NAME

mfbcap − graphics terminal capability data base

SYNOPSIS

˜cad/lib/mfbcap

DESCRIPTION

MFBCAP is a data base describing graphics terminals, used, *e.g.*, by *kic*(1) and *mfb*(3). Terminals are described in *MFBCAP* by defining a set of capabilities that they have, and by describing how operations are performed. Output delays and initialization sequences are also included in *MFBCAP*.

Entries in *MFBCAP* consist of a set of comma (,) separated fields. Entries may continue onto multiple lines by beginning a continuation line with either a tab or space character. The first entry for each terminal gives the names by which the terminal is known, separated by vertical bar (|) characters. The first name is always 2 characters long, the second name given is the most common abbreviation for the terminal, and the last name given should be a long name fully identifying the terminal. The second name should contain no blanks; the last name may contain blanks for readability. For compatability with other operating systems, it is recommended that the device names use all UPPER CASE LETTERS.

CAPABILITIES

The Parms column indicates which of the four possible parameters are used in the encoding/decoding of string variable. Upper case letters signify that the parameter value is passed to the *mfb(3)* routine, and lower case letters identify values that are returned.

Name	**Type**	**Parms**	**Description**
8BB	boolean		transmit in 8 Bit Binary using LITOUT
ALUEOR	string		set ALU writing mode to Exclusive OR
ALUJAM	string		set ALU writing mode to JAM (replace mode)
ALUNOR	string		set ALU writing mode to NOR
ALUOR	string		set ALU writing mode to OR
APT	boolean		Accurately Positionable Text
BELL	string		ring the terminals BELL
BLD	boolean		BLinkers Definable
BLE	string		BLinkers End
BLS	string	XYZT	BLinkers Start
			X = off color ID
			Y = red/hue intensity when blinked
			Z = green/lightness intensity when blinked
			T = blue/saturation intensity when blinked
BU1	numeric		value returned by BUtton 1 of pointing device
BU2	numeric		value returned by BUtton 2 of pointing device
BU3	numeric		value returned by BUtton 3 of pointing device
BU4	numeric		value returned by BUtton 4 of pointing device
BU5	numeric		value returned by BUtton 5 of pointing device
BU6	numeric		value returned by BUtton 6 of pointing device
BU7	numeric		value returned by BUtton 7 of pointing device
BU8	numeric		value returned by BUtton 8 of pointing device
BU9	numeric		value returned by BUtton 9 of pointing device
BU10	numeric		value returned by BUtton 10 of pointing device
BU11	numeric		value returned by BUtton 11 of pointing device
BU12	numeric		value returned by BUtton 12 of pointing device

DBS	string	XYZT	Draw Box Sequence X = lower left Y = lower bottom Z = upper right T = upper top
DCS	string	XYZ	Draw Circle Sequence X = center x coordinate Y = center y coordinate Z = radius of circle
DFP	boolean		Definable Fill Patterns
DLP	boolean		Definable Line Patterns
DLS	string	XYZT	Draw Line Sequence X,Y = start coordinate Z,T = end coordinate
DLT	string	XY	Draw Line To (x,y) sequence X,Y = next current graphics position
DSL	string	XYZT	Draw Solid Line sequence X,Y = start coordinate Z,T = end coordinate
DSLT	string	XY	Draw Solid Line To (x,y) sequence X,Y = next current graphics position
DSB	string	XYZT	Draw Solid Box sequence X = lower left Y = lower bottom Z = upper right T = upper top
FDE	string	X	Fill pattern Define End X = style ID
FDF	string	XY	Fill pattern Define Format X = style ID Y = one 8 bit row/col of the fill pattern array
FDH	numeric		Fill pattern Define Height in rows
FDR	boolean		Fill pattern Define Row major
FDS	string	X	Fill pattern Define Start X = style ID
FDW	numeric		Fill pattern Define Width in columns
FPOLY	boolean		terminal is capable of Filled POLYgons
GCH	numeric		Graphics Character Height
GCS	string		Graphics Clear Screen (in current color)
GCW	numeric		Graphics Character Width
GFS	string		Graphics Finish String
GIS	string		Graphics Initialization String
GTE	string		Graphics Text End
GTH	numeric		Graphics Text Height offset
GTO	boolean		Graphics Text Overstrikes old text
GTR	boolean		Graphics Text Replaces old text
GTS	string	XYZ	Graphics Text Start X,Y = lower left coordinate of text string Z = number of characters in text string
GTW	numeric		Graphics Text Width offset
HLS	boolean		convert RGB color definitions to HLS
ICS	string		Initialize predefined Color Styles
IFP	string		Initialize predefined Fill Patterns

