Performance testing of EPICS user interfaces -- an attempt to compare the performance of MEDM, EDM, CSS-Boy, and EPICS

Authors R. Farnsworth, J. P. Hammonds, B. Pausma, C. Suarez, APS, Argonne, IL 60439, USA
A. Rhyder, A. Starritt, SLSA, Clayton, Vic 3168, Australia.

Abstract

The upgrading of the display manager or graphical user interface at EPICS sites reliant on older display technologies, typically MEDM or EDM, requires attention to both functionality and performance. For many sites, performance is not an issue - all display managers will update a small number of process variables at rates exceeding the human ability to discern changes, but for certain applications typically found at larger sites the ability to respond to updates rates at sub Hertz frequencies for thousands of process variables is a requirement. This paper describes a series of tests performed on both older display managers - MEDM and EDM and also the newer CSS-Boy, epicsQT and CaQtDM. Modestly performing modern hardware is used.

UI	Version	CPU	Max Update Linux	Lossless Update Linux
CSS-BOY	3.1.4	14.88	22000	15000
EPICSQt	2.4.18	13.12	11100	10000
EDM	1.12.40	10.95	35100	20000
MEDM	3.1.7	12.83	65000	45000
CAQtDM	2.8.0	13.71	13500	5000
AS-Delphi	Int	16.54	9900	5000

Table 1 – Linux update rates, text widget

UI	Version	CPU	Max Update W7	Lossless Update W7
CSS-BOY	3.1.4	14.88	23000	16500
EPICSQt	2.4.18	13.12	10000	8000
EDM	1.12.40	10.95	NA	NA
MEDM	3.1.7	12.83	22250	16000
CAQtDM	2.8.0	13.71	13000	5500
AS-Delphi	Int	16.54	NA	NA

Table 2 – Window update rate, text widget

UI	Version	Max Update Linux	Lossless Update Linux
CSS-BOY	3.1.4	9840	8000
EPICSQt	2.4.18	14400	4000
EDM	1.12.40	142880	32000
MEDM	3.1.7	136800	48000
CAQtDM	2.8.0	16600	12000
AS-Delphi	Int	-	-

Table 3 – Linux update rates, Graphical widget

UI	Version	Max Update W7	Lossless Update W7
CSS-BOY	3.1.4	28800	16000
EPICSQt	2.4.18	40000	40000
EDM	1.12.40	NA	NA
MEDM	3.1.7	36000	7360
CAQtDM	2.8.0	-	-
AS-Delphi	Int	-	-

Table 4 – Windows update rates, Graphical widget

Description

Each user interface was monitored manually at 30 frames per second and updating screens were successively generated using the 500 PV data placed until such time as limits were met. The tw limits were:
1/ The user interface skipped updates. This is the lossless threshold; and
2/ The user interface failed altogether. This is the absolute threshold. Typically this is where (or just before) either locked up or failed to updated any PV's.
Video recording and inspection of the recorded video led to the verification of the USER interface limits. The average CPU utilisation was measured on Linux. The result, are generally rounded. The number of displayed PV's and the frequency of the updates was used them to calculate the update rate. To obtain the rate in Hertz, the total number of characters on the screen at the limit point was multiplied by the database update frequency.
The results are presented in the tables to the left.
The comparison of older user interfaces may show significant performance in their favour for text widgets. It is useful to note that have had many years of optimisation. The younger interfaces have different design criteria, faster hardware and less time spent on optimisation for performance. It is anticipated that newer versions of these products may perform significantly differently to these tests. It may also be noted that, in general the performance of CSS-BOY exceeds that of the Qt products. This is a surprising result as Eclipse have been generally considered slower than the C or C++ platforms that Qt is based upon. It may indicate that the greatest future performance improvements may come from the Qt platforms as these products mature.

