Ingenieurbüro für Echtzeitprogrammierung

Dokumentation NBS300

Dok-Rev. 1.8 vom 14.11.2007

Hardware-Rev. 1.0 vom 01.09.1997

Inhaltsverzeichnis

Rev.	Datum	Na.	Änderung
1.0	07.05.1998	Ha	Erstellung
1.1	15.05.1998	La	Schaltpläne eingefügt
1.2	29.10.1998	Ko	Schreibfehler und Blocksatz korrigiert
1.3	22.03.1999	La	Schreibfehler korrigiert
1.4	05.07.2000	Ko	auf Hardware.dot umgestellt
1.5	05.10.2000	Ko	Fehler im Kommentar QSPI aufsetzen behoben
1.6	14.01.2002	Ko	Batterie/Goldcap ergänzt (Kap. 4.4)
1.7	24.01.2002	Ko	Fehler in den Kommentaren zum SPI Initialisieren behoben
1.8	12.07.2004	Ko	Anschlußbelegung (ST13/14) ergänzt

1 Allgemeine Hinweise

1.1 Handhabung

1. Lesen Sie bitte zuerst sorgfältig diese Dokumentation bevor Sie die Hardware auspakken und einschalten. Sie sparen Zeit und vermeiden Probleme.
2. Beachten Sie bitte die Vorsichtsmaßnahmen bei der Handhabung elektrostatisch gefährdeter Hardware.
3. Wenn die Hardware Batterien enthält, legen Sie sie nicht auf elektrisch leitfähige Unterlagen. Die Batterie könnte kurzgeschlossen werden und Schäden verursachen.
4. Achten Sie bitte darauf, daß der spezifizierte Temperaturbereich nicht verlassen wird.

1.2 Installation

1. Überprüfen Sie, ob alle Jumper entsprechend Ihrer Anwendung gesetzt sind.
2. Schalten Sie die Spannungsversorgung der externen Anschlüsse ab, bevor Sie eine Verbindung herstellen.
3. Wenn Sie sicher sind, daß alle Verbindungen korrekt installiert sind, schalten Sie die Spannungsversorgung ein.

1.3 Erklärung

Wir behalten uns das Recht vor, Änderungen, die einer Verbesserung der Schaltung oder des Produktes dienen, ohne besondere Hinweise vorzunehmen. Trotz sorgfältiger Kontrolle kann für die Richtigkeit der hier gegebenen Daten, Schaltpläne, Programme und Beschreibungen keine Haftung übernommen werden. Die Eignung des Produktes für einen bestimmten Einsatzzweck wird nicht zugesichert.

1.4 Reparaturen

Sollte das Produkt defekt sein, so senden Sie es bitte frei in geeigneter Verpackung mit folgender Beschreibung an uns zurück:

- Fehlerbeschreibung
- Trat der Fehler nur unter bestimmten Bedingungen auf?
- Was war angeschlossen?
- Wie sahen die angeschlossenen Signale aus?
- Garantiereparatur oder nicht?

2 Allgemeine Informationen

2.1 Einbau

Die NBS300 ist zum Einbau in EMV-dichte Gehäuse bestimmt. Die Verkabelung ist EMV-gerecht mit abgeschirmten Kabeln durchzuführen. Sie darf nur von EMV-kundigem Personal durchgeführt werden.

2.2 Umgebungsbedingungen

Umgebungstemperatur	Betrieb: 0-50° C, Lagerung: -20-85° C
rel. Luftfeuchte	max. 95%, nicht kondensierend
Höhe	-300m bis +3000m

2.3 Mechanische Abmessungen

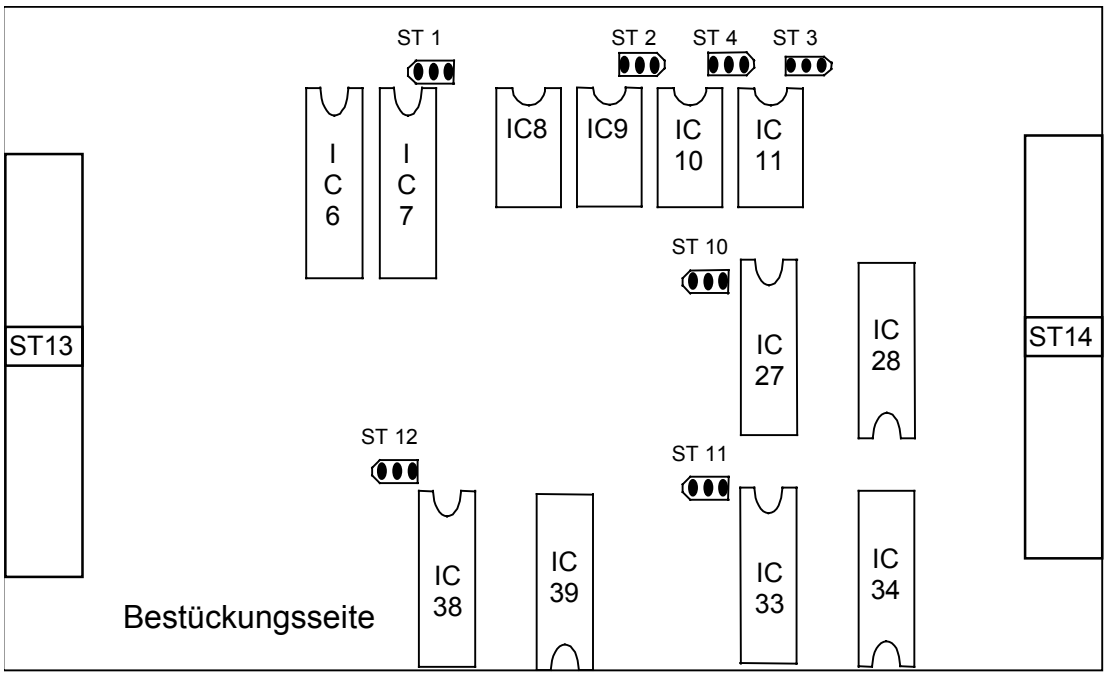
Doppel-Europakarte:	233 x 160 mm
Anschlüsse:	über 2x VG-Leiste DIN 41612, Bauform C, a+c

