EVGui

1. #Known Issues
    a. #Bugs
    b. #To-Do
2. #Documents
3. #Setup
    a. #Environment
    b. #Dependencies
    c. #Test inside Eclipse
4. #Development
    a. #Overview
    b. #Glossary
    c. #Classes and Functions
        i. #User Interface
        ii. #Controllers
        iii. #Model
        iv. #ControlSystem
        v. #Others
5. #Release with Ant
6. #PDUDiag

Known Issues

Bugs

- The following PVs don't accept Strings (have wrong types?)
    - RATE_DEF:GLB0:%d:INPUTS
    - RATE_DEF:GLB0:%d:LONG_NAME

To-Do

- Do a Klystron check (see p. 3, point 11 of EVG upgrade requirements.pdf)
- Provide support for the editable flag on the ModifiersPanel
- Save descriptions of HW Input Bits
- Implement the proper release procedure
- If possible, make the initial saving of the EvgConfig faster on development

Documents

- PABIG Interface
- EVG Upgrade Requirements
- UI Mockup

Setup

Environment

- Effectively, you can only develop EVGui inside Eclipse
- Check out the CVS module physics/evgui into your workspace
    - From now on, the root of project is referred to as $EVGUI_ROOT
- If you run Eclipse in the development environment, check out also the following CVS modules:
    - physics/hlaCommon
    - physics/hlaExtension
    - physics/xal4lcls

Dependencies

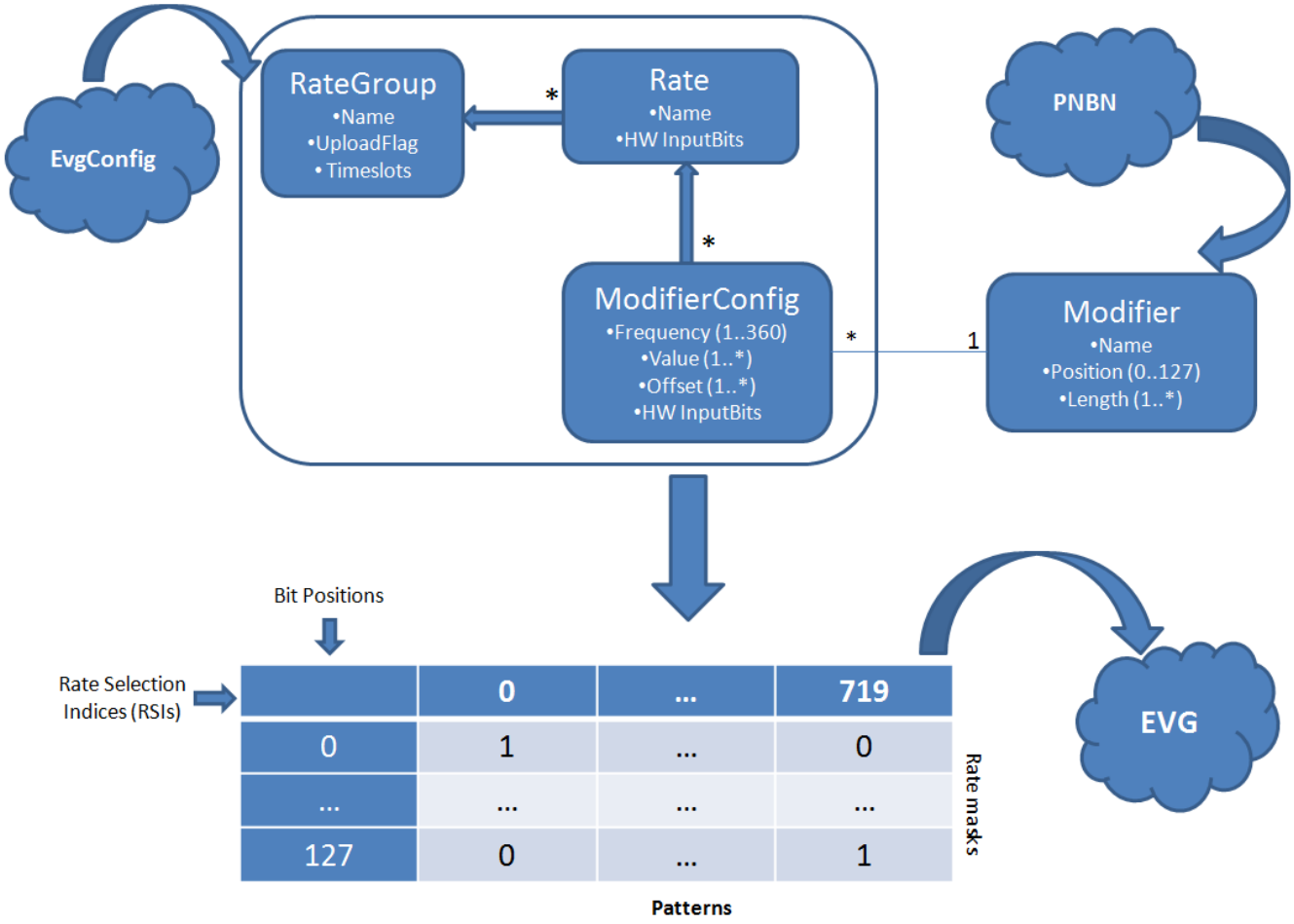
- hlaCommon (latest version)
- hlaExtensions (latest version)
- xal4lcls (latest version)

