Rock
the Robot Construction Kit
Cheat Sheet
(v1.0)

Basic usage

source env.sh
sources the enviromental variables required for your rock installation to be functional

autoproj
allows you to easily install and maintain the rock system.

autoproj update
updates your rock installation.
autoproj update <dir> updates the package located at *<dir>*. **See also** aup below

autoproj build
builds the packages in your rock installation.
autoproj build <dir> builds the package in *<dir>*, as well as its dependencies.
See also amake below

autoproj status
Shows the"difference between the local packages and the remote repositories.

amake *[package_name]*
does an autoproj build for the given package, or the package in the current directory if no name is given

aup *|package_name]*
does an autoproj update for the given package, or the package in the current directory if no name is given

acd
change directories between packages, with shortcuts
Example: $ acd s/ikf will go to slam/quater_ikf
$ acd s/o/ikf to slam/orogen/quater_ikf

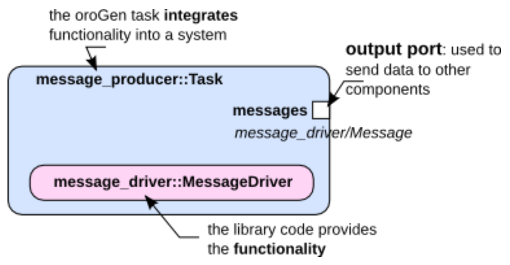
Create commands

rock-create-lib
command line program to generate the basic folder layout for a new library, to develop the core of your algorithms

Example: $ rock-create-lib *MyLibrary*

rock-create-orogen
command line program to generate the basic folder layout for a rock component.

Example: $ rock-create-orogen *MyComponent*


the oroGen task **integrates** functionality into a system
message_producer::Task
output port: used to send data to other components
messages
message_driver/Message
message_driver::MessageDriver
the library code provides the **functionality**

rock-create-vizkit-widget
command line program to generate the basic folder layout and a ready-to-use Qt Designer widget.

Example: $ rock-create-vizkit-widget *MyWidget*

rock-create-vizkit-plugin
command line program to generate the basic folder layout for a vizkit 3D plugin ready-to-use in rock.

Example: $ rock-create-vizkit-plugin *MyPlugin*

rock-create-bundle
command line program to generate a bundle rock package. bundle offers a functional view of your system instead only single components.

Example: $ rock-create-bundle *MyBundle*

Logging commands

pocolog
command-line tool allows you to easily look at log files

Usage: $ pocolog *logfile*

rock-convert
conversion between different types version of log files in case of rock base-types updates.

Usage: $ rock-convert *logfile*

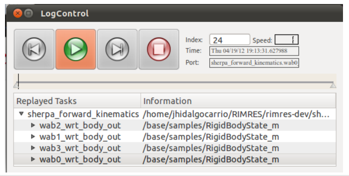
Play commands

rock-run
Starts a new oroGen component

Usage: $ rock-run *project::Task*

rock-replay
command-line tool for replaying the content of log files.

Usage: $ rock-replay *logfile*

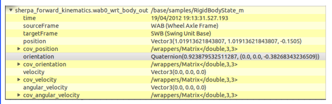


Index: 24
Time: Thu 04/19/12 19:13:31.627988
Port: sherpa_forward_kinematics.wab0

Replayed Tasks	Information
sherpa_forward_kinematics	/home/jhidalgocarrio/RIMRES/rimres-dev/sh...
wab2_wrt_body_out	/base/samples/RigidBodyState_m
wab1_wrt_body_out	/base/samples/RigidBodyState_m
wab3_wrt_body_out	/base/samples/RigidBodyState_m
wab0_wrt_body_out	/base/samples/RigidBodyState_m

rock-display
command line program for displaying the currently running components in your rock system.
As rock-replay, it allows multiple visualization manners of the data flow.

Example: $ rock-display



sherpa_forward_kinematics.wab0_wrt_body_out	/base/samples/RigidBodyState_m
time	19/04/2012 19:13:31.527.193
sourceFrame	WAB (Wheel Axle Frame)
targetFrame	SWB (Swing Unit Base)
position	Vector3(1.01913621843807, 1.01913621843807, -0.1505)
cov_position	/wrappers/Matrix</double,3,3>
orientation	Quaternion(0.923879532511287, (0.0, 0.0, -0.38268343236509))
cov_orientation	/wrappers/Matrix</double,3,3>
velocity	Vector3(0.0, 0.0, 0.0)
cov_velocity	/wrappers/Matrix</double,3,3>
angular_velocity	Vector3(0.0, 0.0, 0.0)
cov_angular_velocity	/wrappers/Matrix</double,3,3>