Debugger Video DLL Configuration Utility (TDWINI.EXE)

TDWINI.EXE

This utility allows you to change the current Video DLL and any of its options quickly and easily, without having to edit the TDW.INI file. Upon startup, TDWINI.EXE searches through your path to check for multiple TDW.INI files. If any additional copies are found, it asks whether you want to delete or rename them. If a copy is found in another directory and not in the Windows directory, it asks if you would like it moved to the Windows directory. It is very important that **EXACTLY ONE** copy of TDW.INI exists and it's in the Windows directory. If you have more than one copy, TDW, TD32, and TDWINI could end up finding and using different copies (which could prevent the video options from takeing effect). If a copy of TDW.INI is marked READONLY or is on a Network or CDROM drive, TDWINI.EXE may not be ableto delete, rename, or copy it and it will then caution you and allow you to leave the extra INI file. If a TDW.INI file is not found anywhere along the path, one is created for you in your Windows directory, with SVGA.DLL as the default video DLL.

Once TDWINI.EXE determines that you have **EXACTLY ONE** copy of TDW.INI and it's in the Windows directory, the main dialog comes up. An attempt at finding your BC4\BIN directory is made and displayed in the Path box. Any valid Video DLLs located in that directory are then displayed in the list box below. The line above the list box shows where the current Video DLL will be loaded from. The line below the list box displays the description of the highlighted DLL. If you want to use a DLL in a different directory, change the Path box and press the **Refresh** button. To select a particular DLL either double-click on that DLL in the list box, or highlight the DLL and press the Set button. To set any DLL specific options, press the **Settings** button, which will take you to an options screen for the currently set DLL. This option screen is unique for each Video DLL.

The TEST button allows you to test the DLL that is currently set.

See also, the Video DLLs:

DUAL8514

STB

SVGA

TDWGUI

Video DLL TEST Button

The **TEST** button on the main TDWINI.EXE dialog allows you to test the DLL that is currently set. It brings up another dialog box that allows you to continue or cancel the test. If you press the **OK** button, the current Video DLL is loaded as if TDW was running, there is a system beep, and the screen is switched to a simulated debugger screen. Four seconds later, the system will beep again, the DLL will be unloaded, and the Windows screen should once again appear. If the debugger screen doesn't appear, or the Windows screen doesn't re-appear, then press <ALT-X>, GDI will be jump-started and the Windows screen should return to normal. If all else fails, at this point, pressing <Shift-Alt-X> will exit and restart Windows (this should rarely be necessary).

If the DLL you chose doesn't work with your video card (in other words, you should have seen the debugger screen and then four seconds later have been back in an uncorrupted Windows screen, otherwise that DLL doesn't work with your card) then pick a different one from the list. If you don't know which one to pick, try either TDWGUI.DLL or SVGA.DLL.

DUAL8514.DLL

General Information

This DLL should only be used with systems that have two monitors attached. One monitor for the 8514 card, and the other connected to the regular VGA card. (All 8514 cards require or have built in, a standard VGA card but not all 8514 systems allow you to connect a separate monitor to the VGA port.) The results are that Windows will appear on the 8514 screen and the Debugger will appear on the VGA screen.

TDW.INI options

The options that DUAL8514.DLL uses are: (* denotes the default selection)

DebugFile=filename	filename = path and name of logging file.
ForceFileSave=Y|N	Y = Forces the log file to be written after each line. N*= opens then file when the DLL is loaded, and closes it when unloaded.
RestoreTextScreen=Y|N|C	Y*= Restore the text screen that TDW was on. N = Don't restore, leave the TDW image there. C = Clear the screen.

Options Screen

The three INI file options are also available from the Options Screen via TDWINI.EXE

STB.DLL

General Information

This DLL is for use with the STB MVP-2 & MVP-4 video cards. These cards have 2 or 4 Tseng ET-4000 VGA "ports" on them. This DLL allows you to have Windows displayed on one (or more) of the screens and the Debugger on a different one.

