Graphik auf VESA Basis

Kernstück einer Plot-Bibliothek

Rainer Ratke

September 1996

Inhaltsverzeichnis

1 Übersicht

1.1 Vorwort

Die IBM VGA (beides Warenzeichen der International Business Machines Corporation) sind zum de-facto Standard der PC Graphikwelt avanciert. Zahllose Hersteller bieten Super-VGAs an, mit BIOS- oder Registerkompatibilität zur IBM VGA und diversen Erweiterungen des VGA-Standards. Diese reichen von höherer Auflösung über größere Farbtiefe hin bis zu schnellerer Abarbeitung. Die neuesten Entwicklungen auf diesem Sektor sind die sogenannten Graphikbeschleuniger. Zum Nutzen des Anwenders hat scharfer Wettbewerb das Kosten/Leistungsverhältnis dramatisch verändert.

Andererseits ist der Anwender vor das nichttriviale Problem gestellt, diese neuen SuperVGA Produkte auch auszunutzen. Die Hardwareimplementationen halten sich an keinen Standard. Nur sehr wenige Software kann die Möglichkeiten der SuperVGAs sofort und direkt ansprechen. Der VESA-Standard für ein erweitertes VGA-BIOS zeigt einen Weg aus dem Dilemma. Dieses allgemeine Softwareinterface zu SuperVGAs wird von fast allen Herstellern unterstützt und erlaubt einfachsten Zugriff auf den weiten Bereich neuester und zukünftiger VGA-Erweiterungen.

1.2 VESA Graphik Paket

Das hier beschrieben Graphikpaket stellt ein Interface dar zu den 256-Farbmodi bei Auflösungen von 320*200 und höher (bis 1280*1024 sofern in einer VGA schon vorgesehen), dabei wird ein Graphikbeschleuniger nicht angesprochen. Es kann auf jeder VGA angewendet werden, wenn ein VESA-Treiber zur Karte mitgeliefert wurde. Das Ganze besteht aus vier Teilen:

- VSVGA.COM ist ein TSR-Programm, daß die Graphikkarte testet und Interrupt 90h umleitet. Dieser, ursprünglich für ROM-Basic reserviert, wird als Schnittstelle von den Graphikprimitiva zur Hardware benutzt.
- Der Konfigurationsdatei VGA.CFG, die im aktuellen Verzeichnis und / oder der Wurzel der Platte C: liegen muß.
- Graphikprimitiva sind der erste Teil von CALCOMP.LIB. Eine Reihe grundlegender Graphikfunktionen kann hier von LAHEY Fortran aufgerufen werden, sie werden ergänzt durch einige Routinen für die Maus.
- CALCOMP.LIB wird ergänzt durch einen Subset von Zeichenroutinen, vor langer Zeit von CALIFORNIA COMPUTER PRODUCTS entworfen und beschrieben. Namens- und Parameterkonventionen folgen diesem CALCOMP-Standard; einige Erweiterungen wurden zugefügt.

2 Zur Nutzung des VESA Graphikpakets

Für eine graphische Anwendung sind zu laden:

- Der passende VESA-Treiber zur VGA-Karte. (Beispielsweise durch Einfügen von TLIVESA.EXE in AUTOEXEC.BAT, wenn eine Tseng Labs ET4000 oder dazu kompatible Karte vorliegt.)
- VSVGA.COM wonach der Bildschirm etwa folgendes zeigt:

```
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
```

  Erläuterung: `imode=0` ist immer vorhanden da IBM Standardmodus. Alle anderen `imode`s werden von dem vorhandenen VESA-Interface unterstützt. Anm.: VSVGA in der vorliegenden Version funktioniert nicht korrekt, falls winsize kleiner 64 KB ist. Ob ein Modus angezeigt wird, hängt ausschließlich von der vorhandenen SuperVGA und dem VESA-Treiber ab!

  Das TSR-Programm VSVGA.COM kann bei Bedarf später aus dem Speicher entfernt werden durch das Kommando

  VSVGA U(nload)

- Die Konfigurationsdatei VGA.CFG muß in das aktuelle Laufwerk und / oder die Wurzel der Festplatte C: kopiert werden.

Eine Fortran-Anwendung – mit LAHEY F77L3 übersetzt und mit CALCOMP.LIB zusammengebunden – kann alle Modi nutzen, die von VSVGA.COM angezeigt wurden. Es kann Aufrufe enthalten:

- Nur der Primitiva. Alle Koordinaten sind dann auf Pixelbasis bezogen. Ursprung ist immer die linke untere Bildschirmecke.
- Nur der CALCOMP-Routinen. Maßeinheit sind dann cm, Zoll oder eine beliebige andere Einheit. Das Bild wird so skaliert, daß der Schirm voll ausgenutzt aber von der Zeichnung nichts abgeschnitten wird. Ursprung ist immer die linke untere Bildecke.
- Primitiva und CALCOMP-Routinen gemischt. Farben, Schreibmodi und ähnliches können dann durch Aufruf einer Primitivroutine in der CALCOMP-Umgebung verändert werden – mit Vorsicht. Wird ein Fenster festgelegt, können auch die CALCOMP-Routinen nicht mehr auf Pixel außerhalb zugreifen.

3 Demonstration des VESA Graphikpakets

Nachdem der spezielle VESA Treiber und VSVGA.COM geladen sind wie vorstehend beschrieben, kann das Demoprogramm GDEMO.EXE gestartet werden. Ist VGA.CFG vorhanden, entweder im aktuellen Laufwerk und / oder in der Wurzel von Platte C: ? Das Programm fragt zuerst nach der Modusnummer. Eingabe von 0 oder einer größeren Zahl bis zu max. mode wählt die gewünschte Auflösung an. Das Programm kommt wieder an diesen Punkt, sodaß alle Modi getestet werden können. Eingabe von -1 hier beendet das Programm.
GDEMO zeigt zuerst die Primitiva: Punkte, Linien, Füllmuster, Animation, Schreibmodi usw. Jeder Teil läuft, bis eine Taste gedrückt wird. Anschließend werden einige CALCOMP-Beispiele gezeigt:

- Symboltabelle der `SYMBOL`-Routine.
- Text, von `SYMBOL` in unterschiedlichen Größen, Winkeln und Neigungen gezeichnet.
  (ESC beendet diesen Teil, andre Tasten bewirken ein neues Textbeispiel.)
- Eine Welle, online mit Finiten Differenzen berechnet. (Durch Tastatureingabe kann das Programm angehalten und weitergestartet werden, mit ESC bricht es ab.)
- Netz und Ergebnisse einer Finite-Element-Berechnung (Grundwasser).

Wenn eine Zeichnung beendet ist, erscheint der blinkende Text: Done. Press ENTER [PRTSC]. Wenn nun gedrückt wird

- Print Screen: Wird eine Hardcopy angefordert und der Flackertext entsprechend gekürzt.
- ENTER: Das Videomemory wird auf Datei ausgegeben, falls eine Vormerkung für Hardcopy vorliegt. Der Bildschirm wird gelöscht und das Programm setzt fort.
- eine andere Taste: Der Flackertext verschwindet und kann mit nochmaliger Eingabe wieder hergestellt werden.

4 CALCOMP.LIB – **Beschreibung**

Abkürzungen: `I` und `R` bezeichnen Integer- bzw. Real-Parameter, eventuell gefolgt von `*2` oder `*4`, um die exakte Wortlänge anzugeben. Reals sind immer einfach genau. Eine Dimensionsangabe in runden Klammern weist auf ein Feld als Parameter hin.
Arg bezeichnet einen nur als Argument benötigten Parameter, dem vor Aufruf ein Wert zugewiesen worden sein muß. Der Wert bleibt unverändert. Res bezeichnet Resultatparameter. Mit A/R werden Parameter markiert, die beiden Zwecken dienen.
Die Beschreibungen sind der Übersichtlichkeit halber in knapper tabellarischer Form gehalten, mehr als Kurzreferenz für den erfahrenen Programmierer denn als mühsames Lehrbuch für den Einsteiger gedacht.

