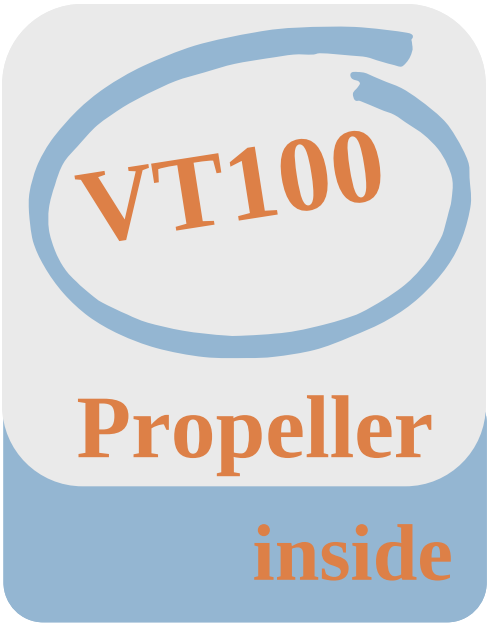
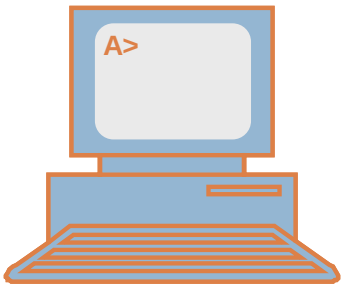
RETROCOMPUTING

VT100
VIDEO TERMIAL



Joe G.

Dokumentation



ÄNDERUNGSVERZEICHNIS

Änderung			geänderte Kapitel	Beschreibung	Autor	neuer Zustand
Nr.	Datum	Version				
1	16.01.2015	V01.00		Startversion	Joe G.	OK
2	27.04.2015	V01.01	RS-232	RS-232-Schnittstelle	Joe G.	OK
3	03.05.2015	V01.02	Hardware	Aufbau und Inbetriebnahme	Joe G.	OK
4	14.09.2015	V01.03	Hardware	Aufbau und Inbetriebnahme	Joe G.	OK
5	21.12.2015	V01.04	Anlagen	Zeichensatz / Control-Codes	Joe G.	OK
6	23.12.2015	V01.05	Anlagen	NRC Sets	Joe G.	OK

Retrocomputing

EINFÜHRUNG

Das VT100 Video Terminal ist ein Computerterminal, welches 1978 von der Digital Equipment Corporation (DEC) aus dem VT52 entwickelt wurde. Es hat sich im laufe der Jahre zum de facto Standard für Terminals entwickelt. Noch heute unterstützen zahlreiche Terminal-Emulatoren diesen Standard. Erst im Jahre 1983 wurde das VT100 von einem verbesserten Standard wie dem VT220 ersetzt.

Die Hardware des ersten VT100 Terminals basierte auf einem Industrie-Standard-Mikroprozessor, dem Intel 8080. Optional gab es zusätzliche Anschlüsse für einen Drucker oder für eine Grafikerweiterung. Auf seinem 12 Zoll Monochrom-Bildschirm konnte das Terminal 24 Zeilen zu 80 Zeichen oder alternativ 14 Zeilen zu 132 Zeichen darstellen. Die Ausgabe über eine Semigrafik wurde durch Sonderzeichen und ANSI-Escape-Sequenzen ermöglicht. Das Terminal verfügte über eine erweiterte Schreibmaschinen-Tastatur (83 Tasten, QWERTY-Belegung) mit einigen Funktionstasten. Weiterhin existierten Folgeprodukte mit unterschiedlicher Hardwareausstattung. Erwähnenswert ist hier z. B. das VT180 (Codename "Robin"). Es beinhaltete zusätzlich noch einen Zilog Z80 Mikroprozessor und konnte direkt CP/M ausführen.

Hardware-Plattform

Das hier vorgestellte VT100 Terminal baut auf einem Parallax Propeller (P8X32A) auf und emuliert die ursprünglichen VT100 Terminalcodes. Der verwendete Mikrocontroller beinhaltet 8 unabhängig arbeitende 32-Bit-RISC CPU Kerne. Sie übernehmen dabei die Funktionen:

- Videodarstellung (VGA-Text-Mode)
- PS/2-Tastaturcontroller
- UART
- GPIO
- USB Schnittstelle

Wie aus Abb. 1 ersichtlich, kann der Controller über eine Logik immer nur auf einen externen seriellen Kanal zugreifen. Dabei dient die USB-Schnittstelle eigentlich nur der Programmierung und der Inbetriebnahme. Im tatsächlichen Terminalbetrieb wird der serielle Kanal des Propeller auf den UART Controller geschaltet. Die USB-Schnittstelle dient nur noch zur externen Stromversorgung des VT-100 Terminals. Dazu kann ein externes 5V Steckernetzteil am USB-Port angeschlossen werden.