ILS	string		Initialize predefined Line Styles
KYB	string		KeYboard Backspace sequence
KYBRD	boolean		Terminal has a KeYBoaRD
KYE	string		KeYboard End sequence
KYS	string	XY	KeYboard Start sequence X,Y = lower left coordinate of keyboard window
KYX	numeric		KeYboard X offset
KYY	numeric		KeYboard Y offset
LDE	string	X	Line Define End X = style ID
LDF	string	XY	Line Define Format X = style ID Y = 8 bit fill pattern
LDL	numeric		Line Define Length (in bytes)
LDS	string	X	Line Define Start X = style ID
MCE	string		device behaves like the following MfbCap Entry
MCL	numeric		Maximum number of Colors
MFP	numeric		Maximum number of Fill Patterns
MLS	numeric		Maximum number of Line Styles
MPS	string	XY	Move Pen Sequence X,Y = coordinate to move graphics cursor
MXC	numeric		Maximum X Coordinate
MYC	numeric		Maximum Y Coordinate
NBL	numeric		Number of BLinkers
NPB	numeric		Number of Pointing device Buttons
OFFDX	numeric		length of OFF screen memory in X Direction
OFFDY	numeric		length of OFF screen memory in Y Direction
OFFMX	numeric		minimum X coodinate of OFF screen Memory
OFFMY	numeric		minimum Y coodinate of OFF screen Memory
OMO	string		Overstrike text Mode On sequence
PDB	boolean		Pointing Device has Buttons
PDE	string		Pointing Device End
PDF	string	xyzt	Pointing Device coordinate Format x,y = input coordinate z = key pushed t = button mask
PDR	string		Pointing Device initiate Read
PDS	string		Pointing Device Start
PLE	string	XY	PoLygon End sequence X,Y = first coordinate in the polygon sequence
PLS	string	XYZ	PoLygon Start sequence X,Y = first of Z coordinates Z = number of coordinates
PLSOL	string	XYZ	PoLygon start sequence for SOLid fill X,Y = first of Z coordinates Z = number of coordinates
PLV	string	XY	send PoLygon Vertex sequence X,Y = next coordinate in the polygon sequence
POD	boolean		terminal has POinting Device
PRBOFF	string		disable Pointing device Rubber Banding
PRBON	string		enable Pointing device Rubber Banding
PRI	boolean		Pointing Read Immediately returns coordinates

RAW	boolean		drive device in RAW mode
RLS	boolean		Reissue Line Style before each line
RMO	string		Replace text Mode On sequence
ROT	boolean		ROTatable graphics text
RTS	string	X	Rotate Text Sequence X = angle of rotation in degrees (-360 <= X <= 360)
RSCPE	string		RaSter CoPy End sequence
RSCPS	string		RaSter CoPy Start sequence
RSCSF	boolean		transmit RaSter Copy Source coordinate First
RSDST	string	XYZT	RaSter copy DeSTination sequence X,Y = destination coordinate Z,T = length,width of area to be copied
RSSRC	string	XYZT	RaSter copy SouRCe sequence X,Y = source coordinate Z,T = length,width of area to be copied
SCS	string	X	Set Color Style X = new color ID
SFP	string	X	Set Fill Pattern X = new fill pattern ID
SLS	string	X	Set Line Style X = new line style ID
SRM	string	X	Set video Read Mask X = channel read mask
SSFP	string	X	Set Solid Fill Pattern X = new fill pattern ID
SSLS	string	X	Set Solid Line Style X = new line style ID
TTY	boolean		device is a TTY
VLT	boolean		Video Lookup Table present
VTE	string	XYZT	Video Table Entry X = color ID of new entry Y = red/hue intensity Z = green/lightness intensity T = blue/saturation intensity
VTI	numeric		Video Table maximum Intensity
VTL	numeric		VLT Length expressed as number of bit planes
VWM	string	X	Video Write Mask X = channel write mask
WPX	string	XY	Write PiXel at coordinate XY

