Funktionsbeschreibung

VT100 + VT52 - Emulation

für MKT-Terminals mit CAN-BUS oder RS232-Schnittstelle

MKT Dokument Nr. 85141

Autor: Wolfgang Büscher, MKT Systemtechnik
Quelle: <WoBu>C:\CPROJ\vt100\DOKU\art85141_VT100_Emulation.odt

Index

1 Versions-Historie dieses Dokuments

Versionsnummer	Datum (ISO)	Autor	Bemerkungen, Änderungsgrund
V1.0	2001-06-15	W.Büscher	Ersterstellung mit Beschreibung der implementierten VT100- und VT52 – Sequenzen, Setup, Beispiele.
V1.1	2001-06-16	W.Büscher	Erweiterung der Beschreibung für Geräte OHNE Tastatur
V1.2	2005-04-20	W.Büscher	Hinweis "Vorläufig - under Construction" entfernt
V1.3	2014-03-05	W.Büscher	Hinweis zum VT100/VT52-Emulator in der Script-Sprache (für programmierbare Displays von MKT) ergänzt

2 Vorwort und Hinweise

Dieses Dokument enthält die Funktionsbeschreibung für *diverse* Bedienterminals von MKT mit VT100 / VT52 – Emulation.

Die Implementierung der VT100-Escape-Sequenzen entspricht in wesentlichen Teilen einem „echten“ VT100-Terminal der Firma Digital Equipment Corporation.

Zusätzlich sind einige VT52-Escape-Sequenzen implementiert, die vom ATARI ST Computer übernommen wurden (darunter Befehle zum Ein- und Ausschalten des blinkenden Cursors, was mit VT100 scheinbar nicht möglich ist).

Die Hardwarebeschreibung ist aus guten Gründen nicht Teil dieses Dokuments. Je nach verwendeter Hardware können nicht alle in diesem Dokument beschriebenen Funktionen realisiert werden, z.B. invertierte Darstellung bei 2*16-Zeichen-Textdisplays.

Je nach verwendeter Terminal-Hardware wird für den Datenaustausch zwischen Terminal und Host der CAN-Bus oder eine einfache RS232-Schnittstelle (ohne Hardware-Handshake) verwendet. Bei der Verbindung per CAN-Bus kann ein einfaches „Xon/Xoff“-ähnliches Protokoll verwendet werden, um Überlauf von Empfangspuffern zu verhindern (**kein CANopen-Protokoll !**).

Im Februar 2014 wurde für die 'programmierbaren' Anzeigegeräte von MKT (z.B. MKT-View II, III, HBG-18, HBG-22, HBG-35, etc) ein Beispielprogramm für die Script-Sprache entwickelt, in welchem ebenfalls die in diesem Dokument vorgestellten Escape-Sequenzen interpretiert werden können. Sie finden das Beispielprogramm nach der Installation des 'Programmiertools für CANdb-Terminals' (zu finden auf der MKT-CD) im Unterverzeichnis programs/script_demos/VT100Emu.cvt) . Eine kurze Beschreibung des Scripts, mit dem die VT100 / VT52-Funktionen emuliert werden, finden Sie hier .

-...-

.-.-.

3 Verwendeter Zeichensatz

Zur Zeit verwendet die Terminal-Emulation einen DOS-Zeichensatz mit 6*8-Pixel (oder mehr). Rechnen Sie damit, daß einige spezielle „VT100-Sonderzeichen“ nicht korrekt dargestellt werden können (z.B. Rautensymbol etc).

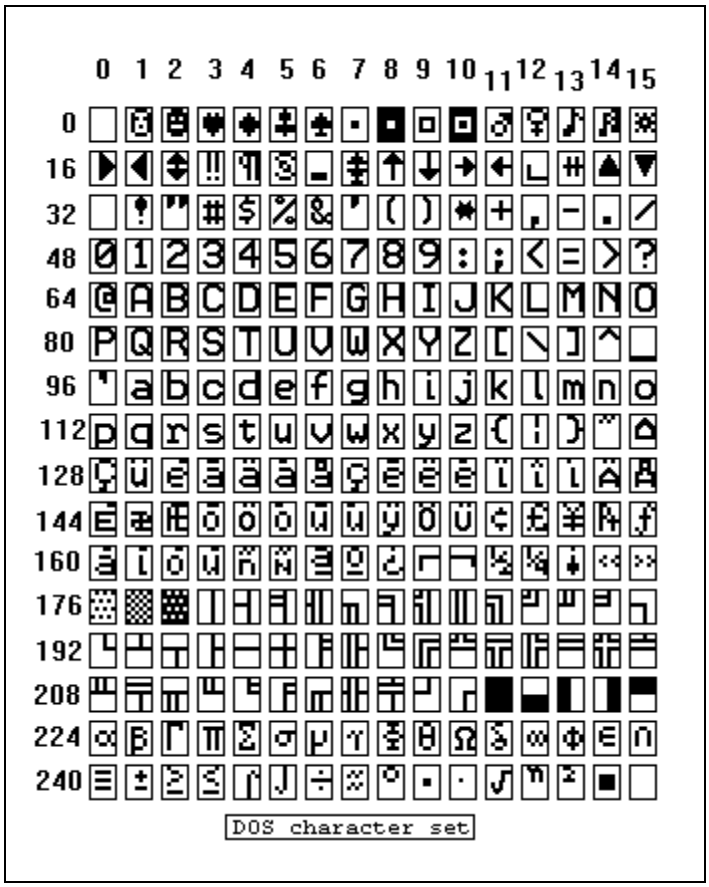

Abbildung eines für die Terminal-Emulation bei grafikfähigen Displays verwendeten Zeichensatzes.

Um den (dezimalen) Code eines Zeichens zu ermitteln, müssen Werte aus der Spalten- und Zeilennumerierung addiert werden.

Codes 176 bis 223 sind die früher oft verwendeten „DOS-Grafikzeichen“.

Simple Text-Displays (z.B. 2 Zeilen a 16 Zeichen) arbeiten mit einem fest im Display eingebauten Zeichensatz, der z.T. stark vom hier gezeigten abweicht ! Viele der Codes 128..255(dez) sind dort nicht nutzbar.

