MVD FEE test hardware and software

25 January 96

Hubert van Hecke - LANL, Nance Ericson - ORNL

In January '96, Nance Ericson from Oak Ridge brought to Los Alamos a prototype of the MVD front-end electronics boards. The boards contained the digital parts of the Heap Manager and FEE. The boards had been tested at oak ridge, using a PC. In order to prepare for the MVD beam test later in the spring, we needed to learn how to communicate to the boards via camac, and receive the data in the CODA data acquisition system. This document is mostly meant for internal record keeping. However, since there is a large amount of overlap between the Heap Manager/FEE of the MVD and that of other Phenix subsystem, the protocols and software may be of interest to the wider Phenix community. The software can be obtained by emailing requests to hubert@lanl.gov.

1. Hardware

The hardware that was tested is shown in a logical diagram in figure 1. Physically, there was the Heap Manager board, and one FEE board. In addition, there was a small diagnostic board with LEDs. The connections to the outside consisted of 3 17-twisted-pair ribbons for the serial communication (connectors J4, J7 and J8). The data were shipped out on another 17 twisted-pair ribbon, along with a single twisted pair for strobing (WSI line). These five ribbons went to 3 locations in a camac crate.

2. Contents of the camac crate

slot	module	FEE connection
1	LeCroy 2366 programmable module	J4,J7,J8 (output)
5	Jorway model 41 NIM output reg. (unused)	
12	BiRa model 2351 NIM input reg.	
14	CES lc 5171 level conv. ECL $\rightarrow$ NIM	J7 (input)
17	BiRa model 3251 NIM output reg. (unused)	
20	LeCroy 4302 memory	data bus and WSI line
21	Kinetic Systems 3291 dataway display	
22-23	Joerger Visual branch Terminator	
24	BiRa model 1302 type A2 controller	

3. Pin assignments

The pin assignments are described in the tables below. During the test, the communications ports on the LeCroy 2366 were all configured as output ports. Since lines 9-11 on J7 need an input port, the J7 cable was split, and the output channels of J7 were routed to an ECL $\rightarrow$ NIM converter, from where they were read in by a NIM input register.

