CONTENTS

This publication may describe designs for which patents are granted or pending. By publishing this information, Tandberg Data A/S conveys no license under any patent or any other rights.

Every effort has been made to avoid errors in text and diagrams. However, Tandberg Data A/S assumes no responsibility for any errors which may appear in this publication.

It is the policy of Tandberg Data A/S to improve products as new techniques and components become available. Tandberg Data A/S therefore reserves the right to change specifications at any time.

We appreciate any comments on this publication.

CONTENTS

1. INTRODUCTION

The graphics option consists physically of one add-on board to be connected to the TDV 2200 S terminal. It has its own micro processor based graphics system solution (see figure 1).
Any communication between host and the graphics subsystem goes through the terminal communication handler, whereas the communication between host and the terminal is as normal. The host can set up a soft-switch to route all information directly between the graphics subsystem and host via the terminal's communication handler. One ASCII control code switches on (US) and one switches off (CAN).

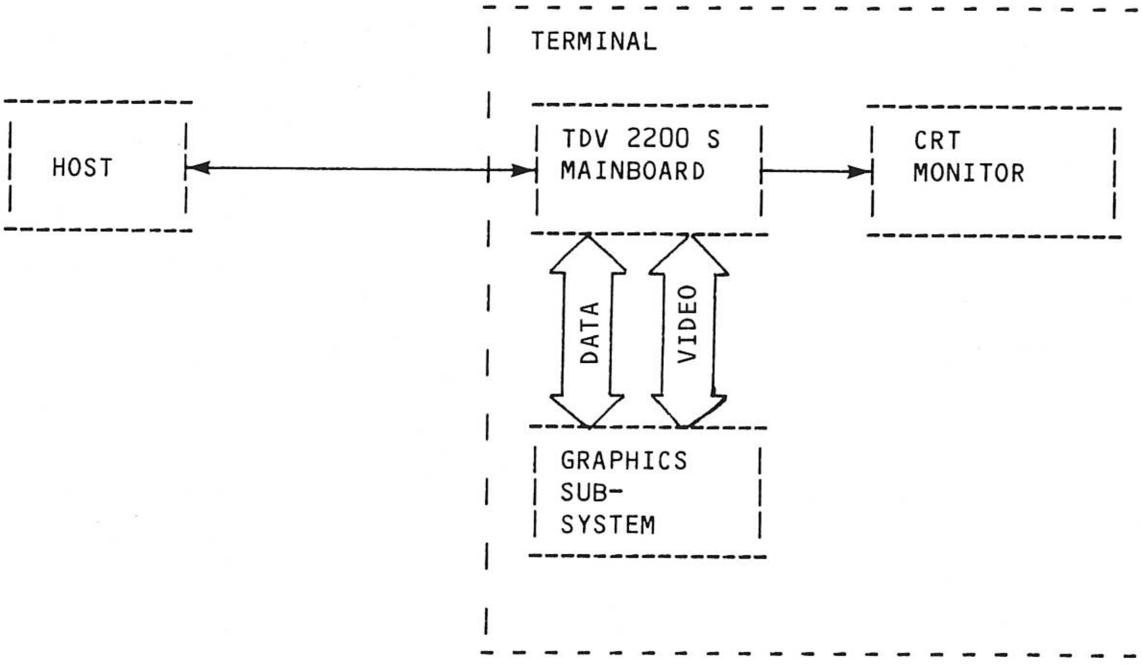
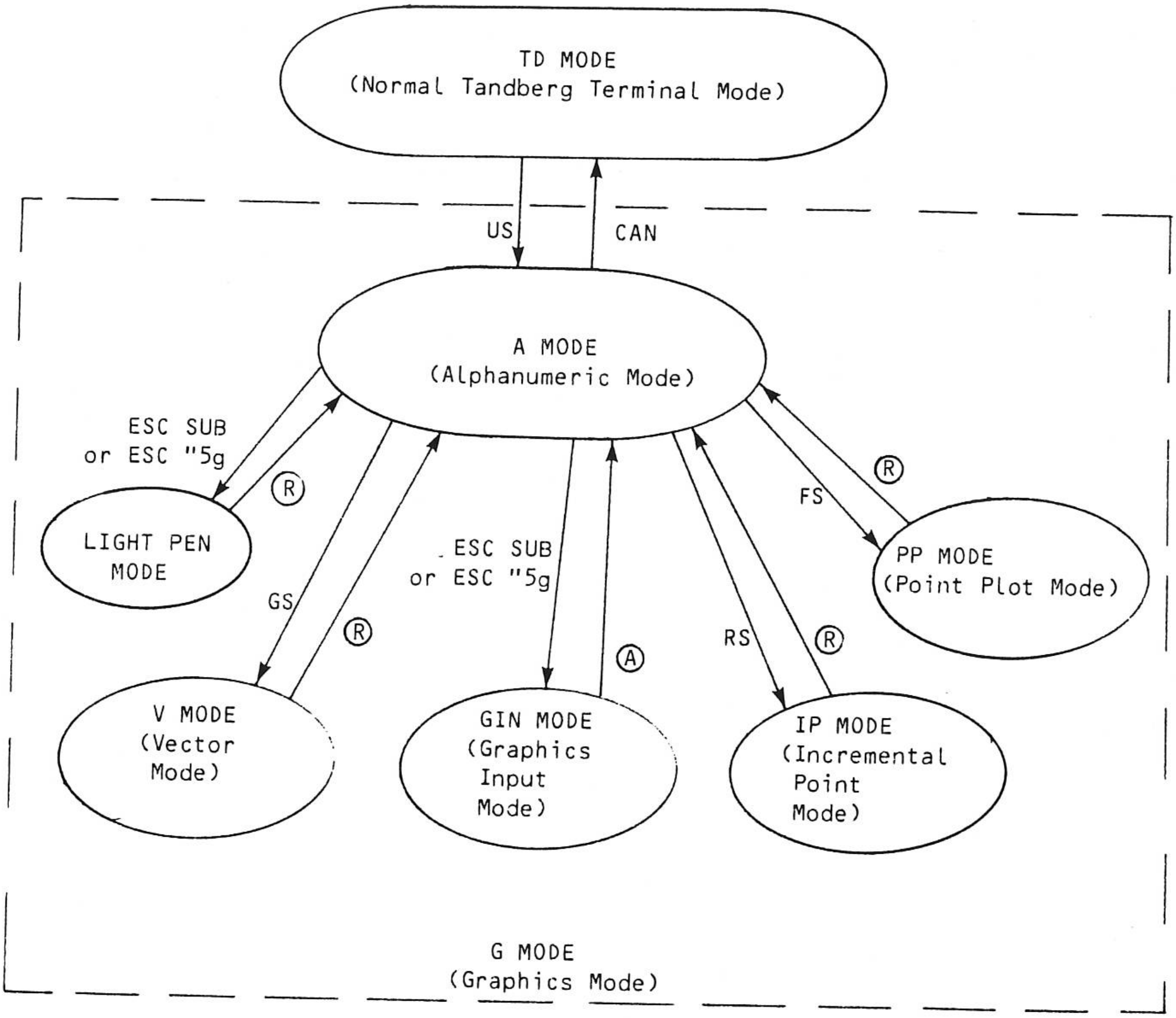


