TPCrt

FrameWindow

```
procedure FrameWindow(LeftCol, TopRow, RightCol, BotRow, FAttr, HAttr :
Byte; Header : shortstring);
```

Draw a frame around a window

WhereXAbs

```
function WhereXAbs: Byte;
```

Return absolute column coordinate of cursor

WhereYAbs

```
function WhereYAbs: Byte;
```

Return absolute row coordinate of cursor

function WhereXY: Word;

```
{-Return absolute coordinates of cursor}
```

function ScreenX: Byte;

```
{-Return absolute column coordinate of cursor}
```

function ScreenY: Byte;

```
{-Return absolute row coordinate of cursor}
```

procedure FastWrite(St : string; Row, Col, Attr : Byte);

```
{-Write St at Row,Col in Attr (video attribute) without snow}
```

procedure SetFrameChars(Vertical, Horizontal, LowerRight, UpperRight,

```
                        LowerLeft, UpperLeft : Char);
{-Sets the frame characters to be used on subsequent FrameWindow calls.}
```

procedure WhereXYdirect(var X, Y : Byte);

```
{-Read the current position of the cursor directly from the CRT controller}
```

function GetCrtMode : Byte; {-Get the current video mode. Also reinitializes internal variables. May

```
 reset: CurrentMode, ScreenWidth, ScreenHeight, CurrentPage, and
 VideoSegment.}
```

procedure GotoXYAbs(X, Y : Byte);

```
{-Move cursor to column X, row Y. No error checking done.}
```

procedure SetVisiblePage(PageNum : Byte);

```
{-Set current video page}
```

procedure ScrollWindowUp(XLo, YLo, XHi, YHi, Lines : Byte);

```
{-Scrolls the designated window up the specified number of lines.}
```

procedure ScrollWindowDown(XLo, YLo, XHi, YHi, Lines : Byte);

```
{-Scrolls the designated window down the specified number of lines.}
```

function CursorTypeSL : Word;

```
{-Returns a word. High byte has starting scan line, low byte has ending.}
```

function CursorStartLine : Byte;

```
{-Returns the starting scan line of the cursor}
```

function CursorEndLine : Byte;

```
{-Returns the ending scan line of the cursor.}
```

procedure SetCursorSize(Startline, EndLine : Byte);

```
{-Sets the cursor's starting and ending scan lines.}
```

procedure NormalCursor;

```
{-Set normal scan lines for cursor based on current video mode}
```

procedure FatCursor;

```
{-Set larger scan lines for cursor based on current video mode}
```

procedure BlockCursor;

```
{-Set scan lines for a block cursor}
```

procedure HiddenCursor;

```
{-Hide the cursor}
```

function ReadCharAtCursor : Char;

```
{-Returns character at the current cursor location on the selected page.}
```

function ReadAttrAtCursor : Byte;

```
{-Returns attribute at the current cursor location on the selected page.}
```

procedure GetCursorState(var XY, ScanLines : Word);

```
{-Return the current position and size of the cursor}
```

procedure RestoreCursorState(XY, ScanLines : Word);

```
{-Reset the cursor to a position and size saved with GetCursorState}
```

procedure FastWriteWindow(St : string; Row, Col, Attr : Byte);

```
{-Write a string using window-relative coordinates}
```

procedure FastText(St : string; Row, Col : Byte);

```
{-Write St at Row,Col without changing the underlying video attribute.}
```

procedure FastTextWindow(St : string; Row, Col : Byte);

```
{-Write St at window Row,Col without changing the underlying video
attribute.}
```

procedure FastVert(St : string; Row, Col, Attr : Byte);

```
{-Write St vertically at Row,Col in Attr (video attribute)}
```

procedure FastVertWindow(St : string; Row, Col, Attr : Byte);

```
{-Write a string vertically using window-relative coordinates}
```

procedure FastFill(Number : Word; Ch : Char; Row, Col, Attr : Byte);

```
{-Fill Number chs at Row,Col in Attr (video attribute) without snow}
```

procedure FastFillWindow(Number : Word; Ch : Char; Row, Col, Attr : Byte);

```
{-Fill Number chs at window Row,Col in Attr (video attribute) without snow}
```

procedure FastCenter(St : string; Row, Attr : Byte);

```
{-Write St centered on window Row in Attr (video attribute) without snow}
```

