

An abstract graphic featuring three blue circles of varying sizes. The top-right circle is the largest, the middle one is medium-sized, and the bottom-right one is the smallest. Two thin, light blue diagonal lines intersect the circles, creating a sense of depth and movement. The circles are composed of concentric layers of different shades of blue, giving them a 3D effect.

# **IDesignSpec Quick Start Guide**

Version 3.9

Introduction.....	3
Basic Concept .....	3
Creating Specification.....	3
IDS Word/OpenOffice Templates .....	4
System .....	4
Board .....	4
Chip .....	4
Block .....	5
RegGroup .....	5
Register.....	5
Reference:.....	6
Specification checking and Address Generation.....	6
Generating Output Files .....	7
UVM Generation.....	7
Usage of Register model.....	8
Coverage.....	10
HDL Path.....	10
reset_value.....	10
TCL based outputs .....	12
Free IDS .....	13
Restrictions:.....	13
Troubleshooting .....	13
Terminology .....	14

## Introduction

This document provides instructions that will enable you to start using IDesignSpec in less than 10 minutes.

If you have not installed IDesignSpec, please double click on the downloaded file to install it.

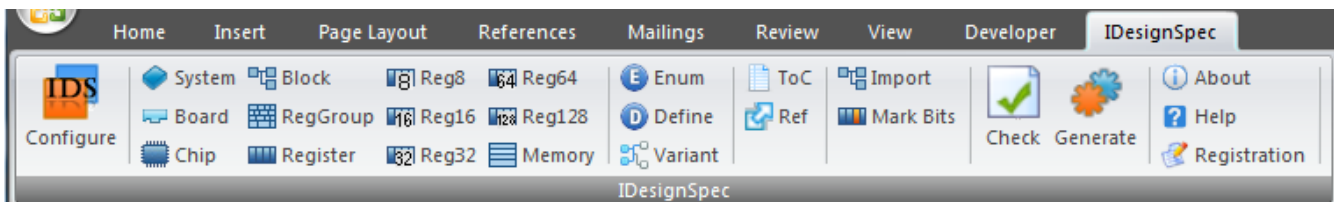
Refer to the IDesignSpec User Guide, IDSBatch user guide and the Tcl-API reference guides for more details.

## Basic Concept

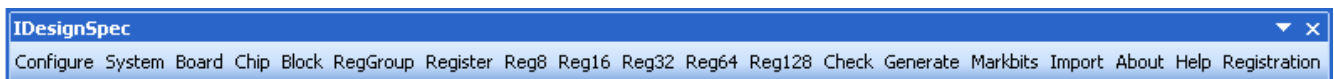
The basic idea behind IDesignSpec is that the specification of a digital system is treated as the Golden Reference from which all information is derived. You create the specification in the editor of your choice (Word, Excel, OpenOffice, LibreOffice etc.) using IDesignSpec provided templates, and generate outputs from it using the provided controls.

## Creating Specification

When the IDesignSpec plugin is installed into MSOffice or OpenOffice, the following tool-bar should appear. If you don't see a tool-bar, refer to the Troubleshooting section below. You should see a toolbar like the one shown below.



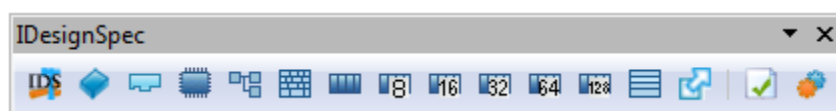
*Figure 1: MSWord 2007/2010 Toolbar*



*Figure 2: MSWord 2003 Toolbar*



*Figure 3: MExcel 2007Toolbar*



*Figure 4: OpenOffice/LibreOffice Toolbar*

IDesignSpec can be used like you would use any editor, except that there are special buttons that have been created to templates to the design. These buttons (or menu picks) are: register, register group, block, chip, board, system, memory and ref etc.

In addition, the Configure, Check and Generate buttons (or menu picks) are provided. Configure is used to specify what gets generated. Check does an on-the-fly checking of the user data. Generate creates the outputs in the user specified directory.

## IDS Word/OpenOffice Templates

A document created using IDesignSpec is no different than a regular document except the user adds special templates using the IDS plugin. In case of Excel, any format may be used as long as the format is described to the IDS system using the ids\_template sheet (see User Guide for more details). The various templates are described in the sections below.