Wird das Terminal im „WINDOWS-ANSI-Modus“ betrieben, durchlaufen die von der Schnittstelle empfangenen Zeichen vor der Bildschirmausgabe noch eine Konvertierungsroutine. Dann sind allerdings die „DOS-Grafikzeichen“ nicht nutzbar. Zur Umschaltung dienen die in Kapitel 5.1 beschriebenen VT52-Sequenzen ESC F („Enter Graphics Mode“, hier: DOS-Zeichensatz) und ESC G („Exit Graphics Mode“, hier: WINDOWS-8-bit-ANSI-Zeichensatz). Die folgende Tabelle zeigt einige Unterschiede zwischen „DOS“- und „Windows“-Zeichensatz:

Sonderzeichen	Code unter „DOS“ (dezimal)	Code unter „Windows“ (dezimal)
ä	132	228
ö	148	246
ü	129	252
Ä	142	196
Ö	153	214
Ü	154	220
°	248	176
ß	225	223

4 VT100-Emulation

(Mai 2001)

Beim VT100-Terminal werden Steuersequenzen durch den sogenannten Control Sequence Introducer (SCI) eingeleitet. Der CSI für „normale“ VT100-Sequenzen besteht aus den Zeichen ESC [(Escape + eckige Klammer auf).
Danach folgen –je nach Befehl- dezimal codierte Parameter, und erst zum Schluß der Kennbuchstabe für den Befehl (ganz im Gegensatz zum einfacheren VT52 !).

Eine detailierte Beschreibung aller VT100-Steuersequenzen finden Sie im Internet bei

http://vt100.net/docs/vt100-ug

oder in kompakterer Form:

http://members.tripod.com/~ilkerf/cdoc/vt100ref.html

Beispieldateien zum Ansteuern einiger CAN-Terminals für MKT's „CAN-Tester“ sind bei MKT verfügbar:
<WoBu>\can\dos_can\VT100can.TXT .
Auszüge aus den Beispielen finden sich im Anhang (Kapitel 8).

4.1 VT100-Steuersequenzen

Die folgenden „normalen“ VT100-Steuersequenzen wurden bislang implementiert:

```
ESC[A          = Cursor Up
ESC[B          = Cursor Runter
ESC[C          = Cursor Rechts
ESC[D          = Cursor Links
ESC[<y>;<x>H = Absolute Positionierung des Cursors in Zeile<y>, Spalte<x>
ESC[<y>;<x>f = Absolute Positionierung des Cursors in Zeile<y>, Spalte<x>
                    (Zählung beginnt mit Zeile 1, Spalte 1 !)
ESC[H          = Absolute Positionierung des Cursors in 'home'-Position
ESC[0J         = Löscht den Bildschirm vom Cursor bis nach 'rechts unten'
ESC[1J         = Löscht den Bildschirm von 'links oben' bis zum Cursor
ESC[2J         = Löscht den ganzen Bildschirm (ohne Cursorbewegung)
ESC[2K         = Löscht die Zeile in der der Cursor steht (ohne Scrollen)
ESC[<n>m       = Setzen von Zeichenattributen
                    (z.B.: n=5:Blinken, 7=Invers, 0=Normal)
ESC[0q         = Schaltet alle programmierbaren LEDs aus
ESC[1q         = Schaltet die erste programmierbare LED ein (1=EINS)
ESC[2q         = Schaltet die zweite programmierbare LED ein
ESC[0;2q       = Schaltet erst alle LEDs aus und dann die zweite LED ein
ESC[?<n>h      = Aktivieren eines DEC-PRIVATE-Modes
                    (z.B.: n=7: wraparound am Zeilenende)
ESC[?<n>l      = Abschalten eines DEC-PRIVATE-Modes (l=Kleinbuchstabe L)
```

Die folgenden „DEC-eigenen“ Steuersequenzen („DEC private codes“):

```
ESC#3        = Double Line Height (doppelte Zeichenhöhe für eine Zeile)
ESC#6        = Double Line Width (doppelte Zeichenbreite für eine Zeile)
ESC#5        = Single Line Width, Single Line Height.
                  (schaltet den ESC#3 und/oder ESC#6-Modus wieder ab).
```

Beachten Sie bezüglich verdoppelter Zeilenhöhe und Zeichenbreite:

- Dies funktioniert nur bei Terminals mit grafikfähigen Displays, nicht bei „nur-Text-Displays“.
- Die Sequenzen „ESC#3“, „ESC#6“ und „ESC#5“ wirken sich auf die **gesamte** Zeile aus, in der der Cursor zum Zeitpunkt des Ausführens dieses Befehls steht. Die Cursorspalte spielt dabei keine Rolle.
- Bei der Verdopplung der Zeichenbreite in einer Zeile „verschwindet“ die halbe Zeile, weil sie nicht mehr auf das Display passt. Wird die Verdopplung rückgängig gemacht, erscheint sie wieder.
- Bei der Verdopplung der Höhe von Zeile N wird Zeile (N+1) unsichtbar.
- Im Gegensatz zum Original-VT100 ist hier die Sequenz „ESC#4“ (=Double Height Line Bottom Half) für die „untere Hälfte einer höhenverdoppelten Zeile“ nicht nötig, die Sequenz ESC#4 wird ignoriert.
- Im Gegensatz zum Original-VT100 ist hier auch die Kombination „doppelte Höhe aber einfache Breite“ möglich. Die Reihefolge, in der Breite und Höhe verdoppelt werden, ist egal.
- Durch „Clear Screen“ (z.B. ESC[2J) erhalten alle Zeilen wieder einfache Höhe und einfache Breite.
- Bei manchen Firmware-Variantem (mit knappem ROM) funktioniert die Kombination „doppelte Zeichenbreite aber einfache Höhe“ nicht.

Die folgenden ASCII-Steuerzeichen lösen besondere Funktionen aus:

```
chr(7)       = BELL .... Blinken der [ersten] LED für ca. 1 Sekunde(!)
chr(8)       = BACKSPACE
chr(10)      = NEW LINE
chr(13)      = CARRIAGE RETURN
```

Beim Betätigen der Tasten des Bedienfelds werden die folgenden Sequenzen gesendet (aber nicht "lokal" auf dem Display ausgeführt!):

```
<Cursor>     = i.A. VT100-ESCAPE-Sequenz, siehe Terminal-Setup
<F1>..<Fn>   = ESCAPE-Sequenz oder Buchstaben *
<ENTER>      = chr(13)   (ASCII- Carriage Return)
```

Da die bei fast allen MKT-Terminals vorhandenen Funktionstasten F1..Fn beim echten VT100-Terminal nicht existieren, gibt es für die Übertragung leider keine standardisierten Sequenzen. Beim Betätigen der Tasten des Bedienfelds werden die im Setup (Kapitel 6.5) definierten Sequenzen gesendet (aber nicht "lokal" auf dem Display ausgeführt !).

4.2 VT100-Grafik-Sonderzeichen

Die im VT100-Manual beschriebenen Steuercodes 152-156, 161, 164-170(oktal!!) zum Zeichnen von rechteckigen Rahmen (nach Umschaltung in den „special graphic character“-Modus) sind hier nicht verfügbar.

