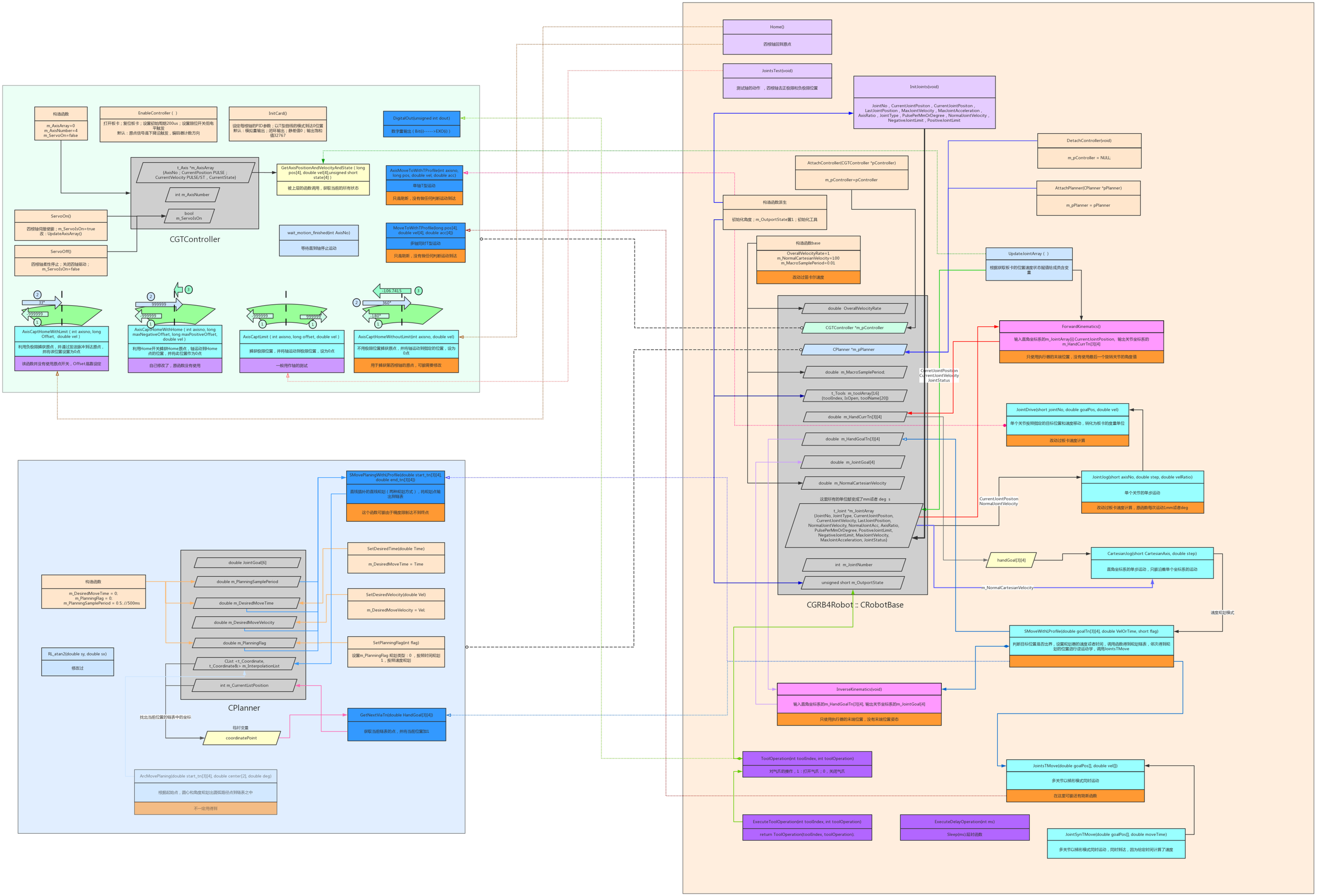
构造函数
m_AxisArray=0
m_AxisNumber=4
m_ServoOn=false
EnableController（ ）
打开板卡；复位板卡；设置初始周期200us；设置限位开关低电平触发
默认：原点信号是下降沿触发，编码器计数方向
InitCard()
设定每根轴的PID参数；以T型曲线的模式到达0位置
默认：模拟量输出；闭环输出；静差值0；输出饱和值32767
DigitalOut(unsigned int dout)
数字量输出（Bit6)----->EXO6)）
t_Axis *m_AxisArray
{AxisNo；CurrentPosition PULSE；CurrentVelocity PULSE/ST，CurrentState}
int m_AxisNumber
bool m_ServoIsOn
CGTController
GetAxisPositionAndVelocityAndState（long pos[4], double vel[4],unsigned short state[4]）
被上层的函数调用，获取当前的所有状态
AxisMoveToWithTProfile(int axisno, long pos, double vel, double acc)
单轴T型运动
只是刚新，没有做任何判断运动到达
ServoOn()
四根轴伺服使能；m_ServoIsOn=true
改：UpdateAxisArray()
ServoOff()
四根轴柔性停止；关闭四轴驱动；m_ServoIsOn=false
wait_motion_finished(int AxisNo)
等待直到轴停止运动
MoveToWithTProfile(long pos[4], double vel[4], double acc[4])
多轴同时T型运动
只是刚新，没有做任何判断运动到达
33°
999999
AxisCaptHomeWithLimit（int axisno, long Offset，double vel）
利用负极限捕获原点，并通过发送脉冲到达原点，并将该位置设置为0点
该函数并没有使用原点开关，Offset是靠设定
999999
-999999
AxisCaptHomeWithHome（int axisno, long maxNegativeOffset, long maxPositiveOffset, double vel）
利用Home开关捕获Home原点，轴运动到Home点的位置，并将此位置作为0点
自己修改了，原函数没有使用
-999999
999999
AxisCaptLimit（int axisno, long offset, double vel）
捕获极限位置，并将轴运动到极限位置，设为0点
一般用作轴的测试
-106.7415
360°
-180°
AxisCaptHomeWithoutLimit(int axisno, double vel)
不用极限位置捕获原点，并将轴运动到指定的位置，设为0点
用于捕获第四根轴的原点，可能需要修改
SMovePlaningWithLProfile(double start_tn[3][4], double end_tn[3][4])
直线插补的直线规划（两种规划方式），将规划点输出到链表
这个函数可能由于精度限制达不到终点
SetDesiredTime(double Time)
m_DesiredMoveTime = Time
SetDesiredVelocity(double Vel)
m_DesiredMoveVelocity = Vel;
SetPlanningFlag(int flag)
设置m_PlanningFlag 规划类型：0，按照时间规划 1，按照速度规划
GetNextViaTn(double HandGoal[3][4])
获取当前链表的点，并将当前位置加1
构造函数
m_DesiredMoveTime = 0;
m_PlanningFlag = 0;
m_PlanningSamplePeriod = 0.5; //500ms
RL_atan2(double sy, double sx)
修改过
double JointGoal[6]
double m_PlanningSamplePeriod
double m_DesiredMoveTime
double m_DesiredMoveVelocity
double m_PlanningFlag
CList <t_Coordinate, t_Coordinate&> m_InterpolationList
int m_CurrentListPosition
CPlanner
找出当前位置的链表中的坐标
临时变量
coordinatePoint
ArcMovePlaning(double start_tn[3][4], double center[2], double deg)
根据起始点，圆心和角度规划出圆弧路径点到链表之中
不一定用得到
Home()
四根轴回到原点
JointsTest(void)
测试轴的动作 ，四根轴去正极限和负极限位置
InitJoints(void)
