CODAC Operator Interface

Software Test Plan (STP)
Based on QA Template Version <1.0>

This document describes the tests that should be performed for CODAC Operator Interface in order to be installed as part of Core System release. Different test cases are described, as well as and test pass-fail criteria.

Contents

1 INTRODUCTION

1.1 Purpose

This document describes the tests that should be performed for CSS BOY - Best Operator Interface (OPI), Yet - in order to be installed as part of CODAC Core System. These tests will ultimately compare the capabilities of BOY against these described in CODAC System Requirement (SRD) Document [IDM 28C2HL].

The Operator Interface is the graphical user interface that displays live control system data to operators and allows them to input data to the control. Particular functions to be tested are Operator Interface (OPI) development and runtime features.

1.2 Scope

The test items are:

- The operational version of BOY,
- The data, including all the configuration data needed to run the operator interface,
- The documentation, including the online help and the release notes.

The installation and uninstallation of the components are not part of this test plan.

1.3 System/Software overview and key features

BOY is a development and runtime environment:

- Edition of Operator Interface screen using graphical widgets and CODAC electrical and fluid standardised symbols. BOY Editor has most of the modern editing features which facilitate the OPI editing greatly, such as copy, paste, undo, redo, arrange multiple widgets, snap to grid or other widgets (geometry), ruler, guide, zoom in/out, change order, create…
- Open and run the Operator Interface, draw widgets on the screen, connect to the control system, update for example text fields, meters, and bar graphs to reflect the current values of EPICS PVs, offer dials or text boxes to enter or adjust PV values, which are then written to the control system.

1.4 References

CODAC Quality assurance plan - https://user.iter.org/?uid=6J7RW4

2 DETAILS OF THE TESTING PROCESS

2.1 Definition of test levels

The described component tests will focus on the desired features of CODAC Operator Interface.

Following test levels are defined in this test plan to organize the testing activity.

Operator Interface Edition Component Test	**EDT**
Test of the operator interface creation	
Operator Interface Runtime Component Test	**RNT**
Test of running the operator interface	
Symbol Library Component Test	**SMB**
Test of archived data plot in CSS and Web Data Browser	
Operator Interface Performance Test	**PRF**
Test of at least 1K widgets running on the operator interface	

2.2 Test administration

2.2.1 Anomaly resolution and reporting

Anomaly Reports shall be submitted in Bugzilla.

2.2.2 Test reporting requirements

The test logs shall be generated to record the outcome of test procedures as described in section *.4 and *.5 of the level test plans.

2.2.3 Test deliverables

The test deliverables include:

- Component Test Logs / Reports
- Anomaly Reports with Bugzilla bug references.

Test input data are registered in SVN code repository.

No other test tool is needed.

The test reports may be submitted on ITER IDM.

3 COMPONENT TEST PLAN

3.1 Scope

3.1.1 Test items and their identifiers

CODAC Operator Interface is included the following unit:

- m-css in the product:
  - org.csstudio.iter.css.product/

CODAC Operator Interface mainly depends on:

  - org.csstudio.iter.opibuilder.feature

CODAC Operator Interface includes ITER specific development:

  - org.csstudio.opibuilder.widgets.symbol
  - org.csstudio.opibuilder.imagelib

3.1.2 Features to be tested

The main CODAC Operator Interface features to be tested are:

- Operator Interface Edition and Run
- CODAC Standard Symbol Library

3.1.3 Features not to be tested

Scripting using Python and Javascript will not be tested.

3.2 Approach

3.2.1 Testing Methods

The overall approach for the level of testing is the Black box method to test the functionality of CODAC Operator Interface.

3.2.2 Item pass/fail criteria

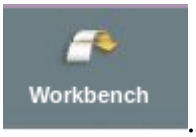
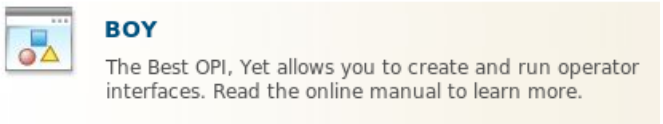
Each major anomaly found determines whether each test item has passed or failed testing.

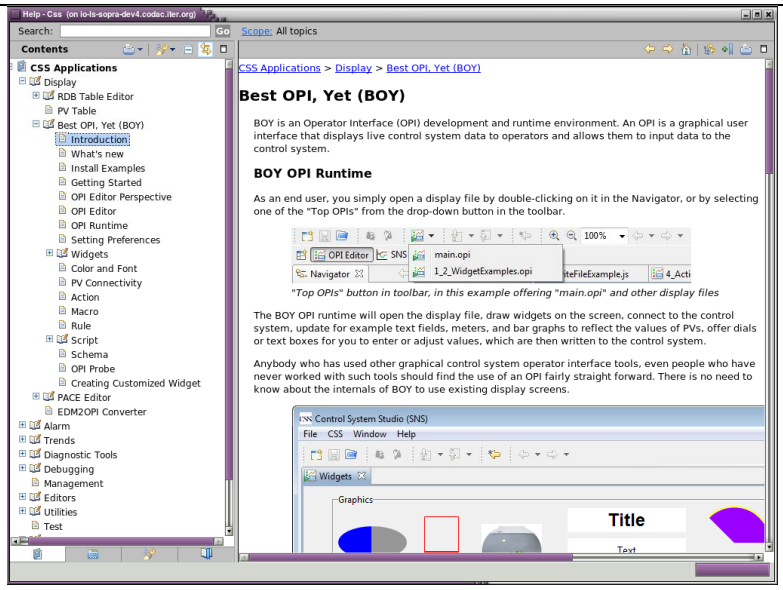
3.3 Environment / Infrastructure

Core System in its development role version should be installed on a CODAC standard machine. Access to SVN is required.

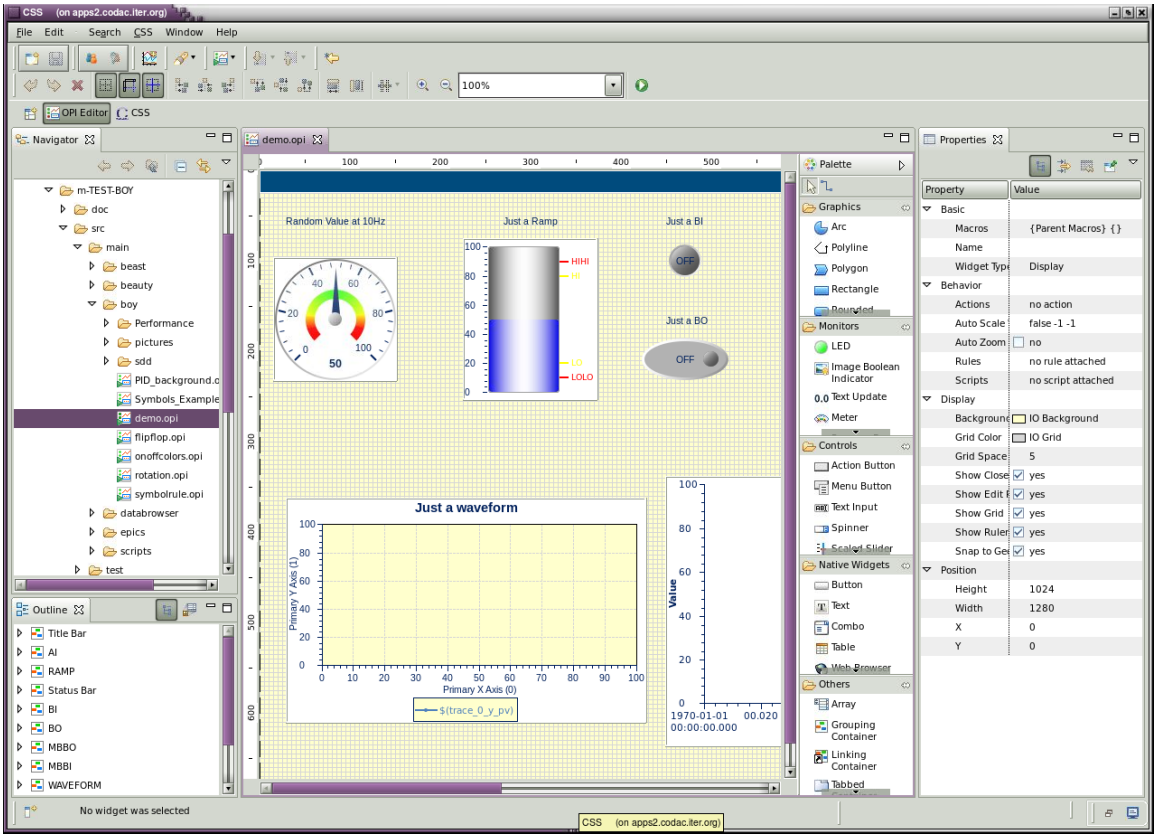
3.4 Component Test Procedures

EDT-01	**3.4.1 Operator Interface Edition**
Prerequisite	In a Linux console, download and start a demo IOC: 1.$ mkdir test 2.$ cd test 3.$ svn co https://svnpub.iter.org/codac/iter/codac/dev/units/m-css/trunk/products/ITER/products/org.csstudio.iter.css.product/demo/m-TEST-BOY A m-TEST-BOY A m-TEST-BOY/.project A m-TEST-BOY/doc A m-TEST-BOY/doc/STP-CODAC_Operator Interface.pdf A m-TEST-BOY/src … A m-TEST-BOY/sdd.xml A m-TEST-BOY/pom.xml Checked out revision xxxx. 4.$ cd m-TEST-BOY 5.$ softIoc -s -d src/main/epics/TEST-BOY0App/Db/PSH0-TEST-BOY0.db Starting iocInit ## ## EPICS R3.14.12.3-rc1 $Date: Mon 2012-12-03 16:39:27 -0600$ ## EPICS Base built Dec 10 2012 ## cas warning: Configured TCP port was unavailable. cas warning: Using dynamically assigned TCP port 37073, cas warning: but now two or more servers share the same UDP port. cas warning: Depending on your IP kernel this server may not be cas warning: reachable with UDP unicast (a host's IP in EPICS_CA_ADDR_LIST) iocRun: All initialization complete 6. List of all defined PVs in this IOC: epics> **dbl** TEST-BOY0:AI1 TEST-BOY0:AI2 TEST-BOY0:BI TEST-BOY0:BO TEST-BOY0:RAMP1 TEST-BOY0:RAMP2 TEST-BOY0:RNDM-AI TEST-BOY0:RNDM-BI TEST-BOY0:RNDM-MBBI TEST-BOY1:RNDM-STATE TEST-BOY0:COMPRESS TEST-BOY0:LONGIN TEST-BOY0:MBBI TEST-BOY0:SWITCHSTATE TEST-BOY0:CMDSWITCH TEST-BOY0:MBBO TEST-BOY0:STRING TEST-BOY0:SETPOINT TEST-BOY0:WAVEFORM epics>
Test Cases	1. Positive confirmation of the operator interface edition
Procedure	In another Linux console, start the development environment to edit a new operator

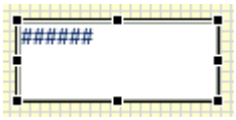
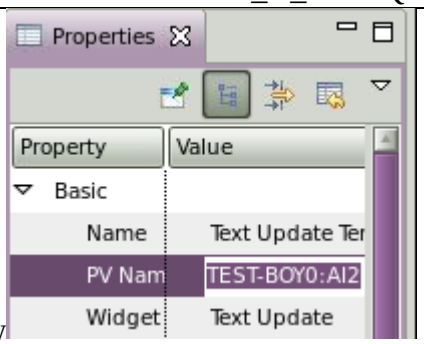
	interface: 1.$ cd test 2.$ css& 3. Browse to select the working directory test Check the Welcome Pages and online Help: 4. From the "Welcome to CSS for ITER!" Page, click on "First Steps". A short description of CSS BOY should be given. Click then on the link "Boy" in this "First Steps" page, just before the short description. The Online Help is displayed. 5. Close the Online Help windows and Close the Welcome screen by clicking on Workbench icon:  . Import the project 6. In the Navigator View, right-click and select the contextual menu Import… -> General -> Existing Projects into Workspace. Click on Next>. To specify the root directory, use the Browse button and select m-TEST-BOY and click on OK. Click on Finish 7. Open the Editor Perspective: menu Window -> Open Perspective -> Other… and select OPI Editor 8. In the Navigator View, browse m-TEST-BOY -> src -> main -> boy. Right-click on demo.opi -> Open With -> OPI Editor 9. From the Palette -> Monitors, use the little arrows to browse the widgets, drag and drop the widget "Text Update" on the grid – for instance below the Gauge displaying a random value at 10Hz 10. Select the new widget on the grid, and in the Properties View, modify the PV Name to "TEST-BOY0:AI2". Click on Enter 11. Save the modified OPI file using the menu File -> Save (Ctrl+S).	
Pass Criteria	4. Welcome First Steps for CSS BOY should appear:  After clicking on BOY link, the Online Help is opened on the BOY topic:	

