Quick search

Search

Docs » 1 TALOS Software manual

1 TALOS Software manual

Thank you for choosing PAL Robotics. This User Manual contains information related to the TALOS robot developed by PAL Robotics. Every effort has been made to ensure the accuracy of this document. All the instructions must be strictly followed for proper product usage. The software and hardware described in this document may be used or replicated only in accordance with the terms of the license pertaining to the software or hardware.

1.1 GENERAL INFORMATION

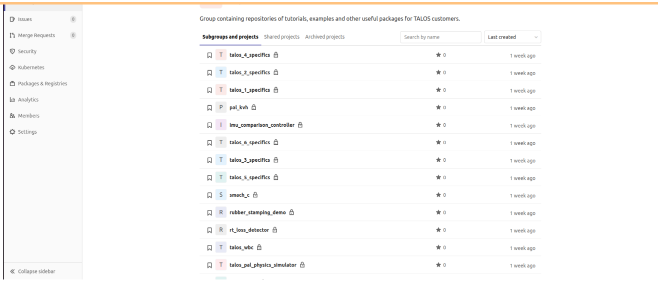
This User Manual is intended to be used as an index for the user, to access the most common topics and usages of the TALOS robot. It doesn't pretend to substitute or override the documentation presented in the handbook or other manuals.

2 PAL repositories

The following table contains a brief description of the packages available at the GitLab repository.

Quick search

Search



Package name	Description
talos_X_specifics	Specific parameters fo TALOS X
pal_kvh	KVH Imu driver by PAL
imu_comparison_controller	Tool to compare two IMUS (ex. KVH vs Orientus)
smach_c	Integration of ROS smach ir C++ to write state machines
rubber_stamping_demo	WBC Kinematic demo of the robot stamping
rt_loss_detector	Tool to check if code is real

PAL ROBOTICS

Quick search

Search

	time safe
talos_wbc	Kinematic WBC Parameters + launch files + documentatio
talos_pal_physics_simulator	PAL physics simulator parameters + documentatio
talos_robot	URDF description + robot parameters
pal_deployer_cfg	Deployer parameters (ex. zero offsets)
pal_locomotion_actions	DCM controller actions + documentatio
talos_pal_locomotion	DCM controller parameters + Dynamic WB(stack + documentatio

Quick search

Search

pal_wbc_utils	Examples + documentatio for Kinematic WBC
pal_walking_tutorials	Examples + documentatio for walking in position
pal_optimization_nl_tutorials	Examples + documentatio for PAL optimization
pal_locomotion_tutorials	Examples + documentatio for walking in torque control
wbc_grasp_demo	WBC Kinematic / Dynamic demo of the robot grasping
pal_base_ros_controller_tutorials	Examples + documentatio of PAL ROS control wrapper
talos_uat_tests	UAT tests to validate the robot after an

PAL ROBOTICS

Quick search

Search

	upgrade
TALOS Handbook	TALOS Latest Handbook

Note

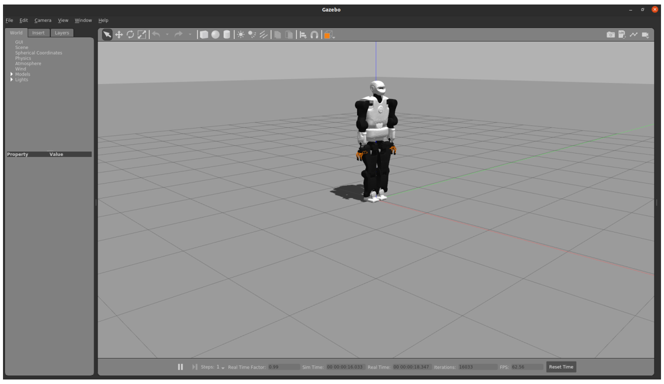
It is recommended to read carefully the **README's** of the above packages to better understand the PAL software architecture.

3 Gazebo simulation

Gazebo is an open-source 3D robotics simulator.

