29 August 1998 HjC, EJ, LG, JE, ZM

An Example of a Trigger Mix for a STAR run

Introduction:

This note describes a possible trigger setup for a STAR run. The key to the trigger is understanding that there is only one Level 0 trigger value per crossing. This value represents one of a set of mutually exclusive conditions[1] placed on signals from the fast detectors. If the value represents a state that is degenerate with respect to L0 conditions but the degeneracy is removed at Level 1, 2, or 3, the value may be modified at these later levels. We count each incidence of each trigger value combined with the BUSY condition of STAR, as presented to the Trigger Control Unit and as accepted throughout the trigger system.

The signal space considered first is the set of signals from the CTB, MWC, and ZDC. These detectors measure the charged particle hits within highly segmented eta,phi space. We define event types based on the signals from these detectors. We further refine the definition of an event type at each level within the trigger/DAQ system as we apply further calculated constraints on the fast signals, including EMC, and as we add signals and constraints from the slower detectors, the SMD, TPC, SVT, FTPC, SSD.

Items explicitly recognized as requiring further discussion are flagged with **.

Simple run:

As an example, consider a run in which only high multiplicity crossings are selected at Level 0. From this set of collisions we may wish to keep only those events having a central vertex within a constrained region. Thus, we will add another constraint to the definition of these central events, namely, the high multiplicity crossings having a well defined primary vertex. We use track stubs at Level 3 to determine the vertex location. We may find the vertex was upstream of STAR in the beampipe, and simply reject the event. Or, we may want to keep a sample of these "background" events to help tune the line and the experiment trigger, so we would keep every 100th such event, thereby defining and applying a new prescale condition to these central events at Level 3. New constraints and prescales can be added at any level of the trigger.

Multiple Trigger Mix:

To examine a multi-trigger (Level 0) run, assume we want a mix of 3 levels of centrality plus a specific set of peripherals. Define m as the multiplicity. The centrality cut is based on multiplicity conditions being met by the sum of the CTB and MWC. The peripheral cut is based on meeting a multiplicity condition, 2.le.m.le.5 real hits in the CTB+MWC, a time condition, at least one hit "in time", and a symmetry condition, that pairs of hits appear in "opposite octants". While we may eventually apply this symmetry condition at Level 0, for the heuristic purposes of this document we assume symmetry conditions must be applied at Level 1. All detectors feeding DSM boards get automatic correction into "true hit counts" through a look-up-table (LUT). While we may also have the EMC, it is not relevant for this

[1] Note that min bias flag is an exception since it is not mutually exclusive with other trigger conditions.

assumed trigger mix, since the definition of the trigger types themselves does not explicitly place any conditions on EMC data. We can see how to add EMC info later.

L0 Trigger Value:

We begin by defining 7 levels of total multiplicity, M. We use a value of maxhits as the largest sum of CTB+MWC expected in the run. The m(total) value is converted to M value in the LUT for the last DSM.

M=1 no hits in CTB or MWC
M=2 1 hit total
M=3 2.le.m.le.5 hits total
M=4 5.lt.m.le.20% maxhits
M=5 20%maxhits.lt.m.le.90% maxhits
M=6 90%maxhits.lt.m.le.maxhits
M=7 .gt.maxhits

M is derived from the CTB sum + MWC sum. M regions do not overlap in m space.

We have additional information from the CTB slats. Each slat provides a single bit to indicate the arrival time window of its hits. This may be useful in cosmic-ray background rejection. The peripheral trigger includes the condition that at least one of its hits fell inside this time window. Thus, when the M=3 condition is met, a check is made to see if any hit set its time bit. If so, this constitutes a potentially valid event for the peripherals program. The symmetry condition will be applied at Level 1.

Next we define signal levels from the ZDC. This detector is meant to verify a central collision, for which it is expected to have a non-zero but not large signal in both the East and West ZDCs. Call them ZDCE and ZDCW. Define ZDCS=sum of hits in ZDCE and ZDCW.

Now we can form some Level 0 trigger types, just based on conditions placed on M, Time, and ZDC. Consider the following list for this run.