4.1 Grundroutinen: Die Primitiva

Die Unterprogramme und Funktionen dieses Unterabschnitts sind die Schnittstelle zwischen LAHEY Fortran und der Graphikkarte. Sie rufen INT 90h auf, dorthin und von dort wird alle notwendige information nur über 16-Bit-Register übertragen. INT 90h liegt innerhalb der residenten Teils von VSVGA.COM. Hier sind schnelle Unterfunktionen zum direkten Schreiben ins Videomemory implementiert, die auch die VESA-Funktionen benutzen.

Allgemeines: Integer-Parameters sind sämtlich intern als `INTEGER*2` deklariert. Werte in die Routinen hinein dürfen aber vom Typ `INTEGER*4` sein, da nur die niederwertigen 16 Bit angesprochen werden. Solche Argumente erzeugen keine Fehler. Nichtnegative Ergebnisse sind korrekt, wenn eine zuvor auf Null gesetzte `INTEGER*4`-Variable als Aktualparameter benutzt wird. Sonst muß immer eine `INTEGER*2`-Variable gebraucht werden. `REAL`-Parameter kommen in diesem Teil nicht vor.

Zweck: Graphikmodus setzen SUBROUTINE: **G_MODE**

Aufruf: `CALL G_MODE (IMODE)`

Parameter	Typ	A/R	Bedeutung			
IMODE	I	Arg	Auflösung:	< 0	:	Rückkehr zum Textmodus
				0	:	320*200
				1	:	640*400
				2	:	640*480
				3	:	800*600
				4	:	1024*768
				5	:	1280*1024

Erläuterung: `G_MODE` stellt die Graphikumgebung im gewählten Modus bereit. Wird der Modus nicht unterstützt, wird der nächstmögliche kleinere benutzt.

Zweck: Graphik Initialisierung ENTRY: **G_INIT**

Aufruf: `CALL G_INIT`

Parameter	Typ	A/R	Bedeutung
			– keine –

Erläuterung: Alle internen Einstellungen werden normiert. Die RGB Palette bleibt ungeändert. Fenster ist der gesamte Bildschirm; er wird gelöscht.
`G_INIT` ist in `G_MODE` enthalten.

Zweck: Graphikinformation ENTRY: **G_INFO**

Aufruf: `CALL G_INFO (MAXMOD)`

Parameter	Typ	A/R	Bedeutung
MAXMOD	I	Res	Maximaler Graphikmodus

Erläuterung: `G_INFO` fragt den `VSVGA`-Kern ab. Der `COMMON`-Block `/SVGA_INFO` wird aktualisiert und kann vom aufrufenden Programm ausgewertet werden. Das Fehlerwort `GERROR` wird intern rückgesetzt. Der nächste Aufruf meldet nur neue Fehler.

`COMMON /SVGA_INFO/` ist vereinbart als

```
      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
```

mit den Bedeutungen

`NPIXH, NPIXV`	Anzahl der Pixel horizontal, vertikal
`ACTCOL`	gesetzte Farbe (low byte: Zeichen-, high: Hintergrundfarbe)
`ACTMOD`	aktueller Graphikmodus (0 to max. imode)
`OLDMOD`	voriger (Text-) Modus
`PTHICK`	Strichstärke, nicht unterstützt, immer 1
`NCOLTX, NROWTX`	Anzahl der Spalten/ Zeilen für Text im aktuellen Modus
`X0CLIP, Y0CLIP`	Koordinaten der unteren linken Ecke des Fensters
`NXCLIP, NYCLIP`	Größe des Graphikfensters
`ACTFIL`	eingestellte Füllmusternummer
`ACTLS`	eingestellte Strichstilnummer
`ACTASP`	Aspectverhältnis (*10000) im eingestellten Graphikmodus
`ACTWRI`	eingestellter Schreibmodus
`HCREQ`	wenn > 0, soll Hardcopy ausgegeben werden
`GERROR`	Fehlerwort (0: ok, 1: kleine 2: schwere Fehler erkannt)

Anm.: Im Textmodus wird außer `MAXMOD` und `GERROR` keine gültige Information erhalten. Siehe auch: **G_CONFIG**.

Zweck: Graphikbildschirm löschen ENTRY: **G_CLS**

Aufruf: `CALL G_CLS`

Parameter	Typ	A/R	Bedeutung
			– keine –

Erläuterung: Der gesamte Bildschirm wird gelöscht.
`G_CLS` ist in `G_INIT` und `G_MODE` enthalten.

Zweck: Graphikfenster setzen ENTRY: **G_SETCLIP**

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

Parameter	Typ	A/R	Bedeutung
`IX1`	`I`	Arg	linke Fenstergrenze
`IY1`	`I`	Arg	untere
`IX2`	`I`	Arg	rechte
`IY2`	`I`	Arg	obere

Erläuterung: Nachher kann kein Pixel außerhalb der Grenzen mehr angesprochen werden. Die untere linke Ecke `(IX1,IY1)` wird der neue Koordinatenursprung.

Zweck: Bewegung auf Punkt ENTRY: **G_MOVE**

Aufruf: `CALL G_MOVE (IX,IY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate
IY	I	Arg	y-Koordinate

Erläuterung: Das aktuelle Pixel wird an der Stelle `(IX,IY)` gesetzt in vorgewählter Farbe, Schreibmodus und Strichstil. Wird gebraucht, um den ersten Punkt eines Polygonzugs zu zeichnen.

Zweck: Zeichne Linie zu Punkt ENTRY: **G_DRAW**

Aufruf: `CALL G_DRAW (IX,IY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate
IY	I	Arg	y-Koordinate

Erläuterung: Eine Gerade vom aktuellen Pixel (ohne dieses) nach `(IX,IY)` wird gezeichnet in vorgewählter Farbe, Schreibmodus und Strichstil. `(IX,IY)` wird das neue aktuelle Pixel.

Zweck: RGB-Werte für einzelne Farbe ENTRY: **G_SETRGB**

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

Parameter	Typ	A/R	Bedeutung
NO	I	Arg	Farbnummer, 0 bis 255
IR	I	Arg	Rotanteil, 0 bis 63
IG	I	Arg	Grünanteil, 0 bis 63
IB	I	Arg	Blauanteil, 0 bis 63

Erläuterung: Die RGB-Werte für Farbnummer `NO` treten sofort inkraft für Pixel, die schon in dieser Farbe gezeichnet sind (Animation).
Anm.: In den 256-Farbmodi sind gewöhnlich nicht alle Farben vordefiniert.

Zweck: Farbe wählen ENTRY: **G_COLORS**

Aufruf: `CALL G_COLORS (IDRAW,IBACK)`

Parameter	Typ	A/R	Bedeutung
IDRAW	I	Arg	Zeichenfarbe, 0 bis 255
IBACK	I	Arg	Füllfarbe, 0 bis 255

Erläuterung: Nachfolgende Graphikaufrufe zeichnen mit diesen Farben.

Zweck: Strichstil setzen ENTRY: **G_LINESTYLE**

Aufruf: `CALL G_LINESTYLE (NO,USER)`

Parameter	Typ	A/R	Bedeutung
NO	I	Arg	Strichstilnummer, 0 bis 4
USER	I	Arg	benutzerdefinierter Strichstil für NO=4

Erläuterung: Stile 0 bis 3 sind vordefiniert. Stilnummer 0 gibt durchgehende Linien. Für Stil 4 definieren die Bits von USER das Muster: Pixel werden mit Zeichenfarbe angelegt, wenn die korrespondierenden Bits gesetzt sind, sonst mit Füllfarbe.

