iCub Courses

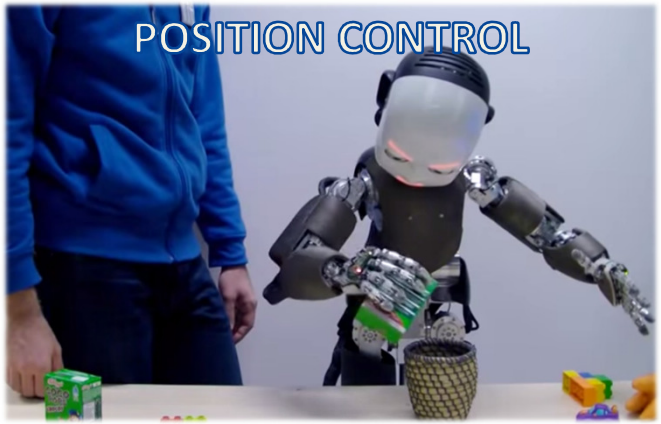
iCub Control Modes & Interaction Modes

ISTITUTO ITALIANO
DI TECNOLOGIA

iCub control modes

- The selection of a control mode allows the user to select a specific control algorithm to actuate the joint. It also implicitly defines the accepted commands (i.e. position commands, velocity command etc.).

- YARP provide overloaded methods to perform set/get operations on the control modes of:
  - individual joints (e.g. joint 0 of the iCub arm)
  - a sub set of joints belonging to a specific robot part (e.g. joints 0 1 2 of the iCub arm)
  - all the joints belonging to a specific part (e.g. the whole iCub arm)



ISTITUTO ITALIANO
DI TECNOLOGIA

iCub control modes

The **old** interface Yarp::dev::IControlMode provides a set of methods which allow to choose only between a predefined set of control modes:

- position
- velocity
- torque
- openloop

With the implementation of new control modes (e.g. position direct), a **new** yarp interface has been implemented, providing greater flexibility.

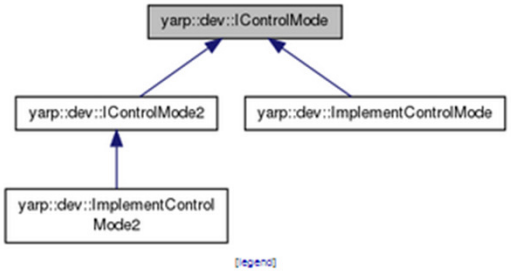
yarp::dev::IControlMode Class Reference abstract

Motor Interfaces

Interface for setting control mode in control board. More...

```
#include <yarp/dev/IControlMode.h>
```

Inheritance diagram for yarp::dev::IControlMode:



Public Member Functions

virtual	**~IControlMode** ()
virtual bool	**setPositionMode** (int j)=0 Set position mode, single axis. More...
virtual bool	**setVelocityMode** (int j)=0 Set velocity mode, single axis. More...
virtual bool	**setTorqueMode** (int j)=0 Set torque mode, single axis. More...
virtual bool	**setImpedancePositionMode** (int j)=0 Set impedance position mode, single axis. More...
virtual bool	**setImpedanceVelocityMode** (int j)=0 Set impedance velocity mode, single axis. More...
virtual bool	**setOpenLoopMode** (int j)=0 Set open loop mode, single axis. More...
virtual bool	**getControlMode** (int j, int *mode)=0 Get the current control mode. More...
virtual bool	**getControlModes** (int *modes)=0 Get the current control mode (multiple joints). More...

