

KLinx DAQ

C# WinForms 应用程序 · 接口与开发文档

`namespace KlinxApp.WinForms`

平台: Windows · .NET 6+

驱动: klinx_api.dll (C cdecl)

参考实现: Qt/C++ MainWindow

1. 程序概述

KLinx DAQ WinForms 是对原 Qt/C++ MainWindow 的 C# 等价移植。程序通过 P/Invoke 调用 klinx_api.dll 驱动库，完成 DAQ 设备的参数配置、采集控制与波形显示。整体逻辑、数据流与 C++ 版本保持一一对应。

1.1 整体架构

模块	C++ 对应	说明
MainForm	MainWindow	主窗口，UI 构建与事件处理
KlinxApi (static)	klinx_api.h P/Invoke	驱动函数的 C# 声明，cdecl 调用约定
WaveformPlotControl	QCustomPlot	GDI+ 自绘波形控件，多通道分栏显示
StatusStateControl	StatusForm	状态栏 LED + 文字指示器
LedIndicatorControl	StatusForm 内部	自绘圆形 LED，支持 Off / On / Blink
ChannelSetting	channel_settings_t	单通道配置：ID、使能、量程

1.2 关键设计决策

原生堆缓冲区（Marshal.AllocHGlobal）

采集缓冲区使用 Marshal.AllocHGlobal 分配，等价于 C++ HeapAlloc(GetProcessHeap())。原因：klinx_api 驱动在首次 klinx_daq_read 时向内核注册 DMA scatter-gather 表并锁定物理页，后续每次读取必须写入同一物理地址。使用托管堆 + GCHandle.Pinned 存在页属性差异，可能导致 DMA 注册失败或数据写入错位，表现为波形跑马灯。原生堆与 C++ std::vector::data() 完全等价，彻底消除此问题。

缓冲区在 Read(false) 时不释放（与 C++ startRead(false) 仅 stop timer 一致），在 Read(true) 时覆盖分配，在 ReleaseDevice() 时最终释放。

定时器精度（timeBeginPeriod）

WinForms.Timer 默认分辨率约 15ms（Windows 系统时钟周期）。程序在 Read(true) 时调用 timeBeginPeriod(1) 将系统时钟提升至 1ms，使 5ms 定时器与 C++ QTimer::setInterval(5) 精度等价。Read(false) 时调用 timeEndPeriod(1) 恢复。

1.3 状态机

设备状态由驱动维护，C# 通过 klinx_daq_state() 查询：

状态	值	含义	LED
DAQ_CLOSED	0	设备未打开	红色 Off
DAQ_OPEN	1	已打开，未建立连接	—
DAQ_ESTABLISHED	2	连接建立，可以 Start	绿色 On
DAQ_WORKING	3	正在采集	绿色 Blink
DAQ_STOPPED	4	已停止，可以重新 Start	黄色 On

1.4 数据流

Read(true) 分配缓冲区并启动 timer → 每 5ms OnTick() 调用 klinx_daq_read() 写入原生缓冲区 → Marshal.Copy 复制到托管 adCopy[] → UpdatePlot() 解交织并刷新 WaveformPlotControl。

步骤	C++ 对应	说明
AllocAdBuf(sz)	make_unique<vector>(sz)	原生堆分配, sz = channels × np × batch
klinx_daq_read(adPtr, adSize)	klinx_daq_read(data(), size())	驱动 DMA 写入, 固定地址
Marshal.Copy(adPtr, adCopy)	直接访问 data()	原生→托管一次 memcpy
解交织循环	updatePlot 内循环	adCopy[n*ch+i] → plotBufs[i][n]
WaveformPlotControl.Invalidate()	plot->replot()	触发 GDI+ 重绘

2. P/Invoke 接口（KlinxApi）

所有接口声明在静态类 KlinxApi 中，DLL 名称 klinx_api.dll，调用约定 CallingConvention.Cdecl。

2.1 枚举类型

KlinxStatus — 返回值状态码

C# 枚举值	整数值	含义
DAQ_OK	0	操作成功
DAQ_ERR_CONTEXT	-1	无效的设备上下文
DAQ_ERR_INVALID	-2	无效参数
DAQ_ERR_HW	-3	硬件错误
DAQ_INVALID_STATE	-4	状态不允许此操作
DAQ_FAILED_OPEN_C2H	-5	打开 C2H 通道失败
DAQ_FAILED_OPEN_USER	-6	打开 USER 通道失败
DAQ_FAILED_SETUP	-7	Setup 失败
DAQ_FAILED_READ	-8	读取失败