Table I : L0 Triggers

Trigger Value (TV)	L0 Conditions[2]	Description
0	M=1, ZDCS=0	no activity detected in fast detectors
1	min bias counter = 1	external min bias counter set
2	(M=6 + M=7) • ZDCE.gt.0 • ZDCW.gt.0	This is a high multiplicity event and is expected to have a non-0 signal in both ZDCs.
3	M=3 • Time = 1	This is a candidate peripheral collision. The Time constraint is for at least one of the hits being in-time. The "opposite octant" constraints will be applied at Level 1.
4	M=2 + M=4 + (M=3 • Time.ne.1)	This is a low multiplicity event. We may eliminate beam-gas events using a symmetry cut in L1.

[2] + means OR and • means AND

5	M=5	This is a mid-range multiplicity event
2000	none of the above	

We have now produced 6 trigger values for this run: the three multiplicity bins requested (0.lt.m.le.20% max, 20%<m.le.90% max, and gt.90%max), plus the peripherals, plus the minimum bias. Trigger value 0 just means there was no activity detected in the CTB, MWC, or ZDC. Value 2000 is selected to contain all the remaining crossings. We may want to keep a subset of these to see what gives signals in other detectors without leaving a trace in the fast trigger detectors.

Prescales:
We could use 6 different Level 0 prescale values for these trigger types. For instance,

TV	L0 prescale (L0ps)	Description
1	1	assume min bias prescale is done in min-bias unit
2	1	want all of these
3	10	All the physics committee would allow
4	20	this gives us "correct" weighting (I'm guessing here)
5	5	Guessing again
2000	1000	just to look

Conditions leading to trigger values having prescales of 1 are checked first in the FPGA that sets the input value sent to the TCU. Thus, if the minimum bias flag is set, the FPGA jumps immediately to its output section and sets the output value=1. If the multiplicity is greater than maxhits and ZDCs are hit, it jumps to output with value=2, etc. Thus, if the crossing happened to have the min bias bit set and (M=6, ZDCE and ZDCW gt 0), the value would be set to 1, not 2, just because the FPGA was programmed to check the min bias flag first. Note that the order in which these conditions are checked implies that conditions checked later in the tree have an automatically applied (not condition) on the conditions checked earlier.

Complications:
Two points should be noted with this scheme. The first is that TV3 contains a subset of the events desired in the TV4 category. TV3 satisfies the 1.le.m.le.20%max criterion of TV4, but TV3 imposes an additional Time condition which is deemed irrelevant for TV4: to make the TV4 selection time independent, we need to include some set of the "in time" M=3 events, since the out-of-time events of 2.le.m.le.5 are included in the OR condition of the definition of TV4. The second point is that we have not met all of the conditions we specified for selecting peripheral collisions, a condition we will call "L0 crudeness", since we have not checked "opposite octant". Let's see how we can correct these apparent difficulties.

TV3 represents a significant difference from the other trigger values in that it requires additional constraints be applied before the event is accepted. This means a TV3 value will invoke additional calculation at Level 1, Level 2, or Level 3. This begs two questions: What should we do if we reject the event at L1? and What should we do if we can't perform any further calculations for the event?

Lets deal with the second question, inability to perform further calculations. We have decided that there will be a configuration-set flag which makes the default condition accept or reject for events which wanted further calculation but did not get it. On Day 1 we will set this flag to Accept as the default. (We could make this depend on TV value if needed, since the decision is performed in the L1CTL processor.)

Rejecting Events:
What about the first question, what to do with events that fail to meet L1 (or L2 or L3) constraints? The original L0ps=10 for TV3 was presumably set by the physics committee assuming that the peripheral program would get some set number of events relative to the L0ps=1 triggers. When a TV3 event is passed to L1, it invokes a specific algorithm to run on the "coarse pixel" array and on the other DSM information resident in the L0/L1 crate. By definition of the TV3, the two hit events must have hits in opposite octants, and the 4 hit events must have both pairs meet the opposite octant condition.