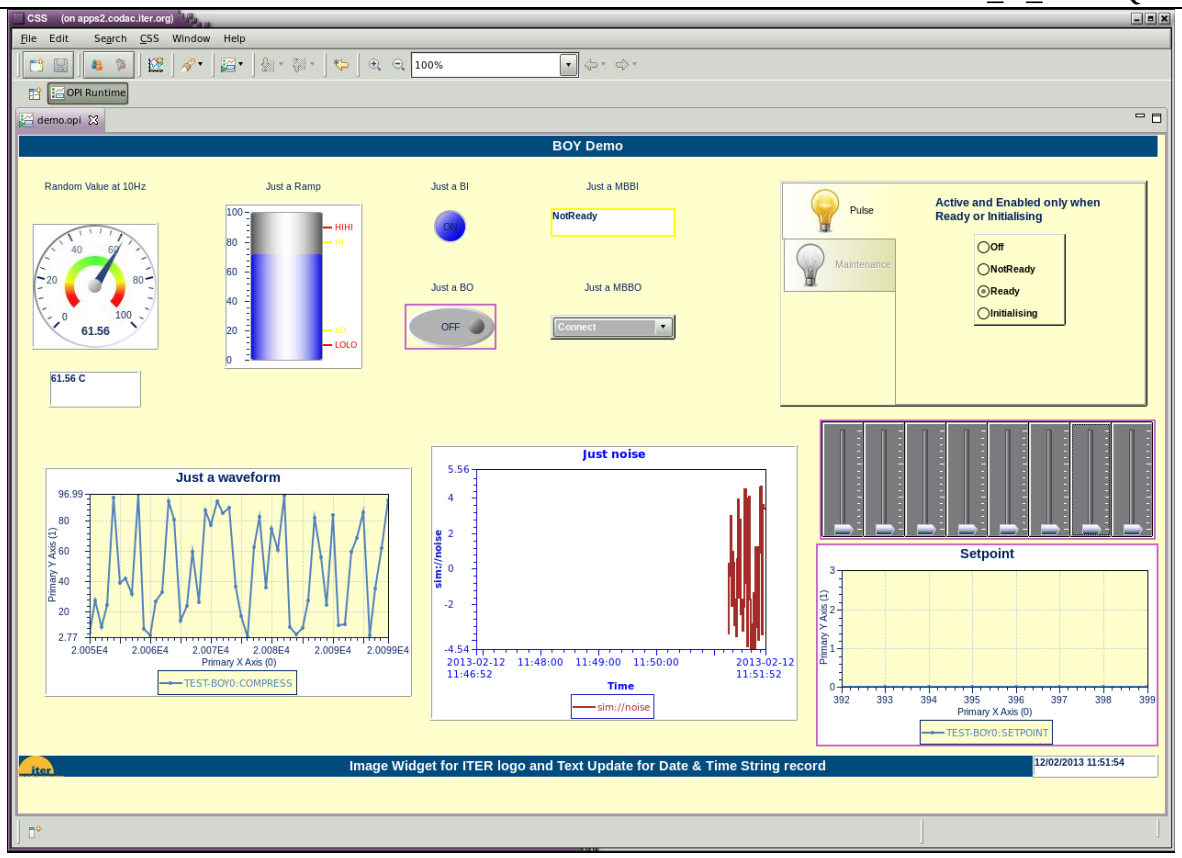


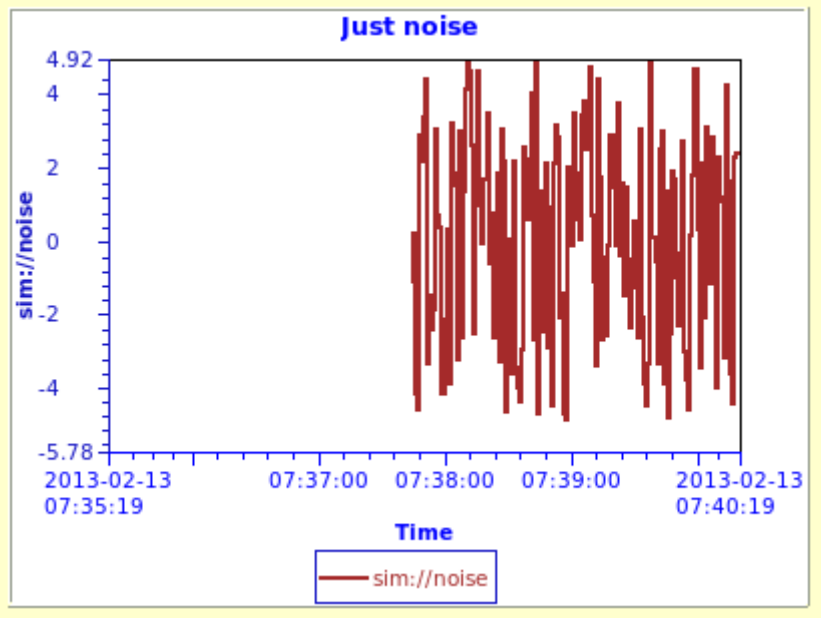
8. The demo.opi BOY screen should be displayed in CSS OPI Editor Perspective:



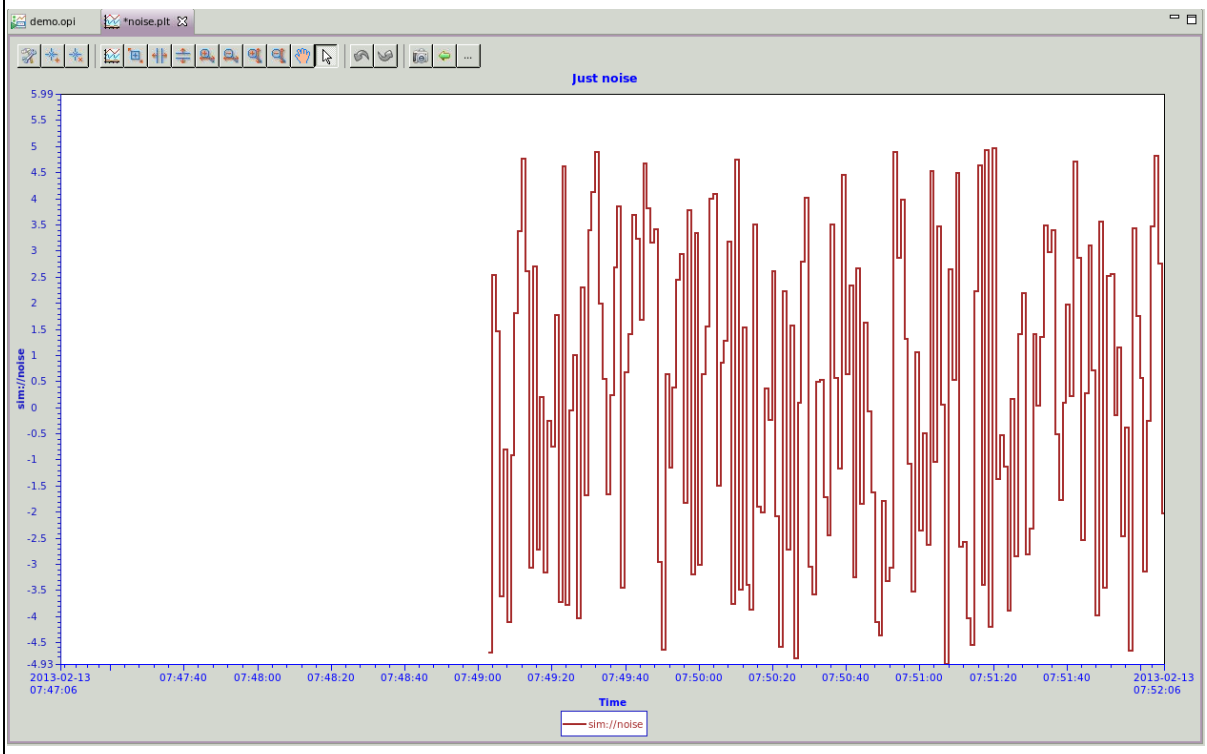
10. The Text Update widget should appear on the grid and its basic Properties should refer to the PV Name to be displayed:

	Widget on the Grid  Properties View 	
RNT-01	**3.4.2 Operator Interface Runtime**	
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened	
Test Cases	1. Positive confirmation of the operator interface running	
Procedure	1. Run the Operator Interface using the button from the tool bar  . Check on the Operator Interface the Title displayed using a Label widget, the AI random value displayed with a Gauge and its border alarm sensitive, the CALC record producing a ramp displayed using a vertical “Progress Bar” widget and its alarm limits, the BI displayed with a LED widget, the BO represented by a Button, the MBBI value displayed with a “Text Update” widget, the MBBO presented as a Combo box, the Waveform of 50 elements displayed on a “XY Graph”, the ITER logo displayed using an Image widget and the Date & Time String record. Finally check the new widget displaying TEST-BOY0:AI2 at 10Hz. The Tabbed Container, Data Browser and Array widgets will be tested afterwards.	
Pass Criteria	1. The Operator Interface to monitor the different types of records should be at Runtime:	

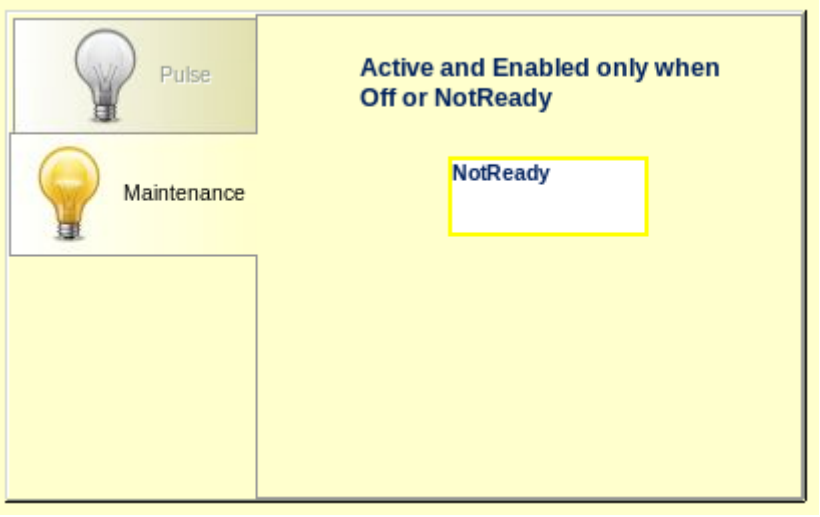
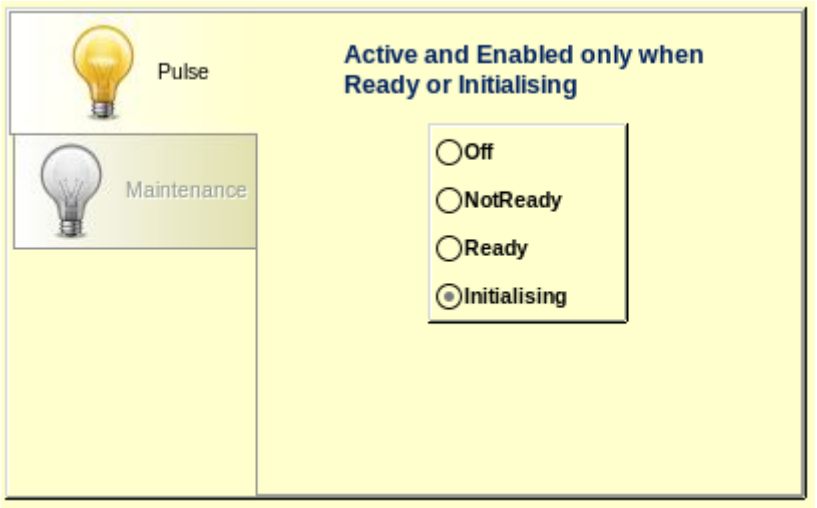
		
EDT-02	**3.4.3 Data Browser widget**	
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened	
Test Cases	1. Positive confirmation of the integration of a plot configuration into BOY	
Procedure	In CSS, from the previous ran Operator Interface demo demonstrating some BOY widgets: 1. Check the noise signal 2. From the bottom bar, click on  icon, select Other…, then General -> Navigator  A Fast View Icon for the Navigator appears in the bottom bar  3. From the Navigator View, double-click on m-TEST-BOY -> src -> databrowser -> noise.plt to open the plot	
Pass Criteria	1. The simulated noise signal should be displayed in the previous ran OPI:	