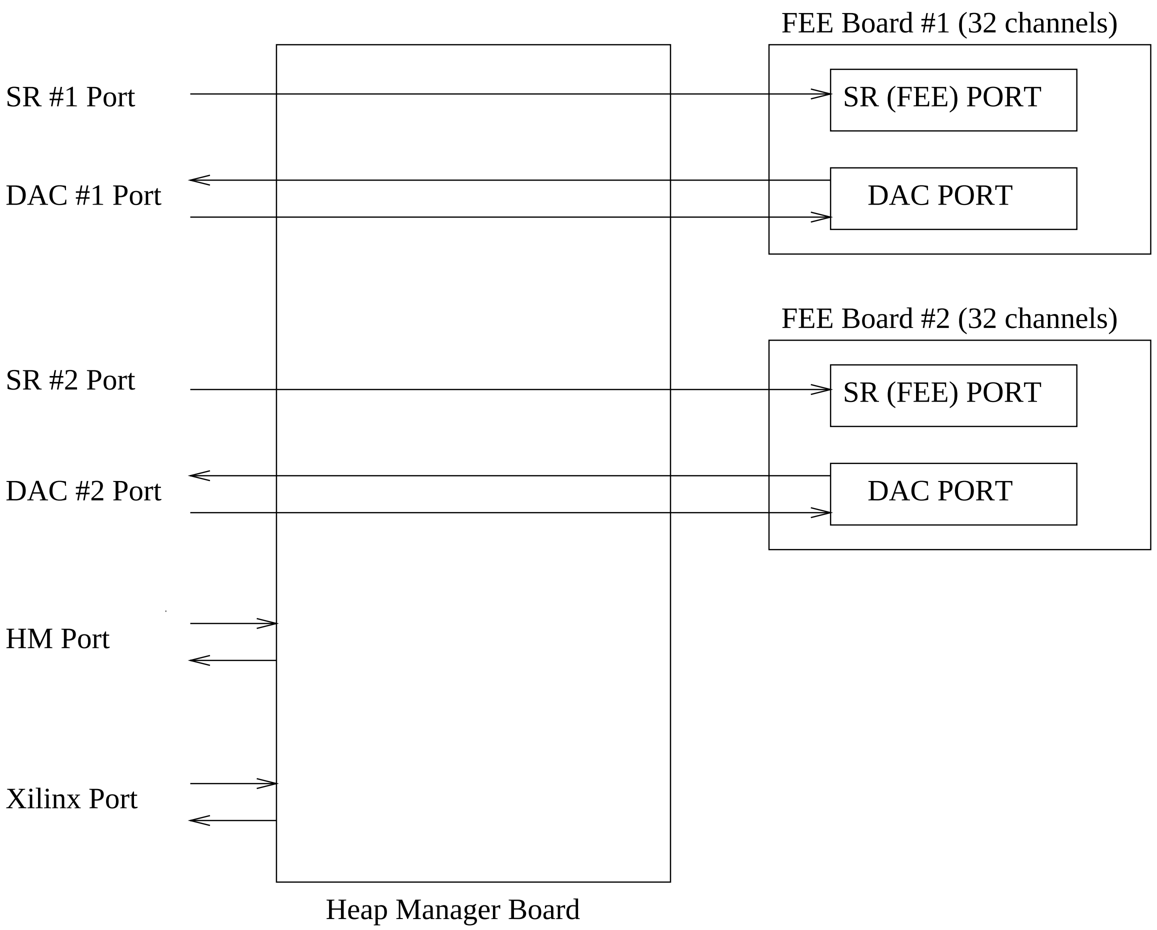

Figure 1. Logical diagram of the tested boards

On J4, channels dac1_csb and dac2_csb were swapped. The Xilinx serial lines were assigned to J8.

connector J4			
channel	pins	name	description
1	1-2	hm_din	Heap Manager - data in
2	3-4	hm_sclk	- clock
3	5-6	hm_sload	- load
4	7-8	dac2_csb	DAC2 - cs-bar
5	9-10	sr1_din	Serial 1 - data in
6	11-12	sr1_sclk	- clock
7	13-14	sr1_rstb	- reset-bar
8	15-16	dac1_csb	DAC1 - cs-bar
9	17-18	sr2_din	Serial 1 - data in
10	19-20	sr2_sclk	- clock
11	21-22	sr2_rstb	- reset-bar
12	23-24		unassigned
...	...		...
17	33-34		unassigned

connector J7			
channel	pins	name	description
1	1-2	dac1_ldac	DAC1 - load
2	3-4	dac1_din	- data in
3	5-6	dac1_sclk	- clock
4	7-8	dac1_rstb	- reset-bar
5	9-10	dac2_ldac	DAC2 - load
6	11-12	dac2_din	- data in
7	13-14	dac2_sclk	- clock
8	15-16	dac2_rstb	- reset-bar
9	17-18	dac1_dout	DAC1 - readback line
10	19-20	dac2_dout	DAC2 - readback line
11	21-22	hm_dout	Heap Manager - readback line
12	23-24		unassigned
...	...		...
17	33-34		unassigned

connector J8			
channel	pins	name	description
1	1-2	mb0	mode control - bit 0
2	3-4	mb1	- bit 1
3	5-6	mb2	- bit 2
4	7-8	mb3	- bit 3
5	9-10	mb4	- bit 4
6	11-12	xil_prog	Xilinx port - program
7	13-14	xil_sclk	- clock
8	15-16	xil_din	- data in
9	17-18		unassigned
...	...		...
17	33-34		unassigned

