

HDL Turbo Writer



User's Guide

Version 6.6a
October 2001

Publication History

November 1993	First Published.
March 1994	Version 1.4 Revision.
October 1994	Version 2.0a Complete revision.
November 1994	Version 2.0e Production and Installation enhancements.
February 1995	Version 2.0h Numerous features added.
March 1995	Version 3.1. Upgraded in line with Codewright 3.1
April 1995	Version 3.1a Extra templates added in line with Codewright 3.1a
November 1995	Version 3.1a Troubleshooting the MTI V-System integration section added.
April 1996	Version 4.0a Windows NT and Windows 95 version completed.
June 1996	Version 4.0c various Windows 95 specific problems fixed. Version 3.1a of Turbo Writer added to the distribution CD's to support Windows 3.1 users.
December 1996	Version 4.0e various minor bugs fixed. 93 templates added.
June 1997	Version 5.0b added outlining and version 5.0b features
October 1997	Version 5.0c fixed various testbench generation bugs.
April 1998	Version 5.0d build 100, added Renoir integration, fixed a problem with the Verilog testbench generator.
April 1999	Version 5.1c build 2. Added menu picks for component creation and instantiation, and fixed bug concerning colour coding of single line verilog comments. Incorporated CodeWright 5.1c.
January 2000	Version 6.0b. Incorporated CodeWright 6.0b. Copy protection changed from proprietary dongles to FlexLM.
October 2000	Version 6.0e. Bug fixes to the Verilog testbench generator, the VHDL error parser, and incorporate the latest CodeWright patches.
October 2001	Version 6.6a. Bug fix to Verilog testbench generator. Incorporated the latest Codewright.

Trademarks

Codewright™ is a registered trademark of Premia Corporation.

Windows™, Windows 95™ and Windows NT™ are registered trademarks of Microsoft Corporation.

ModelSim™ is a registered trademark of Model Technology Corporation.

Renoir™ is a registered trademark of Mentor Graphics Corporation.

Saros Technology Ltd.

The Spirella Building,

Bridge Road,

Letchworth,

Herts.

SG6 4ET

England

Telephone number ++44 (0)1462 476111

Fax number ++44 (0)1462 476112

Support email address support@saros.co.uk

Sales enquiries email address sales@saros.co.uk

Table of Contents

1. INTRODUCTION	1
ABOUT THIS MANUAL	1
ASSUMPTIONS	1
CONVENTIONS	1
SYSTEM REQUIREMENTS	3
INSTALLING TURBO WRITER	3
EVALUATION PERIOD	4
ADDITIONAL NOTES FOR INSTALLATION ON WINDOWS NT	4
IDENTIFYING YOUR COMPUTER FOR LICENSING	5
LICENSE INSTALLATION	6
LICENSING OVERVIEW	6
NODE LOCKED LICENSES	6
ENVIRONMENT VARIABLES	7
FLOATING LICENSES	7
SERVER CONFIGURATION	8
CLIENT CONFIGURATION	9
3. USING TURBO WRITER	11
4. COLOUR CODING	13
ENABLING COLOUR-CODING	13
USING CHANGE BARS	15
SYSTEM FLAGS	16
CHANGING COLOURS	17

5. TEMPLATES	19
INTRODUCTION	19
WHAT IS A TEMPLATE ?	19
CHANGING THE TEMPLATE EXPANSION KEY	20
VHDL TEMPLATE STYLES	20
USING TEMPLATE HISTORY	22
USING SPECIAL VHDL TEMPLATES	23
6. FOLDING	25
INTRODUCTION	25
EXAMPLE	25
7. CODE OUTLINING	27
INTRODUCTION	27
EXAMPLE	27
8. COMMENT HEADERS	31
INTRODUCTION	31
INSERTING A FILE COMMENT HEADER	31
9. AUTOMATIC TESTBENCH GENERATION	35
INTRODUCTION	35
WHAT IS A TESTBENCH ?	35
RUNNING TESTBENCH GENERATION	35
VHDL EXAMPLE	36
VERILOG EXAMPLE	38
10. AUTOMATIC COMPONENT INSTANTIATION	39

INTRODUCTION	39
MAKING THE COMPONENT	39
RE-ASSIGNING THE KEYS	40
11. ADDING HDL FILE TYPES	41
INTRODUCTION	41
ADDING HDL FILE TYPES	41
ADDING VERILOG FILE TYPES	42
12. ADDING COMPILER INTERFACES	43
INTRODUCTION	43
CONFIGURING TURBO WRITER TO USE TWO COMPILERS	43
13. MODELSIM INTERFACE	46
INTRODUCTION	46
MODELSIM INSTALLATION	46
COMPILING VHDL	46
14. RENOIR INTERFACE	49
INTRODUCTION	49
CONFIGURING RENOIR TO CALL TURBO WRITER	49
CONFIGURING TURBO WRITER TO CALL RENOIR	49
15. TROUBLESHOOTING	51
APPENDIX A : VHDL KEYWORDS AND TEMPLATES	59
BASIC VERILOG KEYWORDS	59

VERILOG SYSTEM TASK AND SYSTEM FUNCTION KEYWORDS	59
FUNCTION KEYWORDS	59
VERILOG COMPILER DIRECTIVE KEYWORDS	59
VERILOG TEMPLATE DEFINITIONS	60
VERILOG COMPILER DIRECTIVE TEMPLATE DEFINITIONS	61
<u>APPENDIX B : VHDL KEYWORDS AND TEMPLATES</u>	<u>63</u>
BASIC VHDL-93 KEYWORDS	63
STD 1164 KEYWORDS	63
VITAL 2.2B KEYWORD	64
VHDL TEMPLATE DEFINITIONS	65

1. Introduction

Welcome to Turbo Writer for Windows. Turbo Writer is a tool designed to dramatically improve productivity in the generation of Hardware Description Language (HDL) files. It supports the two most commonly used HDLs, VHDL and Verilog. Turbo Writer provides many facilities including keyword colour coding, rapid code generation through the use of templates, folding features, test-bench generation, Compiler interfaces and many other features.

Turbo Writer is built upon CodeWright, the popular editor for Windows developed by the Premia Corporation. CodeWright is a fully featured editor providing unlimited code size as well as being the fastest Windows-based editor currently available.

ABOUT THIS MANUAL

This Manual covers the HDL extensions to CodeWright provided by Turbo Writer and does not describe the use of CodeWright itself. The user should firstly become familiar with CodeWright by reading the CodeWright user's manual supplied with this product.

Turbo Writer is a comprehensive set of extensions to the CodeWright editor from Premia. Turbo Writer incorporates a fully featured version of Codewright as part of the installation.

Assumptions

This manual assumes that you are familiar with Windows and Windows type applications. It also makes some limited assumptions about your knowledge of CodeWright. This software is supplied with the CodeWright Users Manual from Premia. It is a good idea to make yourself familiar with some of the terms and concepts introduced in this manual.

Conventions

To make information easier to find, this manual adheres to a number of style conventions. They are as follows :-

Courier This style is used to indicate text in a file or a file name.

Introduction

Italic This style is used for emphasis.

<space> This style is used to indicate a keyboard character.

2. Installation

Introduction

The installation of Turbo Writer incorporates a full installation of Codewright automatically. The user need not be concerned with the separate installation of Codewright.

System Requirements

The following is a minimum system on which Turbo Writer can be installed.

<i>System</i>	A 486 machine or better with at least 16MB of RAM is a minimum. Performance will also vary based on the video adapter, driver and video mode (resolution and number of colours) you have selected.
<i>Memory</i>	Turbo Writer requires 7MB to operate.
<i>Windows</i>	Microsoft Windows 95 or Windows NT 4 or greater is required. 16 bit versions of Windows are no longer supported.
<i>Storage</i>	A CD rom drive and a hard drive are required. Turbo Writer will store temporary files on the hard drive. For a full installation 30MB of free space is required.

Installing Turbo Writer

The procedure for installing Turbo Writer should be familiar to anyone who has installed any Windows software. Firstly insert the CD into your disk drive, locate and run the install.exe program. The Turbo Writer installation software will prompt for a target directory; this is where the Codewright and Turbo Writer software will be installed. If the directory you specify does not exist then it will be automatically created.

The installation software will prompt for a Company name. This is for automatic insertion into the example file comment header and is set to default to Saros Technology Ltd. This is merely to give the user an example of how comment templates can be configured. See section 7 for more detail on comment headers.

The installation software will also prompt for a directory in which the Model Technology ModelSim software is installed. If you do not have ModelSim then leave

Installation

the directory as the default. The installation simply adds this directory to your **cwright.ini** file in a set-up command as follows :-

```
[vhdl]  
VsystemSetup=c:\ModelSim\win32pe
```

If you have the ModelSim software then make sure the directory entered is the correct one. You are looking for the directory containing vsim.exe, normally named win32 or win32pe. If after installation you realise that this directory is incorrect then simply edit the **cwright.ini** file and restart Turbo Writer.

After the ModelSim directory has been specified, the installation software copies a number of files from the CD ROM into the target directory.

Finally the user is asked whether Turbo Writer should modify the user's autoexec.bat file with an addition to the path. The install directory is added to the path. If the modifications are required then click yes. The Turbo Writer installation software makes a backup of the existing autoexec.bat in autoexec.bak if it is modified. The modifications are necessary for running the ModelSim interface and you should reboot your machine for these changes to take effect.

In addition to modifying the autoexec.bat, the windows **win.ini** file is modified to add file associations for *.vhd, *.v, *.vlg, *.vsh, *.vsd, *.tb and *.tf files. The installation software then adds a program group and icon for Turbo Writer.

Once the installation is complete, Turbo Writer is ready to be used, ensure that the dongle - if appropriate - is connected to the parallel port and double click on the Turbo Writer icon.

Evaluation Period

Turbo Writer will run for a free trial period of 30 days following the time of installation. After this time the product will cease to run and a license must be purchased.

Additional notes for installation on Windows NT

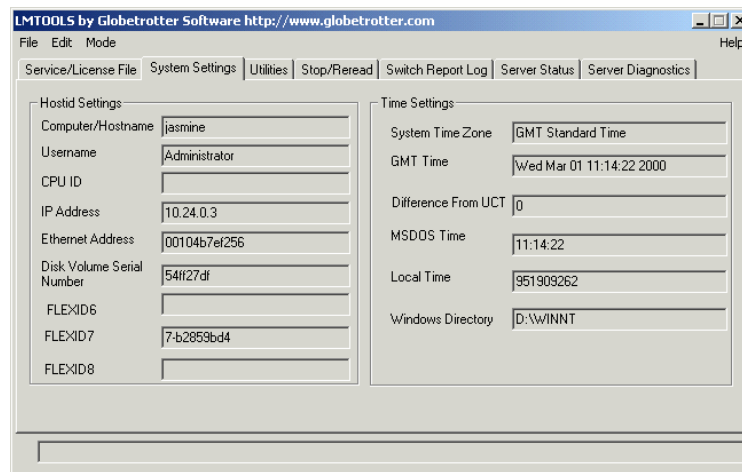
To use a dongle on Windows NT you need to load a dongle driver. The installation program must be run by the administrator or a user with administrator privileges. Without this the registry cannot be updated and the dongle driver will not work.

To load the dongle driver run the setupx86.exe program in the \Rainbow\win_nt directory on the CD ROM. From the Function Menu pick install, click OK, and reboot when the installation has completed.

Identifying your Computer for Licensing

For Saros to be able to generate a license you must supply us some information about how you want it to work. If you buy the Media pack from us we can use the FlexID dongle in that and you do not need to supply us any additional information. If you are not buying a Media Pack or want us to use a disk id, network card address or a FlexID you already own to generate your license, we need you to provide us with a number.

After installing Turbo Writer go to the Start menu, then program files, then HDL Turbo Writer and choose the FLEXlm Utilities menu. From the dialog pick the System Settings tab. You should see a dialog box that looks like this:



Installation

Here the number of interest is the Disk Volume Serial number if you want a disk based license, or the Ethernet Address for a network card based license, or either of the FLEXID7 or FLEXID8 boxes for dongle based license.

Licensing Installation Turbo writer is protected from illegal copying by the industry standard FlexLM licensing software. For information on installing FlexLM licences see the two sections below called “Node locked licenses” and “Floating licences”

Licensing Overview

There are two types of license available, termed floating and node locked. A node locked license is tied to a piece of hardware. This can be a hard disk, a network card or a FlexID dongle. The software will only run on a computer to which the licensed hardware directly connected. Of these methods the dongle is the most flexible because it is easy to move from one machine to another, for example if you want to upgrade the computer or work from home. You may already have a suitable FlexID from another product you would like to use for Turbo Writer also, or you can buy another one from us in the Turbo Writer media pack.

Floating licences are more flexible still. If you put a floating license on a network and you can run Turbo Writer from any other machine on the network. The number of concurrent invocations of Turbo Writer is limited to the number of floating licenses you buy. This concurrent invocation count is maintained by a machine called the license server. You can identify the license server to us by its boot disk volume serial number, its Ethernet card addresses or by the number of a FlexID dongle connected to it.

Node locked licenses

Node locked licenses look like this:

```
FEATURE Twriter SAROS 6.000 permanent uncounted 7DC9092EA016 \  
HOSTID=FLEXID=7-b2859bd4
```

Turbo Writer can run so long as it can find the license file, the HOSTID matches the computer, and 4th field is at least as high as the version number of the product.

Turbo Writer finds its license file using an environment variable called LM_LICENSE_FILE which needs to hold the full path to the license file.

Installation

Environment variables If you are on Windows 95 or 98 set this in your autoexec.bat by adding a line like the one below and reboot.

```
set LM_LICENSE_FILE=d:\tools\flexlm\license.dat
```

On Windows NT 4 you would add these variables to your environment by running the System icon in the Control Panel, selecting the Environment tab and selecting any one of the user environment variables. Now put LM_LICENSE_FILE in as the Variable box and the full path and file name of your license file in the Value box, overwriting the existing contents of these boxes. Now click Set, and the new environment variable should appear in the list. Finally click apply and you are ready to run. A reboot is not necessary on NT.