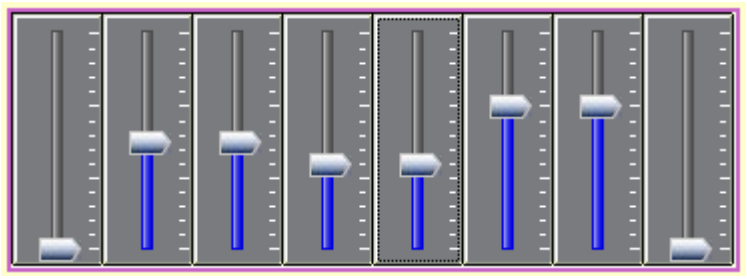
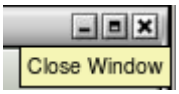
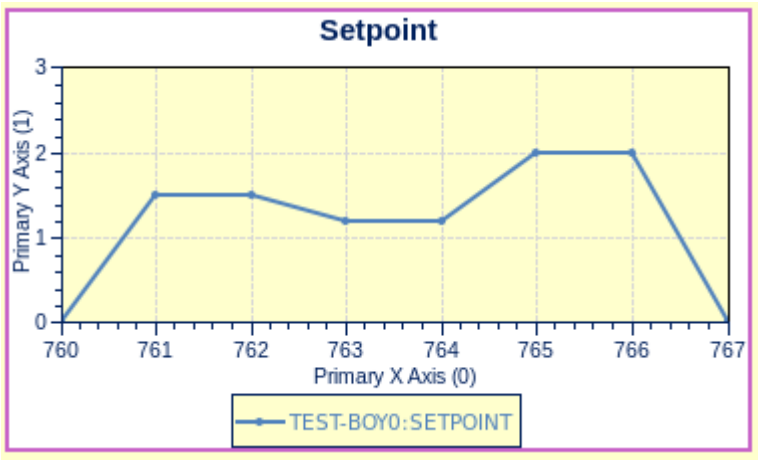


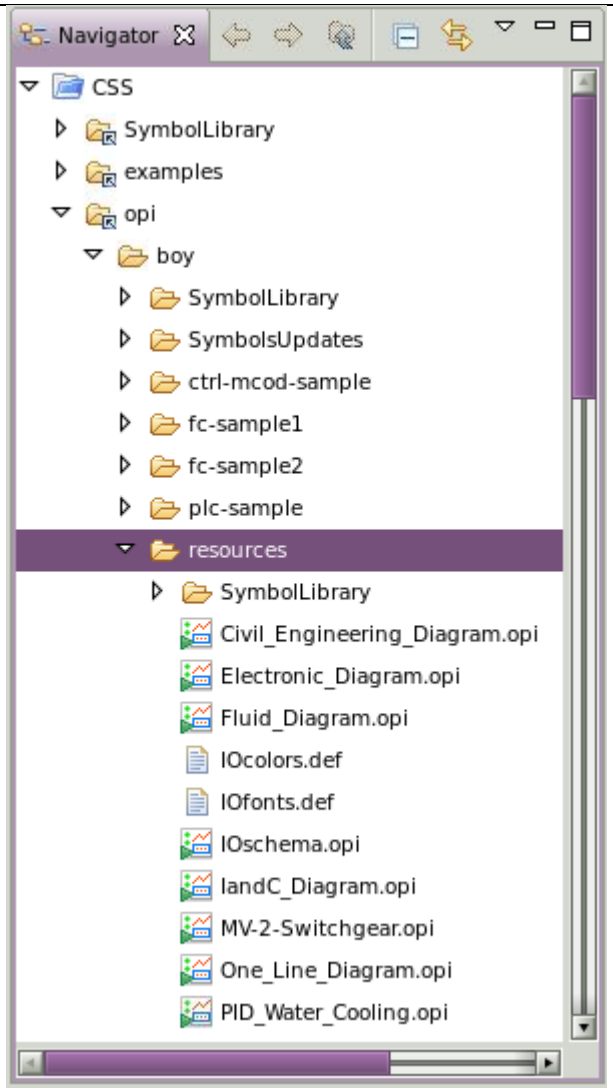
3. The plot should have the same configuration as the noise signal displayed in BOY: same Title label, colour and font, same axis properties and same noise signal:



EDT-03	**3.4.4 Vertical and Enabled Tabbed Container**	
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened	
Test Cases	1. Positive confirmation of the horizontal position and enable/active tab	

Procedure	In CSS, from the previous ran Operator Interface, click on demo.opi tab: 1. Check the vertical position of the tabs 2. Check that the Maintenance tab is activated only when the state is Off or NotReady 3. Check that the Pulse tab is activated only when the state is Ready or Initialising	
Pass Criteria	1-2. The 2 tabs are in vertical position and Maintenance tab is activated when NotReady:  3. Pulse tab is activated when Initialising: 	
EDT-04	**3.4.5 Setpoint adjustment**	
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened	
Test Cases	1. Positive confirmation of the setpoint adjustment	
Procedure	In CSS, from the previous ran Operator Interface demo.opi tab: 1. Adjust the sliders to modify the setpoint: first slider remains 0.0, the second and third are	

	1.5, fourth and fifth are 1.15, sixth and seventh are 2.0 and the eight one remains 0.0:  2. Check that the setpoint waveform 3. Just close the Runtime environment by clicking on the cross close button of the window: 	
Pass Criteria	2. The setpoint should look like that: 	
SMB-01	**3.4.6 CODAC Standard Symbol Lib**	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the standard symbol library	
Procedure	In CSS, in the OPI Editor perspective, run the Operator Interfaces demonstrating for example the use of all electrical and fluid symbols: 1. In the Navigator View, browse CSS -> opi -> boy -> resources which point to the location /opt/codac/opi/boy/resources	



2. From the Navigator View, double-click on Fluid_Diagram.opi file to display all fluid symbols defined in the library. In order to have a better view, use the runtime dedicated Perspective: Window -> Open Perspective -> Other… -> OPI Runtime

3. Switch back to OPI Editor Perspective and from the Navigator View, double-click on One_Line_Diagram.opi file to display all electrical symbols defined in the library. Switch to OPI Runtime Perspective

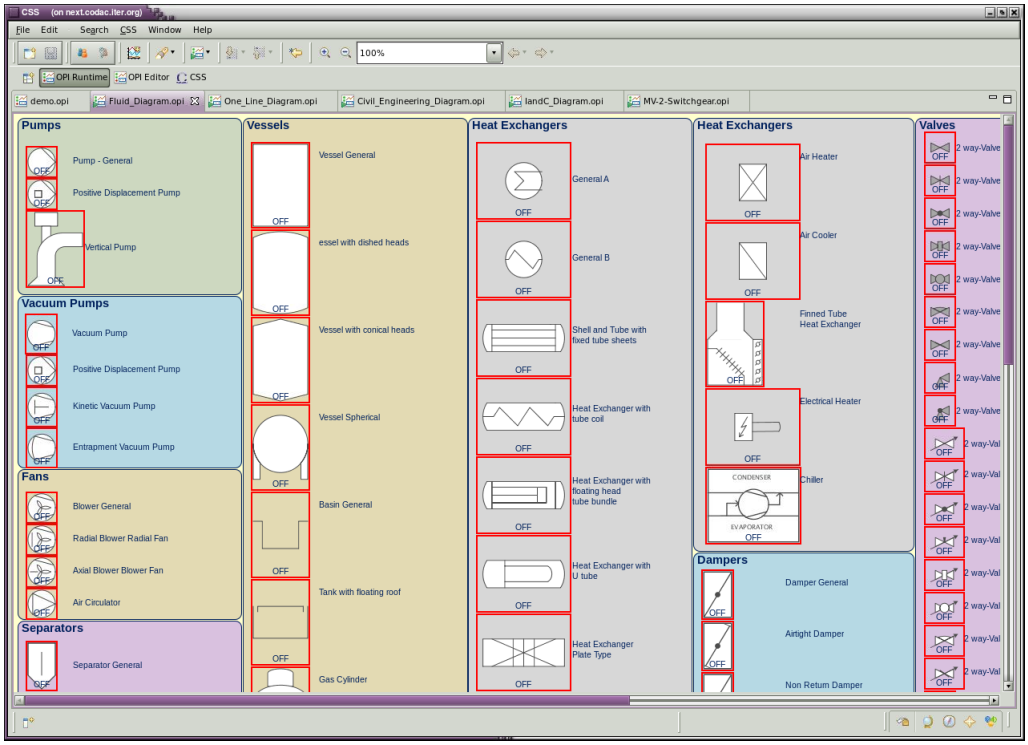
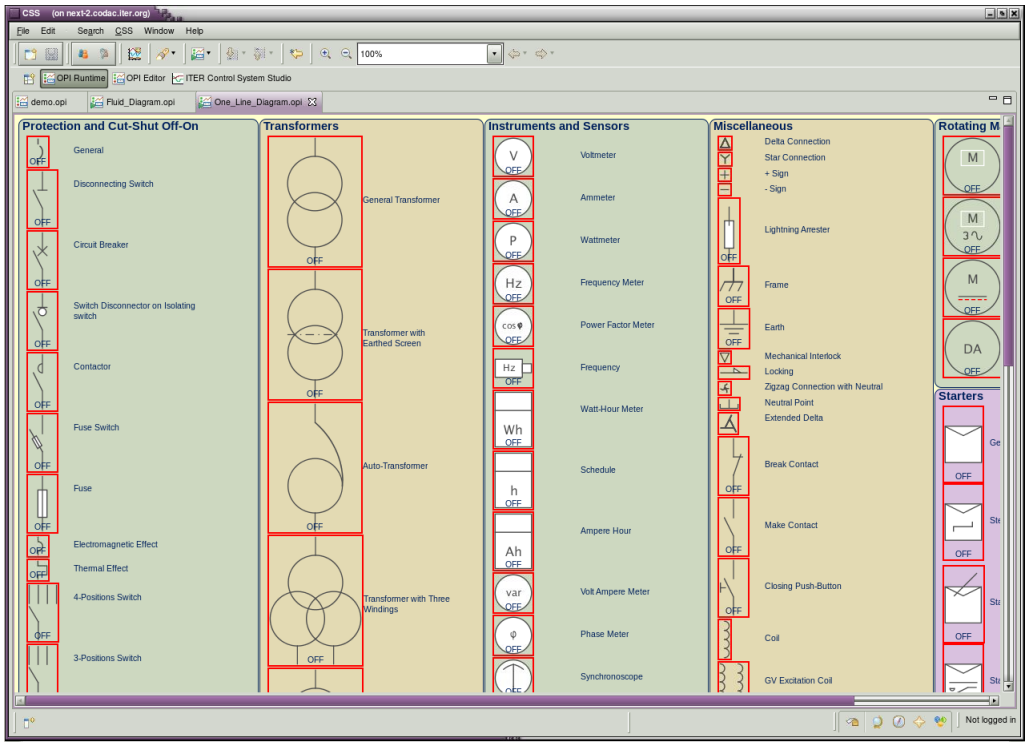
4. Switch back to OPI Editor Perspective and from the Navigator View, double-click on Civil_Engineering_Diagram.opi file to display the related symbols defined in the library. As the BOY screen is small, there is no need to switch to OPI Runtime Perspective to have a better look

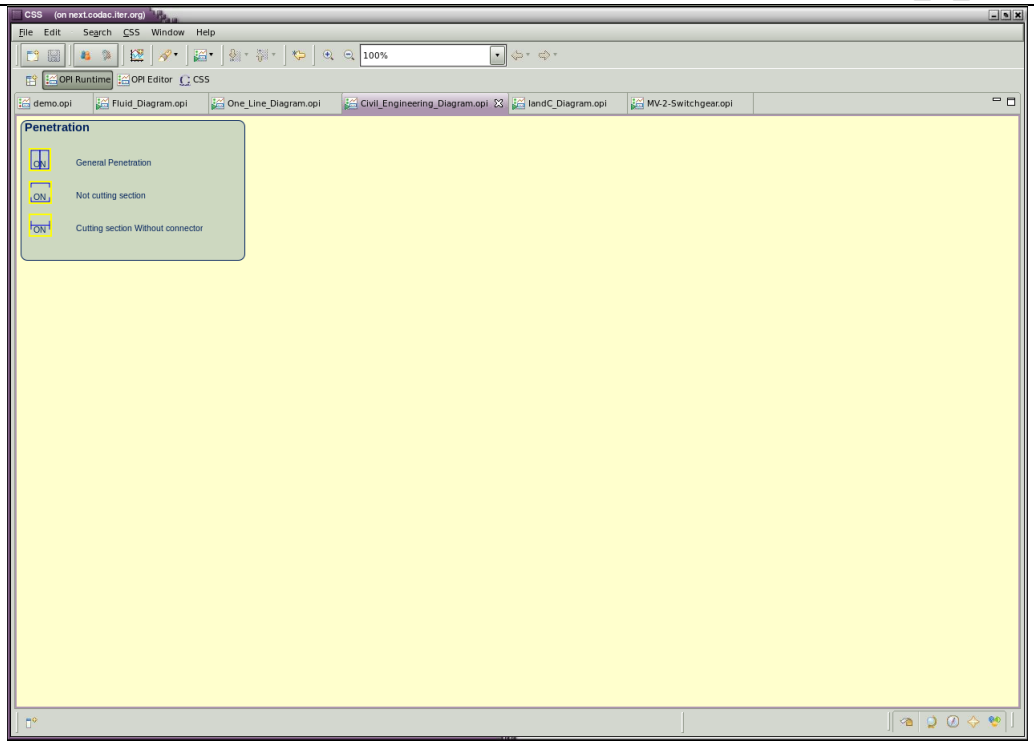
5. From the Navigator View, double-click on IandC_Diagram.opi file to display the related symbols defined in the library

