

案例教程 02 轴使能与点动

在 Visual Studio 2022 里写一个伺服轴控制程序：
使能、按住点动、松开就停，实时显示轴的状态。
每一步都有截图，每一行代码都讲清楚。

Visual Studio 2022

C# WinForms

.NET 8

软卡 SDK

适用软件	SCMotion 4.4.17
配套视频	教程02_轴使能与点动_编程篇_WinForms版.mp4
配套工程	配套工程/AxisDemo
文档版本	V1.0
发布日期	2026-10

ABOUT

关于本教程

适合谁	做完教程01的人。没做过也能跟：新建项目、引用 SDK 这两步在《教程01 图文教程》第 3、4 章讲得最细，本文只列步骤。
做出什么	一个 Windows 窗体程序：连接主软件、给伺服轴使能 / 下使能 / 复位，按住按钮点动、松开就停，还能随时停止；每 200 毫秒刷新一次轴有没有使能、有没有故障，以及位置和速度。
开发环境	Visual Studio 2022 · .NET 8 · C# Windows 窗体应用（WinForms）
适用软件	SCMotion 4.4.17，软卡 SDK 用同一版本主软件生成
配套视频	教程02_轴使能与点动_编程篇_WinForms版.mp4 （16 分钟，17 个章节）。本文各章和视频章节对应，章标题右边标了视频里的时间。
配套工程	配套工程/AxisDemo ：代码和本文附录 A 一字不差，打开就能编译运行。
建议先看	../教程02_轴使能与点动_上篇_主软件操作.mp4 ：先在主软件里把轴使能、点动通，再来写程序。WinForms / WPF / Qt 三个版本共用这一集。

安全

第一次让真实电动机之前：电机和机械负载脱开（或者负载能自由运动）；急停按钮放在手边；速度先用最小值。仿真里做错没有后果，真机上没有撤销。

怎么用这份教程

- 建议一边看视频一边做：视频看「怎么点」，这份文档查「点哪里、写什么、为什么」。
- 带编号圆点的步骤照顺序做；截图里的**红框**就是要点的地方。
- 「」里是界面上的文字，等宽字是代码里的名字。代码块左边的数字是 **Form1.cs 里的行号**，和附录 A、配套工程完全对得上。
- 这一期的重点是第 7 章的**使能**（7.4）和**点动**（7.5），以及第 9 章的三个要点。
- 出了问题先看第 9 章「常见问题」，再按返回码查附录 C。

需要的东西

东西	说明
电脑	Windows 10/11 64 位，装好 SCSMotion 4.4.17
Visual Studio 2022	社区版就可以，安装时勾选「.NET 桌面开发」（见《教程01 图文教程》2.1）
伺服驱动器和电机	一台 EtherCAT 伺服驱动器和电机，网线接驱动器的 IN 口，在主软件里已经扫描、建好轴。 没有伺服也能学 ：把主软件切到「仿真」，仿真里的轴同样能使能、能点动。

CONTENTS 目录

第 1 章 最终效果	视频 00:56	4
第 2 章 准备工作	视频 00:21	5
2.1 机械安全		5
2.2 先在主软件里点动过		5
2.3 让主软件处于运行状态		5
第 3 章 新建项目	视频 01:26	6
第 4 章 引用 SDK	视频 01:50	7
第 5 章 画界面	视频 02:15	8
第 6 章 挂事件	视频 03:19	10
6.1 六个按钮、定时器和关闭事件		10
6.2 点动按钮要用 MouseDown 和 MouseUp		10
第 7 章 写代码	视频 04:20	12
7.1 公共部分：using、三个成员、Log		12
7.2 「连接」按钮		13
7.3 断开：先停轴，再归还		14
7.4 使能、下使能、复位		15
7.5 点动：按下开始走，松开就收手		16
7.6 停止		18
7.7 状态刷新：tmrPoll_Tick		18
第 8 章 编译运行	视频 14:22	19
8.1 编译		19
8.2 运行验证		19
第 9 章 要点回顾和常见问题	视频 15:23	21
9.1 三个要点		21
9.2 常见问题		22
附录 A 完整代码 Form1.cs		23
附录 B 本教程用到的 SDK 指令		26
附录 C 返回码速查		29

第 1 章 最终效果

视频 00:56

先看最后要做出来的程序：



图 1-1 最后做出来的程序（已连接，轴已使能，刚点动过）

编号	控件	作用
①	「连接」「断开」	连接主软件并取得控制权；断开时先停轴，再把控制权还给主软件
②	轴号、点动速度	轴号从 0 开始：主软件里的第一根轴在程序里是 0 号轴。点动速度的单位和主软件轴页里的用户单位一致（比如毫米每秒）
③	「使能」「下使能」「复位」	给伺服上电 / 断电；清除轴故障
④	使能指示、故障指示	程序每 200 毫秒读一次轴状态：使能完成显示绿色「已使能」；有故障显示橙红色「故障」
⑤	「反向点动」「正向点动」「停止」	按住 点动按钮轴就走， 松开 就减速停下；「停止」随时让轴停下
⑥	位置和速度	实时显示轴的实际位置和实际速度，保留三位小数
⑦	日志框	每一步做了什么、返回码是多少，都记在这里

第 2 章 准备工作

视频 00:21

让轴动起来之前，先确认三件事。

2.1 机械安全 视频 00:26

- 电机和负载尽量脱开，急停按钮放在手边。
- 程序里的点动速度第一次用小值（本程序默认 10）。

2.2 先在主软件里点动过 视频 00:33

先在主软件里打开这根轴的「在线」页，低速点动，确认方向和当量（每个用户单位多少脉冲）都正确，再交给自己的程序。上篇视频里演示过这一步。



图 2-1 主软件里轴的「在线」页：先在这里低速点动，确认方向和当量

2.3 让主软件处于运行状态 视频 00:43

- 接了真机：点工具栏的「激活配置参数」，让从站进入 OP。
- 手边没有伺服：点模式条上的「仿真」，仿真里的轴同样能使能、能点动。



图 2-2 没有伺服时，点模式条上的「仿真」

说明 软卡 SDK 包的生成方法、Visual Studio 的安装要求，和教程01 完全一样，见《教程01 图文教程》第 2 章。C# 程序只需要 SDK 包里的 `sd\dotnet\SCmotionSoftCard.Core.dll` 这一个文件。

第 3 章 新建项目

视频 01:26

步骤和教程01 完全一样（每一步的截图见《教程01 图文教程》第 3 章）：

- 1 打开 Visual Studio 2022，点「创建新项目」。
- 2 搜索「窗体」，选 C# 的「Windows 窗体应用」。不要选名字后面带「(.NET Framework)」的那个。点「下一步」。
- 3 「项目名称」填 AxisDemo，勾上「将解决方案和项目放在同一目录中」，点「下一步」。
- 4 「框架」选 .NET 8.0（长期支持），点「创建」。



