

# DK2

---

## Waveform Analyzer Manual

Celoxica, the Celoxica logo and Handel-C are trademarks of Celoxica Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous development and improvement. All particulars of the product and its use contained in this document are given by Celoxica Limited in good faith. However, all warranties implied or express, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Celoxica Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

The information contained herein is subject to change without notice and is for general guidance only.

Copyright © 2003 Celoxica Limited. All rights reserved.

Authors: SB

**Document number: UM-2010-1.2**

# Table of contents

---

## Table of contents

TABLE OF CONTENTS .....	1
CONVENTIONS .....	3
ASSUMPTIONS .....	4
OMISSIONS .....	4
1. OVERVIEW .....	5
1.1.1 Starting the Waveform Analyzer .....	5
1.1.2 Waveform Analyzer file formats .....	5
1.1.3 Time units in the Waveform Analyzer .....	6
1.2 CONNECTING A SIMULATION TO A TRACE .....	6
1.3 CONNECTING A PATTERN TO A PORT .....	8
1.4 CONNECTING IN PARALLEL .....	10
1.5 MEASURING TIME AND VALUE DIFFERENCES IN WINDOWS .....	10
1.6 FINDING A SEQUENCE OF DATA IN A TRACE OR PATTERN .....	11
1.7 GROUPING TRACE AND PATTERN WINDOWS .....	11
1.8 GENERATING PATTERNS .....	12
1.9 USING TRIGGERS .....	12
2. PATTERN GENERATION LANGUAGE .....	14
2.1 USING THE PATTERN GENERATION LANGUAGE .....	14
2.2 PGL STATEMENTS .....	15
2.3 FUNCTIONS IN PGL .....	16
2.4 WILD-CARD MATCHING IN PGL .....	16
2.5 PATTERN GENERATION LANGUAGE SYNTAX .....	17
3. WAVEFORM ANALYZER INTERFACE .....	20
3.1 WAVEFORM ANALYZER TOOLBAR ICONS .....	20
3.2 TRACE/PATTERN WINDOW .....	21
3.3 MENUS .....	22
3.3.1 File Menu .....	22
3.3.2 New window dialog .....	23
3.3.3 Edit menu .....	23
3.3.4 View menu .....	24
3.3.5 Trace Menu .....	24
3.3.6 Trace Properties dialog .....	24
3.3.7 Pattern menu .....	26
3.3.8 Pattern Properties dialog .....	26
3.3.9 Define symbols dialog .....	27
3.3.10 Script menu .....	28
3.3.11 Capture menu .....	28

# Table of contents

---

3.3.12 Set Advance Step dialog .....	28
3.3.13 Window menu.....	28
3.3.14 Help menu .....	29
<b>I NDEX.....</b>	<b>I</b>

# Conventions

---

## Conventions

A number of conventions are used in this document. These conventions are detailed below.



Warning Message. These messages warn you that actions may damage your hardware.



Handy Note. These messages draw your attention to crucial pieces of information.

Hexadecimal numbers will appear throughout this document. The convention used is that of prefixing the number with '0x' in common with standard C syntax.

Sections of code or commands that you must type are given in typewriter font like this:

```
void main();
```

Information about a type of object you must specify is given in italics like this:

```
copy SourceFileName DestinationFileName
```

Optional elements are enclosed in square brackets like this:

```
struct [type_Name]
```

Curly brackets around an element show that it is optional but it may be repeated any number of times.

```
string ::= "{ character }"
```

# Assumptions

---

## Assumptions

This manual assumes that you:

- have used Handel-C or have the Handel-C Language Reference Manual
- are familiar with common programming terms (e.g. functions)
- are familiar with MS Windows

## Omissions

This manual does not include:

- instruction in VHDL
- instruction in the use of place and route tools
- tutorial example programs. These are provided in the Handel-C User Manual

# 1. Overview

---

## 1. Overview


The Waveform Analyzer connects to ports in Handel-C simulations. It displays outputs from Handel-C simulations as waveforms (traces). You can generate inputs to Handel-C simulations and display them as waveforms (patterns). You can manipulate the simulated inputs and outputs in the same way that input and output signals from a real piece of hardware can be manipulated with a logic analyzer.

You can perform the following tasks:

- Connect traces to output ports in Handel-C simulations.
- Connect patterns to input ports in Handel-C simulations.
- Connect the Waveform Analyzer to ports connected to another simulation using the DKShare plugin (connecting in parallel).
- Measure the differences between values and times in traces or patterns, using cursor marks.
- Create patterns by writing scripts using Pattern Generation Language (PGL), or by copying existing traces or patterns into a pattern window. Patterns can also be read from a file.
- Specify triggers in the Pattern Generation Language.
- Capture traces or generate patterns when a specified trigger appears in a trace.
- Find a specified pattern in a trace or pattern window.

### 1.1.1 Starting the Waveform Analyzer

To start the Waveform Analyzer choose one of the following:

- Select **Start>Programs>DK Design Suite>Waveform Analyzer**
- Double-click the icon for the analyzer.exe file  in the **DK\Bin** directory
- Double-click on a Waveform Analyzer project (.apj)
- Double-click on a trace (.trc) or pattern (.pat) file

### 1.1.2 Waveform Analyzer file formats

Waveform Analyzer supports 2 different file formats for storing waveform data. These are:

- ASCII files, where data elements are written in ASCII and separated by white space.
- Value Change Dump (VCD) files. This file format is specified in the IEEE 1364-1995 standard.