Figure 1 Graphic Option on TDV 2200 S

The two control codes, US and CAN, are the same as specified in the TEKTRONIX 4010/4014 formats. The graphics option is also compatible with these two formats and will directly run a Tektronix PLOT 10 software package and many others using the same format. This can be handled without any software modification, which means that we are able to handle TDMODE (a transparent mode) and GMODE (graphics mode) with AMODE (alphanumeric mode), VMODE (vector mode), PPMODE (Point Plot mode), IPMODE (incremental point mode) and GIN MODE (graphic input mode).
To make it clear, TDMODE is the normal terminal mode without a graphics board installed. With a graphics board installed you can switch between TDMODE and GMODE, which gives your terminal a new dimension for solving a wider application area, all in one terminal.

The state diagram of the different modes is illustrated below.



NOTE: Ⓡ is either CR, ESC FF or ESC ENQ
Ⓐ is any alphanumeric character

Figure 2 Graphic Modes State Diagram

The terminal gives you the possibility of choosing between video plane in TDMODE, GMODE or both planes at the same time; a command sequence from host makes that choice. The graphics subsystem operates by receiving a sequence of characters indirectly from host or keyboard. If the terminal is set up with internal echo and if it is online, each keystroke will be routed to both host and the graphics subsystem. If the terminal is set up with external echo, it is up to the host program to give echoes back on the line or not, to control the keyboard actions.

To make a better cost performance of the host resources, many enhanced functions are added to the TEKTRONIX 4010/4014 format. By using these enhancements, the graphics software programmer can remove time-consuming routines in the host; the terminal does the work a lot faster than a TEKTRONIX 4010/4014 written program.
This means that you get more power out of your host with increased speed and better interactive response possibilities, and with these enhancements the terminal is better adjusted to more advanced graphics standards (like the international GKS standard).

In addition to PLOT 10 software, the terminal is compatible with Runit's GPGS, ISSCO's DISSPLA and TELL-A-GRAF, Signal Technology's Interactive Laboratory System and NOVA Graphics International's Nova-GKS to mention some.
If the graphics software has a driver for Retro-Graphics terminals, it will normally be compatible with TDV 2200 S with the graphics option.

The graphics subsystem makes use of a Z-80A microprocessor and performs functions like drawing lines and picture elements in a "bit map" memory. The graphics image shows a raster scan plot of 336 by 720 from the "bit map", which gives the user a good resolution for business graphic, graphics interface in office automation and the lower end of CAD/CAM applications.