图 3-1 选 C# 的「Windows 窗体应用」

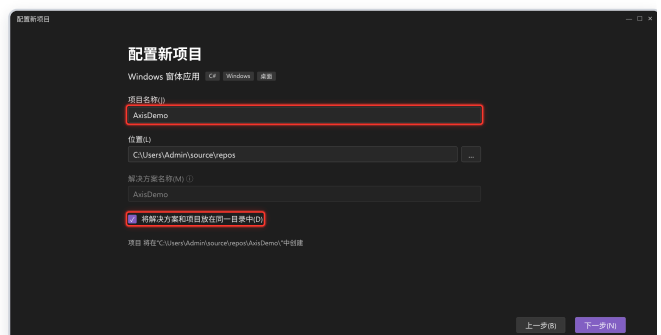


图 3-2 项目名称 AxisDemo，勾上「放在同一目录中」

第 4 章 引用 SDK

视频 01:50

同样和教程01 一样（详细截图见《教程01 图文教程》第 4 章）：

- 1 在项目文件夹 AxisDemo 里新建一个 **sdk** 文件夹，把 **SCmotionSoftCard.Core.dll** 复制进去。
- 2 在解决方案资源管理器里右键「依赖项」→「添加项目引用」，左边点「浏览」，再点右下角「浏览(B)...」，选中 sdk 文件夹里的 DLL，点「添加」。
- 3 确认它前面打了勾，点「确定」。
- 4 右键项目「AxisDemo」→「属性」，在「生成」→「常规」里把「平台目标」从 Any CPU 改成 **x64**，按 **Ctrl + S** 保存。



图 4-1 项目里多了 sdk 文件夹

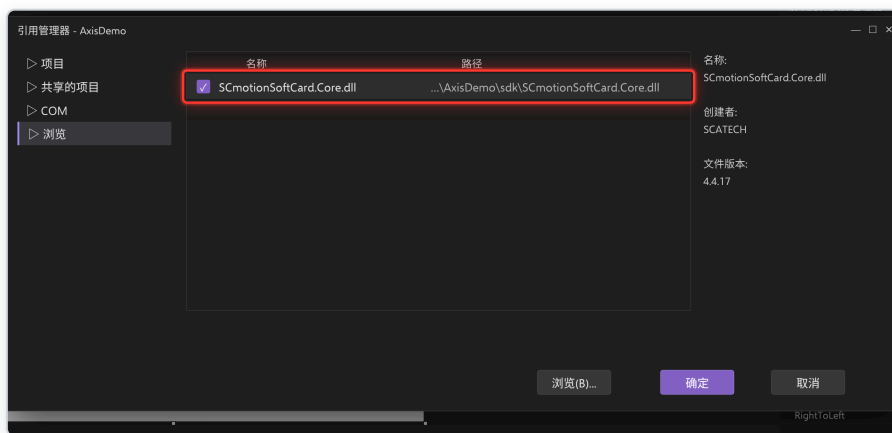


图 4-2 引用管理器里打勾，点「确定」



图 4-3 「平台目标」改成 x64

注意 SDK 是 64 位的，平台目标不改成 x64，编译时会有「处理器架构不匹配」的警告，运行时还可能出错。

第 5 章 画界面

视频 02:15

先选中窗体，把属性「Text」改成 轴使能与点动演示。再从工具箱把下表的控件拖上去，Name 必须一字不差，位置照着图 5-1 摆个大概就行。拖控件、改属性的方法见《教程01 图文教程》第 5 章。

控件类型	Name	Text	还要改的属性
窗体	Form1	轴使能与点动演示	—
Button	btnConnect	连接	
Button	btnDisconnect	断开	
Label	label1	轴号	
NumericUpDown	numAxis		Maximum = 63
Label	label2	点动速度	
NumericUpDown	numSpeed		Minimum = 1, Maximum = 1000, Value = 10
Button	btnPowerOn	使能	
Button	btnPowerOff	下使能	
Button	btnReset	复位	
Label	lblPowered	未使能	AutoSize = False, BorderStyle = FixedSingle, TextAlign = MiddleCenter, BackColor = Gainsboro
Label	lblFaulted	无故障	同 lblPowered
Button	btnJogNeg	反向点动	
Button	btnJogPos	正向点动	
Button	btnStop	停止	
Label	lblInfo	--	同 lblPowered，但 TextAlign = MiddleLeft
TextBox	txtLog		Multiline = True, ReadOnly = True, ScrollBars = Vertical
Timer（在「组件」里）	tmrPoll		Interval = 200

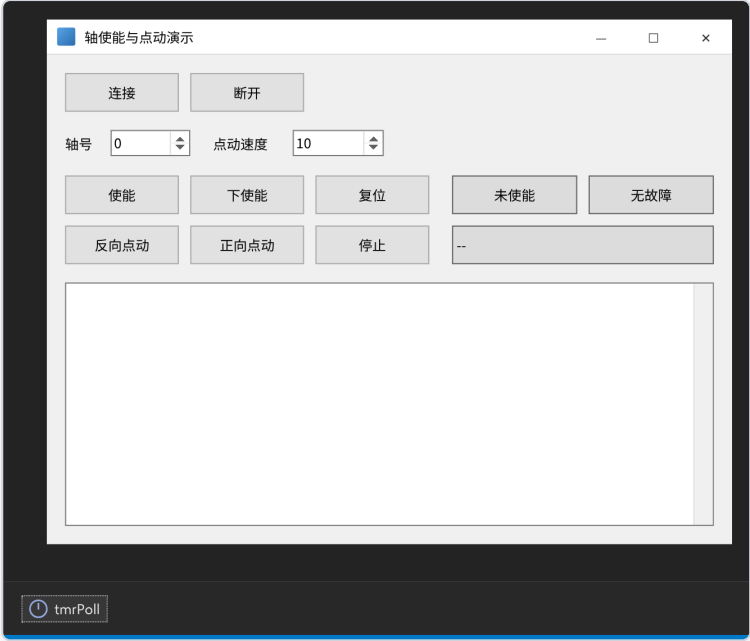


图 5-1 画好的界面（下方是定时器 tmrPoll）

设置	为什么
numAxis: Maximum = 63	轴号从 0 开始：主软件里的第一根轴在程序里是 0 号轴。最大值给够就行
numSpeed: Minimum = 1	点动时速度不能是 0，否则 SDK 返回 10（参数非法）。最小值设成 1，正好避开
numSpeed: Maximum = 1000, Value = 10	速度的单位和主软件轴页里的用户单位一致，比如毫米每秒。第一次用小值
三个标签: AutoSize = False, BorderStyle = FixedSingle	lblPowered、lblFaulted 当指示块，lblInfo 显示位置和速度，都要拉成带边框的方块
tmrPoll: Interval = 200	每 200 毫秒读一次轴状态

第 6 章 挂事件

视频 03:19

6.1 六个按钮、定时器和关闭事件 视频 03:19

- 1 在设计器里，对「连接」「断开」「使能」「下使能」「复位」「停止」六个按钮各双击一次，生成它们的 Click 函数。
- 2 双击设计器下方的 `tmrPoll`，生成 `tmrPoll_Tick`。
- 3 选中窗体，属性窗口点闪电图标切到事件列表，双击「FormClosing」。

