/\*Ch version of the header file\*/

#ifndef LINKBOT\_WRAPPER\_HPP\_

#define LINKBOT\_WRAPPER\_HPP\_

#ifndef \_WIN32

#include <unistd.h>

#else

#include <windows.h>

#define sleep(x) Sleep((x)\*1000)

#endif

#include<array.h>

#ifndef ROBOT\_RECORD\_DATA\_T

#define ROBOT\_RECORD\_DATA\_T

typedef double\* robotRecordData\_t;

#endif

#ifndef ROBOT\_JOINTS\_E

#define ROBOT\_JOINTS\_E

typedef enum robotJoints\_e {

ROBOT\_ZERO,

JOINT1,

JOINT2,

JOINT3,

JOINT4,

ROBOT\_NUM\_JOINTS = 4

} robotJointId\_t;

#endif

#ifndef ROBOT\_JOINT\_STATE\_E

#define ROBOT\_JOINT\_STATE\_E

typedef enum robotJointState\_e

{

ROBOT\_NEUTRAL = 0,

ROBOT\_FORWARD,

ROBOT\_BACKWARD,

ROBOT\_HOLD,

ROBOT\_POSITIVE,

ROBOT\_NEGATIVE,

ROBOT\_ACCEL,

} robotJointState\_t;

#endif

struct LinkbotImpl;

/\*Linkbot I\*/

class CLinkbotI {

public:

CLinkbotI(...);

~CLinkbotI();

int connect();

int connectWithSerialID(const char\* id);

void disconnect();

/\* GETTERS \*/

void getAccelerometerData(double &x, double &y, double &z);

void getBatteryVoltage(double &voltage);

void getDistance(double &distance, double radius);

void getJointAngle(robotJointId\_t id, double &angle);

void getJointAngles(double &angle1, double &angle2, double &angle3);

void getJointAngleInstant(robotJointId\_t id, double &angle);

void getJointAnglesInstant(double &angle1, double &angle2, double &angle3);

void getJointSafetyAngle(double &angle);

void getJointSafetyAngleTimeout(double &timeout);

void getJointSpeed(robotJointId\_t id, double &speed);

void getJointSpeedRatio(robotJointId\_t id, double &ratio);

void getJointSpeeds(double &speed1, double &speed2, double &speed3);

void getJointSpeedRatios(double &ratio1, double &ratio2, double &ratio3);

void getLEDColorRGB(int &r, int &g, int &b);

void getLEDColor(string\_t &color);

/\* SETTERS \*/

void setBuzzerFrequency(int frequency, double time);

void setBuzzerFrequencyOn(int frequency);

void setBuzzerFrequencyOff();

void setJointMovementStateNB(robotJointId\_t id, robotJointState\_t dir);

void setJointMovementStateTime(robotJointId\_t id, robotJointState\_t dir, double seconds);

void setJointSafetyAngle(double angle);

void setJointSafetyAngleTimeout(double timeout);

void setJointSpeed(robotJointId\_t id, double speed);

void setJointSpeeds(double speed1, double speed2, double speed3);

void setJointSpeedRatio(robotJointId\_t id, double ratio);

void setJointSpeedRatios(double ratios1, double ratios2, double ratios3);

void setJointPower(robotJointId\_t id, int power);

void setLEDColorRGB(int r, int g, int b);

void setLEDColor(char \*color);

void setMotorPowers(double p1, double p2, double p3);

void setMovementStateNB( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3);

void setMovementStateTime( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3,

double seconds);

void setMovementStateTimeNB( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3,

double seconds);

void setSpeed(double speed, double radius);