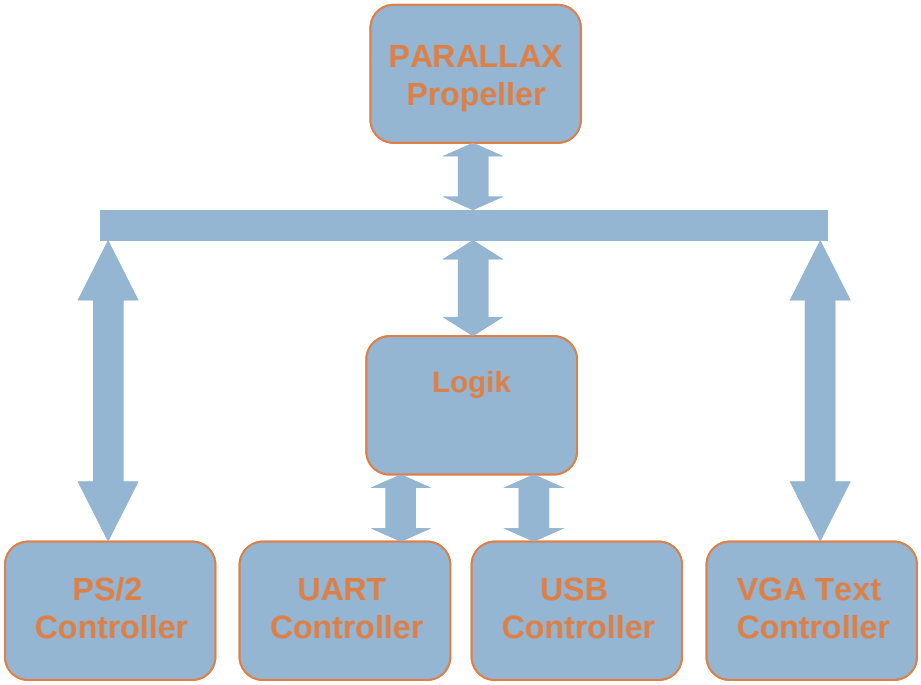


Abb.1: Prinzipaufbau der VT100 Hardware

HARDWARE

Die gesamte Hardware ist auf einer doppelseitigen Leiterplatte (SMD-Bestückung) mit dem Abmaßen 68 mm x 100 mm untergebracht (Abb. 2).

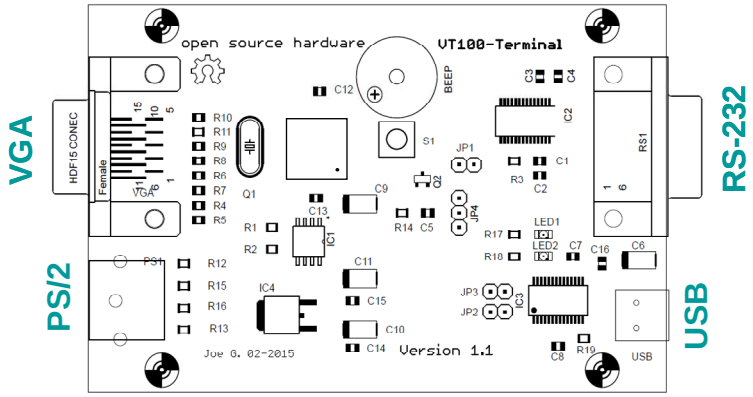


Abb.2: Bestückungsseite mit Anschlussbuchsen des VT100-Terminals

Bestückung und Inbetriebnahme

Die Bestückung und Inbetriebnahme erfolgt Baugruppenweise. Damit werden Bestückungsfehler weitgehend vermieden und eine Funktionskontrolle ermöglicht.

Im ersten Schritt wird die USB-Schnittstelle mit IC3, den zugehörigen passiven Bauelementen und der USB-Buchse bestückt (Abb.4). Wird nun die Schaltung mit dem PC verbunden, sollte im Gerätemanager des

Betriebssystems der USB-Port als virtueller COM-Port angezeigt werden (Abb.3). Die RX und TX LED's leuchten bei einer Verbindungsaufnahme noch nicht auf, da die zugehörige 3.3V Spannungsversorgung noch fehlt.