# 1. Overview

---

When you create a new trace, the extension you type in the **Dump file** box in the **Trace Properties** dialog (**Trace>New**) determines whether a VCD or an ASCII file is produced. If the extension is .vcd, .dmp or .dump the file is a Value Change Dump file, otherwise it is an ASCII file. Similarly, Pattern source files with the extensions .vcd, .dmp or .dump are recognised as VCD files, else the file is assumed to be an ASCII file.

## Saving traces in a VCD file

A VCD file can contain any number of variables. If you need to dump several traces to the same VCD file, enter the same VCD filename in the **Dump File** box on the **Trace Properties** dialog for every trace that should be written to the file. The **Variable** box in the **Trace** dialog is used to enter a reference name which will be used in the VCD file for the signal. All variables are output to a single top-level scope called "top".

## Reading patterns from VCD files

When reading a pattern from a VCD file, you need enter the reference name of the variable in the VCD file which needs to be read in the **Variable** box in the **Pattern** dialog. This name must specify the scope in which the variable occurs and the reference name of the variable within that scope. Scopes are terminated using the '.' character. For example, mai n. scope1. x refers to the variable x in the scope scope1 which is a subscope of mai n.

### 1.1.3 Time units in the Waveform Analyzer

Time units are not explicitly defined in the Waveform Analyzer. Any Handel-C simulation to which you connect the Waveform Analyzer should use the DKSynC plugin with the clock period for the simulation specified in an exti nst string. When you enter the clock period for a trace or pattern window, you determine the sample rate for the trace or patterns in the window relative to the clock period specified for the Handel-C simulation. If the clock period for the trace or pattern is the same as the clock period specified in the exti nst string in the Handel-C program, the trace or pattern will be sampled on every cycle of the Handel-C program. If the clock period for the trace or pattern is twice the clock period specified in the exti nst string in the Handel-C program, the trace or pattern will be sampled on every other cycle of the Handel-C program and so on. It is a matter of convenience to make the clock periods that you use correspond to the clock periods that will be used in the target hardware.

The VCD file reader/writer used by the Waveform Analyzer assumes that the time units used are nanoseconds.

## 1.2 Connecting a simulation to a trace

To connect a simulation to a trace, you must

1. Write and compile Handel-C code to connect a Handel-C port to a terminal using the DKConnect and the DKSynC plugins.
2. Set up a trace window in the analyzer which reads the signal from the named terminal.
3. Simulate the Handel-C code and start capturing.



# 1. Overview

---

## Sample Handel-C program

Open Handel-C, create a new project and enter the following program:

```
set clock = external "P1"
with {extlib = "DKSync.dll", extinst = "50", extfunc = "DKSyncGetSet"};

unsigned 3 x = 0;

interface bus_out() ob1(unsigned 3 out = x)
    with {extlib = "DKConnect.dll", extinst = "t(3)", extfunc =
        "DKConnectGetSet"};

void main(void)
{
    while(1) x++;
}
```

Now compile the program but do not run it.



This program uses the DKConnect plugin to connect the port ob1.out to the terminal t(3).

## Setting up a Trace window

1. Open the Waveform Analyzer.
2. Select **New** from the **File** menu and create a new trace.
3. Select the browse button to specify a filename and location of your choosing.
4. Set **Default Clock Period** to 50 and **Default No. points** to 40.  
An empty Trace window should appear.
5. Select **New Trace** from the **Trace** menu or from the toolbar and enter the following properties in the dialog box:



<b>Name:</b>	testtrace
<b>Width:</b>	3
<b>Type:</b>	Unsigned
<b>Expression:</b>	t(3)
<b>Dump File:</b>	Leave blank
<b>Variable:</b>	Greyed out
<b>Trigger:</b>	Leave box blank. Other settings should be greyed out.
<b>Delay:</b>	Greyed out with 0 as default
<b>Display:</b>	Check the <b>Stepped Waveform</b> radio button.

6. Click **OK**.

# 1. Overview

---

## Start capturing

- Start capturing by clicking the Run icon  on the toolbar, or by selecting **Run** from the **Capture** menu. A red dashed line should appear (jumping around all over the place). This line marks the current position in the trace.
- Run the Handel-C simulation.
- To stop capturing click on the stop button  on the toolbar or select **Stop** from the **Capture** menu. You must also stop the Handel-C simulation.

## 1.3 Connecting a pattern to a port

To connect a pattern to a port, you must

1. Write and compile Handel-C code to connect a Handel-C port to a terminal using the DKConnect and the DKSynC plugins.
2. Set up a pattern window in the analyzer generating a signal to the named terminal.
3. Simulate the Handel-C code and start transmitting the pattern.

### Writing the Handel-C program

Open Handel-C, create a new project and enter the following program:

```
set clock = external "P1"
with {extlib = "DKSync.dll ", extinst = "50", extfunc = "DKSyncGetSet"};

interface bus_in(unsigned 1 in) ib1()
  with {extlib = "DKConnect.dll ", extinst = "t(1)",
    extfunc = "DKConnectGetSet"};

unsigned 5 count = 0;
```

# 1. Overview

---

```
void main(void)
{
    while(!count[4] || !count[2])
    {
        if (ib1.in == 0)
        {
            delay;
            if (ib1.in == 1)
                count++;
        }
        else
            delay;
    }
}
```

Now compile the program but do not run it.