TDW.INI options

The options that STB.DLL uses are: (* denotes the default selection)

DebugFile=filename	filename = path and name of logging file.
ForceFileSave=Y|N	Y = Forces the log file to be written after each line. N*= Opens then file when the DLL is loaded, and closes it when unloaded.
RestoreTextScreen=Y|N|C	Y*= Restore the text screen that TDW was on. N = Don't restore, leave the TDW image there. C = Clear the screen.
FlipScreen=Y|N	Y = Re-draw TDW's text screen each time N*= Don't. (much smoother)
ActiveScreen=1..8	Selects the MVP port that TDW is displayed on. Windows will always be displayed on the Primary or #1 port. For an MVP-2 it must be set to 2 (the default if it's not specified). For an MVP-4, you can select 2-4. With two MVP-4's in the system, 1-4 are the on the first card, 5-8 are on the second.

Options Screen

The five INI file options are also available from the Options Screen via TDWINI.EXE

SVGA.DLL

General Information

This DLL works with all non-coprocessed and some coprocessed video cards. It provides a full screen text-mode session for the Debugger to be displayed in. It does not work with TIGA cards (use TDWGUI.DLL for those cards). This DLL uses the BitBlt API function to save the and restore the Windows screen as well as the documented DEATH and RESURRECTION functions in GDI to perform the switch between graphics and text mode. Because of this, the performance accuracy of this DLL is dependent upon the current Windows display driver. The latest Windows display drivers for most video cards are available from the Microsoft Software Library which is accessible from CompuServe (Go MSL-1 and download the file `WDL.TXT` for the list of available drivers). Some video drivers (Tseng, ATI non co-processed) DO NOT support DEATH and RESURRECTION under intense memory conditions, if this is the case please set Documented=Y.

TDW.INI options

The options that SVGA.DLL uses are: (* denotes the default selection)

DebugFile=filename	filename = path and name of logging file.
ForceFileSave=Y|N	Y = Forces the log file to be written after each line. N*= opens then file when the DLL is loaded, and closes it when unloaded.
BitBlt=Y|N	Y*= Enables BitBlt screen saving (use this if you want to see the user screen when pressing Alt-F5). N = Disables the screen saving (improves speed)
ForceRepaint=Y|N	Y = Forces Windows to repaint the desktop each time the user program is switched to. N*= Doesn't.
Mono=Y|N	Y = Runs TD32 on monochrome screen. N*= Runs TD32 on main screen. (TDW ignores this option. TD32 does not use the -do switch.)
VideoMode=xx	xx= The decimal value of the text-video mode to use for the debugger screen. default is 3. Some special hardware needs to be forced into monochrome video mode (7).
TextLines=xx	xx= Specifies the number of lines to use in 43/50-line-mode. This number can be: 25, 28, 43, or *50. The appropriate VGA text mode will be entered. If the number is not one of the listed sizes, then it defaults to 50.
XGA=Y|N	Y = Forces compliance on XGA systems N*= All other cards. (Some obscure cards will work better with this switch enabled).
EGA=Y|N	Y = Forces the 43/50 line mode to work correctly on EGA systems. N*= Uses VGA modes and the number of lines specified in the TextLines variable (see TextLines above).
RestartGDI=Y|N	Y = Resets GDI when the debugger exits. This should be used if the Windows palette or mouse becomes corrupted (usually on S3 cards) N*= Doesn't
Documented=Y|N	Y = Forces documented video mode switch calls between Text & Graphics modes. N*= Uses the DDK documented Death and Resurrect API calls.

(Some video drivers do not support Death & Ressurect correctly).

Options Screen

The seven INI file options are also available from the Options Screen via TDWINI.EXE

TDWGUI.DLL

General Information

This DLL works on ALL video systems, since it uses GDI to graphically display the debugger in a simulated window on the Windows desktop. If your are unsure of what DLL to use, try this one first. TDWGUI can support a wide range of fonts for displaying the Debugger screen. Since the Debugger thinks it's displaying on a text mode screen, the fonts must be fixed width, contain the OEM characters set (meaning the upper 128 line-drawing characters), and they must not be True Type fonts. The TDWINI.EXE utility can be used to choose from the available fonts, or you can modify the TDW.INI file options directly. Because there is no exact way to specify a particular font size (it's possible for two different fonts to have the same height, width and point size, but appear differently on the screen) TDWGUI uses an index method, that chooses the Xth font from the current list of fonts. The list of fonts that you choose from are available in the Debug File (if you enabled the LogEnums option) or from the Font... button in the TDWGUI options screen. The font can be seen in the simulated debugger screen that comes up from the Position... button in the TDWGUI options screen.

