LBNE SiPM Signal Processor User Manual

J. T. Anderson, P. De Lurgio, Z. Djurcic,
G. Drake, A. Kreps, M. Oberling

Argonne National Laboratory

May 7, 2015
Version 2.05

Table of Contents

Revision Summary

Date	Revision	Changes
10/14	2.04	Initial Release
11/14	2.05	(Internal revision, not released.) Updated section 14.1 Charge Injection Control. Updated section 14.1.1 Charge Injection Voltage Control (DAC Control): Output disable modes described. qi_dac_load bit described. Added new section 14.1.2 Charge Injection Trigger Control. Updated section 14.1.2.1 Switch Polarity. Updated section 14.1.2.2 Trigger Input Control. Added new section 14.1.2.2.2 Timestamp Trigger. Updated section 14.1.2.2.3 Manual/Pulsed control input. Updated section 14.1.2.3 Pulse Duration. Added new section 14.1.2.4 Pulse Delay. Removed section 14.1.5 Frequency. Now described in section 14.1.2.2.2. Removed section 14.2.2 Switch Polarity. Already described in section 14.1.2.1. Removed section 14.2.3 Pulse Duration. Already described in section 14.1.2.3. Fixed in-document links throughout.
3/15	2.05	Updated table of contents: Appendices added. Updated section 1 Introduction. Updated section 2.1 Front Panel Description: Ethernet LEDs colors/descriptions updated. FIFO Status LED colors/description updated. Trig/Sync LED colors/description updated. Channel Act/Enable LED description updated. Clock input description updated. Updated input frequency rang JTAG port description updated. GPIO port description updated. Added pinout table to section 2.1.3 Trigger Output Port. Added pinout table to section 2.1.4 GPIO Port. Updated termination specification in section 2.1.4.2 Update figure of back panel in section 2.2 Back Panel. Updated section 2.2.3 NOvA Interface Port. Added Part number for Rev A SSP Update pinout figure and table. Updated section 3 Communication Interfaces. Updated section 3.1 USB 2.0. Updated section 3.2 10/100/1000 Mbps Ethernet.

		Removed section 3.3 Console Port. Updated section 4 System Timing. Updated frequency specification in section 4.1.1.1 NOvA Clock Input. Updated frequency specification in section 4.1.1.2 Front Panel Clock Input. Update section 4.2 Timestamp Generator Updated Figure 4-6: Timestamp generator block diagram. Updated section 4.2.1 Local Timestamp. Updated section 4.2.1.1 Timestamp Flag Generator. Replaced section 4.2.2 External Timestamp. Added new section 4.2.2.1 Front Panel Based Timestamp Added new section 4.2.2.2 NOvA Based Timestamp Update section 4.3.3 External Clock with Synchronized Timestamping. Update section 4.3.4 NOvA Timing. Added new section 6.1.1 NOvA Synchronous Start. Fixed text in Figure 7-2. Updated section 7.1.1.2 Timestamp Trigger. Corrected caption for Figure 8-3. Fixed caption for Table 9-1. Updated Table 11-2. Updated Table 11-4. Updated section 11.3 Event Buffer. Updated section 13.1 Bias Control. Updated section 13.2 Bias Monitoring. Updated section 15.2.1 Bias Removed section 15.2.1.1 Simple Bias Measurement Calibration Method. Updated Appendix A – Slow Control Packet Structure. Removed Appendix B – Host Data Structures. Removed Appendix C – Register Map. Provided as a separate document. Added new Appendix B – Flash Memory.

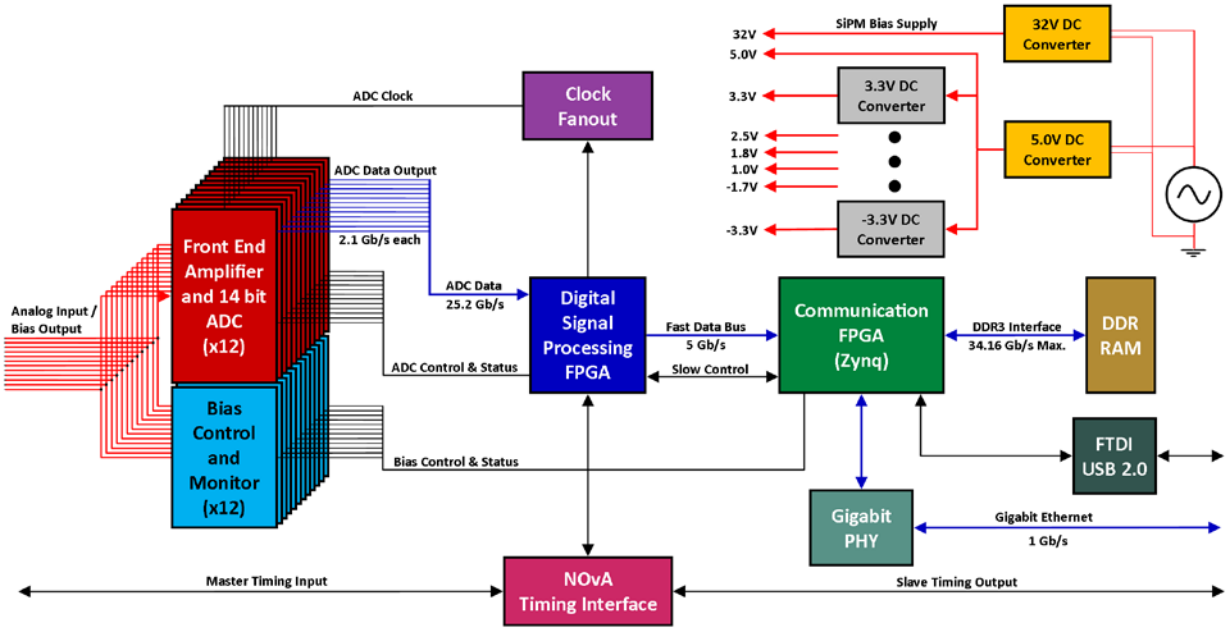
1. Introduction

The Silicon Photo-multiplier (SiPM) Signal Processor, or SSP, is an instrumentation module designed by the Electronics Group of the High Energy Physics division at Argonne National Laboratory to support the development of the photon detectors (PD) for the Long-Baseline Neutrino Experiment (LBNE) [1]. This document is a User Manual for the SSP.

An SSP consists of 12 readout channels packaged in a self-contained 1U module. Each channel contains a fully-differential voltage amplifier and a 14-bit, 150 MSPS analog-to-digital converter (ADC) that digitizes the waveforms received from the SiPMs. The digitized data is then processed by a Xilinx Artix-7 Field-Programmable Gate Array (FPGA) [2]. The FPGA implements an independent Data Processor (DP) for each channel. The processing incorporates a leading edge discriminator for detecting events and a constant fraction discriminator (CFD) for sub clock timing resolution.

In the simplest mode of operation, the module can perform waveform capture, using either an internal trigger or an external trigger. Up to 2046 waveform samples may be read out for each event. When waveform readouts overlap the device can be configured to offset, truncate or completely suppress the overlapping waveform. Pile-up events can also be suppressed.

As an alternative to reading full waveforms, the Data Processors can be configured to perform a wide variety of data processing algorithms, including several techniques for measuring amplitude, and also timing of the event with respect to a reference clock. All timing and amplitude values are reported in a compact event record.



The SSP can be configured to trigger readout in several ways, including self-triggered, use of an external trigger, or use an external gate to readout all events within a time-window. The digitization clock and timestamp across multiple SSP can be synchronized using the timing system developed for the NOvA experiment (http://www-nova.fnal.gov) or other external clock source. A Xilinx Zynq FPGA, onboard the MicroZed system-on-module, handles the slow control and event data transfer. The SSP has two parallel communication interfaces; USB 2.0 and 10/100/1000 Ethernet. The 1 Gbps Ethernet supports full TCP/IP. The module includes a separate 12-bit high-voltage DAC for each channel to provide up to 30 V of bias to each SiPM. The module also feature charge injection for performing diagnostics and linearity monitoring, and also voltage monitoring.

This document is a User Manual for the SSP. It is primarily intended to be used by software professionals, who are writing software to read out data, set up triggering and calibration runs, and general control and monitoring of the unit. Scientists may also find this document useful in understanding the operation and the features, although the verbiage and description are generally technical in nature. The interested reader is referred to [3-4] for additional description of the module.

2. Front Panel & Back Panel Overview

2.1 Front Panel Description

A picture of the front panel of the SSP is shown in Figure 2-1. A description of the controls, indicators, and connectors follows (left to right).

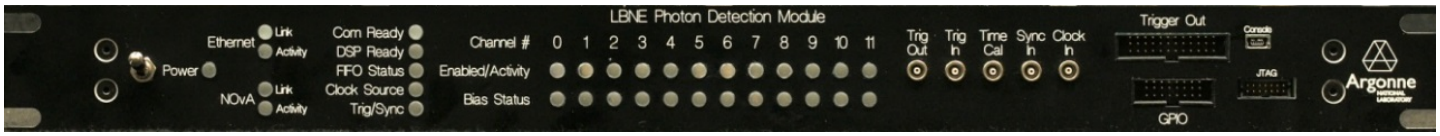


Figure 2-1: Picture of the front panel of the SSP.

Power Switch: The power switch controls the main AC input line voltage to the internal 5V power supply and the on-board linear 31V power supply used for the SiPM bias. All other internal power supplies used in the SSP are powered from the 5V supply.

The power LED: The "Power" LED is powered from an internal voltage monitor that provides an indication that all system voltages are above a lower level threshold set at about 90% of their nominal values:

Off = Power supply out of tolerance or unit off.
Green = Power supplies within tolerance.

Ethernet Status LEDs: These LEDs indicate the state of the Ethernet connection. The "Link" LED indicates a physical Ethernet connection is present. Its color indicates the speed of that connection:

Off = link not established.
Blue = 1 Gb/S link established.
Purple = 100 Mb/S link established.
Red = 10 Mb/S link established.

The "Activity" LED is controlled by the on-board communications processor and will blink/illuminate when there is communication activity:

Off = Idle.
Green = Sending event data.
Red = Slow control activity.

NOvA Status LEDs: These LEDs indicate the state of the NOvA timing interface. These LEDs are controlled by the data processor. When a link with the timing system is established, the "Link"

LED will be illuminated. When timing activity occurs over the NOvA interface, the “Activity” LED will blink.

Com Ready LED: This LED indicates whether the communications processor has booted successfully:

Off = Communication FPGA failed to configure.
Green = Communication successfully configured.

DP Ready LED: This LED indicates whether the data processor is ready:

Off = Data processing FPGA failed to configure.
Green = Data processing successfully configured.

FIFO Status LED: This LED indicates the state of the FIFO in the Zynq FPGA. This LED only reports the status of the DDR buffer, and is not used when the USB interface is selected for transporting event data. The LED will indicate one of four states as described below:

Off = FIFOs are in reset.
Green = FIFO is less than half full.
Yellow/Orange = FIFO is more than half full.
Red = FIFO has less than 1/16 of is space remaining.
Clock Source LED: This LED indicates if a clock source is present:

Off = Selected clock source is not present.
Blue = Indicates that the SSP is running off of the NOvA clock.
Purple = Indicates that the SSP is running off of the front panel clock.
Red = Indicates that the SSP is running off its local clock.

Trig/Sync LED: This LED blinks red when a trigger input is received. It blinks green when a sync pulse is received.

Enabled/Activity LEDs: Each channel has a separate enabled/activity LED. When a channel is enabled but there is no activity, the LED is green. When a channel receives a signal that is above a threshold defined by the setting of the channel discriminator, the LED blinks orange momentarily. As the event rate increases the hue of the LED changes, progressing from green through orange to red. Red indicates a sustained event rate of greater than 10 kHz.

Bias Status LEDs: Each channel has a separate status LED that indicates the state of on-board bias circuit. When the bias has been enabled, the LED glows blue. If the bias circuit for a channel has gone over-temperature (which results in automatic shut-down of the bias circuit), then the LED will glow red.

Trig Out Connector: This is a NIM logic output signal that generates a pulse when a trigger occurs, either internally-generated or externally-generated, depending on the configuration. The pulse width is programmable through the slow control interface.

Trig In Connector: This is a NIM logic input signal that generates an event trigger, forcing a readout of the SSP. Note that the channels must be configured to respond to an external trigger. See Section 7.1 for additional details.

Time Cal Connector: This is a NIM logic input signal that provides an asynchronous timing signal for the charge injection calibration circuit. See chapter 14.

Sync In: This is a NIM logic input signal that resets the time-stamp counter to 0. This is used as to synchronize time-stamps across a system.

Clock In: This is a NIM logic input signal that can be used to provide an external system clock to the module. When operating off of an external clock the module will run at 4x the input frequency. For more details see section 4.1.1.2.The frequency must be between 20MHz and 37.5MHz..

Trigger Out (header): These 12 NIM logic outputs are asserted when the corresponding channel's discriminator triggers. The pulse width is defined by the values of the disc_width_# registers.

GPIO (header): For diagnostic/future use. Treat as TTL outputs.

Console: This is a USB connection that connects to the console port (USB-to-Serial) of the communications processor. This port is provided only for diagnostic communication with the Zynq. User control and/or data access via USB is accomplished via the USB connector on the rear of the module.

JTAG: This connection is used for low-level debugging and to reprogram the firmware in the event that a firmware update over the other communication methods fails.

2.1.1 Trig Out

2.1.1.1 Physical Characteristics

Connector Type:	Coaxial
Connector Manufacturer:	LEMO
Part Number:	EPM-00.250.NTN
Connection Diagram:	See Figure 2-2
Grounding:	Connected to local ground via 10 ohm resistor.

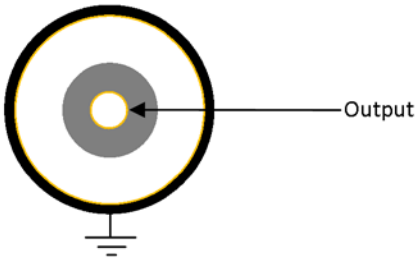


Figure 2-2: Trigger Output Connector Diagram.

2.1.1.2 Electrical Characteristics

Input/Output?:	Output
Termination:	50 ohms
Logic Level:	NIM, negative true
Termination:	None at SSP; Terminate signal at receiver.
Maximum Voltage:	2.1 KV RMS
Maximum Current:	4 A

2.1.2 Trig In / Time Cal / Sync In / Clock In

2.1.2.1 Physical Characteristics

Connector Type: Coaxial
Connector Manufacturer: LEMO
Part Number: EPM-00.250.NTN
Connection Diagram: See Figure 2-3
Grounding: Connected to local ground via 100 ohm resistor.

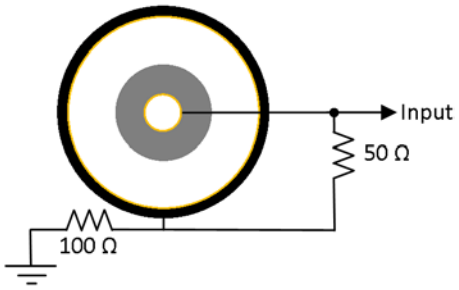


Figure 2-3: LEMO input.

2.1.2.2 Electrical Characteristics

Input/Output?: Input
Termination: 50 ohms
Logic Level: Configurable
Termination: 50 ohms internally to SSP
Maximum Voltage: 2.1 KV RMS
Maximum Current: 4 A

2.1.3 Trigger Output Port

2.1.3.1 Physical Characteristics

Connector Type: Box header, male contacts, 0.1” spacing
Connector Manufacturer: Sullins
Part Number: SBH11-NBPC-D12-SM_BK
Connection Diagram: See Figure 2-4
Grounding: Connected directly to local ground.

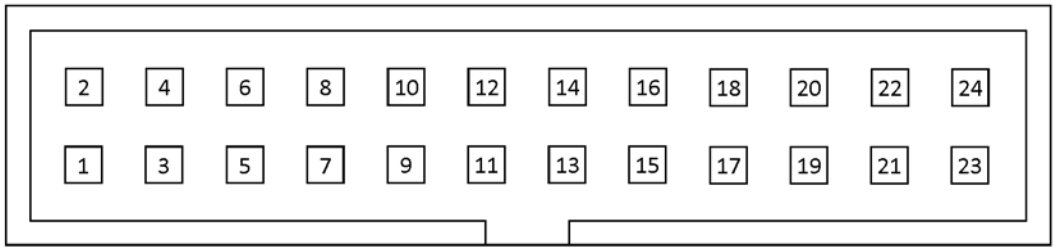


Figure 2-4: Trigger Output Port

Pin #		Pin #	
1	Channel 0 Output	2	Ground
3	Channel 1 Output	4	Ground
5	Channel 2 Output	6	Ground
7	Channel 3 Output	8	Ground
9	Channel 4 Output	10	Ground
11	Channel 5 Output	12	Ground
13	Channel 6 Output	14	Ground
15	Channel 7 Output	16	Ground
17	Channel 8 Output	18	Ground
19	Channel 9 Output	20	Ground
21	Channel 10 Output	22	Ground
23	Channel 11 Output	24	Ground

Table 2-1: Trigger Output Pinout

2.1.3.2 Electrical Characteristics

Input/Output?: Output
Termination: 50 ohms
Logic Level: NIM, negative true.
Termination: None at SSP; Terminate signal at receiver.
Dielectric Breakdown: 1000 V
Maximum Current: 3 A per contact

2.1.4 GPIO Port

2.1.4.1 Physical Characteristics

Connector Type: Box header, male contacts, 0.1” spacing
Connector Manufacturer: ASSMANN
Part Number: AWHW16-G-SMD-R
Connection Diagram: See Figure 2-5
Grounding: 3.3V LVTTL

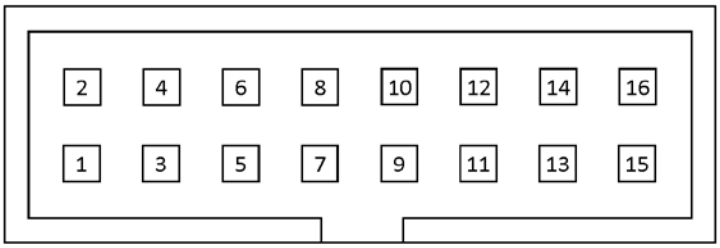


Figure 2-5: GPIO port.

Pin #		Pin #	
1	Comm FPGA GPIO 0	2	Ground
3	Comm FPGA GPIO 1	4	Ground
5	Comm FPGA GPIO 2	6	Ground
7	Comm FPGA GPIO 3	8	Ground
9	DP FPGA GPIO 0	10	Ground
11	DP FPGA GPIO 1	12	Ground
13	DP FPGA GPIO 2	14	Ground
15	DP FPGA GPIO 3	16	Ground

Table 2-2: GPIO Pinout

2.1.4.2 Electrical Characteristics

Input/Output?: Configurable
Logic Level: Configurable
Termination: None
Dielectric Breakdown: 1000 V
Maximum Current: 3 A per contact

2.1.5 Console Port

2.1.5.1 Physical Characteristics

Connector Type: Mini USB 2.0 Type B Receptacle
Connector Manufacturer: Wurth
Part Number: 651005136421
Grounding: Local connection to digital ground; Shell connected to chassis

2.1.5.2 Electrical Characteristics

Input/Output?: I/O
Impedance: USB 2.0 convention
Logic Level: USB 2.0 convention
Termination: USB 2.0 convention
Working Voltage: 30 V
Maximum Current: 1.0 A per contact

2.1.6 JTAG Port

2.1.6.1 Physical Characteristics

Connector Type: Box header, male contacts, 2 mm spacing
Connector Manufacturer: Molex
Part Number: 0878321420
Connection Diagram: See Figure 2-6, Xilinx standard pinout.
Grounding: Connected to digital ground

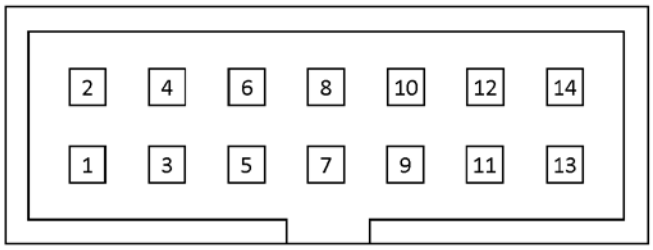


Figure 2-6: JTAG port.

2.1.6.2 Electrical Characteristics

Input/Output?: I/O
Impedance: JTAG
Logic Level: JTAG
Termination: JTAG
Maximum Voltage: 30 V
Maximum Current: 1.0 A per contact

2.2 Back Panel

A picture of the rear panel of the SSP is shown in Figure 2-7. A description of the controls, indicators, and connectors follows (left to right).

Figure 2-7: Picture of the rear panel of the SSP. (Rev A-Top, Prototype-Bottom)

Ethernet Connector: This is a standard 10/100/1000 Cat6 Ethernet port that connects to the internal communications processor.

USB Connector: This is a standard USB B receptacle connecting to the FTDI USB 2.0 interface.

NOvA timing connector: This connector is the NOvA timing interface input.

External Bias connector: This connector allows an external bias voltage to be connected to the module for biasing the SiPMs. The external bias should not exceed 50v due to capacitor voltage ratings. An internal jumper, one per channel, selects whether this input or the internal bias circuit provides bias to the SiPMs.

SiPM Input Connectors: This connector is the differential input for each SiPM front-end channel. The connector is a Lemo 0B connector. The bias voltage selected is asserted upon the SiPM input connector pins. The actual SiPM signal is internally AC coupled to the data processing circuitry.

AC Mains Input Connector: This connector provides 120v 60Hz AC power to the module. The connector is an IEC-320-C14. The AC input power is fused on-board with a rating of 1.0 amp.

2.2.1 10/100/1000 Ethernet Port

2.2.1.1 Physical Characteristics

Connector Type: 8-pin RJ-45 modular jack
Connector Manufacturer: TE-Connectivity
Part Number: 188566-1
Connection Diagram: See Figure 2-8
Grounding: Shield connection to chassis

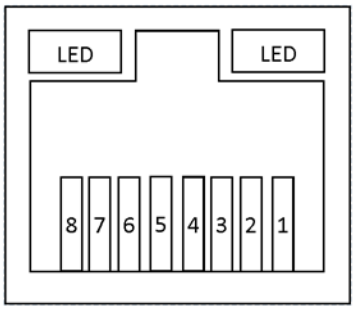


Figure 2-8: Ethernet port.

2.2.1.2 Electrical Characteristics

Input/Output?: I/O
Impedance: Gigabit Ethernet convention
Logic Level: Gigabit Ethernet convention
Termination: Gigabit Ethernet convention
Rating: Cat5e

2.2.2 USB Interface Port

2.2.2.1 Physical Characteristics

Connector Type: USB Type B Receptacle
Connector Manufacturer: FCI
Part Number: 61729-0010BLF
Grounding: Shield connected to chassis

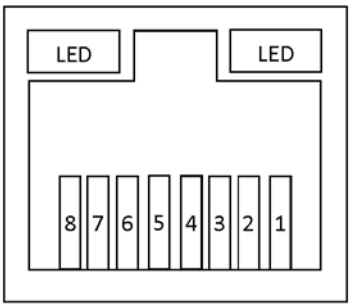
2.2.2.2 Electrical Characteristics

Input/Output?: I/O
Impedance: USB convention
Logic Level: USB convention
Termination: USB convention
Working Voltage: 30 V
Maximum Current: 1.0 A per contact

2.2.3 NOvA Interface Port

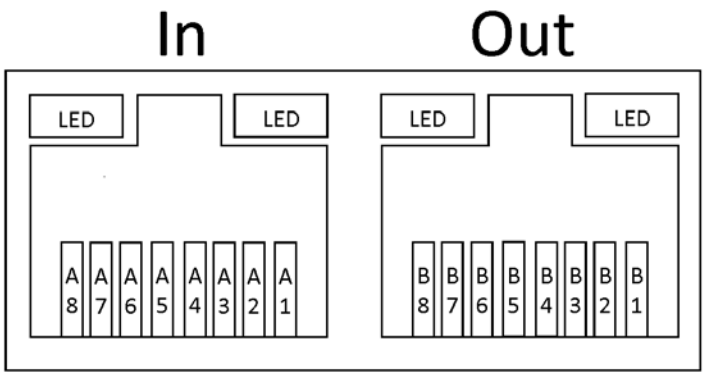
2.2.3.1 Physical Characteristics

Connector Type: 8-pin RJ-45 modular jack
Connector Manufacturer: TE-Connectivity
Part Number: 2-406549-1 or 6368419-1 (Rev A)
Connection Diagram: See Figure 2-9 or Figure 2-10 (Rev A)
Grounding: Shield connection to chassis



Pin #	Direction	Purpose
1	Input	NOvA Sync (+)
2	Input	NOvA Sync (-)
3	Output	NOvA Sync Echo (+)
4	Input	NOvA Clock (-)
5	Input	NOvA Clock (+)
6	Output	NOvA Sync Echo (-)
7	Input	NOvA Data Input (+)
8	Input	NOvA Data Input (-)

Figure 2-9: NOvA connector and pinout.



Pin #	Direction	Purpose
A1	Input	NOvA Sync (+)
A2	Input	NOvA Sync (-)
A3	Output	NOvA Sync Echo (+)
A4	Input	NOvA Clock (-)
A5	Input	NOvA Clock (+)
A6	Output	NOvA Sync Echo (-)
A7	Input	NOvA Data Input (+)
A8	Input	NOvA Data Input (-)

Pin #	Direction	Purpose
B1	Output	NOvA Sync (+)
B2	Output	NOvA Sync (-)
B3	Input	NOvA Sync Echo (+)
B4	Output	NOvA Clock (-)
B5	Output	NOvA Clock (+)
B6	Input	NOvA Sync Echo (-)
B7	Output	NOvA Data Input (+)
B8	Output	NOvA Data Input (-)

Figure 2-10: Rev A NOvA connector and pinout.