To start TALOS using the Gazebo simulator run:

```
roslaunch talos_gazebo talos_gazebo.launch
```



By default it loads the *empty_world*. It is possible to load other environments described in the **pal_gazebo_worlds** package.

```
roslaunch talos_gazebo talos_gazebo.launch
world:=small_office
```

Gazebo tries to replicate through plugins the same behaviour and interface that the one in the real robot. This means that all control applications will be embedded and executed inside the simulation. To

Quick search

Search


```
roslaunch talos_gazebo talos_gazebo.launch debug:=true
```

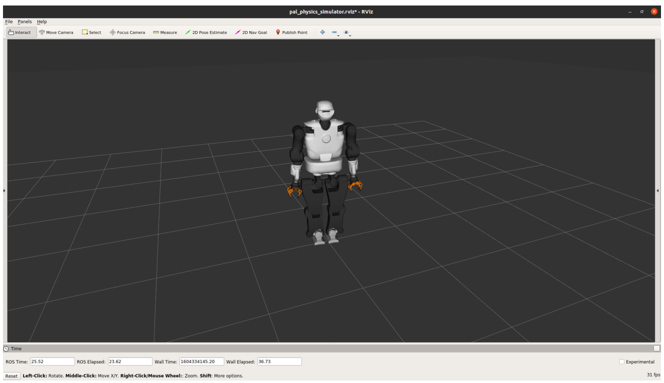
The *default controllers* needs to be launched as follows in Gazebo.

```
roslaunch talos_controller_configuration_gazebo
default_controllers.launch
```

Warning
Before deploying and executing any kind of software in the robot, it is mandatory to validate it in simulation.

4 PAL Physics simulator

The PAL Physics simulator is a custom simulator implemented by PAL Robotics that emulates more realistic physics than Gazebo. It is meant to be used as the main simulator when developing applications in Torque control.



It is possible to start the simulation with two different profiles:

1. Ideal actuators

2. Gaussian noise in the actuators

The **README** of the talos_pal_physics_simulator contains a deeper explanation on how to:

Quick search

Search

- Load a simulated environment.
- Interact with the robot applying an external force.
- How to launch the controllers in each simulator profile.
- Understand the different robot model available.
- How to start the simulation using *GDB* or *Valgrind*.

4.1 Simulation with ideal actuation

This simulates ideal actuators with **zero noise**. The physics are quite similar to the ones of Gazebo.

```
roslaunch talos_pal_physics_simulator
talos_pal_physics_simulator.launch
```

Start torque control (*Direct torque control*):

```
roslaunch talos_pal_locomotion
talos_dcm_walking_controller.launch
```

4.2 Simulation with gaussian noise

This simulates ideal actuators with **Gaussian noise**. The physics are quite close to the real robot.

```
roslaunch talos_pal_physics_simulator
talos_pal_physics_simulator_with_actuators.launch
```

Start torque control using (*Inertia shapping effort torque control*):

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch simulation:=true
```

Warning

Quick search

Search

Any new application in torque control should be previously tested and validated in the PAL physics simulator with Gaussian noise.

5 Initialization script

The initialization script is used for safety to check that the readings from all sensors are valid.

Note
It should be triggered before starting any controller with the robot hanging in air. Wait until the robots says that initialization has finished successfully.

It can be triggered from the WebCommander (Tab Controllers) or from the terminal:

```
rosrun talos_controller_configuration
talos_initialisation.py --yes
```

The script:

- Puts the robot in the default position
- Check that the torques are correct (under 5 N/m deviation)
- Reset the 6 axis F/T sensors of the feet.

For more information check the handbook section.

Warning
If the initialization script fails, check in the logs for the cause. If the reason is a torque sensor deviation, please run the torque offset calibration again.

PAL ROBOTICS

Quick search

Search

6 Position controllers

Talos uses ROS control as a control interface. It has integrated the default **JointTrajectoryControllers** as well as **MoveIt!** for planning.

The available JointTrajectoryControllers are:

- left_arm_controller
- right_arm_controller
- head_controller
- torso_controller
- left_leg_controller
- right_leg_controller
- left_gripper_controller
- right_gripper_controller

6.1 Controller manager

The controller manager interface allow us to:

- List loaded and / or running controllers