/\* MOVEMENT \*/

void accelJointAngleNB(robotJointId\_t id, double acceleration, double angle);

void accelJointTimeNB(robotJointId\_t id, double acceleration, double time);

void accelJointToVelocityNB(robotJointId\_t id, double acceleration, double speed);

void accelJointToMaxSpeedNB(robotJointId\_t id, double acceleration);

void driveAccelJointTimeNB(double radius, double acceleration,

double time);

void driveAccelToVelocityNB(double radius, double acceleration,

double velocity);

void driveAccelToMaxSpeedNB(double radius, double acceleration);

void driveAccelDistanceNB(double radius, double acceleration,

double distance);

void closeGripper();

void closeGripperNB();

void driveAngle(double angle);

void driveAngleNB(double angle);

void driveBackward(double angle);

void driveBackwardNB(double angle);

void driveDistance(double distance, double radius);

void driveDistanceNB(double distance, double radius);

void driveForeverNB();

void driveForward(double angle);

void driveForwardNB(double angle);

void driveTime(double seconds);

void driveTimeNB(double seconds);

void holdJoint(robotJointId\_t id);

void holdJoints();

void holdJointsAtExit();

int isMoving();

int isConnected();

void moveForeverNB();

void moveJoint(robotJointId\_t id, double angle);

void moveJointNB(robotJointId\_t id, double angle);

void moveJointForeverNB(robotJointId\_t id);

void moveJointTime(robotJointId\_t id, double time);

void moveJointTimeNB(robotJointId\_t id, double time);

void moveJointWait(robotJointId\_t id);

void move(double j1, double j2, double j3);

void moveNB(double j1, double j2, double j3);

void moveWait();

void moveJointTo(robotJointId\_t id, double angle);

void moveJointToNB(robotJointId\_t id, double angle);

void moveJointToByTrackPos(robotJointId\_t id, double angle);

void moveJointToByTrackPosNB(robotJointId\_t id, double angle);

void moveTime(double time);

void moveTimeNB(double time);

void moveTo(double angle1, double angle2, double angle3);

void moveToNB(double angle1, double angle2, double angle3);

void moveToByTrackPos(double angle1, double angle2, double angle3);

void moveToByTrackPosNB(double angle1, double angle2, double angle3);

void moveToZero();

void moveToZeroNB();

void openGripper(double angle);

void openGripperNB(double angle);

void relaxJoint(robotJointId\_t id);

void relaxJoints();

void resetToZero();

void resetToZeroNB();

void stop();

void stopOneJoint(robotJointId\_t id);

void turnLeft(double angle, double radius, double tracklength);

void turnLeftNB(double angle, double radius, double tracklength);

void turnRight(double angle, double radius, double tracklength);

void turnRightNB(double angle, double radius, double tracklength);

/\* MISC \*/

void blinkLED(double delay, int numBlinks);

void enableButtonCallback(void\* userdata, void (\*buttonCallback)(void\* data, int button, int buttonDown));

void disableButtonCallback();

void delaySeconds(int seconds);

void systemTime(double &time);

void recordAngleBegin(

robotJointId\_t id,

robotRecordData\_t &time,

robotRecordData\_t &angle,

double seconds);

void recordAngleEnd(robotJointId\_t id, int &num);

void recordAnglesBegin(

robotRecordData\_t &time,

robotRecordData\_t &angle1,

robotRecordData\_t &angle3,

double seconds);

void recordAnglesEnd(int &num);

void recordDistanceBegin(

robotRecordData\_t &time,

robotRecordData\_t &distance,

double radius,

double seconds);

void recordDistanceEnd(int &num);

void enableRecordDataShift();

void disableRecordDataShift();

void recordNoDataShift();

void recordDistanceOffset(double distance);

void recordDataShift();

/\*ROBOSIM COMPATIBILITY\*/

void drivexy(double x, double y, double radius, double trackwidth);

void drivexyNB(double x, double y, double radius, double trackwidth);

void drivexyTo(double x, double y, double radius, double trackwidth);

void drivexyToNB(double x, double y, double radius, double trackwidth);

void drivexyToExpr(double x0, double xf, int n, char \*expr, double radius, double trackwidth);

void drivexyToExprNB(double x0, double xf, int n, char \*expr, double radius, double trackwidth);

void drivexyToFunc(double x0, double xf, int n, double(\*func)(double x), double radius, double trackwidth);

void drivexyToFuncNB(double x0, double xf, int n, double(\*func)(double x), double radius, double trackwidth);

void drivexyWait(void);

void getxy(double &x, double &y);

void line(double x1, double y1, double z1, double x2, double y2, double z2, int linewidth, char \*color);

void point(double x, double y, double z, int pointsize, char \*color);

void recordxyBegin(robotRecordData\_t &x, robotRecordData\_t &y, double seconds, ...);

void recordxyEnd(int &num);

void text(double x, double y, double z, char \*text);

void traceOff(void);

void traceOn(void);