2. OPERATING MODES - DEFINITIONS

Before we start describing the graphics functions in the chapters that follow, let's start with some definitions which may be necessary for understanding the function description.

2.1 Parsing ANSII Style ESCAPE sequence

```
ESC <lead-in> [;<decimal number>]...<trailer>

where

<lead-in>            := '"' or '[' or '/'
<decimal number>     := signed number in the range  -32768 to 32767,
                        unsigned number in the range 0 to  65535
<trailer>            := Any defined ASCII code
                        (default  :=CR)
```

A minimum of 10 parameters are parsed. If a parameter is outside the range, it is discarded. Missing parameters have a value of zero. Next to the leading character, the only valid characters are the numbers 0 - 9. The trailer code(s) terminate the function sequence that the graphics board sends to host; all other characters are discarded.

2.2 6 Bits Data Type

Several of the function sequences make use of 6 bits data type. In this case, the single character which is received will be interpreted as a 6 bits signed or unsigned value. This means that in a 7 bits code, bit 7 is on.
According to the interpretation in an ASCII code it will range from '@' to 'DEL'. The data value carried in this format is given by:

```
      <data> = ASCII char - @
```

where <data> will range from 0 to 3F hex

Now, because 'DEL' can not be used, we use '?' instead as an interpretation of 'DEL' value in a 6 bits data type.
If the signed value is required, it must be 2's complement. Any other character will be discarded.

2.3 Bypass Condition

Whenever data is transmitted from the graphics board to host, the bypass condition is entered, which means that all input of characters to the graphics board will be ignored until a control character (ASCII code 0 to 1F hex) is received. The control code will be handled with action, if defined as in a command code sequence from host.
This feature allows the graphics board to ignore data which is echoed back from the host. This can be in a memory readback transfer, from an INQUIRY response or from a GIN MODE response to host.

3. TEKTRONIX 4010/4014 COMPATIBLE ENTRY-CODES

3.1 Entry to GMODE (graphics mode) and AMODE (alpha mode)

Command code sequence to the graphics from host:

```
                          US    <1F hex>
                ( or      ESC US   <1B,1F hex> )
```

ESC US is only for graphics software which is not completely TEKTRONIX compatible. The US code is used to switch from TDMODE to AMODE, which is the initial part of the GMODE. For this type of switching, ESC US can not be used.
Subsequently received characters are interpreted as ASCII characters to be plotted with the current writing mode (dot off/dot on/complement dot) and the defined character size and font (character set).
The ASCII control characters (CR, LF, etc.) are acted upon; these will be described later in this section.

The cursor is a blinking underline version and indicates the next writing location. The displacement for this position is normally dependent on the character size, but from the host you are able to position the writing location wherever you want.

The AMODE incorporates the MARGIN 1 function for functional compatibility with PLOT 10 software. MARGIN 1 is automatically set when the last available line of text is reached and the terminal receives a LF on the last printable line on the screen. A second column of text can then be displayed on the right-hand side of the screen.

The AMODE can be used for normal alphanumeric operations or in conjunction with one of the other graphics modes. You can make graphs from the host or directly by an input pointing device. Furthermore, you can load a picture into a specified position from host, which gives you the possibility of integrating text and illustrations. The screen image can be printed out by the terminal printer or by read back to host in a compressed facsimile format. A lot of these possibilities are enhancements, which are described in chapters 5 to 9, Enhanced Functions.

To switch into this mode directly from the keyboard, the terminal must be local, or keyboard codes passed to the line must be echoed back by external or internal echo. The US code can be generated from the keyboard by CTRL _(underline), <1F>. This control character may be located on another key on your terminal version; check this out in the functional specifications if the code doesn't work.

A6
A5

3.2 Entry to TDMODE (ordinary TANDBERG terminal mode)

Command code sequence to the graphics from host:

CAN <18 hex>

This control code switches the terminal from GMODE back to TDMODE. Your terminal will then act just as without graphics option, except that you still can see what you have created on the terminal in GMODE. By sending certain codes to the graphics board you can switch off the graphics video or clear the complete graphics memory or only parts of it. This gives you a choice to work in both planes, dependent on how the application programs in the host are designed.

To switch into this mode directly from keyboard, the terminal must be in local mode or the keyboard codes passed to the line have to be echoed back. The CAN control code can be generated from the keyboard by CTRL x, <18>, and can be interpreted as EXIT from GMODE.

4. TEKTRONIX 4010/4014 COMPATIBLE FUNCTION-CODES

4.1 Space Left

Command code sequence to the graphics from host:

```
                    BS     <08 hex>
            or      ESC BS
```

A backspace function is performed. The current location is moved one character position to the left. The cell size of the last character plotted will define the backspace position step.

