

IDesignSpec™

Tcl-Interface API

Version 4.9

Contents

| | |
|---|---|
| Introduction | 3 |
| IDS Object Types and their properties | 3 |
| System | 3 |
| Board | 3 |
| Chip | 3 |
| Block | 4 |
| RegGroup | 4 |
| Reg | 5 |
| Field | 5 |
| Enum | 6 |
| Define | 6 |
| Supported Commands | 7 |
| get_config | 7 |
| get_tcl_api_version | 7 |
| get_system | 7 |
| get_board | 7 |
| get_chip | 7 |
| get_block | 8 |
| get_fields | 8 |
| get_objects | 8 |
| get_object_by_name | 8 |
| get_top | 9 |

| | |
|---------------------------------|----|
| get_prop | 9 |
| get_parent | 9 |
| get_type..... | 9 |
| get_enums | 10 |
| get_defines | 10 |
| get_value_constraint | 10 |
| get_unique_name | 11 |
| get_id..... | 11 |
| Deprecated Commands..... | 11 |
| get_value_list | 11 |
| get_property | 11 |
| Error Handling | 11 |
| User Specified Errors | 12 |

Introduction

IDesignSpec (IDS) enables users to customize and create new outputs using Tcl (Tool Command Language). This document describes the Tcl commands that work on the IDesignSpec documents. For example, “**Templates**” in IDS documents translate into “**Objects**” in Tcl. The Tcl API introduces 7 object types, each corresponding to a document template, e.g., Property/Value pairs on the templates translate into ‘properties’ within Objects. So for a “block”, the value specified for offset can be accessed by getting the ‘offset’ property of the block object in Tcl.

IDS Object Types and their properties

The following types of Objects have been introduced in the Tcl API of IDS.

System

System Properties

name

Name of the register after performing following actions:

- If ‘Preserve name’ is set in the IDS Configuration, then all spaces are trimmed and then converted to “_” and everything other than underscore, numbers and alphabets is removed.
- If ‘Preserve name’ is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

Board

Board Properties

name

Name of the register after performing following actions:

- If ‘Preserve name’ is set in the IDS Configuration, then all spaces are trimmed and then converted to “_” and everything other than underscore, numbers and alphabets is removed.
- If ‘Preserve name’ is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

Chip

Chip Properties

name

Name of the register after performing following actions:

- If 'Preserve name' is set in the IDS Configuration, then all spaces are trimmed and then converted to "_" and everything other than underscore, numbers and alphabets is removed.
- If 'Preserve name' is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

offset

Its value is in the same unit as defined by the settings in the IDS Configuration.

address

byte address of the chip.

size

Total size of the chip. Its value is in the same unit as defined by the settings in the IDS Configuration.

external

true if the chip is external, else its value is false.

Block

Block Properties

name

Name of the register after performing following actions:

- If 'Preserve name' is set in the IDS Configuration, then all spaces are trimmed and then converted to "_" and everything other than underscore, numbers and alphabets is removed.
- If 'Preserve name' is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

offset

Its value is in the same unit as defined by the settings in the IDS Configuration.

address

byte address of the block.

size

Total size of the block. Its value is in the same unit as defined by the settings in the IDS Configuration.

external

true if the block is external, else its value is false.

RegGroup

RegGroup Properties

name

Name of the register after performing following actions:

- If 'Preserve name' is set in the IDS Configuration, then all spaces are trimmed and then converted to "_" and everything other than underscore, numbers and alphabets is removed.
- If 'Preserve name' is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

offset

Its value is in the same unit as defined by the settings in the IDS Configuration.

address

byte address of the RegGroup.

external

true if the RegGroup is external, else its value is false.

Repeat/count

Number of times the RegGroup is repeated.

size

Total size of the RegGroup per iteration. Its value is in the same unit as defined by the settings in the IDS Configuration.

Reg

Register Properties

name

Name of the register after performing following actions:

- If 'Preserve name' is set in the IDS Configuration, then all spaces are trimmed and then converted to "_" and everything other than underscore, numbers and alphabets is removed.
- If 'Preserve name' is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

offset

Its value is in the same unit as defined by the settings in the IDS Configuration.

address

byte address of the register.

external

true if the register is external, else its value is false.

width

width of the register i.e 8/16/ 32 / 64/128.

doc

documentation specified for the register

Field

Field Properties

name

Name of the register after performing following actions:

- If 'Preserve name' is set in the IDS Configuration, then all spaces are trimmed and then converted to "_" and everything other than underscore, numbers and alphabets is removed.
- If 'Preserve name' is NOT set in the IDS Configuration, then everything except alphabets and numbers is removed.

bits

As specified by the user.

sw_access/sw

Type of software access.

hw_access/hw

Type of hardware access.

default_value/default

Default value of the field.

high_offset

Value for the high offset.

low_offset

Value for the low offset.

size

Size of the field

doc

Documentation specified for the register

Enum

Enum is a simple list. It is not an IDS object.

