**Frequently Asked Questions (FAQs)**

**ROS**

A list of the most frequently asked questions and their answers is contained in this document to assist you in the development or modification of ROS robotic control systems.

Table of Contents

[Question: How is Eigen Affine3d modeled? 2](#_Toc461628292)

[Question: How is ROS Time modeled? 5](#_Toc461628293)

[Time and Duration 5](#_Toc461628294)

[Getting the Current Time 5](#_Toc461628295)

[Creating Time and Duration Instances 6](#_Toc461628296)

[Converting Time and Duration Instances 6](#_Toc461628297)

[Time and Duration Arithmetic 6](#_Toc461628298)

[Sleeping and Rates 7](#_Toc461628299)

[Wall Time 7](#_Toc461628300)

[Question: What are the YAML files? 8](#_Toc461628301)

[Question: How about a ROS VERSION NUMBER SCRIPT? 8](#_Toc461628302)

[Question: How to detect if ros master is running in C++? 8](#_Toc461628303)

[Question: Bash script to show Sorted Package Version Info in simple HTML Table Format? 8](#_Toc461628304)

[Question: How does the frame\_id work? 9](#_Toc461628305)

[Question: WARNING: Your current environment's CMAKE\_PREFIX\_PATH is different from the cached CMAKE\_PREFIX\_PATH used the last time this workspace was built? 9](#_Toc461628306)

[Question: Can’t see the robot in Rviz? 9](#_Toc461628307)

[Question: How can I see point values in Rviz? 10](#_Toc461628308)

[Question: How can I see point values in Rviz in C++? 10](#_Toc461628309)

[Question: Capture rviz into animated gif? 11](#_Toc461628310)

# Question: How is Eigen Affine3d modeled?

**Answer:**

An Eigen::Affine3d is a 4x4 homogeneous transformation matrix

Eigen::Affine3d pose;

pose = Eigen::Translation3d(0, 0, 1);

Eigen::Matrix3d m;

m = AngleAxisd(angle1, Vector3f::UnitZ())

\* AngleAxisd(angle2, Vector3f::UnitY())

\* AngleAxisd(angle3, Vector3f::UnitZ());

pose.linear() = m;

#include <Eigen/Geometry>

Eigen::Affine3d create\_rotation\_matrix(double ax, double ay, double az) {

Eigen::Affine3d rx =

Eigen::Affine3d(Eigen::AngleAxisd(ax, Eigen::Vector3d(1, 0, 0)));

Eigen::Affine3d ry =

Eigen::Affine3d(Eigen::AngleAxisd(ay, Eigen::Vector3d(0, 1, 0)));

Eigen::Affine3d rz =

Eigen::Affine3d(Eigen::AngleAxisd(az, Eigen::Vector3d(0, 0, 1)));

return rz \* ry \* rx;

}

int main() {

Eigen::Affine3d r = create\_rotation\_matrix(1.0, 1.0, 1.0);

Eigen::Affine3d t(Eigen::Translation3d(Eigen::Vector3d(1,1,2)));

Eigen::Matrix4d m = (t \* r).matrix(); // Option 1

Eigen::Matrix4d m = t.matrix(); // Option 2

m \*= r.matrix();

return 0;

}

Another method is to do the following:

Eigen::Matrix3d R;

// Find your Rotation Matrix

Eigen::Vector3d T;

// Find your translation Vector

Eigen::Matrix4d Trans; // Your Transformation Matrix

Trans.setIdentity(); // Set to Identity to make bottom row of Matrix 0,0,0,1

Trans.block<3,3>(0,0) = R;

Trans.rightCols<1>() = T;

Explanation below from <http://pointclouds.org/documentation/tutorials/matrix_transform.php>