Windows NT 2000 subtly different from NT 4. From the system icon in the control panel, select the advanced tab. Then click the Environment variables, and click the New button under the user section and fill in the dialog box.

Turbo Writer should now run up. If you get a licensing error message, read the trouble shooting section near the end of this manual. There is a procedure there for debugging node locked and floating licenses.

Floating Licenses

Turbo Writer will run on any computer with a network connection to the licensed machine called the license server. The license server can be specified by its disk id, network card address or the number of a FlexID attached to its parallel port. The dongle is the most flexible because it allows the license server to be changed by moving the dongle to another machine. Floating licenses look like this:

```
SERVER jasmine FLEXID=7-b2859bd4 1967
DAEMON SAROS c:\twriter\saros.exe
FEATURE Twriter SAROS 6.000 permanent 2 3A1A9D8056F5 ck=34
```

Installation

The first line specifies the network name of the license server, jasmine in this case. You will need to edit this field to match the machine you are using as a license server. Then follows the method used to verify the validity of the server, a FlexID dongle number 7-b2859bd4 in this case. It could equally be a disk id or a network card address. Check this matches the license server. The final entry on the sever line is the TCP/IP port used by Turbo Writer to connect to the license server. This needs to be a port that is not used by anything else on the machine. Normally numbers above 1000 and below 30000 are safest.

The second line locates the saros.exe file, which is part of the licensing system. This can be found in the Turbo Writer installation directory. You will probably need to edit this path to reflect the place you installed Turbo Writer.

The third line is the feature line which you shouldn't need to edit unless it got wrapped by the email. Its forth field (6.000 in this case) is the latest licensed version number of the product. Its fifth field (2 in this case) is the maximum number of concurrent invocations of Turbo Writer that are licensed on your network.

There are two halves to configuring a floating license. There is the server and there are the clients.

Server Configuration

The license server is configured using the lmttools icon in the HDL Turbo Writer start menu. Start this program and from the opening screen next to the planet graphic, select the Configuration Using services button. Now select the Configure Services tab and fill in the 3 paths using the browse buttons. You can find the lmgrd.exe in the Turbo Writer directory, the license file is where ever you saved it, and the debug log can go where ever you want. If the boxes are already filled in for another product, change the Service Name in the top box by typing over it with a name of your choosing. For example "Turbo Writer License Manager" should be fine. The other boxes on this dialog should then go blank as you move into them, and you can configure the new license server independently from the existing one. If you want the server to start automatically when the machine is rebooted, Click the "Use NT services", and the Start Server at power up" buttons.

Installation

Once you are happy with the Configure Services screen you can go to the Start/Stop/Reread Control tab and push the Start Server button. The server is now configured and running.

Client Configuration

Client configuration is simple. All the client needs is its LM_LICENSE_FILE environment variable set correctly. This can be done in two ways. Either set the variable to point to the license file from where it can read the server line and find out the machine and port name, or set the variable to point directly to the port on the license server. Saros recommends the latter option because it means you don't need to copy the license or share the disk containing it. Examples of each of the two options are shown below. For details on how to set environment variables, see the end of the previous section on node locked licenses.

Environment variable setting for going via a license file

LM_LICENSE_FILE=\\jasmine\d\tools\flexlm\license.dat

Environment variable setting for going direct to the port on the license server

LM_LICENSE_FILE=1967@jasmine

The actual values will depend on your system.

If you have multiple license servers or multiple license files, you can separate them with semicolons. For example:

LM_LICENSE_FILE=1967@jasmine;1760@holly;1720@mainserver

Turbo Writer should now be ready to run. If you get a licensing error message, read the trouble shooting section at the end of this manual, starting on page 51. There is a procedure there for debugging node locked and floating licenses.

3. Using Turbo Writer

Once Turbo Writer has been installed you are ready to start using it. Double click on the Turbo Writer icon and Turbo Writer should start and look something like Figure 1.

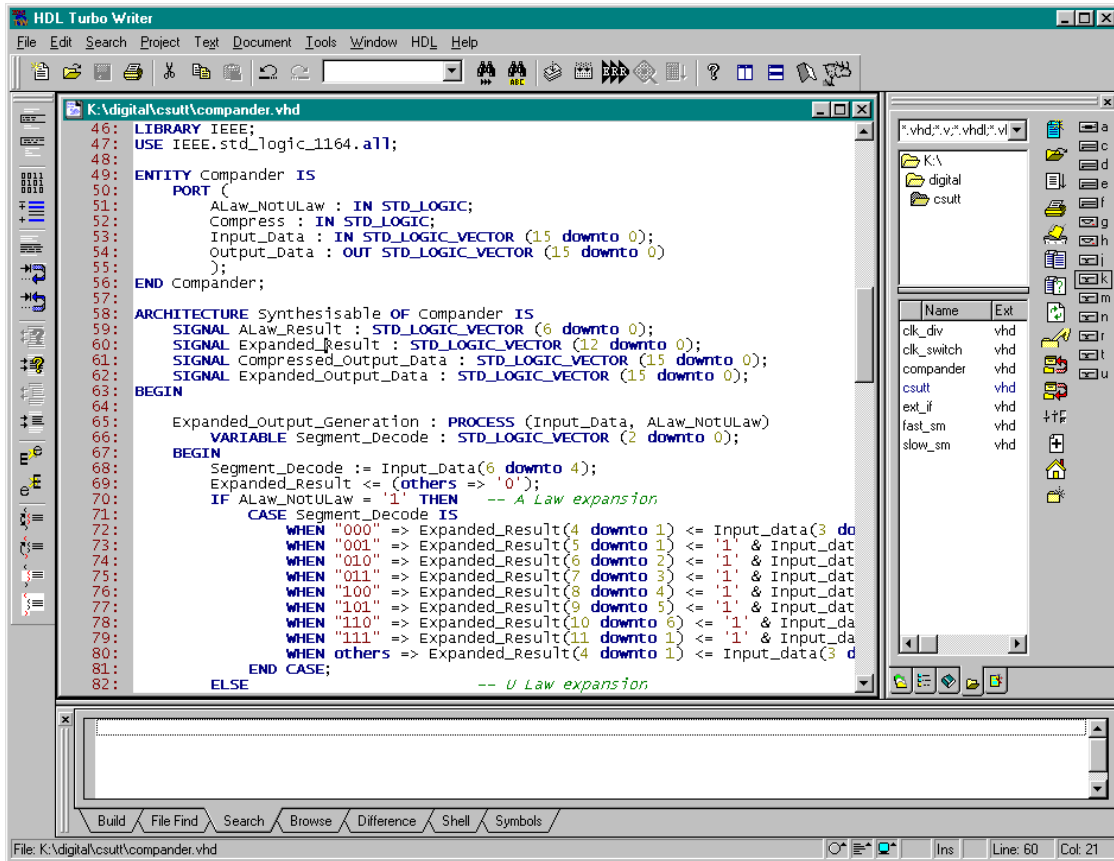


Figure 1. Turbo Writer for Windows

Turbo Writer provides four key command areas :-

- The Menu bar provides a classic Windows menu interface.
- The Ribbon Buttons are a strip of custom buttons underneath the Menu.
- The Side Bar provides additional quick access functions.
- The Project window provides comprehensive file and project functions.

The additional functions provided by Turbo Writer are mainly concentrated in an HDL menu on the main menu bar. As can be seen from Figure 2, there are a number of HDL related features which are discussed in the following sections.

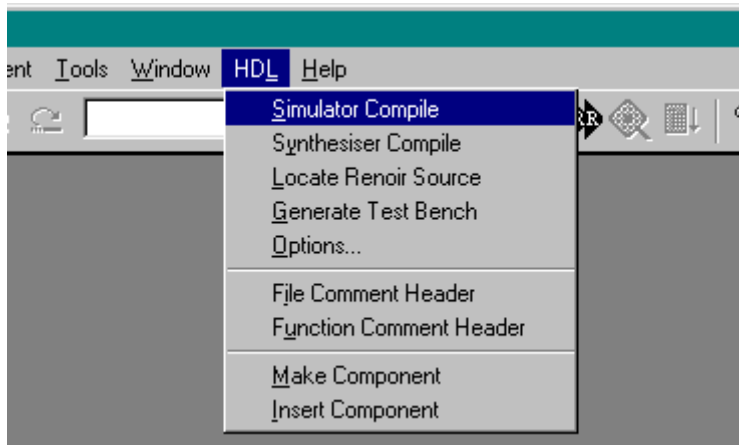


Figure 2. HDL Menu

You may find when you select the HDL menu that all the items of the menu are disabled. This is because the HDL features are only available for file extensions which Turbo Writer "knows" as HDLs. For instance a ".vhd" file is a VHDL file and ".v" file is a Verilog file. To enable the functions in the HDL menu, start editing a file with one of these extensions.

If you have other extensions which you wish to use for HDLs to get all the Turbo Writer facilities then a new file extension needs to be added. This is very simple to do and is fully described in Section 11 : Adding HDL File Types.

4. Colour-coding

Introduction

Turbo Writer provides the facilities for colour-coding languages with a number of predefined colour types. Turbo Writer provides the keyword definitions for VHDL and Verilog as well as further colour definitions for language extensions and compiler directives. A full definition of the VHDL and Verilog keywords for colour-coding are given in Appendix A and Appendix B.

Enabling Colour-coding

Turbo Writer normally enables colour-coding during installation but here is a short description of how to enable colour-coding. Colour-coding is enabled in the Tools..Customize..Language dialogue on a language by language basis. Turbo Writer provides facilities which can be enabled globally for all files or for individual files.

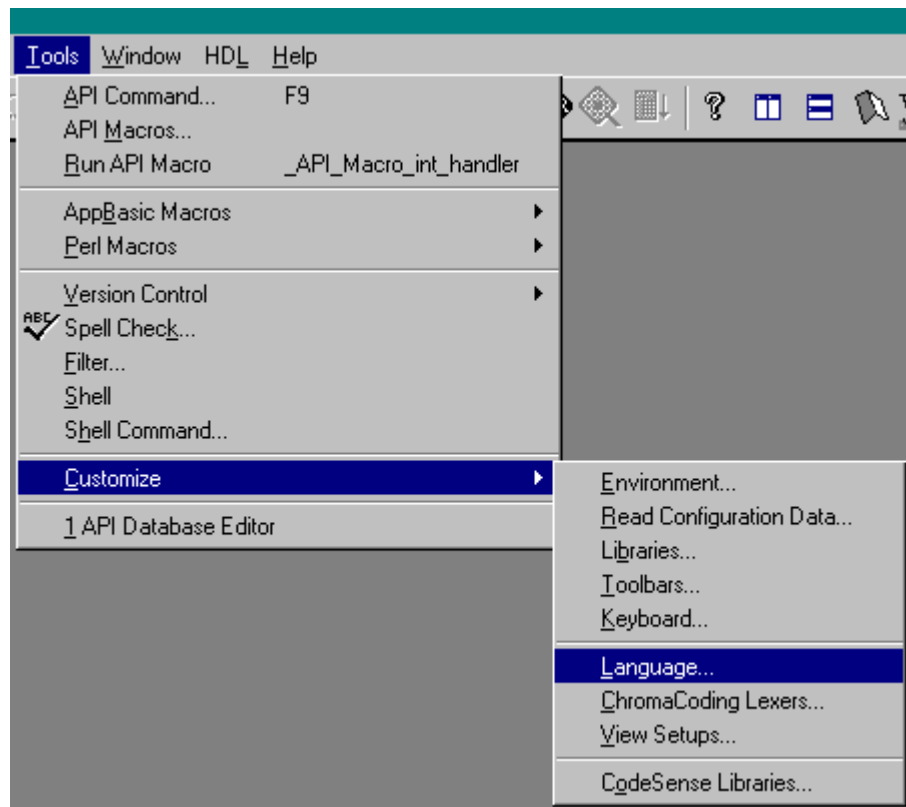


Figure 3. Enabling Colour Coding

As can be seen from Figure 3, the Tools..Customize..Language dialogue is where colour-coding is enabled and disabled. After selecting the menu, the dialogue shown in Figure 4 should appear.

The user must select the particular extension which requires changing from the list on the left. The user can create new extensions based on existing ones if required by mapping from one extension to another.

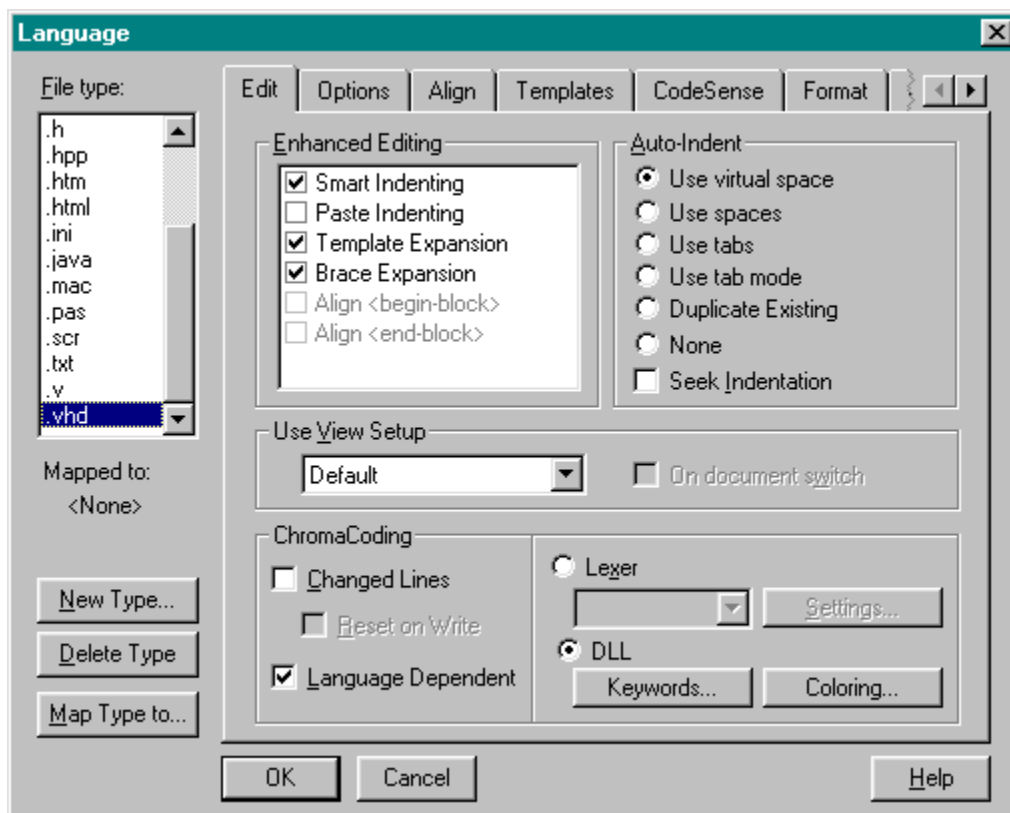


Figure 4. Extension specific dialogue

The important settings for colour-coding are to the left of the dialogue. The Enhanced Editing options section allows the user to choose Template expansion (this is the normal default installation), Smart Indenting (this provides indenting based on certain keywords defined in the language extensions), Paste Indenting (provides intelligent indenting when indenting), and Brace Expansion.