Test inside Eclipse

- Select $EVGUI_ROOT/src/edu.stanford.slac.evgui/EvguiLauncher
- Right-click => select Run As... => Java Application

Development

Overview



Glossary

- Modifier: a chunk of Rate Bits (has position, length, and value)
  - the least significant bit is at position (furthest on the right)
- ModifierConfig: encapsulates HW Input Bits, frequency, offset, and position into one structure
- Mods: 4 integers representing the 128 rate bits
  - Mod1: bits 31-0, Mod2: 63-32, Mod3: 95-64, Mod4: 127-96
- Pattern: a list of values for Rate Selector Indices (aka RSIs) 1 through 720
  - Each RSI can also be seen as a time unit where 360 time units = 1 sec
- Rate Bits: 128 bits (from 127 to 0), assigned to each RSI
- HW Input Bit: essentially, a binary PV whose value may override modifier bits
  - See pabig_icd-1.7.pdf for details

Classes and Functions

User Interface

- The main user interface consists of 5 tabs: "Define Modifiers", "Define Rate Groups", "Define Input Bits", "Define Beam Rates", and "Define Patterns"
- Each tab has a corresponding XYZPanel class under edu.stanford.slac.evgui.ui
  - Each XYZPanel is a subclass of edu.stanford.slac.mpsgui.util.AbstractPanel
  - (Almost) every widget of the panel is stored in a field in the corresponding XYZPanel class
  - The fields are public to allow registration of the listeners (however, it doesn't make sense to set them)

edu.stanford.slac.mpsgui.ui.BeamRatesPanel

- Allows user to add/remove rates of a particular RateGroup; set a short (used by EVG) and a long rate name, as well as HW Input Bits

edu.stanford.slac.mpsgui.ui.EvguiPanel

- The main panel whose child is a JTabbedPane that contains the 5 tabs

edu.stanford.slac.mpsgui.ui.InputBitNamesSelectionPanel

- Pops up in a dialog when user edits the input bits in JTable cells

edu.stanford.slac.mpsgui.ui.InputBitsPanel

- Allows user to enter names and description for HW Input Bits 0 through 9

edu.stanford.slac.mpsgui.ui.ModifierConfigsPanel

- Allows user to configure modifier bits for a rate (HW Input Bits, frequency, offset, and value)
    - Offset can be 1 or higher (not 0)
    - Display the resulting Pattern in cells to the right
- Added dynamically to the #edu.stanford.slac.mpsgui.ui.PatternsPanel when a rate is selected

edu.stanford.slac.mpsgui.ui.ModifiersPanel

- Allows user to configure Modifiers (name, category, position, length, and whether the value should come from MPG)
    - Eventually, the "editable" flag shall prevent actual changes to Modifiers

edu.stanford.slac.mpsgui.ui.PatternsPanel

- Allows user to select a rate group, specify which time slots to display, select a rate, and create ModifierConfigs

edu.stanford.slac.mpsgui.ui.RateGroupsPanel

- Allows users to add/remove a RateGroup, edit its name, and specify on which time slots it shall be active
- Also features a button for uploading the Rate Groups (and patterns) to EVG

Controllers

- Controllers are located under edu.stanford.slac.evgui.controller; the majority of them are Swing TableModels for JTables on corresponding tabs

edu.stanford.slac.evgui.controller.EvguiController

- Sets up the UI, registers all listeners
- Loads the RatesConfig from the EvgConfig IOC and Modifiers from PNBN IOC
- Delegates to EvgModelIO for actual EPICS gets/puts

edu.stanford.slac.evgui.controller.InputBitNamesCellEditor

- A cell editor for HW Input Bits

Model

edu.stanford.slac.evgui.model.BeamRate, edu.stanford.slac.evgui.model.InputBit, edu.stanford.slac.evgui.model.Modifier, edu.stanford.slac.evgui.model.ModifierConfig, edu.stanford.slac.evgui.model.RateGroup

- Components of the EvgConfig (see also #Overview)

edu.stanford.slac.evgui.model.EvgModel

- Container for all components of the EvgConfig

edu.stanford.slac.evgui.model.EvgModelIO

- Adds a progress layer to gets/puts from EPICS IOCs (see also #ControlSystem)
- Provides necessary information for #edu.stanford.slac.evgui.controller.EvguiController to display a progress dialog

edu.stanford.slac.evgui.model.ModifiersCollection

- A custom collection that implements both getFirst() and get(key)

edu.stanford.slac.evgui.model.PatternsProcessor

- A facade for all bit manipulation methods that are used by EVGui

ControlSystem

- The package edu.stanford.slac.evgui.epics contains 3 classes that implement gets/puts to EPICS IOCs that store the EVG (aka PABIG) data, the EvgConfig, and modifiers (PNBN)

Others

edu.stanford.slac.evgui.util.EvguiUtil

- Contains useful utility methods

Release with Ant

```
cd $EVGUI_ROOT
ant
java -jar /afs/slac.stanford.edu/g/lcls/vol8/epics/TestStand/evgui/evgui.jar &
```

- You can change the destination of the jar file (and other parameters) in $EVGUI_ROOT/build.xml

PDUDiag

- Matlab script for viewing data generated by the PDU; located under $TOOLS/matlab/src/pdudiag.m
  - Configured for production; uncomment line 19 to run on development

    ```
    matlab -nosplash -nodesktop -r pdudiag
    ```

- Select location, crate, and channel; press "Collect Data"