# DOCUMENTATION OF TUXDROID PYTHON API 0.2.3 (SVN/UNRELEASED)

**Table of content :**

```
   1.3.6)    Methode : tux.daemon.get_client_name
   1.3.7)    Methode : tux.daemon.get_my_client_id
   1.3.8)    Methode : tux.daemon.get_version
   1.3.9)    Methode : tux.daemon.kill
   1.3.10)   Methode : tux.daemon.print_clients_name
   1.3.11)   Methode : tux.daemon.set_my_client_name
1.4) Object : tux.event (tuxapi_class.TUXevent)
   1.4.1)    Methode : tux.event.clear
   1.4.2)    Methode : tux.event.remote_key_to_string
   1.4.3)    Methode : tux.event.restore
   1.4.4)    Methode : tux.event.store
   1.4.5)    Methode : tux.event.wait_bt_pushed
   1.4.6)    Methode : tux.event.wait_head_bt_pushed
   1.4.7)    Methode : tux.event.wait_head_bt_released
   1.4.8)    Methode : tux.event.wait_lwing_bt_pushed
   1.4.9)    Methode : tux.event.wait_lwing_bt_released
   1.4.10)   Methode : tux.event.wait_remote_bt
   1.4.11)   Methode : tux.event.wait_rwing_bt_pushed
   1.4.12)   Methode : tux.event.wait_rwing_bt_released
   1.4.13)   Methode : tux.event.wait_stable_status
   1.4.14)   Methode : tux.event.wait_status
1.5) Object : tux.hw (tuxapi_class.TUXhw)
   1.5.1)    Methode : tux.hw.alsa_devices_count
   1.5.2)    Methode : tux.hw.alsa_devices_select
   1.5.3)    Methode : tux.hw.audio_get_version
   1.5.4)    Methode : tux.hw.behavior_get_version
   1.5.5)    Methode : tux.hw.donglerf_get_version
   1.5.6)    Methode : tux.hw.oss_device
   1.5.7)    Methode : tux.hw.tuxrf_get_version
1.6) Object : tux.micro (tuxapi_class.TUXmicro)
   1.6.1)    Methode : tux.micro.on
   1.6.2)    Methode : tux.micro.off
   1.6.3)    Methode : tux.micro.capture_start
   1.6.4)    Methode : tux.micro.capture_start_free
   1.6.5)    Methode : tux.micro.capture_stop
1.7) Object : tux.misc (tuxapi_class.TUXmisc)
   1.7.1)    Methode : tux.misc.doc
   1.7.2)    Methode : tux.misc.build_documentation
   1.7.3)    Methode : tux.misc.print_api_version
   1.7.4)    Methode : tux.misc.simulate_remote_key
   1.7.5)    Methode : tux.misc.simulate_tcpip_frame
1.8) Object : tux.monitoring (tuxapi_class.TUXmonitoring)
   1.8.1)    Methode : tux.monitoring.insert
   1.8.2)    Methode : tux.monitoring.remove
1.9) Object : tux.status (tuxapi_class.TUXStatus)
   1.9.1)    Methode : tux.status.charger_state
   1.9.2)    Methode : tux.status.flash_status
   1.9.3)    Methode : tux.status.eyes_closed
   1.9.4)    Methode : tux.status.eyes_counter
   1.9.5)    Methode : tux.status.eyes_motor
   1.9.6)    Methode : tux.status.eyes_opened
   1.9.7)    Methode : tux.status.get
   1.9.8)    Methode : tux.status.head_bt
   1.9.9)    Methode : tux.status.ir_led
   1.9.10)   Methode : tux.status.ir_signal
   1.9.11)   Methode : tux.status.light_level
   1.9.12)   Methode : tux.status.battery_level
   1.9.13)   Methode : tux.status.lled
   1.9.14)   Methode : tux.status.lwing_bt
   1.9.15)   Methode : tux.status.mouth_closed
   1.9.16)   Methode : tux.status.mouth_counter
   1.9.17)   Methode : tux.status.mouth_motor
   1.9.18)   Methode : tux.status.mouth_opened
   1.9.19)   Methode : tux.status.power_plug
   1.9.20)   Methode : tux.status.rf_state
   1.9.21)   Methode : tux.status.rled
   1.9.22)   Methode : tux.status.rwing_bt
```

1.9.23) Methode : tux.status.sounds_count
1.9.24) Methode : tux.status.sound_muted
1.9.25) Methode : tux.status.spin_backward
1.9.26) Methode : tux.status.spin_bt
1.9.27) Methode : tux.status.spin_counter
1.9.28) Methode : tux.status.wings_backward
1.9.29) Methode : tux.status.wings_counter
1.9.30) Methode : tux.status.wings_motor
1.9.31) Methode : tux.status.wings_forward
1.9.32) Methode : tux.status.wings_bt
1.9.33) Methode : tux.status.get_wings_position_switch
1.9.34) Methode : tux.status.to_string
1.10) Object : tux.sys (tuxapi_class.TUXsys)
1.10.1) Methode : tux.sys.add_time_event
1.10.2) Methode : tux.sys.clear_time_events
1.10.3) Methode : tux.sys.delayed_function
1.10.4) Methode : tux.sys.delete_time_event
1.10.5) Methode : tux.sys.looped_function
1.10.6) Methode : tux.sys.shell
1.10.7) Methode : tux.sys.shell_free
1.10.8) Methode : tux.sys.time
1.10.9) Methode : tux.sys.wait
1.11) Object : tux.tts (tuxapi_class.TUXtts)
1.11.1) Methode : tux.tts.auto_connect
1.11.2) Methode : tux.tts.connect
1.11.3) Methode : tux.tts.disconnect
1.11.4) Methode : tux.tts.kill_daemon
1.11.5) Methode : tux.tts.pause
1.11.6) Methode : tux.tts.play
1.11.7) Methode : tux.tts.select_voice
1.11.8) Methode : tux.tts.speak
1.11.9) Methode : tux.tts.speak_free
1.11.10) Methode : tux.tts.stop
1.12) Object : tux.wav (tuxapi_class.TUXwav)
1.12.1) Methode : tux.wav.play
1.12.2) Methode : tux.wav.play_free
1.12.3) Methode : tux.wav.pause
1.12.4) Methode : tux.wav.stop
1.12.5) Methode : tux.wav._continue
1.12.6) Methode : tux.wav.get_duration

**1) Object : tux (tuxapi_class.TUXTCPCommunicator)**

Main class of tux object
Sub class of this class:

| | |
|---|---|
| "cmd" as class | : Class which manages the tux commands |
| "connect" as class | : Class which manages connection functions |
| "daemon" as class | : Class which manages the daemon commands |
| "event" as class | : Class which manages the events |
| "hw" as class | : Class which manages the tux hardware |
| "micro" as class | : Class which manages the microphone functions |
| "misc" as class | : Class which manages the miscellaneous functions |
| "monitoring" as class | : Class which manages the monitoring |
| "status" as class | : Class which manages the request of a status |
| "sys" as class | : Class which manages the system functions |
| "tts" as class | : Class which manages the text to speech |
| "wav" as class | : Class which manages the wav functions |

