enuActor Documentation

Release

Author

CONTENTS

1	enu <i>A</i>	Actor package	•
	1.1	Subpackages	
	1.2	Submodules	1
	1.3	enuActor.DecoratorStateMachine module	
	1.4	enuActor.FSM module	
	1.5	enuActor.MyFSM module	1
	1.6	enuActor.QThread module	1
	1.7	enuActor.main module	
	1.8	Module contents	1
2	Indi	ndices and tables	
Рy	thon]	Module Index	1:
In	dex		1′

Contents:

CONTENTS 1

2 CONTENTS

CHAPTER

ONE

ENUACTOR PACKAGE

1.1 Subpackages

1.1.1 enuActor.Commands package

Submodules

enuActor.Commands.EnuCmd module

enuActor.Commands.RexmCmd module

enuActor.Commands.ShutterCmd module

Module contents

1.1.2 enuActor.Devices package

Submodules

enuActor.Devices.Device module

class enuActor.Devices.Device (actor=None, cfg_file='/home/tpegot/mhsls/devel/enuActor/python/enuActor/Device
Bases: enuActor.QThread.QThread

All device (Shutter, BIA,...) should inherit this class

Attributes:

- link: TTL, SERIAL or ETHERNET
- ser : serial object from serial module @todo: change into link object
- mode: operation or simulated
- cfg_file : path of the communication config file

available_link = ['TTL', 'SERIAL', 'ETHERNET']

load_cfg(device)

Load configuration file of the device.

Parameters device (str.) – name of the device ('SHUTTER', 'BIA', ...)

Returns dict config

Raises CfgFileErr'

```
printstateonchange(e)
          What to display when state change
              Parameters e – event
     send(input buff=None)
          Check communication
              Parameters input_buff (str.) – string to send to check com.
              Returns returns from com.
              Raises CommErr
     startFSM()
          Instantiate the MyFSM class (create the State Machine).
     start_communication(*args, **kwargs)
          Docstring for start_communication.
                  Need first to specify config file and device by calling load_cfq() or in the header of
          Note:
          start_communication()
              Parameters
                  • device (str.) – device name
                  • startCmd (str.) – starting command to check the communication
                  • **kwargs - remaining keywords are not treated
              Returns Communication object (example: serial.Serial object)
              Raises CfgFileErr
     start ethernet()
          @todo: Docstring for start_ethernet.
              Returns @todo
              Raises @todo
     start serial(input buff=None)
          Start a serial communication
              Parameters input_buff (str.) – Send at start to check communication
              Returns serial. Serial
     start ttl()
          @todo: Docstring for start_ttl.
              Returns @todo
              Raises @todo
enuActor.Devices.Device.transition(during_state, after_state=None)
     Decorator enabling the function to trigger state of the FSM.
```

Parameters

- during_state event at beginning of the function
- after_state event after the function is performed if specified

Returns function return

```
Raises DeviceErr
```

enuActor.Devices.Error module

```
exception enuActor.Devices.Error.CfgFileErr (reason, lvl=1)
Bases: enuActor.Devices.Error.RuleError

Docstring for CommErr.

exception enuActor.Devices.Error.CommErr (reason, lvl=1)
Bases: enuActor.Devices.Error.RuleError
```

CommErr are all the error related to the communication between PC and Device.

```
exception enuActor.Devices.Error.DeviceErr (device, reason, lvl=1)

Bases: enuActor.Devices.Error.RuleError
```

DeviceErr are all the error related to the device and controller. When a DeviceErr occures the current state of the FSM go to fail.

```
exception enuActor.Devices.Error.RuleError (reason, lvl=1)
Bases: exceptions.Exception
Define rule and how it is displaied
```

PRIORITY_DEFAULT = 1

enuActor.Devices.rexm module

File: Author: Thomas Pegot-Ogier Email: thomas.pegot@lam.fr Github: gitolite@pfs.ipmu.jp:enuActor Description:

```
class enuActor.Devices.rexm.Rexm(actor=None)
     Bases: enuActor.Devices.Device.Device
     SW Device: Red EXchange Mechanism
     check_status()
     getStatus()
          return status of shutter (FSM)
              Returns 'LOADED', 'IDLE', 'BUSY', ...
     handleTimeout()
          Override method QThread.handleTimeout(). Process while device is idling.
              Returns @todo
              Raises CommErr
     initialise()
          Initialise REXM :returns: @todo :raises: @todo
     move (coord)
          Position defined in Cartesian coordinates with Bryant angles * move to (translate) X, Y, Z * Make a
          clockwise rotation around Z-axis * Make a clockwise rotation around Y-axis * Make a clockwise rotation
```