6.2 点动按钮要用 MouseDown 和 MouseUp 视频 03:41

点动要「按住才走、松开就停」，所以点动按钮不能用 Click：Click 是松开鼠标以后才触发的，而且一次点击只触发一次。我们要的是按下的一刻开始走、松开的一刻收手，正好对应 **MouseDown**（鼠标按下）和 **MouseUp**（鼠标松开）两个事件。

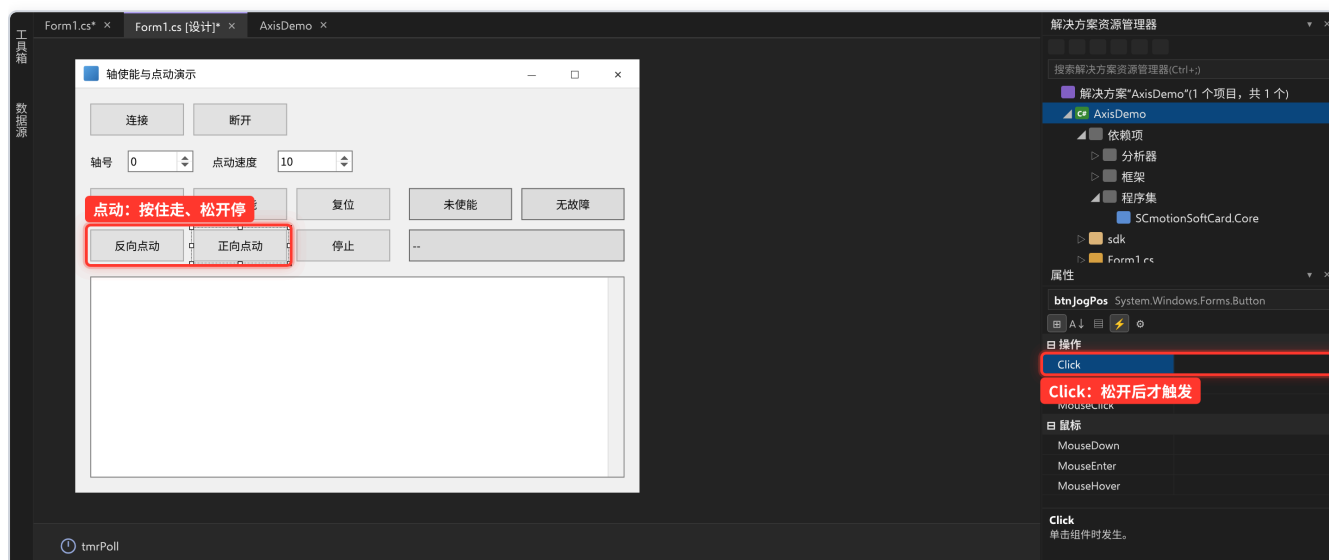


图 6-1 点动按钮不用 Click：Click 松开后才触发

- 4 选中「正向点动」，属性窗口点闪电图标，在事件列表里找到「MouseDown」，双击。
- 5 回到设计器，再找到「MouseUp」，双击。
- 6 「反向点动」也一样，挂上 MouseDown 和 MouseUp。

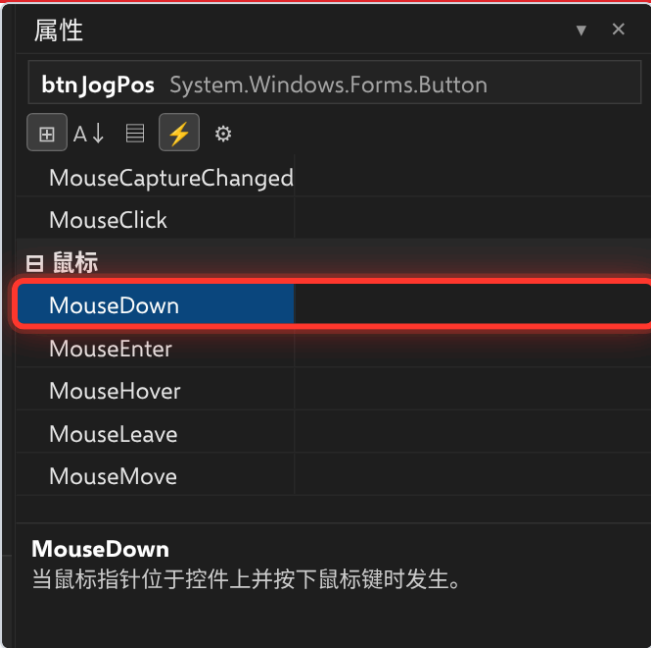


图 6-2 事件列表里双击 MouseDown

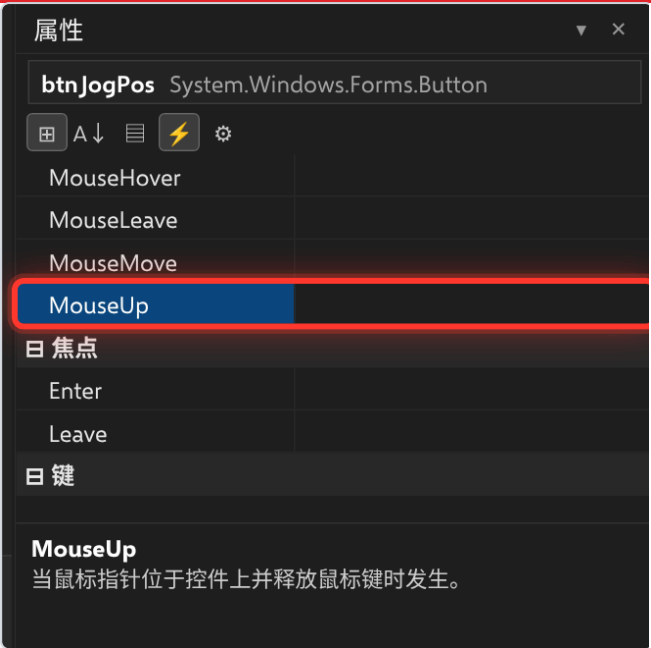


图 6-3 再双击 MouseUp

挂完以后，代码里一共有 12 个自动生成的函数：

函数	什么时候执行
btnConnect_Click / btnDisconnect_Click	点「连接」 / 「断开」
btnPowerOn_Click / btnPowerOff_Click	点「使能」 / 「下使能」
btnReset_Click / btnStop_Click	点「复位」 / 「停止」
btnJogPos_MouseDown / btnJogPos_MouseUp	在「正向点动」上按下 / 松开鼠标
btnJogNeg_MouseDown / btnJogNeg_MouseUp	在「反向点动」上按下 / 松开鼠标
tmrPoll_Tick	定时器启动以后，每 200 毫秒一次
Form1_FormClosing	窗口关闭之前