Global variables of this class:

| | |
|---|---|
| "my_name" as string | : Name of the api instance |
| "print_status" as boolean | : Allow to print the raw statuses |
| "print_warnings" as boolean | : Allow to print the warnings |

Comments:
Call the destroying function at the end of your script for closing the api correctly.

```
>>> tux.destroy()
```

"tts" class is compatible with "tuxttsd" v 0.3.0 or more recent

**1.1) Object : tux.cmd (tuxapi_class.TUXcmd)**

Class which manages the tux commands

Global variables of this class:
"last_ack" as integer : ACK value of the last command
(ACK_CMD_DONGLE_NOT_PRESENT|ACK_CMD_TIMEOUT|
ACK_CMD_OK|ACK_CMD_KO|ACK_CMD_ERROR)
"no_ack" as boolean : Allow to wait a ACK from tuxd

### 1.1.1) Methode : tux.cmd.audio_channel_general

Send a command to tux for selecting the "general" audio channel in the dongle

Example:
>>> tux.cmd.audio_channel_general()

### 1.1.2) Methode : tux.cmd.audio_channel_tts

Send a command to tux for selecting the "TTS" audio channel in the dongle

Example:
>>> tux.cmd.audio_channel_tts()

### 1.1.3) Methode : tux.cmd.eyes_close

Send a command to tux for closing the eyes

Example:
>>> tux.cmd.eyes_close()

### 1.1.4) Methode : tux.cmd.eyes_off

Send a command to tux for stopping the eyes movement

Example:
>>> tux.cmd.eyes_off()

### 1.1.5) Methode : tux.cmd.eyes_on

Send a command to tux for moving the eyes

Parameters:
"count" as integer : Number of movements
(default = 1)

Example:
>>> tux.cmd.eyes_on()
>>> tux.cmd.eyes_on(2)

### 1.1.6) Methode : tux.cmd.eyes_on_free

Send a command to tux for moving the eyes in free mode

Parameters:
"count" as integer : number of movements
(default = 1)

Example:
>>> tux.cmd.eyes_on_free()
>>> tux.cmd.eyes_on_free(2)

### 1.1.7) Methode : tux.cmd.eyes_open

Send a command to tux for opening the eyes

```
        Example:
        >>> tux.cmd.eyes_open()
```

#### ***1.1.8) Methode : tux.cmd.ir_off***

```
    Send a command to tux for turning the IR off

        Example:
        >>> tux.cmd.ir_off()
```

#### ***1.1.9) Methode : tux.cmd.ir_on***

```
    Send a command to tux for turning the IR on

        Example:
        >>> tux.cmd.ir_on()
```

#### ***1.1.10) Methode : tux.cmd.ir_send***

```
    Send a command to tux for sending an IR code

        Parameters:
        "address" as integer        : RC5 address
        "command" as integer        : RC5 command

        Example:
        >>> tux.cmd.ir_send(1,1)
```

#### ***1.1.11) Methode : tux.cmd.ledl_during***

```
    Send a command to tux for turning the left led on during a specified
    time

        Parameters:
        "seconds" as float      : time to wait in seconds

        Example:
        >>> tux.cmd.ledl_during(2.5)
```

#### ***1.1.12) Methode : tux.cmd.ledl_during_free***

```
    Send a command to tux for turning the left led on during a specified
    time in free mode

        Parameters:
        "seconds" as float      : time to wait in seconds

        Example:
        >>> tux.cmd.ledl_during_free(2.5)
```

#### ***1.1.13) Methode : tux.cmd.ledl_off***

```
    Send a command to tux for turning the left led off

        Example:
        >>> tux.cmd.ledl_off()
```

#### ***1.1.14) Methode : tux.cmd.ledl_on***

```
    Send a command to tux for turning the left led on

        Example:
        >>> tux.cmd.ledl_on()
```

#### ***1.1.15) Methode : tux.cmd.ledr_during***

```
    Send a command to tux for turning the right led on during a specified
```

time

Parameters:
"seconds" as float : time to wait in seconds

Example:
>>> tux.cmd.ledr_during(2.5)

### 1.1.16) Methode : tux.cmd.ledr_during_free

Send a command to tux for turning the right led on during a specified time in free mode

Parameters:
"seconds" as float : time to wait in seconds

Example:
>>> tux.cmd.ledr_during_free(2.5)

### 1.1.17) Methode : tux.cmd.ledr_off

Send a command to tux for turning the right led off

Example:
>>> tux.cmd.ledr_off()

### 1.1.18) Methode : tux.cmd.ledr_on

Send a command to tux for turning the right led on

Example:
>>> tux.cmd.ledr_on()

### 1.1.19) Methode : tux.cmd.leds_blink

Send a command to tux for blinking the leds

Parameters:
"count" as integer : number of blink(0..255)
"delay" as integer : delay between 2 states (u=4msec)

Example:
>>> tux.cmd.leds_blink(10,25)

### 1.1.20) Methode : tux.cmd.leds_during

Send a command to tux for turning the leds on during a specified time

Parameters:
"seconds" as float : time to wait in seconds

Example:
>>> tux.cmd.leds_during(2.5)

### 1.1.21) Methode : tux.cmd.leds_during_free

Send a command to tux for turning the leds on during a specified time in free mode

Parameters:
"seconds" as float : time to wait in seconds

Example:
>>> tux.cmd.leds_during_free(2.5)

### 1.1.22) Methode : tux.cmd.leds_off

Send a command to tux for turning the leds off

Example:
>>> tux.cmd.leds_off()

### 1.1.23) Methode : tux.cmd.leds_on

Send a command to tux for turning the leds on

Example:
>>> tux.cmd.leds_on()

### 1.1.24) Methode : tux.cmd.mouth_close

Send a command to tux for closing the mouth

Example:
>>> tux.cmd.mouth_close()

### 1.1.25) Methode : tux.cmd.mouth_off

Send a command to tux for stopping the mouth movement

Example:
>>> tux.cmd.mouth_off()

### 1.1.26) Methode : tux.cmd.mouth_on

Send a command to tux for moving the mouth

Parameters:
"count" as integer : number of movements
(default = 1)

Example:
>>> tux.cmd.mouth_on()
>>> tux.cmd.mouth_on(2)

### 1.1.27) Methode : tux.cmd.mouth_on_free

Send a command to tux for moving the mouth in free mode

Parameters:
"count" as integer : number of movements
(default = 1)

Example:
>>> tux.cmd.mouth_on_free()
>>> tux.cmd.mouth_on_free(2)

### 1.1.28) Methode : tux.cmd.mouth_open

Send a command to tux for opening the mouth

Example:
>>> tux.cmd.mouth_open()

