CRCL FANUC

Generated by Doxygen 1.8.6

Fri Mar 11 2016 14:35:31

Contents

Chapter 1

Fanuc Irmate200id Descarte Demo

Setting up ROS with Descartes

Copy repositories from https://github.com/ros-industrial: fanuc fanuc experimental motoman ros industrial core

Repositories from ROS-I Consortium https://github.com/ros-industrial-consortium github site: descartes descartes_tutorials

Assume catkin_ws has been setup)

```
cd ~/catkin_ws/src
git clone  https://github.com/ros-industrial/fanuc.git
```

How to build just one package using catkin_make?

A: catkin_make –pkg <my_package_name>

so

```
`catkin_make --pkg nist_fanuc
```

Is there a way to enable c++11 support for catkin packages?

```
http://catkin-tools.readthedocs.org/en/latest/cheat_sheet.html (kinda wrong)

set(CMAKE_CXX_FLAGS "-std=c++0x ${CMAKE_CXX_FLAGS}")
set(CMAKE_CXX_FLAGS "-Wwrite-strings ${CMAKE_CXX_FLAGS}")
```

how to resolve g++ warning with -std=c++11: 'auto_ptr' is deprecated [duplicate] https://gcc.gnu.org/bugzilla/show_bug.cgi?id=59325

Compiling Fanuc Demo with Debug Information

```
cd ~/catkin_ws
catkin_make -DCMAKE_BUILD_TYPE=Debug &> log.log
more log.log
```

Using IDE to Debug

After Compiling Fanuc Demo with Debug Information, Use netbeans to create binary C++ project and read binary file to debug (found at /home/michalos/catkin_ws/devel/lib/nist_fanuc/nist_fanuc)

Compiling Fanuc Demo with Debug and Error Information Redirected to log file

```
catkin_make -DCMAKE_BUILD_TYPE=Debug &> log.log
```

Location of exe directory to save ini and other runtime files

path "/home/michalos/catkin_ws/devel/lib/nist_fanuc/nist_fanuc"

Adding urdfdom and independent package

catkin_make_isolated

Installing Xerces c with Ubuntu

https://www.daniweb.com/hardware-and-software/linux-and-unix/threads/409769/ubuntu-11-10-xe
As far as I'm aware libxerces is the same as pretty much any other library in Debian based systems. It should be available in the repositories (the exact version will depend on which version of Ubuntu you're running).

You can use apt-get to install the packages for the library and the dev files. Then to use them in your C/C++ programs you simply #include the appropriate headers and link with the library when compiling/linking.

```
sudo apt-get update
apt-cache search libxerces
sudo apt-get install libxerces-c3.1 libxerces-c-dev
```

Need include file path CMakeLists.txt: include_directories(/usr/include/xercesc)

Link library in CMakeLists.txt: link_directories(/usr/lib/x86_64-linux-gnu/)

Need to link against libxerces.a in CMakeLists.txt: link_directories(/usr/lib/x86_64-linux-gnu/)

Installing CodeSynthesis XSD

http://www.codesynthesis.com/products/xsd/download.xhtml 1) Chose the linux deb install file that matches you computer (below 64 bit amd). 2) Download xsd_4.0.0-1_amd64.deb and it will say open with Ubuntu Software Center 3) Click to install, authenticate and add /usr/include/xsd/cxx/xml as include path.

Need include file path in CMakeLists.txt: include_directories(/usr/include/xsd/cxx/xml)

Running Netbeans without Environment setup properly

Unfortunately, you need to source ~/catkin_ws/devel/setup.bash to set up Unix shell enironment variables properly. Since it was not obvious how to run a bash shell before debugging the executable, it was decided a different approach must be used. (running a bash script with gdb was attempted). Instead environment variables were hard coded into the nist_fanuc program, with the use of the posix function "setenv" to set the environment variables (not perfect, but close enough between addition of new ROS packages).

To make it work, I did > env | grep indigo to find all the related ROS environment variables. From http://answers.ros.org/question/123581/rosrun-cannot-find-my-executable/ found: catkin_find uses the environment variable CMAKE_PREFIX_PATH to find catkin workspaces. These workspaces in turn are used in rosrun. ROS_PACKAGE_PATH is no longer enough.

This works, no claims of robustness. The following code was placed at the beginning of the nist_fanuc.cpp to set up the environment variables before connecting to the roscore (master). (This assumes that all the basic rviz, robot model, etc. has been launched.) In the code:

```
static void SetupRos(std::string  envname, std::string envval, int overwrite=1)
{
        setenv( envname.c_str(),envval.c_str(),1 );
}

       // Setup up environment for netbeans that allows ros utilities to work...
        SetupRos("ROS_MASTER_URI", "http://localhost:11311");
        SetupRos("ROS_DISTRO", "indigo");
        SetupRos("ROS_ROOT", "/opt/ros/indigo/share/ros");
        SetupRos("ROS_ETC_DIR", "/opt/ros/indigo/etc/ros");
        SetupRos("ROS_PACKAGE_PATH", "/home/michalos/catkin_ws/src:/opt/ros/indigo/share:/opt/ros/indigo/stacks")
        SetupRos("ROS_TEST_RESULTS_DIR", "/home/michalos/catkin_ws/build/test_results");
        SetupRos("ROS_ETC_DIR", "/opt/ros/indigo/etc/ros");
        SetupRos("OSLISP_PACKAGE_DIRECTORIES", "/home/michalos/catkin_ws/devel/share/common-lisp");
        SetupRos("PYTHONPATH", "/home/michalos/catkin_ws/devel/lib/python2.7/dist-packages:/opt/ros/indigo/lib/py
        SetupRos("CMAKE_PREFIX_PATH", "/home/michalos/catkin_ws/devel:/opt/ros/indigo");
        SetupRos("PATH", "/home/michalos/catkin_ws/devel/bin:/opt/ros/indigo/bin:/home/michalos/bin:/usr/local/sb
        SetupRos("PKG_CONFIG_PATH", "/home/michalos/catkin_ws/devel/lib/x86_64-linux-gnu/pkgconfig:/opt/ros/indig
```

This does not improve the performance of Netbeans.(Often hogging up to 750M per executable.) I could rant/rave about Java, but won't. And I won't complain about lameness of gdb either.

Running

```
source ~/catkin_ws/devel/setup.bash
cd catkin_ws
roslaunch nist_fanuc lrmate200id_sim.launch
```

Naming problem with Robot links

```
[ WARN] [1456003348.287916295]: World frame 'base_link' does not match model root frame '/base_link', all poses w
```

Problem with Descartes running in Netbeans, even with ROS env set

```
[ INFO] [1456005198.420170539]: Loading robot model 'fanuc_lrmate200id'...
sh: 1: catkin_find: not found
[ERROR] [1456005198.949779328]: The kinematics plugin (manipulator) failed to load. Error: Could not find library
[ERROR] [1456005198.949844984]: Kinematics solver could not be instantiated for joint group manipulator.
[ INFO] [1456005199.039875870]: Generated 10 random seeds
Group 'manipulator' using 6 variables
  * Joints:
    'joint_1' (Revolute)
    'joint_2' (Revolute)
    'joint_3' (Revolute)
    'joint_4' (Revolute)
    'joint_5' (Revolute)
    'joint_6' (Revolute)
    'joint_6-tool0' (Fixed)
  * Variables:
    'joint_1', index 0 in full state, index 0 in group state
        P.bounded [-2.965, 2.965]; V.bounded [-7.85, 7.85]; A.bounded [-1.57, 1.57];
```

```
    'joint_2', index 1 in full state, index 1 in group state
        P.bounded [-1.74533, 2.53073]; V.bounded [-6.63, 6.63]; A.bounded [-1.326, 1.326];
    'joint_3', index 2 in full state, index 2 in group state
        P.bounded [-2.45097, 4.88692]; V.bounded [-9.08, 9.08]; A.bounded [-1.816, 1.816];
    'joint_4', index 3 in full state, index 3 in group state
        P.bounded [-3.315, 3.315]; V.bounded [-9.6, 9.6]; A.bounded [-1.92, 1.92];
    'joint_5', index 4 in full state, index 4 in group state
        P.bounded [-2.18, 2.18]; V.bounded [-9.51, 9.51]; A.bounded [-1.902, 1.902];
    'joint_6', index 5 in full state, index 5 in group state
        P.bounded [-6.285, 6.285]; V.bounded [-17.45, 17.45]; A.bounded [-3.49, 3.49];
  * Variables Index List:
    0 1 2 3 4 5 (contiguous)

[ WARN] [1456005199.040243338]: World frame 'base_link' does not match model root frame '/base_link', all poses w
```

Problem with Descartes joint planning

unable to calculate edge weight of joint transitions for joint trajectories No IK either when run from bash terminal

Standalone method to read URDF file without ROS

THis is a headache since C++ doesn't allow mutliple definitions of the same classes and typedefs, so there will be naming and typing collisions between the ROS urdf model and the Standalone reader. Useful for testing when you dont want the pain and overhead of ROS.

```
#ifdef STANDALONE
        // The following is an example of how to read a urdf file
        RCS::Controller.wm.robot_model = urdf::parseURDFFile(Globals.ExeDirectory + "lrmate200id.urdf");

        if (!RCS::Controller.wm.robot_model) {
            throw std::runtime_error("ERROR: Model Parsing the xml failed");
        }
        std::string str = urdf::GenerateTable(RCS::Controller.wm.robot_model);
        Globals.WriteFile(Globals.ExeDirectory + "robotmodel.html", str);
#endif
```

How does the CrclSession work?

in the main cpp program we declar the controller session thread (with default cycle time).

```
// This thread handles new XML messages received from  crcl asio socket.
RCS::ControllerSession session(DEFAULT_LOOP_CYCLE);
session.Start(); // start the thread
```

It uses the Thread template to run cyclically.

```
 class Thread
       void Cycle ( )
        {
            Init( ); <--- calls ControllerSession::Init()
            _timer.sync( );

            while ( _bThread )
            {
                Action( );
                _timer.wait( );
            }

            Cleanup( );
        }
```

```
   void ControllerSession::Init() {
        std::string info;
       crclinterface= boost::shared_ptr<Crcl::CrclDelegateInterface>(
              new Crcl::CrclDelegateInterface() );
        _kinematics =  boost::shared_ptr<IKinematics>(new DummyKinematics());
        std::string sStatus = DumpHeader(",") + "\n";
        CsvLogging.Timestamping() = false;
        CsvLogging.LogMessage("Timestamp," + sStatus);
    }
```

GDB

```
b main
r
s (step)
n (next)
print *(points._M_impl._M_start)@points.size()

(gdb) break source.cpp:8
(gdb) run
(gdb) p vec.begin()
$1 = {
   _M_current = 0x300340
}
(gdb) p $1._M_current->c_str()
$2 = 0x3002fc "Hello"
(gdb) p $1._M_current +1
$3 = (string *) 0x300344
(gdb) p $3->c_str()
$4 = 0x30032c "world"
```

STOPPING POSTGRESQL-9.4

sudo pkill postgres sudo update-rc.d postgresql-9.4 disable

An easy way to create a ubuntu desktop shortcut?

Open Nautilus Navigate to /usr/share/applications Right-click on the application you want to use and select copy Click on your desktop and select paste Right click on the icon that has just been created and select properties On the Permissions tab check Execute then click Close

```
http://gazebosim.org/tutorials?tut=drcsim_ros_cmds&cat=drcsim
RCSInterpreter::ParseCommand
RobotStatus::Action

PID: http://wiki.ros.org/pr2_mechanism/Tutorials/Adding%20a%20PID%20to%20a%20realtime%20joint%20controller
Joint XML https://github.com/RethinkRobotics/baxter_simulator/blob/master/baxter_sim_controllers/src/baxter_veloc
```

http://gazebosim.org/tutorials?tut=drcsim_ros_cmds&cat=drcsim

How to publish joints to ROS

http://answers.ros.org/question/43157/trying-to-use-get-joint-state-with-my-urdf/

How to visualize trajectory path in moveit

So I think I have a workable answer to my own question. Using the Python API, the following code snippet appears to alter the trajectory speed as desired (here the speed is doubled):

```
     traj = right_arm.plan()
     new_traj = RobotTrajectory()
     new_traj.joint_trajectory = traj.joint_trajectory
     n_joints = len(traj.joint_trajectory.joint_names)
     n_points = len(traj.joint_trajectory.points)

     spd = 2.0

     for i in range(n_points):
         traj.joint_trajectory.points[i].time_from_start =
traj.joint_trajectory.points[i].time_from_start / spd
         for j in range(n_joints):
new_traj.joint_trajectory.points[i].velocities[j] =
traj.joint_trajectory.points[i].velocities[j] * spd
new_traj.joint_trajectory.points[i].accelerations[j] =
traj.joint_trajectory.points[i].accelerations[j] * spd
new_traj.joint_trajectory.points[i].positions[j] =
traj.joint_trajectory.points[i].positions[j]

     self.right_arm.execute(new_traj)
Web references:
https://github.com/ros-controls/ros_controllers/blob/jade-devel/joint_trajectory_controller/test/joint_trajectory
https://github.com/nttputus/wp6_manipulator/blob/master/crops_wp6_arm_navigation_tutorials/src/display_trajectory
```

How the fanuc lrmate 200id IKFast kinematics plug in is installed

http://sdk.rethinkrobotics.com/wiki/MoveIt_Tutorial

rosed <myrobot_name>_moveit_config/config/kinematics.yaml

Edit these parts:

```
<planning_group_name>:
  kinematics_solver: <moveit_ik_plugin_pkg>/IKFastKinematicsPlugin
```

-OR-

```
  kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin
```

IBID: /home/michalos/catkin_ws/src/fanuc_experimental/fanuc_lrmate200ib3l_moveit_config/config/kinematics.yaml manipulator:

kinematics_solver: fanuc_lrmate200id_manipulator_kinematics/IKFastKinematicsPlugin kinematics_solver_attempts: 3 kinematics_solver_search_resolution: 0.005 kinematics_solver_timeout: 0.005

You will now need to recompile and install the IKFast plugins.

```
   catkin_make
   catkin_make install --pkg fanuc_lrmate200ib3l_manipulator_kinematics_plugin
```

```
michalos@rufous:~/catkin_ws$ rospack list
....
fanuc_lrmate200id_moveit_config /home/michalos/catkin_ws/src/fanuc_experimental/fanuc_lrmate200id_moveit_config
fanuc_lrmate200id_moveit_plugins /home/michalos/catkin_ws/src/fanuc_experimental/fanuc_lrmate200id_moveit_plugins
fanuc_lrmate200id_support /home/michalos/catkin_ws/src/fanuc_experimental/fanuc_lrmate200id_support

...
```

installing uncrustify - C++ code formatter

```
sudo apt-get install uncrustify
sudo apt-get install universalindentgui

michalos@rufous:~/catkin_ws$ universalindentgui
```

Diplay contents of rostopic to see current robot pose

rostopic echo /move_group/goal

header: seq: 30 stamp: secs: 1456420731 nsecs: 958832629 frame_id: ” goal_id: stamp: secs: 1456420731 nsecs: 958833341 id: /rviz_rufous_6391_9202004766824814528-31-1456420731.958833341 goal: request: workspace_parameters: header: seq: 0 stamp: secs: 1456420731 nsecs: 958705952 frame_id: /base_link min_corner: x: -1.0 y: -1.0 z: -1.0 max_corner: x: 1.0 y: 1.0 z: 1.0 start_state: joint_state: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: /base_link name: [’joint_1’, ’joint_2’, ’joint_3’, ’joint_4’, ’joint_5’, ’joint_6’] position: [-0.9951596824786128, -1.161361427138559, 0.6456418463565006, 3.1368453900839213, 2.0662861177525222, -1.4036373909368032] velocity: [] effort: [] multi_dof_joint_state: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: /base_link joint_names: [] transforms: [] twist: [] wrench: [] attached_collision_objects: [] is_diff: False goal_constraints:

- name: ” joint_constraints:
  - joint_name: joint_1 position: 1.75594190178 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0
  - joint_name: joint_2 position: -1.50958199864 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0
  - joint_name: joint_3 position: 2.60972229461 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0
  - joint_name: joint_4 position: -1.55001527988 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0
  - joint_name: joint_5 position: 1.68989821207 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0
  - joint_name: joint_6 position: 2.29309630865 tolerance_above: 0.0001 tolerance_below: 0.0001 weight: 1.0 position_constraints: [] orientation_constraints: [] visibility_constraints: [] path_constraints: name: ” joint_constraints: [] position_constraints: [] orientation_constraints: [] visibility_constraints: [] trajectory_constraints: constraints: [] planner_id: ” group_name: manipulator num_planning_attempts: 1 allowed_planning_time: 1.0 max_velocity_scaling_factor: 1.0 planning_options: planning_scene_diff: name: ” robot_state: joint_state: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: ” name: [] position: [] velocity: [] effort: [] multi_dof_joint_state: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: ” joint_names: [] transforms: [] twist: [] wrench: [] attached_collision_objects: [] is_diff: True robot_model_name: ” fixed_frame_transforms: [] allowed_collision_matrix: entry_names: [] entry_values: [] default_entry_names: [] default_entry_values: [] link_padding: [] link_scale: [] object_colors: [] world: collision_objects: [] octomap: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: ” origin: position: x: 0.0 y: 0.0 z: 0.0 orientation: x: 0.0 y: 0.0 z: 0.0 w: 0.0 octomap: header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: ” binary: False id: ” resolution: 0.0 data: [] is_diff: True plan_only: False look_around: False look_around_attempts: 0 max_safe_execution_cost: 0.0 replan: False replan_attempts: 0

    replan_delay: 2.0

Display only the pose of the robot

```
sudo apt-get install ros-indigo-robot-pose-publisher
rosrun robot_pose_publisher robot_pose_publisher
```

FAILED!

Moveit failed to plan also

```
[ INFO] [1456439525.551846365]: Planning request received for MoveGroup action. Forwarding to planning pipeline.
[ WARN] [1456439525.554072295]: Orientation constraint for link 'tool0' is probably incorrect: 0.000000, 0.000000
[ WARN] [1456439525.554147337]: Orientation constraint for link 'tool0' is probably incorrect: 0.000000, 0.000000
[ INFO] [1456439525.554827784]: LBKPIECE1: Starting planning with 1 states already in datastructure
[ INFO] [1456439526.966671637]: Found a contact between 'link_4' (type 'Robot link') and 'base_link' (type 'Robot
[ INFO] [1456439526.966721656]: Collision checking is considered complete (collision was found and 0 contacts are
[ INFO] [1456439526.967459590]: Found a contact between 'link_6' (type 'Robot link') and 'base_link' (type 'Robot
[ INFO] [1456439526.967502013]: Collision checking is considered complete (collision was found and 0 contacts are
[ INFO] [1456439526.968228658]: Found a contact between 'link_4' (type 'Robot link') and 'base_link' (type 'Robot
[ INFO] [1456439526.968269033]: Collision checking is considered complete (collision was found and 0 contacts are
[ INFO] [1456439526.969014693]: Found a contact between 'link_6' (type 'Robot link') and 'base_link' (type 'Robot
[ INFO] [1456439526.969054959]: Collision checking is considered complete (collision was found and 0 contacts are
[ERROR] [1456439528.325962429]: LBKPIECE1: Unable to sample any valid states for goal tree
[ INFO] [1456439528.325996844]: LBKPIECE1: Created 1 (1 start + 0 goal) states in 1 cells (1 start (1 on boundary
[ INFO] [1456439528.326030131]: No solution found after 2.771767 seconds
[ INFO] [1456439528.326055767]: Unable to solve the planning problem
```

Moveit planner hangs

```
    if (!group->plan(my_plan))
        return false;
```

You must have multithreaded enabled before any moveit planning.

```
        // Required for multithreaded communication with moveit components
        ros::AsyncSpinner spinner(1);
        spinner.start();
```

moveit planning with joints

```
    joints.position = moveit.SetRandomJoints();
    std::cout << "Random assigned joints=" << VectorDump<double> (joints.position).c_str();
    moveit.Plan(joints);
    sleep(15.0);
    js = moveit.GetJointValues();
    std::cout << "Current joints=" << VectorDump<double> (js);
```

You must wait for rviz to move. You get all kinds of plans.

rviz not moving

```
pub <topic-name> <topic-type> [data...]

rostopic pub -1 /joint_states sensor_msgs/JointState '{header: auto, name: ['joint_1', 'joint_2', 'joint_3', 'joi

rostopic pub -1 /joint_states sensor_msgs/JointState '{header: auto, name: ['joint_1', 'joint_2', 'joint_3', 'joi

rostopic pub /joint_states sensor_msgs/JointState '{header: auto, name: ['joint_1', 'joint_2', 'joint_3', 'joint_
```

Writing joint values to rviz and then moving robot and "waiting" until rviz reaches goal

Fixed ToVector and made template parameter mandatory. THen added wait until "AtGoal" motion control. Rviz need processing and that only comes when waiting. Again, make sure ros::AsyncSpinner is running!

```
        joint.position=ToVector<double>(6, 0.097, 0.007, -0.590, -0.172, 0.604, -0.142 );
        jointWriter->JointTrajectoryPositionWrite(joint);
```

```
        while( !MotionControl::AtGoal(joint, curjoints) && n++ < 100) {
            curjoints = jointReader->GetCurrentReadings();
            std::cout << "Goal joints=" << VectorDump<double> (joint.position).c_str();
            std::cout << "Cur  joints=" << VectorDump<double> (curjoints.position).c_str();
            Globals.Sleep(100); // 100 milliseconds
        }

template<typename T>
static bool epsiloncompare (T &i, T& j) {
  return (fabs(i-j) < MotionControl::epsilon);
}
template<typename T>
static bool VectorCompare(std::vector<T> vec1,std::vector<T> vec2 )
{
   return std::equal (vec1.begin(), vec1.end(),  vec2.begin() , epsiloncompare) ;
}
bool MotionControl::AtGoal(JointState goal, JointState current,  double epsilon)
{
    MoveitTrajectory::epsilon=epsilon;
    return VectorCompare<double>(goal.position, current.position);
}
```

Turn off Planned Path visualization in Rviz - confusing

In Rviz GUI, navigate to Planned Path, and uncheck "Show Robot Visual"

Not sure how this parameter is set in /move_group/display_planned_path

Using waypoints in moveit

Calculated a waypoint every 1mm

```
std::vector<JointState> MoveitTrajectory::Plan(std::vector<urdf::Pose>& pwaypoints) {
    std::vector<geometry_msgs::Pose> waypoints;
    for(size_t i=0; i< pwaypoints.size(); i++)
    {
        waypoints.push_back(PoseMsg2UrdfPose(pwaypoints[i]));
    }

    moveit_msgs::RobotTrajectory trajectory;
    double fraction = group->computeCartesianPath(waypoints,
            0.001, // cartesian path to be interpolated at a resolution of 1 mm
            0.0, // NO jump_threshold
            trajectory);  // trajectory.joint_trajectory.points  (position)
    std::vector<JointState> points;
    for(size_t j=0; j< trajectory.joint_trajectory.points.size(); j++)
    {
        JointState traj;
        traj.position = trajectory.joint_trajectory.points[j].positions;
        points.push_back(traj);
    }
    return points;
}
```

Test code:

```
       MoveitTrajectory moveit(nh);
        MotionControl motioncontrol;
        urdf::Pose goalpose;
        goalpose.position =urdf::Vector3(.28,-0.7,1.0);
        goalpose.rotation.setFromRPY(0.,0.,0.);

        int nIncr=motioncontrol.computeIncrements (RCS::Controller.status.currentpose, goalpose);
```

```
std::vector<urdf::Pose> poses = motioncontrol.computeWaypoints(RCS::Controller.status.currentpose, goalpo
std::vector<JointState> points = moveit.Plan(poses);
for(size_t k=0; k< points.size(); k++)
{
    std::cout <<  VectorDump<double> (points[k].position);
    jointWriter->JointTrajectoryPositionWrite(points[k]);
}
```

Running Fanuc Robot - notes

Powerup:

1. Turn on power on front of controller (keyed)
2. If auto mode, make sure teach pendant upper left corner knob is OFF
3. If in fault- hold deadman switch halfway, Hold [Shift], press [Reset] key
4. reset to local mode Menu -> [32] Remote/Local/... [F4] Local [Enter]
5. Start ROS programs [Teach][Select] => scroll down to ROS, number 39 hit [Enter] (starts 2 programs)
6. Cycle start Green Auto button on front controller panel, press/release, green light should go on.

Powerdown:

1. Kill ROS programs - DO TWICE - 2 programs running [FCNT] -> 1 -> [ENTER] [FCNT] -> 1 -> [ENTER] If fanuc controller faulted, and have to manually reset joint to safe position

1. Turn controller box to teach pendant from auto
2. hold deadman switch half-on, [SHIFT] Hold, hit [Reset]
3. Now move robot - +/- joint key or xyz key
4. Note to increase traversal- feedoverride in green xx% field in upper right corner

Run ROS Fanuc demo

```
roslaunch fanuc_lrmate200id_moveit_config  moveit_planning_execution.launch
  sim:=false   robot_ip:=129.6.78.111
```

Run RVIZ roslaunch with Fanuc LRMate 200id #!/bin/bash source /home/michalos/catkin_ws/devel/setup.bash roslaunch nist_fanuc lrmate200id_sim.launch

sleep 100

Launch file:

```
<?xml version="1.0"?>
<launch>

  <arg name="sim" default="true" />

  <include file="$(find fanuc_lrmate200id_moveit_config)/launch/moveit_planning_execution.launch">
    <arg name="sim" value="$(arg sim)"/>
  </include>

</launch>
```

Fanuc 200id kinematics plugin

/home/michalos/catkin_ws/src/fanuc_experimental/fanuc_lrmate200id_moveit_config/config/kinematics.yaml

```
manipulator:
  kinematics_solver: fanuc_lrmate200id_manipulator_kinematics/IKFastKinematicsPlugin
  kinematics_solver_attempts: 3
  kinematics_solver_search_resolution: 0.005
  kinematics_solver_timeout: 0.005
```

Maybe:

kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin

Worked!

```
void visualize(ros::NodeHandle nh, moveit_msgs::MotionPlanResponse response) {
    ROS_INFO("Visualizing the trajectory");
    ros::Publisher display_publisher = nh.advertise<moveit_msgs::DisplayTrajectory>("/move_group/display_planned_
    moveit_msgs::DisplayTrajectory display_trajectory;

    display_trajectory.trajectory_start = response.trajectory_start;
    display_trajectory.trajectory.push_back(response.trajectory);
    display_publisher.publish(display_trajectory);
}
```

Ros Cartesian Planning with assigned plugin

```
int main(int argc, char **argv) {
    ros::init (argc, argv, "planning_pipeline");
    ros::AsyncSpinner spinner(1);
    spinner.start();
    ros::NodeHandle node_handle("~");

    //map<int, Controller*> controllersOrder;
    //vector<Controller>  controllers ;//= initControllers(node_handle);


    robot_model_loader::RobotModelLoader robot_model_loader("robot_description");
    robot_model::RobotModelPtr robot_model = robot_model_loader.getModel();

    planning_scene::PlanningScenePtr planning_scene(new planning_scene::PlanningScene(robot_model));
    planning_pipeline::PlanningPipelinePtr planning_pipeline(new planning_pipeline::PlanningPipeline(robot_model,

    //Sleep a little to allow time to startup rviz, etc.
    ros::WallDuration sleep_time(5.0);
    sleep_time.sleep();

    // A tolerance of 0.01 m is specified in position
    // and 0.01 radians in orientation
    vector<double> tolerance_pose(3, 0.01);
    vector<double> tolerance_angle(3, 0.01);

    // Pose Goal
    // ^^^^^^^^^
    // We will now create a motion plan request for the right arm of the PR2
    // specifying the desired pose of the end-effector as input.
    planning_interface::MotionPlanRequest req;
    planning_interface::MotionPlanResponse res;
    geometry_msgs::PoseStamped pose;
    pose.header.frame_id = "torso";
    pose.pose.position.x = -0.000006;
    pose.pose.position.y = 0.05;
    pose.pose.position.z = -0.24;
```

```
    req.group_name = "leg_left";
    req.planner_id = "RRTkConfigDefault";
    req.allowed_planning_time=5;
    req.num_planning_attempts = 5;

    moveit_msgs::Constraints pose_goal = kinematic_constraints::constructGoalConstraints("l_sole", pose, tolerance
    req.goal_constraints.push_back(pose_goal);

    planning_pipeline->generatePlan(planning_scene, req, res);
    if(res.error_code_.val != res.error_code_.SUCCESS) {
        ROS_ERROR("Could not compute plan successfully");
        return 0;
    }

    moveit_msgs::MotionPlanResponse response;
    res.getMessage(response);

    // Visualize the result
        ROS_INFO("Visualizing the trajectory");
    ros::Publisher display_publisher = node_handle.advertise<moveit_msgs::DisplayTrajectory>("/move_group/display_
    moveit_msgs::DisplayTrajectory display_trajectory;

    display_trajectory.trajectory_start = response.trajectory_start;
    display_trajectory.trajectory.push_back(response.trajectory);
    display_publisher.publish(display_trajectory);


    //visualize(node_handle, response);

    ros::waitForShutdown();

    return 0;
```

rosparam - get list of ROS parameter from paramserver

Does not provide the values....

```
michalos@rufous:~/catkin_ws$ rosparam list
/controller_joint_names
/move_group/allow_trajectory_execution
/move_group/allowed_execution_duration_scaling
/move_group/allowed_goal_duration_margin
/move_group/capabilities
/move_group/controller_list
/move_group/jiggle_fraction
/move_group/manipulator/longest_valid_segment_fraction
/move_group/manipulator/planner_configs
/move_group/manipulator/projection_evaluator
/move_group/max_range
/move_group/max_safe_path_cost
/move_group/moveit_controller_manager
/move_group/moveit_manage_controllers
/move_group/octomap_resolution
/move_group/ompl/display_random_valid_states
/move_group/ompl/link_for_exploration_tree
/move_group/ompl/maximum_waypoint_distance
/move_group/ompl/minimum_waypoint_count
/move_group/ompl/simplify_solutions
/move_group/plan_execution/max_replan_attempts
/move_group/plan_execution/record_trajectory_state_frequency
/move_group/planner_configs/BKPIECEkConfigDefault/border_fraction
/move_group/planner_configs/BKPIECEkConfigDefault/failed_expansion_score_factor
/move_group/planner_configs/BKPIECEkConfigDefault/min_valid_path_fraction
/move_group/planner_configs/BKPIECEkConfigDefault/range
/move_group/planner_configs/BKPIECEkConfigDefault/type
```

```
/move_group/planner_configs/ESTkConfigDefault/goal_bias
/move_group/planner_configs/ESTkConfigDefault/range
/move_group/planner_configs/ESTkConfigDefault/type
/move_group/planner_configs/KPIECEkConfigDefault/border_fraction
/move_group/planner_configs/KPIECEkConfigDefault/failed_expansion_score_factor
/move_group/planner_configs/KPIECEkConfigDefault/goal_bias
/move_group/planner_configs/KPIECEkConfigDefault/min_valid_path_fraction
/move_group/planner_configs/KPIECEkConfigDefault/range
/move_group/planner_configs/KPIECEkConfigDefault/type
/move_group/planner_configs/LBKPIECEkConfigDefault/border_fraction
/move_group/planner_configs/LBKPIECEkConfigDefault/min_valid_path_fraction
/move_group/planner_configs/LBKPIECEkConfigDefault/range
/move_group/planner_configs/LBKPIECEkConfigDefault/type
/move_group/planner_configs/PRMkConfigDefault/max_nearest_neighbors
/move_group/planner_configs/PRMkConfigDefault/type
/move_group/planner_configs/PRMstarkConfigDefault/type
/move_group/planner_configs/RRTConnectkConfigDefault/range
/move_group/planner_configs/RRTConnectkConfigDefault/type
/move_group/planner_configs/RRTkConfigDefault/goal_bias
/move_group/planner_configs/RRTkConfigDefault/range
/move_group/planner_configs/RRTkConfigDefault/type
/move_group/planner_configs/RRTstarkConfigDefault/delay_collision_checking
/move_group/planner_configs/RRTstarkConfigDefault/goal_bias
/move_group/planner_configs/RRTstarkConfigDefault/range
/move_group/planner_configs/RRTstarkConfigDefault/type
/move_group/planner_configs/SBLkConfigDefault/range
/move_group/planner_configs/SBLkConfigDefault/type
/move_group/planner_configs/TRRTkConfigDefault/frountierNodeRatio
/move_group/planner_configs/TRRTkConfigDefault/frountier_threshold
/move_group/planner_configs/TRRTkConfigDefault/goal_bias
/move_group/planner_configs/TRRTkConfigDefault/init_temperature
/move_group/planner_configs/TRRTkConfigDefault/k_constant
/move_group/planner_configs/TRRTkConfigDefault/max_states_failed
/move_group/planner_configs/TRRTkConfigDefault/min_temperature
/move_group/planner_configs/TRRTkConfigDefault/range
/move_group/planner_configs/TRRTkConfigDefault/temp_change_factor
/move_group/planner_configs/TRRTkConfigDefault/type
/move_group/planning_plugin
/move_group/planning_scene_monitor/publish_geometry_updates
/move_group/planning_scene_monitor/publish_planning_scene
/move_group/planning_scene_monitor/publish_planning_scene_hz
/move_group/planning_scene_monitor/publish_state_updates
/move_group/planning_scene_monitor/publish_transforms_updates
/move_group/request_adapters
/move_group/sense_for_plan/discard_overlapping_cost_sources
/move_group/sense_for_plan/max_cost_sources
/move_group/sense_for_plan/max_look_attempts
/move_group/sense_for_plan/max_safe_path_cost
/move_group/start_state_max_bounds_error
/move_group/trajectory_execution/allowed_execution_duration_scaling
/move_group/trajectory_execution/execution_duration_monitoring
/move_group/trajectory_execution/execution_velocity_scaling
/robot_description
/robot_description_kinematics/manipulator/kinematics_solver
/robot_description_kinematics/manipulator/kinematics_solver_attempts
/robot_description_kinematics/manipulator/kinematics_solver_search_resolution
/robot_description_kinematics/manipulator/kinematics_solver_timeout
/robot_description_planning/joint_limits/joint_1/has_acceleration_limits
/robot_description_planning/joint_limits/joint_1/has_velocity_limits
/robot_description_planning/joint_limits/joint_1/max_acceleration
/robot_description_planning/joint_limits/joint_1/max_velocity
/robot_description_planning/joint_limits/joint_2/has_acceleration_limits
/robot_description_planning/joint_limits/joint_2/has_velocity_limits
/robot_description_planning/joint_limits/joint_2/max_acceleration
/robot_description_planning/joint_limits/joint_2/max_velocity
/robot_description_planning/joint_limits/joint_3/has_acceleration_limits
/robot_description_planning/joint_limits/joint_3/has_velocity_limits
/robot_description_planning/joint_limits/joint_3/max_acceleration
```

```
/robot_description_planning/joint_limits/joint_3/max_velocity
/robot_description_planning/joint_limits/joint_4/has_acceleration_limits
/robot_description_planning/joint_limits/joint_4/has_velocity_limits
/robot_description_planning/joint_limits/joint_4/max_acceleration
/robot_description_planning/joint_limits/joint_4/max_velocity
/robot_description_planning/joint_limits/joint_5/has_acceleration_limits
/robot_description_planning/joint_limits/joint_5/has_velocity_limits
/robot_description_planning/joint_limits/joint_5/max_acceleration
/robot_description_planning/joint_limits/joint_5/max_velocity
/robot_description_planning/joint_limits/joint_6/has_acceleration_limits
/robot_description_planning/joint_limits/joint_6/has_velocity_limits
/robot_description_planning/joint_limits/joint_6/max_acceleration
/robot_description_planning/joint_limits/joint_6/max_velocity
/robot_description_semantic
/rosdistro
/roslaunch/uris/host_rufous__60560
/rosversion
/run_id
/rviz_rufous_23871_3766625969024100662/manipulator/kinematics_solver
/rviz_rufous_23871_3766625969024100662/manipulator/kinematics_solver_attempts
/rviz_rufous_23871_3766625969024100662/manipulator/kinematics_solver_search_resolution
/rviz_rufous_23871_3766625969024100662/manipulator/kinematics_solver_timeout
/rviz_rufous_23871_3766625969024100662/motionplanning_planning_scene_monitor/publish_geometry_updates
/rviz_rufous_23871_3766625969024100662/motionplanning_planning_scene_monitor/publish_planning_scene
/rviz_rufous_23871_3766625969024100662/motionplanning_planning_scene_monitor/publish_planning_scene_hz
/rviz_rufous_23871_3766625969024100662/motionplanning_planning_scene_monitor/publish_state_updates
/rviz_rufous_23871_3766625969024100662/motionplanning_planning_scene_monitor/publish_transforms_updates
michalos@rufous:~/catkin_ws$
```

Remove installed package from ubuntu

sudo apt-get remove grip

Pose conversion tests

Old way: CRCL conversion tests CRCL Pose 0.4643,0.02436,1.275,0.01676,0.08284,0.9964,0.2896,0.9535,-0.08413, Urdf Pose Translation = 464.3:24.36:1275 Rotation = -95.0423:16.8334:-88.9969

Eigen way: no rpy intermediary step CRCL Pose 0.4643,0.02436,1.275,0.01676,0.08284,0.9964,0.2896,0.9535,-0.-08413, Urdf Pose Translation = 464.3:24.36:1275 Rotation = 174.572:-85.1698:78.5552

urdf::Pose Convert(Crcl::PoseType & pose, double lengthConversion) { urdf::Pose p;

p.position.x = pose.Point().X() ∗ lengthConversion; p.position.y = pose.Point().Y() ∗ lengthConversion; p.position.z = pose.Point().Z() ∗ lengthConversion;

Eigen::Matrix3d mat=GetEigenRotMatrix(GetVector3D(pose.XAxis()), GetVector3D(pose.ZAxis())); Eigen::Quaterniond q(mat); p.rotation.x = q.x(); p.rotation.y = q.y(); p.rotation.z = q.z(); p.rotation.w = q.w(); return p; }

Using ROS whose RPY from quaterion doesnt work!