Parameters coord - @todo

Returns @todo

around X-axis

1.1. Subpackages 5

```
Raises @todo
     start communication (cmd=None)
              Parameters cmd – variable from tron
              Returns 0 if OK
              Raises CommErr
enuActor.Devices.shutter module
class enuActor.Devices.shutter.Shutter(actor=None)
     Bases: enuActor.Devices.Device.Device
     SW device: Shutter
     Attributes:
            • currPos : current position of the shutter
     MASK\_ERROR\_SB\_1 = [0, 0, 1, 1]
     MASK\_ERROR\_SB\_3 = [1, 1, 1, 1, 1, 1, 1]
     MASK\_ERROR\_SB\_4 = [0, 0, 1, 1]
     MASK\_ERROR\_SB\_5 = [1, 1, 1, 1, 1, 1, 1]
     MASK\_ERROR\_SB\_6 = [0, 0, 1, 1]
     STATUS_BYTE_1 = ['S_blade_A_offline', 'S_blade_B_offline', 'S_CAN_comm_error', 'S_error_interlock']
     STATUS BYTE 3 = ['S motor to origin timeout', 'S threshold error', ', 'S limit switch', 'S unknown command', 'S
     STATUS_BYTE_4 = ['S_blade_open', 'S_blade_closed', 'S_error_LED', 'S_error_interlock']
     STATUS_BYTE_5 = ['S_motor_to_origin_timeout', 'S_threshold_error', ', 'S_limit_switch', 'S_unknown_command', 'S
     STATUS_BYTE_6 = ['S_blade_open', 'S_blade_closed', 'S_error_LED', 'S_error_interlock']
     check status()
          Check status byte 1, 3, 4, 5 and 6 from Shutter controller and return current list of status byte.
              Returns [sb1, sb3, sb5, sb6] with sbi list of byte from status byte
              Raises CommErr
     getStatus()
          return status of shutter (FSM)
              Returns 'LOADED', 'IDLE', 'BUSY', ...
     handleTimeout()
          Override method OThread. handle Timeout (). Process while device is idling.
              Returns @todo
              Raises CommErr
          Initialise shutter. Here just trigger the FSM to INITIALISING and IDLE :returns: @todo :raises: @todo
     parseStatusByte(sb)
          Send status byte command and parse reply of device
              Parameters sb – byte 1, 3, 4, 5 or 6
```

```
Returns array_like defining status flag
Raises CommErr

positions = ['undef.', 'open', 'closed(A)', 'closed(B)']
shutter (*args)
shutter_id = ['red', 'blue', 'all']
start_communication (cmd=None)
@todo: Docstring for start_communication.

Parameters cmd - variable from tron
Returns 0 if OK
Raises CommErr

terminal ()
launch terminal connection to shutter device
```

Module contents

:RivCreateContent * Contents:

Returns @todo

- 1 Convention naming
- 2 The State Machine
- 3 The Devices
 - 3.1 Shutter
 - 3.2 BIA
 - 3.3 REXM
 - 3.4 IISOURCE
 - 3.5 ENU
 - 3.6 FPSA

Convention naming

The aim of this interface is to follow this naming convention at large:

```
enu <device> <command> [arguments [= value]]
```

Also others convention are defined like those for motorized devices:

- enu <motorized-device> SetHome = [value|CURRENT]: Set Home position to value or current position
- enu <motorized-device> GetGome: Get Home position
- enu <motorized-device> GoHome: Go to Home

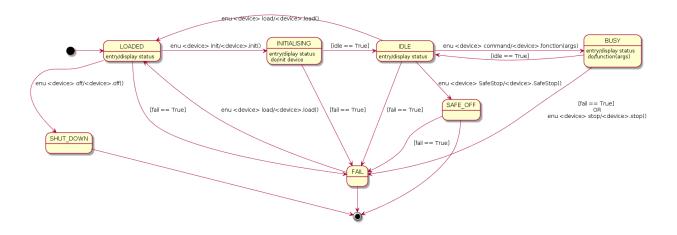
Here are devices classified:

NON	MOTORIZED	MOTORIZED			
BIA	IISOURCE	Environment	Shutter	REXM	FPS
todo	todo	todo	todo	todo	todo

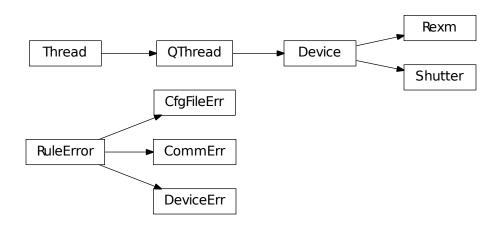
1.1. Subpackages 7

Note: Shutter is a motorized device but the SW device won't provide motorized features.