### 1.1.29) Methode : tux.cmd.ping

Send a command to tux for the "ping-pong" test

Parameters:
"count" as integer : number of pong requested
(default = 200)

Example:
>>> tux.cmd.ping()

```
>>> tux.cmd.ping(200)
```

### 1.1.30) Methode : tux.cmd.raw

Send a raw command to tux

```
Parameters:
"cmd" as integer        : command
"param1" as integer     : parameter 1 of these command
"param2" as integer     : parameter 2 of these command
"param3" as integer     : parameter 3 of these command

Return a ACK if it's required

Example:
>>> tux.cmd.raw(BLINK_EYES_CMD,4,0,0)
```

### 1.1.31) Methode : tux.cmd.sleep_off

Send a command to tux for turning the sleep mode off

```
Example:
>>> tux.cmd.sleep_off()
```

### 1.1.32) Methode : tux.cmd.sleep_on

Send a command to tux for turning the sleep mode on

```
Example:
>>> tux.cmd.sleep_on()
```

### 1.1.33) Methode : tux.cmd.sound_play

Send a command to tux for playing a sound from the flash memory

```
Parameters:
"index" as integer      : index of the sound
"volume" as integer     : volume of the sound(0..7,0=max)
                          (default = 0)

Example:
>>> tux.cmd.sound_play(1)
>>> tux.cmd.sound_play(1,0)
```

### 1.1.34) Methode : tux.cmd.sound_store_index

Send a command to tux for indexing the sound

```
Parameters:
"highAdd" as integer    : high byte address of the sound
"middleAdd" as integer  : middle byte address of the sound
"lowAdd" as integer     : low byte address of the sound

Example:
>>> tux.cmd.sound_store_index(0x00,0x04,0x00)
```

### 1.1.35) Methode : tux.cmd.sound_storing

Send a command to tux for storing a sound collection in the memory flash

```
Parameters:
"number" as integer     : number of sounds to be stored

Example:
>>> tux.cmd.sound_storing(10)
```

***1.1.36)  Methode : tux.cmd.sound_test***

Send a command to tux for testing the sound in tux

```
Example:
>>> tux.cmd.sound_test()
```

***1.1.37)  Methode : tux.cmd.spinl_off***

Send a command to tux for stopping the spinning movement

```
Example:
>>> tux.cmd.spinl_off()
```

***1.1.38)  Methode : tux.cmd.spinl_on***

Send a command to tux to spin to the left

```
Parameters:
"count" as integer      : number of quarter turns
                          (default = 4)
"speed" as integer      : speed of the movement(1..5)
                          (default = 5)

Example:
>>> tux.cmd.spinl_on()
>>> tux.cmd.spinl_on(2)     (count = 2)
>>> tux.cmd.spinl_on(2,5)   (count = 2 and speed = 5)
```

***1.1.39)  Methode : tux.cmd.spinl_on_free***

Send a command to tux to spin to the left in free mode

```
Parameters:
"count" as integer      : number of quarter turns
                          (default = 4)
"speed" as integer      : speed of the movement(1..5)
                          (default = 5)

Example:
>>> tux.cmd.spinl_on_free()
>>> tux.cmd.spinl_on_free(2)     (count = 2)
>>> tux.cmd.spinl_on_free(2,5)   (count = 2 and speed = 5)
```

***1.1.40)  Methode : tux.cmd.spinr_off***

Send a command to tux for stopping the spinning movement

```
Example:
>>> tux.cmd.spinr_off()
```

***1.1.41)  Methode : tux.cmd.spinr_on***

Send a command to tux to spin to the right

```
Parameters:
"count" as integer      : number of quarter turns
                          (default = 4)
"speed" as integer      : speed of the movement(1..5)
                          (default = 5)

Example:
>>> tux.cmd.spinr_on()
>>> tux.cmd.spinr_on(2)     (count = 2)
>>> tux.cmd.spinr_on(2,5)   (count = 2 and speed = 5)
```

### 1.1.42) Methode : tux.cmd.spinr_on_free

Send a command to tux to spin to the right in free mode

```
Parameters:
"count" as integer      : number of quarter turns
                          (default = 4)
"speed" as integer      : speed of the movement(1..5)
                          (default = 5)

Example:
>>> tux.cmd.spinr_on_free()
>>> tux.cmd.spinr_on_free(2)     (count = 2)
>>> tux.cmd.spinr_on_free(2,5)   (count = 2 and speed = 5)
```

### 1.1.43) Methode : tux.cmd.structured

Send a structured command to tux

```
Parameters:
"fct" as integer        : function of tux
"cmd" as integer        : command for this function
"param1" as integer     : parameter 1 for this command
"param2" as integer     : parameter 2 for this command
"param3" as integer     : parameter 3 for this command

Return a ACK as integer

Example:
>>> tux.cmd.structured(TUX_CMD_STRUCT_EYES,TUX_CMD_STRUCT_SUB_ON
,count,0,0)
```

### 1.1.44) Methode : tux.cmd.wings_off

Send a command to tux for stopping the wings movement

```
Example:
>>> tux.cmd.wings_off()
```

### 1.1.45) Methode : tux.cmd.wings_on

Send a command to tux for moving the wings

```
Parameters:
"count" as integer      : number of movements
                          (default = 1)
"speed" as integer      : speed of the movement(1-5)
                          (default = 5)

Example:
>>> tux.cmd.wings_on()
>>> tux.cmd.wings_on(2)     (2 movements)
>>> tux.cmd.wings_on(2,5)   (2 movements and speed=5)
```

### 1.1.46) Methode : tux.cmd.wings_on_free

Send a command to tux for moving the wings in free mode

```
Parameters:
"count" as integer      : number of movements
                          (default = 1)
"speed" as integer      : speed of the movement(1..5)
                          (default = 5)

Example:
>>> tux.cmd.wings_on_free()
>>> tux.cmd.wings_on_free(2)     (2 movements)
```

```
        >>> tux.cmd.wings_on_free(2,5)   (2 movements and speed=5)
```

## 1.2) Object : tux.connect (tuxapi_class.TUXconnect)

Class which manages connection functions

### *1.2.1) Methode : tux.connect.disconnect_from_tux*

```
    Disconnect from tux

        Parameters:
        "seconds" as float : Time to wait in seconds

        Example:
        >>> tux.sys.wait(2.4)
```

### *1.2.2) Methode : tux.connect.connect_to_tux*

```
    Connect to a tux by its ID

        Parameters:
        "id" as uint16 : ID of the tux you want to connect to
                         0 and 0xFFFF (65536) are not valid ID's

        Returns:
        "True" if the command has been sent successfully,
        "False" otherwise.

        Example:
        >>> tux.connect.connect_to_tux(300)
```

### *1.2.3) Methode : tux.connect.random_connect_to_tux*

```
    Connect to the first tux discovered

    Catch any disconnected tux, request it's ID and connect to it.

        Returns:
        "True" if the command has been sent successfully,
        "False" otherwise.
```

### *1.2.4) Methode : tux.connect.id_request*

```
    Get the ID of tux currently connected

        Returns:
        "id" as uint16 (0 < id < 65536) if the command has been sent
                successfully,
        "False" otherwise.
```

### *1.2.5) Methode : tux.connect.id_lookup*

```
    Get the ID of the first disconnected tux which is discoverred

    The first disconnected tux that will detect this command will reply
    with it's ID and disconnect immediately. You can then connect to
    that tux with the ID you just got.

    In order to get the ID's of more than one disconnected tux, you have
    to issue this command multiple times until you don't get any new ID.

        Returns:
        "id" as uint16 (0 < id < 65536) if the command has been sent
                successfully,
        "False" otherwise.
```

#### *1.2.6) Methode : tux.connect.change_id*

Changes the ID of a disconnected tux

You have to push on the head button of tux for XXX seconds while sending this command in order to validate the ID change request, this in order to avoid stealing a tux too easily.