6. From the Navigator View, double-click on MV-2-Switchgear.opi file to display an electrical use case schema. Switch to OPI Runtime Perspective

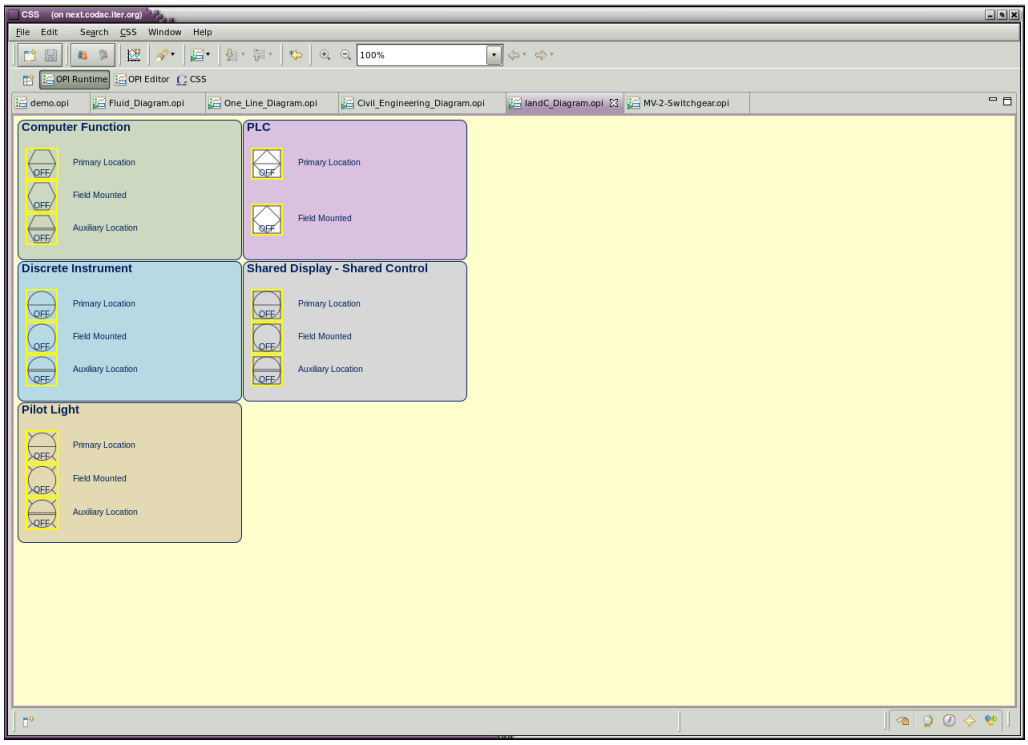
7. Switch back to OPI Editor Perspective and from the Navigator View, double-click on PID_Water_Cooling.opi file to display a fluid use case schema. Switch to OPI Runtime Perspective

8. Switch back to OPI Editor Perspective and from the Navigator View, double-click on

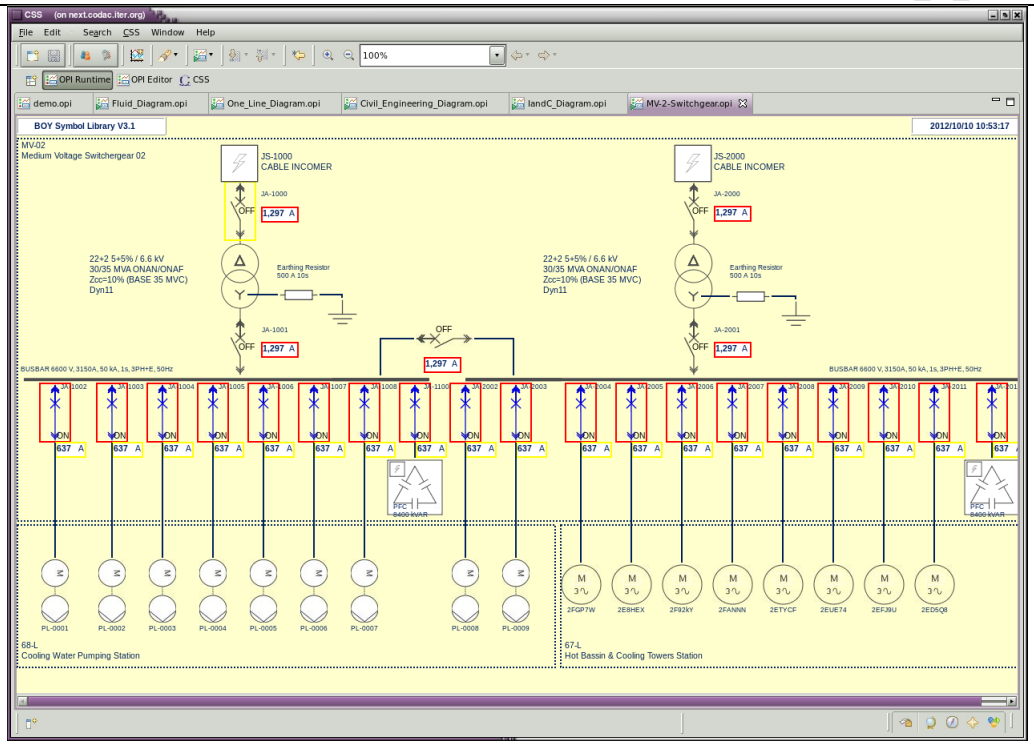
	IOschema.opi file to display the default layout of all BOY widgets for CODAC. Switch to OPI Runtime Perspective 9. Switch back to OPI Editor Perspective and close all screens opened	
Pass Criteria	2. *After some seconds – the time to load the ~130 symbols*, the fluid symbols should be animated on the screen at 1 second:  3. The output for the electrical symbols should be:  4. The output for the civil engineering symbols should be:	



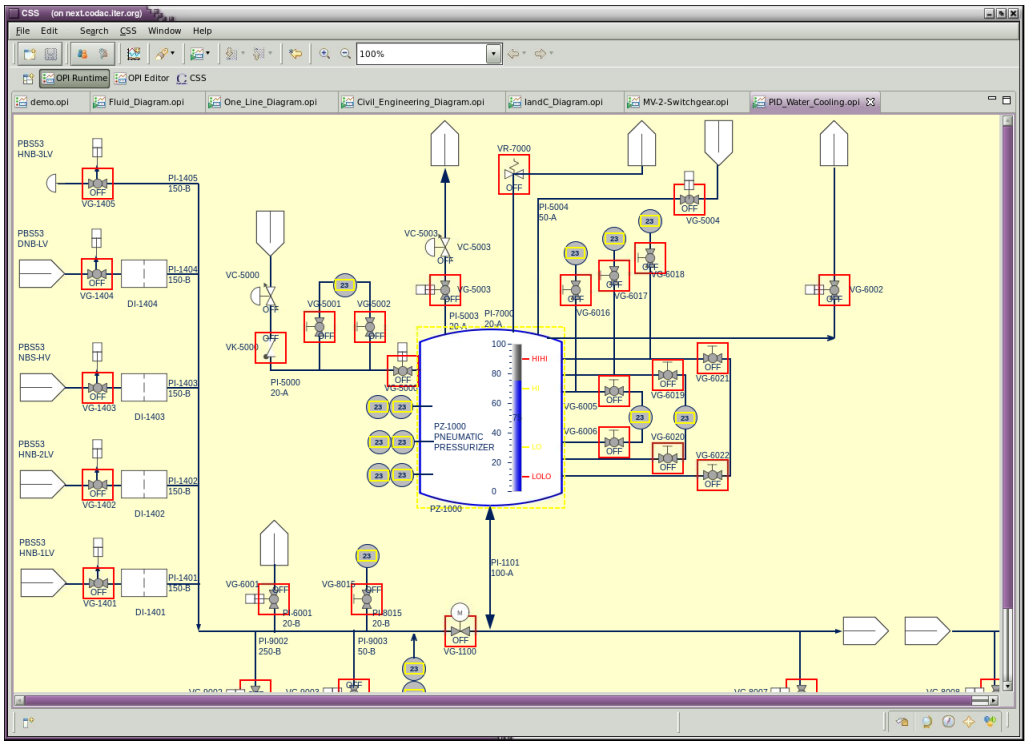
5. The output for the I&C symbols should be:



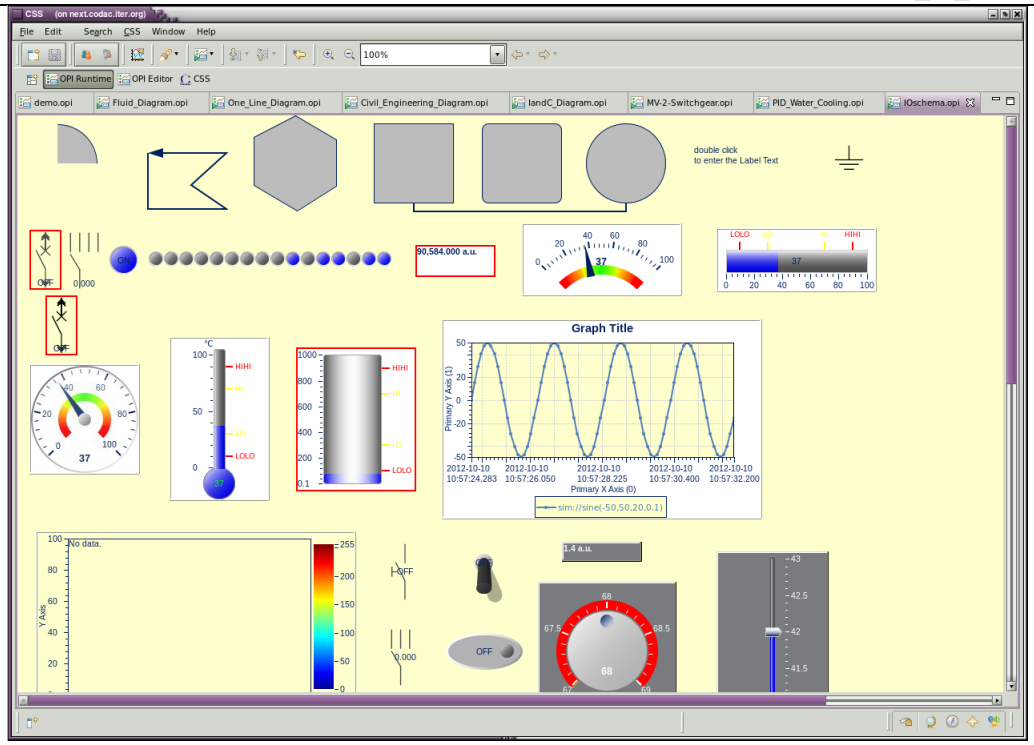
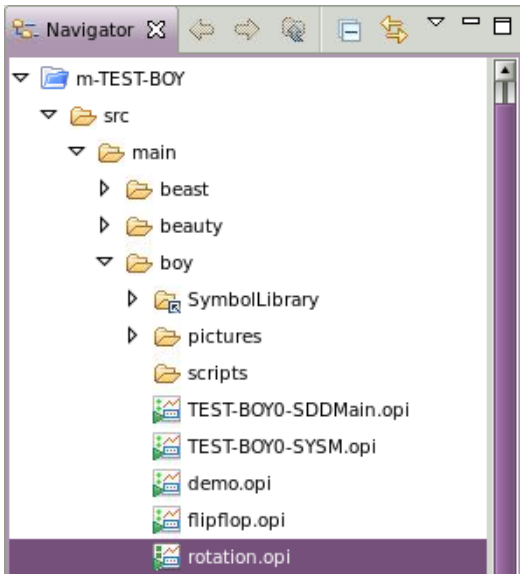
6. The output for the electrical use case should be:

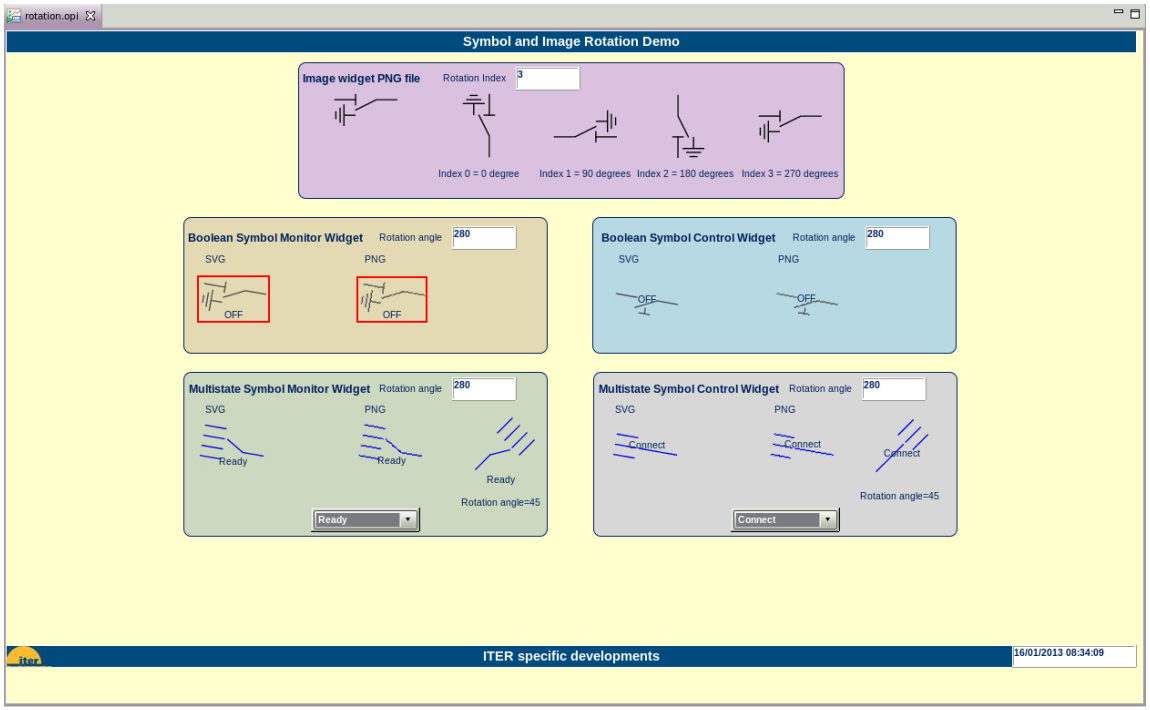


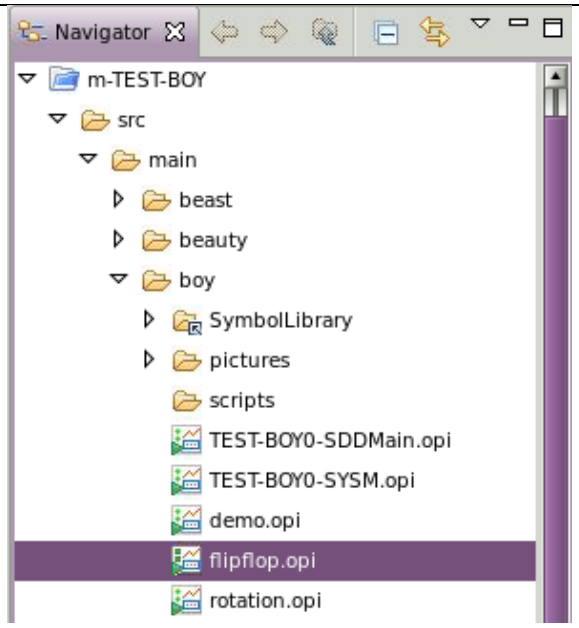
7. The output for the PID use case should be:



8. The output for CODAC BOY Schema should be:

		
SMB-02	**3.4.7 Symbol and Image widgets rotation**	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the rotation of the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the rotation of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy 	

	2. From the Navigator View, double-click on rotation.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Check that the Image widget PNG file rotates from 0 to 270 degrees by 90 degrees. A screen shot of expected positions is given to help the verification. Then check that the Boolean Symbol widgets - Monitor and Control – in SVG and PNG format, rotate by 20 degrees. Finally do the same verification for the Multistate Symbol widgets. 4. Switch back to OPI Editor Perspective and close the rotation BOY OPI	
Pass Criteria	2. The rotation demo BOY screen looks like that: 	
SMB-03	**3.4.8 Symbol and Image widgets flip/flop**	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the horizontal and vertical flip of the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the horizontal and vertical flip of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy	



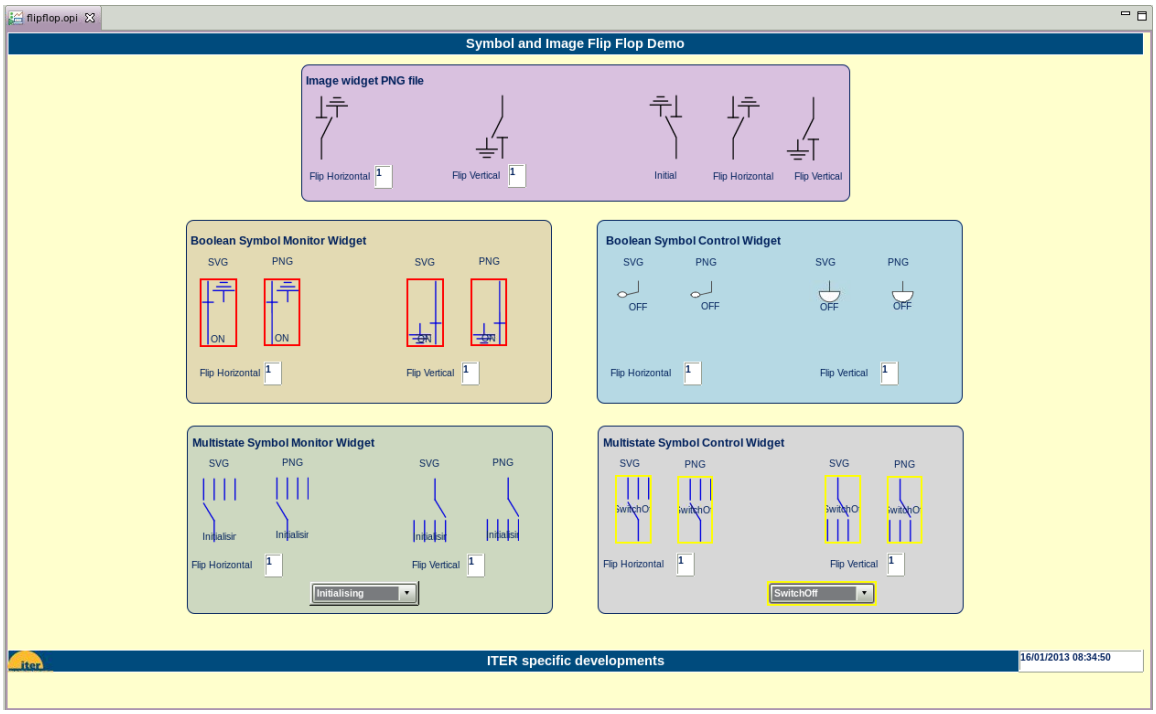
2. From the Navigator View, double-click on flipflop.opi file. In order to have a better view, switch to OPI Runtime Perspective

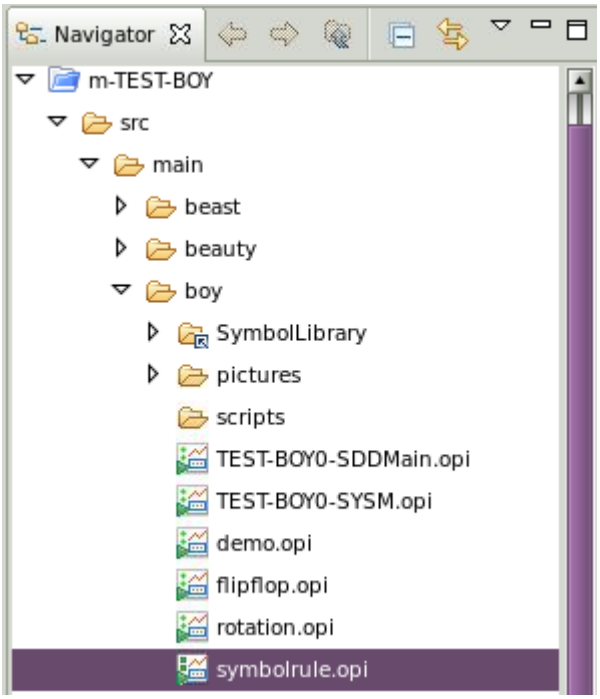
3. Check that the Image widget PNG file flips horizontally and vertically. A screen shot of expected positions is given to help the verification. Then do the same verification for Boolean Symbol widgets - Monitor and Control – in SVG and PNG format and for the Multistate Symbol widgets

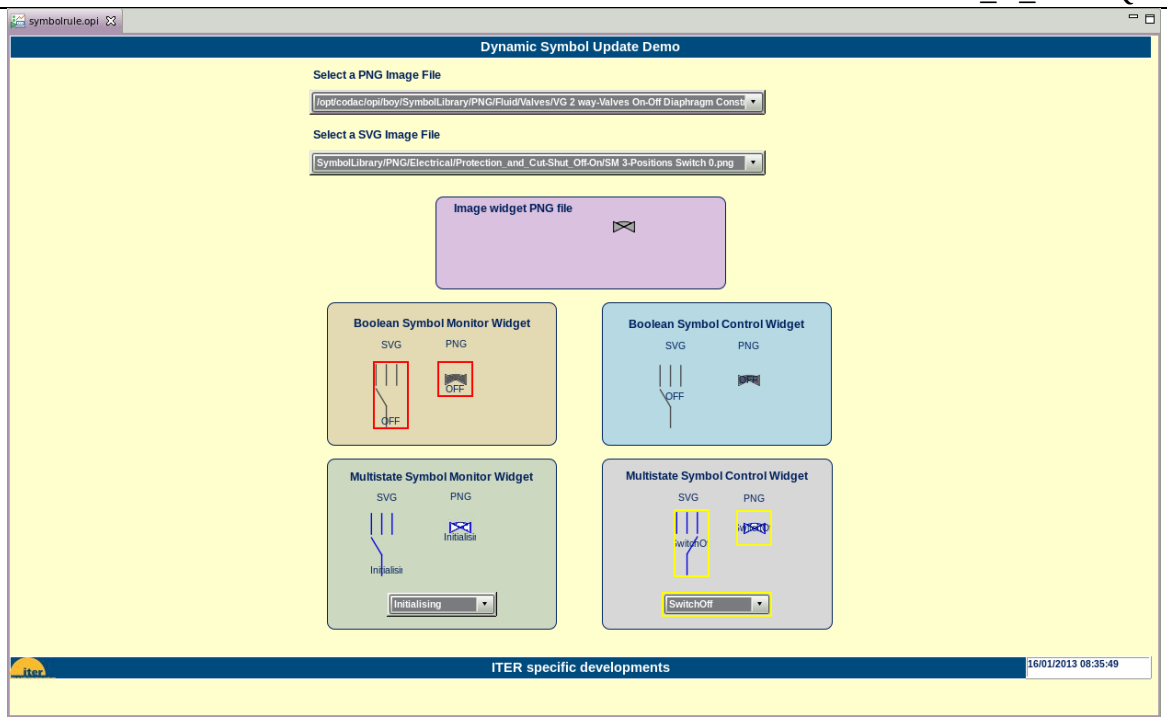
4. Switch back to OPI Editor Perspective and close the OPI screen

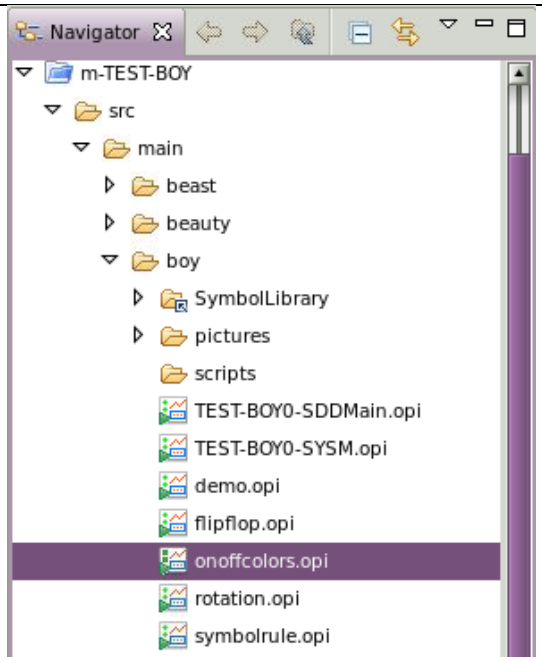
Pass Criteria

2. The rotation demo BOY screen looks like that:



SMB-04	**3.4.9 Symbol and Image File Selection**	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the dynamic change of the image file for the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the dynamic change of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy  2. From the Navigator View, double-click on symbolrule.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Change the PNG Image file using the Combo Box and check that all widgets displaying a PNG file are updated. Do the same with a SVG Image file 4. Switch back to OPI Editor Perspective and close the OPI screen	
Pass Criteria	2. The selection of an PNG and SVG Image file in demo BOY screen looks like that:	

		
SMB-05	**3.4.10 On/Off Symbol Color**	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the dynamic change of the On and Off color for the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the dynamic change of the ON/OFF color of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy	