4.2 Space Right

Command code sequence to the graphics from host:

```
                    HT     <09 hex>
            or      ESC HT
```

A space forward is performed, the current X value is incremented by the current X cell size. If this results in a current X value greater than or equal to Xres (right margin position), the current X value is set to 0 and the current Y value is decremented by the cell size of the last character size definition. This acts in the same way as Carriage Return Line Feed (CRLF) inserted at the end of the line.
Note that (X,Y) = (0,0) is bottom/left corner on the screen.

4.3 Space Down

Command code sequence to the graphics from host:

```
                    LF     <0A hex>
            or      ESC LF
```

A line feed function is performed. The Y value of the current location is decremented by the Y cell size of the last defined character, which means that cursor will be moved one line down. If decremented below bottom Y coordinate value, (Y=0), the Y value is set to a home Y value, independent on character size. This home value is at the first line (smallest character size) on the top of the screen, and the margin X value is toggled as in TEKTRONIX PLOT 10 format for two column text.

4.8 Entry to Vector Mode

Command code sequence to the graphics from host:

```
          GS     <1D hex>
     or   ESC GS
```

This control code sets the terminal in vector drawing mode. The first coordinate after GS command will only initiate the current start position. A second coordinate transmitted to the graphics board will draw a vector to connect these two points with the current line type. If there is a need to break the connection between two coordinate points, this can be done by preceding the next coordinate point with a GS control code. The vector format is illustrated in figure 3 on the next page.

The character 'P' in bit position 7 in this format indicates that the vector format does not use this bit, but 'P' can be used as parity bit or just be ignored. The code in the format uses seven bits, and five of these will be coordinate information. A two-bytes code will therefore give us a 10 bits coordinate value for x or y, and we will be able to address 1024 points/pixels in the x direction and 780 in the y direction, which is expected by the Tektronix format. This coordinate value will automatically be scaled down to the terminal's 720 by 336 bitmap format. You will also have the possibility of addressing directly to the pixels position in the bitmap, but this will be discussed in the 'Enhanced Graphics' section.

If we try to go in more detail about the vector format we will find that the coordinate for high x and y will be similar to an ASCII code between 'space' and '?' (20-3F hex). Low x will in a similar way be associated with the ASCII code between '@' and '-' (40 - 5F hex) and low y with the ASCII code between ''' and 'DEL' (60 - 7F hex). By this code structure in the vector format it is easy for a program to generate or analyze the contents, but a bit more complicated if we try to do it manually.

BIT

	7	6	5	4	3	2	1	0		
GS	P	0	0	1	1	1	0	1	START VECTOR POSITION	
HIGH Y	P	0	1	Y9	Y8	Y7	Y6	Y5	Y COORDINATE	FIRST COORDINATE
LOW Y	P	1	1	Y4	Y3	Y2	Y1	Y0		
HIGH X	P	0	1	X9	X8	X7	X6	X5	X COORDINATE	
LOW X	P	1	0	X4	X3	X2	X1	X0		
HIGH Y										SECOND COORDINATE
LOW Y										
HIGH X										
LOW X										
o o o o										
US	P	0	0	1	1	1	1	1	END VECTOR DRAWING	

After transmission of the first four bytes, subsequent bytes that do not change (except for the LOW X byte) need not be sent. If the HIGH X byte changes, the LOW Y byte must be sent as well.

NOTE: Lines will be drawn between coordinate N-1 and N. Stop drawing for repositioning can be handled by inserting a GS-code between two coordinates.

Figure 3 Vector Drawing Sequence

4.9 Status Readback

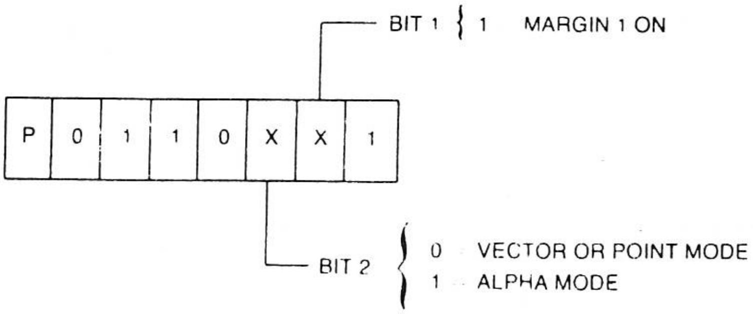
Command code sequence to the graphics from host:

ESC ENQ (1B,05 hex)

A report message is sent to host, containing status and the current cursor location. The bypass condition is entered.
A response to this command function is shown in figure 4 below.

		BIT 7	6	5	4	3	2	1	0
	* GRAPHICS STATUS	P	0	1	1	0	X	X	1
APLPHA CURSOR POSITION	HIGH X	P	0	1	X9	X8	X7	X6	X5
	LOW X	P	0	1	X4	X3	X2	X1	X0
	HIGH Y	P	0	1	Y9	Y8	Y7	Y6	Y5
	LOW Y	P	0	1	Y4	Y3	Y2	Y1	Y0
PROGRAMMABLE	TRAILER (e.g. CR)	P	X	X	X	X	X	X	X
	TRAILER (e.g. EOT)	P	X	X	X	X	X	X	X

* GRAPHICS STATUS



If host sends an inquiry to the graphics by: ESC ENQ, this will be the response to host.

