Verification Continuum™ Verdi®
Python-Based NPI Waveform Model

Version V-2023.12-SP1, March 2024

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Feedback

Contents

Customer Support 5
Synopsys Statement on Inclusivity and Diversity 6

1. **Introduction to Python Based NPI** 7
Packages and Modules 7
Module Functions and Class Objects 8
User Interface and Use Flow 8
Environment and Library Setting 8

2. **Module npisys** 11
Overview 11
L0 APIs 11

3. **Python-Based NPI Waveform Model** 13
Overview 13
Quick Start 13
Enums 14
Scope Enums 15
Signal Enums 16
VCT Enums 18
FT Enums 19
L0 APIs 20
File 20
Scope 33
Signal 37
vct (value change traverse) 52
ft (force tag traverse) 61
vc Iterator 66
Time Based 66
Signal Based 69
L1 APIs 73

Hierarchy Tree 75
Time Conversion 77
Sig Values 80
Find Values 86

Preface

The Python Based NPI Waveform Model User Guide provides convenient APIs to access a waveform.

Customer Support

For any online access to the self-help resources, you can refer to the documentation and searchable knowledge base available in SolvNetPlus.

To obtain support for your Verdi product, choose one of the following:

- Open a case through SolvNetPlus.

  Go to https://solvnetplus.synopsys.com/s/contactsupport and provide the requested information, including:

  - Product L1 as Verdi
  - Case Type

  Fill in the remaining fields according to your environment and issue.

- Send an e-mail message to verdi_support@synopsys.com.

  Include product name (L1), sub-product name/technology (L2), and product version in your e-mail, so it can be routed correctly.

  Your e-mail will be acknowledged by automatic reply and assigned a Case number along with Case reference ID in the subject (ref:_...:ref).

  For any further communication on this Case via e-mail, send e-mail to verdi_support@synopsys.com and ensure to have the same Case ref ID in the subject header or else it will open duplicate cases.

- You can call for support at:

  https://www.synopsys.com/support/global-support-centers.html

Note:

In general, we need to be able to reproduce the problem in order to fix it, so a simple model demonstrating the error is the most effective way for us to identify the bug. If that is not possible, then provide a detailed explanation of the problem along with complete error and corresponding code, if any/permissible.

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Introduction to Python Based NPI

Python-Based NPI APIs support six models. They are as follows:

- Waveform
- Netlist
- Text
- Coverage
- Language
- Waveform Writer

Each model have their own APIs to let you be able to traverse data objects and obtain objects' properties like the existing C-Based or Tcl-Based NPI APIs.

In this guide, the environment setting for using **Python-Based NPI APIs for Waveform** is demonstrated.

Packages and Modules

Packages

The Python-based NPI package name is "pynpi", and it is placed at `$VERDI_HOME/share/NPI/python`.

Modules

There are seven modules inside the "pynpi" package: npisys, lang, netlist, text, cov waveform and waveformw. The first module, npisys, is the system model for initialization, loading design and exit. The other modules represent language model, netlist model, text model, coverage model, wave model and waveform writer model respectively

Module Functions and Class Objects

L0 Module Functions

Every module provides some L0 (level 0) functions to let you get the class objects. These functions return a class object or a list of class objects, and they follow the specification of the existing L0 APIs provided in C or Tcl.

L1 Module Functions

Similar to L0 module functions, every module also provides some L1 (level 1) functions to let you get advanced information based on the results obtained by L0 module functions. These functions follows the specification of the existing L1 APIs provided in C or Tcl.

Class Objects

The class object is similar to the so-called handle in NPI C APIs. The most difference is that some basic L0 APIs in C and Tcl will become class method function. These L0 APIs are usually to get integer value, string value, 1-to-1 method to get a handle, and 1-to-many method to get handle iterator.

User Interface and Use Flow

This chapter describes the user interface and use flow for Python-Based NPI APIs.

Environment and Library Setting

The python library setting flow of using Python-Based NPI APIs contains four parts:

1. Check your Python's version:

   Python-Based NPI APIs need the Python version greater than 3.6.0.

2. Environment setting for "`VERDI_HOME`" is required for Python-based NPI. Remember to set it well before running program.

3. Add python library path into your python code before loading Python-Based NPI by using the following commands:

```
rel_lib_path = os.environ['VERDI_HOME'] + '/share/NPI/python'

sys.path.append(os.path.abspath(rel_lib_path))
```

4. Import module “npisys” for using the function of NPI initialization and exit from pynpi package.

```
from pynpi import npisys
```

5. Import the module you need from pynpi package. For example, if you want to use waveform model, you can import the module as follows:

```
from pynpi import waveform
```

6. Note that initialization function `npisys.init()` must be called before writing your code by using any other modules. Also, `npisys.end()` must be called after finishing your code. Following is a simple example to demonstrate how to use waveform model by Python-Based NPI APIs.

   Python program to use NPI waveform model: (`demo.py`)

```
#!/global/freeware/Linux/2.X/python-3.6.0/bin/python
import sys, os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
# Initialize NPI
if not npisys.init(sys.argv):
      print("Error: Fail to initialize NPI")
      assert 0
# Load design (if needed, depends on models)
if not npisys.load_design(sys.argv):
      print("Error: Fail to load design")
      assert 0
# Beginning of your code here --------------------
#
# Example code can be found in later chapters
#
# End of your code -------------------------------
# End NPI
npisys.end()
```

   C shell script to setup environment and execute Python program on 64-bit machine: (`run_demo`)

```
#!/bin/csh -f
# Setup your $VERDI_HOME here
setenv VERDI_HOME [YOUR_VERDI_HOME_PATH]
# run the python program
# - Input arguments depend on your program design
# - If loading design is required, you can pass the options like
./demo.py -sv demo.v
```

To run the files, put the files in the same directory and execute the `run_demo` C shell script.

```
./run_demo
```

2

Module npisys

This chapter includes the following topics:

- Overview
- L0 APIs

Overview

Module npisys is for setting Python-based NPI. You must call `npisys.init()` before using any other NPI modules and call `npisys.end()` after using any other NPI modules.

L0 APIs

Following are the public L0 APIs for system module:

npisys.init(pyArgvList)

System initialization for Python-Based NPI.

Parameters: pyArgList (str list) – input argument list, for example, sys.argv

Returns: Return 1 if successful. Otherwise, return 0.

Return type: int

Example:

```
>>>npisys.init(sys.argv)
```

npisys.load_design(pyArgvList)

Load design for Python-Based NPI.

Parameters: pyArgList (str list) – input argument list. For example, sys.argv

Returns: Return 1 if successful. Otherwise, return 0.

Return type: int

Feedback

Example:

```
>>>npisys.load_design(sys.argv)
```

---

npisys.end()

Clean NPI-related settings and data.

Parameters: none

Returns: Return 1 if successful. Otherwise, return 0.

Return type: int

Example:

```
>>>npisys.end()
```

3

Python-Based NPI Waveform Model

This chapter includes the following topics:

- Overview
- Quick Start
- Enums
- L0 APIs
- vc Iterator
- L1 APIs

Overview

This model provides convenient APIs to access a waveform. Performance of database access and the programming usability are both considered in this model.

Quick Start

Following are the Environment and library setting:

1. Add python library path using the following commands:

```
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/
python"
sys.path.append(os.path.abspath(rel_lib_path))
```

2. Import `npisys` to use the function of NPI initialization and exit.

   Import `waveform` to use the APIs of Waveform Model.

```
from pynpi import npisys

from pynpi import waveform
```

Feedback

3. If any error exists in LD_LIBRARY_PATH, add "`$VERDI_HOME/share/NPI/lib/linux64`" and "`$VERDI_HOME/platform/linux64/bin`" to `LD_LIBRARY_PATH`:

```
os.environ['LD_LIBRARY_PATH'] =
os.environ['VERDI_HOME']+'/share/NPI/lib/
linux64:'+os.environ['VERDI_HOME']+'/platform/linux64/
bin:'+os.environ['LD_LIBRARY_PATH']
```

Enums

- Enum lists
- Scope Enums

ScopeType_e	Waveform scope type.

- Signal Enums

DirType_e	Apply to: Fsdb Signal Diration Type.
SigAssertionType_e	Apply to: Fsdb Signal Assertion Type.
SigCompositeType_e	Apply to: Fsdb Signal Composite Type.
SigSpiceType_e	Apply to: Fsdb Signal Power Type.
SigSpiceType_e	Apply to: Fsdb Signal Spice Type.

- VCT Enums

VctFormat_e	vctFormat used in return vct value format, the input of vct value format and the value format.

- FT Enums

ForceTag_e	Apply to: Fsdb Force Tag.
ForceSource_e	Apply to: Fsdb Force Source.

Feedback

Scope Enums

ScopeType_e

`class` waveform.**ScopeType_e**

Waveform scope type

Apply to System Verilog type Scope

SvModule = 0

SvTask = 1

SvFunction = 2

SvBegin = 3

SvFork = 4

SvGenerate = 5

SvInterface = 6

SvInterfacePort = 7

SvModport = 8

SvModportPort = 9

Apply to VHDL type Scope

VhArchitecture = 10

VhProcedure = 11

VhFunction = 12

VhProcess = 13

VhBlock = 14

VhGenerate = 15

Apply to System C type Scope

ScModule = 16

Apply to spice type Scope

Spice = 17

Apply to Power Type Scope

Feedback

PwScope = 18

PwDomain = 19

PwSupplySet = 20

PwStateTable = 21

PwStateGroup = 22

PwSwitch = 23

PwIsoStrategy = 24

PwRetStrategy = 25

PwLsStrategy = 26

Apply to Unknown type Scope

Unknown = 27

Signal Enums

DirType_e

`class` waveform.**DirType_e**

Apply to Fsdb Signal Diration Type

DirNone = 0

DirInput = 1

DirOutput = 2

DirInout = 3

SigAssertionType_e

`class` waveform.**SigAssertionType_e**

Apply to Fsdb Signal Assertion Type

Assert = 0

Assume = 1

Cover = 2

Restrict = 3

Unknown = 4

Feedback

SigCompositeType_e

`class` waveform.**SigCompositeType_e**

Apply to Fsdb Signal Composite Type

Array = 0

Struct = 1

Union = 2

TaggedUnion = 3

Record = 4

SigPowerType_e

`class` waveform.**SigPowerType_e**

Apply to Fsdb Signal Power Type

DomainState = 0

DomainUpfSimState = 1

SwitchLogicPort = 2

SwitchState = 3

SupplyNet = 4

SupplyPort = 5

SupplyState = 6

SupplyVoltage = 7

SupplySimState = 8

SupplySetState = 9

StateTable = 10

GroupState = 11

LogicNet = 12

LogicPort = 13

Unknown = 14

Feedback

SigSpiceType_e

`class` waveform.**SigSpiceType_e**

Apply to Fsdb Signal Spice Type

SpNone = 0

Logic = 1

Voltage = 2

AvgRmsCurrent = 3

Mathematics = 4

InstantaneousCurrent = 5

DiDt = 6

Power = 7

VCT Enums

VctFormat_e

`class` waveform.**VctFormat_e**

vctFormat used in return vct value format, the input of vct value format and the value format.

