

CASE TUTORIAL 01 · 编程篇 · Qt 版 (图文版)

案例教程 01 数字量 IO 控制

在 Qt Creator 里从新建项目开始，
用设计器拖界面、用 C 头文件调用 SDK，
写出一个读输入、写输出的 Qt 程序。

Qt Creator

C++

Qt 6

MSVC 2022 64 位

适用软件	SCMotion 4.4.17
配套视频	教程01_数字量IO控制_编程篇_Qt版.mp4
配套工程	配套工程/IoDemoQt
文档版本	V1.0
发布日期	2026-10

ABOUT 关于本教程

适合谁	第一次用软卡 SDK、第一次用 Qt 写程序的人。会装软件、会用 Windows 就行：每一行代码都在这里讲清楚。
做出什么	一个 Qt 窗口程序：连接主软件、取得控制权，写一个数字量输出点（让灯亮、灭），读一个输入点，并且每 200 毫秒自动刷新输入指示灯。界面、按钮、日志的文字和 WinForms 版、WPF 版一样。
和另外两个版本的差别	① 界面在 Qt Creator 的 设计器 里拖出来；② SDK 直接用 C 语言的头文件和导入库，C++ 里包含头文件就能调用，不用再声明一遍；③ 按钮用「转到槽」接到代码。
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01_数字量IO控制_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/IODemoQt ：代码和本文附录 A 一字不差，用 Qt Creator 打开 IoDemoQt.pro 就能编译运行。
建议先看	../教程01_数字量IO控制_上篇_主软件操作.mp4 ：先在主软件里把 IO 测通、进 OP，再来写程序。WinForms / WPF / Qt 三个版本共用这一集。

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**就是要点、要看的地方；红色圆点上的数字对应表格里的编号。
- 「」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**文件里的行号**，和附录 A、配套工程完全对上。
- 出了问题先看第 10 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
Qt 6 和 Qt Creator	用 Qt 在线安装程序安装，组件勾选 MSVC 2022 64-bit （见 2.1）
Visual Studio 2022	提供 MSVC 编译器：安装时勾选「使用 C++ 的桌面开发」。社区版就可以；只装「Visual Studio 生成工具 2022」也行
EtherCAT IO 模块	DI0 接一个按钮，DO0 接一个指示灯。 没有硬件也能学 ：主软件切到「仿真」，连接、读写的流程都能走通。

CONTENTS 目录

第 1 章 最终效果	视频 00:27	4
第 2 章 准备工作	视频 00:55	5
2.1 Qt 6、Qt Creator 和 MSVC 编译器		5
2.2 生成软卡 SDK 包		5
2.3 让主软件处于运行状态		6
第 3 章 新建项目	视频 02:46	7
第 4 章 加入 SDK	视频 04:28	9
4.1 放好三个文件		9
4.2 在 .pro 里加四行		9
第 5 章 画界面	视频 07:22	11
5.1 设计模式		11
5.2 窗口本身		11
5.3 第一个控件		11
5.4 照表摆好其余控件		12
第 6 章 按钮接到代码：转到槽	视频 10:21	14
第 7 章 写代码 · 头文件	视频 11:31	15
第 8 章 写代码 · mainwindow.cpp	视频 13:03	17
8.1 常量、构造函数、析构函数		17
8.2 工具函数 log		18
8.3 「连接」按钮		18
8.4 断开		19
8.5 写输出		19
8.6 读输入		20
8.7 定时刷新		20
第 9 章 编译运行	视频 19:51	22
9.1 构建		22
9.2 运行和验证		22
第 10 章 常见问题	视频 21:09	24
附录 A 完整代码		25
loDemoQt.pro		25
mainwindow.h		25
mainwindow.cpp		26
附录 B 本教程用到的 SDK 指令		29
附录 C 返回码速查		31

第 1 章 最终效果

视频 00:27

先看最后要做出来的程序。界面一共五块：

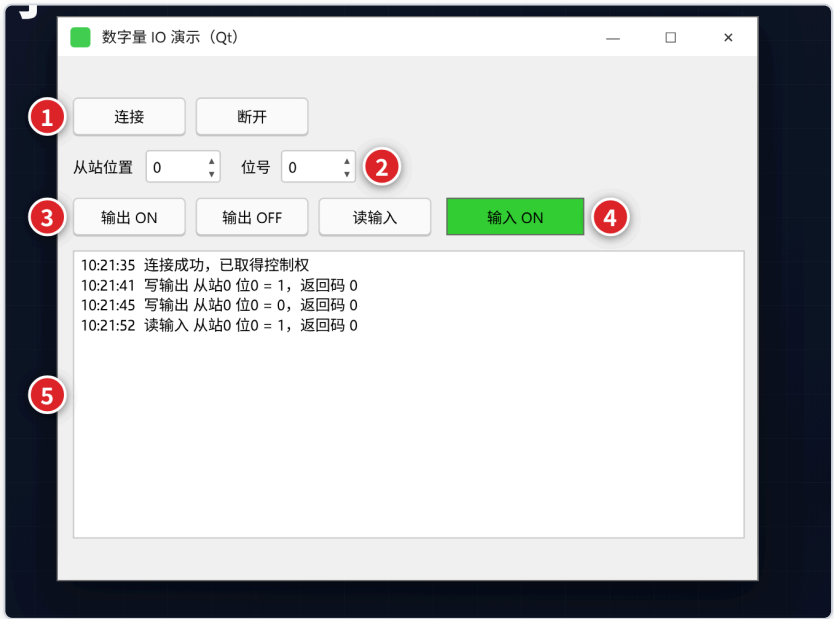


图 1-1 最后做出来的程序（运行中，已经连接）

编号	控件	作用
①	「连接」「断开」	连接主软件并取得控制权；断开时把控制权还给主软件
②	从站位置、位号	两个数字框。从站位置是扫描到的顺序，第一台是 0；位号是这台从站上的第几个点，也从 0 开始
③	「输出 ON」「输出 OFF」「读输入」	给输出点写 1（接通）或写 0（断开）；读一次输入点
④	输入指示灯	每 200 毫秒自动读一次输入：有信号显示绿色「输入 ON」，没有信号显示灰色「输入 OFF」
⑤	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 00:55

2.1 Qt 6、Qt Creator 和 MSVC 编译器

用 Qt 的在线安装程序安装 Qt 6 和 Qt Creator。组件里一定要勾选「MSVC 2022 64-bit」这一套：SDK 的库是 64 位、用微软的 MSVC 编译器编译的，MinGW 那一套链接不上。