Abb.3: USB-Schnittstelle als virtueller COM-Port

Nach der Bestückung von IC4 (Spannungsregler) und den zugehörigen Abblockkondensatoren können die Versorgungsspannungen auf der Leiterplatte kontrolliert werden (Abb.4).

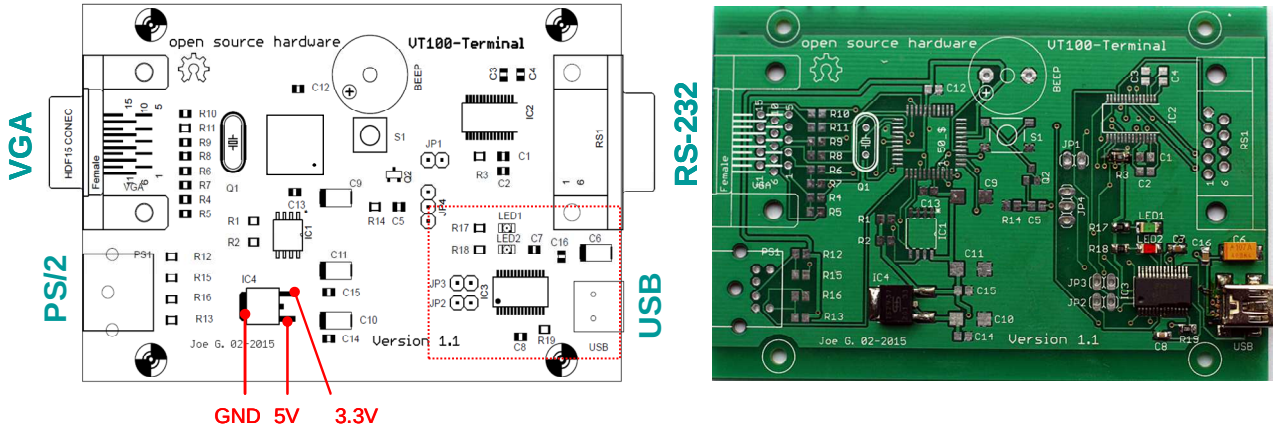


Abb.4: USB-Schnittstelle und Spannungsregler

Unabhängig ob der virtuelle COM-Port Treiber des Host PC IC3 erkennt, sollten bei einem fehlerfreien Aufbau +5 V am Festspannungsreglereingang anliegen. Der Ausgang des Spannungsreglers liefert genau +3.3 V für die Versorgung des Propeller IC. Können am Festspannungsregler keine +5V und keine +3.3V gemessen werden, ist der Aufbau auf mögliche Kurzschlüsse an der Mini-USB Buchse bzw. am IC3 zu untersuchen. Das gleiche gilt für einen fehlenden virtuellen COM-Port-Eintrag im Betriebssystem. Auch hier sind IC3 und Mini-USB auf mögliche Lötbrücken zu kontrollieren.

Im nächsten Schritt wird IC5 (P8X32A) bestückt. Da das Gehäuse quadratisch ist, kann es schnell zu einer Fehlbestückung kommen. Achtung, Pin 1 ist durch einen Punkt (Abb.5) gekennzeichnet. Weiterhin ist darauf zu achten, dass alle Pins mittig auf den zugehörigen Pads liegen. Er dann darf das IC komplett verlötet werden. Anschließend wird IC1 (EEPROM) sowie die RESET-Schaltung um Q2 und der 5 MHz Quarz bestückt. JP4 ist so zu stecken, dass Pin 1 und Pin 2 gebrückt sind (siehe Abb.5). In dieser Jumperstellung wird der RESET-Impuls durch das /DTR-Signal an der USB-Schnittstelle ausgelöst.