Instead use posemath, gomotion rpy from matrix via quaterion

```
GotoCRCL Pose 0.465,0,0.745,-2.051e-10,-8.979e-11,1,1,0,2.051e-10,
Goto urdf Pose Translation = 465:0:745
Rotation = 180:-90:0
```

Position Only IK

Position only IK can easily be enabled (only if you are using the KDL Kinematics Plugin) by adding the following line to your kinematics.yaml file (for the particular group that you want to solve IK for):

position_only_ik: True

I suppose this is "easily" if you dont ever want to change it dynamically or programmatically without modifying the yaml file....

Problems iwth ikfast

http://answers.ros.org/question/205781/moveit-inverse_kinematics-c-api/ Side note: I am aware that FastIK is a possible alternative to KDL, however it requires to install OpenRave, which seems to be problematic under Ubuntu 14.04. Also the ros converter urdf_to_collada for indigo is broken and not working. There is just generally speaking a lot of hassle to get an IK solver at the moment under indigo.

JM: Agree!

Person used KDL IK directly

Markdown Previewer

https://visualstudiogallery.msdn.microsoft.com/0855e23e-4c4c-4c82-8b39-24ab5c5a7f79

The chrome markdown preview was soooo flaky, gave up on it, although it would have been convenient.

And there were 85 markdown previewers to choose from. As alinux user, I'm getting used to an abundance of broken/useless/limited software crap.

CRCl Program

```
CrclDelegateInterface::SetLengthUnits
CrclDelegateInterface::SetTransSpeed
CrclDelegateInterface::SetTransAccel
CrclDelegateInterface::SetEndPoseTolerance
CrclDelegateInterface::SetIntermediatePoseTolerance
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 1.5,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 1500:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 1.5,1,0.0001,1,0,0,0,0,-1,
Goto urdf Pose Translation = 1500:1000:0.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 1.5,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 1500:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 4,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 4,1,0.5001,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:500.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 4,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:1000
```

```
Rotation = 180:-0:0
GotoCRCL Pose 8.25,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.25,1,0.4,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:400
Rotation = 180:-0:0
CrclDelegateInterface::OpenToolChanger
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 8.25,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.75,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.75,1,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:500
Rotation = 180:-0:0
CrclDelegateInterface::CloseToolChanger
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 8.75,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 5.659,1.1,1.8,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:1100:1800
Rotation = 180:-0:0
GotoCRCL Pose 5.659,1.1,0.1501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:1100:150.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 5.659,1.1,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:1100:500
Rotation = 180:-0:0
GotoCRCL Pose 3.86,1.07,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:1070:1000
Rotation = 180:-0:0
GotoCRCL Pose 3.86,1.07,0.6501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:1070:650.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 3.86,1.07,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:1070:1000
Rotation = 180:-0:0
GotoCRCL Pose 5.659,0.9,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:900:500
Rotation = 180:-0:0
GotoCRCL Pose 5.659,0.9,0.1501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:900:150.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 5.659,0.9,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 5659:900:500
Rotation = 180:-0:0
GotoCRCL Pose 3.86,0.93,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:930:1000
Rotation = 180:-0:0
GotoCRCL Pose 3.86,0.93,0.6501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:930:650.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 3.86,0.93,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 3860:930:1000
Rotation = 180:-0:0
GotoCRCL Pose 6.42,1,0.5,1,0,0,0,0,-1,
```

```
Goto urdf Pose Translation = 6420:1000:500
Rotation = 180:-0:0
GotoCRCL Pose 6.42,1,0.1501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 6420:1000:150.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 6.42,1,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 6420:1000:500
Rotation = 180:-0:0
GotoCRCL Pose 4.14,0.93,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:930:1000
Rotation = 180:-0:0
GotoCRCL Pose 4.14,0.93,0.6501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:930:650.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 4.14,0.93,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:930:1000
Rotation = 180:-0:0
GotoCRCL Pose 7.61,1.02,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 7610:1020:500
Rotation = 180:-0:0
GotoCRCL Pose 7.61,1.02,0.1501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 7610:1020:150.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 7.61,1.02,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 7610:1020:500
Rotation = 180:-0:0
GotoCRCL Pose 4.14,1.07,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:1070:1000
Rotation = 180:-0:0
GotoCRCL Pose 4.14,1.07,0.6501,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:1070:650.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 4.14,1.07,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4140:1070:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.75,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.75,1,0.475,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:475
Rotation = 180:-0:0
CrclDelegateInterface::OpenToolChanger
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 8.75,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8750:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.25,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 8.25,1,0.5,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:500
Rotation = 180:-0:0
CrclDelegateInterface::CloseToolChanger
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 8.25,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 8250:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 4,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:1000
Rotation = 180:-0:0
```

```
GotoCRCL Pose 4,1,0.5001,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:500.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 4,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 4000:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 2.5,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 2500:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 2.5,1,0.0001,1,0,0,0,0,-1,
Goto urdf Pose Translation = 2500:1000:0.1
Rotation = 180:-0:0
CrclDelegateInterface::StopMotion
CrclDelegateInterface::MoveThroughTo
GotoCRCL Pose 2.5,1,1,1,0,0,0,0,-1,
Goto urdf Pose Translation = 2500:1000:1000
Rotation = 180:-0:0
GotoCRCL Pose 0.5,0,2,1,0,0,0,0,-1,
Goto urdf Pose Translation = 500:0:2000
Rotation = 180:-0:0
```

Where does the fanuc robot go?

```
(0,0,0,0,0k,0)
.465 0 .695 0 0 1 1 0 0  (180 -90 0)

(33,0,0,0,0,0)
.39 .253 .695  0 .545 .839 1  0 0   (90 -57 90)


CrclDelegateInterface::MoveTo
GotoCRCL Pose 0.39,0.253,0.695,0,0.545,0.839,1,0,0,
Goto urdf Pose Translation = 390:253:695
Rotation = 180:-89.9853:-180

==========================================

CrclDelegateInterface::SetLengthUnits
CrclDelegateInterface::SetTransSpeed
CrclDelegateInterface::SetTransAccel
CrclDelegateInterface::SetEndPoseTolerance
CrclDelegateInterface::SetIntermediatePoseTolerance
CrclDelegateInterface::SetEndEffector
CrclDelegateInterface::MoveTo
GotoCRCL Pose 0.39,0.253,0.695,0,0.545,0.839,1,0,0,
Goto urdf Pose Translation = 390:253:695
Rotation = 180:-89.9853:-180
CrclDelegateInterface::Dwell

RCSInterpreter::ParseCommand
RCSInterpreter::ParseCommand
RCSInterpreter::ParseCommand
Current Pose Translation = 464.975:-0.00919851:695.009
Rotation = -150.214:-89.9977:-29.7826
Goal Pose Translation = 390:253:695
Rotation = 180:-89.9853:-180
CartesianMotion  Poses Translation = 464.975:-0.00919851:695.009
Rotation = -150.214:-89.9977:-29.7826
CartesianMotion  Poses Translation = 439.983:84.3272:695.006
Rotation = -163.179:-89.9993:-113.995
CartesianMotion  Poses Translation = 414.992:168.664:695.003
Rotation = -176.021:-89.9941:-177.603
CartesianMotion  Poses Translation = 390:253:695
Rotation = 180:-89.9853:-180
```

```
New IK Joints -1.97821e-05:-7.22635e-05:-3.84365e-05:6.22758e-05:-6.83954e-05:-1.77077e-06:
GotoPose Translation = 464.975:-0.00919851:695.009
Rotation = -150.214:-89.9977:-29.7826
New Joints -1.97821e-05:-7.22635e-05:-3.84365e-05:6.22758e-05:-6.83954e-05:-1.77077e-06:
New IK Joints -2.91129:-1.44586:0.298472:0.234648:1.40243:-3.29449:
GotoPose Translation = 439.983:84.3272:695.006
Rotation = -163.179:-89.9993:-113.995
New Joints -2.91129:-1.44586:0.298472:0.234648:1.40243:-3.29449:
New IK Joints -2.67506:-1.44189:0.319451:0.474739:1.39747:-3.43398:
GotoPose Translation = 414.992:168.664:695.003
Rotation = -176.021:-89.9941:-177.603
New Joints -2.67506:-1.44189:0.319451:0.474739:1.39747:-3.43398:
New IK Joints -2.45751:-1.41568:0.427933:0.701811:1.34968:-3.54969:
GotoPose Translation = 390:253:695
Rotation = 180:-89.9853:-180
New Joints -2.45751:-1.41568:0.427933:0.701811:1.34968:-3.54969:
New Joint Position -1.97821e-05:-7.22635e-05:-3.84365e-05:6.22758e-05:-6.83954e-05:-1.77077e-06:
New Joint Position -2.91129:-1.44586:0.298472:0.234648:1.40243:-3.29449:
New Joint Position -2.67506:-1.44189:0.319451:0.474739:1.39747:-3.43398:
New Joint Position -2.45751:-1.41568:0.427933:0.701811:1.34968:-3.54969:
RCSInterpreter::ParseCommand


Current joints=0:0:0:0:0:0:
CrclDelegateInterface::SetAngleUnitsDEGREE
Current=0:0:0:0:0:0:
Translation =    465.0000:     0.0000:   695.0000
CrclDelegateInterface::SetLengthUnits=meter
CrclDelegateInterface::SetTransSpeed
CrclDelegateInterface::SetTransAccel
CrclDelegateInterface::SetEndPoseTolerance
CrclDelegateInterface::SetIntermediatePoseTolerance
CrclDelegateInterface::SetEndEffector= 0.00
CrclDelegateInterface::MoveTo
GotoCRCL Pose 0.39,0.253,0.695,0,0.545,0.839,1,0,0,
Goto urdf Pose Translation = 390:253:695
Rotation = 180:-89.9853:-180
```

THIS TIME IK WORKED????!!!!??!!

```
CrclDelegateInterface::Dwell=%5.2
RCSInterpreter::ParseCommand
RCSInterpreter::ParseCommand
RCSInterpreter::ParseCommand
Current Pose Translation = 465:0:695
Rotation = 180:-90:0
Goal Pose Translation = 390:253:695
Rotation = 180:-89.9853:-180
CartesianMotion  Poses Translation = 465:0:695
Rotation = 180:-90:0
CartesianMotion  Poses Translation = 440:84.3333:695
Rotation = -180:-89.9983:-180
CartesianMotion  Poses Translation = 415:168.667:695
Rotation = -180:-89.9933:-180
CartesianMotion  Poses Translation = 390:253:695
Rotation = 180:-89.9853:-180
New IK Joints 0:0:0:0:0:0:
GotoPose Translation = 465:0:695
Rotation = 180:-90:0
New Joints 0:0:0:0:0:0:
New IK Joints 0.22962:-0.0461163:-0.0437904:-1.60459:0.227424:-1.79647:
GotoPose Translation = 440:84.3333:695
Rotation = -180:-89.9983:-180
New Joints 0.22962:-0.0461163:-0.0437904:-1.60459:0.227424:-1.79647:
New IK Joints 0.466449:-0.0303704:-0.0331881:-1.53475:0.466877:-1.91761:
GotoPose Translation = 415:168.667:695
Rotation = -180:-89.9933:-180
```

```
New Joints 0.466449:-0.0303704:-0.0331881:-1.53475:0.466877:-1.91761:
New IK Joints 0.684214:0.0459945:0.0494784:-1.59129:0.68255:-2.1703:
GotoPose Translation = 390:253:695
Rotation = 180:-89.9853:-180
New Joints 0.684214:0.0459945:0.0494784:-1.59129:0.68255:-2.1703:
New Joint Position 0:0:0:0:0:0:
New Joint Position 0.22962:-0.0461163:-0.0437904:-1.60459:0.227424:-1.79647:
New Joint Position 0.466449:-0.0303704:-0.0331881:-1.53475:0.466877:-1.91761:
New Joint Position 0.684214:0.0459945:0.0494784:-1.59129:0.68255:-2.1703:
RCSInterpreter::ParseCommand
```

Setting Hard/Soft Joint Limits Dynamically

https://github.com/ros-controls/ros_control/wiki/joint_limits_interface

But of course, if you use moveit this is completely null and void! Beautiful modular code.

```
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_model/robot_model.h>

void GetJointLimits(std::vector<std::string> names, std::vector<double> lower, std::vector<double> upper) {
     boost::shared_ptr<robot_model_loader::RobotModelLoader>  urdf(new robot_model_loader::RobotModelLoader("robo


     moveit::core::JointModel* jm = urdf->getModel()->getJointModel("joint_1");
...
}
```

Mr Dave Coleman

https://github.com/davetcoleman?tab=repositories

Install rqt-moveit

http://wiki.ros.org/rqt/UserGuide/Install/Groovy

rostopic echo rviz_rufous_20055_8662231662545915181/motionplanning_planning_scene_monitor/parameter_descriptions

```
michalos@rufous:~/catkin_ws$ rostopic echo rviz_rufous_20055_8662231662545915181/motionplanning_planning_scene_mo
groups:
  -
    name: Default
    type: ''
    parameters:
      -
        name: publish_planning_scene
        type: bool
        level: 1
        description: Set to True to publish Planning Scenes
        edit_method: ''
      -
        name: publish_planning_scene_hz
        type: double
        level: 2
        description: Set the maximum frequency at which planning scene updates are published
        edit_method: ''
      -
        name: publish_geometry_updates
```

```
        type: bool
        level: 3
        description: Set to True to publish geometry updates of the planning scene
        edit_method: ''
      -
        name: publish_state_updates
        type: bool
        level: 4
        description: Set to True to publish geometry updates of the planning scene
        edit_method: ''
      -
        name: publish_transforms_updates
        type: bool
        level: 5
        description: Set to True to publish geometry updates of the planning scene
        edit_method: ''
    parent: 0
    id: 0
max:
  bools:
    -
      name: publish_planning_scene
      value: True
    -
      name: publish_geometry_updates
      value: True
    -
      name: publish_state_updates
      value: True
    -
      name: publish_transforms_updates
      value: True
  ints: []
  strs: []
  doubles:
    -
      name: publish_planning_scene_hz
      value: 100.0
  groups:
    -
      name: Default
      state: True
      id: 0
      parent: 0
min:
  bools:
    -
      name: publish_planning_scene
      value: False
    -
      name: publish_geometry_updates
      value: False
    -
      name: publish_state_updates
      value: False
    -
      name: publish_transforms_updates
      value: False
  ints: []
  strs: []
  doubles:
    -
      name: publish_planning_scene_hz
      value: 0.1
  groups:
    -
      name: Default
      state: True
```

```
      id: 0
      parent: 0
dflt:
  bools:
    -
      name: publish_planning_scene
      value: False
    -
      name: publish_geometry_updates
      value: True
    -
      name: publish_state_updates
      value: False
    -
      name: publish_transforms_updates
      value: False
  ints: []
  strs: []
  doubles:
    -
      name: publish_planning_scene_hz
      value: 4.0
  groups:
    -
      name: Default
      state: True
      id: 0
      parent: 0
---
```

catkin_make clean

problem with controller.o linking

rufous disk problem

Clean the disk fsck -As

Had to force a mount mount -o remount,rw / then deleted trash: cd ∼/michalos/.local/share/Trash rm -rf files/∗.∗

sudo apt-get autoremove (problem with kuka ros downloads)

Then removed excess 12.04 kernal images

```
cd /boot
sudo rm *.3.13.0-58*
```

github

git commit -a (added files) normally git commit . VIM nightmare: esc and :wq (write and quit)

```
michalos@rufous:~/github/usnistgov/el-robotics-core$ git push origin master
Username for 'https://github.com': johnmichaloski
Password for 'https://johnmichaloski@github.com':
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 264 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/usnistgov/el-robotics-core
   395d561..b74a274  master -> master
michalos@rufous:~/github/usnistgov/el-robotics-core$
```

Did not add nist_fanuc: 526 git add nist_fanuc 527 git commit . 528 git push origin master

git remote changes and incorporate into local repository

You cannot just "git fetch" remote changes, they will not be incorporated into your local repository unless you **merge** them. So must use "git merge remotebranchname"

```
https://help.github.com/articles/fetching-a-remote/

michalos@rufous:~/github/usnistgov/el-robotics-core$ git merge origin
Updating 5f50122..bfc7439
Fast-forward
 nist_fanuc/CMakeLists.txt           |   7 ++-
 nist_fanuc/scripts/runrvizdemo.bash |   2 +-
 nist_kitting/src/move_group.cpp     |  59 +------------------
 nist_kitting/src/mover.cpp          | 111 ++++++---------------------------
 ulapi/package.xml                   |   2 +-
 5 files changed, 20 insertions(+), 161 deletions(-)
michalos@rufous:~/github/usnistgov/el-robotics-core$
```

creating doxygen documentation

1) Install sudo apt-get install ros-indigo-rosdoc-lite 2) Run rosdoc-lite cd src/nist_fanuc rosdoc_lite ../nist_fanuc 3) Output in doc 4) cd doc/html then double click index.html

Problems with exclude, so hard coded doxygen ....

Chapter 2

Installation Readme

1) Set up ROS with Descartes and ROS Industrial

Copy repositories from https://github.com/ros-industrial:

```
fanuc
fanuc experimental
motoman
ros industrial core
```

Repositories from ROS-I Consortium https://github.com/ros-industrial-consortium github site:

```
descartes
descartes_tutorials
```

2) Installing Xerces c with Ubuntu

https://www.daniweb.com/hardware-and-software/linux-and-unix/threads/409769/ubuntu-11-10-xe
As far as I'm aware libxerces is the same as pretty much any other library in Debian based systems. It should be available in the repositories (the exact version will depend on which version of Ubuntu you're running).

You can use apt-get to install the packages for the library and the dev files. Then to use them in your C/C++ programs you simply #include the appropriate headers and link with the library when compiling/linking.

```
sudo apt-get update
apt-cache search libxerces
sudo apt-get install libxerces-c3.1 libxerces-c-dev
```

Need include file path CMakeLists.txt:

```
include_directories(/usr/include/xercesc)
```

Link library in CMakeLists.txt:

```
link_directories(/usr/lib/x86_64-linux-gnu/)
```

Need to link against libxerces.a in CMakeLists.txt:

```
target_link_libraries(nist_fanuc
libxerces-c.a
${catkin_LIBRARIES}
${Boost_LIBRARIES}
)
```

3) Installing CodeSynthesis XSD

http://www.codesynthesis.com/products/xsd/download.xhtml

1. Chose the linux deb install file that matches your computer (below 64 bit amd).
2. Download xsd_4.0.0-1_amd64.deb and it will say open with Ubuntu Software Center
3. Click to install, authenticate and add /usr/include/xsd/cxx/xml as include path.

Need include file path in CMakeLists.txt:

```
include_directories(/usr/include/xsd/cxx/xml)
```

Chapter 3

CJointReader

CJointReader is a reader which listens to sensor_msgs/JointState messages. The sensor_msgs/JointState describe the latest robot joint readers, either simulated or real joint values. The CJointReader class handles the subscription to the JointState message, the updating of the latest joint readings, and any responding to any queries for the latest joint states.

```
class CJointReader {
public:
    CJointReader(ros::NodeHandle &nh);
    sensor_msgs::JointState GetCurrentReadings();
    void Start();
    std::vector<double> GetJointValues();
    ////////////////////////////////////////////////////////////////
    void callback(const sensor_msgs::JointState::ConstPtr& msg);
    ros::NodeHandle &_nh;
    sensor_msgs::JointState _latestreading;
    sensor_msgs::JointState _lastreading;
    ros::Subscriber sub;
    // Not sure why you'd want to keep ring of last n readings ...
    std::vector<sensor_msgs::JointState> _readings;
    static boost::mutex _reader_mutex;
};
```

CJointReader class will listens for changes to JointState messages by "subscribing" to the joint_states message and providing callback routine (i.e., CJointReader::callback) to handle updates. This subscription and callback is initiated with the "Start" method. It depends on the constructor providing a ROS node handle, so it can call the central ROS services to subscribe to the message and receive Joint reading updates.

```
void CJointReader::Start() {
    sub = _nh.subscribe("joint_states", 10, &CJointReader::callback, this);
}
```

Per ROS subscribe interaction, there is a callback to receive the joint messages and record the latest position. void CJointReader::callback(const sensor_msgs::JointState::ConstPtr& msg) { boost::unique_lock<boost::mutex> scoped_lock (_reader_mutex); _lastreading=_latestreading; _latestreading.position.clear(); _latestreading.position.insert(_latestreading.position.begin(), msg->position.begin(), msg->position.end()); _latestreading.velocity = msg->velocity; _latestreading.effort = msg->effort; } To get the latest Joints value, a thread who has pointer to the joint reader, calls the method GetCurrentReadings() to retrieve the latest values.

```
sensor_msgs::JointState CJointReader::GetCurrentReadings() {
    boost::unique_lock<boost::mutex> scoped_lock(_reader_mutex);
    return _latestreading;
}
```

In general, one instance of a joint reader is shared throughout a program. Also, use of the boost shared pointer construct will handle reference counting and deletion of the heap object , so one declares one instance of a boost::shared_ptr<CJointReader> and passes it to any listeners.

```
boost::shared_ptr<CJointReader>jointReader = boost::shared_ptr<CJointReader>(new CJointReader(nh));
```

That's all there is to joint sensor reading. ROS only seems to support joint reading, so if you wanted the latest robot pose (position and orientation) you would need to compute it with the forward kinematics.

In the package.xml manifest file, which lists all the dependencies for the ROS package (build, install, run, etc). CJointReader does not really depend on Moveit!, (I think). However, its does depend on reading sensor_msgs messages that contain the "joint_states" so this the sensor_msg package is a build and runtime dependency. In the package.xml manifest, you will need to add:

```
<build_depend>sensor_msgs</build_depend>
...

<run_depend>sensor_msgs</run_depend>
And in the CMakeLists.txt file, which describes how to build the package, you will need to add:
find_package(catkin REQUIRED COMPONENTS
  moveit_core
  roscpp
  cmake_modules
  sensor_msgs
)
```

In my package, the joint readings are read once cyclically and updated.

Chapter 4

CJointWriter

CJointWriter is a class that handles updating the joint value to be displayed by RVIZ, or if a real robot is running, update the robot position.

```
class CJointWriter {
public:
    CJointWriter(ros::NodeHandle &nh);

    // Position only for now
    bool JointTrajectoryWrite(std::vector<sensor_msgs::JointState>);
    bool JointTrajectoryPositionWrite(sensor_msgs::JointState joint);
    ////////////////////////////////
    ros::Publisher traj_pub;
    std::vector<std::string> jointnames;
    static boost::mutex _writer_mutex;
};
```

CJointWriter constructor requires the ROS node handle, to advertise that it will be publishing joint values. In order to publish joint values, it must have the names of the joints that it will be updating. Thus, the command geParam with the parameter name controller_joint_names retrieves a list of the jointnames. Then, the constructor advertises to ROS that it will be publishing to the topic “joint_path_command”.

```
CJointWriter::CJointWriter(ros::NodeHandle &nh)
{
     nh.getParam("controller_joint_names", jointnames);
     // Trajectory publisher
    traj_pub = nh.advertise<trajectory_msgs::JointTrajectory>("joint_path_command", 1);
}
```

The JointTrajectoryPositionWrite method publishes updated joint values that the ROS system will publish to all listeners (which of interest in our case is RVIZ). It accepts a sensor_msg JointState structure containing the updated joint values. In theory, the joint_path_command topic could accept many points to display, however, only 1 point at a time is written to the topic.

```
bool CJointWriter::JointTrajectoryPositionWrite(sensor_msgs::JointState joint) {

    ActionGoal traj_goal;
    trajectory_msgs::JointTrajectory traj;
    std::vector<trajectory_msgs::JointTrajectoryPoint> points;
    size_t n_joints=joint.position.size();

    // Where we are going
    trajectory_msgs::JointTrajectoryPoint point;
    point.positions.resize(n_joints);
    point.positions=joint.position;
    point.velocities.resize(n_joints, 0.0);
    point.accelerations.resize(n_joints, 0.0);
```

```
    traj.joint_names = jointnames;
    traj.points.resize(1, point);
    // Send trajectory
    traj.header.stamp = ros::Time(0); // Start immediately
    traj_pub.publish(traj);
    return true;
}
```

The JointTrajectoryWrite method to updates a vector of joint values. (Unclear if this is useful.) It accepts a std vector of sensor_msg JointState values, and will update and then write each value (containing a vector of joint positions).

```
bool CJointWriter::JointTrajectoryWrite(std::vector<sensor_msgs::JointState> joints ) {

   for (size_t i = 0; i < joints.size(); i++) {
        JointTrajectoryPositionWrite(joints[i]);
    }
    return true;
}
```

The package.xml manifest contains all the following Moveit! entries, although it is unclear which ones are necessary.

```
    <build_depend>moveit_core</build_depend>
    <build_depend>moveit_ros_planning_interface</build_depend>
    <build_depend>moveit_ros_move_group</build_depend>
    <build_depend>moveit_ros_planning</build_depend>
    <build_depend>moveit_ros_manipulation</build_depend>

    <run_depend>moveit_core</run_depend>
    run_depend>moveit_ros_planning_interface</run_depend>
    <run_depend>moveit_ros_move_group</run_depend>
    <run_depend>moveit_ros_planning</run_depend>
    <run_depend>moveit_ros_manipulation</run_depend>
```

Likewise the CMakeLists.txt contains the following moveit entries. Of note, the trajectory_msgs

```
## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  moveit_core
  roscpp
  cmake_modules
  trajectory_msgs
  sensor_msgs
    moveit_ros_planning_interface
    moveit_ros_move_group
    moveit_ros_planning
    moveit_ros_manipulation
)

catkin_package(
  INCLUDE_DIRS
    include
  LIBRARIES
  CATKIN_DEPENDS
    roscpp
    moveit_core
    sensor_msgs
    moveit_ros_planning_interface
    moveit_ros_move_group
    moveit_ros_planning
    moveit_ros_manipulation
  DEPENDS
    Boost
    Eigen
```

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Namespace Documentation

9.1 Crcl Namespace Reference

Classes

- struct GripperStatus
- struct JointReport
- struct CrclStatus
- class CrclDelegateInterface
- class CrclClientCmdInterface
- class CrclStatusMsgInterface

Typedefs

- typedef urdf::Vector3 Vector3D

Enumerations

- enum CRCLCmdStatus { CRCL_DONE = 0, CRCL_WORKING, CRCL_ERROR }
- enum CrclReturn {
  CANON_REJECT = -2, CANON_FAILURE = -1, CANON_SUCCESS = 0, CANON_STATUSREPLY = 1,
  CANON_RUNNING }

Functions

- RosMatrix GetXZRotMatrix (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- Eigen::Matrix3d GetEigenRotMatrix (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- bool GetRPY (Crcl::PoseType pose, double &roll, double &pitch, double &yaw)
- bool GetRPY (urdf::Vector3 Xrot, urdf::Vector3 Zrot, double &roll, double &pitch, double &yaw)
- urdf::Pose Convert (Crcl::PoseType &pose, double lengthConversion)
- Crcl::VectorType VectorZero ()
- urdf::Vector3 GetVector3D (Crcl::PointType &point)
- urdf::Vector3 GetVector3D (Crcl::VectorType &vector)
- bool GetPoseToRPY (Crcl::PoseType &pose, double &dRoll, double &dPitch, double &dYaw)

- urdf::Rotation Convert (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- Crcl::PoseType Init (std::vector< double > terms)
- Crcl::PoseType Convert (urdf::Pose pose)
- sensor_msgs::JointState Convert (Crcl::JointStatusSequence jout, double angleConversion)
- Crcl::JointStatusSequence Convert (JointState joints, double _angleConversion)
- JointStatusSequence Convert (Crcl::ActuatorJointSequence joints, double _angleConversion)
- ::PointType GetPoint (RCS::Vector3 &point)
- ::VectorType GetVector (RCS::Vector3 &point)
- Crcl::PoseType NullPose ()
- Crcl::PoseType IdentityPose ()
- Crcl::PoseType PoseHome ()
- std::vector< double > ConvertToAnglePositionVector (Crcl::ActuatorJointSequence &joints, double dAngleConversion)
- std::string DumpCrclPose (Crcl::PoseType pose, std::string separator)
- std::string DumpPose (Crcl::PoseType pose, std::string separator)
- std::vector< double > ConvertToPositionVector (ActuatorJointSequence &, double dConversion)
- JointStatusSequence Convert (ActuatorJointSequence jin)
- Crcl::JointStatusSequence Convert (JointState joints)
- std::ostream & operator<< (std::ostream &os, const Crcl::PoseType &pose)

Variables

- typedef::ActuateJointsType::ActuateJoint_sequence ActuatorJointSequence
- typedef::PoseType PoseType
- typedef::JointStatusType JointStatus
- typedef::CommandStateEnumType CommandStateEnum
- typedef::PointType PointType
- typedef::VectorType VectorType
- typedef::JointStatusesType::JointStatus_sequence JointStatusSequence
- typedef::PoseToleranceType PoseToleranceType

