**AESOP-Lite DAQ Board**

**Event-PSOC Firmware Documentation**

## Channel Connections

Table 1. Calibration-pulse and PMT input connections, from left to right when looking down at the connector-end of the DAQ board. Note that the TOF chip SSN is 2. The SSN codes given here are bits 2,1,0. Bit 3 is always 1. The main PSOC, then, is addressed by 0000, and 1000 is “none”. Also note that Channel 5 (T2) has an external DAC and comparator, so the input to P2[0] is digital, whereas for the other channels it is analog. The four PSOC DACs are labeled by channel number. The guard signal, G, does not participate in the trigger logic but is discriminated and registered in the data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Detector | T3 | G | T4 | T1 | T2 |
| PMT Connector | J10 | J2 | J17 | J15 | J25 |
| Calibration Connector | J12 | J11 | J18 | J16 | J26 |
| PSOC pin for preamp output | P4[6] | P3[2] | P3[3] | P4[5] | P2[0] |
| Schematic PSOC pin for preamp output | 9 | 16 | 15 | 14 | 19 |
| Channel | 2 | 1 | 4 | 3 | 5 |
| Time-of-Flight channel |  |  | 2 (B) | 1 (A) |  |
| Trigger bit | 1 | N/A | 0 | 3 | 2 |
| External SAR SPI SSN | 3 | 1 | 6 | 7 | 4 |

## SPI

An SPI master is used to configure the TOF chip and to send data back to the Main PSOC (in principle it could also receive information, but so far that communication direction is handled by UART). Software pin Pin\_SSN\_Main is used to select the Main PSOC as the slave, by lowering to logic zero. Control register Control\_Reg\_SSN is used to select the TOF chip as well as to select which of the five SAR ADCs to read via a separate SPI channel. The codes are as follows:

|  |  |  |
| --- | --- | --- |
| **Control\_Reg\_SSN** | **Pin\_SSN\_Main** | **Device Selected** |
| 000 | 0 | Main PSOC |
| 001 | 1 | Guard ADC |
| 010 | 1 | TOF Chip |
| 011 | 1 | T3 ADC |
| 100 | 1 | T2 ADC |
| 101 | 1 | none |
| 110 | 1 | T4 ADC |
| 111 | 1 | T1 ADC |

The SPI mode is CPHA=1, CPOL=0, shift direction MSB-first, 8 data bits, and 1 Mbps bit rate.

## Triggers

There are three triggers defined in the Event-PSOC logic:

1. The main physics trigger, which should require a Cerenkov signal and is not prescaled. Most likely, it will be a coincidence of T1, T2, and T3. The coincidence level is defined by the control register Control\_Reg\_Trg1. The LSB is T4, the 2’s bit is T3, the 4’s bit is T2, and the 8’s bit is T1. If a bit is set to logic-high, then that counter is *removed* from the coincidence. Therefore, the setting 0x01 will implement the T1+T2+T3 trigger. Use the command 0x36 to modify the coincidence level.
2. The proton trigger, which should not require a Cerenkov signal, e.g. T1, T3, and T4. It is prescaled by counter Cntr8\_V1\_PMT, and the default count is the maximum of 255, meaning that only 1 in 255 of these triggers will be accepted. Use the command 0x39 to modify the prescaling. The coincidence level is defined by the control register Control\_Reg\_Trg2. The LSB is T4, the 2’s bit is T3, the 4’s bit is T2, and the 8’s bit is T1. If a bit is set to logic-high, then that counter is *removed* from the coincidence. Therefore, the setting 0x04 will implement the T1+T3+T4 trigger. Use the command 0x36 to modify the coincidence level.
3. The track trigger, which is an AND or an OR of the separate bending and non-bending trigger signals. It is prescaled by counter Cntr8\_V1\_TKR, and the default count is the maximum of 255, meaning that only 1 in 255 of these triggers will be accepted. Use the command 0x39 to modify the prescaling. Note that even though the two track triggers are OR’ed together here, they are registered separately in the status register Status\_Reg\_Trg, which is captured and read out for every event. The choice between OR versus AND of the two tracker views is set by the second bit of Control\_Reg\_Trg.

## Channel Gains

The PHA preamplifier gains are defined by feedback capacitors and are set lower for T1 and T4 than for T3 and G, because of the special high-gain PMTs used for time-of-flight in T1 and T4. The T2 gain is also set different, but the feedback resistor in each channel is chosen to make the decay time constant of the preamplifier output about the same for all channels. The peak detector output also has a non-inverting op-amp for each channel, with gains set differently per channel by the associated feedback resistors. Note that for the purpose of injecting signals via the test inputs, the capacitanc connecting the test input to the channel input is 36 pF in all cases.

Table 2. Gain settings of the PHA channels. The preamplifier gain is theoretical (just the inverse of the capacitance) and somewhat overestimated, whereas the peak detector output-amplifier gain is accurately set by 1% resistors. The measured overall gain was obtained by reading the average ADC results from a handful of events when stimulating the inputs via a voltage step on the 36 pF calibration capacitors.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Channel** | **C (pF)** | **R (kΩ)** | **(μs)** | **Preamp Gain (mV/pC)** | **PD Gain** | **Meas. Gain (mV/pC)** |
| T1 | 220 | 10.2 | 2.24 | 4.5 | 1.61 | 9.53 |
| T4 | 220 | 10.2 | 2.24 | 4.5 | 1.61 | 9.63 |
| T3 | 36 | 62 | 2.23 | 28 | 2 | 89.7 |
| G | 36 | 62 | 2.23 | 28 | 2 | 89.9 |
| T2 | 100 | 22.1 | 2.21 | 10 | 1.61 | 19.8 |

## SAR ADCs

Each PHA channel feeds its peak-detector output into an AD7091R 12-bit SAR ADC, and all of them digitize in parallel. Instead of using the internal 2.5 volt reference, we use a single external precision 3.3 volt reference (LT1460). Therefore, the LSB of the ADC output represents 0.81 mV. That can be converted to input charge by using the channel-dependent measured gain from Table 2.