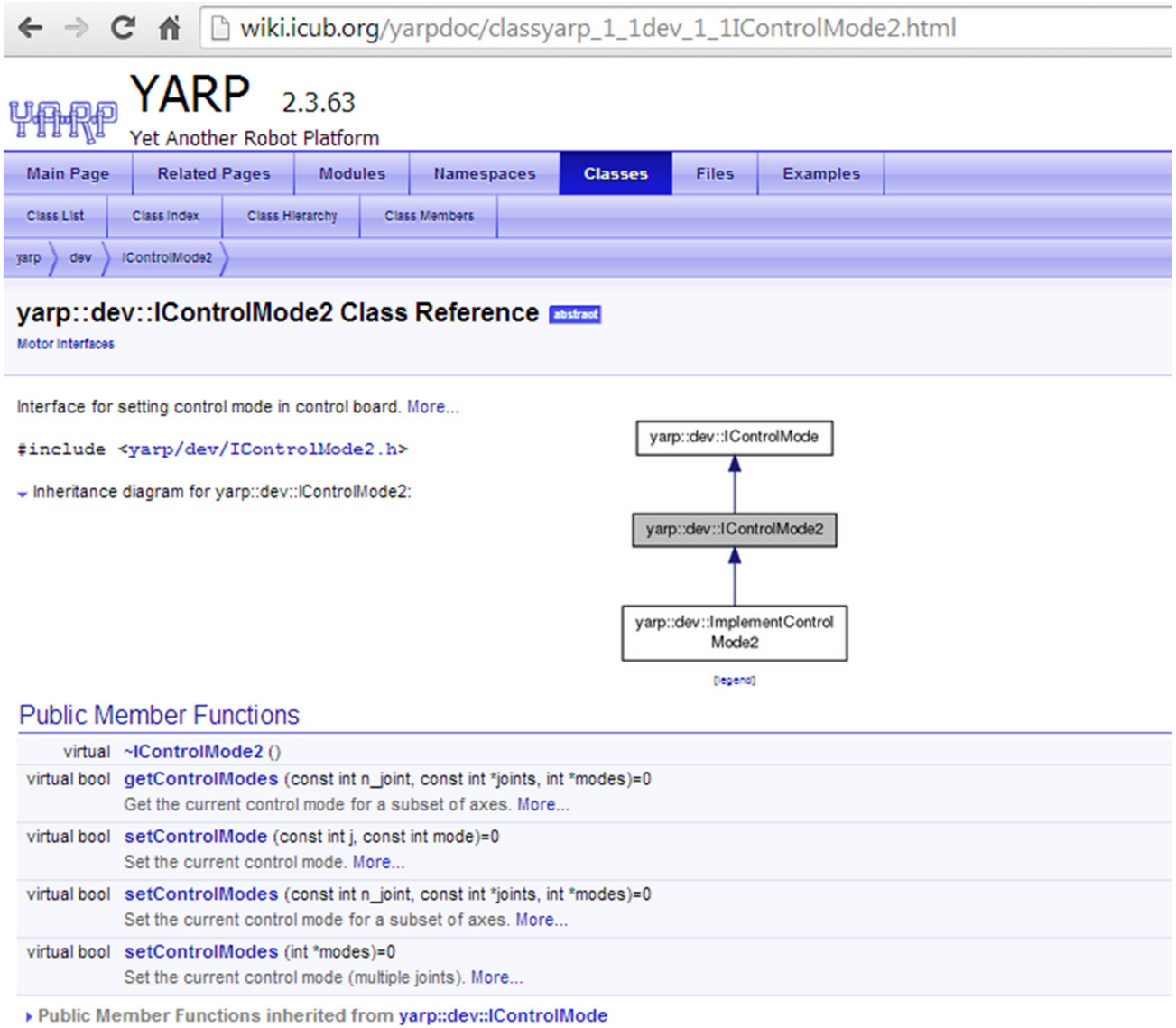
iCub control modes

Since yarp 2.3.63,
there exists a new interface:

Yarp::dev::IControlMode2

- It is now possible to operate on multiple joints.
- Control mode is now identified by an integer (yarp::os::vocab)
  e.g. setControlMode (int j, int mode)

Use old interface will be deprecated soon. Use the new interface for your application !



ISTITUTO ITALIANO
DI TECNOLOGIA

iCub control modes

The protocol defines the following control modes, identified by a yarp::os::Vocab code.

• Position control with trajectory generation[1]	VOCAB_CM_POSITION
• Direct position control	VOCAB_CM_POSITION_DIRECT
• Velocity control	VOCAB_CM_VELOCITY
• Mixed Position-Velocity control	VOCAB_CM_MIXED
• Torque Control	VOCAB_CM_TORQUE
• OpenLoop Control	VOCAB_CM_OPENLOOP
• Idle	VOCAB_CM_IDLE

[1] In the iCub trajectories are implemented following a minimum jerk profile, but this is not strictly enforced by this specifications (other robots can implement conventional trapezoidal profile).