BinStrVal: string of binary format (for example, “1111” for 4’d15)

OctStrVal: string of decimal format (for example, “15” for 4’d15)

HexStrVal: string of hex format (for example, “f” for 4’d15)

SintVal: signed integer type (for example, -1 for 4’d15)

UintVal: unsigned integer type (for example, 15 for 4’d15)

RealVal: double type (for example, -1.234E+01)

StringVal: ASCII string type (for example, Synopsys)

EnumStrVal:string of enum literal (for example, “R” for 0 in enum {R, G, B})

Sint64Val: signed 64-bit integer type (for example, -1 for 64’d15)

Uint64Val: unsigned 64-bit integer type (for example, 15 for 64’d15)

Uint64Val: unsigned 64-bit integer type (for example, 15 for 64’d15)

BinStrVal = 0

OctStrVal = 1

DecStrVal = 2

HexStrVal = 3

SintVal = 4

UintVal = 5

RealVal = 6

StringVal = 7

EnumStrVal = 8

Sint64Val = 9

Uint64Val = 10

ObjTypeVal = 11

FT Enums

ForceTag_e

`class` waveform.**ForceTag_e**

Apply to Fsdb Force Tag

InitialForce = 0

Force = 1

Release = 2

Deposit = 3

Unknown = 4

ForceSource_e

`class` waveform.**ForceSource_e**

Apply to Fsdb Force Source

Design = 0

External = 1

Unknown = 2

Feedback

L0 APIs

File

- Function list

waveform.open(name)	Open Waveform file.
waveform.close(file)	Close Waveform file.
waveform.is_fsdb(name)	Check if the given file is FSDB file.

Example:

Following is an example showing how to open an fsdb file named `CPU.fsdb`.

example.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
fileName = "CPU.fsdb"
fileHandle = waveform.open(fileName)
if fileHandle is None:
   print("open file failed")
ret = waveform.is_fsdb(fileName)
if ret is True:
   print("this is FSDB")
waveform.close(fileHandle)
npisys.end()
```

Result:

```
this is FSDB
```

Waveform function

waveform.open(*name*)

Open Waveform file.

Parameters: name – file name

Returns:

- File object, if success
- None, if fail

Return type: class waveform.FileHandle(fileObj)

Examples

```
>>> file = waveform.open("CPU.fsdb")
```

waveform.close(file)

Close waveform file.

Parameters: file – class waveform.FileHandle(fileObj)

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples

```
>>> waveform.close(file)
```

waveform.is_fsdb(name)

Check if the given file is an FSDB file.

Parameters: name – file name

Returns:

- True, if it is FSDB.
- False, if it is not FSDB.

Return type: bool

Examples

```
>>> print(waveform.is_fsdb("CPU.fsdb"))
True
```

class waveform.**FileHandle**(fileObj)

min_time()	Get minimum time of file object.
max_time()	Get maximum time of file object.

name()	Get name of file object.
scale_unit()	Get scale unit of file object.
dump_off_range()	Get dump off range of file object.
has_seq_num()	Check if file object has sequence number.
is_completed()	Check if file object is completed.
has_glitch()	Check if file object has glitch.
has_assertion()	Check if file object has assertion type signal.
has_force_tag()	Check if file object has force tag.
has_reason_code()	Check if file object has reason code.
has_power_info()	Check if file object has power information.
version()	Get the file version.
sim_date()	Get the simulation date.
has_gate_tech()	Check if file object has FSDB-Gate technology.
top_scope_list()	Get top scope list.
top_sig_list()	Get top signal list.
add_to_sig_list(signal)	Add a signal of interest into the load list.
reset_sig_list()	Reset the load list.
load_vc_by_range(start, end)	For those signals in the load signal list, load their value changes in the specified time range into memory.
unload_vc()	Unload value changes from memory that are already loaded.
scope_by_name(name, scope=None)	Get a scope object with the specified name.
sig_by_name(name, scope=None)	Get a signal object with the specified name.
update()	Update current file object.

Feedback

Example:

file.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
# file L0 API
def test():
    fileName = "CPU.fsdb"
    fileHandle = waveform.open(fileName)
    ret = waveform.is_fsdb(fileName)
    if ret is True:
      print("this is FSDB")
    time = fileHandle.min_time()
      print("min time:"+str(time))
    time = fileHandle.max_time()
      print("max time:"+str(time))
      print("scale unit: " + fileHandle.scale_unit())
    if fileHandle.dump_off_range() is None:
      print("No dump off range")
    else:
      print("dump_off_range: " + fileHandle.dump_off_range())
    hasSeq = fileHandle.has_seq_num()
    print("has_seq_num: " + str(hasSeq))
    boolVal = fileHandle.is_completed()
    print("is_completed: " + str(boolVal))
    boolVal = fileHandle.has_glitch()
    print("has_glitch: " + str(boolVal))
    boolVal = fileHandle.has_assertion()
    print("has_assertion: " + str(boolVal))
    boolVal = fileHandle.has_force_tag()
    print("has_force_tag: " + str(boolVal))
    boolVal = fileHandle.has_reason_code()
    print("has_reason_code: " + str(boolVal))
    boolVal = fileHandle.has_power_info()
    print("has_power_info: " + str(boolVal))
    print("version: " + fileHandle.version())
    print("sim_date: " + fileHandle.sim_date())
    boolVal = fileHandle.has_gate_tech()
    print("has_gate_tech: " + str(boolVal))
    waveform.close(fileHandle)
if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('file.log', 'w')
    sys.stdout = f
    npisys.init(sys.argv)
    test()
    npisys.end()
```

```
    sys.stdout = orig_stdout
    f.close()
```

Result: file.log

```
this is FSDB
min time:0
max time:14000
scale unit: 1ns
No dump off range
has_seq_num: True
is_completed: True
has_glitch: True
has_assertion: False
has_force_tag: False
has_reason_code: False
has_power_info: False
version: 4.3
sim_date: Tue Jun 8 17:40:56 2010
has_gate_tech: False
```

update()

Update the current file object.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> print(file.update())
True
```

min_time()

Get the minimum time of file object.

Returns:

- Time, if success.
- None, if fail.

Return type: int

Examples:

```
>>> print(file.min_time())
0
```

Feedback

max_time()

Get the maximum time of file object.

Returns:

- Time, if success.
- Time, if success.
- None, if fail.

Return type: int

Examples:

```
>>> print(file.max_time())
14000
```

name()

Get the name of file object.

Returns:

- File name, if success.
- None, if fail.

Return type: str

Examples:

```
>>> print(file.name())
./myFolder/CPU.fsdb
```

scale_unit()

Get the scale unit of file object.

Returns:

- Scale unit, if success.
- None, if fail.

Return type: str

Examples:

```
>>> print(file.scale_unit())
1ns
```

Feedback

dump_off_range()

Get the dump off range of file object.

Returns:

- Dump off range, if success.
- None, if fail.

Return type: str

Examples:

```
>>> print(file.dump_off_range())
None
```

has_seq_num()

Check if the file object has sequence number.

Returns:

- True, if it has sequence number.
- False, if it does not have sequence number.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.has_seq_num())
True
```

is_completed()

Check if the file object is completed.

Returns:

- True, if it is completed.
- False, if it is not completed.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.is_completed())
True
```

Feedback

has_glitch()

Check if the file object has glitch.

Returns:

- True, if it has glitch.
- False, if it does not have glitch.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.has_glitch())
True
```

has_assertion()

Check if the file object has assertion type signal.

Returns:

- True, if it has assertion type signal.
- False, if it does not have assertion type signal.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.has_assertion())
False
```

has_force_tag()

Check if the file object has force tag.

Returns:

- True, if it has force tag.
- False, if it does not have force tag.
- None, if fail.

Return type: bool

Feedback

Examples:

```
>>> print(file.has_force_tag())
False
```

has_reason_code()

Check if the file object has reason code.

Returns:

- True, if it has reason code.
- False ,if it does not have reason code.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.has_reason_code())
False
```

has_power_info()

Check if the file object has power information.

Returns:

- True, if it has power information.
- False, if it does not have power information.
- None, if fail

Return type: bool

Examples:

```
>>> print(file.has_power_info())
False
```

version()

Get the file version.

Returns:

- File version, if success.
- None, if fail.

Return type: str

Feedback

Examples:

```
>>> print(file.version())
4.3
```

sim_date()

Get the simulation date.

Returns:

- Simulation date, if success.
- None, if fail.

Return type: str

Examples:

```
>>> print(file.sim_date())
Tue Jun 8 17:40:56 2010
```

has_gate_tech()

Check if the file object has FSDB-Gate technology.

Returns:

- True, if it has gate technology.
- False, if it does not have gate technology.
- None, if fail.

Return type: bool

Examples:

```
>>> print(file.has_gate_tech())
False
```

top_scope_list()

Get the top scope list.

Returns:

- List of top scope if success.
- Empty list if fail.

Return type: ScopeHandle list

Examples

```
>>> scope_list = file.top_scope_list()
for scope in scope_list:
print(scope.name())
tb_CPUsystem
dump_fsdb
```

Detail example at class waveform.ScopeHandle(scopeObj).

top_sig_list()

Get top signal list.