## DACs

The Event PSOC has four internal 8-bit DACs, which are used to set the trigger thresholds for the four PMTs that read out scintillator signals. Each threshold applies to the output of the preamplifier stage of the PHA channel. The DACs are set to slow speed (low-power) mode, whereas the associated comparators are set to fast (high-power) mode, with no hysteresis. In addition, there are three external 12-bit DACs (AD5602) on the board, which are set via the I2C bus. One sets the trigger threshold for the Cerenkov (T2) channel, again applied to the preamplifier output, and the other two are used to set negative thresholds for the raw PMT signals (using op-amps to invert the voltage), to discriminate the T1 and T4 signals and generate stop signals for the TOF chip.

Table 3. Characteristics of the seven DACs used by the Event PSOC.

|  |  |  |  |
| --- | --- | --- | --- |
| **Channel** | **Address** | **Num. Bits** | **LSB (mV)** |
| T1 trigger | VDAC8\_Ch3 | 8 | 4 |
| T4 trigger | VDAC8\_Ch4 | 8 | 4 |
| T3 trigger | VDAC8\_Ch2 | 8 | 4 |
| Guard | VDAC8\_Ch1 | 8 | 4 |
| T2 trigger | I2C 0001110 | 12 | 0.806 |
| T1 TOF | I2C 0001100 | 12 | 0.806 |
| T4 TOF | I2C 0001111 | 12 | 0.806 |

## Event Output Format

The first 6 bytes consist of stuff that precede every type of record from the Event PSOC.

* 0xDC00FF
* 1-byte record length
* 0xDD (instead of the command echo), or 0xDB if TOF debug bytes are included
* 0x00 (zero length for the command-data echo)
* 0: 0x5A45524F (“ZERO” in ASCII)
* 4: 2-byte run number (MSBs first)
* 6: 4-byte event number (count of accepted triggers, i.e. GO signals)
* 10: 4-byte time stamp (LSB increments each 5 milliseconds)
* 14: 4-byte missed trigger count (GO1)
* 18: 4-byte time and date
  + Bits 26-31: the year minus 2000
  + Bits 22-25: month (1-12)
  + Bits 17-21: day of the month (1-31)
  + Bits 12-16: hour (0-23)
  + Bits 6-11: minute (0-59)
  + Bits 0-5: second (0-59)
* 22: 1-byte trigger status bits captured with the trigger
  + Bit 0: primary trigger (electrons)
  + Bit 1: secondary trigger (typically without T2)
  + Bit 2: tracker trigger 0
  + Bit 3: tracker trigger 1
  + Bit 4: T1
  + Bit 5: T2
  + Bit 6: T3
  + Bit 7: T4
* 23: 2-byte T1 ADC counts
* 25: 2-byte T2 ADC counts
* 27: 2-byte T3 ADC counts
* 29: 2-byte T4 ADC counts
* 31: 2-byte G ADC counts
* 33: 2-byte TOF time difference, in 10 ps units
* 35: 2-byte tracker trigger count
* 37: 1-byte tracker command count
* 38: 1-byte status register and tracker trigger capture
  + Bits 6 and 7: tracker trigger capture
  + Bit 5: StoreADC
  + Bit 4: GO
  + Bit 3: Done
  + Bit 2: TOF interrupt
  + Bit 1: tcB
  + Bit 0: tcA
* Optional 10 bytes for debugging the TOF (turn on with the 4th argument of the 0x3C start-run command)
  + 1-byte, number of TOF readouts since the last trigger in channel A
  + 1-byte, number of TOF readouts since the last trigger in channel B
  + 2-byte, channel A reference clock
  + 2-byte, channel B reference clock
  + 2-byte, channel A internal PSOC clock count
  + 2-byte, channel B internal PSOC clock count
* 39 or 49: 1-byte, number of tracker boards in the readout
* 40 or 50: Ordered sequence of tracker board data records of variable length
  + All installed boards are included unless the available space in the data record runs out
  + Individual boards come out with zero hits if
    - there really were no hits.
    - there was not enough space to write out the hits. These will have a bad CRC and an error bit set.
* 0x46494E49 (“FINI” in ASCII)
* Padding bytes (if needed) and 0xFF00FF trailer

