# Instructions for Using Matlab Controller for UMI RTX Robots

Donald Nairn, Bob Fisher

January 28, 2004

## 1 Starting Up

To start using the Matlab controller firstly you must have a copy of the Matlab executable files from `/home/rbf/D11/RTX`:

- abs_angle_rtx.mexglx
- abs_position_rtx.mexglx
- calibrate_rtx.mexglx
- close_grippers_rtx.mexglx
- location_xy_rtx.mexglx
- location_an_rtx.mexglx
- open_grippers_rtx.mexglx
- rel_position_rtx.mexglx
- rel_angle_rtx.mexglx
- rtx_exit.mexglx
- rtx_home.mexglx

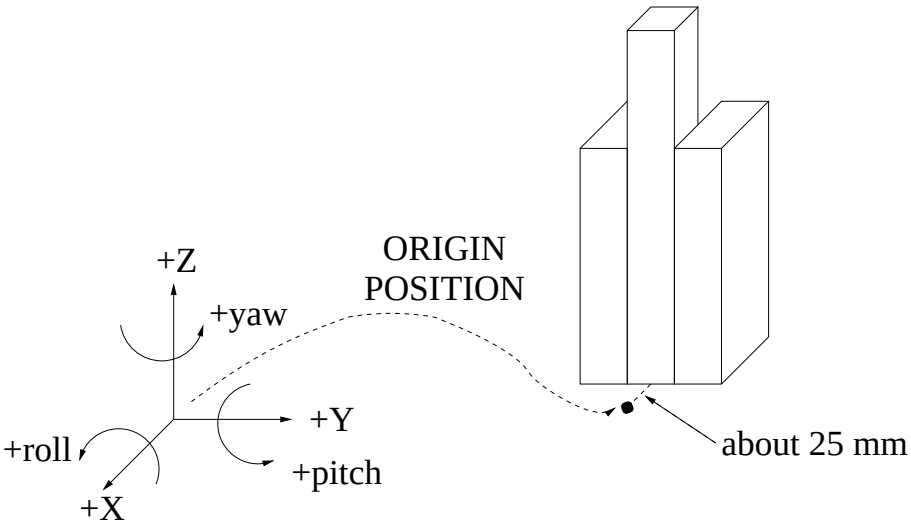
You also need these files:

- monitor
- rtx.sh
- rtx_manual.pdf

The program operates by creating 2 FIFO communication channels, one for Matlab to send commands to the monitoring process and the other for the monitoring process to send feedback to. This operates by having the background process wait until a command is sent to the command FIFO through the Matlab executable, doing this command and then sending the feedback FIFO, where the Matlab executable picks up the feedback from the FIFO. On closing of Matlab it will kill the background C process.

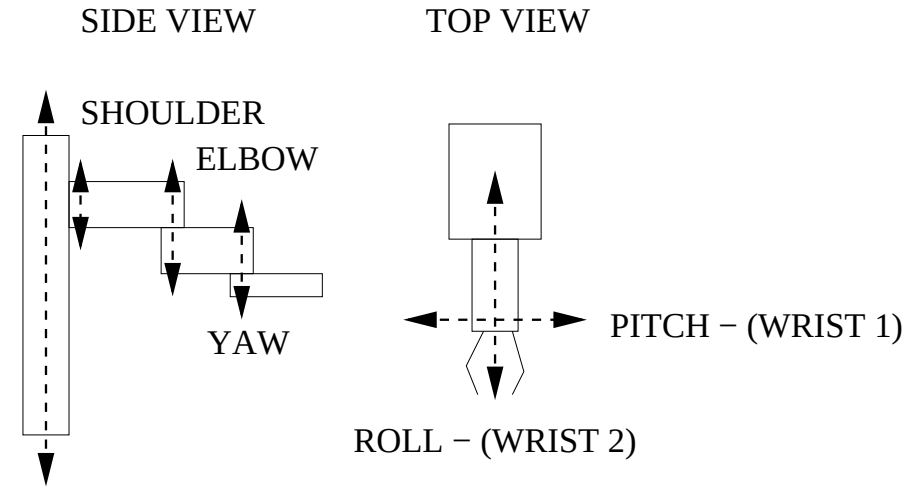
## 2 Coordinate System

The position and orientation of the robot coordinate system is:




The position that the robot is at refers to the position of the gripper tip.

The naming of the joints is:



Maximum and minimum values for joint angles, etc are:

| | Maximum | Minimum | Units |
|---|---|---|---|
| X | 680.0 | -25.0 | mm |
| Y | 550.0 | -450.0 | mm |
| Z | 912.0 | 27 | mm |
| Yaw | 110.0 | -110.0 | degrees |
| Pitch | 176.0 | 90.0 | degrees |
| Roll | 125.0 | -125.0 | degrees |
| Elbow | 140 | -140 | degrees |
| Shoulder | 85 | -85 | degrees |
| Gripper | 89.0 | 0.0 | mm |

# 3 Running the Shell Script

To run the shell script type '. rtx.sh' in the directory from which the mexglx files listed in Section 1 are stored. This will create the FIFOs, begin the monitor process and also open Matlab. The monitor process will initially set up the communications between the RTX and the Linux box and also do a cold calibration of the RTX.

