

JSK Enshu robot_programming Euslisp Manual リファレンスマニュアル

平成 26 年 11 月 29 日

目 次

第I部

robot_programming Models

turtlebot-with-sensors-robot

[クラス]

```

:super    turtlebot-robot
:slots    sensors bumper-sensors

```

:bumper-sensors *nil*

[メソッド]

Returns bumper sensors.

:bumper-sensor *sensor-name*

[メソッド]

Returns bumper sensor of given name

:init *&rest args &key (name turtlebot-with-sensors-robot)*

[メソッド]

:simulate *objs*

[メソッド]

dxl-7dof-arm-robot

[クラス]

```

:super    robot-model
:slots    jc0 jc1 jc2 jc3 jc4 jc5 jc6

```

:arm *&rest args*

[メソッド]

Accessor for arm methods.

:reset-pose *nil*

[メソッド]

Reset pose.

:reset-pose2 *nil*

[メソッド]

Reset pose2.

:tuckarm-pose *nil*

[メソッド]

Folding arm pose.

:tuckarm-pose2 *nil*

[メソッド]

Folding arm pose2.

:inverse-kinematics *target-coords &rest args &key (link-list) (move-target) (stop 300) (use-base nil) (start-coords (send self :copy-worldcoords)) (thre (cond ((atom target-coords) 10) (t (make-list (length target-coords) :initial-element 10)))) (rthre (cond ((atom target-coords) (deg2rad 5)) (t (make-list (length target-coords) :initial-element (deg2rad 5))))) (base-range (list :min #f(-30.0 -30.0) :max #f(30.0 30.0))) &allow-other-keys*

[メソッド]

Inverse kinematics method for arm robot.

:init *&rest args &key (name dxl-7dof-arm-robot)*

[メソッド]

:make-root-link *nil*

[メソッド]

:make-arm-links *nil*

[メソッド]

:arm_joint1 *nil*

[メソッド]

:arm_joint2 <i>nil</i>	[メソッド]
:arm_joint3 <i>nil</i>	[メソッド]
:arm_joint4 <i>nil</i>	[メソッド]
:arm_joint5 <i>nil</i>	[メソッド]
:arm_joint6 <i>nil</i>	[メソッド]
:arm_joint7 <i>nil</i>	[メソッド]

dxl-armed-turtlebot-robot [クラス]

```

:super    turtlebot-with-sensors-robot
:slots    arm-robot arm-base-fixed-joint

```

```

:init &rest args &key (name dxl-armed-turtlebot-robot) (arm-origin-coords (make-coords :pos (float-vector 85.725 9.525 402) :rpy (list 0 0 pi))) [メソッド]
:method-copying substr &optional (use-args nil) [メソッド]
:arm &rest args [メソッド]

```

turtlebot-with-sensors *nil* [関数]

Generation function for turtlebot-with-sensors-robot.

dxl-7dof-arm *nil* [関数]

Generation function for dxl-7dof-arm-robot.

dxl-armed-turtlebot *nil* [関数]

Generation function for dxl-armed-turtlebot-robot.

make-dynamixel-ax-12a-motor-body *nil* [関数]

```

nil  make-dynamixel-ax-12a-frame1-body nil [関数]
nil  make-dynamixel-ax-12a-frame2-body nil [関数]
nil  make-dxl-7dof-arm-gripper-body nil [関数]
nil  make-dxl-7dof-arm-base-body nil [関数]
nil  make-dynamixel-ax-12a-motor-unit-bodyset &key (use-frame1 (list :bottom :left)) [関数]
nil  make-dxl-7dof-arm-root-link nil [関数]
nil  make-dxl-7dof-arm-link1 nil [関数]
nil  make-dxl-7dof-arm-link2 nil [関数]
nil  make-dxl-7dof-arm-link3 nil [関数]
nil  make-dxl-7dof-arm-link4 nil [関数]
nil  make-dxl-7dof-arm-link5 nil [関数]
nil  make-dxl-7dof-arm-link6 nil [関数]
nil  make-dxl-7dof-arm-link7 nil [関数]

