IU ADC Maintainence

December 10, 1994

Rob Lindenbusch

Department of Physics, Indiana University, Bloomington IN 47405

1 Introduction

This note descibes the tools and techniques of maintaining the IU ADCs for the LGD. There are several ADC specific programs available, as well general FASTBUS utilities.

2 Hardware Layout

The 4 crates of IU ADCs, numbered 0-3, are located in the LGD hut. The ADCs draw quite a bit of power (4 kW/crate), and a special FATSBUS power supply is needed to run a full crate of 24 ADCs. In addition, due to the high power consumption, a great deal of heat is generated, neccesitating a robust cooling system. Each crate has a Kinetic Systems FASTBUS fan unit above and below, and a 5 ton air conditioning unit is ducted into the racks to provide cooling. There is, in addition, a temperature interlock for each crate. An interlocking thermostat is located above each crate, and is set to aproximately 100 degrees. If the output temperature of the air from the crate goes above this setpoint, the line power to the supply is cut. This is done in one of the grey boxes located on the power supply rack. These boxes plug into the 208V for the supplies, and also into a 110V outlet strip. The 110V powers the relay that controls the 208V line. The outlet strip also has the fans for the crates plugged into it. These should not be moved, as this insures that the fans are powered if the power supply is. The thermostat that controlls this air conditioner is located above crate 0. This means that crate 0 must be on at all times. If it is turned off and the others are left on, they tend to overheat and trip off.

Each crate contains 24 ADC's in slots 1-24, a readout controller (ROC) in slot 25, and a Kinetic Systems segment interconnect (SI) in slot 0. A ribbon cable runs along the front of the crate, connecting the ADC's and the ROC.

3 General FASTBUS Information

Communication between the crates and the computer is handled by BBFC 2, located in the primary crate in the hut. Due to limitations of the BBFC, it cannot communicate directly with the SI's, and it must go through a SLAC Scanner Processor (SSP). There are two SSP's located in the primary crate, one in slot 1 for use in debugging the ADC's, and one in slot 6 for readout of the ADC's by the DAQ. The FASTBUS addresses of the crates containing the ADC's are 0x10000000, 0x11000000, 0x12000000, and 0x13000000, corresponding to crates 0,1,2,and 3. These addresses should not be changed.

To prepare to run the ADC test programs, the SSP has to be loaded with it's software, and the SI's have to be programmed. To load the SSP, the user runs:

$/home/mpsdaq/e852/daq/bin/bsi\_load < BBFC\# >< SSPslot >$, with $< BBFC\# > = 2$ and $< SSPslot > = 1$.

If errors occur at this stage, there may be somthing wrong with the SSP. The first thing to try is toggling the reset switch on the front of the SSP. If this does not cure the problem, a DAQ expert should be consulted.

Next the SI's need to be loaded. The user should run $/home/mpsdaq/e852/daq/bin/bsi_s i21$. This should print out something like this :

SI at 0x16 ...
SI at 0x80000010 ...
SI at 0x80000011 ...
SI at 0x80000012 ...
SI at 0x80000013 ...

The first line tells the user that the SI in the primary crate is locate in slot 0x16 (22). The logical address of the cable segement is 0x80000000, and there are SI's at addresses 0x10,0x11,0x12,and 0x13 on the cable segment. To address modules in the crates containing the SI's, the user would use (for example) address 0x1n000001 to address slot 1 and 0x1n000018 to address slot 24 in crate n, where n is 0,1,2,or 3.

Errors at this stage can indicate one of several problems. The SI's are all connected via a pair of ribbon cables that connect to a card on the back of the crate. One of these cables (or cards) may have come loose. If one SI seems to be missing, try pulling it and re-inserting it. If it still does not work, try switching it with another that does work in a different crate. If the problem follows the SI, it is probably bad. If the problem stays in the same crate, the cable may be bad, or something may be wrong with the crate (very unlikely). If you cannot get the SI's to work, call a DAQ expert.

