编号：CC042020041801
密级：内部
版本：V1.0.2.0

CC04 动态链接库使用说明

（版本:1.0.2.0）

编写：王瑞恩

审核：

二〇二〇年五月二十日

一、动态库文件：

1、CC04Sdk.dll 动态库主文件（32 位）；

2、CC04Sdk.key 控制器授权文件；

＊ 调用时必须把两个文件同时复制到与项目主程序同一目录下，否则动态库将调用失败。

二、函数说明：

1、函数名：OpenCom

功能说明：初始化通信串口
函数原型：int __stdcall OpenCom(DWORD dwPort)
输入参数：dwPort：通信串口号（1～1024）
输出参数：无
返 回 值：0：初始化失败
1：初始化成功

2、函数名：CloseCom

功能说明：关闭通信串口
函数原型：void __stdcall CloseCom(void)
输入参数：无
输出参数：无
返 回 值：无

3、函数名：DllBind

功能说明：绑定控制器授权文件
函数原型：int __stdcall DllBind(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：1：绑定成功
-1：通信串口未初始化
-2：制器地址无效
-3：控制器通信失败
-4：未找到授权文件
-5：授权文件无效
-6：授权文件与控制器不匹配
-7：系统时间异常

-8：授权文件已过期
-99：未知错误

* 初始化通信串口后，必须先调用此函数，绑定成功后，其他函数才能使用。

4、函数名：IsComOpen

功能说明：检查通信串口是否初始化
函数原型：int __stdcall IsComOpen(void)
输入参数：无
输出参数：无
返 回 值：0：未初始化
1：已初始化

5、函数名：SetDefault

功能说明：控制器恢复出厂设置
函数原型：int __stdcall SetDefault(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（0-16）[0 为广播模式，慎用！]
输出参数：无
返 回 值：0：调用失败
1：调用成功

* nSlaver 参数为 0 时，函数执行为广播模式，既连接在同一总线上的所有控制器将同时被恢复出厂设置；执行此函数成功后，被恢复出厂设置的控制器地址同时被恢复为 1。

6、函数名：GetSerial

功能说明：获取控制器序列号
函数原型：int __stdcall GetSerial(unsigned int nSlaver,
char** szID)
输入参数：nSlaver：控制器地址（1-16）
输出参数：szID：控制器序列号
返 回 值：0：调用失败
1：调用成功

7、函数名：GetSwichState

功能说明：获取控制器开关状态
函数原型：int __stdcall GetSwichState(unsigned int nSlaver,
char** szSwichState)
输入参数：nSlaver：控制器地址（1-16）
输出参数：szSwichState：控制器开关状态(顺序：从“左→右” 对应
“开关 1→开关 6”；0：关；1：开）
返 回 值：0：调用失败
1：调用成功

8、函数名：GetRightExtremity

功能说明：获取控制器右极限状态
函数原型：int __stdcall GetRightExtremity(unsigned int nSlaver,
int* nLimit)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nLimit：控制器右极限状态(0：断开；1：闭合）
返 回 值：0：调用失败
1：调用成功

9、函数名：GetLeftExtremity

功能说明：获取控制器左极限状态
函数原型：int __stdcall GetLeftExtremity(unsigned int nSlaver,
int* nLimit)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nLimit：控制器左极限状态(0：断开；1：闭合）
返 回 值：0：调用失败
1：调用成功

10、函数名：GetRightLimit

功能说明：获取控制器右限状态
函数原型：int __stdcall GetRightLimit(unsigned int nSlaver,
int* nLimit)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nLimit：控制器右限状态(0：断开；1：闭合）
返 回 值：0：调用失败
1：调用成功

11、函数名：GetLeftLimit

功能说明：获取控制器左限状态
函数原型：int __stdcall GetLeftLimit(unsigned int nSlaver,
int* nLimit)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nLimit：控制器左限状态(0：断开；1：闭合）
返 回 值：0：调用失败
1：调用成功

12、函数名：GetRunState

功能说明：获取控制器运行状态

函数原型：int __stdcall GetRunState(unsigned int nSlaver,
int* nRunState)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nRunState：控制器运行状态(0-停止；1-自动；2-左移；3-右移)
返 回 值：0：调用失败
1：调用成功

13、函数名：GetRemoteKeyState

功能说明：获取控制器遥控器按键状态
函数原型：int __stdcall GetRemoteKeyState(unsigned int nSlaver,
int* nKeyState)
输入参数：nSlaver：控制器地址（1-16）
输出参数：nKeyState：控制器按键状态(0-无按键；1-后退键；2-前进键；
3-自动键；4-停止键)
返 回 值：0：调用失败
1：调用成功
* 每次遥控器按钮新状态只存在 3 秒时间，3 秒后自动恢复为 0。

14、函数名：GetBiosDate

功能说明：获取控制器 BIOS 日期
函数原型：int __stdcall GetBiosDate(unsigned int nSlaver,
char** szDate)
输入参数：nSlaver：控制器地址（1-16）
输出参数：szDate：控制器 BIOS 日期
返 回 值：0：调用失败
1：调用成功

15、函数名：GetFactroyDate

功能说明：获取控制器出厂日期
函数原型：int __stdcall GetFactroyDate(unsigned int nSlaver,
char** szDate)
输入参数：nSlaver：控制器地址（1-16）
输出参数：szDate：控制器出厂日期
返 回 值：0：调用失败
1：调用成功