图 2-1 Qt 在线安装程序：勾选 MSVC 2022 64-bit

MSVC 编译器本身来自 Visual Studio 2022：安装 Visual Studio 的时候，勾选「使用 C++ 的桌面开发」。只想要编译器的话，装「Visual Studio 生成工具 2022」并勾选同一个工作负荷也可以。

2.2 生成软卡 SDK 包

- 在主软件左侧工程树的「SYSTEM - 配置」下面，点「软卡设置」（图中 ①）。新版本的主软件里，这一页改名叫「SDK包」。
- 点「生成 SDK 包」（图中 ②），选一个输出目录。

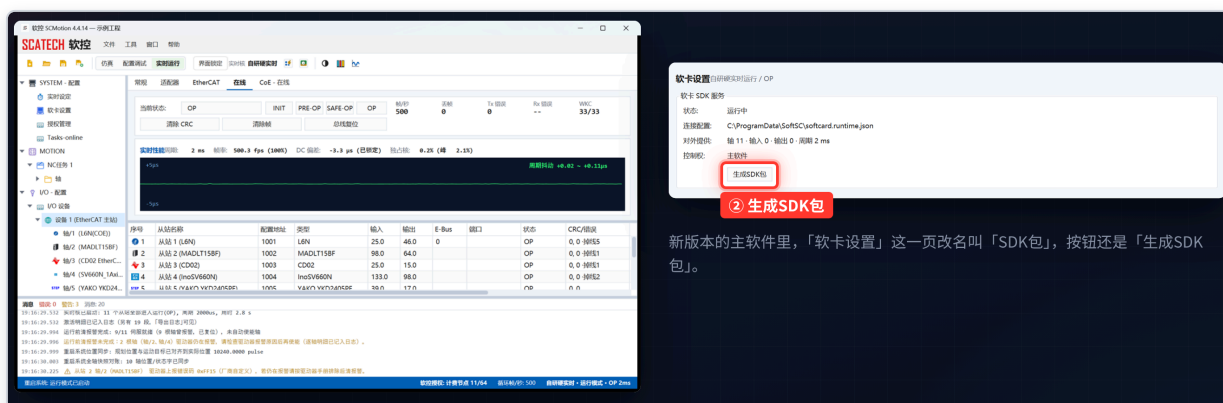


图 2-2 主软件里生成 SDK 包：① 软卡设置 ② 生成 SDK 包

Qt 版要用到 `sdk\c` 下面的三个文件：

```
SCmotionSoftCardSdk_工程名_时间\  
├─ sdk\  
│   └─ dotnet\      (WinForms 版用)  
│       └─ c\  
│           └─ include\scmotion_card.h    ← 头文件：写代码时包含它  
│               └─ win-x64\  
│                   └─ SCmotionSoftCard.lib ← 导入库：链接时用  
│                       └─ SCmotionSoftCard.dll ← 动态库：运行时用  
├─ docs\软卡SDK指令说明.html  
└─ README.txt
```

注意 主软件升级以后，要重新生成一次 SDK 包，三个文件一起换掉。SDK 必须和主软件是同一个版本。

2.3 让主软件处于运行状态

真机点工具栏上的「激活配置参数」进 OP；没有硬件就点模式条上的「仿真」。做法见《上篇 主软件操作》。进 OP（或仿真）以后，主软件会自动写出连接配置文件 `C:\ProgramData\SoftSC\softcard.runtime.json`，程序读它来连接。

第 3 章 新建项目

视频 02:46

- 1 打开 Qt Creator，点菜单「文件」→「新建项目」（**Ctrl+Shift+N**）。
- 2 左边选「Application (Qt)」，中间选「Qt Widgets Application」：带窗口、能用设计器拖控件的桌面程序。点「选择...」。

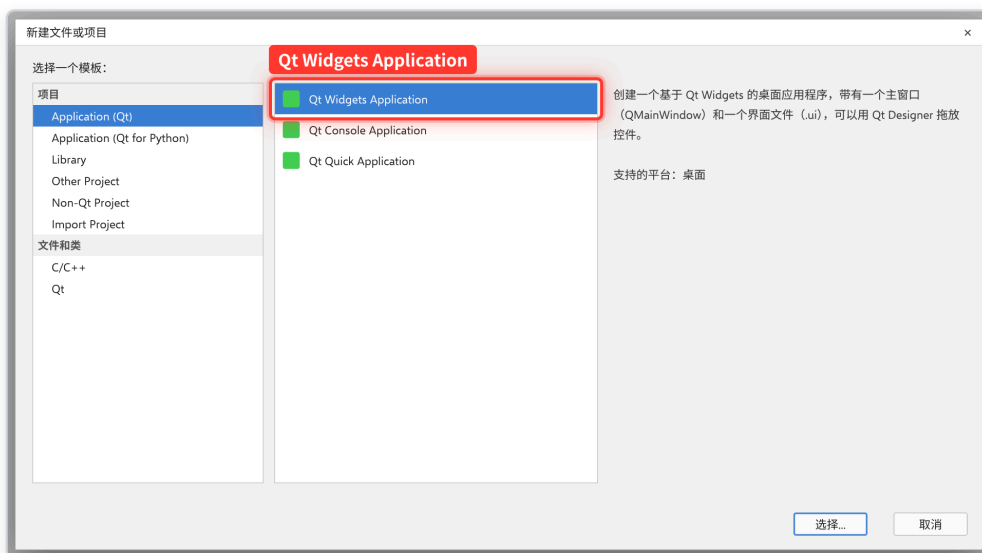


图 3-1 选 Qt Widgets Application

- 3 名称填 `IoDemoQt`，创建路径用默认的，点「下一步」。
- 4 构建系统选 `qmake`。配套工程是 `qmake` 工程，后面加 SDK 只要在 `.pro` 文件里加几行。
- 5 类信息用默认的：类名 `MainWindow`，基类 `QMainWindow`，勾着「创建界面」。翻译文件不用。
- 6 构建套件勾上 `Desktop Qt 6.x MSVC2022 64bit`，一定要选 MSVC 64 位这一套。最后一页点「完成」。