Figure 4 Response to ESC ENQ

4.10 Entry to PPMODE, Point Plot Mode

Command code sequence to the graphics from host:

```
          FS      (1C hex)
    or    ESC FS
```

This function is similar to vector mode. In this mode you are allowed to mark dots at each coordinate point specified in the vector format.

4.11 Entry to IPMODE, Incremental Point Mode

Command code sequence to the graphics from host:

```
          RS      (1E hex)
    or    ESC RS
```

Subsequently received characters in IPMODE are interpreted as "point plotter" type commands. Points are plotted relatively to the current cursor location in one of eight directions. The commands are given in the following table:

CODE	HEX VALUE	FUNCTION GENERATED
space	20	Pen up
P	50	Pen down
D	44	N (up or north)
E	45	NE (up & right or north east)
A	41	E (right or east)
I	49	SE (down & right or south east)
H	48	S (down or south)
J	4A	SW (down & left or south west)
B	42	W (left or west)
F	46	NW (up & left or north west)

4.12 Entry to GIN MODE, Graphics Input Mode

Command code sequence to the graphics from host:

ESC SUB (1B,1A hex)

This code sequence puts the terminal in a graphics input mode. We are able to select between light pen option or crosshair, and therefore, the GIN MODE depends on this selection. Other kinds of input devices will act just as crosshair.
Connect a light pen directly to the graphics board by a cable, and point to a particular location on the display screen. A photoelectric cell in the light pen senses the CRT beam as it passes over the pen tip; the pen is activated and sends a response to the host with the coordinate of the pen tip. This device may give an inaccurate coordinate position, because the light pen can be pointing to the screen at different angles. The response format is described in figure 5 on the next page.

The 'Crosshair' is a cursor marked with a blinking or a non-blinking cross; it can be moved across the screen by using the cursor control keys on the keyboard, with the repeat feature enabled or disabled.

Repeat disabled (off) : the crosshair will move with a constant rate of one dot per keystroke.
Repeat enabled (on) : the crosshair will move over the screen with an increasing speed if a constant pressure is applied on the cursor control keys.

Once the crosshair is positioned and an alphanumeric key on the keyboard is hit, a response is sent to host (as described in figure 5). This mode will exit to AMODE after the character and crosshair cursor has been sent to host.

		BIT							
		7	6	5	4	3	2	1	0
	CHARACTER (7 bits)	P	X	X	X	X	X	X	X
CROSS HAIR CURSOR POSITION	HIGH X	P	0	1	X9	X8	X7	X6	X5
	LOW X	P	0	1	X4	X3	X2	X1	X0
	HIGH Y	P	0	1	Y9	Y8	Y7	Y6	Y5
	LOW Y	P	0	1	Y4	Y3	Y2	Y1	Y0
PROGRAMMABLE	TRAILER (e.g. CR)	P	X	X	X	X	X	X	X
	TRAILER (e.g. EOT)	P	X	X	X	X	X	X	X

NOTE: By ESC SUB the terminal is set to GIN MODE. This sequence will be sent to host when a character is sent to the Graphics Board from keyboard or other input devices.

Figure 5 Response to host from the graphics input device in GIN MODE.

4.13 Set Line Type

The following table gives the line type primitives which follow the Tektronix 4014 format and in addition three user-defined line types:

Sequence	Hex value	Line type
ESC `	1B,1E	Normal solid line (default)
ESC h	1B,68	"
ESC p	1B,70	"
ESC a	1B,61	Dotted
ESC i	1B,69	"
ESC q	1B,71	"
ESC b	1B,62	Dot-dashed
ESC j	1B,6A	"
ESC r	1B,72	"
ESC c	1B,63	Short dashed
ESC k	1B,6B	"
ESC s	1B,73	"
ESC d	1B,64	Long dashed
ESC l	1B,6C	"
ESC t	1b,74	"
ESC x	1B,78	User defined no. 1
ESC y	1B,79	User defined no. 2
ESC z	1B,7A	User defined no. 3

5. ENHANCED GRAPHICS

All functions described up to this point are defined by the Tektronix 4010/4014 format on a monochrome screen. These will be enough to drive PLOT 10 programs and most of the other graphics packages using this format. Remember, however, that this format is more than ten years old and the technology has taken an enormous step forward since then. So, in order to be better adjusted to more interactive solutions with software written under GKS and other modern standards, an add-on enhancement is included to give the total solution a better cost/performance.
In the following paragraphs you will find add-on enhancements to the Tektronix 4010/4014.

5.1 Text Attributes and Definitions

WARNING:

If you are using an existing software package with soft generated text and drawing routines (instead of text generated with the character fonts in AMODE), there is a chance that the soft generated text does not take notice of the change in resolution of the terminal. In that case, the character body may be scaled down to a small size, quite unreadable on the screen.