Returns:

- List of top signal if success.
- Empty list if fail.

Return type: class waveform.ScopeHandle(scopeObj) list

Examples:

```
>>> topSigFileHandle = waveform.open("top_sig.fsdb")
signalList = topSigFileHandle.top_sig_list()
for signal in signalList:
print(signal.name())
realSig
```

Detail example at class waveform.SigHandle(signalObj).

add_to_sig_list(signal)

Add a signal of interest into the load list.

Parameters: signal – The target signal object.

***Returns**:*

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> print(file.add_to_sig_list(sig))
True
```

Detail example at class waveform.SigHandle(signalObj)

reset_sig_list()

Reset the load list.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> print(file.reset_sig_list())
True
```

Detail example at class waveform.SigHandle(signalObj).

load_vc_by_range(start, end)

For those signals in the load list, load their value changes in the specified time range into memory.

Parameters:

- **start** – The start time to load vc.
- **end** – The end time to load vc.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> print(file.load_vc_by_range(10, 2000))
True
```

Detail example at class waveform VctHandle(vctObj).

unload_vc()

Unload the value changes from memory.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples

Feedback

```
>>> print(file.unload_vc())
True
```

Detail example at class waveform VctHandle(vctObj).

scope_by_name(*name, scope=None*)

Get a scope object with the specified name.

Parameters:

- **name** – The string representing the scope name (e.g. top.subscope1.subscope2). (The scope delimiter is fixed to ".")
- **scope** – A scope object for localizing the search space. (If the scope object is none, this function searches the scope name from the root space)

Returns:

- Scope object, if success.
- None, if fail.

Return type: ScopeHandle

Examples

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.name())
i_BJsource
```

Detail example at class waveform.SigHandle(signalObj).

sig_by_name(*name, scope=None*)

Get a signal object with the specified name.

Parameters:

name – The string representing the signal name.

scope – A scope object for localizing the search space. (If the scope object is none, this function searches the signal name from the root space).

Returns:

- Signal object, if success.
- None, if fail.

Return type: class waveform.SigHandle(signalObj)

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
print(signal.name())
Card_temp
```

Detail example at class waveform.SigHandle(signalObj).

Scope

class waveform.**ScopeHandle**(*scopeObj*)

ScopeHandle Function list:

name()	Get the name of scope object.
full_name()	Get the full name of scope object.
def_name()	Get the defined name of scope object.
type([isEnum])	Get the scope type of scope object.
parent()	Get the parent scope.
child_scope_list()	Get the child scope list.
sig_list()	Get the signal list.
file()	Get the file object.

Example:

scope.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
def print_scope(scope):
    print("name: " + scope.name())
    print("full name: " + scope.full_name())
    print("def name: " + scope.def_name())
    print("type(enum): ", scope.type(True))
    print("type(string): " + scope.type(False))
    if scope.parent() is None:
        print("no parent")
    else:
```

Feedback

```
        print("parent name: " + str(scope.parent().name()))
    print("file version:", scope.file().version())
def test():
fileHandle = waveform.open("CPU.fsdb")
# FileHandle API: to get top scope list
scopeList = fileHandle.top_scope_list()
scope = scopeList[0]
    print("[ top scope ]")
    print_scope(scope)
    print("child")
    childs = scope.child_scope_list()
    child = childs[0]
    print(child.name())
    print(child.parent().name())
    print("[scope by name]")
    scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
      print(scope.name())
    waveform.close(fileHandle)
if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('scope.log', 'w')
    sys.stdout = f
    npisys.init(sys.argv)
    test()
    npisys.end()
    sys.stdout = orig_stdout
    f.close()
```

Result: scope.log

```
[ top scope ]
name: tb_CPUsystem
full name: tb_CPUsystem
def name: tb_CPUsystem
type(enum):ScopeType_e.SvModule
type(string): npiFsdbScopeSvModule
parent name: None
file: 4.3
child
i_CPUsystem
tb_CPUsystem
[scope by name]
i_CPUsystem
```

name()

Get the name of scope object.

Returns:

- The scope name, if success.
- None, if fail.

Feedback

Return type: str

Examples:

```
>>> scope_list = file.top_scope_list()
for scope in scope_list:
print(scope.name())
tb_CPUsystem
dump_fsdb
```

full_name()

Get the full name of scope object.

Returns:

- The full name of scope object, if success.
- None, if fail.

Return type: str

Examples:

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.full_name())
tb_CPUsystem.i_BJsource
```

def_name()

Get the defined name of scope object.

Returns:

- The defined name, if success.
- None, if fail.

Return type: str

Examples:

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.def_name())
BJsource
```

type(*isEnum=True*)

Get the scope type of scope object.

Parameters: isEnum – Specify the type in enum or string.

Returns:

- If isEnum is True:

  The scope type, if success.

  None, if fail.

- If isEnum is False:

  The string_value, if success.

  None, if fail.

Return type: ScopeType_e/str

Examples

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.scope.type(True))
print(scope.scope.type(False))
ScopeType_e.SvModule
npiFsdbScopeSvModule
```

parent()

Get the parent scope.

Returns:

- The parent scope, if success.
- None, if fail.

Return type: class waveform.ScopeHandle(scopeObj)

Examples:

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.parent().name())
tb_CPUsystem
```

child_scope_list()

Get the child scope list.

Returns:

- The list of child scope, if success.
- Empty list, if fail.

Return type: class waveform.ScopeHandle(scopeObj) list

Feedback

Examples:

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
scopeList = scope.child_scope_list()
```

sig_list()

Get the signal list.

Returns:

- The signal list, if success.
- Empty list, if fail.

Return type: class waveform.SigHandle(signalObj) list

Examples:

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem")
signalList = scope.sig_list()
print(signalList[0].name())
NextCard
```

Detail example at class waveform.SigHandle(signalObj).

file()

Get the file object.

Returns:

- The file object, if success.
- None, if fail.

Return type: class waveform.FileHandle(fileObj)

Examples

```
>>> print(signal.file().name())
./myFoloder/CPU.fsdb
```

Signal

class waveform.**SigHandle**(*signalObj*)

SigHandle Function list:

name()	Get the name of signal.
full_name()	Get the full name of signal.

is_real()	Check if signal is real type.
has_member()	Check if signal has member.
left_range()	Get the left range of signal.
right_range()	Get the right range of signal.
is_string()	Check if signal is string type.
direction(isEnum=True)	Get the direction of signal.
assertion_type(isEnum=True)	Get the assertion type.
composite_type(isEnum=True)	Get the composite type.
is_packed()	Check if signal is packed.
has_reason_code	Check if signal has reason code.
reason_code()	Get the reason code of signal.
reason_code_desc()	Reason code description.
is_param()	Check if signal is parameter.
has_enum()	Check if signal has enum member.
power_type(isEnum=True)	Get the power type.
has_force_tag()	Check if signal has force tag.
sp_type(isEnum=True)	Get the spice type.
scope()	Get the scope object that this signal belongs to.
parent_sig()	Get the parent signal.
scope()	Get the file object of signal.
member_list()	Get a list of signal members.
create_vct()	Create value change traverse object.
create_ft()	Create force tag traverse object.

Example

sig.py

Feedback

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
fileHandle = waveform.open("CPU.fsdb")
    def print_scope(scope):
    print("name: " + scope.name())
    print("full name: " + scope.full_name())
    print("def name: " + scope.def_name())
    print("type(enum): ", scope.type(True))
    print("type(string): " + scope.type(False))
    if scope.parent() is None:
        print("no parent")
    else:
        print("parent name: " + str(scope.parent().name()))
        print("file:", scope.file().name())
        print("////////////////////////")
def print_signal(signal):
    print("[sigName]:"+signal.name())
    print("fullName:"+signal.full_name())
    print("isreal:" + str(signal.is_real()))
    print("hasMember: "+str(signal.has_member()))
    print("left_range: "+str(signal.left_range()))
    print("right_range: "+str(signal.right_range()))
    print("range_size: "+str(signal.range_size()))
    print("is_string: "+str(signal.is_string()))
    print("direction(enum): ", str(signal.direction(True)))
    print("direction(str): ", str(signal.direction(False)))
    print("assertion_type(enum): ", str(signal.assertion_type(True)))
    print("assertion_type(str): ", str(signal.assertion_type(False)))
    print("composite_type(enum): ", str(signal.composite_type(True)))
    print("composite_type(str): ", str(signal.composite_type(False)))
    print("is_packed: "+str(signal.is_packed()))
    print("has_reason_code: "+str(signal.has_reason_code()))
    print("reason_code: "+str(signal.reason_code()))
    print("reason_code_desc: "+str(signal.reason_code_desc()))
    print("is_param: "+str(signal.is_param()))
    print("has_enum: "+str(signal.has_enum()))
    print("power_type(enum): "+str(signal.power_type(True)))
    print("power_type(str): "+str(signal.power_type(False)))
    print("has_force_tag: "+str(signal.has_force_tag()))
print("file: ",signal.file().name())
parentSig = signal.parent_sig()
if parentSig is not None:
print("parent_sig: ", signal.parent_sig().name())
print("top signal list -------------")
topSigFileHandle = waveform.open("top_sig.fsdb")
signalList = topSigFileHandle.top_sig_list()
print_signal(signalList[0])
print("----------------------------")
```

Feedback

```
print("scope's signal list -------------")
scope = fileHandle.scope_by_name("tb_CPUsystem")
signalList = scope.sig_list()
print_signal(signalList[0])
print("----------------------------")
sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
if signal is None:
print("signal is None")
fileHandle.add_to_sig_list(signal)
print_signal(signal)
print("signal.scope----------------------")
print_scope(signal.scope())
print("-----------------------------------")
print("signal.member_list----------------------")
sigList = signal.member_list()
print_signal(sigList[0])
print("-----------------------------------")
fileHandle.reset_sig_list()
waveform.close(fileHandle)
waveform.close(topSigFileHandle)
npisys.end()
```

Result