## Housekeeping Output Formats

### Event-PSOC housekeeping

* 0xDC00FF
* 1-byte record length
* 0xDE (instead of the command echo)
* 0x00 (zero length for the command-data echo)
* 0: 0x48415553 (“HAUS” in ASCII)
* 4: 2-byte run number
* 6: 4-byte time and date (packed the same as in the event record)
* 10: 2-byte last command
* 12: 2-byte command count
* 14: 1-byte count of bad commands
* 15: 1-byte count of errors
* 16: 4-byte number of triggers (GO signals)
* 20: 4-byte number of missed triggers (GO1 signals)
* 24: 2-byte average readout time in microseconds
* 26: 2-byte T1 average singles rate
* 28: 2-byte T2 average singles rate
* 30: 2-byte T3 average singles rate
* 32: 2-byte T4 average singles rate
* 34: 2-byte Guard average singles rate
* 36: 2-byte most recent tracker command count
* 38: 1-byte percentage of events with tracker trigger bit 1 set
* 39: 1-byte percentage of events with tracker trigger bit 2 set
* 40: 1-byte number of tracker DAQ errors
* 41: 1-byte number of tracker DAQ timeouts
* 42: 1-byte average number of ASICs hit in board 0
* 43: 1-byte average number of ASICs hit in board 1
* 44: 1-byte average number of ASICs hit in board 2
* 45: 1-byte average number of ASICs hit in board 3
* 46: 1-byte average number of ASICs hit in board 4
* 47: 1-byte average number of ASICs hit in board 5
* 48: 1-byte average number of ASICs hit in board 6
* 49: 1-byte average number of ASICs hit in board 7
* 50: 2-byte channel-OR rate in tracker board 0, Hz
* 52: 2-byte channel-OR rate in tracker board 1, Hz
* 54: 2-byte channel-OR rate in tracker board 2, Hz
* 56: 2-byte channel-OR rate in tracker board 3, Hz
* 58: 2-byte channel-OR rate in tracker board 4, Hz
* 60: 2-byte channel-OR rate in tracker board 5, Hz
* 62: 2-byte channel-OR rate in tracker board 6, Hz
* 64: 2-byte channel-OR rate in tracker board 7, Hz
* 66: 2-byte PSOC die temperature, in degrees Celsius
* 68: 2-byte temperate of tracker board 0 (divide by 256 to get degrees Celsius)
* 70: 2-byte temperature of tracker board 7 (divide by 256 to get degrees Celsius)
* 72: average number of Channel-A TOF hits per event
* 73: average number of Channel-B TOF hits per event
* 74: maximum number of Channel-A TOF hits in one event
* 75: maximum number of Channel-B TOF hits in one event
* 76: percent Main-PSOC busy fraction
* 77: percent trigger live fraction
* 78: percent live fraction for ADC state machine
* 0xFF00FF trailer

### Tracker housekeeping

* 0xDC00FF
* 1-byte record length
* 0xDF (instead of the command echo)
* 0x00 (zero length for the command-data echo)
* 0: 0x5452414B (“TRAK” in ASCII)
* 4: 2-byte run number
* 6: 4-byte time and date (packed the same as in the event record)
* 10: 2-byte temperature of board 0 (divide by 256 to get degrees Celsius)
* 12: 2-byte SSD bias current of board 0 (multiply by 2.5/100 for micro-amps)
* 14: 2-byte board 0 digital 1.2-V regulator voltage (FPGA core; multiply by 0.00125 for volts)
* 16: 2-byte board 0 digital 1.2-V regulator current (multiply by 0.0025/0.03 for milliamps; typically 24 mA)
* 18: 2-byte board 0 digital 2.5-V regulator voltage (FPGA and other logic, including ASIC digital logic)
* 20: 2-byte board 0 digital 2.5-V regulator current (typically 440 mA)
* 22: 2-byte board 0 digital 3.3-V regulator voltage (i2c bus)
* 24: 2-byte board 0 digital 3.3-V regulator current (typically 44 mA)
* 26: 2-byte board 0 analog 2.1-V regulator voltage (amplifier input transistor)
* 28: 2-byte board 0 analog 2.1-V regulator current (typically 80 mA)
* 30: 2-byte board 0 analog 3.3-V regulator voltage (ASIC amplifiers)
* 32: 2-byte board 0 analog 3.3-V regulator current (
* Repeat for 7 more boards, for a total of 201 bytes
* 0xFF00FF trailer

The voltage and current conversions are the same for all five regulators. Only the bias-current conversion is different, as it uses a 100-ohm shunt resistor instead of the 0.03 ohms used for the low-voltage supplies.

## Begin of Run Record

The following packet of information goes out into the data stream whenever a run is started. There is the usual header, followed by 85 data bytes.

* 0xDC00FF
* 1-byte record length
* 0x3C
* 0x04
* 4 bytes of echo of data supplied with the start-of-run command
  + 2-byte run number (should match the run number below)
  + 0 or 1 for including tracker data (should normally be 1!)
  + 0 or 1 for including TOF data (normally 0)
* 0: 0x424F4652 (BOFR in ASCII)
* 4: 2-byte run number
* 6: 4-byte packed time and date
* 10: Event PSOC code major version number
* 11: Event PSOC code minor version number
* 12: 4 bytes of threshold-DAC settings for the 4 internal 8-bit DACs
* 16: 2-byte T2 threshold-DAC setting
* 18: 2-byte channel-1 TOF DAC threshold setting
* 20: 2-byte channel-2 TOF DAC threshold setting
* 22: TOF channel-1 Count7 counter period setting (should be 29)
* 23: TOF channel-2 Count7 counter period setting (should be 29)
* 24: ADC-control Count7 counter period setting
* 25: 4-bytes trigger window settings for channels 2 through 5
* 29: delay setting for the PMT trigger
* 30: PMT trigger prescale setting
* 31: TKR trigger prescale setting
* 32: main electron-trigger mask
* 33: prescaled PMT trigger mask
* 34: 8 bytes threshold bumps for each of the tracker boards
* 42: main trigger delay in tracker Verilog
* 43: tracker trigger source
* 44: tracker trigger logic setting
* 45: tracker board 0 FPGA code version
* 46: tracker board 0 FPGA configuration register
* 47: tracker board 0 number of layers in the readout
* 48: tracker board 0 trigger primitive output length
* 49: tracker board 0 trigger primitive output delay
* Repeat for the remaining 7 tracker boards

## End of Run Record

The following packet of information goes out into the data stream when the end-of-run command (0x44) is issued. There is the usual 6-byte header, followed by 146 data bytes.

