

# ***OVERVIEW***

mouseLib is a Turbo (Borland) Pascal 6.0+ mouse support unit, designed to provide event driven mouse support, using a standard (default) event handling mechanism. The unit includes all of the normal mouse library functions, (show, hide cursor, define tresh-hold, detect mouse etc..).

mouseLib was used for over 3 years now, with many programs and products written by the author, and other programmers. Several mouseLib based packages are distributed by ISoft D&M, and can be downloaded for evaluation from BBSes around the world.

Related Topics : [Technical Documentation](#) [Contact](#) [Registration](#) [Credits](#)

# ***CONTACT***

Please contact :

ISoft D&M,  
P.O.B 5517  
Coralville IA 52241,  
U.S.A

To contact the author directly :

Contact :      Loewy Ron,  
                    9 Haneveem st.  
                    Herzeliya, 46465  
                    ISRAEL.

Related Topics : [Overview](#) [Registration](#)

# ***REGISTRATION***

mouseLib is a shareware product, if you find this product valuable, please register it. This section describes the reasons you should register.

By registering mouseLib you will receive the latest mouseLib version.

By registering you will help us to create the next version of mouseLib that will support more INT 33H functions, and will include even more high-level support for TSRs, EVENT-DRIVEN programming, and more.

Related Topics : [Contact](#) [Overview](#)

# ***TECHNICAL***

The mouse library package is built out of 3 different pascal units :

mouseLib - The core mouse support unit.  
tpDESQ - DESQview support unit.  
video - Video hardware support unit.

Related Topics : [Overview](#) [Tips](#)

# ***CREDITS***

Turbo Pascal, Borland Pascal are trademarks of Borland International.

Genius Mouse is a trademark of KUN YING ENTERPRISE CO.

Yaniv Golan is the one to blame for the "true vga cursor" feature, he was the one that triggered my interest in the implementation of this feature.

Dave Kirsch's MOU105 package was used as a reference for the development of the true vga cursor. I used Duncan Murdoch's port of this code to help me identify bugs in my own code. I would like however, to point that the code was written from scratch, and is (to my opinion) superior regarding things such as execution (cpu) and storage (memory) overhead.

Related Topics : [Overview](#)

# TIPS

- a.> use the setMouseGraph and resetMouseGraph before using the initMouse procedure, to choose if you want text or graphics mouse support.
- b.> look at the defaultHandler assembler routine, this is the heart of my event driven applications. (use setDefaultHandler to ...).
- c.> If you want to use True VGA text cursor (vgaTextGraphicCursor mode), call setVGATextGraphicCursor before initMouse, and be sure to call setDefaultHandler, or provide a handler that supports mouse movement the way the vga true cursor handler does.
- d.> Do not set or reset vgaTextGraphicCursor boolean flag by yourself, let the supplied set.. and reset.. procedures do that for you, this variable is supplied in the interface of the unit for browse (read) purpose only, if this variable is changed not through the suggested methods, unpredictable things can happen.
- e.> In true vga cursor mode, the default handler does not trigger an event when a "cursor changed location" condition occurs! - this is a design feature needed to support the author's wintext library, if your code needs to be notified of eventhappened when the mouse moves, you will have to change the supplied default handler, or provide one of your own.

Related Topics : [Overview](#) [Tech. Docs](#)

# ***TPDESQ***

tpDESQ is a Turbo-Pascal unit implementation written for TP6.0+, (I think it will work with TP5.0, and TP5.5 as well, but I never tried), to the DESQview API basic functions that allow the running program to figure out if DESQview is active, what the virtual text buffer is, and allows a program to run in a DESQview window. (not just as a full screen application after you Zoom the window).