This program uses the DKConnect plugin to connect the port `ib1.in` to the terminal `t(1)`. The program will only terminate when it has detected 20 rising edges from the port `ib1.in`.

## Setting up a Pattern Window



1. Open the Waveform Analyzer.
2. Select **New** from the **File** menu and create a new pattern with a filename of your choosing, with 40 as the number of points and 50 as the clock period. An empty Pattern window should appear.
3. Select **New Pattern** from the **Pattern** menu or from the toolbar and enter the following properties in the dialog box:

<b>Name:</b>	testpattern
<b>Width:</b>	1
<b>Type:</b>	Unsigned
<b>Source:</b>	Select <b>Script</b> radio button. Enter <code>loop(20) {0; 1; }</code> in the box.
<b>Variable:</b>	Greyed out
<b>Destination:</b>	<code>t(1)</code>
<b>Trigger:</b>	Leave box blank. Other settings should be greyed out.
<b>Delay:</b>	Greyed out with 0 as default
<b>Display:</b>	Check <b>Stepped Waveform</b> radio button
4. Click **OK**.

# 1. Overview

---

## Starting transmission

1. Run the Handel-C simulation.
2. Start transmission by clicking the Run icon  on the toolbar, or by selecting **Run** from the **Capture** menu. The Handel-C program should terminate shortly after transmission is started.
3. To stop capturing click on the stop icon  on the toolbar or select **Stop** from the **Capture** menu.

## 1.4 Connecting in parallel

If you wish to connect the Waveform Analyzer to ports that are connected to other plugins, you may do so using the DKShare. dll .

### Example

```
interface bus_out() seg7_output(unsigned 7 output1 = encode_out)
with {extlib="DKShare.dll",
     extinst = \
       "Share={extlib=<7segment.dll>, extinst=<A>, extfunc=<PluginSet>}"
       "Share={extlib=<DKConnect.dll>, extinst=<SS(7)>,
               extfunc=<DKConnectGetSet>}" \
     extfunc="DKShareGetSet"
};
```

This example uses DKShare.dll to share the output port seg7\_output.output1 between the 7-segment display and DKConnect (connected to terminal SS(7)). You can then trace the output going to the 7-segment display by using SS(7) as the expression in the **Trace properties** window.

## 1.5 Measuring time and value differences in windows

You can measure the time between two events and the difference in the value of a signal at two different times by placing marks in Trace and Pattern Windows. These marks are represented by coloured triangles and will be referred to as cursors. One cursor is always selected.


The cursor triangles are placed in the time pane of the trace or pattern window. If there is more than one cursor in the time pane, the time pane displays the differences in time between cursors. The bottom-centre pane displays the absolute position in time of all cursors. The differences in values between the cursors are displayed in the bottom-left pane.

If multiple traces or patterns are displayed in a window, the values given are those of the selected trace or pattern.

# 1. Overview

---

## Creating cursors

- Click on the New cursor icon  on the toolbar or select **New Cursor** from the **View** menu.  
The cursor will be added to the centre of the time pane of the active trace or pattern window.


## Moving cursors

- Drag the cursor across the time pane

## Selecting cursors

- Double-click a cursor. A red bar will appear beneath it to show that it is selected. By default, the first cursor created is the selected cursor. Only one cursor can be selected at a time.

## Deleting cursors

- Select the cursor you wish to delete.
- Click the Delete Cursor icon  on the toolbar or select Delete Cursor from the **View** menu.

## 1.6 Finding a sequence of data in a trace or pattern

To find a sequence of data in a trace or pattern:

1. Activate the window containing the trace or pattern you wish to search.
2. Click on the trace or pattern you want to search.
3. Select **Edit>Find**.
4. Enter a PGL statement or function in the **Find what:** box in the **Find** dialog.

## 1.7 Grouping trace and pattern windows

You can group Trace Windows and Pattern Windows together into projects. Only one project may be open at a time.

You must create a project if you want to use a Pattern Generation Language script file.

### Creating a project

Open the Waveform Analyzer and select **New Project** from the **File** menu.

A dialog box will appear asking you to select a file name for the new project. Project filenames have an .apj extension.

# 1. Overview

---

## 1.8 Generating patterns

### Generating a pattern from an existing trace or pattern

You can copy data from a trace or pattern into the clipboard, and then paste the contents of the clipboard into a pattern.

1. Drag the mouse pointer over a region of a trace or pattern to select it.
2. Copy the region to the clipboard by selecting **Copy** from the **Edit** menu or with the **Copy** icon on the toolbar.
3. Make a pattern window active and either select a region to paste over or select a cursor.
4. Select **Paste** from the **Edit** menu or click on the **Paste** icon on the toolbar. If you select a region, the clipboard contents are pasted into the selected pattern starting at the beginning of the selected region. If you select a cursor, the clipboard contents are pasted into the window starting at the selected cursor .

### Generating a pattern from a PGL statement:

1. Select **Script** as the pattern source in the **Pattern Properties** dialog.
2. Enter the PGL statement or function call in the box to the right of the button.

### Generating a pattern from a file:

1. Select **File** as the pattern source in the **Pattern Properties** dialog.
2. Enter the filename in the box to the right of the button. Use the **Browse** button to browse for a file

### Pattern generation limitations

It is an error to use the '?' expression, expression statements starting with '!' and assert statements in PGL programs that are used to generate patterns.