KlinxState — 设备状态

C# 枚举值	整数值	含义
DAQ_CLOSED	0	已关闭
DAQ_OPEN	1	已打开
DAQ_ESTABLISHED	2	连接建立
DAQ_WORKING	3	采集中
DAQ_STOPPED	4	已停止

KlinxTrigMode — 触发模式

C# 枚举值	整数值	含义
DAQ_TRIG_HOST	0	主机控制模式（连续采集）
DAQ_TRIG_SINGLE	1	外部单次触发
DAQ_TRIG_MULTIPLE	2	外部多次触发
DAQ_EXT_TRIG_HOST_STOP	3	外部触发、主机停止

KlinxTrigEdge — 触发边沿

C# 枚举值	整数值	含义
FALLING	0	下降沿触发
RISING	1	上升沿触发

KlinxTrigPole — 极性

C# 枚举值	整数值	含义
LEVEL_HIGH	0	高电平
LEVEL_LOW	1	低电平

KlinxRangeFlag — ADC 输入量程

C# 枚举值	整数值	含义
ADC_RANGE_FLAG_1V	1	±1 V 峰峰值
ADC_RANGE_FLAG_2V	2	±2 V 峰峰值
ADC_RANGE_FLAG_4V	4	±4 V 峰峰值
ADC_RANGE_FLAG_6V	6	±6 V 峰峰值

2.2 生命周期接口

C# 声明	C 原型	说明
IntPtr klinx_daq_create(int idx)	device_handle_t klinx_daq_create(int device_index)	创建设备上下文。idx=0 对应第一个设备。返回 IntPtr.Zero 表示失败。
KlinxStatus klinx_daq_open(IntPtr ctx)	daq_status_t klinx_daq_open(device_handle_t ctx)	打开设备，状态变为 DAQ_OPEN。
void klinx_daq_close(IntPtr ctx)	void klinx_daq_close(device_handle_t ctx)	关闭设备，状态回到 DAQ_CLOSED。
KlinxState klinx_daq_state(IntPtr ctx)	state_t klinx_daq_state(device_handle_t ctx)	查询当前设备状态，可传 IntPtr.Zero（返回 DAQ_CLOSED）。

2.3 控制接口

C# 声明	说明
KlinxStatus klinx_daq_setup(IntPtr ctx)	应用参数配置，状态变为 DAQ_ESTABLISHED。必须在 open 后、start 前调用。
KlinxStatus klinx_daq_start(IntPtr ctx)	启动采集，状态变为 DAQ_WORKING。
KlinxStatus klinx_daq_stop(IntPtr ctx)	停止采集，状态变为 DAQ_STOPPED。
KlinxStatus klinx_daq_clear(IntPtr ctx)	清除设备状态，用于 close 前的清理。

2.4 触发输入参数

C# 声明	C 参数类型	说明
klinx_daq_set_trig_in_mode(IntPtr ctx, KlinxTrigMode m)	daq_trig_mode_t	设置触发模式
klinx_daq_set_trig_in_edge(IntPtr ctx, KlinxTrigEdge e)	daq_trig_edge_t	设置触发边沿
klinx_daq_set_trig_in_pole(IntPtr ctx, KlinxTrigPole p)	daq_trig_pole_t	设置触发极性
klinx_daq_set_sample_point_count_per_trig(IntPtr ctx, ulong np)	uint64_t	每次触发采样点数 (UI: 1 ~ 250,000,000, 默认 12500)
klinx_daq_set_sample_trig_times(IntPtr ctx, ulong t)	uint64_t	触发次数 (UI: 1 ~ 99999)
klinx_daq_set_sample_delay_nclocks_after_trig(IntPtr ctx, ulong n)	uint64_t	触发后延迟时钟数 (UI: 0 ~ 99999)
klinx_daq_set_scan_rate(IntPtr ctx, ulong r)	uint64_t	主机模式扫描速率 Hz (UI: 1 ~ 50000, 默认 10000)

