机器人接口

**typedef** **struct** robjoint{

**double** angle[10];

**int** dof;

}robjoint;

**typedef** **struct** robpose{

**double** xyz[3];

**double** kps[3];

}robpose;

**typedef** **struct** speed{

**double** per[10];

**int** per\_flag;

**double** tcp;

**int** tcp\_flag;

**double** orl;

**int** orl\_flag;

**int** dof;

}speed;

**typedef** **struct** zone{

**int** zone\_flag;//0:不采用转弯区平滑过度，1：百分比，2：距离或圆周角

**double** zone\_size;

}zone;

**typedef** **struct** tool{

**int** robhold;

robpose tframe;

}tool;

**typedef** **struct** wobj{

**int** robhold;

**int** ufprog;

**int** ufmec;

robpose uframe;

robpose oframe;

}wobj;

**typedef** **enum** robdatatype{

*\_robjoint*,

*\_robpose*,

*\_speed*,

*\_zone*,

*\_tool*,

*\_wobj*

}robdatatype;

/\*system initialize

\* robot\_flag: select robot ,0: get from configure file;1:sia; 2:ur

\* path: configure file path

\*

\* return 0:ritgth; other: wrong

\* \*/

**extern** **int** **rob\_initialize**(**int** robot\_flag, **char**\* path);

/\*move robot by joint space

\* rjoint: target position(joint)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\* \*/

**extern** **void** **moveA**(robjoint\* rjoint, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj);

/\*move dual-robot by joint space

\* robot1---

\* rjoint1: target position(joint)

\* rspeed1: move speed

\* rzone1: area of turning

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* rjoint2: target position(joint)

\* rspeed2: move speed

\* rzone2: area of turning

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveA**(robjoint\* rjoint1,robjoint\* rjoint2, speed\* rspeed1,speed\* rspeed2, zone\* rzone1, zone\* rzone2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*move robot by joint space

\* rjoint: target position(joint)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\*\_index: robot index

\* \*/

**extern** **void** **multi\_moveA**(robjoint\* rjoint, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*move robot by joint space

\* rpose: target position(pose)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\* \*/

**extern** **void** **moveJ**(robpose\* rpose, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj);

/\*move dual-robot by joint space

\* robot1---

\* rpose1: target position(pose)

\* rspeed1: move speed

\* rzone1: area of turning

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* rpose2: target position(pose)

\* rspeed2: move speed

\* rzone2: area of turning

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveJ**(robpose\* rpose1,robpose\* rpose2, speed\* rspeed1,speed\* rspeed2, zone\* rzone1, zone\* rzone2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*move robot by joint space

\* rpose: target position(pose)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\*\_index: robot index

\* \*/

**extern** **void** **multi\_moveJ**(robpose\* rpose, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*move robot in a straight line

\* rpose: target position(pose)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\* \*/

**extern** **void** **moveL**(robpose\* rpose, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj);

/\*move dual-robot in a straight line

\* robot1---

\* rpose1: target position(pose)

\* rspeed1: move speed

\* rzone1: area of turning

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* rpose2: target position(pose)

\* rspeed2: move speed

\* rzone2: area of turning

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveL**(robpose\* rpose1,robpose\* rpose2, speed\* rspeed1,speed\* rspeed2, zone\* rzone1, zone\* rzone2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*move robot in a straight line

\* rpose: target position(pose)

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\*\_index: robot index

\* \*/

**extern** **void** **multi\_moveL**(robpose\* rpose, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*move robot in a circular arc

\* rpose: target position(pose)

\* rpose\_mid: middle position

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\* \*/

**extern** **void** **moveC**(robpose\* rpose, robpose\* rpose\_mid, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj);

/\*move dual-robot in a straight line

\* robot1---

\* rpose1: target position(pose)

\* rpose\_mid1: middle position

\* rspeed1: move speed

\* rzone1: area of turning

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* rpose2: target position(pose)