```
rosservice call /controller_manager/list_controllers
```

- Start or stop controllers

```
rosservice call /controller_manager/switch_controllers
```

- Load a controller if the parameters are loaded in the specific namespace

```
rosservice call /controller_manager/load_controller
```

- Unload a controller

PAL ROBOTICS

Quick search

Search


```
rosservice call /controller_manager/unload_controller
```

6.2 Default controllers

The default controllers include all the **upperbody JointTrajectory controllers** and the **walking controller**. It allows to run any kind of motion that doesn't include the legs as a resource.

```
roslaunch talos_controller_configuration
default_controllers.launch
```

In Gazebo:

```
roslaunch talos_controller_configuration_gazebo
default_controllers.launch
```

Check the walking section in the handbook to see how to make the robot walk.

6.3 Full body position controllers

To start all the **JointTrajectory** controllers run:

```
roslaunch talos_controller_configuration
position_controllers.launch
```

6.4 Play motion

Those controllers allow to send trajectories in joint space as predefined motions.

```
rosrun play_motion run_motion NAME_OF_THE_MOTION
```

Or using the GUI:

```
rosrun actionlib axclient.py /play_motion
```

Note

PAL ROBOTICS

Quick search

Search

> For the motions of the legs, the argument *skip_planning* needs to be set to True. Always validate in simulation first.

7 Walking controller

The walking controller is a ROS controller developed at PAL to make the robot walk in position using the **ZMP** based algorithm.

The controller is launched in the default controllers.

Z default kontrolerji

```
roslaunch talos_controller_configuration
default_controllers.launch
```

In the Gazebo simulator the command is different.

```
roslaunch talos_controller_configuration_gazebo
default_controllers.launch
```

Once the controller is started there are different ways to make the robot goal:

- Using the joystick **R1 + analog button**.
- Publishing in the topic `/walking_controller/cmd_vel`
- Using the service `/walking_controller/do_step` to perform a single step. The distances in the message are related to the stance foot.
- Using the service `/walking_controller/walk_steps` to perform n steps forward or backward.

For more information, check the documentation in the section walking controller of the handbook or check the examples in pal_walking_tutorials

7.1 Walking offsets calibration

Baje so iz Pala Kristini rekl

Quick search

Search

The walking controller uses offsets for each legs joints in order to have an even force distribution between both legs during the double support. The `init_offset_controller` allow for tuning the walking offsets for the legs joints. This procedure is requested for achieving a better walk. The controller uses the information provided by the force and torque sensors of the ankles for moving slightly the legs joints in order to reduce the weight difference between the two feets and for reducing as much as possible the torques along axis y and z of every foot.

7.1.1 Load controller

The `init_offset_controller` is taking ownership of the lower body joints only, so `joint_trajectory_controllers` are needed for the upper body. With the robot in the air, run the following commands:

```
roslaunch talos_controller_configuration
upper_body_position_controllers.launch
```

```
roslaunch talos_controller_configuration
init_offset_controller.launch
```

```
rosrun play_motion run_motion walk_pose
```

7.1.2 Start calibration

Put the robot down and call the calibration service:

```
rosservice call /init_offset_controller/init_offsets
```

The robot will start moving sligthly and after a while it will stop. In order to check if the calibration has succeeded you could run `pal-log deployer cat` and inspect the output file. The corrected offsets should be printed in the log file with the corresponding output measurements of the feet force torque sensors.

Quick search

Search

7.1.3 Save the results

```
rosrun reemc_init_offset_controller offsets_saver
```

This will output a file `.pal/init_offset_controller/config/offsets.yaml` . This file should be copied in the folder``.pal/talos_controller_configuration/config/`` and replace the walking_specific_offsets.yaml file. If this folder doesn't exist, create it with the following command:

```
configure_specifics.sh -c
talos_controller_configuration
```

Then copy the file:

```
cp .pal/init_offset_controller/config/offsets.yaml
.pal/talos_controller_configuration/config
/walking_specific_offsets.yaml
```

8 Torque offset calibration

To naj bi se ob redni uporabi

This procedure is meant to estimate the offsets of the torque sensors. During this proces the robot performs some movements to estimate the torque sensor offsets.

> **Note**
> The robot needs to be on air and the position controllers as well as the dispatcher needs to be launched:

```
roslaunch talos_controller_configuration
position_controllers.launch
```

```
roslaunch trajectory_dispatcher
dispatcher_controller.launch
```

Quick search

Search

1. First perform the calibration movements. The argument type is used to specify leg or arm.

```
roslaunch pal_torque_offset_calibration_talos
pal_torque_offset_calibration.launch type:=leg
side:=both
```

2. Once finished the result can be seen here.

```
roslaunch pal_torque_offset_calibration_talos
compute_torque_offset_parameters.launch type:=leg
side:=both
```

Make a backup of the old params.

```
cp ~/.pal/talos_controller_configuration/config
/local_joint_control/actuator_parameters
/actuator_parameters_specific_params.yaml ~/.pal
/talos_controller_configuration/config
/local_joint_control/actuator_parameters
/actuator_parameters_specific_params.bkp
```

3. Reset the torque offsets with the last one computed.