Using Change Bars

Turbo Writer has the ability to display change bars alongside your code. Within the ChromaCoding section of the dialogue illustrated in Figure 4 are two switches for enabling change bars for changed lines and the reset on write switch.

Change bars are displayed as either blue or red bars alongside the text within the Turbo Writer window. Blue bars are displayed when existing lines are edited. Red bars indicate that lines have been added. As a default Turbo Writer is configured to display change bars until a file is written. The reset-on-write switch can be turned off for situations where the user may want the change bars to persist after a write. However change bars only persist during the current editing session.

Change bars can be turned off by simply deselecting it as an option on the dialogue.

System Flags

For completeness it is useful at this point to discuss the System Flags section of the Extension-specific Setup. As a reminder the flags are shown in Figure 5.

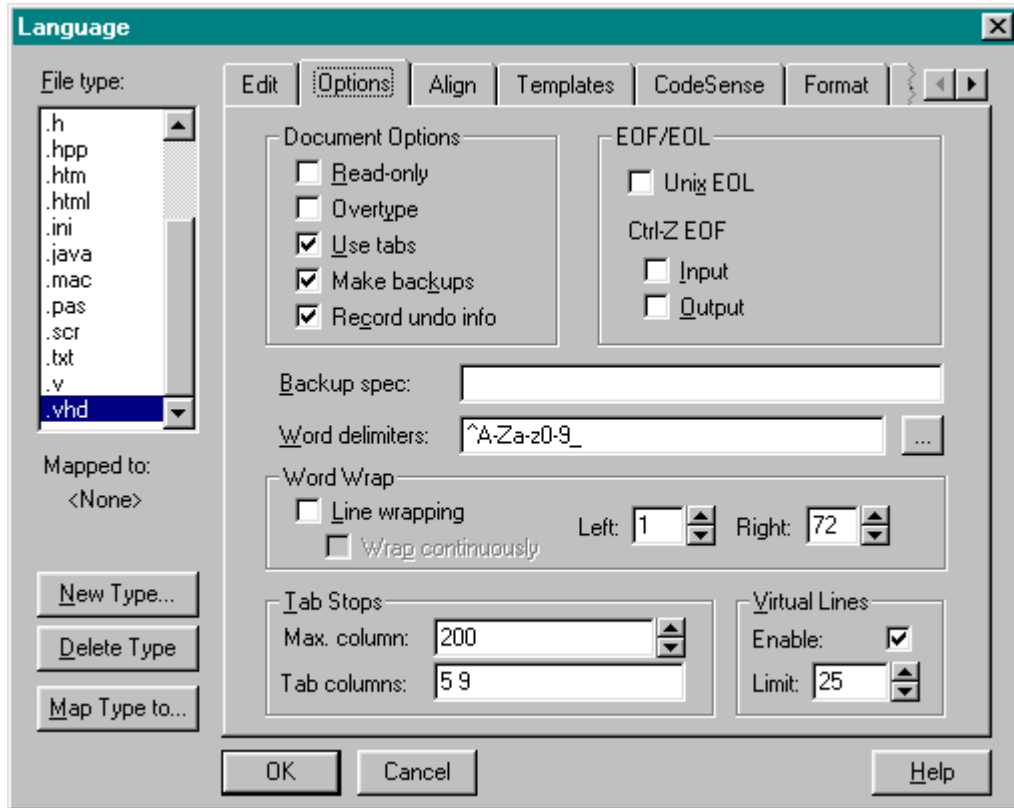


Figure 5. System Flags

Many of the flags are self-evident but some need additional explanation. The Unix EOL option allows Turbo Writer to correctly read/write Unix text files. This feature is invaluable when using Turbo Writer on a mixed network of PCs and Unix machines. Note however that this setting only has an effect on lines added to a file by Turbo Writer.

Changing Colours

The colours for the different types of colour-coding are configured in the Tools..Customize..View Setups dialogue, on the Colors tab. The dialogue shown in Figure 6 should appear.

By scrolling the colour list up and down the user can find the particular colour-coding feature which needs changing. As can be seen from Figure 6 a number of HDL specific colours are configured by Turbo Writer.

To change a colour for a particular text type, click on it, then select the foreground and background colours from the colour palette. At this point the Test button can be pressed in order to see the effects of the change on your current file. Once the change is completed click Apply, or OK buttons.

For VHDL, the Std_1164 library keywords and Vital library keywords can be picked out in different colours.

For Verilog, the OVI standard Compiler Directives and System Task and System Function keywords can be picked out in different colours.

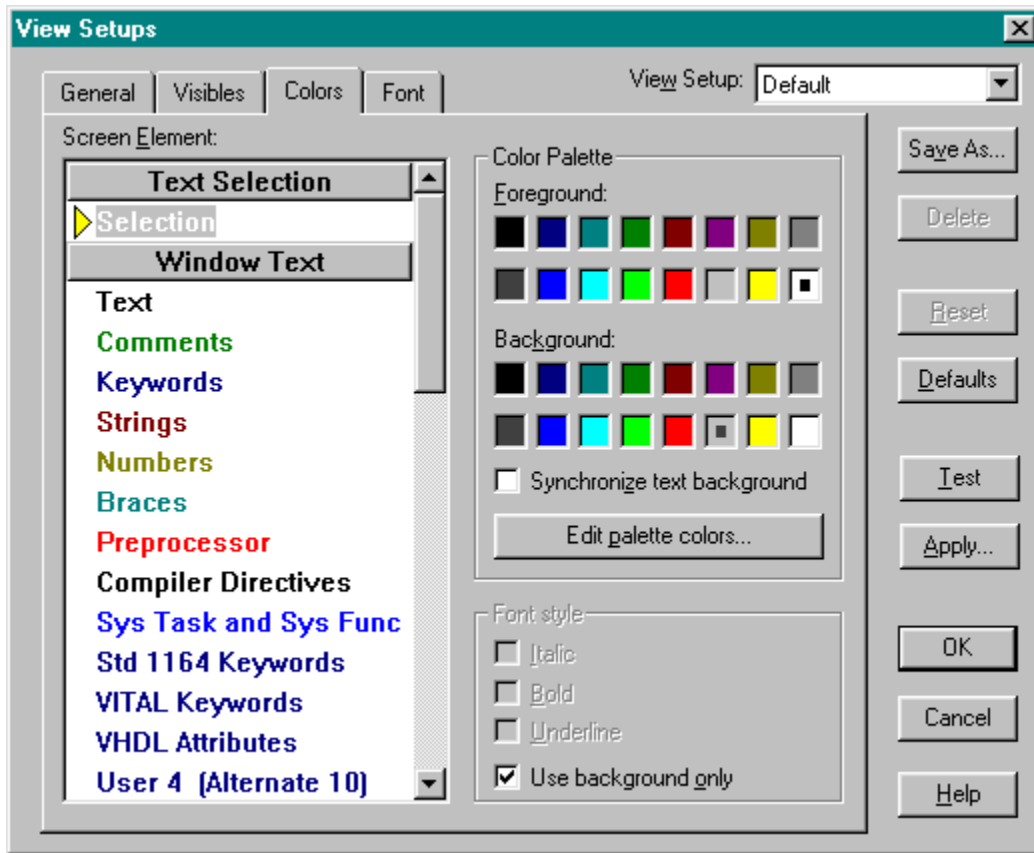


Figure 6. Configuring Colours for language dependent colour-coding.

5. Templates

Introduction

Language-dependent templates allow the user to rapidly generate repetitive HDL code sequences which are syntactically correct. The templates supplied with Turbo Writer can be added to as well as modified by the user. A predefined keypress provides template expansion but the choice of key can also be quickly and easily modified by the user.

Template expansion is enabled in the Language dependent Setup dialogue described in Section 4. If you have problems with template expansion try checking the Enhanced Editing options section of that dialogue .

What is a Template ?

A template comprises a template name which is usually a minimal number of characters and a template definition into which the template name will be converted. For example, a common Verilog language construction is :-

```
always @(sensitivity_list)
    begin
        additional_code
    end
```

A template **a1** exists to insert such a language construction. By typing **a1<tab>** the user is prompted for the sensitivity list. After entering the list the full construction is inserted into your file with the cursor positioned to enter the additional code in the above example.

Templates may contain a number of built-in macro functions which can enhance the flexibility and sophistication of the templates. The macros are preceded by a % character and perform a variety of functions such as cursor movement, popup user prompts, insertion of environment variables, insertion of the current date and time and many more functions. A full listing of all macro functions can be found in the Template Macros section of the CodeWright Users Manual.

A full definition of the VHDL and Verilog templates is given in Appendix A and Appendix B respectively. However if you are unsure which templates are available or what they do, a quick way of seeing the available templates is to select the Templates tab button on the Tools..Customize..Language dialogue .

Changing the Template Expansion Key

The **<tab>** key is the default template expansion key for Turbo Writer, this perhaps provides the quickest template insertion scheme but it does sometimes interfere with other code being entered; eg. when a variable called **a1** is required. To change the key assigned to template expansion requires a change to the CodeWright initialisation file **cwright.ini**.

NOTE : Some early versions of Turbo Writer are delivered with the template expansion key configured to be the **<space>** key. Simply check your **cwright.ini** to confirm which system you are running. The **cwright.ini** file can be found in the CodeWright installation directory.

By default Codewright uses the **<space>** key to expand templates. The Turbo Writer installation overrides this with the **<tab>** key. The following code is inserted in the **cwright.ini** file and can be modified by the user to re-assign template expansion to a different key.

```
[editor]
ExtKmapAssign='<Space>' 'Space'
ExtKmapAssign='<Tab>' '__ext_expand 1'
```

The first statement re-assigns the **<space>** key to simply entering a space. The second statement re-assigns the template expansion function to the **<tab>** key. The user can simply substitute another key name to assign the template expansion to a different key.

VHDL

Unlike Verilog, VHDL is a case-insensitive language. This means that

Template Styles

templates could be written in a variety of styles. Turbo Writer provides three of the most common styles for VHDL templates. These are selected via the HDL..Options dialogue . This is illustrated in Figure 7.

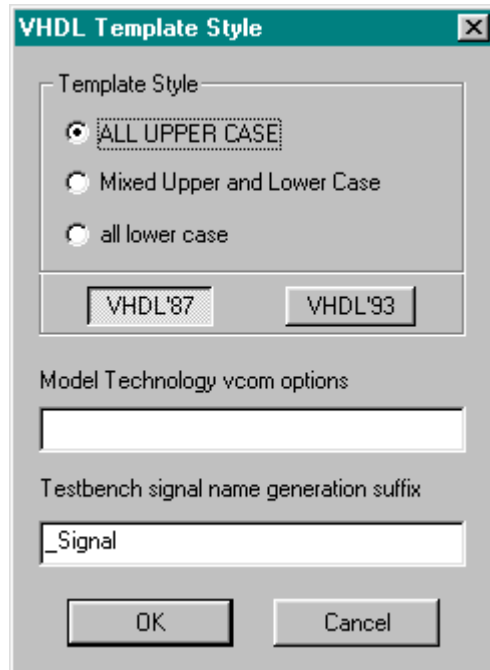


Figure 7. VHDL Template Styles

The three styles supported cover the most common keyword styles used in VHDL. They are all capital letters, first letter capitalised and all lower case. Click on the style required.

When the OK button is pressed the selected style is written into the **[vhdl]** section of the **cwright.ini** file. This ensures that this style is the default each time Turbo Writer is loaded. The additions to **cwright.ini** are as follows:-.

```
[vhdl]
VHDLTemplateStyle=1
```

The number following the **VHDLTemplateStyle** command determines which of the three styles is to be used.

0 = All capitals

1 = Mixed upper and lower case

2 = All lower case.

If no style is specified in the **cwright.ini** file, the system defaults to number 0.

Using Template History

In several of the VHDL templates, the user is prompted for information in a pop-up dialogue box. Alongside the text entry box is a down arrow button. See Figure 8.