Parameters:
"id" as uint16 : new ID you want to set to your tux
0 and 0xFFFF (65536) are not valid ID's

Returns:
"True" if the command has been sent successfully,
"False" otherwise.

#### *1.2.7) Methode : tux.connect.sleep*

Set tux in sleep mode

Returns:
"True" if the command has been sent successfully,
"False" otherwise.

#### *1.2.8) Methode : tux.connect.wakeup*

Wake-up a tux if it's in sleep mode.

Parameters:
"id" as uint16 : ID of the tux you want to wake-up
0 and 0xFFFF (65536) are not valid ID's

Returns:
"True" if the command has been sent successfully,
"False" otherwise.

#### *1.2.9) Methode : tux.connect.avoid_wifi*

Configure the RF module to avoid a given wifi channel

Parameters:
"channel" : wifi channel number that tux should avoid using
Channels from 1 to 14 are valid wifi channels. Set the wifi channel to 0 to disable channel avoidance and use the complete range of frequencies.

Returns:
"True" if the command has been sent successfully,
"False" otherwise.

### 1.3) Object : tux.daemon (tuxapi_class.TUXdaemon)

Class which manages the daemon commands

Global variables of this class:
"connected" as boolean : State of the connection to tuxd

#### *1.3.1) Methode : tux.daemon.auto_connect*

Allow to connect the api to tuxd automatically

Parameters:
"value" as boolean : turn on/off the auto_connect mode
"address" as string : Tcp/IP Host address
(default = 'localhost')
"port" as integer : Tcp/IP Port number

```
                                (default = 5000)

        Examples:
        >>> tux.daemon.auto_connect(True)
        >>> tux.daemon.auto_connect(True,'192.168.0.1')
        >>> tux.daemon.auto_connect(True,'192.168.0.1',5000)
```

### 1.3.2) Methode : tux.daemon.connect

```
    Connect tux object to tuxd

        Parameters:
        "port" as integer       : Tcp/IP Port number
                                  (default = 5000)
        "address" as string     : Tcp/IP Host address
                                  (default = 'localhost')

        Examples:
        >>> tux.daemon.connect()
        >>> tux.daemon.connect('192.168.0.1')
        >>> tux.daemon.connect('192.168.0.1',5000)

        Comment:
        The variable "tux.daemon.connected" contains the result of
        this method
```

### 1.3.3) Methode : tux.daemon.disconnect

```
    Disconnect tux object from tuxd

        Example:
        >>> tux.daemon.disconnect()
```

### 1.3.4) Methode : tux.daemon.disconnect_client

```
    Disconnect a client from tuxd

        Parameters:
        "id_client" as integer  : id of the client to disconnect

        Example:
        >>> tux.daemon.disconnect_client(0)
```

### 1.3.5) Methode : tux.daemon.get_client_count

```
    Get the number of clients connected to tuxd

        Return an integer

        Example:
        >>> print tux.daemon.get_client_count()
```

### 1.3.6) Methode : tux.daemon.get_client_name

```
    Get the name of a client of tuxd

        Return a string

        Example:
        >>> print tux.daemon.get_client_name(0)
```

### 1.3.7) Methode : tux.daemon.get_my_client_id

```
    Get the client id of my connection to tuxd

        Return an integer
```

Example:
>>> print tux.daemon.get_my_client_id()

#### ***1.3.8) Methode : tux.daemon.get_version***

Get the version of tuxd

Return a string

Example:
>>> print tux.daemon.get_version()

#### ***1.3.9) Methode : tux.daemon.kill***

Kill tuxd

Example:
>>> tux.daemon.kill()

#### ***1.3.10) Methode : tux.daemon.print_clients_name***

Print the name of all the clients connected to tuxd

Example:
>>> tux.daemon.print_clients_name()

#### ***1.3.11) Methode : tux.daemon.set_my_client_name***

Set my client name on tuxd

Parameters:
"name" as string : name of this client (max 11 char)

Example:
>>> tux.daemon.set_my_client_name('Py client')

Comment:
The variable 'tux.my_name' is affected by this function

### 1.4) Object : tux.event (tuxapi_class.TUXevent)

Class which manages the events

Global variables of this class:

```
"on_bt_pushed" as pof           : On tux button pushed
"on_head_bt_pushed" as pof      : On tux head button pushed
"on_lwing_bt_pushed" as pof     : On tux left wing button pushed
"on_rwing_bt_pushed" as pof     : On tux right wing button pushed
"on_bt_released" as pof         : On tux button released
"on_head_bt_released" as pof    : On tux head button released
"on_lwing_bt_released" as pof   : On tux left wing button released
"on_rwing_bt_released" as pof   : On tux right wing button released
"on_remote_bt" as list of pof   : On remote controller button pressed
"on_status" as pof              : On status arrival
"on_remote" as pof              : On remote controller event
                                  param 1 : Key as integer
"on_light_level" as pof         : On light level event
                                  param 1 : light value as integer
"on_connected" as pof           : On api connected to tuxd
"on_disconnected" as pof        : On api disconnect from tuxd
"on_mouth_open" as pof          : On mouth open event
"on_mouth_close" as pof         : On mouth close event
"on_power_plugged" as pof       : On power plugged event
"on_power_unplugged" as pof     : On power unplugged event
"on_left_blue_led_on" as pof    : On left blue led changed to on
"on_left_blue_led_off" as pof   : On left blue led changed to off
"on_right_blue_led_on" as pof   : On right blue led changed to on
```

```
"on_right_blue_led_off" as pof  : On right blue led changed to off
"on_eyes_open" as pof           : On eyes open event
"on_eyes_close" as pof          : On eyes close event
"on_rf_connected" as pof        : On RF is connected
"on_rf_disconnected" as pof     : On RF is disconnected
"on_pong_received" as pof       : On pong status received
"on_mouth_stop" as pof          : On mouth stop event
"on_eyes_stop" as pof           : On eyes stop event
"on_wings_stop" as pof          : On wings stop event
"on_spin_stop" as pof           : On spin stop event
(pof = pointer of function)

Example of associating a function to a remote event:
>>> def my_function(key):
>>>     print "Button %s is pressed"%remote_bt_name[key]
>>> tux.event.on_remote=my_function

Example of associating a function to a specific remote event:
>>> def play_pause():
>>> tux.sys.shell("audacious --play-pause")
>>> tux.event.on_remote_bt[K_PLAYPAUSE]=play_pause

Key constants of the remote controller:
(K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_8,K_9,K_STANDBY,
K_MUTE,K_VOLUMEPLUS,K_VOLUMEMINUS,K_ESCAPE,K_YES,
K_NO,K_BACKSPACE,K_STARTVOIP,K_RECEIVECALL,K_HANGUP,
K_STAR,K_SHARP,K_RED,K_GREEN,K_BLUE,K_YELLOW,
K_CHANNELPLUS,K_CHANNELMINUS,K_UP,K_DOWN,K_LEFT,
K_RIGHT,K_OK,K_FASTREWIND,K_FASTFORWARD,K_PLAYPAUSE,
K_STOP,K_RECORDING,K_PREVIOUS,K_NEXT,K_MENU,K_MOUSE,
K_ALT)
```

