

---

# **connectal Documentation**

***Release 14.11.6***

**Jamey Hicks, Myron King, John Ankcorn**

December 11, 2014



## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Connectal BSV Libraries</b>	<b>5</b>
2.1	HostInterface Package . . . . .	5
2.2	Leds Package . . . . .	5
2.3	CtrlMux Package . . . . .	5
2.4	Portal Package . . . . .	6
<b>3</b>	<b>Connectal Examples</b>	<b>7</b>
3.1	Simple Example . . . . .	7
<b>4</b>	<b>Python</b>	<b>9</b>
4.1	AST . . . . .	9
4.2	bsvgen . . . . .	11
4.3	cppgen . . . . .	11
4.4	discover_tcp . . . . .	12
4.5	makefilegen . . . . .	13
4.6	util . . . . .	13
<b>5</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Bsv Package Index</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



Contents:



## INTRODUCTION

Introduction goes here.





## CONNECTAL BSV LIBRARIES

### 2.1 HostInterface Package

The HostInterface package provides host-specific typedefs and interfaces.

**HostInterface::DataBusWidth**

Width in bits of the data bus connected to host shared memory.

**HostInterface::PhysAddrWidth**

Width in bits of physical addresses on the data bus connected to host shared memory.

**HostInterface::NumberOfMasters**

Number of memory interfaces used for connecting to host shared memory.

**interface HostInterface::BsimHost**

Host interface for the bluesim platform

**interface HostInterface::PcieHost**

Host interface for PCIe-attached FPGAs such as vc707 and kc705

**interface HostInterface::ZynqHost**

Host interface for Zynq FPGAs such as zedboard, zc702, zc706, and zybo.

The Zc706 is a ZynqHost even when it is plugged into a PCIe slot.

### 2.2 Leds Package

**interface Leds::LEDS**

**Leds::LedsWidth**

Defined to be the number of default LEDs on the FPGA board.

The Zedboard has 8, Zc706 has 4, ...

### 2.3 CtrlMux Package

**CtrlMux::mkInterruptMux**

Used by BsimTop, PcieTop, and ZynqTop

**CtrlMux::mkSlaveMux** → PhysMemSlave#(addrWidth,dataWidth)

Takes a vector of MemPortals and returns a PhysMemSlave combining them.

## 2.4 Portal Package

### 2.4.1 PipePortal Interface

```
interface Portal :: PipePortal
:parameter: numeric type numRequests, numeric type numIndications, numeric type slaveDataWidth

    messageSize (Bit#(16) methodNumber) → Bit#(16)
        Returns the message size of the methodNumber method of the portal.

    requests → Vector#(numRequests, PipeIn#(Bit#(slaveDataWidth)))

    indications → Vector#(numIndications, PipeOut#(Bit#(slaveDataWidth)))
```

### 2.4.2 MemPortal Interface

```
interface Portal :: MemPortal

    slave → PhysMemSlave#(slaveAddrWidth,slaveDataWidth)

    interrupt → ReadOnly#(Bool)

    top → WriteOnly#(Bool)

Portal :: getSlave (MemPortal#(_a,_d) p) → PhysMemSlave(_a,_d)
Portal :: getInterrupt (MemPortal#(_a,_d) p) → ReadOnly#(Bool)
Portal :: getInterruptVector (Vector#(numPortals, MemPortal#(_a,_d)) portals) → Vector#(16,
    ReadOnly#(Bool))
```

### 2.4.3 ShareMemoryPortal Interface

```
interface Portal :: SharedMemoryPortal
    Should be in SharedMemoryPortal.bsv

    readClient;

    writeClient → MemWriteClient#(dataBusWidth)

    interface cfg → SharedMemoryPortalConfig

    Portal :: interrupt → ReadOnly#(Bool)
```