2.5 ADC 通道参数

C# 声明	C 参数类型	说明
<code>klinx_daq_set_adc_enable(IntPtr ctx, uint ch, bool en)</code>	<code>uint32_t ch, bool en</code>	使能/禁用指定通道。bool 以 MarshalAs(UnmanagedType.I1) 传递 (1 字节)。
<code>klinx_daq_set_adc_in_attenuator(IntPtr ctx, ulong ch, KlinxRangeFlag r)</code>	<code>uint64_t ch, adc_range_flag_t range</code>	设置指定通道输入量程。
<code>int klinx_daq_ad_channel_count(IntPtr ctx)</code>	<code>int</code>	返回设备总 ADC 通道数。

2.6 触发输出参数

C# 声明	C 参数类型	说明
<code>klinx_daq_set_trig_out_pole(IntPtr ctx, KlinxTrigPole p)</code>	<code>daq_trig_pole_t</code>	设置触发输出极性
<code>klinx_daq_set_trig_out_pulse_width(IntPtr ctx, int w)</code>	<code>int32_t</code>	设置脉冲宽度 (时钟数)。UI: 0 ~ 1,000,000, 默认 25。
<code>klinx_daq_set_trig_out_rate(IntPtr ctx, int r)</code>	<code>int32_t</code>	设置触发输出频率 Hz。UI: 0 ~ 10,000,000, 默认 10000。

2.7 数据读取

C# 声明	C 参数类型	说明
<code>klinx_daq_read_ptr(IntPtr ctx, IntPtr buf, ulong sz)</code>	<code>int16_t* buf, uint64_t sz</code>	读取采集数据至 buf。sz 为元素个数 (int16_t)。buf 必须为原生堆指针 (Marshal.AllocHGlobal)，地址在整个 session 中保持不变以满足 DMA 要求。EntryPoint="klinx_daq_read"。
<code>ulong klinx_daq_point_count_per_scan(IntPtr ctx)</code>	<code>uint64_t</code>	返回每次扫描的采样点数。在 klinx_daq_start 后可能返回与 start 前不同的值，因此 UpdatePlot 中每 tick 重新查询 (与 C++ updatePlot 行为一致)。

klinx_daq_read 的 buf 声明为 IntPtr 而非 short[]，原因是驱动要求地址稳定 (DMA 物理页锁定)。使用 [Out] short[] 时 CLR 每次调用可能生成新的中间缓冲区，破坏地址一致性。

3. C → C# 类型映射

C 类型	C# P/Invoke 类型	备注
int	int	32 位有符号
uint32_t	uint	32 位无符号
int32_t	int	32 位有符号 (trig_out_rate / pulse_width)
uint64_t	ulong	64 位无符号 (scan_rate、np、delay 等)
int64_t / size_t	ulong / long	视语义选择
bool	[MarshalAs(UnmanagedType.I1)] bool	1 字节，匹配 C bool (adc_enable)
int16_t* (数据缓冲区)	IntPtr	原生堆指针，禁止使用 [Out] short[]
device_handle_t (opaque ptr)	IntPtr	不透明句柄
daq_status_t (enum)	KlinxStatus (enum : int)	值完全对应
state_t (enum)	KlinxState (enum : int)	值完全对应
daq_trig_mode_t (enum)	KlinxTrigMode (enum : int)	值完全对应
adc_range_flag_t (enum)	KlinxRangeFlag (enum : int)	值完全对应

4. UI 控件参数对照

以下参数源自 mainwindow.ui, C# 中通过 QtMainWindowUiValues 常量类统一管理。

4.1 触发输入参数 (External Trig 模式)

控件	C++ UI 名称	范围	默认值	接口
Sample Point Count Per Trig	spinBoxSamplePointCountPerTrig	1 ~ 250,000,000	12500	set_sample_point_count_per_trig
Sample Times	spinBoxSampleCircleTimesPerTrig	1 ~ 99999	1	set_sample_trig_times
Delay Clocks	spinBoxBeginSampleNClocksAfterTrig	0 ~ 99999	0	set_sample_delay_nclocks_after_trig
Edge	buttonTrigINRising / buttonTrigINFalling	Rising / Falling	Rising	set_trig_in_edge
Pole	buttonTrigINHigh / buttonTrigINLow	High / Low	High	set_trig_in_pole

