

Motor Abstraction Reference

Author

April 2022



TABLE OF CONTENTS

1	Abstract Motor Class	1
2	Motor Configuration Load	5
	Bibliography	9
	Python Module Index	10
	Index	11

ABSTRACT MOTOR CLASS

```
class AbstractMotor (motor_id, **kwargs)
```

```
    Bases: abc.ABC
```

Abstract Motor Class

Required attributes:

- `motor_id` Motor ID (hexadecimal or otherwise). This is used both for low level communication and to refer to individual motors in communication with the user.

Recommended attributes:

- `motor` Motor object containing methods for low-level motor control. It may be provided by the user or by a library. For further reference consider the T-Motor AK80-6 and AK80-9 interfaces provided by [mini-cheetah-tmotor-python-can](#) (pip).

```
__getattr__ (name)
```

Override attribute retrieval.

Conduct motor configuration validation at attribute retrieval time, and apply boilerplate to communicate the motor's state after all commands and provide the user with logs.

1. Motor configuration validation

- Ensure motor instance has all required attributes
- Ensure motor instance has a declared `rest_state` before calling any function other than `__init__` or `rest_state`

2. Function boilerplate

- Push motor state via provided communication protocol after all motor commands
- Log motor disabling

```
__getattr__ (name)
```

Retrieve unknown attributes from motor controller.

```
__command (fn_command, *args, **kwargs)
```

```
__disable (fn_disable, *args, **kwargs)
```

rest_state(*x_r*)

Set rest state

abstract __init__(*motor_id*, ***kwargs*)

Abstract Motor Class

Required attributes:

- *motor_id* Motor ID (hexadecimal or otherwise). This is used both for low level communication and to refer to individual motors in communication with the user.

Recommended attributes:

- *motor* Motor object containing methods for low-level motor control. It may be provided by the user or by a library. For further reference consider the T-Motor AK80-6 and AK80-9 interfaces provided by [mini-cheetah-tmotor-python-can](#) (pip).

abstract enable()

Enable motor

abstract zero()

Zero at current motor position

abstract rest()

Rest position command

abstract command(*u*)

Motor command

abstract disable()

Disable motor

```
__abstractmethods__ = frozenset({'__init__', 'command', 'disable',  
'enable', 'rest', 'zero'})
```

```
__dict__ = mappingproxy({'__module__':  
'motor_abstraction.abstract_motor', '__getattr__': <function  
AbstractMotor.__getattr__>, '__getattribute__': <function  
AbstractMotor.__getattribute__>, '_command': <function  
AbstractMotor._command>, '_disable': <function  
AbstractMotor._disable>, 'rest_state': <function  
AbstractMotor.rest_state>, '__init__': <function  
AbstractMotor.__init__>, 'enable': <function AbstractMotor.enable>,  
'zero': <function AbstractMotor.zero>, 'rest': <function  
AbstractMotor.rest>, 'command': <function AbstractMotor.command>,  
'disable': <function AbstractMotor.disable>, '__dict__': <attribute  
'__dict__' of 'AbstractMotor' objects>, '__weakref__': <attribute  
'__weakref__' of 'AbstractMotor' objects>, '__doc__': None,  
'__abstractmethods__': frozenset({'rest', '__init__', 'command',  
'zero', 'disable', 'enable'}), '_abc_impl': <_abc_data object>,  
'__annotations__': {}})
```

```
__module__ = 'motor_abstraction.abstract_motor'
```

```
__slots__ = ()
```

```
__weakref__
```

list of weak references to the object (if defined)

```
_abc_impl = <_abc_data object>
```

```
class Protocol (*args, **kwargs)
```

Bases: `abc.ABC`

Initialize communication protocol for individual device

```
__init__ (*args, **kwargs)
```

Initialize communication protocol for individual device

```
abstract generate_bindings (*args, **kwargs)
```

