Tubii: A Field Of Dreams

Eric Marzec

February 16, 2015

Abstract

Here the functionality of the TUBII, as it is designed, is described in detail. The goal of this document is to inform any interested parties of what the TUBII will actually do. This will hopefully prevent the TUBII from lacking functions that everyone just assumed would be present. Or, conversly, prevent the TUBII from having functions that are unecessary. Because this document is meant to be comprehensive if there is something that is not described here, it is likely that it will not be on the TUBII. If some function that should be present is not, contact Eric Marzec (marzece@hep.upenn.edu)

Additionally, this document will continue to be updated as the TUBII progresses.

1 Motivation for TUBii

Unavoidably, in the creation SNO+'s DAQ the primary focus was what features were needed. But now that much of the DAQ is"on an even keel" so to speak there is a long list of features that could be beneficial to the experiment, though they are not strictly necessary. This is where the TUBII comes in, it serves to improve the DAQ by providing a lot of functions and utilties to the user that haven't previously been availible. It seeks to solve the problems that has plagued the experiment in the past. And hopefully it can even help address problems that may occur in the future. In summary, the TUBII is meant to solve everyone's problems forever. Obviously SNO+ will not exist in some state of utopic, experiment nirvana upon the completion of TUBII. But hopefully the TUBII will at least allow operators to have a few more degrees of freedom for how they take data and address problems with the detector as they occur. The remainder of the document will describe exactly what new functionality is provided.

2 The MicroZed

Nearly all user interactions with the TUBII will be done through the MicroZed that is mounted onto the TUBII. A MicroZed is a board with a Zynq

7010/7020 FPGA chip, USB port, microUSB and an ethernet port (along with a few blinky LEDs). The idea is that the user/ORCA will be able to send and receive data to and from the FPGA via ethernet packets. The zynq chip comes with the ability to use a linux processing system (PS) which can use C code to read and write to ethernet ports, interpret commands and send them to the programming logic (PL) of the chip by writing to registers. This allows the user to select clock outputs, set delay lengths, pulser rates, etc...

Currently I have been doing this by using telnet to access the MicroZed and start an ethernet server. The user, in the guise of, e.g. ORCA, can then connect as a client and send a packet containing a command and an argument, e.g. 'clock-0/1', 'pulser-rate', 'delay-length' which the code understands and writes to the appropriate register to act on the inputs/outputs of the board.

3 Clocks

The TUBii is designed to have two clocks. One is the 100MHz ECL clock from the TUB and the other is a 200MHz LVPECL clock. Since the 200MHz clock comes from FOX Electronics, I've taken to calling it "The Fox Clock". The Fox Clock gets converted to ECL and divided down to 100MHz to match the TUB clock. Then the user can decide which of the two clocks will be the default used clock, and which will be the backup. From there the clock signal that is decided to be the default clock is watched for if any clock signals get missed. And if more then x clocks pulses are missed the backup clock automatically takes over for the default clock. This continues until the default clock resumes normal operation. x is a value that can be set by the user. However, due to the speed of the clock x cannot go lower than a few clock pulses. Testing and prototyping will determine the exact minimum. The design for this automatic fault detection system is similar as that of the UGBoard's.

4 ELLIE

The TUBii provides a pair of the exactly the same utilities for ELLIE, one for TELLIE and one for SMELLIE. The utilities are a synchronous pulse, a synchronous delay, and an asynchronous delay. The asynchronous delay is really another synchronous delay followed by a small tune-able asynchronous delay.

5 GT Timing

The TUBii has a copy of the Global Trigger (GT) signal that is sent from the MTCD. This copy of GT is used to create two more signals, Delayed Global Trigger (DGT), and Delayed Delayed Global Trigger ($DDGT$). DGT can be set to come between 0 and 510ns after GT in steps of 2ns. Similarly $DDGT$ can be up to 1275ns after GT in steps of 5ns. From here DGT goes off the board to the ***MTCD?***. From here the $DDGT$ signal may or maynot be sent to the MTCD to be used as the $LO*$ (Lock Out*) signal. If $DDGT$ is not used as $LO*$ then another signal, that I'll call $LO*_{MTCD}$, is used as $LO*$. $LO*_{MTCD}$ is generated on the MTCD and previously has been the only signal used for $LO*$. A bit, that is set by the user, on the MicroZed is used to decide if $DDGT$ or $LO*_{MTCD}$ is used as $LO*$. The $LO*$ and DGT signals are then used on the MTCD to determine when there should be the detector should re-trigger.