The State Machine

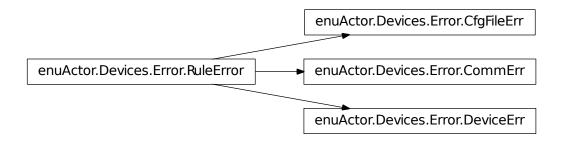


The Devices





Device



Error



Shutter

Shutter is open or close ...

BIA



REXM

IISOURCE

ENU

FPSA

1.1. Subpackages 9

1.2 Submodules

1.3 enuActor.DecoratorStateMachine module

class enuActor.DecoratorStateMachine.ContextBase

Bases: object

class enuActor.DecoratorStateMachine.TransitionTable (stateVarblName)

Bases: object

Defines a state table for a state machine class

A state table for a class is associated with the state variable in the instances of the class. The name of the state variable is given in the constructor to the StateTable object. StateTable objects are attributes of state machine classes, not intances of the state machine class. A state machine class can have more than one StateTable.

initialize(ctxt)

Create a new state variable in the context. State variable refs this transition table.

nextStates (subState, nslList)

Sets up transitions from the state specified by substate

subState is one of the derived state classes, subclassed from the context state machine class. nslList is a list of states to which the context will transition upon the invocation of one of the transition methods. 'None' may be specified instead of an actual state if the context is to remain in the same state upon invocation of the corresponding method.

enuActor.DecoratorStateMachine.event(state_table)

Decorator for indicating an Event or 'Action' method.

The decorator is applied to the methods of the state machine class to indicate that the method will invoke a state dependant behavior. States are implemented as subclasses of the context(state machine) class .

enuActor.DecoratorStateMachine.transition(state_table)

Decorator used to set up methods which cause transitions between states.

The decorator is applied to methods of the context (state machine) class. Invoking the method may cause a transition to another state. To define what the transitions are, the nextStates method of the TransitionTable class is used.

enuActor.DecoratorStateMachine.transitionevent(state_table)

A decorator which is essentially the combination of the above two.

Can both invoke state dependent method and trigger a state transition. Mostly equivalent to : @Transition(xitionTable) @Event(xitionTable)

enuActor.DecoratorStateMachine.truncated(alist, cmprsn)

1.4 enuActor.FSM module

1.5 enuActor.MyFSM module

 ${f class}$ enuActor.MyFSM.Fysom (cfg)

Bases: object

Wraps the complete finite state machine operations.

can (event)

Returns if the given event be fired in the current machine state.

cannot (event)

Returns if the given event cannot be fired in the current state.

is_finished()

Returns if the state machine is in its final state.

isstate(state)

Returns if the given state is the current state.

trigger(event)

Triggers the given event. The event can be triggered by calling the event handler directly, for ex: fsm.eat() but this method will come in handy if the event is determined dynamically and you have the event name to trigger as a string.

exception enuActor.MyFSM.FysomError

Bases: exceptions. Exception

Raised whenever an unexpected event gets triggered.

1.6 enuActor.QThread module

```
class enuActor.QThread.QMsg (method, *argl, **argd)
```

Bases: object

DEFAULT PRIORITY = 5

Bases: threading. Thread

exitMsq(cmd=None)

handler for the "exit" message. Spits out a message and arranges for the .run() method to exit.

handleTimeout()

Called when the .get() times out. Intended to be overridden.

pingMsg(cmd=None)

handler for the 'ping' message.

putMsg (method, *argl, **argd)

send ourself a new message.

Parameters

- method a function or bound method to call
- *argl the arguments to the method.
- *argd the arguments dict to the method

run()

Main run loop for this thread.

```
sendLater (msg, deltaTime, priority=1)
```

Send ourself a QMsg after deltaTime seconds.