\* rpose\_mid2: middle position

\* rspeed2: move speed

\* rzone2: area of turning

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveC**(robpose\* rpose1,robpose\* rpose2, robpose\* rpose\_mid1, robpose\* rpose\_mid2, speed\* rspeed1,speed\* rspeed2, zone\* rzone1, zone\* rzone2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*move robot in a circular arc

\* rpose: target position(pose)

\* rpose\_mid: middle position

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\*\_index: robot index

\* \*/

**extern** **void** **multi\_moveC**(robpose\* rpose, robpose\* rpose\_mid, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*move robot in a spiral line

\* rpose: target position(pose)

\* rpose\_mid: middle position

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\* \*/

**extern** **void** **moveT**(robpose\* rpose, robpose\* rpose\_mid, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj);

/\*move dual-robot in a spiral line

\* robot1---

\* rpose1: target position(pose)

\* rpose\_mid1: middle position

\* rspeed1: move speed

\* rzone1: area of turning

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* rpose2: target position(pose)

\* rpose\_mid2: middle position

\* rspeed2: move speed

\* rzone2: area of turning

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveT**(robpose\* rpose1,robpose\* rpose2, robpose\* rpose\_mid1, robpose\* rpose\_mid2, speed\* rspeed1,speed\* rspeed2, zone\* rzone1, zone\* rzone2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*move robot in a spiral line

\* rpose: target position(pose)

\* rpose\_mid: middle position

\* rspeed: move speed

\* rzone: area of turning

\* rtool: tool

\* rwobj: coordinate system

\*\_index: robot index

\* \*/

**extern** **void** **multi\_moveT**(robpose\* rpose, robpose\* rpose\_mid, speed\* rspeed, zone\* rzone, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*B spline move in joint space

\* filename Necessary points sequence data filename

\* rspeed velocity (time)

\* rtool tool

\* rwobj coordinate system

\* \*/

**extern** **void** **moveAJBS**(**char**\* filename, speed\* rspeed, tool\* rtool, wobj\* rwobj);

/\*dual-robot B spline move in joint space

\* robot1---

\* filename1: data file

\* rspeed1: move speed

\* rtool1: tool

\* rwobj1: coordinate system

\* robot2---

\* filename2: data file

\* rspeed2: move speed

\* rtool2: tool

\* rwobj2: coordinate system

\*

\* \*/

**extern** **void** **dual\_moveAJBS**(**char**\* filename1, **char**\* filename2, speed\* rspeed1, speed\* rspeed2, tool\* rtool1, tool\* rtool2, wobj\* rwobj1, wobj\* rwobj2);

/\*B spline move in joint space

\* filename Necessary points sequence data filename

\* rspeed velocity (time)

\* rtool tool

\* rwobj coordinate system

\* \_index robot index

\* \*/

**extern** **void** **multi\_moveAJBS**(**char**\* filename, speed\* rspeed, tool\* rtool, wobj\* rwobj, **int** \_index);

/\*start move initialize or power

\* \*/

**extern** **void** **move\_start**();

/\*stop move or poweroff

\* \*/

**extern** **void** **move\_stop**();

/////////////////////////////////////////////////////////////////////////////////////////

/\*get joint data by name of the data

\*J: data name

\*rjoint: joint data

\*\_index: robot index

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getrobjoint2**(**char**\* J, robjoint\* rjoint, **int** \_index);

/\*get joint data by name of the data

\*J: data name

\*rjoint: joint data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getrobjoint**(**char**\* J, robjoint\* rjoint);

/\*get pose data by name of the data

\*P: data name

\*rpose: pose data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getrobpose**(**char**\* P, robpose\* rpose);

/\*get speed data by name of the data

\* S: data name

\* sp: speed data

\* \_index: robot index

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getspeed2**(**char**\* S, speed\* sp, **int** \_index);

/\*get speed data by name of the data

\* S: data name

\* sp: speed data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getspeed**(**char**\* S, speed\* sp);

/\*get zone data by name of the data

\* Z: data name

\* zo: zone data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getzone**(**char**\* Z, zone\* zo);

