encoreio::Module Class Reference

List of all members.

Public Member Functions

- def **get_instance**
- def **__init__**
- def **wait_irq**
- def **wait**
- def **wait_or_pending**
- def **read**
- def **write**
- def **timeout**
- def **qsize**
- def **pprint**

Detailed Description

```
Module class is the software abstraction of an HW Module.
```

Member Function Documentation

def encoreio::Module::__init__ (self, drvname, devname, lun)

```
Module's constructor
```

def encoreio::Module::get_instance (cls, drvname, lun)

```
Class's method returning a Module instance
Parameters
----------
drvname: str
    name of the driver driving the HW module
lun: int
    logical unit identifying the right HW module.

Returns
```

```
-------
out: Module
    Module instance

Examples
--------
# get module instances corresponding to the HW module DAB lun 0 and 1
mod1 = Module.get_instance('bi_bctfdab', 0)
mod2 = Module.get_instance('bi_bctfdab', 1)
```

def encoreio::Module::pprint (self)

```
print Module's register definition
Can be use in case of doubt regarding the register definition
```

def encoreio::Module::qsize (self,
qsize
)

```
set irq queue size

Parameters
----------
qsize: int
    desired size for the irq queue.

Returns:
res: int
    returns 0 in case of success, a negative value in case of
    failure
```

def encoreio::Module::read (self,
regname,
am = -1,
start = 0,
to = -1,
dma = 0,
time = 0
)

```
Read the content of the given register

Parameters
----------
regname: str
    name of desired register
am: int (optional)
    the desired address modifier. By default VME access are
    using the address modifier specified in the Module
    description
start: int (optional)
```

```
    the desired start index in case one wants to read a slice of
    the register content. By default start is 0
to: int (optional)
    the desired end index in case one wants to read a slice of
    the register content. By default end is equal to the
    register size
dma: int (1 or 0)
    enable/disable a DMA transaction. By default dma = 0 (DMA
    is disabled)
time: int (1 or 0)
    generates a trace on the stdout giving the time spent to
    complete the read. By default time = 0 (trace is disabled)

Returns
-------
value: tuple
    the returned value is a tuple even for scalar register
    In case of error, None object is returned and an error
    message is sent to stdout.

Examples
--------
# read register capBufA of the dab module 0 using dma and
# checking the time spent to read 512KB
val = dab0.read("capBufA", dma=1, time=1)
if val is not None:
    # read succeeds print the value
    print 'val:',val[0]
```

def encoreio::Module::timeout (self, timeout)

```
set the timeout to the desired value

Parameters
----------
timeout: int
    desired value in ms.

Returns:
res: int
    returns 0 in case of success, a negative value in case of
    failure
```

def encoreio::Module::wait (self)

```
Deprecated: should be replaced by wait_irq()
waits till an interrupt is raised or the timeout expired
Note: by default the timeout is not set, therefore teh wait
migth block for ever if the module doesn't raised any interrupt.
Check timeout() method to set a timeout
```

```
Returns
-------
three parameters:
irq_val: int
    content of the irq source register
irq_count: int
    the irq counter
res: int
    the error code. It should be 0 if the wait terminates on
    interrupt reception or negative value if the timeout expired. In
    case of failure irq_val and irq_count are set to 0

Examples
--------
# wait interrupt raised by the module dab0
irq_val, irq_count, res = dab0.wait()
```

def encoreio::Module::wait_irq (self)

```
waits till an interrupt is raised or the timeout expired
Note: by default the timeout is not set, therefore the wait
migth block for ever if the module doesn't raised any interrupt.
Check timeout() method to set a timeout

Returns
-------
2 parameters:
irq:
    irq data returned using this structure.
logical_unit: device lun having raised the irq
irq_value: content of the irq source register
irq_count: the irq counter
pending: how many irq remain in the queue
queued: tells if the irq was queued or not
nsdelay: nanosecond delay between the irq arrival and its
consumption by the client

res: int
    the error code. It should be 0 if the wait terminates on
    interrupt reception or negative value if the timeout expired.

Examples
--------
# wait interrupt raised by the module dab0
irq, res = dab0.wait_irq()
if not res:
     print ("lun: %d, irq_src: %d, irq_count: %d, queued: %d, "
    "pending: %d, nsdelay: %d"
    %(irq.logical_unit, irq.irq_value, irq.irq_count, irq.queued,
    irq.pending, irq.nsdelay) )
```

def encoreio::Module::wait_or_pending (self)

```
Deprecated: should be replaced by wait_irq()
```

```
Behaves like the wait function but before waiting for a new
interrupt, this function checks if an interrupt is still
pending, that is to say an interrupt which occured while no
client was listening. The queue size of pending interrupt is 1
```

```
def encoreio::Module::write (  self,
                               regname,
                               data,
                               am = -1,
                               start = 0,
                               to = -1,
                               dma = 0,
                               time = 0
                             )
```

```
write the desired value into the given register

Parameters
----------
regname: str
    name of desired register
data: tuple
    value to be written into the register
    data is a tuple even for a scalar value
am: int (optional)
    the desired address modifier. By default VME access are
    using the address modifier specified in the Module
    description
start: int (optional)
    the desired start index in case one wants to read a slice of
    the register content. By default start is 0
to: int (optional)
    the desired end index in case one wants to read a slice of
    the register content. By default end is equal to the
    register size
dma: int (1 or 0)
    enable/disable a DMA transaction. By default dma = 0 (DMA
    is disabled)
time: int (1 or 0)
    generates a trace on the stdout giving the time spent to
    complete the read. By default time = 0 (trace is disabled)

Returns
-------
res: int
    returns 0 in case of success, a negative value in case of
    failure

Examples
--------
# Set register thresholdA of the dab module 0
# The scalar value is passed via a tuple
```

```
res = dab0.write("capBufA", (0x20,))
if res:
    # write failed
    ...
```

The documentation for this class was generated from the following file:

- encoreio.py

All Classes Functions