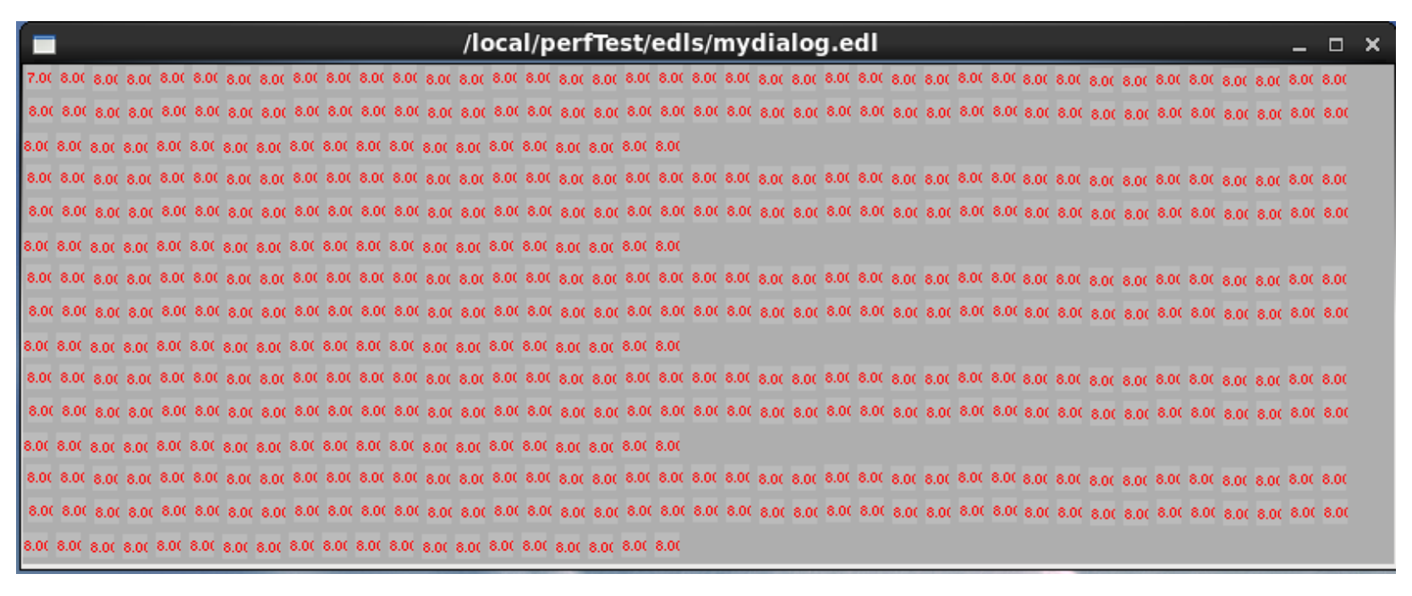
Fig 5 EDM - text