Zweck: Rechteck füllen ENTRY: **G_PATTERN**

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

Parameter	Typ	A/R	Bedeutung
IX1	I	Arg	x-Koordinate der 1. Ecke
IY1	I	Arg	y-Koordinate der 1. Ecke
IX2	I	Arg	x-Koordinate der 2. Ecke
IY2	I	Arg	y-Koordinate der 2. Ecke
NO	I	Arg	Füllstilnummer, <0 bis 12
USER	I(4)	Arg	Benutzerfüllstil

Erläuterung: Das Rechteck wird gefüllt mit dem Muster NO. NO=0 füllt vollständig mit Zeichenfarbe, 1 mit Füllfarbe. Diverse Muster sind vordefiniert durch NO 2 bis 11. Für Stilnummer <0, muß das INTEGER*2-Feld USER 8 Bytes für das Muster bereitstellen. Die 8*8 Bit zeigen Zeichen- bzw. Füllfarbe für einen 8*8 Pixelblock an. Das Muster wird intern gespeichert und kann später als Füllmuster 12 wiederverwendet werden.

Zweck: Text Interface ENTRY: **G_TEXT**

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

Parameter	Typ	A/R	Bedeutung
`IX`	`I`	Arg	x-Koordinate der linken unteren Textecke
`IY`	`I`	Arg	y-Koordinate
`IFACT`	`I`	Arg	Vergrößerungsfaktor: 1 bis 8
`ISWITCH`	`I`	Arg	< 0: Koordinaten und Faktor speichern, keine Ausgabe. $= 0$: Text schreiben ab `(IX,IY)` mit Vergrößerung `IFACT`. > 0: `(IX,IY)` und `IFACT` ignorieren. Ausgabe ab alter Textposition mit altem Faktor.
`TEXT`	`CHAR`	Arg	Zu schreibender Text

Erläuterung: Textposition und Faktor werden intern mitgeführt. Der 8*8 ROM-Zeichensatz wird benutzt. Zeichen- und Füllfarbe geben Text und Hintergrund.
Anm.: Es kann mit `ISWITCH` > 0 fortlaufend geschrieben werden.

Zweck: Zeichnen einer Pixelzeile ENTRY: **SET_PIX**

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

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate des linkesten Pixels
IY	I	Arg	y-Koordinate der Zeile
NPIX	I	Arg	Anzahl der Pixel
ICOL	I	Arg	Farbe für Zeile

Erläuterung: Es wird eine horizontale Reihe aus `NPIX` Pixeln in Farbe `ICOL` gezeichnet.

Zweck: Bestimme Pixelfarbe ENTRY: **GET_PIX**

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

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate des Pixels
IY	I	Arg	y-Koordinate
ICOL	I	Res	Colour

Erläuterung: `ICOL` wird die gesuchte Farbnummer des Pixels.

Zweck: Hardcopy: Bildschirmabzug ENTRY: **G_DUMP**

Aufruf: `CALL G_DUMP (LFN)`

Parameter	Typ	A/R	Bedeutung
LFN	I	Res	Nummer der Abzugsdatei (<0 bei Fehler)

Erläuterung: Das Video-Memory wird zeilenweise, 1 Byte je Pixel, auf Datei geschrieben. Die Dateinamen sind Hexadezimalzahlen 0000 bis 7FFF mit der Erweiterung .RAW. Die Routine ermittelt einen freien Namen. Die Zahl wird in LFN rückgemeldet, wenn der Abzug erfolgreich ausgeführt werden konnte. Zusammen mit der Datei COLOR.MAP wird die gesamte Videoinformation gespeichert. (Auf dem Markt sind verschiedene Programme erhältlich, die daraus diverse Graphikmetafiles erstellen können – nicht in diesem Paket enthalten.)

Zweck: Schreibmodus setzen ENTRY: **SET_WRMOD**

Aufruf: `CALL SET_WRMOD (MODE)`

Parameter	Typ	A/R	Bedeutung
MODE	I	Arg	Schreibmodus 0 bis 3

Erläuterung: Wirkung der Modi:

0	absolutes Schreiben:	Pixel bekommt aktuelle Farbe
1	XOR-Modus:	Pixelfarbe wird mittels XOR geändert
2	OR-Modus:	Pixelfarbe wird mittels OR geändert
3	transparent:	Vordergrundfarbe wird absolut geschrieben Hintergrundfarbe wird nicht berücksichtigt

Zweck: VGA-Status sichern / wiederherstellen ENTRY: **VGA_STATE**

Aufruf: `CALL VGA_STATE (MODE)`

Parameter	Typ	A/R	Bedeutung
MODE	I	A/R	Argument: 0: Sichern, 1: Wiederherstellen Resultat: 0: Fehler, 1: ok

Erläuterung: Videomodus und RGB-Palette sollen gespeichert werden. Bei zu kleinen internen Puffern wird dieser Fehler durch `MODE` < 1 gemeldet.

Zweck: Konfiguration lesen SUBROUTINE: **G_CONFIG**

Aufruf: `CALL G_CONFIG (PENS,MXMOD)`

Parameter	Typ	A/R	Bedeutung
PENS	I (0:15)	Res	Stifttabelle: Farbnummern der ersten 16 Stifte (für `NEWPEN`)
MXMOD	I	A/R	< 0 als Argument: nur der Maximalmodus wird in `MXMOD` zurückgegeben Resultat: Maximalmodus in VGA.CFG, -1 bedeutet keine Einschränkung.

Erläuterung: Routine liest die Konfigurationsdatei, die zuerst im aktuellen, dann im Wurzelverzeichnis der Platte C: gesucht wird. VGA.CFG muß in einem von beiden liegen!

Im Textmodus kann nur der Maximalmodus abgefragt werden. Im Graphikmodus werden die Stifttabelle belegt und alle RGB-Register wie in der Konfiguration vorgegeben gesetzt. In Datei COLOR.MAP werden die RGB-Werte gespeichert als Ergänzung zum Speicherabzug.

Bei Erstaufruf werden alle Werte intern gespeichert. Weitere Aufrufe greifen darauf zurück und sind daher schneller.

Anm.: `MXMOD` hier und `MAXMOD` als Parameter von `G_INFO` nicht verwechseln! Letzterer gibt den höchsten Modus an, der von VGA und VESA-Treiber unterstützt wird; `MXMOD` muß in VGA.CFG gesetzt werden, um beispielsweise einen Monitor vor für ihn zu hohe Modi zu schützen.

Zweck: Bogenstück zeichnen SUBROUTINE: **G_ARC**

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

Parameter	Typ	A/R	Bedeutung
MX	I	Arg	x-Koordinate des Zentrums
MY	I	Arg	y-Koordinate des Zentrums
IR1	I	Arg	Distanz vom Zentrum zum Bogenanfang
IR2	I	Arg	Distanz vom Zentrum zum Bogenende
IA1	I	Arg	Winkel in Grad vom Zentrum zum Bogenanfang
IA2	I	Arg	Winkel in Grad vom Zentrum zum Bogenende

Erläuterung: Koordinaten und Entfernungen in Pixelangaben. Winkel sind positiv im mathematisch positiven Sinne (Gegenuhrzeigersinn), mit 0.0 in Richtung der x-Achse. Schreibmodus und Linienstil werden berücksichtigt. Ist das Ansichtsverhältnis von 1.0 verschieden, hat nur die x-Richtung die wahren Längen, y-Entfernungen werden modifiziert.

`CALL G_ARC (MX,MX,25,25,0,360)` ergibt einen Vollkreis mit Durchmesser 50 um `(MX,MY)`. Er sieht auch wie ein Kreis aus!