At this point, with the SSP and SI's loaded, you are ready to begin checking the ADC's

4 ADC Testing

The primary program used to test ADC's is $/home/mpsdaq/e852/daq/bin/adc\_mdt$. This program tests all of the capabilties of the ADC's. There are two flags that are normally used. The $-f$ flag tells it not to pause between tests (At one time, problems were observed with the interface on mpsdaq when pauses were not inserted). The $-c < num >$ flag tells it to use num as the base address for the test. For example, to test the ADC's in crate 2, one would run $adc\_mdt - f - c0x12000000$. The program does the following

1) The crate corresponding to the base address given by the user is scanned for ADC's. The user is prompted for the first ADC he/she wants to check, and the last. The first **must** be to the left of the last (i.e. a higher numbered slot).

2) After the user has selected the modules to be tested, the MDT config bits is CSR#0 are set, and the pedestals are trimmed to a nominal value of 15 counts.

3) The user is then asked if he/she wants to do all the tests, which is what you normally do. The tests are :

I. All channels on.
II. All channels off.
III. Only one channel on at a time.
IV. Only one channel off at a time.
V. Only two channels on at a time.
VI. Random channels on.

Each "." printed represents a set of four tests being run. For example, the first test run (all channels on) consists of the following:

I. Fire the test pulser on all channels, with the lower and upper discriminators set below the test amplitude. Read out the modules in

   (1) Block mode from DSR#0
   (2) MDT mode from DSR#0x100
   (3) MDT mode from DSR#0x200
   (4) MDT mode from DSR#0x300

   After each readout, the data is checked to make sure that the discriminators have fired at the appropriate times. If an error is detected, an error dump is generated. The first item in this dump is a section of the data that was read out containing the "bad" channel, generally followed by the next channel, even if that channel is good. Next is an example of what was expected for these channels. The important bits are 0x2000 (lower discriminator on) and 0x4000 (upper discriminator on). The first 16 bits (0xffff0000) correspond to the id that channel has been given by the program - the upper 8 bits correspond to the slot, and the next 8 bits correspond to the channel number (from 0 to 31). The last 12 bits are the signal. This will usually not agree with the expected data, since if the channel has a cable attached and is told to fire with a value of 0x800, half of the test charge goes down the cable, and the value read out will be closer to 0x400. The discriminators and test pulse amplitude are set so that this affect will not change whether or not a discriminator should fire. The last item in the dump describes the readout mode, the test pulse amplitude, and the discriminator settings for that test.

   If a problem is found, the first thing to do is see which discriminator did not fire. Chances are that that integrator is bad and needs to be replaced (the integrator cards contain the decriminators as well). See the section below on debugging for more information.

II. Fire the test pulser on all channels, with the lower discriminator set below the test amplitude, and the upper discriminator set above the test amplitude. Do the same readout and data checking as before.

III. Fire the test pulser on all channels, with the lower discriminator set above the test amplitude, and the upper discriminator set below the test amplitude. Do the same readout and data checking as before. Note that the terms "upper" and "lower" do not require that one be set lower than the other, that is just how they are normally used.

IV. Fire the test pulser on all channels, with the lower and upper discriminators set below the test amplitude. Do the same readout and data checking as before.

5 Debugging Problems

5.1 General tools

There are several utility programs avaiable to aid in the debugging of FASTBUS problems. Most of these can be found in ~*ostrov/e852/fbtest*. Some of the more usefull ones are listed below. All require that the environment variable **BBFC** be set.

(1) **rc PA SA** - Reads one word from PA, SA is CSR space. For example, to read CSR#0 from the ADC in slot 17 of crate 2, one would use *rc 0x12000017 0*.

(2) **rcb PA SA WC** - Reads WC words from PA, SA in CSR space. For example, to read CSR#0xc0000000 through CSR#0xc00000019 from the ADC in slot 17 of crate 2, one would use *rcb 0x12000017 0xc0000000 32*.

(3) **rd PA SA** - Reads one word from PA, SA in DSR space. For example, to read DSR#0 from the ADC in slot 17 of crate 2, one would use *rd 0x12000017 0*.