2.2.3.2 Electrical Characteristics

Input/Output?:	I/O
Impedance:	100 ohms within a pair
Logic Level:	LVDS
Termination:	Received signals terminated on SSP; Output signals have no back termination.
Maximum Voltage:	30 V
Maximum Current:	1.0 A per contact

2.2.4 External Bias Input

2.2.4.1 Physical Characteristics

Connector Type:	Coaxial
Connector Manufacturer:	LEMO
Part Number:	EPM-00.250.NTN
Connection Diagram:	See Figure 2-11
Grounding:	Shield connected local analog ground

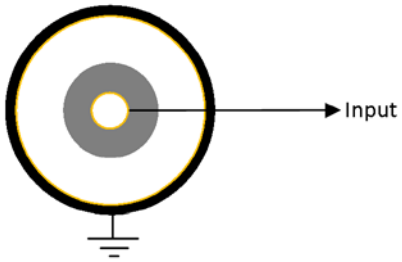


Figure 2-11: Bias input.

2.2.4.2 Electrical Characteristics

Input/Output?:	Input
Impedance:	50 ohms
Logic Level:	
Termination:	100 K
Maximum Voltage:	2.1 KV RMS
Maximum Current:	4 A

2.2.5 SiPM Inputs

2.2.5.1 Physical Characteristics

Connector Type: Differential (2-conductor) Coaxial
Connector Manufacturer: LEMO
Part Number: EPG-0B.302.HLN
Connection Diagram: See Figure 2-12
Grounding: Connected to local analog ground

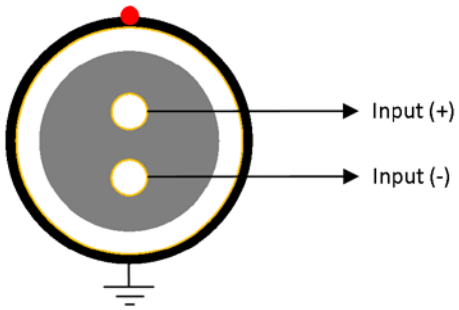


Figure 2-12: SiPM input port diagram.

2.2.5.2 Electrical Characteristics

Input/Output?: Input
Impedance: 50 ohms
Signal Type: Analog
Termination: 100 ohms differential termination; 50 ohms signal to shield
Maximum Voltage: 1.2 KV
Maximum Current: 4.5 A each contact

2.2.6 AC Input

2.2.6.1 Physical Interface

Connector Type: IEC-320-C14 Male chassis mount AC receptacle, with fuse
Connector Manufacturer: Qualtech
Part Number: 701W-X2/09-ND
Grounding: Safety ground connected to SSP chassis

2.2.6.2 Electrical Characteristics

Input/Output?: Input
Signal Type: 120V, 60 Hz AC power
Maximum Voltage: 250V
Maximum Current: 15A

3. Communication Interfaces

The SSP supports two communication interfaces: USB 2.0, and 10/100/1000 Mbps Ethernet. These are implemented by a MicroZed system-on-module [5] that is incorporated into the SSP, which contains a Xilinx Zynq FPGA [6]. The SSP also hosts a console port as an auxiliary method for providing slow controls and monitoring. The event_data_interface_sel register selects which interface is used to send event and waveforms data. Setting the register to 0 selects the USB interface, while setting it to 1 selects the Ethernet interface. Slow control can still occur on either interface regardless of this setting. All interfaces (USB, UDP and TCP) may be used simultaneously for slow control on monitoring.

3.1 USB 2.0

The USB interface is driven by a FTDI 2232H USB controller. The FTDI 2232H provides two internal data ports to the SSP (only one physical USB interface). One port is used for slow control and monitoring and the other for transporting event data. Both interfaces require the use of the FTDI's D2XX driver on the host computer that communicates with the SSP, which may be downloaded from their website (http://www.ftdichip.com/FTDrivers.htm). The slow control port will be recognized by the D2XX driver as a serial port, and the event data port will be recognized as a FTDI FIFO interface. Note that when using USB to transfer event data the event buffering memory will be very limited. See section 11.3 for more details.

3.2 10/100/1000 Mbps Ethernet

The Ethernet interface is driven by Marvell Alaska 88E1512 10/100/1000 Mbps physical layer. This interface supports TCP/IP protocol for both slow control and event data transfer. Currently the SSP provides two TCP ports for slow control and monitoring on ports 50001 and 50002. Only one client may connect to each TCP port. A UDP port is also provided on port 50001 and may be used by multiple clients.

Event data is sent via TCP port 50010. Only one client may connect to each TCP port. Any available event data will immediately be sent as it become available. Flow control is handled by the TCP protocol. No data is sent from the client to the SSP over this port.

The default IP settings are:

IP4 Address: 192.168.1.123
IP4 Netmask: 255.255.255.0
IP4 Gateway: 192.168.1.1
MAC Address: 32-02-02-02-02-02

These default values must be changed when multiple SSP/LCMs exist on the same network. The convention has been to set to LSB of the IP to the serial number of the SSP + 100, and the LSB of the MAC address to just the serial number.

The module will fall back to these values if the communication configuration is erased. (See section B.2 Configuration Memories for more details.)

Two diagnostic features are also provided over the Ethernet interface. One is a loop back interface hosted on TCP port 50000. Any data sent to this port will be echoed back to the client. The other diagnostic feature, is that the SSP will broadcast a beacon packet every 10 seconds. The interval between beacons is controlled by the beacon_interval register. The register value sets the interval in ms. The pay load of the packet is shown in Table 3-1 below:

Word/Byte =>	0	1	2	3
1	Fixed String [0..3]			
2	Fixed String [4..7]			
3	Fixed String [8..11]			
4	0x000000 (Fixed Value)			Is IPv6
5	Beacon Count			
6	0x00	0x00	0x00	0x00
…	…	…	…	…
325	0x00	0x00	0x00	0x00

Table 3-1: Beacon Packet Format.

This can be monitored to detect loss of communication, or used to discover SSPs on a network.

4. System Timing

The clock management logic allows the ADCs in the SSP to be clocked from either the onboard oscillator or one of two external clock and timing sources. The user may either connect the module to the NOvA timing system or provide an external clock to the front panel. A jitter-attenuating PLL may be used with external clock sources. Figure 4-1 shows an overview of the clock manager and clock distribution on board the SSP.

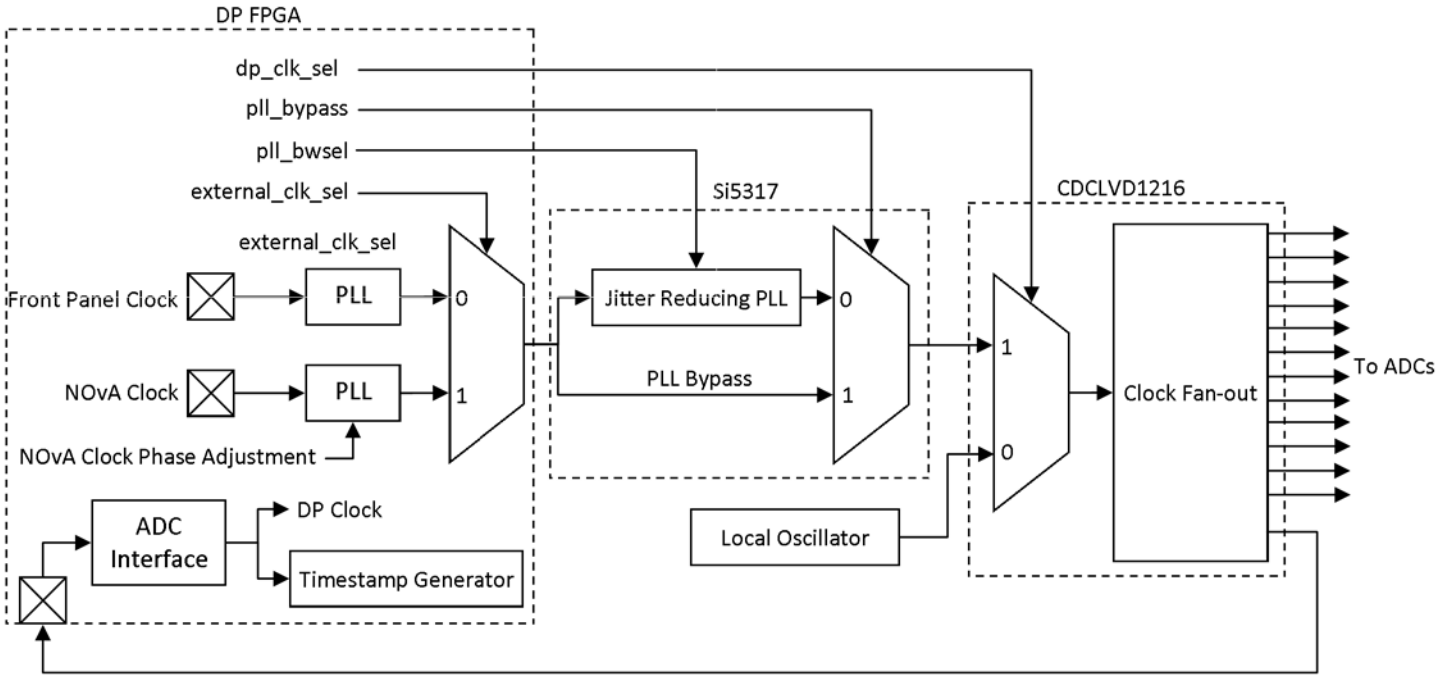


Figure 4-1: Clock manager block diagram.

Two methods of time-stamping events are also provided. The SSP always provides a 48-bit timestamp in the header of each event. The 48-bit timestamp is clocked by the Data Processor clock, and can be reset manually or automatically via NOvA. When using an external clock or timing source, an additional 64-bit event timestamp field is also reported in the header. The meaning of this timestamp depends on the external clock/timing mode used. This is described in more detail in section 4.2. Typical clocking and timing configurations are also provided at the end of this section.

4.1 Data Processor Clock

The Data Processor clock manager consists of 3 sections, as illustrated in Figure 4-1 the external clock selection mux logic within the DP FPGA, the jitter-attenuating PLL, and the clock fan-out. When operating the SSP's ADCs using the local oscillator only the configuration of the clock fan-out section is relevant.

4.1.1 External Clock Mux

An expanded block diagram of the external clock mux section and its inputs is shown in Figure 4-2.

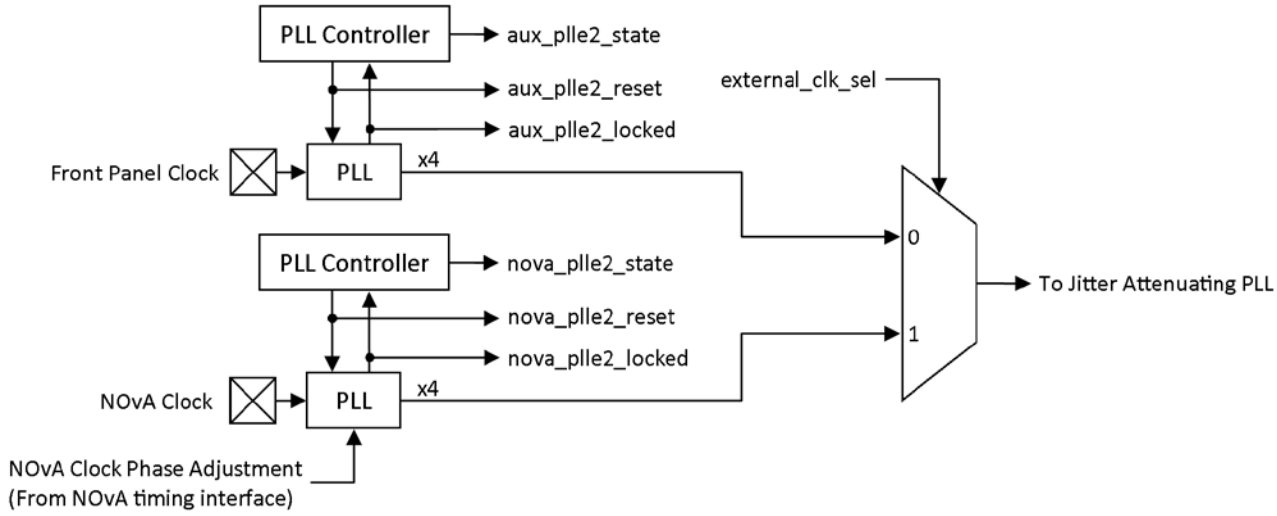


Figure 4-2: External clock mux block diagram.

4.1.1.1 NOvA Clock Input

The NOvA clock input is provided to the SSP as an LVDS signal on the NOvA interface connector, located on the back panel of the module (see Section 2.2). The NOvA clock input is received on pins 4/5 of the NOvA port, as shown in Figure 2-9. The NOvA Timing system is described in [7].

The LVDS clock is differentially terminated into 100 ohms inside the SSP module. Other termination configurations and support of other signaling standards are possible if required. If used, the provided clock frequency must be between 20.0MHz and 37.5MHz. See section 2.2.3 for the full electrical specification and pin-out.

4.1.1.2 Front Panel Clock Input

The front panel clock input is provided to the SSP as a NIM, TTL, or LVTTL signal, depending on how the front panel is configured (factory setting.) The external clock source must be connected to the LEMO connector labeled Clock In on the front panel of the module (see section 2.1.) The outer conductor is connected to ground within the SSP module.

The front panel clock input is terminated to ground via a 49.9 ohm resistor. If used, the provided clock frequency must be between 20.0MHz and 37.5MHz. (The Input PLL does permit the use of other frequencies, but this requires changes to the firmware. Please consult with ANL representatives if this is needed.) See section 2.1.2 for the full electrical specification and pin-out of the clock input port.

4.1.1.3 External Clock Mux Control

The external_clk_sel bit in the dp_clock_control register selects the external clock source. Setting the external_clk_source to ‘0’ selects the front panel clock. Setting it to ‘1’ selects the

NOvA clock. However, this bit does not control if the (selected) external clock source or the local oscillator is used to drive the ADCs and DP logic. That is done by the dp_clk_sel also in the dp_clock_control register. See section 4.1.3 for details regarding the dp_clk_sel control bit.

4.1.1.4 Input PLLs

Each external clock input is passed through an Input PLL, within the DP FPGA, that multiplies the input clock frequency by 4, prior to the external clock selection multiplexer. Figure 4-3 below shows a single generalized Input PLL.

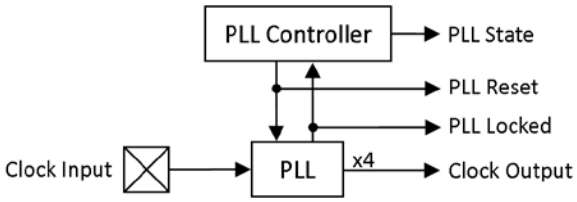


Figure 4-3: Input PLL.

Operation of the Input PLL is completely automated and no user controls are provided. Once an appropriate clock source is connected, the PLL automatically locks onto the input. Each PLL provides three status indictors for monitoring purposes: PLL Reset, PLL Locked, and PLL State. These outputs can be read from the dp_clock_status register. Table 4-1 provides the register bit-field names for each of these status indicators.

Clock Input Source	PLL State	PLL Reset	PLL Locked
Front Panel	aux_plle2_state	aux_plle2_reset	aux_plle2_locked
NOvA	nova_plle2_state	nova_plle2_reset	nova_plle2_locked

Table 4-1: Register bit-field names for PLL status outputs.

4.1.1.4.1 PLL Reset

The PLL reset indicator does not provide an effective means of monitoring the Input PLLs and should not be relied upon for determining the status of the external clock. PLL reset is provided primarily as a diagnostic tool for firmware development and is set whenever the PLL Controller demands a reset of the PLL. This can occur if the input clock source stops oscillating or becomes disconnected.

4.1.1.4.2 PLL Locked

The PLL locked indicator is set to '1' when the Input PLL has detected the external clock input, and has locked onto the frequency and phase of that clock. This is an indicator that the clock is present, but the PLL State indicator should be used to determine if the clock source is stable and clean.

4.1.1.4.3 PLL State

Each Input PLL has a PLL controlling state machine that manages its operation. The state of the PLL Controller can be monitored via the PLL State indicator. This is the most reliable way of monitoring the status of the external clock source. Table 4-2 lists the possible values of the PLL State.

PLL State Value	State Description	External Clock Source Ready?
0xF, 0xD	PLL controller is being held in reset.	No
0xC	PLL controller is resetting the PLL.	
0xB	PLL controller is waiting for the PLL to lock.	
0xA	PLL is locked, waiting for the PLL's output to stabilize.	
0x8	Loss of lock detected, resetting PLL.	
0x1	PLL locked and stable.	Yes

Table 4-2: PLL State codes.

Any state other that 0x1 means that the input clock source is not ready for use.

4.1.1.4.4 Phase Control (NOvA PLL Only)

The NOvA clock Input PLL has an additional capability to shift the output clock phase in steps of 17ps. Support of NOvA clock phase alignment will be added when NOvA timing support is integrated into the DP firmware. Currently, there are no monitors or controls associated with this feature.

4.1.2 Jitter-attenuating PLL

The Input PLLs only serve to multiply/divide the input clock frequency to obtain the 150MHz clock used to drive the ADCs and DP logic. They do not effectively attenuate jitter. Therefore, an additional jitter-attenuating PLL is provided external to the DP FPGA to improve the clock quality when using an external clock source. This PLL receives the external clock as selected by the external_clk_sel control bit. An expanded block diagram of the jitter-attenuating PLL is shown in Figure 4-4.

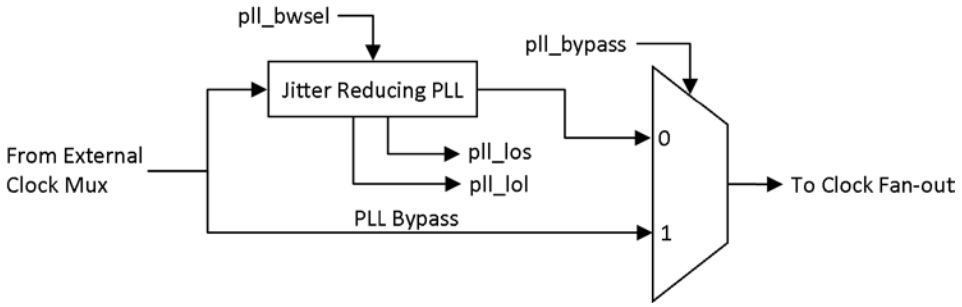


Figure 4-4: Jitter-attenuating PLL block diagram.

The jitter-attenuating PLL may be bypassed by setting the pll_bypass bit in the dp_clock_control register. Otherwise, the PLL's loop bandwidth can be adjusted via the pll_bwsel bit-field in the dp_clock_control register. There are five bandwidth settings as shown in Table 4-3.

pll_bwsel Value	PLL Loop Bandwidth
0x0	107Hz
0x1	214Hz
0x2	429Hz
0x3	1735Hz
0x4	7240Hz
All other values.	7240Hz

Table 4-3: Jitter-attenuating PLL loop bandwidth settings.

The PLL provides two status bits pll_lol and pll_los in the dp_clock_status register. The pll_lol bit indicates if the PLL is phase locked onto the selected external input clock. The pll_los bit indicates a loss of signal condition. This is set when the jitter-attenuating PLL does not detect a clock at its input. The most likely cause of this condition is a failure of the external clock source.

4.1.2.1 Response to External Failure

The SSP can be configured to respond in one of two ways to an external clock failure. If the jitter reducing PLL is bypassed and the external clock stops oscillating or becomes disconnected, the clock to the ADCs stops and the data processing pipeline is held in reset. Buffered events that are pending readout may still be read out but no new data will be taken.

If the jitter reducing PLL is in use, it freezes the current state of its VCO at the moment the external clock is lost and holds the current clock frequency. In this mode, event processing continues. However, all events that are processed while operating under this condition are flagged in the event headers (this feature is to be implemented at a later release.) This mode is identifiable via monitoring of the vco_freeze bit in the dp_clock_status register. If the bit is set, the jitter-attenuating PLL is operating in the fail-over state. If cleared, the PLL is operating normally.

4.1.2.2 Advanced PLL Function

The jitter-attenuating PLL also provides a manual phase shifting function to the user. This allows the phase of the 150MHz clock to be manually shifted in 180ps steps. The phase shift is continuous through 360 degrees, and there is no adjustment limit. To increment the phase, write '1' to the pll_phase_inc bit in the dp_clock_phase_control register. To decrement the phase, write '1' to the pll_phase_dec bit. To reset the phase skew to 0, write a '1' to the pll_phase_reset bit. All bits in this register self-clear.

4.1.3 Clock Fan-Out

The clock fan-out logic is where either the selected external clock or the internal oscillator clock is driven to the 12 ADCs. In addition, one copy is driven back to the DP FPGA, as the source of the common data processing clock. Figure 4-5 shows an expanded block diagram of the clock fan-out logic.

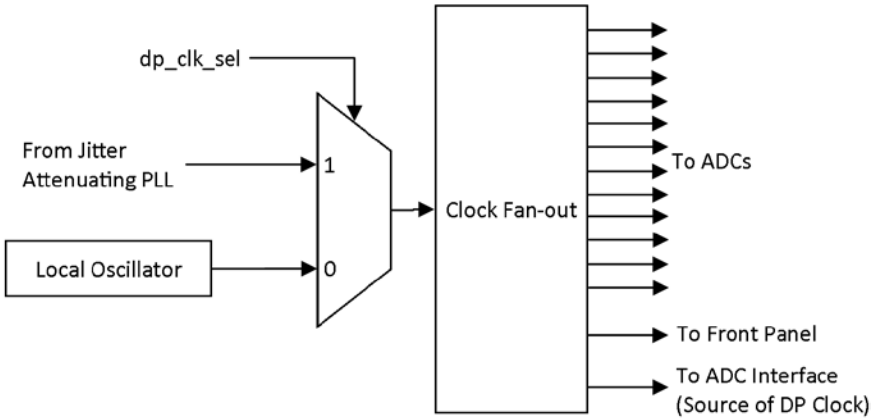


Figure 4-5: Clock fan-out block diagram.

The dp_clk_sel bit of the dp_clock_control register selects if the module runs off of the local oscillator or the external clock as selected by the external_clk_sel bit. No direct status monitoring is available for the clock fan-out. However, monitoring the status of the DP PLL provides indication of whether the clock fan-out is currently driving the 150MHz clock out to the ADCs.

4.2 Timestamp Generator

The SSP maintains two timestamps: a 48-bit local timestamp and a 64-bit "external" timestamp. Both values are reported in the event header. In addition to generating the timestamp, the timestamp generator also produces 7 diagnostic triggers (or timestamp flags) that can be used by the DP to trigger a channel at a fixed rate and/or used to triggering charge injection. Figure 4-6 show the block diagram of the timestamp generator.

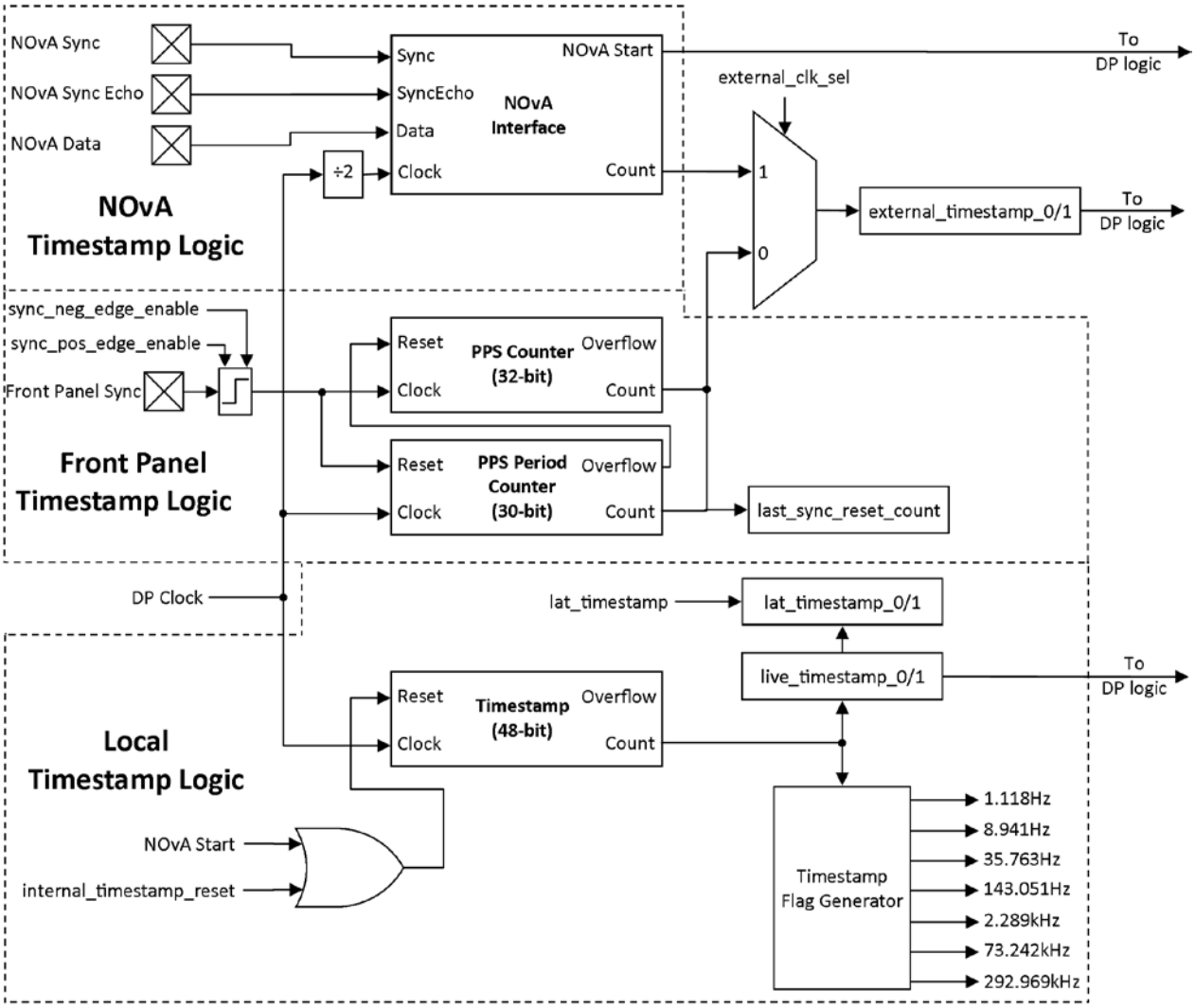


Figure 4-6: Timestamp generator block diagram.