### System

System is the highest level in the IDS hierarchy. A system is the collection boards. This template is optional.

Figure 5: System Template

### Board

A board is a physical arrangement of the logic design including multiple chips. This template is optional.

Figure 6: Board Template

### Chip

A chip is a major building block of an electronic system such as an ASIC or an FPGA. It is a semiconductor component which provides the memory, logic and virtually all other intelligence functions in an electronic system. It contains large number of electronic circuits made of Logic Gates. This template is also optional.

Figure 7: Chip Template

## Block

A block is logical grouping of digital circuitry. A single chip may contain multiple blocks of logic. A block contains registers or RegGroups.

Figure 8: Block Template

## RegGroup

RegGroup is a set of registers. A RegGroup can have registers and RegGroups in it. This can be used to construct an arbitrary multidimensional register set.

Figure 9: RegGroup Template

## Register

When the register button on the toolbar is clicked, an IDS template for register is created, and all of the details can be specified, including the register's constituent fields and their attributes, like software access, hardware access, default or reset value etc.

offset		reg_name										external										Reg.											
Enter Register Offset here		Enter Register Name here																															
Enter Register Description Here																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
bits		name		s/w		h/w		default		description																							
31:0										Enter Field Description Here																							
										Insert Field Details here																							
										Hit Tab to add more Fields																							

Figure 10: Register Template

## Reference:

Any Register, Block, RegGroup or any other element defined previously or in any other document.

The reference will automatically include the element being referred at the correct address location.

Note: The path can either be absolute path or the relative path to the document being referred.

offset		name		instance_name		path		type	
Enter the instance offset here				Enter the instance name of the element being referred					
				Enter the name of the element being referred					
						Enter the path of the document containing the element being referred (leave blank if referring elements from same document)			

Figure 11: Ref Template

The specification can be started at any level from block all the way up to the system. Your specification with these IDS templates can be mixed with the rest of the documentation as you see fit to optimize the readability of the specification.

## Specification checking and Address Generation

Once the design is specified, and all registers are created, clicking on the Check button verifies that the entered data are correct. Errors are shown in the document so that they do not get propagated on to the derived files. The IDesignSpec tool then proceeds to recalculate addresses and shows them in the IDS template.

By default the addresses shown are the 32-bit addresses for the various registers, however, the address unit can be changed by clicking on the Configure button.

Address calculation is done based on the size and contents of the blocks/chips etc. The default sizes of blocks/chips etc. are specified in the Configure window.

## Generating Output Files

Click on the Configure button to choose which outputs are generated. The Configure dialog box is shown in the following figure:

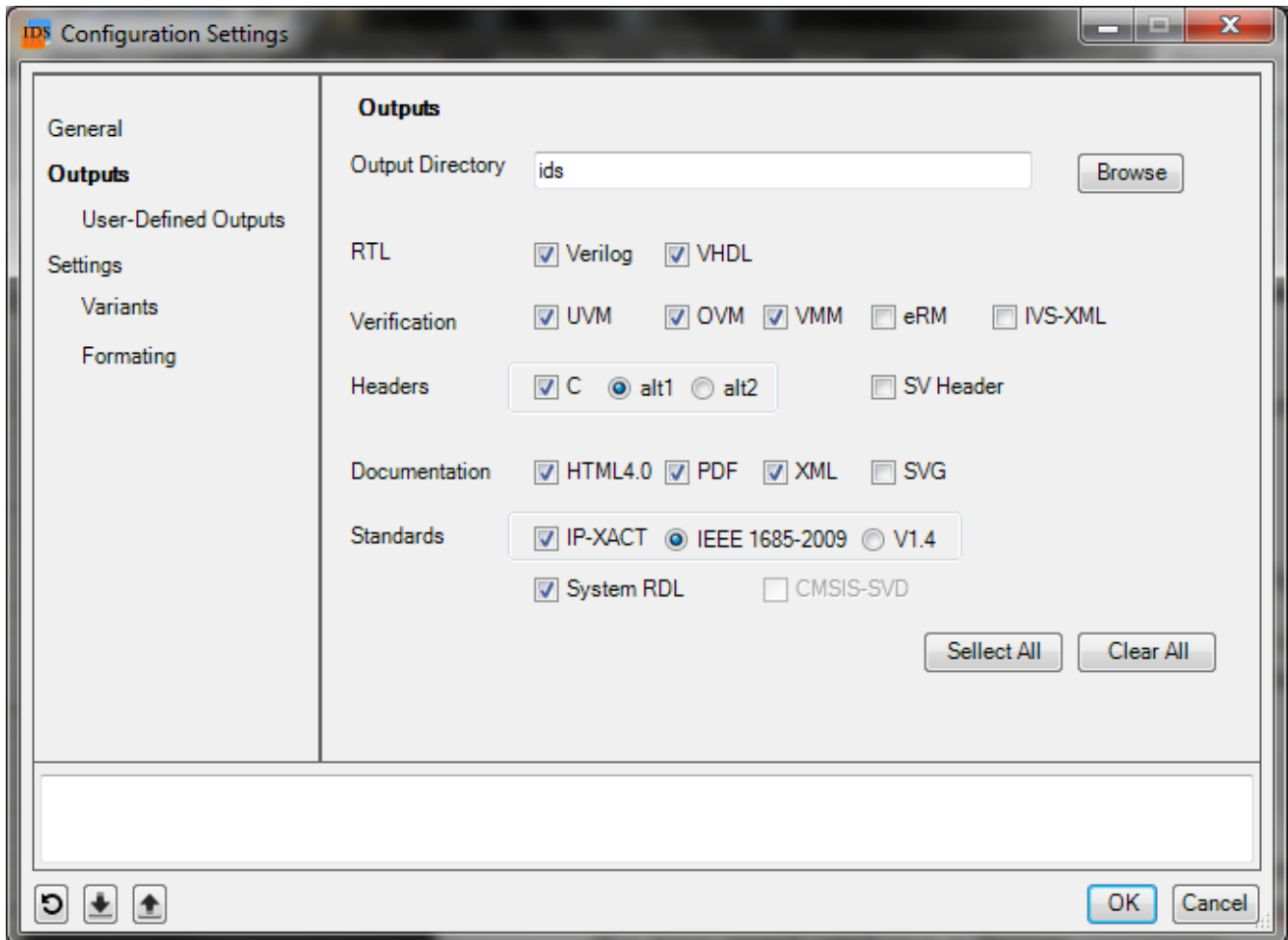


Figure 6: Configure Dialog box

Once the appropriate outputs are selected, click on the Generate button to create the outputs. All outputs are created in an "ids" directory next to the IDS file. This directory may be changed by the user.

## UVM Generation

UVM provides the best framework to achieve coverage-driven verification (CDV). CDV combines automatic test generation, self-checking testbenches, and coverage metrics to significantly reduce the time spent verifying a design.

The UVM register layer defines several base classes that, when properly extended, abstract the read/write operations to registers and memories in a DUV (Design Under Verification). This abstraction mechanism allows the migration of verification environments and tests from block to system levels without any modifications. It also can move uniquely named fields between physical registers without requiring modifications in the verification environment or tests. Finally, UVM

provides a register test sequence library containing predefined test-cases you can use to verify the correct operation of registers and memories in a DUV.

A register model is an instance of a register block, which may contain any number of registers, register files, memories, and other blocks. Each register file contains any number of registers and other register files. Each register contains any number of fields, which mirror the values of the corresponding elements in hardware.

Consider the following example of a register defined inside the block, in IDS Word (the Word version of IDesignSpec).

IDS_BLK		IDS_blk		<div>Block</div>		0x00001	
offset		01		External		Size	

IDS_reg_A		IDS_reg_A		<div>Reg.</div>			
offset				external			

Sample Register of Block IDS_blk																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bits		name		s/w		h/w		default		description																					
31:0		Fld_1		Rw		Ro		'hA		Field for the IDS_reg_A register with reset value equal to 'hA																					

This will create a Register Model containing 'IDS\_reg\_A' register inside the 'IDS\_blk' block. IDS\_reg\_A register class will have the Field named 'Fld\_1' with configurations taken from the above template.

```

Fld_1.configure(this, 32, 0, "RW", 0, 'd10, 1, 0, 0);
  
```

Field name      Field size (in bits)      Field LSF      Software Access      Reset Value

Likewise, IDS can generate Register Arrays, Memories, Indirect Access Registers and Coverage Models that are supported by the UVM register model in its UVM output.