## CONNECTAL EXAMPLES

### 3.1 Simple Example



AST
bsvgen
cppgen
discover_tcp
makefilegen
util

## 4.1 AST

### Functions

<code>classInfo(item)</code>	
<code>declInfo(mitem)</code>	
<code>dtInfo(arg)</code>	
<code>piInfo(pitem)</code>	
<code>serialize_json(interfaces, globalimports, ...)</code>	Returns json serialized data

### Classes

<code>Enum(elements)</code>	Represents a BSV Enum
<code>EnumElement(name, qualifiers, value)</code>	
<code>Function(name, return_type, params)</code>	
<code>Interface(name, params, decls, ...)</code>	Represents a BSV Interface
<code>InterfaceMixin</code>	
<code>Method(name, return_type, params)</code>	
<code>Module(moduleContext, name, params, ...)</code>	represents a BSV Module
<code>Param(name, t)</code>	
<code>Struct(elements)</code>	Represents a BSV Struct
<code>StructMember(t, name)</code>	
<code>Type(name, params)</code>	Represents a BSV Type
<code>TypeDef(tdtype, name, params)</code>	
<code>Typeclass(name)</code>	
<code>TypeclassInstance(name, params, provisos, decl)</code>	
<code>Variable(name, t)</code>	

```
class AST.Enum (elements)
    Represents a BSV Enum

    instantiate (paramBindings)

class AST.EnumElement (name, qualifiers, value)

class AST.Function (name, return_type, params)

class AST.Interface (name, params, decls, subinterfacename, packagename)
    Bases: AST.InterfaceMixin

    Represents a BSV Interface

    Create an Interface with name, params, decls, subintfacename, packagename

    instantiate (paramBindings)

    interfaceType ()

class AST.InterfaceMixin

    getSubinterface (name)

    parentClass (default)

class AST.Method (name, return_type, params)

    instantiate (paramBindings)

class AST.Module (moduleContext, name, params, interface, provisos, decls)
    represents a BSV Module

    instantiates a BSV Module

class AST.Param (name, t)

    instantiate (paramBindings)

class AST.Struct (elements)
    Represents a BSV Struct

    instantiate (paramBindings)

class AST.StructMember (t, name)

    instantiate (paramBindings)

class AST.Type (name, params)
    Represents a BSV Type

    instantiate (paramBindings)

class AST.TypeDef (tdtype, name, params)

class AST.Typeclass (name)

class AST.TypeclassInstance (name, params, provisos, decl)

class AST.Variable (name, t)

AST.classInfo (item)

AST.declInfo (mitem)
```

AST.**dtInfo** (*arg*)

AST.**piInfo** (*pitem*)

AST.**serialize\_json** (*interfaces, globalimports, dutname, interfaceList*)  
Returns json serialized data

## 4.2 bsvgen

### Functions

---

`collectElements`(*mlist, workerfn, name*)

---



---

`fixupSubsts`(*item, suffix*)

---



---

`generate_bsv`(*project\_dir, noisyFlag, jsongdata*)

---



---

`toBsvType`(*titem*)

---

`bsvgen.collectElements` (*mlist, workerfn, name*)

`bsvgen.fixupSubsts` (*item, suffix*)

`bsvgen.generate_bsv` (*project\_dir, noisyFlag, jsongdata*)

`bsvgen.toBsvType` (*titem*)

## 4.3 cppgen

### Functions

---

`accumWords`(*s, pro, memberList*)

---



---

`cName`(*x*)

---



---

`collectMembers`(*scope, pitem*)

---



---

`emitCD`(*item, generated\_hpp, indentation*)

---



---

`emitEnum`(*item, name, f, indentation*)

---



---

`emitMethodDeclaration`(*mname, params, f, ...*)

---



---

`emitStruct`(*item, name, f, indentation*)

---



---

`emitStructMember`(*item, f, indentation*)

---



---

`formalParameters`(*params, insertPortal*)

---



---

`gatherMethodInfo`(*mname, params, itemname*)

---



---

`generate_class`(*className, declList, parentC, ...*)

