

案例教程 02 轴使能与点动

在 Qt 程序里让一根轴使能，
按住按钮点动、松开就停，
并实时显示轴的状态。

Qt Creator

C++

Qt 6

MSVC 2022 64 位

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

ABOUT 关于本教程

适合谁	做过教程01 Qt 版（数字量 IO）的人。没做过也能跟：新建项目、加入 SDK、拖控件的步骤这里都写了，但讲得简略些。
做出什么	一个 Qt 窗口程序：连接主软件，让一根轴使能、下使能、复位；按住按钮点动，松开就停；每 200 毫秒刷新一次轴的使能、故障、位置和速度。
这一期的重点	① 受理不等于完成：使能、复位返回 0 只表示命令收下了；② 点动必须收手：按下接 <code>pressed</code> 、松开接 <code>released</code> ，定时器里再加一道保险；③ 发轴命令前再取一次控制权。
开发环境	Qt 6（6.5 以上）· Qt Creator · qmake · MSVC 2022 64 位编译器
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_Qt版.mp4 。本文各章和视频章节一一对应，章标题右边标了视频里的时间。
配套工程	配套工程/AxisDemoQt ：代码和本文附录 A 一字不差，用 Qt Creator 打开 <code>AxisDemoQt.pro</code> 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 ：先在主软件里扫描建轴、进 OP、在轴的「在线」页低速点动过，再来写程序。

怎么用这份教程

- 建议一边看视频一边做：视频看「怎么点、怎么敲」，这份文档查「点哪里、写什么、为什么」。
- 截图里的**红框**就是要看的地方；红色圆点上的数字对应表格里的编号。代码块左边的数字是**文件里的行号**。
- 出了问题先看第 9 章「常见问题」，再按返回码查附录 C。

注意 轴是会动的。第一次上真机：电机和负载尽量脱开，急停按钮放在手边，点动速度先用小值；先在主软件的轴「在线」页里低速点动过、确认方向，再用自己的程序控制。

CONTENTS 目录

第 1 章 最终效果	视频 00:28	4
第 2 章 准备工作	视频 00:56	5
第 3 章 新建项目、加入 SDK	视频 01:47	6
3.1 新建项目		6
3.2 放好 SDK 文件		6
3.3 .pro 里加同样的四行		6
第 4 章 画界面	视频 02:44	7
第 5 章 按钮接到代码	视频 04:00	9
5.1 点动按钮：pressed 和 released		9
第 6 章 写代码 · 头文件	视频 05:06	11
第 7 章 写代码 · mainwindow.cpp	视频 06:06	13
7.1 和教程01 一样的部分		13
7.2 断开：先收手、再停轴		13
7.3 发命令前的检查：ready		13
7.4 使能、下使能、复位		14
7.5 点动		14
7.6 停止		16
7.7 状态刷新：pollStatus		16
第 8 章 编译运行	视频 13:30	18
第 9 章 常见问题	视频 14:32	19
第 10 章 三个要点	视频 15:17	20
附录 A 完整代码		21
AxisDemoQt.pro		21
mainwindow.h		21
mainwindow.cpp		22
附录 B 本教程用到的轴指令		26
附录 C 返回码速查		28

第 1 章 最终效果

视频 00:28



图 1-1 最后做出来的程序（运行中，已经连接并使能）

编号	控件	作用
①	「连接」「断开」	连接主软件并取得控制权；断开时先让点动收手、停轴，再归还控制权
②	轴号、点动速度	两个数字框。轴号从 0 开始（主软件里的「轴 1」是 0 号）；点动速度的单位是主软件轴页里的用户单位每秒，必须大于 0
③	「使能」「下使能」「复位」	发出使能、下使能、清除故障的命令
④	两个状态块	已使能显示绿色「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，松开就停；「停止」是普通停止
⑥	位置和速度	实际位置、实际速度，保留三位小数
⑦	日志框	每一步的返回码都记在这里

第 2 章 准备工作

视频 00:56

- ❶ **机械安全：**电机和负载尽量脱开，急停按钮放在手边，速度先用小值。
- ❷ **先在主软件里点动过：**在轴的「在线」页里低速点动，确认方向是对的。
- ❸ **主软件进 OP，或者切到仿真：**没有伺服就切仿真，仿真轴一样能使能、点动。

后两件都是主软件这边的操作，《上篇 主软件操作》里一步一步做过。Qt、Qt Creator、MSVC 编译器的安装，以及 SDK 包的生成，见教程01 Qt 版第 2 章：要用的还是头文件 `scmotion_card.h`、导入库 `SCmotionSoftCard.lib` 和动态库 `SCmotionSoftCard.dll`。

第 3 章 新建项目、加入 SDK

视频 01:47

3.1 新建项目 视频 01:47

和教程01一样：「文件」→「新建项目」，选「Qt Widgets Application」；名称填 `AxisDemoQt`；构建系统选 `qmake`；构建套件勾 `MSVC2022 64bit`；其他用默认的。