Generate bindings

```
abstract push (*args, **kwargs)
```

```
abstract pull (*args, **kwargs)
```

```
__abstractmethods__ = frozenset({'generate_bindings', 'pull', 'push'})
```

```
__dict__ = mappingproxy({'__module__': 'motor_abstraction.communicator', '__init__': <function Protocol.__init__>, 'generate_bindings': <function Protocol.generate_bindings>, 'push': <function Protocol.push>, 'pull': <function Protocol.pull>, '__dict__': <attribute '__dict__' of 'Protocol' objects>, '__weakref__': <attribute '__weakref__' of 'Protocol' objects>, '__doc__': None, '__abstractmethods__': frozenset({'push', 'pull', 'generate_bindings'})}, '_abc_impl': <_abc_data object>, '__annotations__': {}})
```

```
__module__ = 'motor_abstraction.communicator'
```

```
__slots__ = ()
```

```
__weakref__
```

list of weak references to the object (if defined)

```
_abc_impl = <_abc_data object>
```

```
class lcm (topic, freq, generate_bindings=False)
```

Bases: `motor_abstraction.communicator.Protocol`

Initialize LCM protocol for individual device

```
__init__(topic, freq, generate_bindings=False)  
    Initialize LCM protocol for individual device  
  
generate_bindings(*args, **kwargs)  
    Generate bindings  
  
push(content)  
  
pull(content)  
  
__abstractmethods__ = frozenset({})  
  
__annotations__ = {}  
  
__dict__ = mappingproxy({'__module__':  
'motor_abstraction.communicator', '__init__': <function  
lcm.__init__>, 'generate_bindings': <function lcm.generate_bindings>,  
'push': <function lcm.push>, 'pull': <function lcm.pull>, '__doc__':  
None, '__abstractmethods__': frozenset(), '_abc_impl': <_abc_data  
object>, '__annotations__': {}})  
  
__module__ = 'motor_abstraction.communicator'  
  
__slots__ = ()  
  
__weakref__  
    list of weak references to the object (if defined)  
  
_abc_impl = <_abc_data object>
```

MOTOR CONFIGURATION LOAD

load (*robot*)

mjbots (*robot*)

mjbots motor configuration

1. Create transport with all

exception **_AddendumException** (*msg, add="", lst=[], ind=' '*)

Bases: `Exception`

Exception with addendum for user guidance.

__init__ (*msg, add="", lst=[], ind=' '*)

__cause__

exception cause

__context__

exception context

__delattr__ (*name, /*)

Implement `delattr(self, name)`.

__dict__ = `mappingproxy({'__module__':
'motor_abstraction.utils.exceptions', '__doc__': '\n Exception with
addendum for user guidance.\n ', '__init__': <function
_AddendumException.__init__>, '__weakref__': <attribute '__weakref__'
of '_AddendumException' objects>, '__annotations__': {}})`

__getattr__ (*name, /*)

Return `getattr(self, name)`.

__module__ = `'motor_abstraction.utils.exceptions'`

__new__ (***kwargs*)

__reduce__ ()

Helper for pickle.

__repr__()

Return repr(self).

__setattr__(name, value, /)

Implement setattr(self, name, value).

__setstate__()

__str__()

Return str(self).

__suppress_context__

__traceback__

__weakref__

list of weak references to the object (if defined)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception ConfigurationError (msg, add="", lst=[], ind='')

Bases: *motor_abstraction.utils.exceptions._AddendumException*

Raised when configuration errors are detected.

__annotations__ = {}

__cause__

exception cause

__context__

exception context

__delattr__(name, /)

Implement delattr(self, name).

__dict__ = mappingproxy({'__module__':

'motor_abstraction.utils.exceptions', '__doc__': '\n Raised when configuration errors are detected.\n ', '__annotations__': {}})

__getattr__(name, /)

Return getattr(self, name).

__init__(msg, add="", lst=[], ind='')

__module__ = 'motor_abstraction.utils.exceptions'

__new__(kwargs)**

__reduce__()

Helper for pickle.

__repr__()

Return repr(self).

__setattr__(name, value, /)

Implement setattr(self, name, value).

__setstate__()

__str__()

Return str(self).

__suppress_context__

__traceback__

__weakref__

list of weak references to the object (if defined)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception SafetyException(msg, add="", lst=[], ind='')

Bases: `motor_abstraction.utils.exceptions._AddendumException`

Raised when operational safety is compromised.

__annotations__ = {}

__cause__

exception cause

__context__

exception context

__delattr__(name, /)

Implement delattr(self, name).

