# DSP接口使用方法

提供CDspController和CDsp用于外部与Dsp进行交互，其中CDspController用于与CDsp相关的底层操作，如启动，退出，收发原始数据帧和判断Dsp运行状态（是否启动成功），而CDsp类是基于CDspController的上层封装，使用户不需要关心Dsp的帧的收发过程，只需要按照DspArmProtocol.h文件中描述的协议进行数据的发送和读取，这样就将Dsp的底层操作和上层数据操作隔离开来。

另外，由于现在Dsp端的收发采用了异步的方式，所以CDsp也采用了异步操作的方式来提高效率，其中，对于每种格式数据的发送都提供相应的函数，因而用户可以用多个线程同时进行多个数据发送操作，但对于数据接收则统一用一个线程接收，接收的数据会存储到另外一个CDspDataSet类的实例中（该实例由用户创建），而用户则统一通过CDspDataSet中提供的接口进行读取。而且在CDspDataSet中对数据进行了加锁处理，保证了数据的安全性。

将Dsp初始化和释放操作放到DspController中并提供给用户，而不是统一封装到CDsp中的原因是：虽然所有的数据操作都可以由CDsp进行，但基于项目的考虑，Dsp的初始化和释放在程序的整个生命周期中只进行一次（要由用户保证），而CDsp并不是单例类，更加难以保证多线程中对Dsp资源初始化和释放的安全。

## CDspController接口使用

CDspController是个单例类，因此这样来对Dsp进行初始化：

CDspController::GetInstance()->DspInit();

释放Dsp的时候，务必调用DspDelete()方法，如果只是delete CDspController实例是不会释放Dsp的，即：

CDspController::GetInstance()->DspDelete();

一般在程序开始对Dsp进行初始化，在程序结束对Dsp进行释放。

## CDsp接口使用

对于如下这一组发送函数，只需在外部构造好相应的数据，然后调用相应的函数即可

bool SendSmv(const SmvData &smvData);

bool SetConfig(const Config &dspConfig);

bool ResetCache();

bool SendSmvGen(const SmvGenData &smvGenData);

bool SendMsgTime(const MSGTIME &msgTime);

bool GetDspInfo();

而对于接收，提供了两个函数：

bool StartRead();

void StopRead();

通过调用StartRead()启动接收线程，调用StopRead()停止接收线程，接收到的数据会存放到一个CDspDataSet实例中，这个实例由用户创建并在CDsp初始化的时候作为引用传入。用户通过调用CDspDataSet中的方法读取相应的数据。

## 交互方法

Arm与Dsp的交互采用DspArmProtocol.h中定义的多种结构体来进行，具体来说，对于数据数据交互用户只需按照协议中定义的数据结构构造和存储数据，然后调用相应接口进行交互即可

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 操作 | ARM端发送数据 | ARM端发送次数 | DSP端回送数据 | DSP回送次数 | 描述 |
| SendSmv | SmvData | 多次 | SMV\_DATA\_CALC\_RT | 1 | 发送采样值，由DSP计算谐波等数据 |
| ResetCache |  | 1 |  |  | 将DSP采样值缓冲区清空 |
| SetConfig | Config | 1 | Error | 1 | 发送设置信息，设置相关计算参数，由DSP返回设置结果，设置成功Error.nErrno=0，失败为其他错误码 |
| SendSmvGen | SmvGenData | 1 | SMV\_GEN\_DATA\_CALC\_RT | 1 | 发送波形生成参数，由DSP计算得出生成的波形数据 |
| SendMsgTime | MsgTimeData | 多次 | float [11] | 1 | 发送报文时间数据，由DSP计算得到报文离散度 |
| GetDspInfo |  | 1 | DspInfo | 1 | 返回DSP相关信息 |