3.2 放好 SDK 文件 视频 02:07

项目文件夹里建一个 `sdk` 文件夹，放头文件和导入库；`SCmotionSoftCard.dll` 直接放在项目文件夹里。

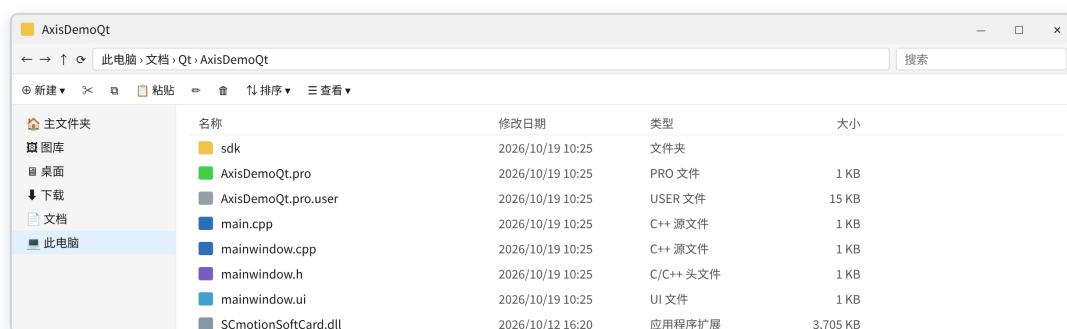


图 3-1 AxisDemoQt 项目文件夹

3.3 .pro 里加同样的四行

```
26 # ----- 软卡 SDK: 下面 4 行是自己加的 -----
27 msvc: QMAKE_CXXFLAGS += /utf-8           # 源码里有中文：用 MSVC 编译时按 UTF-8 读源码
28 INCLUDEPATH += $$PWD/sdk                 # 到 sdk 文件夹找 scmotion_card.h
29 LIBS += -L$$PWD/sdk -lSCmotionSoftCard    # 链接 sdk 文件夹里的 SCmotionSoftCard.lib
30 DESTDIR = $$PWD                          # exe 生成到项目文件夹，和 SCmotionSoftCard.dll 放在一起
```

每一行的意思见教程01 Qt 版 4.2 节：按 UTF-8 读源码，到 `sdk` 找头文件，链接导入库，`exe` 生成到项目文件夹。按 **Ctrl+S** 保存。

第 4 章 画界面

视频 02:44

双击 `mainwindow.ui` 进设计模式。窗口的宽改成 640、高改成 460，`windowTitle` 改成「轴使能与点动演示（Qt）」，再照下表摆控件。拖控件、改 `objectName` 和属性的做法，教程01 Qt 版第 5 章一个一个演示过。

控件盒里的控件	objectName	text / 其他属性	geometry (X, Y, 宽, 高)
Push Button	<code>btnConnect</code> 、 <code>btnDisconnect</code>	连接、断开	X 12 / 110, Y 12; 90 × 30
Label	label (不改)	轴号	12, 54, 30, 26
Spin Box	<code>spinAxis</code>	maximum 63	44, 54, 60, 26
Label	label_2 (不改)	点动速度	120, 54, 56, 26
Double Spin Box	<code>spinSpeed</code>	decimals 1, maximum 10000, value 10	178, 54, 80, 26
Push Button	<code>btnPowerOn</code> 、 <code>btnPowerOff</code> 、 <code>btnReset</code>	使能、下使能、复位	X 12 / 110 / 208, Y 92; 90 × 30
Label	<code>lblPowered</code>	未使能; <code>frameShape</code> Box, <code>alignment</code> 水平居中, <code>styleSheet</code> <code>background: gainsboro;</code>	310, 92, 90, 30
Label	<code>lblFaulted</code>	无故障; 其余同上	408, 92, 90, 30
Push Button	<code>btnJogNeg</code> 、 <code>btnJogPos</code> 、 <code>btnStop</code>	反向点动、正向点动、停止	X 12 / 110 / 208, Y 132; 90 × 30
Label	<code>lblInfo</code>	--	310, 132, 250, 30
Plain Text Edit	<code>txtLog</code>	<code>readOnly</code> 勾上	12, 174, 616, 230

这一期新出现的控件只有一个：点动速度用 **Double Spin Box**，它能填小数。选中 `spinSpeed` 看它的属性：

- **decimals**：小数位数，填 1。
- **maximum**：最大值，填 10000。
- **value**：打开程序时的默认值，填 10。速度的单位和主软件轴页里的用户单位一样，每秒。

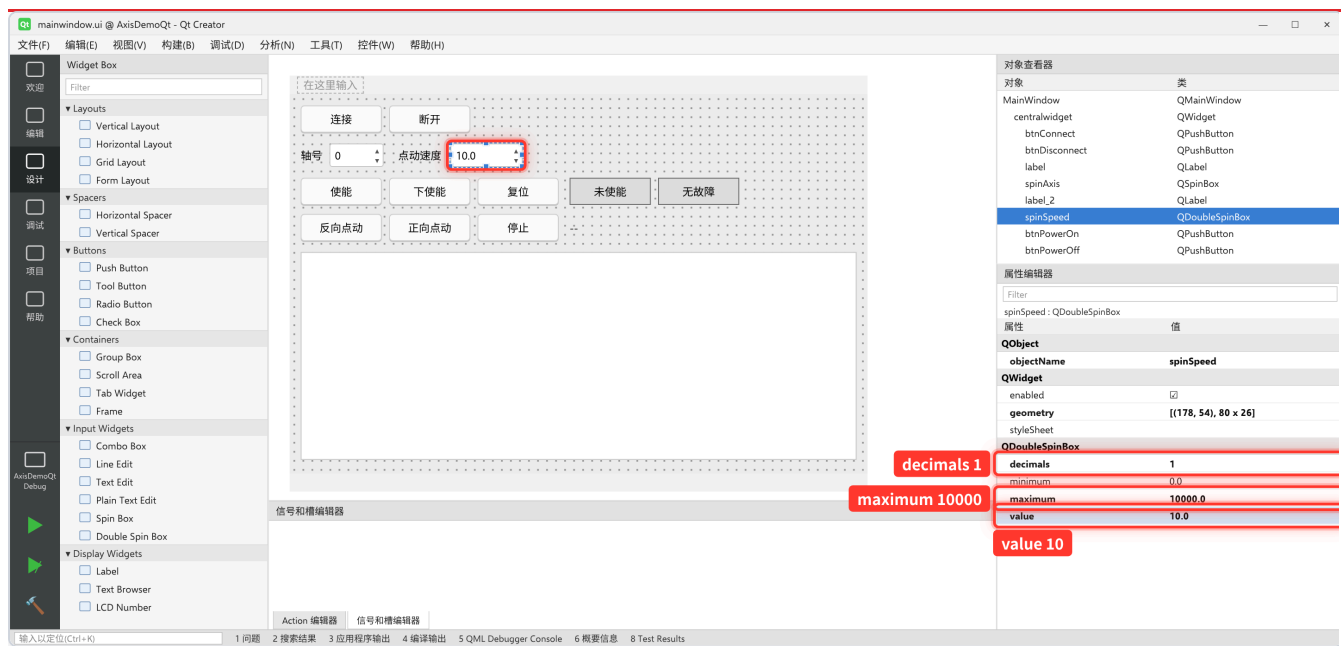


图 4-1 spinSpeed 的属性: decimals、maximum、value

两个状态块 `lblPowered`、`lblFaulted` 跟教程01 的指示灯一样: Box 边框、文字居中、浅灰底色。`lblInfo` 显示位置和速度, 先写两个短横。按 **Ctrl+S** 保存。