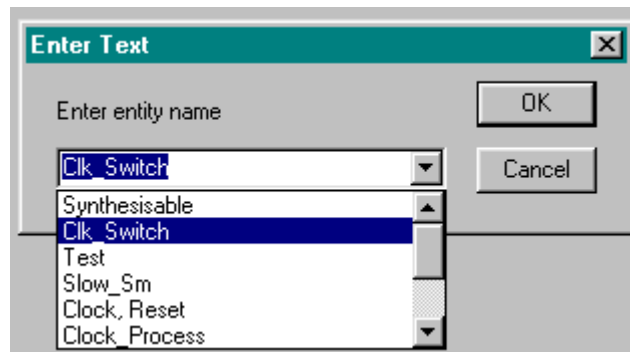


Figure 8. Template history mechanism

This allows the user to browse back through the history of values entered in templates. The down arrow cursor key can be used in addition to clicking the arrow button with the mouse. The feature is especially useful when a value from a previous template needs to be entered again, e.g. enter an entity template followed by an architecture template. The entity name is needed in the architecture statement. Simply press the cursor key twice to pop up the entity name.

Using Special VHDL Templates

In several of the VHDL templates, there are special features designed to provide flexibility and productivity to VHDL code generation. For example a number of VHDL statements can be preceded by a label. The process statement is one of them. The **ps** template provides a number of options making it productive yet extremely flexible.

After pressing **ps<tab>** the user is prompted for a label; if one is required type it in, if not press **<esc>** or click the cancel button. Next, the user is prompted for a sensitivity list. Type in a comma-separated list and the template will insert it surrounded automatically by parentheses. Press **<esc>** and the list is omitted. With these options the following styles of process can be generated from one template as well as other combinations.

```
ExampleLabel : PROCESS (X,Y,RESET)
BEGIN
END PROCESS ExampleLabel;
```

```
Process
Begin
End Process;
```

Signal, Variable and Constant templates have default types which can be configured by the user. These are simply configured in the **cwright.ini** file with statements as follows :-

```
[vhdl]
VHDLSignalType="Std_logic"
VHDLVariableType="Bit"
VHDLConstantType="integer range 0 to 1023"
```

Another VHDL specific feature of Turbo Writer is included in the architecture template. If an entity template is used before an architecture template, Turbo Writer automatically includes the entity name as the default entity for the architecture template.

6. Folding

Introduction

Folding is a method of compacting large HDL files into key sections which can be folded and unfolded. For large HDL files the benefit of folding is immense. It allows complex detail to be hidden until it is required to be modified, the user simply unfolds the section being worked on. Once it is complete it can be folded up and another section unfolded.

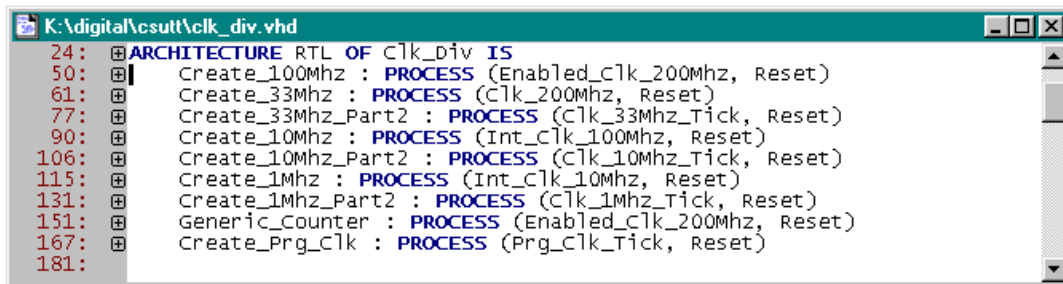
For each HDL there are certain constructs which are configured to trigger folding. By pressing the Fold button on the Ribbon bar the user can toggle between the folded and unfolded modes. The Fold button is shown below:-



Example

To illustrate the benefit of the folding feature the following are examples of how a file can be folded and then a selected area unfolded for editing or viewing.

After loading an example VHDL file and pressing the Fold button, Turbo Writer displays the file as shown in Figure 9.

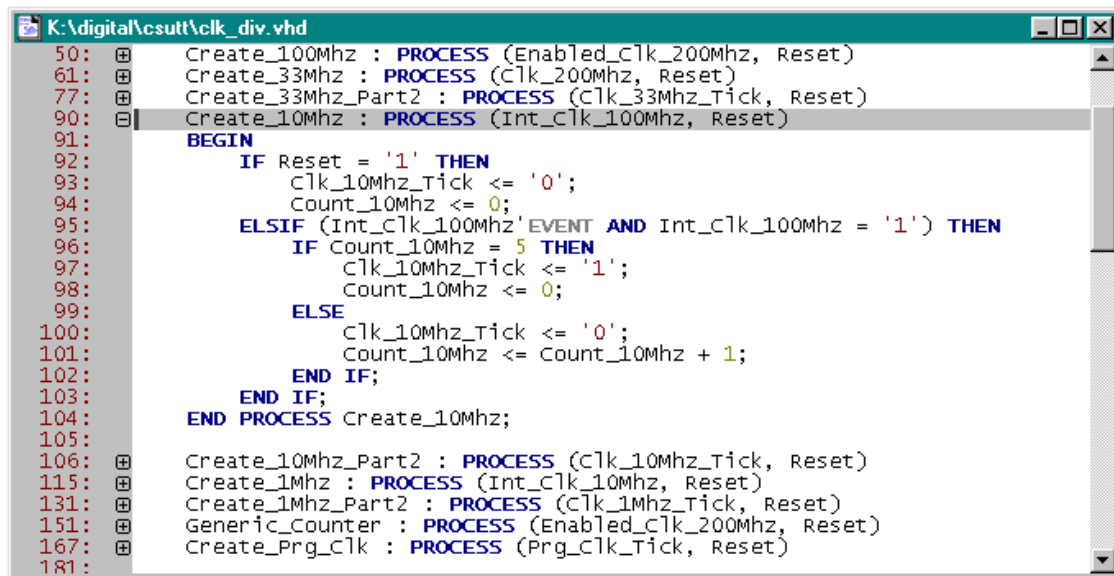


```
K:\digital\csutt\clk_div.vhd
24:  ⊕ ARCHITECTURE RTL OF Clk_Div IS
50:  ⊕   Create_100Mhz : PROCESS (Enabled_Clk_200Mhz, Reset)
61:  ⊕   Create_33Mhz  : PROCESS (Clk_200Mhz, Reset)
77:  ⊕   Create_33Mhz_Part2 : PROCESS (Clk_33Mhz_Tick, Reset)
90:  ⊕   Create_10Mhz   : PROCESS (Int_Clk_100Mhz, Reset)
106: ⊕   Create_10Mhz_Part2 : PROCESS (Clk_10Mhz_Tick, Reset)
115: ⊕   Create_1Mhz    : PROCESS (Int_Clk_10Mhz, Reset)
131: ⊕   Create_1Mhz_Part2 : PROCESS (Clk_1Mhz_Tick, Reset)
151: ⊕   Generic_Counter : PROCESS (Enabled_Clk_200Mhz, Reset)
167: ⊕   Create_Prg_Clk  : PROCESS (Prg_Clk_Tick, Reset)
181:
```

Figure 9. Folding a VHDL File.

Folding

As can be seen, each of the processes in the file is compressed to a title and a + button. The + indicates that the section can be expanded. Note that the line numbers are correctly shown in the left column. By double clicking on the +, a section can be unfolded. Figure 10 illustrates the result.



```
K:\digital\csutt\clk_div.vhd
50: ⊕ Create_100Mhz : PROCESS (Enabled_Clk_200Mhz, Reset)
61: ⊕ Create_33Mhz : PROCESS (Clk_200Mhz, Reset)
77: ⊕ Create_33Mhz_Part2 : PROCESS (Clk_33Mhz_Tick, Reset)
90: ⊖ Create_10Mhz : PROCESS (Int_Clk_100Mhz, Reset)
91: BEGIN
92:   IF Reset = '1' THEN
93:     Clk_10Mhz_Tick <= '0';
94:     Count_10Mhz <= 0;
95:   ELSIF (Int_Clk_100Mhz'EVENT AND Int_Clk_100Mhz = '1') THEN
96:     IF Count_10Mhz = 5 THEN
97:       Clk_10Mhz_Tick <= '1';
98:       Count_10Mhz <= 0;
99:     ELSE
100:       Clk_10Mhz_Tick <= '0';
101:       Count_10Mhz <= Count_10Mhz + 1;
102:     END IF;
103:   END IF;
104: END PROCESS Create_10Mhz;
105:
106: ⊕ Create_10Mhz_Part2 : PROCESS (Clk_10Mhz_Tick, Reset)
115: ⊕ Create_1Mhz : PROCESS (Int_Clk_10Mhz, Reset)
131: ⊕ Create_1Mhz_Part2 : PROCESS (Clk_1Mhz_Tick, Reset)
151: ⊕ Generic_Counter : PROCESS (Enabled_Clk_200Mhz, Reset)
167: ⊕ Create_Prg_Clk : PROCESS (Prg_Clk_Tick, Reset)
181:
```

Figure 10. Unfolding a section

Although the file is displayed in what CodeWright calls compacted mode, the text can still be edited, copied and deleted just as normal text. In fact any edit operation that can be performed on a file can be performed in Folded mode. Note also when a section is unfolded the + button for that section changes to a -. Clicking the “-” button folds that section back up.

By clicking the Fold button again the whole file shall be displayed in unfolded mode. Alternatively pressing the <ESC> key also exits the Folded mode.

7. Code Outlining

Introduction

Code outlining is a more sophisticated method of viewing key elements of your code than folding. It takes the essence of folding and provides a new outline window containing icons to represent key elements of your HDL code. The outline view is one of the tabs of the Project Window.

Example

To illustrate the benefit of the outlining feature the following are examples of how a VHDL file or a Verilog file would look alongside the outliner.

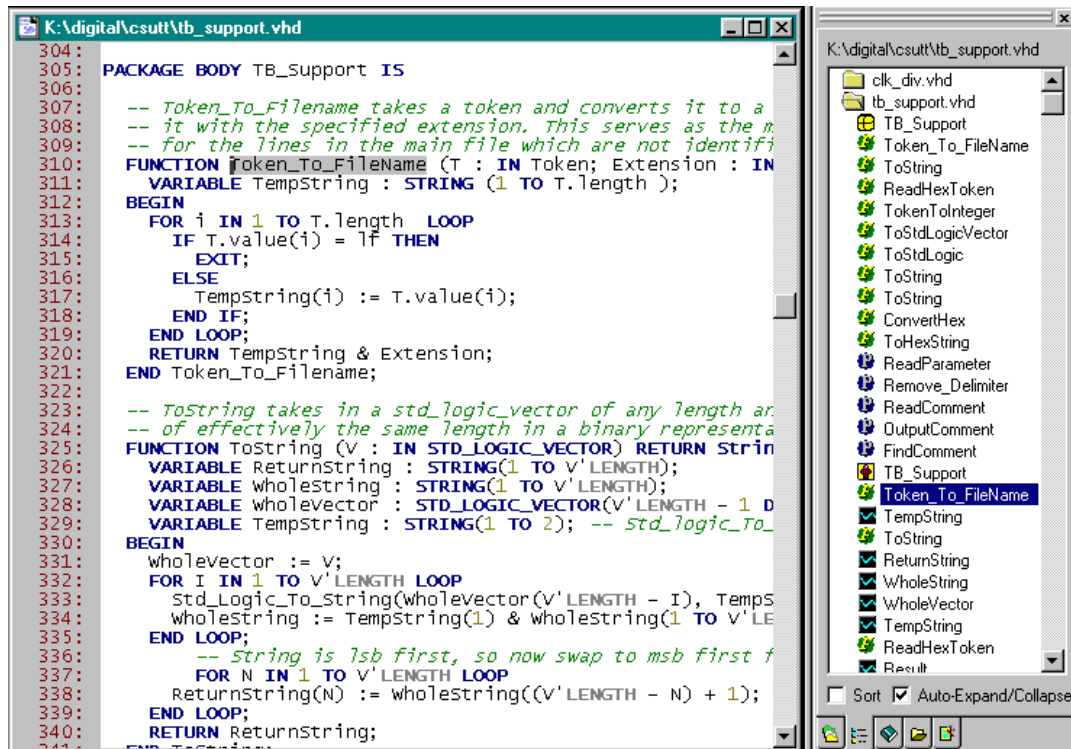


Figure 11. VHDL Code Outlining

As can be seen from the VHDL example, there are a variety of icons to represent elements of the VHDL language. By simply double-clicking the appropriate icon in the outline window, the cursor is placed at that point in the source file. The outline window is dynamically updated every few seconds so that icons appear as you enter code.

One of the attractions of the outlining system is that the user can choose which of the key elements should appear in the outline window. The outline symbols to be displayed are set in the Tools..Customize..Language dialogue. From the CodeSense tab select the Symbol Patterns button (Figure 12). By default all outline symbols are enabled, but by deselecting various symbols the user can customise the system to present their chosen outline view.