### Complex pattern generation

If you need to generate more complex patterns than those which can be generated with PGL, you should consider writing a separate Handel-C program to perform your pattern generation.

## 1.9 Using triggers

You can specify a sequence of data that you want to use as a trigger or re-use an existing specification.

When the trigger sequence occurs you can:

- Start capturing a trace before, at or after the specified trigger.
- Start generating a pattern at or after the specified trigger.
- Pause simulations once a trace has been captured or a pattern has been generated.

# 1. Overview

---

## To specify a trigger:

1. Open the **Pattern** or **Trace Properties** dialog.
2. Enter ***trace name***: in the **Trigger** box followed by a PGL statement. ***trace name*** is the name of a pre-defined trace (note that it must be followed by a colon). The PGL statement will be matched against the named trace. For example:  
b : {0; 1; }  
would cause the pattern to be generated on a rising edge of trace b.
3. Select the appropriate radio button:
  - No trigger:** No triggering.
  - Single:** Transmit or capture the first time the trigger is received.
  - Auto:** Transmit or capture each time the trigger is received.

## To re-use a specified trigger:

1. Open the **Pattern** or **Trace Properties** dialog.
2. Enter "***name***" in the **Trigger** box, where ***name*** is the name of the trace or pattern that uses a trigger. Note that ***name*** must be preceded by a double-quote.  
For example:  
"trace1  
would cause the trace or pattern whose details are being entered to use the same trigger as the trace named trace1.
3. Select the appropriate radio button:
  - No trigger:** No triggering.
  - Single:** Transmit or capture the first time the trigger is received.
  - Auto:** Transmit or capture each time the trigger is received.

## To specify the delay between the trigger and the action

1. Open the **Pattern** or **Trace Properties** dialog.
2. Specify a trigger.
3. Enter the number of time units in the **Delay** box on the **Properties** dialog. The delay is in the time units for that window. Delays can be positive or negative for a trace, (negative delays capture before the trigger, positive after) and positive or zero for a pattern.

## To pause on trigger

1. Open the **Pattern** or **Trace Properties** dialog.
2. Specify a trigger.
3. Check the **Pause** box.

## 2. Pattern Generation Language

---

## 2. Pattern Generation Language

The Waveform Analyzer uses Pattern Generation Language (PGL) as a scripting language to generate patterns. PGL has a similar expressive power to regular expressions, but uses a C-like syntax.

You can use it to trigger on a sequence of data and search for a sequence of data in a trace or pattern.

When executed, a PGL program generates a sequence of values. You can use a PGL program to trigger or search for any pattern that could be generated by passing parameters to the program.

### 2.1 Using the Pattern Generation Language

You can use the Pattern Generation Language (PGL) to:

- Generate patterns that are fed into a port.
- Identify a sequence of data in a trace that you wish use as a trigger. You can use the trigger to start recording the trace or to start generating a pattern. If you have defined a trigger associated with a trace or pattern, you may re-use it as a trigger for other traces or patterns.
- Find a sequence of data in a trace or a pattern.

#### Entering PGL statements

When you need to enter a PGL statement in a dialog box, you can enter any statement, including compound statements and function calls. For example, the following are all valid PGL statements:

- `2;`
- `{0; 1}`
- `rising_edge(); //valid if the rising_edge function has been defined`

But these are not:

- `2 //missing semi-colon`
- `0; 1; //curly brackets must surround compound statements`

You can write PGL functions in the `script.pgl` file associated with a project.

#### Complex pattern-matching and pattern-generation

If you need to generate more complex patterns than those that can be generated with PGL, or if you need to trigger efficiently on a complex sequence of words, consider writing a separate Handel-C program to perform your pattern generation or pattern matching. For triggering, you could output a trigger signal from this Handel-C program to Waveform Analyzer, and then use a simple PGL statement to trigger on this signal.



## 2. Pattern Generation Language

---

### 2.2 PGL statements

You can use any of the following statements as a PGL statement:

#### Expression statement

For example: `1;`

When an expression statement is executed, it generates the value of the expression.

An expression statement used for matching may also be of the form: `!1;`

This statement will match any value except 1.

#### Compound statement

For example: `{0;1;}`

The statements enclosed in the braces get executed sequentially.

#### Loop statement

For example: `loop(3) {0;1;}`

The body of this loop will get executed 3 times.

#### Conditional statement

For example: `if (a==1) {0;1;} else {1;0;}`

Here, the statements that get executed depend upon the value of the variable a.

You can build Boolean tests using the following operators:

`==`      `!=`      `!`      `||`      `&&`

#### Switch statement

For example:

```
switch(a)
{
    case 1:
        0; 1; break;
    default:
        1; 0; break;
}
```

This switch statement achieves the same thing as the conditional statement described above.

#### Assert statement

For example: `assert(a != 0);`

This kind of statement can be used to place constraints on matched variables when matching.

## 2. Pattern Generation Language

---

### 2.3 Functions in PGL

PGL allows you to define and call functions which can take parameters.

You can only define functions in an open project. The functions are stored in the `script.pgl` file associated with that project. You may edit this file outside the Waveform Analyzer.

#### Defining functions

1. Open the project in which you want to use the functions.
2. Click on the **Edit Script** icon on the toolbar or select **Edit Script** from the **Script** menu. This opens `script.pgl` in Notepad. This file resides in the same directory as the project file.
- 3.