Stattdessen funktionieren bei allen grafikfähigen Displays die bekannten DOS-Linienzeichen mit Codes 179-218(dezimal). Siehe Kapitel 3.

Hinweis: Im VT100-Handbuch wird ein sogenannter „Grafikmodus“ erwähnt. Dabei handelt es sich allerdings nicht um das, was der Anwender eines „modernen“ PCs darunter verstehen würden! Auch wenn die in manchen MKT-Terminals verwendeten Displays prinzipiell grafikfähig sind, ist - und bleibt- ein VT100-Terminal aus Sicht des Anwenders ein reiner „Textbildschirm“.

* Da das VT100 nicht über Funktionstasten F1..F10 verfügt, sind hierfür verschiedene Möglichkeiten implementiert. Siehe Kapitel 6.5.3.

5 VT52-Emulation

(Mai 2001)

VT52 kann als Alternative zur VT100 verwendet werden. Das Original-VT100-Terminal von DEC unterstützt ebenfalls VT52. Viele Anwendungen die angeblich „VT100“-Sequenzen verwenden, arbeiten in Wirklichkeit im VT52-kompatiblen Modus. Beim „Original-VT100“ muss mit ESC [? 2 l in den VT52-Modus umgeschaltet werden, und mit ESC < vom VT52-Modus zurück in den VT100-Modus (der z.T. „ANSI“-Modus genannt wird). Beim hier beschriebenen Terminal von MKT sind diese Umschaltsequenzen unnötig, weil das Terminal grundsätzlich VT52 und VT100-Sequenzen gleichermaßen akzeptiert.

Die implementierte VT52-Terminal-Emulation umfaßt einige Erweiterungen, die einstmals im Atari ST (genialer Computer mit 68000er-CPU) zum Einsatz kamen. Die VT52-Sequenzen sind einfacher anzuwenden als das VT100-Terminal und reichen für die Fähigkeiten der MKT-Terminal-Hardware eigentlich aus. Darüberhinaus waren beim ATARI auch Steuerbefehle zum Einschalten und Ausschalten des Cursors definiert, die in der Original-VT100-Beschreibung nicht zu finden waren.

Ein Beispielprogramm zum Ansteuern einiger CAN-Terminals für MKT's „CAN-Tester“ ist bei MKT verfügbar:
<WoBu>\can\dos_can\VT52can.TXT

5.1 VT52-Steuersequenzen

Die VT52-Sequenzen beginnen mit ESCAPE (chr(27)), direkt gefolgt einem Kennbuchstaben für die Funktion, anschließend ein oder zwei Parameter jeweils als einzelne Buchstaben (keine mehrziffrigen Dezimalzahlen !).
Achtung ! Einige der hier beschriebenen Sequenzen sind VT52-Erweiterungen der Firma ATARI.
Die folgende Beschreibung basiert teilweise auf dem historischen Werk „ATARI ST Intern“ von Data Becker (1986).
Der Zwischenraum zwischen ESC und dem Kennbuchstaben in der folgenden Beschreibung dient nur der Übersichtlichkeit.

ESC A Cursor hoch
Damit wird der Cursor um eine Zeile nach oben bewegt. Befand sich der Cursor schon in der obersten Zeile, passiert nichts.

ESC B Cursor runter
Diese ESC-Sequenz positioniert den Cursor um eine Zeile nach unten. Befindet sich der Cursor schon in der untersten Zeile, passiert nichts.

ESC C Cursor rechts
Um eine Spalte nach rechts wird der Cursor durch diese Sequenz bewegt.

ESC D Cursor links
Damit wird der Cursor um eine Position nach links bewegt. Befindet sich der Cursor schon in der ersten Spalte, geschieht nichts (im Gegensatz zur Beschreibung von Data Becker ist diese Funktion nicht identisch mit dem Kontrollcode Backspace !).

ESC E Clear Home
Diese Sequenz löscht den kompletten Bildschirm und positioniert den Cursor in die linke obere Bildschirmecke (Home-Position).

ESC F Enter Graphics Mode
Schaltet den verwendeten Zeichensatz auf „Grafikzeichen“ um. Bei den MKT-Terminals sind dies die in Kapitel 3 beschriebenen DOS(!!)-Zeichen.

ESC G Exit Graphics Mode
Schaltet den verwendeten Zeichensatz zurück auf den Standard-ASCII-Zeichensatz (in dem wegen 7-Bit-Übertragung nur 128 Zeichen definiert sind). Bei den MKT-Terminals wird stattdessen nach ESC G soweit wie möglich der von WINDOWS verwendete „ANSI-Zeichensatz“(!!) verwendet, in dem alle 256 Codes definiert sind. Siehe auch Kapitel 3 (Zeichensatz).

ESC H Cursor Home
Setzt den Cursor in die Home-Position ohne den Bildschirm zu löschen

ESC I Cursor nach oben + evtl scrollen
Diese Sequenz bewegt den Cursor um eine Zeile nach oben. Im Gegensatz zu ESC A wird jedoch dann, wenn der Cursor sich schon in der obersten Zeile befand, eine Leerzeile eingefügt und der Rest des Bildschirms entsprechend um eine Zeile nach unten gescrollt. Die Spaltenposition des Cursors bleibt dabei unverändert.

ESC J Rest des Bildschirms löschen
Mit dieser Funktion wird der Rest des Bildschirms ab der augenblicklichen Cursorposition einschließlich gelöscht. Der Cursor bleibt dabei an derselben Stelle.

ESC K Rest der Zeile löschen
Diese ESC-Sequenz löscht den Rest der Zeile, in der sich der Cursor befindet. Gelöscht wird ab Cursorposition einschließlich; die Cursorposition selbst wird nicht verändert.

ESC L Zeile einfügen
Damit ist es möglich, an der augenblicklichen Cursorposition eine Leerzeile einzufügen. Der Rest des Bildschirms wird dabei nach unten geschoben; die unterste Zeile geht dabei verloren. Nach dem Einfügen steht der Cursor am Beginn der neuen Zeile.

ESC M Zeile löschen + scrollen
Diese Funktion löscht die Zeile, in der sich der Cursor befindet und rückt den Rest des Bildschirms nach oben. Die unterste Bildschirmzeile wird dadurch frei. Der Cursor befindet sich nach dem Löschen in der ersten Spalte der nachgerückten Zeile.