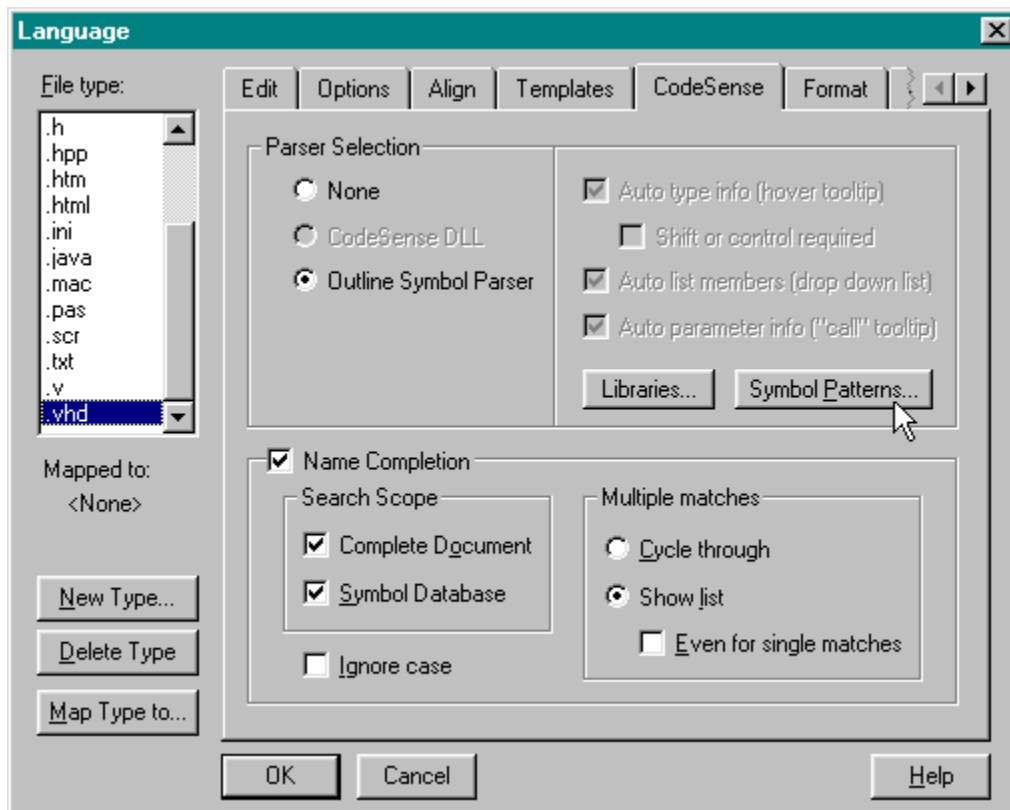


Figure 12: Selecting the Symbol Patterns button

As can be seen, both VHDL and Verilog outline displays can be individually tailored to the users needs from this dialogue. However the Symbol Patterns dialogue is language dependant, so you must select the correct File type prior to selecting the Symbol Patterns button.

Here is the Symbol Patterns dialogue :-

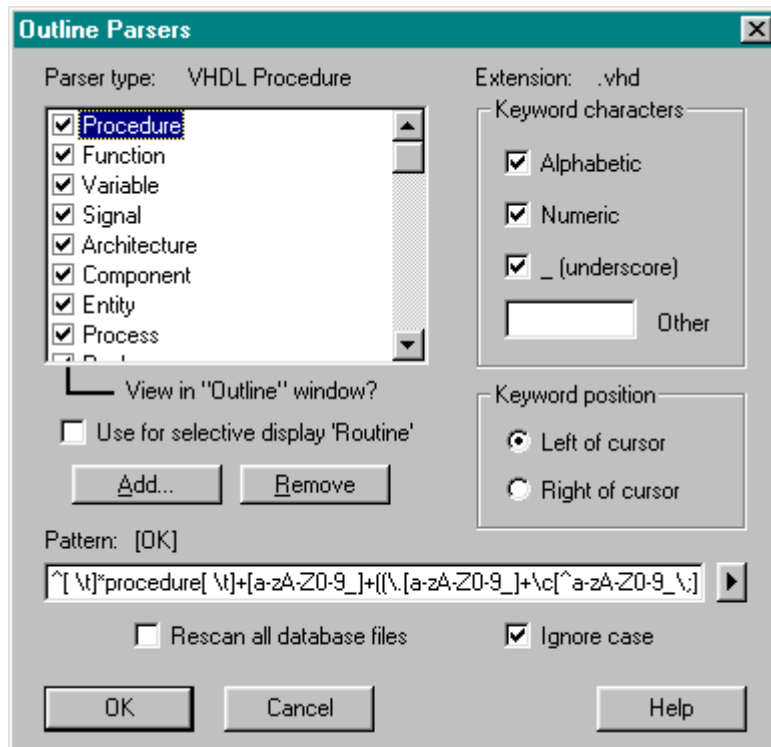
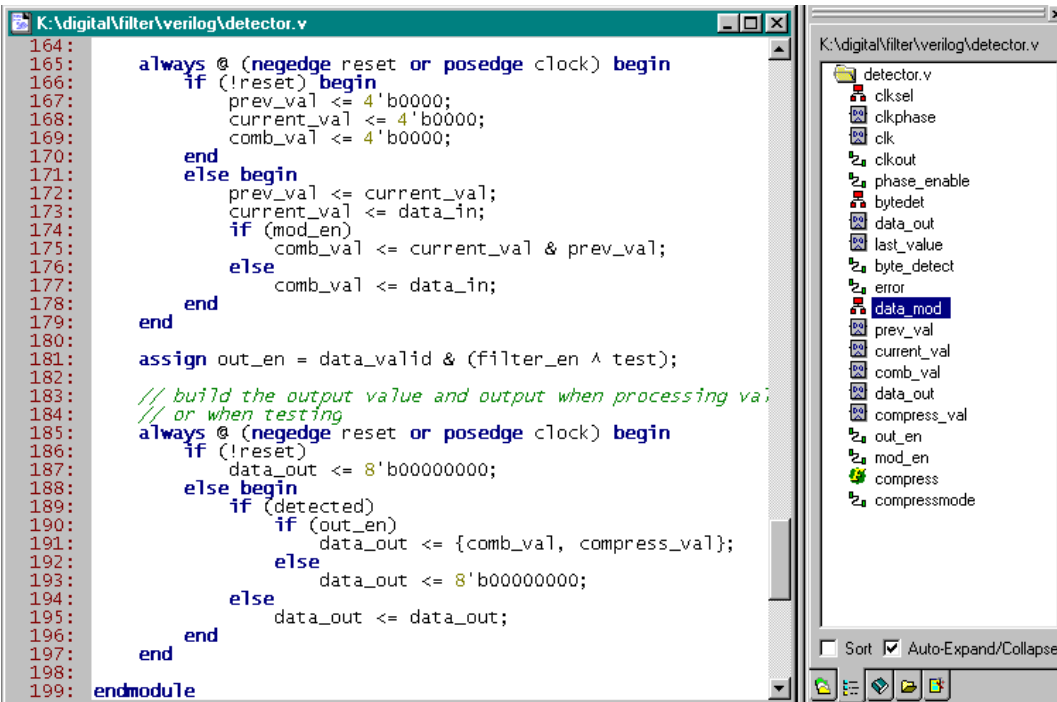


Figure 13 : Symbol Patterns Dialogue for VHDL Files

Deselection of the tick box next to a VHDL construct in the list in this dialogue will remove that construct from the outline view. For example, you may want to exclude signals and variables from your outline as there may be large numbers of these in a typical file.

Code Outlining

For completeness, here is a small Verilog example :-



```
164:
165:
166:   always @ (negedge reset or posedge clock) begin
167:     if (!reset) begin
168:       prev_val <= 4'b0000;
169:       current_val <= 4'b0000;
170:       comb_val <= 4'b0000;
171:     end
172:     else begin
173:       prev_val <= current_val;
174:       current_val <= data_in;
175:       if (mod_en)
176:         comb_val <= current_val & prev_val;
177:       else
178:         comb_val <= data_in;
179:     end
180:   end
181:   assign out_en = data_valid & (filter_en ^ test);
182:
183:   // build the output value and output when processing va
184:   // or when testing
185:   always @ (negedge reset or posedge clock) begin
186:     if (!reset)
187:       data_out <= 8'b00000000;
188:     else begin
189:       if (detected)
190:         if (out_en)
191:           data_out <= {comb_val, compress_val};
192:         else
193:           data_out <= 8'b00000000;
194:       else
195:         data_out <= data_out;
196:     end
197:   end
198: endmodule
199:
```

The screenshot shows a Verilog code editor window titled 'K:\digital\filter\verilog\detector.v'. The code is a Verilog module named 'detector.v'. It features two 'always' blocks triggered by a clock edge (negedge reset or posedge clock). The first block handles reset and data processing, updating 'prev_val', 'current_val', and 'comb_val'. The second block handles output generation, setting 'data_out' based on 'detected' and 'out_en' signals. A right-hand pane shows a project tree with various signals and variables like 'clk', 'data_out', and 'compress_val'. The 'data_mod' variable is highlighted in the tree.

Figure 14 : Verilog Code Outlining

8. Comment Headers

Introduction

Comment headers are blocks of program comments which are designed to give source code a consistent look and structure. They provide invaluable information for tracking source code history and bug fixes. Turbo Writer provides a feature for inserting standard comment blocks for file headers and for functions.

Since each user may have different ideas and styles for implementing these headers, Turbo Writer loads these comment headers from predefined external files. These external files are simple text files which are readily edited by the user. Sample files are supplied with Turbo Writer which illustrate the powerful macro features which can be incorporated into the standard files.

Inserting a File Comment Header

Comment headers are inserted via two means. From the HDL menu select the File Comment Header menu item. Dependent on the language currently being edited, a suitable comment header will be inserted at the top of the file regardless of where the cursor is currently positioned. Here is an example of the Verilog File Comment Header.

```
/*
**
** DEMONSTRATION FILE HEADER
** Copyright (c) SAROS Technology Ltd 1994
**
**
** Project Name       : DEMO
**
** Author             : Nick Heaton
** Creation Date      : 10/22/94 15:10:10
** Version Number     : 3.0
**
** Revision History   :
**
** Date              Initials      Modification
*/
```

Comment Headers

```
**
**
** Description          :
**
**
**
**
**
*****/
```

In addition to being able to insert comment headers from the HDL menu, two buttons on the sidebar are specifically provided for the same job. The two buttons are :-



The File Header button



The Function Header button

Simply pressing these buttons has the same effect as using the menu options.

The VHDL file comment header has a number of features which may be useful to users.

```
-----
-- File name      : d:\projects\lmt\vhdl\fiforam
-- Title         : Demo
-- Library       : WORK
--              :
-- Purpose       :
--              :
-- Created On    : 02/11/95 12:10:56
--              :
-- Comments      :
--              :
-- Assumptions   : none
-- Limitations   : none
-- Known Errors  : none
```


Comment Headers

in macro %10 in the `cwright.ini` file and used in the file header. The header is stored in `file.vhd` and can be loaded and edited just like any other text file.

A function header is very similar to a file header except that it is usually used to comment each function or process. The header is inserted at the start of the line on which the cursor is currently positioned. The demo VHDL function comment header prompts the user for a function name and then inserts the following into your file:-

```
-----  
--  
-- DEMONSTRATION FUNCTION HEADER  
-- Copyright (c) SAROS Technology Ltd 1994  
--  
-- Function name       : demo  
-- Creation Date       : 10/23/94 09:07:09  
--  
-----
```

The demo function header shows how the date and time can be automatically inserted giving your code an accurate stamp of when the function was generated.

9. Automatic Testbench Generation

Introduction

Testbenches are an important part of HDL development and use. They are a repetitive task to write since much of the code is almost repeated but with subtle differences.

Turbo Writer aims to remove some of the tedium from testbench generation by automatically generating a skeleton testbench.

What is a Testbench ?

A testbench is a harness around an HDL module specifically designed for testing. A testbench usually instantiates a test module alongside the module under test (MUT). A number of signals are declared which fully connect the test module to the MUT. All inputs to the MUT are connected to outputs from the test module and all outputs from the MUT are connected to inputs on the test module.

Running Testbench Generation

To generate a testbench or test fixture (Verilog nomenclature) file simply open the source HDL file and select the following menu item.

The Testbench generation software looks at the current file in order to find the first Entity (VHDL) or module (Verilog). A testbench based on the first declaration is created in a new file with a name based on the original file name.

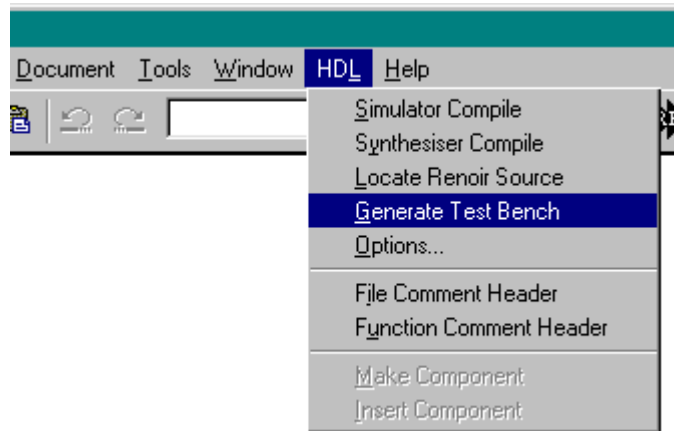


Figure 15. Generating Test Bench HDL.

VHDL Example

The following is an example of Testbench generation in VHDL. The first file listing is the source. The second is the Testbench automatically generated from it.

```
ENTITY FifoRam IS
    GENERIC (
        size : integer := 8
    );
    PORT (
        DataIn  : IN Std_logic_vector(size-1 DOWNTO 0);
        WE      : IN Std_logic;
        WAddr   : IN Std_logic_vector(1 DOWNTO 0);
        RAddr   : IN Std_logic_vector(1 DOWNTO 0);
        DataOut : OUT Std_logic_vector(size-1 DOWNTO 0)
    );
END FifoRam;
```

The resulting testbench looks like this :-

```
LIBRARY IEEE;
USE IEEE.STD_Logic_1164.ALL;

ENTITY FifoRam_tb IS
END FifoRam_tb;

ARCHITECTURE HTWTestBench OF FifoRam_tb IS

    COMPONENT FifoRam
        GENERIC (
            size : integer := 8
        );
        PORT (
            DataIn  : IN Std_logic_vector(size-1 DOWNTO 0);
            WE      : IN Std_logic;
            WAddr   : IN Std_logic_vector(1 DOWNTO 0);
```

Automatic Testbench Generation

```

        RAddr  : IN Std_logic_vector(1 DOWNTO 0);
        DataOut : OUT Std_logic_vector(size-1 DOWNTO 0)
    );
END COMPONENT;

SIGNAL DataIn_Signal  : Std_logic_vector(size-1 DOWNTO 0);
SIGNAL WE_Signal      : Std_logic;
SIGNAL WAddr_Signal   : Std_logic_vector(1 DOWNTO 0);
SIGNAL RAddr_Signal   : Std_logic_vector(1 DOWNTO 0);
SIGNAL DataOut_Signal : Std_logic_vector(size-1 DOWNTO 0);

BEGIN
    U1 : FifoRam
        GENERIC MAP (size => 8)
        PORT MAP (
            DataIn => DataIn_Signal,
            WE     => WE_Signal,
            WAddr  => WAddr_Signal,
            RAddr  => RAddr_Signal,
            DataOut => DataOut_Signal);
END HTWTestBench;
```

As can be seen, the generator takes the entity declaration and constructs a complete testbench around it. All relevant signals are declared and named appropriately.