LinkbotImpl \*m;

static void \*g\_chlinkbot\_dlhandle;

static int g\_chlinkbot\_dlcount;

private:

double mMaxSpeed;

};

/\*Linkbot L\*/

class CLinkbotL {

public:

CLinkbotL(...);

~CLinkbotL();

int connect();

int connectWithSerialID(const char\* id);

void disconnect();

/\* GETTERS \*/

void getAccelerometerData(double &x, double &y, double &z);

void getBatteryVoltage(double &voltage);

void getJointAngle(robotJointId\_t id, double &angle);

void getJointAngles(double &angle1, double &angle2, double &angle3);

void getJointAngleInstant(robotJointId\_t id, double &angle);

void getJointAnglesInstant(double &angle1, double &angle2, double &angle3);

void getJointSafetyAngle(double &angle);

void getJointSafetyAngleTimeout(double &timeout);

void getJointSpeed(robotJointId\_t id, double &speed);

void getJointSpeedRatio(robotJointId\_t id, double &ratio);

void getJointSpeeds(double &speed1, double &speed2, double &speed3);

void getJointSpeedRatios(double &ratio1, double &ratio2, double &ratio3);

void getLEDColorRGB(int &r, int &g, int &b);

void getLEDColor(string\_t &color);

/\* SETTERS \*/

void setBuzzerFrequency(int frequency, double time);

void setBuzzerFrequencyOn(int frequency);

void setBuzzerFrequencyOff();

void setJointMovementStateNB(robotJointId\_t id, robotJointState\_t dir);

void setJointMovementStateTime(robotJointId\_t id, robotJointState\_t dir, double seconds);

void setJointSafetyAngle(double angle);

void setJointSafetyAngleTimeout(double timeout);

void setJointSpeed(robotJointId\_t id, double speed);

void setJointSpeeds(double speed1, double speed2, double speed3);

void setJointSpeedRatio(robotJointId\_t id, double ratio);

void setJointSpeedRatios(double ratios1, double ratios2, double ratios3);

void setJointPower(robotJointId\_t id, int power);

void setLEDColorRGB(int r, int g, int b);

void setLEDColor(char \*color);

void setMotorPowers(double p1, double p2, double p3);

void setMovementStateNB( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3);

void setMovementStateTime( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3,

double seconds);

void setMovementStateTimeNB( robotJointState\_t dir1,

robotJointState\_t dir2,

robotJointState\_t dir3,

double seconds);

/\* MOVEMENT \*/

void closeGripper();

void accelJointAngleNB(robotJointId\_t id, double acceleration, double angle);

void accelJointTimeNB(robotJointId\_t id, double acceleration, double time);

void accelJointToVelocityNB(robotJointId\_t id, double acceleration, double speed);

void accelJointToMaxSpeedNB(robotJointId\_t id, double acceleration);

void holdJoint(robotJointId\_t id);

void holdJoints();

void holdJointsAtExit();

int isMoving();

int isConnected();

void moveForeverNB();

void moveJoint(robotJointId\_t id, double angle);

void moveJointNB(robotJointId\_t id, double angle);

void moveJointForeverNB(robotJointId\_t id);

void moveJointTime(robotJointId\_t id, double time);

void moveJointTimeNB(robotJointId\_t id, double time);

void move(double j1, double j2, double j3);

void moveNB(double j1, double j2, double j3);

void moveWait();

void moveJointTo(robotJointId\_t id, double angle);

void moveJointToNB(robotJointId\_t id, double angle);

void moveJointToByTrackPos(robotJointId\_t id, double angle);

void moveJointToByTrackPosNB(robotJointId\_t id, double angle);

