Chris Perkins
04/26/2012

Implementation of QT Algorithm for STAR ZDC : Run 12 – Heavy Ion

QT Code Version: 0x6a
MCS File: qt32b_l0_v6_a.mcs

Description:

This algorithm compares the East and West Analog Sums to four different thresholds, compares the East and West Attenuated Analog Sums to two thresholds, and outputs the upper bits of the E1TAC and W1TAC signals if they are within some range.

This algorithm does **not** use the "Good Hit" requirements.

The upper 10 bits of the E1TAC signal (Daughter A, ch4) is passed on to L0 if:

TAC_MIN < E1TAC < TAC_MAX

Otherwise, 0x000 is passed on to L0 for E1TAC. An equivalent condition is required to pass on W1TAC (Daughter C, ch4).

Note that the lowest two bits from each TAC value have been dropped to make room for additional thresholds as compared to last year's algorithm.

The ESum, WSum, ESumA, and WSumA threshold bits have no requirements on their TAC signals; a threshold bit is '1' if the corresponding channel is greater than the corresponding threshold with no other requirements.

Inputs:

QT8A: ESum (ch2), ESumA (ch3), E1TAC (ch4)
QT8B: None
QT8C: WSum (ch18), WSumA (ch19), W1TAC (ch20)
QT8D: None

Registers (1 Set Per Daughter Card)**:**

Alg. Reg. 0 (Reg 13): Not Used
Alg. Reg. 1 (Reg 14): TAC_MIN
Alg. Reg. 2 (Reg 15): TAC_MAX
Alg. Reg. 3 (Reg 16): E/W Analog Sum Threshold 0
Alg. Reg. 4 (Reg 17): E/W Analog Sum Threshold 1
Alg. Reg. 5 (Reg 18): E/W Analog Sum Threshold 2
Alg. Reg. 6 (Reg 19): E/W Analog Sum Threshold 3
Alg. Reg. 7 (Reg 20): E/W Attenuated Analog Sum Threshold 4
Alg. Reg. 8 (Reg 21): E/W Attenuated Analog Sum Threshold 5
Reg. 11: Channel Mask

LUT:

Timing adjustments/pedestal subtraction for each PMT

Algorithm Latch: 1

L0 Output to DSM:

(0-9) : W1TAC (if within range) (Upper 10 bits)
(10-19) : E1TAC (if within range) (Upper 10 bits)
(20) : WSumA > Th4
(21) : WSumA > Th5
(22) : ESumA > Th4
(23) : ESumA > Th5
(24) : WSum > Th0
(25) : WSum > Th1
(26) : WSum > Th2
(27) : ESum > Th0
(28) : ESum > Th1
(29) : ESum > Th2
(30) : WSum > Th3
(31) : ESum > Th3

Actions :

Tick	QT8A	QT8B	QT8C	QT8D
1	Mask Channels / Latch Inputs	-	Mask Channels / Latch Inputs	-
2	TAC > R1 → **TAC_MIN_GOOD** TAC < R2 → **TAC_MAX_GOOD** ADCx > THy → **ADCxTHy**	-	TAC > R1 → **TAC_MIN_GOOD** TAC < R2 → **TAC_MAX_GOOD** ADCx > THy → **ADCxTHy**	-

3	ADCxTHy → **ADCxTHy_Del1**
	If(TAC_MIN_GOOD && TAC_MAX_GOOD)
	TAC → **TAC_OUT**
	Else
	0 → **TAC_OUT**
		-	ADCxTHy → **ADCxTHy_Del1**
			If(TAC_MIN_GOOD && TAC_MAX_GOOD)
			TAC → **TAC_OUT**
			Else
			0 → **TAC_OUT**
				-

4	**Latch out ADCxTHy_Del1** **Latch out TAC_OUT**	-	ADCxTHy_Del1 → **ADCxTHy_Del2** TAC_OUT → **TAC_OUT_Del1**	-
5	-	Latch In	ADCxTHy_Del2 → **ADCxTHy_Del3** TAC_OUT_Del1 → **TAC_OUT_Del2**	-
6	-	**Latch Out**	ADCxTHy_Del3 → **ADCxTHy_Del4** TAC_OUT_Del2 → **TAC_OUT_Del3**	-
7	-	-	Latch In $ADCxTHy_A$ Latch In TAC_OUT_A ADCxTHy_Del4 → **ADCxTHy_Del5** TAC_OUT_Del3 → **TAC_OUT_Del4**	-
8	-	-	**Latch out ADCxTHy_Del5** **Latch out TAC_OUT_Del4** **Latch out $ADCxTHy_A$** **Latch out TAC_OUT_A**	-
9	-	-	-	Latch In
10	-	-	-	**Latch out $ADCxTHy_A$** **Latch out TAC_OUT_A** **Latch out $ADCxTHy_C$** **Latch out TAC_OUT_C**