QtEDM: A Modern Qt Reimplementation of MEDM for EPICS Control System Displays

Robert Soliday

Advanced Photon Source, Argonne National Laboratory, Lemont, IL 60439, USA

Abstract

QtEDM is a modern Qt-based reimplementation of MEDM (Motif Editor and Display Manager), the widely-used graphical user interface for EPICS control systems. QtEDM supports both Qt5 and Qt6 and runs natively on Linux, macOS, and Windows. As the Motif toolkit becomes harder to maintain on contemporary systems, QtEDM provides a sustainable path forward while maintaining full backward compatibility with existing ADL display files. This paper describes the motivation for QtEDM, its architecture and implementation, key features and enhancements over the original MEDM, and the transition path for existing installations.

1. Introduction

MEDM has served as a cornerstone application for EPICS-based control systems for over three decades. Originally developed at Argonne National Laboratory by Mark Anderson, with significant contributions from Frederick Vong and Kenneth Evans Jr., MEDM provides operators and engineers with the ability to design and operate graphical control screens that interact with EPICS process variables through Channel Access.

Despite its proven reliability and widespread adoption, MEDM faces a significant challenge: its dependence on the Motif toolkit. Motif, once the standard GUI toolkit for X11/Unix systems, has seen declining support, uneven packaging on modern Linux distributions, and no native path for macOS or Windows deployments. Font rendering issues, widget appearance inconsistencies, and maintenance difficulties have made continued reliance on Motif increasingly untenable.

QtEDM addresses these challenges by reimplementing MEDM's functionality using Qt (versions 5 or 6), a modern, actively-maintained, cross-platform GUI framework. The result is a display manager that reads and writes the same ADL file format, supports all existing widget types, and provides equivalent functionality—while benefiting from Qt's superior rendering, font handling, and long-term support.

2. Motivation

2.1 Motif Deprecation

The Motif toolkit, while historically significant, presents several challenges for modern deployments:

- **Declining distribution support**: Many Linux distributions have reduced or eliminated Motif packages
- **Limited native platform reach**: Motif remains tied to X11-style environments rather than native macOS and Windows desktops
- **Font rendering issues**: Motif's font handling struggles with contemporary font configurations
- **Visual inconsistency**: Motif widgets appear dated and may conflict with modern desktop themes
- **Limited development**: Minimal ongoing development or bug fixes

2.2 Preserving Investment in ADL Files

EPICS facilities have accumulated substantial investments in ADL display files. These displays represent significant engineering effort in designing operator interfaces. Any successor to MEDM must preserve this investment by maintaining full compatibility with existing ADL files.

2.3 Modern Feature Requirements

Operators and engineers have expressed needs for features difficult to implement in Motif:

- Improved font scaling and rendering
- Better printing and image export capabilities
- Enhanced data export from trend displays
- Interactive zoom and pan for plots
- Audit logging for regulatory compliance

3. Architecture

3.1 Design Philosophy

QtEDM follows a strict separation between visual presentation and EPICS Channel Access operations:

- **Element classes** handle widget painting and user interaction as QWidget subclasses
- **Runtime classes** manage EPICS channel connections and data updates as QObject subclasses

This separation allows for cleaner code organization and easier testing. Runtime objects are created when entering Execute mode and destroyed when returning to Edit mode, ensuring clean resource management.

3.2 Channel Access and PVAccess Integration

QtEDM uses the standard EPICS Channel Access library for process variable communication and accepts `pva://` PV prefixes when PVAccess is required. A centralized `ChannelAccessContext` manages CA initialization and periodic polling. Each runtime class that requires PV connectivity follows established patterns for channel creation, callback registration, and event handling.

3.3 ADL File Compatibility

QtEDM reads and writes the identical ADL file format used by MEDM. The ADL parser is implemented in C++ and carefully replicates MEDM's parsing behavior. Display files created by either application can be opened in the other without modification.

3.4 Dual Build System

Both MEDM and QtEDM are built from the same source repository. The build system detects available dependencies and produces both executables. This ensures ongoing maintenance of both implementations during the transition period.

4. Supported Widget Types

QtEDM implements the complete set of MEDM widget types:

4.1 Graphics Widgets

- Rectangle, Oval, Arc
- Line, Polyline, Polygon
- Text (static labels)
- Image (GIF and other formats)
- Composite (grouped widgets)

4.2 Monitor Widgets

- Text Monitor

- Bar Monitor
- Byte Monitor
- LED Monitor
- Scale Monitor (indicator)
- Meter
- Thermometer
- Strip Chart
- Cartesian Plot
- Heatmap
- Waterfall Plot
- PV Table
- Waveform Table
- Expression Channel

4.3 Controller Widgets

- Text Entry
- Text Area
- Slider (Valuator)
- Wheel Switch
- Choice Button
- Menu
- Message Button
- Related Display
- Shell Command

Standard MEDM widgets support the same dynamic attributes (visibility, color modes) as MEDM, ensuring behavioral compatibility. QtEDM-only widgets extend ADL for newer diagnostic and operator workflows.