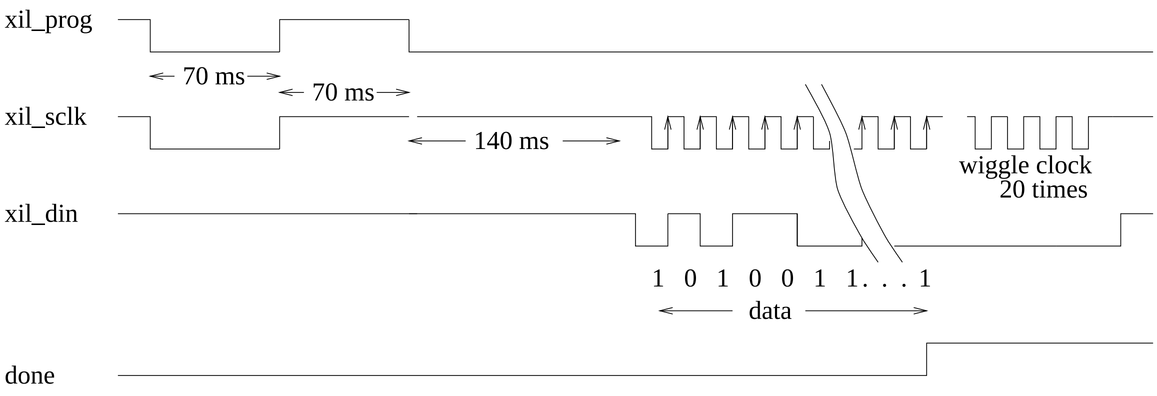

Figure 2. Timing of the Xilinx serial signals

4. Programming the Heap Manager Xilinx chip

The Xilinx chip needs to be configured with the appropriate software. The configuration is developed at Oak Ridge, and the product is a file of bits that need to be downloaded via the Xilinx serial link to the HM board. The file is an ascii file, containing about 180K bits.

Figure 2 shows the timing of the Xilinx serial signals. Initially, the xil_prog and xil_sclk lines are set and reset on timescales of tens of msec. After that, the data are clocked in as fast as the program will run, on the rising edge of the clock, as indicated. After the data are done, the clock line is wiggled 20 times for good measure. The 'done' line should come up at the end when all is done. This line is not brought out on a connector. I reset the data line at the end. The xil_prog line must be kept low. If it is allowed to go high, the program is lost and needs to be reloaded again. Note that the xil_** lines reside on connector J8, where the mode lines and the trigger line are housed. Care must be taken to not affect the xil_prog line when manipulating these other signals.

The downloading process is slow, since one bit at a time is transferred from the HP over the net to the VME cpu, from there to the LeCroy 2366 communications board, and from there to the HM. It takes about 20 minutes to download the bits. If the downloading is successful, 12 LEDs on the little diagnostic board all go out. There are several solutions to the speed problem:

1) one can have the software on the HP download a 24 bit-word to the LeCroy 2366, and have the 2366 reprogrammed so that it ships them out serially. This would speed up the process by a factor of 24 to under 1 minute. This would require the Oak Ridge Xilinx programmers to design such a configuration. It is on their list of things to do, but not near the top.

2) one can write a program in C, which executes in the VME cpu, which reads the file and loads the bits. This would completely get rid of all the network overhead and run extremely fast.

5. Mode bits

The mode bits determine the 'run state' of the onboard machine. For the current board, there are only 3 states, with mode bit patterns for run, halt all, and pipeline reset, as shown here:

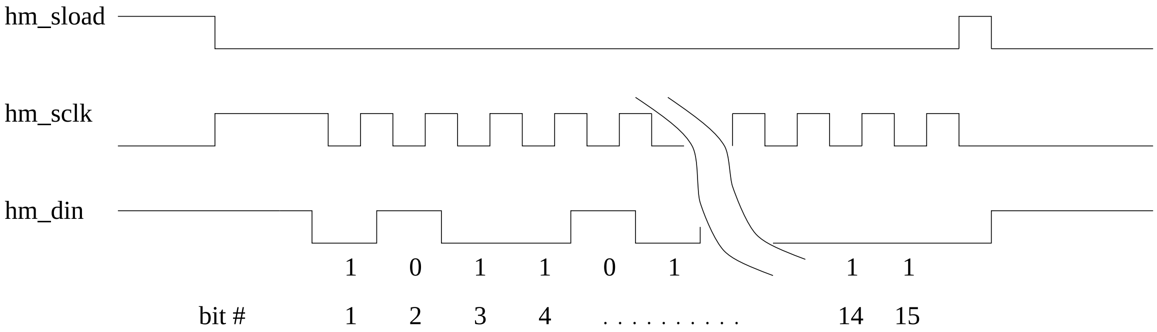

Figure 3. Timing of the Heap Manager serial port signals

command	mode bits 0-3
HALT	1111
RUN	0011
PRES	0001

In addition, setting and resetting mode bit 4 represents a level-1 trigger. These 5 lines are static, so there are no timing considerations.

