arg	Character size
FPN	R	Arg	Real number to draw
ANGLE	R	Arg	Angle in degrees from x axis
NDEC	I	Arg	Number of digits right of the decimal point / code

Remarks: The number is plotted in standard format if NDEC < 10. For NDEC $= -1$ only the integer part is plotted and the decimal point is omitted.

For NDEC > 9 FPN is taken as time in seconds and displayed in d-h-m-s format, depending on NDEC:
Example for FPN=100000 s:

```
NDEC   displayed
1000   1d
1100   1d4h
1010   1d3h47m              (the leftmost 1 gives the biggest unit,
1001   1d3h46m40s           the rightmost 1 denotes the smallest.)
0100   27h
0110   27h47m
0101   27h46m40s
0010   1667m
0011   1666m40s
```

Purpose: Draw an axis, extended SUBROUTINE: **AXIS**

Call: `CALL AXIS (XPAGE,YPAGE,TEXT,NCHAR,AXLEN,ANGLE,FIRSTV,DELTAV)`

Parameter	Type	A/R	Description
XPAGE	R	Arg	x coordinate or 999.0, start of the axis
YPAGE	R	Arg	y coordinate or 999.0
TEXT	CHAR	Arg	Title for the axis
NCHAR	I	Arg	Number of characters in the title
AXLEN	R	Arg	Length of the axis
ANGLE	R	Arg	Angle of the axis in degrees
FIRSTV	R	Arg	Value corresponding to the start of the axis
DELTAV	R	Arg	Value to length ratio

Remarks: The sign of `NCHAR` defines which side of the axis ticks, values and text are placed: + is on the left, - on the right side, seen from the start of the axis. Sizes and distances are preset by `NEWPLOT` for cm as unit of length. They can easily be altered for inches or other customers needs by the entries described below.

Purpose: Define tick height and distance ENTRY: **AXMARK**

Call: `CALL AXMARK (HTIC,DTIC)`

Parameter	Type	A/R	Description
HTIC	R	Arg	Height of ticks at the axis
DTIC	R	Arg	Distance between ticks

Remarks: Standard setting is 0.18 and 1.00 (cm). For inches as unit of lenght, use `CALL AXMARK (0.075, 0.5)` as standard.

Purpose: Define axis numbering ENTRY: **AXZIFF**

Call: `CALL AXZIFF (NZ,AZ,WZ,SZ,HZ,JZ)`

Parameter	Type	A/R	Description
NZ	I	Arg	`NDEC` for numbers at the axis ticks
AZ	R	Arg	Angle between numbers and the axis in degrees
WZ	R	Arg	Width of the number digits
SZ	R	Arg	Slant for numbers
HZ	R	Arg	Height to width ratio
JZ	I	Arg	Every `JZ`th tick will become numbered

Remarks: Default settings for cm units are obtained by
`CALL AXZIFF (2, 0.0, 0.26, 0.0, 1.0, 2).`
`CALL AXZIFF (2, 0.0, 0.1, 0.0, 1.0, 2)` will work well for inches.

Purpose: Define axis text settings ENTRY: **AXTEXT**

Call: `CALL AXTEXT (IT,AT,WT,ST,HT)`

Parameter	Type	A/R	Description
IT	I	Arg	centering code
AT	R	Arg	Angle between text and the axis in degrees
WT	R	Arg	Width of the characters
ST	R	Arg	Slant
HT	R	Arg	Height to width ratio

Remarks: `IT` =1: text starts at origin of the axis, =2: centers text, =3: text and axis have common ends.
Default settings for cm units are obtained by `CALL AXTEXT (2, 0.0, 0.35, 0.0, 1.0)`. `CALL AXTEXT (2, 0.0, 0.15, 0.0, 1.0)` will work well for inches.

Purpose: Number of digits INTEGER FUNCTION: **NZZIFF**

Call: `CALL NZZIFF (FPN,NDEC)`

Parameter	Type	A/R	Description
FPN	R	Arg	Real number
NDEC	I	Arg	Number of decimals

Remarks: `NZZIFF` computes the number of characters `FPN` will consist of if output is done by the `NUMBER` routine. All special cases are handled.
(Used by `AXIS`.)