注意 MouseUp 一定要挂上。只挂 MouseDown 的话，按下去轴开始走，松开却没有任何代码去让它停。

第 7 章 写代码

视频 04:20

下面按视频的順序往 Form1.cs 里填代码。代码块左边是最终文件里的行号，写完以后整个文件和附录 A 一模一样。和教程01 相同的部分讲得简短一些，详细解释见《教程01 图文教程》第 7 章。

这一期用到 10 条 SDK 指令：

指令	作用	在哪用
SCM_OpenProjectConfig	读连接配置文件，连上主软件，拿到句柄	7.2 连接
SCM_GetOpenErrorText	连接失败时取出具体原因	7.2 连接
SCM_RequestOwner	取得控制权，之后才能使能、点动	7.2 连接
SCM_ReleaseOwner / SCM_Close	归还控制权 / 关闭句柄	7.3 断开
SCM_AxisPower	使能（1）或下使能（0）	7.4 使能
SCM_AxisReset	清除轴故障	7.4 复位
SCM_AxisJog	点动：方向 1 / -1 开始走，方向 0 收手	7.5 点动
SCM_AxisStop	普通停止（最终停）	7.3、7.6
SCM_AxisGetStatus	读轴的完整状态：是否使能、是否故障、位置、速度……	7.7 状态刷新

7.1 公共部分：using、三个成员、Log 视频 04:20

和教程01 一字不差。文件最上面写三行 using：

```
1 using System.Text;
2 using SCmotion.SoftCard.Sdk;
3 using static SCmotion.SoftCard.Sdk.ScMotionApi;
```

类里面、构造函数上方定义三个成员：

```
9 // 连接配置文件：主软件进 OP（或切到仿真）时自动生成，路径固定
10 const string ConfigPath = @"C:\ProgramData\SoftSC\softcard.runtime.json";
11
12 // 本程序的身份证号：自己定一个，多个程序同时连接时要各不相同
13 const int ClientId = 88;
14
15 // 软卡句柄：连接成功后才有值，null 表示还没连接
16 ScMotionCard? card;
```

类的最后面写日志函数 Log：

```
174 // 在日志框末尾追加一行，前面带上时间
175 private void Log(string text)
176 {
177     txtLog.AppendText($"{DateTime.Now:HH:mm:ss} {text}\r\n");
178 }
```

行	说明
1-3	引入 StringBuilder 所在的 <code>System.Text</code> 、SDK 命名空间； <code>using static</code> 让我们直接写 <code>SCM_</code> 开头的指令名。
10	连接配置文件路径。主软件进 OP 或切到仿真时自动写出这个文件，路径固定。
13	本程序的身份号，自己定，这里用 88。几个程序同时连接时要各不相同。
16	软卡句柄。 <code>?</code> 表示可以是 null：没连接时是 null，连接成功后才有值。
177	在日志框末尾追加一行：前面带当前时间，最后的 <code>\r\n</code> 是换行。

7.2 「连接」按钮 视频 06:07

写法和教程01 一样：打开连接 → 取得控制权 → 启动定时刷新。

```

23 // 「连接」按钮：打开连接 → 取得控制权 → 启动定时刷新
24 private void btnConnect_Click(object sender, EventArgs e)
25 {
26     if (card != null) { Log("已经连接了"); return; }
27
28     int rc = SCM_OpenProjectConfig(ConfigPath, ClientId, out card);
29     if (rc != SCM_OK)
30     {
31         var why = new StringBuilder(1024);
32         SCM_GetOpenErrorText(why, why.Capacity);
33         Log($"连接失败，返回码 {rc}: {why}");
34         card = null;
35         return;
36     }
37
38     rc = SCM_RequestOwner(card);
39     if (rc != SCM_OK)
40     {
41         Log($"取控制权失败，返回码 {rc}");
42         SCM_Close(card);
43         card = null;
44         return;
45     }
46
47     tmrPoll.Start();
48     Log("连接成功，已取得控制权");
49 }

```

行	说明
26	防止重复连接：card 不是 null 说明已经连上了，写一条日志就返回。
28	SCM_OpenProjectConfig 连接主软件：配置文件路径、身份号，第三个是出参，成功后句柄写回 card。返回码存进 rc，0 表示成功。
29-36	连接失败：用 StringBuilder 接住 SCM_GetOpenErrorText 给出的具体原因（参数对不上时会逐项列出双方的轴数、IO 点数和周期），写日志，card 设回 null，返回。
38	SCM_RequestOwner 取得控制权。使能、点动这些写操作，都必须先拿到控制权。
39-45	拿不到控制权：记下返回码（3 一般是主软件没进 OP；7 是控制权在别的程序手上），关闭句柄，card 设回 null，返回。
47-48	两步都成功：启动定时器，写一条连接成功的日志。

7.3 断开：先停轴，再归还 视频 07:55

Disconnect 写在 Log 的上面。和教程01比，多了一行停轴：

```

161 // 断开：停定时器 → 停轴 → 归还控制权 → 关闭句柄
162 private void Disconnect()
163 {
164     if (card == null) return;
165
166     tmrPoll.Stop();
167     SCM_AxisStop(card, (int)numAxis.Value);
168     SCM_ReleaseOwner(card);
169     SCM_Close(card);
170     card = null;
171     Log("已断开");
172 }

```

行	说明
164-166	没连接就返回；然后停掉定时器。
167	新增：SCM_AxisStop 先让轴停下来。因为归还控制权并不会停止正在进行的运动：要是轴还在动就把控制权还了，它会继续走。所以断开之前，先停轴，再归还。
	注意这一行只停当前选中的这根轴，停下以后轴还保持使能。所以运行时我们先点「下使能」，再点「断开」。
168-171	归还控制权、关闭句柄，card 设回 null，写日志。

「断开」按钮和窗口关闭事件里，都调用 Disconnect：

```

51 // 「断开」按钮
52 private void btnDisconnect_Click(object sender, EventArgs e)
53 {
54     Disconnect();
55 }

```

```

110 // 关闭窗口前也断开，把控制权还给主软件
111 private void Form1_FormClosing(object sender, FormClosingEventArgs e)
112 {
113     Disconnect();
114 }

```

7.4 使能、下使能、复位 视频 08:46

这是这一期的重点。「使能」按钮：

```

57 // 「使能」按钮：返回 0 只表示命令已受理，使能完成要看 powered
58 private void btnPowerOn_Click(object sender, EventArgs e)
59 {
60     if (card == null) { Log("请先连接"); return; }
61
62     int axis = (int)numAxis.Value;
63     int rc = SCM_AxisPower(card, axis, 1);
64     Log($"使能 轴{axis}, 返回码 {rc}");
65 }

```

行	说明
57	注释写清楚：返回 0 只表示命令已受理，使能完成要看 powered。
60	先检查有没有连接。
62	从数字框读出轴号。numAxis.Value 是 decimal，用 (int) 转成整数。
63	发出使能命令 SCM_AxisPower：三个参数依次是句柄、轴号，最后一个 1 表示使能，0 表示下使能。
64	把返回码写进日志。

