• TRIGGER
  – General Trigger Architecture
  – Trigger Tower Level Signals
  – Trigger Veto Generator
  – Trigger Scheduler
  – Trigger Message Generator
  – Trigger Data Contribution
  – Summary of Features / Limitations

• FILTER
  – Review of development path
  – Event Size/Event Format
  – CPU Comparisons
  – Results
  – Summary/Problems

• TRIGGER & FILTER
  – Bringing it all together
What I Am Not Talking About

• Conditioning of the Front-End Signals
  – This is an important issue, if we’ve got time, let’s talk

• Details of GLT Commanding, Control, and Configuration

• Internal Timing (how long to form the trigger decision)

• How to Test the GLT / How to Determine the Correct Timing
  – Note that this does not mean we have not thought about these problems and how to address them
    • For example, the CAL beam test offers a unique opportunity in this area
  – It is just beyond the scope and time for this presentation
  – For those ultimately reviewing the GLT design, this is a good place to stick your finger in
Trigger Block Diagram – Trg Path Viewpoint
Trigger Front – End Signals

• ACD Primitives
  – 216 ACD Lo Threshold Signals
  – 12 ACD Hi Threshold Signals

• CAL – 2 / tower:
  – 16 tower level ORs of LO discriminator.
  – 16 tower level ORs or HI discriminator

• TKR – 1 / tower:
  – 16 tower 3-in-a-row coincidence of layer ORs.

• Remember the number of wires is 2x the numbers above
  – IO Pins are a problem
Trigger Veto Generator

- Acts as a receiver for the ACD signals
  - Addresses the IO problem
  - Reduces 216 signals to 18 outputs destined for Trigger Scheduler

- ACD signals are logically grouped
  - Tile signals (96 signals, only 89 active)
  - Ribbon signals (12 signals)
    - These are basically ignored by the Veto Generator

- All the tile signals are routed to 18 OR gates
  - Each of the 96 inputs can be enabled/disabled
  - 16 of these OR gates are used as the TKR tower vetos
  - The remaining 2 are (can be?) used to define
    - A set of UPPER ACD tiles
    - A set of LOWER ACD tiles
    - May be used with the CAL signals as an additional veto
      - Working with Steve Ritz on this point
Trigger Scheduler

- Processes the trigger signals and generates a trigger request message

- So what are the received signals?
  - From the towers and AEM
    - TKR 3-in-a-row (16 processed to 16 raw/1 summary)
    - CAL LO (16 processed to 16 raw/1 summary)
    - CAL HI (16 processed to 16 raw/1 summary)
    - ACD HI or CNO (12 processed to 12 raw/1 summary)

  - From the Trigger Veto Generator
    - ACD Tower Level Veto (16)
    - ACD UPPER/LOWER (1 or 2)

  - Internal Signals
    - CPU trigger request (solicited)
    - Periodic
Trigger Scheduler

• Before any coincidence can be formed a coincidence window needs to be generated
  – Width of the window is determined by the trigger jitter
    • Complicated ways to dynamically optimize this
    • We have just fixed this at a fixed, but programmable width
      – Likely around ~500nsecs

  – Only signals that may result in a trigger can initiate a window start, called a window turn
    • For example, ACD veto signals cannot produce a trigger, so they cannot start a window turn
      – Well, at least not in normal running
      – They can for diagnostic purposes
    • Therefore, all inputs to the trigger have an enable/disable to control whether they can start a window turn
Can now form coincidences

- A coincidence is determined by integrating the signals during the time the window is open
  - If a signal is TRUE any time during the time the window is open, it is considered TRUE at window close time

- At window close time
  - 3-in-a-row TKR tower signals are ANDed with their corresponding veto signals
  - The UPPER ACD is ANDED with NOT CAL_HI
    - Again, talking to Steve about this
  - This reduces 18 signals to 2 signals.
Trigger Scheduler

- Scheduler forms an 8 bit vector consisting of
  - TKR 3-in-a-row
  - CAL LO
  - CAL HI
  - ACD HI (CNO)
  - TKR 3-in-a-row, vetoed
  - CAL vetoed
  - Solicited (cpu trigger)
  - Periodic