Purpose: Polygon fill – scanline SUBROUTINE: **FILLZZ**

Call: `CALL FILLZZ (XY,NXY,ICOL,H)`

Parameter	Type	A/R	Description
XY	R(2,NXY)	Arg	Coordinates of the polygon corners
NXY	I*2	Arg	Number of polygon corners
ICOL	I*2	Arg	Fill colour
H	I(2*NXY+..)	–	Temporary storage

Remarks: `FILLZZ` fills any arbitrary polygon. There must be enough room in `H` for `2*NXY` pixel coordinates plus additional 3* the maximum number of intersections of the polygon with a single scanline.

Purpose: Triangle: Interpolate colours SUBROUTINE: **INTER3**

Call: `CALL INTER3 (XY,IDXY,Z,IDZ,ELE,ZMIN,DZ,PALETT,NF)`

Parameter	Type	A/R	Description
XY	R(IDXY,0:*)	Arg	Coordinates of all system nodes
IDXY	I	Arg	Number of rows in XY; (XY(1,j),XY(2,j)) are the coordinates of node j.
Z	R(IDZ,0:*)	Arg	Function values of all system nodes
IDZ	I	Arg	Number of rows in Z; Z(1,j) is the function value of node j.
ELE	I(3)	Arg	Node numbers of a single triangle element. Numbers are beginning with 0.
ZMIN	R	Arg	Minimum function value for colours, see below
DZ	R	Arg	Function increment for colours, see below
PALETT	I(1,NF)	Arg	External palette
NF	I	Arg	Number of colours

Remarks: The triangle is shaded pixel by pixel. A linear interpolation is used for the function values. Function values <ZMIN result in colour PALETT(1) otherwise the pixels colour is PALETT (2+ MAX(NF, INT((Z−ZMIN)/DZ))).
For presentation of scalar results (Finite Elements).
Not suitable for use in XOR write mode!

Purpose: Quadrilateral: Interpolate colours SUBROUTINE: **INTER4**

Call: `CALL INTER4 (XY,IDXY,Z,IDZ,ELE,ZMIN,DZ,PALETT,NF)`

Parameter	Type	A/R	Description
XY	R(IDXY,0:*)	Arg	Coordinates of all system nodes
IDXY	I	Arg	Number of rows in XY; (XY(1,j),XY(2,j)) are the coordinates of node j.
Z	R(IDZ,0:*)	Arg	Function values of all system nodes
IDZ	I	Arg	Number of rows in Z; Z(1,j) is the function value of node j.
ELE	I(3)	Arg	Node numbers of a single quadrilateral element. Numbers are beginning with 0.
ZMIN	R	Arg	Minimum function value for colours, see below
DZ	R	Arg	Function increment for colours, see below
PALETT	I(1,NF)	Arg	External palette
NF	I	Arg	Number of colours

Remarks: The quadrilateral is shaded pixel by pixel. A bilinear interpolation is used for the function values. Function values $<$ZMIN result in colour PALETT(1) otherwise the pixels colour is PALETT (2+ MAX(NF, INT((Z-ZMIN)/DZ))).
For presentation of scalar results (Finite Elements).
Not suitable for use in XOR write mode!

4.3 Conclusion

The routines described above are the skeleton only of a full CALCOMP library. Other routines as `SCALE, LINE, LGAXS` etc. may be added liberately. The extensions add a portion of functionality especially designed for use in scientific and technical computing.

5 A Commented Example of VGA.CFG

The configuration file is used to inform the graphic routines about the users needs concerning resolution, colours and hardcopy. The example given `verbatim` is commented in *italics*.

```
$VGA.CFG - an example
$Section 1: Define hardcopy printer type
```

