SSD slow control (simulation)

The code can be found in **ssd-upgrade.star.bnl.gov:/home/yanwh/epics/ssdY**

OR **garbo.star.bnl.gov:/home/yanwh/epics/x86/ssdY**

In order to test slow control without hardware (w/o power supply, too), there are mainly two difficulties:

1. The old code was running on VME, it should be changed for PC.
2. JTAG simulation without hardware.

# VME->PC

1. The functions provided by vxWorks are all changed or disabled.
2. The makefile and the directory structure are adjusted.

# JTAG support

Reimplement related functions for JTAG protocol:

**scan\_ir(unsigned short \*output, unsigned short length, unsigned short \*input)**

**scan\_dr(unsigned short \*output, unsigned short length, unsigned short \*input)**

**circulate\_dr(unsigned short length, unsigned short \*data)**

init\_stream(unsigned short \*, unsigned short)

get\_stream(unsigned short \*,unsigned short, unsigned short)

set\_stream(unsigned short \*, unsigned short, unsigned long, unsigned short)

**The above 3 functions has nothing to do with the hardware.**

# Changelog

The main changes in ssdApp directory are listed below.

## ssdApp/adl

The <anyName>\_BAK.adl are automatically generated when you change anyName.adl. They can be safely ignored.

### ladder.adl

1. The direction of Byte Monitor changed from “up” to “down” to avoid warnings of MEDM.
2. The location of bias files changed.

### power\_chan.adl & power\_chan\_summary.adl

1. The direction of Byte Monitor changed from “up” to “down” to avoid warnings of MEDM.

### other

**No change:** avdd.adl, avss.adl, bias.adl, calib.adl, guard.adl, ladder\_config.adl, ladder\_expert2.adl, ladder\_summary.adl, module.adl, pedoff.adl, power\_expert\_1\_10.adl, power\_expert\_11\_20.adl, power\_expert\_help.adl, ssd\_main.adl, ssd\_main\_expert.adl, ssd\_messages.adl, ssd\_readout.adl, ssd\_readout\_summary.adl, temperatures\_cnx.adl, vme\_ps\_9u.adl.

**Complete rewritten needed (power supply):** crate\_map.adl, caen\_crate\_control.adl, crate\_map.adl.

**Not used:** crate\_map\_current.adl, crate\_map\_proposal.adl, ladder2.adl, ladder\_small.adl, ssd10lad\_new.adl, ssd10lad\_new2.adl, SSDcanbus.adl, ssd10lad\_new.adl, ssd10lad\_new2.adl, ssd\_ladder\_confog.adl, ssd\_main\_10lad.adl, ssd\_main\_20060119.adl, ssd\_main\_expert2.adl, ssd\_main\_expert\_20060119.adl, temperatures.adl.

## ssdApp/adl

### ladder\*.db

To enable all chips,

1. In A128C record:

field(BDIR, “/home/yanwh/epics/ssdY/iocBoot/iocssd/bias/”)

field(HYBY, “65535”)

field(FBIA, “0)

1. In costar record:

field(byPs, “1”)

### other

**No change:** 10ladder\_readout.db, power\_ladder\*.db

**Complete rewritten:** Makefile

## ssdApp/src

1. The header files from vxWorks are all removed.
2. semBCreate, semTake, semGive functions are disabled.
3. taskSpawn function was replaced by call the process directly.

After Considering all this, jtagA128C.c, jtagConnexion.c, jbiCostar.c, jbiReadout.c are the same.

### caenRecord.c

1. Disable the infinite loop while(1)

### jbimain.c,

1. Remove some vxWorks function.

### ReadoutFPGARecord.c

1. “rdnb<<4” to “rdnb\*10” for initiation of Readout board.

### A128crecord.c, costarRecord.c, & caenSimul.c

1. Some small corrections.

### jtagCommonSimul.c

This is the most important function in simulation. The JTAG related functions are defined in this file.

1. Remove some unused functions.
2. Add some output.
3. Fix some bugs.

### caen.h, jtagCommon.h, jtagCostar.h & jtagreadout.h

1. A few changes in function declaration.

### sequencer files

1. The suffix change to stt.
2. Some unimportant changes in ssd\_ladder.st, ssd\_main.st.

### other

**Added:** A128CSupport.dbd, caenSupport.dbd, ReadoutFPGASupport.dbd, ssdMain.cpp, ssd\_seq.dbd

**Removed:** caen.c, caen\_2b.c, caen\_2b.h, jtagCommon.c

**No change:** A128CRecord.dbd, constarRecord.dbd, jbicomp.c, jbicomp.h, jbiexprt.h, jbijtag.c, jbijtag.h, jbiport.h, jtagA128.h, jtagConnexion.h, ReadoutFPGARecord.dbd, ssd\_10lad.st,

**Complete rewritten:** Makefile

# Known problems

1. There are some problems in x86\_64 architecture, so I rebuilt a 32-bit version EPICS.

**export EPICS\_HOST\_ARCH=linux-x86**

1. You may need to create a soft link in /usr/lib to compile the medm extension.

**cd /usr/lib**

**sudo ln -s libXt.so.6.0.0 libXt.so**

**sudo ln -s libXp.so.6.2.0 libXp.so**

**sudo ln -s libXmu.so.6.2.0 libXmu.so**

1. A 32-bit flex library is not directly available in the 64-bit ssd-upgrade computer. So I download flex-2.5.37.tar.bz2 from <http://sourceforge.net/projects/flex/>.