5.2 Select Character Font

Command code sequence to the graphics from host:

```
ESC " 1; <char set> h
```

where

```
<char set> := 0    International ASCII character font (default)
           := 1    Optional character font no. 1
           := 2    Optional character font no. 2
           := 3    Optional character font no. 3
           := 4    Optional character font no. 4
           := 7    Downloaded character font no. 1
           := 8    Downloaded character font no. 2
           := 9    Downloaded character font no. 3
```

This sequence is used to select a particular font to be used in AMODE.
It is also used to select the downloadable font to be used during the DEFINE Downloadable character sequence (see section 5.3).
The optional character font must be specified by the customer, and to generate downloadable character fonts, an optional font editor program written in 'C' is available.
This solves the problem with national font variants in a downloadable format instead of an optional national font version in PROM. Naturally, a downloadable font is more flexible, and can be used to define a picture element or to describe a logo or an icon.

5.3 Define Download Characters

Command code sequence to the graphics from host:

ESC ' 4;~~<character>~~<X><Y><y disp><data>........<data>.

where

<character> := character to be defined in the selected font (see 5.1)

<X> := X matrix size (range: 0< X < 64)

<Y> := Y matrix size (range: 0< Y < 64)

<y disp> := Displacement in y-direction relative to baseline. The value will be signed in range -32 to +31.

<data>.... := data to define the character matrix. Each of <X>, <Y>, <y disp> and <data> is defined in 6 bits data format as defined in section 2.2.

. := termination character

This sequence is used to download a character into a font library where the font number is specified. It doesn't matter in what order the characters are sent to the graphics board. You can send one character to a font and keep all the other characters unchanged if you want.
<data>.......<data> define the image of the character, packed as a binary bit stream. This bit stream will start at the bottom left position of the matrix, traversing to the right through the matrix to the upper right. The displacement value defines start address relatively to the baseline in AMODE from the bottom of a character.
An ASCII code of the character in range from 20 to 7F hex defines the character value, and one font is able to store up to 96 characters. There is, however, a restriction on the total amount of data available for defining fonts. The internal data structure allocates data for each font as follows:

1. number of characters up to 96
2. define bucket size to be equal to 15 bytes (120 bits)
3. total numbers of bucket per font equal to 219 .

Data is allocated in units of buckets, one bucket at a time. The number of buckets allocated depends on the size of the character to be defined. For example a 7x9 character requires 63 bits and will use one bucket, whereas a 12x12 character requires 144 bits and needs two buckets. When the total data of 219 buckets are used, the incoming data is discarded. If less data is sent than that required to fill the cell, the character cell is padded with zero bits.
With pixel multiplication you can handle a downloadable character in a similar way as ordinary characters. Three downloadable fonts are available (see section 5.2); when a font is defined, the current font will be used until a new one is defined.

example:

Define an 'H' character on a 5x7 matrix. In bitmap, the character will look like:

```
10001
10001
10001
11111
10001
10001
10001
```

and in 6 bits data stream format:

```
100011  000110  001111  111000  110001  10001x
(x is for padding )
```

and in ASCII code:

```
c   F   0   x   q   b
```

Definition of a character 'H' to be downloaded is given by:

~~ESC'4;HFG~~@c F0xqb.

Esc'H;EG....

5.4 Definition of Character Size

Command code sequence to the graphics from host:

```
ESC 0    set character size to 1x (default)
ESC 1    set character size to 2x
ESC 2    set character size to 3x
ESC 3    set character size to 4x
```

Characters are drawn into the graphics memory with a pixel multiplication factor determined by the character size equally in both x and y direction.

5.5 Select Pixel Multiplication Factor

Command code sequence to the graphics from host:

```
ESC " 3;<x fac>;<y fac> h
```

where

<x fac>,<y fac> := A decimal integer ranging from 1 to 10 and representing how many times a pixel is repeated when alpha mode characters are drawn into the graphics memory in each direction.
This command can be used instead of the ones described in section 5.4.

5.5 Init Downloadable Character Font

Command code sequence to the graphics from host:

```
                    ESC " 6 h
```

which "un-defines" all data in the currently selected downloadable character font.

5.6 Define Normal and Italic

Command code sequence to the graphics from host:

```
            ESC " 11; 0 h    set normal  (default)
            ESC " 11; 1 h    set Italic
```

This command gives you the possibility of slanting a character to the right, to simulate an italic type font from a normal font.

5.7 Enter/Exit Proportional Spacing Mode

Command code sequence to the graphics from host:

```
            ESC " 2; <enable/disable> h

        where

        <enable/disable>    := 0   disable (default)
                            := 1   enable
```

When proportional spacing mode is selected and a character is sent to the terminal in AMODE, the character will be drawn into the graphics memory. The spacing to the next character is given by a new character or a byte defined in a 6 bits data format. Rather than spacing right one cell position, the next received character is interpreted as the amount to add to the current X location. If pixel multiplication factors are used, the spacing value will be multiplied by this factor in x-direction. A '@' as spacing value will give zero displacement.