What we have done is taken the TV3 events and broken them down further into two separate event types, meeting the new conditions or not meeting them. We only care about the rejected events because for this run they should be counted in a different trigger type. For simplicity, let's make the new conditions remove the degeneracy of TV3 by introducing two new trigger types. We can think of these as 3a and 3b, or we can simply map them into our original 64k trigger value space. This is a bookkeeping issue. We choose to map these onto two new words because it means we get to keep the formality of a single table, so we say these become TV300 and 301. That is, TV301 is really TV3 PLUS the L1 conditions OpOct.

TV	L1 Conditions	Description
3 -> 300	OpOct not met	
3 -> 301	2.le.m.le.5, Time = 1, OpOct.	require all events to have opposite octant hits

What do we do with the TV3 events that do not meet the criteria for TV301? We create another trigger type, TV300, for those events which began life as TV3 but were rejected for inclusion in the TV301 type. We can now introduce a separate prescale for TV300 events to match their sample size with the TV4 sample size by introducing a prescale whose product with L0ps(3) gives L0ps(4): in this case, L1ps(300)=2.

What about the case where the ratio of prescales is not an integer? For instance, say L0ps(3)=10 and L0ps(4)=25. How can we get L0ps(3)*L0ps(300)=25? We could do this in principal alternating between 2 and 3 for L0ps(300), since this is just a coding issue.

Level 1,2,3
Why do we go to this level of complexity for bookkeeping? The problem arose because of the imposition of the time condition to an L0 class that was otherwise a subset of another L0 trigger condition. L0 selection is expected to have significant degeneracy which can only

be removed at L1, L2, and L3. If we adopt the concept that each set of conditions leads to a new trigger type, we can keep all counting clear.

ADDING EMC:

What if we also want to include a subset of events that have some form of jet in the EMC, preserving the other types (TV1-TV5) in the mix for a run? Define a new trigger value, TV7, for events passing the condition "at least one jet". Define the jet however you want. Such a jet may be included in an event having any of the other trigger conditions unless we explicitly exclude "events possessing jets" from the other trigger definitions. To preserve the required orthogonality in the L0 trigger definitions we MUST include such an explicit exclusion. This is trivial in our FPGA approach to L0 event characterization, since we could first check for the jet condition and then ignore all others if it is found.

What if we set a prescale for the "jet events" other than 1? Since the "jet event" has ONLY the characterization that it posses a "jet", we do not need to check any other conditions to define the event once the jet has been discovered. The prescale is checked in the TCU after the event has been characterized. If MOST of the other events also have jets, then the trigger frequency will be dominated by this jet prescale value. If most do not have a jet, then the "normal" frequencies will be realized. However, the "normal" events will now have an explicit condition added to their definition, namely, that they DO NOT include a jet. Note that a prescale is not the same as an additional condition, such as the OpOct of the peripheral case, so there is no opportunity to recover events rejected by the prescale condition.

Appendix: Typical trigger (old)

We define the data trigger types to be BEAM, INTERACTION, and CENTRAL collisions, defined through the following logical conditions on detector signals:
BEAM=BC1(A+B).BC2(A+B)
INT = BEAM.MLT(Lo)
CENT= BEAM.MLT(hi)
Note that these are not mutually exclusive: CENT is a subset of both INT and BEAM, and INT is a subset of BEAM. We allow a separate prescale value for each, with PS(CENT)=1, PS(INT)=10, and PS(BEAM)=100 as an example. Each channel of prescaler is decremented each time the condition was encountered. Thus, on a CENT interaction, all three prescalers were decremented, while on an BEAM interaction, only the BEAM prescaler decremented. The trigger is formed as the OR of the prescale outputs: whenever any one went to 0 we get a trigger. Furthermore, each bit corresponding to a condition met by this event is set in the input register. It is straightforward to cut on the CENT bit to gather all of the CENT events for the experiment. This is viewed as a standard way to set up prescalers - namely, define a set of trigger types by placing conditions on various signals, and then prescale each trigger type independently to get the desired mix.

If we defined the trigger types exclusively as:
NONI = BEAM.MLTbar
PERIPH = BEAM.MLT(lo).MLT(hi)bar
CENT = BEAM.MLT(hi)
with BEAM = BC1(A+B).BC2(A+B),
then we would have three non-overlapping trigger types, and only 1 bit would be set for each interaction. You could still see all of the CENT events by cutting on the CENT bit, but

to see all interactions you would have to cut on either CENT or PERIPH being set. In the preceding setup, INT = PERIPH + CENT.