```
roslaunch pal_torque_offset_calibration_talos
apply_torque_offsets.launch
```

> **Warning**
> Compare the results with the previous offsets. If the standard deviation or some torque offset has a big difference with respect to the previous offsets don’t start the DCM controller. Check how to validate torque control for a single joint in the Torque control documentation.

For more information check the torque offset calibration section in the handbook.

Quick search

Search

9 Torque control

TALOS is equipped with a single axis joint torque sensor in every joint, except the wrists, grippers and head. This allows to control the robot in torque control.

9.1 Joint Torque control

The torque control is performed at joint level.

To validate that the torque control is behaving properly for a single joint, start the **sine sweep controller**.

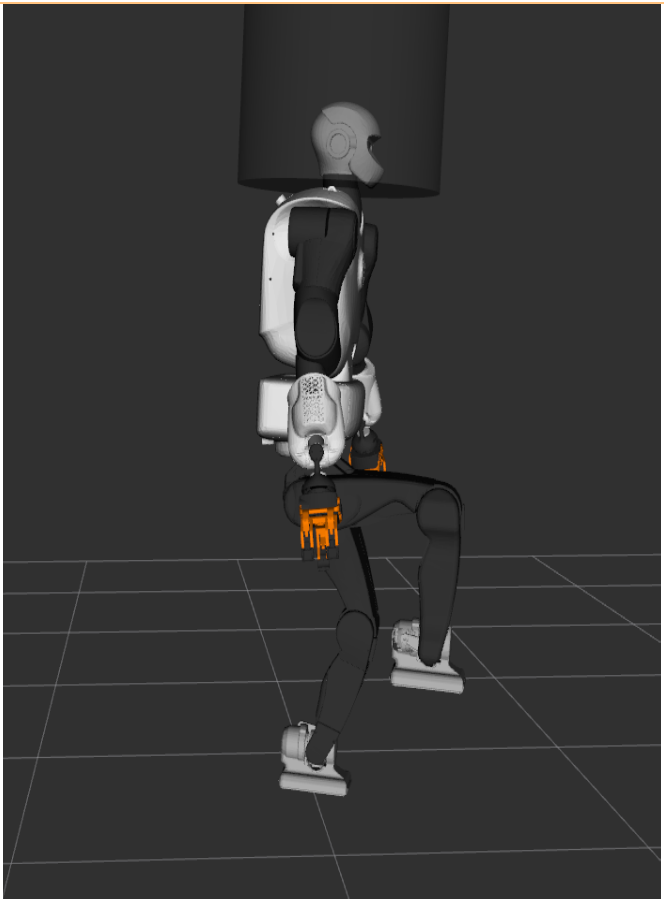
```
 roslaunch local_sine_sweep_torque_controller
sine_sweep_controller.launch
local_joint_control:=inertia_shaping_effort_analytic_dob_control
local_joint_control_pkg:=inertia_shaping_effort_control
joint_name:=leg_right_3_joint
controller_pkg_name:=talos_controller_configuration
```

Note
The previous command starts torque control for the joint specified. In the case of the arms and the torso, the **local_joint_control** type is different.

Warning
Before starting the controller the joint needs to be in zero gravity position, i.e the consequent links of the robot must be aligned with gravity. For example for the knee joint, the hip should be at 90° and the knee should also be at 90°.

Quick search

Search



Note

If torque control is working fine the joint doesn't move from it's position, and when exciting manually in both directions, it should behave as a pendulum with almost no friction and no acceleration in any direction.

Error

Please be cautious and always have the emergency stop during the proceess before validating torque control at joint level.

9.2 The DCM controller

The DCM is controller is the dynamic whole body controller implemented by PAL. This controllers uses

Quick search

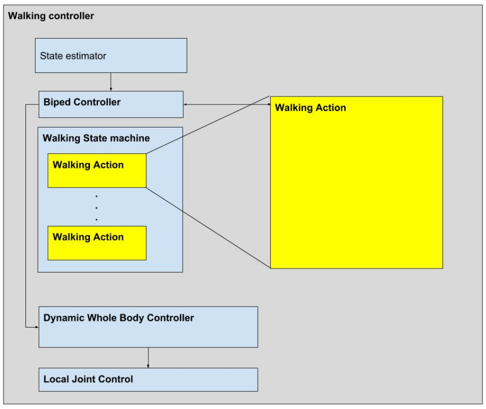
Search

as an input to the local torque control for each joint.

The default *WBC* stack, as well as the main parameters involved are defined in the talos_pal_locomotion package.

This controller contains a state machine internally where the user can push the desired walking actions. The walking actions available are:

- Balancing in double support (maintaining the DCM at the center of the support polygon)
- Balancing the DCM from one foot to the other
- Stand in one leg / Push down the leg
- Static walking (the DCM never goes out from the support polygon)
- Dynamic walking **(only validated in simulation)**
- Kinematic WBC wrapper (used for manipulation in dynamics ex. wbc_grasp_demo)



To start the DCM controller in **Gazebo** or **PAL physics simulator** with ideal actuation.

```
roslaunch talos_pal_locomotion talos_dcm_walking_controller.launch
```

To start the DCM controller in **PAL physics simulator with noise**.

Quick search

Search


```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch simulation:=true
```

To start the DCM controller on the **real robot**.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
```

To start the DCM controller with the **upper body controlled in position**.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
robot:=full_v2_fixed_upper_body
```

To start the DCM controller with **modified parameters** that overwrite the default ones.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
biped_parameters:=debris
```

For more information about:

- Safety in the DCM controller, or the parameters involved please check the talos_pal_locomotion package.
- Available actions, how to start / replace them, or how to implement a new action please check the pal_locomotion_actions package.
- Tutorials with examples on how to send steps to the robot, squat action, etc. please check the pal_locomotion_tutorials package.

10 Walking torque control

First start torque control on the robot with fixed upper body.