图 3-2 名称 IoDemoQt



图 3-3 构建系统 qmake

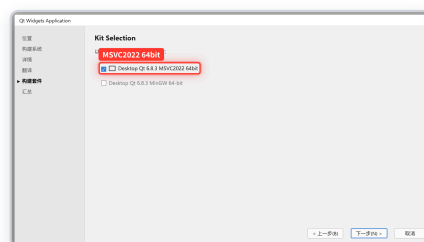


图 3-4 构建套件 MSVC2022 64bit

项目建好了。左边的项目树里，`IoDemoQt.pro` 是项目文件；`mainwindow.ui` 是窗口的界面；`mainwindow.h` 和 `mainwindow.cpp` 是这个窗口的代码；`main.cpp` 是程序入口，不用改。

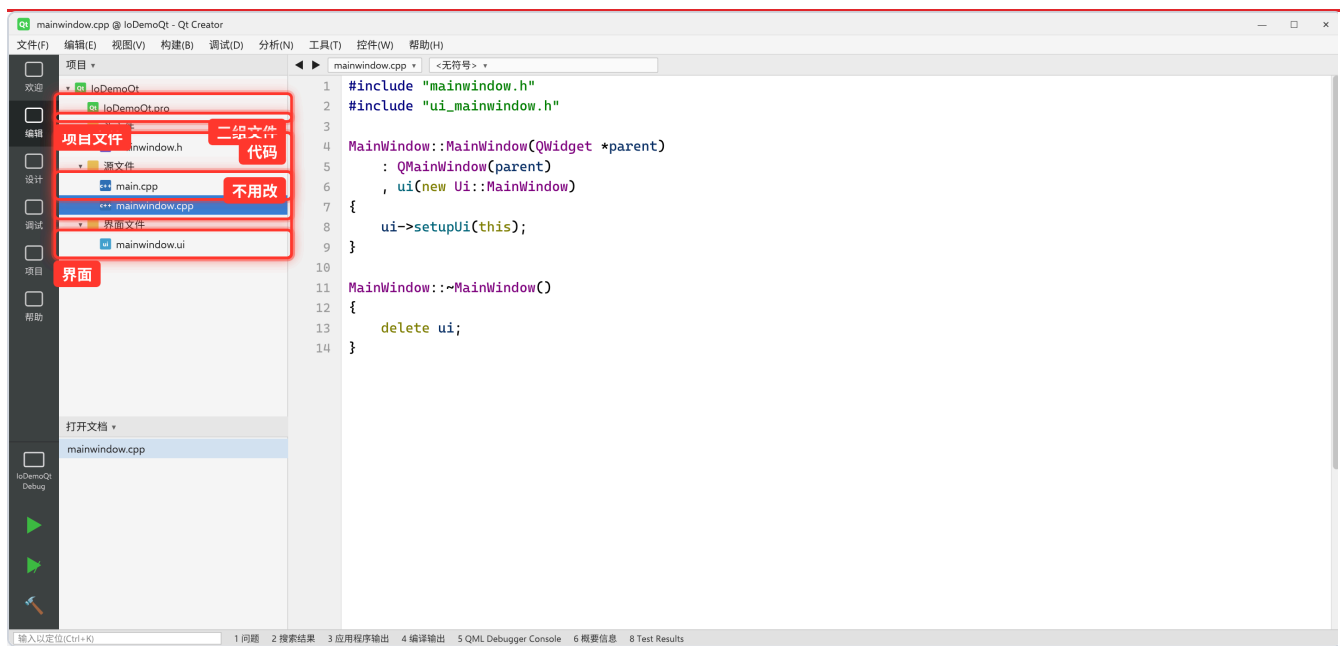


图 3-5 新建好的项目

第 4 章 加入 SDK

视频 04:28

4.1 放好三个文件

- 1 打开项目文件夹 `IoDemoQt`，新建一个文件夹，名字叫 `sdk`。
- 2 把 SDK 包 `sdk\c\include` 里的 `scmotion_card.h`、`sdk\c\win-x64` 里的 `SCmotionSoftCard.lib`，复制到这个 `sdk` 文件夹里。
- 3 把 `SCmotionSoftCard.dll` 直接放在项目文件夹里，和 `.pro` 文件在一起。

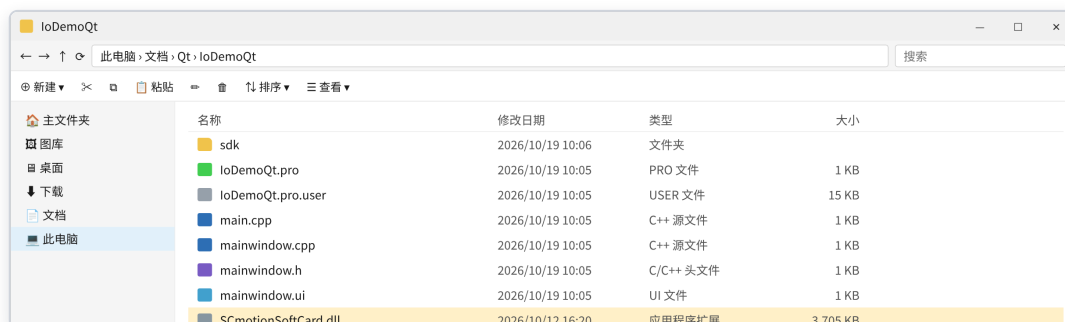


图 4-1 项目文件夹：sdk 文件夹 + SCmotionSoftCard.dll

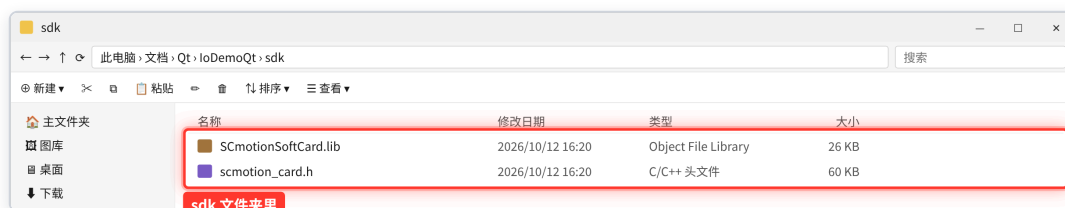


图 4-2 sdk 文件夹里：导入库和头文件

这样放，整个项目拷到别的电脑上也能直接编译。

4.2 在 .pro 里加四行

双击项目树里的 `IoDemoQt.pro` 打开它。这是 qmake 的项目文件：`SOURCES`、`HEADERS`、`FORMS` 列出了项目里的源文件、头文件和界面文件。要告诉 qmake 三件事：头文件在哪、库在哪、程序生成到哪。在文件最后加一行注释和四行：

```
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
```