TDW.INI options

The options that SVGA.DLL uses are: (* denotes the default selection)

DebugFile=filename	filename = path and name of logging file.
ForceFileSave=Y|N	Y = Forces the log file to be written after each line. N*= opens then file when the DLL is loaded, and closes it when unloaded.
Rows=xx	xx= 25, or 50. Specifies the initial mode for TDW to start up in. This option only determines the mode (25 or 50), NOT the number of lines (see MaxLines to specify the number of lines you want to use in "43\50" line mode). This option is for compatibility with the 3.0 and 3.1 versions of TDW. (Default = 25)
FontSize=xx	xx= requested point size for the font. (default=9)
FontName=ss	ss= a string specifying the name of the requested font. The font must be fixed width and use the OEM character set. TDWGUI.DLL calls CreateFont with this font name after matching the selected point size with the fonts available from EnumFonts. If it does not find a usable font with these attributes, it defaults back to the TERMINAL font. i.e.: FontName=TERMINAL The matching process goes like this; for all the fonts in the specified font name: 1. Look for fixed pitch & OEM char set 2. Then for an exact point match 3. Else look for nearest point match 4. Else use OEM_FIXED_FONT Nearest point match is defined as the first font that is the closest in points to the requested point size. Use the LogEnums=1 switch to see the list of fonts available to TDWGUI (or use the Font... button in the Options Screen from TDWINI.EXE). If more than one font matches the selected point size, the first one will be selected. In order to select one of the other fonts that also match, an index override switch is provided. The EnumIndex=xx switch can be set to the index number that is listed in the 'Enumerated Fonts List' in the

	log file. This will then be the exact font selected (assuming that the specified index exists in the list). True Type fonts cannot be used and are not listed in the font list. They can be forced to be listed with the LogTTs=Y switch, even though they cannot be indexed, matched, or used by TDWGUI.
LogEnums=Y|N	Y = Logs the available fonts to the Debug File. N*= Doesn't.
LogTTs=Y|N	Also logs the True Type fonts in the Debug File (NOTE: True Type fonts can NOT be specified or used with TDWGUI) (Default = N).
EnumIndex=xx	xx= Index of font to use from the font list in the Debug File. (see LogEnums and FontName above) (Default = -1 which disables this feature)
TimerRez=xx	xx= number of milliseconds between updates of the TDW screen. (Default = 55, range = 55 - 5000) TD32 does not use the timer.
MaxLines=xx	xx= the number of lines to use when TDW goes into 43/50 line mode. anything over 100 is set back to 100. (See also Rows) (Default = 50)
PosX=xx	xx= horizontal position of TDW "window". -1 to center. (Default = -1)
PosY=xx	xx= vertical position of TDW "window". -1 to center. (Default = -1)
BorderWidth=xx	xx= the width in pixels for the 3D border. if a value less than 6 is specified, then 6 is used for the 3D border. (default = -1)
RestartGDI=Y|N	Y = Resets GDI when the debugger exits. This should be used if the Windows palette or mouse becomes corrupted (usually on S3 cards) N*= Doesn't

Options Screen

The most commonly used INI file options are available from the Options Screen via TDWINI.EXE. There are also two buttons on the Options Screen: Position... and Font...:

Font...

This button brings up a dialog that allows the current TDWGUI font and point size to be selected. Most of the usable (fixed, OEM & non True Type) fonts will be the TERMINAL fonts in DOSAPP.FON. These are the same ones that are used with "windowed" DOS boxes. You can choose from any of the fonts in the list box, or add your own font in the Font File box and enter the font name in the Face Name box. Press the **Refresh** button and any compatible fonts will be entered into the list box. The field called **Req.Pnt.Size** has the point size number that will be matched the closest if there is no **Enum Index** set (this is simply the FontSize= value).

Position...

This button brings up a simulated Debugger window using the current font and screen coordinates that the real Debugger will use. This allows you to position the window anywhere on the screen (simply left-click anywhere in the window and drag). A pop-up menu appears when you right-click anywhere in the window, from this menu you can:

Toggle the screen between 25 lines and the current value of MaxLines.

Center the window (sets PosX and PosY to -1).

Get to this help file.

Revert to the positions previous value) and exit back to the Options Screen.

Save the current position and exit back to the Options Screen.

The centering command will set the position values to -1, requesting the Debugger window to remain centered even if the screen resolution changes in the future.