All timestamps always run synchronously to the data processing clock. This is the same clock that drives the ADCs. Refer back to Figure 4-1 for the source of the DP Clock.

4.2.1 Local Timestamp

The DP maintains an internal 48-bit timestamp as an absolute time base, used for many internal functions such as the timing of waveform readouts. This timestamp is reported in the event header and can be used as the primary method of time stamping events when operating a single SSP in isolation or when no external timing correlation with other hardware is required. The local timestamp increments at 150MHz (6.667ns/tick), and is driven by the same clock that is used for the ADCs.

The local timestamp can be reset via the internal_timestamp_reset bit in the master_logic_control register. When internal_timestamp_reset is set the local timestamp is held to 0. Event handling will not function properly while the internal_timestamp_reset bit is set, since the waveform readout logic relies on the operation of the timestamp. Events generated while the readout is in reset will not be made available for readout.

The local timestamp is also reset at the start of a NOvA initiated run. This "NOvA Start" function is described in section 6.1.1.

The local timestamp can be monitored through either the live_timestamp_0/1 registers or the lat_timestamp_0/1 registers. The live timestamp registers are continuously updated, and may rollover between reads of 0 and 1 registers. To prevent this condition, the lat_timestamp_0/1 registers latches the value of the live_timestamp_0/1 registers when the latch_timestamp bit of the dp_pulsed_control register is set. This bit automatically clears after the timestamp has been latched.

4.2.1.1 Timestamp Flag Generator

The timestamp flag generator creates seven periodic trigger sources that may be used to perform baseline measurement or charge injection. The flags are generated by comparing various numbers of least significant bits of the local timestamp against zero. The flags are asserted for one clock cycle. There is no monitoring or control of this feature. Table 4-4 lists the timestamp trigger rates.

Number of bits compared against zero	Effective trigger rate
Lower 9 bits	292.969kHZ
Lower 11 bits	73.242kHz
Lower 16 bits	2.289kHz
Lower 20 bits	143.051Hz
Lower 22 bits	35.763Hz
Lower 24 bits	8.941Hz
Lower 27 bits	1.118Hz

Table 4-4: Timestamp trigger rates.

Note that these flags will not be asserted synchronously across multiple SSPs unless the NOvA Synchronous Start method is used at described in section 6.1.1.

4.2.2 External Timestamp

The DP FPGA maintains a 64-bit external timestamp. The format of this 64-bit time stamp depends on which external timestamping mode is selected. Synchronization of the external timestamp across multiple SSPs can either be done via the front panel or the NOvA timing interface. The external_clk_sel bit of the dp_clock_control register selects which mode is used. (0 = Front Panel, 1 = NOvA)

The current value of the external timestamp can be read from the external_timestamp_0/1 registers.

4.2.2.1 Front Panel Based Timestamp

The 32-bit PPS Counter counts the number of sync (PPS) pulses received from the selected sync input source. The 30-bit PPS Period Counter counts the number of 150MHz DP clock cycles that have elapsed since the last sync pulse was received. In this mode the external_timestamp_0 register holds the value of the PPS Period Counter, whereas, the external_timestamp_1 register holds the value of the PPS Counter. The last_sync_reset_count register latches the PPS Period Counter value when the selected sync pulse input is asserted. The value of this register is a measure of the period of the sync pulse in terms of the dp_clock periods.

When using the front panel, synchronization is achieved by externally gating the 1PPS for 8 or more seconds. This holds the 1 PPS counters in all SSPs in reset. All modules will begin counting simultaneously upon receipt of the first sync pulse. The front panel sync input may be configured to trigger on either rising or falling edges. Setting the sync_pos_edge_enable bit in the misc_config register enables positive edge sync triggers. Similarly, setting the sync_neg_edge_enable bit enables negative edge sync triggers. If neither bit is set, sync pulses are ignored. This effectively disables external timing.

4.2.2.1.1 Front Panel Sync Input

The front panel sync input is provided to the SSP as a NIM, TTL, or LVTTL signal, depending on the configuration of the front panel. The external sync must be connected to the LEMO connector labeled Sync In. As illustrated in Figure 2-2, the outer conductor is connected to ground within the SSP module. The front panel sync input is terminated to ground via a 49.9 ohm resistor. See section 2.1.2 for the full electrical specification and pin-out.

4.2.2.2 NOvA Based Timestamp

When using the NOvA timing system, the NOvA timing master is responsible for timestamp synchronization via the NOvA interface. In this mode external_timestamp_0/1 hold the 56-bit NOvA timestamp. The upper most 8-bits are always 0. The timestamp counts at 64MHz, one half of the dp_clock frequency.

4.2.2.2.1 NOvA Sync Input

The NOvA sync input is provided to the SSP as an LVDS signal on the NOvA interface connector, located on the back panel of the module. The NOvA sync input should be driven on pins 4/5 of the NOvA port. Figure 2-7 details the pin-out of the NOvA connector. The LVDS sync is differentially terminated into 100 ohms inside the SSP module. Other termination configurations and support of other signaling standards are possible if required. See section 2.2.3 for the full electrical specification and pin-out.

4.3 Typical Configuration Examples

Timing and timestamping of events can be handled in different ways depending on the needs of the experiment. This section describes the typical configurations.

Timing Mode	Using External Clock?	Synchronized Timestamping?	external_clk_sel	dp_clk_sel
Local	No	No	Don't Care	0
Local Clock with Synchronized Timestamping	No	Yes	0	0
External Clock with Synchronized Timestamping	Yes	Yes	0	1
NOvA Timing	Yes	Yes	1	1

Table 4-5: Typical timing configurations.

4.3.1 Local Timing Mode

In local timing mode, the SSP uses its internal oscillator and local timestamp. Correlating timestamps between SSP modules is not possible when operating in this mode.

4.3.2 Local Clock with Synchronized Timestamping

This is the most basic method of synchronization, and allows the user to supply a one pulse per second (PPS) signal to one or more modules to synchronize their timestamps.

The ADCs are still driven by the oscillator on board the SSP, but since the timestamps are synchronized every second, the time of events across multiple SSPs can be correlated to one another to within a few ticks of the 150 MHz clock. The accuracy of the timing correlation between SSPs in this mode is limited by the frequency tolerance of the internal oscillator. The oscillators onboard the SSP are rated at a frequency accuracy of 1Hz and a stability of 25ppm.

To help decrease the timing uncertainty, the SSP provides a register that reports the number of its 150 MHz clocks that have transpired between the previous two PPS signals. This allows the mean frequency of each SSP to be better characterized when operating in this mode.

4.3.3 External Clock with Synchronized Timestamping

For better timing correlation between SSPs, the modules can be synchronized to a single, external clock source. In this way all the ADCs across multiple SSPs can be driven from a single clock source, eliminating frequency uncertainty between modules. This is achieved by supplying an external clock to the external clock input on the front panel of the SSP. The external clock source must be within the allowed frequency range.

4.3.4 NOvA Timing

The SSP's hardware implements support for the NOvA timing system. This feature is not yet implemented in the current firmware, but a future firmware release will allow for direct interfacing with the NOvA timing system including timestamp and clock synchronization. The 62.5 MHz NOvA clock will be multiplied by 2 to generate the 125 MHz ADC clock using clock generation modules within the DP FPGA.

5. ADC Interface

The ADC interface has two primary functions. First, at power-on, the ADCs are configured via an SPI interface by the ADC configuration controller. Second, the interface receives the data from the ADCs and compensates for channel to channel propagation delays of the digitized waveform samples. Both functions are completely automated and require no action by the user. This section is provided to document the various status values that the ADC interface reports, as well as advanced diagnostic features. These can be used to diagnose hardware problems.

5.1 ADC Configuration Controller

A block diagram of the ADC configuration controller is shown in Figure 5-1.

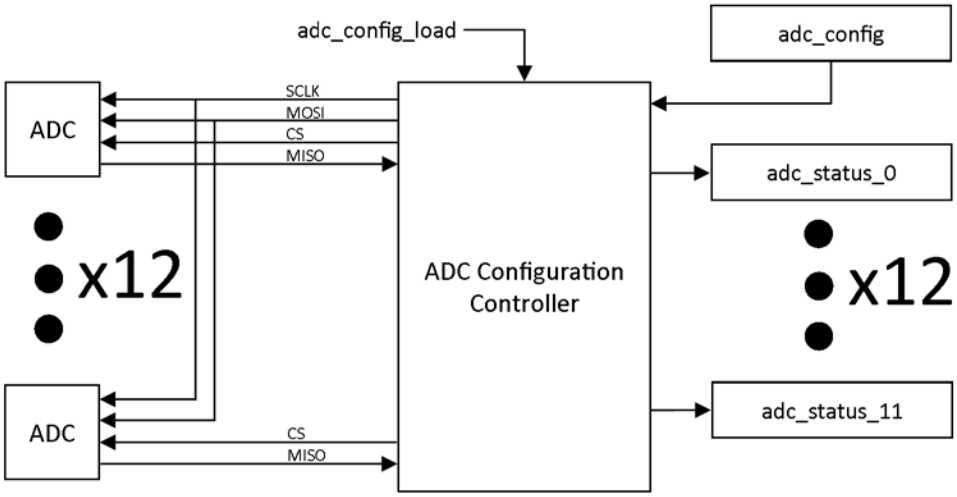


Figure 5-1: ADC configuration controller.

At power on, the ADC configuration controller reads the value stored in the adc_config register and loads it into the ADCs. As a diagnostic feature, it then reads back the configuration state of all ADCs and reports it in the adc_status_# registers, where # is the channel number. The adc_status and adc_config registers have identical formatting. The data in the status register should match the adc_config register, after power on. If these values do not match after power, the likely cause is a hardware problem within the SSP module. The structure of the adc_status/control register is described in Table 5-1. Refer to the SSP register map and the LTC2262-14 datasheet for more information. (Warning: These values should not be modified, except during servicing or repair of the module. Inappropriate values will cause data corruption.)

To load a new ADC configuration, first write the configuration in the adc_config register, then set the adc_config bit in the adc_config_load register to '1'. This bit will self-clear.

Bit Field	Maps to
7:0	ADC register A1
15:8	ADC register A2
23:16	ADC register A3
31:24	ADC register A4

Table 5-1: ADC configuration data format.

5.2 ADC Data Phase Alignment

The ADC interface does not compensate for analog propagation delays nor does it have any effect on when the ADCs samples the data. The ADCs always sample synchronously with each other since the length of all clock lines are matched on the PCB. However, the data lines from the ADCs are only matched to other data lines within the same channel. The sole purpose of the ADC interface is to compensate for the channel to channel skew in the data lines, and bring all channels into a single common clock domain. A simplified block diagram of the ADC interface is shown in Figure 5-2.

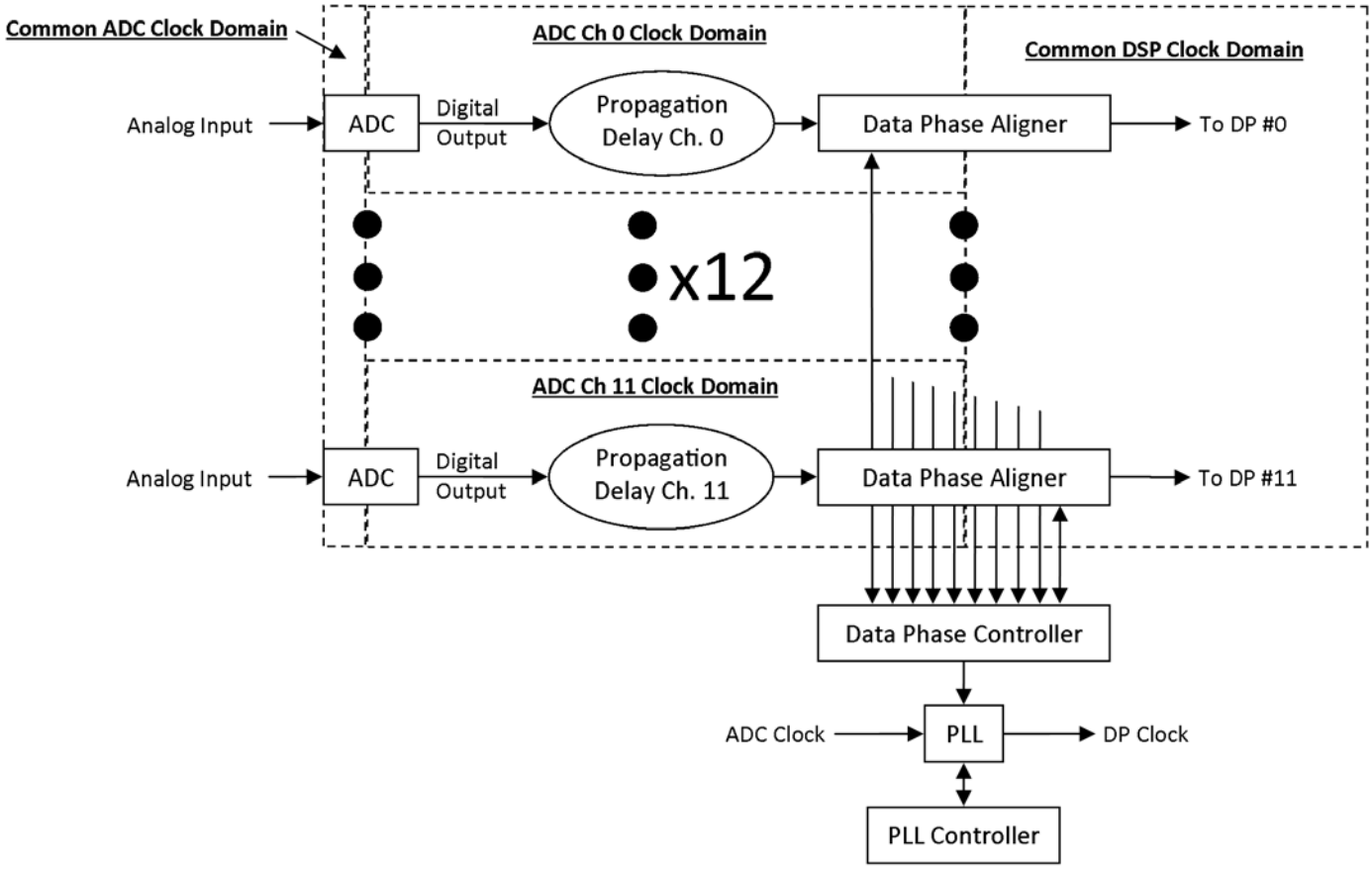


Figure 5-2: ADC interface block diagram.

Data phase alignment occurs in two steps. First, the Data Phase Controller performs a course phase alignment, to compensate for the net offset between the ADC Clock and the DP Clock. The Data Phase Controller then releases the Data Phase Alignment logic on each channel, to perform fine channel-by-channel alignment. This fine alignment runs continuously to compensate for any effects of process, temperature and voltage variation over time.

5.2.1 Data Phase Controller and DP PLL

The Data Phase Controller and DP PLL have several status outputs. An expanded block diagram of this area is shown in Figure 5-3.

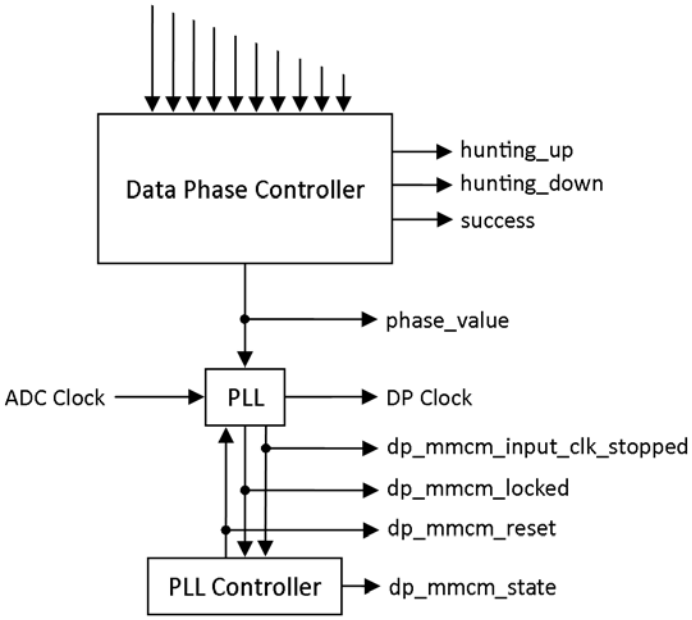


Figure 5-3: Data phase controller and DP PLL.

5.2.1.1 Data Phase Controller

To monitor the status of the course phase alignment the Data Phase Controller provides three outputs: hunting_up, hunting_down, and success. All three are located in the dp_clock_status register. If the course phase alignment is successful, then the Data Phase Controller will set the success bit high. Otherwise, if hunting_up or hunting_down is set, then the Data Phase Controller is either still compensating for the ADC to DP clock phase, or a hardware problem exists. The course phase alignment step should take less than 1ms to complete. Once completed, the phase_value register provides a measure of the relative phase difference between the ADC clock and DP clock. Each count of the phase value register equates to approximately 17ps of offset. These values may be positive or negative.

5.2.1.2 DP PLL

The structure of the DP PLL is similar to that of the Input PLLs discussed in section 4.1.1.3. The PLL provides four status indictors for monitoring purposes: dp_mmcm_reset (PLL Reset), dp_mmcm_locked (PLL Locked), dp_mmcm_state (PLL State), and

dp_mmcm_input_clk_stopped (clock stopped indication). These outputs can be read from the dp_clock_status register.

5.2.1.2.1 PLL Reset

The PLL reset indicator does not provide an effective means of monitoring the DP PLL and should not be relied on for determining the status of the ADC clock. The PLL reset is provided primarily as a diagnostic tool for firmware development and is set whenever the PLL Controller demands a reset of the DP PLL. This can occur if the input clock source stops oscillating.

5.2.1.2.2 PLL Locked

The PLL locked indicator is set to '1' when the DP PLL has detected the ADC clock input, and has locked onto the frequency and phase of that clock. This is an indicator that the clock is present, but the PLL State indicator should be used to determine if the clock source is stable and clean.

5.2.1.2.3 PLL State

Like the Input PLLs, the DP PLL has a PLL controlling state machine that manages its operation. The state of the PLL Controller can be monitored via the PLL State indicator. This is the most reliable way of monitoring the status of the external clock source. Table 4-2 lists the possible values of the PLL State.

5.2.1.2.4 Input Clock Stopped Indicator

The DP PLL provides an additional status indicator that is set if the ADC clock stops toggling.

5.2.2 Data Phase Aligner

A data phase aligner block on each channel performs the fine data de-skew. Again, this is a completely automated process with no effect on the timing of the ADC sampling, and no user control is possible. Each Data Phase Aligner provides two status outputs, as shown in Figure 5-4.

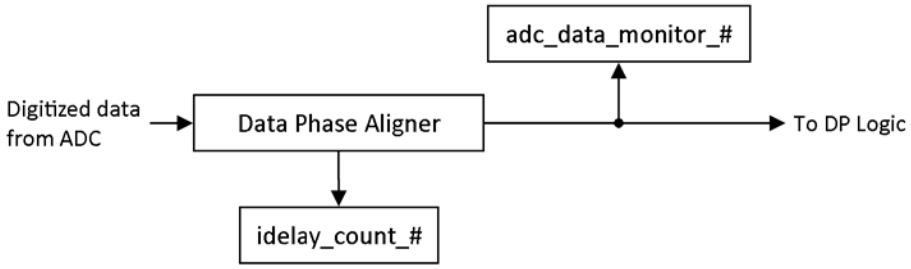


Figure 5-4: ADC phase aligner block diagram.

5.2.2.1 IDelay Count

The idelay_count_# registers report the de-skew offset applied to the digitized data received from the ADC. This value will tend to be higher for middle channels and lower for outside channels. The de-skew applied to the data is equal to the value of the idelay_count_# register multiplied by 78ps.

5.2.2.2 ADC Data Monitor

The adc_data_monitor_# register allows the user to spy directly on the live ADC data. The lower 14 bits of the register contain one sample from the ADC. Bits 14 and 15 report the state of the over range flag as sampled on the rising and falling edge of the clock. These bits should always be equal, either both '1' or both '0'.

6. General DAQ Controls

This chapter explains basic channel controls and features that do not fall under any of the major DP functions of triggering, amplitude, pile-up, timing, or event readout.

6.1 DP Logic Enable

The dp_logic_enable control bit in the master_logic_status register, serves as a master enable for all channels. When the dp_logic_control bit is clear all channels are disabled. Events that were generated while the device was active may still be read out after clearing this bit. The intent of this control is to provide a single bit to enable or disable data acquisition in the module.

6.1.1 NOvA Synchronous Start

The SSP provide an option to enable digitization synchronously across many modules via the NOvA timing interface. The feature is enabled by setting the nova_start_mode control bit in the master_logic_status register. Once this bit has been set the SSP is armed, and the next sync that is receive over the NOvA timing interface will cause the equivalent of setting the dp_logic_enable control bit. To stop the run when operating in this mode the nova_start_mode bit must be cleared.

The NOvA synchronous start will also reset the internal timestamp to 0. Therefore the internal timestamp in all SSPs will run synchronously when this method is used. Additional NOvA sync commands received after starting will not cause additional resets of the internal timestamp. The internal timestamp can then be used as a relative timing base for the event reported during the run. This also results in the synchronization of the periodic timestamp flags, allowing for synchronized triggering during charge injection or calibration runs.

The dp_logic_status bit of the master_logic_status register may be monitored to determine if the module is currently active. The bit is ‘0’ when the module is idle, and is set to ‘1’ when it is active. The bit is set when the module is active, regardless of whether that is caused by setting the dp_logic_enable bit or from a NOvA synchronized start.

6.2 Channel Enable

The channel_enable bit in the channel_control_# registers is used to disable individual channels. Its effect is identical to using the dp_logic_enable bit, but only acts on a specific channel. When a channel is enabled by setting both the channel_enable and the dp_logic_enable (independent of order), the channel will take time to initialize before it can begin processing events. The initialization time is described in more detail in section 8.4.

6.3 Input Inversion

The invert_enable bit in the channel_control_# register allows the polarity of the sampled analog waveform to be inverted. This bit has no effect on the polarity of the SiPM bias. Set the bit to 1 to invert the digitized analog data.

7. Triggering

The chapter discusses how to configure the channel to trigger on events. Triggering is the first step in the data processing sequence. When a channel "triggers" it arms the amplitude and timing logic and prepares the DP to process the event. The trigger can be used to timestamp the event as discussed in section 10.1.3; however, normally event timing information is obtained from the constant fraction discriminator (CFD). The triggering logic in the SSP consists of two primary components: the external trigger logic and the leading edge discriminator (DISC). While these components may be used in many combinations, typical configuration examples are given at the end of this chapter in section 7.3. Figure 7-1 shows a generalized overview of the event processing flow within the SSP. This chapter discusses the rectangular blocks.

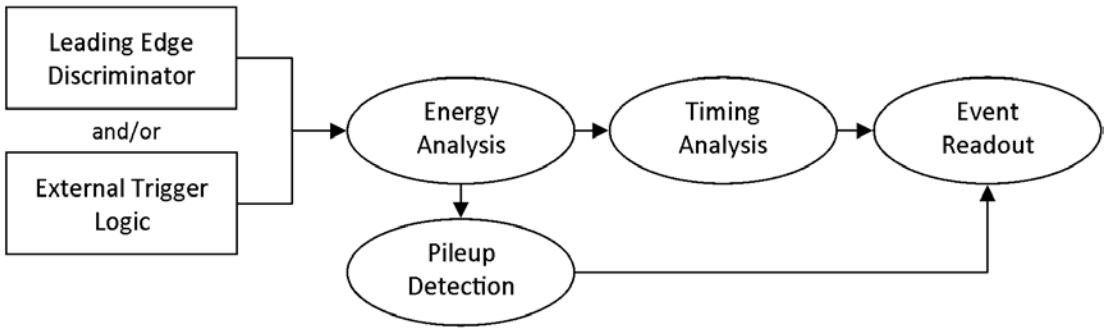


Figure 7-1: Triggering and event processing overview.

7.1 External Trigger

The external triggering logic allows the user to trigger the channel using an external trigger source connected to the front panel of the SSP. This input may either be used to directly trigger the channel or to initiate a gate for the DISC. The channel may also be triggered at a fixed rate by one of the timestamp flags. An expanded block diagram of the triggering logic is shown in Figure 7-2.

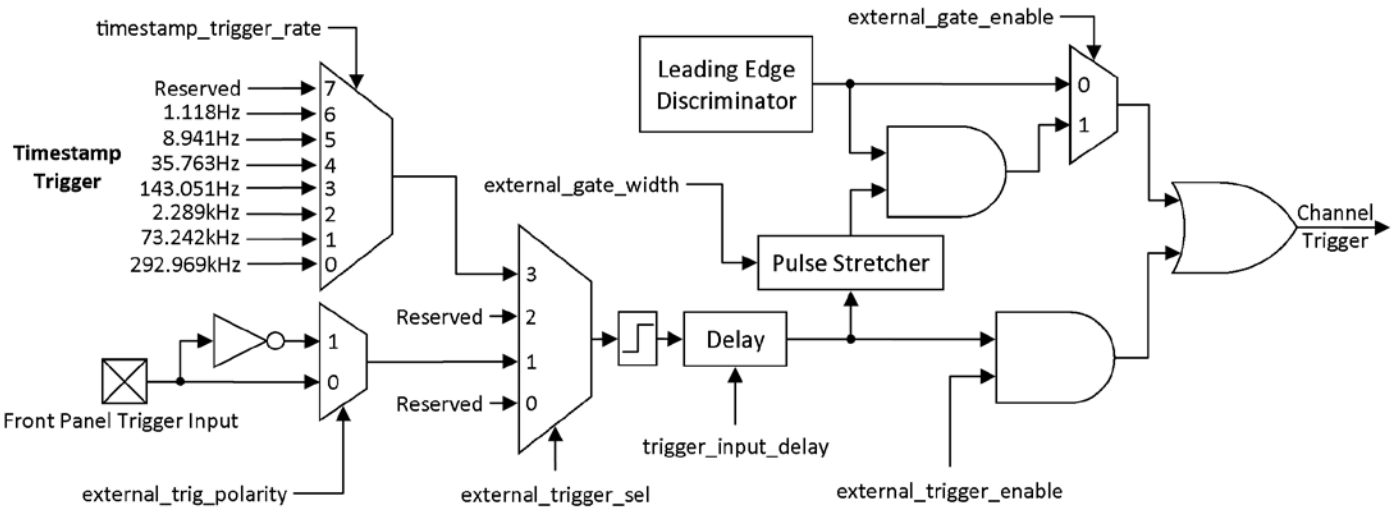


Figure 7-2: Block diagram of triggering logic.