16、函数名：GetCurrentPoint

功能说明：获取当前坐标(cm)
函数原型：int __stdcall GetCurrentPoint(unsigned int nSlaver,
float* fPoint)

输入参数：nSlaver：控制器地址（1-16）
输出参数：fPoint：当前坐标(cm)
返 回 值：0：调用失败
1：调用成功

17、函数名：SetInit

功能说明：初始化控制器（开关全关，停止移动）
函数原型：int __stdcall SetInit(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

18、函数名：SetMoveLeft

功能说明：发送向左移动指令
函数原型：int __stdcall SetMoveLeft(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

19、函数名：SetMoveRight

功能说明：发送向右移动指令
函数原型：int __stdcall SetMoveRight(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

20、函数名：SetMoveAuto

功能说明：发送自动移动指令
函数原型：int __stdcall SetMoveAuto(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

21、函数名：SetMoveStop

功能说明：发送停止移动指令

函数原型：int __stdcall SetMoveStop(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

22、函数名：SetSwichAllOff

功能说明：发送控制器开关全关指令
函数原型：int __stdcall SetSwichAllOff(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

23、函数名：SetSwichAllOn

功能说明：发送控制器开关全开指令
函数原型：int __stdcall SetSwichAllOn(unsigned int nSlaver)
输入参数：nSlaver：控制器地址（1-16）
输出参数：无
返 回 值：0：调用失败
1：调用成功

24、函数名：SetSwichState

功能说明：设置控制器开关状态
函数原型：int __stdcall SetSwichState(unsigned int nSlaver,
unsigned int nSwich,
unsigned int nState)
输入参数：nSlaver：控制器地址（1-16）
nSwich：开关顺序号（1-6）
nState：开关状态（0：关；1：开）
输出参数：无
返 回 值：0：调用失败
1：调用成功

三、动态库调用流程：

1、初始化通信串口（OpenCom）；
2、绑定控制器授权文件（DllBind）；
3、调用函数（…）；
4、调用函数（…）；
5、……
6、关闭通信串口（CloseCom）。

四、调用示例（C#）

1、函数声明：

```
[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "OpenCom")]
public static extern int OpenCom(int iPort);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "CloseCom")]
public static extern void CloseCom();

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "DllBind")]
public static extern int DllBind(uint iSlaver);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "IsComOpen")]
public static extern int IsComOpen();

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "GetSerial")]
public static extern int GetSerial(uint iSlaver, ref string strId);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "GetRunState")]
public static extern int GetRunState(uint iSlaver, ref int iRunState);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "GetCurrentPoint")]
public static extern int GetCurrentPoint(uint iSlaver, ref float fPoint);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "SetInit")]
public static extern int SetInit(uint iSlaver);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "SetMoveLeft")]
public static extern int SetMoveLeft(uint iSlaver);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "SetMoveRight")]
public static extern int SetMoveRight(uint iSlaver);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "SetMoveAuto")]
```

```
public static extern int SetMoveAuto(uint iSlaver);

[DllImport("CC04Sdk.dll", CharSet = CharSet.Ansi, EntryPoint = "SetMoveStop")]
public static extern int SetMoveStop(uint iSlaver);
```

2、调用示例：

```
var iRet = CC04Sdk.OpenCom(7);
if (iRet != 1)
{
    MessageBox.Show(@"通信串口初始化失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return;
}
MessageBox.Show(@"通信串口初始化成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

iRet = CC04Sdk.DllBind(1);
if (iRet == 1)
{
    timer1.Enabled = true;
    MessageBox.Show(@"绑定控制器授权文件成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show(@"绑定控制器授权文件失败！返回码：" + iRet, @"提示", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
}

iRet = CC04Sdk.SetInit(1);
if (iRet == 1)
{
    MessageBox.Show(@"初始化控制器成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show(@"初始化控制器失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

var    str = "";
iRet = CC04Sdk.GetSerial(1, ref str);
if (iRet == 1)
{
    MessageBox.Show(@"获取控制器序列号成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    MessageBox.Show(@"序列号：" + str, @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

```
else
{
    MessageBox.Show(@"获取控制器序列号失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

var iRun = -1;
iRet = CC04Sdk.GetRunState(1, ref iRun);
if (iRet == 1)
{
    MessageBox.Show(@"获取控制器运行状态成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    MessageBox.Show(@"运行状态：" + iRun, @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show(@"获取控制器运行状态失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

float fRet = 0;
iRet = CC04Sdk.GetCurrentPoint(1, ref fRet);
if (iRet == 1)
{
    MessageBox.Show(@"获取当前坐标成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    MessageBox.Show(@"当前坐标：" + fRet.ToString("0.0"), @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show(@"获取当前坐标失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

 iRet = CC04Sdk.SetMoveLeft(1);
if (iRet == 1)
{
    MessageBox.Show(@"发送向左移动指令成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show(@"发送向左移动指令失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

 iRet = CC04Sdk.SetMoveRight(1);
if (iRet == 1)
{
    MessageBox.Show(@"发送向右移动指令成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

```
else
{
   MessageBox.Show(@"发送向右移动指令失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

 iRet = CC04Sdk.SetMoveAuto(1);
if (iRet == 1)
{
   MessageBox.Show(@"发送自动动指令成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
   MessageBox.Show(@"发送自动动指令失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

 iRet = CC04Sdk.SetMoveStop(1);
if (iRet == 1)
{
   MessageBox.Show(@"发送停止动指令成功！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
   MessageBox.Show(@"发送停止动指令失败！", @"提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

CC04Sdk.CloseCom();
```