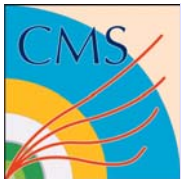


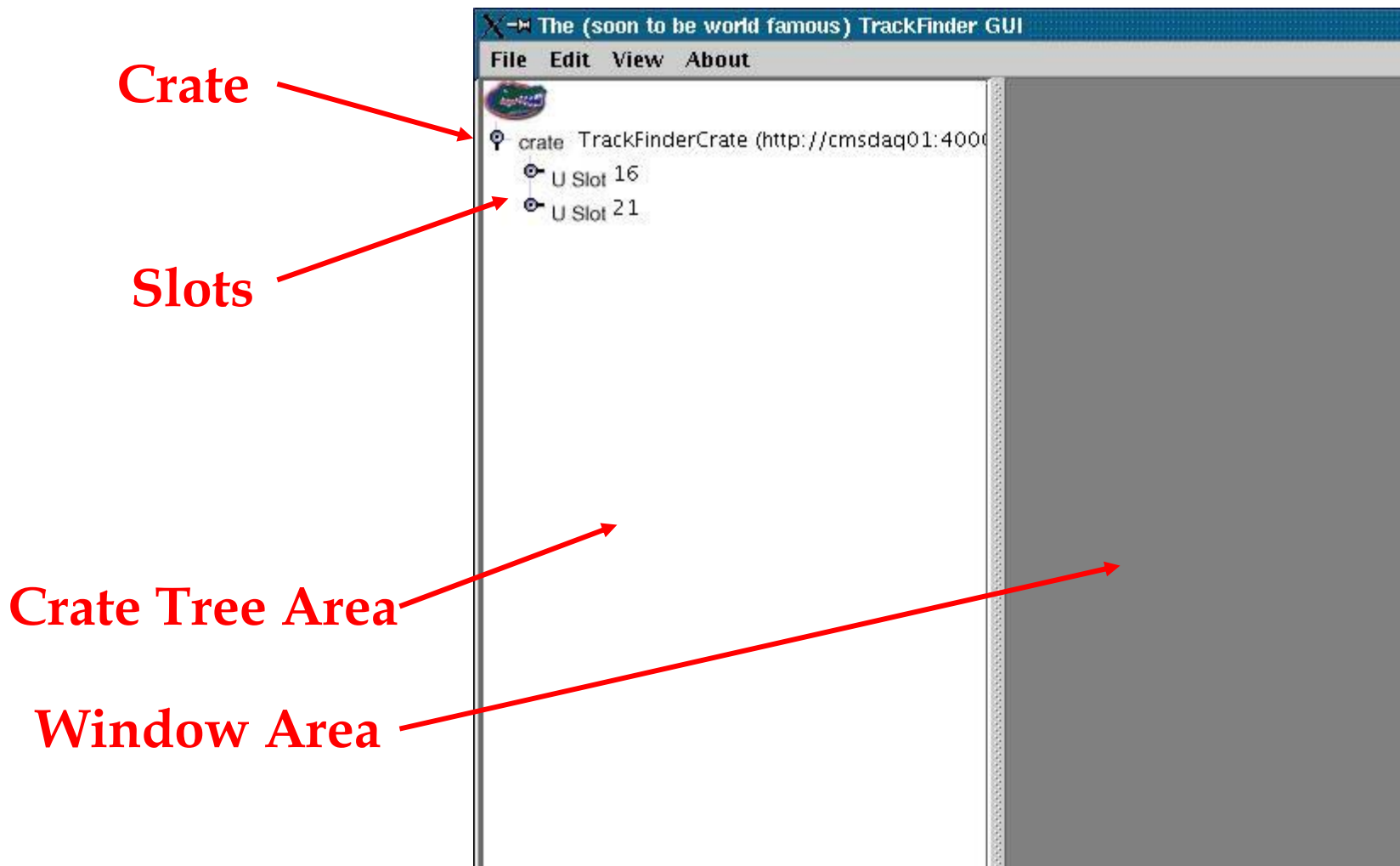


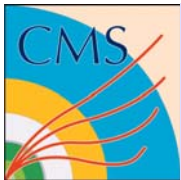
The TrackFinder GUI