* 0xDC00FF
* 1-byte record length
* 0x44
* 0x00
* 0: 0x454F52 (“EOR” in ASCII)
* 3: 2-byte run number
* 5: 4-byte missed-trigger count (GO1)
* 9: 4-byte accepted-trigger count (GO); equals the number of events acquired
* 13: 1-byte number of bad CRC codes (or 0 if CRC checking is not turned on)
* 14: 4-byte number of tracker read commands issued when the tracker says it is ready
* 18: 2-byte number of tracker read commands issued even though no ready confirmation was received
* 20: 1-byte average number of TOF stops per event in channel A
* 21: 1-byte average number of TOF stops per event in channel B
* 22: 1-byte maximum number of TOF stops per event in channel A
* 23: 1-byte maximum number of TOF stops per event in channel B
* 24: 4-byte count of events created when the SPI link is busy
* 28: 2-byte number of GO signals received by the tracker master board
* 30: 2-byte number of triggers received by tracker board 0
* 32: 2-byte number of read commands received by tracker board 0
* 34: 1-byte missed-trigger count from tracker board 0
* 35: 1-byte count of read commands received with no trigger, for board 0
* 36: 1-byte of error bits from board 0
* 37: 1-byte of error bits from ASIC #4 on board 0
* 38: 1-byte number of bad command addresses or command codes seen by board 0
* 39: 2-byte number of triggers received by tracker board 1
* 41: 2-byte number of read commands received by tracker board 1
* 43: 1-byte missed-trigger count from tracker board 1
* 44: 1-byte count of read commands received with no trigger, for board 1
* 45: 1-byte of error bits from board 1
* 46: 1-byte of error bits from ASIC #4 on board 1
* 47: 1-byte number of bad command addresses or command codes seen by board 1
* The same for boards 2 through 7
* 102: 2 bytes global command count
* 104: 2 bytes command count
* 106: number of time-outs by the command interpreter
* 107: number of tracker resets
* 108: number of ASIC error events
* 109: number of ASIC parity errors
* 110: number of bad ASIC headers
* 111: number of bad clusters
* 112: number of bad commands
* 113: number of oversized clusters
* 114: number of tracker overflows
* 115: number of tracker trigger tag mismatches
* 116: number of oversized, truncated events
* 117: number of errors incurred while unpacking tracker data
* 118: number of tracker events that fail the formal check on the data packet length
* 119: 2 bytes number of tracker time-outs
* 121: 4 bytes number of tracker triggers on view 1
* 125: 4 bytes number of tracker triggers on view 2
* 129: 4 bytes number of PMT-only triggers
* 133: 4 bytes number of tracker-only triggers
* 137: 4 bytes number of events with all triggers fired
* 141: 4 bytes number of events with no Cerenkov in the trigger
* 145: 2 bytes 100 times the percent live time of the ADC state machine
* 147: 2 bytes number of NOOP commands received

## Command Format

* Each command is formatted as "S1234<sp>xyW" repeated 3 times, followed by <cr><lf>
  + where 1234 are 4 ASCII characters, each representing a nibble. Only the ASCII characters for the hex numbers 0 through F are allowed. Either case is allowed for A through F. All other ASCII codes will be translated to zero.
  + 12 gives us the data byte and 34 the address byte
* data byte: {7:0} gives the command code
* address byte: {7:6} and {1:0] give the number of data-byte "commands" to follow, 0 to 15
* address byte {5:2} = 0x8 indicate the event PSOC
* All data arrive in up to 15 subsequent data-byte "commands" For those,
  + bits {7:0} of the data byte are the data for the command in progress
  + bits {7:6} and {1:0} of the address byte give the data-byte number, 1 through 15
  + bits {5:2} of the address byte must match, as usual, the PSOC address of 0x8.
* Subsequent commands must wait until after the correct number of data bytes has arrived

