主管道

传入主管道消息队列句柄和.so句柄

struct rt\_messagequeue \*p\_mq;

struct rt\_dlmodule \*p\_lib;

分管道

main

1. 初始化组件消息队列

res=rt\_mq\_init(&COMA\_pipe,"COMA\_pipe",COMA\_msg\_pool,sizeof(GMS\_STRU),sizeof(COMA\_msg\_pool),RT\_IPC\_FLAG\_FIFO); //建立COMA管道

1. 将 主管道消息队列句柄赋值到 CP\_NAME\_ctl

 COMA\_ctl.p\_mainpipe=((CP\_ENTRY\_ARGS\_STRU \*)p)->p\_mq;

* 1. typedef struct
  2. {
  3. unsigned char is\_registered;
  4. struct rt\_messagequeue \*p\_mainpipe;
  5. struct rt\_dlmodule \*p\_lib;
  6. unsigned char id;
  7. unsigned char cur\_n;
  8. } \_\_attribute\_\_((packed)) CP\_CTL\_STRU;

1. 初始化组件线程，并运行CP\_NAME\_register\_entry
2. 给CP\_NAME\_reg.p\_pipe赋值。本组件的消息队列句柄
3. rt\_sem\_init(&sem\_reg,"CP\_NAME\_reg",0,RT\_IPC\_FLAG\_FIFO);
4. //初始化用于消息同步的信号量，接受者在完成消息处理后，必要时需要对信号量进行释放
5. //向多维消息体中装入消息、消息协议栈数据帧填充函数
   1. mb\_make\_dmgms(
   2. &tmp\_dmgms,//数据帧结构体数组地址
   3. 0,//数组下标
   4. &sem\_reg,//信号量句柄赋值
   5. MAINPIPE\_CMD\_CPREG,//命令类型
   6. MB\_STATN\_MAIN,//目的组件消息mac编号（对应消息队列句柄）
   7. MB\_STATN\_CP\_NAME,//源组件消息队列编号
   8. &CP\_NAME\_reg,//数据区指针
   9. sizeof(CP\_REG\_STRU),//数据长度
   10. &resp\_reg); //返回数据存放的地址
6. rt\_mq\_send(CP\_NAME\_ctl.p\_mainpipe,&tmp\_dmgms,sizeof(DM\_GMS\_STRU));//向主管道发送注册命令
7. rt\_sem\_take//等待接收者释放信号量，超时将造成组件注册失败

主管道

1. MP\_TRANS\_TABLE\_STRU Trs\_Tbl[TRS\_TBL\_SIZE];// 20
2. CP\_REG\_STRU \*p\_reg=(CP\_REG\_STRU \*)tmp\_dmgms.dm\_gms[i].d\_p; //取出消息中的注册信息,cpid与管道指针
3. Trs\_Tbl[n].statn=p\_reg->cpid; //放入转发表
4. Trs\_Tbl[n].p\_mq=p\_reg->p\_pipe; n++;
5. tmp\_dmgms.dm\_gms[i].r\_p->r\_src=MB\_STATN\_MAIN;//返回数据来源
6. tmp\_dmgms.dm\_gms[i].r\_p->r\_res=MB\_RESP\_SUC; //返回数据结果，注册成功
7. if(tmp\_dmgms.dm\_gms[i].psem) rt\_sem\_release(tmp\_dmgms.dm\_gms[i].psem); //如果有信号量需要释放，则释放