- State of this 8 bit vector is used to drive the Trigger Message Generator
  - Well, not quite, 256 was a bit much to implement
  - So the 256 inputs are mapped down to 16 via a lookup table
  - Effectively means 16 independently controlled trigger slots
  - This 4-bit value is passed on to the Trigger Message Generator
Trigger Message Generator

• Purpose is to generate a trigger message using
  – The 4 bit vector from the Trigger Scheduler
  – The throttle line
  – The event number

• How does it do this?
  – The 4 bit vector indexes a 16 entry table giving the static portion of the trigger message
    • 8 bit prescaler
    • 5 bit destination address (which CPU to send it to)
    • 1 bit ACD/CAL Zero Suppress enable/disable select
    • 1 bit CAL auto-range enable/disable
    • Miscellaneous other techno-geek bits
  – The prescale is checked, then decremented
  – The throttle is checked
Trigger Message Generator

– If not inhibited by the prescale or throttle, then
  • If the destination CPU address is dynamic (one of the techno-geek bits)
    – The CPU address is selected from the next entry in a circular buffer
      » 32 entries allows static load balancing to ~3%
  • If the destination is static, it is just used
  • The event number is added to the trigger message
– The trigger message is broadcast to 16 TEMs/AEM/GLT
Features

- The event number *names* the event and is tacked onto every contributor’s data
  - Allows integrity checking after the event is assembled
  - No longer any need for a local timestamp
    - A global clock is used throughout the system
    - Timestamp is rigidly tied to the event number in the GLT

- The destination address can be a broadcast address
  - Allows synchronization events to passed to the system
    - For example, orderly shutdown of the data pipeline
    - Shuttles diagnostic event classes to a particular CPU
      - CNO events
      - Pedestal monitoring events
Trigger Block Diagram – Data Viewpoint
GLT Data Contribution

<table>
<thead>
<tr>
<th>Tracker 3-in-a-row</th>
<th>Trigger Request Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAL HI</td>
<td>CAL LO</td>
</tr>
<tr>
<td>ACD HI</td>
<td>ACD Tower Vetoes</td>
</tr>
</tbody>
</table>

**ACD Veto List (4-8 words)**

<table>
<thead>
<tr>
<th>Inhibited by deadtime</th>
<th>Deadtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>(count of window turns)</td>
<td>(count of SYSTEM clocks)</td>
</tr>
<tr>
<td>Sent</td>
<td>Inhibited by prescaler</td>
</tr>
<tr>
<td>(count of window turns)</td>
<td>(count of window turns)</td>
</tr>
</tbody>
</table>

**TIME**

(count of free running 20MHz clock)

**PPS**

(value of free running count register at last 1PPS time hack)
GLT Data Contribution

- Logic can be checked by comparing with TEM based signals
  - Trigger Request vector
  - 3-in-a-row
  - CAL Low
  - CAL Hi
  - CNO
  - Tower Vetoes
  - ACD Veto List
- Deadtime from Window Turns can be the monitored
  - # of Window Turns = ‘Prescale’ + ‘DeadTime’ + ‘Sent’
  - Essentially
    - Prescale counts voluntary deadtime
    - DeadTime counts involuntary deadtime
- The time of the event is precisely tagged using PPS + TIME
  - Well not quite, the PPS must reference the actual GPS msg.
  - It has the value of the 20MHz at the 1PPS and a 1PPS counter
GLT Flexibility

• Veto Generation
  – Arbitrary definition of tiles shadowing the towers

• Trigger Generation
  – Trigger is an arbitrary combination of the input signals
    • For example: TKR + CAL_LO
  – Ability to define and prescale monitor triggers
  – Ability to take non-threshold suppressed ACD/CAL events
    • Allows one to continuously monitor the pedestals
  – Ability to take non-autoranged CAL events
    • Allows one to monitor range overlap in CNO events

• Trigger Data
  – Monitors the logical consistency of the GLT itself
  – Can be beat against the TEMs to monitor consistency
  – Precise deadtime monitoring
    • Some help in determining the source of the deadtime
Additional Flexibility in the GLT