All parameters are uniquely configurable for each channel with exception of the external_trig_input and the extertnal_gate_width.

7.1.1 External Trigger Sources

There are two possible sources for the external trigger. The external trigger input source is selected by the external_trigger_sel bit-field in the channel_control_# registers. Setting the external_trigger_sel to 1 selects the front panel, while setting it to 3 selects the one of the periodic trigger sources generated from the timestamp flag generator. All other settings are reserved for future use.

7.1.1.1 Front Panel

The front panel trigger input is provided to the SSP as a NIM, TTL, or LVTTL signal, depending on the configuration of the front panel. The external trigger must be connected to the LEMO connector labeled Trig In. As illustrated in Figure 2-3, the outer conductor is connected to ground within the SSP module. The front panel trigger input is terminated to ground via a 49.9 ohm resistor. See section 2.1.2 for the full electrical specification and pin-out.

The external_trig_polarity bit in the front_panel_config register allows the front panel input to be inverted if necessary. Setting the bit inverts the input.

7.1.1.2 Timestamp Trigger

The timestamp triggers are one clock cycle-wide pulses that occur on the rollover a pre-selected number of timestamp bits as described in section 4.2.1.1. This mode allows the channel to be triggered periodically to perform baseline measurements. It may be used in conjunction with charge injection to synchronously trigger on charge injection pulses for calibration purposes. (See section 14.1.2.2 for more details on charge injection triggering.) One of seven charge

injection rates may be selected by the timestamp_trigger_rate bit-field in the channel_control_# registers. Refer to Figure 7-2 for details. These triggers can be synchronized across multiple modules by using the NOvA Synchronous Start described in section 6.1.1.

7.1.2 External Trigger Modes

There are two ways to use the external trigger to trigger; either directly as the trigger, or to initiate a gate signal to the output of the DISC. Direct trigger mode is enabled by setting the external_trigger_enable bit in the channel_control_# registers. Gated mode is enabled by setting the external_gate_enable bit.

In direct trigger mode, the channel triggers on the rising edge of the external trigger input. The edge detection logic requires that the external trigger input be more than 6.667ns wide. When the channel triggers directly from the external trigger input, the amplitude and timing logic will not know the polarity of the pulse. This information is normally provided by the DISC. Because of this, the user must select to have all events that are triggered in this way to be processed as either positive or negative polarity. This is set by the external_trigger_polarity bit in the channel_control_# register. (1 = positive, 0 = negative)

When operating in gated mode, the channel only triggers if the DISC logic detects an event during the gating time as defined by the external trigger. The duration of the gating time is set by the external_gate_width register. The trigger input delay may be used to adjust the position of the gating signal. The external_trigger_polarity bit is ignored for triggers occurring by this method, since the polarity information from the DISC is still forwarded to the amplitude and timing logic.

If the external_gate_enable bit is clear, DISC may be used in parallel with an external trigger source. The external trigger is disabled by clearing both the external_trigger_enable and the external_gate_enable bits.

7.1.3 Timing Considerations

For proper operation of the amplitude and timing logic, the external trigger should be timed to occur prior to the leading edge of the event. The data processing algorithms work best if the external trigger occurs within +/- the d window setting from the start of the pulse. The d window is discussed in more detail in section 7.2.2. This only applies if the external trigger is used to directly trigger the channel.

Typically, the pipelining delays within the ADCs and the DP FPGA result in the digitized data lagging by approximately 32 clocks or 213.33ns with respect to when the analog signal is received at the front panel input. The external trigger will need to be timed properly to acquire the signal of interest.

In gated mode, the typical application will require positioning the gating window, as defined by the external_gate_width register, to coincide with the time-window of interest in the waveform. In general, this window could begin before or after the external trigger with respect the analog waveform.

To facilitate proper timing alignment of the external trigger or external gate, the DP FPGA provides the facility to delay the external trigger input or the digitized data.

7.1.3.1 External Trigger Input Delay

If the external trigger is too early, it can be delayed in 6.666 ns steps via the trigger_input_delay_# register. The external trigger can be delayed by as much as 6.820 μs. When using the external trigger as a gating signal to the DISC, this delay allow the gating window to be moved later in time.

7.1.3.2 Waveform Input Delays

If the external trigger is too late, the waveform data can be delayed in 6.666 ns steps via the p_window_# register. The waveform data can be delayed by as much as 6.820 μs. The waveform input delay can also be used as a course compensation for cable delays. When using the external trigger as a gating signal to the DISC, this delay allow the gating window to be moved forward in time. The load_vals bit of the channel_pulsed_control register must be set for changes the p_window_# registers to take effect. This bit self clears.

7.1.3.3 Verifying Direct External Trigger Timing

Verification of proper timing should be done by examining events. There are two ways to perform this verification. If operating in CFD Bypass mode, the readout of the waveform will be positioned about the point where the channel was triggered. The leading edge of the waveform should occur at the position specified by the readout_pretrigger_# register. If the leading edge occurs after this position then the external trigger is too early. If the leading edge occurs earlier then the external trigger is too late.

If operating with the CFD enabled, the readout of the waveform will be positioned about the point where the CFD triggered. However, the SSP can be configured to provide a marker in the waveform data where the DP saw the external trigger fire. Again, this marker should occur on the leading edge of the event. If the marker occurs before the leading edge then the external trigger is too early. If the marker occurs after the leading edge then the external trigger is too late. See section 11.2.2 for information on enabling waveform timing marks.

7.2 Leading Edge Discriminator

The leading edge discriminator (DISC) is used to trigger the channel on the rising or falling edge of an event. The input signal into the DISC is filtered to help reduce high frequency noise that can cause false triggers. When the difference between two data samples, separated by a programmable number of clocks cycles, exceeds the user-specified threshold, the DISC triggers and the event is acquired for processing. The time between samples can range from 0 to 127

samples. When the DISC is configured properly, the threshold sets the minimum pulse amplitude that the SSP will process. This effectively allows the user to set a lower amplitude threshold. Figure 7-3 provides an overview of the DISC design.

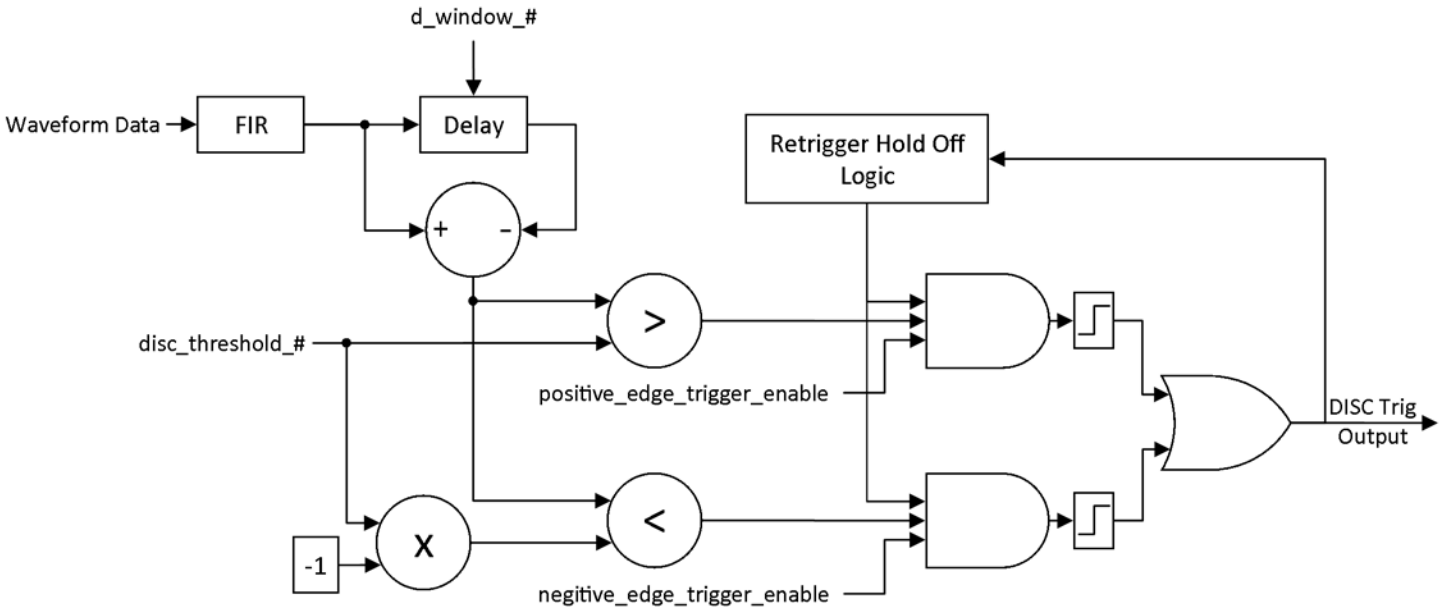


Figure 7-3: Block diagram of the leading edge discriminator.

7.2.1 FIR Filter

The input into the DISC is filtered by a symmetric FIR filter, with a corner frequency of 27.6MHz to reduce high frequency noise that may cause false triggers. The frequency response of this filter is shown in Figure 7-4.

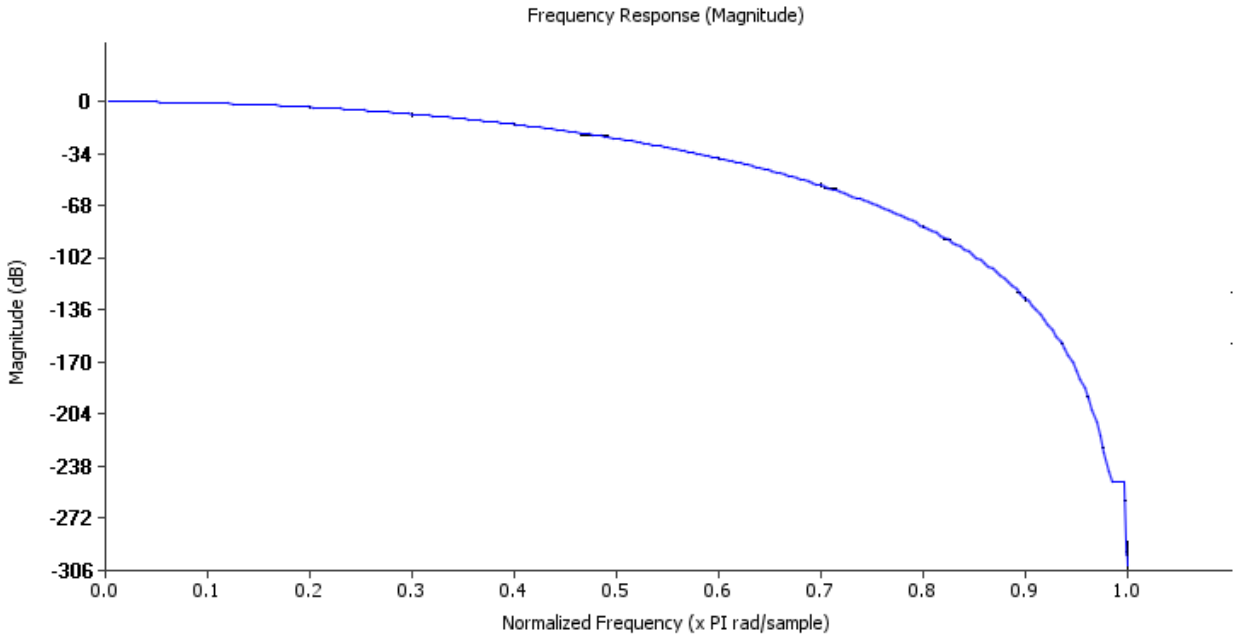


Figure 7-4: FIR filter response.

7.2.2 DISC Tap separation (D Window)

For proper operation the d_window_# register should be set to a value that is at least as long as the rise/fall time of the leading edge of the waveform. This value affects several aspects of the logic. For the DISC, it sets the separation of the taps over which the DISC is taking the difference. If the separation is too short, the leading edge of the waveform will span the delay time and the amplitude of the pulse, as observed by the DISC, will be diminished. Setting the d window value to be about 20% to 40% longer than the actual rise works well in most cases.

Setting the d window to a value much longer than the leading edge of the event will not significantly affect the operation of the DISC. However, it will limit the maximum trigger rate (as discussed in section 7.2.4) and may adversely affect the performance CFD. If timing performance is not critical, or the CFD is bypassed, large d values will loosen the timing tolerance for positing an external trigger in time (See section 7.1.3). Each count of the d window equates to 6.666ns of delay. The separation time is adjustable from 0 to 127 samples.

After changing the values of any window registers, the load_vals bit in the channel_pulsed_control register must be written to '1' in order to apply the new settings. This bit automatically resets back to '0'.

7.2.3 Threshold and Edge Selection

When the d_window_# register is set to be at least as long the leading edge of the event, the led_threshold_# register effectively sets the minimum event amplitude that will be processed by the DP. The DISC can trigger on positive edges, negative edges, or both. Setting the positive_edge_trigger_enable bit in the channel_control_# register enables positive edge triggers. Setting the negitive_edge_trigger_enable bit in the channel_control_# register enables negative edge triggers. To disable the DISC clear both edge enable bits. The DISC reports the polarity of each trigger in the event header.

7.2.4 Retrigger Hold Off

After the DISC has triggered, it is disabled for a time equal to the d_window setting. Both positive and negative triggers are disabled during this time. This prevents triggering multiple times on the same event. Also when two events occur within one d window of each other, the amplitude and timing logic is not able to distinguish/process them as separate events. Figure 7-5 shows an example of two events that are too close to be processed separately. Figure 7-6 shows two events that will be detected as separate events for the given d window setting of 10.

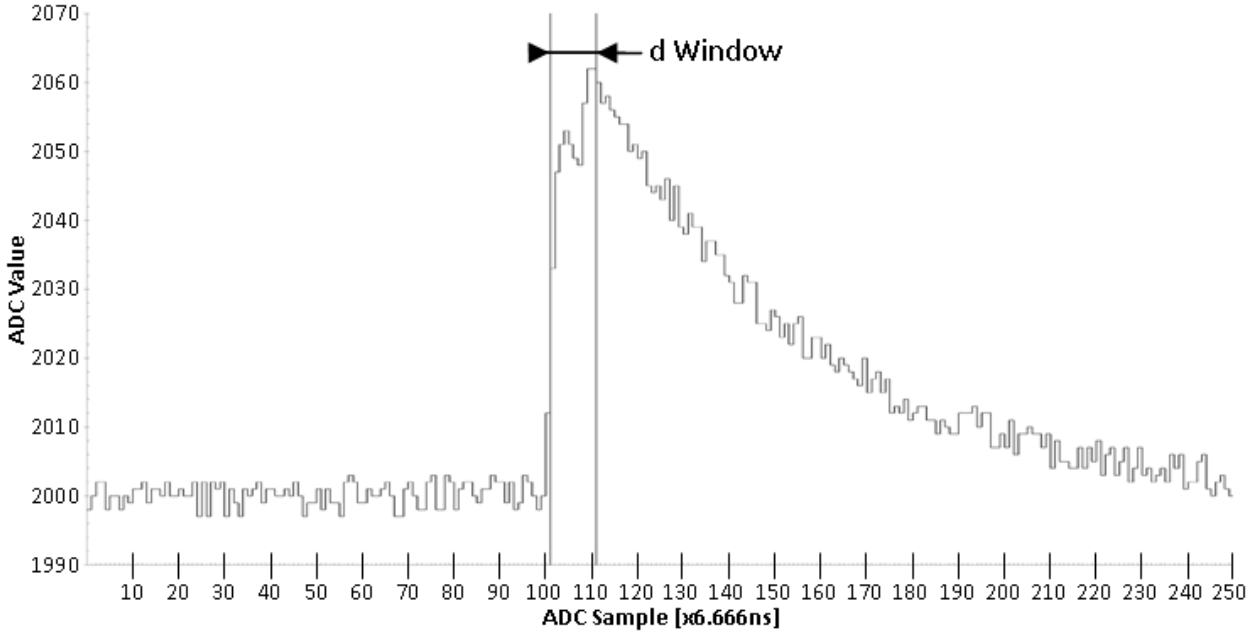


Figure 7-5: Example of two events occurring with less than d separation. These are not detected as separate events and will be processed as a single event by the SSP. (d = 10)

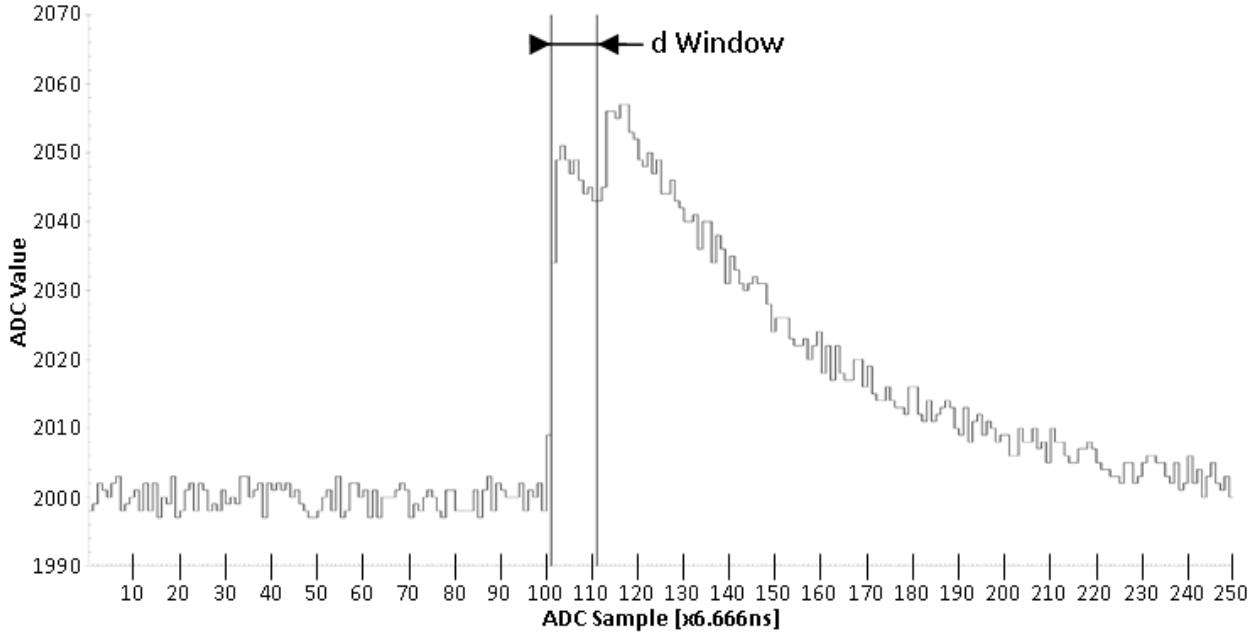


Figure 7-6: Example of two events occurring with more d separation. The SSP will recognize and process these as separate events. (d = 10)

Since events occurring within d of each other are indistinguishable to the DP they will not be marked as pile-up. For more detailed information on pile-up detection refer to section 9. Although not shown in Figure 7-3, the retrigger hold off logic also applies to the external trigger.

7.3 Typical Configuration Examples

There are several ways to configure the triggering logic in the SSP. The following lists a few of the most typical configurations:

A: "Positive Edge Trigger Mode" – Trigger on positive going events with amplitudes greater than a specified threshold.
B: "Negative Edge Trigger Mode" – Trigger on negative going events with amplitudes greater than a specified threshold.
C: "Direct External Trigger Mode" – Trigger the channel directly from an external trigger source.
D: "Externally Gated Trigger Mode" – Trigger on positive and/or negative going events greater than a specified threshold that occur within a specified time window of an external trigger input.
E: "Timestamp Trigger Mode" – Trigger the channel periodically at a known rate.

Table 7-1 below provides the settings of the trigger mode control bits needed to achieve the example trigger modes listed above.

Mode	positive_edge_trigger_enable	negitive_edge_trigger_enable	external_trigger_enable	external_gate_enable	external_trigger_sel
A	1	0	0	0	X
B	0	1	0	0	X
C	0	0	1	0	1
D	0 or 1	0 or 1	1	0	3
E	0	0	0	1	1
X="Don't Care"					

Table 7-1: Typical triggering configurations.

8. Amplitude Analysis

Amplitude analysis is the second step in the data processing sequence, as shown in Figure 8-1.

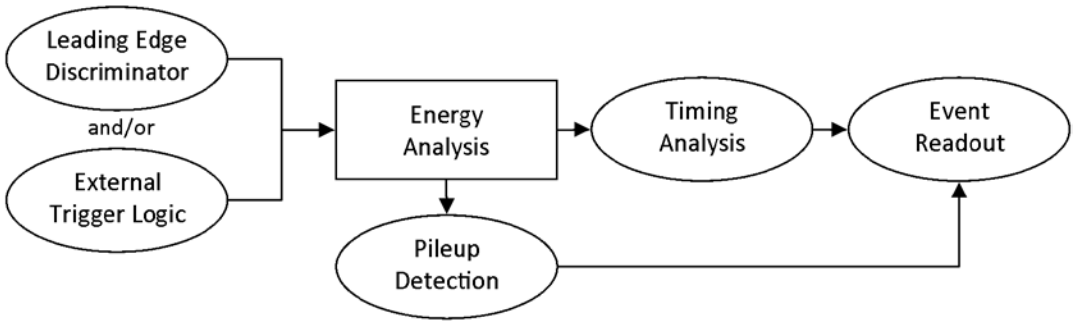


Figure 8-1: Triggering and event processing overview.

All amplitude parameters reported by the DP are either summations of waveform samples or a difference between two summations. The event amplitude is characterized via four customizable parameters, calculated by the DP:

1. Peak Sum or Peak Difference (user selectable)
2. Baseline Sum
3. Integrated Sum
4. Pedestal Value

Figure 8-2 provides an overview the amplitude processing logic. Figure 8-3 show a typical waveform and the relative locations of the summation window boundaries.

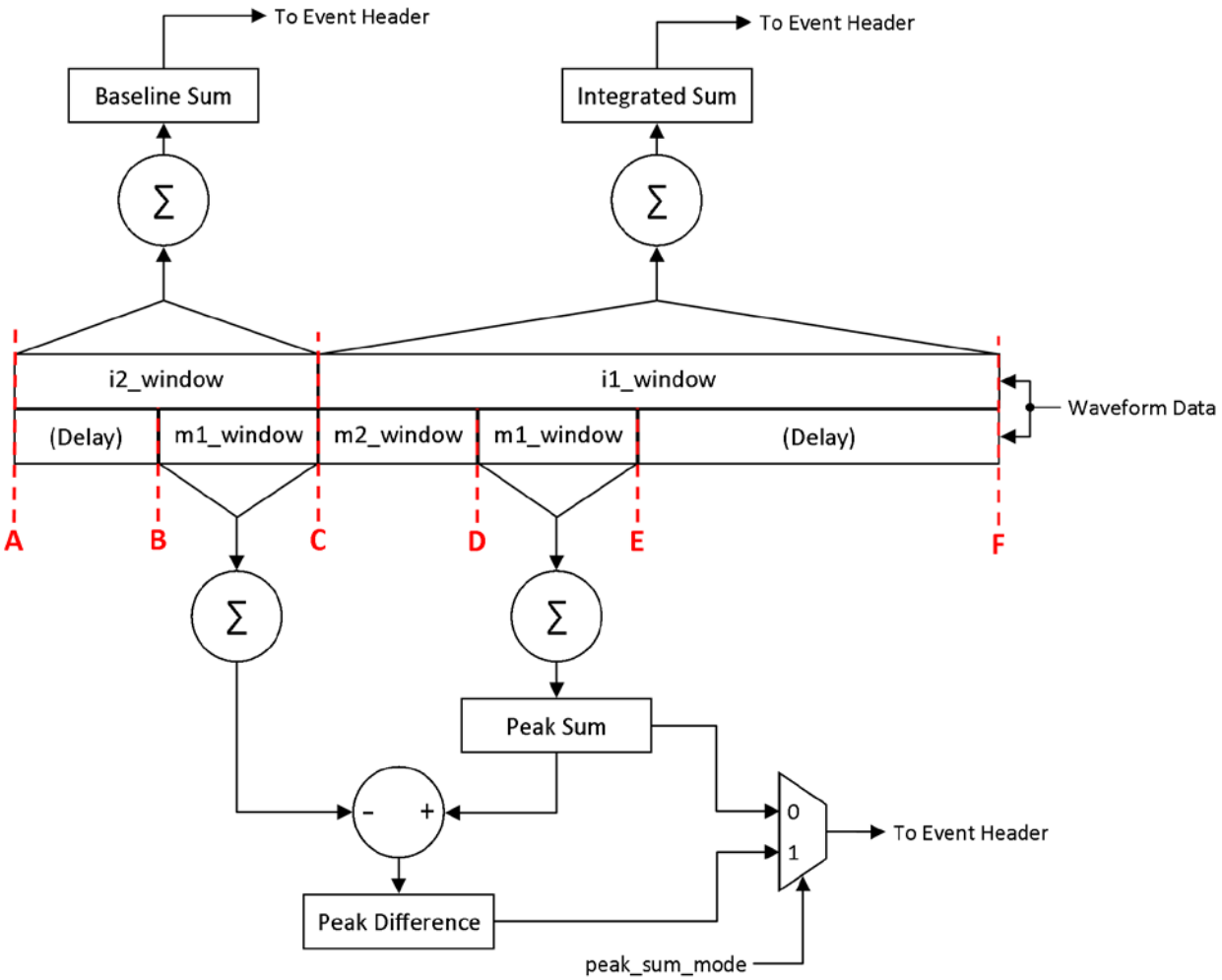


Figure 8-2: Overview of amplitude processing logic.

