OTA 升级协议说明

帧类型说明

1、获取设备 ID （通信类型 0） ；获取设备的 ID 和 64 位 MCU 唯一标识符

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	0	bit15~8:用来标识主机 CAN_ID	目标电机 CAN_ID	0

应答帧：

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	0	目标电机 CAN_ID	0XFE	64 位 MCU 唯一标识符

2、升级启动帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	11	bit15~8:用来标识主机 CAN_ID	目标电机 CAN_ID	64 位 MCU 唯一标识符

反馈帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	11	Bit23~16: 0x00: 发送成功 0xF0：发送失败 bit15~8:目标电机 CAN_ID	主机 CAN_ID	0

3、bin 文件解析，升级信息帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	12	bit15~8:用来标识主机 CAN_ID	目标电机 CAN_ID	Byte0~Byte3 数据包字节数 binsize Byte4~Byte7 数据包字数 PackNumber 低字节在前

反馈帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	12	Bit23~16: 0x00: 发送成功 0xF0：发送失败 bit15~8:目标电机 CAN_ID	主机 CAN_ID	0

4、升级帧

4.1 过程帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	13	bit15~8:升级包当前位置	目标电机 CAN_ID	数据包数据

反馈帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	11	Bit23~16: 0x00: 发送成功 0xF0：发送失败 bit15~8:目标电机 CAN_ID	主机 CAN_ID	0

4.2 结束帧

29 位 ID			8Byte 数据区
Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
14	bit15~8:升级包当前位置	目标电机 CAN_ID	数据包数据

反馈帧

数据域	29 位 ID			8Byte 数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	14	Bit23~16: 0x00: 发送成功 0xF0：发送失败 bit15~8:目标电机 CAN_ID	主机 CAN_ID	Byte0~Byte1 升级包当前位置

详细流程说明（基于 QT）

下方为定义参数

```
enum canComMode{
    CANCOM_ANNOUNCE_DEVID = 0,//通告设备 ID
    CANCOM_MOTOR_CTRL,          //MOTOR-电机控制
    CANCOM_MOTOR_FEEDBACK,      //MOTOR-电机反馈
    CANCOM_MOTOR_IN,            //MOTOR-进入电机模式
    CANCOM_MOTOR_RESET,         //MOTOR-复位模式
    CANCOM_MOTOR_CALI,          //MOTOR-高速编码器标定
    CANCOM_MOTOR_ZERO,          //MOTOR-设置机械零位
    CANCOM_MOTOR_ID,            //MOTOR-设置 ID
    CANCOM_PARA_WRITE,          //参数-写入
    CANCOM_PARA_READ,           //参数-读取
    CANCOM_PARA_UPDATE,         //参数-更新上传
```

```
    CANCOM_OTA_START,           //OTA-启动
    CANCOM_OTA_INFO,            //OTA-升级文件描述
    CANCOM_OTA_ING,             //OTA-升级中
    CANCOM_OTA_END,             //OTA-升级完成
    CANCOM_CALI_ING,            //编码器标定中
    CANCOM_CALI_RST,            //编码器标定结果
    CANCOM_SDO_READ,            //sdo
    CANCOM_SDO_WRITE,           //sdo
    CANCOM_PARA_STR_INFO,       //参数-字符串信息
    CANCOM_MOTOR_BRAKE,         //MOTOR-进入刹车模式
    CANCOM_FAULT_WARN,          //故障和警告信息
    CANCOM_MODE_TOTAL,
};

struct exCanIdInfo{
        quint32 id:8;
        quint32 data:16;
        enum canComMode mode:5;
        quint32 res:3;
};

struct  canPack{
    Struct  exCanIdInfo  exId;
    quint8  len;
    quint8  data[8];
};

struct canPack pack;
memset(&pack,0,sizeof (struct canPack));
pack.len = 8;
pack.exId.id = devCanId;//赋值电机 canid
quint32 addr;
QString FilePath;
QByteArray BinData;
```

下面为具体流程

1、识别 bin 文件路径，读取文件信息

```
void MainWindow::on_pushButtonOpenFile_clicked()
{
    BinData.clear();//清空
    FilePath = QFileDialog::getOpenFileName(this,tr("Open Binary File"),
                                            ".",
                                            tr("Binary File(*.bin *.hex)"));
```

```
    if(FilePath.length())
    {
        QFile file(FilePath);
        if(file.open(QIODevice::ReadOnly))
        {
            BinData = file.readAll();//read file
            file.close();
        }
    }
}
```

2、发送升级启动帧

```
    pack.exId.data = 0xFD; //主机 id
    pack.exId.mode = CANCOM_OTA_START;
    memcpy(&(pack.data[0]),&(mcuBuf.id[mcuBuf.usePos]),8);//数据位是 mcuid，通过类型 0 获得
    txdPack(&pack);
    break;
```

如接收到反馈帧可继续下一步

3、发送升级信息帧，包含包大小的信息

```
binSize = BinData.size();
if(binSize==0)
{
   //该路径表示未识别到文件
}
else if(binSize > 0X80000)
{
    //该路径表示识别的文件大小过大，非目标文件
}
else
{
    PackNumber = binSize / 8;
    if(binSize % 8)
    {
        PackNumber += 1;
    }
    PackCnt = 0;
    memcpy(&(pack.data[0]),&binSize,4);
    memcpy(&(pack.data[4]),&PackNumber,4);
    pack.exId.data = 0xFD; //主机 id
    pack.exId.mode = CANCOM_OTA_INFO;
    txdPack(&pack);
}
break;
```

如接收到反馈可继续下一步

5、循环发送，发送一帧反馈一帧，接收到反馈后再发送下一帧
先发送一个过程帧

```
pack.exId.data = PackCnt;
addr = PackCnt*8;
for(uint8_t i=0;i<8;i++,addr++)
{
   if(addr<binSize)      pack.data[i]=BinData[PackCnt*8+i];
   else                  pack.data[i]=0XFF;
}
txdPack(&pack);
```

等到反馈，接收中断后就可以循环下方函数

```
if(rxFrame.exId.data == devCanId) //反馈帧错误位置 0，表示发送成功，顺序号加 1
{
    PackCnt++;
}
else if(rxFrame.exId.data == ((quint16)devCanId|0X0F00))   //反馈帧错误位为 0x0F，表示发送失败，重新发送该帧，断点续接
{
    memcpy(&PackCnt,rxFrame.data,2);
}
if(PackCnt >= PackNumber)   //结束帧，该帧表示升级完成
{
    pack.exId.data = 0;
    pack.exId.mode = CANCOM_OTA_END;
    memcpy(&(pack.data[0]),&PackNumber,4);
    txdPack(&pack);
    break;
}
else   //过程帧
{
    pack.exId.data = PackCnt;
    addr = PackCnt*8;
    for(uint8_t i=0;i<8;i++,addr++)
    {
       if(addr<binSize)      pack.data[i]=BinData[PackCnt*8+i];
       else                  pack.data[i]=0XFF;
    }
       txdPack(&pack);
       QThread::msleep(1);     //延时 1ms
       break;
}
```

可以加一个升级延时判定，如果发现长时间没有反馈帧，那就表示通信出现问题，可以电机

重新上电再次升级