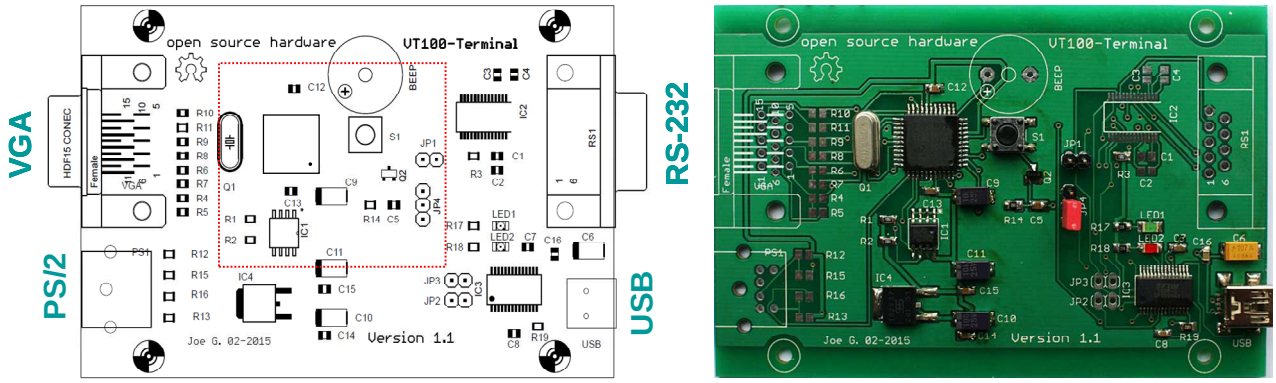


Abb.5: Bestückung des P8X32A, EEPROM, RESET-Schaltung und Quarz

An dieser Stelle kann bei einem korrektem Aufbau testweise eine Verbindung zum P8X32A (Propeller) hergestellt werden. Dazu wird zunächst die entsprechende Software von der PARALLAX-Internetseite geladen und installiert [1]. Neben dem offiziellen SPIN-Compiler von PARALLAX, existieren noch eine ganze Reihe von ähnlichen Compilerlösungen unter den Betriebssystemen Linux und Windows. Die vorliegende Beschreibung beschränkt sich jedoch nur auf die Windows-Version des Editor/Development Systems. Arbeiten USB-Schnittstelle und Propeller fehlerfrei, kann über die Funktionstaste F7 die korrekte Kommunikation mit dem Parallax-IC überprüft werden (Abb.6). Der zugehörige virtuelle COM-Port wird dabei automatisch durch die Software bestimmt. Wird der Propeller und die RESET-Logik korrekt erkannt, erfolgt eine entsprechende Mitteilung (Abb.6).

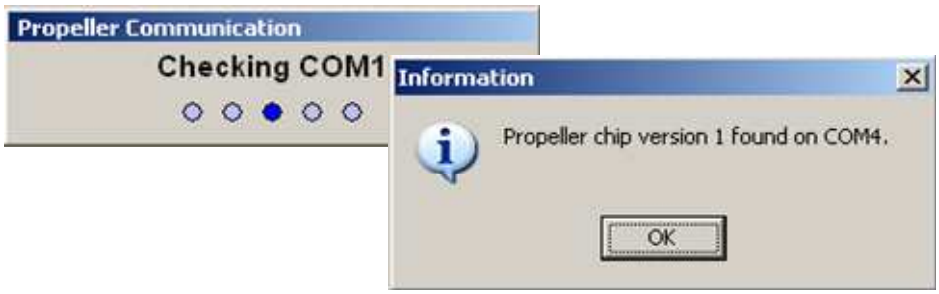


Abb.6: Kommunikationstest des Editor/Developmentsystems mit der Hardware

Nun können die Steckverbinder für den VGA-Monitor und die PS/2 Tastatur sowie die zugehörigen passiven Bauelemente bestückt werden (Abb.7).

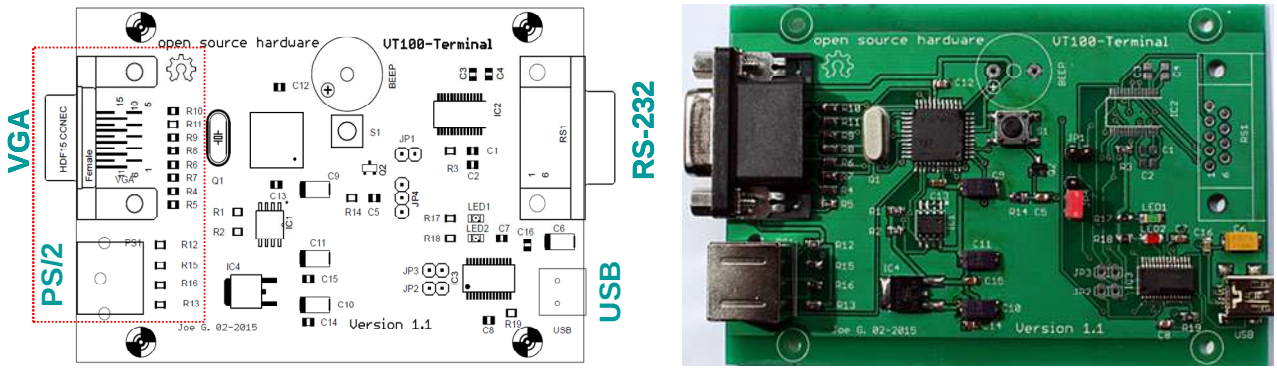


Abb.7: VGA- und PS/2 Anschlüsse

Im letzten Arbeitsschritt werden der RS232-Treiber sowie der zugehörige DE-9 Steckverbinder bestückt (Abb.8).