2.4 Technische Eigenschaften

Versorgungspannung:	18...36 VDC
Prozessor:	MC68332, 16,7 MHz, 16-Bit Datenbus
Speicher:	2x 32p JEDEC für EPROM, max 2MB SRAM, max 2MB, optional batteriegepuffert FLASH, max 1MB, on-board programmierbar
Batteriepufferung:	für Echtzeituhr und SRAM, über Lithium-Batterie oder Gold-Cap-Kondensator
Motorregler:	3 Inkrementalgeber A/B, Indeximpuls, TTL-Pegel 3 Analog-Spannungsausgänge ± 10V, 12 Bit
Analogeingänge	2 single-ended, unipolar Eingangsempfindlichkeit 0..5V, Auflösung 12 Bit
Digitaleingänge:	16 Stück, TTL-Pegel 4 einlesbare Dip-Switches
Digitalausgänge:	16 Stück TTL-Pegel 4 Stück TTL, offener Kollektor
Matrix-Tastatur:	bis 7x8 Tasten, frei konfigurierbar
LCD-Display:	Grafik-Module mit dem Controller T6963
Ser. Schnittstellen	1x 5-Draht RS232 1x 3-Draht, TTL

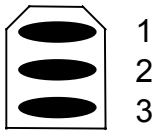
3 Inbetriebnahme

3.1 Lage der Jumper



3.2 Beschreibung der Jumper

Die Löt-Jumper werden folgendermaßen gezählt:



3.2.1 ST1 – EPROM-Konfiguration

ST1 muß entsprechend der eingesetzten EPROMs gesetzt werden:

Position	EPROM-Typ
1-2 verbunden	27040 (512kB)
2-3 verbunden	27010 / 27020 (128kB / 256kB)

3.2.2 ST2 –Konfiguration SRAM-Bank 1

ST2 muß entsprechend der eingesetzten SRAMs gesetzt werden:

Position	SRAM-Typ
1-2 verbunden	128 kB
2-3 verbunden	512 kB

3.2.3 ST3 –Konfiguration SRAM-Bank 2

ST3 muß entsprechend der eingesetzten SRAMs gesetzt werden:

Position	SRAM-Typ
1-2 verbunden	128 kB
2-3 verbunden	512 kB

3.2.4 ST4: Batteriepufferung der SRAMs

ST4 legt gemeinsam die Spannungsversorgung der beiden SRAM-Bänke fest:

Position	SRAM-Versorgung
1-2 verbunden	Keine Batteriepufferung
2-3 verbunden	SRAMs bei Spannungsausfall über Batterie gepuffert

3.2.5 ST10, ST11, ST12: Indeximpuls der Motorregler

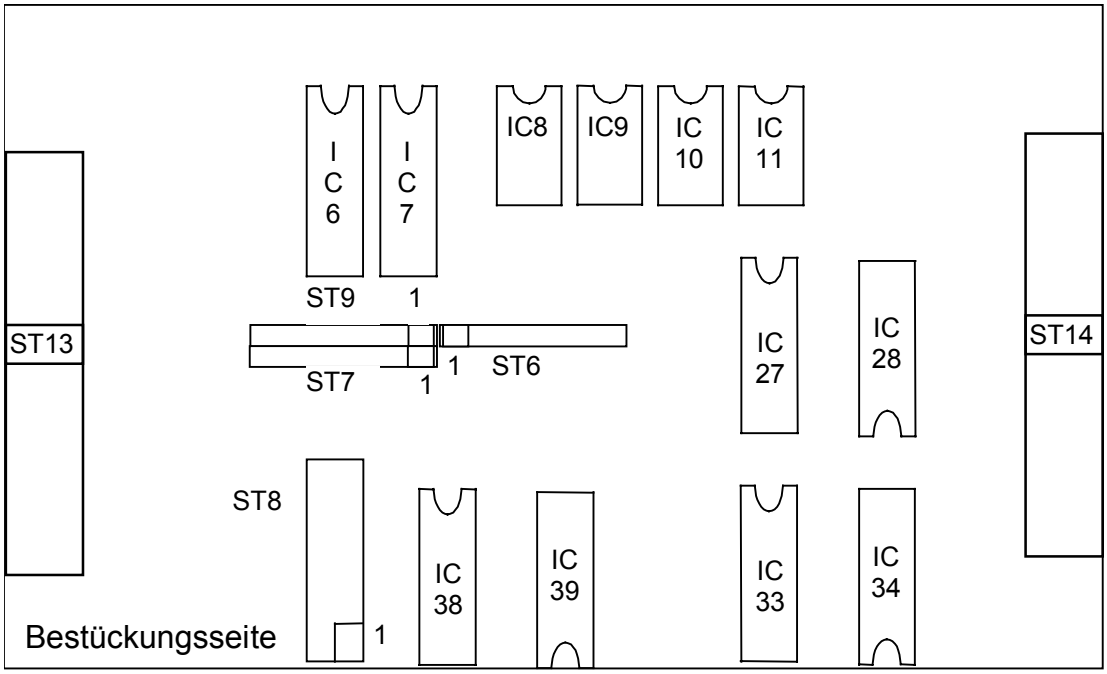
Mit diesen Jumpern wird der Weitergabe eines Indeximpulses von den Inkrementalgebern an die Motorregler festgelegt.

Position	Indeximpuls wird
1-2 verbunden	vom Motorregler genutzt
2-3 verbunden	nicht verwendet

Werkszustand: 2-3 verbunden, d.h. Indeximpuls wird nicht genutzt.

Diese Verbindung ist über eine Leiterbahn als Default-Jumperstellung realisiert. Diese Leiterbahn ist vor einer Änderung zu trennen (Kratzbrücke)!

3.3 Lage der Steckfelder



3.4 Beschreibung der Steckfelder

3.4.1 ST6, ST7, ST9 – Matrix-Tastatur

Mit den Steckfeldern ST6, ST7 und ST9 ermöglichen den Anschluß von Matrix-Tastaturen mit bis zu 56 Tasten in 8 Zeilen und 7 Spalten. Die Steckfelder sind so angeordnet, daß sich eine möglichst große Zahl von unterschiedlichen Matrix-Tastaturen anschliessen lässt.

Die Belegung der Steckfelder ist wie folgt:

	S6	S5	S4	S0	S1	S2	S3	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0								

3.4.2 ST8 – LCD-Display

Über ST8 ist ein LCD-Display mit dem Controller T6963 anschliessbar.

Die Kontrasteinstellung erfolgt über TR1 (direkt neben ST8).

Die Signale FS (Font Select, 0: 6x8, 1: 8x8) und RV (Reverse, 0: Helle Schrift, dunkler Grund, 1: dunkle Schrift, heller Grund) sind über die I/O-Leitungen auf Port E des Prozessors schaltbar.