Zweck: Ellipse zeichnen SUBROUTINE: **G_ELLIPS**

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

Parameter	Typ	A/R	Bedeutung
XM	I	Arg	x-Koordinate des Zentrums
YM	I	Arg	y-Koordinate des Zentrums
DX	I	Arg	Durchmesser in x-Richtung
DY	I	Arg	Durchmesser in y-Richtung
B	I	Arg	Randdicke
FILL	I	Arg	Füllmuster, < 0 :ungefüllt

Erläuterung: Koordinaten und Durchmesser auf Pixelbasis. Y-Entfernungen werden durch das Ansichtsverhältnis modifiziert. Der Rand wird mit Zeichenfarbe `B` Pixel breit gezeichnet. Bei nichtnegativem `FILL` wird das Innere entsprechend Füllstil gefüllt.

4.2 Hilfsroutinen

Zweck: Tastaturabfrage INTEGER*2 FUNCTION: **INKEY**

Aufruf: `CALL INKEY (WAIT)`

Parameter	Typ	A/R	Bedeutung
`WAIT`	`I`	Arg	0: Kein Warten auf Eingabe sonst: Auf Tastendruck warten und lesen

Erläuterung: Ergebnis von `INKEY` ist:

- 0: keine Eingabe erfolgt (nur bei `WAIT` =0)
- > 0: `ASCII`-Code
- < 0: negativer Scancode einer erweiterten Tastenkombination

4.3 Mausunterstützung

Zweck: Maus initialisieren SUBROUTINE: **MS_INIT**

Aufruf: `CALL MS_INIT (ERROR)`

Parameter	Typ	A/R	Bedeutung
ERROR	L	Res	Fehler, wenn .TRUE.

Erläuterung: Zuerst wird nachgesehen, ob ein Maustreiber vorhanden ist (`ERROR` wird `.TRUE.`, wenn er fehlt.) Sonst wird die aktuelle Größe des Bildschirms ermittelt und ein Fadenkreuz in seine Mitte plaziert. Der gesamte Bildschirm steht für Mausbewegungen zur Verfügung und die Mauskoordinaten (0,0) zeigen auf seine linke untere Ecke.

Zweck: Mausposition setzen SUBROUTINE: **MS_POS**

Aufruf: `CALL MS_POS (IX,IY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate der Maus
IY	I	Arg	y-Koordinate der Maus

Erläuterung: Die Koordinaten werden geprüft, gegebenenfalls an die Bildschirmgröße angepaß und die Mausposition an den Maustreiber weitergereicht. Der Fadenkreuzcursor bleibt unverändert!

Zweck: Mausabfrage SUBROUTINE: **MS_KEY**

Aufruf: `CALL MS_KEY (LEFT,RIGHT,IX,IY)`

Parameter	Typ	A/R	Bedeutung
LEFT	L	Res	Linke Maustaste gedrückt, wenn .TRUE.
RIGHT	L	Res	Rechte Maustaste gedrückt, wenn .TRUE.
IX	I	Res	x-Koordinate der Maus
IY	I	Res	y-Koordinate der Maus

Erläuterung: Mausposition und gegebenenfalls auch der Cursor werden aktualisiert wie durch den letzten Aufruf von MS_POS und / oder Mausbewegungen definiert. Die beiden ersten Parameter zeigen an, welche der Maustasten seit dem letzten Aufruf betätigt wurden, die anderen geben die aktuelle Mausposition zurück.

Zweck: Fadenkreuz setzen SUBROUTINE: **MS_CURSOR**

Aufruf: `CALL MS_CURSOR (IX,IY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Cursorkoordinate
IY	I	Arg	y-Cursorkoordinate

Erläuterung: Bei IX $\geq$ 0 wird ein Fadenkreuz im XOR-Modus an diese Position gezeichnet, nach Löschen eines eventuell zuvor gezeichneten Fadenkreuzes. Die Lage des neuen Fadenkreuzes wird intern in der Routine festgehalten.

Bei IX = −1 wird diese intern gespeicherte Position zu "undefiniert" gesetzt.
IY steuert dabei das Löschen eines Fadenkreuzes:
−1: nichts zu löschen (bei Cursorinitialisierung)
0: Fadenkreuz war gezeigt und soll gelöscht werden.

Zweck: Fadenkreuz setzen ENTRY: **MS_0CURS**

Aufruf: `CALL MS_0CURS (IX,IY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Cursorkoordinate
IY	I	Arg	y-Cursorkoordinate

Erläuterung: Wie `MS_CURSOR`, jedoch wird der Aufruf ignoriert, wenn momentan kein Fadenkreuz gezeichnet ist. Die Position bleibt dann "undefiniert".

Zweck: Fadenrechteck zeichnen SUBROUTINE: **MS_BOX**

Aufruf: `CALL MS_BOX (IX,IY,JX,JY)`

Parameter	Typ	A/R	Bedeutung
IX	I	Arg	x-Koordinate der ersten Ecke
IY	I	Arg	y-Koordinate der ersten Ecke
JX	I	Arg	x-Koordinate der zweiten Ecke
JY	I	Arg	y-Koordinate der zweiten Ecke

Erläuterung: Die erste Ecke ist beliebig und die zweite liegt ihr diagonal gegenüber. Bei `IX` $\geq$ `0` zeichnet `MS_BOX` das so definierte Rechteck im `XOR`-Modus, nach Löschen eines eventuell zuvor gezeichneten. Die Lage des neuen Rechtecks wird intern in der Routine festgehalten.

Bei `IX = -1` wird diese intern gespeicherte Position zu "undefiniert" gesetzt.
`IY` steuert dabei das Löschen eines Rechtecks:
`-1`: nichts zu löschen (Initialisierung)
`0`: Eine Box war gezeigt und soll gelöscht werden.

Zweck: Ausschnitt öffnen, Version 1 SUBROUTINE: **MS_1WIN**

Aufruf: `CALL MS_1WIN (RIGHT,IX,IY,JX,JY)`

Parameter	Typ	A/R	Bedeutung
RIGHT	L	Res	`.TRUE.`: rechte Maustaste gedrückt, Ende
IX	I	Res	Kleinste x-Koordinate
IY	I	Res	Kleinste y-Koordinate
JX	I	Res	Größte x-Koordinate
JY	I	Res	Größte y-Koordinate

Erläuterung: Beim ersten Aufruf wird zunächst im linken unteren Bildschirmviertel eine kleine Box gezeigt, die mit der Maus auf passende Größe aufgezogen werden kann. Klicken der linken Maustaste beendet das Vergrößern. Die Box kann nun verschoben werden; ein weiterer Druck auf die linke Maustaste beendet die Routine. Die Koordinaten `IX, IY, JX` und `JY` beschreiben die letzte Fensterposition.

Folgeaufrufe erlauben weiteres Verschieben der Box bei unveränderter Größe und Rückgabe von Koordinaten wie oben.

`MS_1WIN` soll so oft wiederholt werden, bis der Benutzer durch Druck auf die rechte Maustaste ein Ende signalisiert. Das Fenster verschwindet dann und die Routine befindet sich wieder im Anfangszustand.

Zweck: Ausschnitt öffnen, Version 2 SUBROUTINE: **MS_2WIN**

Aufruf: `CALL MS_2WIN (RIGHT,IX,IY,JX,JY)`

Parameter	Typ	A/R	Bedeutung
RIGHT	L	Res	`.TRUE.`: rechte Maustaste gedrückt, Abbruch
IX	I	Res	Kleinste x-Koordinate
IY	I	Res	Kleinste y-Koordinate
JX	I	Res	Größte x-Koordinate
JY	I	Res	Größte y-Koordinate

Erläuterung: `MS_2WIN` definiert ein Rechteck durch Anklicken zweier Eckpunkte (linke Taste). Es wird eine kleine Box gezeigt, deren linke untere Ecke zuerst auf den einen Eckpunkt zu positionieren ist. Nach erstem Tastendruck bleibt sie fest und die diagonal gegenüberliegende Ecke wird auf gleiche Weise festgelegt.