void moveJointWait(robotJointId\_t id);

void moveTime(double time);

void moveTimeNB(double time);

void moveTo(double angle1, double angle2, double angle3);

void moveToNB(double angle1, double angle2, double angle3);

void moveToByTrackPos(double angle1, double angle2, double angle3);

void moveToByTrackPosNB(double angle1, double angle2, double angle3);

void moveToZero();

void moveToZeroNB();

void openGripper(double angle);

void openGripperNB(double angle);

void relaxJoint(robotJointId\_t id);

void relaxJoints();

void resetToZero();

void resetToZeroNB();

void stop();

void stopOneJoint(robotJointId\_t id);

/\* MISC \*/

void blinkLED(double delay, int numBlinks);

void enableButtonCallback(void\* userdata, void (\*buttonCallback)(void\* data, int button, int buttonDown));

void disableButtonCallback();

void delaySeconds(int seconds);

void systemTime(double &time);

void recordAngleBegin(

robotJointId\_t id,

robotRecordData\_t &time,

robotRecordData\_t &angle,

double seconds);

void recordAngleEnd(robotJointId\_t id, int &num);

void recordAnglesBegin(

robotRecordData\_t &time,

robotRecordData\_t &angle1,

robotRecordData\_t &angle2,

double seconds);

void recordAnglesEnd(int &num);

void enableRecordDataShift();

void disableRecordDataShift();

void recordNoDataShift();

void recordDataShift();

/\*ROBOSIM COMPATIBILITY\*/

void getxy(double &x, double &y);

void line(double x1, double y1, double z1, double x2, double y2, double z2, int linewidth, char \*color);

void point(double x, double y, double z, int pointsize, char \*color);

void recordxyBegin(robotRecordData\_t &x, robotRecordData\_t &y, double seconds, ...);

void recordxyEnd(int &num);

void text(double x, double y, double z, char \*text);

void traceOff(void);

void traceOn(void);

LinkbotImpl \*m;

static void \*g\_chlinkbot\_dlhandle;

static int g\_chlinkbot\_dlcount;

private:

double mMaxSpeed;

};

/\*GroupI\*/

class CLinkbotIGroup {

public:

CLinkbotIGroup();

~CLinkbotIGroup();

void addRobot(CLinkbotI& robot);

void addRobots(array CLinkbotI robots[], ...);

void connect();

/\* MOVEMENT FUNCTIONS \*/

void closeGripper();

void closeGripperNB();

void driveAngle(double angle);

void driveAngleNB(double angle);

void driveBackward(double angle);

void driveBackwardNB(double angle);

void driveDistance(double distance, double radius);

void driveDistanceNB(double distance, double radius);

void driveForeverNB();

void driveForward(double angle);

void driveForwardNB(double angle);

void driveTime(double time);

void driveTimeNB(double time);

void holdJoint(robotJointId\_t id);

void holdJoints();

void holdJointsAtExit();

int isMoving();

int isConnected();

void move(double j1, double j2, double j3);

void moveForeverNB();

void moveJoint(robotJointId\_t id, double angle);

void moveJointNB(robotJointId\_t id, double angle);

void moveJointForeverNB(robotJointId\_t id);

void moveJointTime(robotJointId\_t id, double time);

void moveJointTimeNB(robotJointId\_t id, double time);

void moveJointTo(robotJointId\_t id, double angle);

void moveJointToNB(robotJointId\_t id, double angle);

void moveJointToByTrackPos(robotJointId\_t id, double angle);

void moveJointToByTrackPosNB(robotJointId\_t id, double angle);

void moveJointWait(robotJointId\_t id);

void moveNB(double j1, double j2, double j3);

void moveTime(double time);

void moveTimeNB(double time);

void moveTo(double angle1, double angle2, double angle3);

void moveToNB(double angle1, double angle2, double angle3);

void moveToByTrackPos(double angle1, double angle2, double angle3);

void moveToByTrackPosNB(double angle1, double angle2, double angle3);

void moveToZero();

void moveToZeroNB();

void moveWait();

void openGripper(double angle);

void openGripperNB(double angle);

void relaxJoint(robotJointId\_t id);

void relaxJoints();

void resetToZero();

void resetToZeroNB();

void stop();

void turnLeft(double angle, double radius, double tracklength);

void turnLeftNB(double angle, double radius, double tracklength);

void turnRight(double angle, double radius, double tracklength);

void turnRightNB(double angle, double radius, double tracklength);