6. GRAPHICS ATTRIBUTES

6.1 Set Writing Mode

Command code sequence to the graphics from host:

```
ESC / 0 d     set writing dot on
ESC / 1 d     set writing dot off (delete)
ESC / 2 d     set writing dot to the
               complement value
```

6.2 Define User-specified Line Type

Command code sequence to the graphics from host:

```
ESC / <ps>;<ps>;.......a          User def no 1
ESC / <ps>;<ps>;.......b          User def no 2
ESC / <ps>;<ps>;.......c          User def no 3
```

The value of ps is defined by a decimal number and ranges between 1 and 63. The maximum number of <ps> is 10, and the sequence starts with the number of dots on followed by the number of dots off, and so on.

```
Example:   ESC/63a          define solid line on line 1.
           ESC/2;5;3;5b     define user line no. 2 with
                            2 dots on, 5 off, 3 on and 5 off.
```

Select line type as shown in section 4.13.

6.3 Rectangle Fill Operation

Command code sequence to the graphics from host:

```
ESC " 8; <X1>;<Y1>;<X2>;<Y2> h
```

The rectangle is defined by the coordinates (X1,Y1) and (X2,Y2). If the current writing mode is set to 'dot on', the bitmap area will be filled; if set to 'dot off', the rectangle area of the bitmap will be cleared and if the writing mode is complemented, all bits in the bitmap area will be complemented. This is a very useful function, particularly for area deletion.
The coordinates correspond to the scaling state as defined in section 7.4.

6.4 Deleting by Backspace

Command code sequence to the graphics from host:

```
ESC " 0 r   Disable deleting by backspace
ESC " 1 r   Enable  deleting by backspace
```

When you have set up the graphics with deleting by backspace and then are using backspace <08 hex>, you have to remember that you are deleting backwards with the current size or pixel multiplication factor.

6.5 Inking Mode

Command code sequence to the graphics from host:

```
ESC " 16; 0 h    Disable inking in GIN MODE
ESC " 16; 1 h    Enable  inking in GIN MODE
```

This sequence enables or disables "inking" in Graphics input mode. It can be controlled by the cursor control keys or by a pointing device, if available. Inking results in vectors connecting each point where the crosshair is moving.

7. CURSOR AND DEVICE ATTRIBUTES

7.1 Load Crosshair Cursor

Command code sequence to the graphics from host:

```
ESC / f
```

The current vector location is loaded into the crosshair cursor location. This gives you the possibility of addressing the crosshair, and it will appear at this location whenever GIN MODE is activated.

7.2 Cursor in AMODE

Command code sequence to the graphics from host:

```
ESC " 10; 0 h          Disable alpha cursor
ESC " 10; 1 h          Enable  alpha cursor
```

Tektronix 4010 alpha cursor can be disabled or enabled on the screen.

7.3 Crosshair Flashing

Command code sequence to the graphics from host:

```
ESC " 12; 0 h    Disable crosshair flashing
ESC " 12; 1 h    Enable  crosshair flashing
```

As a default the crosshair will flash. This sequence allows the flashing to be disabled. The crosshair, however, will always be displayed.

7.4 Repeating Cursor Control Keys

Command code sequence to the graphics from host:

```
ESC " 15; 0 h       Disable repeating cursor control keys
ESC " 15; 1 h       Enable  repeating cursor control keys
```

This sequence enables or disables the repeat feature of the keys. When disabled, the crosshair will move one pixel for each cursor movement code sent to the graphics board. The default condition is 'enabled'.

7.5 Scaling

Command code sequence to the graphics from host:

```
ESC " 5; 0 h        Set Tektronix 4010 scaling
ESC " 5; 1 h        Set direct scaling
```

Tektronix 4010 scaling is the default value, and the addressing range with coordinates between (0,0) and (1024,780). The same addressing range for direct scaling is with coordinates between (0,0) and (720,336).

7.6 Graphics Video

Command code sequence to the graphics from host:

```
ESC " 7; 0 h        Disable graphics video
ESC " 7; 1 h        Enable  graphics video
```

This command gives us the possibility of disabling graphics video plane. You can communicate, load and store information in the graphics board without displaying the results. Enable graphics video plane is a default state.

8. INITIALIZATION AND PRINTER SETUP

8.1 Clear Graphics Only

Command code sequence to the graphics from host:

```
ESC " 9 h
```

Erases the graphics display without affecting the terminal in TDMODE.

8.2 Report Version

Command code sequence to the graphics from host:

```
ESC " 17 h
```

The graphics terminal is asked to report a version number back to host, consisting of the firmware identification and revision. The message format will be:

```
TD720  FW067  Vx.yy (C) Copyright  Digital  Engineering,
Inc 1984
```

where x is the release, and yy is one or two digit numbers representing the level. The report version information is used by the host driver program; the host needs to know whether a function is to be solved by host or by the terminal. Later version will normally be able to do more in the terminal and thereby increase speed to handle different applications.