#### Example

The following example defines two functions, one called `rising_edge` and the other called `rectangular_wave`.

```
rising_edge()  
{0; 1; }
```

```
rectangular_wave(hi_val , hi_count , lo_val , lo_count , cycles)  
{  
  loop (cycles)  
  {  
    loop(hi_count)  
      hi_val ;  
    loop(lo_count)  
      lo_val ;  
  }  
}
```

You can call the `rectangular_wave` function with a statement like this:

```
rectangular_wave(1, 5, 0, 5, 10);
```

### 2.4 Wild-card matching in PGL

If you use a PGL program for triggering or searching, you can have a '?' character in any place where a number or variable could go. This character stands for 'any value'. For example the compound statement `{ 1; ?; 1; }` would match against any 3 word sequence starting and ending with a 1.

If '?' is used as a parameter in a function call, when the function is called, the formal parameter which corresponds to the '?' has no value assigned to it.

## 2. Pattern Generation Language

---

If a variable is encountered which has no value assigned to it, it gets assigned a value according to the values encountered during matching.

Context-sensitive matches can be carried out in this way. For example, if a function is defined as follows:

```
count_fives(a)
{a; loop(a) 5; }
```

and called using the statement 'count\_fives(?);', it will match any sequence consisting of a number, followed by that number of fives (including the sequence '0').

This feature should be used carefully, since it is possible to use it write functions which take a very long time to match.



Do not use the '?' expression, expression statements starting with '!' or assert statements in PGL programs; these are used to generate patterns.

### 2.5 Pattern Generation Language syntax

```
subprogram_def ::= identifier ( [parameter-list] )
                    compound-statement
```

```
parameter-list ::= identifier
                    | identifier , parameter-list
```

```
statements ::= statement
              | statement statements
```

```
statement ::= subprogram_call
              | compound-statement
              | loop-statement
              | if-statement
              | if-else-statement
              | switch-statement
              | break-statement
              | expression-statement
              | assert-statement
```

```
subprogram_call ::= identifier ( [expression-parameter-list] ) ;
```

```
expression-parameter-list ::= expression
                               | expression , expression-parameter-list
```

## 2. Pattern Generation Language

---

*compound-statement* ::= { *statements* }

*loop-statement* ::= loop ( *expression* ) *statement*  
                   | loop forever *statement*

*if-statement* ::= if ( *boolean-expression* ) *statement*

*if-else-statement* ::= if ( *boolean-expression* ) *statement* else *statement*

*switch-statement* ::= switch ( *expression* ) { *case-list* [default :  
   *statements* ] }

*case-list* ::= case  
               | case *case-list*

*case* ::= case number : *statements*

*break-statement* ::= break ;

*expression-statement* ::= *expression* ;  
                           | ! *expression* ;

*assert-statement* ::= assert ( *boolean-expression* );

*boolean-expression* ::= *expression* == *expression*  
                           | *expression* != *expression*  
                           | *expression*  
                           | *boolean-expression* && *boolean-expression*  
                           | *boolean-expression* || *boolean-expression*  
                           | ! *boolean-expression*  
                           | ( *boolean-expression* )

(here, && has higher precedence than || and both are left associative)

## 2. Pattern Generation Language

---

*expression* ::= ?

- | number
- | identifier

Numbers must be binary, octal, decimal or hexadecimal integers, and use the same syntax as Handel-C. (i.e. 0b... for binary numbers, 0... for octal numbers, 0x... for hex numbers; all other numbers are treated as decimals).

Identifiers are C-style identifiers.

## 3. Waveform Analyzer interface

---



























### 3. Waveform Analyzer interface

The Waveform Analyzer interface consists of:

- A menu bar
- A toolbar
- A workspace area
- Any trace or pattern windows you have open
- A log output window: used by the program to report errors to the user.

You may group trace or pattern windows together in a project. Projects contain a number of trace or pattern windows (those open when you last saved the project) and any scripts you have written in the project.