This API provides access to the 'enum' definitions and 'enum' values used in the register fields.

To access 'enum' definitions you can use the functions 'get_enums'.

IDS enum is represented as a Tcl list, with first index being the enum name and the second index has the list of enum fields. Each 'enum' field is further a list of mnemonic, value and optional description at indexes 0, 1 and 2 respectively.

Example of 'enum':

```
{Enum1 {{ON 1 {1 is on}} {OFF 0 {0 is off}}}}
```

Define

Define is a simple list. It is not an IDS object.

This API provides access to the 'define' definitions. To access 'define' definitions you can use the functions 'get_defines'.

IDS define is represented as a Tcl list having define var, value and optional description at indexes 0,1 and 2 respectively.

Example of 'define':

```
{ON 1 {1 is ON}}
```

Supported Commands

get_config

Usage:

get_config

Returns: List: Specifying the configuration setting.

Description: Return the list of all configuration setting set in "Configure" window.

get_tcl_api_version

Usage:

get_tcl_api_version

Returns: String.

Description: Return the version of TCL API.

get_system

Usage:

get_system <obj>

Arguments: <obj>: Object of type: board, chip, block, memory, reggroup, reg, and field.

Returns: Object of type: system

Description: Get the enclosing system for a given board, chip, block, memory, reggroup, register or field. Returns a NULL* Object, in case a system doesn't exist.

get_board

Usage:

get_board <obj>

Arguments: <obj>: Object of type: chip, block, memory, reggroup, reg and field.

Returns: Object of type: board

Description: Get the enclosing board for a given chip, block, memory, reggroup, register or field. Returns a NULL* Object, in case a board doesn't exist.

get_chip

Usage:

get_chip <obj>

Arguments: <obj>: Object of type: block, memory, reggroup, reg, field

Returns: Object of type: chip

Description: Get the enclosing chip for a given block, memory, reggroup, register or field. Returns a NULL* Object, in case a chip doesn't exist.

get_block

Usage:

```
get_block <obj>
```

Arguments: <obj>: Object of type: memory, reggroup, reg or field.

Returns: Object of type: block

Description: Get the enclosing block for a given memory, regroup, register or field. Returns a NULL* Object, in case a block doesn't exist.

get_fields

Usage:

```
get_fields <obj>
```

Arguments: <obj>: Object of type: reg

Returns: List of Objects of type: field.

Example:

```
foreach blk_obj $list_of_blocks {
    set all_regs_or_RegGroups [get_objects $blk_obj]
    foreach reg_or_RegGroup $all_regs_or_RegGroups {
        if {[get_type $reg_or_RegGroup]=="reg"} {
            set reg $reg_or_RegGroup reg
            set all_fields [get_fields $reg]
        }
    }
}
```

Description: Returns a list of field Objects for a given register.

get_objects

Usage:

```
get_objects <obj>
```

Arguments: <obj>: Object of type: system, board, chip, block or reggroup

Returns: Object of type: Board, chip, block, regroup, reg or memory.

Example:

```
set top_elem [get_top] // if this is a system
set all_board_elems [get_objects $top_elem]
```

Description: Get all objects inside a given object.

get_object_by_name

Usage:

```
get_object_by_name <hierarchical_name>
```

Arguments: <hierarchical_name>: Complete name of the desired object separated with a "/" character specifying the complete hierarchy.

Returns: Object of type: system or board or chip or block or reg or field.

Description: Return the object based on the hierarchical name.

get_top

Usage:

```
get_top
```

Arguments: None

Returns: *Object* of type: system, board, chip or block.

Example:

```
set top_elem [get_top]
```

Description: Get the top most Object in the source document

get_prop

Usage:

```
get_prop <obj> [<property_name>]
```

Arguments: <obj>: Object of any valid type

<property_name> : Name of any valid property

Returns: String: Value of the property.