/\*get tool data by name of the data

\*T: data name

\*to: tool data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **gettool**(**char**\* T, tool\* to);

/\*get wobj data by name of the data

\* w: data name

\* wo: wobj data

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **getwobj**(**char**\* W, wobj\* wo);

/\*get robot current joint

\* joint: current joint data

\* \_index: robot index

\* \*/

**extern** **void** **getRobotJoint**(**double**\* joint, **int** \_index);

/\*get robot current pose

\* pospose: current pose data

\* \_index: robot index

\* \*/

**extern** **void** **getPosAndPose**(**double**\* pospose, **int** \_index);

/\*sleep

\* \_time time (s) resolution ratio ms

\* \*/

**extern** **void** **RSleep**(**double** \_time);

/\*sleep

\* \_time time (s) resolution ratio ms

\* \*/

**extern** **void** **RSleep2**(**double** \_time);

/\*set robot move the percentage of the acceleration and jerk

\* a: acceleration

\* aa: jerk

\* \*/

**extern** **void** **AccSet**(**int** a, **int** aa);

/\*set do

\* id: do index

\* flag: 0 or 1

\* \*/

**extern** **void** **SetDo**(**int** id,**int** flag);

/\*get do

\* id: do index

\* flag: return 0 or 1

\* \*/

**extern** **void** **GetDi**(**int** id,**int**\* flag);

/\*wait di

\* di :di

\* value: 0 or 1

\* \*/

**extern** **void** **WaitDi**(**int** di, **int** value);

/\*set ao

\* id: ao index

\* flag: ao value

\* \*/

**extern** **void** **SetAo**(**int** id,**double** flag);

/\*get ao

\* id: ao index

\* flag: return ao value

\* \*/

**extern** **void** **GetAi**(**int** id,**double**\* flag);

/\*get offset position

\* rpose: positon

\* x: offset x value

\* y: offset y value

\* z: offset z value

\* k: offset roll

\* p: offset pitch

\* s: offset yaw

\*

\* return offset position

\* \*/

**extern** robpose **Offs**(**const** robpose\* rpose, **double** x, **double** y, **double** z, **double** k, **double** p, **double** s);

//----------------------------------------------------------------------------------------communication interface--------------------------------------------------------------------------

/\*create socket server

\* ip: ip

\* port: port

\* sName: server name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketCreate**(**char**\* ip, **int** port, **char**\* sName);

/\*create socket client

\* ip: ip

\* port: port

\* sName: client name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **ClientCreate**(**char**\* ip, **int** port, **char**\* sName);

/\*close socket server or client

\* sName: server or client name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketClose**(**char**\* sName);

/\*sent byte

\* data: byte data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketSendByte**(**int** data, **char**\* sName);

/\*receive byte

\* data: return byte data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketRecvByte**(**int**\* data, **char**\* sName);

/\*sent string

\* data: string data

\* sName: socket name

\*

\* return >=0string lenth other:wrong

\* \*/

**extern** **int** **SocketSendString**(**char**\* data, **char**\* sName);

/\*receive string

\* data: retrun string data

\* sName: socket name

\*

\* return >=0string lenth other:wrong

\* \*/

**extern** **int** **SocketRecvString**(**char**\* data, **char**\* sName);

/\*sent double

\* data: double data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketSendDouble**(**double** data, **char**\* sName);

/\*receive double

\* data: retrun double data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketRecvDouble**(**double**\* data, **char**\* sName);

/\*sent int

\* data: int data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketSendInt**(**int** data, **char**\* sName);

/\*receive int

\* data: retrun int data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **SocketRecvInt**(**int**\* data, **char**\* sName);

/\*sent byte array

\* data: byte array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketSendByteArray**(**int**\* data,**int** n,**char**\* sName);

/\*receive byte array

\* data: return byte array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketRecvByteArray**(**int**\* data, **char**\* sName);

/\*sent double array

\* data: doube array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketSendDoubleArray**(**double**\* data, **int** n,**char**\* sName);