PAL ROBOTICS

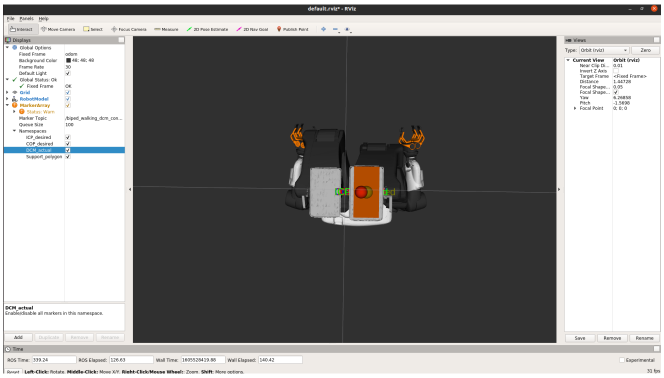
Quick search

Search

> but the performance may not be as optimal as fixed upper body.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
robot:=full_v2_fixed_upper_body
```

Then open RViz with the *MarkerArray* **(/biped_walking_dcm_controller/markers)** as shown in the picture.



> **Warning**
> The **DCM / ICP actual (red)** and the **COP desired (blue)** should be close to the **DCM / ICP desired (green)**, otherwise the robot might fall after few steps.

> **Warning**
> If the offset between the DCM / ICP actual and desired is big please run the torque offset calibration again.

> **Note**
> If at any point the DCM goes out of the support polygon (orange rectangle) the controller will shut down because the robot is unstable.

- Using the static walk action.

Quick search

Search


```
roslaunch pal_locomotion push_action.launch
action:=pal_locomotion::StaticWalkAction
```

Controlling with the joystick.

```
rosrun topic_tools relay
/walking_controller/cmd_vel
/biped_walking_dcm_controller/cmd_vel
```

Or sending predefined steps from a node of pal_locomotion_tutorials.

```
rosrun pal_locomotion_tutorials
static_steps_command
```

- Other option is to make the robot walk using the Stand one leg / Stand down leg actions. In this case, there is no need to start the static walk action.

```
rosrun pal_locomotion_tutorials
static_steps_execution_node -d 2.0 -u 2.0 -s
5 -x 0.15 -y 0.1
```

> **Note**
> To debug, after the torque offset calibration, or after change some parameters better use the previous node with *x=0* to ensure that the robot is able to do steps in place.

- To make the robot stand on one leg.

```
rosrun pal_locomotion_actions
push_stand_one_leg_action -d 3.0 -u 3.0 -z
0.15 -s LEFT
```

To put the robot back in double support.

```
rosrun pal_locomotion_actions
push_stand_down_leg_action -d 3.0 -u 3.0
```

PAL ROBOTICS

Quick search

Search

Changing *LEFT* by *RIGHT* makes the robot stand on the other leg.

> **Note**
> For debugging is also recommended to first stand the robot in one leg and put it down before start walking.

11 PAL Whole Body Control

The PAL Whole Body Control (WBC) is PAL's implementation of the **Stack of Tasks**. It includes a hierarchical quadratic solver able to accompish different tasks with different priorities.

11.1 Kinematic WBC

To start the Kinematic simulation of the WBC in which all the joints are controlled in position.

```
roslaunch talos_wbc talos_wbc_standalone.launch
```

The previous command starts the simulator and the Whole Body Kinematic controller.

To start the controller standalone inside the robot or the simulator.

```
roslaunch talos_wbc talos_wbc.launch
```

For printing the list of active tasks / constrains in the stack.

```
rosservice call /whole_body_kinematic_controller
/get_stack_description
```

Quick search

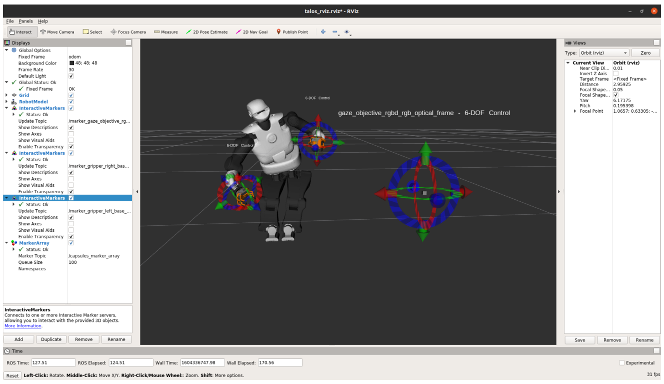
Search

To control both end effectors and the head of the robot using interactive markers execute.