• Sufficient range on various timing registers to map out the timing response

• Input signals are maskable to remove hot/noisy channels

• Intermediate signals are maskable
GLT Trigger Limitations

• Prescales limited to 8 bit counters
  – Not a problem for physics triggers
    • Even a 10K trigger can be scale down to 40 Hz
  – Periodic trigger is a problem
    • It will work from a scaled down system clock, but…
    • Not enough dynamic range
      – Want ~100Hz to 10KHz for Calibration/Testing purposes
      – Want ~.1Hz to 100 Hz for other purposes

• Clock time registers have finite width necessitating keep alive triggers

• Considering adding ‘trigger received counters’ on the TEMs
  – Allows cross-checking when the system timeouts

• Should we record the 216 individual ACD signals or just the ORs?

• And the biggie, can one really get a reliable coincidence between the CAL and TKR?
Event Filtering

• How we are doing it and how we are doing at doing it

• Input/Output Event Size estimates

• CPU Comparisons

• Results

• Summary/Problems
Event Filtering – Development Path

- **Using Monte Carlo Events Generated In GLASTsim**
  - Event size includes noise at prescribed rates (see next slide)
  - Event layout is “near final” DAQ format
    - Since dPDR we have taken another step closer to this goal
    - We now have real hardware to check data format

- **Algorithmic Development**
  - Designing and debugging on SUN/LINUX boxes
  - Measuring performance on Motorola MV2303 and RAD750

- **Event Features Used In Current Round Of Analysis**
  - TKR layer hit bits (very fast access)
  - ACD tile hit bits (disordered but access still fast)
  - CAL energy sums (slowest access … needs coarse calibration constants)
  - Minimal track finding
Event Filtering – Event Size

- Event Sample Generated With
  - $1 \times 10^{-2}$ CAL noise occupancy x 1536 x 2 logs ends
  - $1 \times 10^{-4}$ TKR noise occupancy x 884k strips

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Hits</th>
<th>Noise</th>
<th>Fixed Overhead</th>
<th>Bits/hit</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>26 logs</td>
<td>30 logs</td>
<td>32 bits / tower * 16 towers</td>
<td>32</td>
<td>2304</td>
</tr>
<tr>
<td>TKR</td>
<td>93 strips</td>
<td>88 strips</td>
<td>72 bits / tower * 16 towers</td>
<td>20</td>
<td>4772</td>
</tr>
<tr>
<td>ACD</td>
<td>5 tiles</td>
<td>1 tile</td>
<td>216</td>
<td>16</td>
<td>312</td>
</tr>
<tr>
<td>TRG</td>
<td></td>
<td></td>
<td>320</td>
<td>40</td>
<td>320</td>
</tr>
</tbody>
</table>

  “Typical” Event 7708 963

  Monte Carlo Measurement 750-850 Bytes

- This is the input event size
  - Output event size will be smaller
  - My guess, factor of 2, into the 400-500 byte range
  - Dan Wood did some work on the tracker data
## Event Filtering – CPU Comparisons

### Feature Comparison

<table>
<thead>
<tr>
<th>Processor Board</th>
<th>CPU</th>
<th>Instructions Per Cycle</th>
<th>Clock Speed (MHz)</th>
<th>Memory Wait States</th>
<th>Execution Units</th>
<th>L1 Cache (kByte)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU</td>
<td>Memory</td>
<td>Integer</td>
</tr>
<tr>
<td>Motorola MV2303</td>
<td>603</td>
<td>1.25</td>
<td>200</td>
<td>66</td>
<td>9 + 1 + 2 + 1</td>
<td>1</td>
</tr>
<tr>
<td>NRL custom</td>
<td>603</td>
<td>1.25</td>
<td>133</td>
<td>50</td>
<td>6 + 4 + 4 + 4</td>
<td>1</td>
</tr>
<tr>
<td>BAE RAD 750</td>
<td>750</td>
<td>1.86</td>
<td>133</td>
<td>33</td>
<td>4 + 1 + 1 + 1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Performance Comparison

<table>
<thead>
<tr>
<th>Processor Board</th>
<th>CPU</th>
<th>Memory</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola MV2303</td>
<td>603</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>NRL custom</td>
<td>603</td>
<td>1.0-1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>BAE RAD750</td>
<td>750</td>
<td>2.0</td>
<td>1.0-1.5</td>
</tr>
</tbody>
</table>
## Event Filtering - Results