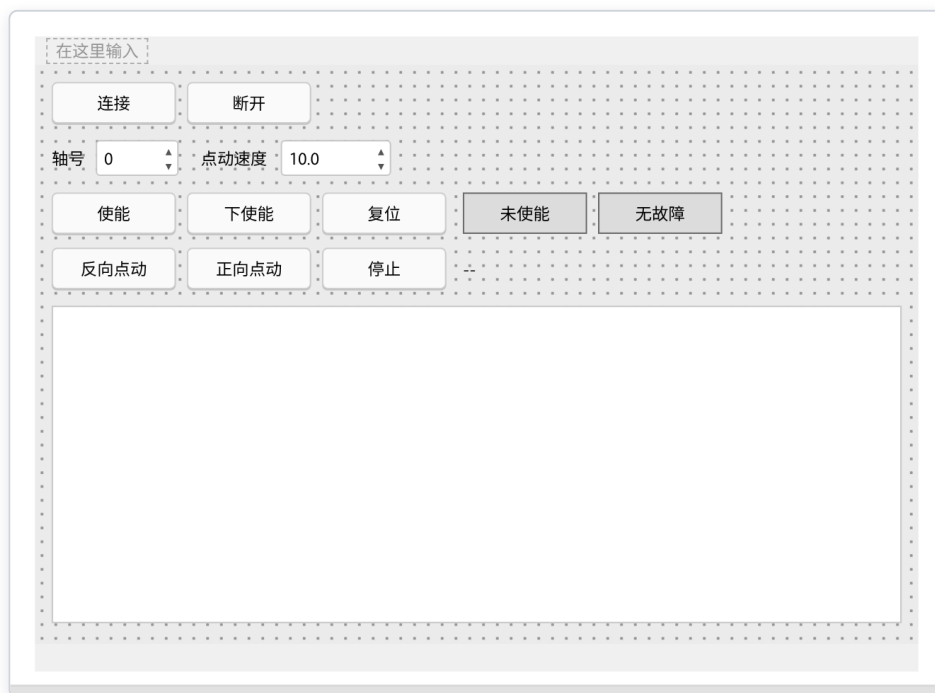


图 4-2 画好的界面

第 5 章 按钮接到代码

视频 04:00

连接、断开、使能、下使能、复位、停止，这六个按钮和教程01 一样：右键，「转到槽...」，选 `clicked()`，确定。

5.1 点动按钮：pressed 和 released

两个点动按钮不能用 `clicked`：clicked 要等鼠标松开才发出来。点动要按下就走、松开就停，所以用另外两个信号：

信号	什么时候发	接到哪个槽
<code>pressed()</code>	鼠标按下的时候	<code>on_btnJogPos_pressed</code> / <code>on_btnJogNeg_pressed</code> ：开始点动
<code>released()</code>	鼠标松开的时候；按住时把鼠标拖出按钮外也会发	<code>on_btnJogPos_released</code> / <code>on_btnJogNeg_released</code> ：收手

- 1 右键「正向点动」，转到槽，选 `pressed()`，确定。
- 2 再右键一次，转到槽，选 `released()`，确定。
- 3 「反向点动」也一样，转到 `pressed()` 和 `released()` 两个槽。

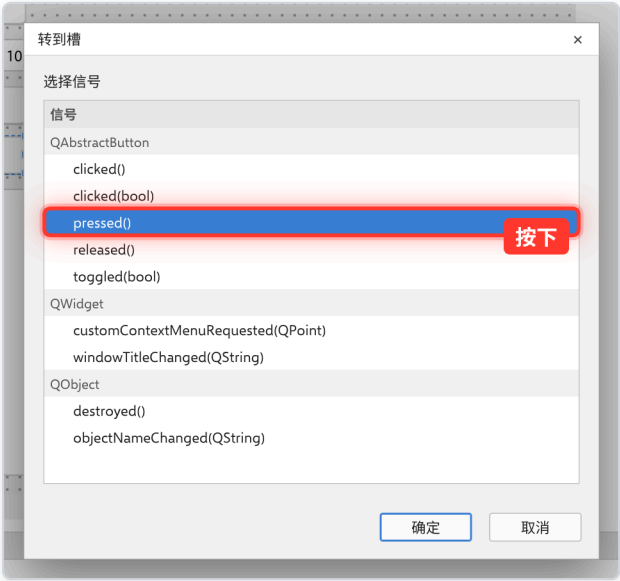
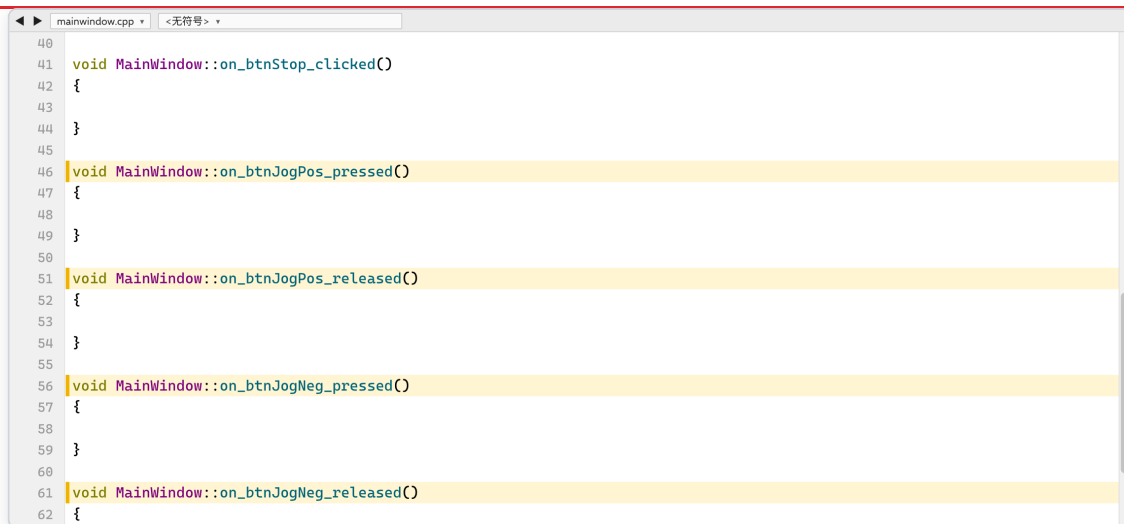