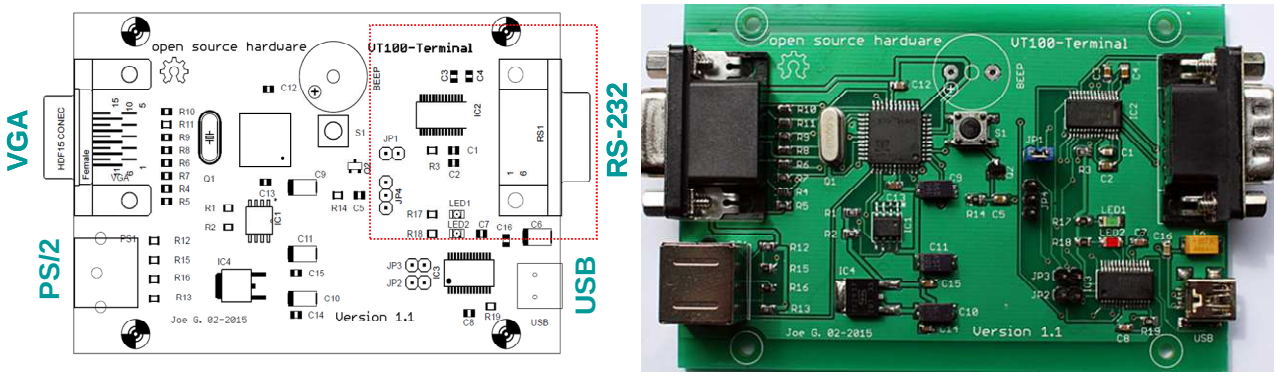


Abb.8: RS-232 Treiber und DE-9 Steckverbinder

Damit ist der Aufbau abgeschlossen und die VT-100 Software kann in den EEPROM programmiert werden.

Das geschieht wiederum über das auf dem PC installierte Editor/Development System. Über die Funktionstaste F8 wird dazu das Object Info Fenster aufgerufen (Abb.9). Nun kann das compilierte Binärfile über den Button **Open File** geladen werden. Mit dem Button **Load EEPROM** wird das Binärfile in den EEPROM übertragen und ein RESET am Propeller ausgelöst. Das VT100-System ist nun funktionsfähig.

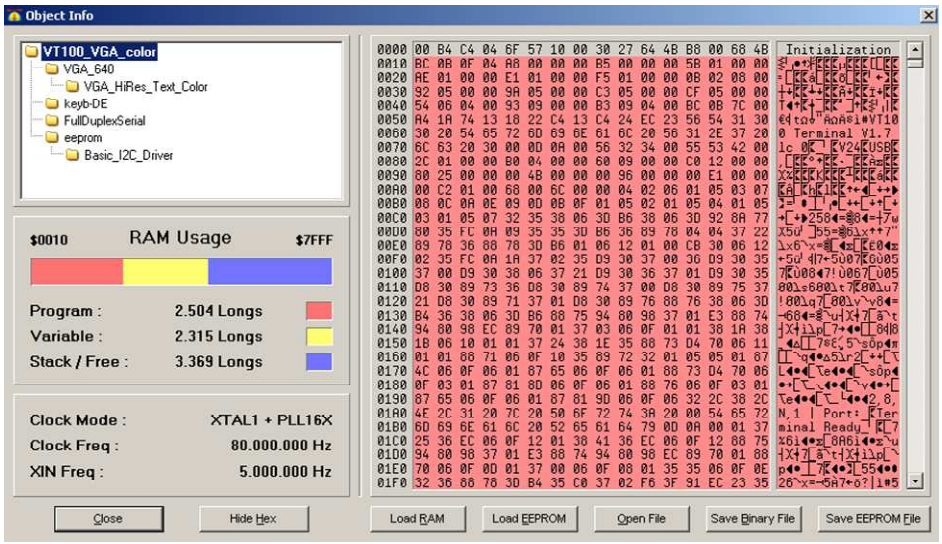


Abb.9: Object Info Tool zur Programmierung

VT100 Firmware

Die Propellerfirmware stellt ein Terminal mit VT100 Funktionalität bereit. Entgegen der ursprünglichen Monitorauflösung von 24 Zeilen zu 80 US-ASCII-Zeichen (12 Zoll monochrom) werden 30 Zeilen zu 80 ASCI-Zeichen incl. der deutschen Umlaute dargestellt.

