

Aria Reference Manual
2.1.0

Generated by Doxygen 1.2.10

Wed Oct 6 18:36:32 2004

Contents

1	Aria Hierarchical Index	1
1.1	Aria Class Hierarchy	1
2	Aria Compound Index	3
2.1	Aria Compound List	3
3	Aria Class Documentation	5
3.1	ArPanTiltCube Class Reference	5
3.2	ArPowerCubeArm Class Reference	7
3.3	ArPowerCubeController Class Reference	12

Chapter 1

Aria Hierarchical Index

1.1 Aria Class Hierarchy

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

ArASyncTask	
ArPanTiltCube	5
ArPowerCubeArm	7
ArPowerCubeController	12



Chapter 2

Aria Compound Index

2.1 Aria Compound List

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

ArPanTiltCube (An async thread that controls the power cube joint)	5
ArPowerCubeArm (A class for interfacing with the Amtec arm via CANbus card)	7
ArPowerCubeController (A class for interfacing with the Amtec arm via CANbus card)	12



Chapter 3

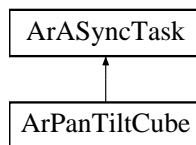
Aria Class Documentation

3.1 ArPanTiltCube Class Reference

An async thread that controls the power cube joint.

```
#include <ArPanTiltCube.h>
```

Inheritance diagram for ArPanTiltCube::



Public Methods

- **ArPanTiltCube** (ArRobot *robot)
Constructor.
 - **~ArPanTiltCube** ()
Destructor.
 - void **setPanAcc** (double acceleration)
Set the acceleration and velocities.
 - bool **isTilting** ()
Is the joint moving?
-

- void **setMaxTilt** (double max)
Set the max and min joint positions.
- void **tilt** (double deg)
Move the joint to an absolute position.
- void **tiltRel** (double deg)
Move the joint to a relative position.
- double **getPan** ()
Get pan and tilt positions.
- void * **runThread** (void *ptr)
The thread that updates the joint's current position.

Protected Attributes

- **ArPowerCubeController * myController**
The power cube joint.
- **ArRobot * myRobot**
The robot.
- double **currentPitch**
The pitch (tilt) and yaw (pan).
- **ArInterpolation * jointHistory**
The history of positions.

3.1.1 Detailed Description

An async thread that controls the power cube joint.

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

- ArPanTiltCube.h
- ArPanTiltCube.cpp

3.2 ArPowerCubeArm Class Reference

A class for interfacing with the Amtec arm via CANbus card.

```
#include <ArPowerCubeArm.h>
```

Public Methods

- **ArPowerCubeArm** ()
Constructor.
- virtual **~ArPowerCubeArm** ()
Destructor.
- int **init** (void)
Initialization routine.
- void **reset** (void)
Reset error codes in modules.
- void **resetJoint** (int modNum)
Reset error codes in one module.
- void **halt** (void)
Halt all modules.
- void **homeArm** (void)
Home all joints on the arm, and move to zero.
- void **homeJoint** (int modNum)
Home a single joint.
- void **park** (void)
Park the arm.
- void **moveArmPTP** (float pos1, float pos2, float pos3, float pos4, float pos5, float pos6)
Move all joints of the arm to given positions (in degrees).
- void **moveJoint** (int modNum, float pos)
Move a single joint to a given position (in degrees).
- void **setJointCurrent** (int modNum, float current)

Set the current of a joint in +/- amps.

- void **moveGrip** (float pos)
Move the gripper to a position (in mm).
- void **closeGrip** (float current)
Close the gripper. Takes negative current.
- void **getArmPos** (float *pos1, float *pos2, float *pos3, float *pos4, float *pos5, float *pos6)
Get the six main joint positions (in degrees).
- void **getJointPos** (int modNum, float *pos)
Get a single joint position (in degrees).
- void **getGripPos** (float *pos)
Get the gripper position (in mm).
- void **getArmState** (unsigned long *state1, unsigned long *state2, unsigned long *state3, unsigned long *state4, unsigned long *state5, unsigned long *state6)
Get the states of the six main joints.
- void **getJointState** (int modNum, unsigned long *state)
Get a single joint state.
- void **getGripState** (unsigned long *state)
Get the gripper state.
- void **waitForArm** (void)
Block until the six joints complete their movements.
- void **waitForJoint** (int modNum)
Block until a single joint finishes its movement.
- void **waitForGrip** (void)
Block until the gripper finished its movement.
- bool **isParked** (void)
Returns true if the arm is close to a parked position.
- bool **isHomed** (void)
Returns true if the arm has been homed.