***/\* Reminder: how transformation matrices work :***

***|-------> This column is the translation***

***| 1 0 0 x | \***

***| 0 1 0 y | }-> The identity 3x3 matrix (no rotation) on the left***

***| 0 0 1 z | /***

***| 0 0 0 1 | -> We do not use this line (and it has to stay 0,0,0,1)***

***METHOD #1: Using a Matrix4f***

***This is the "manual" method, perfect to understand but error prone !***

***\*/***

**Eigen::Matrix4f transform\_1 = Eigen::Matrix4f::Identity();**

***// Define a rotation matrix (see https://en.wikipedia.org/wiki/Rotation\_matrix)***

**float theta = M\_PI/4; *// The angle of rotation in radians***

***// Define a translation of 2.5 meters on the x axis.***

***/\* METHOD #2: Using a Affine3f***

***This method is easier and less error prone***

***\*/***

**Eigen::Affine3f transform\_2 = Eigen::Affine3f::Identity();**

***// Define a translation of 2.5 meters on the x axis.***

**transform\_2.translation() << 2.5, 0.0, 0.0;**

***// The same rotation matrix as before; theta radians around Z axis***

**transform\_2.rotate (Eigen::AngleAxisf (theta, Eigen::Vector3f::UnitZ()));**

***// Print the transformation***

**printf ("\nMethod #2: using an Affine3f\n");**

**std::cout << transform\_2.matrix() << std::endl;**

**int main()**

**{**

**Eigen::ArrayXf v(6);**

**v << 1, 2, 3, 4, 5, 6;**

**cout << "v.head(3) =" << endl << v.head(3) << endl << endl;**

**cout << "v.tail<3>() = " << endl << v.tail<3>() << endl << endl;**

**v.segment(1,4) \*= 2;**

**cout << "after 'v.segment(1,4) \*= 2', v =" << endl << v << endl;**

**}**

**Spherical interpolation   
(**[Rotation2D](https://eigen.tuxfamily.org/dox/classEigen_1_1Rotation2D.html)**and**[Quaternion](https://eigen.tuxfamily.org/dox/classEigen_1_1Quaternion.html)**only)**

**rot3 = rot1.slerp**(alpha,rot2);

**Component** **accessors**

|  |  |
| --- | --- |
| full read-write access to the internal matrix | t.matrix() = matN1xN1; // N1 means N+1  matN1xN1 = t.matrix(); |
| coefficient accessors | t(i,j) = scalar; <=> t.matrix()(i,j) = scalar;  scalar = t(i,j); <=> scalar = t.matrix()(i,j); |
| translation part | t.translation() = vecN;  vecN = t.translation(); |
| linear part | t.linear() = matNxN;  matNxN = t.linear(); |
| extract the rotation matrix | matNxN = t.rotation(); |

|  |  |
| --- | --- |
|  | **procedural API** |
| [**Translation**](https://eigen.tuxfamily.org/dox/classEigen_1_1Translation.html) | t.translate(Vector\_(tx,ty,..));  t.pretranslate(Vector\_(tx,ty,..)); |

# Question: How is ROS Time modeled?

**Answer:**

TIME: from <http://library.isr.ist.utl.pt/docs/roswiki/roscpp%282f%29Overview%282f%29Time.html>

## Time and Duration

ROS has builtin time and duration primitive types, which [roslib](http://library.isr.ist.utl.pt/docs/roswiki/roslib.html) provides as the ros::Time and ros::Duration classes, respectively. A Time is a specific moment (e.g. "today at 5pm") whereas a Duration is a period of time (e.g. "5 hours"). Durations can be negative.

Times and durations have identical representations:

int32 sec

int32 nsec

ROS has the ability to setup a simulated [Clock](http://library.isr.ist.utl.pt/docs/roswiki/Clock.html) for nodes. Instead of using platform time routines, you should use roscpp's time routines for accessing the current time, which will work seamlessly with simulated [Clock](http://library.isr.ist.utl.pt/docs/roswiki/Clock.html) time as well as wall-clock time.