Related Topics : [tpDESQ interface](#) [Overview](#) [Tech. Docs](#)

# ***TPDESQINT***

This is the tpDESQ unit interface :

```
const
  DESQviewActive : boolean = false; { true if running under DESQview }
  DESQviewMajor  : byte    = 0;
  DESQviewMinor  : byte    = 0;

procedure detectDESQview;
function getDESQviewTextBuffer : pointer;
procedure DESQviewApiCall(func : word);
procedure DESQviewPause;
procedure DESQviewBeginCritical;
procedure DESQviewEndCritical;
procedure makeDESQviewAware;
procedure DESQviewHercules43Lines(page : byte);
function  DESQviewCurrentWindow : byte;
function  DESQviewDirectScreenWrite : boolean;
function  extendedDESQview : boolean; { true if xdv }
```

For the mousselib unit only the detectDESQview function is needed, and it is called automatically by the inclusion of this unit in the uses clause of mousselib's interface section. The DESQviewActive boolean flag is used to decide if vgaTextGraphicCursor mode (True cursor) can be entered.

Related Topics : [tpDESQ Tech. Docs Overview](#)



# ***VIDEO***

Video is a support unit for hardware adapters. This unit is used in order to support the True vga cursor on vga screens.

This unit is used to define and support constants related to the video memory buffer, memory buffer size, memory adapter, font lines, and other general video statistics.

Related Topics : [Video Interface](#) [Tech. Docs](#) [Overview](#)

# VIDEOINT

This is the video unit interface :

```
type    fontSize = (font8,font14,font16, unknownFontSize);
adapterType = (none,mda,cga,egaMono,egaColor,vgaMono,vgaColor,mcgaMono,mcgaColor);
var      textBufferOrigin  : pointer; {pointer to text buffer}
          textBufferSeg     : word;
extBufferSize    : word;    {size in bytes of...}
isibX,visibleY : byte;
          fontLines         : byte;
const   maxX : integer = 79;
        axY  : integer = 24;

function queryAdapterType : adapterType;
function fontCode(h : byte) : fontSize; {convert from byte to enum}
function getFontSize : fontSize; {normal 25 lines,ega 25 lines,vga 25 lines}
function fontHeight(f : fontSize) : byte;
procedure getTextBufferStats(var BX      : byte; {visible x dimentions}
                             var BY      : byte; {visible y dimentions}
                             var buffSize : word {refresh buffer size}
                             );
```

The mousselib unit uses this unit in order to validate the existence of a vga color adapter, in order to support the true vga cursor in vgaTextGraphicCursor mode.

Related Topics : [Video Tech. Docs Overview](#)

# ***MOUSELIB***

The mousselib unit is the core of the mouse support in the mousselib package. This unit implement the mouse driver API support, as well as the extended services and support data structures and utilities.

The inclusion of this unit ensures the automatic initialization and detection of the mouse driver, and general mousselib unit variables.

If you want to use the mousselib unit in a cursor mode which is not the default text mode hardware cursor, use the setMouseGraph and setVgaTextGraphicCursor to use graphic or vga true cursor modes, followed by a call to initMouse. For the use of the true vga cursor a call to setDefaultHandler is recommended.

Related Topics : [Overview](#) [Tech. docs](#) [tpDESQ](#) [Video](#)  
[Constants](#) [Types](#) [Variables](#)  
[Functions](#) [Procedures](#)

# CONSTANTS

The following constants are defined in the mouselib unit's interface :

```
OUSEINT = $33; {mouse driver interrupt}  
  LEFTBUTTON = 1; {bit 0}  
  RIGHTBUTTON = 2; {bit 1}  
  MIDDLEBUTTON = 4; {bit 2}
```

```
URSOR_LOCATION_CHANGED = 1; {event mask bits}  
  LEFT_BUTTON_PRESSED = 2;  
  LEFT_BUTTON_RELEASED = 4;  
  RIGHT_BUTTON_PRESSED = 8;  
  RIGHT_BUTTON_RELEASED = 16;  
  MIDDLE_BUTTON_PRESSED = 32;  
  MIDDLE_BUTTON_RELEASED = 64;
```

```
click_repeat = 10; { Recommended value for waitForRelease timeOut }  
mouseTextScale = 8;
```