6 CAEN Interface

There are two functions which the TUBII provides for the CAEN. The first of which is to convert the $SYNC$ and $SYNC24$ signals to LVDS before they are sent to the CAEN. Similarly it provides a NIM copy of each Global Trigger. Currently these conversions are done by the TIB(***ASK ANDY***), which will be replaced once the TUBII is completed.

The second function provided for the CAEN is that of a channel selector for the various analog signals that may be plugged into the CAEN. Currently any time an operator would like to change which channels are seen by the CAEN he/she would have to physically remove a cable and plug in their desired cable. This is undesireable for several reasons, not the least of which is that it leaves no record of what changes occured. And ultimately leads to no one being quite sure what the CAEN is telling them. With the TUBII changing the channels in the CAEN will be able to be done in software which can they automatically keep track of which signals are availible and which aren't. Unfortunately, the TUBII's channel selecting ability is somewhat limited, it provides eight output channels. Four of those channels have two possible inputs that can be chosen between, meaning there are 12 total inputs and 8 outputs. It also means that some combinations of channels are not possible. For example, if ESUMH and NHIT_100H were plugged into two input channels that share an output, then the user could never have both ESUMH and NHIT_100H availible to the CAEN. This scheme was chosen because it's believed to be sufficient for most nearly any user while being of maneagble complexity.

In addition to allowing the user to choose which of the plugged in channels are sent to the CAEN the TUBII will attenuate signals to ensure they

fit within the dynamic range of the CAEN. The user can also choose to send any signal to an oscilloscope instead of the CAEN. If the signal is sent out to the oscilloscope the signal will be clipped so that it is withing -2V and 0V. In summary, the TUBII will allow for 12 channels to be plugged in, 8 of which can either attenuated and sent to the CAEN or clipped and sent to an oscilloscope.

7 Baseline Monitoring

The TUBII's Baseline Monitoring provides a simple buffer between the signals from the backplane and the MTCD. This is done by putting several unity gain op-amps between the two. The goal is to prevent noise from propagating from the MTCD back into the backplane.

8 Ecal Control

I CAN'T REMEMBER WHY WE WANTED THIS. ASK SOMEONE(ANDY? RICHIE? GDOG?) Ian: Currently running Ecals involves manually switching over a cable (which cable?). The aim of putting this in TUBII is basically to make a switch were the user can select whether we want normal or ecal running without Noel having to go in and change it.

9 General Utilities

Part of the goal of the TUBii is to provide utilities that have no exact purpose in mind except that they may someday be useful to the user.

9.1 Comparators

The TUBII has two separate blocks of circuitry that each act as mimics to the MTCD's ***IT IS THE MTCD, RIGHT?*** trigger logic. In the TUBII i refer to these blocks as the Comparators, though I realize this is a horrible abuse of terminology. The difference between the two Comparators is that one is faster but the trigger threshold is set by a dial. The other is slower but the threshold can be set through software. Otherwise, both compare whatever analog signal is plugged into them to their threshold value and depending on the state of GT, DGT, and $LO*$ will emit a trigger. The reason for having the Comparators is that someone in the future may have some analog pulse that they would like to trigger on and currently doing so would be difficult. By having the Comparators this difficulty can hopefully be avoided.

9.2 Delays

The first of these is a synchronous delay. The synchronous delay is done completely on the MicroZed. There is also an asynchronous delay that is performed by doing a synchronous delay followed by a small asynchronous delay. The asynchronous part of the delay can be up to 127.5ns in length in steps of 0.5ns. The synchronous portion of the delay can be as long as necessary.

9.3 Pulser

Similar to the delays availible, the TUBII also can act as a pulser. It can provide two independent pulses, one that is synchronous, and one that is asynchronous. The asynchronous pulse is created similar to how the asynchronous delay is created, it is a synchronous pulse followed by some tuneable asynchronous offset. The offset can be as large as 127.5ns in steps of 0.5ns.