### Getting the Current Time

ros::Time::now()

* Get the current time as a ros::Time instance:

ros::Time begin = ros::Time::now();

#### Time zero

When using simulated [Clock](http://library.isr.ist.utl.pt/docs/roswiki/Clock.html) time, now() returns time 0 until first message has been received on /clock, so 0 means essentially that the client does not know clock time yet. A value of 0 should therefore be treated differently, such as looping over now() until non-zero is returned.

### Creating Time and Duration Instances

You can create a Time or Duration to a specific value as well, either floating-point seconds:

1 ros::Time a\_little\_after\_the\_beginning(0.001);

2 ros::Duration five\_seconds(5.0);

or through the two-integer constructor:

1 ros::Time a\_little\_after\_the\_beginning(0, 10000);

2 ros::Duration five\_seconds(5, 0);

### Converting Time and Duration Instances

Time and Duration objects can also be turned into floating point seconds:

1 double secs =ros::Time::now().toSec();

2

3 ros::Duration d(0.5);

4 secs = d.toSec();

### Time and Duration Arithmetic

Like other primitive types, you can perform arithmetic operations on Times and Durations. People are often initially confused on what arithmetic with these instances is like, so it's good to run through some examples:

* 1 hour + 1 hour = 2 hours (duration + duration = duration)

2 hours - 1 hour = 1 hour (duration - duration = duration)

Today + 1 day = tomorrow (time + duration = time)

Today - tomorrow = -1 day (time - time = duration)

Today + tomorrow = error (time + time is undefined)

Arithmetic with Time and Duration instances is similar to the above examples:

1 ros::Duration two\_hours = ros::Duration(60\*60) + ros::Duration(60\*60);

2 ros::Duration one\_hour = ros::Duration(2\*60\*60) - ros::Duration(60\*60);

3 ros::Time tomorrow = ros::Time::now() + ros::Duration(24\*60\*60);

4 ros::Duration negative\_one\_day = ros::Time::now() - tomorrow;

## Sleeping and Rates

bool ros::Duration::sleep()

* Sleep for the amount of time specified by the duration:

1 ros::Duration(0.5).sleep(); // sleep for half a second

ros::Rate

* roslib provides a ros::Rate convenience class which makes a best effort at maintaining a particular rate for a loop. For example:

ros::Rate r(10); // 10 hz

while (ros::ok())

{

... do some work ...

r.sleep();

}

In the above example, the Rate instance will attempt to keep the loop at 10hz by accounting for the time used by the work done during the loop.

**Note:** It is generally recommended to use Timers instead of Rate. See the [Timers Tutorial](http://library.isr.ist.utl.pt/docs/roswiki/roscpp_tutorials%282f%29Tutorials%282f%29Timers.html) for details.

## Wall Time

For cases where you want access to the actual wall-clock time even if running inside simulation, roslib provides Wall versions of all its time constructs, i.e. ros::WallTime, ros::WallDuration, and ros::WallRate which have identical interfaces to ros::Time, ros::Duration, and ros::Rate respectively.

# Question: What are the YAML files?

**Answer:**

YAML FILES:

michalos@rufous:gripper\_ws> find . -type f -name "\*.yaml"

./.catkin\_tools/default/build.yaml

./build/.catkin\_tools.yaml

./src/nistgripper/fanuc\_lrmate200id\_support/config/joint\_names\_lrmate200id.yaml

./src/nistgripper/fanuc\_lrmate200id\_moveit\_config/config/fake\_controllers.yaml

./src/nistgripper/fanuc\_lrmate200id\_moveit\_config/config/kinematics.yaml

./src/nistgripper/fanuc\_lrmate200id\_moveit\_config/config/joint\_limits.yaml

./src/nistgripper/fanuc\_lrmate200id\_moveit\_config/config/controllers.yaml

./src/nistgripper/fanuc\_lrmate200id\_moveit\_config/config/ompl\_planning.yaml

# Question: How about a ROS VERSION NUMBER SCRIPT?

**Answer:**

!/bin/bash

pkgs=`rospack list | awk '{print $1}'`

for pkg in $pkgs

do

echo -n $pkg " " && rosversion $pkg

done

# Question: How to detect if ros master is running in C++?

**Answer:**

In C++ you can check for the master being online, returns true if up, false if not.