### ***1.4.1) Methode : tux.event.clear***

Clear all events

Example:
```
>>> tux.event.clear()
```

### ***1.4.2) Methode : tux.event.remote_key_to_string***

Get the string name of a remote key value

Parameters:
```
"key" as integer    : key to translate to string
```

Return a string

Example:
```
>>> print tux.event.remote_key_to_string(10)
```

### ***1.4.3) Methode : tux.event.restore***

Restore all events

Example:
```
>>> tux.event.restore()
```

### ***1.4.4) Methode : tux.event.store***

Store all events

Example:
```
>>> tux.event.store()
```

### ***1.4.5) Methode : tux.event.wait_bt_pushed***

Wait until a tux button is pushed

Parameters:
"time_out" as integer : Time-out in seconds
(9999 = infinite wait)

Return the button value as integer
(HEAD_BT|LEFT_WING_BT|RIGHT_WING_BT|NONE_BT)

Example:
>>> tux.event.wait_bt_pushed(10)

### ***1.4.6) Methode : tux.event.wait_head_bt_pushed***

Wait until head button is pushed

Parameters:
"time_out" as integer : Time-out in seconds
(9999 = infinite wait)

Return a boolean

Example:
>>> tux.event.wait_head_bt_pushed(2)

### ***1.4.7) Methode : tux.event.wait_head_bt_released***

Wait until head button is released

Parameters:
"time_out" as integer : Time-out in seconds
(9999 = infinite wait)

Return a boolean

Example:
>>> tux.event.wait_head_bt_released(2)

### ***1.4.8) Methode : tux.event.wait_lwing_bt_pushed***

Wait until left wing is pushed

Parameters:
"time_out" as integer : Time-out in seconds
(9999 = infinite wait)

Return a boolean

Example:
>>> tux.event.wait_lwing_bt_pushed(2)

### ***1.4.9) Methode : tux.event.wait_lwing_bt_released***

Wait until left wing is released

Parameters:
"time_out" as integer : Time-out in seconds
(9999 = infinite wait)

Return a boolean

Example:
>>> tux.event.wait_lwing_bt_released(2)

### ***1.4.10) Methode : tux.event.wait_remote_bt***

Wait until a specified key of the remote is pressed

```
        Parameters:
        "time_out" as integer : Time-out in seconds
                                (9999 = infinite wait)

        Return a boolean

        Example:
        >>> tux.event.wait_remote_bt(K_OK,2)
```

***1.4.11) Methode : tux.event.wait_rwing_bt_pushed***

```
    Wait until right wing is pushed

        Parameters:
        "time_out" as integer : Time-out in seconds
                                (9999 = infinite wait)

        Return a boolean

        Example:
        >>> tux.event.wait_rwing_bt_pushed(2)
```

***1.4.12) Methode : tux.event.wait_rwing_bt_released***

```
    Wait until right wing is released

        Parameters:
        "time_out" as integer : Time-out in seconds
                                (9999 = infinite wait)

        Return a boolean

        Example:
        >>> tux.event.wait_rwing_bt_released(2)
```

***1.4.13) Methode : tux.event.wait_stable_status***

```
    Wait for stable status

        Parameters:
        "DATA_STATUS" as integer  : Desired status
        "DATA_VALUE" as integer   : Desired value
        "time_out" as integer     : Time-out in seconds
                                    (9999 = infinite wait)
        "stable_out" as integer   : Time of stable status request in seconds

        Return a boolean

        Example:
        >>> var=tux.event.wait_status(DATAS_STATUS_HEAD_PUSH_SWITCH,1,5,2)
```

***1.4.14) Methode : tux.event.wait_status***

```
    Wait until the specified status arrives

        Parameters:
        "DATA_STATUS" as integer  : Desired status
        "DATA_VALUE" as integer   : Desired value
        "time_out" as integer     : Time-out in seconds
                                    (9999 = infinite wait)

        Return a boolean

        Example:
        >>> var=tux.event.wait_status(DATAS_STATUS_HEAD_PUSH_SWITCH,1,2)
        (see 'tuxapi_const.py' for the complete list of statuses)
```

## 1.5) Object : tux.hw (tuxapi_class.TUXhw)

Class which manages the tux hardware

```
    Global variables of this class
    "TUX_devices" as tuple          : tuple of tux devices as tuple
                                      (param 0 = alsa device name)
                                      (param 1 = sound card)
                                      (param 2 = sound device)
                                      (param 3 = usb bus)
                                      (param 4 = usb device)
                                      (param 5 = usb PID)
                                      (param 6 = usb VID)
    "TUX_devices_count" as integer  : number of tux devices
    "alsa_device" as string         : current alsa device name
                                      (is selected on the beginning
                                       of the api with the first tux
                                       sound card)
```

### *1.5.1) Methode : tux.hw.alsa_devices_count*

```
    Return the number of tux alsa devices

        Return an integer

        Example:
        >>> print tux.hw.alsa_devices_count()
```

### *1.5.2) Methode : tux.hw.alsa_devices_select*

```
    Get the alsa device name of a tux sound card

        Return a string (ex: 'hw:1,0')

        Example:
        >>> print tux.hw.alsa_devices_select(0)
```

### *1.5.3) Methode : tux.hw.audio_get_version*

```
    Get the version of the audio firmware

        Return a tuple  : (major version, minor version, update version,
                           revision, author_id)

        Example:
        >>> print tux.hw.audio_get_version()
```

### *1.5.4) Methode : tux.hw.behavior_get_version*

```
    Get the version of the behavior firmware

        Return a tuple  : (major version, minor version, update version,
                           revision, author_id)

        Example:
        >>> print tux.hw.behavior_get_version()
```

### *1.5.5) Methode : tux.hw.donglerf_get_version*

```
    Get the version of the donglerf firmware

        Return a tuple  : (major version, minor version, update version,
                           revision, author_id)

        Example:
        >>> print tux.hw.donglerf_get_version()
```

#### *1.5.6) Methode : tux.hw.oss_device*

Get the oss compatibility device of the tux sound card

Return a string (ex : '/dev/dsp')

#### *1.5.7) Methode : tux.hw.tuxrf_get_version*

Get the version of the tuxrf firmware

```
Return a tuple  : (major version, minor version, update version,
                   revision, author_id)

Example:
>>> print tux.hw.tuxrf_get_version()
```

### 1.6) Object : tux.micro (tuxapi_class.TUXmicro)

Class which manages the microphone functions.