### Usage of Register model

Once the Register Model is generated from the Register Specifications of the DUV, it is to be tested in the UVM Environment. User can also use RTL (Verilog/VHDL) generated by IDS in the UVM Environment as a DUV. Design and BUS Interface is created where bus (controlling the traffic of the transactions) signals are connected to the design. This Interface is the Top Level HDL Path.

For the support of the UVM Environment, Register Sequence classes need to be developed for the Read/Write operations in the Registers. User can also use the UVM Built-in Register Sequences "uvm\_reg\_mem\_built\_in\_seq". Or can create own sequence classes using "uvm\_reg\_sequence" as its base class. This class provides base functionality for both user-defined RegModel test sequences and "register translation sequences". Here, **uvm\_ids\_reg\_seq** class is created with **uvm\_reg\_sequence** as its base class.



```
class uvm_ids_reg_seq extends uvm_reg_sequence #(uvm_sequence #(uvm_reg_item));
```

After, the sequencers are created for the Registers and Memories, the testbench environment class needs to be created. The testbench environment class with `uvm_component` as its base class, is architected to provide a flexible and extendable verification component. The main function of this class is to model behavior by generating constrained random traffic, monitoring DUV responses and checking the validity of the protocol activity

```
class tb_env extends uvm_component;
. . . . .
```

This class consists of Register Model, Bus agent and Sequence class. In this class, HDL root path is defined for the backdoor accessibility. Here, the bus adapter and the run task are implemented.

```
IDS_blk_block regmodel;           // IDS Generated UVM Register model
apb_agent      apb;               // APB BUS (UVM already supports this bus interface)
uvm_reg_sequence seq;            // UVM Sequence class
. . . . .
    string hdl_root = "top.DUV";   // Top level HDL Path
    . . . . .
    regmodel.set_hdl_path_root(hdl_root); // Defining HDL Path to Rg-Model for Back-
                                         door access
end
. . . . .
    regmodel.default_map.set_sequencer(apb.sqr, reg2apb); //Connecting the Bus Adapter
. . . . .
virtual task run_phase();          // run Task
. . . . .;
```

After the testbench environment is setup, the testbench is created, where all the above classes are used. Testbench program collects all the files in the Environment.

```
`include "apb.sv"                // Include APB Bus interface
`include "uvm_top_DUV.sv"         // Include TOP level DUV
`include "uvm_seqlib.sv"          // Include Register Sequence Classes
`include "ids/IDS_blk_regmem.sv"  // Include IDS generated Register Model
import IDS_blk_regmem_pkg::*;     // Import the Package containing Register Model
                                  Classes
program tb;                       // testbench "tb" program
. . . . .
    `include "uvm_env.sv"         //Include the Testbench Environment Class
. . . . .
endprogram
```

With this the Environment is build up and is ready to be tested. On compiling the Environment user have to just compile the testbench file which will automatically compile all the Environment files. Then run the test giving the sequence class name on the command line. UVM Environment will run the sequence name mentioned on the command line. This is how the IDS generated register model is used.

User can also run the Makefile available in the `ids_uvm_example` which will do all the above mentioned tasks for the user. To run the Makefile to run the UVM environment with Verilog RTL simply run

```
%make verilog
```

## Coverage

In IDS, coverage code can be generated in the UVM Register model defined for Blocks, Registers, Memories and Fields. User can get the coverage for that element, by adding a property named "Coverage" with value "on", "a", "b", "f" or any combination of "abf" to enable the desired coverage. By default the coverage will be "off".

Coverage type	Description
on/ON	All coverage on
a/A	Address coverage
b/B	Reg bits coverage
f/F	Field Vals coverage
off/OFF	No coverage on


"Coverage" code is generated for Blocks, Memories and Registers depending on where it is written.

## HDL Path

The UVM register library can specify arbitrary hierarchical path components for blocks, register files, registers and memories that, when strung together, provide a unique hierarchical reference to a register or memory.

In IDS user can mention the hdl path from the property name 'hdl\_path' having hierarchical path value e.g. 'top.dut.R'. In IDS, user can mention hdl paths for memories and registers.