- bool **isError** (void)
Returns true if there is an error on the arm.
- bool **isJointError** (int modNum)
Returns true if there is an error on a joint.
- void **setJointVel** (int modNum, float velPercent)
Sets the joint velocities as a percent of the max (deg/s).
- void **setJointAcc** (int modNum, float accPercent)
Sets the joints accelerations as a percent of the max (deg/s/s).
- void **setArmVel** (float velPercent)
Sets a single joint velocity to a percent of its max (deg/s).
- void **setArmAcc** (float accPercent)
Sets a single joint acceleration to a percent of its max (deg/s/s).
- void **exit** (void)
Closes the device connection to the CANBus card.

3.2.1 Detailed Description

A class for interfacing with the Amtec arm via CANbus card.

This class is for controlling the Amtec robotic arm on the powerbot platform via a PCI331 CANBus card

This is a basic wrapper for the libraries supplied by Amtec. It provides a simpler interface, as well as some pre-packaged commands, like parking the arm on robot. This is a very simple class, and if more complex function are needed, then the Amtec libraries should be used.

Always exercise care when using the arm! It is very heavy and very strong!

3.2.2 Member Function Documentation

3.2.2.1 void ArPowerCubeArm::homeArm (void)

Home all joints on the arm, and move to zero.

This homes the arm in a fashion designed to work from the park position. It first raises joint 2 to straight up in the air, then works on the others. This is done to reduce the turning radius of the arm.

It centers all the joints when it completes.

3.2.2.2 int ArPowerCubeArm::init (void)

Initialization routine.

This initializes the CANBus card, checks the number of modules, and sets the min and max vel and acc ratings.

It must be called before the other functions can be called.

3.2.2.3 bool ArPowerCubeArm::isParked (void)

Returns true if the arm is close to a parked position.

This checks for whether the arm positions are within PARK amount of the original park positions.

3.2.2.4 void ArPowerCubeArm::park (void)

Park the arm.

Parking the arm will attempt to lay it flat on top of the powerbot's topplate. This is done in way that gets all of the joints within the boudnaries of the plate, minizing risk of hitting anything with the arm while driving around.

3.2.2.5 void ArPowerCubeArm::waitForArm (void)

Block until the six joints complete their movements.

The waitForArm command waits for all six modules to finish a variety of states. The wait will block if there is motion on any joint, there used to be motion, but the arm is not at the end, yet, there are no error conditions, and none of the joints are in braking only condition.

3.2.2.6 void ArPowerCubeArm::waitForGrip (void)

Block until the gripper finished its movement.

The gripper is different than the other joints, in that it may stop motion for a period, while the object it is gripping compresses, but then may resume motion.

Thus, two velocities are checked, and it is assumed that if both times they're zero, then the gripper has really stopped.

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

- ArPowerCubeArm.h
- ArPowerCubeArm.cpp

3.3 ArPowerCubeController Class Reference

A class for interfacing with the Amtec arm via CANbus card.

```
#include <ArPowerCubeController.h>
```

Public Methods

- **ArPowerCubeController** ()
Constructor.
- virtual **~ArPowerCubeController** ()
Destructor.
- int **init** (void)
Initialization routine.
- void **reset** (void)
Reset error codes in modules.
- void **haltAll** (void)
Halt all modules.
- void **homeAll** (void)
Home all joints, and move to zero.
- void **homeJoint** (int modNum)
Home a single joint.
- void **moveJoint** (int modNum, float pos)
Move a single joint to a given position (in degrees).
- void **setJointCurrent** (int modNum, float current)
Set the current of a joint in +/- amps.
- void **getJointPos** (int modNum, float *pos)
Get a single joint position (in degrees).
- void **getJointState** (int modNum, unsigned long *state)
Get a single joint state.
- void **waitForAll** (void)
Block until the six joints complete their movements.