3.5 ST13, ST14 – I/O-Anschlüsse

Belegung ST13:

Pin	A-Reihe	Gruppe	C-Reihe	Gruppe
1	LED 1	Leuchtdioden	TxD	Serielle Schnittstelle 1
2	LED 2		RxD	
3	LED 3		RS2_RxD	Serielle Schnittstelle 2
4	LED 4		RS2_TxD	
5			RS2_CTS	
6			RS2_RTS	
7				
8	D_OUT1.0	digitale Ausgänge	D_OUT1.1	digitale Ausgänge
9	D_OUT1.2		D_OUT1.3	
10	D_OUT1.4		D_OUT1.5	
11	D_OUT1.6		D_OUT1.7	
12	D_OUT2.0	digitale Ausgänge	D_OUT2.1	digitale Ausgänge
13	D_OUT2.2		D_OUT2.3	
14	D_OUT2.4		D_OUT2.5	
15	D_OUT2.6		D_OUT2.7	
16				
17	D_IN1.0	digitale Eingänge	D_IN1.7	digitale Eingänge
18	D_IN1.1		D_IN1.6	
19	D_IN1.2		D_IN1.5	
20	D_IN1.3		D_IN1.4	
21	D_IN2.0	digitale Eingänge	D_IN2.7	digitale Eingänge
22	D_IN2.1		D_IN2.6	
23	D_IN2.2		D_IN2.5	
24	D_IN2.3		D_IN2.4	
25				
26				
27	V_ANA	analoger Eingang	V_ANA	analoger Eingang
28	A_CH0		A_CH1	
29	A_GND		A_GND	
30				
31	GND1	Masseanschluß	+24Volt	Versorgungsspannung
32	GND1		+24Volt	

Belegung ST14:

Pin	A-Reihe	Gruppe	C-Reihe	Gruppe
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11	LM1_OUT	Motor 1		
12	AGND			
13	LM1_INDEX			
14	LM1_A			
15	LM1_B			
16	GND1			
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27	LM2_OUT	Motor 2	LM3_OUT	Motor 3
28	AGND		AGND	
29	LM2_INDEX		LM3_INDEX	
30	LM2_A		LM3_A	
31	LM2_B		LM3_B	
32	GND1		GND1	

4 Programmierung

4.1 Adreßbelegung

4.1.1 Externe Peripherie am Prozessorbus

Die Adreßbelegung der NBS300 wird im wesentlichen von der Programmierung der Chip-Select-Leitungen des Prozessors bestimmt. Die Standardkonfiguration ist:

Chip-Select	Anschluß	Größe	Adresse	Programmierung
CSBOOT	EPROM	1 MB	$C00000-$CFFFFF	0 WS
CS0	RAM 1 high	512 KB	$000000-$0FFFFF	0 WS
CS1	RAM 1 low	512 KB	$000000-$0FFFFF	0 WS
CS2	RAM 1 high	512 KB	$100000-$1FFFFF	0 WS
CS3	RAM 1 low	512 KB	$100000-$1FFFFF	0 WS
CS4	FLASH high	512 KB	$D00000-$DFFFFF	0 WS
CS5	FLASH low	512 KB	$D00000-$DFFFFF	0 WS
CS6	---	---		A19 +Autovector ($FFF8)
CS7	RTC	2 K	$6C0000-$6DFFFF	3 WS, Byte
CS8	auf 138 De-coder	64 K	$600000-$60FFFF	externer DTACK
	Display	4 K	$600000-$6007FF	
	Tastatur	4 K	$601000-$601FFF	
	LM628 Kanal 1	4 K	$602000-$602FFF	
	LM628 Kanal 2	4 K	$603000-$603FFF	
	LM628 Kanal 3	4 K	$604000-$604FFF	
CS9	---	---		frei
CS10	---	---		frei

4.1.2 Peripheriebausteine am SPI

Die digitalen Ein- und Ausgänge sowie der A/D-Wandler sind als Peripheriebausteine über das SPI des MC68332 angeschlossen. Es sind angeschlossen:

Digitale Eingänge	PCS1
Digitale Ausgänge	PCS0
A/D-Wandler	PCS2

4.1.3 Portleitungen des Prozessors

An den Portleitungen des Prozessors sind angeschlossen:

LED-Ausgänge - Port F (0xFFFA18)

X	X	X	X	X	X	X	X	LED 4	LED 3	LED 2	LED 1	X	X	X	X

DIP-Switch und Display-Konfiguration- Port E (0xFFFA10)

X	X	X	X	X	X	X	X	O	O	I	I	I	I	O	X
X	X	X	X	X	X	X	X	LCD RV	LCD FS	DIP 4	DIP 3	DIP 2	DIP 1	RS2_RTS	X

O: Ausgang

I: Eingang

LCD FS: High=6x8 Pixel

LCD RV: High=Positiv Mode

Die Ausgänge dürfen nur mit unteilbaren Bit-Manipulationsbefehlen gesetzt werden, da die Integrität der Ausgangsdaten sonst nicht gewährleistet ist.

4.2 Funktionsbeschreibung des SPI

Nach einer Programmierung des QSM mit folgenden RTOS-UH-Monitorbefehlen arbeitet das SPI automatisch und unabhängig von der CPU:

```
/* Initialisierung */
SM -SW FFFC16 3BFE ; /* QPAR:  PCS2=PCS1=PCS0=MISO=MOSI=1            */
SM -SW FFFC14 00E7 ; /* QPDR:  PCS1=PCS0=0, sonst 1                  */
SM -SW FFFC18 B72A ; /* SPCR0: Master, 13 Bits, CPOL=CPHA=1, 200 kHz */
SM -SW FFFC1C 4500 ; /* SPCR2: WREN, ENDQP=5, NEWQP=0                */

/* TX-Data vorbesetzen */
SM -SW FFFD20 00FF ; /* Start AD 1. Kanal       */
SM -SW FFFD22 0000 ; /* DO1 alle Ausgänge auf 0 */
SM -SW FFFD24 0000 ; /* DO2 alle Ausgänge auf 0 */
SM -SW FFFD26 00D0 ; /* Start AD 2. Kanal       */
SM -SW FFFD28 0000 ; /* Dummy                   */
SM -SW FFFD2A 0000 ; /* Dummy                   */

/* Command-Ram aufsetzen */
                            /* PCS2 PCS1 PCS0 */
SM -SB FFFD40 8A    ; /*  CONT,  8 Bit, 010 */
SM -SB FFFD41 CA    ; /*  CONT, 13 Bit, 010 */
SM -SB FFFD42 CE    ; /*  CONT, 13 Bit, 110 */
SM -SB FFFD43 8B    ; /*  CONT,  8 Bit, 011 */
SM -SB FFFD44 CA    ; /*  CONT, 13 Bit, 010 */
SM -SB FFFD45 4E    ; /* !CONT, 13 Bit, 110 */

/* QSM Starten */
SM -SW FFFC1A B020    ; /* SPE */
```

Die Ein- und Ausgangsdaten stehen dann im QSM-Ram zur Verfügung Sie können jederzeit gelesen und geschrieben werden. Die tatsächliche Ein/Ausgabe erfolgt zyklisch ca. alle 300µs.