Bei Druck auf die rechte Maustaste erfolgt abbruch, die Koordinaten sind nicht gültig.

Achtung: Vor Aufruf soll der Cursor positioniert und gezeigt werden, möglichst in Nähe der Stelle, wo eine Ausschnittsecke erwartet wird. Die Box wird bei Verlassen der Routine gelöscht und der Fadenkreuzcursor erscheint wieder.

Zweck: Maus-Interrupt SUBROUTINE: **MS_INT**

Aufruf: `CALL MS_INT (REGS)`

Parameter	Typ	A/R	Bedeutung
REGS	`I*4(9)`	A/R	Feld für Registerblock

Erläuterung: Reine Hilfsroutine, um `INT 33h (51 dez.)` aufzurufen. Version: LAHEY F77L3

4.4 CALCOMP-kompatible Routinen

Die Unterprogramme dieses Abschnitts sind die Grundlage einer Bibliothek im Stile von CALCOMP. Sie rufen nie den `VESA`-Treiber direkt auf sondern benutzen die Graphikprimitiva des vorigen Abschnitts.

Aufruffolge: Die Reihenfolge zum Erstellen einer Graphik sieht etwa wie folgt aus:

- `CALL PLOTS (...)` um die Auflösung festzulegen. Kann fehlen, dann wird in maximaler Auflösung gezeichnet.
- `CALL NEWPLOT (...,XL,YL,...)` Zeichnungsgröße festlegen.
  `CALL PLOT (XL,YL,1)` hat gleiche Wirkung. Beide Aufrufe schalten in den Graphikmodus.
- Aufruf anderer Graphikroutinen .....
- `CALL PLOT (...,...,0)` oder `CALL PLOT (...,...,999)` beenden die Zeichnung. Das Ende wird angezeigt und das Bild bleibt bis zur Eingabe von ENTER stehne. Es gibt Gelegenheit, einen Pixeldump zu erzeugen.
  `CALL PLOT (XL,YL,1)` oder `CALL NEWPLOT (...,XL,YL,...)` sind gleichwertig dazu, initieren aber eine neue Zeichnung, ohne in den Textmodus zurückzukehren.
- `PLOTS` und `NEWPLOT` können beliebig oft verwendet werden.

Parametertypen: Alle Real-Parameter sind `REAL*4`. Fast alle Integer-Parameter sind `INTEGER*4`, einige Hilfsroutinen nutzen jedoch `INTEGER*2` wegen der Schnelligkeit. Solche Parameter sind mit `I*2` in den Beschreibungen gekennzeichnet, `INTEGER*4`-Werte können nur als aktuelle **Argumente** benutzt werden!

Zweck: Hauptplotroutine SUBROUTINE: **PLOT**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Arg	x-Koordinate oder Länge
YPAGE	R	Arg	y-Koordinate oder Länge
IPEN	I	Arg	Stiftcode

Erläuterung: `IPEN` bestimmt die auszuführende Aktion:

IPEN	Aktion
0	(`XPAGE, YPAGE` irrelevant) Zeichnungsende
1	Grafikstart mit virtueller Blattgröße `XPAGE * YPAGE`. Die Zeichnung wird so skaliert, daß der Bildschirm ausgenutzt wird. Koordinatenursprung liegt in der unteren linken Zeichnungsecke. Alle internen Voreinstellungen werden normalisiert.
2	Zeichnen absolut: Gibt Linie vom letzten Punkt zu `(XPAGE, YPAGE)`.
3	Bewegen absolut: Nur die Stiftposition wird zu `(XPAGE, YPAGE)`.
-2, -3	Wie 2, 3. Der Ursprung wird nach `(XPAGE, YPAGE)` verschoben.
12, 13	Zeichnen / Bewegen relativ: `XPAGE, YPAGE` sind Änderungen.
-12, -13	Wie 12, 13. Verschiebung des Ursprungs wie bei -2, -3.
4	(`XPAGE, YPAGE` irrelevant) absoluten Schreibmodus einstellen.
5	(`XPAGE, YPAGE` irrelevant) XOR-Schreibmodus einstellen.
6	(`XPAGE, YPAGE` irrelevant) OR-Schreibmodus einstellen.
7	(`XPAGE, YPAGE` irrelevant) transparenten Schreibmodus einstellen.
999	Gleiche Wirkung wie 0

Anm.:

a) Anstelle `XPAGE` oder `YPAGE` kann 999.0 angegeben werden bei `IPEN` gleich -2, -3, 2, 3,-12, -13, 12 oder 13. Die entsprechende Koordinate bleibt dann unverändert. In allen anderen Fällen müssen diese Parameter < 999.0 sein!

b) `PLOT` schneidet alle über das Graphikfenster hinausragenden Zeichnungsteile sauber ab. Ursprung und aktuelle Stiftposition sind jedoch immer richtig, eventuell auch außerhalb des Fensters.

Zweck: Auflösung wählen ENTRY: **PLOTS**

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

Parameter	Typ	A/R	Bedeutung
`I1`	`I`	Arg	unbenutzt
`I2`	`I`	Arg	unbenutzt
`MODE`	`I`	Arg	Modusnummer (0 – 5) für die nächsten Zeichnungen

Erläuterung: Siehe auch `G_MODE`.

Zweck: Stiftwahl ENTRY: **NEWPEN**

Aufruf: `CALL NEWPEN (NPEN)`

Parameter	Typ	A/R	Bedeutung
`NPEN`	`I`	Arg	Stiftnummer für die nächsten Zeichenbefehle

Erläuterung: `NPEN` ist Füllstiftnummer *256 + Zeichenstiftnummer (oder nur Zeichenstiftnummer, wenn Hintegrund in Farbe 0 gewünscht). Stiftnummern < 16 werden mittels der Stifttabelle konvertiert, größere sind schon Farbnummern.

Zweck: Stiftfarben bestimmen ENTRY: **CHKPEN**

Aufruf: `CALL CHKPEN (NPEN,ICOLOR)`

Parameter	Typ	A/R	Bedeutung
NPEN	I*2	Arg	Stiftnummer(n)
ICOLOR	I*2	A/R	Farbnummer(n)

Erläuterung: `NPEN` wird anhand der Stifttabelle konvertiert, wenn die Bytes für Zeichnen und Füllen > 15 sind. Ist `ICOLOR` als Argument Null, erhält es als Ergebnis nur die Zeichenfarbe.

Zweck: Transformation, Konstanten ENTRY: **KOORZZ**

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

Parameter	Typ	A/R	Bedeutung
AX	R	Res	siehe unten
BX	R	Res	siehe unten
AY	R	Res	siehe unten
BY	R	Res	siehe unten
LX1	I*2	Res	Minimum x-Koordinate, immer 0
LX2	I*2	Res	Maximum x-Koordinate
LY1	I*2	Res	Minimum y-Koordinate, immer 0
LY2	I*2	Res	Maximum y-Koordinate

Erläuterung: Bildschirmkoordinaten `(IX,IY)` des `(X,Y)` nächstliegenden Pixels erhält man mit:

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

Zweck: Aktuelle Stiftposition ENTRY: **WHERE**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Res	Aktuelle x-Koordinate
YPAGE	R	Res	Aktuelle y-Koordinate
FACT	R	Res	Aktueller Faktor, unbenutzt, immer 1.0

Erläuterung: Koordinaten relativ zur unteren linken Ecke des virtuellen Zeichnungsblattes, nicht auf eventuell verschobenen Ursprung bezogen!

Zweck: Vergößerung ENTRY: **FACTOR**

Aufruf: `CALL FACTOR (FACT)`

Parameter	Typ	A/R	Bedeutung
FACT	R	Arg	Faktor

Erläuterung: Unbenutzt. `FACT` ist ohne Wirkung. Siehe `NEWPLOT`.
Wegen Kompatibilität zur Stiftplotterversion bereitgestellt.