A Sample Entry

The following entry describes the HP 2648. (This particular 2648 entry may be outdated, and is used as an example only.)

```
#
# HP2648 with keyboard cursor control
#
h0|H0|2648|HP2648|HP2648A|Hewlett-Packard 2648A,
        TTY, APT, MXC#719, MYC#359, MCL#2, MFP#8, MLS#2,
        GTO, DFP, DLP,
        MPS=\E*pa%X%d\,%Y%dZ,
        DLT=\E*pf%X%d\,%Y%dZ, RLS,
        DBS=\E*m3b%X%d\,%Y%d\,%Z%d\,%T%dE,
        DSL=\E*m1B\E*pa%X%d\,%Y%d\,%Z%d\,%T%dZ,
        DLS=\E*pa%X%d\,%Y%d\,%Z%d\,%T%dZ,
        WPX=\E*pa%X%d\,%Y%d\,%X%d\,%Y%dZ,
        PLS=\E*pa%X%d\,%Y%d,
        PLV=\,%X%d\,%Y%d,
        PLE=\,%X%d\,%Y%dZ,
        LDL#1, LDF=\E*m%Y%d 1C,
        GCS=\E*d%X%+A%c$<#500>, GCH#11, GCW#7,
        GFS=\EH\EJ\E*mR\E*dlaeD$<#2500>,
        GIS=\E*mR\E*dlafC$<#3500>,
        GTE=\E*dT, GTH#1, GTW#1,
        GTS=\E*pa%X%d\,%Y%dZ\E*dS,
        KYBRD, KYB=^H,
        KYS=\E*pa%X%d\,%Y%dZ\E*m4a\E*dS,
        KYE=\E*dT, KYX#1, KYY#1,
        SFP=\E&f%X%+1%cE\21,
        SCS=\E*m%X%+1%cA,
        SLS=\E*m2B,
        FDH#8, FDW#8, FDR, FDF= %Y%3,
        FDS=\E&f1a%X%+1%ck36L\E*m,
        FDE=D$<#90>,
        POD, PDR=\E*s4\^\021, PDS=\E*dK,
        PRBON=\E*dM, PRBOFF=\E*dN,
        PDF=\+%d\,%X\+%d\,%Y%3%Z%c, PDE=\021\E*dL,
```

Capabilities in *MFBCAP* are of three types: Boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal, and string capabilities which give a sequence that can be used to perform particular terminal operations.

Types of Capabilities

All capabilities have an identifying code. For instance, because the HP2648 has ‘‘accurately positionable text’’ (*i.e.,* graphics text may be positioned with lower left corner at any pixel on the screen) is indicated by the boolean **APT**. Hence the description of the HP2648 includes **APT**. Numeric capabilities are followed by the character ‘#’ and then the value. Thus **MXC** which specifies the maximum value of the X coordinate on the terminal viewport gives the value ‘719’ for the HP2648.

Formatting String Capabilities

String variables have a formatting capability to be used for encoding numbers into ASCII strings and decoding ASCII strings into numbers. An example of the former is the capability **DBS** (for **D**raw **B**ox **S**equence), which takes four numbers (X, Y, Z, and T) and generates the proper sequence to draw a box from the lower left corner (X,Y) to the upper right corner (Z,T). An example of a string decode is the capability **PDF** (for **P**ointing **D**evice **F**ormat), which takes an ASCII string from the input stream and extracts from it an x and y coordinate, a key (if one was pushed) and a buttonmask (if a cursor button was pushed).