Die Abbildung der Daten im QSM-Ram ist wie folgt:

Eingangsdaten:

0xFFFD00

X	X	X	X	X	X	X	X	IN1.7	IN1.6	IN1.5	IN1.4	IN1.3	IN1.2	IN1.1	IN1.0

0xFFFD02

X	X	X	IN2.7	IN2.6	IN2.5	IN2.4	IN2.3	IN2.2	IN2.1	IN2.0	X	X	X	X	X

0xFFFD04

x	x	x	MSB	Analog-Eingangskanal 1 LSB	x

0xFFFD0A

x	x	x	MSB	Analog-Eingangskanal 0 LSB	x

Ausgangsdaten:

0xFFFD24

X	X	X	O2.4	O2.3	O2.2	O2.1	O2.0	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0

0xFFFD22

X	X	X	X	X	X	X	X	X	X	X	X	X	O2.7	O2.6	O2.5

4.3 Motorregler

Über die Inkrementalgebereingänge und die Analogausgänge wird mit Hilfe eines LM628 ein Regelkreis realisiert.

Zur Programmierung dieses integrierten Reglers ist das Datenblatt erforderlich.

4.4 Batterie/Goldcap

Das Gerät ist ggf. mit einem gepufferten RAM ausgestattet. Die Pufferung erfolgt entweder über eine 3,6 Volt Lithiumbatterie oder einen 1 F Goldcap. Die Batterie wird erst bei der Lieferung eingesetzt und hat dann eine Mindestlebensdauer von 5 Jahren, unabhängig von der Einschaltdauer des Gerätes. Die Pufferdauer mit dem Goldcap hängt vom Ladezustand ab, bei geladenem Goldcap ist eine Pufferdauer von min. 14 Tagen gegeben.

5 Abgleich

Alle Karten sind bei Werksauslieferung getestet und abgeglichen und erfordern keinen regelmäßigen Neuabgleich.

6 Terminalemulation

6.1 Allgemeines

Die Terminalemulation ermöglicht den Anschluß von getrennten Tastaturen und Displays an einem RTOS-UH-Rechner und verhält sich wie eine übliche serielle Schnittstelle.
Unter den mnemotechnischen Bezeichnungen `/AT`, `/BT` und `/CT` ist die Terminalemulation in der A-, B- oder C-Betriebsart unter der LDN 4 mit den Drives 0, 2 und 6 erreichbar. Unter der Bezeichnung `/DT` wird der Ausgabekanal im Vollduplex--Betrieb mit der LDN 14 angesprochen.
Zusätzlich zu den üblichen Betriebsarten kann die Terminalemulation bei LDN 4 noch unter den Drives 64, 66 und 70 angesprochen werden. Die Terminalemulation liefert in diesem Fall bei Eingaben die Tastatur-Scancodes ohne Übersetzung in ASCII-Zeichen. Auch hier ist ein Betrieb entsprechend der A-, B- oder C-Betriebsart möglich, jedoch findet grundsätzlich kein Echo statt. Besonders ist zu beachten:

- Bei Betriebsartenumschaltung Scan-Code/ASCII kann der Empfangspuffer noch Zeichen in der jeweils anderen Codierung enthalten. Zur Sicherheit sollte der Empfangspuffer daher durch ein erstes Lesen in der A-Betriebsart gelöscht werden.
- Bei Scan-Code-Betrieb müssen die Ausgaben ebenfalls über die Drives 64, 66 oder 70 erfolgen, da ansonsten wieder eine Rückschaltung in den ASCII-Betrieb erfolgt.

Der Tastaturtreiber bietet keine Auto-Repeat-Funktion. Statt dessen sind Treiberversionen verfügbar, die bei Loslassen einer gedrückten Taste den ASCII-Wert NUL liefern. Hierdurch kann die Dauer einer Tastenbetätigung erfasst und ggf. ein Auto-Repeat emuliert werden.

6.2 Befehlsvorrat Terminalemulation

Die Terminalemulation verhält sich weitgehend Televideo-kompatibel. Sie versteht die folgende Befehlssequenzen (Erläuterung der Angaben *Ps*, *Pc*, *Pn*, *r* und *c* siehe unten). Es ist nicht gewährleistet, daß alle Funktionen auf allen Terminals zur Verfügung stehen! So können auf einen schwarz/weiß--Display natürlich keine Farben eingestellt werden, oder auf einen Textdisplay wird es schwierig, Grafik auszugeben.

Cursor style (0..4)			
	ESC . *Ps*		
		0	Cursor off
		1	Blinking block
		2	Steady block
		3	Blinking underline
		4	Steady underline

Define Visual attributes																	
	ESC G *Ps*		0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
		Underline									X	X	X	X	X	X	X	X
		Reverse					X	X	X	X					X	X	X	X
		Blinking			X	X			X	X			X	X			X	X
		Invisible		X		X		X		X		X		X		X		X

Operating Modes			
	ESC	)	Enable invers
		(	Disable invers
		V	Autoscroll off
		W	Autoscroll on
		$	Grafics mode on
		%	Grafics mode off
		U	Monitor mode on
		X	Monitor mode off
		[=7h	Auto Wrap on
		[=7l	Auto Wrap off
		! *c*	c >= ‚A‘: Schreibfarbe, c<‘A‘: Hintergrundfarbe setzen
		[*Pc* ; *Pc* r	Set scroll region rows (*Pc*: start, end)
		[*Pc* ; *Pc* s	Set scroll region columns (*Pc*: start, end)