## Command Codes

The codes in the following table are all implemented in a single switch statement within the interpretCommand routine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **# data bytes out** | **Command Data bytes** |
| 0x01 | Load a PHA threshold DAC  CH 1-4: 8 bit DAC internal to the PSOC  CH 5: 12 bit external DAC | 0 | 0: PHA channel  1: High-order byte  2: Low-order byte (CH 5 only) |
| 0x02 | Read a PHA threshold DAC | 2 | 0: PHA channel |
| 0x03 | Read accumulated error codes | Variable | N/A |
| 0x04 | Load a 12-bit TOF threshold DAC | 0 | 0: TOF channel number  1: High-order byte  2: Low-order byte |
| 0x05 | Read a 12-bit TOF DAC | 2 | 0: TOF channel number |
| 0x06 | Turn red LED-2 on or off | 0 | 0: 1 for on, 0 for off |
| 0x07 | Return the code version number | 1 | N/A |
| 0x10 | Send arbitrary command to the tracker | Variable | 0: FPGA (0-7)  1: Tracker command code  2: Number of data bytes to follow  3 & up: Data bytes |
| 0x0C | Reset the TOF chip | 0 | N/A |
| 0x0D | Modify the TOF chip configuration | 0 | 0: which byte to modify  1: new value for the selected byte |
| 0x0E | Read the TOF chip configuration | 17 | N/A |
| 0x20 | Read bus voltages from INA226 chips, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the INA226 chips can be accessed by only the Main PSOC.* | 2 | 0: I2C address of the INA226 chip |
| 0x21 | Read current-shunt-voltages from INA226 chips, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the INA226 chips can be accessed by only the Main PSOC.* | 2 | 0: I2C address of the INA226 chip |
| 0x22 | Read the board temperature from the TMP100 chip. *This has been disabled by separating the I2C bus on the DAQ board, so that the TMP100 chip can be accessed by only the Main PSOC.* | 2 | N/A |
| 0x23 | Read a register from the Real-Time-Clock chip, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the RTC chip can be accessed by only the Main PSOC.* | 1 | 0: register number to read |
| 0x24 | Write a register in the Real-Time-Clock chip, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the RTC chip can be accessed by only the Main PSOC.* | 0 | 0: register number to write  1: register value to write |
| 0x26 | Read a leak-detector barometer register, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the leak-detector barometer can be accessed by only the Main PSOC.* | 1 | 0: register number to read |
| 0x27 | Write a leak-detector barometer register, via the I2C bus. *This has been disabled by separating the I2C bus on the DAQ board, so that the leak-detector barometer can be accessed by only the Main PSOC.* | 0 | 0: register number to write  1: register value to write |
| 0x30 | Set the output mode (default is SPI). The USB-UART output is for bench testing of the Event-PSOC alone, for which case this needs to be the first command issued. | 0 | 0: 0 for SPI, 1 for USB-UART |
| 0x31 | Initialize the SPI interface (normally not needed, as it is done at start-up). | 0 | N/A |
| 0x32 | Initiate sending of TOF data directly to the USB-UART output. This is for benchtop debugging work only! | 0 | N/A |
| 0x34 | Return the number of TOF events currently stored. This is for benchtop debugging work only! | 2 | N/A |
| 0x35 | Read the most recent TOF event from channel A or B. This is for benchtop debugging work only! | 9 | 0: 0 for A, 1 for B |
| 0x36 | Set a trigger mask for a PMT-based trigger | 0 | 0: 1 for the main electron trigger,  0 for the proton trigger  1: mask setting |
| 0x37 | Read a channel counter  The 1st 4 bytes are the software count.  The 5th byte is the hardware count.  Add together for the total count. | 5 | 0: channel number 1 through 5 |
| 0x33 | Read a saved channel counter, after EOR  The 1st 4 bytes are the software count.  The 5th byte is the hardware count.  Add together for the total count. | 5 | 0: channel number 1 through 5 |
| 0x38 | Reset the logic and counters  3-byte clock count is returned | 3 | N/A |
| 0x39 | Set trigger prescales | 0 | 0: 1 for tracker, 0 for PMT proton trig.  1: prescale value |
| 0x3A | Set the PMT signal settling time | 0 | 0: number of 24 MHz clock cycles |
| 0x3B | Enable or disable the trigger in both the PSOC and the tracker | 0 | 0: 1 to enable, 0 to disable |
| 0x3C | Start a run | 0 | 0,1: two-byte run number  2: 1=read tracker; 0=no tracker  3: 1=TOF debugging on |
| 0x3D | Return trigger enable status  1= enabled; 0= not enabled | 1 | N/A |
| 0x3E | Return a trigger mask register for a PMT-based trigger | 1 | 0: 1 for the main electron trigger,  0 for the proton trigger |
| 0x3F | Stop sending TOF data directly to the USB-UART output. This is for benchtop debugging work only! | 0 | N/A |
| 0x40 | Read all TOF data (for bench testing) | variable | N/A |
| 0x41 | Load a Tracker ASIC mask register.  *For the trigger and data masks, this should always be used instead of command 0x10,* so that the settings will get retained in memory, as needed in case there is an automatic reset and reconfiguration.  For all of the masks this command is more useful than the direct 0x10 command, as it allows just selected channels to be specified. | TKR echo | 0: FPGA (0-7)  1: Chip address (0-11)  2: Register type:  1 = calibration mask  2 = data mask  3 = trigger mask  3: Option  0 = mask selected channels off  1 = set selected channels on  4: Number of 2-byte items to follow:  5 and up: for each item  First byte = starting channel  Second byte = # of channels |