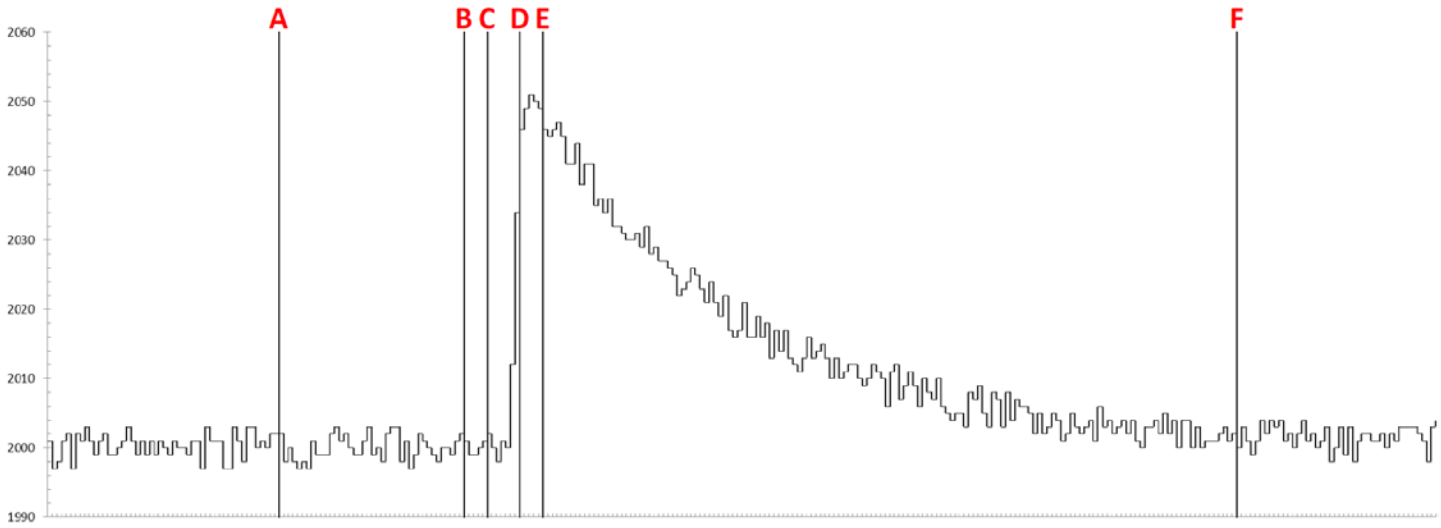


Figure 8-3: Example waveform showing key timing points (m1_window = 5, m2_window = 7, i1_window = 150, i2_window = 40).

8.1 Peak Sum/Peak Difference

The SSP finds and reports information regarding the peak amplitude of each event. The peak of the waveform is found by continuously taking the difference between two equally sized summations of consecutive waveform samples, as shown by the sections of waveform highlighted in green in Figure 8-4 below. The widths of the summations are defined by the m1_window_# register. These summations may be set from 1 to 1023 samples. The separation of the summations is defined by the m2_window_# register. This is shown in purple in the figure below. This value may be set from 0 to 127 samples. Typically, m2 is set to be as at least as long as the leading edge of the events.

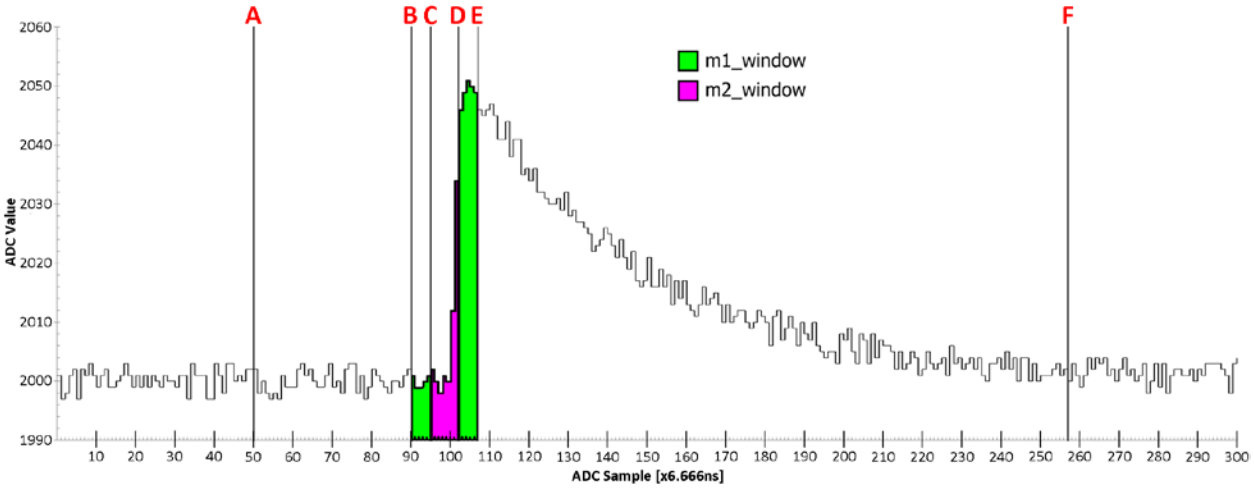


Figure 8-4: Peak sum/peak difference timing plot.

For each event, the SSP detects and stores the maximum difference between these two summations. The peak_sum_mode bit in the channel_control_# registers sets whether the DP reports the peak difference between the two summations or the peak sum of just the second summation. If the peak_sum_mode = 0 then the value reported in the header is:

$$A_{sum} = \sum_{i=D}^{E} x_i$$

Where x_i are the digitized ADC values. Otherwise, if peak_sum_mode = 1 the peak value is reported as:

$$A_{sum} = \sum_{i=D}^{E} x_i - \sum_{j=B}^{C} x_j$$

Where:
(E – D) = (C – B) = m1_window_#
and
(D – C) = m2_window_#.

At the same point that the DP finds the peak sum, the baseline sum and integrated sum are also recorded as described in the following sections. The baseline and integrated sums always have a fixed timing relationship with respect to the position of peak sum. Note that the actual peak value, expressed in ADC counts, is obtained by dividing these sums and differences by the number of samples that went into the sums as defined by the user. This operation is to be done by the DAQ as a post-readout operation.

8.2 Integrated Sum

The integrated sum is defined as the summation of samples starting at a point coincident with the start of the m2 window. Given a properly sized m2 window, this ensures that the integrated sum starts at a point prior to the rising edge of the signal, as shown in Figure 8-5.

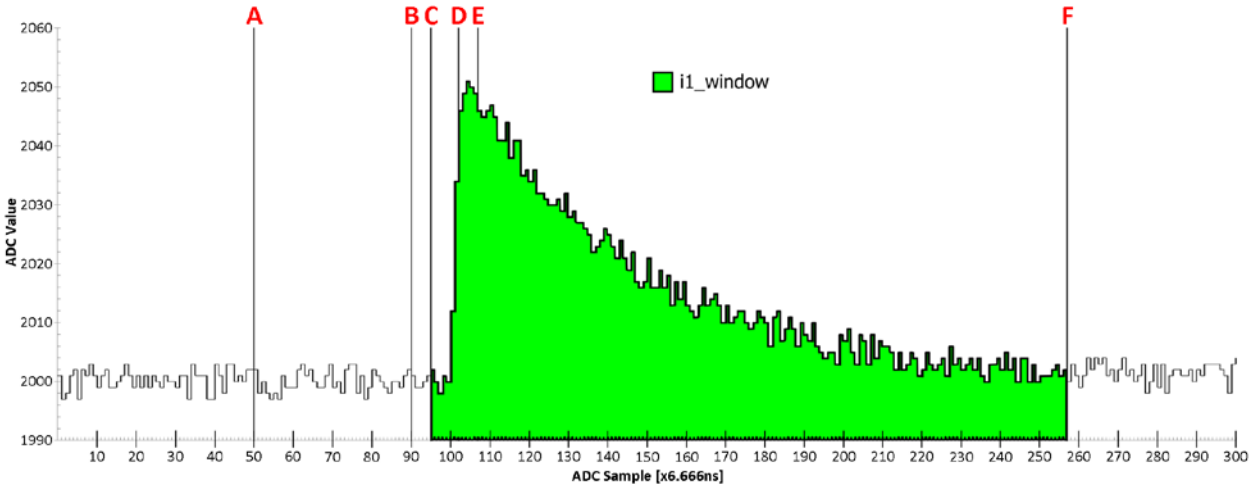


Figure 8-5: Integrated sum timing plot.

The width of the integrated sum is set by the i1_window and may be set from 1 to 1023 samples. Note that the actual integral is obtained by multiplying the sampling period, 6.666 nS. Again, this operation is done by the DAQ as a post-readout operation.

8.3 Baseline Sum

The baseline sum provides information regarding the baseline level of the signal at the moment just before the event. It accounts for any offset voltages that might be present due to the either the electronics, or low frequency leakage current from the detector, or signal residuals due to a previous pulse. This information can be used to subtract off the signal baseline from the integrated sum value to calculate the integrated signal from the event. The baseline sum is defined as the summation of a samples ending at a point coincident with the end of the first peak summation, as shown in Figure 8-6.

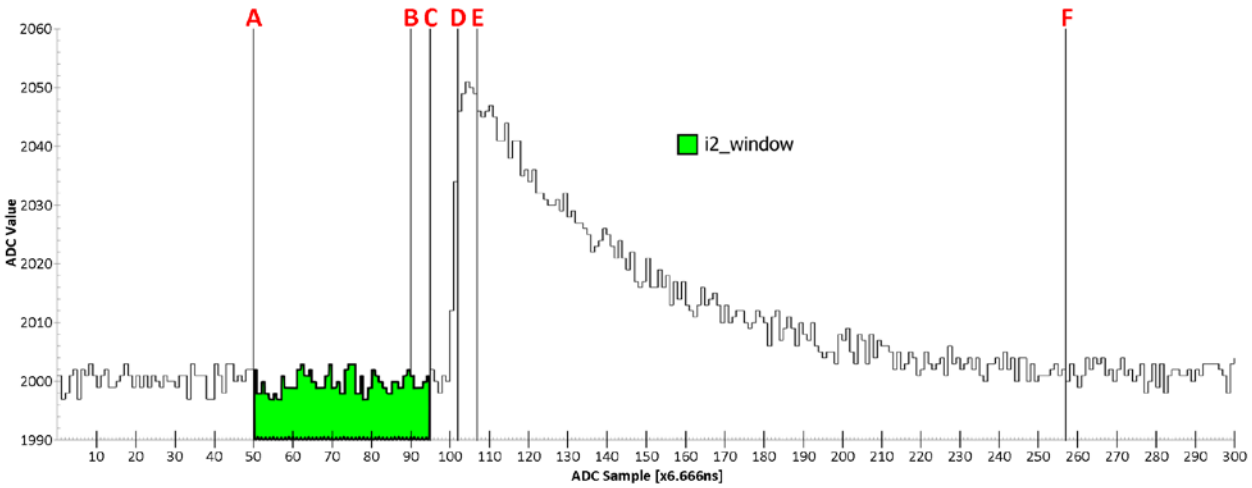


Figure 8-6: Baseline sum timing plot.

Given a properly sized m2 window, this ensures that none of the leading edge of the event enters into the baseline summation. This also means the end of the baseline summation and the beginning of the integrated sum always occurs on consecutive waveform samples. The width of the baseline summation is defined by the i2_window_# register. The value may be set from 1 to 1023 samples. Note that the baseline value, expressed in ADC counts, is obtained by dividing this sum by the number of samples that went into the sums as defined by the user. This operation is to be done by the DAQ as a post-readout operation. This operation should be done before using it as a subtraction operation as the baseline, since the number of samples that go into the baseline sum may be different than the number of samples that go into the integrated sum.

8.4 Load Delays

Changes to the m1_window_#, m2_window_#, i1_window_#, i2_window_#, p_window_# and d_window_# registers do not immediately take effect on the Data Processor logic. The load_vals bit in the channel_pulsed control register must be set to '1' to apply any changes to these registers. This bit automatically resets back to '0'. When the bit is set to '1' all channels are reset and re-initialized. The initialization process typically takes several microseconds to complete and can be calculated by taking the longer of two equations below:

Initialization time = (p + 4d + i1 + m1 + 1048) * 6.666 ns
or
Initialization time = (p + d + i1 + i2 + 2 * m1 + m2) * 6.666 ns

Readout of events may continue during this process. Writing the load_vals bit loads and resets all channels.

8.5 Pedestal Value

In addition to the baseline sum, which provides information on the level of the signal just before the event, the DP also provides another measure of baseline called the pedestal value. Use of the pedestal value provides an effective means for monitoring the offset voltages of a channel in time, as well as to compare the baseline sum to assess possible corruption of an event due to pile-up. The Data Processor measures the pedestal of the signal only when the channel has been idle for a period that is user-programmable, which would typically be set to five or more decay time constants of the signal. When no event occurs for the set hold off time, the pedestal measurement begins. The pedestal value is measured by passing waveform samples through a dedicated digital filter that has a very long effective time constant. The resulting measurement is reported in units of 0.25 ADC counts/LSB. When an event is detected by the DISC, pedestal tracking is suspended far enough in advance to prevent corruption of the pedestal value by the event waveform. The operation is equivalent to taking an average over a long period. In the current firmware the pedestal value is only valid when using the leading edge discriminator or trigger the channel, and the threshold is set appropriately.

8.6 Pulse Pile-up Considerations

If two events are closer than m1 samples from one another, the amplitude analysis logic may not have sufficient time to find the peak of the first waveform prior to being interrupted by the second event. The SSP detects this type of pile-up condition as an “m-type” pile-up, as described in the next chapter. This condition is reported in the event header. This can only occur if the d window is set smaller than m1 window.

When the amplitude analysis logic is interrupted by an m-type pileup condition the integrated sum and baseline sum of the first event are typically not strongly affected, unless either the i1_window is set short relative to the decay time of the pulse or the m1_window is set long relative to the rise time of the pulse. The reason for this is that the m1_window effectively defines the maximum mis-alignment that can occur during an m-type pileup. Looking at figures Figure 8-5 and Figure 8-6, one can see that if the pulse is shifted to the right by up to m1 samples, which is se tot 5 in these examples, that the integral of the shaded area is not significantly affected. Of course, the integrated sum as shown by Figure 8-5 will report the integration of both pulses.

However, the peak sum/peak difference is strongly affected by misalignment that can occur in an m-type pileup, due to the small number of samples in the sum, which makes them position sensitive. If however, the d window is set larger than the m1 window, m-type pileups will never occur since the d window sets the minimum (firmware) resolvable pulse pair.

9. Pile-up

Pile-up detection occurs in parallel to the amplitude analysis. A flowchart of the algorithm is shown in Figure 9-1.

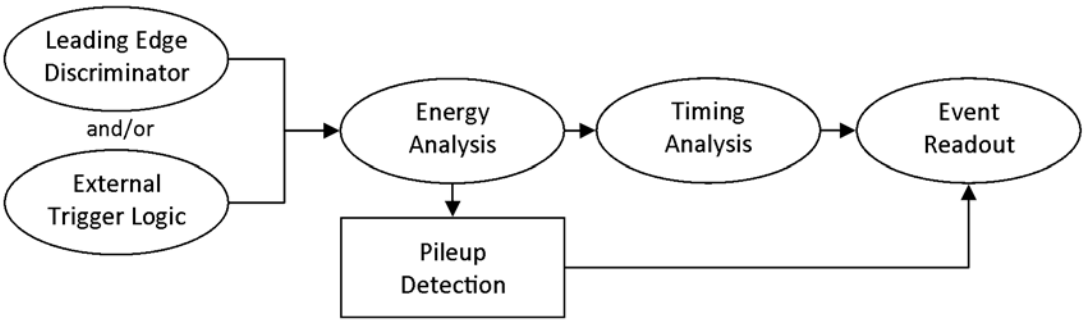


Figure 9-1: Triggering and event processing overview.

9.1 Pile-up Detection

The SSP provides two indications of pile-up which are reported in the event header. The first is the i-pile-up flag, which is set when two events occur within the i1 window time from one another. The second is the m-pile-up flag, which is set when two events occur within the m1 window time from one another. If either condition is true then the event is considered to have pile-up. For illustration Figure 9-2 shows an example event with the i-pile-up detection time shaded in green.

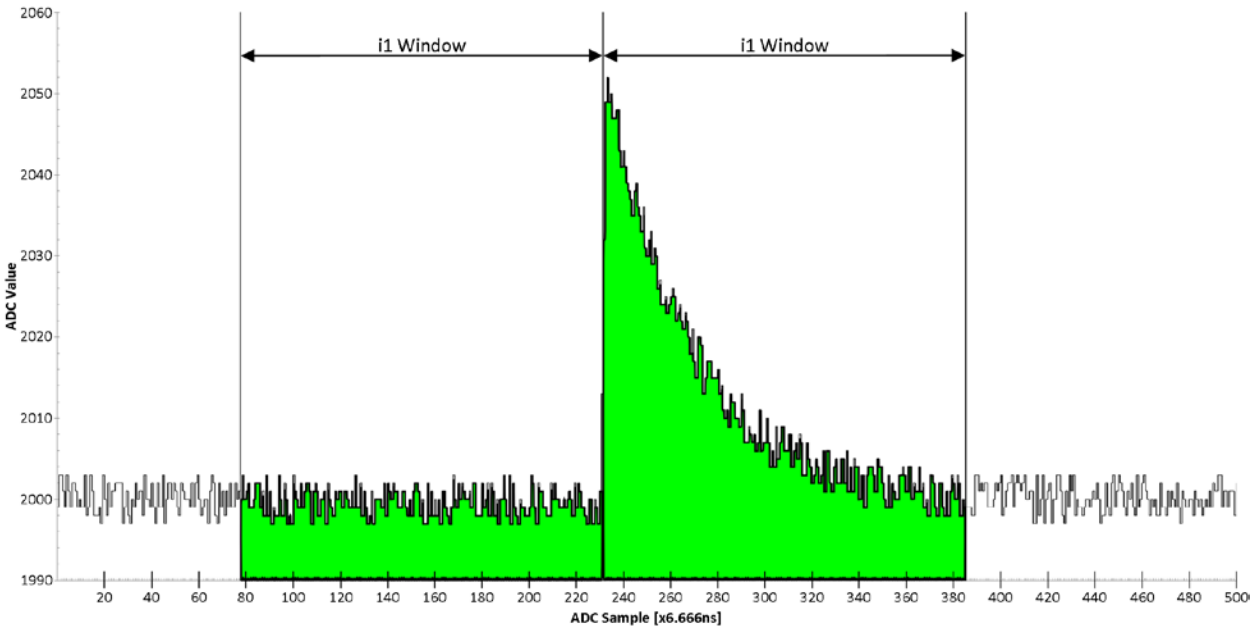


Figure 9-2: I-pile-up detection window. (i1 = 200)

The m-pile-up detection window is similar, in that only the width of the window is different; being dependent on m1_window_# instead of the i1_window_#.

An i-type pile-up indicates that the integrated sum value has been affected by another event that was detected within the i1 window of the event of interest. An m-type pile-up indicates that the event was close enough to another event that the peak difference value will have been strongly affected. When two or more events pile upon one another, the SSP also reports which event is the first in the pile-up and which are the proceeding events.

Detection of events is done by the DISC. Events that are of insufficient amplitude to trigger the DISC do not affect pile-up detection.

The position of the mid-point of the pile-up window is set by the position where the DISC triggers. Since the DISC will trigger on the leading edge and the integrated sum begins before the leading edge this will result in the pile-up detection window extending a few clock past the end of the integration window. If another event is detected anywhere within the area shaded in green then both events are marked as an i-type pile-up. Similarly if they occur within m samples of one another they are also marked as an m-type pile-up. Figure 9-3 and Figure 9-4 provide a few examples of what would and would-not be considered to be piled-up events.

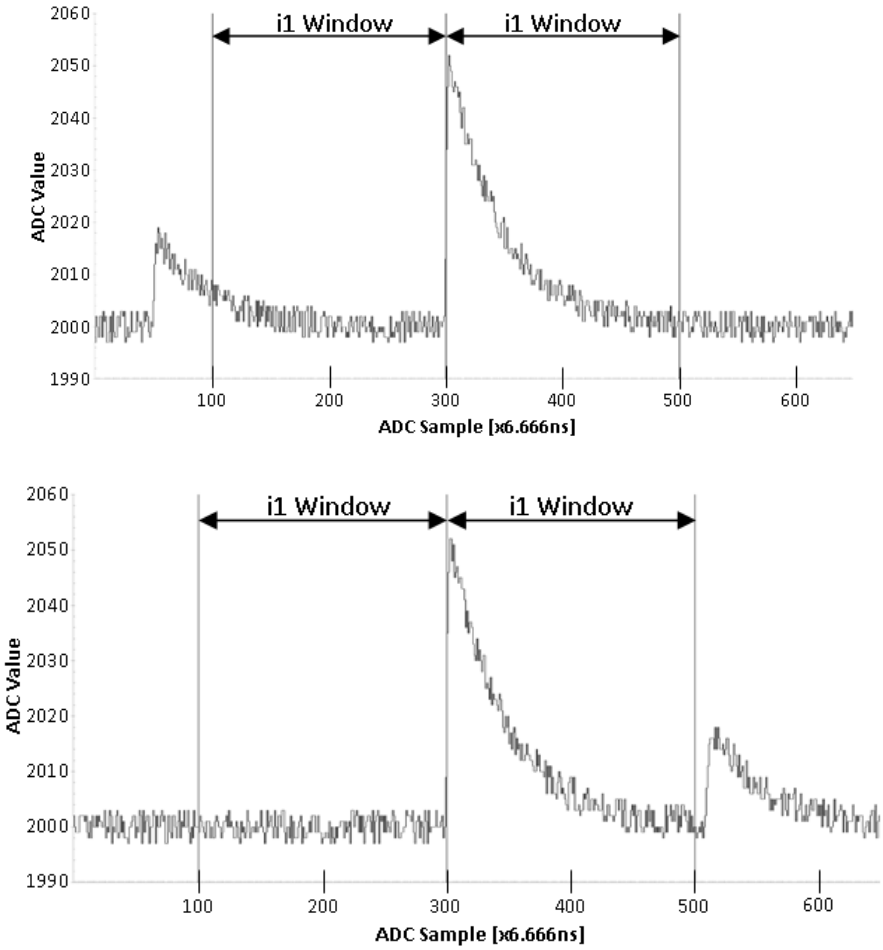


Figure 9-3: Two examples of events that are not considered to be in pile-up, since they lay further than i1 samples from one another. (i1 = 200)

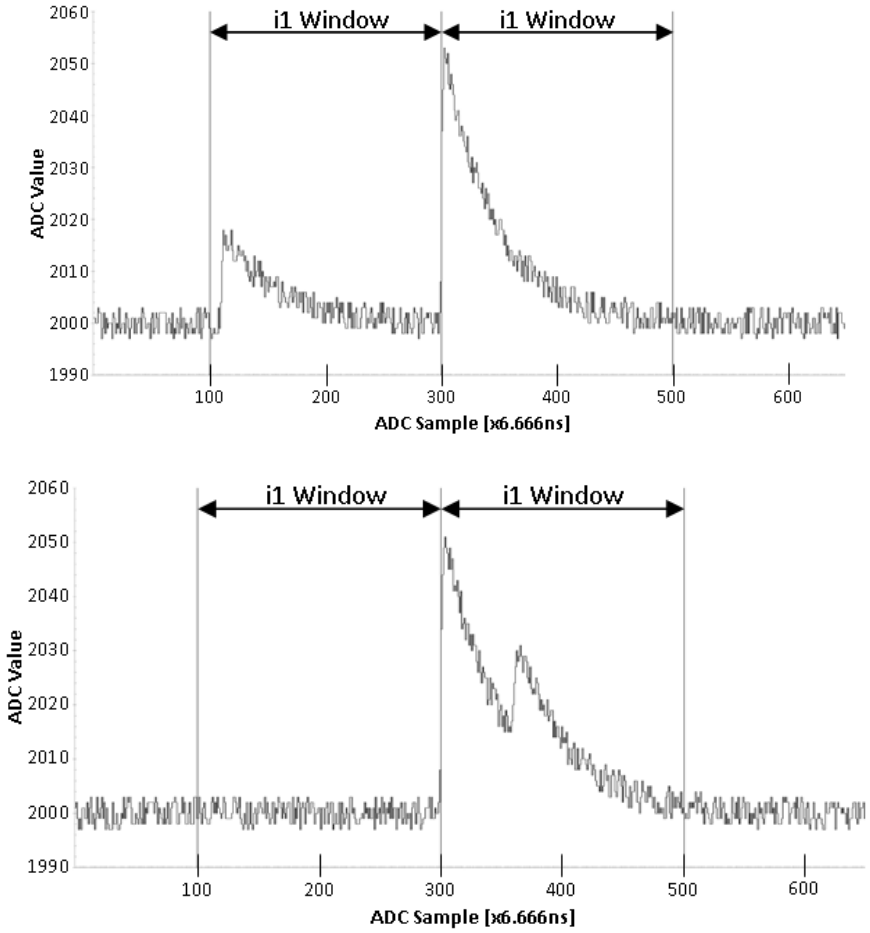


Figure 9-4: Two examples of events that are considered to be piled-up by the DP. (i1 = 200)

When two or more events pile upon one another, the second and the ones that follow are marked as "extended" events in addition to be marked as piled-up. These, extended, events are handled no differently in terms of the event processing, but allow the first event of a series of piled-up event to be easily identified in offline analysis. The pile-up rejection logic also allows for selective waveform readout of events depending on if they are the first event in a pulse pile-up.

9.2 Pile-up Suppression

The SSP provides several methods of suppressing pile-up.

- The SSP can drop all piled-up events.
- The SSP can drop all extended events.
- The SSP can suppress waveform readout for all non-piled up events.

Each of these options has an associated control bit within the channel_control_# registers. While these control bits can be used in any combination, the following section provides some typical configurations.

