WI99

Window Manager for

TURBO-PASC'99

Table of content

WI99, Window Manager for TP'99

1. Content of the disk.

WI99@	The Window Manager Library module that can be linked to a pascal program using LK99.
WI99DEMO	TP'99 program, which shows some possibilities of WI99. It is well documented and can help beginners with problems.
WI99DEMO-O	Object code of WI99DEMO. This file must be linked with WI99@.
WI99MASK	TP'99 Demo-Programm. Must also be loaded with WI99@.
MASKDOC_E	WI99MASK manual. Read this manual before starting WI99MASK.
MASKDOC_G	WI99MASK German manual.
WI99MASK1	Program image file from WI99MASK. Can be started with the RUN PROGRAM FILE option 5 of the Editor/ Assembler.
WI99PROC	External declarations of all WI99 procedure.

The standard pascal procedures: CLS, CURSOR, WRITE, WRITELN, READ and READLN can be used simultaneously with WI99 without any modification.

2. WI99 systeem procedures.

PROCEDURE W_init(VAR returncode :integer);

This procedure must be called before any other WI99 procedure can be called.

W_init ensures that a number of standard pascal I/O procedures are bypassed and thus enable window operation.

The WI99 internal data structures are initialized. The character codes 128..255 are the inverted codes of 0..127.

Attempting to initialize WI99 multiple times will result in an error message.

PROCEDURE W_exit(VAR returncode :integer);

Leave the WI99 environment. Turns the normal standard pascal I/O back on.

```
PROCEDURE W_open( window#    : integer;
                  headline   :string[40];
                  frametyp   :integer;
                  row, col   :integer;
                  hight, width:integer;
                  VAR retcode :integer);
```

Draws a window on the screen.

Window # is an integer between 0..255. This number must be used in every subsequent window procedure. Multiple windows may not be opened with the same number.

Headline is a line of text that is printed on the top line of the window. This makes it possible to quickly recognize a certain window.

Frame type is an integer that specifies the type of frame to use. You can choose from five different frames:

0 No frame.
1 Simple narrow frame.
2 Double narrow frame.
3 Simple wide frame.
4 Wide and narrow frame combined.

If frame type <0> is chosen, the size of the window indicates exactly the number of lines and columns. With the other frame types, two lines and two columns are lost.

Row and col indicate where the top left corner of the window will end up.

Row IN [1..24]

Col IN [1..40]

Height and width indicate the height and width of the defined window.

Height (type 0) IN [1..24]
(type 1,2,3,4) IN [3..24]

Width (type 0) IN [1..40]
(type 1,2,3,4) IN [3..40]

A window defined by W_open becomes active immediately.

PROCEDURE W_pop(window# :integer; VAR retcode : integer);

Makes a window with window # active. If several windows are defined, a specific window can be made active. The window is brought to the front at the same time.

PROCEDURE W_close(VAR retcode :integer);

The window with window # is removed. The window that is on top of the staple is then activated.

PROCEDURE W_move(row, col :integer; VAR retcode :integer);

Shifts the active window to the specified row and col.

PROCEDURE W_inv(VAR retcode : integer);

Inverts the active window.

PROCEDURE W_pos(VAR row, col height, width : integer; VAR retcode : integer);

Displays the row and col position of the active window as well as the width and height.

PROCEDURE W_activ(VAR window# :integer; VAR retcode : integer);

Returns the active windown number.

PROCEDURE W_get(number_of_bytes :integer; VAR text_buffer :string[255];
VAR retcode :integer);

Allows to read some characters from the active window. Reading starts from the current cursor position, which is then naturally adjusted. Number_of_bytes can be up to 255. If a window is smaller than the desired number of bytes to be read, a smaller string is of course returned.

PROCEDURE W_match(row, col : integer; VAR match :boolean; VAR window# :integer;
VAR retcode :integer);

Match = TRUE if a window is found.

Match = FALSE if row and col are not inside a window.

Window # is the windown number of the found window.

Example with mouse control:

```
REPEAT
  REPEAT UNTIL ClcMs;
  CrsMs(row,col);
  W_match(row,col,match,wn#,err);
UNTIL match;
W_pop(wn#,err);
```

Note: Routines ClcMs and CrsMs must be taken from the WiPoMouse module.

FUNCTION S_inv(text : string[255]) :string[255];

Inverts all characters in text.

FUNCTION lastky : integer;

Provides the code of any key that closed the last read or readln.

If WI99 is not initialized and / or no read / readln has been performed then lastky will provide an undefined character.

3. Changes to the redirected routines.

GRAPHICS: The invocation of this procedure is ignored. WI99 can only be used in text mode.

TEXT: The call is not executed because at startup the pascal system is already in text mode.

CLS: This procedure only clears the active window.

SCREEN: Works as usual. It is possible that the colors have to be set again after W_init because W_init changes various buffers of RUNLIB.

CURSOR: Works as usual. Since windows can take different sizes, you have to be aware that the specified position for the cursor is within the active window, otherwise a runtime error will occur immediately.

WRITE: As usual.

WRITELN: As usual.

READ: As usual. It can happen that in a window there is not enough space to contain all the characters that must be entered. The other characters must then be typed in blindly.

An entry can be closed with <enter> or with <fctn 6> (code 12), <fctn 7> (code 1), <fctn 8> (code 6), <fctn 9> (code 15), <fctn E > (code 11), or <fctn X> (code 10). With the help of lastky you can then test with which key the READ has been closed.

READLN: See read.

Appendix A.

Error code	Meaning.
0	Not an error.
1	WI99 not initialized.
2	WI99 initialized several times.
3	more than 20 open windows.
4	wrong window number
5	no window opened with used window number in procedure call.
6	window number used several times during W_open.
7	wrong frame number
8	wrong row, col, height or width.
9	wrong parameter.
10	too little memory. (Too many large windows defined)