Control System Studio Training
- BOY

Kay Kasemir
ORNL/SNS

kasemirk@ornl.gov

A lot of material from Nadine Utzel, ITER and BOY online help by Xihui Chen, SNS

Oct. 2012 EPICS Meeting, PAL, Korea
BOY – Best OPI, Yet

Operator Interface Editor

Similar to EDM, MEDM, SDS, DM2K, but possibly better

Runtime
Example: ITER
Examples: SNS

- Top-level displays created by operators
Examples: SNS
Examples: SNS “Steering” Tool

Try to get spot into the green, at least into orange

Tim Southern, Nick Luciano
Main Idea: Simple Things are Easy

1. Drag a widget, e.g. Knob, from palette to editor
2. Enter the PV name in Properties view
3. Click the “Run” button to execute!

What you will get

✓ PV value as text and via knob position
✓ PV severity reflected in border color
✓ PV name and value shown in tool-tip
✓ PV’s display limits set the knob’s default range
✓ Indicate ‘disconnected’ state via a pink border
✓ Widget will be greyed-out if read-only
Exercise: First Display

- Menu CSS,
  - Display, OPI Editor Perspective
  - Display, Install OPI Examples

- Navigator Context menu on CSS: New, OPI File, call it “first.opi”
  - Or Menu File, New, BOY, OPI File

- Locate in Palette: Monitors, Text Update
  - Drag’ Text Update onto display grid
  - Move widget around, resize

- Locate Properties View
  - Enter PV Name “sim://sine”

- Press Run button in Toolbar
Widget Palette Hints

Many widgets, hard to see them all

• Scroll

• Click on section header

• Try the ‘pins’

• Header Context menu offers **Columns** mode to display Widgets as small icons in columns
Widget Properties

- Widgets are configured by setting Properties in the Properties view.

- Common Properties:
  - Name
  - Position*
  - Background color
  - Border

- Widgets that read/write PVs:
  - Basic: PV Name
  - Border: Alarm Sensitive
  - Behavior: Limits from PV

* Position can also be modified by moving or resizing the widget in the editor, or via Toolbar buttons to align etc.
Exercise: Extend First Display

• Locate in Palette: Controls, Knob
• Drag Knob onto display
• Move Knob around, resize
• Locate Property PV Name for Knob
• Enter “sim://sine”
• Create another Knob:
  – PV Name = “loc://test”,
  – “Increment” = 0.1
  – “Limits from PV” = no
• Run 

• Note how the “sim://sine” Knob is really read-only, but you can change the “loc://test” PV via the Knob
OPI Files: Run or Edit?

- Default: Double-click on *.opi in Navigator opens in “OPI Runtime”, i.e. executes the display

- Context menu allows to select
  a) Editor to edit?
  b) Runtime to execute?

- Once you select “Editor”, that will become the double-click default
  - Select “Runtime” once to restore previous default
Exercise: Edit vs. Runtime Mode

- Close all CSS Editors (Menu File, Close All)

- In the Navigator, double-click on the first.opi that you created before
  - Does it open in the Editor or Runtime?

- In the Navigator, open the Context Menu on first.opi and select Open With, OPI Editor.
  - Close first.opi, now double-click the file in the Navigator. Does it open in the Editor?

- In the Navigator, open the Context Menu on first.opi and select Open With, OPI Runtime.
  - Close first.opi, now double-click the file in the Navigator. Does it open in the Runtime?
Exercise: Send PV to other CSS tools

- Run the OPI that you created
- Use CSS Process Variable context menu on a widget that displays a PV to open Probe
Example Displays

- Installed via Menu CSS, Display, Install OPI Examples

Note new project named BOY Examples
Double-click on main.opi file to open
Explore the examples

Check the “Start Up” page, which is similar to the first two exercises

Remember: You can Open With, .. Editor to see implementation
Exercise: Screen Navigation

- Similar to hyperlinks in a Web Browser:
  - Default: Linked display replaces the current display.
  - Zoom in/out, go “back” via toolbar:
  - Use context menu to open in ‘tabs’ or new Window

Try with OPI Examples: Open in tab, … Window

OPIs in ‘Tabs’
Exercise: Editing Features

Add, duplicate Widgets in various ways

- Drag & Drop from Palette
- Copy/paste, Ctrl+Drag existing widgets to duplicate

Arrange them on the display

- Snap to grid, guideline, other widgets
- Align, distribute

Select multiple widgets to

- Edit common properties
- Adjust size or move around
Hint: Drop PV Names

- Assume you have some text document with a list of PVs
- How to quickly create a display with Text Update widgets for these PVs?
  - Just drag the names into the display
  - Will be prompted for the type of widget
Exercise: View Online Help

- Find the “Widgets” section
Font and Color Names

When configuring a color (foreground, background, border, ...) or font (Text Update font, ...), you have two options:

a) Pick any color or font
   - RGB resp. Name, Typeface, Size

b) Pick a Predefined Color resp. Font
   - Remember BOY Preferences, Color and Font file?