```
roslaunch talos_wbc interactive_markers.launch
```

The previous command adds new tasks in the stack. If the user want to control by topic the *source_data* argument needs to be modified.

```
roslaunch talos_wbc interactive_markers.launch
source_data:=topic_reflexx_typeII
```



The pal_wbc_utils package contains:

- Brief explanation of the WBC and the default tasks loaded in the stack.
- Description of the available kinematic tasks.
- How to create a new task or constraint.
- How to create a specific stack.
- How to start or remove a task online.
- Code API and examples to push tasks online.
- Available references.

The pal_wbc_tutorials package explains how to create your own constrains. It contains an example on how to create a joint reference task.

The talos_wbc contains:

- The available kinematic stacks.

PAL ROBOTICS

Quick search

Search

- The parameters involved in the Kinematic WBC.
- The available launch files to start the controller and push the different tasks.

11.2 Dynamic WBC

The Dynamic WBC is based on torque control. The inverse dynamic constraint is added (among other constraints) in the stack to produce the optimal torques that minimize the desired objective.

To start the dynamic torque control in simulation (without noise):

```
roslaunch talos_pal_locomotion
talos_dcm_walking_controller.launch
```

To start it on the real robot:

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
```

Once the DCM controller is started, push a wrapper of the Kinematic WBC in the state machine of the DCM controller.

```
rosrun pal_locomotion StartWBCAction
```

Then is it possible to start the same demo of the interactive_markers or use any of the previous instructions by specifying the new namespace of the controller.

```
roslaunch talos_wbc interactive_markers.launch
ns:=/biped_walking_dcm_controller/wbc
```

```
rosservice call /biped_walking_dcm_controller
/wbc/get_stack_description
```

The talos_pal_locomotion package contains:

PAL ROBOTICS

Quick search

Search

- The default dynamic stack loaded when starting the DCM controller.
- The WBC parameters related with the default dynamic stack.

The talos_wbc package contains:

- Examples of dynamics stacks.
- Standalone dynamic WBC demos.

> **Warning**
> Except the wbc grasp demo, the rest of dynamic WBC demos are not mature enough at this moment. Control both end effectors using the interactive markers may trigger safety preventions due to the discontinuities in torque.

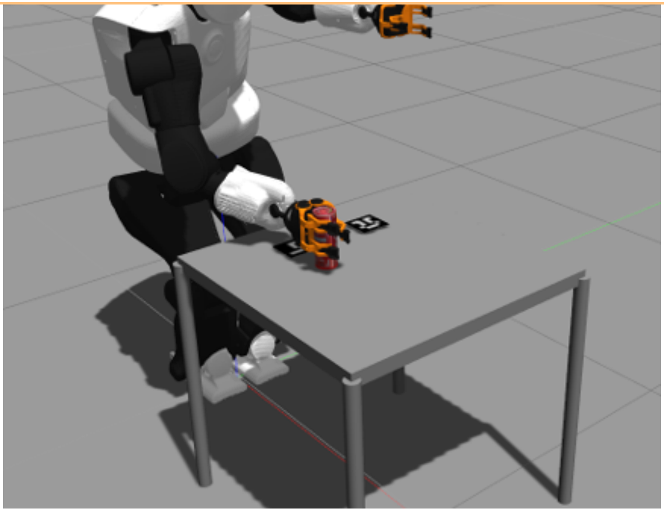
11.3 WBC grasp demo

This package is devoted to exemplify how to command the robot to do tasks such as to approach a table, and to grasp and place an object on this table, by using WBC.

The demo is ready to be executed, either in Kinematic WBC or in Dynamic WBC, on the real robot as well as in simulation.

Quick search

Search



12 Introspection controller

The introspection controller is used to publish all the statistics and information coming from the sensors, motors and control applications.