(4) **rdb PA SA WC** - Reads WC words from PA, SA in DSR space. For example, to read DSR#0x0 through CSR#0x19 from the ADC in slot 17 of crate 2, one would use *rcb 0x12000017 0 32*.

(5) **rd2b PA SA WC** - Reads WC words from PA, SA in DSR space, using an SS=2 terminated read. This is for MDT transfer testing. The user needs to make sure that the modules are configured correctly. For example, to read all the data in crate 2 from DSR#0x200, one would use *rd2b 0x12000018 0x200 768* ($24 \times 32 = 768$). Note that in this case, WC only gives the maximum number of words to be read.

5.2 No ADC's were found

(1) Was a valid base address given on the command line? The base address being used is echoed by the program.

(2) Are the SI routing tables set (by running *si*)?

(3) Call a DAQ expert.

5.3 Some ADC's are not found

(1) Is the ADC correctly seated in the slot?

(2) Can CSR#0 be read using *rc*? Do the upper 16 bits = 0x01e0 (this is the IU ADC module ID). If CSR#0 is wrong, the module is bad and should be replaced. If you cannot read CSR#0 at all, a fuse may have blown. Pull the module and check the 5 fuses at the back. Replace any that are bad and try the module again. **Do not replace with fuses rated for higher current than the original!** If the fuses are ok, the module is probably bad and should be replaced.

5.4 Pedestals fail to trim on one or more channels

When a channel fails to trim (i.e. the program cannot set it to 15 counts), a message like this will be printed:

Can't trim pedestal (0xff), slot 2, channel 6, last data value 0x2060000.

The number in parenthesis refers to the ending value of the pedestal trim value for the channel. The last number is the last data read from the channel. Remember that it is the last three digits that contain the value of the pedestal. In this case, the pedestal trim was set to it's maximum value, but the pedestal was still zero. Recall that with the pedestal trim set to 0, a channel actually has a "negative" pedestal, which will show up as zero. As the pedestal trim value increases, the pedestal will also increase, eventually becoming positive. Possible problems and their solutions are:

(1) Two consecutive channels fail to trim. The first channel is an even number (0,2,4...), and both channels fail to trim with a trim value of 0xff but zero signal. This is the famous "VPED" problem, and indicates a problem with the ADC board itself. The board needs to be replaced and sent back for repairs.

(2) The channel fails to trim with it's trim value at 0x0 and a high pedestal. This is usually a bad integrator. Replace the integrator and try again.

(3) The channel fails to trim with it's trim value at 0xff and zero pedestal. This may be a bad integrator, but it could also be a shorted cable. Try pulling the cable out and trimming the channel again. If this fails, replace the integrator and try again. If it still fails to work, there is probably something wrong with the ADC, and it needs to be replaced and sent back for repair.

(4) The channel fails to trim and it's trim value is not at an extreme. This can once again be a couple of things. Remove the cable for the channel and try again. If it is ok now, there may be a lot of noise on the signal cable. Check the connector, and replace if needed. If it still will not trim, try replacing the integrator. If this fails, replace the ADC and return it for repair.

5.5 Errors during the tests

Only the first two tests ("all bits on" and "all bits off") need to be run to check for proper functioning of the ADC's. The other tests exist primarily for historical reasons. If a channel has trimmid properly but falis the test, there are a few possible reasons. In all cases, one or both of the discriminators will be wrong - this is what determines failure.

(1) Discriminators are wrong, value is zero or pedestal (15 (0xf)). The first thing to do is unplug the signal cable and test again. If it still fails, replace the integrator. If this still fails, replace the ADC.

(2) Discriminators are wrong, value looks ok. Replace the integrator. If the problem persists, replace the ADC.

(3) All of the discriminators (upper or lower) are wrong for a board. This is probably a bad DAC on the ADC board. These chips are socketed and are replaceable. They are at the top and bottom of the row of IC's closest to the front of the board. The chip at the top controls the upper discriminator, anf the chip at the bottom controls the lower discriminator. Be sure that you are well grounded while replacing these chips, as they are static sensitive. If the problem persists, the board will have to be replaced.