Zweck: Koordinatenkonvertierung ENTRY: **CHK999**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Arg	x-Koordinate oder 999.0
YPAGE	R	Arg	y-Koordinate oder 999.0
IPEN	I	Arg	Stiftcode
XP	R	Res	x-Koordinate
YP	R	Res	y-Koordinate

Erläuterung: Werte 999.0 oder größer werden in die aktuellen Stiftkoordinaten oder Null gewandelt, je nachdem, ob `IPEN` absolutes oder relatives Zeichnen anzeigt.

Zweck: Neue Zeichnung SUBROUTINE: **NEWPLOT**

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

Parameter	Typ	A/R	Bedeutung
IUNIT	I	Arg	Unbenutzt: (Logische Gerätenummer des Plotters)
IDEV	I	Arg	Unbenutzt: (Plottertyp)
XL	R	Arg	x-Länge der Zeichnung
YL	R	Arg	y-Länge der Zeichnung
IANZ	I	Arg	Unbenutzt: (Abbruch nach `IANZ` Fehlern)

Erläuterung: Wegen Kompatibilität zur Stiftplotterversion bereitgestellt.
`CALL PLOT (XL,YL,1)` wird ausgeführt.

Zweck: Zeichnung beenden SUBROUTINE: **PL_END**

Aufruf: `CALL PL_END`

Parameter	Typ	A/R	Bedeutung
			– keine –

Erläuterung: Wartet auf ENTER, Bildschirmabzug auf Datei bei Anforderung.

Zweck: Erweitertes Symbolunterprogramm SUBROUTINE: **SYMBOL**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Arg	x-Koordinate oder 999.0
YPAGE	R	Arg	y-Koordinate oder 999.0
WIDTH	R	Arg	Symbolgröße
ASCII	CHAR	Arg	Zu zeichnender Text bei NCHAR > 0
IORD	I	Arg	Nummer eines zentrierten Symbols / ASCII-Code
ANGLE	R	Arg	Winkel in Grad
NCHAR	I	Arg	Anzahl Zeichen in ASCII / Code

Erläuterung: Drei verschiedene Aufrufe von SYMBOL sind möglich:

- NCHAR > 0: Text ASCII wird gezeichnet. Ist das erste Zeichen die Unterstreichung, wird es fortgelassen und der ganze Text understrichen. Es wird Proportionalschrift benutzt. WIDTH ist die mittlere Zeichenbreite und -Höhe. XPAGE, YPAGE bezeichnen die untere linke Ecke des Textes. Anschließend ist die Stiftposition rechts vom Text.
- NCHAR $= 0$: Ein Einzelzeichen mit ASCII-Code IORD wird gezeichnet. Stiftposition nacher rechts vom Zeichen.
- NCHAR -2 oder -1: Das zentrierte Symbol mit der Nummer IORD wird gezeichnet. Ist NCHAR gleich -2, mit Linie von der alten Stiftposition zum Zentrum des Symbols. Stiftposition ist nachher das Zentrum bei XPAGE, YPAGE.

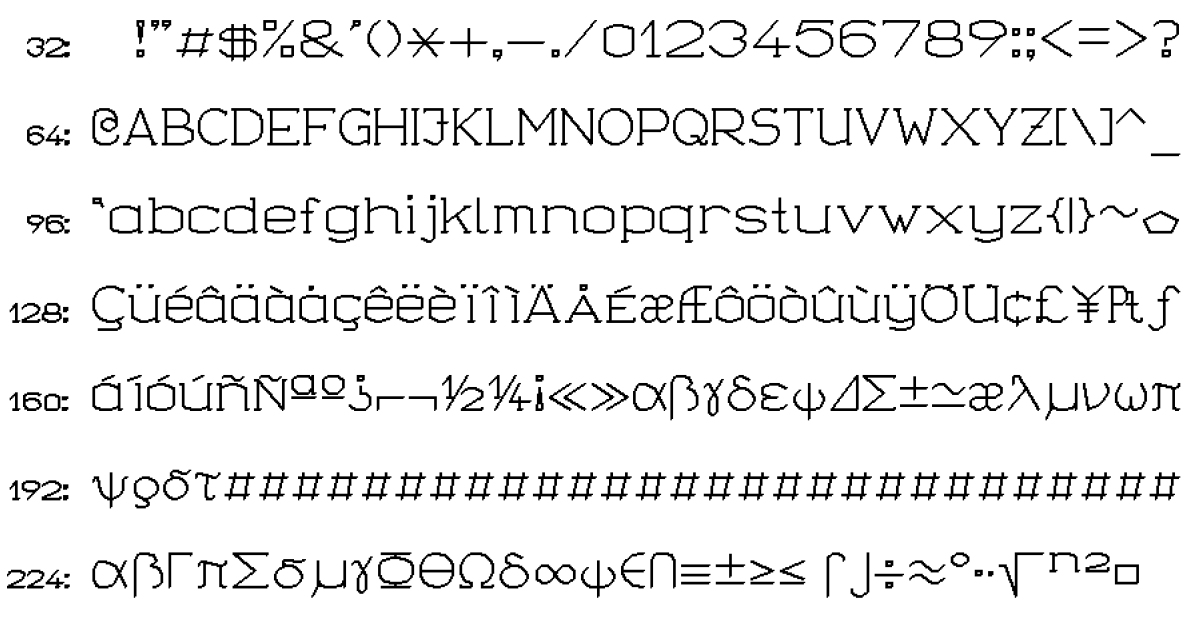
Der IBM PC-Zeichensatz ist implementiert mit speziellen griechisch-technischen anstelle der Graphikzeichen. ASCII-Codes < 32 fehlen. Undefinierte Zeichen ergeben das Nummernzeichen (ASCII 35) im Bild.
Alle Teile eines Zeichens werden nur einfach gezeichnet, sodaß SYMBOL auch im XOR- und anderen Schreibmodi brauchbar ist.

($\hookrightarrow$ Folgeseite)

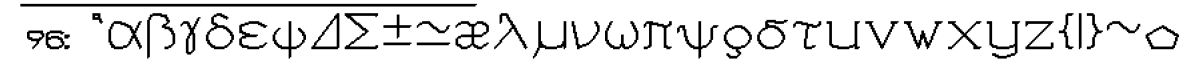
Fortsetzung: SYMBOL

SYMBOL: Zeichensatz

ASCII–Set



GREEK–TECHNICAL Set



CENTERED SYMBOLS 0 – 14

Zweck: Zeichenneigung ENTRY: **SYMBSL**

Aufruf: `CALL SYMBSL (SLANT)`

Parameter	Typ	A/R	Bedeutung
SLANT	R	Arg	Neigung in Grad

Erläuterung: `SYMBOL` zeichnet die nächsten Textzeichen mit der angegebenen Neigung. `NEWPLOT` belegt `SLANT` mit 0.0 vor.

Zweck: Zeichenhöhe ENTRY: **SYMBHB**

Aufruf: `CALL SYMBHB (HB)`

Parameter	Typ	A/R	Bedeutung
HB	R	Arg	Höhen- zu Breitenverhältnis

Erläuterung: `SYMBOL` berechnet die Zeichenhöhe aus der mittleren Zeichenbreite und dem Verhältnis. `NEWPLOT` belegt `HB` mit 1.0 vor.

Zweck: Zeichensatz ENTRY: **SYMBGR**

Aufruf: `CALL SYMBGR (GREEK)`

Parameter	Typ	A/R	Bedeutung
GREEK	I	Arg	Code, siehe unten

Erläuterung: `GREEK` > 0 ersetzt die Kleinbuchstaben a bis t durch die griechisch-technischen Zeichen. `GREEK` < 1 stellt den Originalzustand wieder her. `NEWPLOT` belegt `GREEK` mit 0 vor.

