| IM Release # LLNL-AR-741177 | Revision: 2.1 |
|---|---|

# Nanosecond Gated CMOS Camera (NSGCC) ICD

## *LLNL v4 Camera Board*

Jack Dean
*hCMOS Project Manager*

Signature Date

Matthew Dayton
*Diagnostic Project Manager*

Signature Date

Brad Funsten
*Project FPGA Engineer*

Signature Date

Jeremy Martin Hill
*Project Software Engineer*

Signature Date

| Rev. | Date | Section Edits | Eng. | Description of Change |
|---|---|---|---|---|
| **2.1** | 7/6/2021 | - | JMH | nsCamera software releases 2.1.1 |
| **2.0.4** | 6/28/2021 | 12.1 | JMH | Added SW_COARSE_CONTROL register |
| **2.0.3** | 2/1/2021 | 12.1 | JMH | Rename of STAT_POTSCONFIGURED to STAT_DACSCONFIGURED |
| **2.02** | 1/8/2021 | 12.1 | BTF | Added five more bits to MISC_SENSOR_CTL register for accumulation mode control and reordered the register. |
| **2.01** | 11/3/2020 | 12.1 | BTF | Added MISC_SENSOR_CTL register to control miscellaneous Icarus sensor pins. |
| **2.0** | 10/16/2020 | 6.3, 12.1, 14, 15, 16, 18 | BTF | Added Section 16 mentioning the RS422 USB driver location that is used and primarily tested with the hardware. Added Section 18 to discuss ELM-U references to the board. Rewrote Daedalus sections for bypass Phi Clock and RSL programming in Section 15 . Updated temperature data in STAT_REG as 12-bits instead of 11 bits and confirmed Daedalus RSL left and right signals as always '0' in Section 12.1 . Updated Dual Edge Trigger in Section 6.3 . Also updated the title of the ICD as LLNL v4 Camera Board instead of " " " FPGA Board. Moved Power Save Mode Section to Icarus Implementation Section 14 since the mode applies only to Icarus. POWERSAVE subregister was also changed to Icarus only. LED_EN subregister was removed since there are no controllable LEDs on the Ver 4.0 board. Synchronized to nsCamera 2.1 |
| **1.22** | 9/22/2020 | 2, 7, 13.1, 16 | BTF | Added the sensor readoff time for Daedalus. Added new section on Radiation-Tolerant Modes. Fixed up register map with respect to Daedalus implementation. Updated Daedalus Implementation section. |

Previous change notes may be found in Section 19

**Disclaimer**



# Contents

# 1 Background

The Ultra-Fast X-ray Imager (UXI) program is an ongoing effort at Sandia National Laboratories to create high speed, multi-frame, time-gated Read Out Integrated Circuits (ROICs), and a corresponding suite of photodetectors to image a wide variety of High Energy Density (HED) physics experiments on both Sandia's Z-Machine and LLNL's National Ignition Facility (NIF). Several cameras have been designed over the length of the program; one of the most recent is the Icarus, which is an improvement on past imagers (Furi and Hippogriff). A second sensor that can be connected is the Daedalus sensor. The Icarus is a 1024 × 512-pixel array with either 25 µm or 8 µm spatial resolution containing four frames of storage per pixel and has improved timing generation and distribution components while achieved 2 ns time gating. The Daedalus sensor is also a 1024 x 512-pixel array with 25 µm special resolution containing three frames of storage per pixel and has an increased set of features for a wider variety of applications from interlacing of rows in each frame to configurability of all shutters.[1] See Section 14 for details regarding the Icarus implementation of the firmware and Section 15 for details regarding the Daedalus implementation.

Due to the unique test environments UXI sensors are targeted for, full custom hardware was required to physically mount an Icarus or Daedalus sensor, manage its various functions, and read out pixel data for transfer to a host computer. Beyond experimental functionality, the hardware also needed to accommodate sensor characterization requirements. Lawrence Livermore National Laboratory's 'Version 4.0 Board' was the result of these efforts. It mounts all the components required to fully utilize the Icarus and Daedalus sensors including analog to digital converters to convert pixel data and various system voltages to digital form for readout and analysis, DAC channels for remote configuration of critical bias

[1] This paragraph was sourced from References 2, 4, 5, and 6.

voltages, static random-access memories to buffer pixel data, RS422 and Gigabit Ethernet communications for remote access, and an FPGA to tie these components together.

This document describes the FPGA electrical interfaces in detail to allow the reader a greater understanding of the device, and to facilitate implementation of custom software to control and manage it.

The Version 4.0 Board is a continuation of the Nano-second Gated CMOS hardware design that retains much of the functionality of the Version 1.0 Board while adding features including a DAC instead of digital potentiometers, as well as sensors for pressure and radiation. The Version 4.0 board is intended for applications requiring tight form-factor enclosures. It is composed of two stacking boards; one holds the FPGA and regulators to power the various components of the board while the other contains the mating connector to the sensor, image-readoff ADCs, the DAC, and other components.

# 2 Design Summary

The Nanosecond Gated CMOS Camera (NSGCC) FPGA is a design targeted for the Microsemi A3PE3000-FG484 device residing on the LLNL Version 4.0 Board. It controls interaction between a host computer and the functions on the board, whose major components include an image sensor mounted on a daughter board, SRAM, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), temperature sensors, and power circuitry.

| Port | Readoff Time | | |
|---|---|---|---|
| | *ICARUS 2-Frame Readout* | *ICARUS 4-Frame Readout* | *Daedalus 3-Frame Readout* |
| **RS-422** | ~ 27 seconds | ~ 54 seconds | ~38 seconds |
| **Gigabit Ethernet** | < 1 second | < 1 second | < 1 second |

Table 1: Readoff time - Start of sensor readoff to images downloaded by host

A summary of the feature set of the FPGA:

- RS422 and Gigabit Ethernet Ports for complete control of FPGA and sensor
- ARM/Disarm function which aggregates critical system status into one status bit
- Controls four NoBL (QDR) 72 Mbits SRAMs and four 8-channel ADCs for digitizing, buffering, and read out of pixel data to a host computer
- Utilizes an 8-channel DAC and two 8-channel ADCs for calibration, monitoring, and tuning of critical bias voltages
- Timer/Counter which increments at 1 second intervals while FPGA is running
- Temperature sensor to monitor the system temperature
- Radiation-tolerant logic
- Supports the board's sensor voltage protection circuitry and enables image sensor power only when the correct sensor is installed, and the connected power supply provides the proper voltage.
- Power Supply requirement is 8 Volts at 2 Amps. The maximum current limit is 2.5 A.

A preliminary block diagram of the Version 4.0 Board, including the FPGA, is shown in Figure 1. The image sensor attaches to the Version 4.0 Board via a SamTec SEAF connector.