9.1.1 Typedef Documentation

9.1.1.1 typedef urdf::Vector3 Crcl::Vector3D

9.1.2 Enumeration Type Documentation

9.1.2.1 enum Crcl::CRCLCmdStatus

Enumerator

CRCL_DONE

CRCL_WORKING

CRCL_ERROR

9.1.2.2 enum Crcl::CrclReturn

Enumerator

CANON_REJECT

CANON_FAILURE

CANON_SUCCESS

CANON_STATUSREPLY

CANON_RUNNING

9.1.3 Function Documentation

9.1.3.1 JointStatusSequence Crcl::Convert (ActuatorJointSequence *jin*)

9.1.3.2 Crcl::JointStatusSequence Crcl::Convert (JointState *joints*)

9.1.3.3 RCS::Pose Crcl::Convert (Crcl::PoseType & *pose,* double *lengthConversion*)

9.1.3.4 urdf::Rotation Crcl::Convert (urdf::Vector3 *Xrot,* urdf::Vector3 *Zrot*)

9.1.3.5 Crcl::PoseType Crcl::Convert (urdf::Pose *pose*)

9.1.3.6 sensor_msgs::JointState Crcl::Convert (Crcl::JointStatusSequence *jout,* double *angleConversion*)

9.1.3.7 Crcl::JointStatusSequence Crcl::Convert (JointState *joints,* double *_angleConversion*)

9.1.3.8 JointStatusSequence Crcl::Convert (Crcl::ActuatorJointSequence *joints,* double *_angleConversion*)

9.1.3.9 std::vector<double> Crcl::ConvertToAnglePositionVector (Crcl::ActuatorJointSequence & *joints,* double *dAngleConversion*)

9.1.3.10 std::vector<double> Crcl::ConvertToPositionVector (ActuatorJointSequence & , double *dConversion*)

9.1.3.11 std::string Crcl::DumpCrclPose (Crcl::PoseType *pose,* std::string *separator*)

9.1.3.12 std::string Crcl::DumpPose (Crcl::PoseType *pose,* std::string *separator*)

9.1.3.13 Eigen::Matrix3d Crcl::GetEigenRotMatrix (urdf::Vector3 *Xrot,* urdf::Vector3 *Zrot*)

9.1.3.14 ::PointType Crcl::GetPoint (RCS::Vector3 & *point*)

9.1.3.15 bool Crcl::GetPoseToRPY (Crcl::PoseType & *pose,* double & *dRoll,* double & *dPitch,* double & *dYaw*)

9.1.3.16 bool Crcl::GetRPY (Crcl::PoseType *pose,* double & *roll,* double & *pitch,* double & *yaw*)

9.1.3.17 bool Crcl::GetRPY (urdf::Vector3 *Xrot,* urdf::Vector3 *Zrot,* double & *roll,* double & *pitch,* double & *yaw*)

9.1.3.18 ::VectorType Crcl::GetVector (RCS::Vector3 & *point*)

9.1.3.19 urdf::Vector3 Crcl::GetVector3D (Crcl::PointType & *point*) [inline]

9.1.3.20 urdf::Vector3 Crcl::GetVector3D (Crcl::VectorType & *vector*) [inline]

9.1.3.21 RosMatrix Crcl::GetXZRotMatrix (urdf::Vector3 *Xrot,* urdf::Vector3 *Zrot*)

9.1.3.22 Crcl::PoseType Crcl::IdentityPose ( )

9.1.3.23 Crcl::PoseType Crcl::Init (std::vector< double > *terms*)

9.1.3.24 PoseType Crcl::NullPose ( )

9.1.3.25 std::ostream& Crcl::operator<< (std::ostream & *os,* const Crcl::PoseType & *pose*) [inline]

9.1.3.26 PoseType Crcl::PoseHome ( )

9.1.3.27 Crcl::VectorType Crcl::VectorZero ( ) [inline]

9.1.4 Variable Documentation

9.1.4.1 typedef::ActuateJointsType::ActuateJoint_sequence Crcl::ActuatorJointSequence

9.1.4.2 typedef::CommandStateEnumType Crcl::CommandStateEnum

9.1.4.3 typedef::JointStatusType Crcl::JointStatus

9.1.4.4 typedef::JointStatusesType::JointStatus_sequence Crcl::JointStatusSequence

9.1.4.5 typedef::PointType Crcl::PointType

9.1.4.6 typedef::PoseToleranceType Crcl::PoseToleranceType

9.1.4.7 typedef::PoseType Crcl::PoseType

9.1.4.8 typedef::VectorType Crcl::VectorType

9.2 NIST Namespace Reference

Functions

- void getRPY (const geometry_msgs::Quaternion &qmsg, double &roll, double &pitch, double &yaw)

9.2.1 Function Documentation

9.2.1.1 void NIST::getRPY (const geometry_msgs::Quaternion & *qmsg,* double & *roll,* double & *pitch,* double & *yaw*) [inline]

9.3 RCS Namespace Reference

Classes

- struct RdfJoint

- class ChainRobotModel
- struct CController

  The CController provides an collection for all the relevant controller pieces. The CController is the main controller class to collect all the references/pointers to instances in the project. A global instance, call Controller, is created that is used through out the code to reference various instances of control objects (e.g., kinematics, joint writer, joint reader, etc.)
- class RobotStatus

  The RobotStatus is a thread to updates the status of the robot. The RobotStatus is a separate thread that updates the robot status. Currently, it uses a JointReader to read joint values from the controller. It uses a Kinematics pointer reference to compute the current pose using the FK routine. It also uses a CrclDelegate pointer reference to update the status reported by CRCL.
- class RobotProgram

  The RobotProgram is a thread to handle crcl programs. Crcl programs are not in fact legitimate, however, debugging and verification are assisted by programs. However, program as in the Crcl XSD specification, so it doesn't hurt to handle. They require special handling as only one command should be done at a time. Uses codesynthesis to parse Crcl xml into C++ data structures.
- class CMessageQueue

  The CMessageQueue offers a mutexed front to a stl deque. The queue is a LIFO data structure. Useful for safely sharing data between multiple threads.
- class Thread

  Thread is an RCS ulapi equivalent for timed thread. Given a cycle time, the thread provides a wait function to sleep to exactly the amount of the thread cycle time. It keeps track of busy/idle time for diagnostic purposes.
  Notes: https://www.quantnet.com/threads/c-multithreading-in-boost.10028/.
- class Timer

  Timer is a general-purpose timer. The Timer is a general-purpose timer, which can be used for waiting until a synchronous time tick, slept on for any period at all, or to obtain a time in system clock ticks from creation of the timer.
- struct CanonCmd

  CanonCmd is the controller command structure.
- struct CanonWorldModel

  CanonWorldModel describes the controller state. Includes reference to robot model.

Typedefs

- typedef int(∗ RCS_TIMERFUNC)(void ∗_arg)
- typedef urdf::Pose Pose
- typedef urdf::Vector3 Position
- typedef urdf::Rotation Rotation
- typedef urdf::Vector3 Vector3
- typedef double Length
- typedef double LinearVelocity
- typedef double AngularVelocity
- typedef std::vector< double > robotAxes

Enumerations

- enum CanonLengthUnit { METER = 0, MM, INCH }

  enumeration of length units. Conversion into ROS compatible meters.
- enum TrajPointType { WAYPOINT = 1, GOAL }

  enumeration of trajector pose points.
- enum CanonAngleUnit { RADIAN = 0, DEGREE }

  enumeration of angle units. Conversion into ROS compatible radians.

- enum CanonForceUnit { NEWTON = 0, POUND, OUNCE }

  enumeration of force units.

- enum CanonTorqueUnit { NEWTONMETER = 0, FOOTPOUND }

  enumeration of torque units.

- enum CanonReturn {
  CANON_REJECT = -2, CANON_FAILURE = -1, CANON_SUCCESS = 0, CANON_STATUSREPLY = 1,
  CANON_RUNNING }

  enumeration of return type from Crcl intepretation. If statusreply, requires status sent to Crcl client.

- enum CanonCmdType {
  CANON_NOOP = 0, CANON_DWELL, CANON_END_CANON, CANON_INIT_CANON,
  CANON_MOVE_JOINT, CANON_MOVE_TO, CANON_MOVE_THRU, CANON_SET_MAX_CART_ACC,
  CANON_SET_MAX_CART_SPEED, CANON_SET_MAX_JOINT_ACC, CANON_SET_MAX_JOINT_SPEED,
  CANON_SET_GRIPPER,
  CANON_STOP_MOTION, CANON_UNKNOWN }

  enumeration of Crcl commands. Many Crcl commands are wm parameter setting and require no motion component.

- enum CanonStopMotionType { UNSET = -1, IMMEDIATE = 0, FAST, NORMAL }

  enumeration of stopping motion, e.g., estop equivalent to immediate.

- enum CanonAccProfile {
  MS_IS_UNSET = 0, MS_IS_DONE = 1, MS_IS_ACCEL = 2, MS_IS_CONST = 3,
  MS_IS_DECEL = 4, MS_IS_ESTOPPING = 5, MS_IS_PAUSED = 6 }

  enumeration of trajectory acceleration profile.

- enum MovementType { MOVE_DEFAULT = 0, MOVE_CARTESIAN, MOVE_JOINT }

  enumeration of trajectory motion type, joint or cartesian.

- enum CanonStatusType {
  CANON_DONE = 0, CANON_WORKING, CANON_PAUSED, CANON_ERROR,
  CANON_ABORT, CANON_WAITING }

  enumeration of controller status types for individual commands. Note, even though command types are listed, not all used or supported.

Functions

- ::CRCLProgramType::MiddleCommand_sequence & DummyInit ()
- template<class Rep , class Period >
  double ToNanoseconds (boost::chrono::duration< Rep, Period > d)
- template<class Rep , class Period >
  double ToSeconds (boost::chrono::duration< Rep, Period > d)
- std::string DumpPose (urdf::Pose &pose)

  DumpPose takes a urdf pose and generates a string describing pose. Can be used as std::cout << DumpPose(pose);.

- std::string DumpQuaterion (std::ostream &os, const urdf::Rotation &rot)

  DumpQuaterion takes a urdf quaterion and generates a string describing x,y,z,w coordinates. Can be used as std::cout << DumpQuaterion(urdf::rotation);.

Variables

- boost::mutex cncmutex
- RCS::CController Controller (DEFAULT_LOOP_CYCLE)

9.3.1 Typedef Documentation

9.3.1.1 typedef double RCS::AngularVelocity

9.3.1.2 typedef double RCS::Length

9.3.1.3 typedef double RCS::LinearVelocity

9.3.1.4 typedef urdf::Pose RCS::Pose

9.3.1.5 typedef urdf::Vector3 RCS::Position

9.3.1.6 typedef int(∗ RCS::RCS_TIMERFUNC)(void ∗_arg)

9.3.1.7 typedef std::vector<double> RCS::robotAxes

9.3.1.8 typedef urdf::Rotation RCS::Rotation

9.3.1.9 typedef urdf::Vector3 RCS::Vector3

9.3.2 Enumeration Type Documentation

9.3.2.1 enum RCS::CanonAccProfile

enumeration of trajectory acceleration profile.

Enumerator

MS_IS_UNSET

MS_IS_DONE

MS_IS_ACCEL

MS_IS_CONST

MS_IS_DECEL

MS_IS_ESTOPPING

MS_IS_PAUSED

9.3.2.2 enum RCS::CanonAngleUnit

enumeration of angle units. Conversion into ROS compatible radians.

Enumerator

RADIAN

DEGREE

9.3.2.3 enum RCS::CanonCmdType

enumeration of Crcl commands. Many Crcl commands are wm parameter setting and require no motion component.

Enumerator

CANON_NOOP
CANON_DWELL
CANON_END_CANON
CANON_INIT_CANON
CANON_MOVE_JOINT
CANON_MOVE_TO
CANON_MOVE_THRU
CANON_SET_MAX_CART_ACC
CANON_SET_MAX_CART_SPEED
CANON_SET_MAX_JOINT_ACC
CANON_SET_MAX_JOINT_SPEED
CANON_SET_GRIPPER
CANON_STOP_MOTION
CANON_UNKNOWN

9.3.2.4 enum RCS::CanonForceUnit

enumeration of force units.

Enumerator

NEWTON
POUND
OUNCE

9.3.2.5 enum RCS::CanonLengthUnit

enumeration of length units. Conversion into ROS compatible meters.

Enumerator

METER
MM
INCH

9.3.2.6 enum RCS::CanonReturn

enumeration of return type from Crcl intepretation. If statusreply, requires status sent to Crcl client.

Enumerator

CANON_REJECT
CANON_FAILURE
CANON_SUCCESS
CANON_STATUSREPLY
CANON_RUNNING

9.3.2.7 enum RCS::CanonStatusType

enumeration of controller status types for individual commands. Note, even though command types are listed, not all used or supported.

Enumerator

CANON_DONE

CANON_WORKING

CANON_PAUSED

CANON_ERROR

CANON_ABORT

CANON_WAITING

9.3.2.8 enum RCS::CanonStopMotionType

enumeration of stopping motion, e.g., estop equivalent to immediate.

Enumerator

UNSET

IMMEDIATE

FAST

NORMAL

9.3.2.9 enum RCS::CanonTorqueUnit

enumeration of torque units.

Enumerator

NEWTONMETER

FOOTPOUND

9.3.2.10 enum RCS::MovementType

enumeration of trajectory motion type, joint or cartesian.

Enumerator

MOVE_DEFAULT

MOVE_CARTESIAN

MOVE_JOINT

9.3.2.11 enum RCS::TrajPointType

enumeration of trajector pose points.

Enumerator

WAYPOINT

GOAL

9.3.3 Function Documentation

9.3.3.1 ::CRCLProgramType::MiddleCommand_sequence& RCS::DummyInit ( )

9.3.3.2 std::string RCS::DumpPose (urdf::Pose & *pose*) [inline]

DumpPose takes a urdf pose and generates a string describing pose. Can be used as std::cout << DumpPose(pose);.

9.3.3.3 std::string RCS::DumpQuaterion (std::ostream & *os*, const urdf::Rotation & *rot*) [inline]

DumpQuaterion takes a urdf quaterion and generates a string describing x,y,z,w coordinates. Can be used as std::cout << DumpQuaterion(urdf::rotation);.

9.3.3.4 template<class Rep , class Period > double RCS::ToNanoseconds (boost::chrono::duration< Rep, Period > *d*)

9.3.3.5 template<class Rep , class Period > double RCS::ToSeconds (boost::chrono::duration< Rep, Period > *d*)

9.3.4 Variable Documentation

9.3.4.1 boost::mutex RCS::cncmutex

9.3.4.2 CController RCS::Controller

global declaration of ONE controller

9.4 tf Namespace Reference

Functions

- void pointMsgToEigen (const geometry_msgs::Point &m, Eigen::Vector3d &e)
  Converts a Point message into an Eigen Vector.
- void pointEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Point &m)
  Converts an Eigen Vector into a Point message.
- void poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Affine3d &e)
  Converts a Pose message into an Eigen Affine3d.
- void poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Isometry3d &e)
  Converts a Pose message into an Eigen Isometry3d.
- void poseEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Pose &m)
  Converts an Eigen Affine3d into a Pose message.

- void poseEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Pose &m)

  Converts an Eigen Isometry3d into a Pose message.
- void quaternionMsgToEigen (const geometry_msgs::Quaternion &m, Eigen::Quaterniond &e)

  Converts a Quaternion message into an Eigen Quaternion.
- void quaternionEigenToMsg (const Eigen::Quaterniond &e, geometry_msgs::Quaternion &m)

  Converts an Eigen Quaternion into a Quaternion message.
- void transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Affine3d &e)

  Converts a Transform message into an Eigen Affine3d.
- void transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Isometry3d &e)

  Converts a Transform message into an Eigen Isometry3d.
- void transformEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Affine3d into a Transform message.
- void transformEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Isometry3d into a Transform message.
- void vectorMsgToEigen (const geometry_msgs::Vector3 &m, Eigen::Vector3d &e)

  Converts a Vector message into an Eigen Vector.
- void vectorEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Vector3 &m)

  Converts an Eigen Vector into a Vector message.
- void twistMsgToEigen (const geometry_msgs::Twist &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Twist message into an Eigen matrix.
- void twistEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Twist &m)

  Converts an Eigen matrix into a Twist message.
- void wrenchMsgToEigen (const geometry_msgs::Wrench &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Wrench message into an Eigen matrix.
- void wrenchEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Wrench &m)

  Converts an Eigen matrix into a Wrench message.
- template<class Derived >
  void matrixEigenToMsg (const Eigen::MatrixBase< Derived > &e, std_msgs::Float64MultiArray &m)

  Converts an Eigen matrix into a Float64MultiArray message.

9.4.1 Function Documentation

9.4.1.1 template<class Derived > void tf::matrixEigenToMsg (const Eigen::MatrixBase< Derived > & *e*, std_msgs::Float64MultiArray & *m*)

Converts an Eigen matrix into a Float64MultiArray message.

9.4.1.2 void tf::pointEigenToMsg (const Eigen::Vector3d & *e*, geometry_msgs::Point & *m*)

Converts an Eigen Vector into a Point message.

9.4.1.3 void tf::pointMsgToEigen (const geometry_msgs::Point & *m*, Eigen::Vector3d & *e*)

Converts a Point message into an Eigen Vector.

9.4.1.4 void tf::poseEigenToMsg (const Eigen::Affine3d & *e*, geometry_msgs::Pose & *m*)

Converts an Eigen Affine3d into a Pose message.

9.4.1.5 void tf::poseEigenToMsg (const Eigen::Isometry3d & *e*, geometry_msgs::Pose & *m*)

Converts an Eigen Isometry3d into a Pose message.

9.4.1.6 void tf::poseMsgToEigen (const geometry_msgs::Pose & *m*, Eigen::Affine3d & *e*)

Converts a Pose message into an Eigen Affine3d.

9.4.1.7 void tf::poseMsgToEigen (const geometry_msgs::Pose & *m*, Eigen::Isometry3d & *e*)

Converts a Pose message into an Eigen Isometry3d.

9.4.1.8 void tf::quaternionEigenToMsg (const Eigen::Quaterniond & *e*, geometry_msgs::Quaternion & *m*)

Converts an Eigen Quaternion into a Quaternion message.

9.4.1.9 void tf::quaternionMsgToEigen (const geometry_msgs::Quaternion & *m*, Eigen::Quaterniond & *e*)

Converts a Quaternion message into an Eigen Quaternion.

9.4.1.10 void tf::transformEigenToMsg (const Eigen::Affine3d & *e*, geometry_msgs::Transform & *m*)

Converts an Eigen Affine3d into a Transform message.

9.4.1.11 void tf::transformEigenToMsg (const Eigen::Isometry3d & *e*, geometry_msgs::Transform & *m*)

Converts an Eigen Isometry3d into a Transform message.

9.4.1.12 void tf::transformMsgToEigen (const geometry_msgs::Transform & *m*, Eigen::Affine3d & *e*)

Converts a Transform message into an Eigen Affine3d.

9.4.1.13 void tf::transformMsgToEigen (const geometry_msgs::Transform & *m*, Eigen::Isometry3d & *e*)

Converts a Transform message into an Eigen Isometry3d.

9.4.1.14 void tf::twistEigenToMsg (const Eigen::Matrix< double, 6, 1 > & *e*, geometry_msgs::Twist & *m*)

Converts an Eigen matrix into a Twist message.

9.4.1.15 void tf::twistMsgToEigen (const geometry_msgs::Twist & *m*, Eigen::Matrix< double, 6, 1 > & *e*)

Converts a Twist message into an Eigen matrix.

9.4.1.16 void tf::vectorEigenToMsg (const Eigen::Vector3d & *e*, geometry_msgs::Vector3 & *m*)

Converts an Eigen Vector into a Vector message.

9.4.1.17 void tf::vectorMsgToEigen (const geometry_msgs::Vector3 & *m*, Eigen::Vector3d & *e*)

Converts a Vector message into an Eigen Vector.

9.4.1.18 void tf::wrenchEigenToMsg (const Eigen::Matrix< double, 6, 1 > & *e*, geometry_msgs::Wrench & *m*)

Converts an Eigen matrix into a Wrench message.

9.4.1.19 void tf::wrenchMsgToEigen (const geometry_msgs::Wrench & *m*, Eigen::Matrix< double, 6, 1 > & *e*)

Converts a Wrench message into an Eigen matrix.

Chapter 10

Class Documentation

10.1 ALogger Class Reference

`#include <Logging.h>`

Public Member Functions

- ALogger ()
- void Close ()
- void Open (std::string filename, int bAppend=false)
- int LogMessage (std::string filename, std::string msg, int level=-1)
- int LogMessage (std::string msg, int level=-1)
- int Fatal (std::string msg)
- int Error (std::string msg)
- int Warning (std::string msg)
- int Info (std::string msg)
- int Debug (std::string msg)
- int Status (std::string msg)
- int & DebugLevel ()
- bool & Timestamping ()
- int & OutputConsole ()
- operator std::ostream & ()

Static Public Member Functions

- static std::string StrFormat (const char *fmt,...)
- static std::string FormatString (const char *fmt, va_list ap)
- static std::string Timestamp ()

Public Attributes

- int _debuglevel
- bool _bTimestamp
- std::ofstream DebugFile
- int _nOutputConsole
- std::string filename

10.1.1 Constructor & Destructor Documentation

10.1.1.1 ALogger::ALogger ( ) [inline]

10.1.2 Member Function Documentation

10.1.2.1 void ALogger::Close ( ) [inline]

10.1.2.2 int ALogger::Debug (std::string *msg*) [inline]

10.1.2.3 int& ALogger::DebugLevel ( ) [inline]

10.1.2.4 int ALogger::Error (std::string *msg*) [inline]

10.1.2.5 int ALogger::Fatal (std::string *msg*) [inline]

10.1.2.6 static std::string ALogger::FormatString (const char * *fmt,* va_list *ap*) [inline], [static]

10.1.2.7 int ALogger::Info (std::string *msg*) [inline]

10.1.2.8 int ALogger::LogMessage (std::string *filename,* std::string *msg,* int *level* = -1) [inline]

10.1.2.9 int ALogger::LogMessage (std::string *msg,* int *level* = -1) [inline]

10.1.2.10 void ALogger::Open (std::string *filename,* int *bAppend* = false) [inline]

10.1.2.11 ALogger::operator std::ostream & ( ) [inline]

10.1.2.12 int& ALogger::OutputConsole ( ) [inline]

10.1.2.13 int ALogger::Status (std::string *msg*) [inline]

10.1.2.14 static std::string ALogger::StrFormat (const char * *fmt,* ...) [inline], [static]

10.1.2.15 static std::string ALogger::Timestamp ( ) [inline], [static]

10.1.2.16 bool& ALogger::Timestamping ( ) [inline]

10.1.2.17 int ALogger::Warning (std::string *msg*) [inline]

10.1.3 Member Data Documentation

10.1.3.1 bool ALogger::_bTimestamp

10.1.3.2 int ALogger::_debuglevel

10.1.3.3 int ALogger::_nOutputConsole

10.1.3.4 std::ofstream ALogger::DebugFile

10.1.3.5 std::string ALogger::filename

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Logging.h

10.2 RCS::CanonCmd Struct Reference

CanonCmd is the controller command structure.

`#include <RCS.h>`

Public Member Functions

- CanonCmd ()

  CanonCmd constructor.
- void Init ()
- VAR (CommandID, unsigned long long)
- VAR (ParentCommandID, unsigned long long)
- VAR (StatusID, unsigned long long)

Public Attributes

- CanonCmdType cmd
- CanonStatusType status
- TrajPointType type
- CanonStopMotionType stoptype
- bool bCoordinated
- bool bStraight
- double absTransAcc
- double absTransSpeed
- double absRotAcc
- double absRotSpeed
- double absJointAcc
- double absJointSpeed
- double dwell
- double gripperPos
- CanonAccProfile accprofile
- std::vector< double > speed
- std::vector< int > jointnum
- JointState joints
- urdf::Pose pose
- urdf::Pose tolerance
- std::vector< urdf::Pose > waypoints

Static Public Attributes

- static unsigned long long _cmdid =0

10.2.1 Detailed Description

CanonCmd is the controller command structure.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 RCS::CanonCmd::CanonCmd ( ) [inline]

CanonCmd constructor.

10.2.3 Member Function Documentation

10.2.3.1 void RCS::CanonCmd::Init ( )

10.2.3.2 RCS::CanonCmd::VAR (CommandID , unsigned long *long*)

10.2.3.3 RCS::CanonCmd::VAR (ParentCommandID , unsigned long *long*)

10.2.3.4 RCS::CanonCmd::VAR (StatusID , unsigned long *long*)

10.2.4 Member Data Documentation

10.2.4.1 unsigned long long RCS::CanonCmd::_cmdid =0 [static]

10.2.4.2 double RCS::CanonCmd::absJointAcc

joint max acceleration

10.2.4.3 double RCS::CanonCmd::absJointSpeed

joint max velocity

10.2.4.4 double RCS::CanonCmd::absRotAcc

cartesian rotation acceleration

10.2.4.5 double RCS::CanonCmd::absRotSpeed

cartesian rotation velocity

10.2.4.6 double RCS::CanonCmd::absTransAcc

cartesian translational acceleration

10.2.4.7 double RCS::CanonCmd::absTransSpeed

cartesian translational velocity

10.2.4.8 CanonAccProfile RCS::CanonCmd::accprofile

current trajectory acceleration profile

10.2.4.9 bool RCS::CanonCmd::bCoordinated

coordinated joint trajectory motion boolean

10.2.4.10 bool RCS::CanonCmd::bStraight

straigth cartesian trajectory motion boolean

10.2.4.11 CanonCmdType RCS::CanonCmd::cmd

command type

10.2.4.12 double RCS::CanonCmd::dwell

time for dwelling in seconds

10.2.4.13 double RCS::CanonCmd::gripperPos

gripper position 0 to 1

10.2.4.14 std::vector<int> RCS::CanonCmd::jointnum

vector of joint numbers used by command

10.2.4.15 JointState RCS::CanonCmd::joints

commanded joint state

10.2.4.16 urdf::Pose RCS::CanonCmd::pose

commanded pose state

10.2.4.17 std::vector<double> RCS::CanonCmd::speed

vector of joint velocities

10.2.4.18 CanonStatusType RCS::CanonCmd::status

status type

10.2.4.19 CanonStopMotionType RCS::CanonCmd::stoptype

stop trajectory choice

10.2.4.20 urdf::Pose RCS::CanonCmd::tolerance

commanded tolerance

10.2.4.21 TrajPointType RCS::CanonCmd::type

trajectory points type

10.2.4.22 std::vector<urdf::Pose> RCS::CanonCmd::waypoints

commanded cartesian waypoints in trajectory

The documentation for this struct was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RCS.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/RCS.cpp

10.3 RCS::CanonWorldModel Struct Reference

CanonWorldModel describes the controller state. Includes reference to robot model.

`#include <RCS.h>`

Public Member Functions

- CanonWorldModel ()
  CanonWorldModel constructor that initializes parameterization.
- void Init ()
- double getCycleTime ()
  Cycletime of the world model. /fixme what is this.

Public Attributes

- CanonCmdType echo_cmd
- CanonStatusType echo_status
- ModelInterfaceSharedPtr robot_model
- double maxTransAccel
- double maxTransVel
- double maxRotAccel
- double maxRotVel
- double maxJointAccel
- double maxJointVel
- double _cycleTime
- CanonCmd echocmd
- JointState currentjoints
- urdf::Pose currentpose

10.3.1 Detailed Description

CanonWorldModel describes the controller state. Includes reference to robot model.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 RCS::CanonWorldModel::CanonWorldModel ( ) [inline]

CanonWorldModel constructor that initializes parameterization.

10.3.3 Member Function Documentation

10.3.3.1 double RCS::CanonWorldModel::getCycleTime ( ) [inline]

Cycletime of the world model. /fixme what is this.

10.3.3.2 void RCS::CanonWorldModel::Init ( )

10.3.4 Member Data Documentation

10.3.4.1 double RCS::CanonWorldModel::_cycleTime

cycle time

10.3.4.2 JointState RCS::CanonWorldModel::currentjoints

current joint state

10.3.4.3 urdf::Pose RCS::CanonWorldModel::currentpose

current robot pose

10.3.4.4 CanonCmdType RCS::CanonWorldModel::echo_cmd

copy of current command type

10.3.4.5 CanonStatusType RCS::CanonWorldModel::echo_status

copy of current status type

10.3.4.6 CanonCmd RCS::CanonWorldModel::echocmd

copy of current command

10.3.4.7 double RCS::CanonWorldModel::maxJointAccel

max joint acceleration

10.3.4.8 double RCS::CanonWorldModel::maxJointVel

max joint velocity

10.3.4.9 double RCS::CanonWorldModel::maxRotAccel

max rotational acceleration

10.3.4.10 double RCS::CanonWorldModel::maxRotVel

max rotational velocity

10.3.4.11 double RCS::CanonWorldModel::maxTransAccel

max translation acceleration

10.3.4.12 double RCS::CanonWorldModel::maxTransVel

max translation velocity

10.3.4.13 ModelInterfaceSharedPtr RCS::CanonWorldModel::robot_model

pointer to robot model

The documentation for this struct was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RCS.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/RCS.cpp

10.4 CAsioCrclServer Class Reference

The CAsioCrclServer provides an boost asio server which accepts new connections and starts a Crcl listener session. The CAsioCrclServer is based on the Boost Asio library which can process network communication asynchronously. Because CRCL data can only be received after a connection has been established, and because a connection can only be established after the name has been resolved, the various asynchronous operations are started in separate callback handlers. Thus in boost asio a callback to async_connect() is then followed by a method call to the handler connect-_handler() which starts a new Crcl session. Readers can read more at: http://theboostcpplibraries.-com/boost.asio-network-programming The CAsioCrclServer is divided into a number of main funcitons (e.g. wait for socket connection, handle new session by spawning new CAsioCrclSession, repeat. These operations are done asynchronously on a separate thread with notification done by the boost asio io server and it is assumed to be thread-safe. The CAsioCrclServer listens for connections on port 64444 and when a connection is initiated starts a new Crcl session to read xml messages from the devices.

```
#include <AsioCrclServer.h>
```

Public Member Functions

- CAsioCrclServer (boost::asio::io_service &io_service)

Constructor for asio crcl server that listens on socket port 64444, and spawns a new session.

- void server (boost::asio::io_service &io_service, short port)
- void HandleAsyncAccept (session_ptr pSession, const boost::system::error_code &error)

  Handles new tcp/ip endpoint connection, and starts session thread.Restarts async accept.
- void StartAsyncAccept ()

  Starts asio acceptor to wait for connections.
- void StopAsyncAccept ()

  Stop async accept by cancelling asio acceptor.
- int Init (std::string domain, long portnumber, std::string devicename)

  initializes the server.
- int Stop (void)

  Stop all connection session and future connections.
- void Start ()

  start the asynchronous thread to listen for connections.

Public Attributes

- boost::asio::io_service & _io_service
- tcp::acceptor * m_pAcceptor
- bool _bInited
- int _portnumber
- std::string _domainname
- std::string _deviceName
- bool _bLastConnected
- unsigned int _nHeartbeat
- unsigned int(* ErrorMessage)(std::string)

Static Public Attributes

- static bool bRunning = true
- static int nCount = 0
- static bool _bTrace = false

10.4.1 Detailed Description

The CAsioCrclServer provides an boost asio server which accepts new connections and starts a Crcl listener session. The CAsioCrclServer is based on the Boost Asio library which can process network communication asynchronously. Because CRCL data can only be received after a connection has been established, and because a connection can only be established after the name has been resolved, the various asynchronous operations are started in separate callback handlers. Thus in boost asio a callback to async_connect() is then followed by a method call to the handler connect-_handler() which starts a new Crcl session. Readers can read more at: http://theboostcpplibraries.-com/boost.asio-network-programming The CAsioCrclServer is divided into a number of main funcitons (e.g. wait for socket connection, handle new session by spawning new CAsioCrclSession, repeat. These operations are done asynchronously on a separate thread with notification done by the boost asio io server and it is assumed to be thread-safe. The CAsioCrclServer listens for connections on port 64444 and when a connection is initiated starts a new Crcl session to read xml messages from the devices.

-

10.4.2 Constructor & Destructor Documentation

10.4.2.1 CAsioCrclServer::CAsioCrclServer (boost::asio::io_service & *io_service*)

Constructor for asio crcl server that listens on socket port 64444, and spawns a new session.

Parameters

io_service	reference tot he asio service providers. only one per program.

10.4.3 Member Function Documentation

10.4.3.1 void CAsioCrclServer::HandleAsyncAccept (session_ptr *pSession,* const boost::system::error_code & *error*)

Handles new tcp/ip endpoint connection, and starts session thread.Restarts async accept.

Parameters

pSession	is the latest session pointer associated with connection.
error	is asio error if any.

10.4.3.2 int CAsioCrclServer::Init (std::string *domain,* long *portnumber,* std::string *devicename*)

initializes the server.

Parameters

domain	gives the domain name (usually localhost).
portnumber	is asio socket port number (usually 64444).
devicename	is the name of the device running this server.

10.4.3.3 void CAsioCrclServer::server (boost::asio::io_service & *io_service,* short *port*)

brief Creates acceptor for tcp/ip endpoint, and starts async accept.

Parameters

io_service	is asio service.
port	is tcp port to listen for connections on.

10.4.3.4 void CAsioCrclServer::Start ( )

start the asynchronous thread to listen for connections.

10.4.3.5 void CAsioCrclServer::StartAsyncAccept ( )

Starts asio acceptor to wait for connections.

10.4.3.6 int CAsioCrclServer::Stop (void)

Stop all connection session and future connections.

Returns

error or ok.

10.4.3.7 void CAsioCrclServer::StopAsyncAccept ( )

Stop async accept by cancelling asio acceptor.