/\*receive double array

\* data: return double array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketRecvDoubleArray**(**double**\* data, **char**\* sName);

/\*sent int array

\* data: int array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketSendIntArray**(**int**\* data, **int** n, **char**\* sName);

/\*receive int array

\* data: return int array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **SocketRecvIntArray**(**int**\* data, **char**\* sName);

/\*UDP create socket server

\* ip: ip

\* port: port

\* sName: server name

\*

\* return 0:right other:wrong

\* \*/

**extern** **int** **UDPServerCreate**(**char**\* ip, **int** port, **char**\* sName);

/\*UDP create socket client

\* ip: ip

\* port: port

\* sName: client name

\*

\* return 0:right other:wrong

\* \*/

**extern** **int** **UDPClientCreate**(**char**\* ip, **int** port, **char**\* sName);

/\*UDP close socket server or client

\* sName: server or client name

\*

\* return 0:right other:wrong

\* \*/

**extern** **int** **UDPClose**(**char**\* sName);

/\*UDP sent byte

\* data: byte data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPSendByte**(**int** data, **char**\* sName);

/\*UDP receive byte

\* data: return byte data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPRecvByte**(**int**\* data, **char**\* sName);

/\*UDP sent string

\* data: string data

\* sName: socket name

\*

\* return >=0string lenth other:wrong

\* \*/

**extern** **int** **UDPSendString**(**char**\* data, **char**\* sName);

/\*UDP receive string

\* data: retrun string data

\* sName: socket name

\*

\* return >=0string lenth other:wrong

\* \*/

**extern** **int** **UDPRecvString**(**char**\* data, **char**\* sName);

/\*UDP sent double

\* data: double data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPSendDouble**(**double** data, **char**\* sName);

/\*UDP receive double

\* data: retrun double data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPRecvDouble**(**double**\* data, **char**\* sName);

/\*UDP sent int

\* data: int data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPSendInt**(**int** data, **char**\* sName);

/\*UDP receive int

\* data: retrun int data

\* sName: socket name

\*

\* return 1:right other:wrong

\* \*/

**extern** **int** **UDPRecvInt**(**int**\* data, **char**\* sName);

/\*UDP sent byte array

\* data: byte array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPSendByteArray**(**int**\* data,**int** n,**char**\* sName);

/\*UDP receive byte array

\* data: return byte array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPRecvByteArray**(**int**\* data, **char**\* sName);

/\*UDP sent double array

\* data: doube array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPSendDoubleArray**(**double**\* data, **int** n,**char**\* sName);

/\*UDP receive double array

\* data: return double array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPRecvDoubleArray**(**double**\* data, **char**\* sName);

/\*UDP sent int array

\* data: int array data

\* n: data number

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPSendIntArray**(**int**\* data, **int** n, **char**\* sName);

/\*UDP receive int array

\* data: return int array data

\* sName: socket name

\*

\* return >0: number of the data; other: wrong

\* \*/

**extern** **int** **UDPRecvIntArray**(**int**\* data, **char**\* sName);

//------------------------------------------------------------------------modbus----------------------------------------------------------------------------

