

FUNCTIONAL SPECIFICATION

Real-Time Multi-Channel DAQ Visualizer

ImGui + ImPlot / C++14 / CODAC Core OS 7

Field	Value
Document version	1.0 — DRAFT
Status	For review
Target platform	CODAC Core OS 7 (RHEL 7 base)
Prepared by	—
Date	2026-05-22

Table of Contents

1. Purpose and Scope

This document specifies the functional requirements for a real-time multi-channel data acquisition (DAQ) visualizer. The software receives high-rate sample data over UDP, stores it in lock-free ring buffers, decimates it dynamically for display, and provides an oscilloscope-style interface including triggered capture mode.

The application is self-contained — all third-party UI code is compiled in-tree — and imposes no system package dependencies beyond a basic OpenGL/X11 stack already present on the target platform.

2. Background and Constraints

2.1 Operational Environment

Parameter	Value
Operating system	CODAC Core OS 7 (RHEL 7 / glibc 2.17)
Network	Isolated — no internet access
GPU	No discrete GPU; Mesa software rasterizer (swrast) assumed
Package management	No yum/pip access; all deps must be in-tree
Compiler	GCC 4.8+ (C++11/14)

2.2 Key Design Constraints

- All third-party library source (ImGui, ImPlot, GLFW or SDL2) is vendored into the repository and compiled together with the application. No shared library installation is required.
- The software must operate correctly with software-only OpenGL rendering (Mesa swrast) and must not depend on hardware-accelerated paths.
- Memory footprint must be manageable on constrained hardware: full-rate ring buffers cover a configurable short window; long-term history uses a separately maintained decimated overview buffer.
- The ingest path must never block on UI operations; all inter-thread communication uses lock-free single-producer / single-consumer (SPSC) ring buffers.

3. Architecture Overview

The system is divided into three independent layers that communicate only through shared ring buffers. This separation ensures that a slow render frame never causes sample drops, and a burst of incoming data never stalls the UI.

Layer	Responsibility
Ingest	One dedicated thread per channel. Receives UDP datagrams, validates packet headers, and writes samples to the channel ring buffer. No processing beyond type conversion.
Store	Per-channel lock-free SPSC ring buffer holding the full-rate sample history for the configurable short window. A second, lighter ring buffer holds a pre-decimated overview stream for long-term display.
Render	Runs on the main thread at the display frame rate (~30–60 fps). Reads from the ring buffer tail, applies min/max envelope decimation to fit the current viewport width, and calls ImPlot draw primitives.

3.1 Technology Stack

Component	Choice and rationale
Language	C++14 — best balance of performance, portability, and library compatibility on GCC 4.8+
UI framework	Dear ImGui (immediate-mode) — vendored source (~15 files), no system deps, designed for real-time data tools
Plotting	ImPlot — vendored companion to ImGui, native support for large datasets with stride-based rendering and built-in zoom/pan
Window / GL context	GLFW 3.x or SDL2 — whichever is available on the target, or compiled from source; both have minimal dependencies
OpenGL version	OpenGL 2.1 + GLSL 1.20 — supported by Mesa swrast on any hardware
Build system	CMake 2.8+ (matches RHEL 7 available version)

4. Functional Requirements

4.1 UDP Ingest

The ingest subsystem receives sample data from multiple simultaneous UDP streams and writes it into per-channel ring buffers with minimal latency.

ID	Requirement	Priority	Status
ING-01	The application shall accept configuration specifying the number of channels (1–N), the UDP port for each channel, and the expected sample rate.	Must	Draft
ING-02	Each channel shall be serviced by a dedicated OS thread pinned, where possible, to a specific CPU core via <code>pthread_setaffinity</code> .	Must	Draft
ING-03	The socket receive buffer (<code>SO_RCVBUF</code>) shall be set to the maximum permitted by the kernel on startup; the application shall log the negotiated value.	Must	Draft
ING-04	The ingest thread shall perform no heap allocation after initialisation. Datagrams are decoded into a fixed-size stack buffer and the payload written directly to the ring buffer.	Must	Draft
ING-05	The ingest thread shall count and expose: datagrams received, bytes received, sequence-number gaps detected, and ring-buffer overflows.	Should	Draft
ING-06	The application shall support at least 4 simultaneous channels each running at up to 2 MSps (sample rate configurable per channel).	Must	Draft
ING-07	Packet format shall be configurable: raw IEEE-754 float32 payload, int16, or a simple header (sequence number + timestamp + samples) defined in a compile-time struct.	Should	Draft