10.4.4 Member Data Documentation

10.4.4.1 bool CAsioCrclServer::_bInited

server initialized flag

10.4.4.2 bool CAsioCrclServer::_bLastConnected

10.4.4.3 bool CAsioCrclServer::_bTrace = false [static]

trace input messages

10.4.4.4 std::string CAsioCrclServer::_deviceName

copy of device name

10.4.4.5 std::string CAsioCrclServer::_domainname

copy of domain name

10.4.4.6 boost::asio::io_service& CAsioCrclServer::_io_service

reference to asio io server to be passed to each session

10.4.4.7 unsigned int CAsioCrclServer::_nHeartbeat

heartbeat incremented to show server alive

10.4.4.8 int CAsioCrclServer::_portnumber

copy of socket port number to listen to

10.4.4.9 bool CAsioCrclServer::bRunning = true [static]

boolean that all sessions monitor for termination

10.4.4.10 unsigned int(* CAsioCrclServer::ErrorMessage)(std::string)

10.4.4.11 tcp::acceptor* CAsioCrclServer::m_pAcceptor

pointer to asio acceptor

10.4.4.12 int CAsioCrclServer::nCount = 0 [static]

count of active sessions

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/AsioCrclServer.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/AsioCrclServer.cpp

10.5 CAsioCrclSession Class Reference

The CAsioCrclSession provides an boost asio session (which listens for each connected client). The CAsioCrclSession listens for XML messages and constructs. The CAsioCrclSession uses mostly asynchronous operation for waiting, reading, and timeout of a socket connection. The operation is started by creating a session which starts an aynchronous thread, that is supplied IO communication events by the asio io service provider. After connection to the socket client, an StartAyncRead() that is paired with a timer is used to wait for communicatin from a socket. There is no trailing marker on CRCL XML so any socket communication must be buffered and when a complete message has been received, it is pushed onto the inmsgs message queue. During the socket communication, a timeout can occur, which at this point only causes a new to beStartAyncRead() initiated. Because CRCL Xml does not have a trailing marker (e.g., zero or line feed), the CAsioCrclSession must determine the trailing XML tag to search for, by inspecting the communication for a XML leading tag. It works, but is dubious. However, if the communicating socket is disconnected, an error is returned by asio, and the session is terminated cleanly.

Useful web sites:

```
#include <AsioCrclServer.h>
```

Public Member Functions

- CAsioCrclSession (boost::asio::io_service &io_service)

  Constructor for each listener on the socket.
- void AppendBuffer (std::string read)

  Appends a socket buffer. Add previous buffer is exists.
- void BufferHandler (std::string &endtag)

  Looks for matching end xml tag. If found, saves message into queue, and restarts read process.
- void Disconnect ()

  Disconnect timer, socket, and running flags.
- std::string FindLeadingElement (std::string xml)

  Finds the leading XML tag to create matching end tag. If none, return nonsense tag. Uses boost regex. xml is the text to search for starting tag.
- size_t HandleRead (const error_code &error, size_t bytes_read)

  Handles notification from asio via socket or timeout error or other error.
- virtual void Session ()

  For each connection a new Session is started.

- void SaveMessage (std::string xmlmessage)

  Queues message onto message queue.
- tcp::socket & Socket ()

  Return the TCP/IP socket from this session.
- void StartAyncRead ()

  Starts async read operation. Deadline timer going at same time.
- void SyncWrite (std::string str)

  Handles synchronous socket write of string.
- void TagReset ()

  resets the XML ending tag to be non-matchable nonsense.
- void TimerReset ()

  Restarts asio timeout timer for async read. Set at 2 seconds. Invokes callback if timer expires.

Static Public Member Functions

- static std::string NonsenseTag ()

  NonsenseTag to be used as dummy ending xml to test against.
- static void Join (CAsioCrclSession ∗device)

  Keeps track of devices that created an asio session.
- static void Leave (CAsioCrclSession ∗device)

  Keeps track of devices that left and are no longer an asio session.
- static CAsioMessages & InMessages ()

  Keeps track of devices that left and are no longer an asio session.

Protected Types

- enum { max_length = 4096 }

Protected Attributes

- boost::condition_variable cMessage
- boost::mutex condMutex
- tcp::socket _socket
- boost::asio::deadline_timer _timer
- std::string _current
- std::string _next
- std::string _endtag
- char data_ [max_length]
- unsigned int(∗ ErrorMessage)(std::string)
- bool _bRunning
- friend CAsioCrclServer

Static Protected Attributes

- static std::set
  < CAsioCrclSession ∗ > _devices
- static CAsioMessages _inmsgs
- static CAsioMessages _outmsgs

10.5.1 Detailed Description

The CAsioCrclSession provides an boost asio session (which listens for each connected client). The CAsioCrclSession listens for XML messages and constructs. The CAsioCrclSession uses mostly asynchronous operation for waiting, reading, and timeout of a socket connection. The operation is started by creating a session which starts an aynchronous thread, that is supplied IO communication events by the asio io service provider. After connection to the socket client, an StartAyncRead() that is paired with a timer is used to wait for communicatin from a socket. There is no trailing marker on CRCL XML so any socket communication must be buffered and when a complete message has been received, it is pushed onto the inmsgs message queue. During the socket communication, a timeout can occur, which at this point only causes a new to beStartAyncRead() initiated. Because CRCL Xml does not have a trailing marker (e.g., zero or line feed), the CAsioCrclSession must determine the trailing XML tag to search for, by inspecting the communication for a XML leading tag. It works, but is dubious. However, if the communicating socket is disconnected, an error is returned by asio, and the session is terminated cleanly.

Useful web sites:

asio read socket

http://stackoverflow.com/questions/4933610/boostasio-async-read-guarantee-all-bytes-are-rea

http://www.tagwith.com/question_285175_when-do-i-call-boostasiostreambufconsume-and-boostas

http://www.dahuatu.com/jdyjaPxKWQ.html

https://issues.apache.org/jira/browse/THRIFT-311

http://pastebin.com/Li3wbpvu

http://geekswithblogs.net/JoshReuben/archive/2014/11/17/c-boost-in-a-nutshell.-aspx

http://pastebin.com/YXk9stVA

http://thisthread.blogspot.com/2013/09/simple-asio-tcp-clientserver-example.-html

http://dolinked.com/questions/4792923/uninitialized-read-error-on-asio-tcp-socket

http://onnerby.se/~daniel/mC2/trunk/docs/core/html/_socket_8h-source.html

http://en.pudn.com/downloads245/sourcecode/internet/proxy/detail1143109_en.-html

http://www.gamedev.net/topic/566670-c-implementing-ssl/

http://cpp.knowcoding.com/view/20682-tcp-socket.html

Buffering:

http://stackoverflow.com/questions/4294651/sending-an-xml-message-in-parts-through-a-tcp-so

http://www.codesynthesis.com/pipermail/xsde-users/2014-January/000631.html

Raw:

http://www.bogotobogo.com/cplusplus/sockets_server_client.php

10.5.2 Member Enumeration Documentation

10.5.2.1 anonymous enum [protected]

Enumerator

max_length

10.5.3 Constructor & Destructor Documentation

10.5.3.1 CAsioCrclSession::CAsioCrclSession (boost::asio::io_service & *io_service*)

Constructor for each listener on the socket.

Parameters

io_service	reference tot he asio service providers. only one per program.

10.5.4 Member Function Documentation

10.5.4.1 void CAsioCrclSession::AppendBuffer (std::string *read*)

Appends a socket buffer. Add previous buffer is exists.

Parameters

read	buffer of characters

10.5.4.2 void CAsioCrclSession::BufferHandler (std::string & *endtag*)

Looks for matching end xml tag. If found, saves message into queue, and restarts read process.

*

Parameters

endtag	is the ending tag, e.g., </ENDTAG to match against. Includes backslash.

10.5.4.3 void CAsioCrclSession::Disconnect ( )

Disconnect timer, socket, and running flags.

-

10.5.4.4 std::string CAsioCrclSession::FindLeadingElement (std::string *xml*)

Finds the leading XML tag to create matching end tag. If none, return nonsense tag. Uses boost regex. xml is the text to search for starting tag.

Returns

end tag or nonsense tag if none. e.g., </TAG>

10.5.4.5 size_t CAsioCrclSession::HandleRead (const error_code & *error,* size_t *bytes_read*)

Handles notification from asio via socket or timeout error or other error.

Parameters

error	is potential communiationerror.
result	is the buffer size of the socket read.

Returns

size of buffer read.

10.5.4.6 static CAsioMessages& CAsioCrclSession::InMessages ( ) [inline], [static]

Keeps track of devices that left and are no longer an asio session.

10.5.4.7 static void CAsioCrclSession::Join (CAsioCrclSession * *device*) [inline], [static]

Keeps track of devices that created an asio session.

10.5.4.8 static void CAsioCrclSession::Leave (CAsioCrclSession * *device*) [inline], [static]

Keeps track of devices that left and are no longer an asio session.

10.5.4.9 static std::string CAsioCrclSession::NonsenseTag ( ) [inline], [static]

NonsenseTag to be used as dummy ending xml to test against.

10.5.4.10 void CAsioCrclSession::SaveMessage (std::string *xmlmessage*)

Queues message onto message queue.

Parameters

xmlmessage	to queue onto this session message queue.

10.5.4.11 void CAsioCrclSession::Session ( ) [virtual]

For each connection a new Session is started.

10.5.4.12 tcp::socket& CAsioCrclSession::Socket ( ) [inline]

Return the TCP/IP socket from this session.

10.5.4.13 void CAsioCrclSession::StartAyncRead ( )

Starts async read operation. Deadline timer going at same time.

10.5.4.14 void CAsioCrclSession::SyncWrite (std::string *str*)

Handles synchronous socket write of string.

Parameters

str	is the string to write out on the socket.

10.5.4.15 void CAsioCrclSession::TagReset ( ) [inline]

resets the XML ending tag to be non-matchable nonsense.

10.5.4.16 void CAsioCrclSession::TimerReset ( )

Restarts asio timeout timer for async read. Set at 2 seconds. Invokes callback if timer expires.

10.5.5 Member Data Documentation

10.5.5.1 bool CAsioCrclSession::_bRunning [protected]

boolean running loop flag

10.5.5.2 std::string CAsioCrclSession::_current [protected]

current string read from socket

10.5.5.3 std::set< CAsioCrclSession * > CAsioCrclSession::_devices [static], [protected]

list of devices being listened to

10.5.5.4 std::string CAsioCrclSession::_endtag [protected]

endtag to designate the end of Crcl XML message, found from beginning

10.5.5.5 CAsioMessages CAsioCrclSession::_inmsgs [static], [protected]

queue of inbound crcl xml messages from device

10.5.5.6 std::string CAsioCrclSession::_next [protected]

leftover string after pulling out Crcl XML message

10.5.5.7 CAsioMessages CAsioCrclSession::_outmsgs [static], [protected]

queue of outbound crcl xml messages to device

10.5.5.8 tcp::socket CAsioCrclSession::_socket [protected]

tcp/ip asio socket

10.5.5.9 boost::asio::deadline_timer CAsioCrclSession::_timer [protected]

socket reader timer

10.5.5.10 friend CAsioCrclSession::CAsioCrclServer [protected]

10.5.5.11 boost::condition_variable CAsioCrclSession::cMessage [protected]

10.5.5.12 boost::mutex CAsioCrclSession::condMutex [protected]

mutex to

10.5.5.13 char CAsioCrclSession::data_[max_length] [protected]

asio tcp/ip character read buffer

10.5.5.14 unsigned int(* CAsioCrclSession::ErrorMessage)(std::string) [protected]

function pointer to error message emitter

The documentation for this class was generated from the following files:

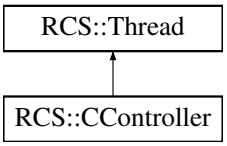
- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/AsioCrclServer.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/AsioCrclServer.cpp

10.6 RCS::CController Struct Reference

The CController provides an collection for all the relevant controller pieces. The CController is the main controller class to collect all the references/pointers to instances in the project. A global instance, call Controller, is created that is used through out the code to reference various instances of control objects (e.g., kinematics, joint writer, joint reader, etc.)

`#include <Controller.h>`

Inheritance diagram for RCS::CController:



Public Types

- enum DebugLevel { FATAL = 0, WARNING = 2, INFORM = 4, FULL = 5 }
- enum DebugType { CRCL = 0, RPY }
- enum MotionPlannerEnum {
  NOPLANNER = 0, MOVEIT, DESCARTES, BASIC,
  WAYPOINT, GOMOTION }
- typedef std::list< RCS::CanonCmd > xml_message_list

Public Member Functions

- CController (double cycletime)

  CController constructor that requires a cycle time for RCS thread timing.
- ~CController (void)
- bool Verify ()

  Verifies that all the pointer references in the controller have been instantiated (i.e., not null).
- virtual void Action ()

  Cyclic loop for the controller. Reads Crcl input mexsage queue, interprets into canon cmds if any, reads canon cmds queue, interprets into robot command messages.
- virtual void Init ()

  Initialization routine for the controller..
- std::string Dump (std::string separator=",")

  Creates a comma separated string of current state of robot. (Can use other separator).
- std::string DumpHeader (std::string separator=",")

  Creates a header line containing names of comma separated string fields that describes the current state of robot. (Can use other separator).
- NVAR (CrclDelegate, boost::shared_ptr< Crcl::CrclDelegateInterface >, crclinterface)
- VAR (Kinematics, boost::shared_ptr< IKinematics >)
- VAR (TrajectoryModel, boost::shared_ptr< CTrajectory >)
- VAR (JointWriter, boost::shared_ptr< CJointWriter >)
- VAR (MoveitPlanner, boost::shared_ptr< MoveitPlanning >)
- void SetKinematics (boost::shared_ptr< IKinematics > k)

  Routine to set the kinematics reference pointer. Uses the interface class IKinematics, but can have any implementation instance.

Public Attributes

- RCSInterpreter _interpreter
- RCS::CanonCmd _newcc
- RCS::CanonCmd _lastcc
- std::string lastlogstatus
- MotionPlannerEnum eCartesianMotionPlanner
- MotionPlannerEnum eJointMotionPlanner

Static Public Attributes

- static bool bSimulation = true
- static RCS::CanonWorldModel wm
- static RCS::CanonWorldModel status
- static RCS::CanonWorldModel laststatus
- static RCS::CMessageQueue < RCS::CanonCmd > cmds
- static xml_message_list donecmds
- static RCS::CMessageQueue < RCS::CanonCmd > robotcmds
- static size_t _NumJoints
- static bool bGenerateProgram = false
- static unsigned long _debugtype = (unsigned long) RPY
- static unsigned long _debuglevel = 0
- static unsigned long _csvlogFlag = 0
- static ALogger CsvLogging

Additional Inherited Members

10.6.1 Detailed Description

The CController provides an collection for all the relevant controller pieces. The CController is the main controller class to collect all the references/pointers to instances in the project. A global instance, call Controller, is created that is used through out the code to reference various instances of control objects (e.g., kinematics, joint writer, joint reader, etc.)

10.6.2 Member Typedef Documentation

10.6.2.1 typedef std::list<RCS::CanonCmd> RCS::CController::xml_message_list

10.6.3 Member Enumeration Documentation

10.6.3.1 enum RCS::CController::DebugLevel

Enumerator

FATAL

WARNING

INFORM

FULL

10.6.3.2 enum RCS::CController::DebugType

Enumerator

CRCL

RPY

10.6.3.3 enum RCS::CController::MotionPlannerEnum

Enumerator

NOPLANNER

MOVEIT

DESCARTES

BASIC

WAYPOINT

GOMOTION

10.6.4 Constructor & Destructor Documentation

10.6.4.1 RCS::CController::CController (double *cycletime*)

CController constructor that requires a cycle time for RCS thread timing.

Parameters

cycletime	in seconds.

10.6.4.2 RCS::CController::~CController (void)

10.6.5 Member Function Documentation

10.6.5.1 void RCS::CController::Action ( ) [virtual]

Cyclic loop for the controller. Reads Crcl input mexsage queue, interprets into canon cmds if any, reads canon cmds queue, interprets into robot command messages.

Reimplemented from RCS::Thread.

10.6.5.2 std::string RCS::CController::Dump (std::string *separator* = " , ")

Creates a comma separated string of current state of robot. (Can use other separator).

10.6.5.3 std::string RCS::CController::DumpHeader (std::string *separator* = " , ")

Creates a header line containing names of comma separated string fields that describes the current state of robot. (Can use other separator).

10.6.5.4 void RCS::CController::Init ( ) [virtual]

Initialization routine for the controller..

Reimplemented from RCS::Thread.

10.6.5.5 RCS::CController::NVAR (CrclDelegate , boost::shared_ptr< Crcl::CrclDelegateInterface > , crclinterface)

10.6.5.6 void RCS::CController::SetKinematics (boost::shared_ptr< IKinematics > *k*) [inline]

Routine to set the kinematics reference pointer. Uses the interface class IKinematics, but can have any implementation instance.

10.6.5.7 RCS::CController::VAR (Kinematics , boost::shared_ptr< IKinematics >)

10.6.5.8 RCS::CController::VAR (TrajectoryModel , boost::shared_ptr< CTrajectory >)

10.6.5.9 RCS::CController::VAR (JointWriter , boost::shared_ptr< CJointWriter >)

10.6.5.10 RCS::CController::VAR (MoveitPlanner , boost::shared_ptr< MoveitPlanning >)

10.6.5.11 bool RCS::CController::Verify ( )

Verifies that all the pointer references in the controller have been instantiated (i.e., not null).

10.6.6 Member Data Documentation

10.6.6.1 unsigned long RCS::CController::_csvlogFlag = 0 [static]

10.6.6.2 unsigned long RCS::CController::_debuglevel = 0 [static]

level of debugging, 0 least, 5 most

10.6.6.3 unsigned long RCS::CController::_debugtype = (unsigned long) RPY [static]

output crcl xz rotation or roll,pitch, yaw

10.6.6.4 RCSInterpreter RCS::CController::_interpreter

interprets canon commands into robot commands

10.6.6.5 RCS::CanonCmd RCS::CController::_lastcc

last canon command interpreted

10.6.6.6 RCS::CanonCmd RCS::CController::_newcc

current new canon command to interpret

10.6.6.7 size_t RCS::CController::_NumJoints [static]

number of joints in controller robot - assuming serial link manipulator

10.6.6.8 bool RCS::CController::bGenerateProgram = false [static]

global flag to create program from Crcl XML

10.6.6.9 bool RCS::CController::bSimulation = true [static]

simulation flag - not connected to robot

10.6.6.10 RCS::CMessageQueue< RCS::CanonCmd > RCS::CController::cmds [static]

queue of commands interpreted from Crcl messages

10.6.6.11 ALogger RCS::CController::CsvLogging [static]

controller status csv logging instance

10.6.6.12 RCS::CController::xml_message_list RCS::CController::donecmds [static]

list of commands interpreted from Crcl messages that have completed

10.6.6.13 MotionPlannerEnum RCS::CController::eCartesianMotionPlanner

type of cartesian motion to use

10.6.6.14 MotionPlannerEnum RCS::CController::eJointMotionPlanner

type of joint motion to use

10.6.6.15 std::string RCS::CController::lastlogstatus

10.6.6.16 RCS::CanonWorldModel RCS::CController::laststatus [static]

last status of controller

10.6.6.17 RCS::CMessageQueue< RCS::CanonCmd > RCS::CController::robotcmds [static]

list of commands to be sent to robot

10.6.6.18 RCS::CanonWorldModel RCS::CController::status [static]

current status of controller

10.6.6.19 RCS::CanonWorldModel RCS::CController::wm [static]

the world model of the controller

The documentation for this struct was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Controller.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Controller.cpp

10.7 CGlobals Class Reference

CGlobals is a catch-all data structure for collecting global functions, extensions, parameters, etc. Functions here usually vary between windows and linux, or there is no easy mechanism in C++ to extend classes (e.g., string) like in C#.

`#include <Globals.h>`

Public Types

- enum TimeFormat { HUM_READ, GMT, GMT_UV_SEC, LOCAL }

Public Member Functions

- CGlobals ()

  Constructor for globals function. Functions here usually vary between windows and linux, or there is no easy mechanism in C++ to extend classes (e.g., string) like in C#.

- std::string StrFormat (const char *fmt,...)

  StrFormat accepts a traditional C format string and expects parameter to follow on calling stack and will produce a string from it.

- bool IsDebug ()
- void Dump ()

  dumps to std out global parameters set at runtime parameters.

- void Sleep (unsigned int ms)

  sleep milliseconds. Equivalent to Sleep in windows.

- bool ReadFile (std::string filename, std::string &contents)

  Reads a file all at once into a string. Include file open, read, close. If fails, empty string is only diagnostic.

- void WriteFile (std::string filename, std::string &contents)

  Writes entire string contents to a file all at once. Include file open, write, close. No error messages.

- void AppendFile (std::string filename, std::string contents)

  Appends entire string contents to a file all at once. Include file open, write, close. No error messages.

- std::string Trim (std::string s)

  Trim cleans blank characters from the front and back of a string. Blank chars are white space, tab, carriage return.

- unsigned int DebugMessage (std::string errmsg)

  Prints an diagnostic message to the debug reporting mechanism. (cout or OutputDebugString)

- unsigned int ErrorMessage (std::string errmsg)

  Prints an error message to the error reporting mechanism.

- unsigned int DebugStrFormat (const char *fmt,...)

  Prints a format string and arguments as a diagnostic message to the debug reporting mechanism. (cout or OutputDebugString)

- std::string GetTimeStamp (TimeFormat format=GMT_UV_SEC)

  GetTimeStamp returns a timestamp string depending on the input format.

Public Attributes

- std::map< std::string, std::string > _appproperties
- int & Debug
- std::string ExeDirectory
- std::string inifile
- std::string SocketPort

10.7.1 Detailed Description

CGlobals is a catch-all data structure for collecting global functions, extensions, parameters, etc. Functions here usually vary between windows and linux, or there is no easy mechanism in C++ to extend classes (e.g., string) like in C#.

10.7.2 Member Enumeration Documentation

10.7.2.1 enum CGlobals::TimeFormat

Enumerator

HUM_READ

GMT

GMT_UV_SEC

LOCAL

10.7.3 Constructor & Destructor Documentation

10.7.3.1 CGlobals::CGlobals ( ) [inline]

Constructor for globals function. Functions here usually vary between windows and linux, or there is no easy mechanism in C++ to extend classes (e.g., string) like in C#.

10.7.4 Member Function Documentation

10.7.4.1 void CGlobals::AppendFile (std::string *filename,* std::string *contents*)

Appends entire string contents to a file all at once. Include file open, write, close. No error messages.

Parameters

filename	is the name of the file to write to
contents	is a reference to a string in which to write string.

10.7.4.2 unsigned int CGlobals::DebugMessage (std::string *errmsg*)

Prints an diagnostic message to the debug reporting mechanism. (cout or OutputDebugString)

Parameters

str	errmsg is the error message that is posted to the debug reporting mechanism.

Returns

a error result integer. (e.g., E_FAIL or -1).

10.7.4.3 unsigned int CGlobals::DebugStrFormat (const char * *fmt,* ...)

Prints a format string and arguments as a diagnostic message to the debug reporting mechanism. (cout or Output-DebugString)

Parameters

fmt	is the error format statement that uses parameters that follow and is posted to the debug reporting mechanism.

Returns

a error result integer. (e.g., E_FAIL or -1).

10.7.4.4 void CGlobals::Dump ( ) [inline]

dumps to std out global parameters set at runtime parameters.

10.7.4.5 unsigned int CGlobals::ErrorMessage (std::string *errmsg*)

Prints an error message to the error reporting mechanism.

Parameters

str	errmsg is the error message that is posted to the error reporting mechanism.

Returns

a error result integer. (e.g., E_FAIL or -1).

10.7.4.6 std::string CGlobals::GetTimeStamp (TimeFormat *format* = GMT_UV_SEC)

GetTimeStamp returns a timestamp string depending on the input format.

Parameters

format	is one of an enumeration describing how to format timestamp.

Returns

a formated timestamp string.

10.7.4.7 bool CGlobals::IsDebug ( ) [inline]

10.7.4.8 bool CGlobals::ReadFile (std::string *filename,* std::string & *contents*)

Reads a file all at once into a string. Include file open, read, close. If fails, empty string is only diagnostic.

Parameters

filename	is the name of the file to read from
contents	is a reference to a string in which to store file contents.

10.7.4.9 void CGlobals::Sleep (unsigned int *ms*) [inline]

sleep milliseconds. Equivalent to Sleep in windows.

Parameters

ms	number of milliseconds to sleep

10.7.4.10 std::string CGlobals::StrFormat (const char * *fmt,* ...) [inline]

StrFormat accepts a traditional C format string and expects parameter to follow on calling stack and will produce a string from it.

Parameters

fmt	is the C format string.

10.7.4.11 std::string CGlobals::Trim (std::string *s*)

Trim cleans blank characters from the front and back of a string. Blank chars are white space, tab, carriage return.

Parameters

str	is the string to trim. Will trim a copy.

Returns

a new trimmed string

10.7.4.12 void CGlobals::WriteFile (std::string *filename,* std::string & *contents*)

Writes entire string contents to a file all at once. Include file open, write, close. No error messages.

Parameters

filename	is the name of the file to write to
contents	is a reference to a string in which to write string.

10.7.5 Member Data Documentation

10.7.5.1 std::map< std::string, std::string > CGlobals::_appproperties

map of application properties, e.g., ["prop"]="value"

10.7.5.2 int& CGlobals::Debug

10.7.5.3 std::string CGlobals::ExeDirectory

the path to directory where exe is located

10.7.5.4 std::string CGlobals::inifile

inifile path name

10.7.5.5 std::string CGlobals::SocketPort

socket port to listen for Crcl clients

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Globals.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Globals.cpp

10.8 RCS::ChainRobotModel Class Reference

`#include <ChainRobotModel.h>`

Public Member Functions

- ChainRobotModel ()
- size_t GetJointNum ()
- size_t GetMovingJoints ()
- RdfJoint & GetJoint (int num)
- void RdfFromXmlFile (std::string xml_string)
- std::string DumpRdfJoints (std::vector< RdfJoint > &jointspec)

Public Attributes

- std::vector< RdfJoint > prejointspec
- std::vector< RdfJoint > postjointspec
- std::vector< RdfJoint > jointspec

10.8.1 Constructor & Destructor Documentation

10.8.1.1 RCS::ChainRobotModel::ChainRobotModel ( ) [inline]

10.8.2 Member Function Documentation

10.8.2.1 std::string RCS::ChainRobotModel::DumpRdfJoints (std::vector< RdfJoint > & *jointspec*)

10.8.2.2 RdfJoint& RCS::ChainRobotModel::GetJoint (int *num*) [inline]

10.8.2.3 size_t RCS::ChainRobotModel::GetJointNum ( ) [inline]

10.8.2.4 size_t RCS::ChainRobotModel::GetMovingJoints ( ) [inline]

10.8.2.5 void RCS::ChainRobotModel::RdfFromXmlFile (std::string *xml_string*)

10.8.3 Member Data Documentation

10.8.3.1 std::vector<RdfJoint> RCS::ChainRobotModel::jointspec

10.8.3.2 std::vector<RdfJoint> RCS::ChainRobotModel::postjointspec

10.8.3.3 std::vector<RdfJoint> RCS::ChainRobotModel::prejointspec

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/ChainRobotModel.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/ChainRobotModel.cpp

10.9 CJointReader Class Reference

The CJointReader is a thread to accept joint update callbacks from ROS. Uses a ros node handle to tell roscore we are subscribing to joint_state topic. Then, when joint updates occur, the callback routine is invoked and the latest joint values saved.

`#include <Communication.h>`

Public Member Functions

- CJointReader (ros::NodeHandle &nh)

  CJointReader constructor that requires a ROS node handle.
- sensor_msgs::JointState GetCurrentReadings ()

  GetCurrentReadings returns the latest joint readings for position, velocity, and effort. It is thread safe.
- void Start ()

  Start sets up subscriber to joint_state topic messages.
- void Stop ()

  Stop unsubscribes to joint_state topic.
- std::vector< double > GetJointValues ()

  GetJointValues sets up subscriber to joint_state topic messages.
- void callback (const sensor_msgs::JointState::ConstPtr &msg)

Public Attributes

- ros::NodeHandle & _nh
- sensor_msgs::JointState _latestreading
- sensor_msgs::JointState _lastreading
- ros::Subscriber sub

Static Public Attributes

- static boost::mutex _reader_mutex

10.9.1 Detailed Description

The CJointReader is a thread to accept joint update callbacks from ROS. Uses a ros node handle to tell roscore we are subscribing to joint_state topic. Then, when joint updates occur, the callback routine is invoked and the latest joint values saved.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 CJointReader::CJointReader (ros::NodeHandle & *nh*)

CJointReader constructor that requires a ROS node handle.

Parameters

nh	ros node handle so joint reader can tell roscore we are subscribing to joint_state topic.

10.9.3 Member Function Documentation

10.9.3.1 void CJointReader::callback (const sensor_msgs::JointState::ConstPtr & *msg*)

type of joint motion to use

10.9.3.2 sensor_msgs::JointState CJointReader::GetCurrentReadings ( )

GetCurrentReadings returns the latest joint readings for position, velocity, and effort. It is thread safe.

10.9.3.3 std::vector< double > CJointReader::GetJointValues ( )

GetJointValues sets up subscriber to joint_state topic messages.

Returns

a std vector of double containing joint positions

10.9.3.4 void CJointReader::Start ( )

Start sets up subscriber to joint_state topic messages.

10.9.3.5 void CJointReader::Stop (void)

Stop unsubscribes to joint_state topic.

10.9.4 Member Data Documentation

10.9.4.1 sensor_msgs::JointState CJointReader::_lastreading

previous joint readings

10.9.4.2 sensor_msgs::JointState CJointReader::_latestreading

latest joint readings

10.9.4.3 ros::NodeHandle& CJointReader::_nh

reference pointer to ROS node handle

10.9.4.4 boost::mutex CJointReader::_reader_mutex [static]

for mutexed reading access

10.9.4.5 ros::Subscriber CJointReader::sub

ros subscriber information

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Communication.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Communication.cpp

10.10 CJointWriter Class Reference

The CJointWriter is a thread to publish new joint values to ROS. Uses a ros node handle to tell roscore we are pusblishing to the joint_path_command topic. Then, when joint updates occur, these are published on joint_path_command the topic.

`#include <Communication.h>`

Public Member Functions

- CJointWriter (ros::NodeHandle &nh)
  CJointWriter constructor that requires a ROS node handle.
- bool JointTrajectoryWrite (std::vector< sensor_msgs::JointState >)
  JointTrajectoryWrite writes the joint values to joint_state topic messages.
- bool JointTrajectoryPositionWrite (sensor_msgs::JointState joint)
  JointTrajectoryWrite writes the joint values to joint_state topic messages.
- void Start ()
  Start creates publisher by telling roscore we are advertising new joint values.
- void Stop ()
  Stop publishing by unadvertising.

Public Attributes

- ros::Publisher traj_pub
- std::vector< std::string > jointnames
- ros::NodeHandle & _nh

Static Public Attributes

- static boost::mutex _writer_mutex

10.10.1 Detailed Description

The CJointWriter is a thread to publish new joint values to ROS. Uses a ros node handle to tell roscore we are pusblishing to the joint_path_command topic. Then, when joint updates occur, these are published on joint_path_command the topic.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 CJointWriter::CJointWriter (ros::NodeHandle & *nh*)

CJointWriter constructor that requires a ROS node handle.

Parameters

nh	ros node handle so joint reader can tell roscore we are advertising new joint values.

10.10.3 Member Function Documentation

10.10.3.1 bool CJointWriter::JointTrajectoryPositionWrite (sensor_msgs::JointState *joint*)

JointTrajectoryWrite writes the joint values to joint_state topic messages.

Parameters

a	sensor_msgs::JointState describing robot joints

Returns

boolean if write occurred as expected.

10.10.3.2 bool CJointWriter::JointTrajectoryWrite (std::vector< sensor_msgs::JointState > *joints*)

JointTrajectoryWrite writes the joint values to joint_state topic messages.

Parameters

a	std vector of sensor_msgs::JointState describing a series of joints

Returns

boolean if all writes occurred as expected.

10.10.3.3 void CJointWriter::Start ( )

Start creates publisher by telling roscore we are advertising new joint values.

10.10.3.4 void CJointWriter::Stop (void)

Stop publishing by unadvertising.

10.10.4 Member Data Documentation

10.10.4.1 ros::NodeHandle& CJointWriter::_nh

reference pointer to ROS node handle

10.10.4.2 boost::mutex CJointWriter::_writer_mutex [static]

for mutexed writing access to joint values

10.10.4.3 std::vector<std::string> CJointWriter::jointnames

ros requries joint names for each joint update

10.10.4.4 ros::Publisher CJointWriter::traj_pub

ros publisher information used for joint updates

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Communication.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Communication.cpp

10.11 RCS::CMessageQueue< T > Class Template Reference

The CMessageQueue offers a mutexed front to a stl deque. The queue is a LIFO data structure. Useful for safely sharing data between multiple threads.

`#include <RCSMsgQueue.h>`

Public Types

- typedef std::deque< T > xml_message_queue

Public Member Functions

- CMessageQueue ()
- void ClearMsgQueue ()

  ClearMsgQueue clears all contents in message queue. T.
- size_t SizeMsgQueue ()

  SizeMsgQueue returns number of items in message queue.
- T PopFrontMsgQueue ()

  PopFrontMsgQueue mutex pop of front item of message queue.
- T BackMsgQueue ()

  BackMsgQueue mutex pop of back item of message queue.
- void AddMsgQueue (T t)

  AddMsgQueue mutex push to back an item onto message queue.
- void InsertFrontMsgQueue (T t)

  InsertFrontMsgQueue mutex push to front an item onto message queue.