**extern** **int** **modbusServerCreate\_tcp**(**char**\* ip, **int** port, **int** server\_id, **char**\* sName);

**extern** **int** **modbusServerCreate\_tcpip**(**char**\* ip, **int** port, **int** server\_id, **char**\* sName);

**extern** **int** **modbusServerCreate\_rtu**(**const** **char** \*device, **int** baud, **char** parity,

**int** data\_bit, **int** stop\_bit,**int** rtu\_mode, **int** server\_id,**char**\* sName);

**extern** **int** **modbusClientCreate\_tcp**(**char**\* ip, **int** port, **int** server\_id, **char**\* sName);

**extern** **int** **modbusClientCreate\_tcpip**(**char**\* ip, **int** port, **int** server\_id, **char**\* sName);

**extern** **int** **modbusClientCreate\_rtu**(**const** **char** \*device, **int** baud, **char** parity,

**int** data\_bit, **int** stop\_bit,**int** rtu\_mode, **int** server\_id,**char**\* sName);

**extern** **int** **modbusClose**(**char**\* sName);

**extern** **int** **modbusSendCoils**(**int** address, **int** nb, **int** \*in, **char**\* sName);

**extern** **int** **modbusRecvCoils**(**int** address, **int** nb, **int** \*out, **char**\* sName);

**extern** **int** **modbusRecvDiscretesInput**(**int** address, **int** nb, **int** \*out, **char**\* sName);

//nb:寄存器个数，也是in的维度，in的每个元素存放一个寄存器数据

**extern** **int** **modbusSendRegistersArray**(**int** address, **int** nb, **int** \*in, **char**\* sName);

//nb:寄存器个数，也是out的维度，out的每个元素存放一个寄存器数据

**extern** **int** **modbusRecvRegistersArray**(**int** address, **int** nb, **int** \*out, **char**\* sName);

//nb:寄存器个数，也是out的维度，out的每个元素存放一个寄存器数据

**extern** **int** **modbusRecvInputRegistersArray**(**int** address, **int** nb, **int** \*out, **char**\* sName);

//n:数组维度

**extern** **int** **modbusSendRegistersIntArray**(**int** address, **int** n, **int** \*in, **char**\* sName);

//n:数组维度

**extern** **int** **modbusRecvRegistersIntArray**(**int** address, **int** n, **int** \*out, **char**\* sName);

//n:数组维度

**extern** **int** **modbusRecvInputRegistersIntArray**(**int** address, **int** n, **int** \*out, **char**\* sName);

//n:数组维度

**extern** **int** **modbusSendRegistersFloatArray**(**int** address, **int** n, **float** \*in, **char**\* sName, **char**\* format);

//n:数组维度

**extern** **int** **modbusRecvRegistersFloatArray**(**int** address, **int** n, **float** \*out, **char**\* sName, **char**\* format);

//n:数组维度

**extern** **int** **modbusRecvInputRegistersFloatArray**(**int** address, **int** n, **float** \*out, **char**\* sName, **char**\* format);

//n:数组维度

**extern** **int** **modbusSendRegistersDoubleArray**(**int** address, **int** n, **double** \*in, **char**\* sName);

//n:数组维度

**extern** **int** **modbusRecvRegistersDoubleArray**(**int** address, **int** n, **double** \*out, **char**\* sName);

//n:数组维度

**extern** **int** **modbusRecvInputRegistersDoubleArray**(**int** address, **int** n, **double** \*out, **char**\* sName);

//----------------------------------------------------------------------------------------thread interface--------------------------------------------------------------------------

/\*create thread

\* fun: thread execute function

\* arg: execute function

\* name: thread name

\* detached\_flag: thread attribute,0:PTHREAD\_CREATE\_JOINABLE other:PTHREAD\_CREATE\_DETACHED

\*

\* return 0:right; other: wrong

\* \*/

**extern** **int** **ThreadCreat**(**void**\* (\*fun)(**void**\*), **void**\* arg, **char**\* name, **int** detached\_flag);

/\*thread data free

\* name: thread name

\* \*/

**extern** **int** **ThreadDataFree**(**char**\* name);

/\*thread join (It works when thread attribute is PTHREAD\_CREATE\_JOINABLE)

\* name thread name

\* return 0:right; other: wrong

\* \*/

**extern** **int** **ThreadWait**(**char**\* name);

/////////////////////////////////////////////////////////////////////////////////////////

/\*close grip

\*pos: \_ROBOTIQ:夹具夹紧程度，从0.0~1.0之间取值，0.0完全伸展开，1.0处于夹紧状态；

\*

\* return 0:right other:wrong

\* \*/

**extern** **int** **grip\_position**(**double** pos);

//=========================custom interface===============================

/\*constom interface\*/

**int** **custom\_interface**(**const** **char** \* \_\_format);

**int** **custom\_interface1**(**void**\* arg,**int** mode);

**int** **custom\_interface2**(**int** argc, **char** \*argv[]);