All lines starting with $ are treated as comments.
*Section 1 is not processed, provided only for compatibility with old versions and may be omitted. Cols 1-10 hold the keyword *HARDCOPY*, cols 11-20 a printer type keyword. First definition is active, subsequent redefinitions will be ignored.*

```
*HARDCOPY*9_NEEDLES
*HARDCOPY*EPSON24
$Section 2: Define the mode selection parameters
$FORMAT (A10,8I5,A10,A10)  name, 8 integers, palette name, Hardcopy palette
$MODNAM   MXMOD<---------- unused ---------------> PALETTE  H.PALETTE comment
$ MODNAM : arbitrary name, not referenced later
$ MXMOD  : if not blank and value > -1: maximum mode is restricted to MXMOD,
$          should be set to the maximum mode allowed for the connected monitor.
$          (modes in VSVGA.COM are 0 to 5)
$ PALETTE:   name must appear in cols. 1-10 in one of the later lines
$            (section 3), followed by a palette definition, see below.
$ H.PALETTE: A non blank field in cols. 61-70 must contain a HARDCOPY PALETTE
$            name. The same name is used in section 4 in cols. 1-10.
$            A blank field in cols. 61-70 means dump RBG values to COLOR.MAP
$ All names are case sensitive!
$ 1st definition appearing here will be in effect!
VSVGA.COM    ??<---------- unused ---------------> BLUE_RED GREYSCALE P.SCRIPT
VSVGA.COM     4<---------- unused ---------------> BLUE_RED REVERSE   DIAs
VSVGA.COM     4<---------- unused ---------------> BLUE_RED           RGB-vals
```

The first integer only and the palette names are read by VSVGA.COM. A blank or non numeric field for MXMOD results in -1.
Older versions of the driver used 8 integer values, given here for completeness:

```
$OLD:  8 integers to set registers and resolutions for a specific mode
$MODNAM      AX   BX   MX   MY  MXT  MYT   MC  XOR PALETTE  H.PALETTE comment
$ AX,  BX  :register values, leading # means hexadecimal value
$ MX,  MY  :max. graphic coordinates
$ MXT, MYT :max. columm, row for text in that mode
$ XOR      :0  XOR-mode is not supported by INT 10h
```

(↪ see next page)

continued

```
OLD configurations, not for use with VSVGA.COM
"$" is not needed, if lines do not begin with palette names
ET1024/256  #38    0 1023  767  127   47  255    0 BLUE_RED   TSENG-ET4000
IBM320/256  #13    0  319  199   39   19  255    0 PALETTE_1  IBM STANDARD VGA
V7 640/256#6F05  #67  639  479   79   29  255    0 SEACLOUD   V_7 1024i/512K
GN 640/256  #5E    0  639  479   79   29  255    0 SEACLOUD   GENOA 6400

$Section 3: Palette definitions for VGA, selected by palette name of 2nd sect.
1st line: (A10,16I4) palette name, numbers of colours for NEWPEN(0 to 15)
```

Used to change pens easier, without changing RGB values. Here intensive colours first!

```
 PALETTE_1   0   9  10  11  12  13  14  15   8   1   2   3   4   5   6   7
$define colours: (4I5) number, R, G, B
$if number is negative: interpolate from last number to current number
$   I   RI   GI   BI (RGB range: 0 to 63)
    0    0    0    0  BLACK
    1    0    0   40  BLUE
    2   40    0    0  RED
    3   40    0   40  MAGENTA
    4    0   40    0  GREEN
    5    0   40   40  CYAN
    6   40   40    0  YELLOW
    7   40   40   40  WHITE
    8   20   20   20  GREY
    9    0    0   63  INT. BLUE
   10   63    0    0  INT. RED
   11   63    0   63  INT. MAGENTA
   12    0   63    0  INT. GREEN
   13    0   63   63  INT. CYAN
   14   63   63    0  INT. YELLOW
   15   63   63   63  INT. WHITE
$ shades, any order, misses and redifinitions (last values valid) allowed
   16    4    4    4 GREY SHADES
  -63   63   63   63     -WHITE
   64   63    4    4 RED
 -127    4   32   32     -CYAN
  128    4   63    4 GREEN
 -191    4    4    4     -GREY
  192    4    4   63 BLUE
 -255    4    4    4     -GREY
#### place a non numeric line after last line to indicate end of palette
```

(↪ see next page)