You need to wait about 1-2 minutes for the RTX to self-calibrate. Ignore the two initial `joint limit exceeded` error messages. Wait for `Begin polling`. Type `rtx_home` in the MATLAB window. After the robot stops, you can then run your code.

# 4 Using Matlab

After the arm has calibrated itself (it should be in a straight line forward) you will be able to control it via Matlab. You should not give commands to it until it is finished calibrating itself. To see when the calibration sequence is finished look at the terminal from which you started the shell script, 'Begin polling' should be printed to the terminal if it has finished its calibration sequence. You may enter commands but they will not return until they have been fully executed, which will not happen until the robot has been calibrated.

The following sections explain the use of each of the commands:

```
calibrate_rtx
rtx_home
abs_position_rtx
rel_position_rtx
location_xy_rtx
abs_angle_rtx
rel_angle_rtx
location_an_rtx
open_grippers_rtx
close_grippers_rtx
rtx_exit
```

## 4.1 Calibrate

There are two different types of calibration. Cold calibration resets the robot's joints by forcing the joints to their maximum, and also warm calibration which simply resets the z axis (the conveyer belt which the arm moves upon). Warm calibration is done using the command:

```
[X,Y,Z,Yaw,Pitch,Roll] = calibrate_rtx(0)
```

This returns the (x,y,z) coordinates along with the (yaw,pitch,roll) rotations, which is the default home position (home position is approximately (680,0,700,0, 90,0)). This is the position the RTX will return to upon a call to rtx_home and also when the RTX is initially calibrated upon startup (this is a cold calibration so that on startup the RTX knows its location).

Cold calibration is done by using the command:

```
[X,Y,Z,Yaw,Pitch,Roll] = calibrate_rtx(1)
```

This does a full calibration of the joints and returns to the home position, rather than just resetting the z axis.

Use the cold calibration command if the RTX freezes (will not move but gives feedback as if it had moved) for some reason. This can be caused by an invalid angle/position has been given to it by one of the following commands. If it does not return to the home position use the rtx_home command to return it. Otherwise use the rtx_exit command and restart the shell script.

## 4.2 Move to Home Position

Moving to the home position is done using the command:

```
[X,Y,Z,Yaw,Pitch,Roll] = rtx_home
```

Very similar in operation to calibration, however does not recalibrate the RTX's joints, simply returning to the home position. Returns exactly the same list as before, (x,y,z,yaw,pitch,roll) positions, which should have approximately the values (680,0,700,0,90,0).

## 4.3 Absolute Position Movement

This function will move the position of the RTX's end effector to exactly the position specified by parameters passed to the function. It is done using the command:

```
[NX,NY,NZ,NYaw,NPitch,NRoll] = ...
        abs_position_rtx(Ta,Tp,X,Y,Z,Yaw,Pitch,Roll)
```

The "..." is Matlab's way of allowing a command to continue on the next line. If you can type the whole command on one line, then this is clearer.

Although it seems that NX should be equal to X, NY equal to Y and so on this is not necessarily the case as some coordinates may be out of reach of the RTX arm due to a variety of reasons, including the handedness of the arm which is currently set at left handed. Therefore it is important to inspect the

new position of the RTX to check whether the RTX has indeed moved. If the arm has not moved an error message will be printed out to the terminal where the shell script was initially executed, error messages are prefixed by "**".

Unfortunately the only real way to check from Matlab whether or not a command has been executed it to check whether the coordinates are the same as they were before, and this is the case for all functions that move the robot arm. Ta and Tp are the angular and position tolerances for the movement. Normally these are set to the recommended default of 4 (mm) and 4 (degrees), but several joint motions can interact with each other and so users might need to increase the tolerances.

The robot has 3 parallel axes (shoulder, elbow, yaw). When moving to a given position and orientation, the control software may choose to use different angles for these three axes than the ones you directly or indirectly specified, trading off one for another. So, these values cannot be checked. The other 5 variables can be checked (x, y, z, pitch, roll) and are compared using the tolerances mentioned above.

When a motion fails, the reported intended and actual positions/angles are reported. Also, the returned new positions and angles are all set to -1000000.

You may needed to recalibrate with `calibrate_rtx(0)` after motion failures.

## 4.4 Relative Position Movement

This function is used to do a relative movement of the RTX's position. It is executed using the command:

```
[NX,NY,NZ,NYaw,NPitch,NRoll] = ...
        rel_position_rtx(Ta,Tp,DX,DY,DZ,DYaw,DPitch,DRoll)
```

This command updates the RTX using its current position and adding the appropriate updates DX, DY, DZ, DYaw, DPitch and DRoll to it, ie. assuming currently in position (X,Y,Z,Yaw,Pitch,Roll), the updated position is therefore (X+DX,Y+DY,Z+DZ, Yaw+DYaw,Pitch+DPitch,Roll+DRoll).