6. Heap Manager serial port

Several functions are encoded in the 15-bit serial string to the HM port. The first 2 bits give the number of clock cycles between integrator resets (00 = 1K beam clocks, 01 = 2K, 10 = 4K, and 11 = 8K beam clocks). The next 11 bits set multiplexer channels; bit 14 enables or disables multiplexing, and bit 15 turns the correlator on or off. Note that in the hardware under test, the correlator was always off. Figure 3 shows the timing of the HM serial signals. The multiplexer channels M1A0 through M3A3 are connected to 11 LEDs on a small diagnostic board. A quick test of the system is to see if you can turn these LEDs on and off.

7. DAC serial ports

Each of the two DAC serial ports talks to 3 8-bit DACs, each one of which has 4 output channels (A-D), for a total of 24 settable voltages. Only DAC2 was present during the test. Full scale on these DACs was 5V. An added complication is that the data line is chained through the three chips, whereas the clock, load and cs-bar lines are parallel, as in figure 4.

To program the 3 DAC chips on a FEE board, a 36-bit package needs to be sent, consisting of 3 12-bit sub-packages, one for each of the three DACs. The first 4 bits of these 12 (labeled A1, A0, C1 and C0), determine which of the 4 channels is to be affected, and the remaning 8 bits correspond to the voltage to be set:

A1	A0	C1	C0	which DAC channel
0	0	1	1	channel A
0	1	1	1	channel B
1	0	1	1	channel C
1	1	1	1	channel D
x	0	0	0	all
x	1	0	0	no-op