图 5-1 点动按钮选 pressed()（再转一次选 released()）

回到 `mainwindow.cpp`，十个槽函数都生成好了：



```
40
41 void MainWindow::on_btnStop_clicked()
42 {
43 }
44
45
46 void MainWindow::on_btnJogPos_pressed()
47 {
48 }
49
50
51 void MainWindow::on_btnJogPos_released()
52 {
53 }
54
55
56 void MainWindow::on_btnJogNeg_pressed()
57 {
58 }
59
60
61 void MainWindow::on_btnJogNeg_released()
62 {
```

图 5-2 十个槽函数：六个 clicked，四个 pressed / released

第 6 章 写代码 · 头文件

视频 05:06

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QTimer>
6 #include "scmotion_card.h"
7
8 QT_BEGIN_NAMESPACE
9 namespace Ui {
10 class MainWindow;
11 }
12 QT_END_NAMESPACE
13
14 class MainWindow : public QMainWindow
15 {
16     Q_OBJECT
17
18 public:
19     MainWindow(QWidget *parent = nullptr);
20     ~MainWindow();
21
22 private slots:
23     void on_btnConnect_clicked();
24     void on_btnDisconnect_clicked();
25     void on_btnPowerOn_clicked();
26     void on_btnPowerOff_clicked();
27     void on_btnReset_clicked();
28     void on_btnStop_clicked();
29     void on_btnJogPos_pressed();
30     void on_btnJogPos_released();
31     void on_btnJogNeg_pressed();
32     void on_btnJogNeg_released();
33     void pollStatus();
34
35 private:
36     void startJog(int direction);
37     void stopJog();
38     bool ready();
39     void disconnectCard();
40     void log(const QString &text);
41
42     Ui::MainWindow *ui;
43     SCM_CARD_HANDLE card = nullptr; // 软卡句柄, nullptr 表示没连接
44     QTimer timer; // 定时读轴状态
45     int jogAxis = -1; // 正在点动的轴号, -1 表示没在点动
46 };
47 #endif // MAINWINDOW_H
```

行	加了什么	说明
5-6	<code>#include <QTimer></code> 、 <code>#include "scmotion_card.h"</code>	定时器；SDK 的头文件（函数、句柄类型、 <code>SCM_AXIS_STATUS</code> 结构体、返回码都在里面）
23-32	十个槽	转到槽时 Qt Creator 自动加的
33	<code>void pollStatus();</code>	定时器每 200 毫秒调用它，读一次轴状态
36-40	五个自己用的函数	<code>startJog</code> 开始点动、 <code>stopJog</code> 点动收手、 <code>ready</code> 发命令前的检查（返回 <code>bool</code> ，通过返回 <code>true</code> ）、 <code>disconnectCard</code> 断开、 <code>log</code> 写日志
43-44	<code>card</code> 、 <code>timer</code>	句柄和定时器，和教程01 一样
45	<code>int jogAxis = -1;</code>	新加的：记住现在正在点动的是哪根轴，-1 表示没在点动（第 7.5 节讲为什么要它）

第 7 章 写代码 · mainwindow.cpp

视频 06:06

7.1 和教程01 一样的部分 视频 06:06

- 第 1–9 行：include QDateTime、配置文件路径 ConfigPath、身份号 ClientId。
- 第 15–17 行：定时器 200 毫秒，timeout 连到 pollStatus。
- 第 22 行：析构函数里先调用 disconnectCard —— 关窗口时也断开，而且会先停轴，再把控制权还给主软件。
- 第 26–50 行：「连接」按钮；第 194–198 行：log。

7.2 断开：先收手、再停轴 视频 06:54

```
180 // 断开：停定时器 → 点动收手、停轴 → 归还控制权 → 关闭句柄
181 void MainWindow::disconnectCard()
182 {
183     if (card == nullptr) return;
184
185     timer.stop();
186     stopJog();
187     SCM_AxisStop(card, ui->spinAxis->value());
188     SCM_ReleaseOwner(card);
189     SCM_Close(card);
190     card = nullptr;
191     log("已断开");
192 }
```

和教程01 比多了两步（第 186–187 行）：先让点动收手（stopJog），再用 SCM_AxisStop 把轴号框里的那根轴停下来，然后才归还控制权、关闭句柄。因为 SCM_ReleaseOwner 和 SCM_Close 都不会停轴：点动还在走的时候直接断开，轴会一直走下去。「断开」按钮的槽（第 52–56 行）调用它。

7.3 发命令前的检查：ready 视频 07:41

```
166 // 发轴命令之前的检查：连上了没有，再取一次控制权
167 // 使能走完以后主软件会把控制权收回去；控制权本来就在自己手上，这里直接返回 0
168 bool MainWindow::ready()
169 {
170     if (card == nullptr) { log("请先连接"); return false; }
171
172     int rc = SCM_RequestOwner(card);
173     if (rc != SCM_OK) {
174         log(QString("取控制权失败，返回码 %1").arg(rc));
175         return false;
176     }
177     return true;
178 }
```

使能、下使能、复位、点动、停止，发之前都要做同样的检查。最后再调用一次 SCM_RequestOwner：帮助里写着，使能流程走完以后，主软件会把控制权收回去；调试时程序停在断点上超过 5 秒、主软件收不到心跳，也会收回。所以每

次发轴命令之前都再取一次。控制权本来就在自己手上的话，它直接返回 0，没有副作用。取不到就写日志，返回 false。

7.4 使能、下使能、复位 视频 08:29

