

DebugDelphi User Guide

(Release 2.3)

Copyright Helmuth J.H. Adolph 2001 / 2002

The Debug Terminal for Delphi 2 .. 7

Purpose

The purpose of DebugDelphi is to create a possibility to display debugging information of an application in a debugging window. This should be done with a minimum of effort.

How To

Producing output in DebugDelphi is done in three steps:

- Insert a Uses-statement for the unit 'DebugInterface' into the Unit you want to output debug text
- Insert WriteLn - statements anywhere you need debugging output.
- Start DebugDelphi

That's all.

Date: 01/02/2003

Contents of this description

- A. Setup process
- B. Simple Usage - using the main channel
- C. Advanced usage - using the minor channels
- D. Using DebugDelphi in multi threaded applications
- E. Defining auxiliary output files
- F. Using Write and WriteLn
- G. Special Functions
- H. Limitation of use
- I. Upgrading to professional mode
- J. Built-in functions
- K. History

A. Setup process

The setup program copies the necessary unit 'DebugInterface' into your Delphi library directory and the DLL with the needed functions into the Windows system directory. DebugDelphi itself and the setup program into the installation directory. Also it enters DebugDelphi into the Delphi tools menu and into the windows start menu. For uninstalling start the setup program in the installation directory, it will automatically delete all files it has copied before.

DebugDelphi can be used with or without the Delphi IDE. Output of Debug information is very fast, every information transferred from the testee to DebugDelphi is buffered in a shared memory. If the information is not processed yet, a message is posted to DebugDelphi to start output. If the buffersize (64 entries) would be exceeded, DebugDelphi is forced to process the buffer, the application waits until done so. Normally DebugDelphi works when the testee is waiting for a message, so that the testee is not slowed down because of displaying error messages.

If automatic start is checked when installing, the application that uses DebugDelphi automatically starts DebugDelphi at initialization. This feature can also be activated later on in DebugDelphi itself.

B. Simple Usage - using the main channel

When a Uses-statement for the Unit DebugInterface is inserted into the DPR-file of the project, output with the WriteLn-statement can be produced if DebugDelphi is started (e.g. by the Delphi tools menu).

Example:

```
WriteLn('This is a debug output on the standard output device (main channel) with DebugDelphi');
```

Output without naming a file-variable uses the standard output device. This device is redefined by the unit DebugInterface and is the **Main Channel**.

Output in DebugDelphi's window would be:

```
-M- [date ][time ]This is a debug output on the standard output device (main channel) with DebugDelphi
```

-M- stands for main channel.

If date and time are displayed depends on the options set in DebugDelphi.

If DebugDelphi is not started, there is no output, also no error message. By starting and ending DebugDelphi you decide, if you want to see error messages or not.

C. Advanced usage - using the minor channels

By using 'DebugInterface' in your application, automatically the standard output device for your application is redefined. Also 10 extra channels, **Minor Channels**, are automatically defined. The extra channels can be used by naming one of the defined file variables DBChannel in the WriteLn - statement.

Example:

```
WriteLn(DBChannel[1], 'This is an output on channel 1 (minor channel) with DebugDelphi');
```

Output in DebugDelphi's window would be:

```
-1- [date ][time ]This is an output on channel 1 (minor channel) with DebugDelphi
```

The unit DebugInterface automatically creates 10 minor channels to DebugDelphi. These channels can be used for different purposes. Every channel can be switched on and off separately on-line. If you want to use minor channels, you need to place the 'USES DebugInterface' statement into every unit in which you want to display something in the DebugDelphi window.

A good idea would be to build error classes in your application. Depending on the settings in DebugDelphi you can

display these classes or not.

In Freeware mode only minor channels 0, 1 and 2 can be used, in Professional mode all channels. Using channels 3 to 9 in Freeware mode causes I/O- errors !!!