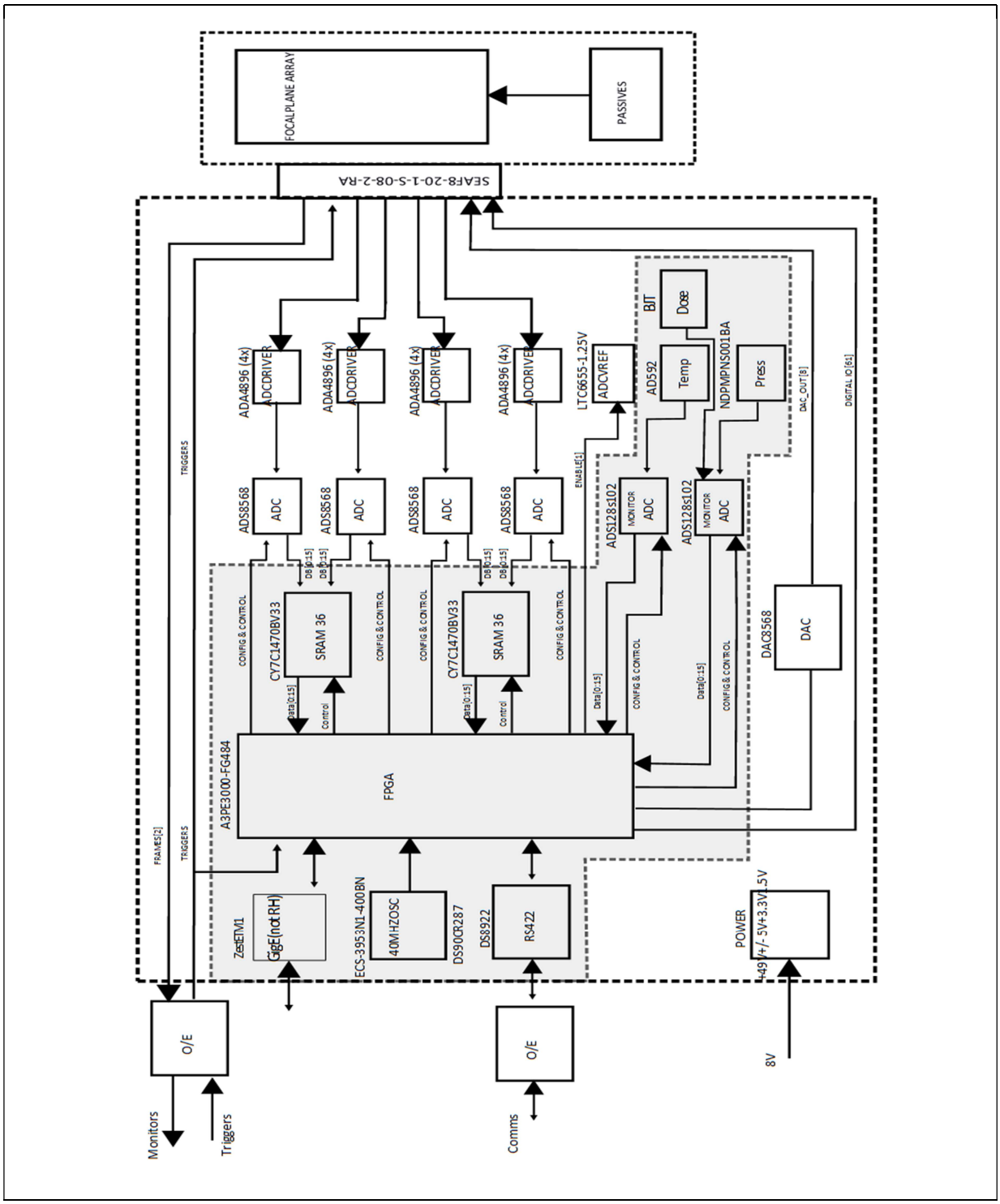


Figure 1: Version 4.0 Board, System Block Diagram

# 3 Block Diagram

A block diagram of the FPGA internal architecture is shown in Figure 2.

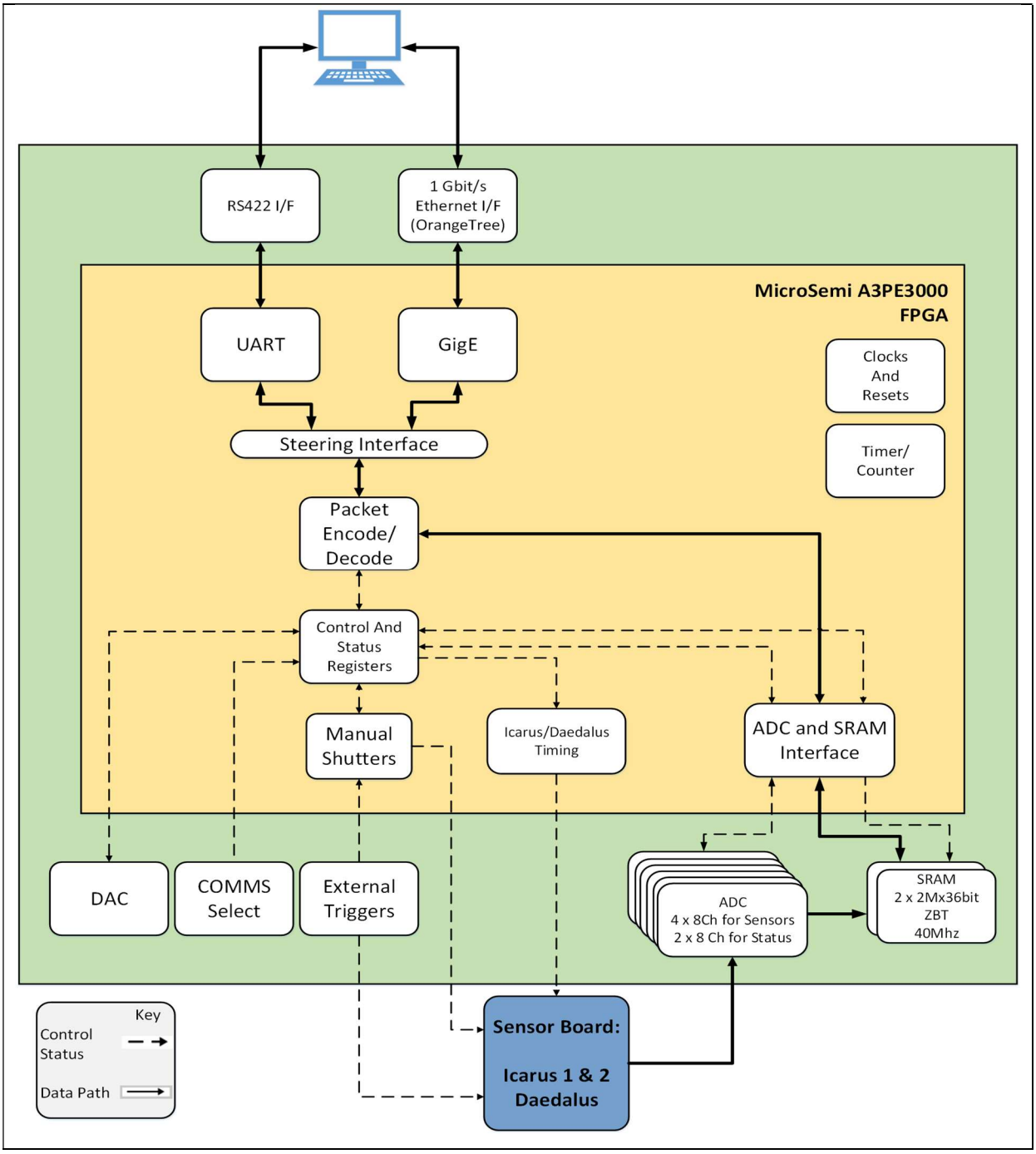


Figure 2: FPGA Block Diagram

# 4 Host Communications Interfaces

The NSGCC FPGA contains support for two serial interfaces for connectivity to a remote host computer—RS422 via a DB-9 connector, and Gigabit Ethernet via an RJ-45 connector. Only one port can be active at a time.

The RS422 port has a fixed configuration of 921.6 kbaud, 8 data bits, 0 parity bits, and 1 stop bit; these settings cannot be modified by the user. The Gigabit Ethernet port can be accessed by connecting a standard Gigabit Ethernet adapter via a Cat 5a or better cable.

## 4.1 Communications Port Selection

Selection of the active communications port is performed through a configuration pin on the FPGA (called *comm_port_sel_i*). Table 2 illustrates communications port selection. This table applies to all revisions of the LLNL V4.0 board.

| com_port_sel_i Jumper State | | Port Selected |
|---|---|---|
| *comm_port_sel_i logic level* | *Configuration* | |
| 0 | Short JMP1 to GND (Rev AA or AB) or short J5 (Rev AB or later) | RS422 |
| 1 | Open JMP1 to GND (Rev AA or AB) or open J5 (Rev AB or later) | Gigabit Ethernet |

Table 2: Communications Port Selection using FPGA Input Pins

# 5 Sensor Interface

The NSGCC FPGA controls the configuration and readout of the sensor in timing coordination with ADC conversion and SRAM storage. These functions are implemented with state machines, control logic, and command/status registers that respond to commands received from the host computer. ADC and SRAM functions will be discussed in the following sections: consult the associated sensor-specific ICD for relevant details.

## 5.1 ADC Interface and Control

The FPGA controls the six analog-to-digital converters (ADCs) on the board. Four of the ADCs are used to convert the image sensor's analog pixel information into 16-bit digital data, while the fifth and sixth are used to monitor DAC channel voltages. The four ADCs used to convert the image sensor's analog pixel information are Texas Instruments ADS8568SPM, which are eight-channel 16-bit devices.

The fifth and sixth ADCs (ADC5 and ADC6) are used to monitor a subset of the board's DAC channels. See Sections 14 and 15 for the mapping of ADC5 inputs for the Icarus and Daedalus sensors, respectively.

The FPGA reads ADC5 and ADC6 periodically and writes the voltage data into the **ADC5_DATA_1** through **ADC5_DATA_4** and **ADC6_DATA_1** through **ADC6_DATA_4** registers; these can be read by the user at any time. The time between updates of these registers is controlled by the **ADC_PPER** register. The associated DAC channels that are monitored are also described in Sections 14.2 and 15.2 for Icarus and Daedalus implementations respectively.

Prior to use, host software must configure the ADCs properly using the **ADC_CTL** and **ADC*X*_CONFIG_DATA** registers (see Section 12.1 ). First, the appropriate **ADC*X*_CONFIG_DATA** register(s) must be configured (see the ADS8568SPM data sheet for configuration information). Then the **ADC_CTL** register must be written to force the FPGA to write the configuration data to the targeted ADC(s).

## 5.2 SRAM Interface and Control

Each of the image sensor ADCs presents its output to the SRAM one channel at a time such that 4×16-bits, or 64 bits of data must be buffered simultaneously and continuously until all intended data is read from the sensor and stored in the SRAM. To handle storage of this data, the Version 4.0 Board contains two Cypress CY7C1470BV33-167AXI 2Mbit×36 SRAMs. When all pixel data has been stored, the FPGA asserts a status bit (**STAT_REG_SRC** bit 0, or **SRAM_READY**). Software waits for this bit to go high and commence readout of pixel data from SRAM to the host for storage and processing.