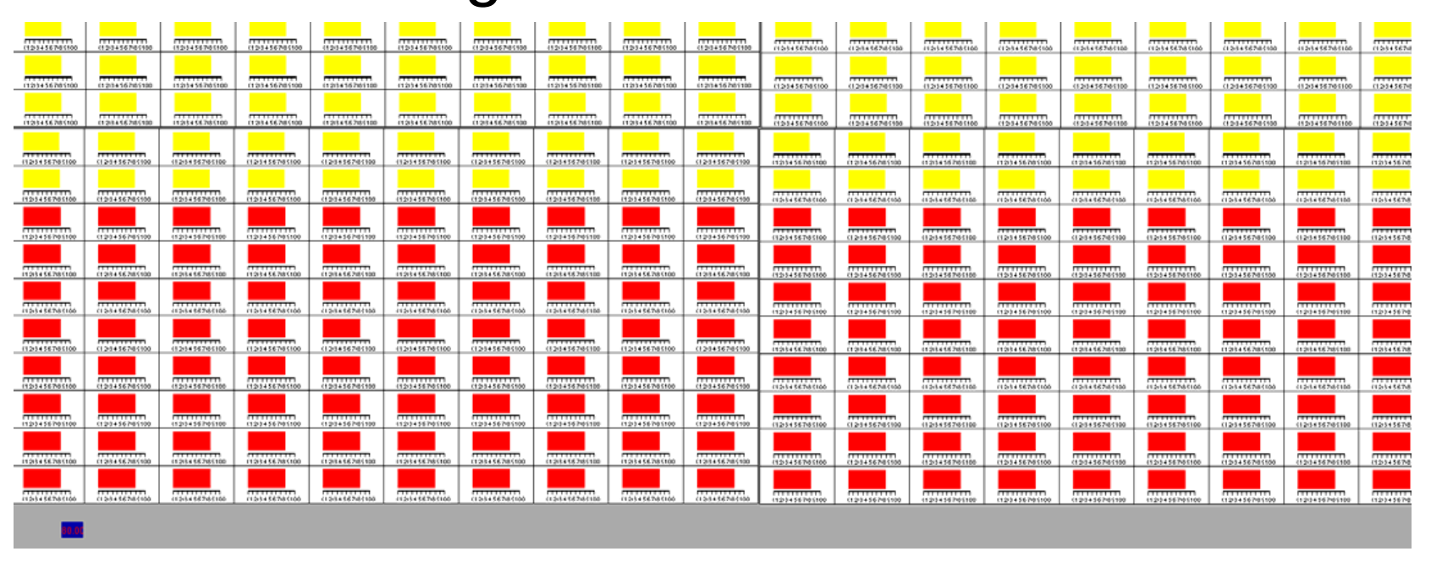
Fig 6 EDM - graphic