D. Using DebugDelphi in multi threaded applications

All output done with WriteLn is done by means of the Delphi text-file device driver. This driver uses file variables for the output. As every other data too, these variables have to be protected against simultaneous access by different threads. To do this, you have different possibilities, e.g. use critical sections or mutexes. Critical sections afford an EnterCriticalSection call before the WriteLn and a LeaveCriticalSection call after the WriteLn and is a lot of coding effort.

Easier ways are:

- To use one of the standard minor channels per thread.
- To define auxiliary file variables as described in the next chapter.

E. Defining auxiliary output files

DebugInterface offers calls to open and close additional files for your output with WriteLn. Of course the output is not stored in a file but transferred to DebugDelphi.

```
PROCEDURE OpenAuxDebugOutput( VAR AuxChannel : Text; ChannelNo : Integer);  
  
PROCEDURE CloseAuxDebugOutput(VAR AuxChannel : Text );  
  
// ChannelNo must have a value between 0 and 9. (0 = Main channel / 1..9 = Minor channels)  
// - In freeware mode auxiliary channels can not be used !
```

Example:

```
VAR  
  MyOutput : Text;  
BEGIN  
  OpenAuxDebugOutput(MyOutput, 5); // Open file and use channel number 5  
  WriteLn(MyChannel, 'Write something on MyChannel using channel 5');  
  CloseAuxDebugOutput(MyOutput);
```

By this way you can open as much file variables as you like (or as Windows accepts).

A good idea would be to let every thread use an own file variable and use the standard channels in the Main Thread.

The auxiliary file variables can only be used in professional mode. Using auxiliary file variables in the freeware mode causes lines not to be displayed, range check or I/O- errors !!!

F. Using Write and WriteLn

Instead of producing a line in DebugDelphi by a single WriteLn statement you also can use several Write - statement plus a WriteLn - statement. The output in DebugDelphi appears after the WriteLn - statement.

The total text length may not exceed 255 characters.

G. Special Functions

- To insert an empty line into the debugging window just use a WriteLn statement without an output string, e.g.

```
WriteLn;                                or  
WriteLn(DBChannel[1]);
```

- To delete all lines of the debugging window use a WriteLn statement with a form feed character, e.g.

```
WriteLn(#12);                           or  
WriteLn(Chr(12));                       or  
WriteLn(DBChannel[1], #12);
```

G. Limitation of use

- Length of the strings to be displayed: max. 255 characters (DebugDelphi adds channel / Date / Time)
- Maximum of stored and displayed lines in DebugDelphi: 32000
- In FreewareMode only the main channel and two minor channels are usable.
- In FreewareMode auxiliary file variables can not be used.

H. Upgrading to professional mode

Upgrading to professional mode is done by registering DebugDelphi. You can order a registration key via ShareIt registration service (<http://www.shareit.com>). Together with the registration key you'll get a special version of the included DebugDel.DLL.

After entering the registration information and copying DebugDel.DLL to the Windows system directory (drive:\windows\system for Win 9x/Me and drive:\windows\system32 for Win NT/2000/XP) the minor channels 3 .. 9 are unlocked. Also it is possible to use auxiliary file variables for output.

I. Built-in functions

DebugDelphi has a lot of possibilities:

- Saving all lines to a file,
- Copying, deleting or printing selected lines (copying to clipboard),
- Deleting all lines,
- Search of text,
- Switch date and time on and off,
- Keep the window on top,
- Set automatic scrolling on and off,
- Select a display font and
- Set the window colour.

J. History

Version 1.0 : 9/01	First release
Version 1.3 : 3/02	Automatic start of applications using DebugDelphi's API
Version 1.4 : 4/02	Optional start of DebugDelphi with a minimized window
Version 1.5 : 5/02	Bugfix: Strings with more than 127 characters caused range check error
Version 2.0 : 7/02	Adaption to Delphi 7
Version 2.1 : 9/07	Bugfix
Version 2.2 : 11/02	Inserting Empty lines in the debugging window / deleting the lines in the debugging window
Version 2.3 : 1/03	Minor channel 0 added, font problem when executed on a Windows system with large fonts solved.