## class MainWindow : public QmainWindow

public:

MainWindow(QWidget \*parent = 0);

~*MainWindow*();

enum{

MainWidgetWidth = 650,

MainWidgetHight = 310,

MidWidth = 480,

LeftWidth = 140

};

void *closeEvent*(QCloseEvent \*); //重载该事件，让串口窗口也关闭

signals:

void SendReadyData(HandleToComData); //向串口界面发送封装好的数据

void ComToRun(ComToRunData); //串口传回的数据，发给运动设定

public slots:

void GetPowerData(PowerToHandleData powerdata); //从电源设定界面接受到的数据

void GetRunData(RunToHandleData rundata); //从运动设定界面接受到的数据

void GetComRecvData(QByteArray comdata); //串口接收到的数据

void GetConData(QByteArray condata); //获取点位控制传来的数据

void HideAllButton();

void ShowAllButton();

## class PowerWidget : public QWidget

public:

explicit PowerWidget(QWidget \*parent = 0);

void HideButton();

void ShowButton();

signals:

void SendPowerData(PowerToHandleData); //向主界面发送数据

public slots:

## class RunWidget : public QWidget

public:

explicit RunWidget(QWidget \*parent = 0);

void HideButton();

void ShowButton();

signals:

void SendRunData(RunToHandleData); //向主界面发送数据

public slots:

void GetRecvData(ComToRunData); //从主界面接受串口处理号的数据

## class ControlWidget : public QWidget

public:

explicit ControlWidget(QWidget \*parent = 0);

void HideButton();

void ShowButton();

signals:

void ConToCom(QByteArray);

## class ComPortWidget : public QWidget

public:

explicit ComPortWidget(QWidget \*parent = 0);

signals:

void EmitRecvData(QByteArray); //将接收到的数据发送给主界面

void HideAllButton();

void ShowAllButton();

public slots:

void GetSendData(HandleToComData); //从主界面获取需要发送的数据

void MySendData(int DataFlat = SENDDATA); //用来发送信息的槽函数

void MyRecvData(); //接受信息的槽函数

## class DataHandle

public:

DataHandle();

HandleToComData PowerDataHandle(PowerToHandleData &powerdata); //封装电源设定数据

HandleToComData RunDataHandle(RunToHandleData &rundata); //封装运动设定数据

HandleToComData ConDataHandle(QByteArray &condata); //封装点位控制数据

ComToRunData AnalysisComData(QByteArray comdata); //解析串口发回的数据

## struct PowerToHandleData //电源设定数据

{

QString PulseWidethString; //脉冲宽度

QString PulseRateString; //脉冲频率

QString PowerString; //幅值电压

};

## struct RunToHandleData //运动设定数据

{

enum{

ShowPositon = 0,

SetPositon = 1,

SetSpeed = 2 //展示，设置位置和设置速度

};

QString xData[3]; //x位置展示、设定和速度设定

QString yData[3]; //y位置展示、设定和速度设定

QString zData[3]; //z位置展示、设定和速度设定

QString pztData[2]; //pzt位置展示、设定

};

## struct ComToRunData

{

enum{

XIndex = 0,

YIndex = 1,

ZIndex = 2,

PZTIndex = 3

};

bool isOk = false; //

bool isPlus[3]; //是否为正方向

double PositionData[4]; //接收到的信息

};

## struct HandleToComData //传给串口处理好的数据

{

HandleToComData();

HandleToComData(const HandleToComData &other);

void clear();

enum{

POWERDATA,

RUNDATA,

CONDATA,

DATALEN = 32

};

char flag;

QByteArray data;

};

## class PromptMessage : public QDialog

//串口发送信息时的提示窗口

## CRCHandle

unsigned short CRCHandle(QByteArray &data,unsigned int len)

{

unsigned short CrcFlag = 0xFFFF; //初值与下位机协定好

unsigned int m\_dataIndex = 0;

while(len--)

{

for(unsigned char i=0x80; i != 0; i >>= 1) //8次循环

{

if((CrcFlag & 0x8000) != 0) //上一位存在余式，CRC乘以2再求CRC

{

CrcFlag <<= 1;

CrcFlag ^= 0x1021; //CRC生成多项式 0x1021，欧洲推荐标准

}

else //上一位不存在余式

{

CrcFlag <<= 1;

}

if((data[m\_dataIndex]&i) != 0)

CrcFlag ^= 0x1021; //最后加上本位的异或值，产生本位CRC

}

m\_dataIndex++;

}

return CrcFlag; //Crc校验码

}

## ByteArrayClear //QByteArray清0

void ByteArrayClear(QByteArray &data)

{

for(auto &tmp : data)

tmp = 0x00;

}