Appendix: unfolding cross sections

Say we apply the multiple prescale idea (non-orthogonal states). Assume we have 4 trigger types as:

bit 1	= BEAM	= BCs	PS=100
bit 2	= INT	= BEAM.MLTlo	PS= 50
bit 3	= CENT	= INT.MLThi	PS= 10
bit 4	= MAX	= CENT.MLTmax	PS= 1

Whenever any of the prescale conditions has decremented to 0, the OR of such conditions is used as the trigger. Thus events may be accepted from any single bit or any combinations of these bits. Now we want to get the sample size for INT events. We then go through the data set taking all events in which bit 2 is set. If we ignore the other bits and simply take the number of bit2 on events and multiply by 50 to get their "true" number, we would be incorrect. This is because we haven't considered how many come from CENT and MAX events which have a different prescale. How do we get the correct number of INT events sampled? We have to consider the set of $((1.2.\overline{3}.\overline{4})*50)+((1.2.3.\overline{4})*10)+(1.2.3.4)$, since the INT is a sub-condition on CENT and MAX. But what fraction of the $(1.2.3.\overline{4})$ events actually came from the bit2 prescale select, and not from the bit 3 prescale select? We don't know unless we also record which bit caused the event to be selected. If we use the event select bits as well as the event describe bits, then we might think we could get a good sample of INT from the events having the INT bit in the selected field. Again, what do we do with the events having more than bit 2 high? This shows that you can not simply cut on a given bit to get the correct sample size for an event type, but must consider all of the sets available of which the type is a subset.

What if we define the types to be orthogonal? thus,

bit 1	= BEAM	= BCs.$\overline{\mathrm{MLT}}$	PS=100
bit 2	= INT	= BCs.MLTlo.$\overline{\mathrm{hi}}$.$\overline{\mathrm{max}}$	PS= 50
bit 3	= CENT	= BCS.MLThi.$\overline{\mathrm{max}}$	PS= 10
bit 4	= MAX	= BCs.MLTmax	PS= 1

Again we can use the OR of the prescale decrement counters to act as the trigger, but this time only one bit can cause any event to be accepted. There is no ambiguity which condition set the accept. For the MAX type, it is irrelevant whether the lo and hi have fired, since they are both below the max setting. Now, again assuming that INT really should include CENT and MAX since they are also interactions, we would get the correct INT sample size by taking bit2*50+bit3*10+bit4. Since only one condition is met for each event, there is no question of double counting and the bookkeeping is quite simple.

**We need a discussion of Level1 and Level2 counters in this bookkeeping problem.

Appendix: The multiple parallel prescale option

Can we set the STAR trigger to allow an arbitray number of independent prescales to operate at the same time? We believe this is possible using DSM boards. We have shown that we can program a value into the FPGA, and we have shown that we can increment a value each crossing. We can then compare the incremented value with the stored value to see if a prescale has been met. We can do this on many individual input bits, probably more than 20 and perhaps 31, in parallel. We can then check to see if any of the incremented values is equal to its register value and set an output bit to indicate that at least one such value was. All of these bits and operations are done in parallel. Thus we can certainly tell when any individual input bit has met its prescale condition.

Can we also tell which of the possible input bits met the condition? This is straightforward to do, but the information is not involved in the Level 0 trigger. The fact that any one of the bits met the condition can cause a Level 0 trigger, but we will not know which bits were involved until Level 1. Presumably this is not a problem. This is easily accomplished for up to 31 bits per DSM board, and we can use more than 1 DSM board in parallel if we want to prescale more than 31 such bits. We can read these bits in the present DSM design by making the 31 bits act as input to another layer of DSM. Alternatively, we are pursuing the option of altering the DSM design to allow reading output as well as input bits for each triggered crossing.

Appendix: Open questions

Do we want to keep constant trigger values over the lifetime of STAR? This way we could cut on TV=2, for instance, and always know we are getting highest multiplicity central collisions. The same question can be asked if we use a bit-wise striping method - do we want to define bits for STAR′s lifetime? It is easier to keep constantly defined trigger values than bits, I believe.