String Formatting

The string variables have a formatting capability which uses four variables (X, Y, Z, and T) to generate a formated string (with *MFBGenCode),* or generates four variables (X, Y, Z, and T) from a formated string (with *MFBDecode).* Two temporary registers represented by the letters **R** and **r** are available. All operations begin with a percent sign '**%**', and they are listed below:

Com	**Command Description encode/(decode)**
%X	set value/(X variable) to the X variable/(value).
%Y	set value/(Y variable) to the Y variable/(value).
%Z	set value/(Z variable) to the Z variable/(value).
%T	set value/(T variable) to the T variable/(value).
%C	set value to the current foreground color ID.
%F	set value to the current fill pattern ID.
%L	set value to the current line style ID.
%d	output/(input) value in variable length decimal format
%2	output/(input) value converting to/(from) two decimal digits.
%3	output/(input) value converting to/(from) three decimal digits.
%c	output/(input) least significant byte of value withoutconversions.
%h1	output/(input) least significant four bits converting to/(from) one ASCII hex character.
%h2	output/(input) least significant byte converting to/(from) two ASCII hex characters.
%h3	output/(input) least significant twelve bits converting to/(from) three ASCII hex characters.
%h4	output/(input) least significant sixteen bits converting to/(from) four ASCII hex characters.
%o1	output/(input) least significant three bits converting to/(from) one ASCII octal character.
%o2	output/(input) least significant six bits converting to/(from) two ASCII octal characters.
%o3	output/(input) least significant nine bits converting to/(from) three ASCII octal characters.
%o4	output/(input) least significant twelve bits converting to/(from) four ASCII octal characters.
%o5	output/(input) least significant fifteen bits converting to/(from) five ASCII octal characters.
%o6	output/(input) least significant sixteen bits converting to/(from) six ASCII octal characters.
%t1	output/(input) X and Y in Tektronix format.
%t2	output/(input) Z and T in Tektronix format.
%t3	output X and R in Tektronix format (MFBGenCode only).
%t4	output R and Y in Tektronix format (MFBGenCode only).
%t5	output R and r in Tektronix format (MFBGenCode only).
%ti	output/(input) value in Tektronix integer format.

%tr	output value in Tektronix real format.
%R	store/(retrieve) value in temporary register 1.
%r	store/(retrieve) value in temporary register 2.
%+x	add x to value.
%-x	subtract x from value.
%*x	multiply value by x.
%/x	divide value by x.
%>>x	shift value right by x bits.
%<<x	shift value left by x bits.
%|x	OR x with value.
%&x	AND x with value.
%^x	EOR x with value.
%=x	set value equal to x.
%ax	set value equal to the absolute value of x.
%~	Complement value (1's complement).
%@	output a single null character (MFBGenCode only).
%%	gives '%'.
%B	BCD (2 decimal digits encoded in one byte).
%D	Delta Data (backwards bcd).

Where x can be:

(1) One byte - the numeric value of this byte is used as x.

(2) The character "#" followed by a decimal integer value for x.

(3) The character "%" followed by C, F, L, X, Y, Z, T, r, or R - the value of C, F, L, X, Y, Z, T, r or R is used.

The command formats are similar to those found in *termcap(5)* or *terminfo(5),* but are more complicated due to the more rigorous requirements of graphics terminals.

Preparing Descriptions

We now outline how to prepare *MFBCAP* descriptions of graphics terminals. The most effective way to prepare a terminal description is to build up a description gradually, using partial descriptions with simple *mfb(3)* test routines to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the *MFBCAP* file to describe it. To easily test a new terminal description you can set the environment variable **MFBCAP** to a pathname of a file containing the description you are working on. After setting the environment variable, any program that uses *mfb(3), e.g., kic,* will look at the pathname defined by the environment variable instead of *~cad/lib/mfbcap.*

Delays

Delays may be embedded anywhere in a string capability and is distinguished by the **$<** and **>** brackets. The number contained within these brackets describes the delay in milliseconds to be generated and must conform to the above description for the variable 'x' (*e.g.,* an integer constant must be preceded by the character "#"). Before each delay, the output buffer is flushed.

