# EPICS driver for scaler readout from JLab Discriminator (Current state and future plans)

Nerses Gevorgyan
Yerevan Physics Institute
2 Alikhanian Brothers, Yerevan 0036, Armenia
Hovanes Egiyan
Jefferson Laboratory
12000 Jefferson Ave, Newport News, VA 23606, USA

December 16, 2011

**Abstract**

The JLab Fast Electronic group developed a 16-channel discriminator/scaler VME module. It has scalers and programmable thresholds, digital delays and widths of pulses. Hall-D needs a software to remotely monitor and control these discriminator modules. In this paper the EPICS interface for scaler readout will be described and details of preliminary driver presented.

## 1 Introduction

The Fast Electronics group of JLab developed a 16-channel VME discriminator-scaler module (see Ref. [1]). The board contains 16-dual threshold discriminators with programmable digital delays and two 32-bit scalers per discriminator and threshold. One of the scalers is gated from NIM input on the front-panel and the second is free running. The discriminator outputs differential ECL logic signals through two front-panel connectors. Both TDC and trigger output channels can individually be enabled or disabled with

output widths and delays being user programmable. All programming is done through VME registers. The TDC output is driven from discriminator channel and is not routed through the FPGA to minimize jitters and delays. The trigger output is the second threshold per discriminator and is routed through the FPGA.

The goal of this project is to develop EPICS support for this module to fit into slow-controls framework of Hall-D. The preliminary version of EPICS support for the scalers was developed based on the driver from Data Acquisition group of JLab. A user interface was created to enable graphical monitoring of scalers from discriminator board.

# 2 Location of files and directories

Currently, all software described here is located under directory and which could be changed in the future.

**/group/halld/Online/controls/epics/app**

All files and directories refereed in this document should be looked starting the above mentioned directory. The URL of corresponding subversion repository is:

**https://halldsvn.jlab.org/repos/trunk/controls/epics/app**

The software is in the directory **JLabDiscr_subApp/src** and the boot configuration is under directory **iocBoot/iocJLabDiscr_sub**. The database configuration and the operator interface files are in the directories **JLabDiscr_subApp/Db** and **JLabDiscr_subApp/op** respectively. See the complete tree structure of these directories below:

```
JLabDiscr_subApp
|-- Db
|   |-- dbJLabDiscr.db
|   |-- Makefile
|-- Makefile
|-- op
|   `-- css
|       |-- JLabDiscr_sub.opi
|       `-- run_opi.csh
```

```
‘-- src
    |-- all_rocs.c
    |-- dmaPList.c
    |-- dmaPList.h
    |-- dsc2.c
    |-- dsc2.h
    |-- JLabDiscr.c
    |-- JLabDiscr.h
    |-- JLabDiscr_subConfig.c
    |-- JLabDiscr_subConfig.dbd
    |-- JLabDiscr_subMain.cpp
    |-- JLabDiscr_subrecord.c
    |-- JLabDiscr_subrecord.dbd
    |-- jlabgef.c
    |-- jlabgefDMA.c
    |-- jlabgefDMA.h
    |-- jlabgef.h
    |-- jvme.c
    |-- jvme.h
    |-- Makefile
    |-- tsi148.h
    ‘-- usrvme.c
iocBoot/iocJLabDiscr_sub
|-- envPaths
|-- Makefile
|-- README
‘-- st.cmd
```

# 3 Hardware tests and constrains observed

We tested the hardware for correctness and predictability of results obtained, and to estimate the range of the accumulation time interval of the scalers. Quality checks were conducted to ensure that we observe a signal from a pulser when it is attached to particular port with a particular threshold.

The DAQ group of JLab (Bryan Moffit) provided a driver written in C programming language. The driver has a several ways to readout scalers, one of them is direct memory access (DMA). The DMA access is suitable

for data acquisition (DAQ) purposes, while for monitoring and managing the device we can use direct access of hardware registers of FPGA on the discriminator/scaler module via VME bus. There are 2 types of readout of registers implemented on FPGA firmware[1]: latching and non-latching. The speed tests were done using latching readout. Latching for these devices consists of the following steps:

1. stop the scaler counting
2. copy the value to the memory from where it would be transferred
3. reset the scaler value
4. start the scaler counting

The difference among latching and non-latching readouts is that non-latching is does not stop and reset the scaler. While not resetting the scalers is good for DAQ purposes, and it will not alter the event consistency, from other hand at least the ungated reference clock working at 125MHz is going to fill 32-bit space in $2^{32}/(125 * 10^6 Hz) \approx 34$ seconds. And if DAQ is not running, it will not reset scalers and we will not get reliable results from it, while the slow control should be able to continue to work flawlessly. Currently, we have to use latching readout in order to get consistent results.