- Control modes are always explicitly chosen by the user using the proper YARP interfaces: no automatic switch are performed by the system (with the only exceptions of the special board statuses described in Section 3)

- Only the mixed position-velocity control is allowed to accept two command types (i.e. position and velocity commands). All other control modes accept only the command types specific of their interface.

Control modes and interaction modes

In addition to the previously described control modes, YARP provides two interaction modes: stiff interaction and compliant interaction.

• Stiff Interaction	VOCAB_INTERACTION_STIFF
• Compliant Interaction	VOCAB_INTERACTION_COMPLIANT

- Stiff interaction is the typical interaction mode of 'industrial' robots, which are required to execute accurate position/velocity trajectories in controlled environments.

- Using compliant interaction mode the user can set specific joint impedance (i.e. stiffness σ and damping μ) during the execution of position or velocity commands.

The new interface yarp::dev::IInteractionMode replaces the old control modes ImpendancePositionMode / ImpedanceVelocityMode

YARP 2.3.63
Yet Another Robot Platform

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples

Class List | Class Index | Class Hierarchy | Class Members

yarp | dev | IInteractionMode

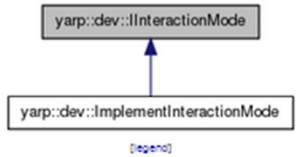
yarp::dev::IInteractionMode Class Reference abstract

Motor Interfaces

Interface settings the way the robot interacts with the environment: basic interaction types are Stiff and Compliant. More...

```
#include <yarp/dev/IInteractionMode.h>
```

Inheritance diagram for yarp::dev::IInteractionMode:



Public Member Functions

virtual	**~IInteractionMode** () Destructor. More...
virtual bool	**getInteractionMode** (int axis, **yarp::dev::InteractionModeEnum** *mode)=0 Get the current interaction mode of the robot, values can be stiff or compliant. More...
virtual bool	**getInteractionModes** (int n_joints, int *joints, **yarp::dev::InteractionModeEnum** *modes)=0 Get the current interaction mode of the robot for a set of joints, values can be stiff or compliant. More...
virtual bool	**getInteractionModes** (**yarp::dev::InteractionModeEnum** *modes)=0 Get the current interaction mode of the robot for a all the joints, values can be stiff or compliant. More...
virtual bool	**setInteractionMode** (int axis, **yarp::dev::InteractionModeEnum** mode)=0 Set the interaction mode of the robot, values can be stiff or compliant. More...
virtual bool	**setInteractionModes** (int n_joints, int *joints, **yarp::dev::InteractionModeEnum** *modes)=0 Set the interaction mode of the robot for a set of joints, values can be stiff or compliant. More...
virtual bool	**setInteractionModes** (**yarp::dev::InteractionModeEnum** *modes)=0 Set the interaction mode of the robot for a all the joints, values can be stiff or compliant. More...

Control modes and interaction modes

- The selection of the interaction mode will affect only the joint control algorithm, but will not alter the type of accepted commands.

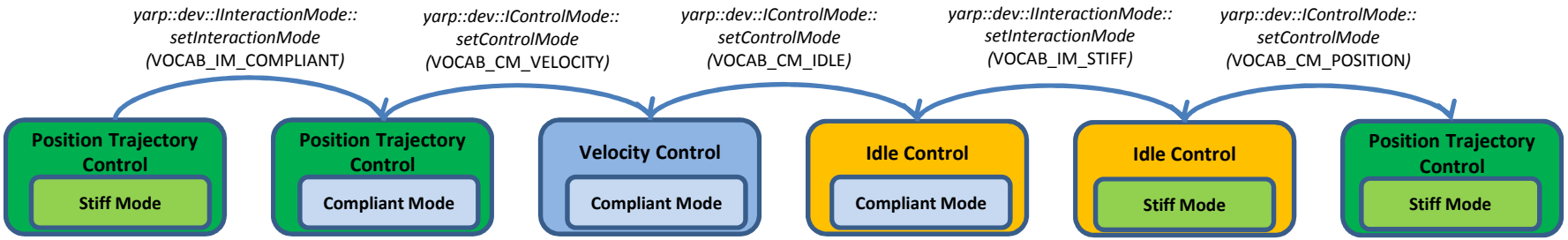
- The full set possible control modes/interaction modes combinations are summarized in the following table:

Control Mode	Accepted motor commands	Interaction mode	Additional parameters
Position control with trajectory generation	q Yarp::dev::IPositionControl::positionMove()	Stiff mode	---
		Compliant mode	σ, μ
Direct position control	q Yarp::dev::IPositionDirect::setPosition()	Stiff mode	---
		Compliant mode	σ, μ
Velocity control	$\dot{q}$ Yarp::dev::IVelocityControl::velocityMove()	Stiff mode	---
		Compliant mode	σ, μ
Mixed Position-Velocity control	$q, \dot{q}$ Yarp::dev::IPositionControl::positionMove() Yarp::dev::IVelocityControl::velocityMove()	Stiff mode	---
		Compliant mode	σ, μ
Torque Control	τ Yarp::dev::ITorqueControl::setRefTorque()	---	
OpenLoop Control	φ Yarp::dev::IOpenLoopControl::setRefOutput()	---	
Idle	---	---	

Transitions between control modes

joint control mode and interaction mode are selected by the methods:

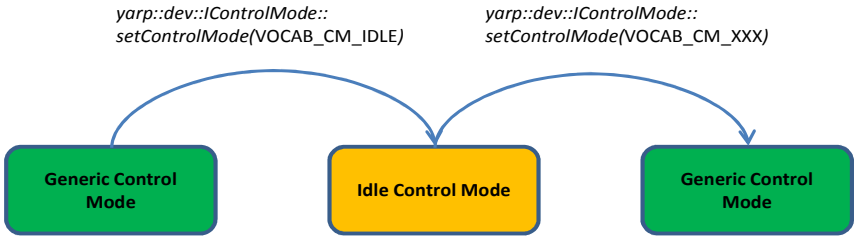
- *yarp::dev::IControlMode::setControlMode()*
- *yarp::dev::IInteractionMode::setInteractionMode()*

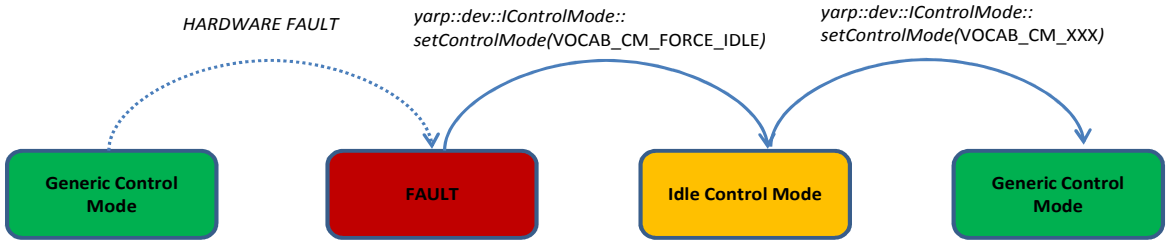


- The selection of stiff/compliant interaction mode is meaningful only when the current control mode is position, velocity or mixed.
- In all other control modes, such as idle mode, torque mode, openloop etc., choosing the interaction mode has no effect on the control, but is accepted and stored in the internal status of the board.
- The current the controller when the controller will be again set to a control mode which supports the interaction mode.
- To turn off a motor user can set a joint in idle control mode (PWM is off)

Board status

- There exists control modes which cannot be set by the user and are used to indicate a particular status of the board (e.g. fault, calibration in progress etc.)
- Board statuses are handled like control modes and are retrieved by getControlMode()
- The transition from a particular board status to a normal control mode with setControlMode() may be restricted, depending on the board status.





- If a control board is faulted, the user has to send the special command setControlMode(VOCAB_CM_FORCE_IDLE) to reset the fault before choosing any other desired control mode.
- Sending the VOCAB_CM_FORCE_IDLE special command when the board is not in fault status has the same effect of setting VOCAB_CM_IDLE control mode.