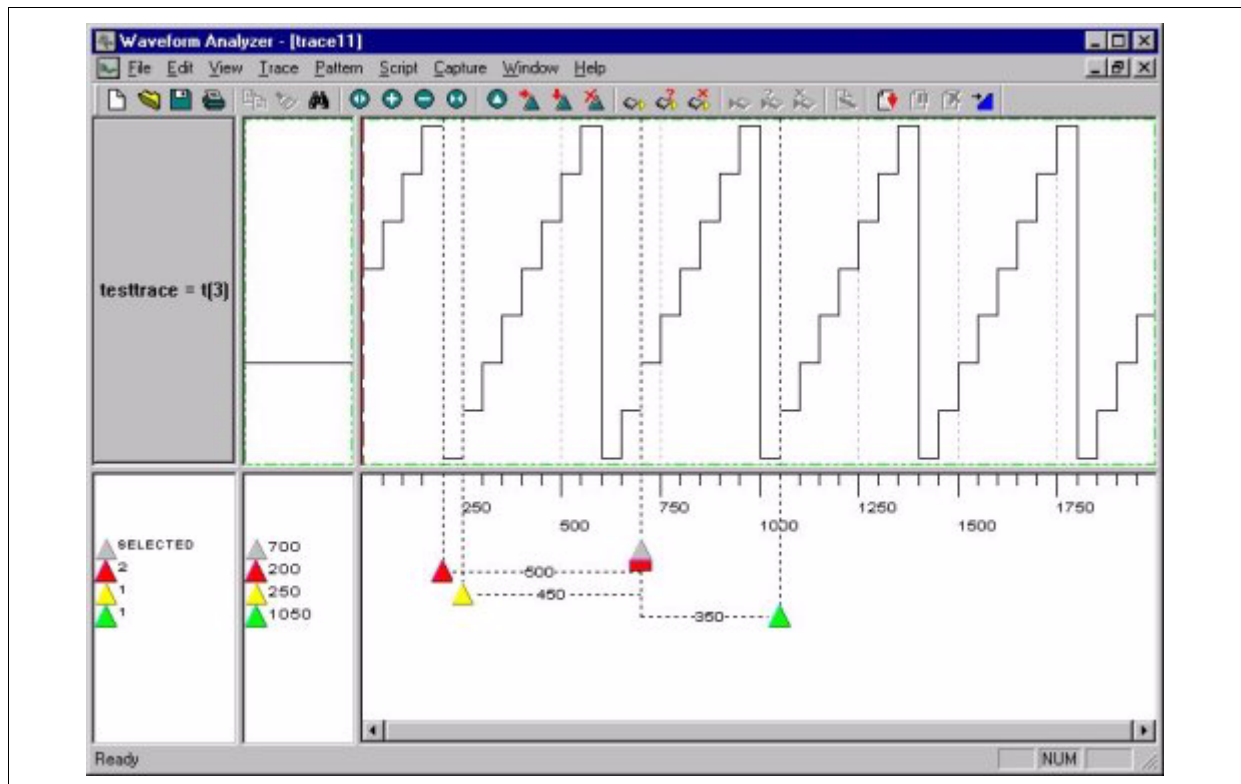
#### 3.1 Waveform Analyzer toolbar icons

	New		New trace
	Open		Edit trace
	Save		Delete trace
	Print		New pattern
	Copy		Edit pattern
	Paste		Delete pattern
	Find		Edit script
	Zoom max		Run
	Zoom in		Pause
	Zoom out		Stop
	Zoom min		Advance
	Zoom on cursor		
	Jump to cursor		
	New cursor		
	Delete cursor		

## 3. Waveform Analyzer interface

### 3.2 Trace/Pattern window

The Trace and Pattern windows use the same layout. The top half of the window shows the trace or pattern details. The bottom half of the window shows the values and positions of marks you have set on the trace or pattern. The marks are referred to as cursors and are represented by coloured triangles.



TRACE WINDOW

The six panes visible in the above diagram contain (reading from left to right):

- Trace/pattern name(s) and port(s)
- Current value at selected cursor
- Waveform over time
- Cursor list with values relative to current cursor
- Cursors with absolute time values
- Time and cursors

The current trace is out-lined with a green dashed line.

The current cursor has a red underline.

Right-clicking the trace waveform or the current value pane gives you a menu of possible display formats for that pane.

## 3. Waveform Analyzer interface

---

You can have multiple traces or patterns in a single window, but they all use the same cursors, the same number of points and the same clock period.

### Zooming

You can zoom in and out of the active Trace or Pattern window using the zoom icons or the **Zoom** options from the **View** menu.

## 3.3 Menus

### 3.3.1 File Menu

Command	Description
<b>New</b> (Ctrl+N)	Creates a new trace or pattern window. You will be prompted for the type of window (Trace or Pattern), a filename for the window and the clock period and number of points for the window. The clock period and the number of points that you specify will be used for all traces or patterns in the window.
<b>Open</b> (Ctrl+O)	Opens an existing trace or pattern file.
<b>Close</b>	Closes the active trace or pattern window.
<b>Save</b> (Ctrl+S)	Saves the active trace or pattern window.
<b>Save As</b>	Saves the active trace or pattern window with a different name.
<b>Save All</b>	Saves all open trace and pattern windows.
<b>New Project</b>	Creates a new project.
<b>Open Project</b>	Opens an existing project.
<b>Close Project</b>	Closes the current project.
<b>Save Project</b>	Saves the current project.
<b>Print</b>	Prints the active trace or pattern window.
<b>Print Setup</b>	Sets up the printer details.
<b>Print Preview</b>	Previews the active trace or pattern window.
<b>Recent Files</b>	List of recently used trace or pattern files.
<b>Recent Projects</b>	List of recently used projects.
<b>Exit</b>	Closes all windows and exits the application.



## 3. Waveform Analyzer interface

---

### 3.3.2 New window dialog

The **New window** dialog box (**File>New**) defines the default clock period and the number of points in the window.

Field	Description
Untitled box	Select the window type (Pattern Window or Trace Window).
Default clock period	Enter the default clock period for the window.
Default No. points	Enter the number of points recorded in the window.
Filename and location	File where the window details are stored (use the browse button to choose a directory).

### 3.3.3 Edit menu

Command	Description
<b>Find</b> (Ctrl+F)	Searches for a specified sequence of data words in the selected trace or pattern. You will be prompted for a PGL statement describing the sequence of words to search for, the search direction, and whether to scroll to the sequence if it is found. Searching starts at the position of the selected cursor. If there is no cursor, searching starts at the beginning of the selected trace or pattern. If the sequence is found, the selected cursor is positioned at the start of the sequence. If there is no cursor, a cursor is created at the start of the sequence.
<b>Copy</b> (Ctrl+C)	Copies the selected portion of a trace or pattern to the clipboard.
<b>Paste</b> (Ctrl+V)	Pastes the contents of the clipboard into the selected portion of the pattern.
<b>Save Selection As...</b>	Saves the selected portion of a trace or pattern to a file. You will be prompted for a file name and, if the file is a VCD file, a reference name to use for the signal in the VCD file.