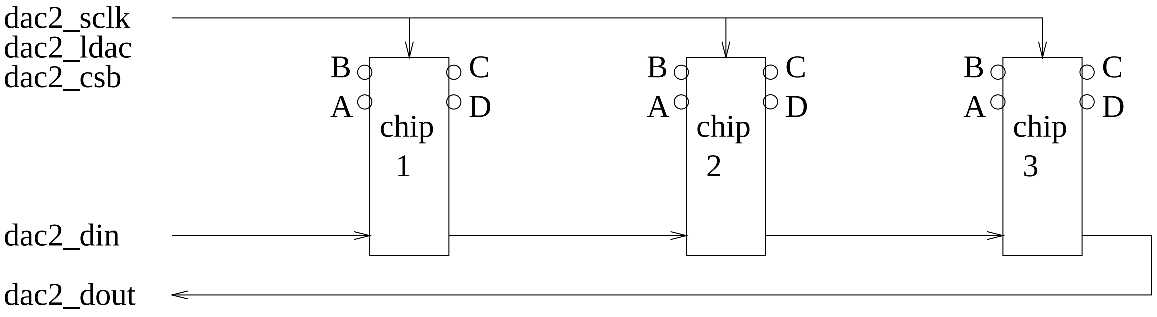

Figure 4. Lines to the DAC chips

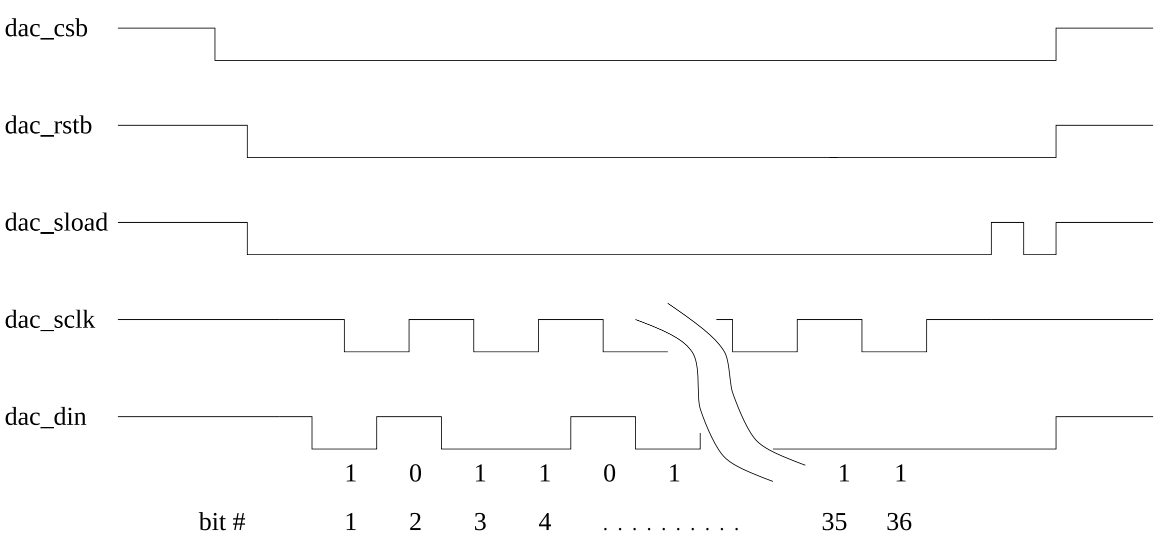

Figure 5. Timing of the DAC serial signals

Where 'x' means that the bit can be either 0 or 1. One cannot shift a packet into one chip and strobe the load line without also shifting data through the other chips, and strobing all three chips at the same time. In this context, the 'no-op' command caused lots of confusing behavior of chips. The solution chosen was to save the status for all the bits in the control program, and shift out the entire configuration each time any of the 12 voltages is changed. Thus all voltages are reloaded every time. As an example, the following string of 144 bits sets all voltages to 0, except for channel 3D, which is set to 100% of full voltage, or 5V.

```
         ------ A ----  ------ B ----  ------ C ----  ------ D ----
chip 1:  0011 00000000  0111 00000000  1011 00000000  1111 00000000
chip 2:  0011 00000000  0111 00000000  1011 00000000  1111 00000000
chip 3:  0011 00000000  0111 00000000  1011 00000000  1111 11111111
```

Figure 5 shows the timing of the DAC serial signals.

8. FEE boards serial signals

These are the serial lines labeled SR1 and SR2 in figure 1 Figure 6 shows the timing of the mode bits serial signals.