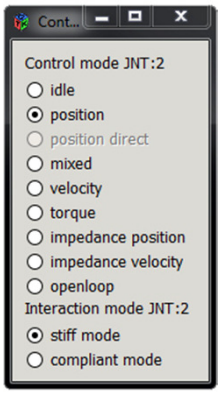
Control modes in the robotMotorGui

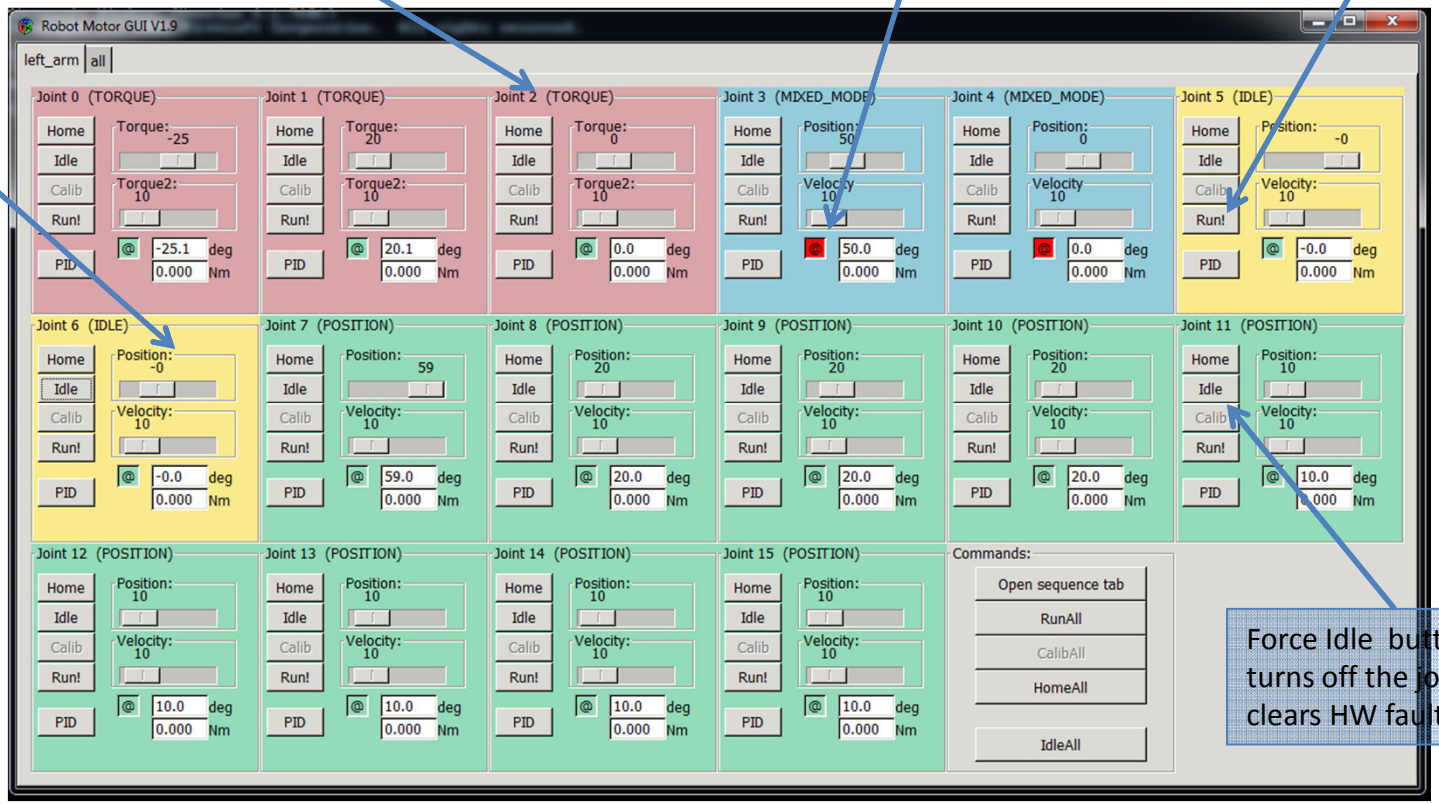
Background color and box title indicates the current control mode