#include <ros/master.h>

...

ros::init(argc, argv, "nistcrcl");

# Wait til roscore up and running

while(!ros::master::check()) Globals.Sleep(10000);

# Question: Bash script to show Sorted Package Version Info in simple HTML Table Format?

**Answer:**

!/bin/bash

pkgs=`rospack depends $1 |sort `

echo "<HTML><BODY>"

echo "<h1> " $1 " Package Version Dependencies </h1>"

echo "<TABLE>"

echo "<TR><TH>Package</TH><TH>Version</TH></TR>"

for pkg in $pkgs

do

ver=`rosversion $pkg `

echo "<TR><TD> $pkg </TD><TD> $ver </TD></TR>"

done

echo "</TABLE></BODY></HTML>"

# Question: How does the frame\_id work?

**Answer:**

From http://answers.ros.org/question/34684/header-frame\_id/

The frame\_id in a message specifies the point of reference for data contained in that message.

For example, if I want to specify a goal that is 10 meters directly in front of my robot, I would first declare the "origin" of the robot, which I will name the /base\_link frame. The origin of this /base\_link frame (0,0,0) is located at the center of mass of my robot. I would send a goal of (10,0,0) in the /base\_link frame.

# Question: WARNING: Your current environment's CMAKE\_PREFIX\_PATH is different from the cached CMAKE\_PREFIX\_PATH used the last time this workspace was built?

**Answer:**

If you want to use a different CMAKE\_PREFIX\_PATH you should call`catkin clean` to remove all references to the previous CMAKE\_PREFIX\_PATH.

$ catkin clean

Y

$ catkin build -DCMAKE\_BUILD\_TYPE=Debug

# Question: Can’t see the robot in Rviz?

**Answer:** The Robot Model must be enabled in rviz if you want to see it move when you move joint\_state

Open RVIZ, [ADD] Robot Model (not Robot)

# Question: How can I see point values in Rviz?

**Answer:** Clicked point - in rviz Panels-> Tool Properties

Publish Point shows up in dialog box

rostopic echo /clicked\_point

header:

seq: 0

stamp:

secs: 1472590453

nsecs: 790104667

frame\_id: base\_link

point:

x: 0.374454498291

y: -0.108664080501

z: 0.00685179233551

---

1. First, click Publish Point Red Balloon Arrow
2. Then clicked on center of one of lower X neg Y bolt holder slots
3. (facing robot - lower left holder) This is kinda the centroid

# Question: How can I see point values in Rviz in C++?

**Answer:**

Add subscriber to poin

#include <ros/ros.h>

#include <geometry\_msgs/PointStamped.h>t\_position

#include <ros/ros.h>

#include <geometry\_msgs/PointStamped.h>

...

sub = \_nh.subscribe("clicked\_point", 10, &RvizCheckers::callback, this);

...

void callback(const geometry\_msgs::PointStamped::ConstPtr& msg) {

geometry\_msgs::Point pt = msg->point;

std::cout << Globals.StrFormat("geometry\_msgs::PointStamped=%f:%f:%f\n", pt.x,pt.y,pt.z );

}

# Question: Capture rviz into animated gif?

**Answer:** Have to record screen and then convert to animated gif. I used gtk-recordmydesktop to record desktop window, and then <http://www.online-convert.com/file-format/ogv> to convert ogv file format into animated gif. A bit kludgy. Imagemagick did not have proper codec?

sudo apt-get install recordmydesktop

sudo apt-get install gtk-recordmydesktop

sudo apt-get install imagemagick mplayer gtk-recordmydesktop

http://askubuntu.com/questions/107726/how-to-create-animated-gif-images-of-a-screencast

sudo apt-get install byzanz

http://www.online-convert.com/file-format/ogv