



OpenCPI - Open Component Portability Infrastructure

**An Open Source Infrastructure for
Heterogeneous/Embedded Component-
Based Systems and Applications**

OpenCPI Debugging Tools

- Use **ocpihdl** and **ocpixml** to collect HDL application worker and HDL device worker information
- Review the worker lifecycle
- Use **ocpihdl** to control the worker state
- Change and view worker properties
- Use the **ocpi_debug** built-in worker build configuration property

- You can use `ocpihdl` to:
 - Probe the XML of the currently loaded bitstream
 - Display currently loaded workers
 - View/change worker status
 - Set/change worker properties
 - Reset workers
 - Step through an application
- When invoked without options, `ocpihdl` lists all of its uses
- See the `ocpihdl(1)` man page for command usage details
- You cannot currently use `ocpihdl` on simulators (this function may be enabled in a future release)

- **ocpihdl search** – provides information on available HDL devices, such as:
 - Device name
 - Date of bitstream creation
 - HDL platform
 - Part identifier
 - Unique identifier (UUID) of the bitstream

```
ocpihdl search
```

```
OpenCPI HDL device found: 'PL:0': bitstream date Fri Jan  7 09:35:37  
2022, platform "matchstiq_z1", part "xc7z020", UUID 10226754-2e6e-  
11e7-88f7-573bc8e8124c
```

- **ocpihdl probe** – similar to search, but you can specify an HDL device

- Examine the XML packaged into *any* artifact (including bitstreams) with **ocpixml get**:

```
ocpixml get <artifact-file>.bitz |  
<artifact-file>.so
```

(See the **ocpixml(1)** man page for tool usage details)

- Load a bitstream with **ocpihdl load**:

```
ocpihdl load <path-to-bitstream>
```

- Examine the XML packaged into a *currently loaded* bitstream with **ocpihdl getxml**:

```
ocpihdl getxml <output-file>
```

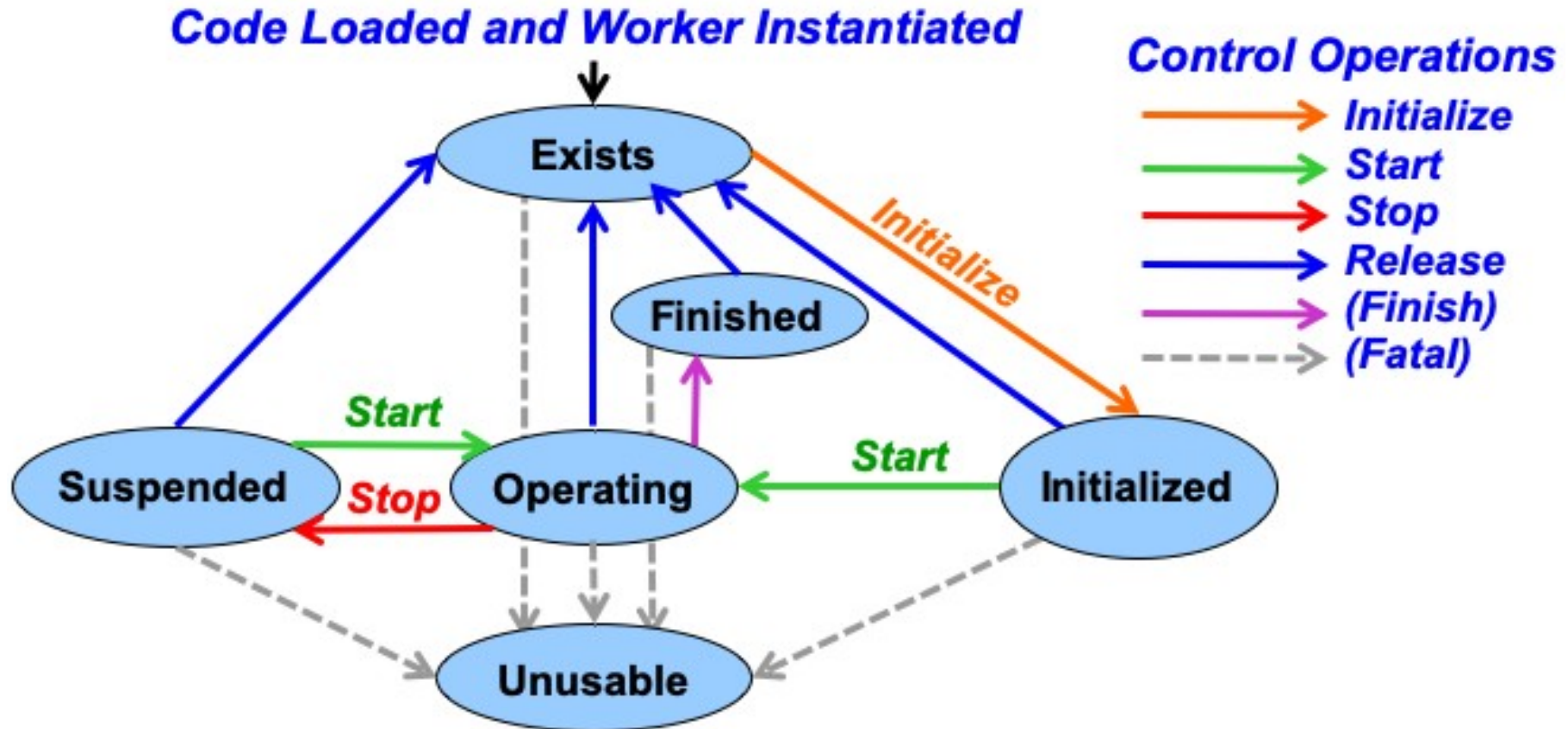
```
ocpihdl getxml <output-file>
```

```
grep controlOperation <output-file>
```