这一行	做什么
<code>msvc: QMAKE_CXXFLAGS += /utf-8</code>	源码里有中文，MSVC 编译器默认不按 UTF-8 读，日志里的中文会乱码。 <code>msvc:</code> 表示只对 MSVC 生效
<code>INCLUDEPATH += \$\$PWD/sdk</code>	<code>\$\$PWD</code> 是 .pro 文件所在的文件夹。编译器到它下面的 sdk 里找 <code>scmotion_card.h</code>
<code>LIBS += -L\$\$PWD/sdk -lSCmotionSoftCard</code>	大写 <code>-L</code> 跟库所在的文件夹，小写 <code>-l</code> 跟库名（不写 .lib）。链接时用上 <code>SCmotionSoftCard.lib</code>
<code>DESTDIR = \$\$PWD</code>	生成的 exe 直接放进项目文件夹，和 <code>SCmotionSoftCard.dll</code> 在一起，程序一运行就能找到它

按 `Ctrl+S` 保存，Qt Creator 会自动重新读一遍项目文件。



```
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
```

图 4-3 IoDemoQt.pro 最后加的四行

第 5 章 画界面

视频 07:22

5.1 设计模式

双击项目树里的 `mainwindow.ui`，Qt Creator 切到「设计」模式。左边是**控件盒**，中间是正在设计的**窗口**，右上是**对象查看器**，右下是**属性编辑器**：选中谁，就显示谁的属性。

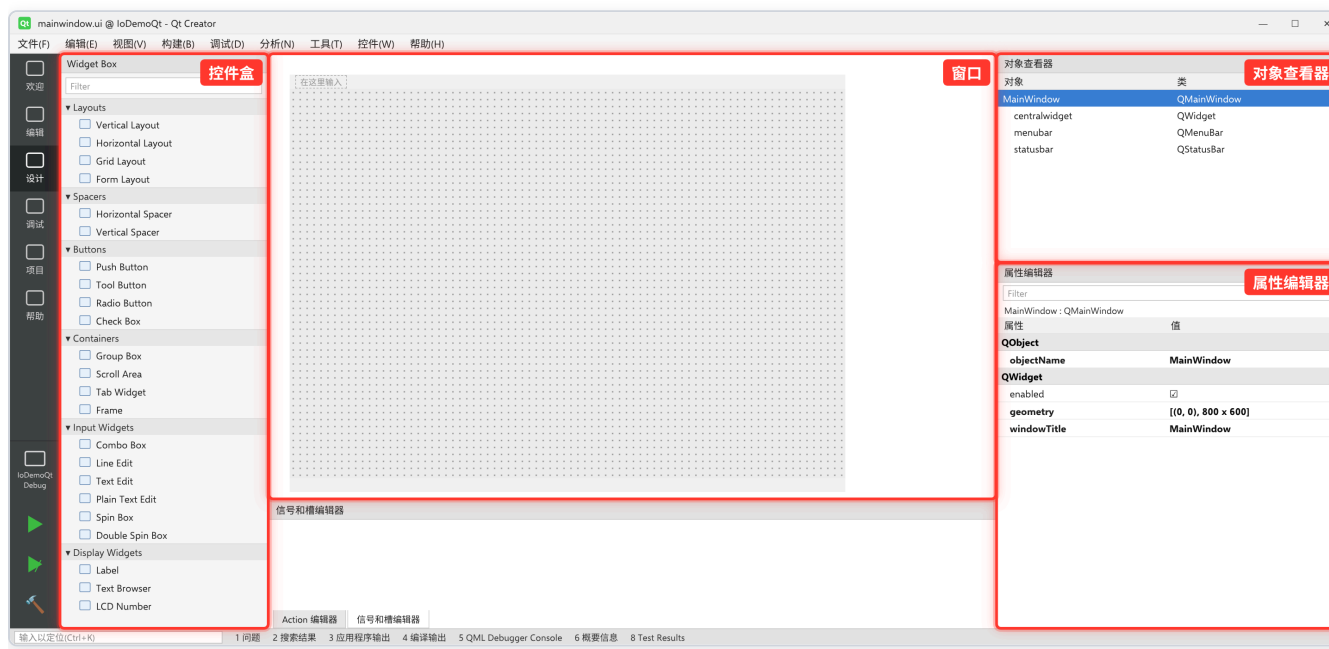


图 5-1 设计模式：控件盒、窗口、对象查看器、属性编辑器

5.2 窗口本身

- 1 点一下窗口的空白处，选中 `MainWindow`。
- 2 属性编辑器里展开「`geometry`」，宽改成 560、高改成 420（像素）。
- 3 「`windowTitle`」改成「数字量 IO 演示（Qt）」，这是窗口标题栏上的文字。

5.3 第一个控件

- 1 在控件盒的「Buttons」里找到「Push Button」，按住鼠标拖到窗口左上角。
- 2 属性编辑器里，「`objectName`」改成 `btnConnect`：这是按钮在代码里的名字，写代码时靠它找到这个按钮。
- 3 「`text`」改成「连接」：按钮上显示的文字。
- 4 「`geometry`」里把位置和大小填准：X 12, Y 12, 宽 90, 高 30。用数字填，比拖出来的整齐。

