

# Hall A High Voltage EPICS Screens

## CS-Studio Phoebus

Brian Eng  
DSG  
2021/09/09

# Why Phoebus?

- No longer dependent on Eclipse
  - Standard Widget Toolkit (SWT) is one of the toolkits to create GUIs in Java
  - More than just SWT, Eclipse is an Integrated Development Environment (IDE)
    - Heavyweight (30+ min vs ~3 min compile time)
    - More code, e.g. 11.2 kloc vs 4.5 kloc for Channel Finder
- Since ~2016, various parts of CS-Studio have been migrating to using JavaFX as the GUI toolkit
  - Included with the Java Development Kit (JDK) since Java 11
    - also available separately for older JDKs
  - Phoebus has all components in JavaFX

# Why not stay with CSS-BOY?

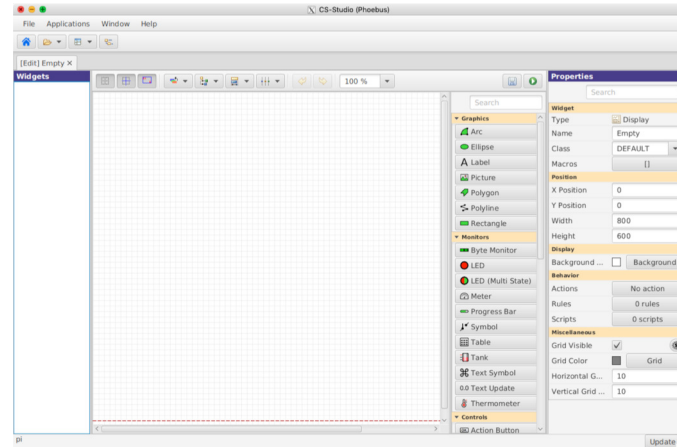
- Phoebus is more performant: less CPU % and less memory use
  - Varies based on screen, Heater Demo:  $\sim \frac{1}{4}$  CPU ( $\sim 40\%$  vs  $\sim 10\%$ ) and  $\sim \frac{1}{2}$  memory usage ( $\sim 9$  MB vs  $\sim 4$  MB)
- Phoebus can import and run BOY screens with no changes
  - Major exception are scripts
    - Generally, just need to change import path
    - Highly dependent on the script (one reason to avoid script use)
- Most BOY developers have moved to Phoebus
  - New widgets and features not in BOY

# Hall A EPICS Screens

- Hall C recently upgraded from some Tcl/Tk screens to CSS-BOY
  - Scripts to generate CSS-BOY done by DSG (Lemon & Bonneau)
  - Upgrading screens and creating new scripts for Hall A
- Initial focus will be on detector high voltage

# Creating Screens

- Phoebus screens are XML files with an extension of .bob (BOY is the same, but with .opi)
- One method of creating screens is with the built-in Display Builder editor
  - The default choice
  - Good when layout won't change
- Some Hall A screens need to be easily customizable (Example #1) or use external parameter files (Example #2)
  - Using Python to generate .bob files



# Python Example #1

- Hall A will have experiments that use a suite of detectors in slightly different configurations
- For the main menu, easier to generate the screen dynamically from a dictionary

```
Desktop — aslow@adaqsc:~/EPICS/HV/CSS — ssh -Y hallgw.jlab.or...
#!/usr/bin/env python3

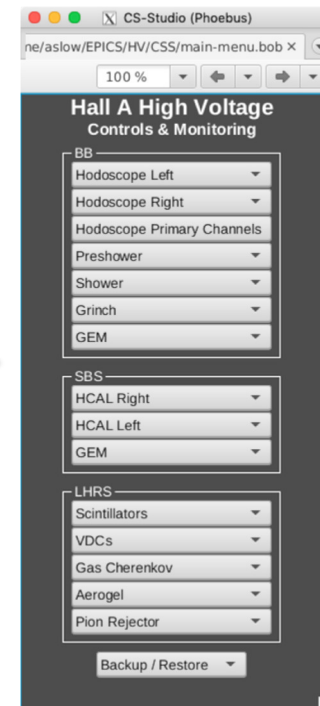
import sys
if len(sys.argv)>1:
    OUTFILE=sys.argv[1]
else:
    OUTFILE = "main-menu.bob"

ALL_PATHS=["../bb_all","../sbs_all"]

WIDTH = 300
# Width of the button groups
BWIDTH = 236
BGCOLOR = 'red="77" green="77" blue="77"'
FGCOLOR = 'red="255" green="255" blue="255"'

# Text Labels
TITLE = "Hall A High Voltage"
SUBTITLE = "Controls & Monitoring"

def makebuttonlist(spec,detlist):
    blist = []
    for det in detlist:
```