See Section 4.3 for the discussion on movement failure, Ta and Tp.

## 4.5 Coordinate Location

This returns the coordinates that the RTX currently believes it is at. It is not necessarily the case that these are correct as the RTX robots are not greatly reliable. Recalibration should sort this. The Matlab command to use for this is:

```
[X,Y,Z,Yaw,Pitch,Roll] = location_xy_rtx
```

## 4.6 Absolute Joint Angle Movement

This function is used to set the angles of the joints of the RTX to exact values specified by the parameters passed to it. To use this command type the following:

```
[NElbow,NShoulder,NZ,NYaw,NPitch,NRoll] = ...
        abs_angle_rtx(Ta,Tp,Elbow,Shoulder,Z,Yaw,Pitch,Roll)
```

The parameter names above stand for:

- Elbow: elbow angle
- Shoulder: shoulder angle
- Z: zed (black belt material arm sits on, equivalent to z coordinate)
- Yaw: yaw
- Pitch: pitch
- Roll: roll

The values of the rotation of the yaw, pitch and roll are the same as in the previous functions. See Section 4.3 for the discussion on movement failure, Ta and Tp.

## 4.7 Relative Joint Angle Movement

This function takes the relative change offsets for the elbow, shoulder, zed, yaw, pitch and roll and adds them to the current parameters to get the relative position (as explained in the section on relative position movement). It operates using the command:

```
[NElbow,NSshoulder,NZ,NYaw,NPitch,NRoll] = ...
        rel_angle_rtx(Ta,Tp,DElbow,DShoulder,DZ,DYaw,DPitch,DRoll)
```

See Section 4.3 for the discussion on movement failure, Ta and Tp.

## 4.8 Location Joint Angles

This function returns the joint angles also returned by the above methods. Like location_xy_rtx it does not move the robot and only returns the angles it believes the robot to be at. Again these may not be correct if the RTX believes a move command has executed but it has not. In this case the robot must be recalibrated. The command is:

```
[Elbow,Shoulder,Zed,Yaw,Pitch,Roll] = location_an_rtx
```

Again error messages are dealt with as explained above.

## 4.9 Opening Grippers

This function opens the RTX's gripper. It takes only 1 argument - the width of the gripper opening (approximately) in mm:

```
open_grippers_rtx(Value)
```

If an invalid value is entered, it will not execute and will display an error message. Otherwise, it will open the grippers to the desired amount. However, the function will not return until the RTX has finished opening its grippers. Valid values are between 0 mm (closed) and 89 mm (fully open).

### 4.10 Close Grippers

Simple function that simply closes the RTX's gripper, equivalent to:

- open_grippers_rtx(0):

Instruction is:

```
close_grippers_rtx
```

`close_grippers_rtx` will fully close the grippers and you may want to use a a call to `open_grippers_rtx(X)` if you only want the grippers to close to value X; otherwise, the object will liable to be crushed.

### 4.11 Exit Program

This command stows the RTX, closes all files and ports it opens, closes the monitoring process and exits Matlab. It is preferable to use this instruction before closing down the application as it resets the serial port to its previous settings and close all files cleanly, which is not necessarily the case if you exit using the Matlab 'exit' command (or another equivalent).

```
rtx_exit
```

## 5 Demo Program

To help you follow the commands, the demo program `demo.m` demonstrates all of the commands, by this sequence of actions:

1. Move to the home position
2. Open grippers
3. Move to an given position (500,0,400,0,95,0) by absolute movement
4. Partially close the grippers as if picking up an object (if between the grippers when they close)
5. Observe the current joint angles.
6. Apply a relative movement to the joints
7. Find current coordinate position
8. Increment the Yaw by 30 degrees
9. Change the position relatively
10. Open the grippers as if releasing the object
11. Change the position relatively again
12. Close the grippers completely
13. exit (if uncommented)

# 6 Emergency Stop Button

Notice the red emergency stop button. When getting the robot to move, always keep this within reach, as it is easy to write programs that bump into things (esp. the table). This could damage either the object or the robot. Keep an eye on the gripper's position and hit stop if it looks like a collision is going to happen.

After hitting stop, you will need to 1) twist the stop button to release it, 2) turn the power on again (green button), 3) re-calibrate the robot and 4) fix your program.

# 7 If RTX Locks Up

Sometimes the RTX seems to lock up - when you calibrate it it doesn't move to the home position. In this case: 1) turn off power at the switch (not red button), 2) exit matlab, 3) restart RTX and use `rtx_home` when it thinks it is calibrated.

# 8 Other Oddities

Sometimes you need to issue a command twice before it does anything.

The robots seem to switch their motors off after a small rest of doing nothing. This results in no response, delayed movements or incorrect (too early) readings coming off the robot. This should not be a problem if something is being run as a program.

The grippers occasionally fail to open (often due to an incorrect position being given to the robot), and therefore will fail to close properly because of the location the robot believes the grippers to be in is inaccurate. this is alleviated by a recalibration. If this happens it will also affect `close_grippers_rtx`.