Motor Abstraction Reference

Author

April 2022

TABLE OF CONTENTS

CHAPTER

ONE

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).

__getattribute__(*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', '__getattribute__': <function AbstractMotor.__getattribute__>, '__getattr__': <function AbstractMotor.__getattr__>, '_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>`

CHAPTER TWO

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__': {}})`

`__getattribute__`(*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__': {}})`

`__getattribute__``(`*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__': {}})`

`__getattribute__`(*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

INDEX

Non-alphabetical