Global variables of this class:
"on_buffer" as EventControl : Event on new buffer from micro

Example of associating a function to 'on_buffer' event:

```
>>> def new_buffer(values):
>>> ... print values
>>> ...
>>> tux.micro.on_buffer.connect(new_buffer)
>>> tux.micro.on()
```

#### *1.6.1) Methode : tux.micro.on*

Function to turn on the micro capture.

```
Example:
>>> tux.micro.on()
```

#### *1.6.2) Methode : tux.micro.off*

Function to turn off the micro capture.

```
Example:
>>> tux.micro.off()
```

#### *1.6.3) Methode : tux.micro.capture_start*

Write the stream in a wav file.

```
Parameters:
"out_path" as string    : path of the wave file
"length" as float       : duration of the capture in seconds
                          When this parameters is omitted, the length
                          is infinite. You must stop the recording
                          with 'tux.micro.capture_stop()'

Examples:
>>> tux.micro.capture_start('/home/remi/Desktop/test.wav', 10.0)
>>> tux.micro.capture_start('/home/remi/Desktop/test.wav')
```

#### *1.6.4) Methode : tux.micro.capture_start_free*

Write the stream in a wav file.

```
Parameters:
"out_path" as string    : path of the wave file
```

```
            "length" as float       : duration of the capture in seconds
                                      When this parameters is omitted, the length
                                      is infinite. You must stop the recording
                                      with 'tux.micro.capture_stop()'

            Examples:
            >>> tux.micro.capture_start_free('/home/remi/Desktop/test.wav', 10.0)
            >>> tux.micro.capture_start_free('/home/remi/Desktop/test.wav')
```

#### ***1.6.5) Methode : tux.micro.capture_stop***

Stop the capture of the stream.

### **1.7) Object : tux.misc (tuxapi_class.TUXmisc)**

Class which manages the miscellaneous functions

#### ***1.7.1) Methode : tux.misc.doc***

Print the docstring of an element of tux api

```
            Parameters:
            "element" as methode or class

            Examples:
            >>> tux.misc.doc(tux)
            >>> tux.misc.doc(tux.cmd.eyes_on)
```

#### ***1.7.2) Methode : tux.misc.build_documentation***

Build the documentation of this api

```
            Parameters:
            "doc_path" as string    : path of the output text file

            Example:
            >>> tux.misc.build_documentation('/home/remi/tuxapi_doc')
```

#### ***1.7.3) Methode : tux.misc.print_api_version***

To print the version of the API

```
            Example:
            >>> tux.misc.print_api_version()
```

#### ***1.7.4) Methode : tux.misc.simulate_remote_key***

To simulate the receiving of a remote key

```
            Parameters:
            "key" as integer    : remote key

            Example:
            >>> tux.misc.simulate_remote_key(K_OK)
```

#### ***1.7.5) Methode : tux.misc.simulate_tcpip_frame***

To simulate the receiving of a frame

```
            Parameters:
            "frame" as list of 16 char : fake frame

            Example:
            >>> tux.misc.simulate_tcpip_frame()
```

### **1.8) Object : tux.monitoring (tuxapi_class.TUXmonitoring)**

Class which manages the monitoring

```
    Status constants list:
    (STATUS_WINGS_MOTOR_BACKWARD,       STATUS_SPIN_MOTOR_BACKWARD,
     STATUS_SPIN_MOTOR_FORWARD,         STATUS_MOUTH_OPEN_POSITION,
     STATUS_MOUTH_CLOSED_POSITION,      STATUS_HEAD_PUSH_POSITION,
     STATUS_CHARGER_INHIBIT_SIGNAL,     STATUS_WINGS_POSITION_SWITCH,
     STATUS_MOTOR_FOR_WINGS,            STATUS_LEFT_BLUE_LED,
     STATUS_I2C_SDA_LINE,               STATUS_I2C_SCL_LINE ,
     STATUS_HEAD_MOTOR_FOR_MOUTH,       STATUS_HEAD_MOTOR_FOR_EYES,
     STATUS_IR_RECEIVER_SIGNAL,         STATUS_SPIN_POSITION_SWITCH,
     STATUS_WINGS_MOTOR_FORWARD,        STATUS_IR_LED,
     STATUS_EYES_OPEN_POSITION_SWITCH,  STATUS_EYES_CLOSED_POSITION_SWITCH,
     STATUS_LEFT_WING_PUSH,             STATUS_RIGHT_WING_PUSH,
     STATUS_POWER_PLUG_SWITCH,          STATUS_HEAD_PUSH_SWITCH,
     STATUS_CHARGER_LED,                STATUS_MUTE_STATUS,
     STATUS_LIGHT_LEVEL,                STATUS_EYES_POSITION_COUNTER,
     STATUS_MOUTH_POSITION_COUNTER,     STATUS_WINGS_POSITION_COUNTER,
     STATUS_SPIN_POSITION_COUNTER,      STATUS_RIGHT_BLUE_LED,
     STATUS_RF_CONNECTED,               STATUS_IR_CODE,
     STATUS_SOUND_COUNT,                STATUS_PONG,
     STATUS_BATTERY,                    STATUS_MICRO_ENERGY,
    )
```

***1.8.1) Methode : tux.monitoring.insert***

Insert a monitoring event.

```
    Parameters:
    "status" as integer                 : Status index
                                          (See status contants list)
    "function" as pointer of function   : Function to bind

    Returns:
    The index of your event in the monitoring manager.
    You must save this index if you want removing your event.

    Exemple:
    >>> monitor_idx = tux.monitoring.insert(STATUS_LIGHT_LEVEL, my_function)
```

***1.8.2) Methode : tux.monitoring.remove***

Remove a monitoring event.

```
    Parameters:
    "event_id" as integer   : Index of the event.

    Exemple:
    >>> tux.monitoring.remove(monitor_idx)
```

**1.9) Object : tux.status (tuxapi_class.TUXStatus)**

Class which manages the request of a status

```
    Global variables of this class:
    "rf_connected" as boolean   : State of the droid/dongle connection
```

***1.9.1) Methode : tux.status.charger_state***

Get the status of the charger

```
    Return 1 for charging and 0 for not charging

    Example:
    >>>var = tux.status.charger_state()
```

***1.9.2) Methode : tux.status.flash_status***

Get the last sound flash status

Return a tupple with the audio flash status:
(play state, record state, sound number)
play state : Return the sound number or 0 if no sound is played
record state : Return the recording state
sound number : Return the track which is recorded.