| 0x42 | Start a Tracker calibration sequence (pulses the channels selected by the calibration mask, using a pulse height set for the calibration DAC) |  | 0: FPGA (0-7)  1: Trigger delay  2: Trigger tag |
| 0x43 | Read a calibration event | Variable | 0: Trigger tag (must match 0x42 tag) |
| 0x44 | **End a run** and send the run summary.  See the output format above. | 84 | N/A |
| 0x45 | Set the real-time-clock. Normally this should be used by the Main PSOC to set the clock to match the time of the i2c RTC that has backup power. | 0 | 0: seconds  1: minutes  2: hours  3: day-of-week  4: day-of-month  5,6: day-of-year  7: month  8,9: year |
| 0x46 | Get the time and date from the RTC  0: Seconds  1: Minutes  2: Hours  3: Day of the week  4: Day of the month  5,6: Day of the year  7: Month  8,9: Year | 10 | N/A |
| 0x47 | Reset the Tracker state machines & ASICs  ASICs are reset only if they are showing DAQ errors in their configuration registers. | 0 | N/A |
| 0x48 | Calibrate tracker FPGA input timing |  | 0: FPGA address, or 8 for all layers |
| 0x49 | Read accumulated tracker layer rates  0: 0x6D  1: 8 (for 8 tracker boards, default)  2,3: uint16 rate of first board  4,5: uint16 rate of the second board  Etc. | 18 | N/A |
| 0x4B | Set the delay between the trigger GO and readout of the peak detectors | 0 | 0: delay, in 24 MHz clock cycles, from 0 to 127 |
| 0x4C | Enable or disable TOF data accumulation | 0 | 0: 1=enable, 0=disable |
| 0x4D | Select TOF acquisition mode | 0 | 0: 1=use DMA (preferred default)  0=use interrupt |
| 0x4E | Select whether or not to verify the tracker CRC in the Event PSOC. Turning this on will increase somewhat the dead time. | 0 | 0: 1=yes, 0=no (default) |
| 0x4F | Set the tracker trigger delay for PMT triggers. | 0 | 0: value, in units of the 12 MHz clock period |
| 0x50 | Return counter information  2 bytes count of all commands seen  2 bytes count of PSOC commands received  1 byte count of command timeouts  1 byte count of tracker resets  1 byte count of events with ASIC error flag  1 byte count of events with ASIC parity err | 8 | N/A |
| 0x51 | Get the average event readout rate  4-byte sum of readout times (5 ms units)  4-byte number of readouts  Take the ratio to get the average, but keep in mind that many readouts require less than the 5 ms quantization. | 8 | N/A |
| 0x53 | Get the PMT counter rates  0,1: uint16 monitoring time in 5ms units  2,3: uint16 counts for the 1st channel  4,5: uint16 counts for the 2nd channel  Etc.  Divide counts by time for the rate. | 12 | N/A |
| 0x54 | Load ASIC configuration registers.  *This should be used instead of command 0x10,* so that the settings will get retained in memory, as needed in case there is an automatic reset and reconfiguration. | 0 | 0: first byte (MSBs)  1: second byte  2: third byte (LSBs) |
| 0x55 | Load ASIC threshold DACs.  *This should be used instead of command 0x10,* so that the settings will get retained in memory, as needed in case there is an automatic reset and reconfiguration. | 0 | 0: FPGA address (0 through 7)  1: ASIC address or 0x1F for all  2: threshold setting |
| 0x56 | Set up the tracker ASIC configuration.  *This will first set the number of boards to be read out and turn on the ASIC power.*  The ASIC register values are read from RAM (which is initialized at power-on from EEPROM but can be modified by various commands). | 0 | 0: number of boards (1 through 8) |
| 0x57 | Start monitoring and sending housekeeping packets. Note that this also turns on the tracker and PMT rate monitoring with the supplied period. No averaging is done with the tracker rate, but that can be changed with command 0x4A. | 0 | 0: period in seconds between packets |
| 0x58 | Stop sending housekeeping packets. This also stops the rate monitors. | 0 | N/A |
| 0x59 | Change the tracker layer configuration. The boards are labeled by letters A through I, but here they are numbered 0 through 8. | 0 | 0: board in layer 0 (master)  1: board in layer 1  Etc. |
| 0x5A | Get the tracker layer configuration | 8 | N/A |
| 0x5B | Add a given amount to each and every tracker threshold. This changes only the value in RAM, so do this before issuing command 0x56. | 0 | 0: value in DAC counts  If only one data byte is supplied, then all 8 layers are set to that. If 8 data bytes are supplied, then they are mapped onto individual layers. |
| 0x5C | Start sending tracker housekeeping packets (temperature, voltage, current) | 0 | 0: period in minutes between sending of packets. |
| 0x5D | Stop sending tracker housekeeping packets | 0 | N/A |
| 0x5E | Send a housekeeping packet *now* | 0 | N/A |
| 0x5F | Send a tracker housekeeping packet *now* | 0 | N/A |
| 0x60 | Change the multiplier for the tracker rate monitoring and temperature update relative to the housekeeping period. |  | 0: multiplier (default = 4) |
| 0x61 | Return all of the ASIC error codes from the configuration registers. Each board requires 3 bytes to hold the 12\*3=36 bits. | 24 | N/A |
| 0x62 | Get the trigger prescale value | 1 | 0: 1 for tracker, 0 for PMT |
| 0x63 | Set the tracker trigger to take OR versus AND of the two views. | 0 | 0: 0 for AND and 1 for OR |
| 0x64 | Get the tracker logic setting established by command 0x63. It returns 0 for AND or 2 for OR. | 1 | N/A |