```
top signal list -------------
[sigName]:realSig
fullName:realSig
isreal:True
hasMember: False
left_range: 0
right_range: 0
range_size: 1
is_string: False
direction(enum): DirType_e.DirNone
direction(str): npiFsdbDirNone
assertion_type(enum): None
assertion_type(str): None
composite_type(enum): None
composite_type(str): None
is_packed: False
has_reason_code: False
reason_code: None
reason_code_desc: None
is_param: False
has_enum: False
power_type(enum): None
power_type(str): None
has_force_tag: False
file: /remote/us01home53/peichun/Ted_code/pynpi_demo/writer_L1.fsdb
----------------------------
scope's signal list -------------
[sigName]:NextCard
```

```
fullName:tb_CPUsystem.NextCard
isreal:False
hasMember: False
left_range: 0
right_range: 0
range_size: 1
is_string: False
direction(enum): DirType_e.DirNone
direction(str): npiFsdbDirNone
assertion_type(enum): None
assertion_type(str): None
composite_type(enum): None
composite_type(str): None
is_packed: False
has_reason_code: False
reason_code: None
reason_code_desc: None
is_param: False
has_enum: False
power_type(enum): None
power_type(str): None
has_force_tag: False
file: /remote/us01home53/peichun/Ted_code/pynpi_demo/CPU.fsdb
----------------------------
[sigName]:Card_temp
fullName:tb_CPUsystem.i_BJsource.Card_temp
isreal:False
hasMember: True
left_range: 0
right_range: 3
range_size: 4
is_string: False
direction(enum): DirType_e.DirNone
direction(str): npiFsdbDirNone
assertion_type(enum): None
assertion_type(str): None
composite_type(enum): SigCompositeType_e.Array
composite_type(str): npiFsdbSigCtArray
is_packed: False
has_reason_code: False
reason_code: None
reason_code_desc: None
is_param: False
has_enum: False
power_type(enum): None
power_type(str): None
has_force_tag: False
file: /remote/us01home53/peichun/Ted_code/pynpi_demo/CPU.fsdb
signal.scope-----------------------
name: i_BJsource
full name: tb_CPUsystem.i_BJsource
def name: BJsource
type(enum): ScopeType_e.SvModule
```

Feedback

```
type(string): npiFsdbScopeSvModule
parent name: tb_CPUsystem
file: /remote/us01home53/peichun/Ted_code/pynpi_demo/CPU.fsdb
//////////////////////
-----------------------------------
signal.member_list----------------------
[sigName]:Card_temp[0]
fullName:tb_CPUsystem.i_BJsource.Card_temp[0]
isreal:False
hasMember: False
left_range: 3
right_range: 0
range_size: 4
is_string: False
direction(enum): DirType_e.DirNone
direction(str): npiFsdbDirNone
assertion_type(enum): None
assertion_type(str): None
composite_type(enum): None
composite_type(str): None
is_packed: True
has_reason_code: False
reason_code: None
reason_code_desc: None
is_param: False
has_enum: False
power_type(enum): None
power_type(str): None
has_force_tag: False
file: /remote/us01home53/peichun/Ted_code/pynpi_demo/CPU.fsdb
parent_sig: Card_temp
-----------------------------------
```

name()

Get the name of signal.

Returns:

- Signal name,if success.
- None, if fail.

Return type : str

Examples

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
print(signal.name())
Card_temp
```

Feedback

full_name()

Get the full name of signal.

Returns:

- Signal full name, if success.
- None, if fail.

Return type: str

Examples

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
print(signal.full_name())
tb_CPUsystem.i_BJsource.Card_temp
```

is_real()

Check if the signal is real type.

Returns:

- True, if signal is real type.
- False, if signal is not real type.
- None, if fail.

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.is_real())
False
```

has_member()

Check if signal has member.

Returns:

- True, if signal has member.
- False, if signal does not have member.
- None ,if fail.

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.has_memeber())
True
```

left_range()

Get the left range of signal.

Returns:

- left_range, if success.
- None, if fail.

Return type: int

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.left_range())
0
```

right_range()

Get the right range of signal.

Returns:

- right_range, if success.
- None, if fail.

Return type: int

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.right_range())
3
```

is_string()

Check if the signal is string type.

Returns:

- True, if signal is string type.
- False, if signal is not string type.
- None, if fail.

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.is_string())
False
```

direction(*isEnum=True*)

Get the direction of signal.

Parameters: isEnum – Specify if direction is enum type or string.

Returns:

- If isEnum is True:

  enum_value, if success.

  None, if fail.

- If isEnum is False:

  string_value, if success.

  None, if fail.

Return type: DirType_e/str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.direction(True))
print(sig.direction(False))
DirType_e.None
npiFsdbDirNone
```

assertion_type(*isEnum=True*)

Get the assertion type.

Parameters: isEnum – Specify if assertion type is enum type or string.

Returns:

- If isEnum is True:

  enum_value, if success.

  None, if fail.

Feedback

- If isEnum is False:

  string_value, if success.

  None, if fail.

Return type: SigAssertionType_e/str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.assertion_type(True))
print(sig.assertion_type(False))
SigAssertionType_e.None
None
```

composite_type(*isEnum=True*)

Get the composite type.

Parameters: isEnum – Specify if compositye type is in enum or string.

Returns:

- If isEnum is True:

  enum_value, if success.

  None, if fail.

- If isEnum is False:

  string,_value, if success.

  None if fail.

Return type: SigCompositeType_e/str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.composite_type(True))
print(sig.composite_type(False))
SigCompositeType_e.Array
npiFsdbSigCtArray
```

is_packed()

Check if signal is packed.

Feedback

Returns:

- True, if signal is packed.
- False, if signal is not packed.
- None, if fail.

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.is_packed())
True
```

has_reason_code()

Check if signal has reason code.

Returns:

- True, if signal has reason code.
- False, if signal does not have reason code.
- None, if fail

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.has_reason_code())
False
```

reason_code()

Get the reason code of signal.

Returns:

- Reason code, if success.
- None, if fail.

Return type: str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.reason_code())
None
```

Feedback

reason_code_desc()

Reason code description.

Returns:

- The description of signals' reason code, if success.
- False, if fail.

Return type: str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.reason_code_desc())
None
```

is_param()

Check if signal is parameter.

Returns:

- True, if this signal is a pararmeter.
- False, if this signal is not a pararmeter.
- None, if fail

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.is_param())
False
```

has_enum()

Check if signal has enum member.

Returns:

- True, if this signal has enum.
- False, if this signal does not have enum.
- None, if fail.

Return type: bool

Feedback

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.has_enum())
False
```

power_type(*isEnum=True*)

Get the power type.

Parameters: isEnum – Specify if power type is enum type or string.

Returns:

- If isEnum is True:

  enum_value, if success.

  None, if fail.

- If isEnum is False:

  string_value, if success.

  None, if fail.

Return type: SigPowerType_e/str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.power_type(True))
print(sig.power_type(False))
None
None
```

has_force_tag()

Check if signal has force tag.

Returns:

- True, if this signal has force tag.
- False, if this signal does not have force tag.
- None, if fail.

Return type: bool

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.has_force_tag())
False
```

sp_type(*isEnum=True*)

Get the spice type.

This API's spec is not draft and the `string_value` support when the spec draw up.

Parameters: isEnum – Specify if power type is enum type or string.

Returns:

- If isEnum is True:

  enum_value, if success.

  None, if fail.

- If isEnum is False:

  string_value, if success.

  None, if fail.

Return type: SigSpiceType_e/str

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.sp_type(True))
print(sig.sp_type(False))
SigSpiceType_e.SpNone
None
```

scope()

Get the scope object that this signal belongs to.

Returns:

- Scope object, if success.
- None, if fail.

Return type: ScopeHandle

Feedback

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.scope().full_name())
tb_CPUsystem.i_BJsource
```

parent_sig()

Get the parent signal.

Returns:

- Parent signal object, if success.
- None, if fail.

Return type: SigHandle

Examples:

```
>>> signal =
 fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp[0]")
print(sig.parent_sig().name())
Card_temp
```

file()

Get the file object of signal.

Returns:

- File object, if success.
- None, if fail.

Return type: FileHandle

Examples:

```
>>> signal = fileHandle.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
print(sig.file().version())
4.3
```

member_list()

Get a list of signal members.

Returns:

- List of member, if success.

  Empty list, if fail.

Return type: SigHandle member list

Feedback

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
sigList = signal.member_list()
for sig in sigList:
print(sig.name())
Card_temp[0]
Card_temp[1]
Card_temp[2]
Card_temp[3]
```

create_vct()

Create value change traverse object.

Returns:

- Value change traverse object, if success.
- None, if fail.

Return type: VctHandle

Examples:

```
>>> signal = file.sig_by_name("tb_CPUsystem.i_BJsource.Card_temp")
vct = signal.create_vct()
Detail example at VctHandle
```

create_ft()

Create force tag traverse object.

Returns:

- Force tag traverse object, if success.
- None, if fail.

Return type: FtHandle

Examples:

```
>>> signal = file.sig_by_name("top.siga")
ft = signal.create_ft()
Detail example at FtHandle
```

---

vct (value change traverse)

class waveform.**VctHandle**(*vctObj*)

Feedback

VctHandle Function list:

time()	Get the current time.
value(format=<VctFormat_e.ObjTypeVal: 11>)	Get the current value.
format()	Get the VctHandle default format.
seq_num()	Get the sequence number.
port_value()	Get the port value.
goto_next()	Increase the index of the value change traverse object if possible.
goto_prev()	Decrease the index of the value change traverse object if possible.
goto_first()	Move the index of the value change traverse object to the first value change if possible.
goto_time(time)	Change the index of the value change traverse object to the last vc at the specified time.
duration()	Get the current time duration of the value change traverse object (for assertion signals).
sig()	Get the signal object of current VC object.
release()	Free the value change traverse handle.