Cursor Movement			
	CTRL	J	Line feed
		K	Up
		V	Down
		L	Right
		M	Carriage Return
		H	Left
	ESC	= *r c*	Set Cursor *row column*
		[*Pn* A	Cursor up *Pn* rows
		[*Pn* B	Cursor down *Pn* rows
		[*Pn* C	Cursor right *Pn* columns
		[*Pn* D	Cursor left *Pn* columns
		[*c* ; *r* H	Set Cursor position *column*, *row*

Tabulator Control			
	ESC	1	Set tab stop at actual column
		2	Clear tab stop at actual column
		3	Clear all tab stops
		i	Move Cursor to next tab stop
		I	Move Cursor back one tab stop

Insertion			
	ESC	Q	Insert space at cursor
		E	Insert line of spaces

Miscellaneous			
	ESC	~	Reset Terminal

Deletion			
		W	Delete char at cursor
		R	Delete current line to spaces
		T	Delete to end of line
		t	
		Y	Delete to end of screen
		y	
		*	Delete Screen
		,	
		.	
		;	
		+	
		:	
	CTRL Z		

Die Befehlssequenz zur Änderung der Schreibfarbe (ESC ! c) ist nicht zu gängigen Terminalemulationen kompatibel. Neben der einfachen Änderung der Schreibfarbe kann durch Änderung des Grundwertes der Farbangabe folgendes Verhalten erzeugt werden:

Grundwert	**Verhalten**
32 (Leerzeichen)	Änderung der Schreibfarbe
64 (A)	Änderung der Hintergrundfarbe

Die Angaben für mit *Ps*, *Pc*, *Pn*, *r* oder *c* gekennzeichnete Zahlenwerte erfolgen durch Werte, die sich ASCII--codiert aus 32 + Zahlenwert errechnen.

6.3 Umsetzung der Eingabezeichen

Bei der Matrixtastatur sind unterschiedliche Belegungen möglich. So können z.B. sämtliche Tasten als frei programmierbare Funktionstasten zur Verfügung gestellt werden. Bei der PC--Tastatur sind die normalen Funktionstasten belegt, die Tasten von 13 bis 24 werden durch gleichzeitiges Drücken von `Shift` und `Funktionstaste` erreicht.
Standardmäßig liefern die Funktionstasten

`SOH` ***zeichen*** `CR`

Mit folgendem *zeichen*

Funktionstaste	***Zeichen***	**Shift+F-Taste**	***Zeichen***
1	@	13	´
2	A	14	a
3	B	15	b
4	C	16	c
5	D	17	d
6	E	18	e
7	F	19	f
8	G	20	g

9	H	21	h
10	I	22	i
11	J	23	j
12	K	24	k

Bei Matrixtastaturen hängt die Zuordnung der einzelnen Tasten zu den Tastencodes von dem Anschluß der Tastatur ab und muß für jeden Anwendungsfall ermittelt werden.

Die Programmierung der Funktionstasten erfolgt mit der Befehlssequenz

ESC | *p1* 1 *text* CTRL-Y

Und den Parametern

p1 Kennzeichnung der zu programmierenden Taste

text gewünschte Tastenbelegung

Funktionstaste	Kennung	Shift+F-Taste	Kennung
1	1	13	<
2	2	14	=
3	3	15	>
4	4	16	?
5	5	17	@
6	6	18	A
7	7	19	B
8	8	20	C
9	9	21	D
10	:	22	E
11	;	23	F
12	G	24	L

6.4 Grafikfunktionen

Bei grafikfähigen Displays stellt die Terminalemulation Funktionen zur Nutzung der grafischen Fähigkeiten des Displays zur Verfügung. Der Leistungsumfang richtet sich nach der Leistungsfähigkeit des jeweiligen Grafik-Displays bzw. des verwendeten Controllers.

Die Grafikfunktionen sind aus Geschwindigkeitsgründen nicht auf Multi-Tasking ausgelegt. Grundsätzlich sollte ein Grafiksystem nur von einer Task angesprochen werden. Besonders der kombinierte Betrieb von Terminalemulation und Grafikfunktionen erfordert besondere Aufmerksamkeit.

Neben den Grafik-Primitiven SETPIX, GETPIX und LINE (s. RTOS-UH-Handbuch Seite C-IV-8) gibt es folgende Prozeduren:

6.4.1 Funktionsumfang und Implementierungsabhängigkeiten

Prozedur	IP-SVGA	PVGA	VHI	NBS300
BACK_PEN	Ja	Ja	—	—
BIN_TEXT	Ja	Ja	Ja	Ja
BOX	Ja	Ja	Ja	Ja
BOX_FILLED	Ja	Ja	Ja	Ja
BOX_MOVE	Ja	Ja	Ja	Ja
BOX_READ	Ja	Ja	—	—
BOX_WRITE	Ja	Ja	—	—
CIRCLE	Ja	Ja	Ja	Ja
CLEAR	Ja	Ja	Ja	Ja
COLOR_CLEAR	Ja	Ja	—	—
DISPLAY_MODE	Ja	Ja	Ja	Ja
DISPLAY_STATE	Ja	Ja	—	—
DOT_LINE	Ja	Ja	Ja	Ja
DOT_LINE_RANGE	Ja	Ja	Ja	Ja
FILLED_HISTOGRAM	Ja	Ja	Ja	Ja
GET_PEN	Ja	Ja	—	—
GET_BACK_PEN	Ja	Ja	—	—
GET_TEXT_FONT	Ja	Ja	—	—
GET_TEXT_WIDTH	Ja	Ja	Ja	Ja
GRAPH_CURSOR	Ja	Ja	—	—
GRAPH_CURSOR_SELECT	Ja	Ja	—	—
HIDE_CURSOR	Ja	Ja	—	—
HISTOGRAM	Ja	Ja	Ja	Ja
PEN	Ja	Ja	—	—
PLANE	—	Ja	—	—
PLINIT	Ja	Ja	Ja	Ja
POLYGON	Ja	Ja	Ja	Ja
SELECT_FONT	Ja	Ja	—	—
SELECT_TEXT_FONT	Ja	Ja	—	—
SET_TEXT_WIDTH	Ja	Ja	Ja	Ja
SET_VGA_SCREEN	Ja	Ja	—	—
TEXT	Ja	Ja	Ja	Ja
VGA_GET_PLANE	Ja	Ja	—	—
VGA_LOCK_TEXT	Ja	Ja	—	—
VGA_SET_PLANE	Ja	Ja	—	—
VGA_UNLOCK_TEXT	Ja	Ja	—	—
WIDTH	Ja	Ja	Ja	Ja

6.4.2 Allgemeines

Zum Verständnis der Funktionsbeschreibung sind Kenntnisse über die folgenden Begriffe erforderlich:

- Farbe

  Farben werden in RGB-Notation angegeben. Die Angabe kennzeichnet die Intensität, in der der jeweilige Farbanteil dargestellt wird.