5. Enhancements Over MEDM

While maintaining compatibility, QtEDM introduces several enhancements:

5.1 Improved Font Handling

QtEDM offers two font modes:

- **Alias mode**: Matches MEDM's font sizing for pixel-perfect compatibility
- **Scalable mode**: Uses Qt's native font scaling for improved readability

5.2 Enhanced Statistics Window

The Statistics Window includes a "PV Details" mode displaying a sortable table of all connected process variables with connection status, update rate, and alarm severity—useful for debugging connection issues.

5.3 Find PV Dialog

A Find PV feature (Ctrl+F) searches for process variables across all open displays, helping operators locate widgets associated with specific PVs.

5.4 Image Export

Displays can be exported as PNG, SVG, JPEG, or BMP images. SVG export provides vector output suitable for documentation.

5.5 Data Export

Strip Charts and Cartesian Plots support data export in SDDS or CSV format, enabling offline analysis of trend data.

5.6 Interactive Plot Navigation

Cartesian Plots support:

- Mouse wheel zoom (centered on cursor)
- Shift+scroll for X-axis only zoom
- Ctrl+scroll for Y-axis only zoom
- Click-and-drag panning
- Right-click reset to original view

5.7 Audit Logging

QtEDM logs all control widget value changes (writes to PVs). Log entries include timestamp, username, widget type, PV name, value written, and display file. Logging can be disabled with `-nolog` or the environment variable QTEDM_NOLOG=1 if not required.

5.8 Native Desktop Platforms

QtEDM runs natively on Linux, macOS, and Windows when the local Qt, EPICS Base, and SDDS dependencies are available. This gives sites a path away from X11/Motif constraints without giving up existing ADL files.

6. Command Line Interface

QtEDM accepts MEDM-compatible command line options:

```
qtedm [options] [display-files]

Options:
  -x              Start in EXECUTE mode
  -macro "..."    Define macro substitutions
  -dg geometry    Specify display geometry
  -displayFont    Select font mode (alias|scalable)
  -nolog          Disable audit logging
  -help           Display usage information
```

7. Transition Considerations

7.1 Deployment Strategy

Facilities can deploy QtEDM alongside MEDM, allowing gradual transition:

1. Install QtEDM as a separate executable
2. Test existing ADL files for visual fidelity

3. Migrate operators to QtEDM incrementally. At the APS, this can be done by setting the environment variable USE_QTEDM=1 prior to launching common MEDM scripts such as "storage-ring" or "linac".
4. Maintain MEDM for edge cases during transition

7.2 Known Differences

While QtEDM strives for exact compatibility, minor differences exist:

- Qt rendering may produce slightly different anti-aliasing
- Window management behaviors depend on desktop environment
- Some Motif-specific widget appearances are approximated

7.3 Training Requirements

The user interface is intentionally similar to MEDM. Operators familiar with MEDM require minimal training—the same menus, keyboard shortcuts, and workflows apply.

8. Implementation Status

QtEDM is ready for widespread testing. The implementation includes:

- Complete widget set with all dynamic attributes
- Full ADL file read/write compatibility
- Execute and Edit mode operation
- Channel Access and PVAccess PV naming
- Printing and image export
- Macro substitution and related displays
- QtEDM-only widgets including Heatmap, Waterfall Plot, PV Table, Waveform Table, Thermometer, LED Monitor, Text Area, and Expression Channel
- All standard EPICS environment variables

9. Future Work

Potential future enhancements include:

- Continue expanding QtEDM-native widgets for workflows that MEDM never covered.
- Implement "Save PV Values" snapshot feature for current display state.
- Add "Restore PV Values" to write saved snapshot back to IOC.
- Continue broadening platform packaging and validation for Linux, macOS, and Windows sites.

10. Availability

QtEDM is based off a fork of the official MEDM source distribution and is available from:

- GitHub: https://github.com/rtsoliday/medm
- EPICS Extensions: http://www.aps.anl.gov/epics/extensions/medm/

The upstream MEDM source distribution is maintained at:

- GitHub: https://github.com/epics-extensions/medm

Building QtEDM requires Qt5 or Qt6 development packages, EPICS Base, and the SDDS library. The same QtEDM source is intended for native Linux, macOS, and Windows builds when the platform-specific toolchain and

dependencies are installed. The build system automatically detects dependencies and produces both MEDM and QtEDM executables where the required components are available.

11. Conclusion

QtEDM provides a sustainable path forward for EPICS display management as Motif support wanes. By maintaining full ADL file compatibility while leveraging Qt's modern capabilities, QtEDM protects existing investments in display files while enabling new features and long-term maintainability. The dual-build approach allows facilities to transition at their own pace while ensuring continued support for both implementations.

Acknowledgments

The author acknowledges the foundational work of Mark Anderson, Frederick Vong, and Kenneth Evans Jr. on the original MEDM implementation.

References

1. Anderson, M., Evans, K., "MEDM Reference Manual," Argonne National Laboratory, 2014.
2. Dalesio, L.R., et al., "The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future," Nuclear Instruments and Methods in Physics Research A, vol. 352, pp. 179-184, 1994.