重点 受理不等于完成：SCM_AxisPower 返回 0，只表示主软件「收下了」这条命令，并不表示轴已经使能好了。

使能是在后台一步一步完成的：成功通常要一百多毫秒到几秒（最长约 8 秒），失败最长要 12 秒多。所以等待的超时时间不要短于 15 秒。

怎么知道使能完成了？看轴状态里的 powered 变成 True。7.7 的定时器每 200 毫秒读一次状态，就能实时看到它。

「下使能」几乎一模一样，只是最后一个参数传 0：

```

67 // 「下使能」按钮
68 private void btnPowerOff_Click(object sender, EventArgs e)
69 {
70     if (card == null) { Log("请先连接"); return; }
71
72     int axis = (int)numAxis.Value;
73     int rc = SCM_AxisPower(card, axis, 0);
74     Log($"下使能 轴{axis}, 返回码 {rc}");
75 }

```

「复位」用来清除轴的故障，调用 SCM_AxisReset，参数是句柄和轴号：

```
77 // 「复位」按钮：清除轴故障，完成要看 faulted 变回 False
78 private void btnReset_Click(object sender, EventArgs e)
79 {
80     if (card == null) { Log("请先连接"); return; }
81
82     int axis = (int)numAxis.Value;
83     int rc = SCM_AxisReset(card, axis);
84     Log($"复位 轴{axis}, 返回码 {rc}");
85 }
```

说明 复位同样是「收下就返回」：完成要看 faulted 变回 False，至少要 0.45 秒。复位成功以后，轴一般处于未使能状态，要再点一次「使能」。

7.5 点动：按下开始走，松开就收手 视频 10:26

在 Disconnect 的上面写两个公用函数：开始点动 StartJog 和点动收手 StopJog。

```
140 // 开始点动: direction 为 1 正向, 为 -1 反向
141 private void StartJog(int direction)
142 {
143     if (card == null) { Log("请先连接"); return; }
144
145     int axis = (int)numAxis.Value;
146     double speed = (double)numSpeed.Value;
147     int rc = SCM_AxisJog(card, axis, direction, speed, 0, 0);
148     Log($"点动 轴{axis} 方向{direction} 速度{speed}, 返回码 {rc}");
149 }
```

行	说明
141	参数 direction 是方向：1 正向，-1 反向。
143-145	检查连接，读出轴号。
146	读出点动速度。SCM_AxisJog 的速度参数是 double（双精度小数），所以把数字框的值转成 double。
147	开始点动：SCM_AxisJog，六个参数依次是句柄、轴号、方向、速度、加速度、减速度。加速度和减速度传 0，表示使用主软件轴页里设置的默认值。
148	把方向、速度和返回码写进日志。

说明 方向只看 direction，不看速度的正负：速度传负数也只按绝对值算，不会反着走。方向不是 0 的时候，速度不能是 0，否则返回 10。我们把速度的最小值设成了 1，正好避开。

```
151 // 点动收手: direction 传 0, 轴按减速度停下
152 private void StopJog()
153 {
154     if (card == null) return;
155
156     int axis = (int)numAxis.Value;
157     int rc = SCM_AxisJog(card, axis, 0, 0, 0, 0);
158     Log($"点动收手 轴{axis}, 返回码 {rc}");
159 }
```

行	说明
154-156	检查连接，读出轴号。
157	收手还是调用 SCM_AxisJog，但方向传 0：轴按减速度减速停下来。这不是急停。
158	写日志。

回到四个鼠标事件：按下时调用 StartJog（正向传 1，反向传 -1），松开时调用 StopJog。

```
116 // 「正向点动」按下：开始正向走
117 private void btnJogPos_MouseDown(object sender, MouseEventArgs e)
118 {
119     StartJog(1);
120 }
121
122 // 「正向点动」松开：收手
123 private void btnJogPos_MouseUp(object sender, MouseEventArgs e)
124 {
125     StopJog();
126 }
127
128 // 「反向点动」按下：开始反向走
129 private void btnJogNeg_MouseDown(object sender, MouseEventArgs e)
130 {
131     StartJog(-1);
132 }
133
134 // 「反向点动」松开：收手
135 private void btnJogNeg_MouseUp(object sender, MouseEventArgs e)
136 {
137     StopJog();
138 }
```

注意 松开时发收手这一步千万不能漏。如果只在按下时发命令、松开时不发，轴会一直走下去。

7.6 停止 视频 12:38

```

87 // 「停止」按钮：普通停止，停下后要重新发运动指令才会再动
88 private void btnStop_Click(object sender, EventArgs e)
89 {
90     if (card == null) { Log("请先连接"); return; }
91
92     int axis = (int)numAxis.Value;
93     int rc = SCM_AxisStop(card, axis);
94     Log($"停止 轴{axis}, 返回码 {rc}");
95 }

```

行	说明
90-92	检查连接，读出轴号。
93	<code>SCM_AxisStop</code> 是普通停止，也是「最终停」：轴停下来以后，要重新发运动指令才会再动。万一一点动松手时没收到 <code>MouseUp</code> （比如按住时切换了窗口），就点它。

7.7 状态刷新：tmrPoll_Tick 视频 13:03

定时器每 200 毫秒读一次轴状态，刷新界面：

```

97 // 定时器：每 200 毫秒读一次轴状态，刷新界面
98 private void tmrPoll_Tick(object sender, EventArgs e)
99 {
100     if (card == null) return;
101
102     SCM_AxisGetStatus(card, (int)numAxis.Value, out SCM_AXIS_STATUS st);
103     lblPowered.Text = st.powered ? "已使能" : "未使能";
104     lblPowered.BackColor = st.powered ? Color.LimeGreen : Color.Gainsboro;
105     lblFaulted.Text = st.faulted ? "故障" : "无故障";
106     lblFaulted.BackColor = st.faulted ? Color.OrangeRed : Color.Gainsboro;
107     lblInfo.Text = $"位置 {st.actualPosition:F3}    速度 {st.actualVelocity:F3}";
108 }

```

行	说明
100	没连接就返回。
102	<code>SCM_AxisGetStatus</code> 读出这根轴的完整状态。读状态是只读操作， 不需要控制权 。最后一个参数是出参： <code>out SCM_AXIS_STATUS st</code> 声明一个状态结构体变量 <code>st</code> ，位置、速度、是否使能、是否故障，全都在它里面。
103-104	<code>st.powered</code> 表示使能是否完成：为真显示「已使能」、背景变绿，否则显示「未使能」。
105-106	<code>st.faulted</code> 表示轴有没有故障：有故障显示「故障」、背景变成橙红色。
107	显示实际位置 <code>actualPosition</code> 和实际速度 <code>actualVelocity</code> 。冒号后面的 <code>F3</code> 表示保留三位小数，单位和主软件轴页一致。