Die Ausgabe zeichenorientierter Grafik (Semigrafik), inverser oder blinkender Schrift wird durch ANSII-Escape-Sequenzen ermöglicht. Der PS/2 Port ermöglicht den Anschluss einer handelsüblichen PC-Tastatur mit 104 Tasten. Über zwei serielle Schnittstellen (RS-232 / USB) kann das VT100 Terminal an einen Host-Rechner angeschlossen werden. Erfolgt der Anschluss des Host-Rechners nur über die RS-232, so kann die Stromversorgung des VT100 Terminals über ein handelsübliches 5V Steckernetzteil über die Mini-USB Buchse erfolgen.

Nach dem Anlegen der Stromversorgung meldet sich das Terminal mit seiner Einschaltmitteilung. Hier wird die aktuelle Firmwareversion, die eingestellte Baudrate und das verwendete Protokoll sowie der aktuell verwendete serielle Port angezeigt. Über die Funktionstaste F1 kann ein Setup-Menü zur Änderung der voreingestellten Parameter aufgerufen werden. Die Änderungen erfolgen über die angegebenen Funktionstasten und anschließend werden diese Einstellungen dauerhaft in einem EEPROM gespeichert. Weitere Spezialfunktionen wie ALT+CTRL+DEL (Reset CP/M) sind dem Anhang im Handbuch zu entnehmen. Auch die derzeit implementierten ASII-Escape-Sequenzen sind im Anhang aufgeführt.

ANLAGEN

RS232 – Schnittstelle

Ein Standard für eine serielle Kommunikation zur Datenübertragung ist die RS232 Schnittstelle. Sie definiert die Signale, die zwischen einem DTE (**D**ata **T**erminal **E**quipment) wie beispielsweise ein Computer-Terminal, und einem DCE (**D**ata **C**ommunication **E**quipment) wie beispielsweise ein Modem ausgetauscht werden. Der RS232-Standard wird sehr häufig bei Computern mit seriellem Anschluss verwendet. Die Norm definiert die elektrischen Eigenschaften sowie das Timing der Signale, die Funktion der Signale sowie die Belegung der Anschlüsse.

Die RS232 Schnittstelle wurde ursprünglich also dafür geschaffen, um Computerterminals (DTE) an langsame Modems (DCE) anzuschließen. Um beide Geräte zu unterscheiden, hat das Terminal einen SUB-D9-Stecker und das Modem eine SUB-D-9-Buchse (Abb.1). Verbunden werden beide Geräte mit einem einfachen seriellen Kabel (ACHTUNG kein Null-Modem-Kabel).

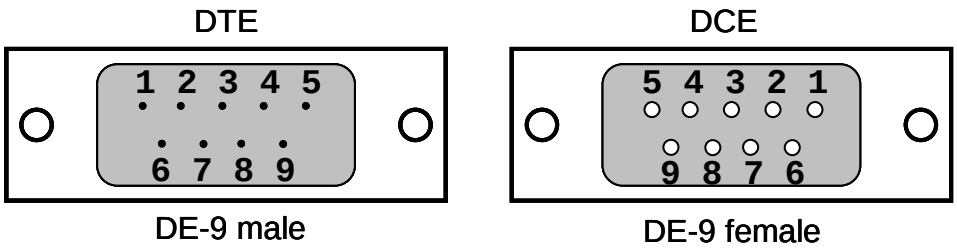


Abb1.: Pinbelegung der Terminalverbinder

RTS/CTS

Bei den frühere DFÜ-Geschwindigkeiten konnte das Modem die Daten nicht so schnell versenden, wie das Terminal sie bereitstellen konnte. Deshalb gibt es im RS232-Standard zwei Steuerleitungen, mit denen der Datenfluss gesteuert werden kann.

Die dafür verwendeten Leitungen sind RTS und CTS. Am Terminal gibt es einen RTS-Ausgang und einen CTS-Eingang. Am Modem gibt es einen RTS-Eingang und einen CTS-Ausgang. Wenn Hardwarehandshake verwendet wird, dann aktiviert das Terminal zuerst die RTS-Leitung (**R**equest **T**o **S**end) und fragt damit beim Modem an, ob es bereit ist Daten zu empfangen. Wenn das Modem bereit ist, aktiviert es nun seinerseits die CTS-Leitung (**C**lear **T**o **S**end). Erst jetzt sendet der Sender auf der TXD-Datenleitung asynchron das Datenbyte mit Start- und Stopbits.