/\* SET FUNCTIONS \*/

void setJointSpeed(robotJointId\_t id, double speed);

void setJointSpeeds(double speed1, double speed2, double speed3);

void setJointSpeedRatio(robotJointId\_t id, double ratio);

void setJointSpeedRatios(double ratios1, double ratios2, double ratios3);

void setSpeed(double speed, double radius);

void setLEDColorRGB(int r, int g, int b);

void setLEDColor(char \*color);

/\*ROBOSIM COMPATIBILITY\*/

void traceOff(void);

void traceOn(void);

private:

CLinkbotI \*\*\_robots;

int \_numRobots;

int argInt;

double argDouble;

int \_numAllocated;

int \_motionInProgress;

void \*\_thread;

char \*\*\_ID; //Array that stores the Id of the robots to add to the group

};

/\*GroupL\*/

class CLinkbotLGroup {

public:

CLinkbotLGroup();

~CLinkbotLGroup();

void addRobot(CLinkbotL& robot);

void connect();

/\* MOVEMENT FUNCTIONS \*/

void holdJoint(robotJointId\_t id);

void holdJoints();

void holdJointsAtExit();

int isMoving();

int isConnected();

void move(double j1, double j2, double j3);

void moveForeverNB();

void moveJoint(robotJointId\_t id, double angle);

void moveJointNB(robotJointId\_t id, double angle);

void moveJointForeverNB(robotJointId\_t id);

void moveJointTime(robotJointId\_t id, double time);

void moveJointTimeNB(robotJointId\_t id, double time);

void moveJointTo(robotJointId\_t id, double angle);

void moveJointToNB(robotJointId\_t id, double angle);

void moveJointToByTrackPos(robotJointId\_t id, double angle);

void moveJointToByTrackPosNB(robotJointId\_t id, double angle);

void moveJointWait(robotJointId\_t id);

void moveNB(double j1, double j2, double j3);

void moveTime(double time);

void moveTimeNB(double time);

void moveTo(double angle1, double angle2, double angle3);

void moveToNB(double angle1, double angle2, double angle3);

void moveToByTrackPos(double angle1, double angle2, double angle3);

void moveToByTrackPosNB(double angle1, double angle2, double angle3);

void moveToZero();

void moveToZeroNB();

void moveWait();

void relaxJoint(robotJointId\_t id);

void relaxJoints();

void resetToZero();

void resetToZeroNB();

void stop();

/\* SET FUNCTIONS \*/

void setJointSpeed(robotJointId\_t id, double speed);

void setJointSpeeds(double speed1, double speed2, double speed3);

void setJointSpeedRatio(robotJointId\_t id, double ratio);

void setJointSpeedRatios(double ratios1, double ratios2, double ratios3);

void setLEDColorRGB(int r, int g, int b);

void setLEDColor(char \*color);

/\*ROBOSIM COMPATIBILITY\*/

void traceOff(void);

void traceOn(void);

private:

CLinkbotL \*\*\_robots;

int \_numRobots;

int argInt;

double argDouble;

int \_numAllocated;

int \_motionInProgress;

void \*\_thread;

char \*\*\_ID; //Array that stores the Id of the robots to add to the group

};

void \*CLinkbotI::g\_chlinkbot\_dlhandle=NULL;

int CLinkbotI::g\_chlinkbot\_dlcount=0;

void \*CLinkbotL::g\_chlinkbot\_dlhandle=NULL;

int CLinkbotL::g\_chlinkbot\_dlcount=0;

#pragma package <chbarobo>

#pragma importf "chlinkboti.chf"

#pragma importf "chlinkbotl.chf"

#endif