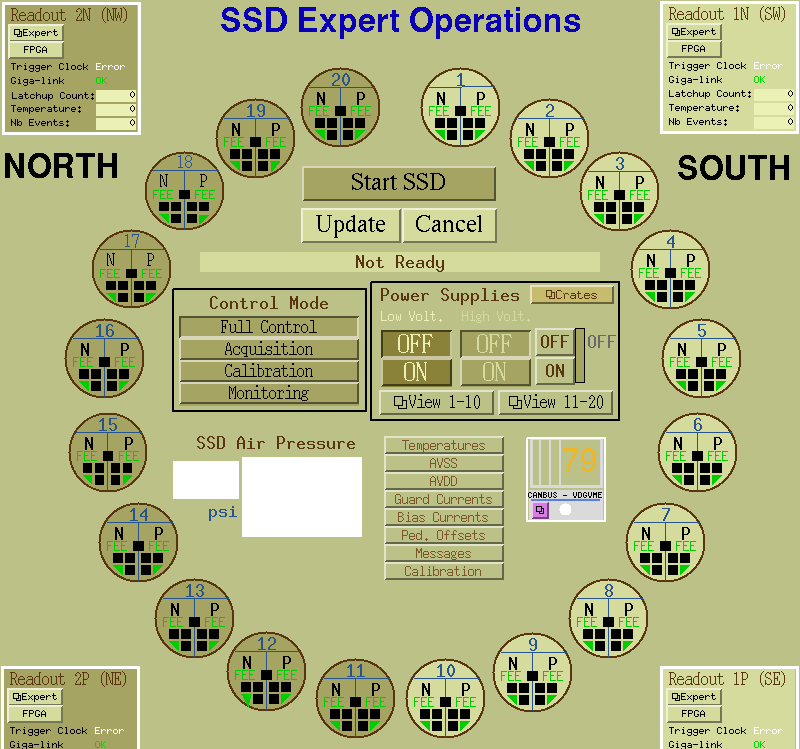
**export CC="gcc -m32" ./configure --prefix=/usr/local/**

**make**

**make install**

1. Don’t trust the return value from scan\_dr..

# How to

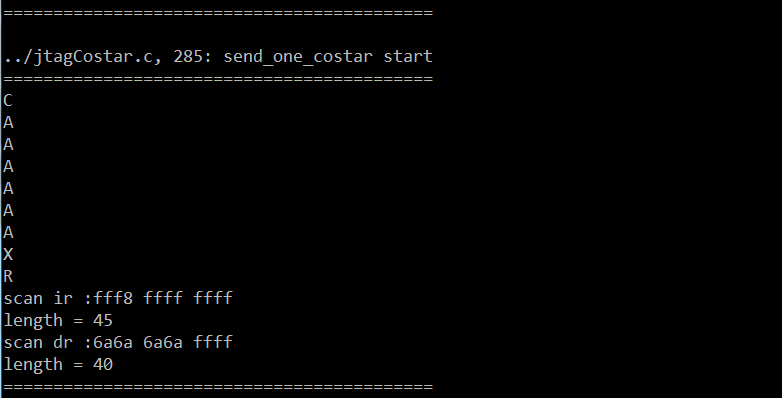


The usage is almost the same as old slow control.

1. Start IOC: cd ssdY/iocBoot/iocssd/; ../../bin/linux-x86/ssd st.cmd
2. Start the GUI: cd ssdY/ssdApp/adl; medm ssd\_main\_expert.adl; click the “Execute” button in medm window.
3. Turn on the 3 buttons in Power Supplies box, then click “Start SSD”.

You can watch the output in IOC terminal.

A Simple example of decoding the simulation result:



The first line shows where scan\_ir, scan\_dr functions are called.

The JTAG chain is R X A A A A A A C (R: Readout boards; A: Alice 128 chip; C: Costar chip)

The instruction is ffff ffff ffff8

The data is ffff 6a6a 6a6a

So in this example, only COSTAR chip is NOT in BYPASS, the instruction is 0x18(last 5 bits). This corresponds to the command CO\_DAC\_0 and it is called by set\_dac\_one\_costar.

# To do

1. Print ladder number and module number in each JTAG command.
2. Translate the magic instruction numbers to their corresponding macros.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Last modified: 12-17-2012

[yanwh@rcf.rhic.bnl.gov](mailto:yanwh@rcf.rhic.bnl.gov)