Exercise: Explain why (b) is better.
Exercise: BOY Font, Color Preferences

Menu **CSS, Preferences:**

- Locate the BOY settings
- Assert that the **Color File, Font File, Top OPIs** settings use files from the BOY Examples that we just installed:
Exercise: Use Predefined Fonts

- Add a Label to your display
  - Set font to the predefined Title font
  - Set text to something like “This is the Title”

- Add another Label
  - Assert that it uses the “Default” font
Exercise: Schema File

- Create a new display file “schema.opi”
  - Add a Text Update
    - Background Color: Yellow
    - Foreground Color: Red
  - Save, close the schema.opi

- Menu CSS, Preferences, CSS Applications, Display, BOY, OPI Editor
  - Set the “Schema OPI” to the schema.opi that you just created

- Create a new OPI file
  - Add a Text Update widget
  - Notice its initial Background & Foreground color?
Portable Usage of Fonts

Fonts differ between operating systems: “Times New” vs. “adobe-times-..” etc.

How can an OPI file “Look the same” on Windows, OS X, Linux?

1. If possible, install the same fonts on all your computers
   – Microsoft “Office” fonts available on most Windows and Mac OS computers because they also run MS Office
   – MS Office fonts are also available for Linux! Google “free office fonts Linux”

2. BOY fonts.def file allows system-specific tweaks

   # Though using the same MS Office font
   # on all operating systems, the sizes seem
   # somewhat different.
   # Fix that by using different sizes for
   # each OS:
   Default=Verdana-regular-10
   Default(macosx_cocoa)=Verdana-regular-14
   Default(linux_gtk)=Verdana-regular-10

   # Same with “Header1”: OS X needs bigger font
   # for same on-screen pixel size
   Header1=Verdana-bold-24
   Header1(macosx_cocoa)=Verdana-bold-36
Preferences: Top OPIs, Sitewide settings

- **Top OPIs: Appear in Toolbar**

- **Path names for color & font files, “Top” OPIs, Schema can be web links**
  - Instead of `/BOY Examples/font.def`
  - use `http://some.server.org/path/font.def`

Good for site-wide files like your top-level control system screen!
Suggestions for your site

- After gaining some experience with BOY, somebody with design talents defines which colors, fonts, … to use for displays at your site

- Pick fonts that look similar on all operating systems

- Create color.def, font.def, schema.opi
  - Place these on a web server
  - Configure CSS for your site to use the http://… paths to the *.def and schema.opi

- You can put your *.opi files into CVS
  - or subversion, Mercurial, GIT, …CSS can include support for these

- Each night, you can publish the current *.opi files from CVS on your web server
  - Point the “Topi OPIs” to http://web.server/opis/main.opi
  - End users can now easily run the “current” version from the Toolbar
Macros

Usage: $(macro) or ${macro}

- Wherever you enter a widget property
- Most often used for (partial) PV name:
  - $(pv)_setpoint
  - $(pv)_readback

Such a display can then be invoked with
pv="PowerSupply1" or "PowerSupply2"

Diagram:

Linking.opi
pv=PS1
pv=PS2

Macros.opi
"$(pv)_setpoint"

Macros.opi
"PS1_setpoint"

Macros.opi
"PS2_setpoint"
Macro Definition

- Predefined Macros: Widget properties, see online help for name mapping
  - Property “X”: Macro $(x)
  - Property “Name”: Macro $(pv_name)
  - Automatic: Macro $(pv_value)
    - See default for the “Tool Tip” property

- User-defined:
  1. BOY Runtime Preference Setting (-pluginCustomization ....)
  2. User Preference settings (CSS, Pref..., ..App.., Display, BOY, OPI Runtime)
  3. Macro parameter of Action that opens the *.opi file
  4. Display *.opi file property “Macros”
  5. Grouping/Linking/Tabbed Container that wraps the widgets

Example:
Macro parameter of Action will override Preference settings.
Exercise: Linking Displays with Macros

- Create display file “Macros opi”
  - Label with Text “$(pv)”
  - Text Update with PV Name “$(pv)”

- Create display file “Linking opi”
  - Action button with “Actions” to “Open OPI”
    - Use File Path for Macros opi
    - Define Macros: pv= “sim://sine”
  - Add another action button (copy previous one)
    - Set macro to pv=“sim://ramp”

- Execute. Check that you can open the linked display

- Extra: Check OPI Examples, “4. Actions”
  - Can have more than one “Open OPI”
  - Any widget can have “Action”. Try Label.
  - Try Linking Container to display Macros opi within Linking opi
Miscellaneous

- Display has an “Auto Zoom” property
  - Size will adjust to fit window
Exercise: Grouping Container

In EDM, MEDM, … we needed lines and rectangles to visually group related displays.