Keyboard Interface:

<Arrow Keys>	Moves the window one pixel.
<SHIFT><Arrows Keys>	Moves the window ten pixels.
<Enter>	Save & exit.
<Escape>	Revert & exit.
<Home>	Center.
<End>	Save & exit (same as **<Enter>**).
<Alt><Space>	Brings up the pop-up menu.

Introduction to the TDW Video DLLs

Turbo Debugger for Windows comes in two flavors: TDW.EXE for debugging 16-bit Windows 3.x applications, and TD32.EXE for debugging 32-bit NT and Win32s applications. When running TD32 on an NT system, all the screen I/O is handled by NT via an NT console window. When running either TD32 or TDW under Windows 3.x or Win32s, the debuggers attempt to switch the video system into full-screen text-mode. In order to keep up with the ever changing video card market, the job of performing the video mode switch has been factored out of the debuggers and placed in the hands of separate DLLs (dynamic link libraries). We provide various DLLs to support the different types of video systems currently available.

One of the main reasons that TDW and TD32 use a text-based user interface instead of the Windows "windowing system" is so that they don't interfere with the application being debugged. If one of the bugs in the program you're debugging has the effect of corrupting the windowing system, a debugger that uses that windowing system may also be effected. By using a text-based user interface we attempt to provide more stability when debugging in an unstable environment.

The DLLs that we provide are:

- DUAL8514
- STB
- SVGA
- TDWGUI

The debugger looks at the file called TDW.INI in the Windows directory to find the correct Video DLL to load:

```
[TurboDebugger]
VideoDLL=c:\bc4\bin\svga.dll
```

Once the DLL loads, it may also look in the same TDW.INI file for options specific to that DLL:

```
[VideoOptions]
DebugFile=c:\bc4\bin\video.log
ForceFileSave=No
```

Select the previous hot-links for an explanation of each DLLs TDW.INI file options.

If you are in doubt of which Video DLL you should use, try TDWGUI.DLL, since it is compatible with all video cards.

Diagnosing Video Problems:

If you have any trouble installing the Video DLLs or if your Windows screen still gets corrupted, here are a few things to try:

1. Run the TDWINI.EXE utility. This will automatically check for missing or multiple TDW.INI files. When it comes up, check the path and name of the DLL that it says is currently installed.
2. There is also a TEST button that will attempt to use the current DLL as if the debugger was calling it. If you can see the screen correctly here, it will work with the debugger too. If it doesn't appear to work correctly in this test screen, then you should either check this DLL's local options (via the **Settings** button) or select a different DLL.
3. Select the **Settings** button for that DLL and make sure that you point the DebugFile= option to the full path and file name of some text file that you will examine after trying to run the Debugger again. Remember, if the debugger simply doesn't display on the screen and all you end up with is an hourglass mouse cursor, try hitting <Escape> followed by <Alt><X> to get back out.
4. If no debug file exists, then the DLL didn't get loaded (see # 1). Otherwise, examine the log file to see that the log date and time is when you just ran the debugger. Look at the DLLs full path listed at the top of the log file to make sure you are using the correct DLL.
5. If it still doesn't work, and you've tried both SVGA.DLL and TDWGUI.DLL, then you may need to contact our Technical Support department. First, you should try looking for any new DLL updates either on CompuServe (Go Borland) or on our local BBS (408)-431-5096 (9600 bps, 8N1). From time to time, when new video card technology emerges we provide updated or additional video DLLs that are available via our various On-line Systems.
6. If you still need help installing or configuring the Debugger, you can contact Borland Technical Support at 408-461-9133 from 6:00am to 5:00pm PST. Please have the video DLL's log file available when the support person answers the phone. Also be prepared to run the Microsoft Diagnostic program (MSD.EXE) that is in your Windows directory (exit Windows before running it, for the most complete results). MSD will give various reports on your system configuration (i.e.: video card, memory...)

Advanced Information:

If you have a special application where a custom video DLL is required, here is a brief overview of the functions required to create a TDW video DLL:

Exported Functions:

```
WORD FAR PASCAL _export VideoInit (void);
```

Called when TDW first loads up. All dynamic allocation and chip/video mode detection should be performed here.