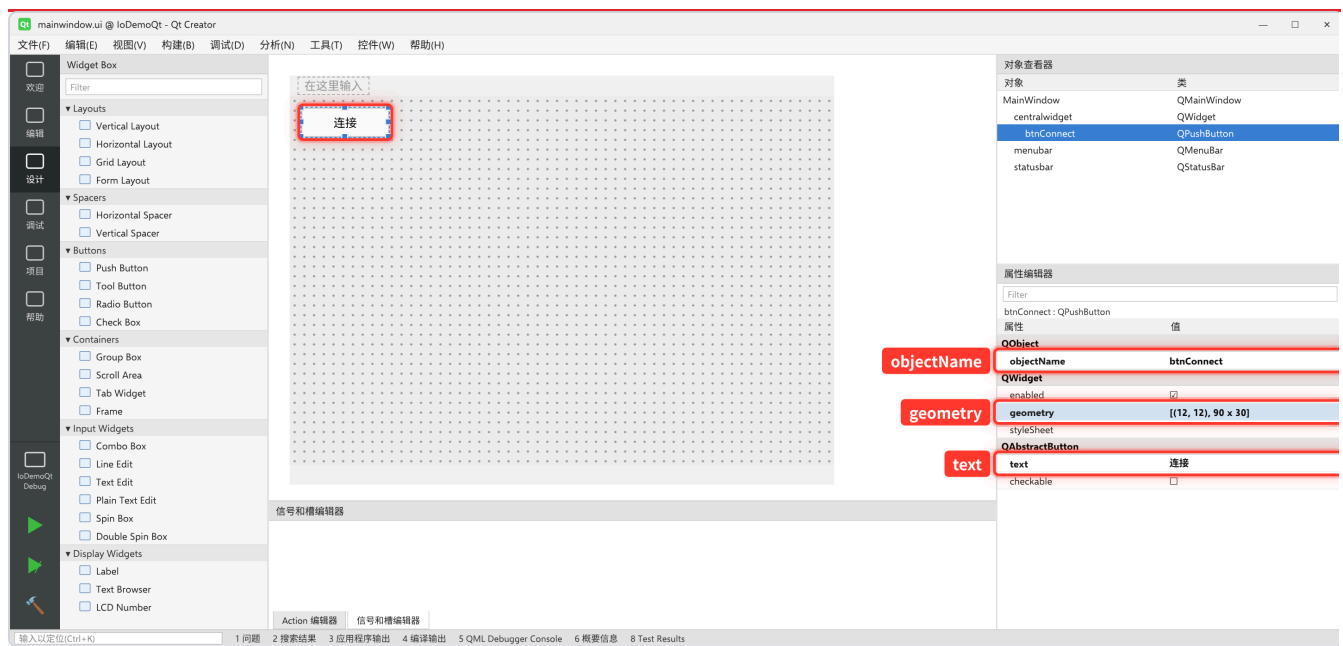


图 5-2 拖进第一个 Push Button，改 objectName、text、geometry

5.4 照表摆好其余控件

一共 11 个控件。按钮、数字框、指示灯和日志框，名字都要和表里一样，后面的代码就是按这些名字写的；两个 Label 只用来显示文字，名字不用改。

控件盒里的控件	objectName	要改的其他属性	geometry (X, Y, 宽, 高)
Push Button	btnConnect	text: 连接	12, 12, 90, 30
Push Button	btnDisconnect	text: 断开	110, 12, 90, 30
Label	label (不改)	text: 从站位置	12, 54, 56, 26
Spin Box	spinSlave	maximum: 255	70, 54, 60, 26
Label	label_2 (不改)	text: 位号	146, 54, 30, 26
Spin Box	spinBit	maximum: 255	178, 54, 60, 26
Push Button	btnOn	text: 输出 ON	12, 92, 90, 30
Push Button	btnOff	text: 输出 OFF	110, 92, 90, 30
Push Button	btnRead	text: 读输入	208, 92, 90, 30
Label	lblDi	text: 输入 --; frameShape: Box; alignment: 水平居中; styleSheet: background: gainsboro;	310, 92, 110, 30
Plain Text Edit	txtLog	readOnly: 勾上	12, 134, 536, 230

- **Spin Box** 是数字框，里面直接就是整数，代码里不用再把文字转成数字。
- **lblDi** 当输入指示灯：frameShape 选 Box 加一圈边框，alignment 选水平居中，styleSheet 填 background: gainsboro; 让背景是浅灰色。

- **txtLog** 勾上 `readOnly`，日志框只读。

界面画完了，按 **Ctrl+S** 保存。



图 5-3 画好的界面

第 6 章 按钮接到代码：转到槽

视频 10:21

按钮点下去，要执行我们写的代码。Qt 用信号和槽把它们连起来：按钮被点击时发出 `clicked()` 信号，接收它的函数叫槽。

- 1 在设计器里右键「连接」按钮，选「转到槽...」。
- 2 弹出的列表是按钮能发出的信号。选 `clicked()`（按钮被点击），点「确定」。



图 6-1 右键 → 「转到槽...」

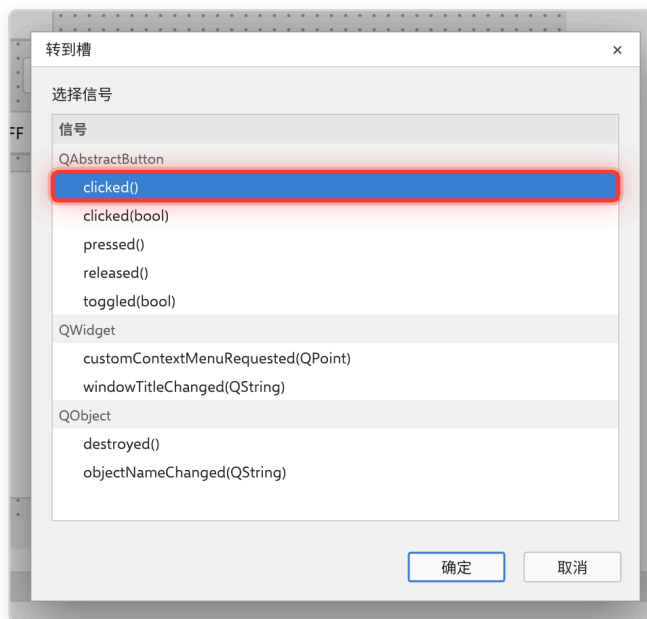


图 6-2 选 clicked()

Qt Creator 自动做了两件事：在 `mainwindow.cpp` 最后加好一个空函数，在 `mainwindow.h` 里加了一段 `private slots` 声明同一个函数。



图 6-3 `mainwindow.cpp` 最后多了 `on_btnConnect_clicked`

提示 函数名是 `on_` + 按钮的 `objectName` + `_` + 信号名。Qt 靠这个名字，自动把按钮的点击和这个函数连在一起。所以 `objectName` 要先改好再转到槽；改名以后要重新转一次。

回到「设计」模式，「断开」、「输出 ON」、「输出 OFF」、「读输入」四个按钮也一样：右键，转到槽，选 `clicked()`，确定。五个按钮就都有了各自的槽函数。