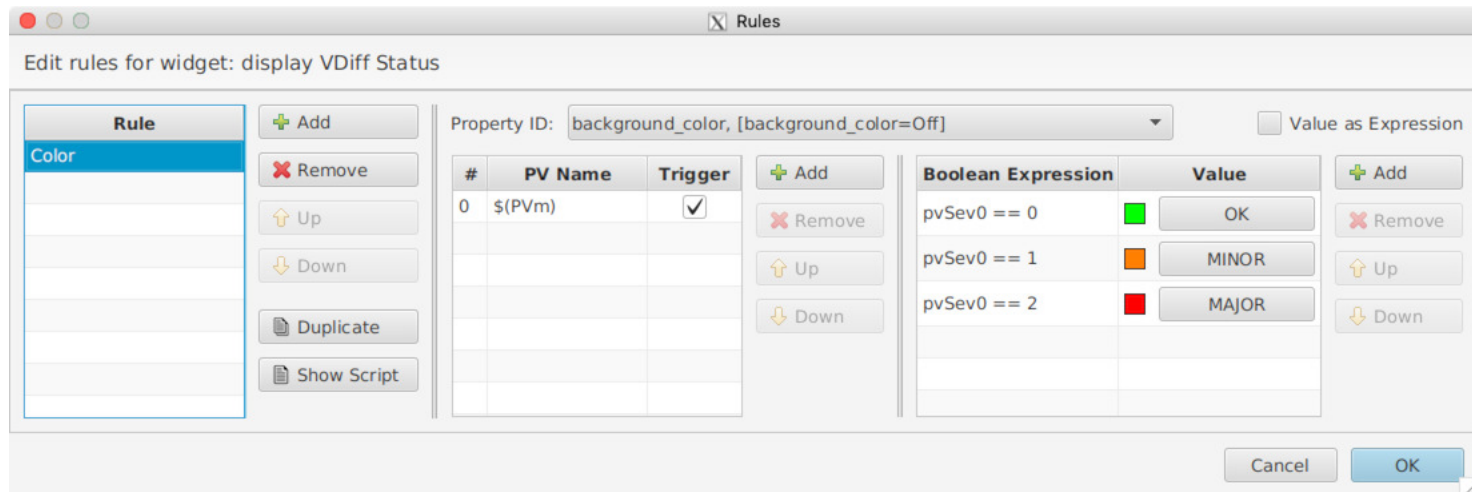
Each of these groups is a separate dictionary

# Python Example #2 (1/4)

- Some high voltage supplies have separate set and read process variable (PVs)
- Each high voltage channel has a calculated PV ( $V_{diff}$ ) that indicates difference between set and read
  - $|V_{set} - V_{read}|$
  - PV alarm limits
    - 25 V for LOW and HIGH
    - 50 V for LOLO and HIHI

## Python Example #2 (2/4)

- When using the monitor widget and its alarm status, default changes only widget border, based on alarm; request was for entire widget to change color
- vdiff.bob uses a rule to change widget background color based on severity level of the PV passed via macro





## Python Example #2 (3/4)

- A mapping file (high voltage name, PV, row, column, etc.) can be used to generate a grid of indicators to show Vdiff status, using Python
- To make the grid, multiple copies of vdiff.bob are embedded into the new detector screen and macros are used to define the individual PV for each monitor widget

# Python Example #2 (4/4)

- Read the high voltage mapping file with Python to generate a grid of embedded displays to create the detector screen

```
beng — pi@rpi: ~/hv — ssh -Y pi@rpi — 80x24
# BigBite Hodoscope HV Map file
# Created 2021-08-20 09:28:46 from HodoMapping_Mar2021.xlsx with ./BBHOD0parse.py
#
# Name Crate Slot Channel Group Row Column
HODO_0_Left 0 6 43 1 0 1
HODO_0_Right 0 0 1 2 0 2
HODO_1_Left 0 2 2 1 1 1
HODO_1_Right 0 0 2 2 1 2
HODO_2_Left 0 2 3 1 2 1
HODO_2_Right 0 4 45 2 2 2
HODO_3_Left 0 2 4 1 3 1
HODO_3_Right 0 0 4 2 3 2
HODO_4_Left 0 2 5 1 4 1
HODO_4_Right 0 0 5 2 4 2
HODO_5_Left 0 2 6 1 5 1
HODO_5_Right 0 0 6 2 5 2
HODO_6_Left 0 2 7 1 6 1
HODO_6_Right 0 4 44 2 6 2
HODO_7_Left 0 2 8 1 7 1
HODO_7_Right 0 0 8 2 7 2
HODO_8_Left 0 2 9 1 8 1
HV.hvc 1,1 Top
```

High Voltage Map File

+

```
beng — pi@rpi: ~/hv — ssh -Y pi@rpi — 80x24
mapf = open(HV_MAP, "r")
for line in mapf:
    # skip comment lines
    if (line[0] == "#"):
        continue

    spl = line.split()

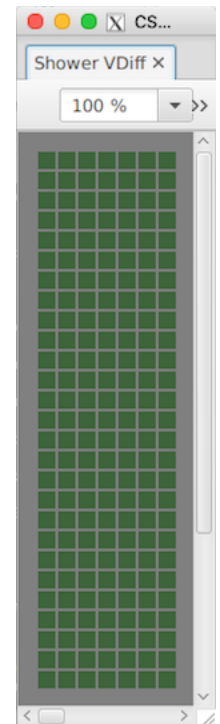
    # generate the PV name
    pv = 'HAHV{:s}:{:02s}:{:03s}:VDiff'.format(spl[1], spl[2], spl[3])

    # figure out which screen this should go to
    for i in range(0):
        if (re.match("^"+ss[i], spl[0])):
            arr[i].append([pv, int(spl[5]), int(spl[6]), spl[0]])

# width of the title group
tw = 120
# size of the LEDs
led = 13
# width of the "LEDs" + space
vdiff.py [+]
```

Python

=



Screen

# Conclusion

- Phoebus will be used for generating local displays going forward
- Current scripts will be modified and new scripts and screens will be created
  - Next step will be embedding detector Vdiff screens into an overall screen