  Der bei den verschiedenen Grafikfunktionen erforderliche `col`-Parameter gibt jedoch üblicherweise nicht direkt die Zeichenfarbe als RGB-Wert an. Vielmehr enthält diese Größe folgende Daten:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zeichenmode				Farbpalette											

  Farbpalette
  gibt die Nummer einer Farbpalette an, die den tatsächlichen RGB-Farbwert enthält. Die Anzahl der verfügbaren Paletten und damit der größtmögliche Farbpaletten-Wert hängen vom verwendeten Grafiksystem ab.

  Zeichenmode
  Bei unterschiedlichen Grafiksystemen stehen unterschiedliche Zeichenmodi zur Verfügung. Folgende Zeichenmodi sind verfügbar:

Kennung	**Mode**	**Verfahren**
0	absolut	Es wird mit der angegebenen Farbe gezeichnet
2	Invers	Es wird mit invertiertem Farbwert gezeichnet.
4	AND	Die Zeichenfarbe ergibt sich aus der bitweisen UND-Verknüpfung der angegebenen Farbe und der aktuellen Farbe des darzustellenden Bildpunkts.
8	OR	Die Zeichenfarbe ergibt sich aus der bitweisen ODER-Verknüpfung der angegebenen Farbe und der aktuellen Farbe des darzustellenden Bildpunkts.
12	XOR	Die Zeichenfarbe ergibt sich aus der bitweisen EXKLUSIV-ODER-Verknüpfung der angegebenen Farbe und der aktuellen Farbe des darzustellenden Bildpunkts.

  Die Verknüpfung unterschiedlicher Zeichenmodi ist möglich.

- Pixel

  Kennzeichnet einen Bildpunkt. Je nach verwendetem Grafiksystem kann ein Bildpunkt unterschiedliche Farben annehmen.

- Pen

  Ein Pen ist durch eine Farbe gekennzeichnet.

  Es gibt einen Schreib- und einen Hintergrund-Pen. Pixel, Linien u.ä. werden nur unter Verwendung des Schreib-Pens gezeichnet.

  Vorgegebene Konstrukte wie Fonts o.ä. füllen den Zeichenhintergrund mit dem Hintergrund-Pen und stellen die Zeichen mit dem Schreib-Pen dar.

- Font

  Ein Font (Zeichensatz) ist als Bitmap abgelegt. Alle Zeichen eines Fonts haben gleiche Größe. Die Größe eines Fonts beinhaltet Ober- und Unterlängen.

  Die Bitmap enthält Informationen darüber, welche Pixel der von einem Zeichen eingenommen Fläche mit dem Schreib-Pen und welche Pixel mit dem Hintergrund-Pen darzustellen sind. Ein Font selber enthält keine Farbinformationen.

- Palette oder Plane

  Die im Bildschirmspeicher abgelegten Pixel-Farbinformationen werden über Paletten in tatsächliche Farbintensitätswerte übersetzt.

  Die Pixel-Farbinformation kann in diesem Fall mit geringem Speicherbedarf abgelegt werden (z.B. 4 Bit). Diese Information wird als Index beim Zugriff auf die Palette verwendet. Die Palette liefert dann die tatsächliche Farbinformation als RGB-Wert mit deutlich höherer Farbauflösung (z.B. 3x 8 Bit).

- Screen

  Ein Screen ist die Menge aller für einen sichtbaren Bildschirm im Bildspeicher abgelegten Daten. Bei Grafiksystemen, die entweder mehrere unterschiedliche Bildschirme unterstützen oder im Bildspeicher gleichzeitig Daten für mehrere Bildschirme ablegen können, ist die Wahl des Screens, auf dem die folgenden Zeichenoperationen erfolgen sollen, möglich.

6.4.3 Funktionsreferenz

PEARL-Spezifikation	**Funktion**

`BACK_PEN: PROC(`
`  col  FIXED(15)`
`) GLOBAL;`
	Setzt die angegebene Farbe `col` als Hintergrundfarbe. Bei folgenden Operationen, die eine Hintergrundfarbe darstellen müssen, aber keine eigene Hintergrundfarbe spezifizieren (`TEXT` und `BIN_TEXT`), wird `col` als Hintergrundfarbe verwendet.

`BIN_TEXT: PROC(`
`  (x, y)  FIXED(15),`
`  col  FIXED(15),`
`  zeichen_adr  FIXED(31)`
`) GLOBAL;`
	Überträgt ein Zeichen aus dem Zeichengenerator--ROM in der Farbe `col` auf den Bildschirm an die Position (`x`, `y`). Die Adresse im ROM wird mit `zeichen_adr` angegeben. Sie hängt von der Größe des auszugebenden Zeichens ab:
	$zeichen\_adr = \frac{xhöhe * ybreite + 7}{8}$

BOX: PROC(
  (x, y) FIXED(15),
  (Xend, Yend) FIXED(15),
  col FIXED(15)
) GLOBAL;
	Zeichnet einen rechteckigen Rahmen in der Farbe `col` mit den diagonalen Punkten (`x`, `y`) (linke obere Ecke) und (`Xend`, `Yend`) (rechte, untere Ecke).

BOX_FILLED: PROC(
  (x, y) FIXED(15),
  (Xend, Yend) FIXED(15),
  col FIXED(15)
) GLOBAL;
	Zeichnet ein in der Farbe `col` gefülltes Rechteck mit den diagonalen Punkten (`x`, `y`) und (`Xend`, `Yend`)

BOX_MOVE: PROC(
  (Xstart, Ystart) FIXED(15),
  (Breite, Hoehe) FIXED(15),
  (Xziel, Yziel) FIXED(15),
  mode FIXED(15)
) GLOBAL;
	Kopiert einen rechteckigen Bildausschnitt von `Breite` und `Höhe` beginnend bei dem Startpunkt (`Xstart`, `Ystart`) auf einen Bildbereich gleicher Größe mit dem Startpunkt (`Xziel`, `Yziel`).
	Der Zielbereich kann außerhalb des sichtbaren Bildschirmbereichs liegen, d.h. `Ywidth` wird überschritten.
	`mode` legt die Zeichenart fest, im unteren Nibble sind folgende Werte zulässig:
	
	Dez. hex Art
	12 $0C absolut
	3 $03 not
	8 $08 and
	14 $0E or
	6 $06 exor
	
	Das nächste Nibble legt die Schreibfarbe fest, d.h. für ein Pixel werden die 4 Bit aus dem Video-RAM entsprechend dem unteren Nibble gelesen, mit der Schreibfarbe "verundet"' und wieder im Video-RAM abgelegt. Wird der Parameter `mode` auf absolut (=12) gesetzt, so wird der original Bildausschnitt an der Zielstelle abgelegt.