- void **waitForJoint** (int modNum)
Block until a single joint finishes its movement.
- bool **isArmMoving** (void)
Returns true if any joints are currently moving.
- bool **isJointMoving** (int modNum)
Returns true if the requested joint is still moving.
- bool **isHomed** (void)
Returns true if all the modules are homed.
- bool **isError** (void)
Returns true if there is an error on a joint.
- void **setJointVel** (int modNum, float velPercent)
Sets the joint velocities as a percent of the max (deg/s).
- void **setJointAcc** (int modNum, float accPercent)
Sets the joints accelerations as a percent of the max (deg/s/s).
- void **setArmVel** (float velPercent)
Sets a single joint velocity to a percent of its max (deg/s).
- void **setArmAcc** (float accPercent)
Sets a single joint acceleration to a percent of its max (deg/s/s).
- void **setJointPID** (int modID, float C0, float damp, float A0)
Sets the PID values for a particular joint - C0, damp, A0.
- void **exit** (void)
Closes the device connection to the CANBus card.
- int **getNumMods** (void)
Returns the number of modules found on the bus.
- int **getMaxNumMods** (void)
This number should be set to the same as the driver, which is currently 31.
- int * **getModIDs** (void)

Returns the IDs of the modules found on the bus. The returned array is a list of the modules, starting with `myModIDs[0]` as the first one. The modules IDs have to be used as the 'joint number'.

- float **getMaxPos** (int modNum)

Returns the maximum position, in radians, that a joint may turn to.

- float **getMinPos** (int modNum)

Returns the minimum position, in radians, that a joint may turn to.

3.3.1 Detailed Description

A class for interfacing with the Amtec arm via CANbus card.

This class is for controlling the Amtec robotic arm on the powerbot platform via a PCI331 CANBus card

This is a basic wrapper for the libraries supplied by Amtec. This will eventually be the base class for **ArPowerCubeArm** (p. 7), though hasn't been set up that way, yet. It also doesn't know about different joint types (i.e. gripper, rotary, linear) and treats them all the same.

3.3.2 Member Function Documentation

3.3.2.1 int ArPowerCubeController::init (void)

Initialization routine.

This initializes the CANBus card, checks the number of modules, and sets the min and max vel and acc ratings.

It must be called before the other functions can be called.

3.3.2.2 void ArPowerCubeController::waitForAll (void)

Block until the six joints complete their movements.

The `waitForArm` command waits for all six modules to finish a variety of states. The wait will block if there is motion on any joint, there used to be motion, but the arm is not at the end, yet, there are no error conditions, and none of the joints are in braking only condition.

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

- ArPowerCubeController.h

- ArPowerCubeController.cpp

Index

- ~ArPanTiltCube
 - ArPanTiltCube, 5
 - ~ArPowerCubeArm
 - ArPowerCubeArm, 7
 - ~ArPowerCubeController
 - ArPowerCubeController, 12
 - ArPanTiltCube
 - ~ArPanTiltCube, 5
 - ArPanTiltCube, 5
 - currentPitch, 6
 - getPan, 6
 - isTilting, 5
 - jointHistory, 6
 - myController, 6
 - myRobot, 6
 - runThread, 6
 - setMaxTilt, 6
 - setPanAcc, 5
 - tilt, 6
 - tiltRel, 6
 - ArPanTiltCube, 5
 - ArPowerCubeArm
 - ~ArPowerCubeArm, 7
 - ArPowerCubeArm, 7
 - closeGrip, 8
 - exit, 9
 - getArmPos, 8
 - getArmState, 8
 - getGripPos, 8
 - getGripState, 8
 - getJointPos, 8
 - getJointState, 8
 - halt, 7
 - homeJoint, 7
 - isError, 9
 - isHomed, 8
 - isJointError, 9
 - moveArmPTP, 7
 - moveGrip, 8
 - moveJoint, 7
 - reset, 7
 - resetJoint, 7
 - setArmAcc, 9
 - setArmVel, 9
 - setJointAcc, 9
 - setJointCurrent, 7
 - setJointVel, 9
 - waitForJoint, 8
 - ArPowerCubeArm, 7
 - homeArm, 9
 - init, 10
 - isParked, 10
 - park, 10
 - waitForArm, 10
 - waitForGrip, 10
 - ArPowerCubeController
 - ~ArPowerCubeController, 12
 - ArPowerCubeController, 12
 - exit, 13
 - getJointPos, 12
 - getJointState, 12
 - getMaxNumMods, 13
 - getMaxPos, 14
 - getMinPos, 14
 - getModIDs, 13
 - getNumMods, 13
 - haltAll, 12
 - homeAll, 12
 - homeJoint, 12
 - isArmMoving, 13
 - isError, 13
 - isHomed, 13
-