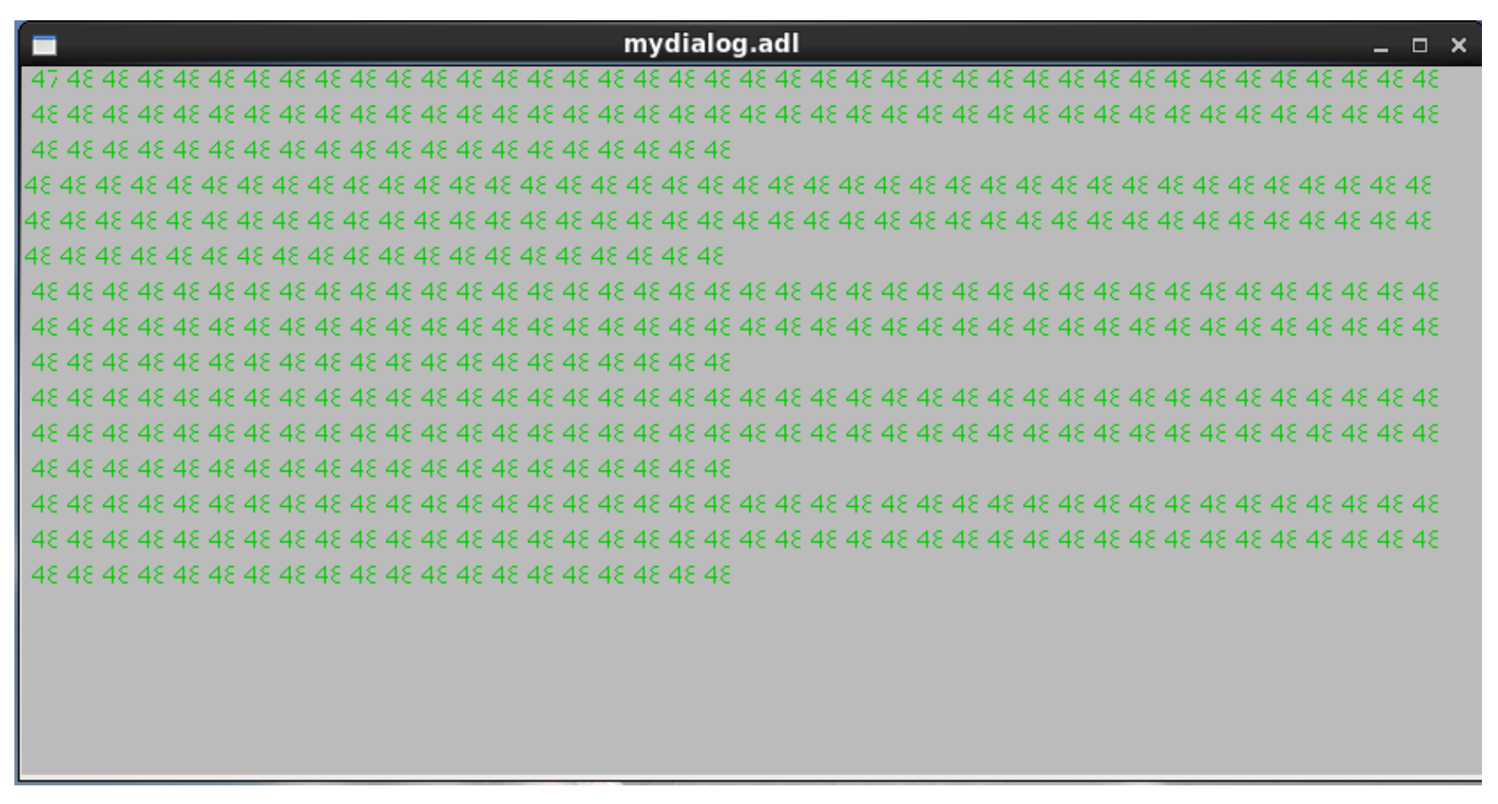
Fig 1 Medm - text

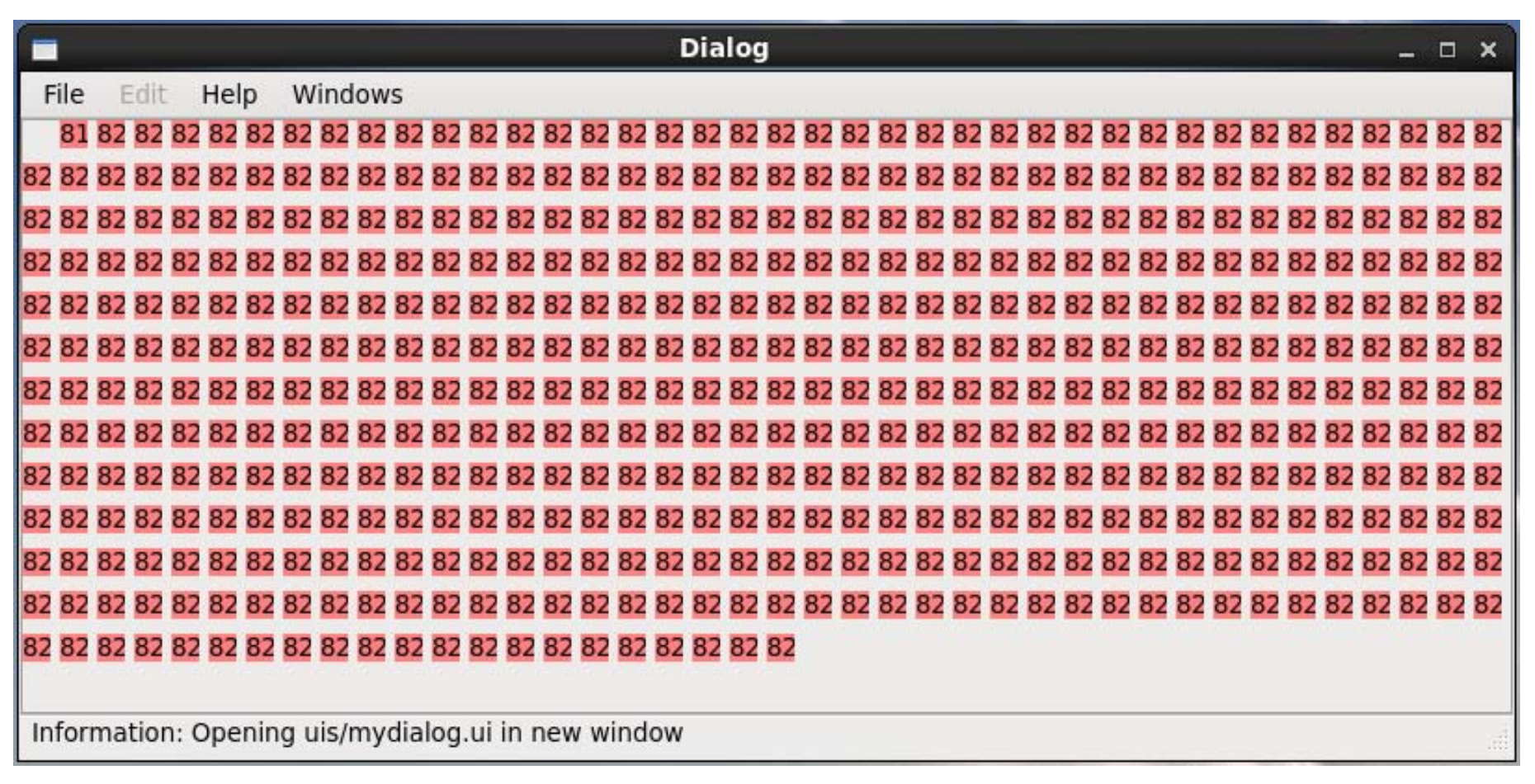