Example:
>>> (play_state, record_state, sound_number) = tux.status.flash_status()

### 1.9.3) Methode : tux.status.eyes_closed

Get the last state of "eyes closed position switch" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.eyes_closed()

### 1.9.4) Methode : tux.status.eyes_counter

Get the number of remaining movements of the eyes

Return an integer (0..255)

Example:
>>> var = tux.status.eyes_counter()

### 1.9.5) Methode : tux.status.eyes_motor

Get the last state of "head motor for eyes" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.eyes_motor()

### 1.9.6) Methode : tux.status.eyes_opened

Get the last state of "eyes open position switch" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.eyes_opened()

### 1.9.7) Methode : tux.status.get

Get a specified status

Parameters:
"DATA_STATUS" as integer : desired status

Return a boolean
(True if the status arrives before 2 seconds)
(The raw of the status response is in tux.tcp_data)

Example:
>>> tux.status.get(DATAS_STATUS_WINGS_MOTOR_BACKWARD)

### 1.9.8) Methode : tux.status.head_bt

Get the status of the head button

Return 1 for on and 0 for off

Example:
>>> var = tux.status.head_bt()

***1.9.9) Methode : tux.status.ir_led***

Get the last state of "IR led" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.ir_led()

***1.9.10) Methode : tux.status.ir_signal***

Get the last state of "IR receiver signal" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.ir_signal()

***1.9.11) Methode : tux.status.light_level***

Get the last light level

Return an integer (0..1024)

Example:
>>> var = tux.status.light_level()

***1.9.12) Methode : tux.status.battery_level***

Get the last battery level

Return a tupple with the battery level and the status:
(level, status)
level: 0..1024
status: True (valid battery level) or False (i.e. a motor was running during the measurement so it's not accurate)

Example:
>>> (level, valid) = tux.status.battery_level()

***1.9.13) Methode : tux.status.lled***

Get the state of the left blue led

Return 1 for on and 0 for off

Example:
>>> var = tux.status.lled()

***1.9.14) Methode : tux.status.lwing_bt***

Get the last state of "left wing push" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.lwing_bt()

***1.9.15) Methode : tux.status.mouth_closed***

Get the last state of "mouth closed position" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.mouth_closed()

***1.9.16) Methode : tux.status.mouth_counter***

Get the number of remaining movements of the mouth

Return an integer (0..255)

Example:
>>> var = tux.status.mouth_counter()

***1.9.17) Methode : tux.status.mouth_motor***

Get the last state of "head motor for mouth" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.mouth_motor()

***1.9.18) Methode : tux.status.mouth_opened***

Get the last state of "mouth open position" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.mouth_opened()

***1.9.19) Methode : tux.status.power_plug***

Get the last state of "power plug switch" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.power_plug()

***1.9.20) Methode : tux.status.rf_state***

Get the last state of the RF status

Return 1 for connected and 0 for disconnected

Example:
>>>var = tux.status.rf_state()

***1.9.21) Methode : tux.status.rled***

Get the state of the right blue led

Return 1 for on and 0 for off

Example:
>>> var = tux.status.rled()

***1.9.22) Methode : tux.status.rwing_bt***

Get the last state of "right wing push" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.rwing_bt()

***1.9.23) Methode : tux.status.sounds_count***

Get the number of sounds stored in the flash memory

Return a integer (0..255)

Example:
```
>>> var = tux.status.sounds_count()
```

***1.9.24) Methode : tux.status.sound_muted***

Get the last state of "tux mute sound" status

Return 1 for on and 0 for off

Example:
```
>>> var = tux.status.sound_muted()
```

***1.9.25) Methode : tux.status.spin_backward***

Get the last state of "spin motor forward" status

Return 1 for on and 0 for off

Example:
```
>>> var = tux.status.spin_forward()
```

***1.9.26) Methode : tux.status.spin_bt***

Get the last state of "spin position switch" status

Return 1 for on and 0 for off

Example:
```
>>> var = tux.status.spin_bt()
```

***1.9.27) Methode : tux.status.spin_counter***

Get the number of remaining movements of 'spinning'

Return an integer (0..255)

Example:
```
>>> var = tux.status.spin_counter()
```

***1.9.28) Methode : tux.status.wings_backward***

Get the last state of "wings motor backward" status

Return 1 for on and 0 for off

Example:
```
>>> var = tux.status.wings_backward()
```

***1.9.29) Methode : tux.status.wings_counter***

Get the number of remaining movements of the wings

Return an integer (0..255)

Example:
```
>>> var = tux.status.wings_counter()
```

***1.9.30) Methode : tux.status.wings_motor***

Get the last state of "motor for wings" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.wings_motor()

***1.9.31) Methode : tux.status.wings_forward***

Get the last state of "wings motor forward" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.wings_forward()

***1.9.32) Methode : tux.status.wings_bt***

Get the last state of "wings position switch" status

Return 1 for on and 0 for off

Example:
>>> var = tux.status.wings_bt()

***1.9.33) Methode : tux.status.get_wings_position_switch***

Deprecated : see 'tux.status.wings_bt'

***1.9.34) Methode : tux.status.to_string***

Convert the current raw statuses to an explicit string

Parameters:
"frame" as tuple of char : raw statuses

Return a string

Example:
>>> print tux.status.to_string()

**1.10) Object : tux.sys (tuxapi_class.TUXsys)**

Class which manages the system functions

***1.10.1) Methode : tux.sys.add_time_event***

Add a time event in the time event handler

Parameters:
"cmd_type" as number : Command type (CT_SHELL|CT_FUNCTION)
"cmd" as string : Command to execute
"year" as integer : (ex : 2006) (9999 : parameter ignored)
"month" as integer : (ex : 12) (99 : parameter ignored)
"day" as integer : (ex : 23) (99 : parameter ignored)
"hour" as integer : (ex : 08) (99 : parameter ignored)
"minute" as integer : (ex : 55) (99 : parameter ignored)
"second" as integer : (ex : 30) (99 : parameter ignored)

Example:
>>> tux.sys.add_time_event(CT_SHELL,'xmms',9999,99,99,8,5,0)

***1.10.2) Methode : tux.sys.clear_time_events***

Clear the time events of the time event handle

Example:
>>> tux.sys.clear_time_events()

### 1.10.3) Methode : tux.sys.delayed_function

To execute a function with a delay

Parameters:
"function" as pointer of function : function to execute
"delay" as float : time to wait before executing the function. In seconds

Example:
```
>>> def test():
...     print "hello world"
...
>>> tux.sys.delayed_function(test,10)
```

