# SecsCore库

概述：

如何实现hsms通信协议，完成设备与主机之间的通信

|  |  |
| --- | --- |
| 类 | 描述 |
| SecsGem类 | 实现HSMS通信 |
| SecsMessage类 | hsms通信中的DataMessage构建 |
| Helper类 | 将SecsMessage实例与SML字符串互相转换 |

|  |  |
| --- | --- |
| 枚举 | 描述 |
| ConnectionState | Hsms的连接状态 |

SecsGem类：

|  |  |
| --- | --- |
| 属性 | 描述 |
| **State** | 通信状态 |
| **LinkTestInterval** | LinkTest(检测网络连接是否断开)间隔时间 |
| **T3** | T3(Reply Timeout)定时间隔 |
| **T5** | T5(Connect Separation Time)定时间隔 |
| **T6** | T6(Control Transactions Timeout )定时间隔 |
| **T7** | T7(NOT SELECTED Timeout)定时间隔 |
| **T8** | T8(Network Intercharacter Timeout)定时间隔 |
| **DeviceId** | 设备ID |
| **LinkTestEnable** | 获取或设置LinkTest定时器是否启用 |

|  |
| --- |
| 构造函数： |
| ***SecsGem(IPAddress,int,bool,Action<SeceMessage,Action<SecsMessage>>,SecsTracer,int)*** |
| 使用指定的IP地址，端口号，通信模式，处理主消息的委托方法，SecsMessage消息调试对象，内部调用的sorket实例的缓冲区大小，初始化SecsGem类的新实例 |

SecsGem(

IPAddress *ip*,

int *port*,

bool *isActive*,

Action<SecsMessage,Action<SecsMessage>> *primaryMsgHandler*,

SecsTracer *tracer*,

int *receiveBufferSize*

)

|  |  |
| --- | --- |
| ip | IP地址，当为主动模式的时候，为host端IP地址，为被动模式时，为equipment端IP地址 |
| port | 端口号，当为主动模式的时候，为host端端口号，为被动模式时，为equipment端端口号 |
| isActive | 主动或被动模式，设备的连接模式，为主动请求或被动响应 |
| primaryMsgHandler | 处理PrimartMessage的委托实例，每个PrimaryMessage要调用一次Reply方法，如不使用，则构造函数默认为Reply委托实例为空 |
| tracer | 信息描述对象，对传入的SecsMessage对象进行可视化描述 |
| receiveBufferSize | 构造函数内部的Socket接收区缓存大小 |

Example：

SecsGem \_secsGem;

\_secsGem = new SecsGem(IPAddress.Parse("192.169.0.49"), 5000,

true,

null,

\_logform.Logger, 0);

|  |  |
| --- | --- |
| 方法 | 描述 |
| Send(SecsMessage) | 发送SECS消息到设备 |
| BeginSend(SecsMessage,AsyncCallback, object) | 将SECS消息异步发送至设备 |
| EndSend(IAsyncResult) | 结束异步发送 |
| Dispose() | 重置所有配置 |

public SecsMessage Send(

SecsMessage *msg*

)

发送SECS消息到设备，如果msg.ReplyExpected为true，则设备的回复msg;否则为null

public IAsyncResult BeginSend(

SecsMessage *msg*,

AsyncCallback *callback*,

object *state*

)

将SECS消息异步发送至设备，如果msg.ReplyExpected为true，则引用异步发送的IAsyncResult;否则为null

public SecsMessage EndSend(

IAsyncResult *asyncResult*

)

结束异步发送，如果参数asyncResult是引用异步发送的IAsyncResult，则设备的回复消息，否则为空。

public void Dispose()

清除连接信息，并将所有配置重置

Example：

SecsGem \_secsGem;

var msg = txtSendPrimary.Text.ToSecsMessage();

\_secsGem.BeginSend(msg, ar => {

try {

var reply = \_secsGem.EndSend(ar);

} catch (SecsException ex) {

}

}, null);

if (\_secsGem != null) {

\_secsGem.Dispose();

\_secsGem = null;

}

SecsMessage类：

|  |  |
| --- | --- |
| 属性 | 描述 |
| S | message stream number |
| F | message function number |
| Name | Stream function name |
| RawDatas |  |
| ReplyExpected | expect reply message |
| SecsItem | the root item of message |

|  |  |
| --- | --- |
| 构造函数 | 描述 |
| SecsMessage() | 使用指定的S字节，F字节，函数名称，是否期待响应，Item实例，初始化SecsMessage类的新实例 |
| SecsMessage(byte,byte,string,bool,Item) |
| SecsMessage(byte,byte,string,Item) |
| SecsMessage(byte,byte,string) |
| SecsMessage(byte,byte,string,bool,byte[],ref int) |

SecsMessage(

byte s,

byte f,

string name,

bool replyExpected,

Item item

)

SecsMessage(byte s, byte f, string name, Item item) : this(s, f, name, true, item) { }

SecsMessage(byte s, byte f, string name) : this(s, f, name, null) { }

SecsMessage(byte s, byte f, bool replyExpected, byte[] itemBytes, ref int index) :

this(s, f, "Unknown", replyExpected, Decode(itemBytes, ref index)) { }

Helper类：

|  |  |
| --- | --- |
| 方法 | 描述 |
| ToSML(this SecsMessage) | 将SecsMessage对象转换为string类型字符串 |
| ToSecsMessage(this string) | 将string类型字符串转换为SecsMessage对象 |

public static string ToSML(

this SecsMessage msg

)

public static SecsMessage ToSecsMessage(

this string str

)