## 5.3 Automatic Sensor Detection

Currently, the LLNL V4 board is not able to provide automatic sensor detection.

# 6 Trigger Control

Two triggers are required to initiate image capture and retrieve images from the board—a Coarse Trigger followed by a Fine Trigger. These triggers can be provided to the board through three different methods: hardware triggers, software triggers, and a dual-edged hardware trigger.

## 6.1 Hardware Triggers

The default trigger mode requires the use of external equipment to provide hardware triggers. The hardware trigger requirements are:

- Hardware triggers are enabled by asserting the **HW_TRIG_EN** subregister while deasserting **DUAL_EDGE_TRIG_EN** and **SW_TRIG_EN** (i.e., setting bit 0 of **TRIGGER_CTL** while clearing bits 1 and 2.)
- Both triggers must adhere to TTL logic levels as measured on the board
- Both triggers must have a minimum 60 ns pulse width as measured on the board
- For Icarus sensor operation, a minimum of 32 µs is needed between the rising edge of the Coarse Trigger and the rising edge of the Fine Trigger to obtain a successful edge detect (w0_top_l_edge1) assertion after the assertion of the Fine Trigger. Please see the 'UXI_Icarus_FPA' document for more details regarding the edge detect signal from the Icarus sensor.[5]

  For Daedalus sensor operation, the Fine Trigger must be held low at least 16 µs for a successful edge detect (SH2_fall_UR). Please see 'UXI_Daedalus_HDD' document for more details regarding the edge detect signal from the Daedalus sensor.[6]
- The signal source must be able to drive a 50 Ω load since both triggers are terminated on the board.

## 6.2 Software Triggers

The Software Triggers are generated internally to the FPGA and exceed the requirements for the Hardware Triggers– the pulse width of both triggers is 10 µs and the rising edges occur 20 µs apart.

The software trigger is enabled by asserting the **SW_TRIG_EN** subregister while deasserting **HW_TRIG_EN** and **DUAL_EDGE_TRIG_EN** (clearing bits 0 and 1 of **TRIGGER_CTL** while setting bit 2). When subregister **SW_TRIG_START** is asserted, the software control logic will activate and assert first the Coarse Trigger then the Fine Trigger as described in the previous paragraph. **SW_TRIG_START** is self-clearing, so software does not need to clear it to disable this function.

Note that inside the FPGA, the hardware and software triggers are logically ORed together. Therefore, when the Software Trigger function is used, the user must not assert the Hardware Triggers.

## 6.3 Hardware Dual-Edge Trigger

When using the Hardware Dual-Edged Trigger, external equipment must drive only the Fine Trigger; the Coarse Trigger is not used. Dual-edge triggers are enabled by asserting subregisters **HW_TRIG_EN** and **DUAL_EDGE_TRIG_EN** and deasserting **SW_TRIG_EN** (setting bits 0 and 1 of **TRIGGER_CTL** while clearing bit 2.) When the dual-edge trigger is enabled, trigger signals will be handled in the following manner:

- External Coarse Triggers are ignored.
- For Icarus sensor operation, the Fine Trigger must be held low at least 32 µs for a successful edge detect (*w0_top_l_edge1*). Please see the 'UXI_Icarus_FPA' document for more details regarding the edge detect signal from the Icarus sensor.[5]
  For Daedalus sensor operation, the Fine Trigger must be held low at least 16 µs. For proper assertion of the edge detect (*SH2_fall_UR*, the diagnostic one-shot falling edge of shutter two), the Fine Trigger must be held high for at least 11.5 µs.[6]
- The FPGA detects the falling edge of the external Fine Trigger and treats this event internally as the Coarse Trigger.
- The FPGA detects the rising edge of the external Fine Trigger and treat this event internally as the Fine Trigger.

## 6.4 Readoff Delay

By default, the FPGA waits 40 ms between the detection of Fine Trigger by the FPGA before initiating image readoff from the sensor to the SRAM. This delay is configurable using the **TIME_ROW_DCD** register.

# 7 Digital-to-Analog Converter Channels

There are eight DAC channels on the board that are used to bias miscellaneous image sensor functions. The DAC is a TI8568 device; configuration is performed via the DAC data registers (**DAC_REG_A_AND_B**, **DAC_REG_C_AND_D**, **DAC_REG_E_AND_F**, and **DAC_REG_G_AND_H**), and the DAC control register (**DAC_CTL**).

To ensure that the DAC channels are programmed to valid values immediately upon turning on system power, the FPGA will detect the de-assertion of the system reset signal, and after a 1 ms delay will automatically program all DAC channels to the default values contained in their respective DAC data registers. See Sections 14 and for the Icarus and Daedalus DAC assignments, respectively. Specific voltages that the Daedalus sensor should bias are described in Table 13 in Section 15.2 .

When the user writes to the **DAC_CTL** register, a DAC write command is initiated. It takes some time for the command to be generated, serialized, sent to the DAC, and for the DAC interface module to then terminate the process. If a DAC write is initiated while a previous DAC write is still in progress, the second DAC write will neither be executed nor buffered; it will be lost. This is not an issue when using the RS422 interface due to the slow communications rate but is potentially a problem when using the Gigabit Ethernet interface. Interface software must not execute a DAC write command for at least 50 µs following a previous DAC write command when the Gigabit Ethernet interface is used.

# 8 Temperature Sensor

Two functionally identical temperature transducers in parallel are used to measure the temperature. The temperature transducers (AD592 and ISL71590SEH) differ in their radiation tolerance. Analog voltage measured at the output voltage can be converted to temperature via the conversion, 1 mV = 1 Kelvin since a 1 kΩ pull-down resistor is used. The on-board 12-bit ADC monitor, ADC128S102, converts the analog reading to digital and stores the reading into the appropriate section of a register, e.g., ADC5_DATA2(11:0).

In order to reduce the number of bits needed for storage, the temperature is recorded in Celsius rather than Kelvin, (the lowest temperature is never expected to be less than 0 degrees Celsius). This shifts the digital value stored lower by 339 (equivalent to 273.15 K) using the calculation:

$$Round\left[(273.15\ Kelvin)\left(\frac{2^{12} counts}{3.3\ V}\right)\left(\frac{1\ V}{1000\ mV}\right)\left(\frac{1\ mV}{1\ Kelvin}\right)\right] = 339\ counts$$

For example, assume that a 12-bit digital reading of `0x1C6` is read by the ADC monitor. The decimal conversion of `0x1C6` is 454 counts. We shift this down by subtracting 339 to get 115. This in turn can be converted to a temperature in Celsius:

$$Temp(^{\circ}C) = (measurement - 339\ counts)\left(\frac{3.3\ V}{2^{12}\ counts}\right) = \frac{(454 - 339) * 3.3\ V}{2^{12}}$$
$$= 92.65\ mV \rightarrow\ 92.65\ ^{\circ}C$$

Shifting the temperature range in this manner allows all expected temperature values to be represented in 7 bits, without requiring a change of scale. The resulting value is stored in the **STAT_REG_SRC** status register and is accessible from the **STAT_TEMP** subregister.

After the system is powered on, the FPGA will continually read the temperature transducers at an interval determined by the **ADC_PPER** register.

# 9 Pressure Sensor

A Honeywell NBPLPNN030PAUNV absolute-type pressure sensor is used to measure the board's environmental pressure. This device exhibits a maximum measurement of 30 PSI. The 12-bit ADC monitor, ADC128S102, polls the monitor of two separate pressure outputs: pressure positive and pressure negative. The polling is performed in the same manner as the temperature sensor. The calculation to convert the monitored readings in mV to PSI using the difference of pressure positive and pressure negative is as follows:

Given a nominal sensitivity of the sensor of $21\frac{mV}{V}$, the dimensional conversion from volts to PSI is

$$21\ \frac{mV}{V/_{span}}\left(\frac{5\ V/_{span}}{30\ PSI}\right) = 3.5\frac{mV}{PSI}$$

The sensitivity of any particular pressure sensor can be anywhere in the range between 15.5 and 26.0 mv/V/span, and there is a potential offset of ±7 mv/V, so accurate pressure readings require sensor calibration.

The maximum pressure reportable by the sensor is 30 PSI, thus the maximum voltage difference is calculated as 140 mV. In hexadecimal this is 0x8C. Since this can be represented in eight bits, we can neglect the higher four bits of the twelve-bit monitor and store the results in eight bits of register space.

# 10 Packet Formats

The NSGCC FPGA supports three packet types: Command, Response, and Burst Response (i.e., pixel) Packets. Their formats and usage are described in this section.