BOX_READ: PROC(
  (Xstart, Ystart) FIXED(15),
  (Breite, Hoehe) FIXED(15),
  (Xziel, Yziel) FIXED(15),
  feld STRUCT[REF CHAR(1)]
) GLOBAL;
	Kopiert einen rechteckigen Bildausschnitt mit `Breite` und `Hoehe` beginnend bei dem Startpunkt (`Xstart`, `Ystart`) in den Speicherbereich, der mit `feld` angegeben wird. Dabei findet **keine** Überprüfung der Größe des Speicherbereiches statt. Ist das angegebene `feld` zu klein, ist der Absturz des Rechners so gut wie sicher!

BOX_WRITE: PROC(
  (Xstart, Ystart) FIXED(15),
  (Breite, Hoehe) FIXED(15),
  (Xziel, Yziel) FIXED(15),
  mode FIXED(15),
  feld STRUCT[REF CHAR(1)]
) GLOBAL;
	Kopiert den Speicherbereich, der mit `feld` angegeben wird, auf einen rechteckigen Bildausschnitt von `Breite` und `Höhe`, beginnend bei dem Startpunkt (`Xstart`, `Ystart`).
	Dabei findet **keine** Überprüfung der Größe des Speicherbereiches statt. Ist das angegebene `feld` zu klein, wird der folgende Speicher auf den Bild-

	schirm geschrieben. Der Parameter `mode` ist bei der Prozedur `BOX_MOVE` beschrieben.
CIRCLE: PROC((Xmitte, Ymitte) FIXED(15), Radius FIXED(15), col FIXED(15)) GLOBAL;	Zeichnet geschlossene Kreislinien auf den Bildschirm.
CLEAR: PROC GLOBAL;	Löscht den sichtbaren Bildschirmbereich mit der Farbe 0.
COLOR_CLEAR: PROC(col FIXED(15)) GLOBAL;	Löscht den sichtbaren Bildschirmbereich mit der angegebenen Farbe `col`.
DISPLAY_MODE: PROC(mode FIXED(31)) GLOBAL;	Ermöglicht bei einigen Systemen die Umschaltung der Betriebsart des Grafiksystems (Text/Grafik).
DISPLAY_STATE: PROC(State FIXED(31)) GLOBAL;	Schaltet Display ein (`State` = 1) oder aus (`State` = 0)
DOT_LINE: PROC((Xstart, Ystart) FIXED(15), (XEnd, YEnd) FIXED(15), col FIXED(15), mask FIXED(15)) GLOBAL;	Zeichnet ein (strichpunktierte) Linie vom Punkt (`Xstart`, `Ystart`) zum Punkt (`XEnd, YEnd`) in der Farbe `col`. Mit `mask` wird das Linienmuster vorgegeben: -1: durchgezogene Linie TOFIXED ´AAAA'B4: einfach punktiert
DOT_LINE_RANGE: PROC((Xstart, Ystart) FIXED(15), (XEnd, YEnd) FIXED(15), col FIXED(15), mask FIXED(15), (YU, YO) FIXED(15), col2 FIXED(15),) GLOBAL;	Zeichnet ein (strichpunktierte) Linie vom Punkt (`Xstart`, `Ystart`) zum Punkt (`XEnd, YEnd`) in der Farbe `col`. Mit `mask` wird das Linienmuster vorgegeben: -1: durchgezogene Linie TOFIXED ´AAAA'B4: einfach punktiert Die Y-Koordinaten `YU` (unten) und `YO` (oben) sind Umschaltschwellen für die Zeichenfarbe. Liegt ein zu zeichnender Bildpunkt unterhalb `YU` oder oberhalb `YO`, so wird er in der Farbe `col2` anstatt in `col` dargestellt.
FILLED_HISTOGRAMM: PROC((XS, YS) FIXED(15) IDENT, count FIXED(15), col FIXED(15), b_col FIXED(15), width FIXED(15), ySave FIXED(15) IDENT) GLOBAL;	Zeichnet ein Histogramm aus `count` nicht gefüllten Balken der Breite `width` in der Farbe `col`. Die einzelnen Histogrammbalken werden durch ihre linke, obere Ecke (`XS, YS`) und die Breite `width` bestimmt. `XS` und `YS` müssen in getrennten FIXED(15)-Feldern gegeben werden. Die Y-Grundlinie des Histogramms wird durch die Initialwerte im Feld `ySave` gegeben. Zur Erhöhung der Zeichengeschwindigkeit bei wiederholter Darstellung eines Histogramms wird nur die Änderung der Balkenhöhe gegenüber der im Feld `ySave` gespeicherten Höhe neu gezeichnet.

GET_PEN: PROC RETURNS(FIXED(15)) GLOBAL;	Gibt die Farbe des aktuell gewählten Schreib-Pens zurück.
GET_BACK_PEN: PROC RETURNS(FIXED(15)) GLOBAL;	Gibt die Farbe des aktuell gewählten Hintergrund-Pens zurück.
GET_TEXT_FONT: PROC RETURNS(FIXED(15)) GLOBAL;	Gibt die Nummer des aktuellen Textfonts zurück (von SELECT_FONT oder SELECT_TEXT_FONT gesetzt). Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem.

GET_TEXT_WIDTH: PROC(
  Xhoehe FIXED(15) IDENT,
  ybreite FIXED(15) IDENT
) GLOBAL;
	Liefert die Größe eines Zeichens des gewählten Zeichensatzes in den Variablen xhoehe und ybreite zurück.

GRAPH_CURSOR: PROC(
  (x, y) FIXED(15),
  status FIXED(15)
) GLOBAL;
	Positioniert den Graphik-Cursor auf den Punkt (x, y). Mit status = 1 wird der Graphik-Cursor sichtbar, mit status = 0 wird er unsichtbar.