9.4 Pulse Inverter

A pulse inverter is provided by the TUBII. It is a simple circuit that takes an incoming pulse, that can be either analog or digital, and inverts it using an op-amp. The limitations of the circuit are that the incoming pulses must be between -5 and +5 volts and should not exceed 500MHz in frequency. Additionally any signal that changes faster than 5,500V/μs will not be faithfully inverted.

9.5 Ribbon Delay

A "ribbon delay" is also availible to the user. The ribbon delay simply takes any ECL signal and sends it along a ribbon cable, which will delay the signal several nanoseconds (depending on how long the cable is). However, thing are made slightly more complex because the delay is designed such that the signal will be sent along the cable five times and is buffered after each trip along the cable by an MC10E116. This means a ribbon cable that has 2ns worth of length will give a delay of 10ns. It also means that the effectiveness of the ribbon delay is limited to signals usable by the MC10E116.

9.6 Pulse Scaler

The final generic utility is a pulse scaler. The pulse scaler is designed to be used as a display of the frequency of any chosen trigger. The MicroZed simply sends out a pulse to the scaler, which then causes a counter to increment. And since the MicroZed recieves many different trigger signals any of these can be copied within the MicroZed and sent to the scaler. The value of the count will be displayed on a small display on the front of the TUBII

and also be availible to user on a connected computer (***CHECKWITH IAN THAT THIS IS TRUE***)

9.7 Translation

The TUBII will provide voltage level translation between the following standards. TTL $\Longleftrightarrow$ ECL, LVDS $\Longleftrightarrow$ ECL, and NIM $\Longleftrightarrow$ ECL.

10 Speaker

Anyone familiar with the TUB's speaker should understand what the idea behind the speaker on the TUBII. The difference between the TUB's and TUBII's speaker is that the TUBII will be able to emit noise for any of the triggers it has availible, opposed to the TUB which emits noise corresponding only to the Global Trigger. Furthermore, the sophistication of the MicroZed will allow for the speaker's clicks to correspond to logical combiations of triggers (e.g ESUMH AND !PED). (***AGAIN CHECK WITH IAN***)

11 Inner Workings

Inevitably, the TUBII will be used as a black box by the many of it's users, so ideally nothing described here will be useful knowledge to those users. But for the sake of completeness and for posterity this section will be devoted to describing the parts of the TUBII which have no direct interaction with the end user.

11.1 Shift Register Addressing

A great many of the TUBII's utilities require a shift register being loaded. Doing so generally require three separate signals, *LatchEnable* (LE), *Clock* (CLK), and *Data*. So if the TUBII had eight shift registers that would use up a minimum of 24 pins on the MicroZed. To prevent such a large consuption of pins a 3-bit addressing scheme is used. This allows for 3 pins on the MicroZed to choose which of the eight shift registers will be programmed. This multiplexed signal acts as the *LatchEnable* signal for the majority of the shift registers on the board. The $DATA$ and CLK signals are shared between all of the shift registers. This means every shift register gets programmed at the same time but only the register that is specified by the 3-bit address will be "paying attention". This allows for a total of 5 pins on the MicroZed to be dedicated to programming shift registers.

11.2 Control Register

One of the shift registers addressed as described above is the "Control Register". The Control Register is meant to hold all the bits that the user will likely only ever set once. An example of this $Select_LO_SRC$ bit, which decides if $DDGT$ or $LO*_{MTCD}$ will be used as $LO*$, as described in 5. The user will probably only ever want to set the $Select_LO_SRC$ bit once, so the bit is stored in the Control Register rather than on the MicroZed. This saves one pin on the MicroZed with effectively no cost. Also a parallel-in serial-out register is connected to the eight outputs of the Control Register, this was added for debugging purposes. It allows the MicroZed to read the state of the Control Register in a non-destructive way.

11.3 Powers

The TUBII requires an unfortunately varied amount of different powers due to the fact that it has to perform both analog and digital operations, and for its digital operations it has to use several different logic families. Each power line that goes onto the board is buffered through an inductor and regulated (with a regulator). The following powers are used by the TUBII, +15V, +5V (VCC), +3.3V, 0V (GND), -2V (VTT), -5V (VEE), and -15V. Additionally a -5V reference is created which is used in the Caen interface circuitry. It is worth noting that with ECL the VCC signal generally corresponds to 0V while with TTL VCC is at 5V. This can obviously lead to some amount of confusion, but in designing the TUBII I have tried to always use the convention that VCC is 5V and that GND is 0V regardless of which logic family is being considered.