## 10.1 Command Packet

A Command Packet is sent by the host to the FPGA; it may not be sent by the FPGA. A Command Packet is used to send an instruction to the FPGA for execution. All Command Packets shall have the format shown in Table 3.

| 16 bits | 4 bits | 12 bits | 32 bits | 16 bits |
|---|---|---|---|---|
| Preamble | Command | Address | Data | CRC16 |

Table 3: Command Packet Format

| | |
|---|---|
| **Preamble** | Bit pattern that precedes actual packet data to assist the receiver in determining the start of the packet. The preamble is fixed to 0xAAAA |
| **Command** | 0x0: Write Single<br>0x1: Read Single<br>0x2: Read Burst (i.e., Read Pixels)<br>All other values not supported |
| **Address** | Bit[11:0]: Defines the address of the target register of a Write Single or Read Single command. For a Read Burst command, this field is not used, but is recommended that it be filled with zeros. |
| **Data** | Bit[31:0]: Contains write data for a Write Single command. This field is not used for Read Single or Read Burst commands, but it is recommended that it be filled with zeros. |

| | |
|---|---|
| **CRC16** | Bit[15:0]: CRC-16 field, calculated over the entire packet, excluding the preamble and CRC field itself. The CRC16 field exists for RS422 packets only; for Ethernet packets, the CRC16 field does not exist. |

Table 4: Command Packet Fields

## 10.2 Response Packet

A response packet is sent by the FPGA to the host; it may not be sent by the host. Response Packets are sent either (1) in response to a Write Single packet where the response packets are not disabled, or (2) in response to a Read Single packet.

The Response Packet format is similar to the Command Packet format; this enables host software to easily correlate a Response Packet to the Command Packet that was the cause of its generation. However, there are a couple of minor differences, such as asserting the MSB of the Command field, and the content of the Data field.

| 16 bits | 4 bits | 12 bits | 32 bits | 16 bits |
|---|---|---|---|---|
| Preamble | Command | Address | Status | CRC16 |

Table 5: Response Packet Format

| | |
|---|---|
| **Preamble** | Bit pattern that precedes actual packet data to assist the receiver in determining the start of the packet. The preamble is fixed to 0xAAAA |
| **Command** | Contains the contents of the command field of the corresponding Command Packet, except that the MSB is asserted.<br>0x8: Write Single<br>0x9: Read Single<br>0xA: Read Burst (i.e., Read Pixels) |
| **Address** | Same as the source Command Packet |
| **Status** | For Read Single Commands: This field contains the data read from the target register.<br>For Write Single Commands: This field contains status information, particularly errors, contained in the transmitted command packet. This status information does NOT refer to the response packet.<br>Bit[0]: CRC error<br>Bit[1]: Invalid Command – command not executed<br>Bit[2]: Invalid Sub-Command – command not executed |
| **CRC16** | Bit[15:0]: CRC-16 field, calculated over the entire packet, excluding the preamble and CRC field itself. The CRC16 field exists for RS422 packets only; for Ethernet packets, the CRC16 field does not exist. |

Table 6: Response Packet Fields

## 10.3 Burst Response Packet

Burst Response (or Pixel) Packets are sent in response to a Read Burst (or Read Pixels) command.

| 16 bits | 4 bits | 12 bits | 32 bits | Variable | 16 bits |
|---|---|---|---|---|---|
| Preamble | Command | Reserved | Payload Length | Payload | CRC16 |

Table 7: Burst Response Packet Format

| | |
|---|---|
| **Preamble** | Bit pattern that precedes actual packet data to assist the receiver in determining the start of the packet. The preamble is fixed to 0xAAAA |
| **Command** | Contains the Command field of the Command Packet, except that the MSB is asserted.<br>0xA: Read Burst (i.e. Read Pixels) |
| **Reserved** | Field is not used, should be set to 0x000 |
| **Payload Length** | Length of the Payload field, in bytes. This is the total number of bytes transmitted for a particular SRAM readout. |
| **Payload** | Payload. This field contains pixel data. Each pixel occupies 16-bits of payload; if the actual pixel data is less than 16 bits, the pixel data shall be zero-justified |
| **CRC16** | CRC-16 field, calculated over the entire packet, excluding the preamble and CRC field itself. The CRC16 field exists for RS422 packets only; for Ethernet packets, the CRC16 field does not exist. |

Table 8: Burst Response Fields

# 11 Instructions

Three instructions are currently supported: Write Single, Read Single, and Read Burst.

The Write Single instruction is used by the host to update and modify the NSGCC FPGA's control registers. By writing to the appropriate control registers in the correct sequence, the host can control all NSGCC FPGA and Version 4.0 Board functions. When the FPGA receives a command packet with a Write Single instruction with response packets enabled, it will return a response packet indicating reception of the packet and whether it was received error-free.

The Read Single instruction is used to read the content of a single NSGCC control or status register. This instruction enables the host to determine the status of all FPGA functions that are supported. When a command packet with a Read Single instruction is received by the FPGA, it *must* return a response packet, which contains the FPGA target register contents. Again, it is up to the host to determine a suitable timeout period while awaiting the response packet and to re-send the packet if required.

The Read Burst instruction is used to read sensor data from the Version 4.0 Board's SRAM; it should be sent by the host only after pixel data is read from the ADC and stored in SRAM. When this instruction is received by the FPGA via a command packet, it will return a single of Burst Response packets with the format described in the previous section.

# 12 Registers

This section lists all NSGCC FPGA registers accessible by software. All registers are 32 bits wide, although not all bits are used in every register. The different register types are defined as follows:

- Read Only: Software can read the register but cannot modify its contents. The register's contents are updated/modified only by internal FPGA hardware.
- Read/Write: Software can read or write the register; hardware cannot update/modify the register contents, unless stated.
- Read Clear: Software can read the register; the register's contents are cleared (i.e., reset to zeros) when read by software. However, if software has not resolved the underlying cause of asserted status bits, reading this type of register may not result in all zeros being read from it. A typical example is an interrupt register, where the underlying source of the interrupt has not been cleared.

## 12.1 Register Map

A pale green background indicates an Icarus-specific setting; pale gold indicates a Daedalus-specific setting. Unless 'Icarus2' is specified in an entry, 'Icarus' refers to both Icarus and Icarus2.