GRAPH_CURSOR_SELECT: PROC(
  Cursor FIXED(31) IDENT,
  mode FIXED(15)
) GLOBAL;
	Gibt eine Bitmap zur Darstellungsform des Graphik-Cursors vor. Cursor muß das erste Element eines FIXED(31)-Feldes mit 32 Elementen sein; damit wird ein 32x32 Bit-Cursor beschrieben. Gesetzte Bits entsprechen darzustellenden Pixeln.
	mode gibt die Darstellungsart des Cursors an.

HIDE_CURSOR: PROC GLOBAL;	Schaltet den Graphik-Cursor aus.

HISTOGRAMM: PROC(
  (XS, YS) FIXED(15) IDENT,
  count FIXED(15),
  col FIXED(15),
  b_col FIXED(15),
  (offs, width) FIXED(15),
  ySave FIXED(15) IDENT
) GLOBAL;
	Zeichnet ein Histogramm aus count nicht gefüllten Balken der Breite width in der Farbe col.
	Die Y-Grundlinie des Histogramms wird in offs gegeben.
	Die einzelnen Histogrammbalken werden durch ihre linke, obere Ecke (XS, YS) und die Breite width bestimmt.
	XS und YS müssen in getrennten FIXED(15)-Feldern gegeben werden.
	Zur Erhöhung der Zeichengeschwindigkeit bei wiederholter Darstellung eines Histogramms wird nur die Änderung der Balkenhöhe gegenüber der im Feld ySave gespeicherten Höhe neu gezeichnet. Bei der erstmaligen Darstellung eines Histogramms muss ySave daher vollständig mit dem Wert offs initialisiert werden.

PEN: PROC(col FIXED(15)) GLOBAL;	Setzt die angegebene Farbe col als Schreibfarbe. Bei folgenden Operationen, die keine eigene Hin-Schreibfarbe spezifizieren, wird col als Schreibfarbe verwendet.

PLANE: PROC(
  Farbnummer FIXED(15),
  Palettenwert FIXED(31)
) GLOBAL;
	Setzt den Wert eines Paletteneintrags. Für Farbnummer sind Werte von 0 bis 15 zulässig. Folgende Bits vom Palettenwert werden genutzt (abhängig

	vom Grafiksystem): **Farbe Bit (PEARL) Bit(Assembler)** Rot 12 ... 4 18 ... 20 Grün 20 ...22 10 ... 12 Blau 28 ...30 2 ... 4 Der Aufbau des Langwortes `Palettenwert` läßt sich durch die Einzelbit-Darstellung `[---- ---- ---r rr-- ---g gg-- ---b bb--]` veranschaulichen.
`PLINIT: PROC GLOBAL;`	Initialisiert den Display--Prozessor. Muß vor Benutzung der Grafik-Routinen einmal aufgerufen werden.

`POLYGON: PROC(`
`  (x,y)  FIXED(15) IDENT,`
`  cnt  FIXED(15,`
`  col  FIXED(15)`
`) GLOBAL;`
	Zeichnet einen (nicht geschlossenen) Linienzug mit `cnt` Stützpunkten in der Farbe `col`.
	`x` und `y` müssen in getrennten, eindimensionalen Feldern abgelegt sein.

`SELECT_FONT: PROC(`
`  Font  FIXED(15)`
`) GLOBAL;`
	Wählt den angegeben Font als aktuellen Textfont aus.
	Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem.

`SELECT_TEXT_FONT: PROC(`
`  Font  FIXED(15)`
`) GLOBAL;`
	Wählt den angegeben Font als aktuellen Textfont aus.
	Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem.

`SET_TEXT_WIDTH: PROC(`
`  Xhöhe  FIXED(15) IDENT,`
`  ybreite  FIXED(15) IDENT`
`) GLOBAL;`
	Setzt die Größe eines Zeichens für die Funktionen `BIN_TEXT` und `TEXT`.

`SET_VGA_SCREEN: PROC(`
`  Screen  FIXED(15)`
`)`
`  RETURNS( FIXED(15) )`
`GLOBAL;`
	Wählt den angegebenen `Screen` als Ziel der folgenden Graphikoperationen aus.
	Der Rückgabewert ist –1, falls der angegebene `Screen` im aktuellen Grafiksystem nicht vorhanden ist.

`TEXT: PROC(`
`  (x, y)  FIXED(15),`
`  col  FIXED(15),`
`  string  STRUCT`
`  [ str CHAR(255),`
`  len FIXED(15)`
`  ]`
`) GLOBAL;`
	Schreibt die in `string.str` abgelegte Zeichenkette beginnend auf der Position (`x`, `y`) (linke, obere Ecke des ersten Zeichens) in der Farbe `col` auf den Bildschirm. Die Ausgabe endet, wenn `string.len` Zeichen ausgegeben wurden.
	Die Schreibrichtung ist horizontal (x-Richtung).

`VGA_GET_PLANE: PROC(`
`  Plane  FIXED(15),`
`  (R, G, B)  FIXED(15) IDENT`
`) GLOBAL;`
	Liest aus der Palette `Plane` den enthaltenen RGB-Wert aus.

`VGA_LOCK_TEXT: PROC`
`  RETURNS( FIXED(15) )`
`GLOBAL;`
	Sperrt das Grafiksystem gegenüber den Aktionen der Terminalemulation.

	Diese Funktion muß jeweils vor einem Block von zusammenhängenden Grafikoperationen aufgerufen werden, falls gleichzeitig die Terminalemulation genutzt werden soll. Der Rückgabewert ist 1, falls die Operation erfolgreich war.

VGA_SET_PLANE: PROC(
  Plane FIXED(15),
  (R, G, B) FIXED(15)
) GLOBAL;
	Setzt die Palette `Plane` auf einen angegebenen RGB-Wert. Plane wird beim Zeichnen über die `col` angesprochen

VGA_UNLOCK_TEXT: PROC RETURNS(FIXED(15)) GLOBAL;	Gibt das Grafiksystem nach einem Aufruf von `VGA_LOCK_TEXT` wieder für Aktivitäten der Terminalemulation frei. Der Rückgabewert ist 1, falls die Operation erfolgreich war.

WIDTH: PROC(
  Xwidth FIXED(15) IDENT,
  Ywidth FIXED(15) IDENT
) GLOBAL;
	Gibt in den Variablen `Xwidth` und `Ywidth` die Größe des Bildschirms in Pixel zurück.