Example:

```
set chip_name [get_prop $chip_obj name]
set chip_properties [get_prop $chip_obj]
```

Description: "property_name" is an optional argument for get_prop, when specified, returns the value of the specified property for the given object. Otherwise returns the list of all the available properties on the specified object.

get_parent

Usage:

```
get_parent <obj>
```

Arguments: <obj>: object of any valid type

Returns: *Object of type:* System, board, chip, block , reggroup or reg

Example:

```
Set parent [get_parent $object]
```

get_type

Usage:

```
get_type <obj>
```

Arguments: <obj>: Object of any valid type

Returns: String: Specifying any of the valid types.

Example:

```
set top_template_type [get_type [get_top]]
if {[string match $top_template_type chip]} {
    puts "Top most template is a chip"
}
```

Description: Returns the type of IDS object. One of "system", "board", "chip", "block", "reg", "reggroup", "field" or "mem". Note that enum and define are not objects but simple Tcl lists.

get_enums

get_enums <obj>

Arguments: <obj>: Object of any valid type

Returns: List of all the 'Enum' definitions.

Example:

```
set all_enums [get_enums $block]
foreach enum $all_enums {
    set enum_name [lindex $enum 0]
    set enum_fields [lindex $enum 1]

    foreach e_field $enum_fields {
        puts "Enum Mnemonic -> [lindex $e_field 0]"
        puts "Enum Value -> [lindex $e_field 1]"
        puts "Enum Desc. -> [lindex $e_field 2]"
    }
}
```

Description: Returns the list of all 'Enum' definitions for the given object.

get_defines

get_defines <obj>

Arguments: <obj>: Object of any valid type

Returns: List of all the 'Define' definitions.

Example:

```
set all_defines [get_defines $block]
foreach define $all_defines {
    puts "Define Var -> [lindex $define 0]"
    puts "Define Value -> [lindex $define 1]"
    puts "Define Desc. -> [lindex $define 2]"
}
```

Description: Returns the list of all 'Define' definitions for the given object.

get_value_constraint

Usage:

get_value_constraint <obj>

Arguments: <obj>: Object of type: field

Returns: List containing the min and max values defined in the user. It's syntax is as follows: {{min <min_value>} {max <max_value>}}

Example:

```
set cons_vals [get_value_cons $field_elem]
set min_val [lindex [lindex $cons_vals 0] 1]
set max_val [lindex [lindex $cons_vals 1] 1]
puts "Range : $min_val - $max_val"
```

Description: Returns a list of set of min and max values for a given field. Returns an empty string if no constraint is defined.

get_unique_name

Usage:

```
get_unique_name <obj> [<ref_obj>] [<separator>]
```

Arguments:

<obj>: Object of any valid type.

<ref_obj>: This must be a valid parent of the <obj>. Accepts, object of type: system, board, chip, block, section.

Returns: String

Example:

For a hierarchy -> systemA > boardA > chipA > blockA > regA

a. [get_unique_name \$regA_obj] :

Returns systemA_boardA_chipA_blockA_regA

b. [get_unique_name \$regA_obj \$boardA_obj] :

Returns chipA_blockA_regA

Description: Returns a name for a given object which is unique in the complete hierarchy by default. The hierarchy can be limited by specifying a reference Object, which must be a valid parent of the 'obj'.

get_id

Usage:

```
get_id <obj>
```

Arguments: <obj>: Object of type: system, chip, block, memory, reggroup, memory, reg or field.

Returns: String

Example:

For a hierarchy -> systemA > boardA > chipA > blockA > regA

[get_id \$regA_obj] :

Returns system/board/chip/block/reg

Description: Returns complete hierarchical path to the given object separated by slash "/". Each node in the path will be type of that node.

Deprecated Commands

get_value_list

This has been replaced by 'get_default_value_enum' as of version 2.2 of the API.

get_property

This has been replaced by 'get_prop'.

Error Handling

All functions defined here which returns an 'IDS Object' or a list of IDS Objects can return an IDS Object of type NULL (in case no object that matches your request is found). You may use the function "is_null" to check if an object is NULL.

User Specified Errors

IDS support generating error messages from your Tcl script. These messages are displayed in the IDS message window at the time of Generate. You may use the Tcl mechanism of generating errors. For example, when the following code is encountered in a IDS tcl script, an error is shown to the user.

```
error "Register name with two '_' are not allowed"
```

This mechanism is useful for creating user defined custom errors that can enforce custom rules.