The pileup_rejection_enable bit controls if any piled-up events are processed. If pileup_rejection_enable = 0, all events that are marked as pile-up are dropped from readout. Otherwise, the readout behavior of piled-up events depends on the settings of the pileup_extend_enable and pileup_waveforms_only bits.

The pileup_extend_enable bit controls if all events that are marked as pile-up are allowed to be readout or if only the first event of a pulse pile-up can be readout. If pileup_extend_enable = 0, then only the first event of a pulse pile-up is processed for readout. Otherwise, all events are readout.

The pileup_waveforms_only bit controls if waveforms are readout for all events or only for piled-up events. If pileup_waveforms_only = 1, then only piled-up events will give waveforms. Otherwise, all events may have waveforms.

9.2.1 Typical Configuration Examples

Table 9-1 provides a few typical configuration examples of the pile-up suppression logic.

	pileup_rejection_enable	pileup_extend_enable	pileup_waveforms_only
Process only clean events.	1	X	0
Process all events.	0	0	1 or 0
Process only clean events and leading events of pulse pile-ups.	0	1	1 or 0
X="Don't Care"			

Table 9-1: Typical pile-up suppression configurations.

10. Timing

The SSP provides three timing parameters to characterize the timing of the event.

1. Event Timestamp
2. Timestamp Interpolation Points
3. Peak Offset

The timestamp and timestamp interpolation information is provided by a constant fraction discriminator (CFD), whereas the peak offset is provided by the amplitude analysis logic

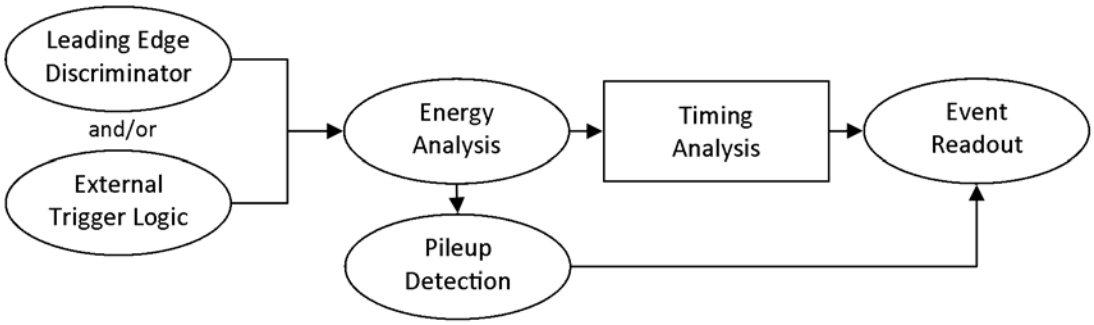


Figure 10-1: Triggering and event processing overview.

10.1 Constant Fraction Discriminator

Each Data Processor implements a CFD to obtain timing information on the event. The CFD operates use of a sliding summation between 1 and 127 samples which is set by the value of the d_window_# register. The start of the sliding summation is defined as the point the DISC triggers minus d samples, as shown in Figure 10-2 below.

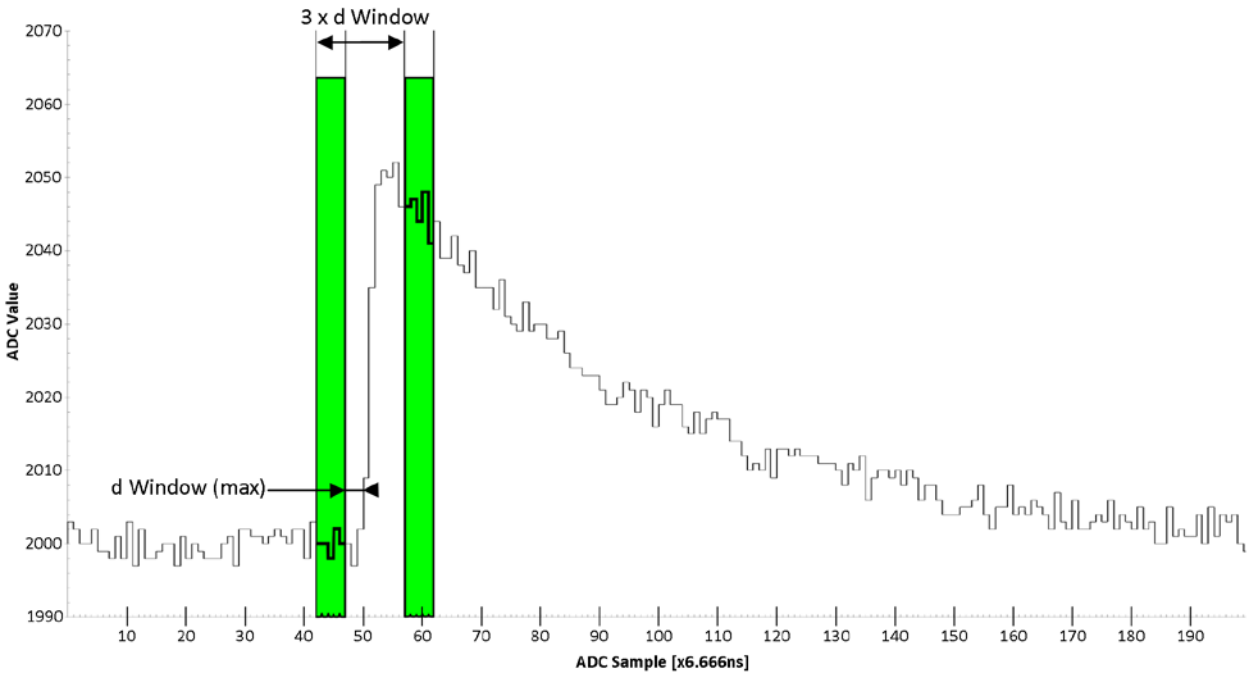


Figure 10-2: Sliding CFD summation. (d=5)

Just as with the DISC, it is important that the d_window_# value be set no shorter than the rise/fall time of the leading edge of the signal. The summation slides along the waveform from a position just ahead of the rising edge to a position past the peak. The DP finds the minimum and maximum values of the sliding sum, and marks the timestamp of the event at the point in which the summation is at a set fraction of the two. This defines the CFD equation, and provides for consistent time stamping of all irrespective of the event's amplitude or rise time. The CFD fraction is set by the cfd_fraction_# register, as defined below:

CFD fraction for Channel # = cfd_fraction_# / 8192

In addition to the event timestamp the CFD provides four additional data points, called "Timestamp Interpolation Points." The timestamp interpolation points represent the difference between the CFD summation and the CFD trigger threshold for that event. The CFD trigger threshold is determined by the CFD fraction and the min / max summation over d samples for each event. The DP reports the two values of the CFD equation on either side of the threshold crossing. Figure 10-3 show a plot of the CFD equation for the example waveform on the previous page. The black points represent the CFD timing points reported in the event header.

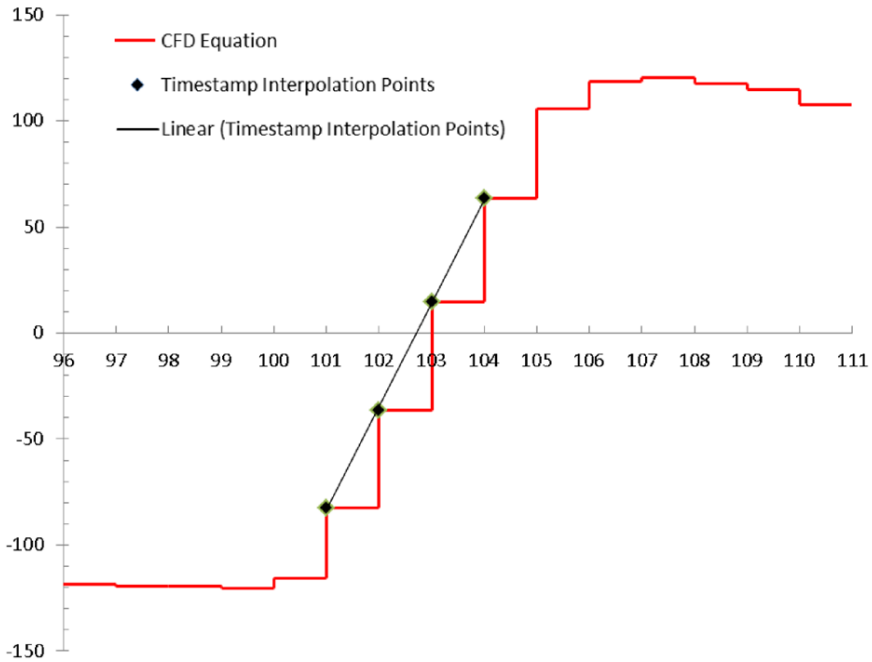


Figure 10-3: CFD equation, showing timing points and linear interpolation. (d=5, CFD fraction=50%)

The point at which the CFD equation cross zero marks the timestamp of the event, 103 in this example. However, by performing a linear interpolation of the CFD points one can obtain sub-clock timing accuracy. This is gives approximately 102.7 for the above example. In a future firmware release the DP will perform this interpolation within the FPGA, to reduce the size of the event header.

10.1.1 Significance of χ^2 for Timestamp Interpolation Fit.

In general, the RMS error or χ^2 of the linear fit to the CFD timestamp interpolation points is not a direct indicator of the accuracy of the timestamp interpolation. It is can be seen in Figure 10-3 that the CFD equation is not linear. However, typically the CFD points in the region of the x-axis crossing for SiPM-like signals will fit closely to a straight line, with a properly sized d window. More importantly, the width of the distribution at any given pulse amplitude of the RMS error or χ^2, rather than its absolute value, will provide some indication of the relative timing uncertainty versus event amplitude.

In addition, an event that has a significantly different RMS error or χ^2 than other events of the same amplitude, may indicate that an undetected pile-up has occurred, where two event occur in less than the d window time of separation.

10.1.2 CFD Valid Status Bit

The CFD report a status bit in the event header to indicate if the CFD timing points are valid. When using the DISC to trigger the channel this bit should always be set. However, when using an external trigger the CFD may report a non-valid status if the timing of the external trigger is

too early or late. See section 7.1.3 for information on how to adjust the timing of the external trigger.

10.1.3 CFD Bypass Mode

The operation of the CFD may be disabled. If the user chooses to bypass the CFD, then the DISC marks the timestamp of the event and no interpolation points will be reported in the event header. CFD bypass mode is enabled by setting the cfd_enable bit of the channel_control_# register to 0. Otherwise, the CFD is enabled.

10.2 Peak Offset

The final point of timing information, provided by the Data Processor, is the peak offset. Peak offset is a value reported for each event that is defined as the time between the event timestamp, as reported by the CFD, and the point at which the amplitude analysis logic finds the peak sum/peak difference. See the amplitude analysis section for more information regarding the peak sum/peak difference.

11. Event Readout

Event readout is the last step in the data processing sequence.

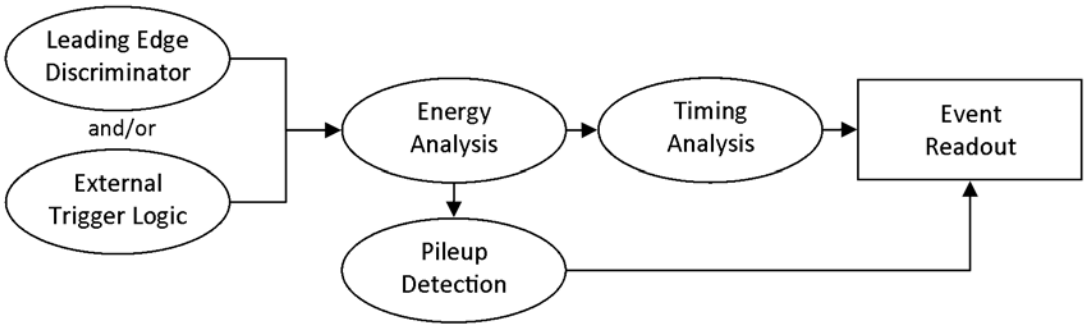


Figure 11-1: Triggering and event processing overview.

After an event has been processed by the Data Processor, it generates an event header followed by a user defined number of waveform samples. Table 11-1 shows the overall structure of each event.

Event Header
Waveform Data

Table 11-1: Event packet structure overview.

The events are collected in a round-robin fashion from all channels and transferred the Comm FPGA for readout by the DAQ system. Each channel implements an event buffer to store up several events should the event data rate momentarily exceed the available bandwidth of the readout interface or the external DAQ system.

11.1 Event Header

The event header allows for the definition of multiple event header formats. However, only one event header type has been defined thus far. The structure of the event header is shown in Table 11-2.

	Bit																													
Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Start of Record Marker																															
1	Reserved								Status Bits				Record Type				Packet Length															
2	Module ID												Channel ID				Reserved															
3	External Event Timestamp 31:0																															
4	External Event Timestamp 63:32																															
5	Peak Offset								Peak Sum/Peak Difference																							
6	Integrated Sum 7:0								Baseline Sum																							
7	Baseline Offset																Integrated Sum 23:8															
8	CFD Point 1																CFD Point 0															
9	CFD Point 3																CFD Point 2															
10	Local Event Timestamp 15:0																Fractional Timestamp															
11	Local Event Timestamp 47:16																															

Table 11-2: Event header structure.

Start of Record Marker – This is a fixed value used to mark the start of an event. This field will always be 0xAAAAAAAA.

Packet Length – Length of the packet in terms of 32 bit words. This includes the length of the header.

Record Type – Type of record. Only one record format has been defined, this field will always read 0.

Status Bits – There are five status bits that are reported for each event.

Bit 20 = I pile-up Flag – Set if an i-type pile-up condition has occurred. (See section 9.1)

Bit 21 = Polarity Flag – If 1 then this is a positive going event. If 0 this is a negative going event. (See section 7.2.3)

Bit 22 = Offset Flag – This bit is set to 1 if the waveform readout was offset due to consecutive event readouts overlapping. For more information see the next section.

Bit 23 = CFD Valid Flag –This is set to 1 if the CFD Points are valid. (See section 10.1.2)

Bit 24 = M pile-up Flag – Set if a m-type pile-up condition has occurred. (See section 9.1)

Channel ID – This field reports the channel number from which the event was generated.

Module ID – The value set in the module_id register. In a system consisting of multiple SSPs this register provides a method of uniquely identifying from which SSP the event originated. This value must be set by the DAQ control system.

External Event Timestamp – When operating using an external clock source this field provides the system synchronized timestamp of the event. (See section 4.2)

Peak Sum/Peak Difference – Holds the peak sum/peak difference value. (See section 8.1)

Peak Offset – Hold the peak offset value. (See section 10.2)

Baseline Sum – Holds the baseline sum. (See section 8.3)

Integrated Sum – Holds the integrated sum. (See section 8.2)

Baseline Offset – Holds the baseline offset value. Units are ¼ ADC count per LSB. (See section 8.5)

CFD Point # - Reports the four timestamp interpolation points. (See section 10.1)

Fractional Timestamp – Reserved for future use.

Local Event Timestamp – Reports the time of the event in terms of the local timestamp. (See section 4.2)

11.2 Waveform Readout

Waveform readout is taken with respect to where the CFD fires, or where the DISC fires in the case where the CFD is bypassed. The length of the waveform readout is set by the readout_window_# registers and may be any even value between 0 and 2046. The position of the waveform within the readout window is set by the readout_pretrigger_# registers. The pretrigger sets the number of samples to read prior to the trigger as shown in Figure 11-2. This may be set to any value from 0 to 2047.

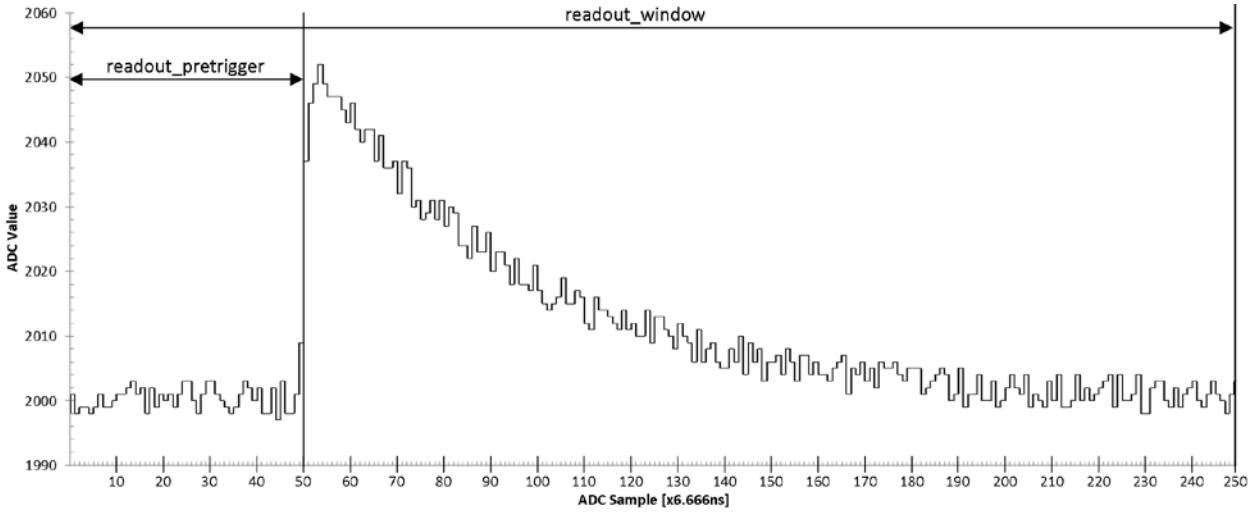


Figure 11-2: Waveform readout parameters. (readout_pretrigger=50, readout_window=250)

The waveform data immediately follows the event header. Each 14-bit sample is packed in to a 16-bit word as shown in Table 11-3.

Table 11-3: Waveform sample format.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISC	CFD	ADC Sample 13:0													

The upper two bits may be configured to mark the location of the DISC trigger and the CFD trigger as described in section 11.2.2. Each 32-bit word of the event record contains two such samples as shown in Table 11-4.

	Bit																													
Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0..11	Event Header																															
12	Waveform Sample 1																Waveform Sample 0															
13	Waveform Sample 3																Waveform Sample 2															
...	...																...															
m	Waveform Sample n																Waveform Sample n-1															

Table 11-4: Waveform data packing order.

The next event record will immediately follow the end of the waveform data.

11.2.1 Waveform Offset

During operation, some events will be positioned close enough to one another, such that their readout windows overlap. Figure 11-3 shows a series of three events, where such a condition exists.

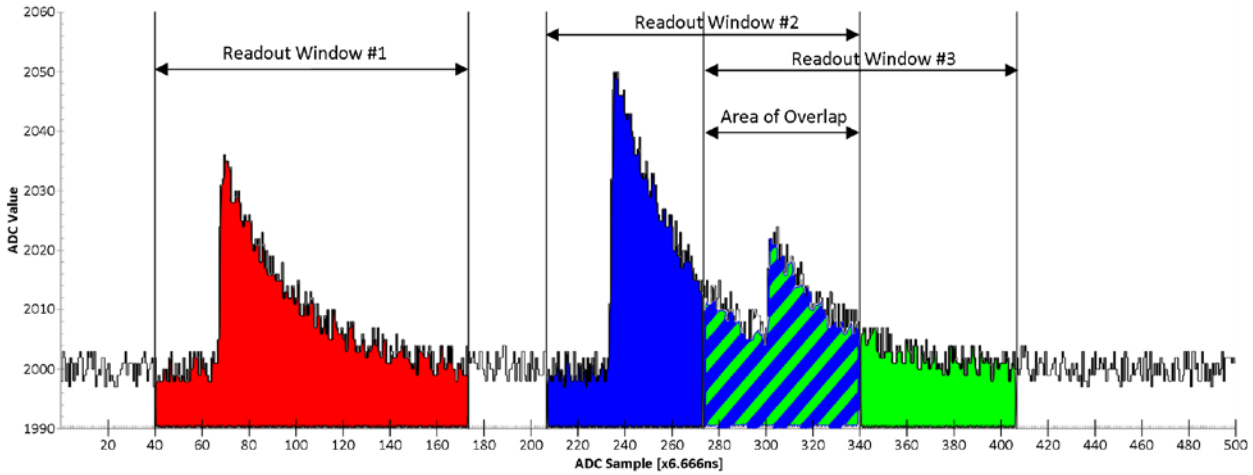


Figure 11-3: Example of overlapping waveform readout windows. (readout_pretrigger=40, readout_window=200)

The SSP can be configured to handle these cases of overlapping readout windows in one of four ways. This is called the offset mode and is set by the event_extend_mode bit field of the channel_control_# register. The available modes are shown in Table 11-5.

Mode	event_extend_mode
Offset Disabled	0
Offset	1
Offset with Truncation	2
Headers Only	3

Table 11-5: Offset modes.

11.2.1.1 Offset Disabled Mode

In offset disabled mode, any event which has a readout window that overlaps with the previous event's readout window is dropped. No event header or waveform will be reported for this event. Figure 11-4 shows how the example from Figure 11-3 would be handled in this mode.

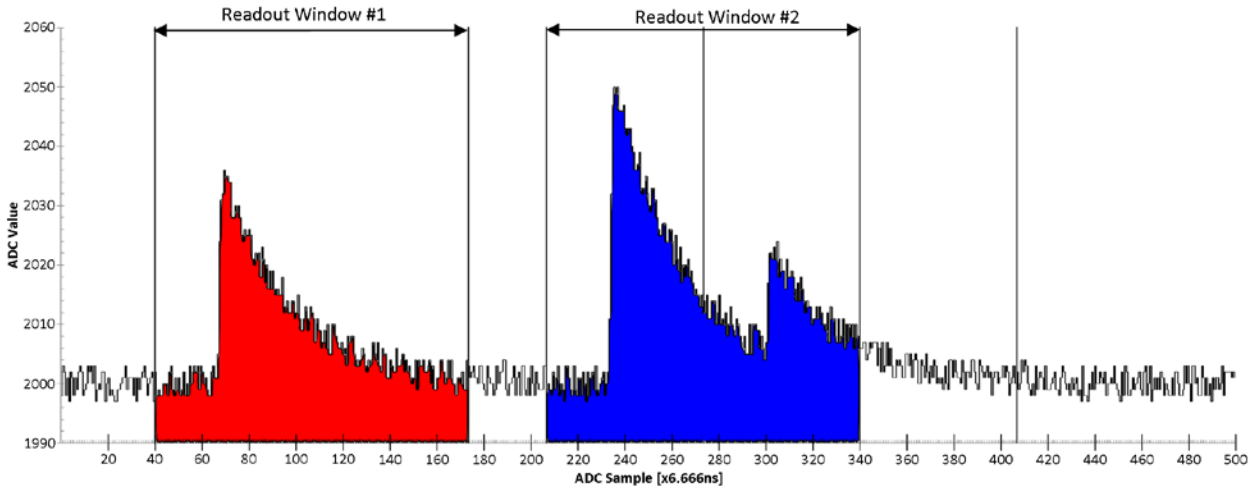


Figure 11-4: Example with offset disabled mode. (readout_pretrigger=40, readout_window=200)

The first and second events are read out normally. However, since the third event's readout window would have overlapped with the second's it is dropped from readout. Since the third event was dropped, if a fourth event were present, it could be readout so long as it did not overlap with the readout of the second.

11.2.1.2 Offset Mode

In offset mode, any event which has a readout window that overlaps with the previous event's readout window is offset in time. When this occurs the offset flag in the header is set to indicate that the readout window was moved. Figure 11-5 shows how the example from Figure 11-3 would be handled in this mode.

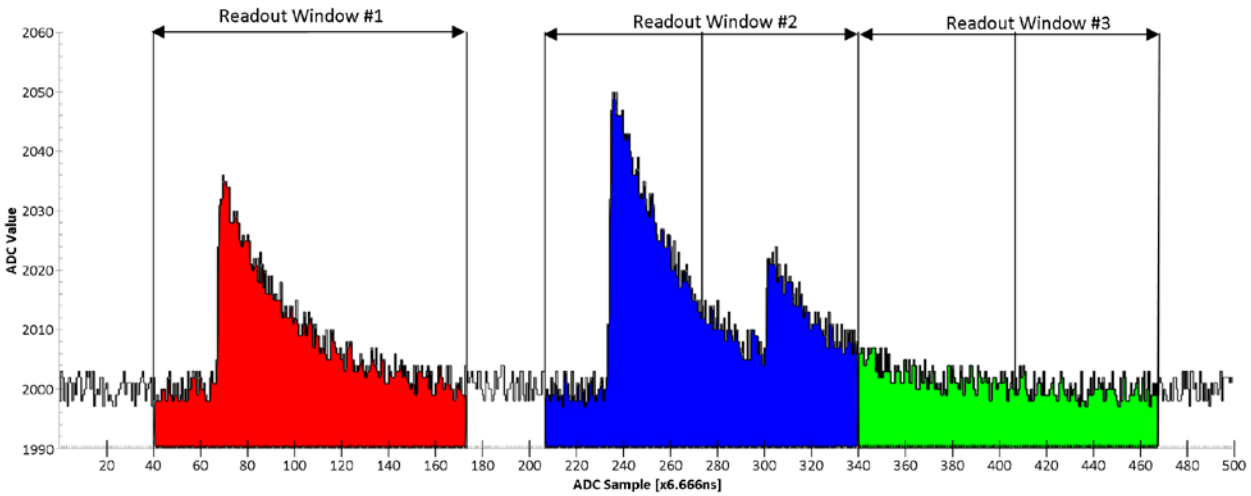


Figure 11-5: Example with offset mode. (readout_pretrigger=40, readout_window=200)

Again, the first and second events are read out normally. However, the third event's readout window is delayed such that it immediately follows the second event. This mode has several disadvantages. First, it causes the readout buffers to be filled with waveform data that would not have normally been readout had the overlap condition not existed. Second, if the SSP has been configured to readout waveforms for piled-up event (section 9.2), a string of piled-up events can generate a very long series of readouts that can, again, fill the readout buffer. This can also cause non-piled up events that proceed shortly after a pulse pile-up to be offset or even dropped, as the readouts for the pulse pile-up complete. This mode is mainly supplied to provide some mode of offset readout, while maintaining a fixed event record length.