- What controlOperations are implemented by each worker?

```
% grep controlOperation output.xml | tail -4
<worker name="pattern" model="hdl" package="ocpi" specname="ocpi.pattern" sizeOfConfigSpace="21
47483712" controlOperations="initialize">
<worker name="bias" model="hdl" package="ocpi" specname="ocpi.bias" sizeOfConfigSpace="42949672
96" controlOperations="initialize" Timeout="444">
<worker name="capture" model="hdl" package="ocpi" specname="ocpi.capture" sizeOfConfigSpace="21
47487744" controlOperations="initialize">
<worker name="ocscp" model="hdl" package="ocpi" specname="ocpi.ocscp" controlOperations="stop">
```

- If you want to remove the bitstream from the FPGA, use **ocpihdl unload**
- Most commonly used when the system's processor functions independently from the programmable logic
- **CAUTION:** Unloading an FPGA that implements a PCIe interface may result in the card becoming unusable and may require a system reboot



(Fatal): fatal error from control operation, other transitions based on success:
non-fatal errors do not change states.
(Finish): is self initiated, not controlled from outside

- Find the worker index and other information:

ocpihdl get

You can also specify **<worker-instance-index>**, **<worker-instance-name>**, and even **<worker-property>** as arguments

- Reset or unreset the control reset signal into workers:

ocpihdl [wreset|wunreset] <worker-instance-index>

- Perform other control operations on workers:

ocpihdl wop <worker-instance-index> [start|initialize|stop]

- Set worker properties:

**ocpihdl set [<worker-instance-index>|<worker-instance-name>] \
<property> <value>**

- Observe worker state:

ocpihdl status [<worker-instance-index>|<worker-instance-name>]

Add the **-v** option to see worker properties

ocpihdl get

```
% ocpihdl get
HDL Device: 'PL:0' is platform 'matchstiq_z1' part 'xc7z020' and UUID '10226754-2e6e-1
1e7-88f7-573bc8e8124c'
Platform configuration workers are:
  Instance p/matchstiq_z1 of io worker matchstiq_z1 (spec ocpi.platform) with index 0
  Instance p/time_server of io worker time_server (spec ocpi.devices.time_server) with
index 1
Container workers are:
  Instance c/ocscp of normal worker ocscp (spec ocpi.ocscp)
  Instance c/unoc_term0_0 of io worker sdp_term (spec ocpi.devices.sdp_term)
  Instance c/unoc_term1_0 of io worker sdp_term (spec ocpi.devices.sdp_term)
  Instance c/unoc_term2_0 of io worker sdp_term (spec ocpi.devices.sdp_term)
  Instance c/unoc_term3_0 of io worker sdp_term (spec ocpi.devices.sdp_term)
  Instance c/tb_bias_wti0_time_client of normal worker time_client (spec ocpi.time_cli
ent)
  Instance c/metadata of normal worker metadata (spec ocpi.metadata)
Application workers are:
  Instance a/pattern of normal worker pattern (spec ocpi.pattern) with index 2
  Instance a/bias of normal worker bias (spec ocpi.bias) with index 3
  Instance a/capture of normal worker capture (spec ocpi.capture) with index 4
```