Protected Attributes

- boost::mutex m
- xml_message_queue xml_msgs

10.11.1 Detailed Description

template<typename T>class RCS::CMessageQueue< T >

The CMessageQueue offers a mutexed front to a stl deque. The queue is a LIFO data structure. Useful for safely sharing data between multiple threads.

10.11.2 Member Typedef Documentation

10.11.2.1 template<typename T> typedef std::deque<T> RCS::CMessageQueue< T >::xml_message_queue

10.11.3 Constructor & Destructor Documentation

10.11.3.1 template<typename T> RCS::CMessageQueue< T >::CMessageQueue ( ) [inline]

10.11.4 Member Function Documentation

10.11.4.1 template<typename T> void RCS::CMessageQueue< T >::AddMsgQueue (T *t*) [inline]

AddMsgQueue mutex push to back an item onto message queue.

Parameters

T	item to place in back of message queue.

10.11.4.2 template<typename T> T RCS::CMessageQueue< T >::BackMsgQueue ( ) [inline]

BackMsgQueue mutex pop of back item of message queue.

Returns

T returns back item from message queue.

10.11.4.3 template<typename T> void RCS::CMessageQueue< T >::ClearMsgQueue ( ) [inline]

ClearMsgQueue clears all contents in message queue. T.

10.11.4.4 template<typename T> void RCS::CMessageQueue< T >::InsertFrontMsgQueue (T *t*) [inline]

InsertFrontMsgQueue mutex push to front an item onto message queue.

Parameters

T	item to place in front of message queue.

10.11.4.5 template<typename T> T RCS::CMessageQueue< T >::PopFrontMsgQueue ( ) [inline]

PopFrontMsgQueue mutex pop of front item of message queue.

Returns

T returns front item from message queue.

10.11.4.6 template<typename T> size_t RCS::CMessageQueue< T >::SizeMsgQueue ( ) [inline]

SizeMsgQueue returns number of items in message queue.

10.11.5 Member Data Documentation

10.11.5.1 template<typename T> boost::mutex RCS::CMessageQueue< T >::m [protected]

10.11.5.2 template<typename T> xml_message_queue RCS::CMessageQueue< T >::xml_msgs [protected]

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSMsgQueue.h

10.12 CPrimitive Class Reference

`#include <Primitive.h>`

Public Member Functions

- CPrimitive (ros::NodeHandle nh, std::string groupname)
- std::string ToString ()
- void GetJointValues ()
- geometry_msgs::Pose GetRobotState_pose (const robot_state::RobotState &state)
- void ForwardKinematics ()
- std::vector< double > SetRandomJoints ()
- Eigen::Affine3d ComputeForwardKinematics (std::vector< double > jointvalues)
- std::vector< double > InverseKinematics ()
- void RandomGoal ()
- void PlanPath (geometry_msgs::Pose &pose, double dGoalTolerance, int bMoveFlag=1, int bRandomFlag=1)
- void PrintPose ()
- void AddObstacle (std::string id)
- void RemoveObstacle (std::string id)

Static Public Member Functions

- static void PrintJoints (std::vector< double > &joint_values, const std::vector< std::string > &joint_names)
- static void SaveJointInfo (sensor_msgs::JointState &joint_state, std::vector< double > &joint_values, const std::vector< std::string > &joint_names)

10.12.1 Constructor & Destructor Documentation

10.12.1.1 CPrimitive::CPrimitive (ros::NodeHandle *nh,* std::string *groupname*)

10.12.2 Member Function Documentation

10.12.2.1 void CPrimitive::AddObstacle (std::string *id*)

10.12.2.2 Eigen::Affine3d CPrimitive::ComputeForwardKinematics (std::vector< double > *jointvalues*)

10.12.2.3 void CPrimitive::ForwardKinematics ( )

10.12.2.4 void CPrimitive::GetJointValues ( )

10.12.2.5 geometry_msgs::Pose CPrimitive::GetRobotState_pose (const robot_state::RobotState & *state*)

10.12.2.6 std::vector< double > CPrimitive::InverseKinematics ( )

10.12.2.7 void CPrimitive::PlanPath (geometry_msgs::Pose & *pose,* double *dGoalTolerance,* int *bMoveFlag* = `1`, int *bRandomFlag* = `1`)

10.12.2.8 void CPrimitive::PrintJoints (std::vector< double > & *joint_values,* const std::vector< std::string > & *joint_names*) `[static]`

10.12.2.9 void CPrimitive::PrintPose ( )

10.12.2.10 void CPrimitive::RandomGoal ( )

10.12.2.11 void CPrimitive::RemoveObstacle (std::string *id*)

10.12.2.12 void CPrimitive::SaveJointInfo (sensor_msgs::JointState & *joint_state,* std::vector< double > & *joint_values,* const std::vector< std::string > & *joint_names*) `[static]`

10.12.2.13 std::vector< double > CPrimitive::SetRandomJoints ( )

10.12.2.14 std::string CPrimitive::ToString ( )

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Primitive.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Primitive.cpp

10.13 Crcl::CrclClientCmdInterface Class Reference

`#include <CrclInterface.h>`

Public Member Functions

- CrclClientCmdInterface ()
- void SetCommandNum (unsigned long long n)

- std::string CloseToolChanger ()
- std::string Dwell (double seconds)
- std::string EndCanon (int reason)
- std::string GetStatus ()
- std::string InitCanon ()
- std::string Message (std::string message)
- std::string MoveScrew (Crcl::PoseType startPost, VectorType axisPoint, double dAxialDistanceFree, double dAxialDistanceScrew, double dTurn)
- std::string MoveTo (Crcl::PoseType pose, bool bStraight=true)
- std::string MoveThroughTo (Crcl::PoseType *poses, int numPoses, double *accelerations=NULL, double *speeds=NULL, Crcl::PoseToleranceType *tolerances=NULL)
- std::string OpenToolChanger ()
- std::string RunProgram (std::string programText)
- std::string SetEndEffector (double fractionalSetting)
- std::string SetEndPoseTolerance (Crcl::PoseToleranceType toleranceSetting)
- std::string SetEndEffectorTolerance (Crcl::PoseToleranceType toleranceSetting)
- std::string SetAngleUnits (std::string UnitName)
- std::string SetLengthUnits (std::string UnitName)
- std::string SetMotionCoordination (bool bCoordinated)
- std::string SetRotAccel (double dAccel)
- std::string SetRotSpeed (double dSpeed)
- std::string StopMotion (int condition)
- std::string GetStatusReply (CrclStatus *wm)

10.13.1 Constructor & Destructor Documentation

10.13.1.1 Crcl::CrclClientCmdInterface::CrclClientCmdInterface ( ) [inline]

10.13.2 Member Function Documentation

10.13.2.1 std::string CrclClientCmdInterface::CloseToolChanger ( )

10.13.2.2 std::string CrclClientCmdInterface::Dwell (double *seconds*)

10.13.2.3 std::string CrclClientCmdInterface::EndCanon (int *reason*)

10.13.2.4 std::string CrclClientCmdInterface::GetStatus ( )

xercesc::XMLPlatformUtils::Initialize(); CrclClientCmdInterface crcl; std::string text = crcl.CRCLGetStatusCmd(); xercesc::XMLPlatformUtils::Terminate ();

10.13.2.5 std::string CrclClientCmdInterface::GetStatusReply (CrclStatus * *wm*)

10.13.2.6 std::string CrclClientCmdInterface::InitCanon ( )

10.13.2.7 std::string CrclClientCmdInterface::Message (std::string *message*)

10.13.2.8 std::string CrclClientCmdInterface::MoveScrew (Crcl::PoseType *startPost*, VectorType *axisPoint*, double *dAxialDistanceFree*, double *dAxialDistanceScrew*, double *dTurn*)

10.13.2.9 std::string CrclClientCmdInterface::MoveThroughTo (Crcl::PoseType * *poses*, int *numPoses*, double * *accelerations* = NULL, double * *speeds* = NULL, Crcl::PoseToleranceType * *tolerances* = NULL)

10.13.2.10 std::string CrclClientCmdInterface::MoveTo (Crcl::PoseType *pose*, bool *bStraight* = true)

10.13.2.11 std::string CrclClientCmdInterface::OpenToolChanger ( )

10.13.2.12 std::string CrclClientCmdInterface::RunProgram (std::string *programText*)

10.13.2.13 std::string CrclClientCmdInterface::SetAngleUnits (std::string *UnitName*)

10.13.2.14 void Crcl::CrclClientCmdInterface::SetCommandNum (unsigned long long *n*) [inline]

10.13.2.15 std::string CrclClientCmdInterface::SetEndEffector (double *fractionalSetting*)

10.13.2.16 std::string CrclClientCmdInterface::SetEndEffectorTolerance (Crcl::PoseToleranceType *toleranceSetting*)

10.13.2.17 std::string CrclClientCmdInterface::SetEndPoseTolerance (Crcl::PoseToleranceType *toleranceSetting*)

10.13.2.18 std::string CrclClientCmdInterface::SetLengthUnits (std::string *UnitName*)

10.13.2.19 std::string CrclClientCmdInterface::SetMotionCoordination (bool *bCoordinated*)

10.13.2.20 std::string CrclClientCmdInterface::SetRotAccel (double *dAccel*)

10.13.2.21 std::string CrclClientCmdInterface::SetRotSpeed (double *dSpeed*)

10.13.2.22 std::string CrclClientCmdInterface::StopMotion (int *condition*)

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/CrclInterface.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/CrclInterface.cpp

10.14 Crcl::CrclDelegateInterface Class Reference

`#include <CrclInterface.h>`

Public Member Functions

- CrclDelegateInterface ()
- unsigned long long & GetCommandNum ()
- CrclReturn DelegateCRCLCmd (std::string str)
- void CrclRunProgram (::CRCLProgramType::MiddleCommand_sequence cmds)
- CrclReturn DelegateCRCLCmd (::CRCLCommandType &crclCommand)
- virtual CrclReturn ActuateJoints (Crcl::ActuatorJointSequence joints)
- virtual CrclReturn CloseToolChanger ()
- virtual CrclReturn ConfigureJointReports (std::vector< Crcl::JointReport > &jointReports)
- virtual CrclReturn Couple (char *targetID)
- virtual CrclReturn Dwell (double seconds)

- virtual CrclReturn EndCanon ()
- virtual CrclReturn InitCanon ()
- virtual CrclReturn GetStatus ()
- virtual CrclReturn Message (std::string message)
- virtual CrclReturn MoveTo (Crcl::PoseType endpose, bool bStraight)
- virtual CrclReturn MoveThroughTo (std::vector< Crcl::PoseType > &poses, bool bStraight)
- virtual CrclReturn OpenToolChanger ()
- virtual CrclReturn RunProgram (std::string programText)
- virtual CrclReturn SetAbsoluteAcceleration (double acceleration)
- virtual CrclReturn SetAbsoluteSpeed (double speed)
- virtual CrclReturn SetAngleUnits (std::string unitName)
- virtual CrclReturn SetAxialSpeeds (std::vector< double > speeds)
- virtual CrclReturn SetEndEffector (double percent)
- virtual CrclReturn SetEndEffectorTolerance (Crcl::PoseToleranceType dTolerance)
- virtual CrclReturn SetEndPoseTolerance (Crcl::PoseToleranceType tolerance)
- virtual CrclReturn SetForceUnits (std::string unitName)
- virtual CrclReturn SetIntermediatePoseTolerance (Crcl::PoseToleranceType tolerance)
- virtual CrclReturn SetLengthUnits (std::string unitName)
- virtual CrclReturn SetMotionCoordination (bool bCoordinatedMotion)
- virtual CrclReturn SetParameter (char *paramName, void *paramVal)
- virtual CrclReturn SetRelativeAcceleration (double percent)
- virtual CrclReturn SetRelativeSpeed (double percent)
- virtual CrclReturn SetRotAccel (double accel)
- virtual CrclReturn SetRotSpeed (double speed)
- virtual CrclReturn SetTorqueUnits (std::string unitName)
- virtual CrclReturn SetTransAccel (double accel)
- virtual CrclReturn SetTransSpeed (double speed)
- virtual CrclReturn StopMotion (int condition)
- virtual CrclReturn GetTool (double *percent)
- std::string FindLeadingElement (std::string xml)

Static Public Attributes

- static CrclStatus crclwm

10.14.1 Constructor & Destructor Documentation

10.14.1.1 Crcl::CrclDelegateInterface::CrclDelegateInterface ( ) [inline]

10.14.2 Member Function Documentation

10.14.2.1 CrclReturn CrclDelegateInterface::ActuateJoints (Crcl::ActuatorJointSequence *joints*) [virtual]

10.14.2.2 CrclReturn CrclDelegateInterface::CloseToolChanger ( ) [virtual]

10.14.2.3 CrclReturn CrclDelegateInterface::ConfigureJointReports (std::vector< Crcl::JointReport > & *jointReports*) [virtual]

10.14.2.4 CrclReturn CrclDelegateInterface::Couple (char * *targetID*) [virtual]

10.14.2.5 void CrclDelegateInterface::CrclRunProgram (::CRCLProgramType::MiddleCommand_sequence *cmds*)

10.14.2.6 CrclReturn CrclDelegateInterface::DelegateCRCLCmd (std::string *str*)

10.14.2.7 CrclReturn CrclDelegateInterface::DelegateCRCLCmd (::CRCLCommandType & *crclCommand*)

10.14.2.8 CrclReturn CrclDelegateInterface::Dwell (double *seconds*) [virtual]

10.14.2.9 CrclReturn CrclDelegateInterface::EndCanon ( ) [virtual]

10.14.2.10 std::string CrclDelegateInterface::FindLeadingElement (std::string *xml*)

10.14.2.11 unsigned long long& Crcl::CrclDelegateInterface::GetCommandNum ( ) [inline]

10.14.2.12 CrclReturn CrclDelegateInterface::GetStatus ( ) [virtual]

10.14.2.13 virtual CrclReturn Crcl::CrclDelegateInterface::GetTool (double * *percent*) [inline], [virtual]

10.14.2.14 CrclReturn CrclDelegateInterface::InitCanon ( ) [virtual]

10.14.2.15 CrclReturn CrclDelegateInterface::Message (std::string *message*) [virtual]

10.14.2.16 CrclReturn CrclDelegateInterface::MoveThroughTo (std::vector< Crcl::PoseType > & *poses,* bool *bStraight*) [virtual]

10.14.2.17 CrclReturn CrclDelegateInterface::MoveTo (Crcl::PoseType *endpose,* bool *bStraight*) [virtual]

10.14.2.18 CrclReturn CrclDelegateInterface::OpenToolChanger ( ) [virtual]

10.14.2.19 CrclReturn CrclDelegateInterface::RunProgram (std::string *programText*) [virtual]

10.14.2.20 CrclReturn CrclDelegateInterface::SetAbsoluteAcceleration (double *acceleration*) [virtual]

10.14.2.21 CrclReturn CrclDelegateInterface::SetAbsoluteSpeed (double *speed*) [virtual]

10.14.2.22 CrclReturn CrclDelegateInterface::SetAngleUnits (std::string *unitName*) [virtual]

10.14.2.23 CrclReturn CrclDelegateInterface::SetAxialSpeeds (std::vector< double > *speeds*) [virtual]

10.14.2.24 CrclReturn CrclDelegateInterface::SetEndEffector (double *percent*) [virtual]

10.14.2.25 CrclReturn CrclDelegateInterface::SetEndEffectorTolerance (Crcl::PoseToleranceType *dTolerance*) [virtual]

10.14.2.26 CrclReturn CrclDelegateInterface::SetEndPoseTolerance (Crcl::PoseToleranceType *tolerance*) [virtual]

10.14.2.27 CrclReturn CrclDelegateInterface::SetForceUnits (std::string *unitName*) [virtual]

10.14.2.28 CrclReturn CrclDelegateInterface::SetIntermediatePoseTolerance (Crcl::PoseToleranceType *tolerance*) [virtual]

10.14.2.29 CrclReturn CrclDelegateInterface::SetLengthUnits (std::string *unitName*) [virtual]

10.14.2.30 CrclReturn CrclDelegateInterface::SetMotionCoordination (bool *bCoordinatedMotion*) [virtual]

10.14.2.31 CrclReturn CrclDelegateInterface::SetParameter (char * *paramName,* void * *paramVal*) [virtual]

10.14.2.32 CrclReturn CrclDelegateInterface::SetRelativeAcceleration (double *percent*) [virtual]

10.14.2.33 CrclReturn CrclDelegateInterface::SetRelativeSpeed (double *percent*) [virtual]

10.14.2.34 CrclReturn CrclDelegateInterface::SetRotAccel (double *accel*) [virtual]

10.14.2.35 CrclReturn CrclDelegateInterface::SetRotSpeed (double *speed*) [virtual]

10.14.2.36 CrclReturn CrclDelegateInterface::SetTorqueUnits (std::string *unitName*) [virtual]

10.14.2.37 CrclReturn CrclDelegateInterface::SetTransAccel (double *accel*) [virtual]

10.14.2.38 CrclReturn CrclDelegateInterface::SetTransSpeed (double *speed*) [virtual]

10.14.2.39 CrclReturn CrclDelegateInterface::StopMotion (int *condition*) [virtual]

10.14.3 Member Data Documentation

10.14.3.1 CrclStatus CrclDelegateInterface::crclwm [static]

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/CrclInterface.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/CrclInterface.cpp

10.15 Crcl::CrclStatus Struct Reference

`#include <crcl.h>`

Public Member Functions

- CrclStatus ()
- Crcl::JointStatusSequence JointsHome ()
- Crcl::ActuatorJointSequence Merge (Crcl::ActuatorJointSequence, bool bIncremental=false)
- void Update (unsigned long long CommandID)
- void Update (Crcl::CommandStateEnum state)
- void Update (Crcl::JointStatusSequence &joints, RCS::TrajPointType type=RCS::WAYPOINT)
- void Update (Crcl::PoseType &pose, RCS::TrajPointType type=RCS::WAYPOINT)
- void Update (JointState &joints)
- void Update (urdf::Pose &pose)
- VAR (CommandID, unsigned long long)
- VAR (StatusID, unsigned long long)
- VAR (CommandStatus, Crcl::CommandStateEnum)

Public Attributes

- GripperStatus gripper
- Crcl::PoseType _CurrentPose
- Crcl::PoseType _GoalPose
- Crcl::JointStatusSequence _GoalJoints
- Crcl::JointStatusSequence _CurrentJoints
- std::vector< double > _speeds
- double _translationSpeed
- double _translationAccel
- double _rotSpeed
- double _rotAccel
- bool _bCoordinatedMotion
- RCS::CanonLengthUnit _lengthUnit
- double _lengthConversion
- RCS::CanonAngleUnit _angleUnit
- double _angleConversion
- RCS::CanonForceUnit _forceUnit
- double _forceConversion
- RCS::CanonTorqueUnit _torqueUnit
- double _torqueConversion
- Crcl::PoseToleranceType _endPoseTolerance
- Crcl::PoseToleranceType _gripperPoseTolerance
- Crcl::PoseToleranceType _intermediatePoseTolerance
- std::vector< JointReport > _vJointReport
- std::string sCommandState
- std::string Alarm

10.15.1 Constructor & Destructor Documentation

10.15.1.1 Crcl::CrclStatus::CrclStatus ( )

10.15.2 Member Function Documentation

10.15.2.1 Crcl::JointStatusSequence Crcl::CrclStatus::JointsHome ( )

10.15.2.2 Crcl::ActuatorJointSequence Crcl::CrclStatus::Merge (Crcl::ActuatorJointSequence *joints*, bool *bIncremental* = `false`)

10.15.2.3 void Crcl::CrclStatus::Update (unsigned long long *CommandID*)

10.15.2.4 void Crcl::CrclStatus::Update (Crcl::CommandStateEnum *state*)

10.15.2.5 void Crcl::CrclStatus::Update (Crcl::JointStatusSequence & *joints*, RCS::TrajPointType *type* = RCS::WAYPOINT)

10.15.2.6 void Crcl::CrclStatus::Update (Crcl::PoseType & *pose*, RCS::TrajPointType *type* = RCS::WAYPOINT)

10.15.2.7 void Crcl::CrclStatus::Update (JointState & *joints*)

10.15.2.8 void Crcl::CrclStatus::Update (urdf::Pose & *pose*)

10.15.2.9 Crcl::CrclStatus::VAR (CommandID , unsigned long *long*)

10.15.2.10 Crcl::CrclStatus::VAR (StatusID , unsigned long *long*)

10.15.2.11 Crcl::CrclStatus::VAR (CommandStatus , **Crcl::CommandStateEnum**)

10.15.3 Member Data Documentation

10.15.3.1 double Crcl::CrclStatus::_angleConversion

10.15.3.2 **RCS::CanonAngleUnit** Crcl::CrclStatus::_angleUnit

10.15.3.3 bool Crcl::CrclStatus::_bCoordinatedMotion

10.15.3.4 **Crcl::JointStatusSequence** Crcl::CrclStatus::_CurrentJoints

10.15.3.5 **Crcl::PoseType** Crcl::CrclStatus::_CurrentPose

10.15.3.6 **Crcl::PoseToleranceType** Crcl::CrclStatus::_endPoseTolerance

10.15.3.7 double Crcl::CrclStatus::_forceConversion

10.15.3.8 **RCS::CanonForceUnit** Crcl::CrclStatus::_forceUnit

10.15.3.9 **Crcl::JointStatusSequence** Crcl::CrclStatus::_GoalJoints

10.15.3.10 **Crcl::PoseType** Crcl::CrclStatus::_GoalPose

10.15.3.11 **Crcl::PoseToleranceType** Crcl::CrclStatus::_gripperPoseTolerance

10.15.3.12 **Crcl::PoseToleranceType** Crcl::CrclStatus::_intermediatePoseTolerance

10.15.3.13 double Crcl::CrclStatus::_lengthConversion

10.15.3.14 **RCS::CanonLengthUnit** Crcl::CrclStatus::_lengthUnit

10.15.3.15 double Crcl::CrclStatus::_rotAccel

10.15.3.16 double Crcl::CrclStatus::_rotSpeed

10.15.3.17 std::vector<double> Crcl::CrclStatus::_speeds

10.15.3.18 double Crcl::CrclStatus::_torqueConversion

10.15.3.19 **RCS::CanonTorqueUnit** Crcl::CrclStatus::_torqueUnit

10.15.3.20 double Crcl::CrclStatus::_translationAccel

10.15.3.21 double Crcl::CrclStatus::_translationSpeed

10.15.3.22 std::vector<JointReport> Crcl::CrclStatus::_vJointReport

10.15.3.23 std::string Crcl::CrclStatus::Alarm

10.15.3.24 GripperStatus Crcl::CrclStatus::gripper

10.15.3.25 std::string Crcl::CrclStatus::sCommandState

The documentation for this struct was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/crcl.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/crcl.cpp

10.16 Crcl::CrclStatusMsgInterface Class Reference

`#include <CrclInterface.h>`

Public Member Functions

- CrclStatusMsgInterface ()
- void ParseCRCLStatus (std::string filename)
- void ParseCRCLStatusString (std::string str)

Public Attributes

- CrclStatus _status

10.16.1 Constructor & Destructor Documentation

10.16.1.1 Crcl::CrclStatusMsgInterface::CrclStatusMsgInterface ( ) [inline]

10.16.2 Member Function Documentation

10.16.2.1 void CrclStatusMsgInterface::ParseCRCLStatus (std::string *filename*)

10.16.2.2 void CrclStatusMsgInterface::ParseCRCLStatusString (std::string *str*)

10.16.3 Member Data Documentation

10.16.3.1 CrclStatus Crcl::CrclStatusMsgInterface::_status

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/CrclInterface.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/CrclInterface.cpp

10.17 CTrajectory Class Reference

`#include <Trajectory.h>`

Public Member Functions

- CTrajectory ()
- void Init (std::string robot_description, std::string group_name, std::string world_frame, std::string tcp_frame, std::vector< std::string > names, bool bCheckCollisions=true)
- TrajectoryVec PlanPath (TrajectoryVec)
- std::vector< JointState > JointTrajectorytoJointStateVector (const TrajectoryVec &trajectory)
- descartes_core::TrajectoryPtPtr makeCartesianPoint (const Eigen::Affine3d &pose)
- descartes_core::TrajectoryPtPtr makeToleroncedCartesianPoint (const Eigen::Affine3d &pose)
- MoveitStateAdapterPtr GetModel ()
- descartes_planner::DensePlanner GetDensePlanner ()
- std::vector< std::string > GetJointNames ()

Public Attributes

- MoveitStateAdapterPtr _model
- descartes_planner::DensePlanner _planner
- std::string _robot_description
- std::string _group_name
- std::string _world_frame
- std::string _tcp_frame
- std::vector< std::string > joint_names
- bool _bInited

10.17.1 Constructor & Destructor Documentation

10.17.1.1 CTrajectory::CTrajectory ( )

10.17.2 Member Function Documentation

10.17.2.1 descartes_planner::DensePlanner CTrajectory::GetDensePlanner ( ) [inline]

10.17.2.2 std::vector<std::string> CTrajectory::GetJointNames ( ) [inline]

10.17.2.3 MoveitStateAdapterPtr CTrajectory::GetModel ( ) [inline]

10.17.2.4 void CTrajectory::Init (std::string *robot_description*, std::string *group_name*, std::string *world_frame*, std::string *tcp_frame*, std::vector< std::string > *names*, bool *bCheckCollisions* = `true`)

10.17.2.5 std::vector< JointState > CTrajectory::JointTrajectorytoJointStateVector (const TrajectoryVec & *trajectory*)

10.17.2.6 descartes_core::TrajectoryPtPtr CTrajectory::makeCartesianPoint (const Eigen::Affine3d & *pose*) [inline]

10.17.2.7 descartes_core::TrajectoryPtPtr CTrajectory::makeToleroncedCartesianPoint (const Eigen::Affine3d & *pose*) [inline]

10.17.2.8 TrajectoryVec CTrajectory::PlanPath (TrajectoryVec *points*)

10.17.3 Member Data Documentation

10.17.3.1 bool CTrajectory::_bInited

10.17.3.2 std::string CTrajectory::_group_name

10.17.3.3 MoveitStateAdapterPtr CTrajectory::_model

10.17.3.4 descartes_planner::DensePlanner CTrajectory::_planner

10.17.3.5 std::string CTrajectory::_robot_description

10.17.3.6 std::string CTrajectory::_tcp_frame

10.17.3.7 std::string CTrajectory::_world_frame

10.17.3.8 std::vector<std::string> CTrajectory::joint_names

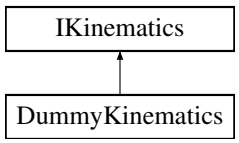
The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Trajectory.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Trajectory.cpp

10.18 DummyKinematics Class Reference

`#include <Kinematics.h>`

Inheritance diagram for DummyKinematics:



Public Member Functions

- virtual std::vector< double > GetJointValues ()
- virtual void SetJointValues (std::vector< double > joint_values)
- virtual urdf::Pose FK (std::vector< double > jv)
- virtual std::vector< double > IK (RCS::Pose &pose, std::vector< double > oldjoints)
- virtual size_t AllPoseToJoints (RCS::Pose &pose, std::vector< std::vector< double > > &newjoints)
- virtual std::vector< double > NearestJoints (std::vector< double > oldjoints, std::vector< std::vector< double > > &newjoints)

10.18.1 Member Function Documentation

10.18.1.1 virtual size_t DummyKinematics::AllPoseToJoints (RCS::Pose & *pose,* std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.18.1.2 virtual urdf::Pose DummyKinematics::FK (std::vector< double > *jv*) [inline], [virtual]

Implements IKinematics.

10.18.1.3 virtual std::vector<double> DummyKinematics::GetJointValues ( ) [inline], [virtual]

Implements IKinematics.

10.18.1.4 virtual std::vector<double> DummyKinematics::IK (RCS::Pose & *pose,* std::vector< double > *oldjoints*) [inline], [virtual]

Implements IKinematics.

10.18.1.5 virtual std::vector<double> DummyKinematics::NearestJoints (std::vector< double > *oldjoints,* std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.18.1.6 virtual void DummyKinematics::SetJointValues (std::vector< double > *joint_values*) [inline], [virtual]

Implements IKinematics.

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Kinematics.h

10.19 Crcl::GripperStatus Struct Reference

`#include <crcl.h>`

Public Attributes

- std::string _name
- double _dPosition

10.19.1 Member Data Documentation

10.19.1.1 double Crcl::GripperStatus::_dPosition

10.19.1.2 std::string Crcl::GripperStatus::_name

The documentation for this struct was generated from the following file:

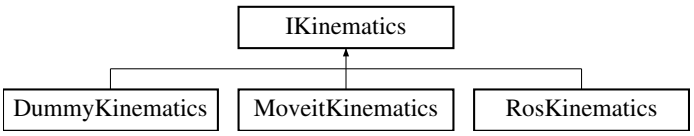
- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/crcl.h

10.20 IKinematics Class Reference

The IKinematics provides is an abstract class with pure virtual functions that are overriden by actual kinematic implementations.

`#include <Kinematics.h>`

Inheritance diagram for IKinematics:



Public Member Functions

- virtual std::vector< double > GetJointValues ()=0
- virtual void SetJointValues (std::vector< double > joint_values)=0
- virtual urdf::Pose FK (std::vector< double > jv)=0
- virtual std::vector< double > IK (RCS::Pose &pose, std::vector< double > oldjoints)=0
- virtual size_t AllPoseToJoints (RCS::Pose &pose, std::vector< std::vector< double > > &newjoints)=0
- virtual std::vector< double > NearestJoints (std::vector< double > oldjoints, std::vector< std::vector< double > > &newjoints)=0
- virtual void Init (std::string groupname, std::string eelinkname)

10.20.1 Detailed Description

The IKinematics provides is an abstract class with pure virtual functions that are overriden by actual kinematic implementations.

10.20.2 Member Function Documentation

10.20.2.1 virtual size_t IKinematics::AllPoseToJoints (RCS::Pose & *pose*, std::vector< std::vector< double > > & *newjoints*) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

10.20.2.2 virtual urdf::Pose IKinematics::FK (std::vector< double > *jv*) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

10.20.2.3 virtual std::vector<double> IKinematics::GetJointValues ( ) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

10.20.2.4 virtual std::vector<double> IKinematics::IK (RCS::Pose & *pose*, std::vector< double > *oldjoints*) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

10.20.2.5 virtual void IKinematics::Init (std::string *groupname*, std::string *eelinkname*) [inline], [virtual]

Reimplemented in MoveitKinematics, and RosKinematics.

10.20.2.6 virtual std::vector<double> IKinematics::NearestJoints (std::vector< double > *oldjoints*, std::vector< std::vector< double > > & *newjoints*) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

10.20.2.7 virtual void IKinematics::SetJointValues (std::vector< double > *joint_values*) [pure virtual]

Implemented in MoveitKinematics, RosKinematics, and DummyKinematics.

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Kinematics.h

10.21 IRate Class Reference

IRate is an interface class for defining the allowed motion rates.

`#include <trajectoryMaker.h>`

Public Member Functions

- IRate ()
- IRate (double maximum_velocity, double maximum_accel, double cycleTime)
- void SetCurrentMotion (double final_velocity, double current_feedrate, double current_velocity)
- NVAR (FinalVelocity, double, _final_velocity)
- NVAR (CurrentFeedrate, double, _current_feedrate)
- NVAR (CurrentVelocity, double, _current_velocity)
- NVAR (MaximumVelocity, double, _maximum_velocity)
- NVAR (MaximumAccel, double, _maximum_accel)
- NVAR (CycleTime, double, _cycleTime)
- NVAR (CurrentAccel, double, _current_accel)
- NVAR (MsFlag, RCS::CanonAccProfile, _msflag)

10.21.1 Detailed Description

IRate is an interface class for defining the allowed motion rates.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 IRate::IRate ( ) [inline]

10.21.2.2 IRate::IRate (double *maximum_velocity,* double *maximum_accel,* double *cycleTime*) [inline]

10.21.3 Member Function Documentation

10.21.3.1 IRate::NVAR (FinalVelocity , double , _final_velocity)

10.21.3.2 IRate::NVAR (CurrentFeedrate , double , _current_feedrate)

10.21.3.3 IRate::NVAR (CurrentVelocity , double , _current_velocity)

10.21.3.4 IRate::NVAR (MaximumVelocity , double , _maximum_velocity)

10.21.3.5 IRate::NVAR (MaximumAccel , double , _maximum_accel)

10.21.3.6 IRate::NVAR (CycleTime , double , _cycleTime)

10.21.3.7 IRate::NVAR (CurrentAccel , double , _current_accel)

10.21.3.8 IRate::NVAR (MsFlag , RCS::CanonAccProfile , _msflag)

10.21.3.9 void IRate::SetCurrentMotion (double *final_velocity,* double *current_feedrate,* double *current_velocity*) [inline]

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/trajectoryMaker.h

10.22 Crcl::JointReport Struct Reference

`#include <crcl.h>`

Public Attributes

- size_t _nJointNumber
- bool _bReportPosition
- bool _bReportTorqueOrForce
- bool _bReportVelocity

10.22.1 Member Data Documentation

10.22.1.1 bool Crcl::JointReport::_bReportPosition

10.22.1.2 bool Crcl::JointReport::_bReportTorqueOrForce

10.22.1.3 bool Crcl::JointReport::_bReportVelocity

10.22.1.4 size_t Crcl::JointReport::_nJointNumber

The documentation for this struct was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/crcl.h

10.23 MotionControl Class Reference

MotionControl is a class that contains some useful motion control methods.