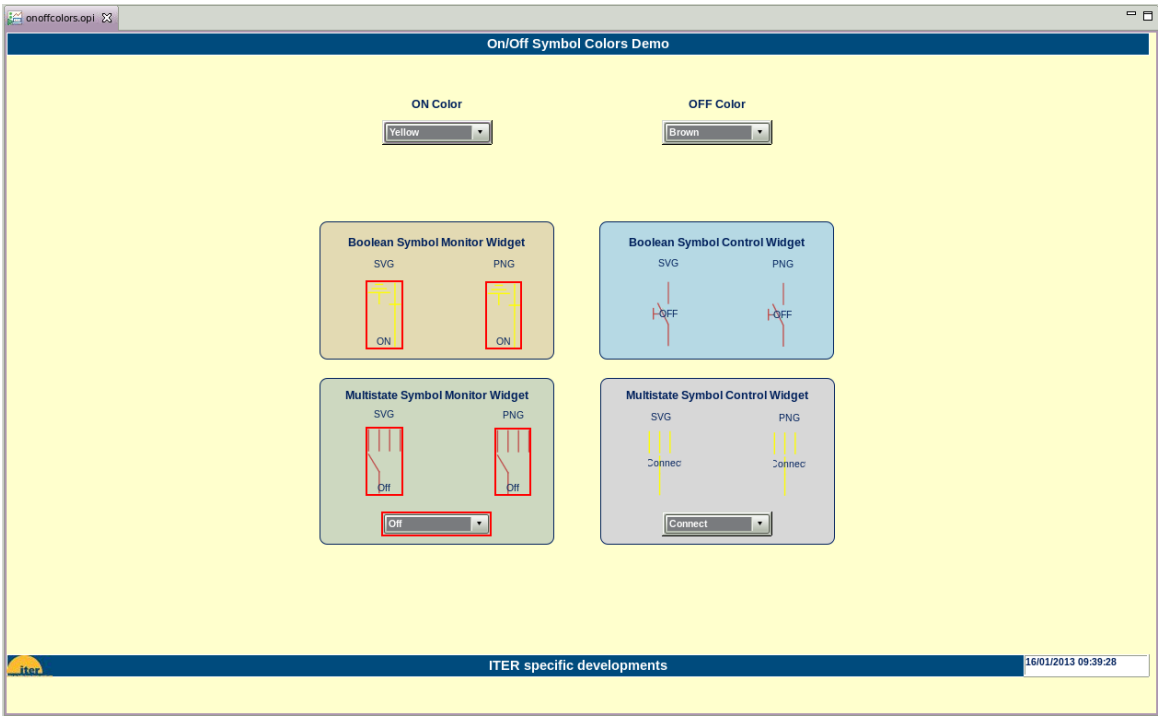
2. From the Navigator View, double-click on onoffcolors.opi file. In order to have a better view, switch to OPI Runtime Perspective

3. Change the ON Color using the Combo Box and check that all symbol widgets displaying an ON value (or index > 0 for a multistate) are updated. Do the same with the Off Color

4. Switch back to OPI Editor Perspective and close the OPI screen

Pass Criteria

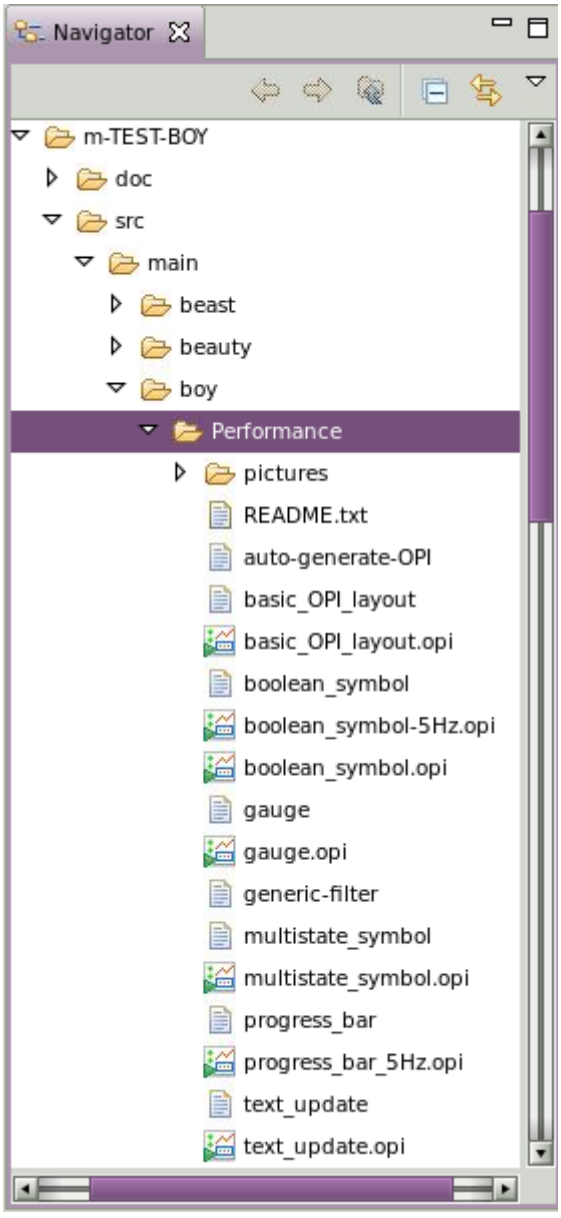
2. The selection of and On and Off color in demo BOY screen looks like that:



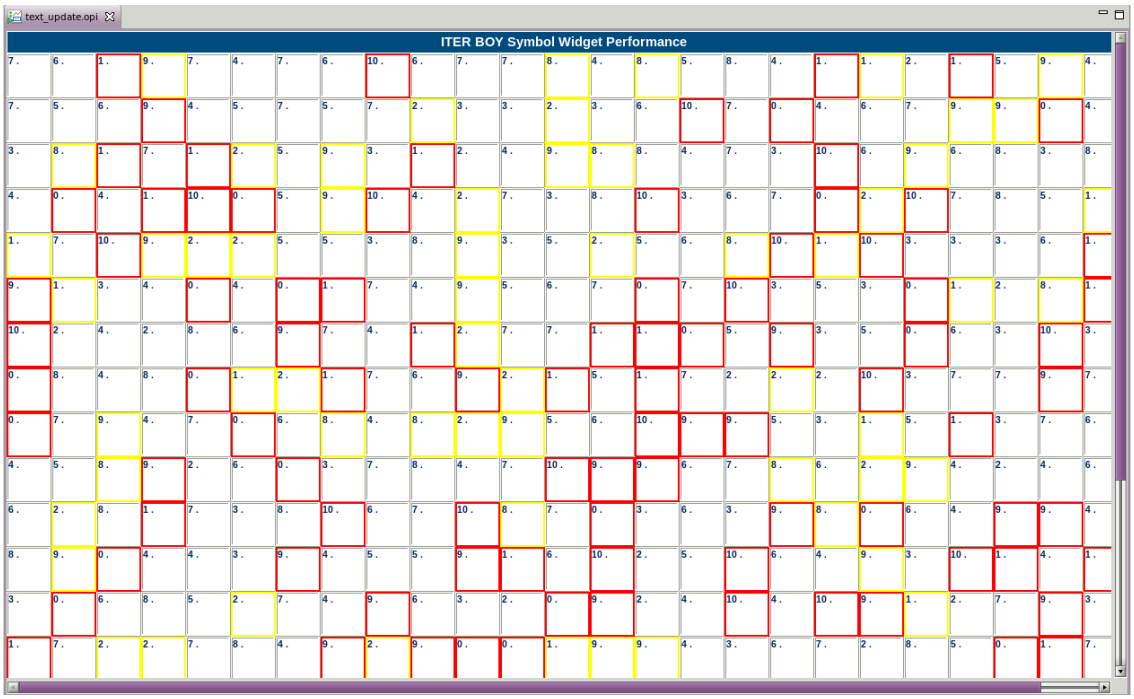
PRF-01

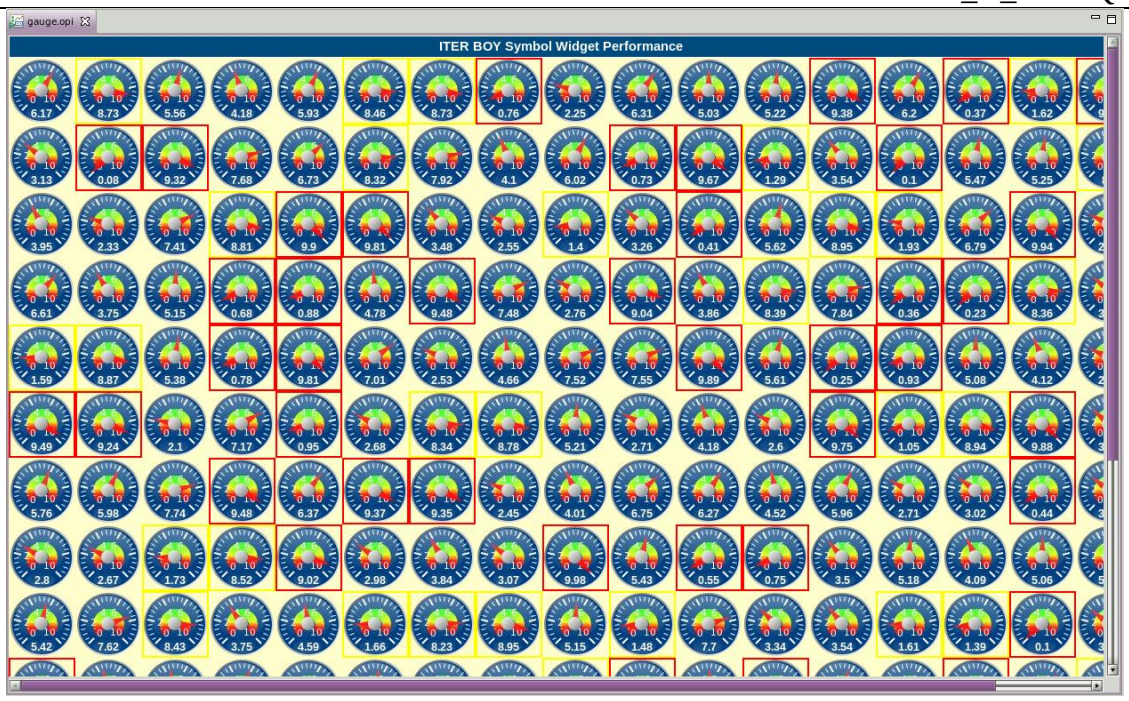
3.4.11 Performance of Symbol Widgets

Prerequisite	1. Performance IOCs downloaded from SVN
	2. You can close the previous EPICS database
	`epics> exit`
	3. Start the EPICS IOC Performance Database:
	`$ softIoc src/main/epics/SharedTemplateApp/Db/all-A-IOCs-start.cmd`	

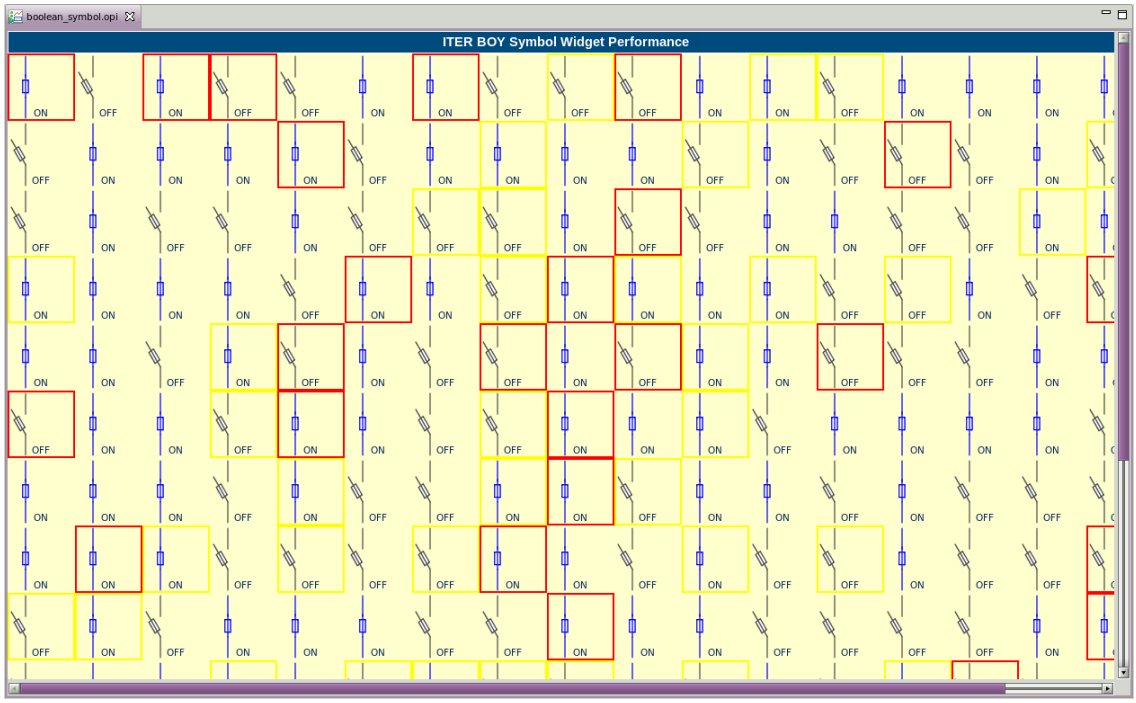
Test Cases	1. Positive confirmation of the loading of more than 250 Boolean and Multistate Symbol widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy/Performance  2. From the Navigator View, right-click first on text_update.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 500 Text Update widgets. Then run the OPI using the button from the tool bar .	