Lindsey Gray / Holger Stöck
University of Florida

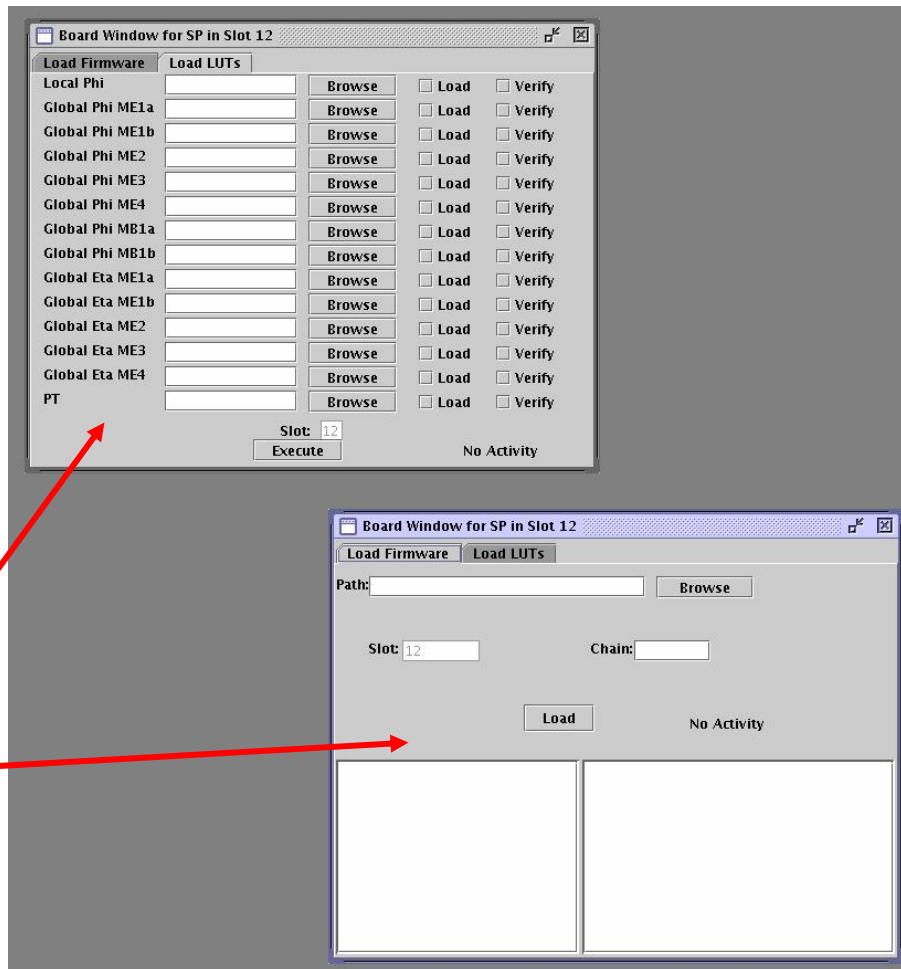
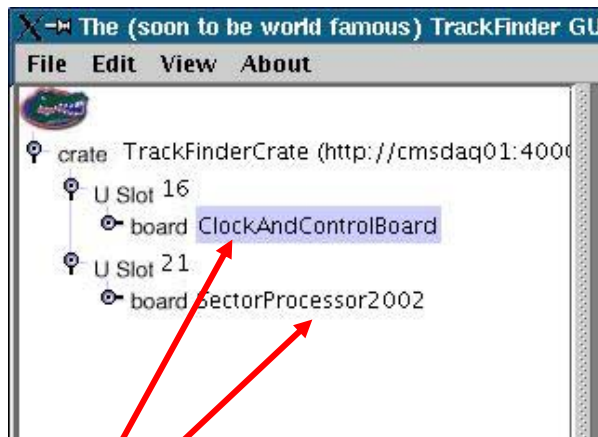