`#include <MotionControl.h>`

Public Member Functions

- bool executeTrajectory (const trajectory_msgs::JointTrajectory &trajectory, const std::string &trajectory_ns)

  executeTrajectory will send a traject to ros to execute
- urdf::Pose computeTranslation (urdf::Pose &_curPos, urdf::Pose &_goalPos, double dIncrement)

  Return linear interpolation (lerp) between current and goal translation.
- std::vector< urdf::Pose > computeWaypoints (urdf::Pose &_curPos, urdf::Pose &_goalPos, double dGap=0.001, bool bAddStart=false)

  Compute waypoints between current and goal poses with assigned distance between poses.
- std::vector< JointState > computeCoorindatedWaypoints (std::vector< double > &_curJts, std::vector< double > &_goalJts, double dGap=0.001, bool bAddStart=false)

  computeCoorindatedWaypoints returns a vector of straightline waypoints between current and goal poses at a given distance. Joints arrive a destination at the same time within the trajectory.
- std::vector< JointState > computeUncoorindatedWaypoints (std::vector< double > &_curJts, std::vector< double > &_goalJts, double dGap=0.001, bool bAddStart=false)

  computeUncoorindatedWaypoints returns a vector of waypoints between current and goal poses at a given distance. Joints arrive a destination at various times in the trajectory.
- int computeIncrements (std::vector< double > &_curJts, std::vector< double > &_goalJts, double gap=0.001)
- int computeIncrements (urdf::Pose &_curPos, urdf::Pose &_goalPos, double dGap=0.001)

Static Public Member Functions

- static bool AtGoal (JointState goal, JointState current, double epsilon=0.001)

  AtGoal will determine if a pair joint state values are equal (within an epsilon tolerance).

Static Public Attributes

- static double epsilon = 0.001

10.23.1 Detailed Description

MotionControl is a class that contains some useful motion control methods.

10.23.2 Member Function Documentation

10.23.2.1 bool MotionControl::AtGoal (JointState *goal,* JointState *current,* double *epsilon =* 0.001) [static]

AtGoal will determine if a pair joint state values are equal (within an epsilon tolerance).

Parameters

goal	description of goal joint state
current	description of current joint state
epsilon	tolerance of equality

Returns

boolean whether robot is at the desired goal described in joint values.

10.23.2.2 std::vector< JointState > MotionControl::computeCoorindatedWaypoints (std::vector< double > & *_curJts*, std::vector< double > & *_goalJts*, double *dGap* = 0.001, bool *bAddStart* = false)

computeCoorindatedWaypoints returns a vector of straightline waypoints between current and goal poses at a given distance. Joints arrive a destination at the same time within the trajectory.

Parameters

_curPos	description of current pose
_goalPos	description of goal pose
gap	distance between waypoints
bAddStart	boolean to determine if starting pose is included in waypoints

Returns

vector of straighline waypoint poses with gap distance between poses.

10.23.2.3 int MotionControl::computeIncrements (std::vector< double > & *_curJts*, std::vector< double > & *_goalJts*, double *gap* = 0.001)

10.23.2.4 int MotionControl::computeIncrements (urdf::Pose & *_curPos*, urdf::Pose & *_goalPos*, double *dGap* = 0.001)

10.23.2.5 urdf::Pose MotionControl::computeTranslation (urdf::Pose & *_curPos*, urdf::Pose & *_goalPos*, double *dIncrement*)

Return linear interpolation (lerp) between current and goal translation.

Parameters

_curPos	description of current pose
_goalPos	description of goal pose
dIncrement	translation amount from [0..1]

Returns

pose containing lerped pose translation.

10.23.2.6 std::vector< JointState > MotionControl::computeUncoorindatedWaypoints (std::vector< double > & *_curJts*, std::vector< double > & *_goalJts*, double *dGap* = 0.001, bool *bAddStart* = false)

computeUncoorindatedWaypoints returns a vector of waypoints between current and goal poses at a given distance. Joints arrive a destination at various times in the trajectory.

Parameters

_curPos	description of current pose
_goalPos	description of goal pose
gap	distance between waypoints
bAddStart	boolean to determine if starting pose is included in waypoints

Returns

vector of straighline waypoint poses with gap distance between poses.

10.23.2.7 std::vector< urdf::Pose > MotionControl::computeWaypoints (urdf::Pose & *_curPos*, urdf::Pose & *_goalPos*, double *dGap* = `0.001`, bool *bAddStart* = `false`)

Compute waypoints between current and goal poses with assigned distance between poses.

Parameters

_curPos	description of current pose
_goalPos	description of goal pose
gap	distance between waypoints
bAddStart	boolean to determine if starting pose is included in waypoints

Returns

vector of waypoint poses with gap distance between poses.

10.23.2.8 bool MotionControl::executeTrajectory (const trajectory_msgs::JointTrajectory & *trajectory*, const std::string & *trajectory_ns*)

executeTrajectory will send a traject to ros to execute

Parameters

trajectory	
trajectory_ns	namespace of trajectory

Returns

boolean whether success or failure

10.23.3 Member Data Documentation

10.23.3.1 double MotionControl::epsilon = 0.001 `[static]`

allowable difference length in equality between two numbers

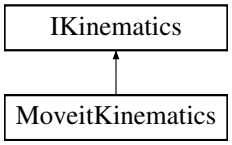
The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/MotionControl.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/MotionControl.cpp

10.24 MoveitKinematics Class Reference

`#include <Kinematics.h>`

Inheritance diagram for MoveitKinematics:



Public Member Functions

- MoveitKinematics (ros::NodeHandle &nh)
- virtual std::vector< double > GetJointValues ()
- virtual void SetJointValues (std::vector< double > joint_values)
- virtual urdf::Pose FK (std::vector< double > jv)
- virtual std::vector< double > IK (RCS::Pose &pose, std::vector< double > oldjoints)
- virtual size_t AllPoseToJoints (RCS::Pose &pose, std::vector< std::vector< double > > &newjoints)
- virtual std::vector< double > NearestJoints (std::vector< double > oldjoints, std::vector< std::vector< double > > &newjoints)
- virtual void Init (std::string groupname, std::string eelinkname)

Public Attributes

- boost::shared_ptr < moveit::planning_interface::MoveGroup > group
- robot_model::RobotModelPtr kinematic_model
- robot_state::RobotStatePtr kinematic_state
- robot_state::JointModelGroup * joint_model_group
- std::vector< double > joint_values
- std::vector< std::string > joint_names
- std::string _groupname
- std::string _eelinkname
- ros::NodeHandle & _nh
- bool _bInit
- boost::mutex kinmutex

10.24.1 Constructor & Destructor Documentation

10.24.1.1 MoveitKinematics::MoveitKinematics (ros::NodeHandle & *nh*)

10.24.2 Member Function Documentation

10.24.2.1 virtual size_t MoveitKinematics::AllPoseToJoints (RCS::Pose & *pose*, std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.24.2.2 urdf::Pose MoveitKinematics::FK (std::vector< double > *jv*) [virtual]

Implements IKinematics.

10.24.2.3 std::vector< double > MoveitKinematics::GetJointValues ( ) [virtual]

Implements IKinematics.

10.24.2.4 std::vector< double > MoveitKinematics::IK (RCS::Pose & *pose*, std::vector< double > *oldjoints*) [virtual]

Implements IKinematics.

10.24.2.5 void MoveitKinematics::Init (std::string *groupname*, std::string *eelinkname*) [virtual]

Reimplemented from IKinematics.

10.24.2.6 virtual std::vector<double> MoveitKinematics::NearestJoints (std::vector< double > *oldjoints*, std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.24.2.7 void MoveitKinematics::SetJointValues (std::vector< double > *joint_values*) [virtual]

Implements IKinematics.

10.24.3 Member Data Documentation

10.24.3.1 bool MoveitKinematics::_bInit

10.24.3.2 std::string MoveitKinematics::_eelinkname

10.24.3.3 std::string MoveitKinematics::_groupname

10.24.3.4 ros::NodeHandle& MoveitKinematics::_nh

10.24.3.5 boost::shared_ptr<moveit::planning_interface::MoveGroup> MoveitKinematics::group

10.24.3.6 robot_state::JointModelGroup* MoveitKinematics::joint_model_group

10.24.3.7 std::vector<std::string> MoveitKinematics::joint_names

10.24.3.8 std::vector<double> MoveitKinematics::joint_values

10.24.3.9 robot_model::RobotModelPtr MoveitKinematics::kinematic_model

10.24.3.10 robot_state::RobotStatePtr MoveitKinematics::kinematic_state

10.24.3.11 boost::mutex MoveitKinematics::kinmutex

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Kinematics.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Kinematics.cpp

10.25 MoveitPlanning Class Reference

`#include <moveit.h>`

Public Member Functions

- MoveitPlanning (ros::NodeHandle &nh)
- ~MoveitPlanning ()
- std::vector< JointState > GetJtsPlan ()
- bool Plan (JointState curjoints, JointState goaljoints)
- bool Plan (urdf::Pose &curpose, urdf::Pose &goalpose)
- bool Plan (urdf::Pose &pose)
- bool Plan (std::vector< urdf::Pose > &waypoints)
- bool Plan (JointState joints)
- bool Plan (Eigen::Affine3d &pose)
- bool Plan (geometry_msgs::Pose &pose)
- urdf::Pose GetCurrentPose ()
- std::vector< double > GetJointValues ()
- urdf::Pose ForwardKinematics ()
- std::vector< double > SetRandomJoints ()
- void DisplayPlan ()
- void SavePlan ()
- boost::shared_ptr < moveit::planning_interface::MoveGroup > GetGroup ()
- std::vector< std::string > GetJointNames ()

Public Attributes

- moveit::planning_interface::MoveGroup::Plan my_plan
- boost::shared_ptr < moveit::planning_interface::MoveGroup > group
- robot_model::RobotModelPtr kinematic_model
- robot_state::RobotStatePtr kinematic_state
- robot_state::JointModelGroup * joint_model_group
- std::string _groupname
- std::vector< std::string > joint_names
- bool _bInited
- ros::Publisher display_publisher
- std::vector< JointState > plannedjts

10.25.1 Constructor & Destructor Documentation

10.25.1.1 MoveitPlanning::MoveitPlanning (ros::NodeHandle & *nh*)

10.25.1.2 MoveitPlanning::∼MoveitPlanning ( )

10.25.2 Member Function Documentation

10.25.2.1 void MoveitPlanning::DisplayPlan ( )

10.25.2.2 urdf::Pose MoveitPlanning::ForwardKinematics ( )

10.25.2.3 urdf::Pose MoveitPlanning::GetCurrentPose ( )

10.25.2.4 boost::shared_ptr<moveit::planning_interface::MoveGroup> MoveitPlanning::GetGroup ( ) [inline]

10.25.2.5 std::vector<std::string> MoveitPlanning::GetJointNames ( ) [inline]

10.25.2.6 std::vector< double > MoveitPlanning::GetJointValues ( )

10.25.2.7 std::vector< JointState > MoveitPlanning::GetJtsPlan ( )

10.25.2.8 bool MoveitPlanning::Plan (JointState *curjoints,* JointState *goaljoints*)

10.25.2.9 bool MoveitPlanning::Plan (urdf::Pose & *curpose,* urdf::Pose & *goalpose*)

10.25.2.10 bool MoveitPlanning::Plan (urdf::Pose & *pose*)

10.25.2.11 bool MoveitPlanning::Plan (std::vector< urdf::Pose > & *waypoints*)

10.25.2.12 bool MoveitPlanning::Plan (JointState *joints*)

10.25.2.13 bool MoveitPlanning::Plan (Eigen::Affine3d & *pose*)

10.25.2.14 bool MoveitPlanning::Plan (geometry_msgs::Pose & *pose*)

10.25.2.15 void MoveitPlanning::SavePlan ( )

10.25.2.16 std::vector< double > MoveitPlanning::SetRandomJoints ( )

10.25.3 Member Data Documentation

10.25.3.1 bool MoveitPlanning::_bInited

10.25.3.2 std::string MoveitPlanning::_groupname

10.25.3.3 ros::Publisher MoveitPlanning::display_publisher

10.25.3.4 boost::shared_ptr<moveit::planning_interface::MoveGroup> MoveitPlanning::group

10.25.3.5 robot_state::JointModelGroup∗ MoveitPlanning::joint_model_group

10.25.3.6 std::vector<std::string> MoveitPlanning::joint_names

10.25.3.7 robot_model::RobotModelPtr MoveitPlanning::kinematic_model

10.25.3.8 robot_state::RobotStatePtr MoveitPlanning::kinematic_state

10.25.3.9 moveit::planning_interface::MoveGroup::Plan MoveitPlanning::my_plan

10.25.3.10 std::vector<JointState> MoveitPlanning::plannedjts

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/moveit.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/moveit.cpp

10.26 RCSInterpreter Class Reference

RCSInterpreter parses a RCS command and generates robot motion commands.

`#include <RCSInterpreter.h>`

Public Member Functions

- RCSInterpreter (IKinematicsSharedPtr k=NULL)

  RCSInterpreter constructor that optionally accepts pointer to kinematic instance.
- ~RCSInterpreter (void)
- int ParseCommand (RCS::CanonCmd cmd)

  ParseCommand parses a RCS command and queues robot motion commands.

Public Attributes

- IKinematicsSharedPtr _kinematics
- TrajectoryVec results
- std::vector< double > times
- IRate rates
- MotionControl motioncontrol

Protected Member Functions

- void AddJointCommands (std::vector< JointState > gotojoints)

  AddJointCommands accepts vector of joint trajectories and adds to robot motion queue.
- std::vector< JointState > PlanCartesianMotion (std::vector< urdf::Pose > poses)

  PlanCartesianMotion accepts vector of poses and generates a vector of joint trajectories. Can use a couple of planning algorithms to generate trajectory.

10.26.1 Detailed Description

RCSInterpreter parses a RCS command and generates robot motion commands.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 RCSInterpreter::RCSInterpreter (IKinematicsSharedPtr *k* = NULL)

RCSInterpreter constructor that optionally accepts pointer to kinematic instance.

Parameters

k	is the kinematics pointer

10.26.2.2 RCSInterpreter::~RCSInterpreter (void)

10.26.3 Member Function Documentation

10.26.3.1 void RCSInterpreter::AddJointCommands (std::vector< JointState > *gotojoints*) [protected]

AddJointCommands accepts vector of joint trajectories and adds to robot motion queue.

Parameters

gotojoints	is the vector of joint states describing the motion.

10.26.3.2 int RCSInterpreter::ParseCommand (RCS::CanonCmd *cmd*)

ParseCommand parses a RCS command and queues robot motion commands.

Parameters

cmd	is the command to interpret

10.26.3.3 std::vector< JointState > RCSInterpreter::PlanCartesianMotion (std::vector< urdf::Pose > *poses*) [protected]

PlanCartesianMotion accepts vector of poses and generates a vector of joint trajectories. Can use a couple of planning algorithms to generate trajectory.

Parameters

poses	is the vector of cartesian motion.

Returns

vector of planned joint states

10.26.4 Member Data Documentation

10.26.4.1 IKinematicsSharedPtr RCSInterpreter::_kinematics

kinematics pointer

10.26.4.2 MotionControl RCSInterpreter::motioncontrol

instance of simple motion control object

10.26.4.3 IRate RCSInterpreter::rates

rates structure for simple motion planner

10.26.4.4 TrajectoryVec RCSInterpreter::results

descartes motion planner results

10.26.4.5 std::vector<double> RCSInterpreter::times

descartes times for trajectory results

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RCSInterpreter.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Archive/RCSInterpreter.cpp

10.27 RCS::RdfJoint Struct Reference

`#include <ChainRobotModel.h>`

Public Types

- enum JointType {
  UNKNOWN = 0, REVOLUTE, CONTINUOUS, PRISMATIC,
  FLOATING, PLANAR, FIXED }

Public Member Functions

- std::string DumpRdfJoint ()

Public Attributes

- std::string name
- int index
- std::string sLink
- enum RCS::RdfJoint::JointType type
- Eigen::Vector3d axis
- Eigen::Vector3d xyzorigin
- Eigen::Vector3d rpyorigin
- double lowerlimit
- double upperlimit
- double effortlimit
- double velocitylimit
- bool bounded

10.27.1 Member Enumeration Documentation

10.27.1.1 enum RCS::RdfJoint::JointType

Enumerator

UNKNOWN

REVOLUTE

CONTINUOUS

PRISMATIC

FLOATING

PLANAR

FIXED

10.27.2 Member Function Documentation

10.27.2.1 std::string RCS::RdfJoint::DumpRdfJoint ( )

10.27.3 Member Data Documentation

10.27.3.1 Eigen::Vector3d RCS::RdfJoint::axis

10.27.3.2 bool RCS::RdfJoint::bounded

10.27.3.3 double RCS::RdfJoint::effortlimit

10.27.3.4 int RCS::RdfJoint::index

10.27.3.5 double RCS::RdfJoint::lowerlimit

10.27.3.6 std::string RCS::RdfJoint::name

10.27.3.7 Eigen::Vector3d RCS::RdfJoint::rpyorigin

10.27.3.8 std::string RCS::RdfJoint::sLink

10.27.3.9 enum RCS::RdfJoint::JointType RCS::RdfJoint::type

10.27.3.10 double RCS::RdfJoint::upperlimit

10.27.3.11 double RCS::RdfJoint::velocitylimit

10.27.3.12 Eigen::Vector3d RCS::RdfJoint::xyzorigin

The documentation for this struct was generated from the following files:

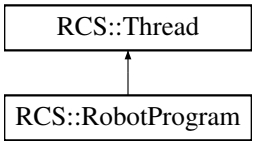
- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/ChainRobotModel.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/ChainRobotModel.cpp

10.28 RCS::RobotProgram Class Reference

The RobotProgram is a thread to handle crcl programs. Crcl programs are not in fact legitimate, however, debugging and verification are assisted by programs. However, program as in the Crcl XSD specification, so it doesn't hurt to handle. They require special handling as only one command should be done at a time. Uses codesynthesis to parse Crcl xml into C++ data structures.

`#include <Controller.h>`

Inheritance diagram for RCS::RobotProgram:



Public Member Functions

- RobotProgram (double cycletime=DEFAULT_LOOP_CYCLE)

  RobotProgram constructor that requires a cycle time for RCS thread timing.
- virtual void ExecuteProgram (std::string programpath)

  ExecuteProgram reads a file path for CRCL XML program. It will set up interpreting the program. It is thread safe.
- virtual void Action ()

  Action is the main loop in the RobotProgram RCS thread.

Public Attributes

- std::string _programname
- Crcl::CrclDelegateInterface _delegate
- std::istringstream istr
- int cmdnum
- ::CRCLProgramType::MiddleCommand_sequence & cmds

Static Public Attributes

- static boost::mutex _progmutex

Additional Inherited Members

10.28.1 Detailed Description

The RobotProgram is a thread to handle crcl programs. Crcl programs are not in fact legitimate, however, debugging and verification are assisted by programs. However, program as in the Crcl XSD specification, so it doesn't hurt to handle. They require special handling as only one command should be done at a time. Uses codesynthesis to parse Crcl xml into C++ data structures.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 RCS::RobotProgram::RobotProgram (double *cycletime* = DEFAULT_LOOP_CYCLE)

RobotProgram constructor that requires a cycle time for RCS thread timing.

Parameters

cycletime	in seconds.

10.28.3 Member Function Documentation

10.28.3.1 void RCS::RobotProgram::Action ( ) [virtual]

Action is the main loop in the RobotProgram RCS thread.

Executes one program command at a time. needs to wait until current command is done before moving on to next command.

Reimplemented from RCS::Thread.

10.28.3.2 void RCS::RobotProgram::ExecuteProgram (std::string *programpath*) [virtual]

ExecuteProgram reads a file path for CRCL XML program. It will set up interpreting the program. It is thread safe.

Parameters

programpath	path of file containing crcl xml program.

10.28.4 Member Data Documentation

10.28.4.1 Crcl::CrclDelegateInterface RCS::RobotProgram::_delegate

crcl delegate used to interpret Crcl XML command

10.28.4.2 boost::mutex RCS::RobotProgram::_progmutex [static]

mutex for thread safe access to RobotProgram commands

10.28.4.3 std::string RCS::RobotProgram::_programname

saved RobotProgram program file path

10.28.4.4 int RCS::RobotProgram::cmdnum

number of Crcl XML command to execute

10.28.4.5 ::CRCLProgramType::MiddleCommand_sequence& RCS::RobotProgram::cmds

reference to crcl program XML commands (from codesynthesis parsing)

10.28.4.6 std::istringstream RCS::RobotProgram::istr

input stream interface for codesynthesis parsing

The documentation for this class was generated from the following files:

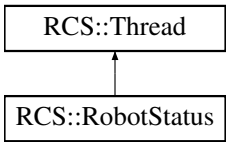
- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Controller.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Controller.cpp

10.29 RCS::RobotStatus Class Reference

The RobotStatus is a thread to updates the status of the robot. The RobotStatus is a separate thread that updates the robot status. Currently, it uses a JointReader to read joint values from the controller. It uses a Kinematics pointer reference to compute the current pose using the FK routine. It also uses a CrclDelegate pointer reference to update the status reported by CRCL.

`#include <Controller.h>`

Inheritance diagram for RCS::RobotStatus:



Public Member Functions

- RobotStatus (double cycletime=DEFAULT_LOOP_CYCLE)

  RobotStatus constructor that requires a cycle time for RCS thread timing.
- NVAR (CrclDelegate, boost::shared_ptr< Crcl::CrclDelegateInterface >, _crclinterface)
- VAR (JointReader, boost::shared_ptr< CJointReader >)
- VAR (Kinematics, boost::shared_ptr< IKinematics >)
- virtual void Action ()

  Action is the main loop in the RCS thread timing. Get latest robot joint readings. Use forward kinematics to get current pose. Then, updates the CRCL world model with the latest readings. Should it keep track of the command id also - in theory only one CRCl command at a time.
- bool Verify ()

  method to determine if the instance is valid, i.e., has all reference pointers.

Additional Inherited Members

10.29.1 Detailed Description

The RobotStatus is a thread to updates the status of the robot. The RobotStatus is a separate thread that updates the robot status. Currently, it uses a JointReader to read joint values from the controller. It uses a Kinematics pointer reference to compute the current pose using the FK routine. It also uses a CrclDelegate pointer reference to update the status reported by CRCL.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 RCS::RobotStatus::RobotStatus (double *cycletime* = DEFAULT_LOOP_CYCLE)

RobotStatus constructor that requires a cycle time for RCS thread timing.

Parameters

cycletime	in seconds.

10.29.3 Member Function Documentation

10.29.3.1 void RCS::RobotStatus::Action ( ) [virtual]

Action is the main loop in the RCS thread timing. Get latest robot joint readings. Use forward kinematics to get current pose. Then, updates the CRCL world model with the latest readings. Should it keep track of the command id also - in theory only one CRCl command at a time.

Reimplemented from RCS::Thread.

10.29.3.2 RCS::RobotStatus::NVAR (CrclDelegate , boost::shared_ptr< Crcl::CrclDelegateInterface > , _crclinterface)

10.29.3.3 RCS::RobotStatus::VAR (JointReader , boost::shared_ptr< CJointReader >)

10.29.3.4 RCS::RobotStatus::VAR (Kinematics , boost::shared_ptr< IKinematics >)

10.29.3.5 bool RCS::RobotStatus::Verify ( ) [inline]

method to determine if the instance is valid, i.e., has all reference pointers.

Returns

boolean to signify whether component is valid.

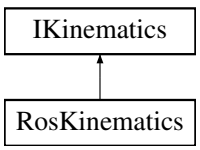
The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Controller.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Controller.cpp

10.30 RosKinematics Class Reference

`#include <Kinematics.h>`

Inheritance diagram for RosKinematics:



Public Member Functions

- RosKinematics ()
- virtual void Init (std::string groupname, std::string eelinkname)
- virtual std::vector< double > GetJointValues ()

- void SetJointValues (std::vector< double > joint_values)
- virtual urdf::Pose FK (std::vector< double > jv)
- virtual std::vector< double > IK (RCS::Pose &pose, std::vector< double > oldjoints)
- bool SatisfiesBounds ()
- void EnforceBounds ()
- virtual size_t AllPoseToJoints (RCS::Pose &pose, std::vector< std::vector< double > > &newjoints)
- virtual std::vector< double > NearestJoints (std::vector< double > oldjoints, std::vector< std::vector< double > > &newjoints)

Public Attributes

- robot_model::RobotModelPtr kinematic_model
- robot_state::RobotStatePtr kinematic_state
- robot_state::JointModelGroup * joint_model_group
- std::vector< double > joint_values
- std::vector< std::string > joint_names
- std::string _groupname
- std::string _eelinkname
- bool _bInit
- boost::mutex kinmutex

10.30.1 Constructor & Destructor Documentation

10.30.1.1 RosKinematics::RosKinematics ( )

10.30.2 Member Function Documentation

10.30.2.1 virtual size_t RosKinematics::AllPoseToJoints (RCS::Pose & *pose,* std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.30.2.2 void RosKinematics::EnforceBounds ( )

10.30.2.3 urdf::Pose RosKinematics::FK (std::vector< double > *jv*) [virtual]

Implements IKinematics.

10.30.2.4 std::vector< double > RosKinematics::GetJointValues ( ) [virtual]

Implements IKinematics.

10.30.2.5 std::vector< double > RosKinematics::IK (RCS::Pose & *pose,* std::vector< double > *oldjoints*) [virtual]

Implements IKinematics.

10.30.2.6 void RosKinematics::Init (std::string *groupname,* std::string *eelinkname*) [virtual]

Reimplemented from IKinematics.

10.30.2.7 virtual std::vector<double> RosKinematics::NearestJoints (std::vector< double > *oldjoints,* std::vector< std::vector< double > > & *newjoints*) [inline], [virtual]

Implements IKinematics.

10.30.2.8 bool RosKinematics::SatisfiesBounds ( )

10.30.2.9 void RosKinematics::SetJointValues (std::vector< double > *joint_values*) [virtual]

Implements IKinematics.

10.30.3 Member Data Documentation

10.30.3.1 bool RosKinematics::_bInit

10.30.3.2 std::string RosKinematics::_eelinkname

10.30.3.3 std::string RosKinematics::_groupname

10.30.3.4 robot_state::JointModelGroup* RosKinematics::joint_model_group

10.30.3.5 std::vector<std::string> RosKinematics::joint_names

10.30.3.6 std::vector<double> RosKinematics::joint_values

10.30.3.7 robot_model::RobotModelPtr RosKinematics::kinematic_model

10.30.3.8 robot_state::RobotStatePtr RosKinematics::kinematic_state

10.30.3.9 boost::mutex RosKinematics::kinmutex

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Kinematics.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/Kinematics.cpp

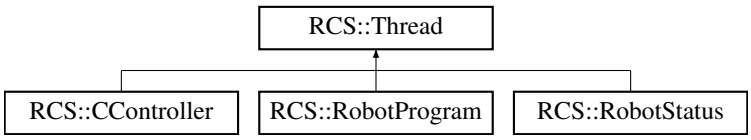
10.31 RCS::Thread Class Reference

Thread is an RCS ulapi equivalent for timed thread. Given a cycle time, the thread provides a wait function to sleep to exactly the amount of the thread cycle time. It keeps track of busy/idle time for diagnostic purposes.

Notes: https://www.quantnet.com/threads/c-multithreading-in-boost.10028/.

`#include <RCSThreadTemplate.h>`

Inheritance diagram for RCS::Thread:



Public Member Functions

- Thread (double cycletime)

  Constructor of thread, that takes cycle time as input.
- ~Thread ()

  Destructor of thread, makes sure thread has stopped.
- std::string & Name ()

  Name returns name of thread.
- void Join ()

  Uses boost thread join routine.
- virtual void Init ()

  Init function called before Action() loop.
- virtual void Cleanup ()

  Cleanup function called after Action() loop done.
- virtual void Action ()

  Action override function called every cycle.
- void Start ()

  Start starts the thread which call Init(), and then does Action() loop.
- void Stop (bool bWait=false)

  Stop stops the thread loop.
- void Suspend ()

  Suspend stops the thread loop until restarted with Resume().
- void Resume ()

  Resume resume execution of the thread loop stopped with Suspend().
- double Load ()

  Load returns the load of the thread cycle.
- double & CycleTime ()

  CycleTime returns the cycle time of the thread cycle in seconds.
- void SetDebugLevel (int n)

  SetDebugLevel sets the debugging level of the thread.
- int & DebugLevel ()

  DebugLevel returns the debugging level of the thread.
- void Cycle ()

  Cycle is the thread main function. It calls init, action, and cleanup. After each cycle waits exactly amount given by cycle time.

Static Public Member Functions

- static boost::thread_group & ThreadGroup ()

  ThreadGroup is a static definition of boost thread group.
- static std::vector< Thread * > & Threads ()

  Threads is a static definition of all the threads that have been created.
- static void StopAll ()

  Static StopAll which stops all the threads created in the boost thread group.

Protected Attributes

- std::string _name
- double _cycletime
- int _debugLevel
- bool _bThread
- bool _bDone
- RCS::Timer _timer
- boost::thread m_thread

10.31.1 Detailed Description

Thread is an RCS ulapi equivalent for timed thread. Given a cycle time, the thread provides a wait function to sleep to exactly the amount of the thread cycle time. It keeps track of busy/idle time for diagnostic purposes.

Notes: https://www.quantnet.com/threads/c-multithreading-in-boost.10028/.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 RCS::Thread::Thread (double *cycletime*) [inline]

Constructor of thread, that takes cycle time as input.

10.31.2.2 RCS::Thread::~Thread ( ) [inline]

Destructor of thread, makes sure thread has stopped.

10.31.3 Member Function Documentation

10.31.3.1 virtual void RCS::Thread::Action ( ) [inline], [virtual]

Action override function called every cycle.

Reimplemented in RCS::RobotProgram, RCS::RobotStatus, and RCS::CController.

10.31.3.2 virtual void RCS::Thread::Cleanup ( ) [inline], [virtual]

Cleanup function called after Action() loop done.

10.31.3.3 void RCS::Thread::Cycle ( ) [inline]

Cycle is the thread main function. It calls init, action, and cleanup. After each cycle waits exactly amount given by cycle time.

10.31.3.4 double& RCS::Thread::CycleTime ( ) [inline]

CycleTime returns the cycle time of the thread cycle in seconds.

Returns

double returns cycle time of thread in seconds.

10.31.3.5 int& RCS::Thread::DebugLevel ( ) [inline]

DebugLevel returns the debugging level of the thread.

Returns

int returns debug dlvel of thread.

10.31.3.6 virtual void RCS::Thread::Init ( ) [inline], [virtual]

Init function called before Action() loop.

Reimplemented in RCS::CController.

10.31.3.7 void RCS::Thread::Join ( ) [inline]

Uses boost thread join routine.

10.31.3.8 double RCS::Thread::Load ( ) [inline]

Load returns the load of the thread cycle.

10.31.3.9 std::string& RCS::Thread::Name ( ) [inline]

Name returns name of thread.

10.31.3.10 void RCS::Thread::Resume ( ) [inline]

Resume resume execution of the thread loop stopped with Suspend().

10.31.3.11 void RCS::Thread::SetDebugLevel (int *n*) [inline]

SetDebugLevel sets the debugging level of the thread.

Parameters

int	specified debug level, as an integer.

10.31.3.12 void RCS::Thread::Start ( ) [inline]

Start starts the thread which call Init(), and then does Action() loop.

10.31.3.13 void RCS::Thread::Stop (bool *bWait* = false) [inline]

Stop stops the thread loop.

Parameters

bWait	indicates whether to wait until thread has finished.

10.31.3.14 static void RCS::Thread::StopAll ( ) [inline], [static]

Static StopAll which stops all the threads created in the boost thread group.

10.31.3.15 void RCS::Thread::Suspend ( ) [inline]

Suspend stops the thread loop until restarted with Resume().

10.31.3.16 static boost::thread_group& RCS::Thread::ThreadGroup ( ) [inline], [static]

ThreadGroup is a static definition of boost thread group.

10.31.3.17 static std::vector<Thread *>& RCS::Thread::Threads ( ) [inline], [static]

Threads is a static definition of all the threads that have been created.

10.31.4 Member Data Documentation

10.31.4.1 bool RCS::Thread::_bDone [protected]

boolean indicating whether thread has finished

10.31.4.2 bool RCS::Thread::_bThread [protected]

boolean loop thread

10.31.4.3 double RCS::Thread::_cycletime [protected]

cycletime of thread in seconds

10.31.4.4 int RCS::Thread::_debugLevel [protected]

debug level of thread

10.31.4.5 std::string RCS::Thread::_name [protected]

name of thread

10.31.4.6 RCS::Timer RCS::Thread::_timer [protected]

RCS timer for coordinating wait and duration of thread

10.31.4.7 boost::thread RCS::Thread::m_thread [protected]

boost thread

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSThreadTemplate.h

10.32 RCS::Timer Class Reference

Timer is a general-purpose timer. The Timer is a general-purpose timer, which can be used for waiting until a synchronous time tick, slept on for any period at all, or to obtain a time in system clock ticks from creation of the timer.

`#include <RCSTimer.h>`

Public Member Functions

- Timer (double _timeout, RCS_TIMERFUNC _function=(RCS_TIMERFUNC) 0)
  timeout is wait interval, rounded up to clock tick resolution; function is external time base, if provided.
- void esleep (double seconds_to_sleep)
  sleep number of seconds to sleep.
- boost::chrono::high_resolution_clock::time_point etime ()
  number of seconds from some epoch, to clock tick resolution.
- double clk_tck ()
  number of clock ticks per second using high resolution timer.
- int wait ()
  wait on synch; returns # of cycles missed.
- double load ()
  Returns % loading on timer, 0.0 means all waits, 1.0 means no time in wait. This is average load.
- double free ()
  Compute free time over all cycles.
- void sync ()
  Synchronize the timing service. Initialize start time and last time called to current time since epoch.
- void suspend ()
  Suspend the timing.
- void resume ()
  Resume the timing. Wakeup timer with boost conditional notify.