	Close CSS runtime instance and go back to OPI Editor to close text_update.opi
	
	3. From the Navigator View, right-click first on gauge.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 graphical widgets. Then run the OPI using the button from the tool bar  .
	
	Close CSS runtime instance and go back to OPI Editor to close gauge.opi. If you fail to close the Operator Interface which is too busy, stop the IOC:
	
	`epics> exit`
	
	4. From the Navigator View, right-click first on boolean_symbol.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 Boolean Symbol Image widgets. Then run the OPI using the button from the tool bar  . If you have stopped the IOC in the previous step, you need to restart it:
	
	`$ softIoc src/main/epics/SharedTemplateApp/Db/rndmIOC-A-start.cmd`
	
	Close CSS runtime instance and go back to OPI Editor to close boolean_symbol.opi. If you fail to close the Operator Interface which is too busy, stop the IOC:
	
	`epics> exit`
	
	5. From the Navigator View, right-click first on multistate_symbol.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 graphical widgets. Then run the OPI using the button from the tool bar  . If you have stopped the IOC in the previous step, you need to restart it:
	
	`$ softIoc src/main/epics/SharedTemplateApp/Db/rndmIOC-A-start.cmd`
	
	Close CSS runtime instance and go back to OPI Editor to close multistate_symbol.opi.	

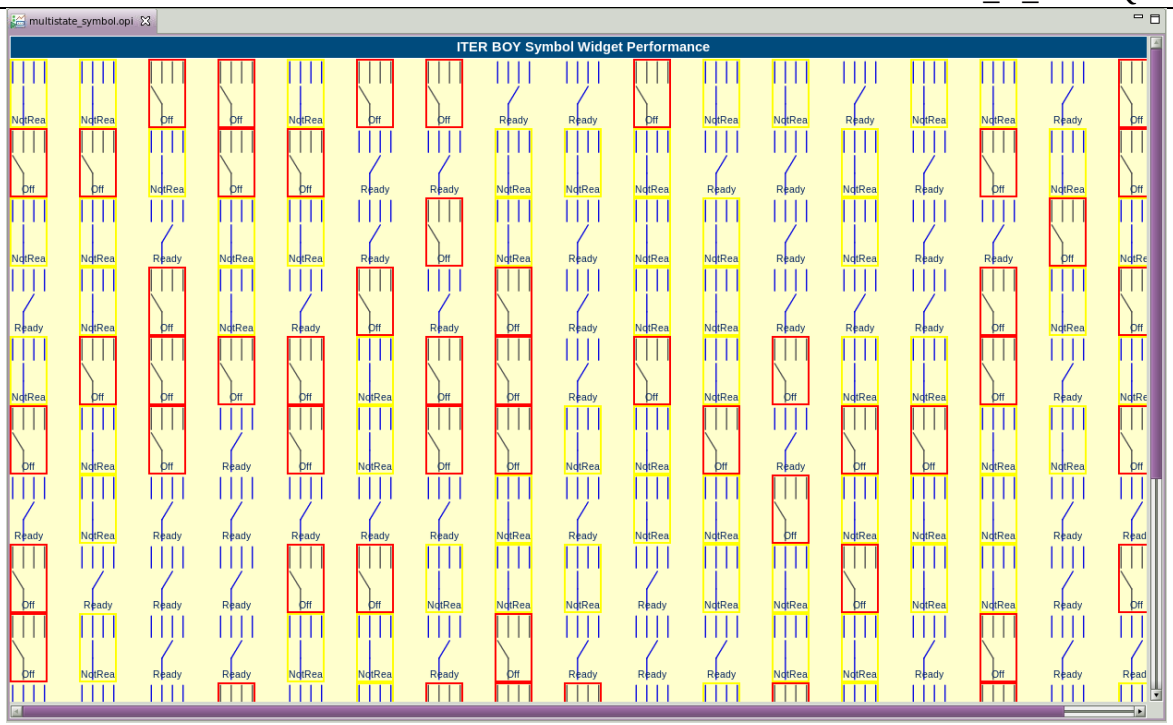
Pass Criteria	2. Text Update widgets used to displayed PV values at 10Hz:  3. The time to load, connect to EPICS PV and animate the graphical widget is definitively longer:	

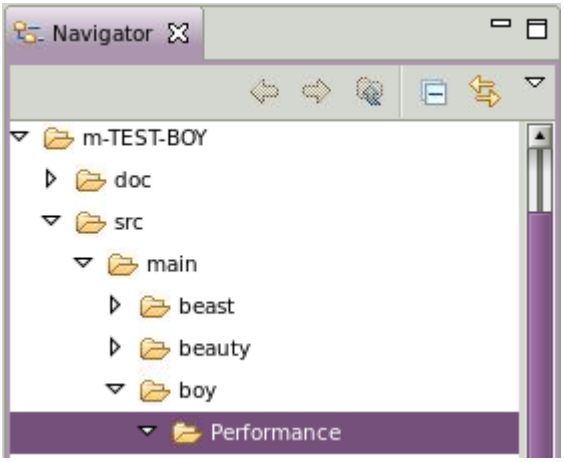


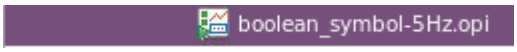
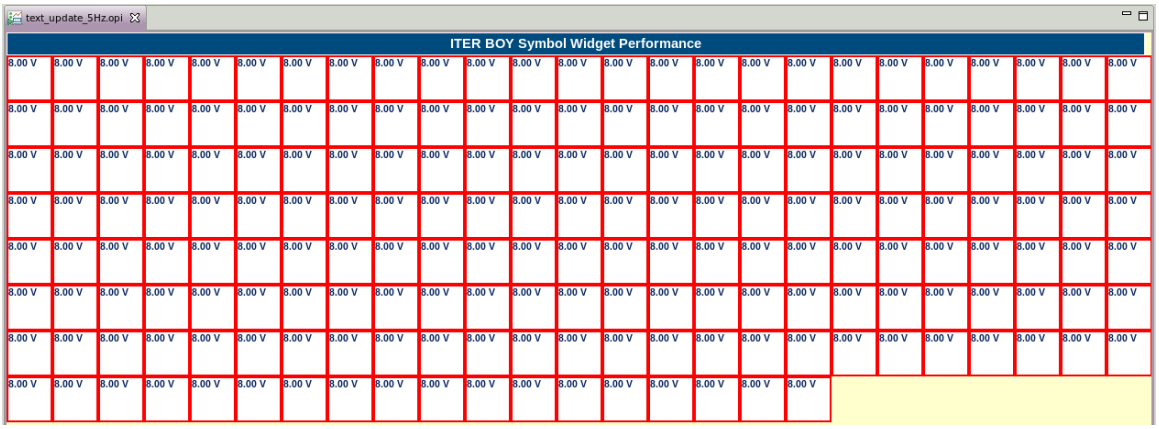
3. The label should be displayed first then the symbol image after some seconds:



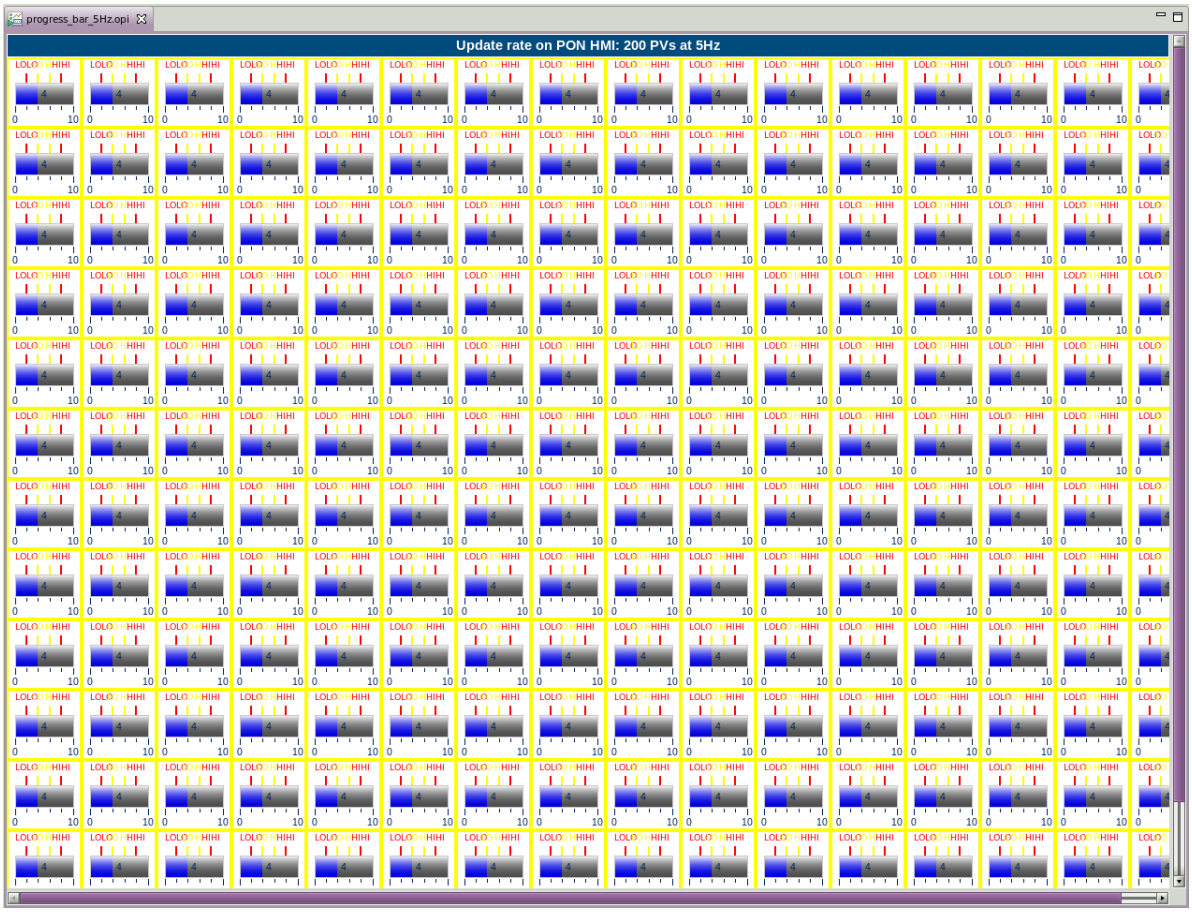
4. Finally the Multistate symbol widgets should be animated at 10Hz from Off state, NotReady state, Initialising state and Ready state:



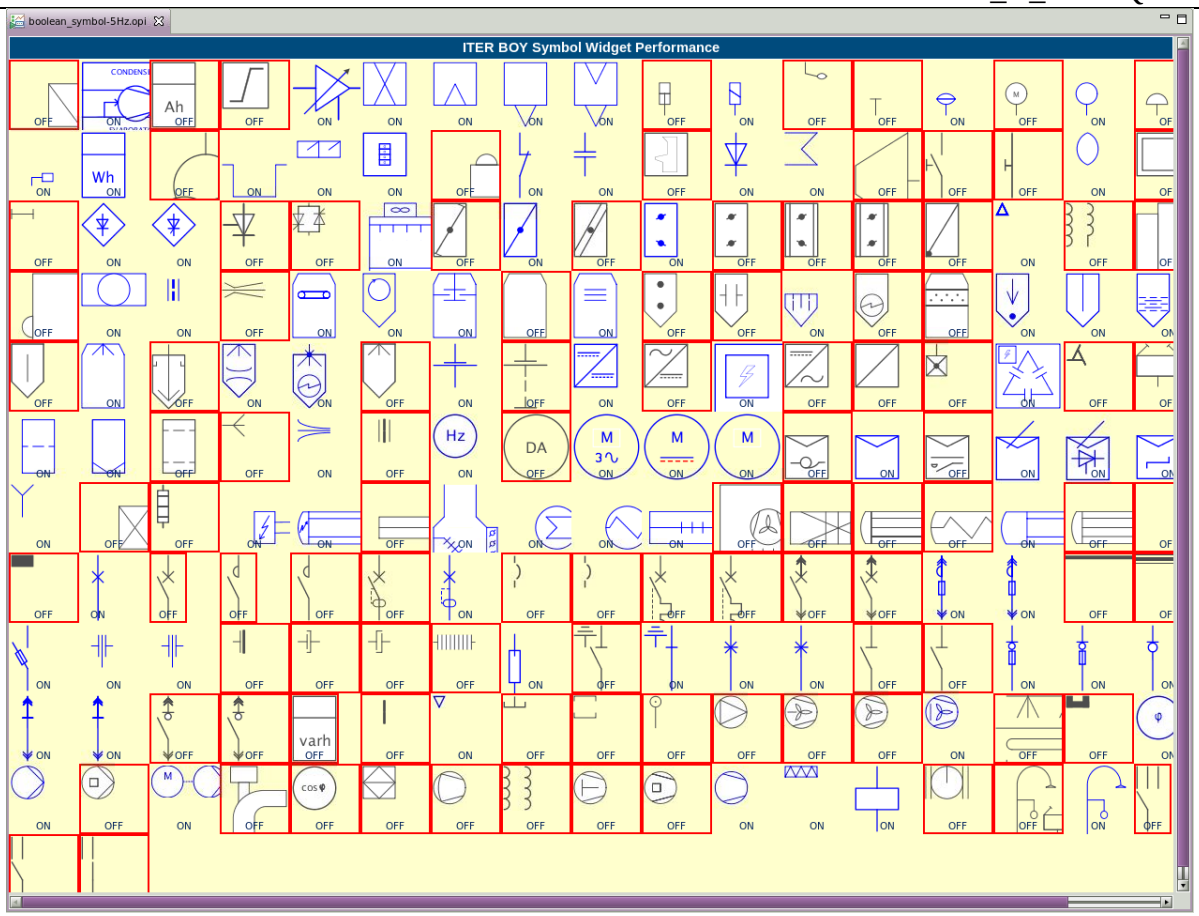
PRF-02	**3.4.12 Update rate on PON HMI: 200 PVs at 5Hz**
Prerequisite	1. Performance IOCs running If you need to restart them: `$ softIoc src/main/epics/SharedTemplateApp/Db/all-A-IOCs-start.cmd`
Test Cases	1. Positive confirmation of ITER control system main parameters - 4.6 Update rate on HMI screen (PON) = 200 PV at 5 Hz (ITER_D_6M58M9 v1.3)
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy/Performance  2. From the Navigator View, right-click first on text_update_5Hz.opi file: 