ESC Y Cursor positionieren
Dies ist eine der wichtigsten Funktionen. Sie erlaubt es, den Cursor beliebig auf dem Bildschirm zu positionieren. Dazu braucht die Funktion noch die Cursorzeile und –spalte als Parameter, die in dieser Reihenfolge mit einem Offset von 32 erwartet werden. Wollen Sie den Cursor in die zweite Zeile, dritte Spalte setzen, so müssen Sie die Sequenz ESC Y chr$(32+1) chr$(32+2) ausgeben. Zeilen und Spalten sind dabei von Null an gezählt, bei einem 4*16-Zeichen-Display sind die Zeilen von 0 bis 3 und die Spalten von 0 bis 15 numeriert. (Wenn Sie als „Basis“ den Offset 31 verwenden, beginnt die Zählung bei 1. Dies entspricht dann der VT52-Beschreibung in der Original-VT100(!)-Spezifikation.)

ESC Z Identify-Funktion (Original-VT52-Funktion)
Wird von VT52-kompatiblen Terminals mit der Sequenz „ESC / Z“ (hex 1B 2F 5A) beantwortet.

Die weiteren ESC-Sequenzen der „ATARI-ST-kompatiblen VT52-Emulation“ beginnen mit einem Kleinbuchstaben:

ESC b Schriftfarbe wählen (nur bei Farbdisplays!)
Mit dieser Funktion können Sie die Schriftfarbe für die weitere Ausgabe wählen. Je nach Farbtiefe werden nur die untersten 2,3 oder 4 Bits des Parameters ausgewertet. Die Farbpalette entspricht den Grundfarben einer CGA/VGA-Karte(!) im Textmodus:
0=schwarz, 1=blau, 2=grün, 3=rot, ... 7=weiß.

ESC c Hintergrundfarbe wählen (nur bei Farbdisplays!)
Mit dieser Funktion können Sie die Hintergrundfarbe für die weitere Ausgabe wählen.

ESC d Bildschirm bis Cursorposition löschen
(NOCH NICHT IMPLEMENTIERT)

ESC e Cursor einschalten
Durch diese ESCAPE-Sequenz wird der Cursor sichtbar, z.B. kann der Cursor eingeschaltet werden, wenn vom Benutzer eine Eingabe erwartet wird.

ESC f Cursor ausschalten
Hiermit wird der blinkende Cursor wieder abgeschaltet.

ESC j Cursorposition speichern
(NOCH NICHT IMPLEMENTIERT)

ESC k Cursor auf gespeicherte Position setzen
(NOCH NICHT IMPLEMENTIERT)

ESC l Zeile löschen ohne zu scrollen
Damit wird die Zeile gelöscht, in der sich der Cursor befindet. Die übrigen Zeilen bleiben davon unbeeinträchtgt. Nach dem Löschen befindet sich der Cursor in der ersten Spalte der gelöschten Zeile.

ESC o Zeilenanfang löschen
(NOCH NICHT IMPLEMENTIERT)

ESC p Reverse ein
Durch diese Sequenz wird auf invertierte Ausgabe geschaltet. Dabei werden für die nachfolgende Ausgabe Schrift- und Hintergrundfarbe vertauscht. Bei einem Monochromdisplay erhalten Sie weiße Schrift auf schwarzem Grund. Bei reinen Textdisplays (z.B. 2 Zeilen a 16 Zeichen) funktioniert dies nicht.

ESC q Reverse aus
Zum Zurückschalten auf Normaldarstellung dient diese Sequenz.

ESC v Automatischer Überlauf ein
Nach Ausführen dieser Sequenz wird bei einer versuchten Ausgabe über das Zeilenende hinaus automatisch eine neue Zeile begonnen.

ESC w Automatischer Überlauf aus
Damit wird die obige Sequenz (ESC v) wieder aufgehoben. Beim Versuch, über das Zeilenende zu schreiben, werden alle nachfolgenden Zeichen in die letzte Spalte der aktuellen Zeile geschrieben.

6 Terminal-Setup

(Juni 2001)

Je nach verwendeter Hardware (Interface, Display, Tastatur, ...) müssen im Terminal-Setup einige Parameter eingestellt werden, um das Terminal an Ihren Einsatzzweck anzupassen.
Die folgende Beschreibung ist eine „Summen-Beschreibung“ aller von MKT eingesetzten VT100-Terminal-Emulationen, das von Ihnen verwendete Gerät bietet evtl. nicht alle der hier aufgeführten Möglichkeiten.

Hinweis: Mittlerweile existieren auch reine „Anzeigen“ ohne Tastatur. Bei diesen Geräten gibt es kein Setupmenü, die Baudrate, CAN-Identifier und das verwendete Protokoll sind dann nicht einstellbar sondern „fest programmiert“ (z.B. beim Mini-CAN-Display mit 97*32 Pixeln). Teilen Sie uns in diesem Fall bei der Bestellung in diesem Fall **unbedingt** mit, welche Baudrate und ggf. CAN-Identifier Sie benötigen ! Andernfalls wird das Gerät mit den in diesem Kapitel erwähnten Defaultwerten ausgeliefert.

6.1 Aufruf des Terminal-Setups

Halten Sie beim Einschalten die erste und dritte Taste des Terminals gedrückt (Zählung beginnt „oben links“). Diese Tastenkombination wird abgefragt, wenn auf dem Display für ca. 0.5 Sekunden die Meldung „Setup ?“ angezeigt wird.
Wenn auf dem Display dann die Meldung „Release Key“ erscheint, lassen Sie alle Tasten los. Sie befinden sich nun im Setup-Menü, durch das Sie sich per Cursor up/down bewegen können. Um eine Funktion im Menü zu starten bzw. den dort gezeigten Parameter zu ändern, drücken Sie „ENTER“ (oder F4, falls keine Enter-Taste vorhanden ist).

Einige Tastenkombinationen für das Terminal-Setup (bei Power-On)

Gerät	Tastenkombination	Hinweise
„Low-Cost-Terminal“, Display: 2 Zeilen a 16 Zeichen, Firmware: ART11060 (RS232) ART11061 (CAN)	Cursor Links + Cursor Down	
„ABE 515“, Display: 128 * 64 Pixel, Firmware: ART11062 (CAN)	F1 + F3	Erst abwarten bis „Setup ?“ erscheint, sonst wird der CAN-Bootloader aktiviert !
„ET-515 Grafik“, Display: 128 * 64 Pixel, Firmware: ART11063 (CAN)	F1 + F3	

6.2 Verlassen des Terminal-Setups

Um geänderte Parameter dauerhaft zu speichern, wählen Sie die Funktion „SAVE+EXIT“ im Setup-Menü (per Cursor Up/Down die Markierung in dieses Menü setzen, dann ENTER (bzw. F4) drücken.
Um das Setup-Menü zu verlassen ohne die geänderten Parameter dauerhaft abzuspeichern wählen Sie die Funktion „EXIT“.