Crate/Slot Level



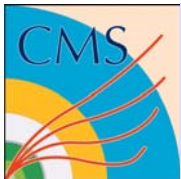


SP02 Board Level



Boards

**Higher level SP02
command panel
windows**

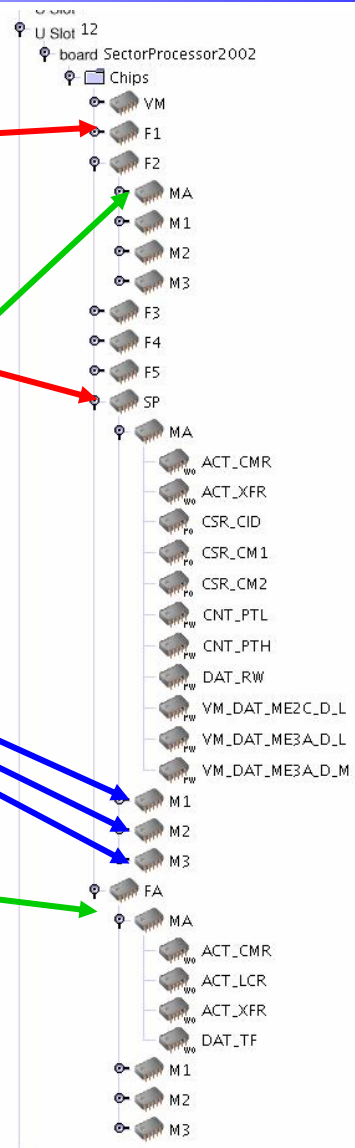


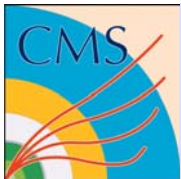
SP02 Chip Level

**SP02
FPGA
Chips**

**FPGA
Muon
Registers**

**FPGA/Muon
Registers for
parallel
addressing**



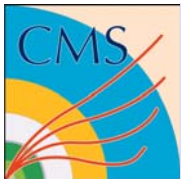


SP02 Register Level

SP02
Registers

The screenshot displays the TrackFinder GUI with a tree view on the left and three editing windows on the right. The tree view shows a hierarchy: crate TrackFinderCrate (http://cmsdaq01:4000) -> U Slot 16 -> board ClockAndControlBoard -> U Slot 21 -> board SectorProcessor2002 -> Chips -> F1 -> Muons -> MA -> Registers -> M1 -> Registers -> M2 -> Registers -> M3 -> Registers. The registers listed are: ACT_HR, ACT_CMR, ACT_LCR, ACT_XFR, ACT_ADR, ACT_TST, STS_CCB, STS_VNA, STS_VPC, CSR_CID, CSR_CLK, CSR_CM1, CSR_CM2, CSR_HR, CSR_CFG, CSR_INI, CSR_CHP, CSR_RDY, CSR_BSY, CSR_OSY, CSR_WOF, CSR_LCT, and CSR_BCO. Three editing windows are open: 'Editing Window for: SP->ACT_HR', 'Editing Window for: SP->CSR_CLK', and 'Editing Window for: SP->DAT_LP'. Each window has fields for Slot Number (21), FPGA (F1), Muon (M3), Register (ACT_HR, CSR_CLK, DAT_LP), and Current Value. They also have a 'New Value' field, 'Read' and 'Write' radio buttons, and an 'Execute' button. Standard Output and Standard Error Output fields are also present.

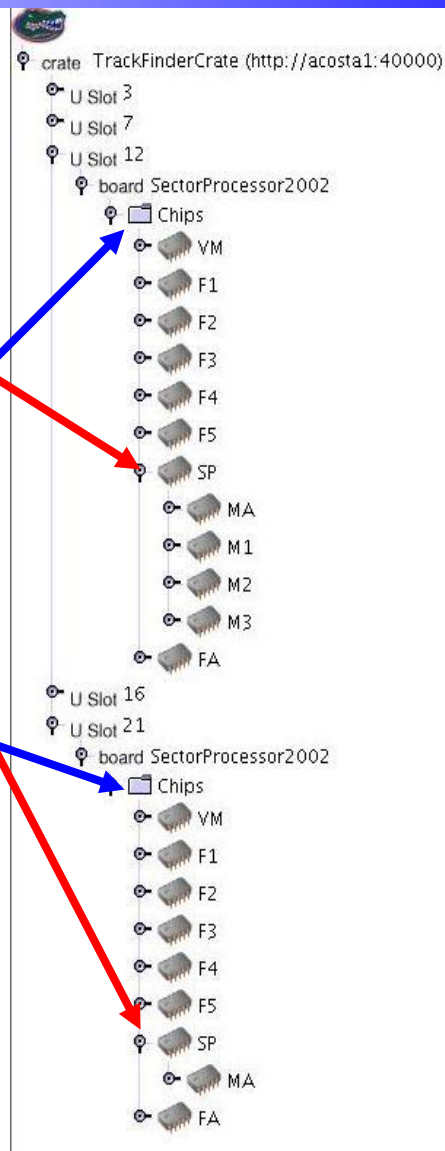
Read/write
hex values
into
individual
registers