	And select Open With -> OPI Editor. State how fast the Editor is able to load more 200 Text Update widgets. Then run the OPI using the button from the tool bar  Close CSS runtime instance and go back to OPI Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 3. From the Navigator View, right-click first on progress_bar_5Hz.opi file:  And select Open With -> OPI Editor. State how fast the Editor is able to load more 200 Progress Bar widgets. Then run the OPI using the button from the tool bar  . Make a right-click on the Operator screen to switch to Full Screen in order to display and animate a maximum of widgets. Exit the Full Screen mode by pressing F11. Close CSS runtime instance and go back to OPI Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 4. From the Navigator View, right-click first on Boolean_symbol_5Hz.opi file:  And select Open With -> OPI Editor. State how fast the Editor is able to load more 200 Monitor Boolean widgets. Then run the OPI using the button from the tool bar  . Make a right-click on the Operator screen to switch to Full Screen in order to display and animate a maximum of widgets. Exit the Full Screen mode by pressing F11. Close CSS runtime instance and go back to OPI Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 5. Stop the IOC (if not already done): `epics> exit`	
Pass Criteria	2. OPI for 200 PVs at 5Hz using exclusively Text Update widgets. The 200 ramps from 0 to 10 are displayed smoothly without missing any increment:  3. OPI for 200 PVs at 5Hz using exclusively Progress bar widgets – one of the most	

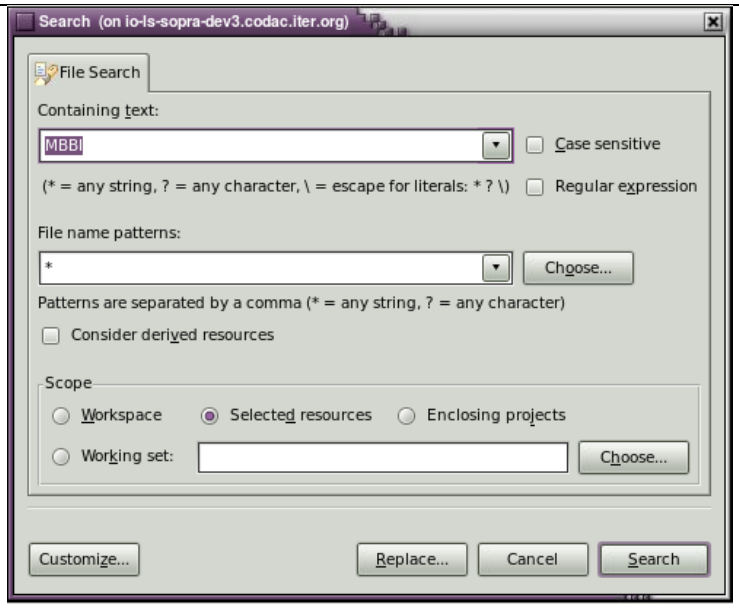
demanding widgets. The 200 ramps from 0 to 10 are not displayed smoothly:



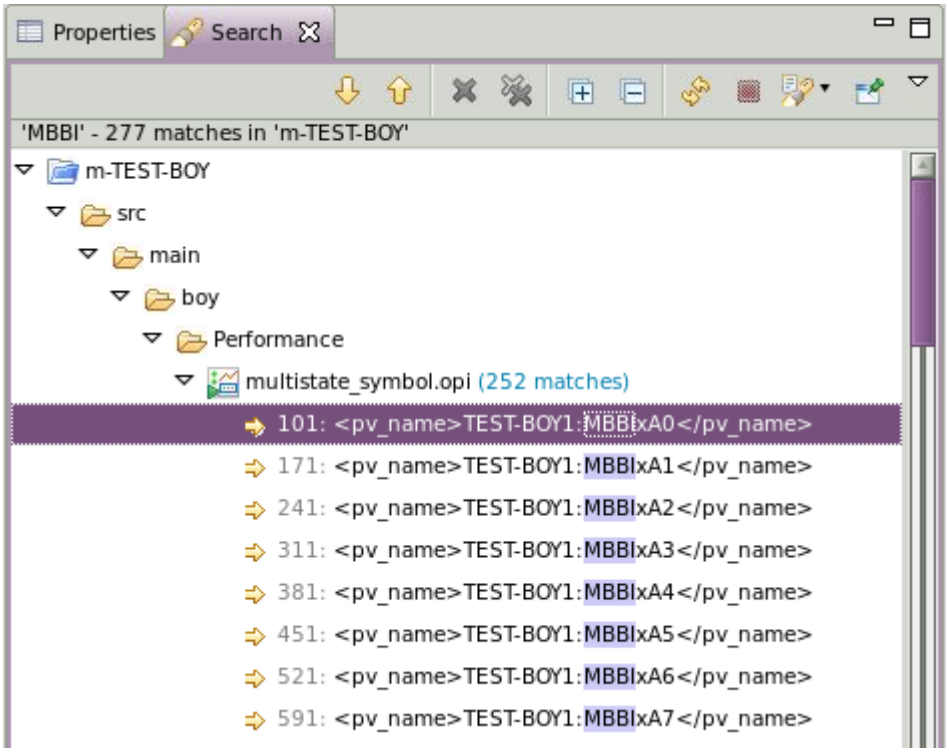
4. Some state changes are being missed on the OPI of 200 different symbols – particularly visible on ON/OFF position switches:



EDT-05	**3.4.13 Eclipse Search in OPI file**	
Prerequisite	1. Development environment started	
Test Cases	1. Positive confirmation of the search feature in OPI files	
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, click on m-TEST-BOY 2. From the menu bar, Search -> Search… (Ctrl+H) 3. Specify “MBBI” ad searched text and click on Selected resources in order to restrict the search to m-TEST-BOY selected previously. Click on Search	

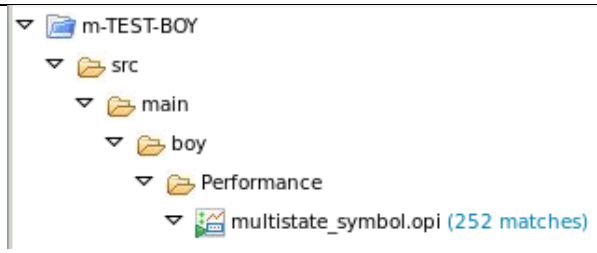


4. From the Search View, double-click on one of the match and the corresponding OPI will be opened and the widget selected

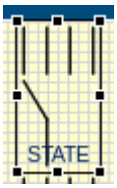


5. Close CSS

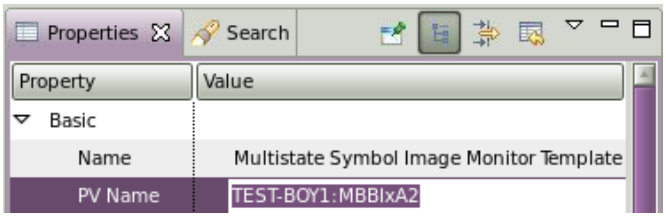
Pass Criteria	4. If the following match is selected for example:  241: <pv_name>TEST-BOY1:MBBIxA2</pv_name> The OPI is open corresponding opi is opened:	



And the widget displaying TEST-BOY1:MBBIxA2 is selected:



To check that the PV name correspond to the match, go to the Property view:



To terminate the tests, stop the slow IOC and close css: 1. $ epics> exit 2. Close CSS using the menu File -> Exit

3.5 Component Test Log

EDT-01	**3.5.1 Operator Interface Edition**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
RNT-01	**3.5.2 Operator Interface Run**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
EDT-02	**3.5.3 Data Browser widget**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
EDT-03	**3.5.4 Vertical and Enabled Tabbed Container**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
EDT-04	**3.5.5 Setpoint adjustment**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

SMB-01	**3.5.6 CODAC Standard Symbol Lib**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
SMB-02	**3.5.7 Symbol and Image widgets rotation**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
SMB-03	**3.5.8 Symbol and Image widgets flip/flop**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
SMB-04	**3.5.9 Symbol and Image File Selection**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
SMB-05	**3.5.10 On/Off Symbol Color**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
PRF-01	**3.5.11 Performance of Symbol Widgets**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
PRF-02	**3.5.12 Update rate on PON HMI: 200 PVs at 5Hz**	[PASS / FAIL]

[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
EDT-05	**3.5.13 Eclipse Search in OPI file**	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

Software Test Plan Checklist

For Assessment of:	
Agency Name	
Project Name	
Document Name	
Date	

Criteria	Yes / No / NA
DOCUMENT STANDARDS COMPLIANCE	
1 Have standards/guidelines been identified to define the work product?	
2 Does the work product format conform to the specified standard/guideline (Template)?	
3 Has the project submitted any request for deviations or waivers to the defined work product?	
4 Have the following areas been addressed completely:	
4a Approval authority?	
4b Revision approval?	
4c Revision control?	
TECHNICAL REFERENCE	
5 Is there evidence that the work product was reviewed by all stakeholders?	
6 Have acceptance criteria been established for the work product?	
7 Does the work product have a clearly defined purpose and scope?	
8 Are references to policies, directives, procedures, standards, and terminology provided?	
9 Does the work product identify any and all constraints/limitations?	
S/W TEST PLAN CONTENTS	
10 Does the S/W Test Plan address the following required information:	
10a Test levels?	
10b Test types (e.g., unit testing, software integration testing, systems integration testing, end-to-end testing, acceptance testing, regression testing)?	
10c Test classes?	
10d General test conditions?	
10e Test progression?	
10f Data recording, reduction, and analysis?	
10g Test coverage (breadth and depth) or other methods for ensuring sufficiency of testing?	
10h Planned tests, including items and their identifiers?	
10i Test schedules, Requirements traceability (or verification matrix)?	

Criteria	Yes / No / NA
10j Qualification testing environment, site, personnel, and participating organizations?	
11 Does the S/W Test Plan identify the environmental exposure as well as requirements for comprehensive, functional, aliveness, end-to-end, and mission simulation testing?	
12 Does the S/W Test Plan provide a System Overview that describes the unique complexities of the system?	
13 Does the S/W Test Plan address user guide, operations / maintenance validation?	
16 Does the S/W Test Plan identify any elements that will not be tested according to the test plan (e.g., externally developed software)?	
17 Does the S/W Test Plan address software architecture in terms of which software components will be based on heritage and which will be mostly or entirely new developments?	
18 Does the S/W Test Plan identify any software reuse? If so, is the extent of reuse or the anticipated modification described?	
S/W TEST ENVIRONMENT	
19 Does the S/W Test Plan include a figure of each system test environment? If so, does it reflect the system hardware approach, simulators, and special development?	
20 Does the S/W Test Plan identify specific test hardware and simulators for each external interface?	
TEST TOOLS	
21 Does the S/W Test Plan address test execution tools?	
TEST PROBLEM REPORTING & CORRECTIVE ACTION	
22 Does the S/W Test Plan provide a description of the problem reporting system to be used by the test team to report problems and/or recommended changes cited during the test activities?	
TEST PROGRESS PLANNING & TRACKING	
23 Does the S/W Test Plan describe the routine test progress reporting approach?	
24 Does the S/W Test Plan describe the Build Test verification methodology? If so, does the description address build verification test level objectives, environment, roles & responsibilities, entry/exit criteria, general guidelines, build test planning, build test scenario development, build test procedure preparation & dry run, build test execution, reporting, and archiving?	