Fig 3 EPICSQt - text

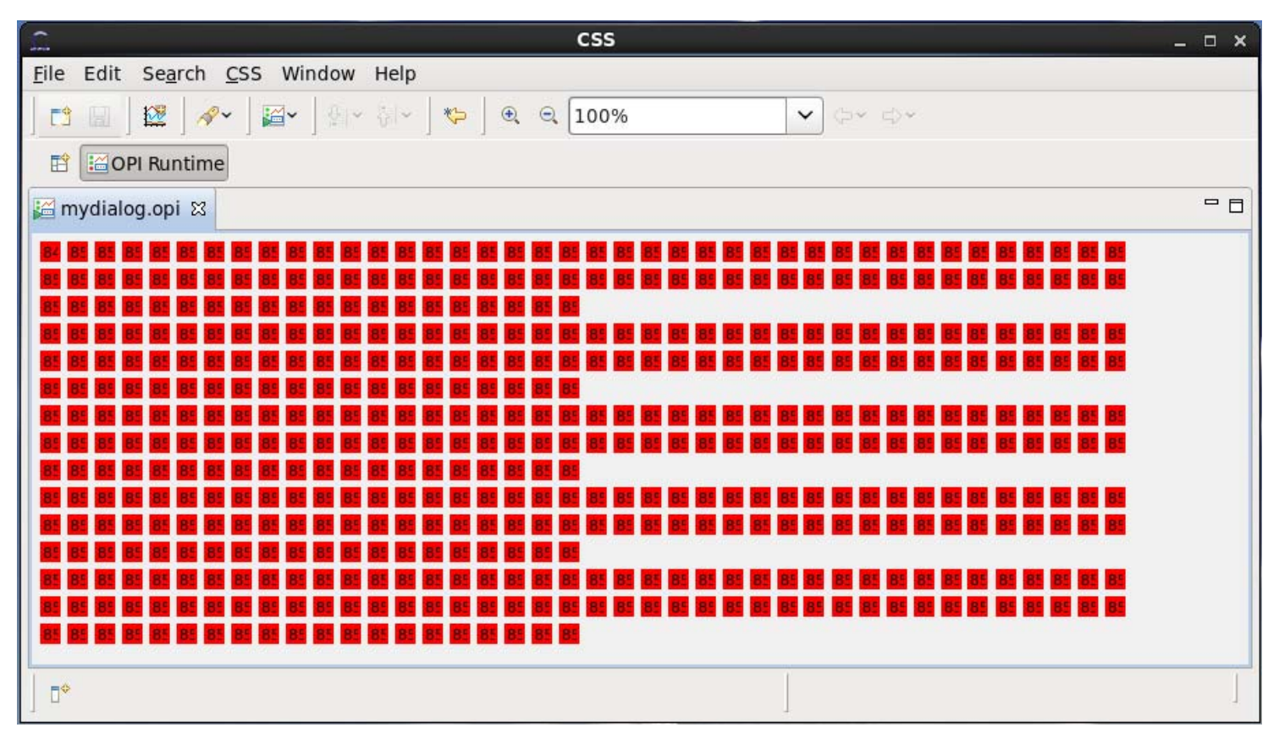