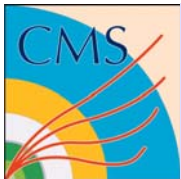


NEW: Sector Processor Registers

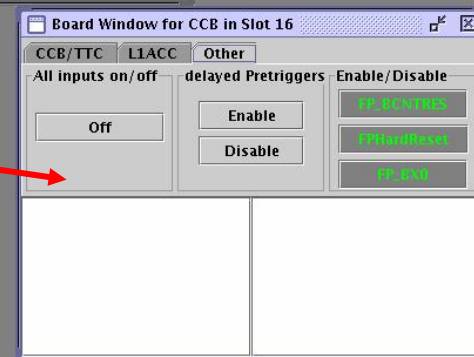
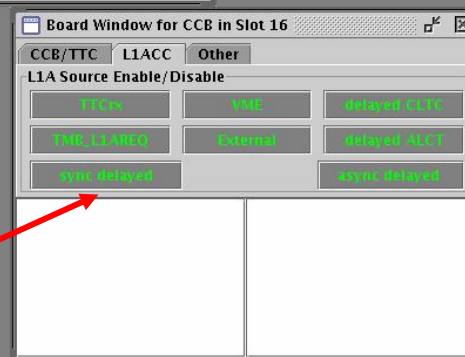
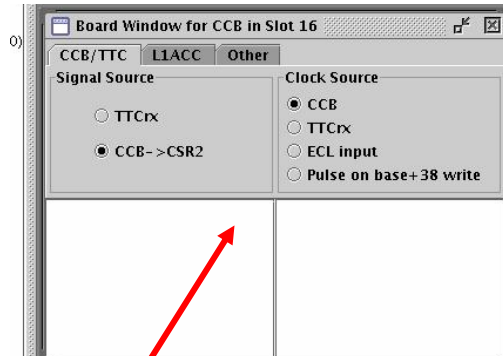
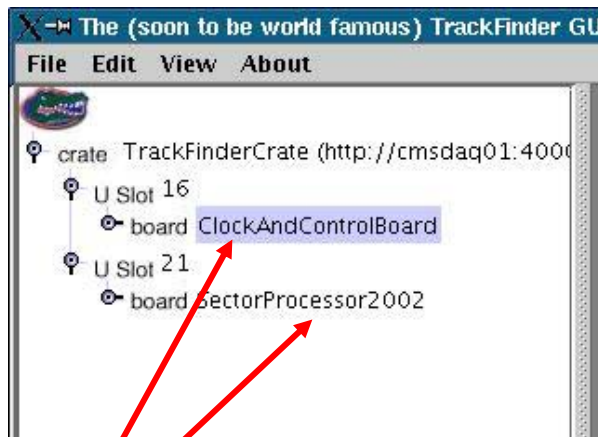
SectorProcessor FPGA

New XML setup routine probes for available registers in all FPGAs



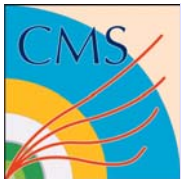


CCB Board Level



Boards

Higher level CCB
command panel
windows

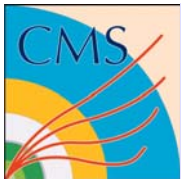


CCB Register Level

CCB
Registers

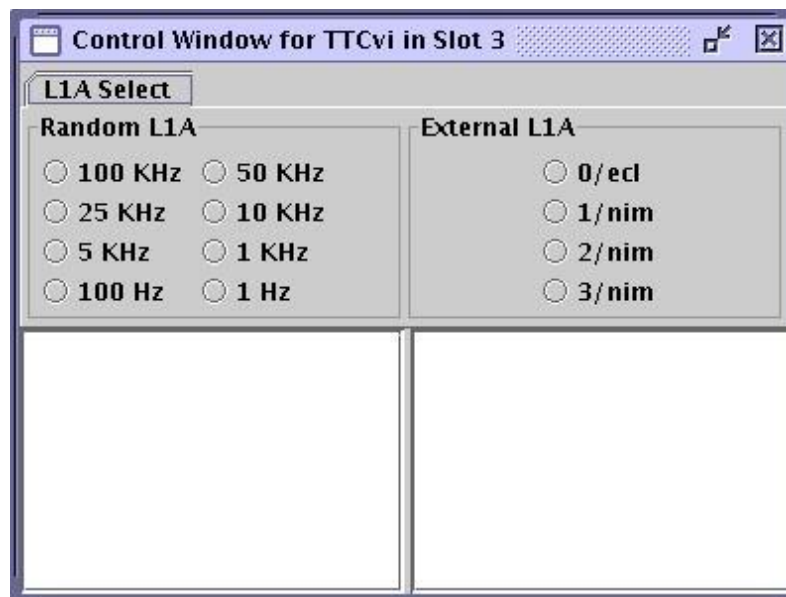
Read/write
hex values
into
individual
registers