By default it's started when launching the DCM controller. Otherwise it needs to be launch as:

```
roslaunch introspection_controller introspection_controller.launch
```

To record a rosbag:

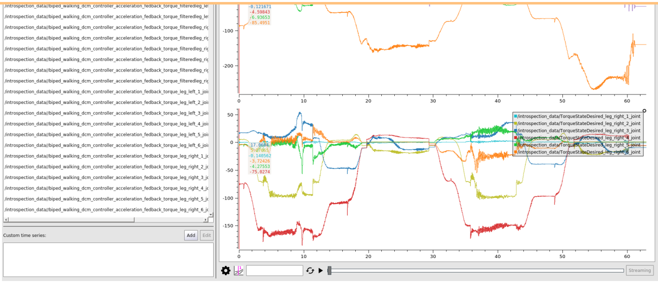
```
rosbag record /introspection_data/names /introspection_data/values -o NAME_OF_THE_BAG
```

Copy the rosbag in your development machine and visualize it with **PlotJuggler**:

```
rosrun plotjuggler PlotJuggler
```

Quick search

Search



> **Note**
> You can create your own layouts in **PlotJuggler**.

Please take a look at the introspection data section of the handbook or at the pal_base_ros_controller_tutorials to understand how to register new variables from your control applications.

13 Validation post upgrade

To properly validate the robot after an upgrade, we recommend the user to run the following UAT tests in this specific order:

13.1 WebCommander

Open the webcommander of the robot. Check in the diagnostics tab and the startup tab that everything is in green.

13.2 Position Control

- First start the initialization script with the robot hanging on air. Wait until the robot says that the initilization finished succesfully.

Quick search

Search

- Start the default controllers
- Run some motions from the WebCommander:
  - Nod
  - Bow
  - Push Walls
  - Close Both Grippers

13.3 Walking Controller

- Copy the *walking_validation.py* script from the talos_uat_tests package inside the robot.

```
scp walking_validation.py pal@talos-Xc:
```

- Execute it:

```
python walking_validation.py
```

> **Note**
> The robot will walk in all directions. Wait until the script finishes and the robot has stopped.

13.4 Whole Body control

- Copy the *wbc_validation.py* script from the talos_uat_tests package inside the robot.

```
scp wbc_validation.py pal@talos-Xc:
```

- Start the Kinematic WBC (before you need to stop the default controllers if they still active):

```
roslaunch talos_wbc talos_wbc.launch
```

- Proceed adding the tasks to control the robot in cartesian space. Open a new terminal connected

PAL ROBOTICS

Quick search

Search


```
roslaunch talos_wbc interactive_markers.launch
source_data:=topic_reflexx_typeII
```

- Execute the script that sends different cartesian poses to each end effector.

```
python wbc_validation.py
```

Warning
You will need to lower the crane for this experiment.

Note
Wait until the script has finished and the robot is back to the default position.

13.5 Torque Offset Calibration

- Run the torque offset calibration for the **legs** (see it on this manual).
- Check that the computed **offsets** and the **standard deviation** is not high for any sensor.
- Apply the offsets.

Repeat the whole process for the **arms**.

13.6 Torque Control

- Put the robot on air and start the initialization script.
- Lower the crane and start the DCM controller.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
```

- Disturb the robot. It should be compliant.

PAL ROBOTICS

Quick search

Search

- Stop the controller.

13.7 Walking Torque Control

- Create a workspace and deploy the pal_locomotion_tutorials package inside the robot.

```
rosrun pal_deploy deploy-py -p
"pal_locomotion_tutorials" talos-Xc
```

- Put the robot on air and start the initialization script.
- Lower the crane and start the DCM controller with fixed upper body.

```
roslaunch talos_pal_locomotion
robot_dcm_walking_controller.launch
robot:=full_v2_fixed_upper_body
```

- Check in RViz that the actual DCM converges to the desired DCM (this is explained in the walking torque control section).
- Execute six steps forward.

```
rosrun pal_locomotion_tutorials
static_steps_execution_node -d 2.0 -u 2.0 -s 6 -x 0.15
-y 0.1
```