## 3. Waveform Analyzer interface

---

### 3.3.4 View menu

Command	Description
Toolbar	Toggles the toolbar on/off.
Status Bar	Toggles the Status Bar on/off.
Zoom Max	Zooms in to the maximum extent at the centre of the active trace or pattern window.
Zoom In	Zooms in at the centre of the active trace or pattern window.
Zoom Out	Zooms out from the centre of the active trace or pattern window.
Zoom Min	Zooms out to the maximum extent from the centre of the active trace or pattern window.
Zoom on Cursor	Zooms in on the selected cursor in the active trace or pattern window.
Jump to Cursor	Scrolls to the selected cursor in the active trace or pattern window.
New Cursor	Creates a new cursor in the centre of the active trace or pattern window.
Delete Cursor	Deletes the selected cursor from the active trace or pattern window.

### 3.3.5 Trace Menu

Command	Description
New Trace	Creates a new trace in the active Trace window.
Edit Trace	Edits the selected trace in the active Trace window.
Delete Trace	Removes the selected trace from the active Trace window.

### 3.3.6 Trace Properties dialog

Fields in the Trace Properties dialog box (**Trace>New Trace**) are:

Field	Description
Name	Name for the trace. This name will be displayed in a box on the left of the Trace window. The name can also be used as part of a trigger specification for this or any other trace. The name must be a C-style identifier.
Width	Width of the data in the trace in bits.
Type	Whether the trace represents signed or unsigned data.

### 3. Waveform Analyzer interface

---

Field	Description
Points	Number of points in the trace. This is the value you entered when you created the Trace window. It cannot be edited.
Clock Period	Rate at which data is read into the trace. This is the value you entered when you created the Trace window. It cannot be edited.
Expression	Port(s) the trace is connected to. The expression may be of the form ' <i>Terminal -Name(width)</i> ' (as for the DKConnect plugin) or a Handel-C expression with expressions of the form ' <i>Terminal -Name(width)</i> ' in place of variables.
Dump File	Enter a filename to capture the trace to a file. Two file formats are supported: ASCII files and Verilog Value Change Dump files. If the filename ends in .vcd, .dmp or .dump a Value Change Dump file will be produced, otherwise an ASCII file will be produced. The <b>Browse</b> button can be used to select a filename. If no filename is entered, no dump file will be produced.
Variable	If the dump file is a Verilog Value Change Dump file, enter the name which will be used as the reference name of the signal in the VCD file.
Trigger	Specifies which sequences of words triggering should occur on. If this box is empty, no trigger is used and all further trigger options are greyed out.
Delay	Specifies the trigger delay. This may be positive or negative. If a positive delay, x, is used, capturing begins x time units after a trigger sequence occurs. If a negative delay is used, capturing begins x time units before a trigger sequence occurs.
No trigger/Single/ Auto	Select trigger mode: <b>No trigger:</b> triggering is disabled. <b>Single:</b> a trace is captured once after a trigger sequence occurs. <b>Auto:</b> a trace is captured after every occurrence of a trigger sequence.
Pause on Trigger	If this checkbox is ticked, capturing will be automatically paused after a trigger sequence has occurred and a trace has been captured.
Interpolated Waveform/Stepped Waveform/Numeric Symbolic	Selects the display format for the trace.
Define Symbols	Opens the <b>Define Symbols</b> dialog where you can select how values are represented.

## 3. Waveform Analyzer interface

---

### 3.3.7 Pattern menu

Command	Description
New Pattern	Creates a new pattern in the active pattern window.
Edit Pattern	Allows you to edit the selected pattern in the active pattern window.
Delete Pattern	Removes the selected pattern from the active pattern window.

### 3.3.8 Pattern Properties dialog

Fields in the Pattern Properties dialog box (**Pattern > New Pattern**) are:

Field	Description
Name	Name for the pattern. The name is displayed in a box on the left of the Pattern window. The name must be a C-style identifier.
Width	Width of the data in the pattern in bits.
Type	Whether the pattern represents signed or unsigned data.
Points	Number of points in the pattern. This is the value you entered when you created the Pattern window. It cannot be edited.
Clock Period	Rate at which data is read into the pattern. This is the value you entered when you created the Pattern window. It cannot be edited.
Source	The source for the pattern may be either a file or a script. Supported file formats are ASCII and VCD. The box to the right of the radio buttons is used to enter a script if the <b>Script</b> radio button is checked, or a file name if the <b>File</b> radio button is checked.
Variable	If the source is a VCD file, this box should be used to enter the reference name of the variable in the VCD file that will be used as the source for this pattern.
Destination	Expression of the form ' <i>Terminal -Name(width)</i> ' as for the DKConnect plugin.
Trigger	Transmission of a pattern can be triggered by the occurrence of a specified sequence of words in any trace. This box is used to specify which sequences of words and which trace triggering should occur on. If this box is empty, no trigger is used and all further trigger options are greyed out.
Delay	Specifies the trigger delay. For patterns, this must be positive. A delay of x means that transmission begins x time units after a trigger sequence occurs.