In the case where generics are declared but have no default value, the testbench generator will insert a default value of `## MISSING GENERIC ##`. This is illegal VHDL and will be caught by the VHDL compiler. The user must specify a generic value.

Another potential trap that the test bench generator can be caught out by is unconstrained arrays as ports on the entity. e.g. fred : in std_logic_vector;

The testbench generator will create a signal with the same declaration which is illegal VHDL. A range must be added to the matching signal declaration for each of this type of port.

Verilog Example

The following is an example of test fixture generation in Verilog. The first file listing is the source. The second is the fixture file automatically generated from it.

```
module demotest (x,y,z);  
    input x,y;  
    output z;  
endmodule;
```

The test fixture file generated looks like this :-

```
module TestFixture;  
  
    reg x_Signal,y_Signal;  
    wire z_Signal;  
  
    demotest U1 (.x(x_Signal),.y(y_Signal),.z(z_Signal));  
  
    // Enter fixture code here  
  
endmodule // TestFixture
```

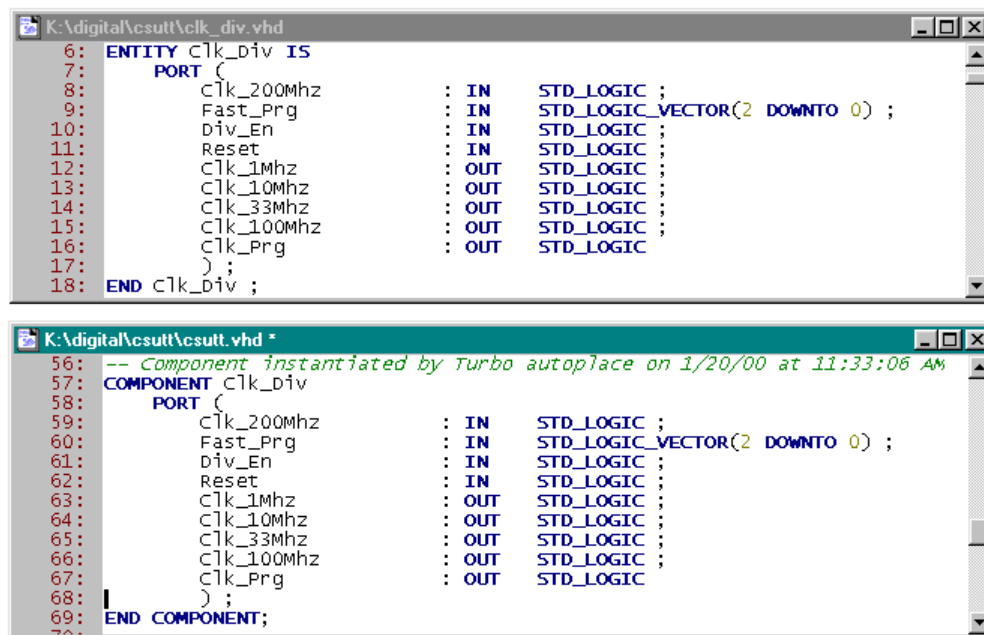
10. Automatic Component Instantiation

Introduction

Turbo Writer provides another productivity tool for VHDL, namely automatic component instantiation. When creating structural levels in VHDL, the user is frequently faced with instantiating an existing entity via a component statement. This feature relieves the tedium of this repetitive process.

Making the Component

The first step in this simple process is to "make" the component. This is done by placing the cursor on the line where the entity declaration is made. Positional accuracy isn't required as the function "looks" at the beginning of the line on which the cursor is placed and steps forward through the file until the first entity is found. Simply press **Alt-M** to "make" the component. This captures the component definition in the scrap buffer. Now all the user must do is place the cursor where the component is required, this could be in the same file or in another file. Once the cursor is placed simply click **Alt-I** to insert the component. This process can be repeated as many times as are necessary. See below for an example of capturing an entity and inserting a component in another file :-



```
K:\digital\csutt\clk_div.vhd
6: ENTITY clk_div IS
7:   PORT (
8:     clk_200mhz      : IN   STD_LOGIC ;
9:     Fast_Prg        : IN   STD_LOGIC_VECTOR(2 DOWNTO 0) ;
10:    Div_En           : IN   STD_LOGIC ;
11:    Reset            : IN   STD_LOGIC ;
12:    clk_1Mhz         : OUT  STD_LOGIC ;
13:    clk_10Mhz        : OUT  STD_LOGIC ;
14:    clk_33Mhz        : OUT  STD_LOGIC ;
15:    clk_100Mhz       : OUT  STD_LOGIC ;
16:    clk_Prg          : OUT  STD_LOGIC ;
17:   );
18: END clk_div ;

K:\digital\csutt\csutt.vhd *
56: -- Component instantiated by Turbo autoplace on 1/20/00 at 11:33:06 AM
57: COMPONENT clk_div
58:   PORT (
59:     clk_200mhz      : IN   STD_LOGIC ;
60:     Fast_Prg        : IN   STD_LOGIC_VECTOR(2 DOWNTO 0) ;
61:     Div_En           : IN   STD_LOGIC ;
62:     Reset            : IN   STD_LOGIC ;
63:     clk_1Mhz         : OUT  STD_LOGIC ;
64:     clk_10Mhz        : OUT  STD_LOGIC ;
65:     clk_33Mhz        : OUT  STD_LOGIC ;
66:     clk_100Mhz       : OUT  STD_LOGIC ;
67:     clk_Prg          : OUT  STD_LOGIC ;
68:   );
69: END COMPONENT;
```

Figure 16. Example of automatic component instantiation.

Re-Assigning the the Keys

The "make" and "instantiate" keys **Alt-M** and **Alt-I** are assigned in the `cwright.ini` file with the commands :-

```
[KmapAssign]
KmapAssign='<Alt-I>' '_vhd_insertcomponent'
KmapAssign='<Alt-M>' '_vhd_makecomponent'
```

If the user wishes to re-assign the functions it is a simple matter of editing the `cwright.ini` to map the functions to different key combinations.

11. Adding HDL File Types

Introduction

Turbo Writer is delivered with the ".vhd", ".v" and ".vlg" file types configured as the only HDL file extensions. It is quite likely that the user has other file types containing VHDL or Verilog code. This section describes the steps needed to enable the HDL functions and colour-coding facilities for a new extension.

Adding VHDL File Types

To add a VHDL file type to the list of Turbo Writer "known" file types, some modifications to the `cwright.ini` file are required. The `cwright.ini` file is separated into different sections denoted by square parenthesis. In addition to the lines already within the following sections, some new lines must be created, and some existing lines need to be appended (note that text added to existing lines is underlined for highlighting purposes only). The following will add a new VHDL file type of ".vhdl".

Close Turbo Writer before editing `cwright.ini` and do not use Turbo Writer as the editor for this file!

In the `[vhdl]` section add this line:

```
AddVhdlType=".vhdl"
```

In the `[Compiler]` section add these lines:

```
CompilerAssign="Vsystem",'.vhdl'
```

```
CompilerNewExt=.vhdl
```

In the `[Extension]` section add this line

```
ExtAlias=.vhdl,.vhd
```

Append these lines:

In the `[Editor]` section append to these lines

```
FilterAdd='HDL (*.vhd, *.v, *.vhdl)', '*.*vhd;*.*v;*.*vhdl',-1
```

```
FilterAdd='VHDL Files (*.vhd, *.vhdl)', '*.*vhd;*.*vhdl',-1
```

Adding Verilog File Types

To add a Verilog file type to the list of Turbo Writer "known" file types, some modifications to the `cwright.ini` file are required. The `cwright.ini` file is separated into different sections denoted by square parenthesis. In addition to the lines already within the following sections, some new lines must be created, and some existing lines need to be appended (note that text added to existing lines is underlined for highlighting purposes only). The following will add a new Verilog file type of ".vlog".

Close Turbo Writer before editing `cwright.ini` and do not use Turbo Writer as the editor for this file!

In the `[verilog]` section add this line:

```
AddVerilogType=".vlog"
```

In the `[Compiler]` section add these lines:

```
CompilerAssign="VsystemVlog",'.vlog'  
CompilerNewExt=.vlog
```

In the `[Extension]` section add this line

```
ExtAlias=.vlog,.v
```

Append these lines:

In the `[Editor]` section append to these lines

```
FilterAdd='HDL (*.vhd, *.v, *.vlog)','*.vhd;*.v;*.vlog',-1  
FilterAdd='Verilog Files (*.v,*.vlg,*.vlog)','*.v;*.vlg;*.vlog',-1
```

After making these modifications to `cwright.ini` Turbo Writer will use the new types next time it starts.

12. Adding Compiler Interfaces

Introduction

Turbo Writer is designed to operate with a variety of language tools and, with a little configuration from the user, any of the user's tools. Since most HDLs are simulatable it is anticipated that most users will have some kind of simulator. In addition users may have a Synthesiser or other compiler. Turbo Writer is designed to cater for situations where the user would like to do syntax checking on source code with two different tools.

Configuring Turbo Writer to use two Compilers

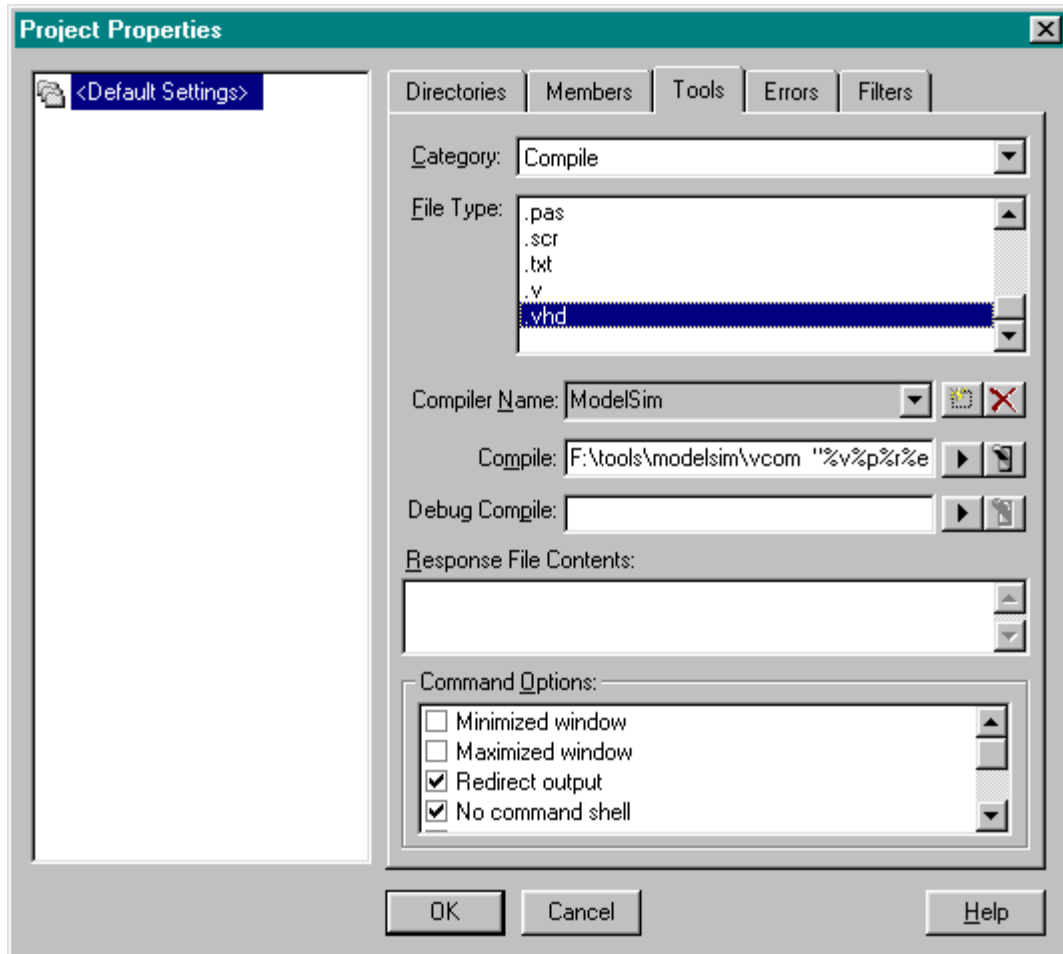
The installation of Turbo Writer hides the Model Technology compiler interface from the user. However if you look at the Project..Project Properties dialogue, select the Tools tab, select the Compile Category and select the relevant File Type, the commands used to interface to Model Technology can be seen and modified (Figure 17). A Compiler is given a name, in the case of Model Technology it is called ModelSim. The user can add new compiler interfaces through this dialogue, by clicking the New button to the right of the Compiler Name selector. For example an alternative Model Technology compiler set up could be defined, called Vsystem. The configuration is then saved in the **cwright.ini** file and can be modified by editing that file.

Once the new compiler interface has been created it is necessary to configure Turbo Writer to enable it to switch compilers. This again is done through the cwright.ini file. Within the relevant HDL section ie. **[vhdl]** or **[Verilog]** the user can place a statement such as

```
[vhdl]
VHDL SimulatorSetup="Vsystem"
VHDL SynthesiserSetup="Exemplar"
```

Figure 17. Setting the compiler options.

When the HDL Simulator Compile command is selected, Turbo Writer installs the correct Compiler interface and then calls it. Similarly when the Synthesiser Compile



command is selected, the Synthesiser Compiler interface is installed and called up. Obviously the nomenclature of Simulator and Synthesiser are purely arbitrary and the user can assign them at will.