6.3 Editieren von Parametern im Setup-Menü

Um einen im Setup-Menü angezeigten Parameter zu ändern, wählen Sie zunächst den entsprechenden Menüeintrag an (per Cursor Up/Down, bei grafikfähigen Displays[1] ein schwarzer Markierungsbalken). Links wird ein Parameterkürzel angezeigt (z.B. „Contrast“), rechts davon der entsprechende Parameterwert (z.B. „127“). Beispiel (bei einem Grafikdisplay mit 128*64 Pixel) :

```
EXIT
SAVE+EXIT
Contrast=127
Wrap=TRUE
Ckeys=VT100
...
```

Per ENTER-Taste wird vom Navigations-Modus in den Parameter-Editier-Modus umgeschaltet. Der Markierungsbalken verschwindet, und es erscheint ein Cursor zum Editieren des Parameters. Bei Dezimalzahlen können Sie per Cursor Links/Rechts die zu ändernde Zehnerstelle anwählen. Die Ziffer unter dem Cursor ändern Sie per Cursor Up/Down (oder, falls Sie ein komfortableres Terminal verwenden, durch direkte Eingabe per Zifferntaste).
Um aus dem Editiermodus wieder in den Menü-Navigationsmodus zu gelangen, drücken Sie nochmals Return. Dann erscheint wieder der Markierungsbalken, und Sie können zum nächsten Parameter umschalten (evtl. durchscrollen).

6.4 Geräte-Parameter

6.4.1 Einstellen des LCD-Kontrasts (Setup: „Contrast“)

Mit dieser Setup-Funktion können Sie ggf. die Lesbarkeit mancher LCD-Anzeigen verbessern. Je höher der hier eingestellte Wert, desto stärker ist die Schwärzung des Displays. Im Idealfall sollte der Wert „127“ (Mitte) einen guten Kontrast ergeben. Werksseitig wird der Wert für übliche Innenraumtemperaturen eingestellt. Manche Displays erfordern bei sehr tiefen Temperaturen (unter 0°C) einen höheren Wert für die Kontrastspannung.

[1] Bei nicht grafikfähigen Displays ("Nur-Text", z.B. 2*16 Zeichen) kann die Anzeige nicht invertiert werden. In dem Fall wird statt des schwarzen Markierungsbalken auch im Navigationsmodus ein blinkender Cursor angezeigt.

6.5 VT100-Parameter

6.5.1 Automatischer Zeilenumbruch (Setup: „Wrap“)

Hiermit definieren Sie das Verhalten des Cursors beim Überschreiten des Zeilenendes.
Bei „Wrap=TRUE“ wird automatisch eine neue Zeile begonnen (und die Anzeige evtl. gescrollt).
Bei „Wrap=FALSE“ bleibt der Cursor am Zeilenende stehen, bei der nächsten Ausgabe wird das dort stehende Zeichen dann überschrieben.
Das Wrap-Verhalten kann auch per VT52- oder VT100-Sequenz beeinflußt werden, allerdings nur „temporär“ bis zum Abschalten der Stromversorgung (ESC[?7h wirkt sich nicht auf die im EEPROM gespeicherte Konfiguration aus).

Der Defaultwert ist TRUE (automatischer Zeilenumbruch aktiv).

6.5.2 Als Cursortasten gesendete Sequenzen (Setup: „Ckeys“)

Im Setup-Menü können Sie festlegen, welche Tastencodes (oder Escape-Sequenzen) das Terminal beim Betätigen der Cursortasten F1...Fn senden soll. Zur Auswahl stehen:

OFF : Beim Betätigen der Cursortasten sendet das Terminal gar nichts.
‚A‘..‘D‘: Cursor hoch,runter,rechts,links wird jeweils als einzelner Großbuchstabe (A,B,C,D) gesendet (1 Byte).
‚a‘..‘d‘: Cursor hoch,runter,rechts,links wird jeweils als einzelner Kleinbuchstabe (a,b,c,d) gesendet (1 Byte).
VT100: Die Cursortasten werden als entsprechende VT100-ESCAPE-Sequenz gesendet (3 Byte pro Taste, z.B. ESC [A = Cursor up).
VT52: Die Cursortasten werden als entsprechende VT52-ESCAPE-Sequenz gesendet (2 Byte pro Taste, z.B. ESC A = Cursor up).

Der Defaultwert ist „ESC [A ... ESC [D“ wie beim „echten“ VT100.

6.5.3 Als Funktionstasten gesendete Sequenzen (Setup: „Fkeys“)

Leider verfügt ein „Original-VT100“ nicht wie der PC (und viele MKT-Terminals) über Funktionstasten F1..Fn. (Die im VT100-Manual erwähnten „Function Keys“ sind Cursor- und Umschalttasten !).
Im Setup-Menü können Sie festlegen, welche Tastencodes (oder Escape-Sequenzen) das Terminal beim Betätigen einer der Funktionstasten F1...Fn senden soll. Zur Auswahl stehen:

OFF : Beim Betätigen von F1.. Fn sendet das Terminal gar nichts
‚0‘.. ‚9‘: Die Funktionstasten F1..F10 werden als ASCII-Codes 48..57 gesendet, dies sind die Ziffern ‚0‘... ‚9‘. (nur sinnvoll, wenn das Terminal keine speziellen Zifferntasten hat)
‚A‘..‘Z‘: Funktionstasten werden als Großbuchstaben ‚A‘.. ‚Z‘ gesendet. Bei Terminals mit nur 4 Funktionstasten können natürlich nur die Buchstaben ‚A‘.. ‚D‘ vorkommen.
‚a‘..‘z‘: Funktionstasten werden als Kleinbuchstaben ‚a‘.. ‚z‘ gesendet. Bei Terminals mit nur 4 Funktionstasten können natürlich nur die Buchstaben ‚a‘.. ‚d‘ vorkommen.
#59..#68: Funktionstasten F1..F10 werden als Codes #59..#68 gesendet. Dies sind die Scancodes der beim PC verwendeten MF2-Tastatur. Zeichencode #59 ist das ASCII-Äquivalent des Semikolons ! (die Übertragung erfolgt nicht als BCD-Sequenz, sondern als einzelner 8-Bit-Wert)
ESC[H..ESC[Q : Funktionstasten F1..F10 werden als VT100-ESCAPE-Sequenz gesendet. F1 wird als „ESCAPE [H“ übertragen, dies entspricht (zufällig?) dem Befehl „Cursor Home“.

Der Defaultwert ist „a ... z“, weil wohl nie ein Mini-Terminal mit „kompletter“ alphanumerischer Tastatur erhältlich sein wird. Die Kleinbuchstaben können daher gut als Codes für die Funktionstasten verwendet werden.

6.6 RS232-Parameter

Für die RS232-Schnittstelle ist das Xon/Xoff-Protokoll (wie beim Original-VT100-Terminal) vorgesehen, bislang aber nicht implementiert ! Die Übertragung von Texten und Steuersequenzen erfolgt immer „zeichenorientiert“.

