absbus.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/26

INFORMATION

Brief

abstraction of the Analog Devices uP bus

License MIT

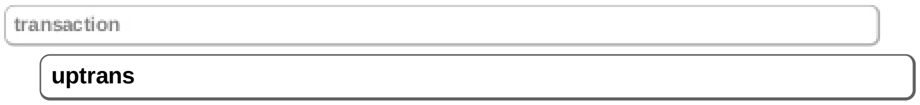
Copyright 2025 Jay Convertino

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.

uptrans



create an object that associates a data member and address for operation.

upState

enum.IntEnum

upState

An enum class that provides the current state and will change states per spec.

upBase

busbase

upBase

upEchoSlave

upMaster

upMonitor

abstract base class that defines uP signals

VARIABLES

_signals

```
_signals
```

List of signals that are required

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
                                                                    *
args,
                                                                   **
kwargs
)
```

Setup defaults and call base class constructor.

_restart_rw

```
def _restart_rw(
self
)
```

kill and restart _run thread.

_run

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

_run thread that deals with read and write.

_run_read

```
async def _run_read(
self
)
```

Abstract method for read thread

_run_write

```
async def _run_write(
self
)
```

Abstract method for write thread