procedure FastFlush(St : string; Row, Attr : Byte);

```
{-Write St flush right on window Row in Attr (video attribute) without snow}
```

procedure FastRead(Number, Row, Col : Byte; var St : string);

```
{-Read Number characters from the screen into St starting at Row,Col}
```

procedure FastReadWindow(Number, Row, Col : Byte; var St : string);

```
{-Read Number characters from the screen into St starting at window Row,Col}
```

procedure ReadAttribute(Number, Row, Col : Byte; var St : string);

```
{-Read Number attributes from the screen into St starting at Row,Col}
```

procedure ReadAttributeWindow(Number, Row, Col : Byte; var St : string);

```
{-Read Number attributes from the screen into St starting at window Row,Col}
```

procedure WriteAttribute(St : String; Row, Col : Byte);

```
{-Write string of attributes St at Row,Col without changing characters}
```

procedure WriteAttributeWindow(St : String; Row, Col : Byte);

```
{-Write string of attributes St at window Row,Col without changing
characters}
```

procedure ChangeAttribute(Number : Word; Row, Col, Attr : Byte);

```
{-Change Number video attributes to Attr starting at Row,Col}
```

procedure ChangeAttributeWindow(Number : Word; Row, Col, Attr : Byte);

```
{-Change Number video attributes to Attr starting at window Row,Col}
```

procedure MoveScreen(var Source, Dest; Length : Word);

```
{-Move Length words from Source to Dest without snow}
```

procedure FlexWrite(St : string; Row, Col : Byte; var FAttrs : FlexAttrs);

```
{-Write St at Row,Col with flexible color handling}
```

procedure FlexWriteWindow(St : string; Row, Col : Byte; var FAttrs : FlexAttrs);

```
{-Write a string flexibly using window-relative coordinates.}
```

function SaveWindow(XLow, YLow, XHigh, YHigh : Byte; Allocate : Boolean;

```
                  var Covers : Pointer) : Boolean;
{-Allocate buffer space if requested and save window contents}
```

procedure RestoreWindow(XLow, YLow, XHigh, YHigh : Byte;

```
                      Deallocate : Boolean; var Covers : Pointer);
{-Restore screen contents and deallocate buffer space if requested}
```

procedure StoreWindowCoordinates(var WC : WindowCoordinates);

```
{-Store the window coordinates for the active window}
```

procedure RestoreWindowCoordinates(WC : WindowCoordinates);

```
{-Restore previously saved window coordinates}
```

function PackWindow(XLow, YLow, XHigh, YHigh : Byte) : PackedWindowPtr;

```
{-Return a pointer to a packed window, or nil if not enough memory}
```

procedure DispPackedWindow(PWP : PackedWindowPtr);

```
{-Display the packed window pointed to by PWP}
```

procedure DispPackedWindowAt(PWP : PackedWindowPtr; Row, Col : Byte);

```
{-Display the packed window pointed to by PWP at Row,Col. If necessary,
 the coordinates are adjusted to allow it to fit on the screen.}
```

procedure MapPackedWindowColors(PWP : PackedWindowPtr);

```
{-Map the colors in a packed window for improved appearance on mono/B&W
 displays}
```

procedure DisposePackedWindow(var PWP : PackedWindowPtr);

```
{-Dispose of a packed window, setting PWP to nil on exit}
```

procedure WritePackedWindow(PWP : PackedWindowPtr; FName : string);

```
{-Store the packed window pointed to by PWP in FName}
```

function ReadPackedWindow(FName : string) : PackedWindowPtr;

```
{-Read the packed window stored in FName into memory}
```

function CreateLibrary(var F : file; Name : string;

```
                      Entries : Byte) : DirectoryPtr;
{-Create a library with the specified # of directory entries}
```

function OpenLibrary(var F : file; Name : string) : DirectoryPtr;

```
{-Open the specified library and return a pointer to its directory}
```

procedure CloseLibrary(var F : file; var DP : DirectoryPtr);

```
{-Close library F and deallocate its directory}
```

procedure PackLibrary(LName : string);

```
{-Pack a library to remove deleted entries.}
```

procedure AddWindowToLibrary(PWP : PackedWindowPtr; var F : file;

```
                             DP : DirectoryPtr; WinName : LibName);
{-Add a packed window to the specified library}
```