Ist das Modem irgendwann nicht mehr in der Lage die vom Terminal eintreffenden Daten schnell genug weiterzuverarbeiten, deaktiviert es die CTS-Leitung. Daraufhin unterbricht das Terminal den Datentransport solange, bis CTS wieder vom Modem aktiviert wird.

DTR/DSR

Zusätzlich existieren noch die beiden Signalleitungen DTR (**D**ata **T**erminal **R**eady) und DSR (**D**ata **S**et **R**eady). Das Terminal besitzt dazu einen DTR-Ausgang und einen DSR-Eingang. Am Modem gibt es passend einen DTR-Eingang und einen DSR-Ausgang. Beide Leitungen werden vom Terminal (DTR) und vom Modem (DSR) ständig auf ON gehalten. Sie zeigen dem jeweiligen Partner nur an, dass das andere Gerät angeschlossen und generell zur Arbeit bereit ist. Für die byteweise Flusskontrolle waren sie ursprünglich jedoch nicht vorgesehen. Bei modernen Geräten lassen sie sich aber ähnlich wie RTS und CTS einsetzen, um den Datenfluss vom Modem zum Terminal synchronisieren.

Zeichengenerator

Aufbau des 8 x 16 Font

Beispiel Zeichen „2“
long $3c000000,$38606666,$7e06060c,$00000000

8	4	2	1	8	4	2	1	Byte	Part
								00	Part 3
								00	
								00	
								00	
	■	■	■	■	■	■		7E	Part 2
					■	■		06	
					■	■		06	
				■	■			0C	
		■	■	■				38	Part 1
	■	■						60	
	■	■			■	■		66	
	■	■			■	■		66	
		■	■	■	■			3C	Part 0
								00	
								00	
								00	

Beispiel Zeichen „Ä“
long $3c002400,$7e666666,$66666666,$00000000

1	2	4	8	1	2	4	8	Byte	Part
								00	Part 0
		■			■			24	
								00	
		■	■	■	■			3C	
	■	■			■	■		66	Part 1
	■	■			■	■		66	
	■	■			■	■		66	
	■	■	■	■	■	■		7E	
	■	■			■	■		66	Part 2
	■	■			■	■		66	
	■	■			■	■		66	
	■	■			■	■		66	
								00	Part 3
								00	
								00	
								00	

NRC Sets

	0		1		2		3	4		5		6	7	
					DE	UK		DE	UK	DE	UK		DE	UK
0	NUL	^@	DLE	^P	SP		0	§	@	P		`	P	
1	SOH	^A	DC1	^Q	!		1	A		Q		a	q	
2	STX	^B	DC2	^R	“	£	2	B		R		b	r	
3	ETX	^C	DC3	^S	#		3	C		S		c	s	
4	EOT	^D	DC4	^T	$		4	D		T		d	t	
5	ENQ	^E	NAK	^U	%		5	E		U		e	u	
6	ACK	^F	SYN	^V	&		6	F		V		f	v	
7	BEL	^G	ETB	^W	‘		7	G		W		g	w	
8	BS	^H	CAN	^X	(		8	H		X		h	x	
9	HT	^I	EM	^Y	)		9	I		Y		i	y	
10	LF	^J	SUB	^Z	*		:	J		Z		j	z	
11	VT	^K	ESC	^[	+		;	K		Ä	[	k	ä	{
12	FF	^L	FS	^\	,		<	L		Ö	\	l	ö	|
13	CR	^M	GS	^]	-		=	M		Ü	]	m	ü	}
14	SO	^N	RS	^^	.		>	N		^		n	ß	~
15	SI	^O	US	^_	/		?	O		_		o	DEL	

Control-Codes

Stand: v1.7.2

Ps Numerischer, optionaler Parameter. bestehend aus einer oder mehrerer Ziffern.
Pm Durch ; getrennte Folge von numerischen Parametern Ps.

Control characters

```
BEL                   Generates bell tone.
BS                    Moves cursor to the left one character position; if cursor is at
                      left margin, no action occurs.
HT                    Moves cursor to next tab stop, or to right margin if there are no
                      more tab stops.
CR                    Moves cursor to left margin on current line.
LF                    Causes a linefeed.
VT                    Processed as LF.
FF                    Processed as LF.
SO                    Invoke G1 character set.
SI                    Invoke G0 character set. (default)
CAN                   If received during an escape or control sequence, cancels the
                      sequence.
SUB                   Processed as CAN.
ESC                   Processed as a sequence introducer.
```