注意 轴号填了一个不存在的轴时，读状态会返回 4，结构体里全是 0，界面同样显示「未使能」。点了使能一直没变化，先确认轴号对不对。

第 8 章 编译运行

视频 14:22

8.1 编译 视频 14:22

- 1 代码全部写完，按 **Ctrl** + **S** 保存。
- 2 按 **Ctrl** + **Shift** + **B** 生成解决方案，输出窗口显示「1 成功，0 失败」。

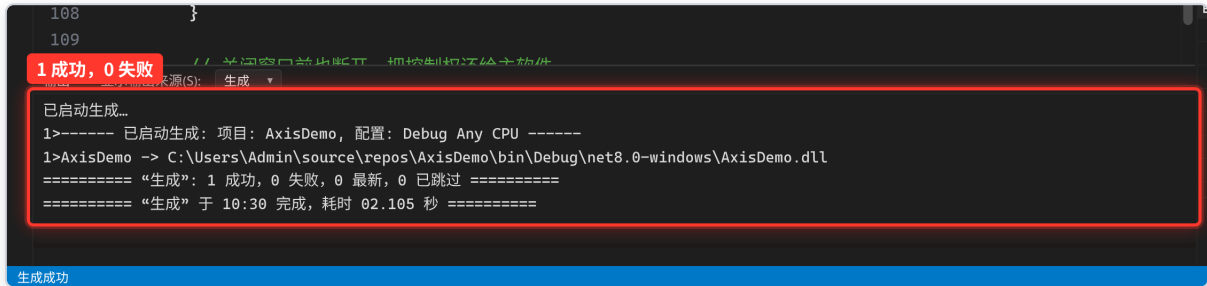


图 8-1 输出窗口：1 成功，0 失败

8.2 运行验证 视频 14:31

- 1 确认主软件已经进 OP，或者切到了仿真。按 **F5** 运行，点「连接」。
- 2 点「使能」。日志里马上出现返回码 0，但这时指示块还是「未使能」：命令只是被收下了。



图 8-2 刚点「使能」：返回码 0，但还没使能（右边是伺服轴的示意）

- 3 过了一会儿，指示块变成绿色的「已使能」，这才是使能真正完成。这就是「受理」和「完成」的区别。

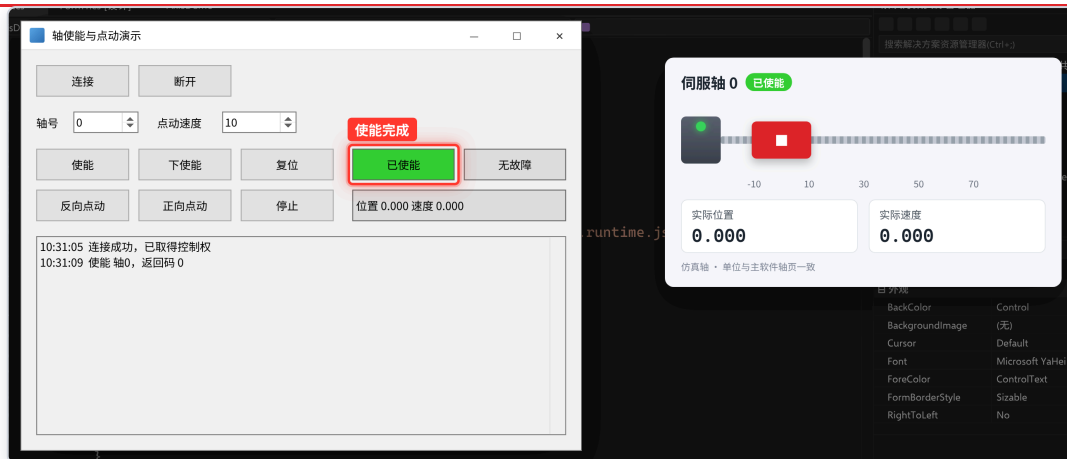


图 8-3 过一会儿变成「已使能」: 使能完成

- 4 按住「正向点动」, 轴开始正向走, 位置一直在增加。松开鼠标, 程序发出收手命令, 轴减速停下, 速度回到 0。



图 8-4 按住「正向点动」: 位置增加, 速度 10

- 5 按住「反向点动」, 位置开始减少; 松开, 同样停下。
- 6 最后先点「下使能」, 再点「断开」, 控制权还给主软件。

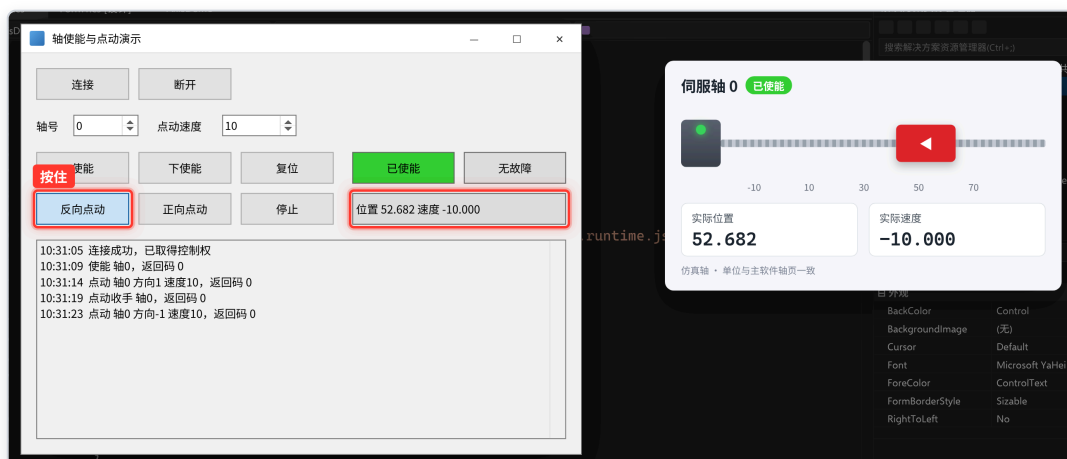


图 8-5 按住「反向点动」: 位置减少, 速度 -10

第 9 章 要点回顾和常见问题

视频 15:23

9.1 三个要点

视频 15:23

要点	具体怎么做
① 受理不等于完成	使能和复位返回 0，只表示命令收下了。使能完成看 <code>powered</code> ，复位完成看 <code>faulted</code> ；使能等待的超时不要短于 15 秒。
② 点动必须收手	按钮绑 <code>MouseDown</code> 和 <code>MouseUp</code> ：按下发方向 1 或 -1，松开一定再发方向 0。
③ 停轴有三种方法	见下表。

指令	效果	什么时候用
<code>SCM_AxisStop</code>	普通停止，最终停：停下以后要重新发运动指令才会再动	要「停了就别再动」的场合，比如本程序的「停止」和「断开」
<code>SCM_AxisHalt</code>	按减速度停，减速过程中可以被新的运动指令直接接管	「松开先减速，但随时可能再按下去」这类交互
<code>SCM_AxisEStop</code>	急停，以最快方式停轴，会让轴进入故障状态	紧急情况。恢复要先复位、等 <code>faulted</code> 变假，再使能、等 <code>powered</code>