Zweck: Einstellungen retten ENTRY: **SYMSAV**

Aufruf: `CALL SYMSAV`

Parameter	Typ	A/R	Bedeutung
			– keine –

Erläuterung: Die Einstellungen `SLANT, HB, GREEK` in `SYMBOL` werden gesichert. Eine zweistufige PUSH-Mimik ist implementiert.
Gebraucht von `NUMBER`.

Zweck: Einstellungen wiederherstellen ENTRY: **SYMRES**

Aufruf: `CALL SYMRES`

Parameter	Typ	A/R	Bedeutung
			– keine –

Erläuterung: Die Einstellungen `SLANT, HB, GREEK` in `SYMBOL` werden wiederhergestellt.
Gebraucht von `NUMBER`.

Zweck: Zahl zeichnen, erweitert SUBROUTINE: **NUMBER**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Arg	x-Koordinate oder 999.0
YPAGE	R	Arg	y-Koordinate oder 999.0
WIDTH	R	Arg	mittlere Zeichenbreite
FPN	R	Arg	Zu zeichnende Real-Zahl
ANGLE	R	Arg	Winkel zur x-Achse in Grad
NDEC	I	Arg	Stellenzahl rechts vom Dezimalpunkt / code

Erläuterung: Die Zahl wird in standardisiertem Format gezeichnet bei NDEC < 10.
Bei NDEC gleich -1 wird nur der ganzzahlige Teil ohne Dezimalpunkt gezeichnet.

Bei NDEC > 9 wird FPN als Zeit in Sekunden aufgefaßt und in d-h-m-s Format dargestellt, gesteuert von NDEC:
Beispiel mit `FPN=100000 s:`

```
NDEC   gezeichnet
1000   1d
1100   1d4h
1010   1d3h47m              (linkeste 1 ergibt maximale Einheit,
1001   1d3h46m40s           die rechteste die kleinste Einheit.)
0100   27h
0110   27h47m
0101   27h46m40s
0010   1667m
0011   1666m40s
```

Zweck: Achse zeichnen, erweitert SUBROUTINE: **AXIS**

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

Parameter	Typ	A/R	Bedeutung
XPAGE	R	Arg	x-Koordinate oder 999.0, Achsenanfang
YPAGE	R	Arg	y-Koordinate or 999.0
TEXT	CHAR	Arg	Titel an der Achse
NCHAR	I	Arg	Zeichenanzahl im Titel
AXLEN	R	Arg	Achsenlänge
ANGLE	R	Arg	Winkel der Achse in Grad
FIRSTV	R	Arg	Wert für Achsenanfang
DELTAV	R	Arg	Wert- zu Längenverhältnis

Erläuterung: Das Vorzeichen von NCHAR gibt an, auf welcher Achsenseite Marken, Werte und Text stehen sollen: + auf der linken, - auf der rechten (vom Achsenanfang her gesehen). Größen und Abstände sind für Längeneinheit cm ausgelegt (in NEWPLOT). Für Zoll und andere Anwenderwünsche sind diese leicht mit den nächsten Routinen zu ändern.

Zweck: Markenhöhe und -Abstand ENTRY: **AXMARK**

Aufruf: `CALL AXMARK (HTIC,DTIC)`

Parameter	Typ	A/R	Bedeutung
`HTIC`	`R`	Arg	Höhe der Marken an der Achse
`DTIC`	`R`	Arg	Abstand zwischen dem Marken

Erläuterung: Standard sind 0.18 und 1.00 für cm.
`CALL AXMARK (0.075, 0.5)` ist für Zollmaß brauchbar.

Zweck: Achsenvermaßung ENTRY: **AXZIFF**

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

Parameter	Typ	A/R	Bedeutung
`NZ`	`I`	Arg	`NDEC` für Zahlen an den Achsmarken
`AZ`	`R`	Arg	Winkel zwischen Zahlen und Achse in Grad
`WZ`	`R`	Arg	Ziffernbreite
`SZ`	`R`	Arg	Ziffernneigung
`HZ`	`R`	Arg	Höhenverhältnis
`JZ`	`I`	Arg	Jede `JZ`te Marke wird beziffert

Erläuterung: `CALL AXZIFF (2, 0.0, 0.26, 0.0, 1.0, 2)` gibt Standard für cm.
`CALL AXZIFF (2, 0.0, 0.1, 0.0, 1.0, 2)` gibt Standard für Zoll.

Zweck: Achsenbeschriftung ENTRY: **AXTEXT**

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

Parameter	Typ	A/R	Bedeutung
IT	I	Arg	Zentriercode
AT	R	Arg	Winkel zwischen Text und Achse in Grad
WT	R	Arg	Zeichenbreite
ST	R	Arg	Zeichenneigung
HT	R	Arg	Höhenverhältnis

Erläuterung: `IT` =1: Text bündig mit Achsenanfang, =2: mittig, =3: bündig mit dem Ende der Achse.
`CALL AXTEXT (2, 0.0, 0.35, 0.0, 1.0)` gibt Standard für cm.
`CALL AXTEXT (2, 0.0, 0.15, 0.0, 1.0)` gibt Standard für Zoll.

Zweck: Stellenzahl ermitteln INTEGER FUNCTION: **NZZIFF**

Aufruf: `CALL NZZIFF (FPN,NDEC)`

Parameter	Typ	A/R	Bedeutung
FPN	R	Arg	Real-Zahl
NDEC	I	Arg	Anzahl Stellen

Erläuterung: `NZZIFF` berechnet die Anzahl Zeichen, aus denen `FPN` bei Ausgabe durch `NUMBER` besteht. Alle Sonderfälle sind eingeschlossen.
(Gebraucht in `AXIS`.)

Zweck: Polygon füllen – Scanline SUBROUTINE: **FILLZZ**

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

Parameter	Typ	A/R	Bedeutung
XY	R(2,NXY)	Arg	Koordinaten der Polygonecken
NXY	I*2	Arg	Anzahl der Ecken
ICOL	I*2	Arg	Füllfarbe
H	I(2*NXY+..)	–	Zwischenspeicher

Erläuterung: `FILLZZ` füllt beliebige Polygone. Es muß ausreichender Platz in `H` für `2*NXY` Pixelkoordinaten und zusätzlich für 3*Maximum (Anzahl Schnitte des Polygons mit eine einzelnen Scanlinie) vorhanden sein.

Zweck: Dreiecke: Farbinterpolation SUBROUTINE: **INTER3**

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

Parameter	Typ	A/R	Bedeutung
XY	R(IDXY,0:*)	Arg	Koordinaten aller Knoten im Sytem
IDXY	I	Arg	Anzahl Zeilen von XY; (XY(1,j),XY(2,j)) sind die Koordinaten von Knoten j.
Z	R(IDZ,0:*)	Arg	Funktionswerte aller Knoten im System
IDZ	I	Arg	Anzahl Zeilen von Z; Z(1,j) ist Funktionswert zu Knoten j.
ELE	I(3)	Arg	Knotennummern eines einzelnen Dreieckselements. Numerierung beginnt mit 0.
ZMIN	R	Arg	Kleinster Funktionswert für Farben, siehe unten
DZ	R	Arg	Abstand der Funktionswerte für Farben, siehe unten
PALETT	I(1,NF)	Arg	Externe Palette
NF	I	Arg	Anzahl Farben in der Palette

Erläuterung: Das Dreieck wird pixelweise eingefärbt. Lineare Interpolation der Funktionswerte Z an jedem Pixel ergibt die Farben: Pixel mit Z <ZMIN erhalten Farbe `PALETT(1)` anderer die Farbe `PALETT (2+ MAX(NF, INT((Z-ZMIN)/DZ)))`.
Für die Darstellung skalarer Ergebnisse aus Finite-Element-Rechnungen.
Nur im absoluten Schreibmodus brauchbar!