---



---

`generate_cpp`(*project\_dir, noisyFlag, jsongdata*)

---



---

`generate_demarshall`(*fmt, w*)

---



---

`generate_marshall`(*pfmt, w*)

---



---

`getNumeric`(*item*)

---



---

`hasBitWidth`(*item*)

---



---

`indent`(*f, indentation*)

---



---

`typeBitWidth`(*item*)

---



---

`typeCName`(*item*)

---



---

`typeNumeric`(*item*)

---

## Classes

---

`paramInfo(name, width, shifted, datatype, ...)`

---

```
cppgen.accumWords(s, pro, memberList)
cppgen.cName(x)
cppgen.collectMembers(scope, pitem)
cppgen.emitCD(item, generated_hpp, indentation)
cppgen.emitEnum(item, name, f, indentation)
cppgen.emitMethodDeclaration(mname, params, f, className)
cppgen.emitStruct(item, name, f, indentation)
cppgen.emitStructMember(item, f, indentation)
cppgen.formalParameters(params, insertPortal)
cppgen.gatherMethodInfo(mname, params, itemname)
cppgen.generate_class(className, declList, parentC, parentCC, generatedCFiles, create_cpp_file,
                       generated_hpp, generated_cpp)
cppgen.generate_cpp(project_dir, noisyFlag, jsondata)
cppgen.generate_demarshall(fmt, w)
cppgen.generate_marshall(pfmt, w)
cppgen.getNumeric(item)
cppgen.hasBitWidth(item)
cppgen.indent(f, indentation)
class cppgen.paramInfo(name, width, shifted, datatype, assignOp)
cppgen.typeBitWidth(item)
cppgen.typeCName(item)
cppgen.typeNumeric(item)
```

## 4.4 discover\_tcp

### Functions

---

`connect_with_adb(ipaddr)`

---

`detect_network()`

---

`do_work(start, end)`

---

`int2ip(addr)`

---

`ip2int(addr)`

---

`open_adb_socket(dest_addr)`

---

```
discover_tcp.connect_with_adb(ipaddr)
```



```
discover_tcp.detect_network()  
discover_tcp.do_work(start, end)  
discover_tcp.int2ip(addr)  
discover_tcp.ip2int(addr)  
discover_tcp.open_adb_socket(dest_addr)
```

## 4.5 makefilegen

## 4.6 util

### Functions

<code>capitalize(s)</code>
<code>createDirAndOpen(f, m)</code>
<code>decapitalize(s)</code>
<code>foldl(f, x, l)</code>
<code>intersperse(e, l)</code>
<code>splitBinding(s)</code>

```
util.capitalize(s)  
util.createDirAndOpen(f, m)  
util.decapitalize(s)  
util.foldl(f, x, l)  
util.intersperse(e, l)  
util.splitBinding(s)
```



## INDICES AND TABLES

- *genindex*
- *modindex*
- *pkgindex*
- *search*



**c**

CtrlMux, 5

**h**

HostInterface, 5

**l**

Leds, 5

**p**

Portal, 6



**a**

AST, [9](#)

**b**

bsvgen, [11](#)

**c**

cppgen, [11](#)

**d**

discover\_tcp, [12](#)

**m**

makefilegen, [13](#)

**u**

util, [13](#)





**A**

accumWords() (in module cppgen), 12  
 AST (module), 9

**B**

BsimHost (interface in HostInterface), 5  
 bsvgen (module), 11

**C**

capitalize() (in module util), 13  
 classInfo() (in module AST), 10  
 cName() (in module cppgen), 12  
 collectElements() (in module bsvgen), 11  
 collectMembers() (in module cppgen), 12  
 connect\_with\_adb() (in module discover\_tcp), 12  
 cppgen (module), 11  
 createDirAndOpen() (in module util), 13  
 CtrlMux (package), 5