Basic Capabilities

The number of pixels on a horizontal row of the display is given by the **MXC** numeric capability, and the number of pixels in a vertical column is given by the **MXY** capability. The number of colors available on the display is specified by the **MCL** capability. For black and white terminals, such as the HP2648, the **MCL** capability is defined as two. The maximum number of stipple fill patterns and line styles is given by the **MFP** and **MLS** numeric capabilities respectively.

Off screen memory refers to an area of the viewport in pixel coordinates which is not displayed. The lower, left corner of the off screen memory is specified by the **OFFMX** and **OFFMY** numeric capabilities. The horizontal length of the off screen memory is specified by the **OFFDX** numeric capability, and the vertical width of the off screen memory is specified by the **OFFDY** numeric capability.

MFBCAP allows two sequences for initializing and uninitializing the terminal. The first initialization string sent to the terminal is given by the **GIS** format string. This will be the first sequence sent to the graphics device. The graphics finish/termination string is given by the **GFS** format string. This will be the last sequence sent to the graphics device.

The initialization character sequences for color styles, fill styles, and line styles are defined respectively by the **ICS**, **IFP**, and **ILS** format strings.

The sequence to ring the terminals bell or alarm is defined by the **BELL** string and defaults to control-G.

Setting Colors and Styles

The character sequence for setting the current foreground color is defined by the **SCS** format string. All subsequent geometries will be drawn in this color. The format for setting the current line style is given by the **SLS** format string, and the format for setting the current fill style is given by the **SFP** format string. All subsequent lines, boxes, and polygons will be drawn with these styles. *MFBCAP* assumes that style zero defines a solid line and fill pattern. If this is not the case for a particular frame buffer, or the format for setting a solid line or fill style is inconsistent with that for other line styles, such as is the case for the HP9872, a character sequence for setting the solid line or fill style is defined by the **SSLS** and **SSFP** format strings respectively. If it is necessary for the current line style to be reissued before a line is drawn (as is the case for the HP 2648), then the **RLS** boolean must be present in the *MFBCAP* entry.

Basic Geometries

The character sequence for moving the current graphics position to a x,y pixel coordinate is defined by the **MPS** format string. The format for drawing a line in the current line style from the current graphics position to a x,y pixel coordinate is defined by the **DLT** format string. The character sequence to draw a line in the current line style from a x,y pixel coordinate to a z,t pixel coordinate is defined by the **DLS** format string. If the command for drawing a solid line is different from that for a non-solid line, the character sequence to draw a solid line from a x,y pixel coordinate to a z,t pixel coordinate may be defined by the **DSL** string capability. The format for drawing a solid line from the current graphics position to a x,y pixel coordinate is defined by the **DSLT** format string.

The sequence for drawing a box in the current foreground color from the lower left x,y pixel coordinate to the upper right z,t pixel coordinate is specified by the **DBS** format string. Because some terminals, such as the Tektronix 4113, have special raster commands for drawing solid boxes, a format for drawing solid boxes may be specified by the **DSB** format string.

The format for setting a pixel in the current color at the x,y pixel coordinate is defined by the **WPX** format string.

The format for drawing a circle with its center at the x,y pixel coordinate and having a radius of z pixels is defined by the **DCS** format string.

The format for clearing the entire screen to the current color is given by the **GCS** format string. If there is no such command sequence, it may be substituted by the command sequence that will write a solid box in the current color over the entire screen.

There are three format strings for defining the terminal's polygon command sequence. First the **PLS** starting sequence is used to define x,y as the first of z pixel coordinates. This character sequence will be followed by z-1 occurrences of the **PLV** format string which defines the remaining vertices of the polygonal path. Finally, an ending sequence that is defined by the **PLE** format string terminates the polygon sequence. For terminals which have inconsistent formats for drawing solid polygons, the **PLSOL** sequence may be used in place of the **PLS** sequence. If the terminal is capable of drawing a filled polygon in the current fill pattern, then the **FPOLY** boolean should appear in the *MFBCAP* entry.