| *Address* | *Register Name* | | *Board* | *Access* | *Default value* |
|---|---|---|---|---|---|
| *Register description* | *Bit range* | *Details of bit range (may include **SUBREGISTER_NAME**)* | | | |
| **0x000** | **FPGA_NUM** | | V1, V4 | | 0x8400_0301 |
| Product Number of the FPGA design<br>Eight-character sequence:<br>1: Board developer<br>2: Board revision number<br>3-5: unused<br>6: Communication interfaces<br>7: Radiation tolerance<br>8: Sensor build | 31 | Board developer | | | |
| | | 0 SNL | | | |
| | | 1 LLNL | | | |
| | 30:28 | Unused | | | |
| | 27:24 | Board major revision number | | | |
| | | 0001 LLNLv1 | | | |
| | | 0100 LLNLv4 | | | |
| | 23:10 | Unused | | | |
| | 9 | '1' indicates Gigabit Ethernet interface implemented | | | |
| | 8 | '1' indicates RS422 interface implemented | | | |
| | 7:5 | Unused | | | |
| | 4 | Radiation tolerance. '1' indicates optimized radiation-tolerant implementation | | | |
| | 3:0 | Sensor implementation | | | |
| | | 0000 Undefined | | | |
| | | 0001 Icarus / Icarus 2 | | | |
| | | 0010 Daedalus | | | |
| | | 0011 Reserved | | | |
| **0x001** | **FPGA_REV** | | V1, V4 | Read-only | --- |
| Revision of FPGA design. | 7:0 | Day of FPGA code release (ex: 0x29 for the 29th day of the month) | | | |
| | 15:8 | Month of FPGA code release (ex: 0x12 for the month of December) | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | 23:16 | Year of FPGA code release (ex: 0x18 for the year 2018) | | |
| | | 27:24 | Unused | | |
| | | 31:28 | Board version (ex: 0x4 for v4 board) | | |
| **0x010** | **HS_TIMING_CTL** | | V1, V4 | Read/Write | 0x0000_0000 |
| Control of HS Timing Function | | 0 | **HST_MODE** - Configure timing. This bit is self-clearing. When '1', HST configuration is initiated with respect to the FPA interface. | | |
| **0x013** | **HS_TIMING_DATA_ALO** | | V1, V4 | Read/Write | 0x0000_0000 |
| Custom high-speed timing A side, LSBs | | 31:0 | Timing pattern bits [31:0] for A side | | |
| **0x014** | **HS_TIMING_DATA_AHI** | | V1, V4 | Read/Write | 0x0000_0000 |
| Custom high-speed timing A side, MSBs | | 7:0 | Timing pattern bits [39:32] for A side | | |
| **0x015** | **HS_TIMING_DATA_BLO** | | V1, V4 | Read/Write | 0x0000_0000 |
| Custom high-speed timing B side, LSBs | | 31:0 | Timing pattern bits [31:0] for B side | | |
| **0x016** | **HS_TIMING_DATA_BHI** | | V1, V4 | Read/Write | 0x0000_0000 |
| Custom high-speed timing B side, MSBs | | 7:0 | Timing pattern bits [39:32] for B side | | |
| **0x017** | **SW_TRIGGER_CONTROL** | | V1, V4 | Write-only | --- |
| Initiates generation of internal coarse and fine triggers | | 0 | **SW_TRIG_START** - When written with '1', initiates coarse and fine triggers internal to FPGA. The coarse trigger will be 10µs long, followed by a delay of 11.5 µs, then followed by a 10 µs fine trigger. | | |
| **0x018** | **HST_READBACK_A_LO** | | V1, V4 | Read-only | --- |
| HST configuration readback | | 31:0 | HST configuration readback with respect to RSL state machine (bits [31:0] for A side) | | |
| **0x019** | **HST_READBACK_A_HI** | | V1, V4 | Read-only | --- |
| HST configuration readback | | 7:0 | HST configuration readback with respect to RSL state machine (bits [39:32] for A side) | | |
| **0x01A** | **HST_READBACK_B_LO** | | V1, V4 | Read-only | --- |
| HST configuration readback | | 31:0 | HST configuration readback with respect to RSL state machine (bits [31:0] for B side) | | |
| **0x01B** | **HST_READBACK_B_HI** | | V1, V4 | Read-only | --- |
| HST configuration readback | | 7:0 | HST configuration readback with respect to RSL state machine (bits [39:32] for B side) | | |
| **0x01C** | **SW_COARSE_CONTROL** | | V4 | Write-only | --- |
| Software Coarse trigger control | | 0 | **SW_COARSE_TRIGGER** - When written with '1', initiates a software coarse trigger | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x024** | **STAT_REG** | | V1, V4 | Read-only | --- |
| Status Register.<br>(Read-only duplicate of STAT_REG_SRC) | 31:0 | Read-only shadow bits of **STAT_REG_SRC** (0x02F). Reading this register has no effect on these bit values. To clear the applicable bits, **STAT_REG_ SRC** must be read. See 0x02F for bit assignments | | | |
| **0x025** | **CTRL_REG** | | V1, V4 | Read/Write | 0x0000_0000 |
| Control Register | 0-1 | Unused | | | |
| | 2 | **COLQUENCHEN** - Column Quench Enable. When '1', enables column quench function | | | |
| | 3 | **POWERSAVE** - Power Save Mode. Controls the assertion of *HST_osc_bias_en* to save power. | | | |
| | | 0 | *HST_osc_bias_en* is tied high continuously | | |
| | | 1 | *HST_osc_bias_en* is asserted upon the rising edge of the Coarse Trigger; *HST_osc_bias_en* is deasserted when sensor readout begins | | |
| | 4 | **REVREAD** - When '1', reverses the frame readout order | | | |
| | 4 | **SLOWREADOFF_0-** Part of a test register for slowing down image readoff. Bit 4 is the LSB and Bit 5 is the MSB. If bits 4 and 5 are set as "01", then readoff is slowed by a factor of 2. If bits 4 and 5 are set as "10", then readoff is slowed by a factor of 3. | | | |
| | 5 | **SLOWREADOFF_1-** Part of a test register for slowing down image readoff. Bit 4 is the LSB and Bit 5 is the MSB. If bits 4 and 5 are set as "01", then readoff is slowed by a factor of 2. If bits 4 and 5 are set as "10", then readoff is slowed by a factor of 3. | | | |
| | 6 | **PDBIAS_LOW** – When '1', the PDBIAS +49 V regulator will be disabled. When '0', the regulator will be enabled. | | | |
| | 7 | **ROWDCD_CTL** – When '1', the *row_dcd-enable* input to the sensor will be asserted indefinitely. When '0', the input to the sensor will be controlled by the FPGA's FPA interface. | | | |
| **0x026** | **DAC_CTL** | | V4 | Read/Write | 0x0000_0000 |
| DAC configuration control for specific channels | 0 | DAC_CONFIG. When written with a '1', the DAC selected by DAC_SEL will be configured with the value in its corresponding register. This bit is self-clearing. | | | |
| | 3:1 | DAC_SEL[2:0]. Selects the DAC channel to configure. Selection is sequential, i.e.,<br>"000" = select DAC A, "001" = select DAC B … "111" = select DAC H | | | |
| **0x027** | **DAC_REG_A_AND_B** | | V4 | Read/Write | --- |
| DAC A and B configuration data | 15:0 | **DACB / HST_A_NDELAY** – see register 0x09C. Control voltage to respective sensor pin. It is the A side n transistor delay buffer voltage. Decrease in voltage increases delay of side n transistor.[5] | | | 3.3 V |

| | | | | | |
|---|---|---|---|---|---|
| See Section 14.2 and 15.2 for DAC channel description of Icarus and Daedalus implementations respectively. | 31:16 | **DACA / HST_A_PDELAY** – see register `0x099`. Control voltage to respective sensor pin. It is the A side p transistor delay buffer voltage. Decrease in voltage increases delay of side n transistor.[5] | | | 0 V |
| **0x028** | **DAC_REG_C_AND_D** | | V4 | Read/Write | --- |
| DAC C and D configuration data<br>See Section 14.2 and 15.2 for DAC channel description of Icarus and Daedalus implementations respectively. | 15:0 | **DACD / HST_B_NDELAY** – see register `0x09A`. Control voltage to respective sensor pin. It is the B side n transistor delay buffer voltage. Decrease in voltage increases delay of side n transistor.[5] | | | 3.3 V |
| | 31:16 | **DACC / HST_B_PDELAY** – see register `0x09B`. Control voltage to respective sensor pin. It is the B side p transistor delay buffer voltage. Decrease in voltage increases delay of side n transistor.[5] | | | 0 V |
| | 31:16 | **DACC / HST_OSC_VREF_IN** – see register `0x09B`. Reference voltage for 500 MHz oscillator.[6] | | | 2.9 V |
| **0x029** | **DAC_REG_E_AND_F** | | V4 | Read/Write | --- |
| DAC E and F configuration data<br>See Section 14.2 and 15.2 **Error! Reference source not found.**for DAC channel description of Icarus and Daedalus implementations respectively. | 15:0 | **DACF / HST_OSC_CTL** – see register `0x09C`. Control voltage to the relaxation oscillator. Decrease in voltage increases the speed of oscillator.[5] | | | 1.45 |
| | 15:0 | **DACF/ COL_TST_IN**– see register `0x09C`. Global column current source analog test input pin.[6] | | | 0 V |
| | 31:16 | **DACE / HST_RO_IBIAS / HST_RO_NC_IBIAS** – see register `0x098`. Control voltage to either the ring with capacitors or without capacitors oscillator determined by register 0x047. Decrease in voltage increases the speed of oscillator.[5] | | | 2.5 V |
| | 31:16 | **DACE / HST_OSC_CTL** – see register `0x098`. Control voltage to the 500 MHz oscillator. Decrease in voltage increases the speed of oscillator. Increase in voltage decreases the speed of oscillator.[6] | | | 1.0 V |
| **0x02A** | **DAC_REG_G_AND_H** | | V4 | Read/Write | --- |
| DAC G and H configuration data<br>See Section 14.2 and 15.2 for DAC channel description of Icarus and Daedalus implementations respectively. | 15:0 | **DACH / VRST** – see register 0x098. Controls the pixel reset voltage.[5] | | | .3 V |
| | 31:16 | **DACG / VAB –** see register 0x097. Controls the pixel anti-bloom transistor voltage.[5, 6] | | | .5 V |
| **0x02D** | **SW_RESET** | | V1, V4 | Write-only | `0x0000_0000` |

| | | | | | |
|---|---|---|---|---|---|
| Software reset | | 0 | **RESET** - *sw_rst.* When asserted, will reset the entire FPGA, including control and status registers. This bit will be automatically cleared after written. | | |
| **0x02E** | **HST_SETTINGS** | | V1, V4 | Read-only | --- |
| High Speed timing control | | 0 | **HST_SW_CTL_EN** - When '1', the hstAllWEn pin to the sensor will be directly controlled by the *sw_hst_all_wen* bit. When '0', hstAllWEn will be controlled by FPGA logic. | | |
| | | 1 | **SW_HSTALLWEN** - Will directly drive the hstAllWEn pin to the sensor when *HST_sw_ctl_en* is '1'; e.g., when both HST_SW_CTL_EN and SW_HSTALLWEN are '1', then the hstAllWEn pin to the sensor will be driven to a logical '1' (i.e. high) | | |
| **0x02F** | **STAT_REG_SRC** | | V1, V4 | Read-clear | --- |
| Status Register, Source.<br><br>Contains the source logic for clearable status bits, whereas STAT_REG contains read-only copies. All bits in STAT_REG_SRC register will be cleared when the register is read, except for the Temperature Sensor and the Pressure Sensor bits. Use register `0x02F` to read these bits | | 0 | **SRAM_READY** - sensor readout is complete | | |
| | | 1 | **STAT_COARSE** - Coarse Trigger detected | | |
| | | 2 | **STAT_FINE** - Fine Trigger detected | | |
| | | 3 | **STAT_W3TOPLEDGE1** - w3_top_L_edge1 (GPIO38) detected | | |
| | | 3 | **STAT_RSLROWOUTL** - Row interlacing output with respect to sensor and input with respect to FPGA to an abutted FPA (left side). Currently, the FPGA ties this to '0' since the current Daedalus sensor does not support RSL programming. | | |
| | | 4 | **STAT_W3TOPREDGE1** - w3_top_R_edge1 (GPIO35) detected | | |
| | | 4 | **STAT_RSLROWOUTR** - Row interlacing output with respect to sensor and input with respect to FPGA to an abutted FPA (right side). Currently, the FPGA ties this to '0' since the current Daedalus sensor does not support RSL programming. | | |
| | | 5 | **STAT_SENSREADIP** - Sensor Readout In Progress; Indicates the start of an ADC read cycle in which 32 pixels will be read from the sensor (8 channels x 4 ADCs) | | |
| | | 6 | **STAT_SENSREADDONE** - Sensor Readout Complete – Asserted by ADC control logic; indicates that sensor readout is complete | | |
| | | 7 | **STAT_SRAMREADSTART** - SRAM Readout Started - Indicates that SRAM readout has started. This is tied to bit 0 of the SRAM_CTL register, which is controlled by software | | |
| | | 8 | **STAT_SRAMREADDONE** - SRAM Readout Complete – Indicates that SRAM readout is complete (all pixels have been read out of SRAM) | | |
| | | 9 | **STAT_HSTCONFIGURED** - HST Configured | | |
| | | 10 | **STAT_ADCSCONFIGURED** - ADC's Configured – Asserted when all five ADCs have been configured | | |