## 3. Waveform Analyzer interface

---

Field	Description
<b>No trigger / Single / Auto</b>	Choose the trigger mode: <b>No trigger:</b> triggering is disabled. <b>Single:</b> a pattern is transmitted once after a trigger sequence occurs. <b>Auto:</b> a pattern is transmitted after every occurrence of a trigger sequence.
<b>Pause on Trigger</b>	If this checkbox is ticked, capturing will be automatically paused after a trigger sequence has occurred and a pattern has been transmitted.
<b>Interpolated Waveform/Stepped Waveform/Numeric Symbolic</b>	Selects the display format for the pattern.
<b>Define Symbols</b>	Opens the <b>Define Symbols</b> dialog where you can select how values are represented.

### 3.3.9 Define symbols dialog

The **Define symbols** dialog consists of a set of radio buttons which allow you to choose how values are represented:

- Binary
- Octal
- Decimal
- Hexadecimal numbers
- ASCII characters
- User defined strings: You must supply the file name of a file which associates symbols with values for the trace being defined. Each line of this file should contain a number (in binary, octal, decimal or hexadecimal, using the Hande-C syntax) followed by a symbol. The symbol should be separated from the number using white space. Any values which may appear in the Trace and which do not have symbols associated with them will be represented using the '?' character.

For example, if the trace is of width 3, is unsigned, and the user defined symbol file contains the following:

```
0b001    A
0b111    D
0b110    C
0b101    B
```

the values 1, 5, 6 and 7 will be represented as A,B,C and D respectively. The values 0, 2, 3 and 4 will all be represented as question marks.

## 3. Waveform Analyzer interface

---

### 3.3.10 Script menu

Command	Description
Edit Script...	Allows you to edit the PGL script for the current project.

### 3.3.11 Capture menu

Command	Description
Run (F5)	Starts reading traces from simulations and sending patterns to simulations.
Pause	Temporarily stops traces being sent to simulations and patterns being read from simulations. This also suspends all connected simulations.
Stop (Shift+F5)	Stops reading traces from simulations and sending patterns to simulations. Simulations continue running after Waveform Analyzer has stopped.
Advance (Ctrl+F11)	Advances all simulations by the interval specified in the <b>Set Advance Step</b> box.
Set Advance Step	Opens the <b>Set Advance Step</b> dialog which allows you to specify the interval by which to advance simulations when <b>Advance</b> is selected.

### 3.3.12 Set Advance Step dialog

The **Set Advance Step** dialog (**Capture > Set Advance Step**) specifies the time in nanoseconds to advance all simulations by.

### 3.3.13 Window menu

Command	Description
Cascade	Cascades all open windows.
Tile	Tiles all open windows.
Arrange Icons	Automatically arranges all minimized trace and pattern windows.

## 3. Waveform Analyzer interface

---

### *3.3.14 Help menu*

Command	Description
Help Topics	Invokes the online help.
About	Displays details about Waveform Analyzer.

# Index

---

## Index

apj files	11	patterns	8, 26
Capture menu	28	creating	12, 22
capturing waveforms	28	grouping	11
connecting a pattern to a port	8	limitations	12
connecting a simulation to a trace	6	properties	8, 26, 27
cursors	10, 21	PGL	11, 12, 14, 15, 16
Define symbols dialog	27	functions	16
DKConnect.dll	6, 8	limitations	16
DKShare.dll	6, 8, 10	scripts	11, 16, 28
DKSync.dll	6, 8	statements	15
Edit menu	23	sub-programs	16
examples		syntax	17
connecting a simulation to a trace	6	wildcards	16
pattern generation	8	ports	6, 8, 10
File menu	22	projects	11
generating patterns	12	scripts	14
Help menu	29	Set Advance Step	28
measuring differences		starting	5
time	10	Waveform Analyzer	5
value	10	time differences	10
menus	20	toolbars	20
New window dialog	23	Trace menu	24
Pattern Generation Language	11, 12, 14, 15, 16	Trace Properties dialog	24
functions	16	Trace window	21, 24
limitations	16	traces	6, 21, 24
scripts	11, 16, 28	creating	22
statements	15	grouping	11
sub-programs	16	properties	6, 24
syntax	17	triggers	12
wildcards	16	user interface	20
Pattern menu	26	value difference	10
Pattern Properties dialog	26	VCD	6
Pattern window	21, 26	Waveform Analyzer	
		overview	5



# Index

---

starting	5	wildcards	16
waveforms	23	Window menu	28
copying	23	zooming	20
finding	11, 23		

Customer Support at <http://www.celoxica.com/support/>

**Celoxica in Europe**

T: +44 (0) 1235 863 656

E: [sales.emea@celoxica.com](mailto:sales.emea@celoxica.com)

**Celoxica in Japan**

T: +81 (0) 45 331 0218

E: [sales.japan@celoxica.com](mailto:sales.japan@celoxica.com)

**Celoxica in Asia Pacific**

T: +65 6896 4838

E: [sales.apac@celoxica.com](mailto:sales.apac@celoxica.com)

**Celoxica in the Americas**

T: +1 800 570 7004

E: [sales.america@celoxica.com](mailto:sales.america@celoxica.com)

Copyright © 2003 Celoxica Ltd. All rights reserved. Celoxica, the Celoxica logo and Handel-C are trademarks of Celoxica Limited. All other trademarks acknowledged. The information herein is subject to change and for guidance only

**[www.celoxica.com](http://www.celoxica.com)**