Video Layer Table

If the terminal has a video layer table, then the **VLT** boolean must be present in the *MFBCAP* entry. *MFBCAP* assumes that the VLT uses the red-green-blue system for defining colors. If the **HLS** boolean capability is specified, then the RGB arguments become HLS (hue-lightness-saturation) values. The maximum intensity of red, green, or blue in the VLT (or the lightness or saturation if using the HLS system) is given by the **VTI** numeric capability. The format for setting a particular entry of the VLT is given by the **VTE** format string. The **VTL** numeric value can be used to define the length of the VLT in terms of the number of bit planes.

Defining Styles

The **LDS** string capability defines the sequence for (re)defining a line style corresponding to a particular style ID. The **LDF** format string is used to define an eight bit mask that represents the new line style. The **LDE** format string terminates the definition of the new line style.

The definition of a new fill pattern is more complicated than is the case for line styles. It is necessary to transform an eight by eight intensity array into whatever command syntax is required by the terminal. The **FDS** string capability is used to begin the (re)definition of a fill pattern. The **FDF** format string defines one row or column of the fill pattern using an eight bit mask (one row of the eight by eight intensity array). If the **FDR** boolean is present, then it is assumed that the fill pattern is being defined by rows in which case the **FDF** sequence is sent by the number of times defined by the **FDH** numeric capability. Otherwise, it is assumed that the fill pattern is defined by columns, and the **FDF** sequence is sent by the number of times defined by the **FDW** numeric capability. If, for example, the number of rows in the fill pattern is ten, the **FDF** sequence is first transmitted using each of the eight rows of the initial eight by eight intensity array, and then the sequence is sent twice using the first and second rows of the initial intensity array. The **FDE** format string terminates the definition of the new fill pattern.

Raster Capabilities

There are four format strings for defining the terminal's raster copy command sequence. First the **RSCPS** starting sequence is used to begin the raster copy command. This character sequence will be followed the **RSSRC** format string which defines the lower, left coordinate and length and width of the source area and the **RSDST** format string which defines the lower left coordinate and the length and width of the destination area. The **RSSRC** sequence appears first only if the **RSCSF** boolean is defined. Finally, an ending sequence that is defined by the **RSCPE** format string terminates the raster copy sequence.

Graphic Text

MFBCAP supports a single font graphic text. The height and width of the text font are given respectively by the **GCH** and **GCW** numeric capabilities. Graphics text is displayed with three format strings. A text string with z characters with a lower left justification at the x,y pixel coordinate is begun with the format string defined by **GTS**. This will be followed by the transmission of the z characters and terminated by the format string defined by **GTE**. The graphic text can offset from the current graphics position by setting the numeric capabilities **GTH** and **GTW**. The following figure demonstrates the assumed character font for the two characters "gh". The character "0" marks a pixel in the character font, and the character "X" marks the x,y pixel coordinate to which the two characters where justified. Note that the **GCH**, **GCW**, **GTH**, and **GTW** numeric capabilities must always be non-negative integers.

```
               _  . . . . . .       0 . . . . .
                 |  .         .       0         .
                 |  .         .       0         .
                 |  . 0 0 0 0 .       0 0 0 0   .
                 |  0         0       0       0 .
                 |  0         0       0         0
    _        GCH  0         0       0         0
    |            |  X 0 0 0 0 0       0         0
    |            |  .         0       .         .
   GTH           |  .         0       .         .
    |            |  0         0       .         .
    |            |  . 0 0 0 0 .       . . . . . .
    -            -
                    | -GCW-  | -GTW-  |
```

If the terminal supports rotatable graphic text, then the **ROT** boolean is present in the *MFBCAP* entry. If rotated text is desired the **RTS** character sequence is issued prior to the the **GTS** sequence and defines a rotation of x degrees, where x is between -360 and 360.

MFBCAP supports two graphic text modes. If the graphic text can be destructive, then the **GTR** boolean is present in the *MFBCAP* entry, and the **RMO** format string specifies the character sequence for entering the destructive graphic text mode. If the terminal has graphic text that can overstrike, then the **GTO** boolean is present in the *MFBCAP* entry, and the **OMO** format string defines the character sequence for entering the overstriking graphic text mode.