**D**

DataBusWidth (in package HostInterface), 5  
 decapitalize() (in module util), 13  
 declInfo() (in module AST), 10  
 detect\_network() (in module discover\_tcp), 12  
 discover\_tcp (module), 12  
 do\_work() (in module discover\_tcp), 13  
 dtInfo() (in module AST), 10

**E**

emitCD() (in module cppgen), 12  
 emitEnum() (in module cppgen), 12  
 emitMethodDeclaration() (in module cppgen), 12  
 emitStruct() (in module cppgen), 12  
 emitStructMember() (in module cppgen), 12  
 Enum (class in AST), 10  
 EnumElement (class in AST), 10

**F**

fixupSubsts() (in module bsvgen), 11  
 foldl() (in module util), 13  
 formalParameters() (in module cppgen), 12  
 Function (class in AST), 10

**G**

gatherMethodInfo() (in module cppgen), 12  
 generate\_bsv() (in module bsvgen), 11  
 generate\_class() (in module cppgen), 12  
 generate\_cpp() (in module cppgen), 12  
 generate\_demarshal() (in module cppgen), 12  
 generate\_marshall() (in module cppgen), 12  
 getInterrupt() (in package Portal), 6  
 getInterruptVector() (in package Portal), 6  
 getNumeric() (in module cppgen), 12  
 getSlave() (in package Portal), 6  
 getSubinterface() (AST.InterfaceMixin method), 10

**H**

hasBitWidth() (in module cppgen), 12  
 HostInterface (package), 5

**I**

indent() (in module cppgen), 12  
 instantiate() (AST.Enum method), 10  
 instantiate() (AST.Interface method), 10  
 instantiate() (AST.Method method), 10  
 instantiate() (AST.Param method), 10  
 instantiate() (AST.Struct method), 10  
 instantiate() (AST.StructMember method), 10  
 instantiate() (AST.Type method), 10  
 int2ip() (in module discover\_tcp), 13  
 Interface (class in AST), 10  
 InterfaceMixin (class in AST), 10  
 interfaceType() (AST.Interface method), 10  
 intersperse() (in module util), 13  
 ip2int() (in module discover\_tcp), 13

**L**

LEDS (interface in Leds), 5  
 Leds (package), 5  
 LedsWidth (in package Leds), 5

**M**

makefilegen (module), 13  
 MemPortal (interface in Portal), 6  
 messageSize() (Portal::PipePortal method), 6

Method (class in AST), [10](#)  
mkInterruptMux (in package CtrlMux), [5](#)  
mkSlaveMux (in package CtrlMux), [5](#)  
Module (class in AST), [10](#)

## N

NumberOfMasters (in package HostInterface), [5](#)

## O

open\_adb\_socket() (in module discover\_tcp), [13](#)

## P

Param (class in AST), [10](#)  
paramInfo (class in cppgen), [12](#)  
parentClass() (AST.InterfaceMixin method), [10](#)  
PcieHost (interface in HostInterface), [5](#)  
PhysAddrWidth (in package HostInterface), [5](#)  
piInfo() (in module AST), [11](#)  
PipePortal (interface in Portal), [6](#)  
Portal (package), [6](#)

## S

serialize\_json() (in module AST), [11](#)  
SharedMemoryPortal (interface in Portal), [6](#)  
SharedMemoryPortal.cfg (interface in Portal), [6](#)  
splitBinding() (in module util), [13](#)  
Struct (class in AST), [10](#)  
StructMember (class in AST), [10](#)

## T

toBsvType() (in module bsvgen), [11](#)  
Type (class in AST), [10](#)  
typeBitWidth() (in module cppgen), [12](#)  
Typeclass (class in AST), [10](#)  
TypeclassInstance (class in AST), [10](#)  
typeCName() (in module cppgen), [12](#)  
TypeDef (class in AST), [10](#)  
typeNumeric() (in module cppgen), [12](#)

## U

util (module), [13](#)

## V

Variable (class in AST), [10](#)

## Z

ZynqHost (interface in HostInterface), [5](#)