第 7 章 写代码 · 头文件

视频 11:31

先改头文件 `mainwindow.h`，把要用到的东西都声明好。改完的样子：

```
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_btnOn_clicked();
26     void on_btnOff_clicked();
27     void on_btnRead_clicked();
28     void pollInput();
29
30 private:
31     void writeOutput(int value);
32     void disconnectCard();
33     void log(const QString &text);
34
35     Ui::MainWindow *ui;
36     SCM_CARD_HANDLE card = nullptr; // 软卡句柄, nullptr 表示没连接
37     QTimer timer;                 // 定时读输入
38 };
39 #endif // MAINWINDOW_H
```

行	加了什么	为什么
5	<code>#include <QTimer></code>	Qt 的定时器
6	<code>#include "scmotion_card.h"</code>	SDK 的头文件：函数、句柄类型 <code>SCM_CARD_HANDLE</code> 、返回码 <code>SCM_OK</code> 都在里面。尖括号是 Qt 自带的头文件，双引号是自己项目里的
23-27	五个 <code>on_..._clicked</code>	转到槽时 Qt Creator 自动加的
28	<code>void pollInput();</code>	定时器每 200 毫秒调用它一次：读输入、刷新指示灯。它要接收定时器的信号，所以放在 private slots 里
31-33	<code>writeOutput</code> 、 <code>disconnectCard</code> 、 <code>log</code>	自己用的三个函数：写一个输出位、断开、往日志框写一行。不接收信号，放在 private 里。 <code>const QString &</code> 表示传进来的字符串不复制、也不能改
36	<code>SCM_CARD_HANDLE card = nullptr;</code>	软卡句柄。初值 <code>nullptr</code> 表示还没连接，后面靠它判断有没有连接
37	<code>QTimer timer;</code>	定时器

按 `Ctrl+S` 保存。

第 8 章 写代码 · mainwindow.cpp

视频 13:03

8.1 常量、构造函数、析构函数 视频 13:03

```
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::pollInput);
18 }
19
20 MainWindow::~MainWindow()
21 {
22     disconnectCard(); // 关闭窗口时也断开，把控制权还给主软件
23     delete ui;
24 }
```

行	做什么
3	<code>#include <QDateTime></code> ：写日志时取当前时间
6	连接配置文件的路径。 <code>static const char *</code> 是一个只读的 C 字符串， <code>static</code> 表示只在这个文件里用。C++ 字符串里的反斜杠要写两个
9	本程序的身份号，自己定一个数。几个程序同时连主软件时，各自的身份号要不一样
16	定时器间隔 200 毫秒
17	<code>connect</code> 把定时器的 <code>timeout</code> 信号连到 <code>pollInput</code> 。四个参数依次是：谁发信号、发什么信号、谁接收、执行哪个函数
22	析构函数在窗口关闭、对象销毁时执行。先调用 <code>disconnectCard</code> ：直接点叉关掉程序，控制权也会马上还给主软件

8.2 工具函数 log 视频 14:19

```
114 // 在日志框末尾加一行，前面带上时间
115 void MainWindow::log(const QString &text)
116 {
117     ui->txtLog->appendPlainText(QDateTime::currentDateTime().toString("HH:mm:ss ") + text);
118 }
```

ui->txtLog 就是界面上的日志框；appendPlainText 在末尾加一行，会自动换行、自动滚到最后。加的内容是：QDateTime::currentDateTime() 取当前时间，toString("HH:mm:ss ") 按「时:分:秒」变成文字、后面空两格，再加上要写的内容。

8.3 「连接」按钮 视频 14:52

```
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 }
```

行	做什么
29	防重复：card 不是 nullptr，说明已经连上了，写一条日志就返回
31	SCM_OpenProjectConfig 打开连接：配置文件路径、身份号，最后传 &card —— & 是取地址，函数才能把句柄写进 card。返回码存进 rc，0 表示成功
32-38	连接失败：准备 1024 字节的字符数组 why，{0} 先全部填 0；SCM_GetOpenErrorText 取回失败原因（sizeof(why) 是数组的大小）。QString("%1...%2").arg(rc).arg(...) 把 %1、%2 依次换成返回码和原因，原因是 UTF-8 文字，用 QString::fromUtf8 转换。card 设回 nullptr，返回
40	SCM_RequestOwner 取得控制权：刚连上只能读，拿到控制权才能写输出。同一时刻只有一个程序能拿到控制权
41-46	取不到：最常见返回 3（主软件没进 OP）或 7（控制权在别的程序手上）。关掉句柄，返回
48-49	两步都成功：启动定时器，写日志

8.4 断开 视频 16:57

```

102 // 断开：停定时器 → 归还控制权 → 关闭句柄
103 void MainWindow::disconnectCard()
104 {
105     if (card == nullptr) return;
106
107     timer.stop();
108     SCM_ReleaseOwner(card);
109     SCM_Close(card);
110     card = nullptr;
111     log("已断开");
112 }

```

停定时器 → SCM_ReleaseOwner 归还控制权 → SCM_Close 关闭句柄 → card 设回 nullptr。如果不还控制权就直接退出，主软件要等大约 5 秒，才会判定程序失联、把控制权收回去。「断开」按钮的槽只要调用它：

```

52 // 「断开」按钮
53 void MainWindow::on_btnDisconnect_clicked()
54 {
55     disconnectCard();
56 }

```

8.5 写输出 视频 17:34

```

91 // 写一个数字量输出位：value 为 1 接通，为 0 断开
92 void MainWindow::writeOutput(int value)
93 {
94     if (card == nullptr) { log("请先连接"); return; }
95
96     int slave = ui->spinSlave->value();
97     int bit = ui->spinBit->value();
98     int rc = SCM_WriteOutputBit(card, slave, bit, value);
99     log(QString("写输出 从站%1 位%2 = %3, 返回码 %4").arg(slave).arg(bit).arg(value).arg(rc));
100 }

```

- 从站位置和位号从两个数字框里读: `ui->spinSlave->value()` 直接就是整数, 不用转换。
- `SCM_WriteOutputBit` 四个参数: 句柄、从站位置、位号、值。从站位置是扫描到的顺序, 从 0 开始; 位号是这台从站自己的第几个点, 也从 0 开始。
- 日志里四个 %, 依次换成从站位置、位号、值和返回码。

注意 没映射的点, 写它也会返回 0, 只是什么都不会发生。第一次接线一定到现场看灯有没有亮。

「输出 ON」「输出 OFF」的槽分别传 1 和 0:

```
58 // 「输出 ON」按钮
59 void MainWindow::on_btnOn_clicked()
60 {
61     writeOutput(1);
62 }
63
64 // 「输出 OFF」按钮
65 void MainWindow::on_btnOff_clicked()
66 {
67     writeOutput(0);
68 }
```