Keyboard Control

For terminals with special keyboard/cursor operations, *MFBCAP* provides a set of string capabilities for controlling keyboard input. The keyboard is initialized, and the current graphics position is moved to the x,y pixel coordinate by the **KYS** format string. The current graphics position can be offset upward from the above x,y pixel coordinate by setting values to the **KYX** and **KYY** numeric capabilities. The keyboard backspace sequence is defined by the **KYB** format string (the is NO default for the backspace format string). The keyboard is uninitialized by the **KYE** format string.

If the terminal does not have the above capabilities, a keyboard input routine, such as that used in *mfb(3),* can use the terminals graphic text capabilities to echo keyboard characters on a command line.

Pointing Device.

If the terminal has a pointing device, then the **POD** boolean is present in the *MFBCAP* entry. If the pointing device has buttons, then the **PDB** boolean is set, the number of buttons is given by the **NPB** numeric capability, and the values returned by the respective buttons of the pointing device are defined by the **BU1** through **BU12** numeric capabilities.

The graphics pointing device is initialized with the **PDS** format string. The **PDR** format string places the terminal in a waiting mode until the first graphic input. When this event occurs, the locator event is decoded by the **PDF** format string. The graphic pointing device is uninitialized by the **PDE** format string.

If the **PRI** boolean is set, one character is read immediately after the pointing device initialization sequence **PDS** and before the pointing device is enabled by **PDR**. This is useful for terminals that have a cursor and can read its current position but do not have the capability of a graphic event (*i.e.,* an x,y pixel coordinate that is read immediately after a key or button is pushed on the terminal).

The pointing device encoding format string **PDF** must assume that the pointing device will send one signature character. After the pointing is activated by the **PDR** format string, the first character transmitted from the terminal must be identical to the first character of the **PDF** format string. If the characters do not match, then the *MFBDecode* routine used by *mfb(3)* will return the first character that was transmitted by the terminal.

The **PRBON** string capability defines the character sequence to enable rubber banding of the pointing device, and **PRBOFF** disables the rubber banding. *MFBCAP* assumes that the center of rubber banding is the current graphics position that can be defined by the **MPS** format string defined above.

Special Modes

If the graphics device is to be handled as a TTY, then the **TTY** boolean must be present in the *MFBCAP* entry. If the graphics encoding can produce 8 bit, nonASCII characters, then the **8BB** boolean must be included. If the graphics device is a TTY and is to be driven in a *RAW* mode, then the **RAW** boolean must be included. See the manual for *tty(4).* Typically, this mode is used only if the device has no keyboard.

MFBCAP supports four ALU writing modes. These are the modes in which a pixel is updated when written over. The four possible modes are JAM (replace mode), OR, EOR, and NOR. The sequences for setting these modes are **ALUJAM**, **ALUOR**, **ALUEOR**, and **ALUNOR** respectively.

Similar Terminals

If there are two very similar terminals, one can be defined as being like the other but with certain exceptions. The string capability **MCE** is given with the name of the similar terminal. The **MCE** must be the last capability defined in the entry, and the combined length of the two entries must not exceed 4096 characters. Because *mfb* routines scan the entry from left to right, and because the **MCE** entry is replaced by the corresponding entry, the capabilities given on the left override identical capabilities defined for the similar terminal. This is useful for defining different modes for a terminal, or for defining terminals with different peripherals.

FILES

~cad/lib/mfbcap file containing terminal descriptions

SEE ALSO

termcap(5), mfb(3), kic(CAD1)

AUTHOR

Giles Billingsley

BUGS

The total length of a single entry (excluding only escaped new lines) may not exceed 8192.

There is a restriction that allows a simple parser to be used for the *MFBCAP* file. The delimiter is assumed to be a comma that is not immediately preceded by a slash (\) character. String capabilities that terminate with a slash character (as is the case for the vt125) must therefore separate the delimiting comma and the slash character with a padding character.

Not all programs support all entries. There are entries that may not be used by any program.