In BOY there is the Grouping Container

• Create a display with Grouping Containers that look like this:

  Border Style=Group Box Style
  Name = Power Supply 1, Power Supply 2
  Add Labels “Setpoint:…”, “Readback:…”

• Note how you can
  – Move the Grouping Container an all its content
  – Move Labels inside and out of the container
Exercise: “Striptool” type Plots of PV over Time

Try both options

- Data Browser Widget
  - New Data Browser Plot, add PV
  - Set desired axis and time range
  - Save as *.plt
  - Add Data Browser Widget to BOY
  - Set its File Name to the *.plt

- XYGraph Widget
  - Behavior, Trigger PV: “sim://noise”
    - This PV updates once a second and will trigger plot updates
  - Primary X Axis(0), Time Format: “HH:MM:ss”
    - To get a “time” axis
  - Trace 0, Trace Type: Step Horizontally
  - Trace 0, Update Mode: Trigger
  - Trace 0, Y PV: Name of PV to plot

✓ Can also display archived data
✓ PV can be ‘monitored’, showing brief spikes
☐ Fewer display options

✓ Has many more display options
☐ Cannot show archived data
☐ PV scanned at update rate, can miss brief spikes
Widgets and Properties Galore

- Compared to EDM, MEDM, ... BOY tries to offer specialized widgets
  - **Grouping Container** instead of Lines
  - **LED** instead of Circle-with-color-rule
  - **Image Button** instead of Images with conditional visibility in front of invisible button
  - **Tabbed Container** instead of embedded window, many invisible buttons, conditionally visible graphics, local PVs to update the display inside the embedded window

- .. with many Properties
  - Alarm sensitive Border/Background/…
  - Precision, Limits, … from PV or direct entry
  - Actions
Widgets and Properties Galore because..

Display file describes **Meaning**: LED to display something, not Circle that happens to change color. Group of **related widgets**, not rectangle that happens to surround something. Border color to reflect **alarm state**, not arbitrary change in color. Font name “**Title**”, not “Arial-bold-12”.

Displays with same **Representation** (Lines, circles with changing color, “**Arial-bold-12**”) look the same as displays with **Meaning** (group, LED, Title). But they are like compiled binaries without source code. Less useful in the long run.

*In the future, files with Meaning will be easier to translate for other, new tools than files with only Representation.*
Rules & Scripts: Disclaimer

... can change any property of any widget:

- Change text of label based on a PV
  - i.e. build your own Text Update
- Change color of an Ellipse based on PV
  - i.e. build your own LED

Based on last slide, that is a bad idea!

Still, there are places where rules and scripts can be very powerful.

A BOY display with Rules/Scripts can replace a custom Java/Python/C/C++ application!
Rules, Scripts

Rules create dynamic displays
- Easy: PV $\rightarrow$ Widget Property

Scripts can to “anything”
- Read PVs,
  - change widget properties,
  - open dialog, ...
- JavaScript or Python (Jython)
Exercise: Rule to change color of Ellipse

- Create *Ellipse* widget
- Locate its *Behavior, Rules* Property
- Click the “no rule attached” value to open the dialog to Attach (or edit) Rules
- Add a rule that changes the background color as shown between Red and Green, triggered by changes in the sim://sine PV
- Press “See Generated Script”, compare with screenshot
- Maybe add another TextUpdate widget to display the same sim://sine PV
- Run the display
Rules vs. Scripts

Rules

- are simpler: One or more PVs change one property
- are closer to describing Meaning
- are internally converted to scripts, but what’s saved in the *.opi file is the Meaning: Property to adjust, expressions for rule, input PVs
- should be preferred to scripts whenever possible

Scripts

- can be pretty much any Java Script of Jython code
- can affect multiple properties, widgets, even add and remove widgets
- should be used with care, because they can be hard to maintain in the long run
  - Use org.cstudio.opibuilder.scriptUtil (PVUtil, ColorFontUtil)
  - Add many source code comments
Exercise: Rules, Scripts in OPI Examples

- Open BOY Examples/5_3_Rules_Script.opi, first in Runtime, then in Edit mode

- Check the rules behind the “Left Win!” text above the two knobs

- Check the Script attached to the left Knob

- Check the Script attached to the moving circle
  - How does it change its color?
Exercise: Script-generated Displays

- Open BOY Examples/Miscellaneous/DynamicLoadWidgets/LoadWidgetsExample.opi in Runtime mode.
  - Notice a difference?
- Open SubPanel.opi in Edit mode, change it slightly by setting the color of the “Group…” label to violet, save, then press “Load” on LoadWidgetsExample.opi.
  - See how it’s using the current version of SubPanel.opi?

Investigate how this is done!

- What PV is attached to the text field where you enter the *.xml file names?
- What PV is attached to the “Load” button?
- Note the script attached to the big Grouping Container that appears empty in edit mode, but is dynamically populated with copies of SubPanel.opi in runtime mode.
- Read that script together with myConfigExample.xml. Writing such a script requires knowledge of the BOY widget model. You don’t have to write such a script, but you should be able to understand what it does.
Scripts can replace custom Applications!

Display how beam loss is increased or reduced relative to a “snapshot”

Save, Adjust, maybe Restore settings

SNS operation group: Tim Southern, Nick Luciano
Summary

There is a lot you can do in BOY

– Macros, Rules, Scripts, …

Remember the Main Idea:

Simply Things are Easy

1. Add widget
2. Enter PV Name
3. Run