8.6 读输入 视频 18:32

```
70 // 「读输入」按钮
71 void MainWindow::on_btnRead_clicked()
72 {
73     if (card == nullptr) { log("请先连接"); return; }
74
75     int slave = ui->spinSlave->value();
76     int bit = ui->spinBit->value();
77     int value = 0;
78     int rc = SCM_ReadInputBit(card, slave, bit, &value);
79     log(QString("读输入 从站%1 位%2 = %3, 返回码 %4").arg(slave).arg(bit).arg(value).arg(rc));
80 }
```

准备一个变量 `value`, 初值 0; `SCM_ReadInputBit` 的最后一个参数传 `&value`, 读到的值就写进 `value`。读输入不需要控制权, 连上就能读。没映射的点读回来也是 0, 返回码也是 0: 读到 0 不一定是没有信号, 也可能是这个点没配置。

8.7 定时刷新 视频 19:06

```
82 // 定时器: 每 200 毫秒读一次输入, 刷新右边的指示块
83 void MainWindow::pollInput()
84 {
85     int value = 0;
86     SCM_ReadInputBit(card, ui->spinSlave->value(), ui->spinBit->value(), &value);
87     ui->lblDi->setText(value != 0 ? "输入 ON" : "输入 OFF");
88     ui->lblDi->setStyleSheet(value != 0 ? "background: limegreen;" : "background: gainsboro;");
89 }
```

- 定时器只在连上以后才启动，所以这里不用再检查 card；定时读取的返回码不处理。
- 条件运算符 `value != 0 ? "输入 ON" : "输入 OFF"`：有信号显示「输入 ON」，否则「输入 OFF」。
- `setStyleSheet` 改背景色：有信号绿色 `limegreen`，没信号浅灰 `gainsboro`。

代码全部写完了，按 **Ctrl+S** 保存。

第 9 章 编译运行

视频 19:51

9.1 构建

点左下角的锤子，或者按 **Ctrl+B** 构建。下面的「编译输出」最后显示 `jom.exe` **正常退出**、没有红字，就是编译通过了。link 那一行能看到：生成的 `IoDemoQt.exe` 就在项目文件夹里（`DESTDIR` 的作用），后面还链接了 `SCmotionSoftCard.lib`。

```
编译输出
10:42:05: 为项目 IoDemoQt 执行构建步骤 ...
10:42:05: 正在启动 "C:\Qt\6.8.3\msvc2022_64\bin\qmake.exe" C:\Users\Admin\Documents\Qt\IoDemoQt\IoDemoQt.pro -spec win32-msvc "CONFIG+=debug"
10:42:06: 进程"C:\Qt\6.8.3\msvc2022_64\bin\qmake.exe"正常退出。
10:42:06: 正在启动 "C:\Qt\Tools\QtCreator\bin\jom.exe"
10:42:06: 正在启动 "C:\Qt\Tools\QtCreator\bin\jom.exe"
cl -c -nologo -Zc:wchar_t -FS ... /utf-8 -Zi -MDd -std:c++17 ... -I..IoDemoQt\sdk ... mainwindow.cpp
link /NOLOGO /DEBUG ... /OUT:C:\Users\Admin\Documents\Qt\IoDemoQt\IoDemoQt.exe ... /LIBPATH:C:\Users\Admin\Documents\Qt\IoDemoQt\sdk SCmotionSoftCard.lib ...
10:42:14: 进程"C:\Qt\Tools\QtCreator\bin\jom.exe"正常退出。
10:42:14: Elapsed time: 00:09.
```

图 9-1 编译输出：正常退出

9.2 运行和验证

确认主软件已经进了 OP，或者切到了仿真，然后点左下角绿色的三角，或者按 **Ctrl+R**。

- 1 点「连接」，日志里出现「连接成功，已取得控制权」。
- 2 从站位置和位号都是 0。点「输出 ON」，返回码 0，接在 DO0 上的灯亮了；点「输出 OFF」，灯灭了。
- 3 按住接在 DI0 上的按钮，右边的指示灯马上变绿；点「读输入」，日志里读到 1。松开按钮，指示灯变回灰色。
- 4 最后点「断开」，控制权还给主软件。



图 9-2 ① 连接成功

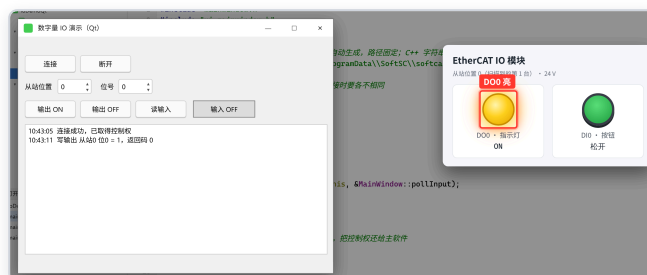


图 9-3 ② 输出 ON：DO0 的灯亮



图 9-4 ③ 按住 DI0 上的按钮：指示灯变绿，「读输入」读到 1

提示 想双击 exe 直接运行（不从 Qt Creator 里运行），要先在 Qt 的命令行里对 exe 跑一次 `windeployqt`，把 Qt 自己的 dll 拷到 exe 旁边。

第 10 章 常见问题

视频 21:09

前四个都和 .pro 里加的那几行有关；返回码的完整含义见附录 C。

现象	原因	怎么办
无法打开包括文件 scmotion_card.h	INCLUDEPATH 没加，或头文件不在 sdk 文件夹	检查 .pro 里 INCLUDEPATH 那一行
无法解析的外部符号 SCM_...	LIBS 没加；sdk 里没有 .lib；或构建 套件不是 MSVC 64 位	检查 LIBS 那一行和 sdk 里的 .lib；套件选 MSVC2022 64bit
一运行就提示找不到 SCmotionSoftCard.dll	exe 旁边没有这个 dll	.pro 里加 DESTDIR = \$\$PWD，dll 放在项目文件夹
日志里的中文是乱码	MSVC 没按 UTF-8 读源码	加上 msvc: QMAKE_CXXFLAGS += /utf-8，重新构建
双击 exe 提示缺 Qt6Core.dll	Qt 自己的 dll 不在 exe 旁边	在 Qt Creator 里运行；或用 windeployqt 拷过去
点按钮没有反应	槽函数名和按钮名对不上（先转到 槽、后改的 objectName）	按 on_按钮名_clicked 核对；改名后重新转到槽
连接失败，返回码 10 / 5	主软件没进 OP 也没切仿真；或 SDK 和主软件版本不一致	看「软卡设置」（SDK包）页状态；必要时重新生成 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；先断开另一个程序，或等 5 秒