Static Public Member Functions

- static double & last_esleep_seconds_to_sleep ()

  return last sleep number of seconds to slept.
- static int & etime_disabled ()
- static double & etime_disable_time ()

10.32.1 Detailed Description

Timer is a general-purpose timer. The Timer is a general-purpose timer, which can be used for waiting until a synchronous time tick, slept on for any period at all, or to obtain a time in system clock ticks from creation of the timer.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 RCS::Timer::Timer (double *_timeout,* RCS_TIMERFUNC *_function* = (RCS_TIMERFUNC) 0) [inline]

timeout is wait interval, rounded up to clock tick resolution; function is external time base, if provided.

Parameters

timeout	period.

10.32.3 Member Function Documentation

10.32.3.1 double RCS::Timer::clk_tck ( ) [inline]

number of clock ticks per second using high resolution timer.

Returns

number of ticks per second.

10.32.3.2 void RCS::Timer::esleep (double *seconds_to_sleep*) [inline]

sleep number of seconds to sleep.

Parameters

seconds	(or fractions) to sleep. Must be positive.

10.32.3.3 boost::chrono::high_resolution_clock::time_point RCS::Timer::etime ( ) [inline]

number of seconds from some epoch, to clock tick resolution.

Returns

high_resolution_clock now

10.32.3.4 static double& RCS::Timer::etime_disable_time ( ) [inline], [static]

10.32.3.5 static int& RCS::Timer::etime_disabled ( ) [inline], [static]

10.32.3.6 double RCS::Timer::free ( ) [inline]

Compute free time over all cycles.

10.32.3.7 static double& RCS::Timer::last_esleep_seconds_to_sleep ( ) [inline], [static]

return last sleep number of seconds to slept.

Returns

last seconds (or fractions) last slept. -199.99 if unused.

10.32.3.8 double RCS::Timer::load ( ) [inline]

Returns % loading on timer, 0.0 means all waits, 1.0 means no time in wait. This is average load.

Returns

double or -1 of time spent busy.

10.32.3.9 void RCS::Timer::resume ( ) [inline]

Resume the timing. Wakeup timer with boost conditional notify.

10.32.3.10 void RCS::Timer::suspend ( ) [inline]

Suspend the timing.

10.32.3.11 void RCS::Timer::sync ( ) [inline]

Synchronize the timing service. Initialize start time and last time called to current time since epoch.

10.32.3.12 int RCS::Timer::wait ( ) [inline]

wait on synch; returns # of cycles missed.

Returns

# of cycles missed.

The documentation for this class was generated from the following file:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSTimer.h

10.33 TrajectoryMaker Class Reference

TrajectoryMaker generates simple trapezoidal velocities. Will accept non-zero final velocity.

`#include <trajectoryMaker.h>`

Public Member Functions

- TrajectoryMaker ()

  constuctor.
- std::vector< JointState > GetJtsPlan ()

  GetJtsPlan returns vector of generated joint state trajectories.
- std::vector< urdf::Pose > GetPosesPlan ()

  GetPosesPlan returns vector of generated pose trajectories.
- void setRates (IRate rates)

  setRates defines the rate to use when generating trajectory.
- bool Plan (JointState curjoints, JointState goaljoints)

  plan a joint trajectory based on current and goal joint states. Assumes rate already set.
- bool Plan (urdf::Pose &curpose, urdf::Pose &goalpose)

  plan a cartesian trajectory based on current and goal pose states. Assumes rate already set.
- bool Plan (std::vector< urdf::Pose > &waypoints)

  plan a cartesian trajectory given a vector of waypoint poses. Assumes rate already set.
- std::vector< double > makePositionVector (std::vector< double > ramp, double start, double end)

  makePositionVector generates a vector of from start to end point.
- bool makeJointPositionTrajectory (IRate rates, JointState &curjoints, JointState &goaljoints)

  makeJointPositionTrajectory constructs a joint trajectory based on current and goal joint states given a rate profile.
- bool makeJointPositionTrajectory (IRate rates, std::vector< double > &curjoints, std::vector< double > &goaljoints)

  makeJointPositionTrajectory constructs a joint trajectory based on current and goal joint states given a rate profile.
- std::vector< double > makeStopJointTrajectory (double startingVelocity, double finalVelocity, double maxAcc, double cycleTime, double current)

  makeStopJointTrajectory constructs a stopping trajectory based on current velocity.
- std::vector< urdf::Pose > makeCartesianTrajectory (IRate rates, urdf::Pose _curPos, urdf::Pose _goalPos)

  makeCartesianTrajectory plans a cartesian trajectory based on current and goal pose states for the given rate profile.
- IRate & Rates ()

  Reference to rates data structure.

Protected Member Functions

- void updateJointCommands (std::vector< double > &curjoints, std::vector< std::vector< double > > &displacements)
- std::vector< urdf::Pose > makeTrajectory (RCS::CanonWorldModel *parameters, urdf::Pose goal, urdf::Pose current)
- std::vector< double > makeJointValues (double current, std::vector< double > displacements)
- std::vector< double > makeJointTrajectory (double current, double goal)
- void setCurrent (urdf::Pose current)
- std::vector< double > makePositionRamp (double maxVel, double maxAccel, double cycletime)
- double makeDeclRamp (double startingVelocity, double finalVelocity, double maxAcc, double cycleTime, std::vector< double > &declRamp)

- double makeAcclRamp (double startingVelocity, double finalVelocity, double maxVelocity, double maxAcc, double cycleTime, std::vector< double > &acclRamp)
- std::vector< boost::tuple < double, double, double > > makeTupleRamp (double maxVelocity, double maxAcc, double cycleTime)
- std::vector< urdf::Pose > makeCartesianTrajectory (double final_velocity, double current_feedrate, double current_velocity, double maximum_accel, double cycleTime, urdf::Pose _curPos, urdf::Pose _goalPos)
- double runTrapezoidalCycle (IRate &trans, double distance_to_go)
- double makeNRamp (int N, double maxVelocity, double maxAcc, double cycleTime, std::vector< double > &acclramp)

10.33.1 Detailed Description

TrajectoryMaker generates simple trapezoidal velocities. Will accept non-zero final velocity.

Author

Stephen Balakirsky, GTRI

Date

July 30, 2014

Copyright

Georgia Tech Research Institute

10.33.2 Constructor & Destructor Documentation

10.33.2.1 TrajectoryMaker::TrajectoryMaker ( )

constuctor.

Constructor for CurrentLocation that sets all points and velocity to 0.

10.33.3 Member Function Documentation

10.33.3.1 std::vector< JointState > TrajectoryMaker::GetJtsPlan ( )

GetJtsPlan returns vector of generated joint state trajectories.

Returns

returns vector of joint state

10.33.3.2 std::vector< urdf::Pose > TrajectoryMaker::GetPosesPlan ( )

GetPosesPlan returns vector of generated pose trajectories.

Returns

returns vector of poses

10.33.3.3 double TrajectoryMaker::makeAcclRamp (double *startingVelocity,* double *finalVelocity,* double *maxVelocity,* double *maxAcc,* double *cycleTime,* std::vector< double > & *acclRamp*) [protected]

max accel is computed by parameters->getMaxAccel(movetype) * parameters->getCycleTime();

10.33.3.4 std::vector< urdf::Pose > TrajectoryMaker::makeCartesianTrajectory (IRate *rates,* urdf::Pose *_curPos,* urdf::Pose *_goalPos*)

makeCartesianTrajectory plans a cartesian trajectory based on current and goal pose states for the given rate profile.

Parameters

rates	defines the motion parameters.
curpose	current pose definition.
goalpose	goal pose definition.

Returns

vector of generated cartesian poses trajectory from start to goal.

10.33.3.5 std::vector< urdf::Pose > TrajectoryMaker::makeCartesianTrajectory (double *final_velocity,* double *current_feedrate,* double *current_velocity,* double *maximum_accel,* double *cycleTime,* urdf::Pose *_curPos,* urdf::Pose *_goalPos*) [protected]

10.33.3.6 double TrajectoryMaker::makeDeclRamp (double *startingVelocity,* double *finalVelocity,* double *maxAcc,* double *cycleTime,* std::vector< double > & *declRamp*) [protected]

max accel is computed by parameters->getMaxAccel(movetype) * parameters->getCycleTime();

10.33.3.7 bool TrajectoryMaker::makeJointPositionTrajectory (IRate *rates,* JointState & *curjoints,* JointState & *goaljoints*)

makeJointPositionTrajectory constructs a joint trajectory based on current and goal joint states given a rate profile.

Parameters

rates	defines motion characteristics.
curjoints	current joint state definition.
goaljoints	goal joint state definition.

Returns

true if successful joint state trajectory was generated.

10.33.3.8 bool TrajectoryMaker::makeJointPositionTrajectory (IRate *rates,* std::vector< double > & *curjoints,* std::vector< double > & *goaljoints*)

makeJointPositionTrajectory constructs a joint trajectory based on current and goal joint states given a rate profile.

Parameters

rates	defines motion characteristics.
curjoints	double vector of current joint position definition.
goaljoints	double vector of goal joint position definition.

Returns

true if successful joint state trajectory was generated.

10.33.3.9 std::vector< double > TrajectoryMaker::makeJointTrajectory (double *current,* double *goal*) [protected]

10.33.3.10 std::vector< double > TrajectoryMaker::makeJointValues (double *current,* std::vector< double > *displacements*) [protected]

10.33.3.11 double TrajectoryMaker::makeNRamp (int *N,* double *maxVelocity,* double *maxAcc,* double *cycleTime,* std::vector< double > & *acclramp*) [protected]

10.33.3.12 std::vector< double > TrajectoryMaker::makePositionRamp (double *maxVel,* double *maxAccel,* double *cycletime*) [protected]

10.33.3.13 std::vector< double > TrajectoryMaker::makePositionVector (std::vector< double > *myramp,* double *start,* double *end*)

makePositionVector generates a vector of from start to end point.

Parameters

ramp	vector of incremental distances up to max velocity attained.
start	defines starting position
end	defines ending position

Returns

vector of points defining trajectory of given velocity profile.

This function creates a vector of doubles that determines the trajectory of the Robot depending on the start position, the end position, the max velocity, and the acceleration. Shows the position values at times incrementing by the cycle time of the robot.

Parameters

start	The start position of the Robot.
end	The desired end position of the Robot.
maxSpeed	The max velocity the Robot can reach.
acc	The acceleration of the Robot.

Returns

The vector of doubles that show the position of the Robot every cycleTime milliseconds.

10.33.3.14 std::vector< double > TrajectoryMaker::makeStopJointTrajectory (double *startingVelocity,* double *finalVelocity,* double *maxAcc,* double *cycleTime,* double *current*)

makeStopJointTrajectory constructs a stopping trajectory based on current velocity.

Parameters

finalVelocity	should be zero.
maxAcc	given maximum deceleration rate.
cycleTime	gives the cycle time for the acceleration rate.
current	is the current double position

Returns

vector of offset distances from current position to stop.

10.33.3.15 std::vector<urdf::Pose> TrajectoryMaker::makeTrajectory (RCS::CanonWorldModel * *parameters,* urdf::Pose *goal,* urdf::Pose *current*) [protected]

10.33.3.16 std::vector< boost::tuple< double, double, double > > TrajectoryMaker::makeTupleRamp (double *maxVelocity,* double *maxAcc,* double *cycleTime*) [protected]

10.33.3.17 bool TrajectoryMaker::Plan (JointState *curjoints,* JointState *goaljoints*)

plan a joint trajectory based on current and goal joint states. Assumes rate already set.

Parameters

curjoints	current joint state definition.
goaljoints	goal joint state definition.

Returns

true if successful joint state trajectory was generated.

10.33.3.18 bool TrajectoryMaker::Plan (urdf::Pose & *curpose,* urdf::Pose & *goalpose*)

plan a cartesian trajectory based on current and goal pose states. Assumes rate already set.

Parameters

curpose	current pose definition.
goalpose	goal pose definition.

Returns

true if successful cartesian trajectory was generated.

10.33.3.19 bool TrajectoryMaker::Plan (std::vector< urdf::Pose > & *waypoints*)

plan a cartesian trajectory given a vector of waypoint poses. Assumes rate already set.

Parameters

waypoints	vector of intermediate pose definition.

Returns

true if successful cartesian trajectory was generated.

10.33.3.20 IRate& TrajectoryMaker::Rates ( ) [inline]

Reference to rates data structure.

10.33.3.21 double TrajectoryMaker::runTrapezoidalCycle (IRate & *trans*, double *distance_to_go*) [protected]

10.33.3.22 void TrajectoryMaker::setCurrent (urdf::Pose *poseIn*) [protected]

Set the current position.

Set the current location from the input pose

10.33.3.23 void TrajectoryMaker::setRates (IRate *rates*)

setRates defines the rate to use when generating trajectory.

Parameters

rates	contains the IRate definition.

10.33.3.24 void TrajectoryMaker::updateJointCommands (std::vector< double > & *curjoints*, std::vector< std::vector< double > > & *displacements*) [protected]

The documentation for this class was generated from the following files:

- /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/trajectoryMaker.h
- /home/michalos/catkin_ws/src/nist_fanuc/src/trajectoryMaker.cpp

Chapter 11

File Documentation

11.1 Installation.md File Reference

11.2 JointReader.md File Reference

11.3 JointWriter.md File Reference

11.4 Readme.md File Reference

11.5 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/AsioCrclServer.h File Reference

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <deque>
#include <set>
#include <boost/bind.hpp>
#include <boost/smart_ptr.hpp>
#include <boost/asio.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include "CrclInterface.h"
#include "RCSMsgQueue.h"
```

Classes

- class CAsioCrclSession

  The CAsioCrclSession provides an boost asio session (which listens for each connected client). The CAsioCrclSession

listens for XML messages and constructs. The CAsioCrclSession uses mostly asynchronous operation for waiting, reading, and timeout of a socket connection. The operation is started by creating a session which starts an aynchronous thread, that is supplied IO communication events by the asio io service provider. After connection to the socket client, an StartAyncRead() that is paired with a timer is used to wait for communicatin from a socket. There is no trailing marker on CRCL XML so any socket communication must be buffered and when a complete message has been received, it is pushed onto the inmsgs message queue. During the socket communication, a timeout can occur, which at this point only causes a new to beStartAyncRead() initiated. Because CRCL Xml does not have a trailing marker (e.g., zero or line feed), the CAsioCrclSession must determine the trailing XML tag to search for, by inspecting the communication for a XML leading tag. It works, but is dubious. However, if the communicating socket is disconnected, an error is returned by asio, and the session is terminated cleanly.
Useful web sites:

- class CAsioCrclServer

  The CAsioCrclServer provides an boost asio server which accepts new connections and starts a Crcl listener session. The CAsioCrclServer is based on the Boost Asio library which can process network communication asynchronously. Because CRCL data can only be received after a connection has been established, and because a connection can only be established after the name has been resolved, the various asynchronous operations are started in separate callback handlers. Thus in boost asio a callback to async_connect() is then followed by a method call to the handler connect-_handler() which starts a new Crcl session. Readers can read more at: `http://theboostcpplibraries.-com/boost.asio-network-programming` *The CAsioCrclServer is divided into a number of main funcitons (e.g. wait for socket connection, handle new session by spawning new CAsioCrclSession, repeat. These operations are done asynchronously on a separate thread with notification done by the boost asio io server and it is assumed to be thread-safe. The CAsioCrclServer listens for connections on port 64444 and when a connection is initiated starts a new Crcl session to read xml messages from the devices.*

Typedefs

- typedef boost::system::error_code error_code
- typedef boost::tuple
  < std::string,
  CAsioCrclSession * > CrclMessage
- typedef RCS::CMessageQueue
  < CrclMessage > CAsioMessages
- typedef boost::shared_ptr
  < CAsioCrclSession > session_ptr

Variables

- boost::asio::io_service myios

11.5.1 Typedef Documentation

11.5.1.1 typedef RCS::CMessageQueue<CrclMessage> CAsioMessages

11.5.1.2 typedef boost::tuple<std::string, CAsioCrclSession *> CrclMessage

11.5.1.3 typedef boost::system::error_code error_code

11.5.1.4 typedef boost::shared_ptr<CAsioCrclSession> session_ptr

11.5.2 Variable Documentation

11.5.2.1 boost::asio::io_service myios

11.6 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/ChainRobotModel.h File Reference

```
#include <string>
#include <vector>
#include <boost/format.hpp>
#include <boost/shared_ptr.hpp>
#include <Eigen/Dense>
```

Classes

- struct RCS::RdfJoint
- class RCS::ChainRobotModel

Namespaces

- RCS

11.7 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Communication.h File Reference

```
#include <boost/thread/mutex.hpp>
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>
#include <moveit/move_group_interface/move_group.h>
#include <actionlib/client/simple_action_client.h>
#include <std_msgs/Float64.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <control_msgs/JointTrajectoryControllerState.h>
#include <control_msgs/QueryTrajectoryState.h>
```

Classes

- class CJointReader

  The CJointReader is a thread to accept joint update callbacks from ROS. Uses a ros node handle to tell roscore we are subscribing to joint_state topic. Then, when joint updates occur, the callback routine is invoked and the latest joint values saved.

- class CJointWriter

  The CJointWriter is a thread to publish new joint values to ROS. Uses a ros node handle to tell roscore we are pusblishing to the joint_path_command topic. Then, when joint updates occur, these are published on joint_path_command the topic.

11.8 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Controller.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <list>
#include "AsioCrclServer.h"
#include "RCSThreadTemplate.h"
#include "CrclInterface.h"
#include "RCSInterpreter.h"
#include "ChainRobotModel.h"
#include "Trajectory.h"
#include "Communication.h"
#include "moveit.h"
```

Classes

- struct RCS::CController

  The CController provides an collection for all the relevant controller pieces. The CController is the main controller class to collect all the references/pointers to instances in the project. A global instance, call Controller, is created that is used through out the code to reference various instances of control objects (e.g., kinematics, joint writer, joint reader, etc.)

- class RCS::RobotStatus

  The RobotStatus is a thread to updates the status of the robot. The RobotStatus is a separate thread that updates the robot status. Currently, it uses a JointReader to read joint values from the controller. It uses a Kinematics pointer reference to compute the current pose using the FK routine. It also uses a CrclDelegate pointer reference to update the status reported by CRCL.

- class RCS::RobotProgram

  The RobotProgram is a thread to handle crcl programs. Crcl programs are not in fact legitimate, however, debugging and verification are assisted by programs. However, program as in the Crcl XSD specification, so it doesn't hurt to handle. They require special handling as only one command should be done at a time. Uses codesynthesis to parse Crcl xml into C++ data structures.

Namespaces

- RCS

11.9 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Conversions.h File Reference

```
#include <Eigen/Core>
#include <Eigen/Geometry>
#include "geometry_msgs/PoseStamped.h"
#include "urdf_model/eigenmath.h"
#include "urdf_model/rosmath.h"
#include "RCS.h"
```

Functions

- urdf::Pose PoseMsg2UrdfPose (const geometry_msgs::Pose &m)
- geometry_msgs::Pose UrdfPose2PoseMsg (const urdf::Pose &m)
- Eigen::Affine3d PoseMsgToEigenAffine (const geometry_msgs::Pose &m)
- geometry_msgs::Pose poseEigenToGeomMsg (const Eigen::Affine3d &e)
- std::vector< geometry_msgs::Pose > UrdfPoses2PoseMsgs (const std::vector< urdf::Pose > &src)
- std::vector< urdf::Pose > PoseMsgs2UrdfPoses (const std::vector< geometry_msgs::Pose > &src)
- std::vector< Eigen::Affine3d > PoseMsgs2AffEigenPoses (const std::vector< geometry_msgs::Pose > &src)
- std::vector< geometry_msgs::Pose > AffEigenPoses2PoseMsgs (const std::vector< Eigen::Affine3d > &src)
- JointState Vector2JointState (const std::vector< double > &src)
- JointState JntPosVector2JointState (const std::vector< double > &src)

11.9.1 Function Documentation

11.9.1.1 std::vector< geometry_msgs::Pose > AffEigenPoses2PoseMsgs (const std::vector< Eigen::Affine3d > & *src*) [inline]

11.9.1.2 JointState JntPosVector2JointState (const std::vector< double > & *src*) [inline]

11.9.1.3 geometry_msgs::Pose poseEigenToGeomMsg (const Eigen::Affine3d & *e*) [inline]

11.9.1.4 urdf::Pose PoseMsg2UrdfPose (const geometry_msgs::Pose & *m*) [inline]

11.9.1.5 std::vector< Eigen::Affine3d > PoseMsgs2AffEigenPoses (const std::vector< geometry_msgs::Pose > & *src*) [inline]

11.9.1.6 std::vector< urdf::Pose > PoseMsgs2UrdfPoses (const std::vector< geometry_msgs::Pose > & *src*) [inline]

11.9.1.7 Eigen::Affine3d PoseMsgToEigenAffine (const geometry_msgs::Pose & *m*) [inline]

11.9.1.8 geometry_msgs::Pose UrdfPose2PoseMsg (const urdf::Pose & *m*) [inline]

11.9.1.9 std::vector< geometry_msgs::Pose > UrdfPoses2PoseMsgs (const std::vector< urdf::Pose > & *src*) [inline]

11.9.1.10 JointState Vector2JointState (const std::vector< double > & *src*)

11.10 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/crcl.h File Reference

```
#include <math.h>
#include "RCS.h"
#include "DataPrimitives.hxx"
#include "CRCLCommands.hxx"
#include "CRCLStatus.hxx"
#include "CRCLCommandInstance.hxx"
#include "CRCLProgramInstance.hxx"
#include "Globals.h"
```

Classes

- struct Crcl::GripperStatus
- struct Crcl::JointReport
- struct Crcl::CrclStatus

Namespaces

- Crcl

Macros

- #define _USE_MATH_DEFINES

Typedefs

- typedef urdf::Vector3 Crcl::Vector3D

Enumerations

- enum Crcl::CRCLCmdStatus { Crcl::CRCL_DONE = 0, Crcl::CRCL_WORKING, Crcl::CRCL_ERROR }
- enum Crcl::CrclReturn {
  Crcl::CANON_REJECT = -2, Crcl::CANON_FAILURE = -1, Crcl::CANON_SUCCESS = 0, Crcl::CANON_STAT-
  USREPLY = 1,
  Crcl::CANON_RUNNING }

Functions

- std::vector< double > Crcl::ConvertToPositionVector (ActuatorJointSequence &, double dConversion)
- JointStatusSequence Crcl::Convert (ActuatorJointSequence jin)
- urdf::Pose Crcl::Convert (Crcl::PoseType &pose, double lengthConversion)
- Crcl::JointStatusSequence Crcl::Convert (JointState joints)
- sensor_msgs::JointState Crcl::Convert (Crcl::JointStatusSequence jout, double angleConversion)
- Crcl::PoseType Crcl::Init (std::vector< double > terms)
- Crcl::PoseType Crcl::IdentityPose ()
- std::string Crcl::DumpPose (Crcl::PoseType pose, std::string separator)
- urdf::Vector3 Crcl::GetVector3D (Crcl::PointType &point)
- urdf::Vector3 Crcl::GetVector3D (Crcl::VectorType &vector)
- bool Crcl::GetPoseToRPY (Crcl::PoseType &pose, double &dRoll, double &dPitch, double &dYaw)
- RosMatrix Crcl::GetXZRotMatrix (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- bool Crcl::GetRPY (urdf::Vector3 Xrot, urdf::Vector3 Zrot, double &roll, double &pitch, double &yaw)
- urdf::Rotation Crcl::Convert (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- Crcl::PoseType Crcl::NullPose ()
- Crcl::PoseType Crcl::PoseHome ()
- Crcl::VectorType Crcl::VectorZero ()
- std::ostream & Crcl::operator<< (std::ostream &os, const Crcl::PoseType &pose)

Variables

- typedef::ActuateJointsType::ActuateJoint_sequence Crcl::ActuatorJointSequence
- typedef::PoseType Crcl::PoseType
- typedef::JointStatusType Crcl::JointStatus
- typedef::CommandStateEnumType Crcl::CommandStateEnum
- typedef::PointType Crcl::PointType
- typedef::VectorType Crcl::VectorType
- typedef::JointStatusesType::JointStatus_sequence Crcl::JointStatusSequence
- typedef::PoseToleranceType Crcl::PoseToleranceType

11.10.1 Macro Definition Documentation

11.10.1.1 #define _USE_MATH_DEFINES

11.11 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/CrclConfig.h File Reference

11.12 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/CrclInterface.h File Reference

```
#include <string>
#include "crcl.h"
#include <iostream>
#include <vector>
#include "Globals.h"
```

Classes

- class Crcl::CrclDelegateInterface
- class Crcl::CrclClientCmdInterface
- class Crcl::CrclStatusMsgInterface

Namespaces

- Crcl

11.13 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/eigen_msg_conversions.cpp File Reference

```
#include <eigen_conversions/eigen_msg.h>
```

Namespaces

- tf

Functions

- void tf::pointMsgToEigen (const geometry_msgs::Point &m, Eigen::Vector3d &e)

  Converts a Point message into an Eigen Vector.
- void tf::pointEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Point &m)

  Converts an Eigen Vector into a Point message.
- void tf::poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Affine3d &e)

  Converts a Pose message into an Eigen Affine3d.
- void tf::poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Isometry3d &e)

  Converts a Pose message into an Eigen Isometry3d.
- void tf::poseEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Pose &m)

  Converts an Eigen Affine3d into a Pose message.
- void tf::poseEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Pose &m)

  Converts an Eigen Isometry3d into a Pose message.
- void tf::quaternionMsgToEigen (const geometry_msgs::Quaternion &m, Eigen::Quaterniond &e)

  Converts a Quaternion message into an Eigen Quaternion.
- void tf::quaternionEigenToMsg (const Eigen::Quaterniond &e, geometry_msgs::Quaternion &m)

  Converts an Eigen Quaternion into a Quaternion message.
- void tf::transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Affine3d &e)

  Converts a Transform message into an Eigen Affine3d.
- void tf::transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Isometry3d &e)

  Converts a Transform message into an Eigen Isometry3d.
- void tf::transformEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Affine3d into a Transform message.
- void tf::transformEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Isometry3d into a Transform message.
- void tf::vectorMsgToEigen (const geometry_msgs::Vector3 &m, Eigen::Vector3d &e)

  Converts a Vector message into an Eigen Vector.
- void tf::vectorEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Vector3 &m)

  Converts an Eigen Vector into a Vector message.
- void tf::twistMsgToEigen (const geometry_msgs::Twist &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Twist message into an Eigen matrix.
- void tf::twistEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Twist &m)

  Converts an Eigen matrix into a Twist message.
- void tf::wrenchMsgToEigen (const geometry_msgs::Wrench &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Wrench message into an Eigen matrix.
- void tf::wrenchEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Wrench &m)

  Converts an Eigen matrix into a Wrench message.

11.14 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/eigen_msg_conversions.h File Reference

```
#include <std_msgs/Float64MultiArray.h>
#include <geometry_msgs/Point.h>
#include <geometry_msgs/Pose.h>
#include <geometry_msgs/Quaternion.h>
#include <geometry_msgs/Transform.h>
#include <geometry_msgs/Twist.h>
#include <geometry_msgs/Vector3.h>
#include <geometry_msgs/Wrench.h>
#include <Eigen/Core>
#include <Eigen/Geometry>
```

Namespaces

- tf

Functions

- void tf::pointMsgToEigen (const geometry_msgs::Point &m, Eigen::Vector3d &e)

  Converts a Point message into an Eigen Vector.
- void tf::pointEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Point &m)

  Converts an Eigen Vector into a Point message.
- void tf::poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Affine3d &e)

  Converts a Pose message into an Eigen Affine3d.
- void tf::poseMsgToEigen (const geometry_msgs::Pose &m, Eigen::Isometry3d &e)

  Converts a Pose message into an Eigen Isometry3d.
- void tf::poseEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Pose &m)

  Converts an Eigen Affine3d into a Pose message.
- void tf::poseEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Pose &m)

  Converts an Eigen Isometry3d into a Pose message.
- void tf::quaternionMsgToEigen (const geometry_msgs::Quaternion &m, Eigen::Quaterniond &e)

  Converts a Quaternion message into an Eigen Quaternion.
- void tf::quaternionEigenToMsg (const Eigen::Quaterniond &e, geometry_msgs::Quaternion &m)

  Converts an Eigen Quaternion into a Quaternion message.
- void tf::transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Affine3d &e)

  Converts a Transform message into an Eigen Affine3d.
- void tf::transformMsgToEigen (const geometry_msgs::Transform &m, Eigen::Isometry3d &e)

  Converts a Transform message into an Eigen Isometry3d.
- void tf::transformEigenToMsg (const Eigen::Affine3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Affine3d into a Transform message.
- void tf::transformEigenToMsg (const Eigen::Isometry3d &e, geometry_msgs::Transform &m)

  Converts an Eigen Isometry3d into a Transform message.
- void tf::twistMsgToEigen (const geometry_msgs::Twist &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Twist message into an Eigen matrix.
- void tf::twistEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Twist &m)

Converts an Eigen matrix into a Twist message.

- void tf::vectorMsgToEigen (const geometry_msgs::Vector3 &m, Eigen::Vector3d &e)

  Converts a Vector message into an Eigen Vector.

- void tf::vectorEigenToMsg (const Eigen::Vector3d &e, geometry_msgs::Vector3 &m)

  Converts an Eigen Vector into a Vector message.

- void tf::wrenchMsgToEigen (const geometry_msgs::Wrench &m, Eigen::Matrix< double, 6, 1 > &e)

  Converts a Wrench message into an Eigen matrix.

- void tf::wrenchEigenToMsg (const Eigen::Matrix< double, 6, 1 > &e, geometry_msgs::Wrench &m)

  Converts an Eigen matrix into a Wrench message.

- template<class Derived >
  void tf::matrixEigenToMsg (const Eigen::MatrixBase< Derived > &e, std_msgs::Float64MultiArray &m)

  Converts an Eigen matrix into a Float64MultiArray message.

11.15 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Globals.h File Reference

```
#include <stdio.h>
#include <vector>
#include <map>
#include <string>
#include <fstream>
#include <boost/thread.hpp>
#include <ctime>
#include <stdarg.h>
#include <sstream>
#include <time.h>
#include "Logging.h"
```

Classes

- class CGlobals

  CGlobals is a catch-all data structure for collecting global functions, extensions, parameters, etc. Functions here usually vary between windows and linux, or there is no easy mechanism in C++ to extend classes (e.g., string) like in C#.

Macros

- #define IfDebug(arg)
- #define _strnicmp strncasecmp
- #define CLEANSTORE(Y, X, Z)
- #define VALIDSTORE(Y, X)
- #define VAR(X, Y)
- #define NVAR(X, Y, Z)
- #define FOREACH(it, v) for(typeof((v).begin()) it = (v).begin(); it != (v).end(); it++)

Functions

- void DebugBreak ()
- template<typename T >
  std::string VectorDump (std::vector< T > v)
- template<typename T >
  std::vector< T > ToVector (int n,...)

Variables

- CGlobals Globals

11.15.1 Macro Definition Documentation

11.15.1.1 #define _strnicmp strncasecmp

11.15.1.2 #define CLEANSTORE(*Y, X, Z*)

Value:

```
try{ Y = X; }           \
    catch ( ... ) { Y = Z; }
```

11.15.1.3 #define FOREACH(*it, v*) for(typeof((v).begin()) it = (v).begin(); it != (v).end(); it++)

11.15.1.4 #define IfDebug(*arg*)

11.15.1.5 #define NVAR(*X, Y, Z*)

Value:

```
protected: Y Z;       \
public: Y & X( ) { return Z; }
```

11.15.1.6 #define VALIDSTORE(*Y, X*)

Value:

```
try{ Y = X; }        \
    catch ( ... ) { }
```

11.15.1.7 #define VAR(*X, Y*)

Value:

```
protected: Y _ ## X; \
public: Y & X( ) { return _ ## X; }
```

11.15.2 Function Documentation

11.15.2.1 void DebugBreak ( )

global definition of windows DebugBreak equivalent.

11.15.2.2 template<typename T > std::vector<T> ToVector (int *n*, ...) [inline]

11.15.2.3 template<typename T > std::string VectorDump (std::vector< T > *v*) [inline]

11.15.3 Variable Documentation

11.15.3.1 CGlobals Globals

global definition of globals

11.16 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Kinematics.h File Reference

```
#include "RCS.h"
#include "Globals.h"
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <vector>
#include <string>
#include <Eigen/Dense>
#include <geometry_msgs/Point.h>
#include <geometry_msgs/Pose.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/Quaternion.h>
#include <moveit/move_group_interface/move_group.h>
#include <moveit/robot_model/joint_model_group.h>
#include <moveit/robot_model/robot_model.h>
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_state/robot_state.h>
#include <moveit_msgs/CollisionObject.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <ros/console.h>
#include <ros/init.h>
#include <ros/node_handle.h>
#include <ros/param.h>
#include <ros/rate.h>
#include <rosconsole/macros_generated.h>
#include <sensor_msgs/JointState.h>
#include <shape_msgs/SolidPrimitive.h>
#include <std_msgs/Header.h>
```

Classes

- class IKinematics

  The IKinematics provides is an abstract class with pure virtual functions that are overriden by actual kinematic implementations.
- class DummyKinematics
- class RosKinematics
- class MoveitKinematics

Typedefs

- typedef boost::shared_ptr < IKinematics > IKinematicsSharedPtr
- typedef moveit::planning_interface::MoveItErrorCode RosErrorCode

11.16.1 Typedef Documentation

11.16.1.1 typedef boost::shared_ptr<IKinematics> IKinematicsSharedPtr

11.16.1.2 typedef moveit::planning_interface::MoveItErrorCode RosErrorCode

11.17 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Logging.h File Reference

Classes

- class ALogger

Macros

- #define LOGONCE static long nLog ## __LINE__ = 0; if (0 == nLog ## __LINE__++)

Variables

- ALogger Logger

11.17.1 Macro Definition Documentation

11.17.1.1 #define LOGONCE static long nLog ## __LINE__ = 0; if (0 == nLog ## __LINE__++)

11.17.2 Variable Documentation

11.17.2.1 ALogger Logger

11.18 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/MotionControl.h File Reference

```
#include <string>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <actionlib/client/simple_action_client.h>
#include "RCS.h"
```

Classes

- class MotionControl

  MotionControl is a class that contains some useful motion control methods.