We can identify the worker instances by their indices

If the worker instance status is “RESET”:
ocpihdl unreset <worker-instance-index>

```
% ocpihdl status 2
```

```
Status of instance 'a/pattern' of worker 'pattern' is 'RESET'
```

```
Worker 2 on device pl:0
```

```
Status:      0x00008000
```

```
Control:      0x00000000 not enabled (reset asserted); timeout value is 1
```

```
ConfigAddr: 0x00000000
```

```
PageWindow: 0x00000000
```

```
Instance a/pattern of normal worker pattern (spec ocpi.pattern) with index 2
```

```
% ocpihdl wunreset 2
```

```
Worker 2 on device pl:0: reset deasserted, was asserted
```

```
% ocpihdl status 2
```

```
Status of instance 'a/pattern' of worker 'pattern' is 'EXISTS'
```

```
Worker 2 on device pl:0
```

```
Status:      0x00008000
```

```
Control:      0x80000000 enabled (reset not asserted); timeout value is 1
```

```
ConfigAddr: 0x00000000
```

```
PageWindow: 0x00000000
```

```
Instance a/pattern of normal worker pattern (spec ocpi.pattern) with index 2
```

If the worker instance implements the **initialize** control option:

ocpihdl wop <worker-instance-index> initialize

```
% ocpihdl wop 2 initialize
Worker 2 on device pl:0: the 'initialize' control operation was performed with result 'success'
01)
% ocpihdl status 2
Status of instance 'a/pattern' of worker 'pattern' is 'INITIALIZED'
Worker 2 on device pl:0
Status:      0x00048000 opValid:0x0:init
Control:     0x80000000 enabled (reset not asserted); timeout value is 1
ConfigAddr: 0x00000000
PageWindow: 0x00000000
Instance a/pattern of normal worker pattern (spec ocp.pattern) with index 2
```

Ultimately, to bring the worker instance status to “OPERATING” :

ocpihdl wop <worker-instance-index> start

- Writeable worker properties can be changed:

```
ocpihdl set [<worker-instance-index>|<worker-instance-name>] \  
<property> <value>
```

- Volatile properties can be read:

```
ocpihdl status -v [<worker-instance-index>|<worker-instance-name>]
```

```
% ocpihdl set 2 step true
```

```
Setting the step property to 'true' on instance 'a/counter'
```

```
% ocpihdl status 2 ; ocpihdl get -v 2
```

```
Status of instance 'a/counter' of worker 'counter-1' is 'OPERATING'
```

```
Worker 2 on device pl:0
```

```
Status:      0x091f0000 addrValid beValid:0x1 opValid:0x1:start wrtValid:1
```

```
Control:     0x80000004 enabled (reset not asserted); timeout value is 16
```

```
ConfigAddr: 0x00000008
```

```
PageWindow: 0x00000000
```

```
Instance a/counter of normal worker counter-1 (spec local.counter) with index 2
```

0	counter: 2
1	max: 9
2	ocpi_debug: true
3	ocpi_endian: little
4	step: true

Using the `ocpi_debug` “Parameter” Property



- `ocpi_debug` is a framework-generated (built-in) worker build configuration “parameter” property for specifying a “debug” build vs. a “production” build
- `ocpi_debug` is set to **true** in the OpenCPI Worker Description (OWD) before building a worker for debugging
- You add the debugging functionality that depends on `ocpi_debug` to the worker
- In VHDL: `ocpi_debug`
`debug_gen : if its(ocpi_debug) generate`
- In C++: `<WORKER-NAME>_OCPI_DEBUG`
`if (COUNTER_OCPI_DEBUG) {`

- Log level:
 export OCPI_LOG_LEVEL=8
 ocpirun [-loglevel=8|-l 8] (l is a lowercase "L")
- **ocpirun** **[-l list|-C]** - to find containers, including simulators
- **ocpiview** - to view simulation waveform results
- **printf/cout** - to display diagnostic output
- **ocpi_debug** - for debug builds
- **gdb** - GNU Project Debugger