```

nil

第II部

robot_programming Robot Interface

turtlebot-interface [クラス]

```

:super    robot-interface
:slots    nil

```

:bumper-vector <i>nil</i>	[メソッド]
Get bumper value vector.	
:button-vector <i>nil</i>	[メソッド]
Get button value vector.	
:wheel-drop-vector <i>nil</i>	[メソッド]
Get wheel drop sensor vector.	
:cliff-vector <i>nil</i>	[メソッド]
Get cliff sensor vector.	
:cliff-bottom-vector <i>nil</i>	[メソッド]
Get cliff bottom vector.	
:imucoords <i>nil</i>	[メソッド]
Get imucoords.	
:power-system-vector <i>nil</i>	[メソッド]
Get power system vector.	
:publish-led <i>id value</i>	[メソッド]
Publish topic to turn on/off LEG. id should be 1-2. Value should be :black, :green, :orange, and :red.	
:publish-sound <i>value</i>	[メソッド]
Publish topic to turn on sound. value should be :on, :off, :recharge, :button, :error, :cleaningstart, and :cleaningend.	
:initialize-turtlebot-ros <i>nil</i>	[メソッド]
:kobuki-bumper-states-callback <i>msg</i>	[メソッド]
:kobuki-button-states-callback <i>msg</i>	[メソッド]
:kobuki-power-system-states-callback <i>msg</i>	[メソッド]
:kobuki-wheel-drop-states-callback <i>msg</i>	[メソッド]
:kobuki-cliff-states-callback <i>msg</i>	[メソッド]
:kobuki-imu-states-callback <i>msg</i>	[メソッド]
:laptop-charge-callback <i>msg</i>	[メソッド]
:def-vector-value <i>key (simulate-func #'(lambda nil (instantiate float-vector 3))) (raw-data-name) (vector-length 3) (state-name :state) (value-name)</i> ソッド]	[メ
:raw-bumper-data <i>nil</i>	[メソッド]
:raw-button-data <i>nil</i>	[メソッド]
:raw-wheel-drop-data <i>nil</i>	[メソッド]
:raw-cliff-data <i>nil</i>	[メソッド]
:raw-imu-data <i>nil</i>	[メソッド]
:imurot <i>nil</i>	[メソッド]
:update-robot-state <i>rest args</i>	[メソッド]
:go-stop <i>Optional (force-stop t)</i>	[メソッド]
:move-to <i>coords key (retry 10) (frame-id /world) (wait-for-server-timeout 5)</i>	[メソッド]
:go-pos <i>x y Optional (d 0)</i>	[メソッド]
:go-velocity <i>x y d Optional (msec 1000) key (stop t) (wait)</i>	[メソッド]
:init <i>rest args</i>	[メソッド]
:add-controller <i>rest args</i>	[メソッド]

dxl-7dof-arm-interface

[クラス]

```

:super      robot-interface
:slots      nil

```

:set-compliance-slope *id slope*

[メソッド]

Set compliance slope for one joint. id should be 1-7. slope is 32 by default.

:compliance-slope-vector *av*

[メソッド]

Set compliance slope vector for all joints. #f(32 32 32 32 32 32 32) by default.

:set-torque-limit *id torque-limit*

[メソッド]

Set torque limit for one joint. id should be 1-7. torque-limit should be within [0, 1].

:torque-enable *id torque-enable*

[メソッド]

Configure joint torque mode for one joint. id should be 1-7. If torque-enable is t, move to torque control mode, otherwise, move to joint position mode.

:servo-on *id*

[メソッド]

Servo On for one joint. id should be 1-7.

:servo-off *id*

[メソッド]

Servo Off for one joint. id should be 1-7.

:servo-on-all *nil*

[メソッド]

Servo On for all joints.

:servo-off-all *nil*

[メソッド]

Servo Off for all joints.

:start-grasp *%optional (arm :arm) %key ((:gain g) 0.5) ((:objects objs) objects)*

[メソッド]

Start grasp mode.

:stop-grasp *%optional (arm :arm) %key (wait nil)*

[メソッド]

Stop grasp mode.

:initialize-arm-robot-ros *nil*

[メソッド]

:dynamixel-motor-states-callback *msg*

[メソッド]

:fullbody-controller *nil*

[メソッド]

:gripper-controller *nil*

[メソッド]

:default-controller *nil*

[メソッド]

:servo-on-off *id on/off*

[メソッド]

:init *%rest args*

[メソッド]

dxl-armed-turtlebot-robot

[クラス]

```

:super      turtlebot-with-sensors-robot
:slots      arm-robot arm-base-fixed-joint

```

:init *%rest args %key (name dxl-armed-turtlebot-robot) (arm-origin-coords (make-coords :pos (float-vector 85.725 9.525 402) :rpy (list 0 0 pi)))*

[メソッド]

:method-copying *substr %optional (use-args nil)*

[メソッド]

:arm *%rest args*

[メソッド]

dxl-armed-turtlebot-interface

[クラス]

:super **robot-interface**
:slots nil

:set-compliance-slope *id slope*

[メソッド]

Set compliance slope for one joint. id should be 1-7. slope is 32 by default.