4.2 Ring Buffer and Memory Management

The ring buffer is the sole point of communication between ingest and render threads. Its design is critical to both performance and data integrity.

ID	Requirement	Priority	Status
BUF-01	Each channel shall have a dedicated SPSC ring buffer. The buffer size shall be a power of two to permit index masking instead of modulo.	Must	Draft
BUF-02	The short-window (full-rate) ring buffer depth shall be configurable at startup (default: enough samples for 60 seconds at the configured sample rate).	Must	Draft
BUF-03	A separate overview ring buffer shall maintain a pre-decimated stream at a fixed reduced rate (default 1000 sps). This buffer shall hold up to 2 hours of overview data.	Must	Draft
BUF-04	All ring buffer memory shall be pre-allocated at startup. No dynamic allocation shall occur during operation.	Must	Draft

BUF-05	The ring buffer head (writer) and tail (reader) indices shall be <code>std::atomic<uint64_t></code> with appropriate memory ordering to avoid data races without mutexes.	Must	Draft
BUF-06	On overflow (ingest catches up to the render read pointer) the oldest samples shall be silently overwritten; the overflow counter (ING-05) shall be incremented.	Must	Draft

4.3 Display and Rendering

The render subsystem reads decimated data from ring buffers and draws waveforms inside an ImPlot canvas at the display refresh rate.

ID	Requirement	Priority	Status
RND-01	The application shall render at a configurable target frame rate (default 30 fps). The render loop shall yield when ahead of the target to avoid spinning.	Must	Draft
RND-02	Before each draw call, the render thread shall decimate the visible sample range to at most (<code>viewport_pixel_width × 2</code>) points using a min/max envelope algorithm.	Must	Draft
RND-03	Min/max envelope decimation shall produce two arrays (min values and max values per pixel column). ImPlot's <code>PlotShaded</code> shall be used to draw the envelope as a filled band, matching the appearance of a hardware oscilloscope at high time compression.	Must	Draft
RND-04	When the visible time window is short enough that the decimation ratio is 1:1 (one sample per pixel or fewer), the application shall switch to a plain polyline draw (ImPlot <code>PlotLine</code>).	Must	Draft
RND-05	The application shall display all active channels on a shared time axis with independent Y-axis scaling per channel. Channels may be stacked vertically or overlaid.	Must	Draft
RND-06	Axes shall display physical units where configured (time in seconds/ms/ μ s, amplitude in user-defined units with configurable scale factor and offset).	Should	Draft
RND-07	The application shall target correct rendering on Mesa swrast (OpenGL 2.1). It shall not use geometry shaders, instanced rendering, or any extension not available in GL 2.1.	Must	Draft

4.4 Zoom, Pan, and Time Navigation

The user must be able to navigate from a full multi-hour overview down to sub-millisecond inspection of individual events without any mode switching.

ID	Requirement	Priority	Status
NAV-01	The visible time window shall be adjustable via scroll-wheel zoom and click-drag pan within the ImPlot canvas.	Must	Draft
NAV-02	When the visible time window exceeds the short-window ring buffer depth, the render thread shall transparently switch to the overview buffer. The transition shall be seamless and require no user action.	Must	Draft

NAV-03	A minimap (overview plot, reduced height) shall always show the full available history with a highlight rectangle indicating the current viewport. Clicking or dragging in the minimap shall pan the main viewport.	Should	Draft
NAV-04	The application shall support a 'live' mode in which the right edge of the viewport tracks the current sample timestamp. Entering live mode shall be possible via a toolbar button or keyboard shortcut.	Must	Draft
NAV-05	Zoom limits shall be enforced: minimum window = 10 samples visible; maximum window = full available history.	Must	Draft
NAV-06	Y-axis scaling shall be adjustable per channel via scroll-wheel when the cursor is over that channel's lane, or via direct numeric input in the channel settings panel.	Should	Draft

4.5 Trigger Mode

Trigger mode allows the user to capture and freeze a time-aligned waveform snapshot on a specific signal condition, equivalent to the triggered sweep of a digital oscilloscope.

