

driver.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/11

INFORMATION

Brief

Bus Driver for APB3

License MIT

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

apb3Master

apb3Base

apb3Master

Drive slave devices over the APB3 bus

FUNCTIONS

init

```
def __init__(
    self,
    entity,
    name,
    clock,
    resetn,

    args,
    kwargs
)
```

Setup defaults and call base class constructor.

read

```
async def read(
    self,
    address
)
```

Read from a address and return data

write

```
async def write(
    self,
    address,
    data
)
```

Write to a address some data

_check_type

```
def _check_type(
    self,
    trans
)
```

Check and make sure we are only sending 2 bytes at a time and that it is a bytes/bytearray

_run

```
async def _run(
    self
)
```

_run thread that deals with read and write queues.

apb3EchoSlave

apb3Base

apb3EchoSlave

Respond to master reads and write by returning data, simple echo core.

FUNCTIONS

__init__

```
def __init__(
    self,
    entity,
    name,
    clock,
    resetn,
    numreg
    =
    256,
    args,
    kwargs
)
```

Setup defaults and call base class constructor.

_check_type

```
def _check_type(
    self,
    trans
)
```

Check and make sure we are only sending a type of apb3trans.

_run

```
async def _run(
    self
)
```

_run thread that deals with read and write request over bus.