```

58 // 「使能」按钮: 返回 0 只表示命令收下了, 使能完成要看 powered
59 void MainWindow::on_btnPowerOn_clicked()
60 {
61     if (!ready()) return;
62
63     int axis = ui->spinAxis->value();
64     int rc = SCM_AxisPower(card, axis, 1);
65     log(QString("使能 轴%1, 返回码 %2").arg(axis).arg(rc));
66 }
67
68 // 「下使能」按钮
69 void MainWindow::on_btnPowerOff_clicked()
70 {
71     if (!ready()) return;
72
73     int axis = ui->spinAxis->value();
74     int rc = SCM_AxisPower(card, axis, 0);
75     log(QString("下使能 轴%1, 返回码 %2").arg(axis).arg(rc));
76 }
77
78 // 「复位」按钮: 清除轴故障, 完成要看 faulted 变回 0
79 void MainWindow::on_btnReset_clicked()
80 {
81     if (!ready()) return;
82
83     int axis = ui->spinAxis->value();
84     int rc = SCM_AxisReset(card, axis);
85     log(QString("复位 轴%1, 返回码 %2").arg(axis).arg(rc));
86 }

```

`SCM_AxisPower` 三个参数: 句柄、轴号, 最后一个 1 表示使能、0 表示下使能。`SCM_AxisReset` 清除轴故障。轴号从 `ui->spinAxis->value()` 直接读整数。

注意 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。使能是主软件在后台一步一步做完的: 顺利的话一百多毫秒到几秒, 最坏要十二秒多, 等待的超时不要短于 15 秒。完成没有, 看轴状态里的 `powered`。复位也一样: 完成要看 `faulted` 变回 0, 至少要 0.45 秒。

7.5 点动 视频 09:42

```

142 // 开始点动: direction 为 1 正向, 为 -1 反向
143 void MainWindow::startJog(int direction)
144 {
145     if (!ready()) return;
146
147     int axis = ui->spinAxis->value();
148     double speed = ui->spinSpeed->value();
149     if (speed <= 0) { log("点动速度要大于 0"); return; }

```

```

150
151     int rc = SCM_AxisJog(card, axis, direction, speed, 0, 0);
152     log(QString("点动 轴%1 方向%2 速度%3, 返回码 %4").arg(axis).arg(direction).arg(speed).arg(rc));
153     if (rc == SCM_OK) jogAxis = axis;
154 }
155
156 // 点动收手: direction 传 0, 轴按减速度停下; 没在点动就什么也不做
157 void MainWindow::stopJog()
158 {
159     if (card == nullptr || jogAxis < 0) return;
160
161     int rc = SCM_AxisJog(card, jogAxis, 0, 0, 0, 0);
162     log(QString("点动收手 轴%1, 返回码 %2").arg(jogAxis).arg(rc));
163     jogAxis = -1;
164 }

```

行	做什么
145-149	先调用 ready; 轴号从轴号框读; 点动速度从 spinSpeed 读, Double Spin Box 的 value 直接就是 double。速度不大于 0 就提示并返回: 方向不是 0 的时候, 速度必须大于 0
151	SCM_AxisJog: 句柄、轴号、方向、速度、加速度、减速度。加减速度传 0, 用主软件轴页里的默认值
153	返回 0 的话, 把轴号记进 jogAxis: 这根轴正在点动
159	没连接, 或者 jogAxis 小于 0 (没在点动), 就什么也不做
161	收手还是 SCM_AxisJog, 方向传 0: 轴按减速度停下来, 这不是急停。轴号用 jogAxis: 就算点动中途改了轴号, 收手也发给正在走的那根轴
163	jogAxis 设回 -1: 收手就算被调用两次, 第二次进来已经是 -1, 不会重复发

四个点动的槽: 按下调用 startJog(1) / startJog(-1), 松开调用 stopJog:

```

99 // 「正向点动」按下: 开始正向走
100 void MainWindow::on_btnJogPos_pressed()
101 {
102     startJog(1);
103 }
104
105 // 「正向点动」松开 (拖出按钮外也算松开): 收手
106 void MainWindow::on_btnJogPos_released()
107 {
108     stopJog();
109 }
110
111 // 「反向点动」按下: 开始反向走
112 void MainWindow::on_btnJogNeg_pressed()
113 {
114     startJog(-1);
115 }
116
117 // 「反向点动」松开: 收手
118 void MainWindow::on_btnJogNeg_released()
119 {
120     stopJog();

```

```
121 }
```

注意 点动必须收手。如果只在按下时发命令、松开不发，轴会一直走下去。

7.6 停止 视频 11:34

```
88 // 「停止」按钮：普通停止，停下后要重新发运动指令才会再动
89 void MainWindow::on_btnStop_clicked()
90 {
91     if (!ready()) return;
92
93     jogAxis = -1;
94     int axis = ui->spinAxis->value();
95     int rc = SCM_AxisStop(card, axis);
96     log(QString("停止 轴%1, 返回码 %2").arg(axis).arg(rc));
97 }
```

SCM_AxisStop 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。停止之前把 `jogAxis` 设回 -1，后面松开鼠标就不会再多发一次收手。

7.7 状态刷新：pollStatus 视频 11:53