11.19 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/moveit.h File Reference

```
#include <vector>
#include <boost/shared_ptr.hpp>
#include <ros/ros.h>
#include <moveit/move_group_interface/move_group.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <moveit_msgs/DisplayRobotState.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <moveit_msgs/AttachedCollisionObject.h>
#include <moveit_msgs/CollisionObject.h>
#include "urdf_model/eigenmath.h"
#include "RCS.h"
```

Classes

- class MoveitPlanning

11.20 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSMsgQueue.h File Reference

```
#include <boost/thread/mutex.hpp>
```

Classes

- class RCS::CMessageQueue< T >

  The CMessageQueue offers a mutexed front to a stl deque. The queue is a LIFO data structure. Useful for safely sharing data between multiple threads.

Namespaces

- RCS

11.21 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSThreadTemplate.h File Reference

```
#include <boost/thread.hpp>
#include "RCSTimer.h"
```

Classes

- class RCS::Thread

  Thread is an RCS ulapi equivalent for timed thread. Given a cycle time, the thread provides a wait function to sleep to exactly the amount of the thread cycle time. It keeps track of busy/idle time for diagnostic purposes.
  Notes: https://www.quantnet.com/threads/c-multithreading-in-boost.10028/.

Namespaces

- RCS

11.22 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/NIST/RCSTimer.h File Reference

```
#include <boost/chrono.hpp>
#include <boost/thread.hpp>
```

Classes

- class RCS::Timer

  Timer is a general-purpose timer. The Timer is a general-purpose timer, which can be used for waiting until a synchronous time tick, slept on for any period at all, or to obtain a time in system clock ticks from creation of the timer.

Namespaces

- RCS

Typedefs

- typedef int(∗ RCS::RCS_TIMERFUNC)(void ∗_arg)

Functions

- template<class Rep , class Period >
  double RCS::ToNanoseconds (boost::chrono::duration< Rep, Period > d)
- template<class Rep , class Period >
  double RCS::ToSeconds (boost::chrono::duration< Rep, Period > d)

11.23 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/nist_fanuc.h File Reference

```
#include <ros/ros.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <actionlib/client/simple_action_client.h>
```

11.24 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Primitive.h File Reference

```
#include <string>
#include <vector>
#include <ros/ros.h>
#include <eigen_conversions/eigen_msg.h>
#include <Eigen/src/Geometry/Transform.h>
#include <geometry_msgs/Point.h>
#include <geometry_msgs/Pose.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/Quaternion.h>
#include <moveit/move_group_interface/move_group.h>
#include <moveit/planning_scene/planning_scene.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <moveit/robot_model/joint_model_group.h>
#include <moveit/robot_model/robot_model.h>
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_state/robot_state.h>
#include <moveit_msgs/CollisionObject.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class CPrimitive

11.25 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RCS.h File Reference

```
#include <math.h>
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_model/robot_model.h>
#include "/usr/include/urdf_model/model.h"
#include "/usr/include/urdf_model/joint.h"
#include "/usr/include/urdf_model/link.h"
#include "/opt/ros/indigo/include/sensor_msgs/JointState.h"
#include "urdf_model/rosmath.h"
#include <stdarg.h>
#include "Globals.h"
```

Classes

- struct RCS::CanonCmd

  CanonCmd is the controller command structure.
- struct RCS::CanonWorldModel

  CanonWorldModel describes the controller state. Includes reference to robot model.

Namespaces

- RCS

Macros

- #define _USE_MATH_DEFINES
- #define LENGTHUNITS 1000
- #define EPSILON 1E-04
- #define DEFAULT_CYCLE 0.010
- #define DEFAULT_LOOP_CYCLE 0.10
- #define DEFAULT_CART_MAX_ACCEL 20.0/LENGTHUNITS
- #define DEFAULT_CART_MAX_VEL 200.0/LENGTHUNITS
- #define DEFAULT_JOINT_MAX_ACCEL 20.0/LENGTHUNITS
- #define DEFAULT_JOINT_MAX_VEL 150.0/LENGTHUNITS

Typedefs

- typedef boost::shared_ptr
  < urdf::ModelInterface > ModelInterfaceSharedPtr
- typedef
  sensor_msgs::JointState_
  < std::allocator< void > > JointState
- typedef boost::shared_ptr
  < JointState > JointStateSharedPtr
- typedef urdf::Pose RCS::Pose
- typedef urdf::Vector3 RCS::Position
- typedef urdf::Rotation RCS::Rotation

- typedef urdf::Vector3 RCS::Vector3
- typedef double RCS::Length
- typedef double RCS::LinearVelocity
- typedef double RCS::AngularVelocity
- typedef std::vector< double > RCS::robotAxes

Enumerations

- enum RCS::CanonLengthUnit { RCS::METER = 0, RCS::MM, RCS::INCH }

  enumeration of length units. Conversion into ROS compatible meters.

- enum RCS::TrajPointType { RCS::WAYPOINT = 1, RCS::GOAL }

  enumeration of trajector pose points.

- enum RCS::CanonAngleUnit { RCS::RADIAN = 0, RCS::DEGREE }

  enumeration of angle units. Conversion into ROS compatible radians.

- enum RCS::CanonForceUnit { RCS::NEWTON = 0, RCS::POUND, RCS::OUNCE }

  enumeration of force units.

- enum RCS::CanonTorqueUnit { RCS::NEWTONMETER = 0, RCS::FOOTPOUND }

  enumeration of torque units.

- enum RCS::CanonReturn {
  RCS::CANON_REJECT = -2, RCS::CANON_FAILURE = -1, RCS::CANON_SUCCESS = 0, RCS::CANON_STATUSREPLY = 1,
  RCS::CANON_RUNNING }

  enumeration of return type from Crcl intepretation. If statusreply, requires status sent to Crcl client.

- enum RCS::CanonCmdType {
  RCS::CANON_NOOP = 0, RCS::CANON_DWELL, RCS::CANON_END_CANON, RCS::CANON_INIT_CANON, RCS::CANON_MOVE_JOINT, RCS::CANON_MOVE_TO, RCS::CANON_MOVE_THRU, RCS::CANON_SET_MAX_CART_ACC,
  RCS::CANON_SET_MAX_CART_SPEED, RCS::CANON_SET_MAX_JOINT_ACC, RCS::CANON_SET_MAX_JOINT_SPEED, RCS::CANON_SET_GRIPPER,
  RCS::CANON_STOP_MOTION, RCS::CANON_UNKNOWN }

  enumeration of Crcl commands. Many Crcl commands are wm parameter setting and require no motion component.

- enum RCS::CanonStopMotionType { RCS::UNSET = -1, RCS::IMMEDIATE = 0, RCS::FAST, RCS::NORMAL }

  enumeration of stopping motion, e.g., estop equivalent to immediate.

- enum RCS::CanonAccProfile {
  RCS::MS_IS_UNSET = 0, RCS::MS_IS_DONE = 1, RCS::MS_IS_ACCEL = 2, RCS::MS_IS_CONST = 3,
  RCS::MS_IS_DECEL = 4, RCS::MS_IS_ESTOPPING = 5, RCS::MS_IS_PAUSED = 6 }

  enumeration of trajectory acceleration profile.

- enum RCS::MovementType { RCS::MOVE_DEFAULT = 0, RCS::MOVE_CARTESIAN, RCS::MOVE_JOINT }

  enumeration of trajectory motion type, joint or cartesian.

- enum RCS::CanonStatusType {
  RCS::CANON_DONE = 0, RCS::CANON_WORKING, RCS::CANON_PAUSED, RCS::CANON_ERROR,
  RCS::CANON_ABORT, RCS::CANON_WAITING }

  enumeration of controller status types for individual commands. Note, even though command types are listed, not all used or supported.

Functions

- std::string RCS::DumpPose (urdf::Pose &pose)

  DumpPose takes a urdf pose and generates a string describing pose. Can be used as std::cout << DumpPose(pose);.
- std::string RCS::DumpQuaterion (std::ostream &os, const urdf::Rotation &rot)

  DumpQuaterion takes a urdf quaterion and generates a string describing x,y,z,w coordinates. Can be used as std::cout << DumpQuaterion(urdf::rotation);.
- std::ostream & operator<< (std::ostream &os, const RCS::CanonCmd &cc)

Variables

- ModelInterfaceSharedPtr
  typedef boost::shared_ptr
  < urdf::Joint > JointSharedPtr

11.25.1 Macro Definition Documentation

11.25.1.1 #define _USE_MATH_DEFINES

11.25.1.2 #define DEFAULT_CART_MAX_ACCEL 20.0/LENGTHUNITS

11.25.1.3 #define DEFAULT_CART_MAX_VEL 200.0/LENGTHUNITS

11.25.1.4 #define DEFAULT_CYCLE 0.010

11.25.1.5 #define DEFAULT_JOINT_MAX_ACCEL 20.0/LENGTHUNITS

11.25.1.6 #define DEFAULT_JOINT_MAX_VEL 150.0/LENGTHUNITS

11.25.1.7 #define DEFAULT_LOOP_CYCLE 0.10

11.25.1.8 #define EPSILON 1E-04

11.25.1.9 #define LENGTHUNITS 1000

11.25.2 Typedef Documentation

11.25.2.1 typedef sensor_msgs::JointState_<std::allocator<void> > JointState

11.25.2.2 typedef boost::shared_ptr<JointState> JointStateSharedPtr

11.25.2.3 typedef boost::shared_ptr<urdf::ModelInterface> ModelInterfaceSharedPtr

11.25.3 Function Documentation

11.25.3.1 std::ostream& operator<< (std::ostream & *os,* const RCS::CanonCmd & *cc*) [inline]

11.25.4 Variable Documentation

11.25.4.1 ModelInterfaceSharedPtr typedef boost::shared_ptr<urdf::Joint> JointSharedPtr

11.26 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RCSInterpreter.h File Reference

```
#include "RCS.h"
#include <vector>
#include "Kinematics.h"
#include "Trajectory.h"
#include "trajectoryMaker.h"
#include "MotionControl.h"
```

Classes

- class RCSInterpreter

  RCSInterpreter parses a RCS command and generates robot motion commands.

11.27 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/RosConversions.h File Reference

Namespaces

- NIST

Functions

- void NIST::getRPY (const geometry_msgs::Quaternion &qmsg, double &roll, double &pitch, double &yaw)

11.28 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Setup.h File Reference

```
#include <ros/ros.h>
#include <map>
```

Functions

- bool SetupAppEnvironment ()

  SetupAppEnvironment will attempt save some of the application settings, e.g., user, nostname.
- bool SetupRosEnvironment ()

  SetupRosEnvironment will attempt to provide an environment equivalent to ROS "source devel/setup.bash". some of the code is hard coded, but unnecessary if executable run in shell.
- std::string ReadRosParams (ros::NodeHandle &nh)

  ReadRosParams will read and record all ros params in system.
- std::string ExecuteShellCommand (std::string)

  ExecuteShellCommand runs a shell script and returns results.

11.28.1 Function Documentation

11.28.1.1 std::string ExecuteShellCommand (std::string)

ExecuteShellCommand runs a shell script and returns results.

Parameters

string	is the shell command.

Returns

string containing pipe recording of output

11.28.1.2 std::string ReadRosParams (ros::NodeHandle & *nh*)

ReadRosParams will read and record all ros params in system.

Parameters

ROS	node handle.

Returns

string containing a list of all ros param definitions

11.28.1.3 bool SetupAppEnvironment ( )

SetupAppEnvironment will attempt save some of the application settings, e.g., user, nostname.

Returns

bool true if success.

11.28.1.4 bool SetupRosEnvironment ( )

SetupRosEnvironment will attempt to provide an environment equivalent to ROS "source devel/setup.bash". some of the code is hard coded, but unnecessary if executable run in shell.

Returns

bool true if success.

11.29 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/Trajectory.h File Reference

```
#include <vector>
#include <boost/shared_ptr.hpp>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <actionlib/client/simple_action_client.h>
#include <descartes_moveit/moveit_state_adapter.h>
#include <descartes_trajectory/axial_symmetric_pt.h>
#include <descartes_trajectory/cart_trajectory_pt.h>
#include <descartes_planner/dense_planner.h>
#include <descartes_planner/sparse_planner.h>
#include "RCS.h"
```

Classes

- class CTrajectory

Typedefs

- typedef std::vector < descartes_core::TrajectoryPtPtr > TrajectoryVec
- typedef TrajectoryVec::const_iterator TrajectoryIter
- typedef boost::shared_ptr < descartes_moveit::MoveitStateAdapter > MoveitStateAdapterPtr

11.29.1 Typedef Documentation

11.29.1.1 typedef boost::shared_ptr<descartes_moveit::MoveitStateAdapter> MoveitStateAdapterPtr

11.29.1.2 typedef TrajectoryVec::const_iterator TrajectoryIter

11.29.1.3 typedef std::vector<descartes_core::TrajectoryPtPtr> TrajectoryVec

11.30 /home/michalos/catkin_ws/src/nist_fanuc/include/nist_fanuc/trajectoryMaker.h File Reference

```
#include <stdio.h>
#include <vector>
#include <iostream>
#include <cmath>
#include <algorithm>
#include <boost/tuple/tuple.hpp>
#include "RCS.h"
#include "Globals.h"
```

Classes

- class IRate

  IRate is an interface class for defining the allowed motion rates.

- class TrajectoryMaker

  TrajectoryMaker generates simple trapezoidal velocities. Will accept non-zero final velocity.

11.31 /home/michalos/catkin_ws/src/nist_fanuc/src/Archive/RCSInterpreter.cpp File Reference

```
#include "CrclConfig.h"
#include "RCSInterpreter.h"
#include "Controller.h"
#include "Globals.h"
#include <algorithm>
```

11.32 /home/michalos/catkin_ws/src/nist_fanuc/src/RCSInterpreter.cpp File Reference

```
#include "CrclConfig.h"
#include "RCSInterpreter.h"
#include "Controller.h"
#include "Globals.h"
#include <algorithm>
#include "urdf_model/eigenmath.h"
#include "Conversions.h"
#include "trajectoryMaker.h"
```

11.33 /home/michalos/catkin_ws/src/nist_fanuc/src/AsioCrclServer.cpp File Reference

```
#include "CrclConfig.h"
#include "AsioCrclServer.h"
#include <boost/exception/all.hpp>
#include <boost/regex.hpp>
#include "Globals.h"
#include "Controller.h"
```

Macros

- #define S_OK 0

Variables

- boost::asio::io_service myios

11.33.1 Macro Definition Documentation

11.33.1.1 #define S_OK 0

11.33.2 Variable Documentation

11.33.2.1 boost::asio::io_service myios

11.34 /home/michalos/catkin_ws/src/nist_fanuc/src/ChainRobotModel.cpp File Reference

```
#include "ChainRobotModel.h"
#include "urdf_model/RobotModel.h"
#include "urdf_model/urdf_parse_model.cpp"
#include "Globals.h"
#include "urdf_model/eigenmath.h"
```

Namespaces

- RCS

Macros

- #define STANDALONEURDF

11.34.1 Macro Definition Documentation

11.34.1.1 #define STANDALONEURDF

11.35 /home/michalos/catkin_ws/src/nist_fanuc/src/Communication.cpp File Reference

```
#include "Communication.h"
```

11.36 /home/michalos/catkin_ws/src/nist_fanuc/src/Controller.cpp File Reference

```
#include "Controller.h"
#include "urdf_model/rosmath.h"
#include <boost/exception/all.hpp>
#include <boost/asio.hpp>
#include <boost/thread.hpp>
#include <strstream>
#include <iostream>
#include "urdf_model/eigenmath.h"
#include "CrclInterface.h"
```

Namespaces

- RCS

Macros

- #define BOOST_ALL_NO_LIB
- #define S_OK 0

Functions

- void DebugBreak ()
- ::CRCLProgramType::MiddleCommand_sequence & RCS::DummyInit ()

Variables

- ALogger Logger
- boost::mutex RCS::cncmutex
- RCS::CController RCS::Controller (DEFAULT_LOOP_CYCLE)

11.36.1 Macro Definition Documentation

11.36.1.1 #define BOOST_ALL_NO_LIB

11.36.1.2 #define S_OK 0

11.36.2 Function Documentation

11.36.2.1 void DebugBreak ( )

global definition of windows DebugBreak equivalent.

11.36.3 Variable Documentation

11.36.3.1 ALogger Logger

11.37 /home/michalos/catkin_ws/src/nist_fanuc/src/crcl.cpp File Reference

```
#include "crcl.h"
#include "Globals.h"
#include "CrclInterface.h"
#include "boost/array.hpp"
#include "urdf_model/rosmath.h"
#include <Eigen/Dense>
#include "urdf_model/eigenmath.h"
```

Namespaces

- Crcl

Functions

- RosMatrix Crcl::GetXZRotMatrix (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- Eigen::Matrix3d Crcl::GetEigenRotMatrix (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- bool Crcl::GetRPY (Crcl::PoseType pose, double &roll, double &pitch, double &yaw)
- bool Crcl::GetRPY (urdf::Vector3 Xrot, urdf::Vector3 Zrot, double &roll, double &pitch, double &yaw)
- urdf::Pose Crcl::Convert (Crcl::PoseType &pose, double lengthConversion)
- Crcl::VectorType Crcl::VectorZero ()
- urdf::Vector3 Crcl::GetVector3D (Crcl::PointType &point)
- urdf::Vector3 Crcl::GetVector3D (Crcl::VectorType &vector)
- bool Crcl::GetPoseToRPY (Crcl::PoseType &pose, double &dRoll, double &dPitch, double &dYaw)
- urdf::Rotation Crcl::Convert (urdf::Vector3 Xrot, urdf::Vector3 Zrot)
- Crcl::PoseType Crcl::Init (std::vector< double > terms)
- Crcl::PoseType Crcl::Convert (urdf::Pose pose)
- sensor_msgs::JointState Crcl::Convert (Crcl::JointStatusSequence jout, double angleConversion)
- Crcl::JointStatusSequence Crcl::Convert (JointState joints, double _angleConversion)
- JointStatusSequence Crcl::Convert (Crcl::ActuatorJointSequence joints, double _angleConversion)
- ::PointType Crcl::GetPoint (RCS::Vector3 &point)
- ::VectorType Crcl::GetVector (RCS::Vector3 &point)
- Crcl::PoseType Crcl::NullPose ()
- Crcl::PoseType Crcl::IdentityPose ()
- Crcl::PoseType Crcl::PoseHome ()
- std::vector< double > Crcl::ConvertToAnglePositionVector (Crcl::ActuatorJointSequence &joints, double dAngleConversion)
- std::string Crcl::DumpCrclPose (Crcl::PoseType pose, std::string separator)
- std::string Crcl::DumpPose (Crcl::PoseType pose, std::string separator)

11.38 /home/michalos/catkin_ws/src/nist_fanuc/src/CrclInterface.cpp File Reference

```
#include "CrclConfig.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <math.h>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/framework/XMLGrammarPoolImpl.hpp>
#include <boost/regex.hpp>
#include <boost/exception/all.hpp>
#include "CrclInterface.h"
#include "Globals.h"
#include "Controller.h"
```

Macros

- #define _USE_MATH_DEFINES

11.38.1 Macro Definition Documentation

11.38.1.1 #define _USE_MATH_DEFINES

11.39 /home/michalos/catkin_ws/src/nist_fanuc/src/demo.cpp File Reference

```
#include <ros/ros.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <actionlib/client/simple_action_client.h>
#include <descartes_moveit/moveit_state_adapter.h>
#include <descartes_trajectory/axial_symmetric_pt.h>
#include <descartes_trajectory/cart_trajectory_pt.h>
#include <descartes_planner/dense_planner.h>
#include <descartes_planner/sparse_planner.h>
#include <descartes_trajectory/cartesian_interpolator.h>
#include <std_msgs/String.h>
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <unistd.h>
#include <fcntl.h>
#include <iostream>
#include <sstream>
```

Typedefs

- typedef std::vector
  < descartes_core::TrajectoryPtPtr > TrajectoryVec

- typedef
  TrajectoryVec::const_iterator TrajectoryIter

Functions

- trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec &trajectory, const descartes_core::-
  RobotModel &model, const std::vector< std::string > &joint_names, const std::vector< double > &time_delay)

- bool executeTrajectory (const trajectory_msgs::JointTrajectory &trajectory, const std::string &trajectory_ns)

- int main (int argc, char **argv)

11.39.1 Typedef Documentation

11.39.1.1 typedef TrajectoryVec::const_iterator TrajectoryIter

11.39.1.2 typedef std::vector<descartes_core::TrajectoryPtPtr> TrajectoryVec

11.39.2 Function Documentation

11.39.2.1 bool executeTrajectory (const trajectory_msgs::JointTrajectory & *trajectory,* const std::string & *trajectory_ns*)

11.39.2.2 int main (int *argc,* char ** *argv*)

11.39.2.3 trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec & *trajectory,* const descartes_core::RobotModel & *model,* const std::vector< std::string > & *joint_names,* const std::vector< double > & *time_delay*)

11.40 /home/michalos/catkin_ws/src/nist_fanuc/src/fanucdemo.cpp File Reference

```
#include <boost/format.hpp>
#include "fanucdemo.h"
#include <string>
#include "sys/stat.h"
#include "fcntl.h"
#include <iostream>
#include <sstream>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/framework/XMLGrammarPoolImpl.hpp>
#include "MotionControl.h"
#include "AsioCrclServer.h"
#include "Globals.h"
#include "Controller.h"
#include "CrclInterface.h"
#include "Kinematics.h"
#include "urdf_model/eigenmath.h"
#include "Communication.h"
#include "Setup.h"
#include "moveit.h"
```

Functions

- std::string DumpUrdfPose (const urdf::Pose &p)
- int main (int argc, char **argv)

11.40.1 Function Documentation

11.40.1.1 std::string DumpUrdfPose (const urdf::Pose & *p*) [inline]

11.40.1.2 int main (int *argc,* char ** *argv*)

11.41 /home/michalos/catkin_ws/src/nist_fanuc/src/Globals.cpp File Reference

```
#include "Globals.h"
#include <map>
#include <iostream>
```

Variables

- CGlobals Globals

11.41.1 Variable Documentation

11.41.1.1 CGlobals Globals

global definition of globals

11.42 /home/michalos/catkin_ws/src/nist_fanuc/src/Kinematics.cpp File Reference

```
#include "Kinematics.h"
#include "urdf_model/eigenmath.h"
#include <eigen_conversions/eigen_msg.h>
#include "RosConversions.h"
#include <iostream>
#include "Conversions.h"
#include "Globals.h"
```

11.43 /home/michalos/catkin_ws/src/nist_fanuc/src/MotionControl.cpp File Reference

```
#include "MotionControl.h"
#include "urdf_model/rosmath.h"
#include <algorithm>
#include "Conversions.h"
```

11.44 /home/michalos/catkin_ws/src/nist_fanuc/src/moveit.cpp File Reference

```
#include "moveit.h"
#include "Globals.h"
#include "Conversions.h"
```

11.45 /home/michalos/catkin_ws/src/nist_fanuc/src/nist_fanuc.cpp File Reference

```
#include <boost/format.hpp>
#include "nist_fanuc.h"
#include <string>
#include "sys/stat.h"
#include "fcntl.h"
#include <iostream>
#include <sstream>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/framework/XMLGrammarPoolImpl.hpp>
#include "MotionControl.h"
#include "AsioCrclServer.h"
#include "Globals.h"
#include "Controller.h"
#include "CrclInterface.h"
#include "Kinematics.h"
#include "urdf_model/eigenmath.h"
#include "Communication.h"
#include "Setup.h"
#include <ros/package.h>
#include "moveit.h"
```

Macros

- #define INITJOINTCONTROLLER
- #define ROBOTSTATUS
- #define BOOSTASIO

Functions

- std::string DumpUrdfPose (const urdf::Pose &p)
- int main (int argc, char **argv)

11.45.1 Macro Definition Documentation

11.45.1.1 #define BOOSTASIO

11.45.1.2 #define INITJOINTCONTROLLER

11.45.1.3 #define ROBOTSTATUS

11.45.2 Function Documentation

11.45.2.1 std::string DumpUrdfPose (const urdf::Pose & *p*)

11.45.2.2 int main (int *argc*, char ** *argv*)

11.46 /home/michalos/catkin_ws/src/nist_fanuc/src/Primitive.cpp File Reference

```
#include "Primitive.h"
```

11.47 /home/michalos/catkin_ws/src/nist_fanuc/src/RCS.cpp File Reference

```
#include "CrclConfig.h"
#include "RCS.h"
#include "Globals.h"
#include "Controller.h"
```

Namespaces

- RCS

11.48 /home/michalos/catkin_ws/src/nist_fanuc/src/RobotModelUrdf.cpp File Reference

Macros

- #define URDFDOM_STATIC

11.48.1 Macro Definition Documentation

11.48.1.1 #define URDFDOM_STATIC

11.49 /home/michalos/catkin_ws/src/nist_fanuc/src/SanityCheckTests.cpp File Reference

Functions

- RCS::Controller Kinematics () -> SetJointValues(cjoints.position)
- std::cout<< "Dump All zero FK
  test Pose "<< RCS::DumpPose(testpose).c_str();_quatToRpy(testpose.rotation,
  roll, pitch, yaw);std::cout
  << "Dump rotation in RPY"
  << roll<< ":"<< pitch
  << ":"<< yaw<< std::endl;testjoints2=RCS::Controller.Kinematics() -> IK (testpose, testjoints)
- Globals ReadFile ("/home/michalos/catkin_ws/src/fanucdemo/doc/programExample.xml", contents)
- mycrcl DelegateCRCLCmd (contents)

Variables

- std::vector< double > testjoints = RCS::Controller.Kinematics()->GetJointValues()
- std::vector< double > testjoints2
- double roll

- double pitch
- double yaw
- urdf::Pose testpose = RCS::Controller.Kinematics()->FK(testjoints)
- cjoints = jointReader->GetCurrentReadings()
- Crcl::CrclDelegateInterface mycrcl

  Program test.
- std::string contents

11.49.1 Function Documentation

11.49.1.1 mycrcl DelegateCRCLCmd (contents)

11.49.1.2 std::cout<< "Dump All zero FK test Pose " << RCS::DumpPose(testpose).c_str(); _quatToRpy (testpose.rotation, roll, pitch, yaw); std::cout << "Dump rotation in RPY" << roll << ":" << pitch << ":"<< yaw << std::endl; testjoints2 = RCS::Controller.Kinematics()-> IK (testpose , testjoints)

11.49.1.3 RCS::Controller Kinematics ( ) -> SetJointValues(cjoints.position)

11.49.1.4 Globals ReadFile ("/home/michalos/catkin_ws/src/fanucdemo/doc/programExample.xml" , contents)

11.49.2 Variable Documentation

11.49.2.1 cjoints = jointReader->GetCurrentReadings()

11.49.2.2 std::string contents

11.49.2.3 Crcl::CrclDelegateInterface mycrcl

Program test.

11.49.2.4 double pitch

11.49.2.5 double roll

11.49.2.6 testjoints = RCS::Controller.Kinematics()->GetJointValues()

11.49.2.7 std::vector<double> testjoints2

11.49.2.8 testpose = RCS::Controller.Kinematics()->FK(testjoints)

11.49.2.9 double yaw

11.50 /home/michalos/catkin_ws/src/nist_fanuc/src/Setup.cpp File Reference

```
#include "Setup.h"
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <vector>
#include <sstream>
#include <boost/algorithm/string.hpp>
#include <ros/ros.h>
#include "Globals.h"
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_model/robot_model.h>
```

Functions

- std::string ExecuteShellCommand (std::string command)

  ExecuteShellCommand runs a shell script and returns results.
- bool SetupAppEnvironment ()

  SetupAppEnvironment will attempt save some of the application settings, e.g., user, nostname.
- bool SetupRosEnvironment ()

  SetupRosEnvironment will attempt to provide an environment equivalent to ROS "source devel/setup.bash". some of the code is hard coded, but unnecessary if executable run in shell.
- std::string ReadRosParam (ros::NodeHandle &nh, std::string paramkey)
- std::string ReadRosParams (ros::NodeHandle &nh)

  ReadRosParams will read and record all ros params in system.
- void GetJointLimits (std::vector< std::string > names, std::vector< double > lower, std::vector< double > upper)

11.50.1 Function Documentation

11.50.1.1 std::string ExecuteShellCommand (std::string)

ExecuteShellCommand runs a shell script and returns results.

Parameters

string	is the shell command.

Returns

string containing pipe recording of output

11.50.1.2 void GetJointLimits (std::vector< std::string > *names,* std::vector< double > *lower,* std::vector< double > *upper*)

11.50.1.3 std::string ReadRosParam (ros::NodeHandle & *nh,* std::string *paramkey*)

11.50.1.4 std::string ReadRosParams (ros::NodeHandle & *nh*)

ReadRosParams will read and record all ros params in system.

Parameters

ROS	node handle.

Returns

string containing a list of all ros param definitions

11.50.1.5 bool SetupAppEnvironment ( )

SetupAppEnvironment will attempt save some of the application settings, e.g., user, nostname.

Returns

bool true if success.

11.50.1.6 bool SetupRosEnvironment ( )

SetupRosEnvironment will attempt to provide an environment equivalent to ROS "source devel/setup.bash". some of the code is hard coded, but unnecessary if executable run in shell.

Returns

bool true if success.

11.51 /home/michalos/catkin_ws/src/nist_fanuc/src/TestDescartes.cpp File Reference

```
#include "Trajectory.h"
#include <descartes_moveit/moveit_state_adapter.h>
#include <descartes_trajectory/axial_symmetric_pt.h>
#include <descartes_trajectory/cart_trajectory_pt.h>
#include <descartes_planner/dense_planner.h>
```

Typedefs

- typedef std::vector < descartes_core::TrajectoryPtPtr > TrajectoryVec
- typedef TrajectoryVec::const_iterator TrajectoryIter

Functions

- descartes_core::TrajectoryPtPtr makeCartesianPoint (const Eigen::Affine3d &pose)
- descartes_core::TrajectoryPtPtr makeToleroncedCartesianPoint (const Eigen::Affine3d &pose)
- trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec &trajectory, const descartes_core::RobotModel &model, const std::vector< std::string > &joint_names, double time_delay)
- bool executeTrajectory (const trajectory_msgs::JointTrajectory &trajectory)
- void TestKinematics (ros::NodeHandle &nh)

11.51.1 Typedef Documentation

11.51.1.1 typedef TrajectoryVec::const_iterator TrajectoryIter

11.51.1.2 typedef std::vector<descartes_core::TrajectoryPtPtr> TrajectoryVec

11.51.2 Function Documentation

11.51.2.1 bool executeTrajectory (const trajectory_msgs::JointTrajectory & *trajectory*)

Sends a ROS trajectory to the robot controller

11.51.2.2 descartes_core::TrajectoryPtPtr makeCartesianPoint (const Eigen::Affine3d & *pose*)

Generates an completely defined (zero-tolerance) cartesian point from a pose

11.51.2.3 descartes_core::TrajectoryPtPtr makeToleroncedCartesianPoint (const Eigen::Affine3d & *pose*)

Generates a cartesian point with free rotation about the Z axis of the EFF frame

11.51.2.4 void TestKinematics (ros::NodeHandle & *nh*)

11.51.2.5 trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec & *trajectory,* const descartes_core::RobotModel & *model,* const std::vector< std::string > & *joint_names,* double *time_delay*)

Translates a descartes trajectory to a ROS joint trajectory

11.52 /home/michalos/catkin_ws/src/nist_fanuc/src/TestMoveit.cpp File Reference

```
#include "moveit.h"
#include "Globals.h"
#include "urdf_model/eigenmath.h"
```

Functions

- trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec &trajectory, const descartes_core::RobotModel &model, const std::vector< std::string > &joint_names, double time_delay)
- bool executeTrajectory (const trajectory_msgs::JointTrajectory &trajectory)
- void TestMoveit (ros::NodeHandle &nh)

11.52.1 Function Documentation

11.52.1.1 bool executeTrajectory (const trajectory_msgs::JointTrajectory & *trajectory*)

11.52.1.2 void TestMoveit (ros::NodeHandle & *nh*)

11.52.1.3 trajectory_msgs::JointTrajectory toROSJointTrajectory (const TrajectoryVec & *trajectory,* const descartes_core::RobotModel & *model,* const std::vector< std::string > & *joint_names,* double *time_delay*)

11.53 /home/michalos/catkin_ws/src/nist_fanuc/src/Trajectory.cpp File Reference

```
#include "Trajectory.h"
#include "RCS.h"
```

11.54 /home/michalos/catkin_ws/src/nist_fanuc/src/trajectoryMaker.cpp File Reference

```
#include "trajectoryMaker.h"
#include <numeric>
#include <functional>
#include "Globals.h"
#include <algorithm>
```

11.55 /home/michalos/catkin_ws/src/nist_fanuc/src/urdf_parse_model.cpp File Reference