For example:

<b>reset_value</b>										reset_value										 Reg.					0x0000004						
offset										External										default					0x0						
{hdl_path=top.dut.reset_value}																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bits		name				s/w		h/w		default		description																			
31:0		field				rw		ro		0x0																					

```
class stopwatch_block extends uvm_reg_block;
    stopwatch_value value;
    stopwatch_reset_value reset_value;
    ...
    ...
    //hdl_path
    reset_value.clear_hdl_path();
    reset_value.add_hdl_path_slice("top.dut.reset_value", 0, 32);
    ...
endclass
```

Following is the complete list of the UVM properties that can be specified in IDS.

Property	Value	Description
<b>coverage</b>	on/off or abf	Include Coverage for that component and to its Lower Hierarchy. a- Coverage of Addresses b- Coverage of Register Bits f- Coverage of Register field Values on- All Coverage on e.g: {coverage= fa} field and address coverage.
<b>index_reg depth</b>	INDEX_reg1 256	Additional properties for indirect register. Specifies name of the index register and depth of the external register.
<b>uvm_class</b>	Class name e.g: uvm_my_class	User can generate its own Class for any component in the Register Model.
<b>coverage.&lt;coverage_options&gt;</b>	per_instance=<Boolean value> weight=<number value> goal=<number value> comment=<string value> at_least=<number value> detect_overlap=<Boolean value> auto_bin_max=<number value> cross_num_print_missing=<number value> get_inst_coverage=<Boolean value>	User can specify the coverage option to be used in the covergroup.
<b>hdl_path</b>	Hierarchy name e.g. (top.dut.reg1)	Hdl path can be given for any component.
<b>no_reg_tests</b> <b>no_reg_hw_reset_test</b> <b>no_reg_bit_bash_test</b> <b>no_reg_access_test</b>	true / false OR 1 / 0	Test sequences are not applied to Registers or Blocks, for which these properties value is true.
<b>no_mem_tests</b> <b>no_mem_access_test</b> <b>no_mem_walk_test</b>	true / false OR 1 / 0	Test sequences are not applied to only Blocks, for which these properties value is true.

## TCL based outputs

IDS enable users to generate their custom outputs for the register specification. Custom outputs can be generated using the IDS TCL-API.

To generate the custom outputs, keep all the TCL scripts in a directory and point the "IDS\_XFORM\_PATH" environment variable to this directory. All the TCL scripts available at the IDS\_XFORM\_PATH will appear in the IDS configuration window along with other IDS default outputs.

For more details on the TCL-API please refer to [IDS tcl api reference guide](#).