Zweck: Vierecke: Farbinterpolation SUBROUTINE: **INTER4**

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

Parameter	Typ	A/R	Bedeutung
XY	R(IDXY,0:*)	Arg	Koordinaten aller Knoten im Sytem
IDXY	I	Arg	Anzahl Zeilen von XY; (XY(1,j),XY(2,j)) sind die Koordinaten von Knoten j.
Z	R(IDZ,0:*)	Arg	Funktionswerte aller Knoten im System
IDZ	I	Arg	Anzahl Zeilen von Z; Z(1,j) ist Funktionswert zu Knoten j.
ELE	I(3)	Arg	Knotennummern eines einzelnen Viereckselements. Numerierung beginnt mit 0.
ZMIN	R	Arg	Kleinster Funktionswert für Farben, siehe unten
DZ	R	Arg	Abstand der Funktionswerte für Farben, siehe unten
PALETT	I(1,NF)	Arg	Externe Palette
NF	I	Arg	Anzahl Farben in der Palette

Erläuterung: Das Viereck wird pixelweise eingefärbt. Bilineare Interpolation der Funktionswerte Z an jedem Pixel ergibt die Farben: Pixel mit Z <ZMIN erhalten Farbe `PALETT(1)` anderer die Farbe `PALETT (2+ MAX(NF, INT((Z-ZMIN)/DZ)))`.
Für die Darstellung skalarer Ergebnisse aus Finite-Element-Rechnungen.
Nur im absoluten Schreibmodus brauchbar!

4.5 Schlußbemerkung

Die vorbeschriebenen Unterprogramme bilden die Basis für eine vollständige CALCOMP-Bibliothek für nahezu alle SuperVGAs. Weitere Routinen wie `SCALE`, `LINE`, `LGAXS` usw. können beliebig zugefügt werden. Die Erweiterungen der Funktionalität sollten besonders für Anwendung in Wissenschaft und Technik nützlich sein.

5 VGA.CFG, kommentiertes Beispiel

Die Konfigurationsdatei enthält Information für das Graphiksystem über Auflösung, Farben und Hardcopy. `Das verbatim gegebene Beispiel` *ist kursiv kommentiert*.

```
$VGA.CFG - Beispieldatei
$Sektion 1: Hardcopydrucker
```

Zeilen mit $ sind Kommentare und werden überlesen.
*Sektion 1 wird nicht gebraucht, nur zwecks Kompatibilität mit voriger Version enthalten, darf fehlen. Spalten 1-10: Schlüsselwort *HARDCOPY*; 11-20 Druckerkennung. Erste Definition gilt, Redefinitionen werden ignoriert.*

```
*HARDCOPY*9_NEEDLES
*HARDCOPY*EPSON24
$Sektion 2: Mode-Parmeter
$FORMAT (A10,8I5,A10,A10)  Name, 8 Integers, Palettenname, Hardcopypalette
$MODNAM   MXMOD<---------- unbenutzt ------------> PALETTE  H.PALETTE comment
$ MODNAM : beliebiger Name, ohne weitere Referenz
$ MXMOD  : wenn nicht leer und > -1: Maximalmodus begrenzt auf MXMOD,
$       setzen auf maximalen Modus, den der angeschlossene Monitor vertraegt!
$       (Modi in VSVGA.COM sind 0 to 5)
$ PALETTE:   Name muss in Sp. 1-10 der Zeilen zu Sektion 3 wieder auftauchen.
$            Leitet dort eine Palettendefinition ein, siehe unten.
$ H.PALETTE: Nichtleeres Feld in Sp. 61-70 enthaelt einen HARDCOPY PALETTE
$            Namen. Wird in Sektion 4 in Sp. 1-10 referiert.
$            Leerfeld in Sp. 61-70 ergibt RBG-Werte auf COLOR.MAP
$ Alle Namen sind fallsensitiv!
$ Auch hier gilt die erste Definition!
VSVGA.COM    ??<---------- unbenutzt ------------> BLUE_RED GREYSCALE P.SCRIPT
VSVGA.COM     4<---------- unbenutzt ------------> BLUE_RED REVERSE   DIAs
VSVGA.COM     4<---------- unbenutzt ------------> BLUE_RED           RGB-Werte
```

Nur die erste Integer und die Palettennamen werden von VSVGA.COM gelesen. Leerfeld oder nichtnumerischer Eintrag ergibt für MXMOD -1.
Alte Treiberversion las 8 Integers, der Vollständigkeit halber beschrieben:

```
$ALT: 8 Integers, um Register und Aufloesung fuer speziellen Modus zu setzen:
$MODNAM      AX   BX   MX   MY  MXT  MYT   MC  XOR PALETTE  H.PALETTE comment
$ AX,  BX  :Registerwerte,  # leitet Hexadezimalzahl ein
$ MX,  MY  :max. Pixelkoordinaten
$ MXT, MYT :max. Textspalte
$ XOR      :0  XOR-Modus nicht unterstuetzt in INT 10h
```

(↪ Folgeseite)

Fortsetzung

```
ALTE Konfigurationen, nicht fuer VSVGA.COM verwendbar!
"$" ueberfluessig, solange Zeilen nicht mit Palettennamen beginnen
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

$Sektion 3: Paletten fuer VGA, Auswahl durch Palettenname aus Sektion 2.
Zeile 1: (A10,16I4) Palettenname, Farbtabelle (Stifte 0--15, NEWPEN)
```

Um Farben ohne Änderung der RGB-Werte anders zuzuordnen. Hier Intensivfarben zuerst!

```
 PALETTE_1   0   9  10  11  12  13  14  15   8   1   2   3   4   5   6   7
$Farben: (4I5) Nummer, R, G, B
$negative Nummer: Interpolation von letzter Nummer bis hier
$   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
$ Schattierungen. Beliebige Folge. Letzte Werte gelten. Auslassungen erlaubt.
   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
#### Nichtnumerische Eingabe zeigt das Ende der Palette an
```

(↪ Folgeseite)

Fortsetzung

Unbenutzte Paletten können enthalten sein, Auswahl nur anhand des Palettennamens!

```
 BLUE_RED    0   9  10  11  12  13  14  15   8   1   2   3   4   5   6   7
    0    0    0    0  BLACK    oben : Stifttabelle
    1    0    0   40  BLUE     links: RGB-Werte
    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 UND WRITE COLOUR 1 D.BLAU
    1             10   (TEXT OUTPUT unterdrueckt)
$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
#### END OF VGA CONFIGURATIONS
```

(↪ Folgeseite)

Fortsetzung

```
$Sektion 4: Hardcopy palette (VSVGA)
----- Hardcopypaletten ------
$VSVGA.COM liest in (2I5) Farbnummer und Grauwert fuer COLOR.MAP
$Farbnummern aufsteigend, fehlende werden interpoliert
```

Letzte Nummer muß 255 sein. Wert 0 ist Weiß, 63 schwärzestes Schwarz. Farbausgabe: keine Hardcopy-Palette angeben!

```
GREYSCALE     (POSTSCRIPT schwarz auf weiss)
    0    0    ;WHITE
    1   63    ;BLACK
    2   50    ;dark grey
   15   20    ;light grey
$-- Ende Stifte ---------
   16    0
  255   63
##### END

REVERSE       Folien weiss auf schwarz
    0   63    ;BLACK
    1    0    ;WHITE
    2   20    ;light grey
   15   50    ;dark grey
$-- Ende Stifte ---------
   16   63
  255    0
##### END
```

Alte Konfigurationen für Matrixdrucker folgen. Kann fehlen, nicht gebraucht für VSVGA.COM.

```
EPSON24       3  1.0 (Vertikale Bytes und Verzerrung)
$ 24 Nadler  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
```

(↪ Folgeseite)

Fortsetzung

```
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  Emulation alter EPSONs auf 24-Nadeldrucker
$ 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 Beispieldatei / VSVGA.COM)
```