ID	Requirement	Priority	Status
TRG-01	The application shall provide a trigger engine that monitors a user-selected channel for a configurable edge condition (rising edge, falling edge, or either) at a configurable threshold level.	Must	Draft
TRG-02	A pre-trigger circular buffer shall retain the most recent configurable duration of samples (default: 10 ms, range: 0 – full short-window depth) before any trigger event.	Must	Draft
TRG-03	On trigger fire, the application shall snapshot the pre-trigger buffer plus a configurable post-trigger duration into a dedicated capture buffer and freeze the display on that capture.	Must	Draft
TRG-04	While a capture is frozen, live data shall continue to be received and buffered in the background. Dismissing the frozen capture shall return the display to the live view.	Must	Draft
TRG-05	The trigger engine shall operate in one of three modes selectable by the user: Single (fires once then disarms), Normal (re-arms after each capture), Auto (fires on timeout if no trigger occurs).	Must	Draft
TRG-06	A trigger status indicator shall be visible at all times: Armed (waiting), Triggered (capture frozen), or Stopped.	Must	Draft
TRG-07	The captured waveform shall be exportable to a CSV file containing timestamps and sample values for all channels.	Should	Draft

5. User Interface Layout

The application window is divided into four regions. All regions are implemented as ImGui windows docked into a fixed layout. The layout is saved to an ini file and restored on next launch.

Region	Contents
Toolbar (top bar)	Play/pause live mode, trigger arm/disarm/status indicator, time-base selector, file export button, settings toggle.
Main plot area (centre)	ImPlot canvas showing all active channels. Time axis shared. Each channel occupies a horizontal lane with its own Y axis. Zoom and pan via mouse.
Overview / minimap (below main)	Compact ImPlot showing full history of all channels at overview resolution. Viewport rectangle overlaid. Click/drag to navigate.
Channel panel (right sidebar)	Per-channel controls: label, colour, Y scale, Y offset, unit string, visibility toggle, trigger source selection.

5.1 Keyboard Shortcuts

Key	Action
Space	Toggle live / paused mode
T	Arm / disarm trigger
R	Reset zoom to full history
Escape	Dismiss frozen capture and return to live
Ctrl + S	Export current capture to CSV
Ctrl + Q	Quit application

6. Configuration

The application is configured via a plain-text INI file passed as a command-line argument. All parameters have documented defaults so a minimal configuration is valid.

6.1 Configuration Parameters

Parameter	Description and default
channel_count	Number of UDP channels to receive (default: 1)
channel[N].port	UDP port for channel N (default: 5000 + N)
channel[N].sample_rate	Expected sample rate in samples/second (default: 1000000)
channel[N].dtype	Sample data type: float32, int16, int32 (default: float32)
channel[N].label	Display label for channel N (default: "CH N")
channel[N].unit	Y-axis unit string (default: "")
channel[N].scale	Y-axis scale factor applied to raw samples (default: 1.0)
channel[N].offset	Y-axis offset applied after scale (default: 0.0)
buffer.short_window_sec	Full-rate ring buffer depth in seconds (default: 60)
buffer.overview_rate	Overview buffer sample rate in sps (default: 1000)
buffer.overview_window_sec	Overview buffer depth in seconds (default: 7200)
trigger.default_channel	Channel index to use as trigger source (default: 0)
trigger.default_threshold	Default trigger threshold in scaled units (default: 0.0)
trigger.pretrigger_ms	Pre-trigger buffer depth in milliseconds (default: 10)
trigger.posttrigger_ms	Post-trigger capture depth in milliseconds (default: 90)
display.fps	Target render frame rate (default: 30)
display.width	Initial window width in pixels (default: 1280)
display.height	Initial window height in pixels (default: 800)
display.opengl_version	OpenGL version to request: 2 or 3 (default: 2)