continued

Unused palette may appear here, selection of palette is always made by palette name!

```
 BLUE_RED    0   9  10  11  12  13  14  15   8   1   2   3   4   5   6   7
    0    0    0    0  BLACK    above: colournumbers for pens 0 to 15
    1    0    0   40  BLUE     left:  colourdefinitions RGB
    2   40    0    0  RED
    3   40    0   40  MAGENTA
    4    0   40    0  GREEN
    5    0   40   40  CYAN
    6   40   40    0  YELLOW
    7   40   40   40  WHITE
    8   18   18   18  GREY
    9    0    0   63  INT. BLUE
   10   63    0    0  INT. RED
   11   63    0   63  INT. MAGENTA
   12    0   63    0  INT. GREEN
   13    0   63   63  INT. CYAN
   14   63   63    0  INT. YELLOW
   15   63   63   63  INT. WHITE
$ ------------------------------ SHADES BLUE TO RED
   16   13    0   36  DARK BLUE
  -27    9    0   28
  -59    7    7   48  LIGHT BLUE
 -136   63   63   25  WHITE / YELLOW
 -219   50    0   20  LIGHT RED
 -255   32    0    0  DARK RED
###### END OF PALETTE

 SEACLOUD    0  11  10  11  12  13  14  15   8   1   2   3   4   5   6   7 #15
    0             10   BACKGROUND AND WRITE COLOUR 1 DARKEST BLUE
    1             10   (SUPPRESSES ALL TEXT OUTPUT)
$2 TO 15 UNUSED, NO DEFINITION
   16    0   40   40   CYAN BELOW RANGE
   17    8    8    8   GREY
  -92    8   63    8   -GREEN
 -168   63   63    0   -YELLOW
 -212   45   31   10   -BROWN
 -254   50   00   00   -RED
  255   40    0   40   MAGENTA: ABOVE RANGE
#### ENDE OF VGA CONFIGURATIONS
```

(↪ see next page)

continued

```
$Section 4: define hardcopy palette (VSVGA)
----- hardcopy palettes ------
$VSVGA.COM driver uses colour number, value (2I5) for COLOR.MAP output
$numbers in ascending order, missing ones will be interpolated
```

Last number must be 255. A value of 0 is white, 63 gives darkest black. For coloured output: don not define a hardcopy palette!

```
GREYSCALE     (for POSTSCRIPT black on white)
    0    0    ;WHITE
    1   63    ;BLACK
    2   50    ;dark grey
   15   20    ;light grey
$-- end of pens ---------
   16    0
  255   63
##### END

REVERSE       used for diapositives white on black
    0   63    ;BLACK
    1    0    ;WHITE
    2   20    ;light grey
   15   50    ;dark grey
$-- end of pens ---------
   16   63
  255    0
##### END
```

Configurations old style for dot matrix printers follow. Can be removed, not used by VSVGA.COM.

```
EPSON24       3  1.0 (vertical bytes and aspect)
$ 24 needles  NEC P5,P6
$ GRAPHIC HEADER:  [CR LF], UNIDIRECTIONAL [ESC U (1)]
HEAD         13   10   27   85    1
$IMPRINT:  ESC J (N) CR   with N=24, 24/180 " LINE FEED
$          ESC * (39) N1,N2  DISTANCE BETWEEN NEDLES IS 1/180 "
IMPR         27   74   24   13   27   42   39   LO   HI
$TRAILER:  [LF LF],  BIDIRECTIONAL [ESC U (0)], CR
TRAIL        10   10   27   85    0   13
#### END 24 NEEDLES
```

(↪ see next page)

continued

```
NEC16IBM      2  1.0
$ NEC-P2,P3 (IBM-INTERFACE)
$ GRAPHIC HEADER:  [CR LF], UNIDIRECTIONAL [ESC U (1)]
HEAD          13   10   27   85    1
$IMPRINT: [ESC J (N)]  CR, FEED N/240"
$VERTICALER PIN DISTANCE 1/120 "  --> N=32 (ADAPTED: 30)
$ [ESC I (N2) (N1)]   COLUMMNS= N1*256+N2 (896)
IMPR          27   74   30   27   73   LO   HI
TRAIL         10   10   27   85    0   13
#### END 16 NEEDLES

9_NEEDLES     1  2.0  old EPSONs (EMULATED ON 24-needle dot printer)
$ GRAPHIC HEADER:  [CR LF], UNIDIRECTIONAL [ESC U (1)]
HEAD          13   10   27   85    1
$IMPRINT:  ESC J (N) CR,    N=24, 24/180 " FEED
$          ESC * (39) N1,N2     DISTANCE 1/180 "
IMPR          27   74   24   13   27   42    1   LO   HI
$TRAILER:  [LF LF],  BIDIRECTIONAL [ESC U (0)], CR
TRAIL         10   10   27   85    0   13
#### END OF FILE  (VGA.CFG example for VSVGA.COM)
```