Proposal to Upgrade the Experimental Physics and Industrial Control System's Graphical User Interface to Control System Studio Phoebus

Brian Eng, Mary Ann Antonioli, Peter Bonneau, Aaron Brown, Pablo Campero, George Jacobs, Mindy Leffel, Tyler Lemon, Marc McMullen, and Amrit Yegneswaran
Physics Division, Thomas Jefferson National Accelerator Facility, Newport News, VA 23606
November 1, 2021

This note presents factors considered in selecting Control System Studio (CSS) Phoebus as the framework of choice for developing new graphical user interfaces (GUIs) that allow users to interact with Experimental Physics and Industrial Control System (EPICS).

An advantage of using EPICS for controls and monitoring is its modularity—libraries can be swapped, e.g. alarming is independent of the logging which is independent of the interface. The EPICS operator interface (OPI) allows several types of GUIs to interact with the system, each of which can run concurrently with other GUIs, e.g. Extensible Display Manager, Best OPI, Yet (BOY), and recently, Phoebus.

Both BOY and Phoebus are OPIs developed by CSS. CSS BOY is based on Eclipse—an integrated development environment that uses the Java Standard Widget Toolkit (SWT) for GUI development. Since 2016, various parts of CSS have been migrating from SWT to JavaFX for GUI development. JavaFX has been included as part of the Java Development Kit since Java 11. All components of Phoebus are developed with JavaFX, not SWT and Eclipse.

Eclipse and SWT-based CSS BOY code is sluggish. For example, it takes ~30 minutes to compile BOY vs ~3 minutes to compile Phoebus. The lines of code (LOC) for the Channel Finder application, a common bundled application with CSS BOY and Phoebus used to find/list PVs, is 11.2 kLOC in BOY compared to 4.5 kLOC in Phoebus.

Along with compile time and LOC being less for Phoebus compared to BOY, Phoebus has less CPU and memory usage. The Phoebus Heater Demo, an example screen available in both CSS BOY and Phoebus, uses a quarter of the CPU and half the memory used by the CSS BOY Heater Demo, enabling Phoebus to load faster and have more screens open concurrently.

Another reason to migrate to Phoebus is that most of the BOY developers are actively developing Phoebus and as such, it has new features and widgets that are not available in BOY.

For developers, Phoebus has the ability to open and run most BOY screens without any changes, except scripts. While most scripts only require changing the import path, changes required are script-dependent.

Two methods to develop Phoebus screens are by using the built-in Display Editor, Fig. 1, or by using a separate programming language to output Phoebus files, which are simply eXtensible Markup Language (XML) files.

The Display Editor allows developers to drag-and-drop widgets and edit their properties with an easy-to-use interface. This built-in tool is a good choice for creating or editing files and is useful for fixed layout screens that are small to moderate in size.

Python is the separate programming language most often used by DSG when creating dynamic screens, whose layout is dependent on other factors, such as detector configuration or large displays (thousands of indicators). Python allows quick modifications to the script, which can then generate the Phoebus screen.

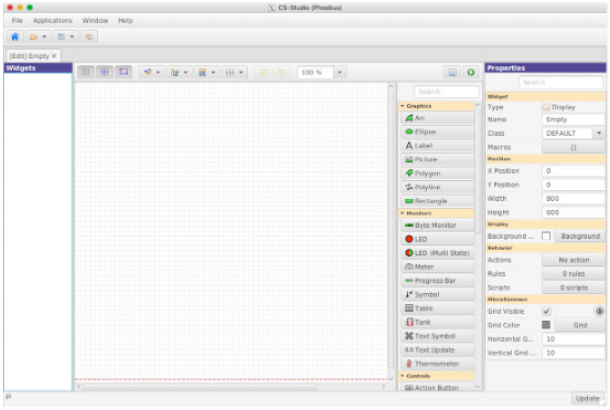

FIG. 1. Phoebus Display Editor.

Because of its increased performance and lower resource requirements, Phoebus has been selected for EPICS screens development. Work has started on implementing screens in Phoebus for Hall A detector high voltage, Fig. 2, with plans to expand to other Halls as new detector and magnet systems are instrumented and brought online.

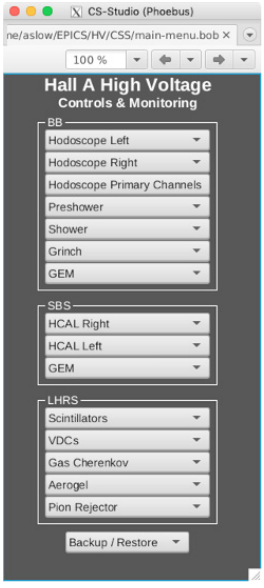

FIG. 2. Example of Phoebus screen for Hall A – Main Menu.