The screenshot displays the TrackFinder GUI interface. On the left, a tree view shows the hardware configuration: crate TrackFinderCrate (http://cmsdaq01:4000) -> U Slot 16 -> board ClockAndControlBoard -> Chips -> CSR1 through CSR16, CSR16B, HardTMRReset, HardDMBReset, HardALCTReset, HardMPCReset, HardCrateReset, CSR9Reset, CSR10Reset, CSR11Reset. On the right, four 'Editing Window' panels are shown for CSR1, CSR2, CSR3, and CSR9. Each panel has fields for Slot Number (16), Register (CSR1-3/9), and Current Value (0x0 or 0x3ff). They also include a 'New Value' field, 'Read' and 'Write' radio buttons, and 'Standard Output' and 'Standard Error Output' text areas. An 'Execute' button is present in each panel. Red arrows point from the text on the left to the GUI elements.



NEW: TTCvi Window

Allows for quick configuration of the TTCvi





Backend: System Interface

Backend currently makes calls to console programs of the TrigDAQ package

Controlled through one class for easy maintenance and possible extensions

Class can easily be modified to use SOAP messaging for XDAQ

```
emacs@localhost.localdomain
File Edit Options Buffers Tools Java Help

public static void execute(java.lang.String cmd, javax.swing.JTextArea textarea, javax.swing.JLabel working)
{
    cmdThread myThread = new cmdThread(cmd, textarea, working);
    myThread.setPriority(java.lang.Thread.MIN_PRIORITY);
    myThread.start();
}

class cmdThread extends java.lang.Thread
{
    public cmdThread(java.lang.String cmd, javax.swing.JTextArea txtarea, javax.swing.JLabel working)
    {
        command = cmd;
        textarea = txtarea;
        notify = working;
    }

    public void run()
    {
        Process p = null;
        Runtime r = Runtime.getRuntime();
        java.lang.String retval = "";

        try
        {
            retval = "";

            notify.setText("Loading!");
            notify.setForeground(java.awt.Color.RED);
            p = r.exec(command);

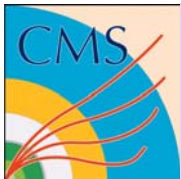
            InputStream in = p.getInputStream();
            int c;
            while ((c = in.read()) != -1)
            {
                retval += ((char)c);
                textarea.append(retval);
                retval = "";
            }
            notify.setText("DONE!");

            sleep(3000);

            notify.setText("No Activity");
            notify.setForeground(java.awt.Color.BLACK);
        }
        catch (Exception e)
        {
            textarea.append("error executing " + command);
        }
    }

    java.lang.String command;
    javax.swing.JTextArea textarea;
    javax.swing.JLabel notify;
}

----- ExecCommand.java (Java CVS-1.2 Abbrev)--L111--63%-----
```

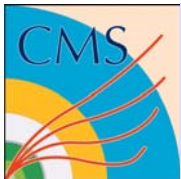


Backend: XML

The TFGUI parses XDAQ address tables to generate available registers for the crate tree

The layout of boards within the crate is given by a XDAQ-style XML file made for the TFGUI

```
<Partition name="TrackFinderCrate">
  <Definitions>
    <ClassDef>SP</ClassDef>
    <ClassDef>CCB</ClassDef>
    <ClassDef>MPC</ClassDef>
    <ClassDef>ITTCvi</ClassDef>
  </Definitions>
  <Host id="0" url="http://acostal:40000">
    <Application class="SP2002">
      <DefaultParameters>
        <Parameter name="hardwareAddressTable" type="string" value="../../TrigData/TrackFinderCrate/SectorProcessor2002/xml/AddressTable_SP2002.xml"/>
        <Parameter name="executable" type="string" value="../../bin/Linux_2.2/readwriteRegister"/>
        <Parameter name="VMEslot" type="int" value="21"/>
      </DefaultParameters>
    </Application>
  </Host>
</Partition>
```



To Come

- **Add MPC interface for backend (for in-Crate tests)**
- **Add more panels for configuration and higher level commands (in progress)**
- **Add panels for crate wide configuration and testing procedures**
- **Replace console program calls with XDAQ SOAP messages**
- **Add B-GO section to the TTCvi control window**
- **...**