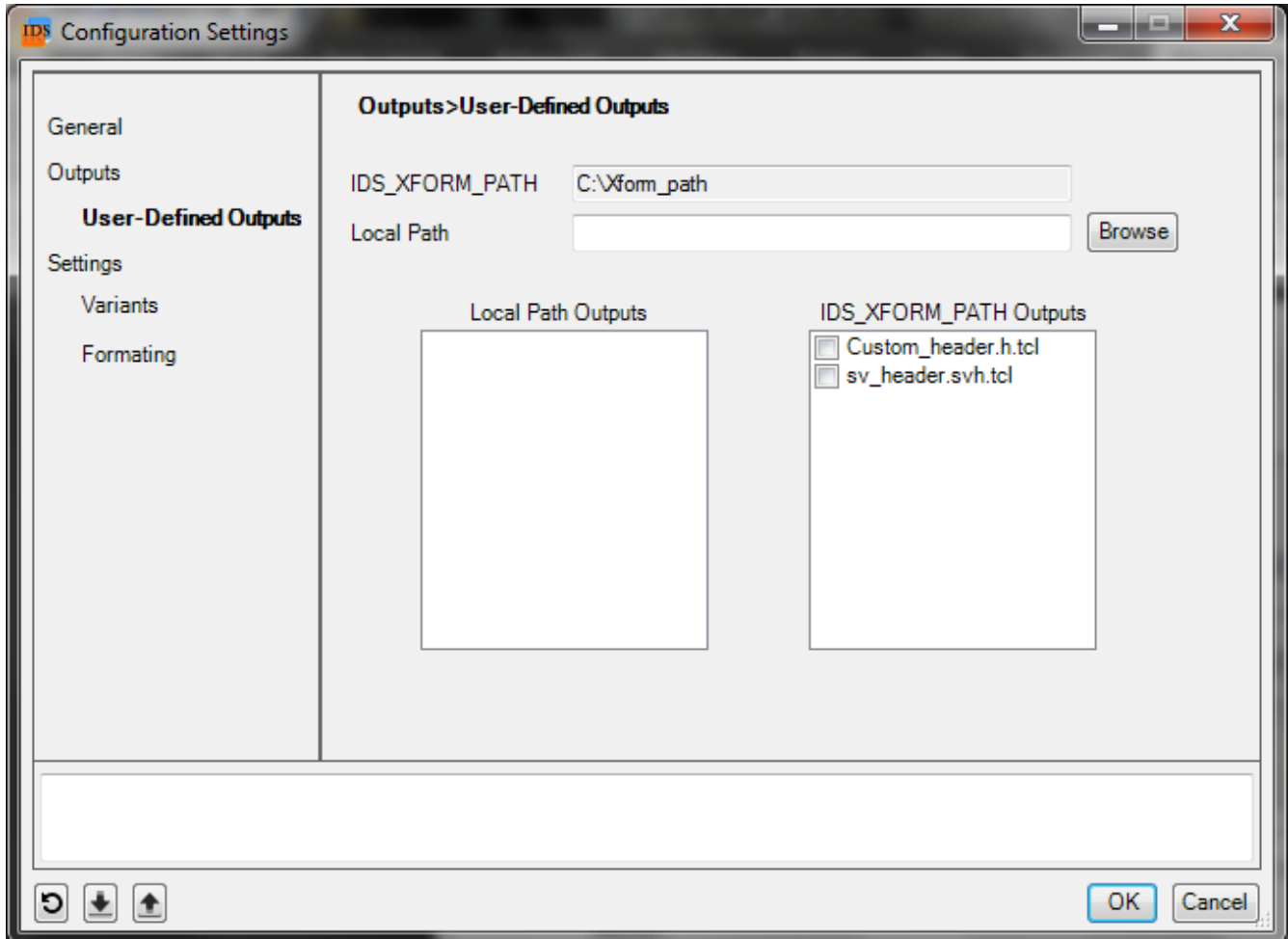


Figure 7: User defined outputs

Note:

The name of the TCL scripts should follow the scheme <script\_name>.<output\_extension>.tcl

## Free IDS

Free version of IDesignSpec™ gives the users a sense of immense power of the tool. IDS free is the complete tool in itself with a few restrictions.

### ***Restrictions:***

- Specification can have maximum up to 25 IDS templates.
- Specification can have only Block and Register templates.
- Table of contents generation for the register specification.
- One user defined output using IDS TCL-API.
  - To use IDS TCL-API please refer to tcl-api reference guide.
- Outputs included
  - UVM model generation with coverage hdl\_path and other UVM properties.
  - Register verification plan generation for IVerifySpec.

To enable all the IDS features and functionality please contact [Agnisys support](#).

## Troubleshooting

If IDesignSpec plug-in does not appear after installation, please refer to IDS troubleshooting guides.

- [IDS Word2007](#)
- [IDS Word 2003](#)
- [IDS Excel2007](#)
- [IDSOO](#)

## Terminology

IDS Template	The graphics that is inserted into the document when one of the six buttons (system, board, chip, block, register group, register) are clicked.
ids_template	This is the name of the template sheet that is read by IDSEExcel.
Software Access (sw access)	On the Register template it specifies what kind of access the software has to the register field. Hit "Enter" to toggle thru all possible values (ro, rw, wo, r/w1c, r/wc)
Hardware Access (hw access)	On the Register template it specifies what kind of access the hardware has to the register field. Hit "Enter" to toggle thru all possible values (ro, rw, wo).
Default	On the Register template it specifies the default value of the field if any.
Bits	Bits specify a range of bits for the register field. Single bit of a range of bits can be specified. Example : 31-0 or 31:0 or 0-31 or 0:31
Offset	Offset is with respect to the enclosing block or register group. The offset is specified in terms of the Register locations, and not in terms of byte addresses.
Repeat	Applies only to the register group. It specifies how many times the content of a register group is repeated.
External	A register or a register group can be marked as external. By default they are considered internal, which means that VHDL/Verilog RTL would be generated for them. If marked external, only appropriate ports for external hook-up are created.
Size	Size of the chip, block, or RegGroup. Note that the units are as specified in the Configuration window.