Fig 5 CSS - text

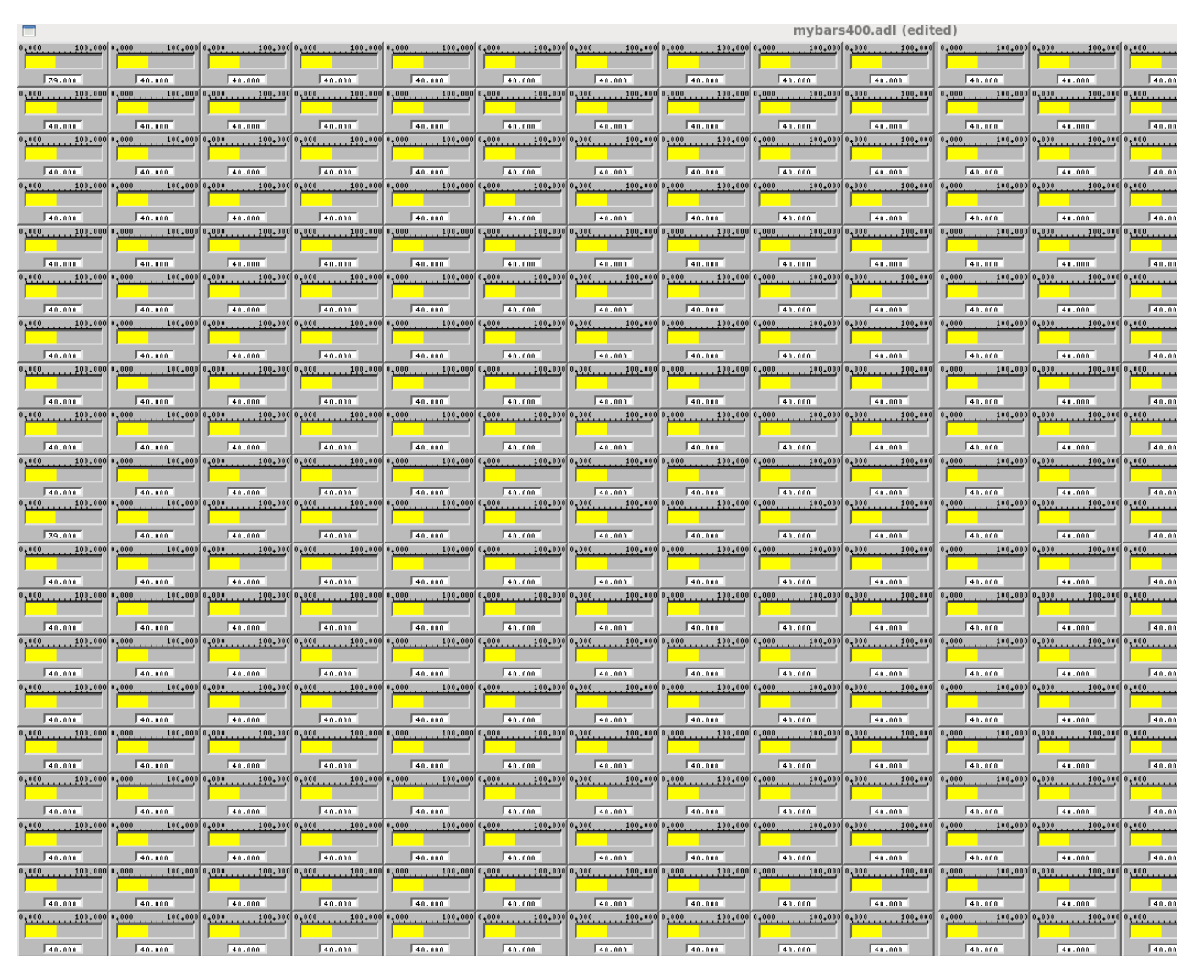