For Verilog the commands are as follows :-

```
[verilog]  
VerilogSimulatorSetup="Vsystem"  
VerilogSynthesiserSetup="Exemplar"
```

The spelling of Synthesiser is somewhat dependent on which side of the Atlantic you reside. To cater for both flavours Turbo Writer accepts **VerilogSynthesizerSetup** and **VHDLSynthesizerSetup** as well as the English spellings.

For a more detailed discussion of implementing compiler interfaces see the Compiler Setup section of the Extension-Specific Features chapter in the Codewright User's Manual.

13. ModelSim Interface

Introduction

The Model Technology ModelSim software is a fully-featured VHDL simulator which is directly supported by the Turbo Writer system. When Turbo Writer is installed, the program prompts the user for a Model Technology installation directory. This is required because Turbo Writer directly invokes a program supplied with the ModelSim software.

ModelSim Installation

The ModelSim installed directory is configured in the **cwright.ini** file in the following way :-

```
[vhdl]
VsystemSetup = c:\vhdl
VsystemVcomOptions = '-work mylib'
```

If the installation directory was entered incorrectly, or if the ModelSim software has been installed after Turbo Writer, then edit the **cwright.ini** file with the correct directory. Turbo Writer must be reloaded after editing this file for the change to take effect.

Compiling VHDL

Once configured correctly, the ModelSim button on the button bar allows direct interfacing to the Model Technology software. The button is illustrated below :-



Clicking on this button performs a sequence of tasks.

- It executes a compile by directly running vcom.exe.
- It captures the output from the compile and checks for errors.
- Any errors are flagged and can be stepped through using the ERR button on the toolbar, or by double clicking the error in the error list, or by using the Find Next Error menu pick.

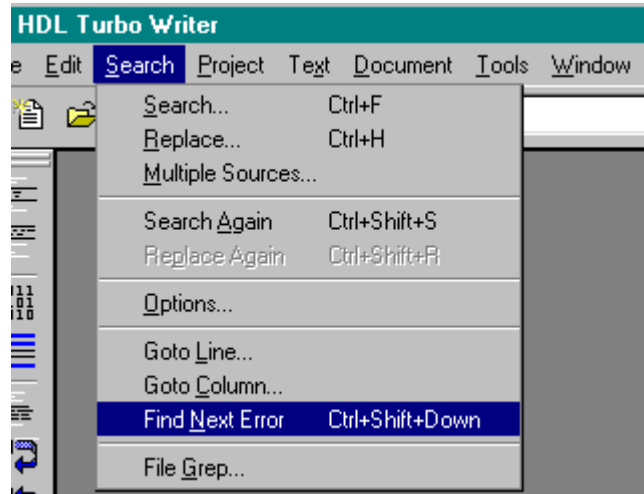
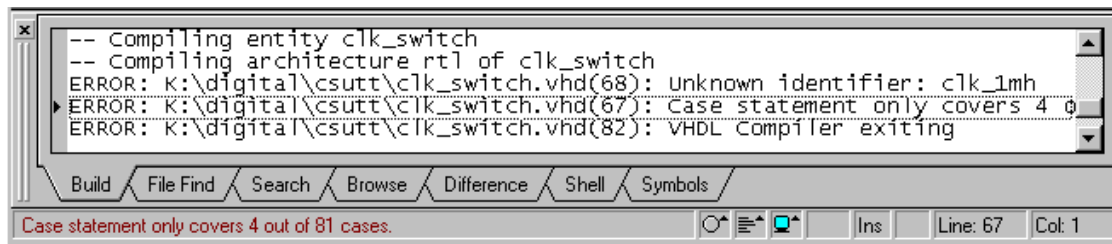


Figure 18. Stepping through errors in the ModelSim Output.

The output from the Model Technology compiler is shown in the tabbed output window in the same way as output from File Grep and File Find. The Tabbed Window can be detached from the bottom of the Turbo Writer main window, or minimised (Auto Hide). For more information on the tabbed output feature please read the Codewright User's manual.

Tabbed output from ModelSim is illustrated as follows :-



```
-- Compiling entity clk_switch
-- Compiling architecture rtl of clk_switch
ERROR: K:\digital\csutt\clk_switch.vhd(68): Unknown identifier: clk_1mh
ERROR: K:\digital\csutt\clk_switch.vhd(67): Case statement only covers 4 o
ERROR: K:\digital\csutt\clk_switch.vhd(82): VHDL Compiler exiting
```

The screenshot shows a ModelSim command window with a menu bar (Build, File Find, Search, Browse, Difference, Shell, Symbols) and a status bar (Case statement only covers 4 out of 81 cases, Line: 67, Col: 1). The output text is as follows:

Figure 19. Tabbed Output from ModelSim

Options to the compiler can be issued through the HDL/Options menu pick and typing in the “Model Technology vcom options” dialog box.

For example, the option **-work myproject** would cause the compilation to go into the library myproject rather than the default library work.

14. Renoir Interface

Introduction

Renoir is a Graphical HDL capture environment that allows VHDL and Verilog designs to be entered via a number of methods more intelligible to humans than pages of HDL code. Renoir allows the user to work with block diagrams, state machines, flow charts and truth tables, as well as HDL files where appropriate. However because the end product from the process is still the HDL code, and it is the HDL code that must be debugged, there needs to be a way of tracking a problem in the code back to the source from which it was generated. HDL Turbo Writer and Renoir work together to make this possible.

Once configured, opening a VHDL file in Renoir will call Turbo Writer to display the file. Conversely placing the Turbo Writer cursor on any line of code generated by Renoir and clicking the "HDL/Locate Renoir Source" menu, will cause Renoir to display the source for the selected line of code.

Configuring Renoir to call Turbo Writer

HDL Turbo Writer can be configured as the default editor within Renoir as follows:

In the Renoir Options menu pick General Preferences.

In the External Editor box enter

```
twriter "%p" -G"%1"
```

assuming Turbo Writer is in your path. If not replace twriter with a full path to the twriter.exe file.

Configuring Turbo Writer to call Renoir

The link back from Turbo Writer to Renoir relies on interprocess communication using TCP/IP ports. For this to work you need to have the TCP/IP network protocol installed. Consult your systems administrator for details on this if you are unsure.

You also need two environment variables to be set correctly. These variables are set automatically if you invoke Turbo Writer from within Renoir, but we recommend you configure them manually so the products can be invoked in either order.

Set the environment variable RENOIR_PORT to the name of the machine running Renoir, followed by a colon and a valid and unused TCP/IP port number. Also set the RENOIRHOME variable to the Renoir installation directory.

Renoir Interface

For example on Windows 95 you could add the following to your autoexec.bat file

```
set RENOIR_PORT=localhost:1782  
set RENOIRHOME=c:\renoir98\
```

And reboot your computer.

On Windows NT you would add these variables to your environment by running the System icon in the Control Panel, selecting the Environment tab and selecting any one of the user environment variables. Now put RENOIR_PORT in as the Variable box and localhost:1782 in the Value box, overwriting the existing contents of these boxes. Now click Set, and the new environment variable should appear in the list. Repeat the procedure for the RENOIRHOME variable. Finally click apply and you are ready to run Renoir. If Renoir was already running when you made the changes above, you will need to close it and open it again.

Renoir must be open for Turbo Writer to be able to communicate with it.

Now for a test. Open a Renoir generated HDL file, put the text cursor on a line of interest and click the HDL menu and pick Locate Renoir Source. Renoir should display the appropriate source document with the focus set to the specific source item of interest.

15. Troubleshooting

Here are a collection of the most common questions asked with suggested answers.

When I start Turbo Writer I get a license error saying checkout failed. I have a node locked license.

Node Locked licenses are where the license file does not contain SERVER or DAEMON lines. If your license is of this type, this is the section for you. If not read the corresponding section on floating licenses starting on page 53 instead.

Node Locked license debug procedure:

1. Checking your license file for email corruption. Open the license in notepad (not word or wordpad) and look at the FEATURE line. It will start with the word FEATURE and end with a \ which is a continuation character. Any line breaks prior to this \ should be deleted. The line after the \ contains the rest of the information and completes the Turbo Writer section of the license file. For example, this is a correct license:

```
FEATURE Twriter SAROS 6.000 permanent uncounted 7DC9092EA016 \  
HOSTID=FLEXID=7-b2859bd4 ck=34
```

But this one has been broken by the email after the uncounted word:

```
FEATURE Twriter SAROS 6.000 permanent uncounted  
7DC9092EA016 \  
HOSTID=FLEXID=7-b2859bd4 ck=34
```

2. Checking the license is right for your machine. Invoke the lmttools program using the menu Start/Programs/HDL Turbo Writer/lmttools. Select the System Settings tab. Now open your license file and find the line that starts FEATURE Twriter. A at the end of this line, or on the next if the line ends in a \ will be a string of letters starting HOSTID= . You need to check the data following this against one of the boxes in this dialog. The box to look at depends on the type of HOSTID in your license file. If you have:

HOSTID=DISK_SERIAL_NUM=number. This is a license locked to your hard disk. Check the number against the Disk Volume Serial Number box.

Troubleshooting

HOSTID=number. This is a license locked to your network card. Check the number against the Ethernet address box.

HOSTID=FLEXID=number. This is a license locked to a parallel port dongle. Check number against the FLEXID7 and FLEXID8 boxes.

If there is a match, continue to step 3.

A mismatch and you have one of the first two license types above means you have the license for a different computer. You need to find the correct license for your machine.

If you have the HOSTID=FLEXID type, but no match in the FLEXID boxes, this indicates a problem with the dongle or its driver. Licenses of this type needs a dongle attached to the parallel port, and on Windows NT a driver loaded. An empty FLEXID7 box in the dialog indicates that the computer has not detected any 7- series dongles of the type we supply with Turbo Writer in the media pack. They are made by Rainbow and are a blue/green colour with flat sides. If the FLEXID7 box is empty and you have such a dongle connected it suggests you have Windows NT and have not loaded the NT dongle driver. See the Windows NT notes on page 5 to correct this.

Turbo Writer also supports (but we do not supply) 8 series Dallas dongles which are either black or light blue, and have a hole in the middle for a button cell. These dongles get their number reported in the FLEXID8 box and the number must match the license file. If you have one of these dongles connected but the FLEXID8 box is empty, consult the documentation supplied to you by whoever provided this dongle.

3. Check Turbo Writer knows where the license file is. The best way to do this is to enter this command into a dos box:

```
notepad %LM_LICENSE_FILE%
```

And the license file should appear in a notepad window. If it does not, then you do not have the LM_LICENSE_FILE environment variable set up correctly. Check your autoexec.bat if you are on Windows 95, or your environment settings on NT. The procedure is explained in the section called Setting Environment Variables on page 7. You have now finished debugging your node locked license.

When I start Turbo Writer I get a license error saying checkout failed. I have a floating license.

Floating Licences are where the license file contains SERVER and DAEMON lines. If your license is of this type, this is the section for you. If not read the previous section on page 51 instead.

It is assumed in this section that you have installed and started the floating license server as described on page 7.

For a floating license to work three things must be in place. The license server must be up and running, the network connection between the client and the server must be operational and the client must know where to look for the server. The following procedure should deal with all three classes of problem.

Lets start with the license server

Server diagnostics.

1. Checking the license server. On the server computer call up a command prompt window and cd to the Turbo Writer installation directory. From there enter this command:
`lmutil lmstat -a -c "path_to_license_file"`
If this reports that there are Turbo Writer licenses available and there are no errors (such as unsupported by license server) then the license server is ok and so proceed to the Client Diagnostics section on page 56.
2. Verifying the path to your license file. The license server is configured using the lmtools icon in the HDL Turbo Writer start menu. Start this program, and from the opening screen next to the planet graphic, select the Configuration Using services button. Now select the Configure Services tab and select in the Service Name list the one you expect to be running the Turbo Writer license server. Verify that the paths to the license file, lmgrd.exe and debug log all point to the correct files.
3. Verify the server name and TCP/IP name resolution. The first line of the license file should be the word SERVER followed by the name of the computer running the server. Make a note of this computer name and add it to the end of a ping command. For example my license file starts
SERVER jasmine FLEXID=7-b2859bd4 1650

So I would type

ping jasmine

The correct response to a ping command is a set of reply lines like this:

Reply from 10.24.0.3: bytes=32 time=10ms TTL=128

If you get unknown host or any timeout messages, then either the name you are using for your computer is incorrect, or there is a problem with the installation of your TCP/IP networking. You can check the computer name as follows:

Windows 2000

Right click on the my computer icon on the desktop and select properties. Select the network identification tab, and push the properties button. The computer name is displayed as the top field in the dialog box.

Windows NT 4

Right click on the Network Neighbourhood icon on the desktop and select properties. Select the identification tab, and look in the Computer Name box.

If the computer name looks correct, but the ping doesn't work, talk to your network administrator. If the computer name differs from the second word of the SERVER line in the license file, correct the license file.

4. Verifying the host id of the license server. The third word of the license file's SERVER line is used to confirm the identity of the license server. Locate this section of the license file and verify the part after HOSTID= against the information displayed in the System Settings tab in lmttools. The box you need to check against in lmttools depends on the type of your HOSTID. Here are the possibilities.

HOSTID=DISK_SERIAL_NUM=number. This validates the server computer by the volume serial number of your hard disk. Check the number against the Disk Volume Serial Number box.

HOSTID=number. This validates the server computer by the address of your network card. Check the number against the Ethernet address.

HOSTID=FLEXID=number. This validates the server computer by the dongle attached to its parallel port. Check number against the FLEXID7 and FLEXID8 boxes.

If there is a match, continue to step 5.

Troubleshooting

A mismatch and your having one of the first two license types above means you have the license for a different computer. You need to find the correct license for your server machine.