VT100-Terminals mit RS485-Interface werden bislang nicht unterstüzt. Wegen des Problems der Umschaltung der Senderichtung ist dies ohne erheblichen Aufwand auch nicht möglich (erfordert ein zusätzliches Protokoll zur Kollisionsvermeidung, weil RS485 im Gegensatz zu RS232 nicht vollduplexfähig ist).

6.6.1 RS232 Baudrate (Setup: „SerBd“)

Wählen Sie aus dieser Combo-Liste die gewünschte „Baudrate“ für die serielle Schnittstelle aus. Bei den meisten Terminals mit seriellem Interface sind die folgenden Übertragungsgeschwindigkeiten verfügbar:
1200, 2400, 4800, 9600 und 19200 bit/sec .
Höhere Geschwindigkeiten sind bei VT100 nicht vorgesehen und können aus Hardwaregründen auch nicht realisiert werden.

Der Defaultwert ist 9600 bit/sec.

6.7 CAN-Parameter

6.7.1 CAN-Baudrate (Setup: „CanBd“)

Wählen Sie aus dieser Combo-Liste die gewünschte CAN-„Baudrate“ aus. Bei den meisten Terminals mit CAN-Interface sind die folgenden Übertragungsgeschwindigkeiten verfügbar:

10, 20, 50, 100, 125, 250, 500kBit/sec und 1MBit/sec.

Der Defaultwert ist 500 kbit/sec.

6.7.2 CAN-RX-Identifier (Setup: „CanRx1“)

Definiert den Identifier des aus der Sicht des Terminals „empfangenen“ CAN-Telegramms. Die Darstellung im Setup-Menü ist dezimal.
Mit diesem Identifier können Sie (als „HOST“) bis zu 8 Zeichen pro CAN-Telegramm an das Terminal senden.
Der Defaultwert des Identifiers ist 513 (dezimal).

6.7.3 CAN-RX-Telegrammstruktur (Setup: „CR1first“, „CR1last“)

Definiert den Aufbau des „empfangenen“ CAN-Telegramms (mit dem Identifier CanRx1). Hiermit können Sie im Bedarfsfall definieren, in welchen der maximal 8 Datenbytes eines CAN-Telegramms die vom Terminal auszuwertenden Zeichen stehen. Einstellbar sind:

CR1first: definiert, an welcher Stelle im CAN-Telegramm das **erste** auszuwertende Zeichen steht.
Defaultwert: 1 (=das erste Byte im Telegramm enthält das erste auszuwertende Zeichen)

CR1last: definiert, an welcher Stelle im CAN-Telegramm das **letzte** auszuwertende Zeichen steht.
Defaultwert: 8 (=das letzte Byte im Telegramm könnte noch ein auszuwertendes Zeichen enthalten)

Beispiel:

Der Host sendet zyklisch ein CAN-Telegramm mit jeweils 6 Datenbytes, von dem aber nur die letzten zwei Bytes innerhalb des Telegramms vom Terminal als „Zeichenkanal“ ausgewertet werden soll. Die passenden Einstellungen im Setup-Menü des Terminals sind dann:

CR1first = 5
CR1last = 6

In einem CANopen-Netz würde dies einem gemappten PDO entsprechen. Falls das Telegramm wegen anderer Busteilnehmer zyklisch gesendet werden muss obwohl keine Zeichen vorliegen, verwenden Sie Zeichencode 0x00 als „Dummy-Zeichen“ ohne Funktion.

6.7.4 CAN-TX-Identifier (Setup: „CanTx1“)

Definiert den Identifier des aus der Sicht des Terminals „gesendeten“ CAN-Telegramms. Die Darstellung im Setup-Menü ist dezimal.
Mit diesem Identifier kann das Terminal bis zu 8 Zeichen pro CAN-Telegramm an den HOST senden. Dazu gehören die Codes der am Terminal betätigten Tasten, je nach Protokoll zusätzliche Steuerzeichen wie XON und XOFF (siehe CAN-Übertragungsprotokoll).

Der Defaultwert ist 514 (dezimal).

6.7.5 CAN-TX-Telegrammstruktur (Setup: „CT1first“, „CT1last“)

Definiert den Aufbau des „gesendeten“ CAN-Telegramms (mit dem Identifier CanTx1). Hiermit können Sie im Bedarfsfall definieren, in welchen der maximal 8 Datenbytes eines CAN-Telegramms die vom Terminal gesendeten Zeichen (druckbare Zeichen + Steuerzeichen) stehen sollen. Einstellbar sind:

CT1first: definiert, an welcher Stelle im CAN-Telegramm das **erste** Zeichen stehen soll.
Defaultwert: 1 (=das erste Byte im Telegramm enthält das erste gesendete Zeichen)

CT1last: definiert, an welcher Stelle im CAN-Telegramm das **letzte** Zeichen stehen darf.
Defaultwert: 8 (=das letzte vom Terminal gesendete Zeichen *könnte* im letzten Byte eines CAN-Telegramms stehen)

Hinweis: Da fast alle Tastaturereignisse weniger als acht Bytes „Sendedaten“ erzeugen, wird das vom Terminal gesendete Telegramm oft kürzer sein als durch „CT1last“ definiert. Das Terminal wartet nicht, bis ein CAN-Telegramm „randvoll“ ist, sondern sendet sobald der Bus frei ist und wenigstens ein Datenbyte zum Senden ansteht (dies entspricht dem RS232-Handling des „Original-VT100-Terminals).

6.7.6 CAN-Übertragungsprotokolle (Setup: „CR1Pr“, „CT1Pr“)

Hiermit können Sie eins der für die CAN-Übertragung implementierten „Protokolle“ auswählen, getrennt für die Übertragung von Host zum Terminal (CR=CanReceive) und vom Terminal zum Host (CT=CanReceive). „Receive“ und „Transmit“ sind hier aus der Sicht des Terminals zu sehen !
Z.Z. verfügbar sind:

„None“: Keine Sicherung durch Übertragungsprotokoll
CAN-Telegramme werden unbestätigt gesendet. Das Senden darf nicht „zu schnell“ erfolgen, um einen Überlauf des CAN-Empfangspuffers im Empfänger zu vermeiden. Im einfachsten Fall sollte der Sender eine (noch zu spezifizierende) Wartezeit zwischen zwei gesendeten Telegrammen einlegen. GROBE SCHÄTZUNG für Terminals mit C515 und Textdisplay: ca. 20 Millisekunden. Bis zu 32 Telegramme kann das Terminal intern zwischenspeichern (wichtig, weil manche VT100-Operationen wie Scrollen oder Löschen des Bildschirms über 100 Millisekunden dauern können. Während dieser Zeit werden bis zu 32 CAN-Telegramme in einem Empfangspuffer abgelegt, die dann „später“ abgearbeitet werden).