11.2.1.3 Truncated Offset Mode

Truncated offset mode is similar to offset mode, except the readout of the overlapping event is truncated to end where is would have ended had it not been offset. Just as with offset mode, the offset flag in the header is set to indicate that the readout window was moved. Figure 11-6 shows how the example from Figure 11-3 would be handled in this mode.

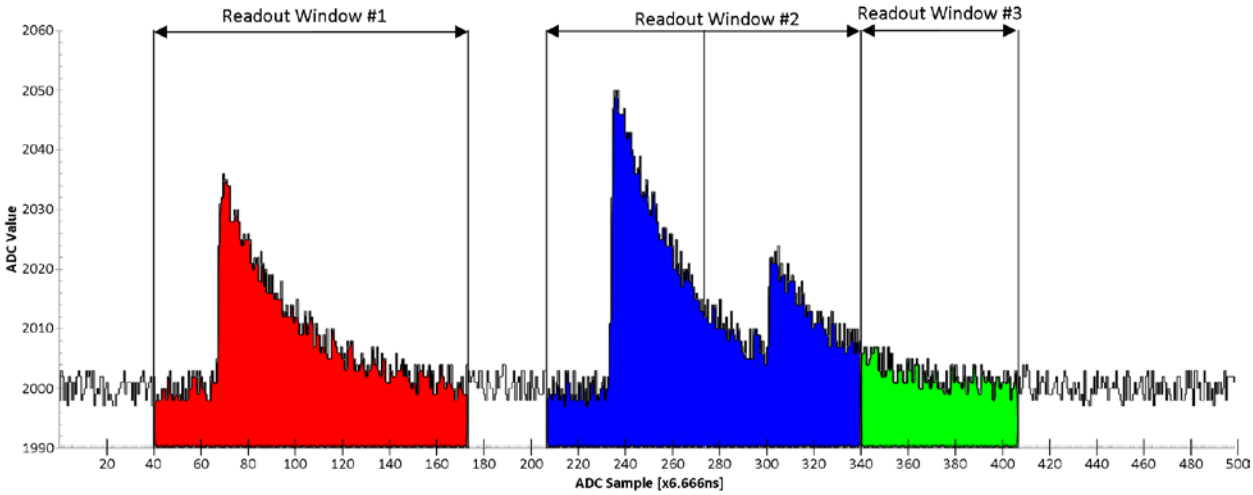


Figure 11-6: Example with offset with truncation mode. (readout_pretrigger=40, readout_window=200)

This will result in variable event record sizes; however, it prevents excessive waveform readout especially in environments where pile-up is common.

11.2.1.4 Headers Only

In headers only mode, no waveform is recorded for any event which has a readout window that overlaps with the previous event's readout window. Only the event header is reported. In terms of the waveform readout this case is identical to the offset disabled mode shown in Figure 11-4.

11.2.2 Timing Marks

Timing marks are injected into the unused bits 14 and 15 of the waveform data, as shown in Table 11-3. The DISC timing flag is set at the point where the DISC triggered. The CFD flag marks the point where the CFD triggered. Timing marks are enabled by setting the write_flags bit of the channel_control_# register to '1'.

11.3 Event Buffer

Each DP channel has a dedicated 64kB event record FIFO to store events pending readout. This is provided to provide fast buffering during period of high activity. So long as these buffers do not fill, there is no dead time between events. Event records written to the Event Record FIFOs are collected and transferred to the collection FIFO in round robin fashion. When operating in USB mode these individual Event Record FIFOs make up the majority of the event buffering capability, with an additional 80kB of buffer space distributed across the collection, transport, and USB FIFOs. However, when using Ethernet, an additional 256MB of buffer space is provided by DDR memory on-board the SSP. Refer to section 3 for detail on how to switch the event data interface between Ethernet and USB.

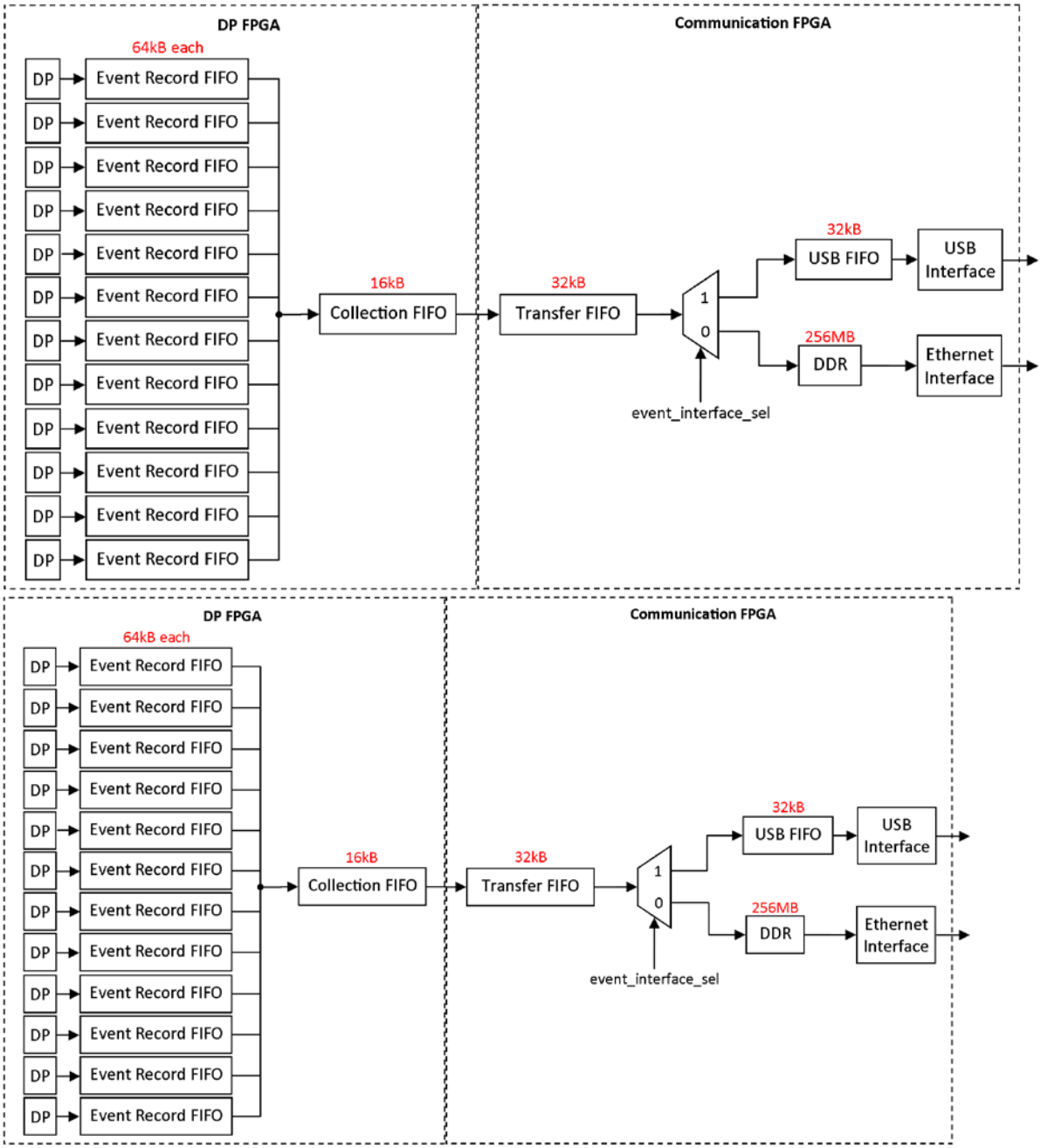


Figure 11-7: Simplified event buffer diagram.

As long as the average data rate does not exceed the Ethernet throughput the SSP will not drop eventevents due to full buffers. However, in the case where the event exceedrate exceeds the read out speed, the equations below define how long the SSP can continue to collect events, without dropping events due to a full event buffers. For simplicity these equations, assume the same average event rate for all active channels and some conservative approximations have been made. These equations also assume that the offset mode is disabled and the event readouts do not overlap.

Terms:
CDR - Channel Data Rate
TR - Transfer Rate = 400,000,000 Bytes/s
ERF_FR - Event Record FIFO Fill Rate
DDR_FR - DDR Fill Rate
TERFF - Time to Event Record FIFO Full
TDDRF - Time to DDR Full

First find the average channel data rate.
CDR = (32 + 2 x waveform_length) x event_rate

Find the Event Record FIFO fill rate, if this number is less than or equal to 0, this Event Record FIFO will be readoutread out faster than it is being written and will not be the limiting factor.
ERF_FR = CDR - TR / number_of_active_channels

Now calculate the fill rate of the DDR. If ERF_FR is less than or equal to 0 use this equation:
DDR_FR = (CDR x number_of_active_channels) - ethernet_readout_rate
Otherwise:
DDR_FR = TR - ethernet_readout_rate

Now find the time to fill both buffers. Again, these numbers may be less than or equal to 0, in which case that buffer will not pose as a limiting factor.
TERFF = 65,000 / ERF_FR
TDDRF = 268,000,000 / DDR_FR

If both numbers are negative, then the SSP's buffers will never fill up (readout can keep up with the event rate). Otherwise, take the smaller of the two (TERFF, TDDRF) that is how long the SSP can operate at that event rate without dropping event due to full event buffers. If using the USB interface instead of Ethernet, take CDR and multiply it by the number of active channels, if your USB transfer rate is less than this number, then the time to fill the event buffers is essentially just:

65,000 / (CDR – usb_transfer_rate / number_of_active_channels)

If the event FIFO does not have enough free space available for the next event record, that record will be dropped. No partial readouts are possible for these events.

12. Data Processor Monitoring

There are three methods of monitor the status of each Data Processor.

1. Front Panel LED
2. Activity Counters
3. Front Panel Trigger Output

12.1 Front Panel LED

The front panel activity LEDs provides the most direct monitor of channel activity.

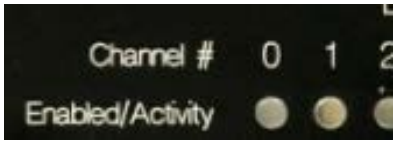


Figure 12-1: Front panel activity LEDs.

The LED is off when the channel is disabled. This may be due to either the channel_enable bit in the channel_control_# register being set to 0 or the dp_logic_enable bit in the master_logic_control register being set to 0. The latter will result in the disabling of all channels.

When the channel is enabled the LED will illuminate. Green is the normal idle state and indicates that the channel is not triggering. The LED blinks yellow every time an event is processed. At rates above 35Hz or so the LED will illuminate solid yellow. As the event rate increases the color of the LED will transition from yellow to red. For event rates of 10 kHz and above the LED will be red.

12.2 Activity Counters

For a non-subjective measure of event rates, each DP provides four 32-bit counters that can either be set to accumulate or rate measurement mode. In accumulate mode, the counters simply count events, in rate mode the value of the counter register represents the event rate in Hz. The four counters are listed in Table 12-1. All counter mode control bits are located in the channel_control_# registers.

Table 12-1: DP counters.

Counter Name	Counter Register Name	Counter Mode Control Bit
Discriminator Hit Counter	disc_count_#	disc_counter_mode
Accepted Hit Counter	ahit_count_#	ahit_counter_mode
Pileup Counter	pileup_count_#	pileup_counter_mode
Dropped Event Counter	dropped_event_count_#	dropped_event_counter_mode

When the counter mode bit is clear the counter is in rate mode, otherwise it is in accumulate mode. Each DP also provides a local counter reset control, the counter_reset bit in the

channel_control_# register. There is also a counter_reset control bit in the master_logic_control register that reset all DP counters in all channels. The counter_enable bit, also in the master_logic_control register, freezes the state of all counters when clear.

12.2.1 Discriminator Hit Counter

This counter tracks the activity of the CFD, or the DISC if the CFD is in bypass mode.

12.2.2 Accepted Hit Counter

The accepted hit counter tracks the events that are processed and handed off to the event packaging logic. Unlike the discriminator hit counter, this counter will treat a train of piled-up pulses as a single event. Also, events that are rejected due to pile-up suppression settings are not handed off to the event packaging logic, and are therefore not counted by this counter.

12.2.3 Pile-up Counter

The pile-up counter tracks events marked as pile-up.

12.2.4 Dropped Event Counter

The dropped event counter tracks the events that were processed for readout but had to be dropped due to either the waveform offset setting or insufficient space in the event record FIFO.

12.3 Front Panel Trigger Output

The third method of monitoring the DPs is via the trigger output port on the front panel of the SSP, as shown in Figure 12-2.

Figure 12-2: Trigger output port.

The twelve outputs on the trigger output port corresponds to the 12 channels of the SSP. See section 2.1.3 for detailed specifications. The trigger out will drive a pulse when either the DISC or CFD triggers, as selected by the channel_trigger_out_select bit of the front_panel_config register (0 = CFD, 1 = DISC). The width of the pulse is controlled by the value of the disc_width_# register.

When using these outputs to drive external logic, it is recommend that the channel_trigger_out_select bit be set to DISC mode. The DISC mode output is generated more promptly with a delay determined primarily by the by the width of the p window and is independent of all other window settings.

13. SiPM Biasing

The SSP provides two bias option selected by a jumper on each channel. This is shown in Figure 13-1. The jumper either connects the SiPM to the common external bias input (Shown as Vbias_EXT) or the channel's internal bias circuit. Note, the external bias should not exceed 50v because of capacitor voltage ratings. The internal bias is provided by separate 12-bit DACs, Analog Devices AD5501, one per channel. The full scale output of the DAC is set to 30v. This results in a LSB precision of 7.3 mV.

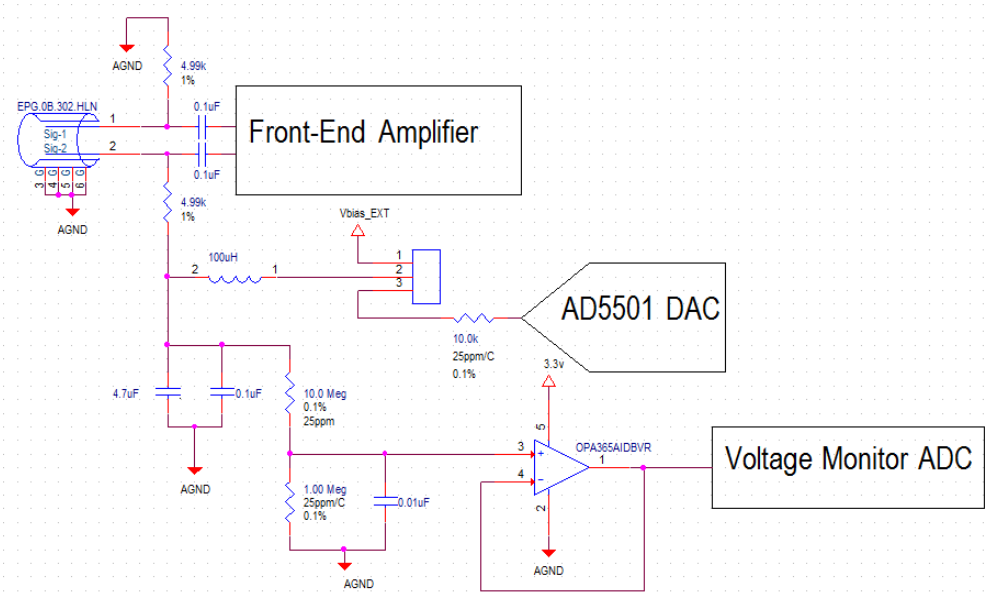


Figure 13-1: Simplified SiPM Bias and Bias Monitoring Schematic

13.1 Bias Control

Each channel's bias is individually controlled by the 12 bias_dac_config_# registers. The value of the 12-bit bias_dac_value bit field in each register sets the channels bias voltage:

Bias Voltage = 30.0 V / 4096 x bias_dac_value

The voltage is a direct control of the DAC output no calibration or software compensation is included.

In addition to the voltage setting is an output enable, controlled by the bias_dac_enable bit of the bias_dac_config_# register. Write this bit to '1' to enable the bias, '0' to disable the bias (power-on default). When the bias is disabled the user has the option to tie the output of the bias DAC to ground through 20kohms. This option is enabled by setting the

bias_dac_disable_option bit to ‘0’ (power-on default), which is also in the bias_dac_config_# registers.

New bias settings (both voltage and output enable settings) must be loaded before they take effect. This is done my writing ‘1’ to the bias_load register. This register will automatically clear back to ‘0’ after being written.

13.2 Bias Monitoring

The SSP measures the output voltage of the SiPM bias circuit after the RLC filter. 0.1% resistors are utilized throughout the SiPM circuitry to provide accuracy to within 30 mV. From observation the tolerance of the components are generally better than the tolerance specified thus the resultant accuracy is better. The bias voltage on the SiPM under ideal conditions with nearly zero bias current is the DAC voltage. However, the total series resistance from the DAC to ground is 20k Ohms which will lower the voltage across the SiPM as it conducts. Note that there is no provision for compensating for gain differences in the monitor and DAC in the SSP firmware and software. See section 15.2.1 for details on how to monitor the bias voltage.

Each bias DAC also has thermal shutdown protection. If one of the 12 DAC’s enter into a thermal shutdown state the corresponding bit in the bias_status register will be set to ‘1’. The state of the bias DAC can also be monitored on the front panel of the SSP, as shown in Figure 13-2.

Figure 13-2: Front panel bias monitor.

When the Bias is on the “Bias Status” LED will light up blue. If the bias DAC enters the thermal shutdown state the LED will turn Red. When the Bias is Off the LED will also turn off.

14. On-Board Charge Injection

The SSP provides charge injection on each channel to allow for the calibration of front-end gain and relative timing. The charge injection circuit is designed to generate a pulse that mimics the nominal pulse generated by a SiPM charge pulse. The resistors and capacitors that govern the charge injection circuitry are all 0.1% and 1% respectively. A simplified schematic of the charge injection circuit is shown in Figure 14-1 below. Note that while the shape of the charge injection pulse is approximately that of a SiPM pulse, the gain and the integral of the pulse are governed primarily by the resistors.

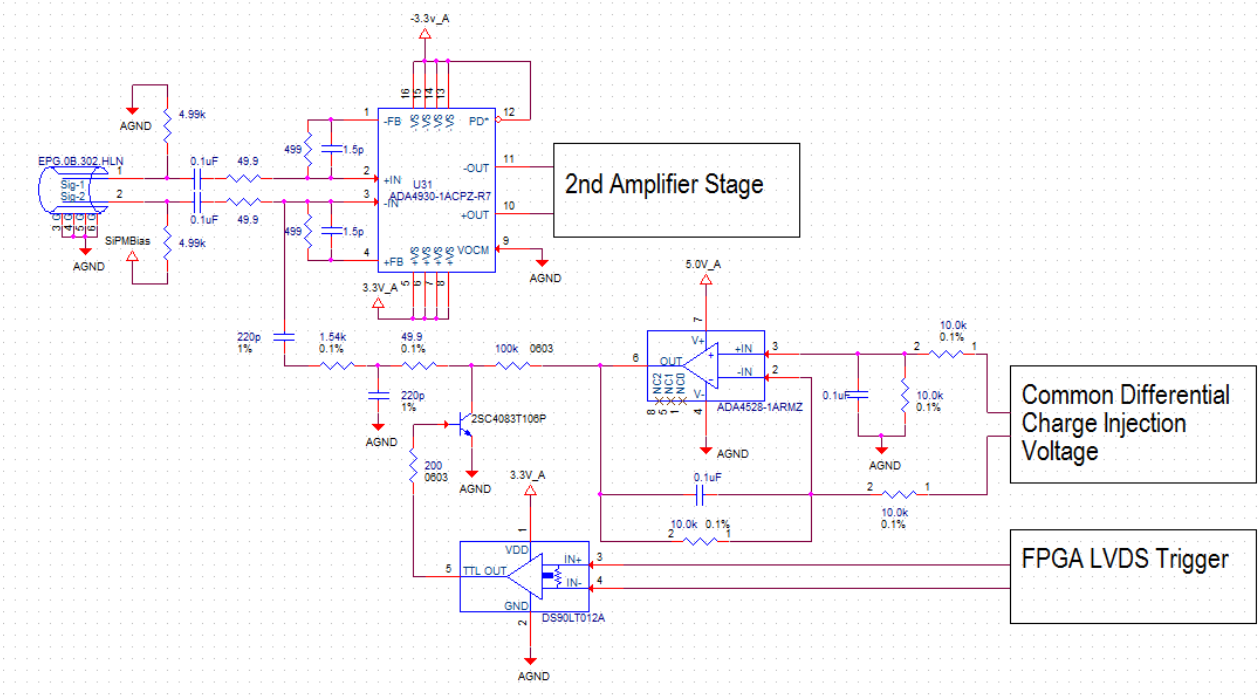


Figure 14-1: Simplified Charge Injection Schematic

14.1 Charge Injection Control

Control of the charge injection function is provided via 5 registers, as described in the following section. The charge injection function can be enabled or disabled for each channel individually. The pulse shape is fixed and is governed by passive components in each front-end channel. The global register for charge injection control the charge injection voltage, switch polarity, pulse duration, trigger input control, and the frequency of triggering which can used for measuring time-stamp uniformity.

14.1.1 Charge Injection Voltage Control (DAC Control)

A single 16-bit DAC sets the charge injection voltage for all channels. The DAC output is controlled by the qi_dac_value field of the qi_dac_config register. The provided 16-bit value sets the DAC output from 0 V to 4.096 V which through a single ended to differential buffer creates a 4.096 V or 0 V charge injection voltage respectively. So, a DAC voltage of 4.096 V (0xFFFF) results in 0 V for the charge injection. Likewise, a 0 V DAC voltage (0x0000), results in 4.096 V for the charge injection voltage. The DAC's output may also be tristated, pulled down to ground via 100kOhms, or pulled down to ground via 1kOhms, by setting the appropriate value for the qi_dac_mode bitfield. This is an internal feature of the DAC. The valid selections are listed below:

- 0 = Normal Output Mode
- 1 = Tristate
- 2 = 100kOhm to Ground
- 3 = 1kOhm to Ground

Before any change to the qi_dac_control register take effect they must be loaded into the DAC by writing the qi_dac_load bit of the qi_dac_control register. This bit self clears. The differential output voltage is sourced to all channels, where it is buffered by the individual channels for charge injection in that channel.

14.1.2 Charge Injection Trigger Control

The qi_config register provides the primary means of controlling the charge injection trigger. The master_enable bit of this register globally enables or disables the charge injection logic. If enabled, bit 11 down to 0 of the channel_enable bitfield serve as channel specific enables. If a bit is set, then the corresponding channel's charge injection is enabled, otherwise it remains disabled. The last two bit fields of this register are the trigger_select and switch_mode which are described below.

14.1.2.1 Switch Polarity

The Artix-7 FPGA controls the state of the LVDS to TTL driver. During charge injection operation, the switch can be set to be normally open or closed, controlled by the switch_mode bit of the qi_config register. When a charge injection "event" is initiated, the state of the switch changes. The correct and default polarity is normally open and closed to generate a charge injection event. The value of the bit should not be modified. Always write 0.

14.1.2.2 Trigger Input Control

The trigger_select bitfield of the qi_config control register selects the trigger source for the charge injection. Several different charge injection trigger inputs are available, as shown in Table 14-1.

Table 14-1: Charge injection trigger sources.

trigger_select	Input Source
0b0XXXX	Direct tcal input (asynchronous)
0b10000	Timestamp Trigger: 292.969kHz.
0b10001	Timestamp Trigger: 73.242kHz.
0b10010	Timestamp Trigger: 2.289kHz.
0b10011	Timestamp Trigger: 143.051Hz.
0b10100	Timestamp Trigger: 35.763Hz.
0b10101	Timestamp Trigger: 8.941Hz.
0b10110	Timestamp Trigger: 1.118Hz.
0b11000	Manual/Pulsed control input.
0b11111	Closed (QI Disabled).
All Others	Unused (Reserved)

X = Don't Care

14.1.2.2.1 Direct t-cal input

14.1.2.2.2 With this selection the front panel tcal input directly and asynchronously controls the charge injection, both in terms of timing and pulse width. The input is nominally configured to use NIM level logic, but may also be configured for other thresholds within +/-3.3V.Timestamp Trigger

The timestamp triggers occur on the rollover of a pre-selected number of timestamp bits as described in section 4.2.1.1. It may also be used in conjunction with the channel triggertriggers described in section 7.1.1.2 to force the channel to trigger synchronously with the charge injection for calibration purposes. One of seven charge injection rates may be selected.

14.1.2.2.3 Manual/Pulsed control input

When this mode is selected, writing the qi_manual_trigger bit of the qi_pulsed register triggers the charge-injection on the next clock. The register automatically resets to zero after the trigger.

14.1.2.3 Pulse Duration

The length of time the switch state is flipped is controlled by the qi_pulsed_width register. The value is set in terms of number of ADC clock cycles. Note that this has no affect the front panel TCAL input. This should be set high enough to not interfere with the charge injection event length of ~2 μs, but should be shorter than the period between charge injection events.

14.1.2.4 Pulse Delay

The precise time of the charge injection trigger may be skewed from that of the selected trigger source in 78ps steps via the qi_delay register. The value must be loaded before is takes effect. This is done by writing 0x00030000 to the qi_pulsed register. These bits will self-clear. Note that this has no affect the front panel TCAL input.

14.2 Charge Injection Recommended Operation

The following settings are recommended for stable charge injection operation. Note that in addition to the negative pulse caused by the base-to-collector capacitance, there is the switch opening and the charge injection capacitors restoring back to the default state. This causes a positive going pulse, with the same polarity as the charge-injection pulse. It is distinct for two reasons. First, the shape is significantly different and is largely invariant to the charge injection voltage. Figure 14-2 shows what a standard charge injection pulse should look like. Figure 14-3 shows the results of a full charge injection scan for all 12 channels.