Interaction mode:
Green -> stiff
Red -> compliant

Run button set control mode to position, it does NOT change interaction mode

Right click to change control mode

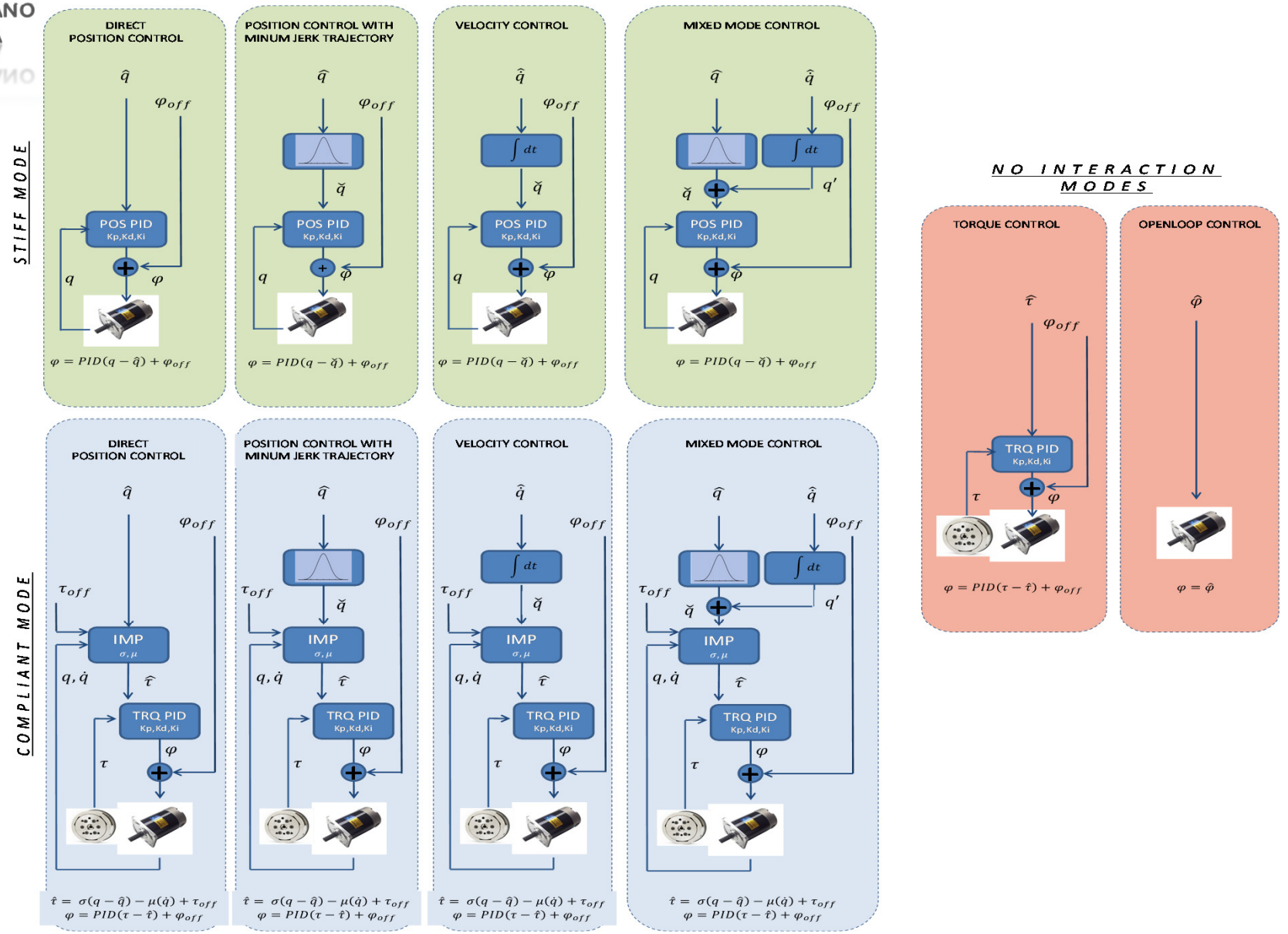




Force Idle button turns off the joint and clears HW fault

ISTITUTO ITALIANO
DI TECNOLOGIA

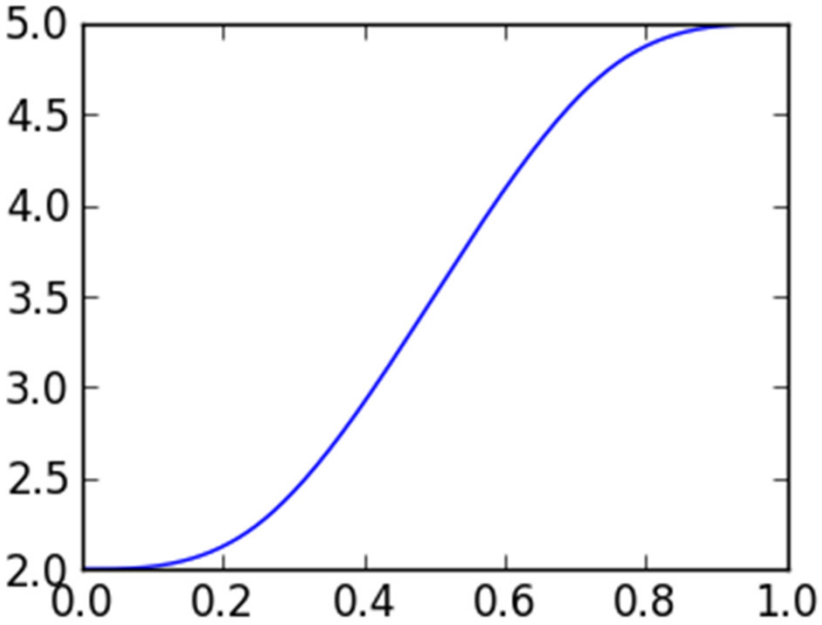
iCub Control Modes



Position mode vs position direct mode

The new yarp::dev::IControlMode2 interface explicitly supports direct position control (VOCAB_CM_POSITION_DIRECT).

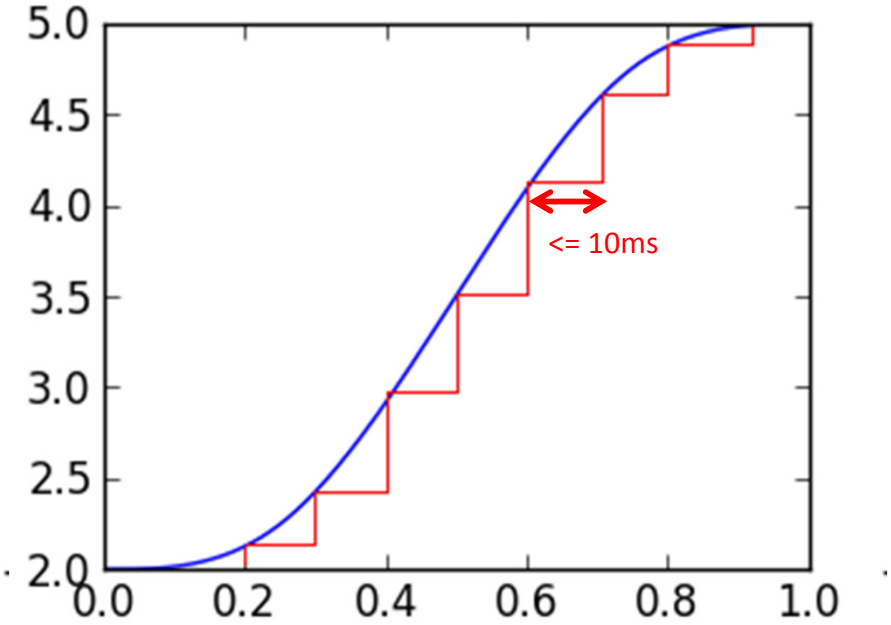
Minimum jerk trajectory



VOCAB_CM_POSITION
Yarp::dev::IPositionControl::positionMove()

- Only minimum jerk trajectory
- Easy to use: only one command needed
- Safe

Trajectory generated with position direct



VOCAB_CM_POSITION_DIRECT
Yarp::dev::IPositionDirect::setPosition()

- More powerful (arbitrary trajectory)
- You need to sample your trajectory, compute position steps and send them continuously (like in velocity control)
- Avoid to use it for large step movement!

Differences respect to old APIs

- Previous APIs allow automatic switching between Position and Velocity mode (i.e. a VelocityMove () command sent in positionMode automatically sets the joint in velocityMode). This is no more supported, a velocityMove() command is rejected, unless the MixedMode is used.

- ImpedancePositionMode and ImpedanceVelocityMode are now deprecated. Use setInteractionMode() to put a joint in compliant mode.

- Previous API do not distinguish between positionMode and positionDirectMode in terms of control mode (both of them are considered “position mode”).

- Previous APIs do not distinguish between idle mode and hardware fault. Now the user is forced to send a special command to reset the fault.

- Previous APIs do not allow to set IdleMode (iAmplifierControl::disableAmp() method is used), but getControlMode() can return “idle” if a fault occurs on the board.