„Frame ACK“: Frame Acknowledge
Das Terminal sendet für jedes empfangene Terminal ein „Bestätigungstelegramm“ (bestehend aus nur einem Byte mit dem ASCII-Steuerzeichen „XON“).

„Block ACK“: Block Acknowledge
Das Terminal sendet ein Antworttelegramm („XON“) wenn die letzte Ausgabe auf dem Bildschirm „komplett“ ist. Der Sender (HOST) kann bis zu 32 CAN-Telegramme mit minimaler Verzögerung senden, bis er einmal auf das „Block Achknowledge“-Telegramm warten ***sollte***.
Nach Befehlen wie Scrollen oder Löschen des Bildschirms können etliche Millisekunden bis zum Block-Acknowledge vergehen (z.B. 250ms beim „ABE 515 V1.2“ mit 128*64-Pixel-Display).

„XON/XOFF“: Puffersteuerung mit XON- und XOFF-Zeichen (wie bei RS232-Interface)
Das Terminal sendet ein XOFF-Zeichen, wenn sein Empfangspuffer mehr als 50 Prozent gefüllt ist, und ein XON-Zeichen, wenn der Puffer die 25%-Füllmarke wieder unterschreitet.
Als XON-Zeichen dient der ASCII-Steuercode „DC1“ (=chr$(17)), als XOFF-Zeichen dient „DC3“ (=chr$(19)). XON und XOFF werden in den vom Terminal gesendeten Zeichenstrom eingefügt und können von der Transportschicht im Host wie bei RS232-Betrieb „ausgefiltert“ werden .

Der Defaultwert ist „NONE“ (für „CR1Pr“ **und** „CT1Pr“), d.h. es werden weder Quittierungstelegramme gesendet noch erwartet. Wenn das Terminal trotzdem Quittierungstelegramme empfängt, werden diese ignoriert (kein Fehler!).

7 VT100 / VT52 - Emulation per Script

Bei Geräten in denen die VT100 / VT52-Emulation per Script realisiert werden soll (z.B. MKT-View II / III, HBG-18, HBG-22, HBG-35, etc) finden Sie Details zum Setup-Menü in Dokument Nr. 85115 („Systemmenü und Setup-Optionen für ***programmierbare*** Terminals“). Zusammen mit zahlreichen weiteren Handbüchern finden Sie diese Datei auf der mittlerweile 'online' gestellten MKT-CD .

Prinzipiell sind damit alle Geräte, die MKT's Script-Sprache unterstützen, auch als VT100- oder VT52-Terminal verwendbar (je nach Hardware per CAN oder/und RS-232).

8 Anhang: Beispiele

Auszug aus Testprogrammen für MKT's CAN-Tester. Der CAN-Tester dient hier zum Senden verschiedener CAN-Telegramme mit folgendem Aufbau :
<IDENTIFIER> <Datenbytes>
Ein vorangestelltes Doppelkreuz kennzeichnet Dezimalzahlen.
Das „at"-Zeichen (bzw. „Klammeraffe", @) kennzeichnet Steuerkommandos für den CAN-Tester oder Variablenzuweisungen. Text in Anführungszeichen kann Byte für Byte in ein CAN-Telegramm eingetragen werden.
Kommentartext innerhalb einer Zeile wird durch das Semikolon abgetrennt.

8.1 Ansteuerung eines Terminals im VT52-Modus per CAN

```
;--- Testprogramm fuer "VT52-Terminals mit CAN-Interface" ---------
; Texte werden hier per CAN-ID #513 an das Terminal geschickt.
; Dabei werden einfache VT52-Sequenzen (ESC<x>) ausgewertet.
; Das Terminal schickt Tastensequenzen und ggf XON,XOFF auf CAN-ID #514.
; Sendepause auf 50ms setzen, sollte reichen... auch ohne XON/XOFF-Steuerung
; Baudrate: meistens 500kBit/sec.
Start:

#513 #27 "Z"    ; ESC Z = "Was Bist Du ?" (Identify-Befehl)
                ; Antwort 1B 2F 5A = ESC / Z -> "ich bin ein VT52-Terminal"

; Intro & Info
#513 #27 "E"      ; ESC E  = Erase Screen
#513 #27 "v"      ; automatic line wrap ON
#513 "DEMOPROG"
#513 "RAM "
#513 "for VT52"   ; send normal (printable) characters
#513 "-Termina"
#513 "ls via C"
#513 "AN ..   "
@delay(500)
#513 #13 #10
#513 "CrsrOn :"
#513 #27 "e"        ; from ATARI ST:   Blinking cursor ON
@delay(1500)
#513 #13
#513 "CrsrOff:"
#513 #27 "f"        ; from ATARI ST:   Blinking cursor OFF
@delay(1000)
#513 #13 #27 "K"  ; CR + clear end of line
#513 #27 "e"      ; turn cursor on again
#513 #27 "w"      ; automatic line wrap OFF
@delay(2000)

; Per VT52 nutzbare Zeichenattribute...
#513 #27 "E"    ; clear screen,  cursor home
#513 "Attribut"
#513 "es: "
#513 #13 #10       ; CR + NL
#513 #27 "pInvers" ; attribute "Revers" (graphic displays only)
#513 #27 "qNormal" ; attribute "Revers" off
@delay(500)
#513 #27 "H"       ; cursor home (no erasing)
; Einfuegen und Loeschen einzelner Zeilen...
@delay(500)
#513 #27 "I"       ; move cursor up, scroll and insert new line if on top
#513 "ESC-I"
@delay(500)
#513 #27 "B"       ; move cursor down, do NOT scroll
```

```
#513 #27 "L"       ; insert a new empty line at the cursor position
#513 "Hello !"
@delay(500)
#513 #27 "M"       ; delete line of the cursor, scroll rest up
@delay(500)
#513 "deleted."
@delay(2000)

; special chars  / Sonderzeichen..
#513 #27 "F"        ; "Enter Graphics Mode", here: use 8-bit "DOS" charset
#513 #13 #10 "DOS-Ch"
#513 "arset:" #13 #10
#513 " "  #132 #148 #129     ; ae oe ue (im DOS-Zeichensatz)
#513 " "  #142 #153 #154     ; Ae Oe Ue (im DOS-Zeichensatz)
#513 " "  #225 #248 #039     ; sz grad accent
@delay(1000)
#513 #27 "G" ; "Exit Graphics Mode", here: use WINDOZE 8-bit "ANSI" charset
#513 #13 #10 "WIN-Ch"
#513 "arset:" #13 #10
#513 " "  #228 #246 #252     ; ae oe ue (im WINDOZE-"ANSI"-Zeichensatz)
#513 " "  #196 #214 #220     ; Ae Oe Ue (im WINDOZE-"ANSI"-Zeichensatz)
#513 " "  #223 #176 #146     ; sz grad accent
@delay(2000)

; Scrolling speed & RX Buffer test
#513 #27 "E"        ; Clear Screen + Cursor home
@N=1
@repeat             ; Loop to send 99 lines of text to the terminal
 #513 #13 #10 "LineNr"
 #513 (#48+(N/#10)) (#48+(N%#10))
 @N=N+1
 @delay(50)         ; wait 50 milliseconds for scroll
@until(N>99)
@delay(5000)

@goto Start

; EOF
```