### 1.10.4) Methode : tux.sys.delete_time_event

Delete a time event from the time event handler

Parameters:
"cmd_type" as number : Command type (CT_SHELL|CT_FUNCTION)
"cmd" as string : Command to execute
"year" as integer : (ex : 2006) (9999 : parameter ignored)
"month" as integer : (ex : 12) (99 : parameter ignored)
"day" as integer : (ex : 23) (99 : parameter ignored)
"hour" as integer : (ex : 08) (99 : parameter ignored)
"minute" as integer : (ex : 55) (99 : parameter ignored)
"second" as integer : (ex : 30) (99 : parameter ignored)

Example:
```
>>> tux.sys.delete_time_event(CT_SHELL,'xmms',9999,99,99,8,5,0)
```

### 1.10.5) Methode : tux.sys.looped_function

Looping on a function with a delay

Parameters:
"function" as pointer of function: function to execute
"delay" as float : time to wait between 2 executions of the function. In seconds

Example:
```
>>> def test():
...     print "hello world"
...     return True
...
>>> tux.sys.looped_function(test,10)
```

Comment:
While the return of the function is true, the loop remains active

### 1.10.6) Methode : tux.sys.shell

Execute a shell command

Parameters:
"command" as string : Shell command

Example:
```
>>> tux.sys.shell('ls -al')
```

### 1.10.7) Methode : tux.sys.shell_free

Execute a shell command in free mode

```
        Parameters:
        "command" as string : Shell command

        Example:
        >>> tux.sys.shell_free('ls -al')
```

#### ***1.10.8) Methode : tux.sys.time***

```
    Get the current time in seconds

        Return an integer

        Example:
        >>> var=tux.sys.time()
```

#### ***1.10.9) Methode : tux.sys.wait***

```
    Wait a time in seconds

        Parameters:
        "seconds" as float : Time to wait in seconds

        Example:
        >>> tux.sys.wait(2.4)
```

## 1.11) Object : tux.tts (tuxapi_class.TUXtts)

```
Class which manages the text to speech

    Global variables of this class:
    "connected" as boolean      : State of the connection to tuxttsd
    "print_status" as boolean   : Allow to print the raw statuses
    "sound_on" as boolean       : Speaking state of the tuxttsd
    "on_connected" as pof       : Event on tuxttsd connected
    "on_disconnected" as pof    : Event on tuxttsd disconnected
    "on_sound_on" as pof        : Event on tts speaking on
    "on_sound_off" as pof       : Event on tts speaking off
    "on_voice_list" as pof      : Event on new authorized voices list
    "authorized_voices_list" as list of string : List of authorized voices
    (pof = pointer of function)

    Example of associating a function to an event:
    >>> def my_function():
    >>>     tux.cmd.mouth_open()
    >>> tux.tts.on_sound_on=my_function
```

#### ***1.11.1) Methode : tux.tts.auto_connect***

```
    Allow to connect the api to tuxttsd automatically

        Parameters:
        "value" as boolean      : turn on/off the auto_connect mode
        "address" as string     : Tcp/IP Host address
                                  (default = 'localhost')
        "port" as integer       : Tcp/IP Port number
                                  (default = 5500)

        Examples:
        >>> tux.tts.auto_connect(True)
        >>> tux.tts.auto_connect(True,'192.168.0.1')
        >>> tux.tts.auto_connect(True,'192.168.0.1',5500)
```

#### ***1.11.2) Methode : tux.tts.connect***

```
    Connect tts object to tuxttsd

       Parameters:
```

```
"port" as integer     : Tcp/IP Port number
                        (default = 5500)
"address" as string   : Tcp/IP Host address
                        (default = 'localhost')

Examples:
>>> tux.tts.connect()
>>> tux.tts.connect('192.168.0.1')
>>> tux.tts.connect('localhost',5500)

Comment:
The variable "tux.tts.connected" contains the result of this method
```

### 1.11.3) Methode : tux.tts.disconnect

Disconnect tts object from tuxttsd

```
Example:
>>> tux.tts.disconnect()
```

### 1.11.4) Methode : tux.tts.kill_daemon

Kill the tuxttsd

```
Example:
>>> tux.tts.kill_daemon()
```

### 1.11.5) Methode : tux.tts.pause

Pause the sound

```
Example:
>>> tux.tts.pause()
```

### 1.11.6) Methode : tux.tts.play

Play the sound if it's in "pause" state

```
Example:
>>> tux.tts.play()
```

### 1.11.7) Methode : tux.tts.select_voice

Select a speaker voice

```
Parameters:
"speaker" as integer    : speaker id (SPK_FR_MALE|SPK_FR_FEMALE|
                                      SPK_US_MALE|SPK_US_FEMALE)
"pitch" as integer      : raised pitch in % (100..330)

Example:
>>> tux.tts.select_voice(SPK_FR_MALE,100)
```

### 1.11.8) Methode : tux.tts.speak

Speak a text with the acapela text to speech engine

```
Parameters:
"text" as string    : text to read

Example:
>>> tux.tts.speak('My name is tux! tux droid !')
```

### 1.11.9) Methode : tux.tts.speak_free

Speak a text with the acapela text to speech engine in free mode

```
Parameters:
"text" as string    : text to read

Example:
>>> tux.tts.speak_free('My name is tux! tux droid !')
```

#### *1.11.10) Methode : tux.tts.stop*

Stop the sound

```
Example:
>>> tux.tts.stop()
```

### 1.12) Object : tux.wav (tuxapi_class.TUXwav)

Class which manages the wav functions.

#### *1.12.1) Methode : tux.wav.play*

Play a wave file with tuxttsd daemon.

```
Parameters:
"wav_path" as string    : Path of the wave file
"begin" as float        : Starting index in seconds(Optional)
"end" as float          : Ending index in seconds(Optional)

Exemple:
>>> tux.wav.play('/home/tux/test.wav')
>>> tux.wav.play('/home/tux/test.wav', 0., 5.5)
```

#### *1.12.2) Methode : tux.wav.play_free*

Play a wave file with tuxttsd daemon in free mode.

```
Parameters:
"wav_path" as string    : Path of the wave file
"begin" as float        : Starting index in seconds(Optional)
"end" as float          : Ending index in seconds(Optional)

Exemple:
>>> tux.wav.play_free('/home/tux/test.wav')
>>> tux.wav.play_free('/home/tux/test.wav', 0., 5.5)
```

#### *1.12.3) Methode : tux.wav.pause*

Pause the current played wave file.

```
Exemple:
>>> tux.wav.pause()
```

#### *1.12.4) Methode : tux.wav.stop*

Stop the current played wave file.

```
Exemple:
>>> tux.wav.stop()
```

#### *1.12.5) Methode : tux.wav._continue*

Continue the playing of a paused wave file.

```
Exemple:
>>> tux.wav._continue()
```

#### *1.12.6) Methode : tux.wav.get_duration*

Get the duration of a wave file.

```
Parameters:
"wav_path" as string    : Path of the wave file

Returns:
The time duration in seconds as float.

Exemple:
>>> print tux.wav.get_duration('/home/tux/test.wav')
```