The problem was discussed with Benjamin Raydo (Fast Electronics Group). We suggested that the firmware to be modified to have a second FIFO for monitoring. He will investigate the feasibility of implementing of additional FIFO. This way there will be no interference among the two systems. If feasible Benjamin will compile a new firmware for FPGA and the boards would be updated.

During tests we observed an about 125 $\mu s$ time spend per one reading of all 66 scalers with latching. Reading was done by calling Bryans scaler reading function and measuring the time.

# 4 Preliminary EPICS driver

## 4.1 Implementation scheme

A preliminary version of EPICS driver was implemented to test in the Hall-B during its last 6 GeV experiment. The driver used based on the EPICS

[1]This is correct as of firmware version 1.9 on the board with identificator 0x44534332

driver for CAEN v1495 FPGA board by Sergey Boiarinov, used in the Hall-B for triggering purposes. Since EPICS clients could randomly make requests to read scaler values, and depending on the frequency, lock the VME bus access, it was decided to separate the readout from hardware from serving to the clients. Therefore, the EPICS Input/Output Controller (IOC) running on VME CPU needs to have a separate thread to readout the discriminator/scaler boards and to copy these values to the RAM of the controller. For simplicity currently this readout is done every second. Then the information from memory is available to other threads of the IOC. An aSubRecord-type process variable (PV) is served by the IOC, which provides the channel access clients with the scaler values.

## 4.2 Readout Configuration

In the IOC start up file the configuration of the discriminator boards are specified by JLabDiscrInit keyword. For example:

*JLabDiscrInit 0xe00000 1 0x10000*

where the first argument is the address of the board with lowest base memory address. These addresses are specified on the board using 4-bit switches. Only last 2 switches (A(23:20) and A(19:16)) are used by the driver. These are making up the bits 16 to 23 inclusively. Therefore, it is possible to have an increment of 0x10000 bytes in the base memory addresses. Since the FPGA registers occupy memory of 0x9004 bytes, the minimum difference in base memory addresses is enough to accommodate registers. In the future, after FPGA firmware update it is possible that the registers will require more address space. In that case, it is possible to configure the driver to use more space per board. For that the jumpers should be incremented by two and the third argument of JLabDiscrInit should be 0x20000. Care should be taken during installation of the boards, so the addresses are sequentially increasing, the board with the lowest address is specified as first argument. The second is the number of boards installed and the memory difference or address increment is the third argument. By specifying these 3 arguments the hardware driver unambiguously recognizes boards and is ready to launch the periodic readout task. After initialization the boards are refereed by their number starting from zero.

Another type of entry in the IOC command script should be the loading of the record database file for PV initialization. The example entry looks like

the following:

*dbLoadRecords "db/dbJLabDiscr.db", "host=hallddisc1, id=0"*

This will create a PV called *JLabDiscr:hallddisc1_0* and that would be associated with the board number 0. For more details on the PV look at section 4.3

Using the following line it is possible to generate additional debugging information.

*var JLabDiscrDebug 1*

If there is no need for debugging information, this variable should be set to zero or the line should be omitted. The default behavior is to not produce debugging information.

## 4.3 The process variable with aSubRecord

The process variable is configured as aSubRecord type which is similar to the subroutine record, but with many output arrays. Like a subroutine record, the aSubRecord calls the user supplied routines in order to initialize and process the record. The processing of the record is configured with SCAN field specified as "2 second". The board number is specified in the INP field. The output scaler values are described below:

**VALA** array of 16 scalers for gated triggers

**VALB** array of 16 scalers for gated TDC-es

**VALC** array of 16 scalers for ungated triggers

**VALD** array of 16 scalers for ungated TDC-es

**VALE** single scaler for gated reference clock

**VALF** single scaler for ungated reference clock

The user supplied routines copy the scaler values from the memory arrays into arrays associated with the VAL* fields of the aSubRecord PV. These routines are in the file JLabDiscr_subrecord.c under the directory:

*JLabDiscr_subApp/src*

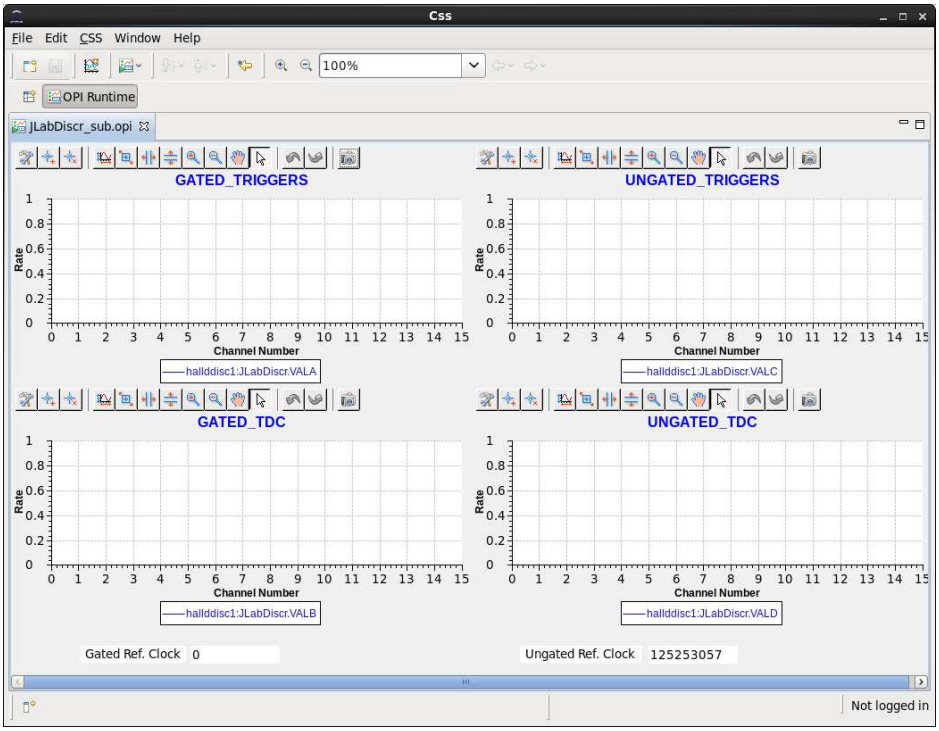


Figure 1: The operator interface for JLab Discriminator. On the left side are gated scalers and the right one are ungated scalers. Top 2 scatter plots are showing the split 16 channels each. At the bottom the corresponding reference clock are displayed.

## 4.4 Readout Task

The readout starts during initialization of driver by launching a POSIX thread (a task in the vxWorks). The thread periodically calls dsc2ReadScalers function from the basic driver library to acquire new scaler values and places them in the memory. The call is semaphore controlled restricting to one instance of readout routine to access the hardware. The period is specified in the file JLabDiscr.h and is restricted to the values from 0 to 100 seconds with a second granularity. It is configured to make a latched readout of all 66 scalers ($16{\times}2\,thresholds{\times}2\,gated\,or\,ungated{+}2\,gated\,and\,ungated\,reference\,clocks$).

### 4.5 Operator interface

Using the CSS/BOY boy a monitoring interface for operator was created. It can be started using **JLabDiscr_subApp/op/css/run_opi.csh** script. The Fig. 1 shows a screen-shot of the GUI.

## 5 Future development plan

In this section we will present our plan for development of the final EPICS interface, starting from drivers up to the user interface. The goal is to monitor and control both scaler and discriminator parameters. This job could be divided into 3 steps.

### 5.1 Scaler Driver

The software should be extended to be interfaced with mcaRecord type instead of using of aSubRecord type PVs. This will allow buffering of the scaler values. Configuration options for the readout period and buffer size should be implemented. The user interface should be modified to enable management and monitoring of these expert settings along with the monitoring of the scalers. The new firmware features to be developed by Benjamin Raydo should also be implemented in the EPICS driver support.

### 5.2 Driver for setting and monitoring parameters

Device support should be developed for setting and monitoring of thresholds, pulse widths and delays. A generic operator interface should be developed to allow monitoring and controlling of these settings.

### 5.3 Releasing, Testing and Documenting

Once the software is ready it will be released to potential users for testing. Testing will involve running conditions with heavy loads or with complex setups prune to fail. The documentation will be created for both users and experts.

Based on the work described above we estimated that development of the final EPICS support for the JLab discriminator modules will take approxi-

mately three man-weeks, and the testing and debugging will take two more man-weeks.

# References

[1] https://clonwiki.jlab.org/wiki/index.php/JLAB_Discriminators