| | | | | | |
|---|---|---|---|---|---|
| | 11 | **STAT_DACSCONFIGURED** – all DACs have been configured | | | |
| | 12 | **STAT_HST_ALL_W_EN_DETECTED** - *hst_all_w_en* detected | | | |
| | 12 | **STAT_RSLNALLWENR -** AllWEn enables all shutter signals to initialize storage caps (right side) | | | |
| | 13 | **STAT_TIMERCOUNTERRESET** – indicates that the timer (see 0x03C and 0x03D) has been reset | | | |
| | 14 | **STAT_ARMED** - 'ADCs configured' AND 'DACs configured' AND **TRIGGER_CTL**[0] & 'HST_Configured' AND NOT 'coarse trigger detected' AND NOT 'fine trigger detected' AND 'PDBIAS is read' (for Icarus only). Note that all these conditions must be met for ARMED to be asserted. | | | |
| | 15 | **STAT_RSLNALLWENL -** AllWEn enables all shutter signals to initialize storage caps (left side) | | | |
| | 16 | **STAT_CONFIGHSTDONE** – Indicates that both HST and RSL is configured. | | | |
| | 23:17 | **STAT_TEMP** - Temperature Sensor [6:0] from **ADC5_DATA_2**[11:0] (0x096); see Section 8 . | | | |
| | 31:24 | **STAT_PRESS** - Pressure Sensor difference [7:0] from the absolute value difference between MON_PRES_MINUS and MON_PRES_PLUS (see 0x095); see Section 9 . | | | |
| **0x030** | **STAT_REG2** | | V1, V4 | Read-only | --- |
| Status Register 2. (Read-only duplicate of **STAT_REG2_SRC**) | 31:0 | Read-only shadow bits of **STAT_REG2_SRC**. Reading this register has no effect on these bit values. To clear the applicable bits, **STAT_REG2_ SRC** must be read. See 0x031 for bit assignments | | | |
| **0x031** | **STAT_REG2_SRC** | | V1, V4 | Read-clear | |
| Status Register 2, Source.<br><br>Contains the source logic for clearable status bits, whereas **STAT_REG2** contains read-only copies. All bits in **STAT_REG2_SRC** register will be cleared when the register is read. See Section 13 for additional details. Use register 0x030 to read these bits | 0 | **FPA_IF_TO** - When this bit is asserted high, a timeout error has occurred during the sensor-to-SRAM readout process | | | |
| | 1 | **SRAM_RO_TO** - When asserted high, this bit indicates that a timeout has occurred while reading an SRAM row. | | | |
| | 2 | **PIXELRD_TOUT_ERR** - When asserted high, this bit indicates that the overall sensor readout process has timed out. It also indicates that the internal transmit data pipeline has reverted from selecting Burst Response (i.e., pixel) data back to Response (i.e., status) data to re-establish communications with the host. | | | |
| | 3 | **UART_TX_TO_RST** - When asserted high and the RS422 port is enabled, this bit indicates that a timeout has occurred within the RS422 transmit logic, and the UART TX module has reset itself to recover from the condition. | | | |
| | 4 | **UART_RX_TO_RST** - When asserted high and the RS422 port is enabled, this bit indicates that a timeout has occurred within the RS422 receive logic, and the UART RX module has reset itself to recover from the condition. | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | 5 | **PDBIAS_UNREADY** – When '1', PDBIAS is considered less than 10 V from the 49 V regulator and therefore is considered not ready. When '0', the 49 V regulator is greater than or equal to 10 V and is therefore considered ready | | |
| **0x032** | **ADC_BYTECOUNTER** | | | V1, V4 | Read-only | --- |
| ADC Byte Counter | | 25:0 | *adc_bytecounter.* This is the byteCounter output of the ADC/SRAM readout module which counts the pixels (in bytes) that have been read from the image sensor and written into the SRAM. If this register is read after a system "hang" has been detected (and prior to a reset), it can indicate if the error occurred during sensor readoff. If it contains a value of zero, then the ADC/SRAM readout module has likely operated normally. If it is a non-zero value, then this may indicate that the error was caused by a failure in the ADC/SRAM module, where the value contained is the pixel/byte number that the image capture hung on. Since this counter is an integral part of the FPGA logic (rather than a test-only feature), it can only be cleared with a system reset. | | |
| **0x033** | **RBP_PIXEL_CNTR** | | | V1, V4 | Read-only | --- |
| Read Burst Processing Pixel Counter | | 23:0 | This is the *pixel_cntr* output of the steering/read_burst_processing module which counts the pixels (in bytes) that have been read from the transmit FIFO after being read from the SRAM. If this register is read after a system "hang" has been detected (and prior to a reset), it can indicate if the error occurred during SRAM readoff, and if the error occurred at the TX FIFO output in the readout pipeline. If it contains a value of zero, then the ADC/SRAM module has likely operated normally, and the error originated elsewhere. If it is a non-zero value, then this may indicate that the error was caused by a failure in the steering/read_burst_processing module, where the value contained is the pixel/byte number that the image capture hung on. Since this counter is an integral part of the FPGA logic (rather than a test-only feature), it can only be cleared with a system reset. | | |
| **0x034** | **DIAG_MAX_CNT_0** | | | V1, V4 | Read/Write | --- |
| Diagnostic Max Count Register 0 | | 7:0 | **MAXERR_SRT** - Maximum number of errors indicated by *sram_ro_to* (see `0x031`) before the counter freezes. Maximum value allowed is `0xFF`. | | |
| | | 15:8 | Unused | | |
| | | 31:16 | **MAXERR_FIT** - Maximum number of errors indicated by *uart_tx_to_rst* (see `0x031`) before the counter freezes. Maximum value allowed is `0xFFFF`. | | |
| **0x035** | **DIAG_MAX_CNT_1** | | | V1, V4 | Read/Write | --- |