JointNo，CurrentJointPositon，CurrentJointPositon，LastJointPosition，MaxJointVelocity，MaxJointAcceleration，AxisRatio，JointType，PulsePerMmOrDegree，NormalJointVelocity，NegativeJointLimit，PositiveJointLimit
DetachController(void)
m_pController = NULL;
AttachController(CGTController *pController)
m_pController=pController
AttachPlanner(CPlanner *pPlanner)
m_pPlanner = pPlanner
构造函数派生
初始化角度；m_OutportState置1；初始化工具
构造函数base
OverallVelocityRate=1
m_NormalCartesianVelocity=100
m_MacroSamplePeriod=0.01
改动过前卡尔速度
UpdateJointArray（ ）
根据获取板卡的位置速度状态赋值给成员含变量
double OverallVelocityRate
CGTController *m_pController
CPlanner *m_pPlanner
double m_MacroSamplePeriod;
t_Tools m_toolArray[16]
{toolIndex, IsOpen, toolName[20]}
double m_HandCurrTn[3][4]
double m_HandGoalTn[3][4]
double m_JointGoal[4]
double m_NormalCartesianVelocity
这里所有的单位都变成了mm或者 deg s
t_Joint *m_JointArray
{JointNo, JointType, CurrentJointPositon, CurrentJointVelocity, LastJointPosition, NormalJointVelocity, NormalJointAcc, AxisRatio, PulsePerMmOrDegree, PositiveJointLimit, NegativeJointLimit, MaxJointVelocity, MaxJointAcceleration, JointStatus}
int m_JointNumber
unsigned short m_OutportState
CGRB4Robot :: CRobotBase
CurretJointPositon
CurrentJointVelocity
JointStatus
ForwardKinematics()
输入直角坐标系的m_JointArray[i].CurrentJointPosition，输出关节坐标系的m_HandCurrTn[3][4]
只使用执行器的末端位置，没有使用最后一个旋转关节的角度值
JointDrive(short jointNo, double goalPos, double vel)
单个关节按照指定的目标位置和速度移动，转化为板卡的度量单位
改动过板卡速度计算
JointJog(short axisNo, double step, double velRatio)
单个关节的单步运动
改动过板卡速度计算，原函数每次运动1mm或者deg
CurrentJointPositon
NormalJointVelocity
handGoal[3][4]
CartesianJog(short CartesianAxis, double step)
直角坐标系的单步运动，只能沿着单个坐标系的运动
m_NormalCartesianVelocity
速度规划模式
SMoveWithLProfile(double goalTn[3][4], double VelOrTime, short flag)
判断目标位置是否出界，设置规划器的速度或者时间，调用函数得到规划链表，依次得到规划的位置进行逆运动学，调用JointsTMove
InverseKinematics(void)
输入直角坐标系的m_HandGoalTn[3][4]，输出关节坐标系的m_JointGoal[4]
只使用执行器的末端位置，没有末端位置姿态
ToolOperation(int toolIndex, int toolOperation)
对气爪的操作，1：打开气爪；0，关闭气爪
JointsTMove(double goalPos[], double vel[])
多关节以梯形模式同时运动
在这里可能还有期斯函数
ExecuteToolOperation(int toolIndex, int toolOperation)
return ToolOperation(toolIndex, toolOperation);
ExecuteDelayOperation(int ms)
Sleep(ms),延时函数
JointSynTMove(double goalPos[], double moveTime)
多关节以梯形模式同时运动，同时到达，因为给定时间计算了速度