If you have the HOSTID=FLEXID type, but no match in the FLEXID boxes, this indicates a problem with the dongle or its driver. Licenses of this type need a dongle attached to the parallel port, and on Windows NT a driver loaded. An empty FLEXID7 box indicates that the computer has not detected any 7- series dongles of the type we supply with Turbo Writer in the media pack, which are made by Rainbow and are a blue/green colour. If you have such a dongle connected it suggests you have Windows NT and have not loaded the NT dongle driver. See the Windows NT notes on page 5 to correct this.

Turbo Writer also supports (but we do not supply) 8 series Dallas dongles which are either black or light blue, and have a hole in the middle for a button cell. These dongles get their number reported in the FLEXID8 box and the number must match the license file. If you have one of these dongles connected but the FLEXID8 box is empty, consult the documentation supplied to you by whoever provided this dongle.

5. Verify that there is a valid port number on the end of the server line. Your server line will look something like this:
SERVER jasmine FLEXID=7-b2859bd4 1967
The port number in this case is the 1967. Check your license file has one. It also needs to be a number unique to Turbo Writer, and less than about 30000. Generally numbers below 1000 are best avoided because many of them are used by the system. If it exists and is within this range, leave it alone for now.
6. Verifying the path to the vendor daemon. Locate the line in your license file starting "DAEMON SAROS". This needs to be followed by the full path to a file called saros.exe which can be found in the Turbo Writer installation area. A quick way to check the path is free of typos is to copy it, run up wordpad, select the file open menu, and paste the path back in. If wordpad can open the file, the path must be correct. If wordpad claims the file doesn't exist, correct the path on the DAEMON line and try again. Once wordpad can open the file, quit and continue to the next step.
7. Verifying the feature line. The end of the feature line contains a checksum which can be used to verify the line against typing errors or corruption from the email system. To verify the line using the checksum run up the lmttools program, and select the Utilities tab. Push the Perform Check Sum button. The window shows the results of check summing each feature line. Check there is an OK at the start of the

Twriter feature line. If instead of OK you get BAD, get the license sent again from Saros Technology Limited.

8. By now the license should be correct, and in the correct place. Its time to restart the license server. From lmttools application, select the Service/License File tab and select the Configuration Using Services button. Then select the Start/Stop/Reread tab. Click the Stop Server button repeatedly until the status line at the bottom of the screen displays “Unable to Stop Server”. Then click the Start Server button.
9. Rerun the test in step 1 of this section. The command should return stating that there are Turbo Writer licenses available. If you still get an error call Saros Technology using the contact details near the start of this manual, and ask for Technical support. We will find out what is wrong with the license and update this procedure.
If the command succeeds try invoking Turbo Writer again on the client machine. If the program fails to start, proceed through the client debug section that follows.

Client Diagnostics

This section is useful if the license server appears to be working, but you still cannot get a license at the client machines. It is assumed that the test in step 1 in the section above has been tried and indicated that licenses are available. With a floating license configuration, there is very little that needs to be set at the client computers. Only two things can go wrong, either the client doesn't know how to access the license server, or network link from the client to the server is not configured correctly.

1. Verify the client knows how to access the server. Inspect the license file on the server and find the SERVER line. It will look something like this:

```
SERVER jasmine FLEXID=7-b2859bd4 1650
```

The second word is the name of the license server, and the number at the end of the line is the port number. Start a command prompt window and run this command in it:

```
echo %LM_LICENSE_FILE%
```

This will return the value of the LM_LICENSE_FILE environment variable, and is the only reliable way to do it, particularly on Windows NT. It should contain either a path to the license file, or a string

of the form `port_number@server_name`. In the latter case, the server line shown above would translate to a `LM_LICENSE_FILE` of `1650@jasmine`. If a path is used, verify the path by opening it with notepad. In the case of the `port_number@server`, check the details against the `SERVER` line in the license file. If you find any errors, change the setting of the environment variable, as described in the “Setting environment variables” on page 7

2. Verify the network is configured correctly. Run up a command prompt and type `ping name_of_your_license_server`
So in the case of the `SERVER` line above, the command would be “ping jasmine”
This should return a list of reply times, like this

```
Reply from 10.24.0.3: bytes=32 time<10ms TTL=128
```

Anything else means your TCP/IP networking is not correctly configured. Contact your network administrator to get this fixed.

Once you can ping the license server, and the `LM_LICENSE_FILE` is set correctly, you should be able to run Turbo Writer. If you are still experiencing problems, please contact Saros Technology Limited using the contact details near the start of this manual.

Folding provides inconsistent results and sometimes seems to hang the machine.

Check the type of files you are editing. If they are Unix files the folding software can exhibit this problem. Select the Document/Manager menu, click the EOL/EOF tab and check if the Unix EOL option is selected. If it is, then change the dialogue and reload the file. The fold function should now work correctly.

The Editor seems to slow down when I'm editing large files. Can anything be done to speed this up?

Codewright allocates a fixed amount of RAM when it initialises. When using large files, memory is swapped to the disk. If you have a sizeable amount of RAM then the default allocation can be increased which can considerably speed up large file handling. The following can be added to your **cwright.ini** file.

```
[editor]
```

```
  SysSwapBlocks=100
```

The **SysSwapBlocks** command allocates a number of 8k blocks of memory. The default is 20.

The Model Technology Interface does not seem to work.

Clicking the Rhino button on the toolbar should compile the current VHDL file. If this does not happen check each of the following points.

- The Turbo Writer and ModelSim directories must both be in your path.
- The VsystemSetup line in the [vhdl] section of the cwright.ini file must contain the path of the directory containing Model Technologies ModelSim, i.e. the directory holding vsystem.exe.
- The executable vcom.exe must be present in the ModelSim directory. If ModelSim was installed under 3.11 this file will not be present, and ModelSim must be reinstalled under Windows 95 or NT.

I don't have any horizontal scroll bars. How do I enable them?

From the Tools menu choose Customize/View Setup. Enable the Horizontal scrollbar checkbox.

Appendix A : Verilog Keywords and Templates

The following are tables of the Verilog Keywords used for Colour Coding.

Table 1. Basic Verilog Keywords

always	And	assign	begin	buf	bufif0
bufif1	Case	casex	casez	cmos	deassign
default	Defparam	disable	edge	else	end
endcase	Endfunction	endmodule	endprimitive	endspecify	endtable
endtask	Event	for	force	forever	fork
function	Highz0	highz1	if	initial	inout
input	Integer	join	large	macromodule	medium
module	Nand	negedge	nmos	nor	not
notif0	Notif1	or	output	pmos	posedge
primitive	Pull0	pull1	pulldown	pullup	rcmos
reg	Release	repeat	rnmos	rpmos	rtran
rtranif0	Rtranif1	scalered	small	specify	specparam
strong0	Strong1	supply0	supply1	table	task
time	Tran	tranif0	tranif1	tri	tri0
tri1	Triand	trior	vectored	wait	wand
weak0	Weak1	while	wire	wor	xnor
xor					

Table 2. Verilog System Task and System

\$bitstoreal	\$countdrivers	\$display	\$fclose	\$fdisplay
\$fclose	\$fdisplay	\$fmonitor	\$fopen	\$fstrobe
\$fwrite	\$finish	\$getpattern	\$history	\$sincsave
\$input	\$itor	\$key	\$list	\$log
\$monitor	\$monitoroff	\$monitoron	\$nokey	\$nolog
\$printtimescale	\$readmemb	\$readmemh	\$realtime	\$realtobits
\$reset	\$reset_count	\$reset_value	\$restart	\$rtoi
\$save	\$scale	\$scope	\$showscopes	\$showvariables
\$showvars	\$sreadmemb	\$sreadmemh	\$stime	\$stop
\$strobe	\$time	\$timeformat	\$write	

Table 3. Verilog Compiler Directive Keywords

'accelerate	'autoexpand_vectornets	'celldefine
'default_nettype	'define	'else
'endcelldefine	'endif	'endprotect
'endprotected	'expand_vectornets	'noremove_gatenames
'noremove_netnames	'nouncconnected_drive	'protect
'protected	'remove_gatenames	'remove_netnames
'resetall	'timescale	'unconnected_drive

Table 4. Verilog Template Definitions

Template Name	Expanded Template	Template Name	Expanded Template
Al	always @()	em	endmodule //
Alb	always @() begin end //always	fe	forever ()
An	always @(negedge)	feb	forever () begin end //forever
Anb	always @(negedge) begin end //always	Fk	fork join
Ap	always @(posedge)	Fo	for (;;) begin end //for
Apb	always @(posedge) begin end //always	Fu	function ; endfunction //
As	assign	If	if ()
Asb	assign begin end //assign	ifb	if () begin end
Ca	case () endcase //case	in	input
Cx	casex () endcase //casex	ini	initial
Cz	casez() endcase //casex	inib	initial begin end
De	deassign	io	inout
Eb	else begin end	mo	module (); endmodule //
Ec	endcase	ou	output
Ei	else if ()	re	reg
Eib	else if () begin end		

Template Name	Expanded Template	Template Name	Expanded Template
Rep	repeat ()	wh	while ()
Repb	repeat () begin end //repeat	whb	while () begin end //while
Ta	task ; begin end endtask	wi	wire

Table 5. Verilog Compiler Directive Template Definitions

Template Name	Expanded Template
`ac	`accelerate
`au	`autoexpand_vectornets
`ce	`celldefine
`de	`define
`defa	`default_nettype
`endc	`endcelldefine
`ex	`expand_vectornets
`if	`ifdef `else `endif
`in	`include
`noac	`noaccelerate
`noun	`nounconnected_drive
`res	`resetall
`ti	`timescale /
`un	`unconnected_drive

Appendix B : VHDL Keywords and Templates

The following are tables of the VHDL Keywords used for Colour Coding.

Table 1.

abs	Access	after	alias	all	and
architecture	Array	assert	attribute	begin	block
body	Buffer	bus	case	component	configuration
constant	Disconnect	downto	else	elsif	end
entity	Exit	file	for	function	generate
generic	Group	guarded	if	impure	in
inertial	Inout	is	label	library	linkage
literal	Loop	map	mod	nand	new
next	Nor	not	null	of	on
open	Or	others	out	package	port
postponed	Procedure	process	pure	range	record
register	Reject	rem	report	return	rol
ror	Select	severity	signal	shared	sla
sll	Sra	srl	subtype	then	to
transport	Type	unaffected	units	until	use
variable	Wait	when	while	with	xnor
xor					

Table 2. Std 1164 Keywords

is_x	resolved	rising_edge	to_bit
to_bitvector	to_stdulogic	to_stdulogicvector	to_stdlogicvector
to_x01	to_x01z	to_ux01z	std_ulogic
std_ulogic_vector	std_logic	std_logic_vector	x01
x01z	ux01	ux01z	

Table 3. VITAL 2.2b Keyword

Delaytypexx	delaytype01	delaytype01z
Glitchkind	pathype	patharraytype
Transitiontype	transitionarraytype	timearray
Timinginfotype	vitalpropagatewiredelay	vitaltimingcheck
Vitalsetupholdcheck	vitalreportsetupholdviolation	vitalreportrtlsermvlviolation
Vitalperiodcheck	vitalextendtofilldelay	vitalcalcdelay
Vitalglitchonevent	vitalglitchondetect	vitalpropagatepathdelay
Vitaland	vitaland2..4	vitalnand
vitalnand2..4	vitalor	vitalor2..4
Vitalnor	vitalnor2..4	vitalxor
vitalxor2..4	vitalxnor	vitalxnor2..4
Vitalbuf	vitalbufif0	vitalbufif2
Vitalinv	vitalinvif0	vitalinvif2
Vitalmux	vitalmux2..4	vitaldecoder
vitaldecoder2	vitaldecoder4	vitaldecoder8
Vitalident	vitaltruthtable	vitalstatetable
Violationtype		

Table 4. VHDL Template Definitions

Template Name	Expanded Template
ab	Abs
ac	access
af	After
ai	alias
al	all;
an	and
ar	architecture <arc_name> of <ent_name> is
	begin
	end <arc_name>;
as	assert <condition>
	[severity <severity_type>]
	[report <report_type>]
at	attribute
be	begin
	end
bl	<block_name> : block
	end block <block_name>;
bo	body
bs	bus
bu	buffer
ca	case () is
	when =>
	when others =>
	end case;
cf	configuration
cn	constant : ;
co	component <comp_name>
	port (
);
	end component;
di	disconnect
do	downto
el	elsif () then
en	entity <ent_name> is
	end <ent_name>;
ex	exit
fd	function <fun_name> () return <ret_type>;
fg	<label> : for <var> in <range> generate
	end generate;
fi	File

Table 4. VHDL Template Definitions cont.

Template Name	Expanded Template
Fu	function <fun_name> () return <ret_type> is begin end <fun_name>;
Ge	generic ();
Gn	generate
Gr	group
Gu	guarded
If	if () then end if;
Ig	<label> : if <condition> generate end generate;
Im	impure
In	integer
Io	inout
Ir	inertial
La	label
Li	library ;
Lk	linkage
Lo	loop
Lt	literal
Lw	<label> : while <condition> loop end loop <label>;
Ma	map
Mo	mod
Ne	new
Nu	null
Nx	next
Op	open
Ot	others
Pa	package <name> is end <name>;
Pb	package body <name> is end <name>;
Pd	procedure <name> () is begin end <name>;
Pdd	procedure <name> ();
Po	port ();
Pp	postponed
Pr	procedure

Table 4. VHDL Template Definitions cont.

Template Name	Expanded Template
Ps	<name> : process () begin end process <name>;
Pu	pure
Ra	range
Re	record
Re	register
Rj	reject
Rm	rem
Rp	report
Rt	return
Se	severity
Sh	shared
Si	signal ;
Sl	select
St	std_logic
Su	subtype
Sv	std_logic_vector
Th	then
Tr	transport
Ty	type
Ua	unaffected
Un	units
Us	use ;
Ut	until
Va	variable ;
Wa	wait until (<clock>'event and <clock>='1');
Wh	when =>
Wi	with
Wl	while