| | | | | | |
|---|---|---|---|---|---|
| Diagnostic Max Count Register 1 | 15:0 | **MAXERR_URTR** - Maximum number of errors indicated by *uart_rx_to_rst* (see 0x031) before the counter freezes. Maximum value allowed is 0xFFFF. | | | |
| | 31:16 | **MAXERR_UTTR** - Maximum number of errors indicated by *fpa_if_to* (see, 0x031) before the counter freezes. Maximum value allowed is 0xFFFF. | | | |
| **0x036** | **DIAG_CNTR_VAL_0** | | V1, V4 | Read-only | --- |
| Diagnostic Counter Value 0 | 7:0 | **SRT_COUNT** - Current value of *sram_ro_to* counter, which increments when *sram_ro_to* is asserted and is reset only with a system reset. Maximum value is 0xFF; when this value is reached, the counter will freeze until reset. | | | |
| | 15:8 | Unused | | | |
| | 31:16 | **FIT_COUNT** - Current value of *fpa_if_to* counter, which increments when *fpa_if_to* is asserted and is reset only with a system reset. Maximum value is 0xFFFF; when this value is reached, the counter will freeze until reset. | | | |
| **0x037** | **DIAG_CNTR_VAL_1** | | V1, V4 | Read-only | --- |
| Diagnostic Counter Value 1 | 15:0 | **URTR_COUNT** - Current value of uart_rx_to_rst counter, which increments when *uart_rx_to_rst* is asserted and is reset only with a system reset. Maximum value is 0xFFFF; when this value is reached, the counter will freeze until reset. | | | |
| | 31:16 | **UTTR_COUNT** - Current value of uart_tx_to_rst counter, which increments when *uart_tx_to_rst* is asserted and is reset only with a system reset. Maximum value is 0xFFFF; when this value is reached, the counter will freeze until reset. | | | |
| **0x038** | **STAT_EDGE_DETECTS** | | V4 | Read-only | --- |
| Status of thirty-one out of thirty-two Icarus edge detect signals based on the rising edge of the FPGA system clock. Sensor must be attached to monitor edge detects. Refer to Icarus HDD document for reference on edge detect signals.[5] | 0 | **W0_TOP_R_EDGE1 -** Diagnostic One Shot: Rising edge of w0 shutter into right/top side of FPA. | | | |
| | 1 | **W0_TOP_L_EDGE2 -** Diagnostic One Shot: Falling edge of w0 shutter into left/top side of FPA. | | | |
| | 2 | **W0_TOP_R_EDGE2 -** Diagnostic One Shot: Falling edge of w0 shutter into right/top side of FPA. | | | |
| | 3 | **W0_BOT_L_EDGE1 -** Diagnostic One Shot: Rising edge of w0 shutter into left/bottom side of FPA. | | | |
| | 4 | **W0_BOT_R_EDGE1 -** Diagnostic One Shot: Rising edge of w0 shutter into right/bottom side of FPA. | | | |
| | 5 | **W0_BOT_L_EDGE2 -** Diagnostic One Shot: Falling edge of w0 shutter into left/bottom side of FPA. | | | |
| | 6 | **W0_BOT_R_EDGE2 -** Diagnostic One Shot: Falling edge of w0 shutter into right/bottom side of FPA. | | | |
| | 7 | **W1_TOP_L_EDGE1 -** Diagnostic One Shot: Rising edge of w1 shutter into left/top side of FPA. | | | |
| | 8 | **W1_TOP_R_EDGE1 -** Diagnostic One Shot: Rising edge of w1 shutter into right/top side of FPA. | | | |
| | 9 | **W1_TOP_L_EDGE2 -** Diagnostic One Shot: Falling edge of w1 shutter into left/top side of FPA. | | | |

| | | |
|---|---|---|
| | 10 | **W1_TOP_R_EDGE2 -** Diagnostic One Shot: Falling edge of w1 shutter into right/top side of FPA. |
| | 11 | **W1_BOT_L_EDGE1 -** Diagnostic One Shot: Rising edge of w1 shutter into left/bottom side of FPA. |
| | 12 | **W1_BOT_R_EDGE1 -** Diagnostic One Shot: Rising edge of w1 shutter into right/bottom side of FPA. |
| | 13 | **W1_BOT_L_EDGE2 -** Diagnostic One Shot: Falling edge of w1 shutter into left/bottom side of FPA. |
| | 14 | **W1_BOT_R_EDGE2 -** Diagnostic One Shot: Falling edge of w1 shutter into right/bottom side of FPA. |
| | 15 | **W2_TOP_L_EDGE1 -** Diagnostic One Shot: Rising edge of w2 shutter into left/top side of FPA. |
| | 16 | **W2_TOP_R_EDGE1 -** Diagnostic One Shot: Rising edge of w2 shutter into right/top side of FPA. |
| | 17 | **W2_TOP_L_EDGE2 -** Diagnostic One Shot: Falling edge of w2 shutter into left/top side of FPA. |
| | 18 | **W2_TOP_R_EDGE2 -** Diagnostic One Shot: Falling edge of w2 shutter into right/top side of FPA. |
| | 19 | **W2_BOT_L_EDGE1 -** Diagnostic One Shot: Rising edge of w2 shutter into left/bottom side of FPA. |
| | 20 | **W2_BOT_R_EDGE1 -** Diagnostic One Shot: Rising edge of w2 shutter into right/bottom side of FPA. |
| | 21 | **W2_BOT_L_EDGE2 -** Diagnostic One Shot: Falling edge of w2 shutter into left/bottom side of FPA. |
| | 22 | **W2_BOT_R_EDGE2 -** Diagnostic One Shot: Falling edge of w2 shutter into right/bottom side of FPA. |
| | 23 | **W3_TOP_L_EDGE1 -** Diagnostic One Shot: Rising edge of w3 shutter into left/top side of FPA. |
| | 24 | **W3_TOP_R_EDGE1 -** Diagnostic One Shot: Rising edge of w3 shutter into right/top side of FPA. |
| | 25 | **W3_TOP_L_EDGE2 -** Diagnostic One Shot: Falling edge of w3 shutter into left/top side of FPA. |
| | 26 | **W3_TOP_R_EDGE2 -** Diagnostic One Shot: Falling edge of w3 shutter into right/top side of FPA. |
| | 27 | **W3_BOT_L_EDGE1 -** Diagnostic One Shot: Rising edge of w3 shutter into left/bottom side of FPA. |
| | 28 | **W3_BOT_R_EDGE1 -** Diagnostic One Shot: Rising edge of w3 shutter into right/bottom side of FPA. |
| | 29 | **W3_BOT_L_EDGE2 -** Diagnostic One Shot: Falling edge of w3 shutter into left/bottom side of FPA. |
| | 30 | **W3_BOT_R_EDGE2 -** Diagnostic One Shot: Rising edge of w3 shutter into right/bottom side of FPA. |

| **0x03A** | **TRIGGER_CTL** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|
| Trigger control | 0 | **HW_TRIG_EN** - When '1' with rest of TRIGGER_CTL bits cleared, coarse and fine triggers are passed to internal logic; when '0', triggers are ignored. | | |
| | 1 | **DUAL_EDGE_TRIG_EN** - When '1' (while HW_TRIG_EN is also '1'), only the fine trigger is required for proper operation. However, the timing of the fine trigger must adhere to the timing described in the Trigger section. The trigger accepts the first edge whether rising or falling edge. The first edge initiates an internal coarse trigger of 10 µs. The second edge initiates an internal fine trigger 10 µs. The internal width between the internal coarse trigger rising edge and internal fine trigger rising edge is 100 µs. | | |
| | 2 | **SW_TRIG_EN** - When '1' with rest of **TRIGGER_CTL** bits cleared, asserting **SW_TRIG_START** will cause the FPGA to generate a coarse and fine trigger. | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x03B** | **SRAM_CTL** | | V1, V4 | Write; self-clearing | 0x0000_0000 |
| Request SRAM Readoff | 0 | **READ_SRAM** - Request SRAM Data when '1'. This bit is self-clearing.<br>When using RS422, software should send no additional command packets after setting this but until all expected burst response data is received. | | | |
| **0x03C** | **TIMER_CTL** | | V1, V4 | Write; self-clearing | --- |
| Timer control register | 0 | **RESET_TIMER** - Resets counter when set to '1'. This bit is self-clearing. | | | |
| **0x03D** | **TIMER_VALUE** | | V1, V4 | Read-only | --- |
| Current value of timer | 23:0 | Current timer counter value. Increments every second | | | |
| **0x03E** | **VRESET_WAIT_TIME** | | V1, V4 | Read/Write | 0x0000_0000 |
| Time to wait for VRESET to ramp high. | 30:0 | *Icarus1 only*. Time to wait for VRESET to ramp high, in 25 ns increments. This register is used to recover the image in a 2-frame Icarus | | | |
| **0x03F** | **HSTALLWEN_WAIT_TIME** | | V1, V4 | Read/Write | 0x0000_0190 |
| hstAllWEn active time | 30:0 | Time for *hstAllWEn* to be active, in 25 ns increments during pixel initialization | | | |
| **0x041** | **ICARUS_VER_SEL** | | V1, V4 | Read/Write | 0x0000_0000 |
| Selects ICARUS type, either 4 or 2 frame version | 0 | 0 | 4-frame 'Icarus2' | | |
| | | 1 | 2-frame 'Icarus' | | |
| **0x042** | **FPA_ROW_INITIAL** | | V1, V4 | Read/Write | 0x0000_0000 |
| The initial pixel row to read off the SRAM | 9:0 | Initial row, between 0 and 1023 (0x000 – 0x3FF) for Icarus and Daedalus | | | |
| **0x043** | **FPA_ROW_FINAL** | | V1, V4 | Read/Write | 0x0000_03FF |
| The final pixel row to read off the SRAM | 9:0 | Final row, between 0 and 1023 (0x000 – 0x3FF) for Icarus and Daedalus. Must be greater than or equal to **FPA_ROW_INITIAL** | | | |
| **0x044** | **FPA_FRAME_INITIAL** | | V1, V4 | Read/Write | 0x0000_0000 |
| The initial pixel frame to read off the SRAM | 1:0 | Initial frame, between 0 and 3 for Icarus2, between 1 and 2 for Icarus, between 0 and 2 for Daedalus | | | |
| **0x045** | **FPA_FRAME_FINAL** | | V1, V4 | Read/Write | 0x0000_0003 |
| The final pixel frame to read off the SRAM | 1:0 | Final frame, between 0 and 3 for Icarus2, between 1 and 2 for Icarus, between 0 and 2 for Daedalus. Must be greater than or equal to **FPA_FRAME_INITIAL** | | | |
| **0x046** | **FPA_DIVCLK_EN_ADDR** | | V1, V4 | Read/Write | 0x0000_0000 |