Escape sequences

```
ESC D                 IND     Linefeed.
ESC E                 NEL     Newline.
ESC M                 RI      Reverse linefeed. (Curser up)
ESC Z                 DECID   DEC private identification. The terminal returns  ESC [ ?
                              6 c, claiming that it is a VT102.
ESC c                 RIS     Reset to Initial State (Power Up)
ESC ( A               SCS     Select Character Set UK  --> G0
ESC ( B               SCS     Select Character Set US ASCII --> G0
ESC ( 0               SCS     Select Character Set DEC special characters and line dra
                              wing set --> G0
ESC ( K               SCS     Select Character Set DE  --> G0
ESC ) A               SCS     Select Character Set UK  --> G1
ESC ) B               SCS     Select Character Set US ASCII  --> G1
ESC ) 0               SCS     Select Character Set DEC special characters and line dra
                              wing set  --> G1
ESC ) K               SCS     Select Character Set DE  --> G1
ESC 7                 DECSC   Save current state (cursor coordinates, attributes, cha
                              racter sets pointed at by G0, G1).
ESC 8                 DECRC   Restore state most recently saved by ESC 7.
ESC # 8               DECALN  DEC screen alignment test - fill screen with E's.
ESC # 9               Show Font (Debug)
```

Control sequences

```
ESC [ Ps @            ICH    Insert Ps blank characters  (defaut = 1) (VT220)
ESC [ Ps A            CUU    Move cursor up
ESC [ Ps B            CUD    Move cursor down
ESC [ Ps C            CUF    Move cursor right
ESC [ Ps D            CUB    Move cursor left
ESC [ Ps ; Ps H       CUP    Move cursor to [row; column] (default = [1,1])
ESC [ Ps J            ED     Erase in Display
        Ps = 0        Erase below (default)
```

```
         Ps = 1         Erase above
         Ps = 2         Erase all
ESC [ Ps K              EL     Erase in line
        PS = 0          EL     Erase to right (default)
        PS = 1          EL     Erase to left
        PS = 2          EL     Erase all
ESC [ Ps L              IL     Insert Ps lines (defaut = 1)
ESC [ Ps M              DL     Delete Ps lines (defaut = 1)
ESC [ Ps P              DCH    Delete Ps characters (defaut = 1)
ESC [ Ps X              ECH    Erase Ps characters  (defaut = 1) (VT220)
ESC [ Ps c              DA     Send device attributes
        PS = 0 or ommitted      Request attributes from terminal
                                Answer ESC [ ? 6 c: "I am a VT102")
ESC [ Ps ; Ps f         HVP    Move cursor to [row; column] (default = [1,1])
ESC [ Pm h              SM Set Mode
        Ps = 4          IRM  Set insert mode
ESC [ ? Pm h            DECSET DEC Private Set Mode Set
        PS = 7          DECAWM  Set autowrap on (default)
        PS = 25         DECTECM Cursor visible
ESC [ Pm l              RM Reset Mode
        Ps = 4          IRM  Set replace mode (default)
ESC [ ? Pm l            DECRST DEC Private Mode Reset
        PS = 7          DECAWM  Set autowrap off
        PS = 25         DECTECM Cursor not visible
ESC [ Pm m              SGR    Set character attributes
        Ps = 0          All attributes off
        Ps = 1          bold (increased intensity)
        Ps = 4          underline (simulated with color)
        Ps = 7          negative (reverse) image
        Ps = 10         Select font 0 (Debug/experimental)
        Ps = 11         Select font 1 (Debug/experimental)
        Ps = 22         bold off (normal intensity)
        Ps = 24         not underlined
        Ps = 27         positive image
        Ps = 30         Set foreground colorto black
        Ps = 31         Set foreground colorto red
        Ps = 32         Set foreground colorto green
        Ps = 33         Set foreground colorto yellow
        Ps = 34         Set foreground colorto blue
        Ps = 35         Set foreground colorto magenta
        Ps = 36         Set foreground colorto cyan
        Ps = 37         Set foreground colorto white
        Ps = 40         Set background colorto black
        Ps = 41         Set background colorto red
        Ps = 42         Set background colorto green
        Ps = 43         Set background colorto yellow
        Ps = 44         Set background colorto blue
        Ps = 45         Set background colorto magenta
        Ps = 46         Set background colorto cyan
        Ps = 47         Set background colorto white
ESC [ Ps n              DSR  Device Status report
        Ps = 5          DSR    Device status report: Answer is ESC [ 0 n (Terminal OK).
        Ps = 6          CPR    Cursor position report: Answer is ESC [ y ; x R
```

QUELLENANGABEN

[1] https://www.parallax.com/downloads/propeller-tool-software