<table>
<thead>
<tr>
<th>Cut</th>
<th>Events</th>
<th>&lt;Time&gt; [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Analyzed (%)</td>
<td>Rejected (%)</td>
</tr>
<tr>
<td>No CAL LO + Veto Tile</td>
<td>15420 (100.0)</td>
<td>9923 (64.4)</td>
</tr>
<tr>
<td>ACD Splash Veto (pass 0)</td>
<td>5497 (35.6)</td>
<td>1566 (10.2)</td>
</tr>
<tr>
<td>CAL &lt; 350MeV + Veto Tile</td>
<td>3931 (25.5)</td>
<td>224 (1.5)</td>
</tr>
<tr>
<td>CAL &lt; 10 Mev + Any Tile</td>
<td>3707 (24.0)</td>
<td>464 (3.0)</td>
</tr>
<tr>
<td>ACD Splash Veto (pass 1)</td>
<td>3243 (21.0)</td>
<td>69 (0.4)</td>
</tr>
<tr>
<td>TKR tower match with ACD top tile</td>
<td>3174 (20.6)</td>
<td>424 (2.7)</td>
</tr>
<tr>
<td>TKR tower match with ACD side tile</td>
<td>2750 (17.8)</td>
<td>304 (2.0)</td>
</tr>
<tr>
<td>No connection between CAL energy &amp; TKR</td>
<td>2446 (15.9)</td>
<td>1152 (7.8)</td>
</tr>
<tr>
<td>CAL Energy Layer 0/Total Energy &lt; .01</td>
<td>1294 (8.4)</td>
<td>156 (1.0)</td>
</tr>
<tr>
<td>CAL Energy Layer 0/Total Energy &gt; .90</td>
<td>1138 (7.4)</td>
<td>94 (0.6)</td>
</tr>
<tr>
<td><strong>Before track finding</strong></td>
<td>1044 (6.8)</td>
<td>14376 (93.2)</td>
</tr>
<tr>
<td>TKR/ACD matching</td>
<td>1044 (6.8)</td>
<td>262 (1.7)</td>
</tr>
<tr>
<td>Projects into skirt region</td>
<td>782 (5.1)</td>
<td>83 (0.5)</td>
</tr>
<tr>
<td>E &lt; 350 Mev, Number of Tracks &lt; 2</td>
<td>699 (4.5)</td>
<td>461 (3.0)</td>
</tr>
<tr>
<td><strong>Final</strong></td>
<td>238 (1.5)</td>
<td>15182 (98.5)</td>
</tr>
</tbody>
</table>
Event Filtering - Summary

• Compared With Jan PDR
  – Rejection Rate has gone from 82.6% → 98.4%
  – Time (MV2303) has gone from 15-20 µsec → 7.5-12.5 µsec

• Still Need To Go From 98.4% → 99.8%
  – But have the numbers on our side
    • >95% rejection in 14 µsec/event (RAD750) leaves 1.4 msec/event
      – To preserve 100% margin in one CPU, still have 700 µsec/event
        » This is 50 times the event processing time used so far

• More Confident In Where We Stand
  – Do not need to extrapolate estimates as far
  – Know the target CPU performance much better
    • 1 BAE 750 or 1.5 NRL 603e is sufficient to do the filtering with 100% margin

• Can Now Return To Estimating Other CPU Demands
  – Previously considered small compared to filtering
Event Filtering Summary

- Given the recent decision to accept a baseline of 3-4 EPUs, the filtering problem no longer drives the # of CPUs hardware decision
  - So, other issues are getting my attention
    - Documents
    - IVV survey
    - ITAR/VISA issues
  - This is NOT to say that this is still not an important issue
    - After all, it has NOT been demonstrated that we can filter sufficiently to fit into the available bandwidth
    - All that is being said here is that CPU cycles are not the problem
Problems

• Photon Efficiency remains a mystery, why?
  – Easy to determine the numerator
    • Just run the filter and count how many survive
  – It’s the denominator
    • Not every photon producing a MC event is analyzable
  – Steve and I need to solve this problem together
    • And here in lies the real problem

• The cross-section of Steve and I having an overlap of free time is vanishingly small
• More on this at the end
Bonus Coverage

• With the simple TKR/ACD shadowing algorithm (in trigger hardware or early stage selections), we have achieved ~70% rejection.
• Is this good enough?