8.2 Ansteuerung eines Terminals im VT100-Modus per CAN

```
;--- Testprogramm fuer "VT100-Terminals mit CAN-Interface" ---------
; Texte werden per CAN-ID #513 an das Terminal geschickt.
; Dabei werden einige VT100-Sequenzen (ESC[<x>) ausgewertet.
; Das Terminal schickt Tastensequenzen und ggf XON,XOFF auf CAN-ID #514.
; Sendepause auf 50ms setzen, sollte reichen... auch ohne XON/XOFF-Steuerung
; Baudrate: meistens 500kBit/sec.

 ; Einige "Sonder-Sequenzen" :
#513 #27 "[6n" ; request cursor position report -> ESC [<line>;<column>R
#513 #27 "[c"  ; request report "What are you"? -> ESC [1;0c (basic VT100)

Start:

#513 #27 "[2J"    ; clear entire screen a la VT100. CURSOR DOES NOT MOVE.
#513 #27 "[H"     ; direct cursor addressing: move to HOME position
#513 #27 "[?7h"   ; automatic line wrap ON (a VT100 "DEC private mode")
#513 "Attribut"
#513 "es: "
#513 #13 #10       ; CR + NL
#513 #27 "[7m"     ; ANSI char attribute "Revers"
#513 "Reverse"
#513 #27 "[0m"     ; ANSI char attribute "Normal"
#513 " Normal"
@delay(800)
#513 #27 "[0;5m"   ; ANSI char attribute "only Blinking"
#513 #13 #10 "Blink "
#513 #27 "[0;7;5m" ; ANSI char attributes only "Reverse"+"Blinking"
#513 "InvBlink"
#513 #27 "[0m"     ; ANSI char attribute "Normal"

@delay(200)
#513 #13 #10
#513 #27 "#3" #27 "#6"  ; double height, double width for the CURRENT LINE
#513 "LEDs.."
#513 #27 "[0;1q"   ; Turn all LEDs off, then LED 1 on
@delay(300)
#513 #27 "[2q"     ; Turn LED 2 on
@delay(300)
#513 #27 "[3q"     ; Turn LED 3 on (if exists..)
@delay(300)
#513 #27 "[4q"     ; Turn LED 4 on (if exists..)
@delay(300)
#513 #27 "[0q"     ; Turn all LEDs off
@delay(300)

#513 #27 "[2J" #27 "[H" ; clear entire screen + move HOME  a la VT100
#513 #27 "[?7h"    ; automatic line wrap ON (a VT100 "DEC private mode")
#513 "DEMOPROG"
#513 "RAM "
#513 "for VT10"
#513 "0-Termin"
#513 "als via "
#513 "CAN ..  "
#513 #27 "[?7l"    ; automatic line wrap OFF (VT100 "DEC private mode")
@delay(1000)

; Double  Width   or    Double Height  (?)
#513 #27 "[2J" #27 "[H" ; clear entire screen + move HOME  a la VT100
#513 "DEC pri"
#513 "vate co"
```

```
#513 "ntrols"
#513 #27 "[2;1H"   ; move cursor to line 2 (=SECOND), column 1 (=left margin)
#513 "Double "
#513 "Height!"
#513 #13 #10
#513 "Invisib"
#513 "le soon"
#513 #27 "[2;1H"   ; move cursor to line 2 , column 1 again
@delay(1000)
#513 #27 "#3"      ; DEC private: Double Height for top half of line
@delay(1000)
#513 #27 "#5"      ; DEC private: Single Height + Single Width LINE
@delay(1000)
#513 #27 "#3"      ; DEC private: Double Height for top half of line
@delay(1000)
#513 #27 "#5"      ; DEC private: Single Height + Single Width LINE
#513 #27 "[3;3H"   ; move cursor somewhere else
; Double Width Demo
#513 #27 "[2;1H"   ; move cursor to line 2 (=SECOND), column 1 (=left margin)
#513 "Double "
#513 "Width !"
#513 "01234567"
#513 #27 "[2;1H"   ; move cursor to line 2 , column 1 again
@delay(1000)
#513 #27 "#6"      ; DEC private: Double Width for current line
@delay(1000)
#513 #27 "#5"      ; DEC private: Single Width, Single Height LINE
@delay(1000)
; Double Width + Double Height Demo
#513 #27 "[2;1H"   ; move cursor to line 2 (=SECOND), column 1 (=left margin)
#513 "Double "
#513 "Width+"
#513 "Height !"
#513 #27 "[2;16H"   ; move cursor to line 2 , column 16 (!)
@delay(1000)        ;   (Note: Cursor will disappear !!!!!)
#513 #27 "#6"      ; DEC private: Double Width for current line
#513 #27 "#3"      ; DEC private: Double Height for top half of line
@delay(1500)
#513 #27 "#5"      ; DEC private: Single Width, Single Height LINE
                   ;    (Note: Formerly invisble cursor APPEARS on old pos)
@delay(1000)
#513 #27 "#6"      ; DEC private: Double Width for current line
#513 #27 "#3"      ; DEC private: Double Height for top half of line
@delay(2000)
#513 #27 "#5"      ; DEC private: Single Width, Single Height LINE

; DOS graphic chars (not really VT100-compatible to use this feature!)
#513 #27 "[2J" #27 "[H" ; clear entire screen + move HOME  a la VT100
#513 #27 "[?7l"         ; automatic line wrap OFF ("DEC private mode")
#513 #27 "#3" #27 "#6"  ; double height, double width for the CURRENT LINE
#513 "DOS-"
#513 "Frames"
@delay(200)
#513 #13 #10 #10               ; CR + 2*NL to jump into a visible line
; single framed box...
#513 DA C4 C4 C4 C4 C4 C4 BF   ; upper line
#513 #13 #10                   ; CR+NL
#513 B3 "Frame1" B3            ; Center
#513 #13 #10                   ; CR+NL
#513 C0 C4 C4 C4 C4 C4 C4 D9   ; lower line
@delay(2000)

@goto Start
```

```
; EOF
```