function ReadWindowFromLibrary(var F : file; DP : DirectoryPtr;

```
                               WinName : LibName) : PackedWindowPtr;
{-Read a packed window from a library}
```

procedure DeleteWindowFromLibrary(var F : file; DP : DirectoryPtr;

```
                                  WinName : LibName);
{-Delete a packed window from the specified library}
```

function MapColor(c : Byte) : Byte;

```
{-Map a video attribute for visibility on mono/bw displays}
```

procedure SetBlink(Status : Boolean);

```
{-Enable text mode attribute blinking if On is True}
```

procedure SetCrtBorder(Attr : Byte);

```
{-Set border to background color if card type and mode allow}
```

function Font8x8Selected : Boolean;

```
{-Return True if EGA or VGA is active and in 8x8 font}
```

procedure SelectFont8x8(Status : Boolean);

```
{-Toggle 8x8 font on or off}
```

function HercPresent : Boolean;

```
{-Return true if a Hercules graphics card is present}
```

procedure SwitchInColorCard(ColorOn : Boolean);

```
{-Activate or deactivate colors on a Hercules InColor card}
```

function HercGraphicsMode : Boolean;

```
{-Return True if a Hercules card is in graphics mode}
```

function HercModeTestWorks : Boolean;

```
{-Return True if HercGraphicsMode will work}
```

procedure SetHercMode(GraphMode : Boolean; GraphPage : Byte);

```
{-Set Hercules card to graphics mode or text mode, and activate specified
 graphics page (if switching to graphics mode).}
```

function ReadKeyWord : Word;

```
{-Waits for keypress, then returns scan and character codes together}
```

function CheckKbd(var KeyCode : Word) : Boolean;

```
{-Returns True (and the key codes) if a keystroke is waiting}
```

function KbdFlags : Byte;

```
{-Returns keyboard status flags as a bit-coded byte}
```

procedure StuffKey(W : Word);

```
{-Stuff one key into the keyboard buffer}
```

procedure StuffString(S : string);

```
{-Stuff the contents of S into the keyboard buffer}
```

procedure ReInitCrt;

```
{-Reinitialize CRT unit's internal variables. For TSR's or programs with
 DOS shells. May reset: CurrentMode, ScreenWidth, ScreenHeight,
 WindMin/WindMax, CurrentPage, CurrentDisplay, CheckSnow, and VideoSegment}
```

{$ifdef WIN32} procedure SetSafeCPSwitching(F: Boolean); procedure SetUseACP(F: Boolean); {$ENDIF}

procedure AssignConToCrt;

procedure ClrScr;

```
{-Clears the screen and returns the cursor to the upper-left corner}
```

procedure TextBackground(Color: Byte);

```
{-Selects the background color}
```

procedure TextColor(Color: Byte);

```
{-Selects the foreground character color}
```

procedure Window(X1,Y1,X2,Y2: Byte);

```
{-Defines a text window on the screen}
```

procedure GotoXY(X,Y: Byte);

```
{-Moves the cursor to the given coordinates within the screen}
```

function WhereX: Byte;

```
{-Returns the X coordinate of the current cursor location}
```

function WhereY: Byte;

```
{-Returns the Y coordinate of the current cursor location}
```

procedure ClrEol;

```
{-Clears all characters from the cursor position to the end of the line }
{ without moving the cursor.                                            }
```

function KeyPressed: Boolean;

```
{-Determines if a key has been pressed on the keyboard and returns True }
{ if a key has been pressed                                             }
```

function ReadKey: Char;

```
{-Reads a character from the keyboard and returns a character or an     }
```

```
{ extended scan code.                                                    }
```

procedure TextMode (Mode: word); procedure InsLine;

```
{-Inserts an empty line at the cursor position}
```

procedure DelLine;

```
{-Deletes the line containing the cursor}
```

procedure LowVideo;

```
{-Selects low intensity characters}
```

procedure HighVideo;

```
{-Selects high-intensity characters}
```

procedure NormVideo;

```
{-Selects normal intensity characters}
```

procedure Delay(MS: Word); procedure Sound(Hz: Word); procedure NoSound; procedure AssignCrt(var F: Text);

```
{-Associates a text file with CRT device.}
```

procedure PlaySound(Freq,Duration: Longint);

```
{-Setups window coordinates }
```

procedure GetLastMode;