```
123 // 定时器：每 200 毫秒读一次轴状态，刷新右边的状态块
124 void MainWindow::pollStatus()
125 {
126     // 保险：按住时窗口被切走，Qt 只把按钮弹起、不发 released，这里发现了也要收手
127     if (jogAxis >= 0 && !ui->btnJogPos->isDown() && !ui->btnJogNeg->isDown())
128         stopJog();
129
130     SCM_AXIS_STATUS st = {};
131     if (SCM_AxisGetStatus(card, ui->spinAxis->value(), &st) != SCM_OK) return;
132
133     ui->lblPowered->setText(st.powered ? "已使能" : "未使能");
134     ui->lblPowered->setStyleSheet(st.powered ? "background: limegreen;" : "background: gainsbor
o;");
135     ui->lblFaulted->setText(st.faulted ? "故障" : "无故障");
136     ui->lblFaulted->setStyleSheet(st.faulted ? "background: orangered;" : "background: gainsbor
o;");
137     ui->lblInfo->setText(QString("位置 %1    速度 %2")
138                             .arg(st.actualPosition, 0, 'f', 3)
139                             .arg(st.actualVelocity, 0, 'f', 3));
140 }
```

行	做什么
126-128	保险： 按住点动时，如果窗口被别的窗口切走，Qt 会让按钮弹起来，却不发 released，轴就收不到收手。这里检查：jogAxis 不小于 0（在点动），两个点动按钮却都不是按下状态（ <code>isDown()</code> 都是假），就调用 stopJog 收手
130	<code>SCM_AXIS_STATUS st = {};</code> ：SDK 头文件里定义的结构体， <code>{}</code> 把所有字段先清成 0
131	<code>SCM_AxisGetStatus</code> 最后一个参数传 <code>&st</code> ，状态写进 st。返回码不是 0（比如轴号不存在）就不刷新
133-136	<code>st.powered</code> 不是 0：已使能，背景变绿； <code>st.faulted</code> 不是 0：有故障，背景橙红
137-139	显示实际位置和实际速度。 <code>arg(数, 0, 'f', 3)</code> ：0 表示不限宽度，'f' 按小数显示，3 保留三位小数。一句太长，分成三行写

第 8 章 编译运行

视频 13:30

按 **Ctrl+B** 构建，编译输出最后显示「正常退出」，没有红字。确认主软件已经进了 OP，或者切到了仿真，按 **Ctrl+R** 运行。

```

编译输出
10:30:05: 为项目 AxisDemoQt 执行构建步骤 ...
10:30:05: 正在启动 "C:\Qt\6.8.3\msvc2022_64\bin\qmake.exe" C:\Users\Admin\Documents\Qt\AxisDemoQt\AxisDemoQt.pro -spec win32-msvc "CONFIG+=debug"
10:30:06: 进程"C:\Qt\6.8.3\msvc2022_64\bin\qmake.exe"正常退出。
10:30:06: 正在启动 "C:\Qt\Tools\QtCreator\bin\jom\jom.exe"
cl -c -nologo -Zc:wchar_t -FS ... /utf-8 -Zi -MDd -std:c++17 ... -I.\AxisDemoQt\sdk ... mainwindow.cpp
link /NOLOGO /DEBUG ... /OUT:C:\Users\Admin\Documents\Qt\AxisDemoQt\AxisDemoQt.exe ... /LIBPATH:C:\Users\Admin\Documents\Qt\AxisDemoQt\sdk SCSMotionSoftCard.1
10:30:15: 进程"C:\Qt\Tools\QtCreator\bin\jom\jom.exe"正常退出。
10:30:15: Elapsed time: 00:10.
  
```

图 8-1 编译输出：正常退出

- 1 点「连接」，日志里出现「连接成功，已取得控制权」。
- 2 点「使能」。日志里马上出现返回码 0，但这时状态块还是「未使能」——命令只是被收下了。
- 3 过了一会，状态块变成绿色的「已使能」，这才是使能真正完成。
- 4 按住「正向点动」，轴开始正向走，位置一直在增加；松开，轴减速停下，速度回到 0。
- 5 按住「反向点动」，位置开始减少；松开，同样停下。
- 6 最后点「下使能」，再点「断开」，控制权还给主软件。



图 8-2 ② 返回码 0，但还是「未使能」



图 8-3 ③ 过一会儿才「已使能」



图 8-4 ④ 按住正向点动



图 8-5 ④ 松开：减速停下

第 9 章 常见问题

视频 14:32

现象	原因	怎么办
点了「使能」一直是「未使能」	使能还没走完；轴号不存在（读状态返回 4）；主软件没进 OP	等够 15 秒；核对轴号（主软件里的「轴 1」是 0 号）；看主软件状态
点动返回 7	控制权不在自己手上	发命令前先 SCM_RequestOwner（本例的 ready）
点动返回 10	方向不是 0 时速度是 0	点动速度填大于 0 的数
松开按钮轴还在走	点动按钮接的是 clicked，或者没写 released	按下接 pressed，松开接 released；定时器里的保险别删
位置、速度读数离谱	头文件、lib、dll 不是同一个 SDK 包	三个文件一起换成主软件生成的同一个 SDK 包
编译、链接、找不到 dll、中文乱码	和教程01 Qt 版一样	见教程01 Qt 版第 10 章：检查 .pro 里加的四行

第 10 章 三个要点

视频 15:17

- ❶ **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 `powered`，复位完成看 `faulted`；使能等待的超时不短于 15 秒。
- ❷ **点动必须收手**：`pressed` 开始，`released` 收手（方向传 0）；按钮被意外弹起，定时器里也会收手。
- ❸ **发轴命令前再取一次控制权**；头文件、lib 和 dll 要用主软件生成的同一个 SDK 包。

附录 A 完整代码

和配套工程 配套工程/AxisDemoQt 一字不差。main.cpp 是模板生成的，没改；mainwindow.ui 用设计器打开看。

AxisDemoQt.pro

```

1  QT      += core gui
2
3  greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
4
5  CONFIG += c++17
6
7  # You can make your code fail to compile if it uses deprecated APIs.
8  # In order to do so, uncomment the following line.
9  #DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0
10
11 SOURCES += \
12     main.cpp \
13     mainwindow.cpp
14
15 HEADERS += \
16     mainwindow.h
17
18 FORMS += \
19     mainwindow.ui
20
21 # Default rules for deployment.
22 qnx: target.path = /tmp/${TARGET}/bin
23 else: unix:!android: target.path = /opt/${TARGET}/bin
24 !isEmpty(target.path): INSTALLS += target
25
26 # ----- 软卡 SDK: 下面 4 行是自己加的 -----
27 msvc: QMAKE_CXXFLAGS += /utf-8           # 源码里有中文: 用 MSVC 编译时按 UTF-8 读源码
28 INCLUDEPATH += $$PWD/sdk                # 到 sdk 文件夹找 scmotion_card.h
29 LIBS += -L$$PWD/sdk -LSCmotionSoftCard  # 链接 sdk 文件夹里的 SCmotionSoftCard.lib
30 DESTDIR = $$PWD                         # exe 生成到项目文件夹, 和 SCmotionSoftCard.dll 放在一起

```

mainwindow.h