• If not
  – I’ve tried moving some of the simpler filter cuts to hardware
  – In particular, if CAL_HI is clear, then pitch the event if any ACD tiles in the UPPER portion are hit
    • This pitches an additional 9%
    • But the more I think about this, the wackier it seems.
      » Why bother with TKR/ACD veto, it’s a subset of this one?
      » Answer is, can tune this cut
# Trigger Classes - Detailed

<table>
<thead>
<tr>
<th>Trig Veto</th>
<th>Splash</th>
<th>Accl Lower</th>
<th>Accl Upper</th>
<th>CAL LO</th>
<th>TKR</th>
<th>CAL LO &amp; TKR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Back</td>
<td>Gamma</td>
<td>Back</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.4</td>
<td>17.4</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9</td>
<td>22.0</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.6</td>
<td>2.8</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.4</td>
<td>1.5</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0</td>
<td>.0</td>
<td>.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>.0</td>
<td>.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAL LO</th>
<th>TKR</th>
<th>CAL LO &amp; TKR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>Gamma</td>
<td>Back</td>
</tr>
<tr>
<td>40.6</td>
<td>4.6</td>
<td>4.2</td>
</tr>
<tr>
<td>12.0</td>
<td>4.6</td>
<td>3.7</td>
</tr>
<tr>
<td>2.4</td>
<td>0.0</td>
<td>1.3</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.7</td>
<td>1.5</td>
<td>6.3</td>
</tr>
</tbody>
</table>
## Trigger Classes Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Background (%)</th>
<th>Gamma (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impossible</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TKR Veto</td>
<td>72.6</td>
<td>15.4</td>
</tr>
<tr>
<td>CAL Veto</td>
<td>8.6</td>
<td>6.5</td>
</tr>
<tr>
<td>Splash Veto</td>
<td>.3</td>
<td>.0</td>
</tr>
<tr>
<td><strong>Total Veto</strong></td>
<td><strong>81.6</strong></td>
<td><strong>22.0</strong></td>
</tr>
<tr>
<td>TKR</td>
<td>8.5</td>
<td>37.0</td>
</tr>
<tr>
<td>CAL LO</td>
<td>7.4</td>
<td>22.0</td>
</tr>
<tr>
<td>TKR &amp; CAL LO</td>
<td>1.9</td>
<td>15.4</td>
</tr>
<tr>
<td>CAL HI</td>
<td>.6</td>
<td>3.9</td>
</tr>
<tr>
<td><strong>Total Triggers</strong></td>
<td><strong>18.4</strong></td>
<td><strong>78.0</strong></td>
</tr>
</tbody>
</table>
Open Issues

• Filtering
  – Common question:
    • Why isn’t the filtering done?
      – Final reduction to the 1:300
      – Studies on losing a tile and other realistic failure scenarios
  – Answer:
    • It has taken a priority hit
    • FSW’s interest in the filtering was how many CPUs does it take.
    • With 3-4 EPUs, no longer interesting, we’ve got enough
  – More serious issue
    • Steve and JJ are swamped with so many other things, that amount free time overlap is vanishingly small
    • Is it time to give this problem to someone else?
      – Honest answer, I suspect for both of us, it is the one thing that keeps our creative juices flowing
Open Issues

• **Pulse Pileup**
  – Analog signals and hence digital signals can persist ~msec in face of CNO particles.
  – These signals can be captured in subsequent events.
    • Could result in (e.g.) ghost tracks in TKR.
  – Not simulated in Monte Carlo
  – Impact not understood
    • Effect on the on-board software filter
    • Effect on offline “analyzability” of events

• **TOT integrity**
  – Pile-up can destroy TOT information
    • Unknown which are destroyed
  – DAQ problem is that TOT is not buffered
    • Fundamental problem is analog signal can be a 0.1-1msec signal
    • Trying to mix this in a system with a 20usec response time