8.3 Select Input Device

Command code sequence to the graphics from host:

```
ESC " 0 q   init crosshair
ESC " 1 q   init light pen
```

Sequence to select input device type before entering GIN MODE with an ESC SUB sequence.

8.4 Enter GIN MODE with Crosshair

Command code sequence to the graphics from host:

```
ESC " 4 g
```

This mode puts the crosshair to any point on the display screen by the cursor control keys. The moving speed depends on how long the keys are pressed. This mode can also be executed by ESC SUB as in section 8.5.

8.5 Enter GIN MODE with Light Pen (optional)

Command code sequence to the graphics from host:

```
ESC SUB      (if input device of light pen is selected)
ESC " 5 g
```

If no light pen is present, crosshair will appear on the screen.

8.6 Specify Printer

Command code sequence to the graphics from host:

```
ESC " 13; 0 h        Epson MX100
```

This command will specify printer to MX100; still, it can drive an Epson MX80 / FX80 or RX80, but only with rotated plot.

8.7 Select Printer Options

Command code sequence to the graphics from host:

```
ESC " <hor/rot>;<sqr>;<opt>;<ff> n
```

where

<hor/rot>	:=	0	Horizontal plot (default)
	:=	1	Rotated plot
<sqr>	:=	0	Without any squaring algorithm (default)
	:=	1	Apply squaring algorithm
<opt>	:=	0	Option (not in use)
<ff>	:=	0	No form feed (default)
	:=	1	Form feed before printing
	:=	2	Form feed after printing
	:=	3	Form feed before and after printing

Can be used in connection with the initialization of printer routine in graphics.

8.8 Print Window Lines

Command code sequence to the graphics from host:

```
ESC " 14 ; <line 1>;<line 2> h
```

This sequence initiates a dump of the Graphics memory to the printer. The <line 1> and <line 2> consist of decimal numbers, which may have values from 0 to max number of line - 1.
<line 1> greater than <line 2>.

8.9 Set Transmission Delay from Graphics Board

Command code sequence to the graphics from host:

```
ESC " <n> d
```

where

<n> := number of line ticks between transmitted characters.

This value represents the delay time in line ticks (50, 60 or 70 per second) between transmitted characters.

8.10 Select Trailer Options

Command code sequence to the graphics from host:

```
ESC " <trailer 1>;<trailer 2> l
```

This command allows selection of trailer codes, <trailer 1> and <trailer 2>. They are given in decimal numbers representing the ASCII code to be used as a trailer code. A value of 255 suppresses transmission of a trailer code. A NULL is a valid code and will be selected by a zero or missing parameter.
Default values are CR for <trailer 1>, while <trailer 2> is suppressed.

9. COMPRESSED BIT-MAP

9.1 Block Transfer Load Address

Command code sequence to the graphics from host:

```
ESC " <X>;<Y> a
```

With this command we can set up loading address in the graphics memory and load down a bitmap picture from host. In compressed bit-map transfer of a picture from host, the addresses <X> and <Y> represent the actual pixel address in range Xmax - 1 and Ymax -1 (719 and 335).
The addressing is in device coordinate system with (0,0) in lower left corner and (719,335) in upper right corner.

9.2 Compressed Bit-map Data Transfer to Graphics

Command code sequence to the graphics from host:

```
ESC  * <data><data>......<data> #
```

Data is in 6 bits format as described in section 2.2 and is loaded into the graphics memory in successive addresses, six bits at a time, starting with the address given by the 'Block Transfer Load Address' sequence (see section 9.1).
The low six bits of <data> are copied into bit-map. Data is loaded successively to the right, and will wrap to the beginning of the next highest line unless this command sequence is terminated by '#'.
Compression can be done by inserting the following sequences in the <data> sequence:

$ <count>	Compressed zeros
% <count>	Compressed ones
& <data> <count>	Compressed data value

<count> is a six bit unsigned value, six bits data format in range 0 to 63. A sequence of <data> <count> gives the possibility of transferring a pattern of six bits.

Example:
A sequence of ten zeroes followed by eight times the bit-pattern 101010 can be expressed in a sequence by:

```
$ J & j H
```

9.3 Compressed Bit-map Data Transfer from Graphics

Command code sequence to the graphics from host:

```
ESC " <X>; <Y>;<count> C
```

This sequence starts reading an image area from the bit-map into host. Start of pixel address is given by <X>,<Y> coordinates in the device coordinate system. Data is transferred in six bits data format in the same form and format as described in section 9.2, transfer to graphics. <count> represents a pixel count for the image we want to read back to memory and can range from 1 to 65535. 'C' is termination of the command. Bypass condition is entered, and the transfer is terminated by the current trailer codes.