```

1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include <QTimer>
6  #include "scmotion_card.h"
7
8  QT_BEGIN_NAMESPACE
9  namespace Ui {
10 class MainWindow;
11 }
12 QT_END_NAMESPACE
13
14 class MainWindow : public QMainWindow
15 {
16     Q_OBJECT
17
18 public:

```

```

19     MainWindow(QWidget *parent = nullptr);
20     ~MainWindow();
21
22 private slots:
23     void on_btnConnect_clicked();
24     void on_btnDisconnect_clicked();
25     void on_btnPowerOn_clicked();
26     void on_btnPowerOff_clicked();
27     void on_btnReset_clicked();
28     void on_btnStop_clicked();
29     void on_btnJogPos_pressed();
30     void on_btnJogPos_released();
31     void on_btnJogNeg_pressed();
32     void on_btnJogNeg_released();
33     void pollStatus();
34
35 private:
36     void startJog(int direction);
37     void stopJog();
38     bool ready();
39     void disconnectCard();
40     void log(const QString &text);
41
42     Ui::MainWindow *ui;
43     SCM_CARD_HANDLE card = nullptr;    // 软卡句柄, nullptr 表示没连接
44     QTimer timer;                    // 定时读轴状态
45     int jogAxis = -1;                // 正在点动的轴号, -1 表示没在点动
46 };
47 #endif // MAINWINDOW_H

```

mainwindow.cpp

```

1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3  #include <QDateTime>
4
5  // 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；C++ 字符串里的反斜杠要写两个
6  static const char *ConfigPath = "C:\\ProgramData\\SoftSC\\softcard.runtime.json";
7
8  // 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
9  static const int ClientId = 88;
10
11 MainWindow::MainWindow(QWidget *parent)
12     : QMainWindow(parent)
13     , ui(new Ui::MainWindow)
14 {
15     ui->setupUi(this);
16     timer.setInterval(200);
17     connect(&timer, &QTimer::timeout, this, &MainWindow::pollStatus);
18 }
19
20 MainWindow::~MainWindow()
21 {
22     disconnectCard();    // 关闭窗口时也断开：先停轴，再把控制权还给主软件
23     delete ui;
24 }
25
26 // 「连接」按钮：打开连接 → 取得控制权 → 启动定时器
27 void MainWindow::on_btnConnect_clicked()
28 {
29     if (card != nullptr) { log("已经连接了"); return; }

```

```

30
31     int rc = SCM_OpenProjectConfig(ConfigPath, ClientId, &card);
32     if (rc != SCM_OK) {
33         char why[1024] = {0};
34         SCM_GetOpenErrorText(why, sizeof(why));
35         log(QString("连接失败, 返回码 %1: %2").arg(rc).arg(QString::fromUtf8(why)));
36         card = nullptr;
37         return;
38     }
39
40     rc = SCM_RequestOwner(card);
41     if (rc != SCM_OK) {
42         log(QString("取控制权失败, 返回码 %1").arg(rc));
43         SCM_Close(card);
44         card = nullptr;
45         return;
46     }
47
48     timer.start();
49     log("连接成功, 已取得控制权");
50 }
51
52 // 「断开」按钮
53 void MainWindow::on_btnDisconnect_clicked()
54 {
55     disconnectCard();
56 }
57
58 // 「使能」按钮: 返回 0 只表示命令收下了, 使能完成要看 powered
59 void MainWindow::on_btnPowerOn_clicked()
60 {
61     if (!ready()) return;
62
63     int axis = ui->spinAxis->value();
64     int rc = SCM_AxisPower(card, axis, 1);
65     log(QString("使能 轴%1, 返回码 %2").arg(axis).arg(rc));
66 }
67
68 // 「下使能」按钮
69 void MainWindow::on_btnPowerOff_clicked()
70 {
71     if (!ready()) return;
72
73     int axis = ui->spinAxis->value();
74     int rc = SCM_AxisPower(card, axis, 0);
75     log(QString("下使能 轴%1, 返回码 %2").arg(axis).arg(rc));
76 }
77
78 // 「复位」按钮: 清除轴故障, 完成要看 faulted 变回 0
79 void MainWindow::on_btnReset_clicked()
80 {
81     if (!ready()) return;
82
83     int axis = ui->spinAxis->value();
84     int rc = SCM_AxisReset(card, axis);
85     log(QString("复位 轴%1, 返回码 %2").arg(axis).arg(rc));
86 }
87
88 // 「停止」按钮: 普通停止, 停下后要重新发运动指令才会再动
89 void MainWindow::on_btnStop_clicked()
90 {
91     if (!ready()) return;
92
93     jogAxis = -1;

```