注意 软件急停不能替代硬件急停回路。

9.2 常见问题

现象	原因	怎么办
连接失败，返回码 10 / 5	主软件没进 OP 也没切仿真；或者 SDK 服务没起来、版本不一致	同教程01：先让主软件进 OP 或切仿真；看「软卡设置」页状态；重新生成 SDK 包
取控制权失败，返回码 3 / 7	3：主软件没进 OP；7：控制权在别的程序手上	进 OP 或切仿真；把另一个程序断开，刚强行停止程序的等 5 秒再连
点了「使能」返回 0，但一直是「未使能」	使能还在进行（最长约 8 秒）；或者使能失败（驱动器有故障、急停没松开）；或者轴号填错了	等够 15 秒；看「故障」指示，有故障先复位再使能；确认轴号（主软件第一根轴是 0）
使能返回 3	总线没进 OP，或者本程序没有控制权	让主软件进 OP；点「断开」再「连接」重新取得控制权
使能、点动、读状态返回 4	轴号非法：轴号从 0 开始，最大是轴数减 1	核对主软件里的轴数和轴号
点动返回 10	方向不是 0 的时候，速度是 0	点动速度至少填 1（数字框 Minimum = 1）
点动、停止返回 7	本程序没有控制权	点「断开」再「连接」，重新取得控制权
松开按钮轴不停	没挂 MouseUp；或者按住时切换了窗口，没收到 MouseUp	检查第 6 章的 MouseUp 事件；点「停止」
点「断开」以后轴还是使能状态	Disconnect 只停轴，不下使能	断开之前先点「下使能」
显示「故障」（橙红色）	驱动器报了故障	点「复位」，等「故障」变回「无故障」，再点「使能」；复位后立刻又报故障，要检查硬件，不要反复重试

附录 A 完整代码 Form1.cs

和配套工程 配套工程/AxisDemo/Form1.cs 一字不差。左边是行号。

```
1 using System.Text;
2 using SCmotion.SoftCard.Sdk;
3 using static SCmotion.SoftCard.Sdk.ScMotionApi;
4
5 namespace AxisDemo
6 {
7     public partial class Form1 : Form
8     {
9         // 连接配置文件：主软件进 OP（或切换到仿真）时自动生成，路径固定
10         const string ConfigPath = @"C:\ProgramData\SoftSC\softcard.runtime.json";
11
12         // 本程序的身份号：自己定一个，多个程序同时连接时要各不相同
13         const int ClientId = 88;
14
15         // 软卡句柄：连接成功后才有值，null 表示还没连接
16         ScMotionCard? card;
17
18         public Form1()
19         {
20             InitializeComponent();
21         }
22
23         // 「连接」按钮：打开连接 → 取得控制权 → 启动定时刷新
24         private void btnConnect_Click(object sender, EventArgs e)
25         {
26             if (card != null) { Log("已经连接了"); return; }
27
28             int rc = SCM_OpenProjectConfig(ConfigPath, ClientId, out card);
29             if (rc != SCM_OK)
30             {
31                 var why = new StringBuilder(1024);
32                 SCM_GetOpenErrorText(why, why.Capacity);
33                 Log($"连接失败，返回码 {rc}: {why}");
34                 card = null;
35                 return;
36             }
37
38             rc = SCM_RequestOwner(card);
39             if (rc != SCM_OK)
40             {
41                 Log($"取控制权失败，返回码 {rc}");
42                 SCM_Close(card);
43                 card = null;
44                 return;
45             }
46
47             tmrPoll.Start();
48             Log("连接成功，已取得控制权");
49         }
50
51         // 「断开」按钮
52         private void btnDisconnect_Click(object sender, EventArgs e)
53         {
54             Disconnect();
55         }
56
57         // 「使能」按钮：返回 0 只表示命令已受理，使能完成要看 powered
```

```

58     private void btnPowerOn_Click(object sender, EventArgs e)
59     {
60         if (card == null) { Log("请先连接"); return; }
61
62         int axis = (int)numAxis.Value;
63         int rc = SCM_AxisPower(card, axis, 1);
64         Log($"使能 轴{axis}, 返回码 {rc}");
65     }
66
67     // 「下使能」按钮
68     private void btnPowerOff_Click(object sender, EventArgs e)
69     {
70         if (card == null) { Log("请先连接"); return; }
71
72         int axis = (int)numAxis.Value;
73         int rc = SCM_AxisPower(card, axis, 0);
74         Log($"下使能 轴{axis}, 返回码 {rc}");
75     }
76
77     // 「复位」按钮：清除轴故障，完成要看 faulted 变回 False
78     private void btnReset_Click(object sender, EventArgs e)
79     {
80         if (card == null) { Log("请先连接"); return; }
81
82         int axis = (int)numAxis.Value;
83         int rc = SCM_AxisReset(card, axis);
84         Log($"复位 轴{axis}, 返回码 {rc}");
85     }
86
87     // 「停止」按钮：普通停止，停下后要重新发运动指令才会再动
88     private void btnStop_Click(object sender, EventArgs e)
89     {
90         if (card == null) { Log("请先连接"); return; }
91
92         int axis = (int)numAxis.Value;
93         int rc = SCM_AxisStop(card, axis);
94         Log($"停止 轴{axis}, 返回码 {rc}");
95     }
96
97     // 定时器：每 200 毫秒读一次轴状态，刷新界面
98     private void tmrPoll_Tick(object sender, EventArgs e)
99     {
100         if (card == null) return;
101
102         SCM_AxisGetStatus(card, (int)numAxis.Value, out SCM_AXIS_STATUS st);
103         lblPowered.Text = st.powered ? "已使能" : "未使能";
104         lblPowered.BackColor = st.powered ? Color.LimeGreen : Color.Gainsboro;
105         lblFaulted.Text = st.faulted ? "故障" : "无故障";
106         lblFaulted.BackColor = st.faulted ? Color.OrangeRed : Color.Gainsboro;
107         lblInfo.Text = $"位置 {st.actualPosition:F3} 速度 {st.actualVelocity:F3}";
108     }
109
110     // 关闭窗口前也断开，把控制权还给主软件
111     private void Form1_FormClosing(object sender, FormClosingEventArgs e)
112     {
113         Disconnect();
114     }
115
116     // 「正向点动」按下：开始正向走
117     private void btnJogPos_MouseDown(object sender, MouseEventArgs e)
118     {
119         StartJog(1);
120     }
121
122