## DAQ Timing

The timing sequences programmed into the PSOC firmware goes as follows:

1. **Trigger window:** in the Discriminator page of the schematic there is a component named TrigWindow\_V1, used for all of T1, T2, T3, and T4 (the Guard uses a simpler device, see below). The component contains a schematic with two components: a custom Verilog component named SignalCrop\_v1 and a 7-bit hardware counter Count7. The Verilog contains a 4-bit counter that determines how long the output will remain high when a rising edge is detected from the comparator that looks at the PHA channel preamp output. A parameter in the symbol currently sets it at 14 clock counts with a 12 MHz clock, i.e. 1.17 microseconds. That determines the *trigger window*, because the trigger logic acts on the outputs, taking the AND of the 1.17 microsecond long pulses. Since it is a 4-bit counter, this window can be increased a bit, up to 1.25 μs by changing the parameter in the SignalCrop\_v1 component. Any more than that would require increasing the size of the counter (or lowering the clock frequency). That can only be changed by modifying the schematic and rebuilding.
2. **Preamp settling time:** after the end of the trigger window the Verilog waits for the comparator output to go low. It then starts the 7-bit hardware counter. The counter is started from a value set in software in the C code by the function *setSettlingWindow*, which is called by the command 0x3A. The default is set to 32 in the schematic but is overridden in the C code by an initial value of 36. When it has counted down to zero (a delay of 2 microseconds), then the Verilog code goes back to the state of looking for another rising edge. Thus the channel deadtime will last for the duration of the time-over-threshold of the preamplifier output plus another 2 microseconds (for the default). Only after that will the channel be ready to sense a new rising edge. The 2 microseconds is there to avoid retriggering on a wiggle back above threshold that might follow the preamplifier signal (especially with hysteresis set to zero for the comparators).
3. **Guard signal processing:** the guard channel is never included in the trigger, so a simpler component, SignalPrep\_v2, is used as an edge detector for it, to save on logic usage. That component does not establish a trigger window.
4. **Singles counting:** each channel includes an 8-bit counter following the edge detector, component Cntr8\_v1. The counters count down from 255, and when they hit zero they send an interrupt to the CPU, which then increments its 32-bit count in memory. When the rate count is read from the Event PSOC, it returns the count in memory times 255 (not 256!) plus the current count in the Cntr8.
5. **PHA timing:** in hardware this is independent of the trigger timing but needs to accommodate it. In the ADC page of the schematic, the SAR\_ADC\_CTRL Verilog component starts acting when any of the channels goes above threshold. There is a 7-bit fixed-function hardware down-counter in the ADC schematic page that is set in software to start from 25 but may be modified easily via command 0x4B (function *setPeakDetResetWait*). Once started, the Verilog waits for a GO signal until the Count7 counts down (1.04 microseconds for the 25 setting and 24 MHz clock). In principle it should wait for at least as long as the trigger window (plus a small logic latency), to be sure never to miss a coincidence for the GO. If it doesn't get a GO, then it waits for the OR of the channels to go back down plus another count-down of the 7-bit counter, while holding high the reset signal for the peak detectors, before it starts looking for the OR of channels to go high again. If it does get a GO, then it sends a signal to the ADCs to digitize. It then waits for the 7-bit counter to count down to zero again, and then it sends a signal to the CPU that the digitization is done. After that it waits for the OR of channels to go low and then holds the reset high on the peak detectors for a full down-count of the 7-bit counter. The CPU initiates the readout of the ADCs after it has received the “done” signal. Note that this Verilog component uses the 7-bit hardware counter for multiple purposes, in order to minimize the number of flip-flops needed (since the PSOC does not have large number of them, unlike an FPGA). It needs to count long enough to cover the trigger window, to give the ADCs enough time to digitize, and to give the peak detectors enough time to drain their charge through the MOSFET switch.
6. **Deadtime:** there are several contributors to the trigger deadtime. First, the individual channels do not recover immediately if they go above threshold, because the preamp must first decay back below threshold, then then the counter has to tick off the settling time, as explained above, before the channel can accept another rising edge. Second, every time any channel goes above threshold, if there is no trigger “GO”, then the system waits for at minimum twice the *setPeakDetResetWait* time described above, i.e. first looking for the GO and second to reset the peak detectors (or more if the preamp takes longer to decay below threshold). That is at minimum a total of 2.6 microseconds, which may not be acceptable if the channel singles rate is too high, in which case the delay setting can be reduced (or the thresholds raised). Third, if there is a trigger “GO”, then the trigger gets disabled for the duration of the event readout. First, the ADCs are read out, the Tracker is readout out, the TOF is read out, and then all the data are formatted and shipped off to the main PSOC. This takes much longer than the above mentioned hardware deadtime components, but it happens only at the trigger rate, not the OR of the singles rates. It is easy to monitor the readout contribution to the deadtime, because we count the triggers that arrive and not accepted pending the readout. One can estimate the other contributions from the singles rates that we monitor for each PMT channel. Or better, the code samples at random times while the trigger is enabled the status of the GO-enable signal to calculate the percentage of time that the ADC-control state machine is not ready to receive a trigger.
7. **Trigger Status Capture:** Status\_Reg\_Trg captures the status of the various triggers and trigger inputs each time there is a “GO”. The status is held by SR flip flops, one for each PMT trigger, one for each Tracker view, and one for the guard, and one for each of channels T1, T2, T3, and T4. The flip flops get reset every time there is a rising edge of the OR from the five comparators. After that, any signal above threshold will set them. An SR flip flop set by the GO signal prevents another channel-OR signal from resetting them before the readout is accomplished. That flip flop gets cleared at the end of the readout. The CPU reads the status register in the interrupt service routine of the “GO” signal, which has the highest priority of all interrupts.

## Triggers and Trigger Timing

There are three triggers in the logic:

1. The main physics trigger, which can be a coincidence of any subset of T1, T2, T3, T4, with no prescaling. It is foreseen to be in flight a T1,T2,T3 coincidence.
2. The secondary PMT trigger, which again can be a coincidence of any subset of T1, T2, T3, T4, but with prescaling by up to 1 in 255. It is foreseen to be a trigger for hadrons and muons, i.e. with no T2 signal required.
3. The tracker trigger, which is completely independent of the PMT triggers and, therefore, can be used to study trigger efficiency. It also can be prescaled by up to 1 in 255. The tracker trigger normally is an OR of the bending-plane coincidence and the non-bending-plane coincidence, but there is an option to change it to AND (see command ‘\x63’). The coincidence levels of the two sides are set by the 0x62 tracker command. By default it is a triple coincidence of the top three layers on the bending side or all three layers on the non-bending side.

The prescales are set by configuration of the period of two 8-bit counters: Cntr8\_V1\_PMT and Cntr8\_V1\_TKR. Use the command 0x39 to set them and 0x62 to read them back.

The coincidence levels are set by the control registers Control\_Reg\_Trg1 and Control\_Reg\_Trg2. A ‘1’ bit in the register is OR’ed with the corresponding PMT logic signal, thereby taking that channel out of the coincidence. Use the command 0x36 to set them. The bit assignment is as follows:

* 1’s bit: T4
* 2’s bit: T3
* 4’s bit: T2
* 8’s bit: T1

For example, T1&T2&T3=0x01 and T1&T3&T4=0x04.