__dict__ = mappingproxy({'__module__':

'motor_abstraction.utils.exceptions', '__doc__': '\n Raised when operational safety is compromised.\n ', '__annotations__': {}})

__getattr__(name, /)

Return getattr(self, name).

__init__(msg, add="", lst=[], ind='')

__module__ = 'motor_abstraction.utils.exceptions'

__new__ (***kwargs*)

__reduce__ ()

Helper for pickle.

__repr__ ()

Return repr(self).

__setattr__ (*name, value, /*)

Implement setattr(self, name, value).

__setstate__ ()

__str__ ()

Return str(self).

__suppress_context__

__traceback__

__weakref__

list of weak references to the object (if defined)

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

fallback_disable (*goal*)

shout_error (*error*)

shout_disabled (*motor*)

BIBLIOGRAPHY

- [1] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). 2021. Downloaded on 28.07.2021 from <http://underactuated.mit.edu/>.

PYTHON MODULE INDEX

a

`motor_abstraction.abstract_motor`, 1

c

`motor_abstraction.communicator`, 3

`motor_abstraction.config`, 4

`motor_abstraction.config.load`, 5

`motor_abstraction.config.postprocessing`, 5

m

`motor_abstraction`, 1

u

`motor_abstraction.utils`, 5

`motor_abstraction.utils.exceptions`, 5

`motor_abstraction.utils.fallback`, 8

`motor_abstraction.utils.shout`, 8

INDEX

Non-alphabetical

- `__abstractmethods__` (*AbstractMotor attribute*), 2
- `__abstractmethods__` (*lcm attribute*), 4
- `__abstractmethods__` (*Protocol attribute*), 3
- `__annotations__` (*ConfigurationError attribute*), 6
- `__annotations__` (*lcm attribute*), 4
- `__annotations__` (*SafetyException attribute*), 7
- `__cause__` (*_AddendumException attribute*), 5
- `__cause__` (*ConfigurationError attribute*), 6
- `__cause__` (*SafetyException attribute*), 7
- `__context__` (*_AddendumException attribute*), 5
- `__context__` (*ConfigurationError attribute*), 6
- `__context__` (*SafetyException attribute*), 7
- `__delattr__` () (*_AddendumException method*), 5
- `__delattr__` () (*ConfigurationError method*), 6
- `__delattr__` () (*SafetyException method*), 7
- `__dict__` (*_AddendumException attribute*), 5
- `__dict__` (*AbstractMotor attribute*), 2
- `__dict__` (*ConfigurationError attribute*), 6
- `__dict__` (*lcm attribute*), 4
- `__dict__` (*Protocol attribute*), 3
- `__dict__` (*SafetyException attribute*), 7
- `__getattr__` () (*AbstractMotor method*), 1
- `__getattr__` () (*_AddendumException method*), 5
- `__getattr__` () (*AbstractMotor method*), 1
- `__getattr__` () (*ConfigurationError method*), 6
- `__getattr__` () (*SafetyException method*), 7
- `__init__` () (*_AddendumException method*), 5
- `__init__` () (*AbstractMotor method*), 2
- `__init__` () (*ConfigurationError method*), 6
- `__init__` () (*lcm method*), 3
- `__init__` () (*Protocol method*), 3
- `__init__` () (*SafetyException method*), 7
- `__module__` (*_AddendumException attribute*), 5
- `__module__` (*AbstractMotor attribute*), 2
- `__module__` (*ConfigurationError attribute*), 6
- `__module__` (*lcm attribute*), 4
- `__module__` (*Protocol attribute*), 3
- `__module__` (*SafetyException attribute*), 7
- `__new__` () (*_AddendumException method*), 5
- `__new__` () (*ConfigurationError method*), 6
- `__new__` () (*SafetyException method*), 7
- `__reduce__` () (*_AddendumException method*), 5
- `__reduce__` () (*ConfigurationError method*), 6
- `__reduce__` () (*SafetyException method*), 8
- `__repr__` () (*_AddendumException method*), 5
- `__repr__` () (*ConfigurationError method*), 7
- `__repr__` () (*SafetyException method*), 8
- `__setattr__` () (*_AddendumException method*), 6
- `__setattr__` () (*ConfigurationError method*), 7
- `__setattr__` () (*SafetyException method*), 8
- `__setstate__` () (*_AddendumException method*), 6
- `__setstate__` () (*ConfigurationError method*), 7
- `__setstate__` () (*SafetyException method*), 8
- `__slots__` (*AbstractMotor attribute*), 3
- `__slots__` (*lcm attribute*), 4
- `__slots__` (*Protocol attribute*), 3
- `__str__` () (*_AddendumException method*), 6
- `__str__` () (*ConfigurationError method*), 7
- `__str__` () (*SafetyException method*), 8
- `__suppress_context__` (*_AddendumException attribute*), 6
- `__suppress_context__` (*ConfigurationError attribute*), 7
- `__suppress_context__` (*SafetyException attribute*), 8
- `__traceback__` (*_AddendumException attribute*), 6
- `__traceback__` (*ConfigurationError attribute*), 7
- `__traceback__` (*SafetyException attribute*), 8