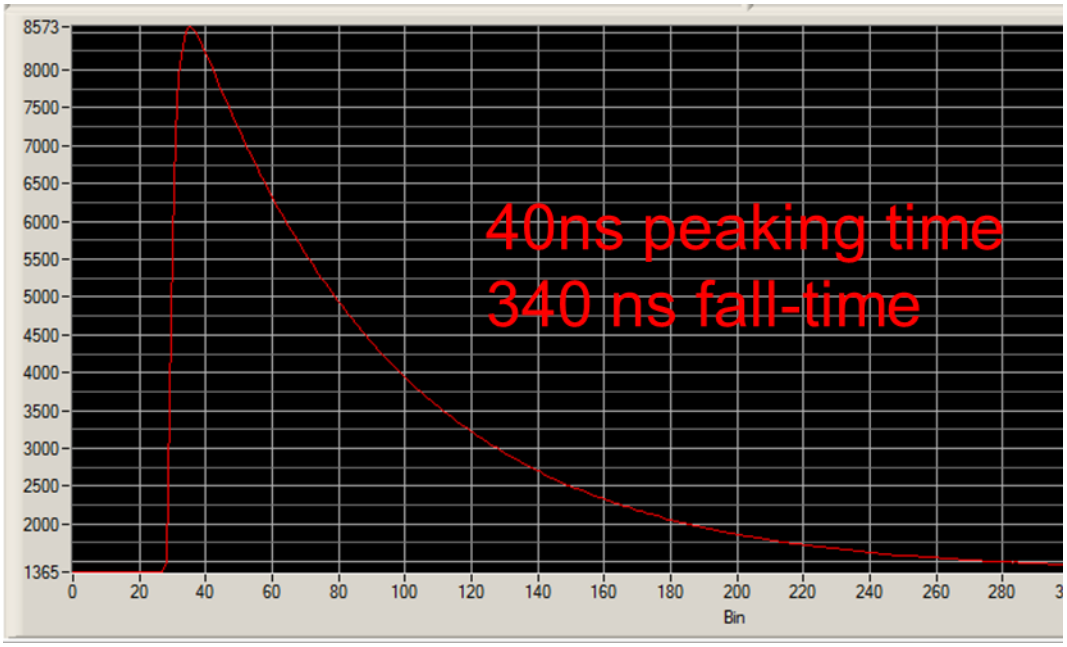


Figure 14-2: Standard Charge Injection Pulse

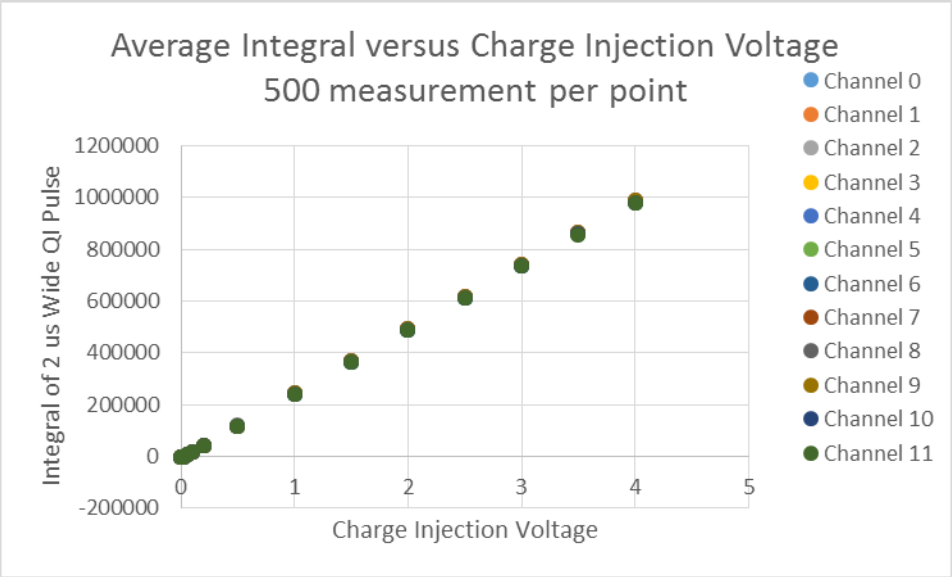


Figure 14-3: Results from a charge injection scan.

14.2.1 Charge Injection Voltage

To do a complete measurement of the gain of the system, the charge injection voltage should be adjusted through a range of voltages. Three regimes exist with respect to the applied charge injection voltage:

Regime 1: Below 0.003 charge injection voltage nothing but the negative pulse is observed.
Regime 2: Between 0.005v to 0.03v a superposition in the waveforms is observed.
Regime 3: Above 0.03 the pulse mimics the SiPM charge pulse.

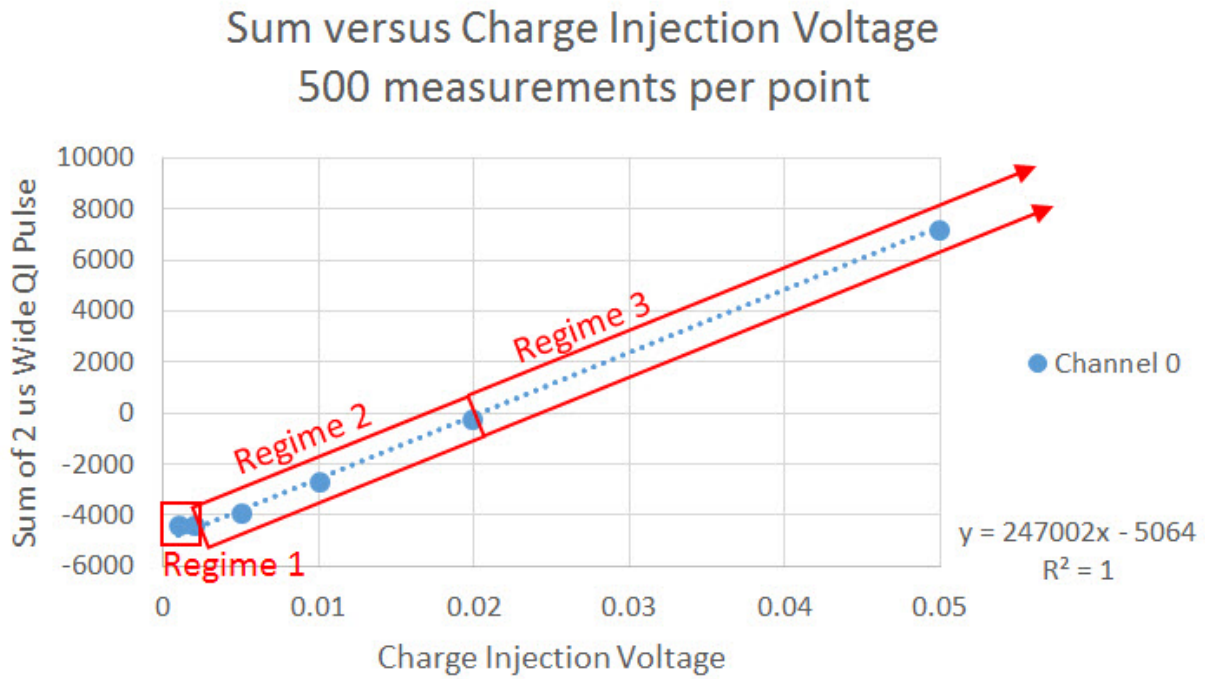


Figure 14-4: Charge injection scan for one channel showing the transition between the three different regimes.

14.2.2 Typical Trigger Input Control

The simplest method for performing charge injection is to enable all channels and use the timestamp trigger to initiate charge injection. The data acquisition cycle records the charge injection event, and ignores the period of time when the charge injection switch opens to recharge the charge injection capacitors.

14.2.3 Frequency

The recharge time-constant of the charge injection circuit is 44μs. Generally, 10 time constants are recommended to recharge the charge injection capacitors. Thus, the minimum period for performing charge injection would be ~450 μs (including signal acquisition time), or a maximum frequency of ~2 KHz. (Typically when characterizing the modules, the testing is done at 36 Hz.)

15. Analog Monitor

The SSP uses a Texas Instrument's ADS1158 multi-channel 16-bit ADC for monitoring the bias, QI voltage, temperature and the 5V power supply. The ADS1158 also provides an internal temperature monitor.

15.1 Configuration

The ADS1158 has 9 8-bit internal registers that control its operation. The configuration of the monitor ADC is controlled by 3 32-bit SSP registers. Registers 0 through 3 are set by the mon_config register, registers 4 through 6 are set by the mon_select_register, and registers 7 and 8 are set by the mon_gpio register.

At power on, the values stored in these registers are loaded into the ADC. As a diagnostic feature, the communication FPGA then reads back the configuration state of all the ADC's registers and reports them in the mon_control_readback, mon_status_readback, and mon_gpio_readback registers. The format of the readback and control registers are identical. The data in the readback registers should match value in the corresponding control register after power on. If these values do not match after power on, the likely cause is a hardware problem within the SSP module. While the SSP provides read/write access to registers 0 through 8 of the ADS1158, it is recommend that the user preserves the default values of reregisters 0,1,2,3, 7 and 8. Refer to the ADS1158 datasheet for details regarding the purpose of these registers, if necessary.

To load a new ADC configuration, first write the new configuration into the mon_config and mon_status registers (do not change the default value of the mon_gpio register). Then set the monitor_load bit in the mon_control register to '1'. This bit will self-clear.

ADC registers 4, 5 and 6 controls which monitor channels are enabled. Read and write access to these registers is provided via the mon_select and mon_select_readback register of the SSP. Table 15-1 lists the enable bits for all the monitor channels. To enable all monitors write 0x00FFFF3D to the mon_select register, then load the setting.

Monitor	Value Register	Enable Control Bit
Channel #0 Bias Voltage	mon_bias_0	bias_enable_0
Channel #1 Bias Voltage	mon_bias_1	bias_enable_1
Channel #2 Bias Voltage	mon_bias_2	bias_enable_2
Channel #3 Bias Voltage	mon_bias_3	bias_enable_3
Channel #4 Bias Voltage	mon_bias_4	bias_enable_4
Channel #5 Bias Voltage	mon_bias_5	bias_enable_5
Channel #6 Bias Voltage	mon_bias_6	bias_enable_6
Channel #7 Bias Voltage	mon_bias_7	bias_enable_7
Channel #8 Bias Voltage	mon_bias_8	bias_enable_8
Channel # 9Bias Voltage	mon_bias_9	bias_enable_9

Channel #10 Bias Voltage	mon_bias_10	bias_enable_10
Channel #11 Bias Voltage	mon_bias_11	bias_enable_11
QI Voltage	mon_value_0	value_enable_0
AGND	mon_value_1	value_enable_1
AREF	mon_value_2	value_enable_2
AMID	mon_value_3	value_enable_3
ADC Offset	mon_value_4	value_enable_4
ADC VCC	mon_value_5	value_enable_5
ADC Temp	mon_value_6	value_enable_6
ADC Gain	mon_value_7	value_enable_7
ADC Ref	mon_value_8	value_enable_8

Table 15-1: ADC monitor channels.

The monitor_enable control bit of the mon_control register serves as the overall enable to the monitor logic. This bit must be set to '1' to enable monitoring.

15.2 Values

The formulas provided in this section describe how to convert the values reported by the monitor ADC into voltages. All values reported by the monitor ADC should be treated as signed 16-bit values. In order to interpret the values reported by the ADC the reported values of VREF and AGND are required. All equations will require the following terms:

K1 = 4096 / (AREF - AGND)
K2 = -K1 x AGND

The following constants are also used:

R1 = 10,000
R2 = 10,000,000
R3 = 1,000,000
R4 = 4,990

15.2.1 Bias Monitor

The following equations convert the reported bias monitor values into the voltage and current applied across the SiPM. First calculate the voltage at the point of measurement for each channel:

MBV# = 11 x (K1 * mon_bias_# + K2) / 1000

MBV is the measured bias voltage in V. This is the voltage across the SiPM assuming that the SiPM current is zero. However current limiting resistors and filtering components means that this value will be in error if the SiPM current is not zero.

While the monitor circuit was not specifically designed to measure the bias current, a coarse (uncalibrated) measurement of the current can be made by assuming the bias DAC's output is correct. In this case one can calculate the bias current and corrected bias voltage by the following equations:

Ch# Bias Current = ((bias_dac_readback_# / 4095) * 30 - MBV#) / R1 – MBV# / (R2 + R3)
Ch# Bias Voltage = MBV# - [Ch# SiPM Current] x 2 x R4

The uncalibrated bias current measurement using the above technique is accurate to approximately +/-(2% of measurement + 5uA).

The uncalibrated bias voltage measurement using the above technique is accurate to approximately +/-(0.25% of measurement + 50mV).

There is a tradeoff between compensating for the effects of the SiPM current and the added uncertainty in the introduced into the Bias Voltage measurement in so doing. Calibration (i.e. a software applied correction, based on a calibration) could be done to significantly improve the accuracy of both the current and voltage measurements.

15.2.2 ADC VCC

The following equation converts the reported ADC VCC value into volts:

ADC VCC [V] = mon_value_# / 3072

This value should be 5V.

15.2.3 Temperature

The following equation converts the reported temperature value into degrees C:

Temp [C] = ((K1 * mon_bias_6 + K2) - 168) / (0.563) + 25

References

[1] M. Diwan, R. Svoboda, J. Strait, "The Long-Baseline Neutrino Experiment," Submitted to the European Strategy Preparatory Group, August 15, 2012, http://lbne2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=6279;filename=LBNE-ESPG.pdf;version=3

[2] Xilinx Artix-7 Field-Programmable Gate Array data sheet: http://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf

[3] Specifications for the LBNE Photon Detection Front-End Electronics (to be published)

[4] J. Anderson, P. de Lurgio, Z. Djurcic, G. Drake, A. Kreps, M. Oberling, "A New Data Acquisition System for Silicon Photomultipliers," to be presented at the 2014 IEEE Nuclear Science Symposium in Seattle, WA, Nov., 2014.

[5] MicroZed users guide: http://www.zedboard.org/sites/default/files/documentations/MicroZed_HW_UG_v1_4.pdf

[6] Xilinx Zynq FPGA data sheet: http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[7] R. Kwarciany, N. Wilcer, "NOvA Timing Distribution System," LBNE internal note, DocDB 71100-v1, Apr. 8, 2013.

Appendix A – Slow Control Packet Structure

The LBNE SSP uses a simple packet structure for the slow control communication path between the SSP module and a host computer. The current implementation only supports the USB interface but will be modified to support UDP/IP or TCP/IP over the Ethernet port. Regardless of the chosen interface, the structure shown below is the data payload sent by the host or the SSP. The orange region is a fixed size header that is included in all transactions. The green region is a variable size data portion that is only included when necessary based on the Command word in the header. There is a 256 word maximum for the data portion of a slow control packet (equal to the constant MAX_CTRL_DATA). Each header field and data word is a 32-bit integer.

WORD	FIELD	DESCRIPTION
0	LENGTH	Length(31:0)
1	ADDR	Address(31:0)
2	CMD	Command(31:0)
3	SIZE	Size(31:0)
4	STAT	Status(31:0)
5	DATA0	Data Word 0 (31:0)
…	…	…
n+5	DATAn	Data Word n (31:0)

n < 256

The slow control protocol is a master/slave interface controlled by the host computer. The host sends a command to the SSP and waits for a response. The SSP never initiates a slow control transaction but it always sends something in response to every command.

When the SSP receives a packet, it inspects the header for a valid combination of Address, Command, and Size. (The host always sends a Status word of zero.) If the header is invalid, the SSP will respond with a packet consisting only of a copy of the received header with the Status word changed to one of the error values defined later in this document. The SSP does not attempt any processing of the packet if the header is invalid.

If the SSP receives a valid header, it performs the requested operation and returns a response packet. The response contains a copy of received header along with an appended data portion, if necessary.

A.1 Slow Control Addresses

The following table summarizes the legal range of addresses and which device within the SSP will respond to the command. All addresses point to 32-bit values within the SSP. Therefore,

the lower 2 bits of the address must always be zero to avoid alignment issues. The SSP treats unaligned addresses as errors.

Address Range	Device	Main Purpose	Valid Commands
0x00000000 - 0x0FFFFFFC	ARM Processor	Diagnostics	0 – 7
0x10000000 - 0x10FFFFFC	DSP Flash Memory	Holds the DSP firmware	0, 1, 4, 8 – 12
0x20000000 - 0x20FFFFFC	Configuration Flash Memory	Holds the SSP configuration	0, 1, 4, 8 – 12
0x30000000 - 0x30FFFFFC	Comm Flash Memory	Holds the Comm firmware	0, 1, 4, 8 – 12
0x40000000 - 0x7FFFFFFC	Zynq FPGA	Communications	0 – 7
0x80000000 - 0xBFFFFFFC	Artix FPGA	DSP	0 – 7

A.2 Slow Control Commands

The following table summarizes the legal combinations of commands and packet formats for the SSP. For each command, it shows the number of header and data words sent between the host computer and SSP. All slow control transactions use 32-bit data words.

Host Packet			SSP Response		CMD Value	ADDR Alignment
CMD Field	Header Words	Data Words	Header Words	Data Words		
cmdRead	5	0	5	1	0	4 Byte
cmdReadMask	5	1	5	1	1	4 Byte
cmdWrite	5	1	5	0	2	4 Byte
cmdWriteMask	5	2	5	0	3	4 Byte
cmdArrayRead	5	0	5	SIZE	4	4 Byte
cmdArrayWrite	5	SIZE	5	0	5	4 Byte
cmdFifoRead	5	0	5	SIZE	6	4 Byte
cmdFifoWrite	5	SIZE	5	0	7	4 Byte
cmdNVWrite	5	1	5	0	8	1 Byte
cmdNVArrayWrite	5	SIZE*	5	0	9	1 Byte
cmdNVEraseSector	5	0	5	0	10	4 KByte
cmdNVEraseBlock	5	0	5	0	11	64 KByte
cmdNVEraseChip	5	0	5	0	12	16 MByte

*For NVArrayWrite SIZE cannot be more than 64. For all others may be up to 256.

The individual commands are described next:

- cmdRead - This operation reads a single 32-bit value from the address in the ADDR field. The SIZE field should always equal 1.
- cmdReadMask - This operation performs a mask read of a single 32-bit value from the address in the ADDR field. Only those bits that are set in the MASK are returned. All other bits will read zero. The SIZE field should always equal 1. The MASK is sent as DATA0.
- cmdWrite - This operation writes a single 32-bit value to the address in the ADDR field. The SIZE field should always equal 1.
- cmdWriteMask - This operation modifies the 32-bit value at the address in the ADDR field using a MASK and VALUE. Only those bits that are set in the MASK are modified to match the corresponding bits in the VALUE. All other bits are not modified. The SIZE field should always equal 1. The MASK is sent as DATA0. The VALUE is sent as DATA1.
- cmdArrayRead - This operation reads a series of 32-bit values starting from the address in the ADDR field. The SIZE field must be less than or equal to 256.
- cmdArrayWrite - This operation writes a series of 32-bit values starting at the address in the ADDR field. The SIZE field must be less than or equal to 256.
- cmdFifoRead - This operation repeatedly reads a series of 32-bit values from the address in the ADDR field. ADDR must correspond to a FIFO access register or the return packet will simply contain SIZE copies of the same value. The SIZE field must be less than or equal to 256.
- cmdFifoWrite - This operation repeatedly writes a series of 32-bit values to the address in the ADDR field. ADDR must correspond to a FIFO access register or the SSP's behavior will be undefined. The address will eventually acquire the final data value in the packet. The SIZE field must be less than or equal to 256.
- cmdNVWrite - This operation writes a single 32-bit value to the flash memory address in the ADDR field. The SIZE field should always equal 1. Only use for testing.
- cmdNVArrayWrite - This operation writes a series of 32-bit values starting at the flash memory address in the ADDR field. The SIZE field must be less than or equal to 64.
- cmdNVEraseSector - This erases 4,096 bytes of flash memory starting at the address in the ADDR field. The address must fall on a 4 KB boundary.
- cmdNVEraseBlock - This erases 65,536 bytes of flash memory starting at the address in the ADDR field. The address must fall on a 64 KB boundary.
- cmdNVEraseChip - This erases an entire region, 16,777,216 bytes. The provided address must be either 0x10000000, 0x20000000, or 0x30000000.

Appendix B – Flash Memory

There are three non-volatile flash memory regions in the SSP. Physically these are 3 flash memory chips inside the SSP. Two hold the firmware images for the communications FPGA and the data processing FPGA. The third holds the configuration settings. The data can be read from these regions using the same methods used for reading registers. However, as described in "Appendix A – Slow Control Packet Structure" these regions can only be written and erased using the "NV…" commands. The address ranges for these regions are:

- 0x10000000 to 0x10FFFFFF = Holds the data processing FPGA firmware.
- 0x20000000 to 0x20FFFFFF = Holds the configuration settings (such as the IP settings). This region is sub-divided into several partitions.
- 0x30000000 to 0x30FFFFFF = Holds the communication FPGA firmware.

The configuration region is divided into smaller regions as shown in the table below:

Address Range	Region Name	Purpose
0x20000000 – 0x2000FFFF	Comm Configuration	Holds communication settings, loads at power-on before configuring DP FPGA.
0x20010000 – 0x2001FFFF	SSP Configuration A	Stores an SSP configuration. Loads at power-on. Blank by default.
0x20020000 – 0x2002FFFF	SSP Configuration B	Stores an SSP configuration. Loaded by command. Blank by default.
0x20030000 – 0x2003FFFF	SSP Configuration C	Stores an SSP configuration. Loaded by command. Blank by default.
0x20004000 – 0x200FFFFF	Unused	*Reserved*
0x20100000 – 0x201FFFFF	Module Info	Store module information in ASCII.
0x20200000 – 0x20FFFFFF	Unused	*Reserved*

The "Comm Configuration" regions primary purpose is to store IP and MAC address information, but it can be utilized for other communication FPGA only settings. The other three SSP configuration blocks, can be ustilized to store and quickly recall SSP configurations. The Module Info section, does not affect operation, and is used to store module information and history.

B.1 Updating the Firmware

This section describes the process needed to update the data processing and communication firmware. Be advised, the corrupting the communication firmware may result loss of communication with the module. This condition can only be resolved by restoring the firmware over the JTAG port on the front panel.

There are three steps required to update the firmware:

1. Erasing the current firmware.
2. Writing the new firmware.
3. (Optional) Verify the firmware image was written without error.
4. Loading the new firmware into the FPGA.

B.1.1 Erasing

Before writing in a new firmware binary, the memory region that holds the firmware must be erased. This is accomplished by sending the NVEraseSector/Block/Chip command to the SSP.

An important note, is that these commands do not return until the requested erase operation has completed. NVEraseChip command can take up to a minute to complete. Because of this, it is recommended that several NVEraseSector or NVEraseBlock command are sent to erase a large region of the flash so that progress can be monitored. Refer back to Appendix A for more information on using these commands.

B.1.2 Writing

There are two commands that write to the flash: NVWrite, and NVArrayWrite. The first, NVWrite, is really intended for testing only. It writes a 32-bit value to 4 bytes of the flash memory starting at the specified address. The second, NVArrayWrite, allows one to write up to one page (256 Bytes) of data at a time. The provided address must fall on a page (256 byte) boundary. While being easier on the flash chips, this method is also much faster.

The firmware files are provided as binary data. The data is taken from the file and written to the proper region of the flash memory byte for byte.

B.1.3 Verifying

An optional step after programming, is to verify that the new contents of the flash have been written properly. The stored flash data can be read back with the same functions used for reading registers.

B.1.4 Loading

After programming the digital signal processing FPGA firmware, the FPGA must be reinitialized for the new firmware to take effect. There are two methods of achieving this:

1. Simply power cycle the module.
2. Write the value of 3 to the dp_control register (0x40000164), followed by writing 0. Do not read from any address above 0x80000000 during this step or the communication processor (as of this writing) will likely crash, requiring a power cycle to recover. Should also delay for about 2 seconds after preforming this operation to allow time for the signal processing FPGA to read in the new firmware.

When programming the communication FPGA, a power cycle is always required for the update to take effect. The IP settings however will be maintained as they are stored in a separate region.

One can verify the firmware version that is currently loaded in the SSP by reading the version registers. There are four in total:

These two registers correspond to the communication firmware:
comm_arm_fw_build = Address 0x00000010
comm_fpga_fw_build = Address 0x40000010

These two registers correspond to the digital signal processing firmware:
dp_fpga_fw_build = Address 0x80000010
dp_fpga_fw_date = Address 0x80000604

An additional version tracking register is provided for the calibration module:
calib_build = Address 0x80000014

These are updated every time the firmware is updated.

B.2 Configuration Memories

There are four configuration memory regions, as shown at the top of this section. All four partions work by storing a series of data pair consisting of “Register Address” and “Register Value.” They are essentially scripts that can be called up with a couple of register accesses.

The first is region is special in that it can only contain settings for the communication FPGA. Typically only the IP settings of the module are stored here. Any communication FPGA setting stored here, are automatically loaded at power on. The communication settings are stored and loaded separately so that even if the digital signal processing firmware is not present, you can still boot and communicate with the module.

The following partitions can be used to store up to three SSP configurations. They may contain settings for any register in the device. Configuration A works the same as the first partition, in that any stored settings are automatically restored at power on. Each of the three configurations are large enough to store every writeable register.

B2.1 Configuration Memory Format

All the configuration partitions use the same data format. The configuration must start with the following key values to indicate that a valid configuration is store in this region:

0xA5A55A5A
0x5A5AA5A5

If these two values are not at the beginning, the configuration will be ignored. These values are followed by any number of 32-bit register address/value pairs:

[32-bit Register Address #1]
[32-bit Register Value #1]
[32-bit Register Address #2]
[32-bit Register Value #2]
...
...
[32-bit Register Address #n]
[32-bit Register Value #n]

The configurations are written using the same commands described in section B.1.2 Writing. Any unused space at the end should be loaded with 0xFFFFFFFF, which is the default state after erasing the memory. When a configuration is loaded, every address/value pair that it contains is written to the SSP's register map.

By default, all SSPs are delivered with all configuration memories erased, which results in the device starting up with the defaults set in the firmware.

B2.2 Loading a SSP Configuration

First, select the configuration to load by writing 0, 1, or 2 to the ssp_config_select register, to select configurations A, B, or C. Then write '1' to the ssp_restore register to load the configuration. This register automatically clears after it is written. The behavior of a configuration that loads itself or other configurations is not defined, and should be avoided.