- The following old methods are now deprecated. Use yarp::dev::IControlMode instead.
  yarp::dev::IAmplifierControl::enableAmp()
  yarp::dev::IAmplifierControl::disableAmp()
  yarp::dev::IPidControl::enablePid()
  yarp::dev::IPidControl::disablePid()
  yarp::dev::iTorqueControl::enableTorquePid()
  yarp::dev::ITorqueControl::disableTorquePid()

ISTITUTO ITALIANO
DI TECNOLOGIA

Examples

OK

```
…
ictrl->setControlMode(j,VOCAB_CM_POSITION);
ipos->positionMove(j,command);
…
ictrl->setControlMode(j,VOCAB_CM_VELOCITY);
ivel->velocityMove(j,command);
…
ictrl->setControlMode(j,VOCAB_CM_POSITION);
ipos->positionMove(j,command);
…
ipos->positionMove(j,command);
…
```

BAD

```
…
ipos->positionMove(j,command);
…
ivel->velocityMove(j,command);
…
ivel->velocityMove(j,command);
…
ipos->positionMove(j,command);
…
```

OK

```
…
ictrl->setControlMode(i,VOCAB_CM_POSITION);
iint->setInteractionMode(i,VOCAB_IM_COMPLIANT);
ipos->positionMove(i,command);
…
iint->setInteractionMode(i,VOCAB_IM_STIFF);
ipos->positionMove(i,command);
…
```

BAD

```
…
ictrl->setImpedancePositionMode(j);
…
ipos->positionMove(j,command);
…
ictrl->setPositionMode(j);
…
ipos->positionMove(j,command);
…
```

ISTITUTO ITALIANO
DI TECNOLOGIA

Interaction Mode example

FROM: icub-tutorials\src\motorControlImpedance\main.cpp
(https://github.com/robotology/icub-tutorials.git)

```
…
IPositionControl *pos;
IControlMode2    *ictrl;
IInteractionMode *iint;
…
robotDevice.view (pos);
robotDevice.view (ictrl);
robotDevice.view (iint);
…
while(true)
{
        times++;
        if (times%2)
        {
             // set the elbow joint in compliant mode
             ictrl->setControlMode(3,VOCAB_CM_POSITION);
             iint->setInteractionMode(3,VOCAB_IM_COMPLIANT);
             // set new reference positions
               command=60;
        }
        else
        {
             // set the elbow joint in stiff mode
             ictrl->setControlMode(3,VOCAB_CM_POSITION);
             iint->setInteractionMode(3,VOCAB_IM_STIFF);
             // set new reference positions
             command=30;
        }
        pos->positionMove(3,command);
        yarp::os::Time::delay(2.0);
}
```

ISTITUTO ITALIANO
DI TECNOLOGIA

Position Direct example

FROM: icub-main\src\modules\demoForceImitation\main.cpp
(https://github.com/robotology/icub-main.git)

```
…
if (left_arm_master)
   {
       for (int i=0; i<5; i++)
       {
           robot->icmd[LEFT_ARM] ->setControlMode(i, VOCAB_CM_TORQUE);
           robot->icmd[RIGHT_ARM]->setControlMode(i, VOCAB_CM_POSITION_DIRECT);
           robot->iint[RIGHT_ARM]->setInteractionMode(i,VOCAB_IM_COMPLIANT);
       }
   }
   else
   {
       for (int i=0; i<5; i++)
       {
           robot->icmd[RIGHT_ARM]->setControlMode(i, VOCAB_CM_TORQUE);
           robot->icmd[LEFT_ARM] ->setControlMode(i, VOCAB_CM_POSITION_DIRECT);
           robot->iint[LEFT_ARM] ->setInteractionMode(i,VOCAB_IM_COMPLIANT);
       }
   }
…
if (left_arm_master)
   {
        robot->ienc[LEFT_ARM] ->getEncoders(encoders_master);
        robot->ienc[RIGHT_ARM]->getEncoders(encoders_slave);
        for (int i=0; i<5; i++) {robot->iposd[RIGHT_ARM]->setPosition(i,encoders_master[i]);}
   }
   else
   {
        robot->ienc[RIGHT_ARM]->getEncoders(encoders_master);
        robot->ienc[LEFT_ARM] ->getEncoders(encoders_slave);
        for (int i=0; i<5; i++) {robot->iposd[LEFT_ARM]->setPosition(i,encoders_master[i]);}
   }
```