

mil_std_1553.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/06

INFORMATION

Brief

MIL-STD-1553 cocotb

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.

Copyright (c) 2020 Alex Forencich

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.

MILSTD1553Source

MILSTD1553Source

A mil-std-1553 transmit test routine.

FUNCTIONS

__init__

```
def __init__(
    self,
    data,
    args,
    kwargs
)
```

Initialize the object

VARIABLES

self._data

self._data

Set internal data connection to 1553 differential bus

self._base_delay

self._base_delay

1 MHz is 1000 nano seconds need half that due to manchester encoding method

self._idle

self._idle

Event trigger for cocotb

self._data

Event trigger for cocotb

self._run_cr

```
self._run_cr
```

Thread instance of `_run` method

FUNCTIONS

`_restart`

```
def _restart(  
    self  
)
```

kill and restart `_run` thread.

`write_cmd`

```
async def write_cmd(  
    self,  
    data  
)
```

Write data to send that uses the command sync

`write_data`

```
async def write_data(  
    self,  
    data  
)
```

Write data to send that uses the data sync

`write_nowait_cmd`

```
def write_nowait_cmd(  
    self,  
    data  
)
```

Write data to send that uses command sync but do not wait after writing.

`write_nowait_data`

```
def write_nowait_data(  
    self,  
    data  
)
```

Write data to send that uses data sync but do not wait after writing.

count

```
def count(  
    self  
)
```

How many items in the queue

empty

```
def empty(  
    self  
)
```

Is the queue empty?

idle

```
def idle(  
    self  
)
```

Is the queue empty and the `_run` is not active processing data.

clear

```
def clear(  
    self  
)
```

Remove all items from queue

_check_type

```
def _check_type(  
    self,  
    data  
)
```

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

_cmd_sync

```
async def _cmd_sync(  
    self,  
    data  
)
```

Generate a command sync on the diff output

_data_sync

```
async def _data_sync(  
    self,  
    data  
)
```

Generate a data sync on the diff output

wait

```
async def wait(  
    self  
)
```

Wait for the run thread to become idle.

_run

```
async def _run(  
    self,  
    data  
)
```

Thread that processing queue and outputs data in mil-std-1553 format.

MILSTD1553Sink

MILSTD1553Sink

A mil-std-1553 transmit test routine.

FUNCTIONS

init

```
def __init__(  
    self,  
    data,  
  
    args,  
  
    kwargs  
)
```

Initialize the object

VARIABLES

self._data

```
self._data
```

Set internal data connection to 1553 differential bus

self._base_delay

```
self._base_delay
```

1 MHz is 1000 nano seconds need half that due to manchester decoding method

self._base_delay

```
self._base_delay_half
```

1 MHz is 1000 nano seconds need half of half that due to manchester decoding method

_cmd_sync

```
self._cmd_sync
```

command sync array value

_data_sync

```
self._data_sync
```

data sync array value

self._run_cr

```
self._run_cr
```

Thread instance of _run method

FUNCTIONS

_restart

```
def _restart(  
    self  
)
```

Kill and restart run function

read_cmd

```
async def read_cmd(  
    self  
)
```

Read any data that was identified with a command sync

read_nowait_cmd

```
def read_nowait_cmd(  
    self  
)
```

Read any data that was identified with a command sync, and do not wait for data to become available.

read_data

```
async def read_data(  
    self  
)
```

Read any data that was identified with a data sync.

read_nowait_data

```
def read_nowait_data(  
    self  
)
```

Read any data that was identified with a data sync, and do not wait for data to become available.

count_cmd

```
def count_cmd(  
    self  
)
```

How many elements are in the command queue?

count_data

```
def count_data(  
    self  
)
```

How many elements are in the data queue?

empty_cmd

```
def empty_cmd(  
    self  
)
```

Is the queue empty?

empty_data

```
def empty_data(  
    self  
)
```

Is the queue empty?

idle

```
def idle(  
    self  
)
```

Is _run waiting to process data?

clear_cmd

```
def clear_cmd(  
    self  
)
```

Clear the command queue

clear_data

```
def clear_data(  
    self  
)
```

Clear the data queue

wait_cmd

```
async def wait_cmd(  
    self,  
    timeout  
    =  
    0,  
    timeout_unit  
    =  
    'nsreg_data'  
)
```

Wait for command data

wait_data

```
async def wait_data(  
    self,  
    timeout  
    =  
    0,  
    timeout_unit  
    =  
    'nsreg_data'  
)
```

Wait for data data.

_run

```
async def _run(  
    self,  
    data  
)
```

Thread that takes input data in mil-std-1553 format and puts it in the proper command or data queue.