The timing for capturing tracker hits has to be different for the PMT versus tracker triggers, because the rise time of the tracker amplifiers is much slower than that of the PMT amplifiers on the DAQ board. The 7-bit down-counter Count7\_Trg sets a delay for the PMT triggers. Use the command 0x4F to set the period of this counter. The default setting is 12 counts of the 12 MHz clock period, corresponding to one microsecond.

The delay of the tracker trigger is set in the FPGA Verilog code, in units of the 10 MHz clock period, i.e. 100 ns. This delay applies to both tracker and PMT triggers, so the total delay of the PMT triggers is the sum of this and the Count7\_Trg period setting. Use the tracker command 0x06 to set the delay in the Verilog. Its default setting is 1.

Note that the settings of the FIFO buffer at the output of the discriminator in each tracker channel are crucial to the trigger timing. These are set in the ASIC configuration register:

* The Trigger Window, bit 12, should be set to ‘1’ to increase the window from two periods of the FIFO clock to three. This makes the timing less critical and allows for more jitter.
* The Buffer Speed setting, bits 4 through 6, determines the period of the FIFO clock. We have always run it, for AESOP-Lite, with a setting of 3. This multiplies the 10 MHz tracker clock by 4 to arrive at a FIFO clock period of 400 ns. That means that the trigger window is 1.2 microseconds. Clearly any change to this parameter would require changing the trigger timing settings.
* The Trigger Delay setting, bits 7 through 11, directly impacts the trigger timing. It sets the difference between the write and read pointers in the FIFO. Our default setting is 4. This could be used to adjust the trigger timing, but it is a course setting, with 400 ns change per click. The Count7\_Trg setting, with an 83 ns adjustment, and the Verilog delay setting, with a 100 ns adjustment, should be used instead, leaving alone the FIFO Trigger Delay setting.

The shaping time setting, bit 3 in the configuration register, also affects the trigger timing, of course. We have been using the slow-amplifier setting. We chose that for the first flight, to try to accommodate the very slow PMT electronics in the old system. That is not relevant for the new PMT electronics, but so far we have stuck with it. If it were changed to the fast setting, then the trigger delays would have to be reduced accordingly.

The trigger operates in a 1-shot mode in which the trigger is disabled immediately in hardware after a trigger is accepted, by setting an SR flip flop in the ADC page. It is not re-enabled until the entire event has been sent out via the SPI link to the Main PSOC. Trigger coincidences that arrive during this “dead” time are counted by the GO1 interrupt. The control register Control\_Reg\_Trg is also used to enable and disable the trigger by CPU control, and the tracker commands 0x65 and 0x66 do the same for the trigger logic in the tracker boards. The ISR for the GO signal disables the trigger by way of this control register, and it gets re-enabled only after the event has been sent out, at the same time that the SR flip flop gets reset by a pulse from the Control\_Reg\_Pls 4’s bit.

## Time of Flight

The time difference is calculated in units of 10 ps as (channel T4 minus channel T1). Each of the two “stop times” is measured independently from the other and also independently from the instrument trigger. The times correspond to threshold crossings of fast discriminators that view the raw PMT signals prior to amplification by the preamplifiers of the analog PHA chain. A 51-ohm resistor separates the point where the TOF voltage is measured from the preamplifier input, so the negative-going signal voltage is given by the PMT current times that resistance (plus the frequency-dependent input impedance of the non-ideal charge amplifier). The threshold voltage is set independently for each channel by 12-bit DACs (and inverted by op-amps to be negative).

The specialized 2-channel timing chip (AS6501\_DS000586) has to be set up prior to the run by loading its volatile configuration register from the Event PSOC via SPI. The SPI is not used to read out the resulting data (although that is an option for this chip). Instead, the data flow out of the chip on LVDS lines at 24 million bits per second and are captured in PSOC 32-bit shift registers. There are two options for reading the shift registers. The first, which was also initial development, interrupts the CPU for every stop event of either channel, after which the interrupt service routine reads the register contents into a large FIFO buffer. The second sends each stop event by DMA to an intermediate buffer than can hold up to about 24 events (limited by the number of DMA “transaction descriptors” available). When the intermediate buffer is full, then the CPU gets interrupted to transfer the buffer to the large FIFO. In both cases, the PSOC clock is captured, to 5 ms resolution, along with the TOF chip time data. The DMA method should be preferable, as it results in fewer CPU interrupts, which in excess could bog down the CPU, depending on the noise rate.

For each channel, the TOF chip delivers 32 bits of data for each stop signal. The lower 16 bits are the stop time, and the upper 16 bits are the reference clock. The reference clock simply counts the 12 MHz clock that is sent to the TOF chip from the Event PSOC via port P15[5]. Each count thus represents a time of 83.33 ns, and it rolls over in a time of about 5.46 ms. Instead of allowing it to roll over, the PSOC resets the reference clock every 5 ms, using a hardware signal derived from the edge of a 200 Hz clock generated within the PSOC and sent to the TOF chip via port P15[4]. The 200 Hz clock is counted by an 8-bit hardware counter that interrupts the CPU once every second. The count of those interrupts is added to the 8-bit hardware count to give the 32-bit clock count used within the C code. The lower 16 bits of that count are captured along with the TOF stop time and reference clock. The three 16-bit numbers put together give an absolute time measurement down to the 10 picosecond level.

The TOF chip is set up such that each count of the stop time represents 10 picoseconds. Therefore, the time is calculated from the stop time plus the reference clock multiplied by 8333, to give a result in units of picoseconds. That could then be directly added to the captured 16-bit clock count, multiplied by , to give a time that rolls over only about every 5.5 hours (but doing so requires a 64-bit integer).