Example:

vct.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
fileHandle = waveform.open("CPU.fsdb")
def print_vct(vct):
    print("time:", vct.time())
    print("value:", vct.value(waveform.VctFormat_e.DecStrVal))
    print("format:", vct.format())
    print("seq num:", vct.seq_num())
    print("sig:", vct.sig().name())
```

```
    print("duration", vct.duration())
def print_sig_vct(sigName, file):
    print("The signal is:", sigName)
    signal = file.sig_by_name(sigName)
    vct = signal.create_vct()
    file.load_vc_by_range(0, 1000)
    if vct is None:
    print('vct is None')
    ret = vct.goto_first()
    if ret is False:
    print("goto_first failed")
    while True:
    ret = vct.goto_next()
    print_vct(vct)
    if ret is False:
    break
    vct.release()
    vct = None
    file.unload_vc()
    def print_sig_vct_reverse(sigName, file):
    print("The signal is:", sigName)
    signal = file.sig_by_name(sigName)
    vct = signal.create_vct()
    file.load_vc_by_range(0, 1000)
    if vct is None:
    print('vct is None')
    ret = vct.goto_time(1000)
    if ret is False:
    print("goto_time failed")
    while True:
    ret = vct.goto_prev()
    print_vct(vct)
    if ret is False:
    break
    vct.release()
    vct = None
    file.unload_vc()
sigName = "tb_CPUsystem.i_BJsource.Card_temp"
print("vct start----------------------")
print_sig_vct(sigName, fileHandle)
print_sig_vct("tb_CPUsystem.NewCard", fileHandle)
print_sig_vct_reverse("tb_CPUsystem.NewCard", fileHandle)
print("-----------------------------------")
waveform.close(fileHandle)
npisys.end()
```

Result:

```
vct start----------------------
The signal is: tb_CPUsystem.i_BJsource.Card_temp
time: 0
value: {4,11,7,10}
format: VctFormat_e.BinStrVal
```

```
seq num: 563
sig: Card_temp
duration None
The signal is: tb_CPUsystem.NewCard
time: 350
value: 1
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 450
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 650
value: 1
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 750
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 950
value: 1
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 1050
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 1050
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
The signal is: tb_CPUsystem.NewCard
time: 750
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
```

Feedback

```
time: 650
value: 1
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 450
value: 0
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 350
value: 1
format: VctFormat_e.BinStrVal
seq num: 8
sig: NewCard
duration None
time: 0
value: 0
format: VctFormat_e.BinStrVal
seq num: 10
sig: NewCard
duration None
time: 0
value: 0
format: VctFormat_e.BinStrVal
seq num: 10
sig: NewCard
duration None
-----------------------------------
```

time()

Get the current time.

Returns:

- Time, if success.
- None, if fail.

Return type: int

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
ret = vct.goto_time(1000)
```

Feedback

```
if ret is True:
print(vct.time())
1000
```

value(format=<VctFormat_e.ObjTypeVal: 11>)

Get the current value.

Parameters: format – VctFormat_e.

Returns: Value with the specified format.

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
ret = vct.goto_time(1000)
if ret is True:
print(vct.value(waveform.VctFormat_e.BinStrVal))
{0100,1011,0111,1010}
```

format()

Get the VctHandle default format.

Returns:

- Format VctFormat_e, if success.
- None, if fail.

Return type: VctFormat_e

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
ret = vct.goto_time(1000)
if ret is True:
print(vct.format())
VctFormat_e.BinStrVal
```

seq_num()

Get the sequence number.

Feedback

Returns: Sequence number of current VC.

Return type: int

Examples:

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = file.sig_by_name(sigName)
vct = signal.create_vct()
print(vct.seq_num())
563
```

port_value()

Get the port value.

Returns:

- [state(string), s0(string), s1(string)], if success.
- None, if fail.

Return type: list

Examples:

```
>>> ret = vct.port_value()
print(ret)
['D','0','6']
```

goto_next()

Increase the index of the value change traverse object if possible.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
vct.goto_time(0)
ret = vct.goto_next()
if ret is True:
print(vct.value())
1
```

Feedback

goto_prev()

Decrease the index of the value change traverse object if possible.

VcIterator:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
vct.goto_time(1000)
ret = vct.goto_prev()
if ret is True:
print(vct.value())
1
```

goto_first()

Move the index of the value change traverse object to the first value change if possible.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
ret = ect.goto_first()
if ret is True:
print(vct.value())
1
```

goto_time(time)

Change the index of the value change traverse object to the last vc at the specified time.

Parameters: time – The target time.

Feedback

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
ret = vct.goto_time(1000)
if ret id True:
print(vct.value())
1
```

duration()

Get the current time duration of the value change traverse object (for assertion signals).

Returns:

- begin_time, end_time if success.
- None if fail.

Return type: int list

sig()

Get the signal object of current VC object.

Returns:

- Signal object, if success.
- None, if fail.

Return type: SigHandle

Examples:

```
>>> signal = file.sig_by_name(sigName)
vct = signal.create_vct()
file.load_vc_by_range(0, 1000)
if vct is None:
print('vct is None')
print(vct.sig().is_packed())
False
```

Feedback

release()

Free the value change traverse handle.

Examples:

```
>>> signal = file.sig_by_name("tb_CPUsystem.NewCard")
vct = signal.create_vct()
vct.release()
vct = None
```

---

ft (force tag traverse)

class waveform.**FtHandle**(*ftObj*)

FtHandle Function list:

time()	Get the current time.
value()	Get the current value.
goto_next()	Increase the index of the force tag traverse object if possible.
goto_prev()	Decrease the index of the force tag traverse object if possible.
goto_first()	Move the index of the force tag traverse object to the first value change if possible.
goto_time(time)	Change the index of the force tag traverse object to the last vc at the specified time.
release()	Free the force tag traverse handle.

Example:

fg.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
def print_ft(ft):
    print("time:", ft.time())
    val = ft.value()
    if val is None:
        print("No value")
```

Feedback

```
    else:
        print("value:", val)
def print_sig_ft(sigName, file):
    print("The signal is:", sigName)
    signal = file.sig_by_name(sigName)
    ft = signal.create_ft()
    if ft is None:
    print('ft is None')
    ret = ft.goto_first()
    if ret is False:
    print("goto_first failed")
    while True:
    ret = ft.goto_next()
    print_ft(ft)
    if ret is False:
    break
    ft.release()
    ft = None
def print_sig_ft_reverse(sigName, file):
    print("The signal is:", sigName)
    signal = file.sig_by_name(sigName)
    ft = signal.create_ft()
    if ft is None:
    print('ft is None')
    ret = ft.goto_time(100)
    if ret is False:
    ret = ft.goto_prev()
    print_ft(ft)
    if ret is False:
    break
    ft.release()
    ft = None
def test():
    fileName = "test_force.fsdb"
    fileHandle = waveform.open(fileName)
    sigName = "top.siga"
    print_sig_ft(sigName, fileHandle)
    print_sig_ft_reverse(sigName, fileHandle)
    waveform.close(fileHandle)
    if __name__ == '__main__':
    npisys.init(sys.argv)
    test()
    npisys.end()
```

Result

```
The signal is: top.siga
time: 30
value: [<ForceTag_e.Force: 1>, <ForceSource_e.Design: 0>, 'test.v', 81]
time: 30
value: [<ForceTag_e.Release: 2>, <ForceSource_e.External: 1>, None, 0]
time: 40
value: [<ForceTag_e.Force: 1>, <ForceSource_e.Design: 0>, 'test.v', 84]
```

```
time: 40
value: [<ForceTag_e.Force: 1>, <ForceSource_e.Design: 0>, 'test.v', 85]
time: 50
value: [<ForceTag_e.Release: 2>, <ForceSource_e.External: 1>, None, 0]
time: 50
No value
The signal is: top.siga
time: 50
value: [<ForceTag_e.Release: 2>, <ForceSource_e.External: 1>, None, 0]
```

time()

Get the current time.

Returns:

- Time, if success.
- None, if fail.

Return type: int

Examples

```
>>> ft = signal.create_ft()
if ft is None:
print('ft is None')
ret = ft.goto_first()
if ret is True:
print(ft.time())
30
```

value()

Get the current value.

Returns:

- [tag, source, file_name, line_num]
- tag:
  - The force tag enum value.
  - None, if fail.
- source:
  - The force tag source enum value.
  - None, if fail.

Feedback

- file_name:
  - The file name of current force tag.
  - None if fail, or there has no file_name.
- line_num:
  - The line number of current force tag.
  - 0, if no line_num.
  - None, if fail.

Return type: [ForceTag_e , ForceSource_e, str, int]

Examples:

```
>>> ft = signal.create_ft()
if ft is None:
  print('ft is None')
ret = ft.goto_first()
if ret is True:
  print(ft.value())
[<ForceTag_e.Release: 2>,<ForceSource_e.External: 1>, 'CPU.v', 0]
```

goto_next()

Increase the index of the force tag traverse object if possible.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> ft = signal.create_ft()
if ft is None:
  print('ft is None')
ft.goto_first()
ret = ft.goto_next()
if ret is True:
  print(ft.value())
[<ForceTag_e.InitialForce: 0>,<ForceSource_e.External: 1>, 'CPU.v', 18]
```

goto_prev()

Decrease the index of the force tag traverse object if possible.

Feedback

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> ft = signal.create_ft()
if ft is None:
  print('ft is None')
ft.goto_time(100)
ret = ft.goto_prev()
if ret is True:
  print(ft.value())
[<ForceTag_e.Release: 2>,<ForceSource_e.Unknown: 2>, 'CPU.v', 0]
```

goto_first()

Move the index of the force tag traverse object to the first value change if possible.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> ft = signal.create_ft()
if ft is None:
  print('ft is None')
ft.goto_time(100)
ret = ft.goto_prev()
if ret is True:
  print(ft.value())
[<ForceTag_e.Release: 2>,<ForceSource_e.Unknown: 2>, 'CPU.v', 0]
```

goto_time(time)

Change the index of the force tag traverse object to the last vc at the specified time.

Parameters: time – The target time.

Returns:

- True, if success.
- False, if fail.

Feedback

Return type: bool

Examples:

```
>>> ft = signal.create_ft()
if ft is None:
  print('ft is None')
ret = ft.goto_time(i00)
if ret is True:
  print(ft.value())
[<ForceTag_e.InitialForce: 0>,<ForceSource_e.Design: 0>, 'CPU.v', 0]
```

release()

Free the force tag traverse handle.

Examples:

```
>>> signal = file.sig_by_name("top.siga")
ft = signal.create_ft()
ft.release()
ft = None
```

vc Iterator

Time Based

class waveform.**TimeBasedHandle**

TimeBased VC Iterator

add(signal,filterEq=False)	Add a signal to the iterator.
iter_start(beginTime, endTime)	Specify the time range to iterate value changes.
iter_next()	Iterate value changes between begin time and end time in a time-based way.
iter_stop()	Stop the iteration.
get_value(format=<VctFormat_e.BinStrVal: 0>)	Get the value of current value changes.
set_max_session_load(num)	Set the max session load number to control memory.

Feedback

Example:

TimeBased.py

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
fileHandle = waveform.open("CPU.fsdb")
tb = waveform.TimeBasedHandle()
sigName = "tb_CPUsystem.NewCard"
signal = fileHandle.sig_by_name(sigName)
tb.add(signal)
signal = fileHandle.sig_by_name(sigName)
tb.add(signal)
tb.iter_start( 0, 500 )
tb.set_max_session_load(6)
currTime = 0
while True:
    idx = 0
    idx,currTime = tb.iter_next()
    if idx == 0:
        break;
    print("idx", idx, "time:", currTime, "value:",
 tb.get_value(waveform.VctFormat_e.BinStrVal))
tb.iter_stop()
waveform.close(fileHandle)
npisys.end()
```

Result

```
idx 1 time: 0 value: 0
idx 2 time: 0 value: 0
idx 2 time: 350 value: 1
idx 1 time: 350 value: 1
idx 1 time: 450 value: 0
idx 2 time: 450 value: 0
```

add(*signal, filterEq=False*)

Add a signal to the iterator.

Note:
all signals added to iterator should be from the same waveform file object.

Parameters

Feedback

- **signal** – A waveform signal object class waveform.SigHandle(signalObj) which will be traversed later in this iterator.
- **filterEq** – Indicate if filtering out equivalent signals (False: not filter, True: filter).

Returns:

- On success, return a number greater than 0. (Note: when filterEq is 1, the return number for equivalent signals will be the same).
- On failure, return 0.

Return type: int

Examples:

```
>>> As TimeBased.py shown
iter_start(beginTime, endTime)
```

iter_start(beginTime, endTime)

Specify the time range to iterate value changes. (All interesting signals should be added before calling this API).

Parameters:

- **beginTime** – The begin time of vc iteration.
- **endTime** – The end time of vc iteration.

Examples:

```
>>> As TimeBased.py shown
```

iter_next()

Iterate value changes between begin time and end time in a time-based way.

Returns:

- [signal_id, time], if success.
- [signal_id, time], if fail.

Return type: [int, int]

Examples:

```
>>> As TimeBased.py shown
```

iter_stop()

Stop the iteration.

Feedback

Examples:

```
>>> As TimeBased.py shown
```

get_value(*format=<VctFormat_e.BinStrVal: 0>*)

Get the value of current value changes. (The spec. of format setting is the same as VctHandle.value()).

Parameters: format – VctFormat_e

Returns:

- The current value with the specified format if success.
- None if fail.

Examples:

```
>>> As TimeBased.py shown
```

set_max_session_load(num)

Set the max session load number to control memory. (by default, the number is 0 and the iterator will load all value changes in the specified time range in one time).

Parameters: num – Maximum loaded session number.

Examples:

```
>>> As TimeBased.py shown
```

Signal Based

*class waveform.***SigBasedHandle**

SigBased VC Iterator

add(signal,filterEq=False)	Add a signal to the iterator.
iter_start(beginTime, endTime)	Specify the time range to iterate value changes.
iter_next()	Iterate value changes between begin time and end time in a signal-based way.
iter_stop()	Stop iteration.

Feedback

get_value(format=<VctFormat_e.BinStrVal: 0>)	Get the value of current value changes.
set_max_session_load(num)	Set the max session load number to control memory.

Example

SigBased.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
fileHandle = waveform.open("CPU.fsdb")
tb = waveform.SigBasedHandle()
sigName = "tb_CPUsystem.NewCard"
signal = fileHandle.sig_by_name(sigName)
tb.add(signal)
sigName = "tb_CPUsystem.i_BJkernel.Total"
signal = fileHandle.sig_by_name(sigName)
tb.add(signal)
tb.iter_start( 0, 1000 )
tb.set_max_session_load(6)
currTime = 0
while True:
idx = 0
idx,currTime = tb.iter_next()
if idx == 0:
break;
print("idx", idx, "time:", currTime, "value:",
 tb.get_value(waveform.VctFormat_e.BinStrVal))
tb.iter_stop()
waveform.close(fileHandle)
npisys.end()
```

Result

```
idx: 1 time: 0 value: 0
idx: 1 time: 350 value: 1
idx: 1 time: 450 value: 0
idx: 1 time: 650 value: 1
idx: 1 time: 750 value: 0
idx: 1 time: 950 value: 1
idx: 2 time: 0 value: xxxxx
idx: 2 time: 300 value: 00000
```

```
idx: 2 time: 500 value: 00100
idx: 2 time: 800 value: 01111
```

add(*signal, filterEq=False*)

Add a signal to the iterator.

Note:
All the signals added to iterator must be from the same waveform file object.

Parameters:

- **signal** – A waveform signal object SigHandle which will be traversed later in this iterator.
- **filterEq** – Indicate if filtering out equivalent signals (False: not filter, True: filter).

Returns:

- On success, return a number greater than 0. (Note: when filterEq is 1, the return number for equivalent signals will be the same).
- On failure, return 0.

Return type: int

Examples

```
>>> As SigBased.py shown
```

iter_start(*beginTime, endTime*)

Specify the time range to iterate value changes. (All interesting signals should be added before calling this API).

Parameters:

- **beginTime** – The begin time of vc iteration.
- **endTime** – The end time of vc iteration.

Examples:

```
>>> As SigBased.py shown
```

iter_next()

Iterate value changes between begin time and end time in a signal-based way.

Returns:

- [signal_id, time], if success.
- [signal_id, time], if fail.

Return type: [int,int]

Examples:

```
>>> As SigBased.py shown
```

iter_stop()

Stop iteration.

Examples:

```
>>> As SigBased.py shown
```

get_value(*format=<VctFormat_e.BinStrVal: 0>*)

Get the value of current value changes. (The spec. of format setting is the same as VctHandle.value()).

Parameters: format – VctFormat_e

Returns:

- The current value with the specified format, if success.
- None, if fail.

Examples:

```
>>> As SigBased.py shown
```

set_max_session_load(*num*)

Set the max session load number to control memory. (by default, the number is 0 and the iterator loads all the value changes in the specified time range in one time).

Parameters: num – Maximum loaded session number.

Examples:

```
>>> As SigBased.py shown
```

Feedback

L1 APIs

- Function list
  - Hierarchy Tree

waveform.hier_tree_dump_scope(file, outFileName, rootScope=None)	Dump the waveform scope tree to a file.
waveform.hier_tree_dump_sig(file, outputFileName, rootScope=None, expand=0)	Dump the waveform signal list to a file.

  - Time Conversion

waveform.time_scale_unit(file)	Get the time scale unit of the waveform file.
waveform.convert_time_in(file, timeValue, timeUnit)	Convert input time value according to the waveform file.
waveform.convert_time_out(file, timeValue, timeUnit)	Convert input time value according to the waveform file.

  - Sig Values

waveform.sig_value_at(file, sigName, time, format=<VctFormat_e.BinStrVal: 0>)	Get the value of a signal at a specific time.
waveform.sig_hdl_value_at(sig, time, format=<VctFormat_e.BinStrVal: 0>)	Get the value of a signal at a specific time.
waveform.sig_vec_value_at(file, sigNameList, time, format=<VctFormat_e.BinStrVal: 0>)	Get the values of a signal vector at a specific time.
waveform.sig_hdl_vec_value_at(sigHdlList, time, format=<VctFormat_e.BinStrVal: 0>)	Get the values of a signal vector at a specific time.

waveform.sig_value_between(file, sigName, beginTime, endTime, format=<VctFormat_e.BinStrVal: 0>)	Get the values of a signal within a specific time range.
waveform.sig_hdl_value_between(sig, beginTime, endTime, format=<VctFormat_e.BinStrVal: 0>)	Get the values of a signal within a specific time range.
waveform.dump_sig_value_between(file, sigName, beginTime, endTime, outputFileName, format=<VctFormat_e.BinStrVal: 0>)	Dump the values of a signal within a specific time range into a file.
waveform.dump_sig_hdl_value_between(sig, beginTime, endTime, outputFileName, format=<VctFormat_e.BinStrVal: 0>)	Dump the values of a signal within a specific time range into a file.

- Find Values

waveform.sig_find_x_forward(file, sigName, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Forward find the signal’s value which contains any x.
waveform.sig_hdl_find_x_forward(sig, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Forward find the signal’s value which contains any x.
waveform.sig_find_x_backward(file, sigName, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Backward find the signal’s value which contains any x.
waveform.sig_hdl_find_x_backward(sig, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Backward find the signal’s value which contains any x.

Feedback

waveform.sig_find_value_forward(file, sigName, value, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Forward find the signal's value which is exactly the same as the input string.
waveform.sig_hdl_find_value_forward(sig, value, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Forward find the signal's value which is exactly the same as the input string.
waveform.sig_find_value_backward(file, sigName, value, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Backward find the signal's value which is exactly the same as the input string.
waveform.sig_hdl_find_value_backward(sig, value, beginTime, format=<VctFormat_e.BinStrVal: 0>)	Backward find the signal's value which is exactly the same as the input string.
waveform.sig_vc_count(file, sigName, beginTime, endTime)	Count value changes of a signal within a specific time range.

Hierarchy Tree

Example:

example.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
file = waveform.open("CPU.fsdb")
if not file:
   print("Error. Failed to open file")
waveform.hier_tree_dump_scope(file, "hier_scope.log", 'tb_CPUsystem')
waveform.hier_tree_dump_sig(file, "hier_sig.log",
 'tb_CPUsystem.i_CPUsystem.i_pram', 0)
waveform.close(file)
npisys.end()
```

Result:

```
In hier_scope.log
tb_CPUsystem
i_CPUsystem
i_CPU
i_PCU
i_ALUB
i_alu
i_CCU
i_maprom
i_mprom
i_pram
i_BJsource
i_BJkernel
```

Result:

```
In hier_sig.log
tb_CPUsystem.i_CPUsystem.i_pram.clock
tb_CPUsystem.i_CPUsystem.i_pram.VMA
tb_CPUsystem.i_CPUsystem.i_pram.R_W
tb_CPUsystem.i_CPUsystem.i_pram.addr
tb_CPUsystem.i_CPUsystem.i_pram.data
tb_CPUsystem.i_CPUsystem.i_pram.BUSY
tb_CPUsystem.i_CPUsystem.i_pram.dataout
tb_CPUsystem.i_CPUsystem.i_pram.Reading
tb_CPUsystem.i_CPUsystem.i_pram.Writing
tb_CPUsystem.i_CPUsystem.i_pram.macroram
```

waveform.hier_tree_dump_scope(*file, outFileName, rootScope=None*)

Dump the waveform scope tree to a file.