- isJointMoving, 13
- moveJoint, 12
- reset, 12
- setArmAcc, 13
- setArmVel, 13
- setJointAcc, 13
- setJointCurrent, 12
- setJointPID, 13
- setJointVel, 13
- waitForJoint, 13
- ArPowerCubeController, 12
 - init, 14
 - waitForAll, 14
- closeGrip
 - ArPowerCubeArm, 8
- currentPitch
 - ArPanTiltCube, 6
- exit
 - ArPowerCubeArm, 9
 - ArPowerCubeController, 13
- getArmPos
 - ArPowerCubeArm, 8
- getArmState
 - ArPowerCubeArm, 8
- getGripPos
 - ArPowerCubeArm, 8
- getGripState
 - ArPowerCubeArm, 8
- getJointPos
 - ArPowerCubeArm, 8
 - ArPowerCubeController, 12
- getJointState
 - ArPowerCubeArm, 8
 - ArPowerCubeController, 12
- getMaxNumMods
 - ArPowerCubeController, 13
- getMaxPos
 - ArPowerCubeController, 14
- getMinPos
 - ArPowerCubeController, 14
- getModIDs
 - ArPowerCubeController, 13
- getNumMods
 - ArPowerCubeController, 13
- getPan
 - ArPanTiltCube, 6
- halt
 - ArPowerCubeArm, 7
- haltAll
 - ArPowerCubeController, 12
- homeAll
 - ArPowerCubeController, 12
- homeArm
 - ArPowerCubeArm, 9
- homeJoint
 - ArPowerCubeArm, 7
 - ArPowerCubeController, 12
- init
 - ArPowerCubeArm, 10
 - ArPowerCubeController, 14
- isArmMoving
 - ArPowerCubeController, 13
- isError
 - ArPowerCubeArm, 9
 - ArPowerCubeController, 13
- isHomed
 - ArPowerCubeArm, 8
 - ArPowerCubeController, 13
- isJointError
 - ArPowerCubeArm, 9
- isJointMoving
 - ArPowerCubeController, 13
- isParked
 - ArPowerCubeArm, 10
- isTilting
 - ArPanTiltCube, 5
- jointHistory
 - ArPanTiltCube, 6
- moveArmPTP
 - ArPowerCubeArm, 7
- moveGrip
 - ArPowerCubeArm, 8
- moveJoint
 - ArPowerCubeArm, 7
 - ArPowerCubeController, 12

myController
 ArPanTiltCube, 6

myRobot
 ArPanTiltCube, 6

park
 ArPowerCubeArm, 10

reset
 ArPowerCubeArm, 7
 ArPowerCubeController, 12

resetJoint
 ArPowerCubeArm, 7

runThread
 ArPanTiltCube, 6

setArmAcc
 ArPowerCubeArm, 9
 ArPowerCubeController, 13

setArmVel
 ArPowerCubeArm, 9
 ArPowerCubeController, 13

setJointAcc
 ArPowerCubeArm, 9
 ArPowerCubeController, 13

setJointCurrent
 ArPowerCubeArm, 7
 ArPowerCubeController, 12

setJointPID
 ArPowerCubeController, 13

setJointVel
 ArPowerCubeArm, 9
 ArPowerCubeController, 13

setMaxTilt
 ArPanTiltCube, 6

setPanAcc
 ArPanTiltCube, 5

tilt
 ArPanTiltCube, 6

tiltRel
 ArPanTiltCube, 6

waitForAll
 ArPowerCubeController, 14

waitForArm
 ArPowerCubeArm, 10

waitForGrip
 ArPowerCubeArm, 10

waitForJoint
 ArPowerCubeArm, 8
 ArPowerCubeController, 13