Fig 2 Medm - Graphical

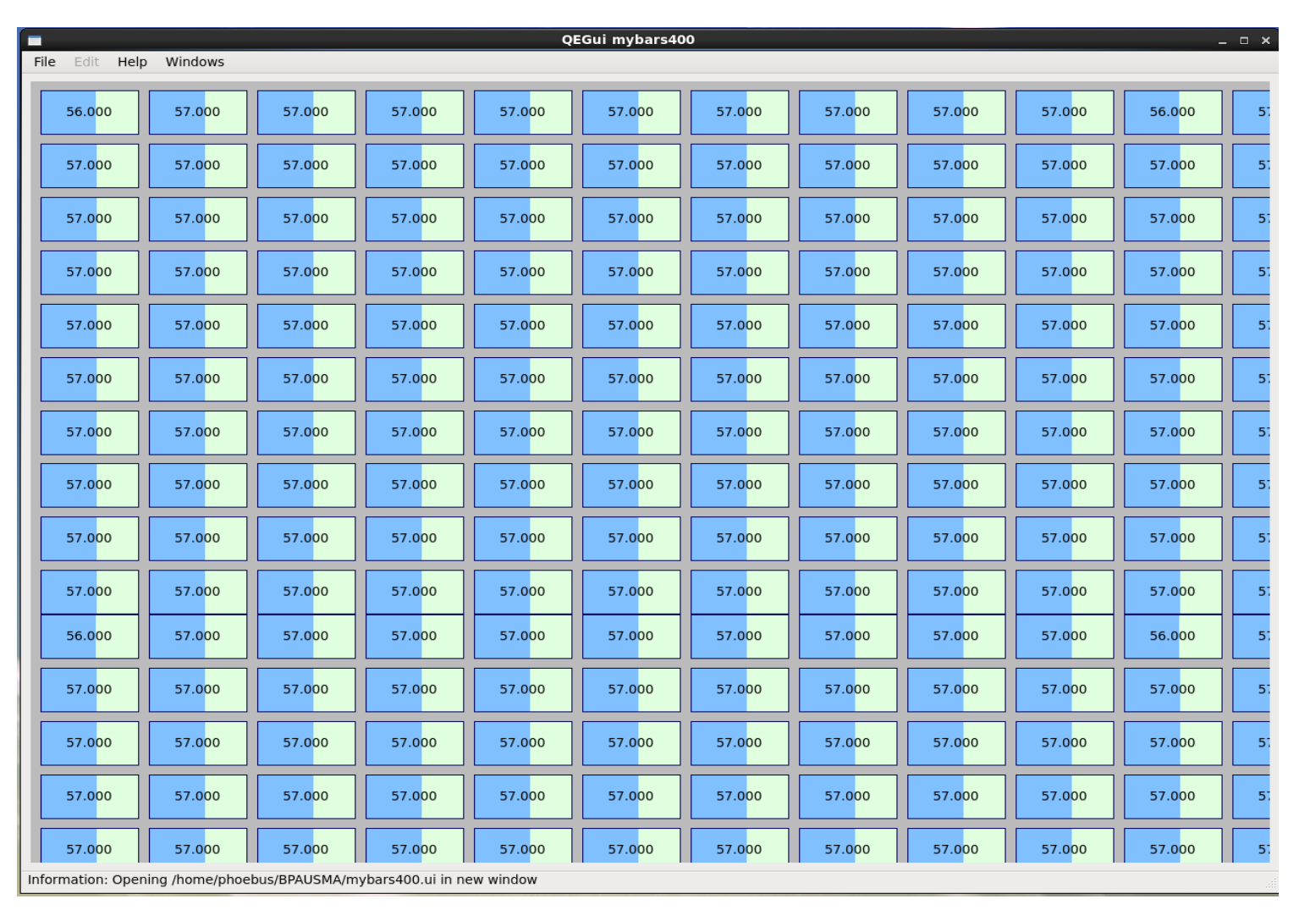
Fig 4 EPICSQt - text