附录 A 完整代码

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

loDemoQt.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_btnOn_clicked();
26     void on_btnOff_clicked();
27     void on_btnRead_clicked();
28     void pollInput();
29
30 private:
31     void writeOutput(int value);
32     void disconnectCard();
33     void log(const QString &text);
34
35     Ui::MainWindow *ui;
36     SCM_CARD_HANDLE card = nullptr;    // 软卡句柄, nullptr 表示没连接
37     QTimer timer;                    // 定时读输入
38 };
39 #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::pollInput);
18 }
19
20 MainWindow::~MainWindow()
21 {
22     disconnectCard();    // 关闭窗口时也断开，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 }

```

```

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 // 「输出 ON」按钮
59 void MainWindow::on_btnOn_clicked()
60 {
61     writeOutput(1);
62 }
63
64 // 「输出 OFF」按钮
65 void MainWindow::on_btnOff_clicked()
66 {
67     writeOutput(0);
68 }
69
70 // 「读输入」按钮
71 void MainWindow::on_btnRead_clicked()
72 {
73     if (card == nullptr) { log("请先连接"); return; }
74
75     int slave = ui->spinSlave->value();
76     int bit = ui->spinBit->value();
77     int value = 0;
78     int rc = SCM_ReadInputBit(card, slave, bit, &value);
79     log(QString("读输入 从站%1 位%2 = %3, 返回码 %4").arg(slave).arg(bit).arg(value).arg(rc));
80 }
81
82 // 定时器: 每 200 毫秒读一次输入, 刷新右边的指示块
83 void MainWindow::pollInput()
84 {
85     int value = 0;
86     SCM_ReadInputBit(card, ui->spinSlave->value(), ui->spinBit->value(), &value);
87     ui->lblDi->setText(value != 0 ? "输入 ON" : "输入 OFF");
88     ui->lblDi->setStyleSheet(value != 0 ? "background: limegreen;" : "background: gainsboro;");
89 }
90
91 // 写一个数字量输出位: value 为 1 接通, 为 0 断开
92 void MainWindow::writeOutput(int value)
93 {
94     if (card == nullptr) { log("请先连接"); return; }
95
96     int slave = ui->spinSlave->value();
97     int bit = ui->spinBit->value();
98     int rc = SCM_WriteOutputBit(card, slave, bit, value);
99     log(QString("写输出 从站%1 位%2 = %3, 返回码 %4").arg(slave).arg(bit).arg(value).arg(rc));
100 }
101

```

```
102 // 断开：停定时器 → 归还控制权 → 关闭句柄
103 void MainWindow::disconnectCard()
104 {
105     if (card == nullptr) return;
106
107     timer.stop();
108     SCM_ReleaseOwner(card);
109     SCM_Close(card);
110     card = nullptr;
111     log("已断开");
112 }
113
114 // 在日志框末尾加一行，前面带上时间
115 void MainWindow::log(const QString &text)
116 {
117     ui->txtLog->appendPlainText(QDateTime::currentDateTime().toString("HH:mm:ss ") + text);
118 }
```

附录 B 本教程用到的 SDK 指令

下面是头文件 `scmotion_card.h` 里的原型。所有指令都返回 `int` 返回码，`SCM_OK`（0）表示成功。

SCM_OpenProjectConfig

```
int SCM_OpenProjectConfig(const char* configPath, int clientId, SCM_CARD_HANDLE* handle);
```

打开连接。configPath 是连接配置文件路径（UTF-8）；clientId 是本程序的身份号；成功时 handle 给回句柄。

返回：SCM_OK = 已连接；10 = 配置文件不存在、读不通或内容非法；5 = 主软件没起，或运行时不可用

SCM_GetOpenErrorText

```
int SCM_GetOpenErrorText(char* buffer, int bufferSize);
```

读最近一次连接失败的详细原因（UTF-8 文字），不需要句柄。

返回：SCM_OK = 取到

SCM_RequestOwner

```
int SCM_RequestOwner(SCM_CARD_HANDLE handle);
```

取得控制权。取得后主软件界面上的输出和运动操作变成只读；程序没归还就退出时，主软件大约 5 秒后自动收回。

返回：SCM_OK = 已取得；7 = 控制权在别人手上；3 = 主软件没进 OP 或当前模式不允许下发；5 = 主软件不可用

SCM_ReleaseOwner

```
int SCM_ReleaseOwner(SCM_CARD_HANDLE handle);
```

归还控制权，主软件立即恢复控制。

返回：SCM_OK = 已归还；7 = 你不是当前的控制权持有者

SCM_Close

```
int SCM_Close(SCM_CARD_HANDLE handle);
```

关闭句柄。关闭前如果还持有控制权，先调用 SCM_ReleaseOwner。

返回：SCM_OK

SCM_WriteOutputBit

```
int SCM_WriteOutputBit(SCM_CARD_HANDLE handle, int slavePosition, int bit, int value);
```

写一个数字量输出位。slavePosition 是从站拓扑位置（扫描顺序，从 0 开始），bit 是输出位号（从 0 开始），value 非 0 接通、0 断开。需要控制权。

返回：SCM_OK = 已下发（没映射时同样返回 SCM_OK）；7 = 没有控制权；5 = 主软件不可用

SCM_ReadInputBit

```
int SCM_ReadInputBit(SCM_CARD_HANDLE handle, int slavePosition, int bit, int* value);
```

读一个数字量输入位，value 给回 0 或 1（没映射或失败时给 0）。只读，不需要控制权。

返回：SCM_OK = 正常（包括「这个点没映射」）；1 = 句柄已关闭；5 = 主软件不可用

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
1	SCM_NOT_OPEN	卡未打开	句柄已经关闭后还在读写
3	SCM_INVALID_STATE	当前状态下该操作无效	取控制权时主软件没进 OP，或当前模式不允许下发
5	SCM_BACKEND_UNAVAILABLE	运行时后端不可用	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；写输出时没有控制权
10	SCM_INVALID_ARGUMENT	参数非法	连接时配置文件不存在或读不了