```

```

122      // 「正向点动」松开：收手
123      private void btnJogPos_MouseUp(object sender, MouseEventArgs e)
124      {
125          StopJog();
126      }
127
128      // 「反向点动」按下：开始反向走
129      private void btnJogNeg_MouseDown(object sender, MouseEventArgs e)
130      {
131          StartJog(-1);
132      }
133
134      // 「反向点动」松开：收手
135      private void btnJogNeg_MouseUp(object sender, MouseEventArgs e)
136      {
137          StopJog();
138      }
139
140      // 开始点动：direction 为 1 正向，为 -1 反向
141      private void StartJog(int direction)
142      {
143          if (card == null) { Log("请先连接"); return; }
144
145          int axis = (int)numAxis.Value;
146          double speed = (double)numSpeed.Value;
147          int rc = SCM_AxisJog(card, axis, direction, speed, 0, 0);
148          Log($"点动 轴{axis} 方向{direction} 速度{speed}, 返回码 {rc}");
149      }
150
151      // 点动收手：direction 传 0，轴按减速度停下
152      private void StopJog()
153      {
154          if (card == null) return;
155
156          int axis = (int)numAxis.Value;
157          int rc = SCM_AxisJog(card, axis, 0, 0, 0, 0);
158          Log($"点动收手 轴{axis}, 返回码 {rc}");
159      }
160
161      // 断开：停定时器 → 停轴 → 归还控制权 → 关闭句柄
162      private void Disconnect()
163      {
164          if (card == null) return;
165
166          tmrPoll.Stop();
167          SCM_AxisStop(card, (int)numAxis.Value);
168          SCM_ReleaseOwner(card);
169          SCM_Close(card);
170          card = null;
171          Log("已断开");
172      }
173
174      // 在日志框末尾追加一行，前面带上时间
175      private void Log(string text)
176      {
177          txtLog.AppendText($"{DateTime.Now:HH:mm:ss} {text}\r\n");
178      }
179  }
180 }

```

附录 B 本教程用到的 SDK 指令

所有指令都返回 `int` 返回码，`SCM_OK`（0）表示成功。连接相关的五条（`OpenProjectConfig`、`GetOpenErrorText`、`RequestOwner`、`ReleaseOwner`、`Close`）和教程01 相同，详细说明见《教程01 图文教程》附录 B。下面是轴相关的五条。

SCM_AxisPower

```
int SCM_AxisPower(ScMotionCard handle, int axis, int enable)
```

参数	方向	说明
<code>handle</code>	入	连接得到的句柄
<code>axis</code>	入	轴号，从 0 开始
<code>enable</code>	入	非 0 = 使能（上电）；0 = 下使能（下电）

返回：SCM_OK = 已受理（不是「已使能」）；3 = 总线没进 OP 或没取得控制权；4 = 轴号非法；7 = 被拒

需要控制权。是否完成以 `SCM_AxisGetStatus` 读到的 `powered` 为准：成功最快约 120 毫秒、最长 8.1 秒；失败最长 12.62 秒，所以等待超时不要短于 15 秒。对已经使能的轴再使能一次，会重新走一遍使能流程，期间 `powered` 可能短暂为 0。

SCM_AxisReset

```
int SCM_AxisReset(ScMotionCard handle, int axis)
```

参数	方向	说明
<code>handle</code>	入	连接得到的句柄
<code>axis</code>	入	轴号，从 0 开始

返回：SCM_OK = 已受理（不是「已复位」）；7 = 没有控制权；4 = 轴号非法

需要控制权。是否完成以 `faulted` 变为 `False` 为准，至少约 0.45 秒。复位成功后轴通常处于未使能状态，要再使能一次。复位不能消除故障的原因。

SCM_AxisJog

```
int SCM_AxisJog(ScMotionCard handle, int axis, int direction, double velocity, double acceleration, double deceleration)
```

参数	方向	说明
handle	入	连接得到的句柄
axis	入	轴号，从 0 开始
direction	入	大于 0 正向；小于 0 反向；0 = 收手（按减速度停）
velocity	入	点动速度（轴用户单位/秒）。取绝对值后按 direction 定方向；direction 不为 0 时必须大于 0
acceleration	入	加速度（轴用户单位/秒 ² ），0 = 用轴页默认值
deceleration	入	减速度（轴用户单位/秒 ² ），0 = 用轴页默认值

返回：SCM_OK = 已下发；10 = direction 不为 0 而 velocity 小于等于 0；7 = 没有控制权；4 = 轴号非法

需要控制权。典型用法：按钮按下发 direction = ±1，松开发 direction = 0。松手一定要发 0，否则轴会一直走。

SCM_AxisStop

```
int SCM_AxisStop(ScMotionCard handle, int axis)
```

参数	方向	说明
handle	入	连接得到的句柄
axis	入	轴号，从 0 开始

返回：SCM_OK = 已下发；7 = 没有控制权；4 = 轴号非法

需要控制权。普通停止，让轴停到静止，不置故障。这是最终停：停下来之后要重新下发运动指令才会再动。

SCM_AxisGetStatus

```
int SCM_AxisGetStatus(ScMotionCard handle, int axis, out SCM_AXIS_STATUS status)
```

参数	方向	说明
handle	入	连接得到的句柄
axis	入	轴号，从 0 开始
status	出	轴状态结构体，失败时给零值

返回：SCM_OK = 读到；4 = 轴号非法（出参给零值）；1 = 句柄已关闭

只读，不需要控制权，可以按界面刷新频率轮询。出参给零值时别把 powered = False 当成「轴确实没使能」，先看返回码。

SCM_AXIS_STATUS 的常用字段

字段	类型	含义
powered	bool	已使能。判断 SCM_AxisPower 是否完成
faulted	bool	故障中。判断 SCM_AxisReset 是否完成
actualPosition	double	实际位置（轴用户单位）
actualVelocity	double	实际速度（轴用户单位/秒，带符号，方向即符号）
commandVelocity	double	规划速度：「要它走多快」；actualVelocity 是「它实际多快」
followingError	double	跟随误差（轴用户单位）
homed	bool	已回零
statusWord	ushort	CiA402 状态字（6041）
errorCode	int	驱动器故障码

来源：《SCMotion 4.0 软卡 SDK 编程手册》5.11~5.13、5.18.14。

附录 C 返回码速查

值	常量	含义	本教程里什么时候会遇到
0	SCM_OK	成功	一切正常（使能、复位是「已受理」）
1	SCM_NOT_OPEN	卡未打开	句柄已经关闭后还在读写
2	SCM_ALREADY_OPEN	卡已打开	—
3	SCM_INVALID_STATE	当前状态下该操作无效	取控制权或使能时，主软件没进 OP；使能时没有控制权
4	SCM_INVALID_AXIS	轴号非法	轴号填了不存在的轴
5	SCM_BACKEND_UNAVAILABLE	运行时后端不可用（主软件没起 / 断连）	连接时主软件没运行、SDK 服务没起来，或版本不一致
6	SCM_TIMEOUT	超时	—
7	SCM_REJECTED	被拒绝	控制权在别的程序手上；点动、复位、停止时没有控制权
8	SCM_FAULTED	轴或运行时处于故障态	轴有故障时
9	SCM_NOT_SUPPORTED	该运行时不支持此操作	—
10	SCM_INVALID_ARGUMENT	参数非法	连接时配置文件不存在；点动时速度为 0

来源：《SCMotion 4.0 软卡 SDK 编程手册》5.18.8「错误码 SCM_RESULT」。被拒绝（7）时，可以再调用 `SCM_GetLastReason` 拿到更细的原因。