Return codes are:

- 0 - success
- 1 - Incorrect video card was detected
- 2 - Unsupported video mode of correct card was detected
- 3 - Could not allocate the memory needed from Windows
- 4 - Regular video mode detected (VIDEO.DLL not required)
- 5 - Misc. error

Any failure will cause TDW to unload the video DLL and exit.

```
WORD FAR PASCAL _export VideoDone (void);
```

Called when TDW exits back to Windows. All memory allocated must be freed by the end of this function (Don't rely on LibMain and WEP). 1 means success, 0 means it failed.

```
WORD FAR PASCAL _export VideoGetTextSelector (int display);
```

Called when TDW needs the selector (protected mode segment) value of the text mode screen requested. If display is 0, you need to return the selector for the address 0xB800 (color). If display is 1, you need to return the selector for the address 0xB000 (mono). This can be done with the Windows pre-defined selectors: _B800H and _B000H. In 'C' you need to take its address after it is externed properly.

```
void FAR PASCAL _export VideoSetCursor (WORD x, WORD y);
```

Called when TDW needs to set the cursor position on the text mode screen. TDW will call this function when it needs to make the cursor disappear (by placing it at a non-displayable position). Historically, TDW 3.x has not called this function, and has set the cursor position itself (directly accessing the VGA cursor registers). In 4.0, however, both TDW and TD32 do call it.

```
void FAR PASCAL _export VideoDebuggerScreen (void);
```

Called when TDW wants to switch to the text mode screen. This function must save the appropriate memory locations, save the VGA palette, and switch to text mode.

```
void FAR PASCAL _export VideoWindowsScreen (void);
```

Called when TDW wants to switch back to the Windows screen. This function must switch back to the original graphics mode, restore the palette, and restore the SuperVGA graphics memory planes that were blown away by text mode. (This will usually be 4 to 8 K in planes 0 and 1 [for the text mode characters and attributes], and 32K in plane 2 [for the character generator data]).

```
WORD FAR PASCAL _export VideoBigSize (void);
```

Called when TDW needs to determine if there is a higher resolution text mode available usually 43 or 50 lines). The maximum number of lines that you are able to support should be returned here.

```
WORD FAR PASCAL _export VideoSetUp (WORD xtra1, WORD xtra2);
```

This function is reserved for future expansion.

```
void FAR PASCAL _export VideoSetSize (WORD bigflag);
```

Called when TDW wants to switch the resolution of the text mode screen. Bigflag will be 1 if high resolution (usually 43 or 50 lines), 0 if low resolution (25 lines).

```
WORD FAR PASCAL _export VideoIsColor (void);
```

Returns 1 for color, and 0 for monochrome. Basically it just passes back the logically inverted value of the 'display' parameter from the `VideoGetTextSelector()` function.

```
void FAR PASCAL _export VideoUpdateWindow (void);
```

This function is only called from TD32, and it is used to tell the Video DLL (mainly TDWGUI.DLL) that something on the screen has changed and it should be updated.

```
void FAR PASCAL _export VideoConfig (HWND hWnd, HINSTANCE hInst, char
  *HelpFile);
```

This function is never called by the Debuggers, but is used instead by TDWINI.EXE which calls it to make the Options Screen appear when the **Settings** button is pressed. hWnd is the parent window handle, hInst is the instance of the calling program, and HelpFile, is the name (path is optional) of the Windows .HLP file that the DLL should use for its help requests. This should be a self-contained function that simply brings up a dialog box that allows you to edit any of the TDW.INI options that it uses.

For TDWINI.EXE to detect the video DLLs, all of the preceding functions should be listed in the .DEF file with the ordinals listed below. This is also so that they appear in the Non-Resident Names table of the DLLs.

Example .DEF file

```
LIBRARY         TDVIDEO
DESCRIPTION     'TDW video driver for ...'
EXETYPE         WINDOWS
CODE            PRELOAD FIXED
DATA            PRELOAD FIXED
HEAPSIZE        8196
EXPORTS
                WEP                             @1
                VIDEODONE                       @2
                VIDEOINIT                       @3
                VIDEOGETTEXTSELECTOR            @4
                VIDEODEBUGGERSCREEN             @5
                VIDEOWINDOWSSCREEN              @6
                VIDEOSETCURSOR                  @7
                VIDEOSETSIZE                    @8
                VIDEOBIGSIZE                    @9
                VIDEOISCOLOR                    @10
                VIDEOUPDATEWINDOW               @11
                VIDEOSETUP                      @12
                VIDEOCONFIG                     @20
```

Source code for a sample DLL is available on CompuServe (Go Borland) and our local BBS (408)-431-5096.

Table of Contents:

Version: 4.02