7. Performance Requirements

ID	Requirement	Priority	Status
PRF-01	The ingest path shall sustain 2 MSps per channel with zero sample loss on a system with sufficient CPU cores, measured over a 60-second continuous test.	Must	Draft
PRF-02	The render thread shall complete one full decimation + draw pass within 33 ms (30 fps budget) for up to 4 simultaneous channels at viewport width \leq 2560 pixels.	Must	Draft
PRF-03	Total process RSS memory shall not exceed 2 GB when running 4 channels at 1 MSps with a 60-second full-rate buffer and a 2-hour overview buffer.	Should	Draft
PRF-04	UI input events (mouse move, scroll) shall be processed within one render frame. There shall be no perceptible lag between user input and viewport change.	Must	Draft
PRF-05	Application startup time from launch to first rendered frame shall be under 3 seconds on the target hardware.	Should	Draft

Performance targets (PRF-01, PRF-02) assume a multi-core x86-64 system. On a single-core system the ingest thread and render thread compete for CPU; in this case target sample rate may need to be reduced or the render FPS target lowered.

8. Build and Deployment

8.1 Repository Layout

Path	Contents
src/	Application source files (main.cpp, udp_receiver, ring_buffer, trigger_engine, ui)
third_party/imgui/	Dear ImGui vendored source (MIT)
third_party/implot/	ImPlot vendored source (MIT)
third_party/glfw/	GLFW vendored source (zlib), or replaced by SDL2 if preferred
config/	Example INI configuration files
CMakeLists.txt	Top-level CMake build file

8.2 Build Procedure

The following commands produce a fully statically-linked executable requiring only glibc, libGL, and libX11 from the target system:

```
mkdir build && cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j$(nproc)
```

The resulting binary (daq_viewer) has no runtime dependencies on ImGui, ImPlot, or GLFW — they are compiled in. It requires libGL.so.1 and libX11.so.6, both present on any RHEL 7 / CODAC OS 7 system with a graphical environment.

If libGL or libX11 are absent (headless server), the application can be built with the SDL2 software renderer backend, which substitutes a pure-CPU framebuffer without requiring any GL driver.

9. Non-Functional Requirements

- The application shall not write to any path outside its working directory and the user home directory (~/.config/daq_viewer.ini).
- All threads shall be named (pthread_setname_np) for debuggability with top and perf.
- The application shall handle SIGINT and SIGTERM gracefully: flush statistics, close sockets, and exit cleanly.
- Log output shall go to stderr with ISO 8601 timestamps and a severity prefix (INFO / WARN / ERROR). A --verbose flag enables debug-level logging.
- The codebase shall compile without warnings under -Wall -Wextra -Wpedantic with GCC 4.8 in C++14 mode.
- The application shall be single-binary deployable: copy the executable and an INI file to the target machine and run.

10. Open Issues and Decisions

Issue	Notes
GLFW vs SDL2	GLFW is lighter and easier to compile from source; SDL2 may already be installed on CODAC. To be confirmed by checking target system packages.
Packet format	The UDP payload format (raw float32 vs framed packet with header) must be agreed with the data source before implementation of ING-07.
Multi-host UDP	If streams originate from multiple hosts, SO_REUSEPORT or separate bind addresses may be needed. To be confirmed from the network topology.
Export format	CSV is specified (TRG-07). Binary formats (HDF5, raw float32 dump) may be added in a later revision if required by downstream analysis tools.
Maximum channel count	The requirement states 4 channels as a minimum (ING-06). The practical maximum at 2 MSps is limited by available CPU cores and memory bandwidth; this should be profiled during integration testing.

This document is a functional specification draft. Requirements marked 'Should' are targets; 'Must' requirements are contractual. Status column tracks review progress.