4.2 主机模式参数

控件	C++ UI 名称	范围	默认值	接口
Scan Rate	spinBoxScanRate	1 ~ 50000	10000	set_scan_rate

4.3 触发输出参数

控件	C++ UI 名称	范围	默认值	接口
Pulse Width	spinBoxPulseWidth	0 ~ 1,000,000	25	set_trig_out_pulse_width
Rate (Hz)	spinBoxTrigOUTRate	0 ~ 10,000,000	10000	set_trig_out_rate
Pole	buttonTrigOUTHigh / buttonTrigOUTLow	High / Low	High	set_trig_out_pole

4.4 采集控制参数

控件	C++ UI 名称	范围	默认值	说明
Batch Read Scan	spinBoxBatchReadScan	1 ~ 100,000	1000	缓冲区大小倍数: adBuf 大小 = activeChannels × np × batch

5. 典型调用流程

5.1 完整操作序列

步骤	C# 调用	C++ 对应	说明
1	<code>klinx_daq_create(0)</code>	<code>klinx_daq_create(0)</code>	创建设备上下文
2	Set Param 按钮	<code>onSetParam()</code>	下发所有参数到驱动
3	<code>klinx_daq_open(handle)</code>	<code>klinx_daq_open()</code>	打开设备
4	<code>klinx_daq_setup(handle)</code>	<code>klinx_daq_setup()</code>	应用参数, 进入 ESTABLISHED
5	<code>Read(true)</code>	<code>startRead(true)</code>	分配原生缓冲区, 启动 5ms 定时器
6	<code>klinx_daq_start(handle)</code>	<code>klinx_daq_start()</code>	启动硬件采集, 进入 WORKING
7	<code>OnTick() × N</code>	<code>QTimer::timeout × N</code>	每 5ms 读数据 + 刷新波形
8	<code>klinx_daq_stop(handle)</code>	<code>klinx_daq_stop()</code>	停止采集, 进入 STOPPED
9	<code>Read(false)</code>	<code>startRead(false)</code>	停止定时器 (不释放缓冲区)
10	<code>klinx_daq_close(handle)</code>	<code>klinx_daq_close()</code>	关闭设备
11	<code>FreeAdBuf()</code>	<code>~MainWindow()</code>	释放原生缓冲区

5.2 缓冲区大小计算

缓冲区大小 (int16_t 元素数) :

```
sz = activeChannelCount() * klinx_daq_point_count_per_scan(handle) * batch
```

数据布局 (交织格式, 与 C++ 一致) :

```
adBuf[n * activeChannel + ch] → 第 n 帧、第 ch 通道的采样值
```

5.3 解交织逻辑

```
for n in [0, np):
    offset = n * activeChannel
    for ch in [0, activeChannel):
        plotBufs[ch][n] = adCopy[offset + ch]
```

6. 自定义控件

6.1 WaveformPlotControl

多通道波形显示控件，纯 GDI+ 自绘，双缓冲。

公开方法 / 属性	说明
SetChannelIds(IReadOnlyList<int>)	设置要显示的通道 ID 列表，清空现有数据并重绘。
SetSeries(IReadOnlyList<int>, IReadOnlyList<short[]>)	更新通道 ID 列表和对应波形数据，触发重绘。
SetXAxisRange(double min, double max)	设置 X 轴范围（采样点索引）。

绘制参数：

参数	值	说明
Y 轴范围	-30000 ~ +30000	固定，对应 int16_t 量程
Y 轴刻度步长	10000	每格 10000 ADC counts
通道间距	10 px	相邻子图间隔
左边距	54 px	为 Y 轴标签留空
底边距	20 px	为 X 轴标签留空

6.2 StatusStateControl

状态栏控件，包含一个 LedIndicatorControl 和一个文字标签。

公开方法	说明
UpdateState(KlinxState st)	根据设备状态更新 LED 颜色、闪烁模式和文字。

6.3 LedIndicatorControl

圆形 LED 指示灯，GDI+ 自绘，支持三色和三种显示模式。

模式	效果	使用场景
LedState.Off	暗淡颜色 (Alpha 70)	DAQ_CLOSED
LedState.On	纯色点亮	DAQ_ESTABLISHED / DAQ_STOPPED
LedState.Blink	350ms 间隔闪烁	DAQ_WORKING