Parameters:

- **file** – Target file object FileHandle.
- **outFileName** – Specify the name of output file.
- **rootScope** – Specify the name of the target scope.

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> waveform.hier_tree_dump_scope(fileHandle, "hier_scope.log",
 'tb_CPUsystem')
(In hier_scope.log)
tb_CPUsystem
i_CPUsystem
i_CPU
i_PCU
i_ALUB
i_alu
i_CCU
```

waveform.hier_tree_dump_sig(*file, outputFileName, rootScope=None, expand=0*)

Dump the waveform signal list to a file.

Parameters:

- **file** – Target file object FileHandle.
- **outFileName** – Specify the name of output file.
- **rootScope** – Specify the name of the target scope.
- **expand** – Specify if including member signals (expand = 0, dump declared signal only; expand = 1, dump declared signals and their member signals).

Returns:

- True, if success.
- False, if fail.

Return type: bool

Examples:

```
>>> waveform.hier_tree_dump_sig(fileHandle, "hier_sig.log",
 'tb_CPUsystem', 0)
tb_CPUsystem.NextCard
tb_CPUsystem.OK
tb_CPUsystem.Fail
```

Time Conversion

Example:

example.py

Feedback

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
file = waveform.open("CPU.fsdb")
if not file:
  print("Error. Failed to open file")
time_unit = waveform.time_scale_unit(file)
  print(time_unit)
time_in = waveform.convert_time_in(file, 1000, "ps")
  print("waveform.convert_time_in: "+str(time_in)+" x "+time_unit+" is
 "+"1000 ps")
time_out = waveform.convert_time_out(file, 1, "ps")
  print("waveform.convert_time_out: "+"1 ns"+" is "+str(time_out)+" ns")
waveform.close(file)
npisys.end()
```

Result:

```
1ns
waveform.convert_time_in: 1 x 1ns is 1000 ps
waveform.convert_time_out: 1 ns is 1000.0 ns
```

waveform.time_scale_unit(*file*)

Get the time scale unit of the waveform file.

Parameters: **file** – Target file object FileHandle.

Returns:

- Time unit, if success.
- None ,if fail.

Return type: str

Examples

```
>>> time_unit = waveform.time_scale_unit(file)
print(time_unit)
1ns
```

waveform.convert_time_in(*file, timeValue, timeUnit*)

Convert input time value according to the waveform file.

Feedback

Parameters:

- **file** – Target file object FileHandle.
- **timeValue** – Specify the input time value.
- **timeUnit** – Specify the input time unit (supported time unit: s, ms, us, ns, ps, fs (case insensitive)).

Returns:

- time_in, if success.
- None, if fail.

Return type: int

Examples:

```
>>> time_in = waveform.convert_time_in(fileHandle, 1000, "ps")
print("waveform.convert_time_in: "+str(time_in)+" x "+time_unit+" is
 "+"1000 ps")
waveform.convert_time_in: 1 x 1ns is 1000 ps
```

waveform.convert_time_out(*file, timeValue, timeUnit*)

Convert input time value according to the waveform file.

Parameters:

- **file** – Target file object FileHandle.
- **timeValue** – Specify the input time value.
- **timeUnit** – Specify the input time unit (supported time unit: s, ms, us, ns, ps, fs (case insensitive)).

Returns:

- time_out, if success.
- None, if fail.

Return type: int

Examples

```
>>> time_out = waveform.convert_time_out(fileHandle, 1, "ps")
print("waveform.convert_time_out: "+"1 ns"+" is "+str(time_out)+" ps")
waveform.convert_time_out: 1 ns is 1000.0 ps
```

Feedback

Sig Values

Example:

example.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(cmd)
file = waveform.open("CPU.fsdb")
if not file:
    print("Error. Failed to open file")
    print("(sig_value_at)")
res = waveform.sig_value_at(file, 'tb_CPUsystem.CLOCK1', 200,
 waveform.VctFormat_e.BinStrVal)
    print(res)
res = waveform.sig_value_at(file, 'tb_CPUsystem.CLOCK1', 250,
 waveform.VctFormat_e.BinStrVal)
print(res)
    print("(sig_hdl_value_at)")
sigHdl = file.sig_by_name('tb_CPUsystem.Card', None)
if not sigHdl:
    print("Error. Failed to get signal")
res = waveform.sig_hdl_value_at(sigHdl, 400,
 waveform.VctFormat_e.DecStrVal)
print(res)
res = waveform.sig_hdl_value_at(sigHdl, 800,
 waveform.VctFormat_e.DecStrVal)
print(res)
    print("(sig_vec_value_at)")
sigNameList = ['tb_CPUsystem.CLOCK1', 'tb_CPUsystem.NewGame']
valList = waveform.sig_vec_value_at(file, sigNameList, 1325,
 waveform.VctFormat_e.BinStrVal)
print(valList)
    print("(sig_hdl_vec_value_at)")
sigHdlList = [file.sig_by_name('tb_CPUsystem.CLOCK1'),
 file.sig_by_name('tb_CPUsystem.NewGame')]
valList = waveform.sig_hdl_vec_value_at(sigHdlList, 1325,
 waveform.VctFormat_e.BinStrVal)
print(valList)
    print("(sig_value_between)")
ret = waveform.sig_value_between(file, 'tb_CPUsystem.CLOCK1', 10, 200,
 waveform.VctFormat_e.DecStrVal)
print(ret)
print("(sig_hdl_value_between)")
ret =
 waveform.sig_hdl_value_between(file.sig_by_name('tb_CPUsystem.CLOCK1'),
 10, 200, waveform.VctFormat_e.DecStrVal)
```

Feedback

```
print(ret)
print("(dump_sig_value_between)")
ret = waveform.dump_sig_value_between(file, 'tb_CPUsystem.CLOCK1', 10,
 200, "dump_sig_value_between.log", waveform.VctFormat_e.DecStrVal)
print(ret)
print("(dump_sig_hdl_value_between)")
ret =
 waveform.dump_sig_hdl_value_between(file.sig_by_name('tb_CPUsystem.CLOCK
1'), 10, 200, "dump_sig_hdl_value_between.log",
 waveform.VctFormat_e.DecStrVal)
print(ret)
waveform.close(file)
npisys.end()
```

Result:

```
(sig_value_at)
0
1
(sig_hdl_value_at)
4
11
(sig_vec_value_at)
['0', '1']
(sig_hdl_vec_value_at)
None
(sig_value_between)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
(sig_hdl_value_between)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
(dump_sig_value_between)
True
(dump_sig_hdl_value_between)
True
```

Result:

```
In dump_sig_value_between.log
10: 0
50: 1
100: 0
150: 1
200: 0
```

Result:

```
In dump_sig_hdl_value_between.log
10: 0
50: 1
100: 0
150: 1
200: 0
```

Feedback

waveform.sig_value_at(*file, sigName, time, format=<VctFormat_e.BinStrVal: 0>*)

Get the value of a signal at a specific time.

Parameters:

- **file** – Specify the waveform file objecti FileHandle.
- **sigName** – Specify the signal's full hierarchy name.
- **time** – Specify the waveform time value.
- **format** – VctFormat_e.

Returns:

- Signal value in specific time with specified format if success.

Return type: str

Examples:

```
>>> res = waveform.sig_value_at(fileHandle, 'tb_CPUsystem.CLOCK1', 200,
 waveform.VctFormat_e.BinStrVal)
print(res)
0
```

waveform.sig_hdl_value_at(*sig, time, format=<VctFormat_e.BinStrVal: 0>*)

Get the value of a signal at a specific time.

Parameters:

- **sig** – Specify the target signal SigHandle.
- **time** – Specify the waveform time value.
- **format** – VctFormat_e.

Returns:

- Signal value in specific time.

Return type: str

Examples:

```
>>> res = waveform.sig_hdl_value_at(sigHdl, 400,
 waveform.VctFormat_e.DecStrVal)
print(res)
4
```

Feedback

waveform.sig_vec_value_at(*file, sigNameList, time, format=<VctFormat_e.BinStrVal: 0>*)

Get the values of a signal vector at a specific time.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigNameList** – Specify the signal vector with full hierarchy names.
- **time** – Specify the waveform time value.
- **format** – VctFormat.

Returns:

- List of signal value.

Return type: List

Examples:

```
>>> valList = waveform.sig_vec_value_at(fileHandle, sigNameList, 1325,
 waveform.VctFormat_e.BinStrVal)
print(valList)
['0', '1']
```

waveform.sig_hdl_vec_value_at(sigHdlList, time, format=<VctFormat_e.BinStrVal: 0>)

Get the values of a signal vector at a specific time.

Parameters:

- **sigHdlList** – Specify the signal object vector SigHandle.
- **time** – Specify the waveform time value.
- **format** – VctFormat_e.

Returns: List of signal value.

Return type: List

Examples:

```
>>> valList = waveform.sig_hdl_vec_value_at(sigHdlList, 1325,
 waveform.VctFormat_e.BinStrVal)
print(valList)
['0', '1']
```

Feedback

waveform.sig_value_between(*file, sigName, beginTime, endTime, format=<VctFormat_e.BinStrVal: 0>*)

Get the values of a signal within a specific time range.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.
- **beginTime** – Specify the begin time of target range.
- **endTime** – Specify the end time of target range.
- **format** – VctFormat_e.

Returns: List of signal value.

Return type: List

Examples:

```
>>> ret = waveform.sig_value_between(fileHandle, 'tb_CPUsystem.CLOCK1',
 10, 200, waveform.VctFormat_e.DecStrVal)
print(ret)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
```

waveform.sig_hdl_value_between*(sig, beginTime, endTime, format=<VctFormat_e.BinStrVal: 0>)*

Get the values of a signal within a specific time range.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **beginTime** – Specify the begin time of target range.
- **endTime** – Specify the end time of target range.
- **format** – VctFormat_e.

Returns: List of signal value.

Return type: List

Examples:

```
>>> ret =
 waveform.sig_hdl_value_between(fileHandle.sig_by_name('tb_CPUsystem.CLOC
K1'), 10, 200, waveform.VctFormat_e.DecStrVal)
```

Feedback

```
print(ret)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
```

waveform.dump_sig_value_between(*file, sigName, beginTime, endTime, outputFileName, format=<VctFormat_e.BinStrVal: 0>*)

Dump the values of a signal within a specific time range into a file.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.
- **beginTime** – Specify the begin time of target range.
- **endTime** – Specify the end time of target range.
- **outputFileName** – The file name of output file.
- **format** – VctFormat_e.

Returns:

- True if success.
- False if fail.

Return type: bool

Examples:

```
>>> ret = waveform.dump_sig_value_between(fileHandle,
 'tb_CPUsystem.CLOCK1', 10, 200, "dump_sig_value_between.log",
 waveform.VctFormat_e.DecStrVal)
print(ret)
True
(In dump_sig_value_between.log)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
```

waveform.dump_sig_hdl_value_between(*sig, beginTime, endTime, outputFileName, format=<VctFormat_e.BinStrVal: 0>*)

Dump the values of a signal within a specific time range into a file.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **beginTime** – Specify the begin time of target range.
- **endTime** – Specify the end time of target range.

- **outputFileName** – The file name of output file.
- **format** – VctFormat_e.

Returns:

- True if success.
- False if fail.

Return type: bool

Examples:

```
>>> ret =
 waveform.dump_sig_hdl_value_between(fileHandle.sig_by_name('tb_CPUsystem
.CLOCK1'), 10, 200, "dump_sig_hdl_value_between.log",
 waveform.VctFormat_e.DecStrVal)
print(ret)
True
(In dump_sig_hdl_value_between.log)
[(10, '0'), (50, '1'), (100, '0'), (150, '1'), (200, '0')]
```

Find Values

Example:

example.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
file = waveform.open(".fsdb")
if not file:
  print("Error. Failed to open file")
xfile = waveform.open("x_value.fsdb")
if not xfile:
  print("Error. Failed to open file")
xSigName = "scope1.arraySig[0]"
xSig = xfile.sig_by_name(xSigName)
vcTuple = waveform.sig_find_x_forward(xfile, xSigName, 60,
 waveform.VctFormat_e.DecStrVal)
print('waveform.sig_find_x_forward', vcTuple)
vcTuple = waveform.sig_hdl_find_x_forward(xSig, 60,
 waveform.VctFormat_e.DecStrVal)
print('waveform.sig_hdl_find_x_forward', vcTuple)
vcTuple = waveform.sig_find_x_backward(xfile, xSigName, 60,
 waveform.VctFormat_e.DecStrVal)
```

Feedback

```
print('waveform.sig_find_x_backward', vcTuple)
vcTuple = waveform.sig_hdl_find_x_backward(xSig, 60,
 waveform.VctFormat_e.DecStrVal)
print('waveform.sig_hdl_find_x_forward', vcTuple)
time = waveform.sig_find_value_forward(file, 'tb_CPUsystem.CLOCK1', "0",
 1000, waveform.VctFormat_e.DecStrVal)
print('waveform.sig_find_value_forward', time)
time = waveform.sig_find_value_forward(file, 'tb_CPUsystem.CLOCK1', "0",
 1000, waveform.VctFormat_e.DecStrVal)
print(time)
SigName = "tb_CPUsystem.CLOCK1"
sigHdl = file.sig_by_name(SigName)
time = waveform.sig_hdl_find_value_forward(sigHdl, "1", 1000,
 waveform.VctFormat_e.DecStrVal)
print('sig_hdl_find_value_forward', time)
time = waveform.sig_find_value_backward(file, 'tb_CPUsystem.CLOCK1', "0",
 1000, waveform.VctFormat_e.DecStrVal)
print('sig_find_value_backward', time)
time = waveform.sig_hdl_find_value_backward(sigHdl, "1", 1000,
 waveform.VctFormat_e.DecStrVal)
print('sig_hdl_find_value_backward', time)
vcCount = waveform.sig_vc_count(file, 'tb_CPUsystem.CLOCK1', 0, 100)
print('sig_vc_count: ' + str(vcCount))
waveform.close(file)
waveform.close(xfile)
npisys.end()
```

Result:

```
waveform.sig_find_x_forward (75, 'X')
waveform.sig_hdl_find_x_forward (75, 'X')
waveform.sig_find_x_backward (35, 'X')
waveform.sig_hdl_find_x_forward (35, 'X')
waveform.sig_find_value_forward 1100
1100
sig_hdl_find_value_forward 1050
sig_find_value_backward 900
sig_hdl_find_value_backward 950
sig_vc_count: 3
```

waveform.sig_find_x_forward(file, sigName, beginTime, format=<VctFormat_e.BinStrVal: 0>)

Forward find the signal's value which contains any x.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.

Feedback

- **beginTime** – Specify the begin time for search.
- **format** – VctFormat_e.

Returns:

- (time, value)

Return type*:* tuple

Examples:

```
>>> vcTuple = waveform.sig_find_x_forward(xfile, xSigName, 60,
 waveform.VctFormat_e.DecStrVal)
print(vcTuple)
(75, 'X')
```

waveform.sig_hdl_find_x_forward(*sig, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Forward find the signal's value which contains any x.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **beginTime** – Specify the begin time for search.
- **format** – VctFormat_e.

Returns:

- (time, value)

Return type: tuple

Examples:

```
>>> vcTuple = waveform.sig_hdl_find_x_forward(xSig, 60,
 waveform.VctFormat_e.DecStrVal)
print(vcTuple)
(75, 'X')
```

waveform.sig_find_x_backward(*file, sigName, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Backward find the signal's value which contains any x.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.

Feedback

- **beginTime** – Specify the begin time for search.
- **format** – VctFormat_e.

Returns:

- (time, value)

Return type: tuple

Examples:

```
>>> vcTuple = waveform.sig_find_x_backward(xfile, xSigName, 60,
 waveform.VctFormat_e.DecStrVal)
print(vcTuple)
(75, 'X')
```

waveform.sig_hdl_find_x_backward(*sig, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Backward find the signal's value which contains any x.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **beginTime** – Specify the begin time for search.
- **format** – VctFormat_e.

Returns:

- (time, value)

Return type: tuple

Examples:

```
>>> vcTuple = waveform.sig_hdl_find_x_backward(xSig, 60,
 waveform.VctFormat_e.DecStrVal)
print(vcTuple)
(75, 'X')
```

waveform.sig_find_value_forward(*file, sigName, value, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Forward find the signal's value which is exactly the same as the input string.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.

- **value** – Specify the search value in string format.
- **beginTime** – Specify the begin time for search.
- **format** – class:.VctFormat_e.

Returns:

- Time if success.
- None if fail.

Return type: int

Examples:

```
>>> time = waveform.sig_find_value_forward(file, 'tb_CPUsystem.CLOCK1',
 "0", 1000, waveform.VctFormat_e.DecStrVal)
print(time)
1100
```

waveform.sig_hdl_find_value_forward(*sig, value, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Forward find the signal's value which is exactly the same as the input string.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **value** – Specify the search value in string format.
- **beginTime** – Specify the begin time for search.
- **format** – class:.VctFormat_e.

Returns:

- Time if success.
- None if fail.

Return type: int

Examples:

```
>>> time = waveform.sig_find_value_forward(file, 'tb_CPUsystem.CLOCK1',
 "0", 1000, waveform.VctFormat_e.DecStrVal)
print(time)
1100
```

Feedback

waveform.sig_find_value_backward(*file, sigName, value, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Backward find the signal's value which is exactly the same as the input string.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.
- **value** – Specify the search value in string format.
- **beginTime** – Specify the begin time for search.
- **format** – class:.VctFormat_e.

Returns:

- Time if success.
- None if fail.

Return type: int

Examples:

```
>>> time = waveform.sig_find_value_backward(file, 'tb_CPUsystem.CLOCK1',
 "0", 1000, waveform.VctFormat_e.DecStrVal)
print(time)
900
```

waveform.sig_hdl_find_value_backward(*sig, value, beginTime, format=<VctFormat_e.BinStrVal: 0>*)

Backward find the signal's value which is exactly the same as the input string.

Parameters:

- **sig** – Specify the signal object SigHandle.
- **value** – Specify the search value in string format.
- **beginTime** – Specify the begin time for search.
- **format** – class:.VctFormat_e.

Returns:

- Time, if success.
- None, if fail.

Feedback

Return type: int

Examples:

```
>>> time = waveform.sig_hdl_find_value_backward(sigHdl, "1", 1000,
 waveform.VctFormat_e.DecStrVal)
print(time)
950
```

waveform.sig_vc_count(*file, sigName, beginTime, endTime*)

Count value changes of a signal within a specific time range.

Parameters:

- **file** – Specify the waveform file object FileHandle.
- **sigName** – Specify the signal with full hierarchy name.
- **beginTime** – Specify the begin time.
- **endTime** – Specify the end time.

Returns:

- The value change count, if success.
- -1, if fail.

Return type: int

Examples:

```
>>> vcCount = waveform.sig_vc_count(file, 'tb_CPUsystem.CLOCK1', 0, 100)
print(vcCount)
3
```