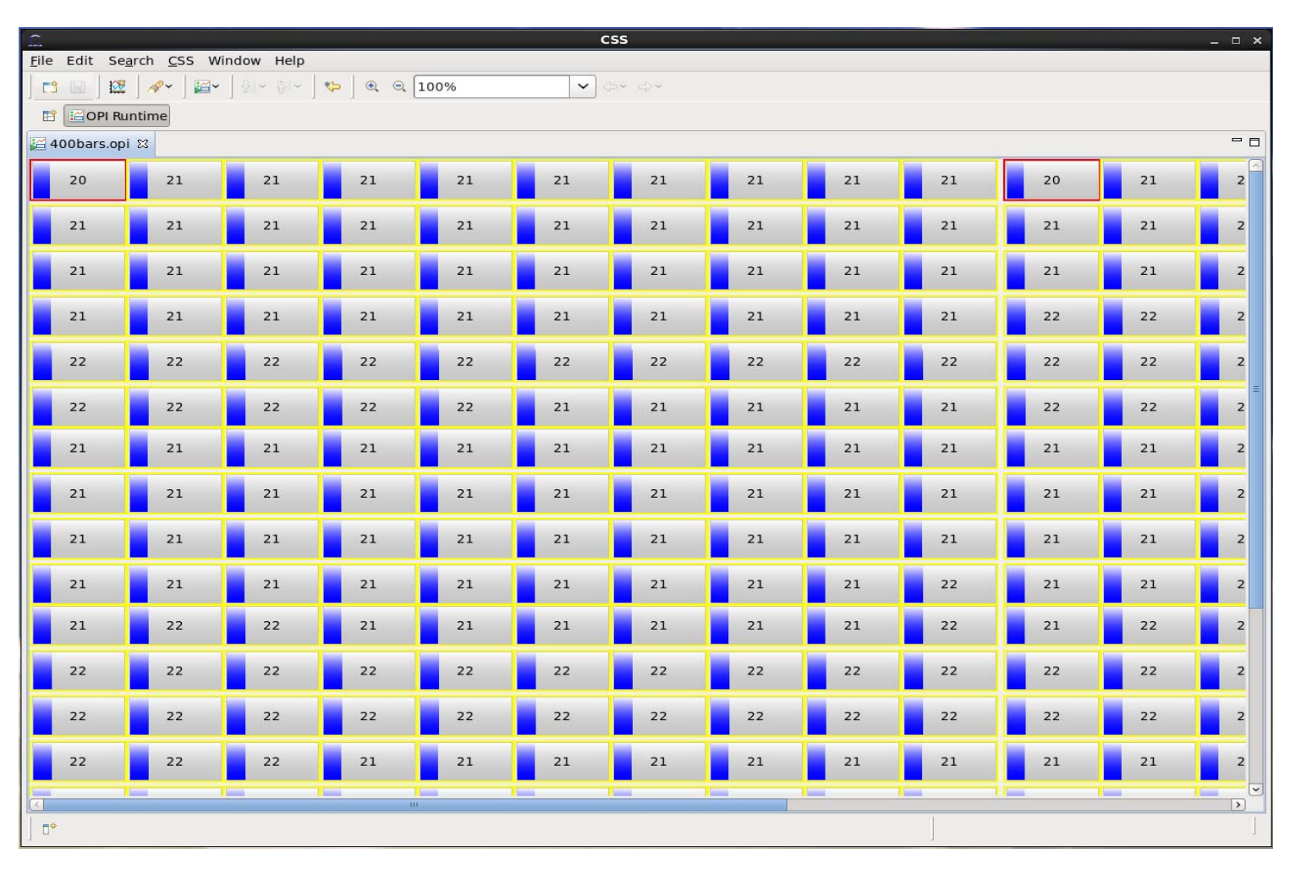
Fig 6 CSS - graphic