`__weakref__` (*AddendumException attribute*), 6
`__weakref__` (*AbstractMotor attribute*), 3
`__weakref__` (*ConfigurationError attribute*), 7
`__weakref__` (*lcm attribute*), 4
`__weakref__` (*Protocol attribute*), 3
`__weakref__` (*SafetyException attribute*), 8
`_abc_impl` (*AbstractMotor attribute*), 3
`_abc_impl` (*lcm attribute*), 4
`_abc_impl` (*Protocol attribute*), 3
`_AddendumException`, 5
`_command()` (*AbstractMotor method*), 1
`_disable()` (*AbstractMotor method*), 1

A

`AbstractMotor` (*class in motor_abstraction.ab-*
stract_motor), 1
`args` (*AddendumException attribute*), 6
`args` (*ConfigurationError attribute*), 7
`args` (*SafetyException attribute*), 8

C

`command()` (*AbstractMotor method*), 2
`ConfigurationError`, 6

D

`disable()` (*AbstractMotor method*), 2

E

`enable()` (*AbstractMotor method*), 2

F

`fallback_disable()` (*in module motor_abstrac-*
tion.utils.fallback), 8

G

`generate_bindings()` (*lcm method*), 4
`generate_bindings()` (*Protocol method*), 3

L

`lcm` (*class in motor_abstraction.communicator*), 3
`load()` (*in module motor_abstraction.config.load*), 5

M

`mjbots()` (*in module motor_abstraction.config.post-*
processing), 5
`module`
 motor_abstraction, 1

motor_abstraction.abstract_motor,
 1
motor_abstraction.communicator, 3
motor_abstraction.config, 4
motor_abstraction.config.load, 5
motor_abstraction.config.postpro-
cessing, 5
motor_abstraction.utils, 5
motor_abstraction.utils.excep-
tions, 5
motor_abstraction.utils.fallback,
 8
motor_abstraction.utils.shout, 8
motor_abstraction
 module, 1
motor_abstraction.abstract_motor
 module, 1
motor_abstraction.communicator
 module, 3
motor_abstraction.config
 module, 4
motor_abstraction.config.load
 module, 5
motor_abstraction.config.postpro-
cessing
 module, 5
motor_abstraction.utils
 module, 5
motor_abstraction.utils.exceptions
 module, 5
motor_abstraction.utils.fallback
 module, 8
motor_abstraction.utils.shout
 module, 8

P

`Protocol` (*class in motor_abstraction.communic-*
ator), 3
`pull()` (*lcm method*), 4
`pull()` (*Protocol method*), 3
`push()` (*lcm method*), 4
`push()` (*Protocol method*), 3

R

`rest()` (*AbstractMotor method*), 2
`rest_state()` (*AbstractMotor method*), 1

S

`SafetyException`, 7

`shout_disabled()` (in module `motor_abstraction.utils.shout`), 8

`shout_error()` (in module `motor_abstraction.utils.shout`), 8

W

`with_traceback()` (`_AddendumException` method), 6

`with_traceback()` (`ConfigurationError` method), 7

`with_traceback()` (`SafetyException` method), 8

Z

`zero()` (`AbstractMotor` method), 2