12 Abandoned Dreams

Here is described all of the things that were considered to be put on the TUBII, but ultimately were abandoned for one reason or another.

12.1 High NHit Alert

The idea behind this is that it would be nice to have some sort of low level alert that would let an operator know everytime a high NHit event occurs. This would be similar to how the TUB speaker lets operator know when the event rate increases. The big advantage to this would be that if ORCA beached then the operator would still have a good idea of the state of the detector.

Currently though, there are no plans to have this alert for the following reasons. It would be very difficult to create a noise that is highly distinct

from the noise of the TUB without adding a decent amount of audio engineering type circuitry to an already very full board. Additionally getting the NHit without going through ORCA would require quite a bit of extra work as well. So all told this alarm would end up increasing the size of an already tubby board significantly.(***GET ANDY TO LOOK AT THIS***)

12.2 LCD Display

Aside from the pulse scaler there had been hopes for any LCD display that could present a variety of information as well as do neat things like having a scrolling marquee of neutrino buddies. Having this is still within the realm of possibility, but the front (and back) panel are expected to have 100 ports for the TUBII. So it is very likely that space constraints will not allow for an extra LCD display.

13 Shift Registers

Here is a list of all the shift registers on the TUBII. The LE MUX which is referenced is a hct238 3-8 multiplexer. The MZ will set 3 parallel bits connected to it and those bits will specify which of 8 shift register is to be programmed. For exact information on how to address each pin refer to the hct238's datasheet.

13.1 Control Register

• 8 bits. Serial in, parallel out
• Connected to pin 13 of LE MUX
• Designed to be programmed once at start up.
• Output gated by flip flops. Toggle CNTRL_RDY once programming is finished
• Each output acts as an input for the Read Control Register

13.2 Read Control Register

• 8 bits. Parallel in, serial out
• Used to read state of the Control Register non-destructively
• MZ toggles reads it serialy by toggling Read_Enable pin (pin 1) each time it wants to see the next bit.
• The register is circular. So reads should be done 8 at a time.

13.3 CAEN Register

• 16 bits. Serial in, parallel out
• Connected to pin 15 of LE MUX
• The bits 0-3 select between analog pulses for the CAEN
• The 4-7 bits do nothing
• The bits 8-15 choose if analog pulses go to the CAEN or the scope outputs
• Bits 8-15 are gated by flip-flops. Toggle CAEN_RDY once programming is finished

13.4 GT Delays Register

• 16 bits. Serial in. No output
• Connected to pin 14 of LE MUX
• The data sets the delay length for the two delays
• Bits 0-7 set the time between GT and DGT
• Bits 8-15 set the time beteween GT and DDGT (LO)
• Bits 0-7 are for a DS1023200 and bits 8-15 are for a DS1023500. See their respective datasheets for programming information.

13.5 Generic Async Pulse Register

• 8 bits. Serial in. No output
• Connected to pin 7 of LE MUX
• The data sets the delay length for the generic pulser.
• The delay length should not exceed the time between pulses. I.e it should not try to delay a 1GHz pulse by 10ns.
• The delay chip is a ds102350. See it's datasheet for programming information.

13.6 Generic Async Delay Register

• 8 bits. Serial in. No output
• Connected to pin 9 of LE MUX
• The data sets the delay length for the generic delay.
• The delay length should not exceed the time between pulses. I.e it should not try to delay a 1GHz pulse by 10ns.
• The delay chip is a ds102350. See it's datasheet for programming information.

13.7 Comparator DAC Register

• 12 bits. Serial in. No output
• Connect to pin 11 of LE MUX
• The data sets the output voltage value for a DAC which acts as a threshold for one of the Comparators.
• The DAC is a AD7243. See datasheet for programming information.

13.8 Missed Clock Count Register

• 8 bits. Serial in, parallel out
• Connected to pin 10 of LE MUX
• Data is gated. Toggle TubiiTime_Data_Time once programming is finished.
• Outputs the value for how many missed counts are allowed before clock switch over