b

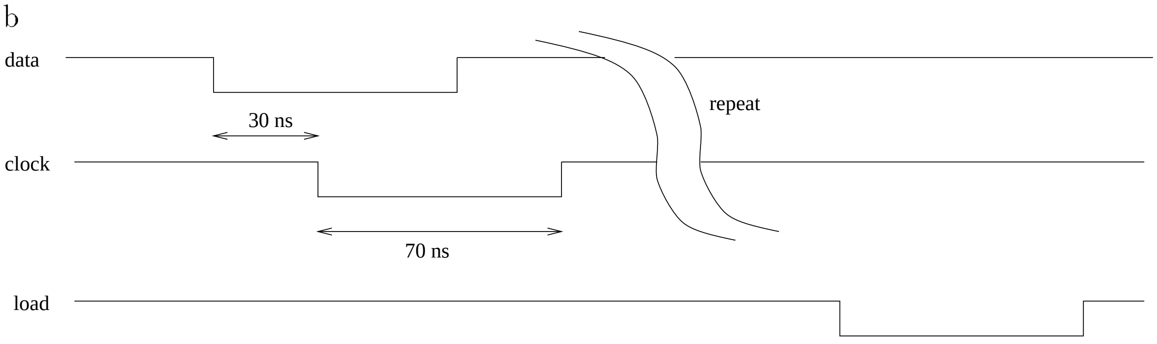

Figure 6. Timing of the SR1, SR2 serial signals

9. User interface

A program has been written that allows the user to execute each of the tasks described in the preceeding sections. It is called /hponl/usr1/mvdonl/camac/mvd_control. It is based on KUIP, and has the following command tree

```
mvd  /lecroy2366              program the LeCroy 2366 camac module
     /xilinx                  program the Heap Manager Xilinx chip
     /mode  /halt             put HM in halt mode
            /run              put HM in run mode
            /pres             pipeline reset
            /trigger          issue a level-1 trigger
     /dac1  /1A <voltage>     set DAC 1A on FEE board 1 to <voltage>
            /1B <voltage>     set DAC 1B ...
            /...              ... and so on for all 12 DACs
            /3D <voltage>     Same again for FEE board 2.
     /dac2  /1A <voltage>
            /...
            /3D <voltage>
     /clear <nwords>          clear the first <nwords> locations
                              in the LeCroy 4302, which receives
                              data packets.
     /hm_serial               download the Heap manager serial port
            /reset  nK        reset the integrator every nK clocks.
                              n=1,2,4,8
            /correlator       on/off
            /enable_mux       on/off
            /mux MxAy         on/off  x=1,2,3, y=0,1,2,3
            /test             flash the LEDs in sequence
            /status           show hm serial bit status
     /sr1
     /sr2
     /clear_data <nwords>     clear the input unit
     /read_data               read what's in the LeCroy 4302
     /inhibit                 clear the camac inhibit
     /test                    run auto test (till 'stop.dat' exists)
```

```
/menu                    show the menu
```

Currently, a command-line interface is used, though at some time later we might switch to point-and-click menus. Command MENU will bring up the following:

```
+------------- MVD CONTROL MENU -----------------------+
| lecroy2366   program the camac module                |
| xilinx       program the FEE chip                    |
| mode         mode control:                           |
|              /run /halt /trig /pres /null            |
| dac1, dac2   set voltages on board 1, 2              |
|              /1A /1B ... /3D <voltage>               |
| hm_serial    heap manager serial port                |
|              /reset /corr /enable /mux /status /test |
| clear_data   zero the input module <nwords>          |
| read_data    read what's in the LeCroy 4302          |
| inhibit      clear the camac inhibit                 |
| test         run auto test (till 'stop.dat' exists)  |
| menu         show this menu                          |
+------------------------------------------------------+

mvd==>
```

From a cold start, command LECROY2366 needs to be issued first, in order to configure the camac programmable logic module as a 4-port output register. This command reads from a binary input file, which was retrieved from the LeCroy website (anonymous FTP to www.lecroy.com, cd to pub/software/2366, file T2366OUT.BIT). About 12K bytes need to be transferred, which takes less than a minute. In the future, the module will be programmed to have some lines be input lines (on connector J7, see section 3), rather than have all of them be outputs.

A this point, we can start talking to the boards. The first thing to do is to load the Xilinx chip with command XILINX. About 180K bits are transferred, which takes 15-20 minutes. When the boards are powered up, all the LEDs on the little diagnostic board will light up. When the Xilinx chip is successfully programmed, all those LEDs will go off.

With the Xilinx chip programmed, all other commands can now be exercised. Some checks that can be done are making the LEDs on the diagnostic board go on and off with the hm/mux or hm/test commands; issuing a trigger with the mode/trig command and checking that a data packet arrives; setting a DAC voltage with the dac2 command and measuring the voltage on the board.

10. postscript

After Nance left, we moved the entire setup from room C124 to C155. In this new location, several problems were seen. The first was that the data packets received were garbled. Eventually this was traced to a reversed ribbon cable. Many ribbon cables around the system are not keyed, so they can easily be inserted upside down.

The bigger problem was that the xilinx chip would crash, usually within several minutes of being loaded. Improving the grounding and shielding had no effect on this. One major problem was spotted in the process, namely that the -5.2 power supply was badly underrated, with voltage sagging to 3.1 V. A bigger power supply solved this, but not the crashing of the xilinx chip. Eventually the problem was traced to unused, unterminated driver chips, which would go into wild oscillations. Removing these chips (c22-25, c28-29) stopped the repeated crashes.

11. CODA

When the TRIG command is issued, the system produces 2 packets of data, 63 words long, plus 6 header and 2 trailed words each. The reason 2 packets are sent is that the correlator is, for the moment, hardwired off. The data are sent over the J2 connector to the LeCroy 4302 memory module. After this is done, an extra pulse is sent to a camac NIM output module in camac slot 17. The strobe output of this module is connected to the INT4 spigot on the camac interface module in the VME crate. This arrangement triggers CODA readout of the LeCroy 4302 memory module. In the beam test, the same arrangement will be used for the FEE readout, but of course the trigger signal will not be computer-generated.

The data received in the coda analyzer program have been checked to be identical to those generated by the HM board. Relevant CODA software is located in /coda/run, files test_camlist.crl and ebana_test.f, and in /coda/rcDatabase, files rcNetwork and physics.config.