- 1.7 enuActor.main module
- 1.8 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

е

```
enuActor, 12
enuActor.Commands, 3
enuActor.DecoratorStateMachine, 10
enuActor.Devices, 7
enuActor.Devices.Device, 3
enuActor.Devices.Error, 5
enuActor.Devices.rexm, 5
enuActor.Devices.shutter, 6
enuActor.MyFSM, 10
enuActor.QThread, 11
```

enuActor	Documentatio	n. Release

16 Python Module Index

A	Н
available_link (enuActor.Devices.Device.Device attribute), 3	handleTimeout() (enuActor.Devices.rexm.Rexm method), 5
	handleTimeout() (enuActor.Devices.shutter.Shutter
C	method), 6
can() (enuActor.MyFSM.Fysom method), 10	handleTimeout() (enuActor.QThread.QThread method),
cannot() (enuActor.MyFSM.Fysom method), 11	11
CfgFileErr, 5 check_status() (enuActor.Devices.rexm.Rexm method), 5	
check_status() (enuActor.Devices.rexiii.Rexiii inetilod), 5 check_status() (enuActor.Devices.shutter.Shutter	initialise() (enuActor.Devices.rexm.Rexm method), 5
method), 6	initialise() (enuActor.Devices.shutter.Shutter method), 6
CommErr, 5	initialize() (enuActor.DecoratorStateMachine.TransitionTable
ContextBase (class in enuActor.DecoratorStateMachine),	method), 10 is_finished() (enuActor.MyFSM.Fysom method), 11
10	isstate() (enuActor.MyFSM.Fysom method), 11
D	
DEFAULT_PRIORITY (enuActor.QThread.QMsg	L
attribute), 11	load_cfg() (enuActor.Devices.Device.Device method), 3
Device (class in enuActor.Devices.Device), 3	M
DeviceErr, 5	
E	MASK_ERROR_SB_1 (enu- Actor.Devices.shutter.Shutter attribute),
enuActor (module), 12	6
enuActor.Commands (module), 3	MASK_ERROR_SB_3 (enu-
enuActor.DecoratorStateMachine (module), 10	Actor.Devices.shutter.Shutter attribute),
enuActor.Devices (module), 7	6
enuActor.Devices.Device (module), 3 enuActor.Devices.Error (module), 5	MASK_ERROR_SB_4 (enu- Actor.Devices.shutter.Shutter attribute),
enuActor.Devices.rexm (module), 5	Actor.Devices.shutter.Shutter attribute),
enuActor.Devices.shutter (module), 6	MASK_ERROR_SB_5 (enu-
enuActor.MyFSM (module), 10	Actor.Devices.shutter.Shutter attribute),
enuActor.QThread (module), 11	6
event() (in module enuActor.DecoratorStateMachine), 10 exitMsg() (enuActor.QThread.QThread method), 11	MASK_ERROR_SB_6 (enu-
exitivisg() (enuActor.QTineau.QTineau method), 11	Actor.Devices.shutter.Shutter attribute),
F	move() (enuActor.Devices.rexm.Rexm method), 5
Fysom (class in enuActor.MyFSM), 10	
FysomError, 11	N
G	nextStates() (enuActor.DecoratorStateMachine.TransitionTable method), 10
getStatus() (enuActor.Devices.rexm.Rexm method), 5	

getStatus() (enuActor.Devices.shutter.Shutter method), 6

P	transition() (in module enu-
parseStatusByte() (enuActor.Devices.shutter.Shutter method), 6	Actor.DecoratorStateMachine), 10 transition() (in module enuActor.Devices.Device), 4
pingMsg() (enuActor.QThread.QThread method), 11 positions (enuActor.Devices.shutter.Shutter attribute), 7	transitionevent() (in module enu- Actor.DecoratorStateMachine), 10
printstateonchange() (enuActor.Devices.Device.Device method), 4	TransitionTable (class in enu- Actor.DecoratorStateMachine), 10
PRIORITY_DEFAULT (enu- Actor.Devices.Error.RuleError attribute), 5	trigger() (enuActor.MyFSM.Fysom method), 11 truncated() (in module enu- Actor.DecoratorStateMachine), 10
putMsg() (enuActor.QThread.QThread method), 11	
Q	
QMsg (class in enuActor.QThread), 11 QThread (class in enuActor.QThread), 11	
R	
Rexm (class in enuActor.Devices.rexm), 5 RuleError, 5 run() (enuActor.QThread.QThread method), 11	
S	
send() (enuActor.Devices.Device method), 4 sendLater() (enuActor.QThread.QThread method), 11 Shutter (class in enuActor.Devices.shutter), 6 shutter() (enuActor.Devices.shutter.Shutter method), 7 shutter_id (enuActor.Devices.shutter.Shutter attribute), 7 start_communication() (enuActor.Devices.Device.Device	
tribute), 6	
STATUS_BYTE_4 (enuActor.Devices.shutter.Shutter attribute), 6	
STATUS_BYTE_5 (enuActor.Devices.shutter.Shutter attribute), 6	
STATUS_BYTE_6 (enuActor.Devices.shutter.Shutter attribute), 6	
Т	
terminal() (enuActor.Devices.shutter.Shutter method), 7	

18 Index