:compliance-slope-vector *av*

[メソッド]

Set compliance slope vector for all joints. #f(32 32 32 32 32 32 32) by default.

:set-torque-limit *id torque-limit*

[メソッド]

Set torque limit for one joint. id should be 1-7. torque-limit should be within [0, 1].

:torque-enable *id torque-enable*

[メソッド]

Configure joint torque mode for one joint. id should be 1-7. If torque-enable is t, move to torque control mode, otherwise, move to joint position mode.

:servo-on *id*

[メソッド]

Servo On for one joint. id should be 1-7.

:servo-off *id*

[メソッド]

Servo Off for one joint. id should be 1-7.

:servo-on-all *nil*

[メソッド]

Servo On for all joints.

:servo-off-all *nil*

[メソッド]

Servo Off for all joints.

:start-grasp *&optional (arm :arm) &key (:gain g) 0.5) (:objects objs) objects)*

[メソッド]

Start grasp mode.

:stop-grasp *&optional (arm :arm) &key (wait nil)*

[メソッド]

Stop grasp mode.

:bumper-vector *nil*

[メソッド]

Get bumper value vector.

:button-vector *nil*

[メソッド]

Get button value vector.

:wheel-drop-vector *nil*

[メソッド]

Get wheel drop sensor vector.

:cliff-vector *nil*

[メソッド]

Get cliff sensor vector.

:cliff-bottom-vector *nil*

[メソッド]

Get cliff bottom vector.

:imucoords *nil*

[メソッド]

Get imucoords.

:power-system-vector *nil* [メソッド]

Get power system vector.

:publish-led *id value* [メソッド]

Publish topic to turn on/off LEG. id should be 1-2. Value should be :black, :green, :orange, and :red.

:publish-sound *value* [メソッド]

Publish topic to turn on sound. value should be :on, :off, :recharge, :button, :error, :cleaningstart, and :cleaningend.

:initialize-arm-robot-ros *nil* [メソッド]

:dynamixel-motor-states-callback *msg* [メソッド]

:fullbody-controller *nil* [メソッド]

:gripper-controller *nil* [メソッド]

:default-controller *nil* [メソッド]

:servo-on-off *id on/off* [メソッド]

:initialize-turtlebot-ros *nil* [メソッド]

:kobuki-bumper-states-callback *msg* [メソッド]

:kobuki-button-states-callback *msg* [メソッド]

:kobuki-power-system-states-callback *msg* [メソッド]

:kobuki-wheel-drop-states-callback *msg* [メソッド]

:kobuki-cliff-states-callback *msg* [メソッド]

:kobuki-imu-states-callback *msg* [メソッド]

:laptop-charge-callback *msg* [メソッド]

:def-vector-value *&key (simulate-func #'(lambda (nil (instantiate float-vector 3))) (raw-data-name) (vector-length 3) (state-name :state) (value-name))* [メソッド]

:raw-bumper-data *nil* [メソッド]

:raw-button-data *nil* [メソッド]

:raw-wheel-drop-data *nil* [メソッド]

:raw-cliff-data *nil* [メソッド]

:raw-imu-data *nil* [メソッド]

:imu-rot *nil* [メソッド]

:update-robot-state *&rest args* [メソッド]

:go-stop *&optional (force-stop t)* [メソッド]

:move-to *coords &key (retry 10) (frame-id /world) (wait-for-server-timeout 5)* [メソッド]

:go-pos *x y &optional (d 0)* [メソッド]

:go-velocity *x y d &optional (msec 1000) &key (stop t) (wait)* [メソッド]

:init *&rest args* [メソッド]

turtlebot-init *&key (objects)* [関数]

Initialization function for *ri*and *turtlebot*.

dxl-7dof-arm-init *nil* [関数]

Initialization function for *ri*and *dxl-7dof-arm*.

dxl-armed-turtlebot *nil* [関数]

Generation function for dxl-armed-turtlebot-robot.

dxl-armed-turtlebot-init *&key (objects)* [関数]

Initialization function for *ri*and *dxl-armed-turtlebot*.

get-method-list-for-turtlebot-interface *nil* [関数]

nil	<code>get-method-list-for-dxl-7dof-arm-interface</code>	<i>nil</i>	[関数]
nil	<code>get-method-list-for-turtlebot-interface</code>	<i>nil</i>	[関数]
nil	<code>get-method-list-for-dxl-7dof-arm-interface</code>	<i>nil</i>	[関数]
nil			