```

94     int axis = ui->spinAxis->value();
95     int rc = SCM_AxisStop(card, axis);
96     log(QString("停止 轴%1, 返回码 %2").arg(axis).arg(rc));
97 }
98
99 // 「正向点动」按下: 开始正向走
100 void MainWindow::on_btnJogPos_pressed()
101 {
102     startJog(1);
103 }
104
105 // 「正向点动」松开 (拖出按钮外也算松开): 收手
106 void MainWindow::on_btnJogPos_released()
107 {
108     stopJog();
109 }
110
111 // 「反向点动」按下: 开始反向走
112 void MainWindow::on_btnJogNeg_pressed()
113 {
114     startJog(-1);
115 }
116
117 // 「反向点动」松开: 收手
118 void MainWindow::on_btnJogNeg_released()
119 {
120     stopJog();
121 }
122
123 // 定时器: 每 200 毫秒读一次轴状态, 刷新右边的状态块
124 void MainWindow::pollStatus()
125 {
126     // 保险: 按住时窗口被切走, Qt 只把按钮弹起、不发 released, 这里发现了也要收手
127     if (jogAxis >= 0 && !ui->btnJogPos->isDown() && !ui->btnJogNeg->isDown())
128         stopJog();
129
130     SCM_AXIS_STATUS st = {};
131     if (SCM_AxisGetStatus(card, ui->spinAxis->value(), &st) != SCM_OK) return;
132
133     ui->lblPowered->setText(st.powered ? "已使能" : "未使能");
134     ui->lblPowered->setStyleSheet(st.powered ? "background: limegreen;" : "background: gainsboro;");
135     ui->lblFaulted->setText(st.faulted ? "故障" : "无故障");
136     ui->lblFaulted->setStyleSheet(st.faulted ? "background: orangered;" : "background: gainsboro;");
137     ui->lblInfo->setText(QString("位置 %1    速度 %2")
138         .arg(st.actualPosition, 0, 'f', 3)
139         .arg(st.actualVelocity, 0, 'f', 3));
140 }
141
142 // 开始点动: direction 为 1 正向, 为 -1 反向
143 void MainWindow::startJog(int direction)
144 {
145     if (!ready()) return;
146
147     int axis = ui->spinAxis->value();
148     double speed = ui->spinSpeed->value();
149     if (speed <= 0) { log("点动速度要大于 0"); return; }
150
151     int rc = SCM_AxisJog(card, axis, direction, speed, 0, 0);
152     log(QString("点动 轴%1 方向%2 速度%3, 返回码 %4").arg(axis).arg(direction).arg(speed).arg(rc));
153     if (rc == SCM_OK) jogAxis = axis;
154 }
155
156 // 点动收手: direction 传 0, 轴按减速度停下; 没在点动就什么也不做
157 void MainWindow::stopJog()

```

```
158 {
159     if (card == nullptr || jogAxis < 0) return;
160
161     int rc = SCM_AxisJog(card, jogAxis, 0, 0, 0, 0);
162     log(QString("点动收手 轴%1, 返回码 %2").arg(jogAxis).arg(rc));
163     jogAxis = -1;
164 }
165
166 // 发轴命令之前的检查：连上了没有，再取一次控制权
167 // 使能走完以后主软件会把控制权收回去；控制权本来就在自己手上，这里直接返回 0
168 bool MainWindow::ready()
169 {
170     if (card == nullptr) { log("请先连接"); return false; }
171
172     int rc = SCM_RequestOwner(card);
173     if (rc != SCM_OK) {
174         log(QString("取控制权失败, 返回码 %1").arg(rc));
175         return false;
176     }
177     return true;
178 }
179
180 // 断开：停定时器 → 点动收手、停轴 → 归还控制权 → 关闭句柄
181 void MainWindow::disconnectCard()
182 {
183     if (card == nullptr) return;
184
185     timer.stop();
186     stopJog();
187     SCM_AxisStop(card, ui->spinAxis->value());
188     SCM_ReleaseOwner(card);
189     SCM_Close(card);
190     card = nullptr;
191     log("已断开");
192 }
193
194 // 在日志框末尾加一行，前面带上时间
195 void MainWindow::log(const QString &text)
196 {
197     ui->txtLog->appendPlainText(QDateTime::currentDateTime().toString("HH:mm:ss ") + text);
198 }
```

附录 B 本教程用到的轴指令

连接、控制权、关闭五条指令见教程01 Qt 版附录 B。下面是 `scmotion_card.h` 里的原型。

SCM_AxisPower

```
int SCM_AxisPower(SCM_CARD_HANDLE handle, int axis, int enable);
```

enable 非 0 使能、0 下使能。**异步受理**：返回码只表示收下了，完成看 SCM_AxisGetStatus 的 powered；使能最坏要十二秒多。需要控制权。

返回：SCM_OK = 已受理；4 = 轴号非法；7 = 没有控制权；3 = 当前状态不允许

SCM_AxisReset

```
int SCM_AxisReset(SCM_CARD_HANDLE handle, int axis);
```

复位轴故障。异步受理，完成看 faulted 转 0，耗时不少于 0.45 秒。需要控制权。

返回：SCM_OK = 已受理；4 / 7 / 3 同上

SCM_AxisStop

```
int SCM_AxisStop(SCM_CARD_HANDLE handle, int axis);
```

普通停止（最终停）：停下后要重新发运动指令才会再动。需要控制权。

返回：SCM_OK = 已下发

SCM_AxisJog

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

direction 大于 0 正向、小于 0 反向、0 收手（按减速度停）。velocity 是点动速度（轴用户单位/秒），direction 不是 0 时必须大于 0。需要控制权。

返回：SCM_OK = 已下发；10 = 参数非法（比如速度为 0）；7 = 没有控制权

SCM_AxisGetStatus

```
int SCM_AxisGetStatus(SCM_CARD_HANDLE handle, int axis, SCM_AXIS_STATUS* status);
```

读轴状态：powered 已使能、faulted 有故障、actualPosition 实际位置、actualVelocity 实际速度等。只读，不需要控制权。

返回：SCM_OK = 正常；4 = 轴号非法（返回非 0 时 status 里的数无意义）

SCM_AXIS_STATUS（头文件里的定义）

```
1 typedef struct SCM_AXIS_STATUS {  
2     int axis;  
3     int state;  
4     int powered;  
5     int faulted;  
6     int homed;  
7     double actualPosition;  
8     double targetPosition;  
9     double actualVelocity;  
10    double followingError;  
11    unsigned short statusWord;  
12    int errorCode;  
13    double commandVelocity;  
14 } SCM_AXIS_STATUS;
```

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：只表示已受理）
1	SCM_NOT_OPEN	卡未打开	句柄已经关闭后还在调用
3	SCM_INVALID_STATE	当前状态下该操作无效	主软件没进 OP，或当前模式不允许下发
4	SCM_INVALID_AXIS	轴号非法	轴号不存在：读状态、使能都会返回 4
5	SCM_BACKEND_UNAVAILABLE	运行时后端不可用	主软件没运行、断连，或 SDK 版本不一致
7	SCM_REJECTED	被拒绝	控制权不在自己手上
8	SCM_FAULTED	轴或运行时处于故障态	轴有故障时发运动指令；先复位
10	SCM_INVALID_ARGUMENT	参数非法	点动方向不是 0 时速度为 0