Related Topics : [MouseLib Tech. Docs Overview](#)

# VARIABLES

The following variables are defined in the mousselib unit :

```
mouse_present : boolean;
mouse_buttons : mouseType;
eventX,eventY,eventButtons : word; {any event handler should update}
eventhappened : Boolean; {these vars to use getLastEvent }
Motions,YMotions : word; {per 8 pixels}
mouseCursorLevel : integer;
if > 0 mouse cursor is visible, otherwise not, contains the level
of showMouseCursor/hideMouseCursor}
fontPoints : byte;
astMask : word = 0;
astHandler : pointer = Nil;
astCursor : grCursorType = (xH : 0; yH : 0; data : nil );
vgaTextGraphicCursor : boolean = false; { this is not the default .. }
```

Related Topics : [MouseLib](#) [Tech. Docs](#) [Overview](#)

# ***TYPES***

The following types are defined in the mousselib unit :

```
mouseType = (twoButton,threeButton,another);  
uttonState = (buttonDown,buttonUp);  
irection = (moveRight,moveLeft,moveUp,moveDown,noMove);  
rCursorType = record  
xH,yH : byte; {x,y Hot Point}  
data   : pointer; {cursor look pointer}  
nd;
```

Related Topics : [MouseLib](#) [Tech. Docs](#) [Overview](#)

# ***FUNCTIONS***

The following functions are defined in the mousselib unit :

```
function getMouseX : word;  
function getMouseY : word;  
function getButton(Button : Byte) : buttonState;  
function buttonPressed : boolean;  
function LastXPress(Button : Byte) : word;  
function LastYPress(Button : Byte) : word;  
function ButtonPresses(Button : Byte) : word; {from last last check}  
function LastXRelease(Button : Byte) : word;  
function LastYRelease(Button : Byte) : word;  
function ButtonReleases(Button : Byte) : word; {from last last check}  
function recentXmovement : direction;  
function recentYmovement : direction;  
function getMouseSaveStateSize : word;
```

Related Topics : [MouseLib](#) [Tech. Docs](#) [Overview](#)

# ***PROCEDURES***

```
procedure initMouse; {when replacing mouse mode do that..!}  
procedure showMouseCursor;  
procedure hideMouseCursor;  
procedure setMouseCursor(x,y : word);  
procedure mouseBox(left,top,right,bottom : word); {limit mouse rectangle}  
procedure graphicMouseCursor(xHotPoint,yHotPoint : byte; dataOfs : pointer);  
procedure HardwareTextCursor(fromLine,toLine : byte);  
procedure softwareTextCursor(screenMask,cursorMask : word);  
procedure setArrowCursor;  
procedure setWatchCursor;  
procedure setUpArrowCursor;  
procedure setLeftArrowCursor;  
procedure setCheckMarkCursor;  
procedure setPointingHandCursor;  
procedure setDiagonalCrossCursor;  
procedure setRectangularCrossCursor;  
procedure setHourGlassCursor;  
procedure setNewWatchCursor;  
procedure setEventHandler(mask : word; handler pointer);  
procedure setDefaultHandler(mask : word);  
procedure enableLightPenEmulation;  
procedure disableLightPenEmulation;  
procedure defineSensitivity(x,y : word);  
procedure setHideCursorBox(left,top,right,bottom : word);  
procedure defineDoubleSpeedTreshHold(treshHold : word);  
procedure disableTreshHold;  
procedure defaultTreshHold;  
procedure setMouseGraph;  
procedure resetMouseGraph;  
procedure waitForRelease(timeOut : word);  
procedure swapEventHandler(mask : word; handler : pointer); { return old in lastMask and lastHandler }  
procedure interceptMouse; { get mouse from interrupted program, and stop it .. }  
procedure restoreMouse;  
procedure setVgaTextGraphicCursor;  
procedure resetVgaTextGraphicCursor;
```

Related Topics : [MouseLib](#) [Tech. Docs](#) [Overview](#)