| | | | | | |
|---|---|---|---|---|---|
| Enable the HST *divClk* output | 0 | 0 | Disable HST *divClk* | | |
| | | 1 | Enable HST *divClk* | | |
| **0x047** | **FPA_OSCILLATOR_SEL_ADDR** | | V1, V4 | Read/Write | 0x0000_0000 |
| Select the oscillator for the ROIC. | 1:0 | 00 | Relaxation/500 MHz oscillator | | |
| | | 01 | Ring/100 MHz oscillator | | |
| | | 10 | Ring oscillator (without caps) | | |
| | | 11 | External clock | | |
| **0x04A** | **VRESET_HIGH_VALUE** | | V1, V4 | Read/Write | 0x0000_0000 |
| Frame 0 and 3 VRESET value | 15:0 | **VRESET_HIGH** - *Icarus1 only*. Determines programmed VRESET value when Frame 0 and 3 shutters are open during the image recovery period for a 2-frame Icarus. Used in conjunction with **VRESET_WAIT_TIME** and **ICARUS_VER_SEL**. Voltage is scaled to 0xFFFF ≙ 3.3 V. Outside of these two frames, Vreset is set according to the to the value of DACH | | | |
| **0x04B** | **FRAME_ORDER_SEL** | | V1, V4 | Read/Write | 0x0000_0000 |
| Reorders the frame readout | 2:0 | 000 | Readout frame order (forwards): [0, 1, 2] | | |
| | | 001 | Readout frame order: [2, 0, 1] | | |
| | | 010 | Readout frame order: [1, 2, 0] | | |
| | | 011 | Readout frame order: [0, 2, 1] | | |
| | | 100 | Readout frame order: [1, 0, 2] | | |
| | | 101 | Readout frame order (reverse): [2, 1, 0] | | |
| **0x04C** | **MISC_SENSOR_CTL** | | V1, V4 | Read/Write | 0x0000_01BE |
| Miscellaneous sensor control for Icarus | 0 | **ACCUMULATION_CTL** – If '1', invoke accumulation mode; relevant sensor pins are controlled by bits 1-4 of this register. If '0', those sensor pins are managed by manual shutter control.[5] | | | |
| | 1 | **HST_TST_ANRST_EN** – Must have bit 0 enabled of this register. If '1', assert the High-Speed Timing manual pixel reset enable pin on the A hemisphere of the sensor. If '0', disable this pin. [5] | | | |
| | 2 | **HST_TST_BNRST_EN** – Must have bit 0 enabled of this register. If '1', assert the High-Speed Timing manual pixel reset enable pin on the B hemisphere of the sensor. If '0', disable this pin. [5] | | | |

| | | |
|---|---|---|
| | 3 | **HST_TST_ANRST_IN** – Must have bit 0 enabled of this register. If '1', assert the High-Speed Timing manual w1 shutter pin on the A hemisphere of the sensor. If '0', disable this pin. [5] |
| | 4 | **HST_TST_BNRST_IN** – Must have bit 0 enabled of this register. If '1', assert the High-Speed Timing manual w1 shutter pin on the B hemisphere of the sensor. If '0', disable this pin. [5] |
| | 5 | **HST_PXL_RST_EN** – If '1', assert the pixel reset transistor enable pin on the sensor. If '0', disable this pin. [5] |
| | 6 | **HST_CONT_MODE** – If '1', enable continuous mode. Shutter timing generation repeats if the oscillator is enabled.[5] |
| | 7 | **COL_DCD_EN** – If '1', enable column decode.[5] |
| | 8 | **COL_READOUT_EN** – If '1', enable the pad drivers for the analog image channels.[5] |

| **0x050** | **MANUAL_SHUTTERS_MODE** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|

| | | | |
|---|---|---|---|
| Manual Shutters Mode select. When bit 0 is set, the FPGA will generate manual shutters signals for the ROIC when the fine trigger is detected. Otherwise, the on-chip High Speed Timing will be used. | 0 | **MANSHUT_MODE** | |
| | | 0 | Normal 'High Speed' Mode |
| | | 1 | Manual Shutters Mode |

| **0x051** | **W0_INTEGRATION** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|
| Manual Shutters image integration time register for frame 0 of ICARUS A-side. | 29:0 | Amount of integration time in 25 ns steps. | | |

| **0x052** | **W0_INTERFRAME** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|
| Manual Shutters image interframe time- time between acquisitions for frames 0 and 1 of ICARUS A-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |

| **0x053** | **W1_INTEGRATION** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|
| Manual Shutters image integration time register for frame 1 of ICARUS A-side. | 29:0 | Amount of integration time in 25 ns steps. | | |

| **0x054** | **W1_INTERFRAME** | V1, V4 | Read/Write | 0x0000_0000 |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Manual Shutters image interframe time- time between acquisitions for frames 1 and 2 of ICARUS A-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |
| **0x055** | **W2_INTEGRATION** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image integration time register for frame 2 of ICARUS A-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x056** | **W2_INTERFRAME** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image interframe time- time between acquisitions for frames 2 and 3 of ICARUS A-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |
| **0x057** | **W3_INTEGRATION** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image integration time register for frame 3 of ICARUS A-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x058** | **W0_INTEGRATION_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image integration time register for frame 0 of ICARUS B-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x059** | **W0_INTERFRAME_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image interframe time- time between acquisitions for frames 0 and 1 of ICARUS B-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |
| **0x05A** | **W1_INTEGRATION_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image integration time register for frame 1 of ICARUS B-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x05B** | **W1_INTERFRAME_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image interframe time- time between acquisitions for frames 1 and 2 of ICARUS B-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |
| **0x05C** | **W2_INTEGRATION_B** | V1, V4 | Read/Write | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| Manual Shutters image integration time register for frame 2 of ICARUS B-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x05D** | **W2_INTERFRAME_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image interframe time- time between acquisitions for frames 2 and 3 of ICARUS B-side. | 29:0 | Amount of interframe time in 25 ns steps. | | |
| **0x05E** | **W3_INTEGRATION_B** | V1, V4 | Read/Write | 0x0000_0000 |
| Manual Shutters image integration time register for frame 3 of ICARUS B-side. | 29:0 | Amount of integration time in 25 ns steps. | | |
| **0x05F** | **TIME_ROW_DCD** | V4 | Read/Write | 0x0002_7100 |
| Row Decode Counter | 27:0 | Set counter for the assertion of ROW_DCD sensor input in 25 ns steps | | |
| **0x090** | **ADC_CTL** | V1, V4 | Write; Self-clearing | --- |
| Control of TI8548 (LLNLv1) ADCs | 0 | Configure ADC 1 | | |
| | 1 | Configure ADC 2 | | |
| | 2 | Configure ADC 3 | | |
| | 3 | Configure ADC 4 | | |
| **0x091** | **ADC1_CONFIG_DATA** | V1, V4 | Read/Write | 0x83A8_81FF |
| Configuration data to be manage ADC 1 | 9:0 | Internal reference DAC setting (1 LSB = internal Vref / 1024) | | |
| | 13 | Internal reference voltage (‘0’ = 2.5 V, ‘1’ = 3 V) | | |
| | 15 | Internal reference enable | | |
| | 24:19 | Voltage multiplier controls | | |
| **0x092** | **ADC2_CONFIG_DATA** | V1, V4 | Read/Write | 0x83A8_81FF |
| Configuration data to be manage ADC 2 | 9:0 | Internal reference DAC setting (1 LSB = internal Vref / 102a4) | | |
| | 13 | Internal reference voltage (‘0’ = 2.5 V, ‘1’ = 3 V) | | |
| | 15 | Internal reference enable | | |
| | 24:19 | Voltage multiplier controls | | |
| **0x093** | **ADC3_CONFIG_DATA** | V1, V4 | Read/Write | 0x83A8_81FF |
| Configuration data to be manage ADC 3 | 9:0 | Internal reference DAC setting (1 LSB = internal Vref / 1024) | | |

| | | | | |
|---|---|---|---|---|
| | 13 | Internal reference voltage (‘0’ = 2.5 V, ‘1’ = 3 V) | | |
| | 15 | Internal reference enable | | |
| | 24:19 | Voltage multiplier controls | | |
| **0x094** | **ADC4_CONFIG_DATA** | V1, V4 | Read/Write | 0x83A8_81FF |
| Configuration data to be manage ADC 4 | 9:0 | Internal reference DAC setting (1 LSB = internal Vref / 1024) | | |
| | 13 | Internal reference voltage (‘0’ = 2.5 V, ‘1’ = 3 V) | | |
| | 15 | Internal reference enable | | |
| | 24:19 | Voltage multiplier controls | | |
| **0x095** | **ADC5_DATA_1** | V4 | Read-only | --- |
| Monitors 1 and 2 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH1 / MON_PRES_MINUS** | | |
| | 23:12 | **MON_CH2 / MON_PRES_PLUS** | | |
| **0x096** | **ADC5_DATA_2** | V4 | Read-only | --- |
| Monitors 3 and 4 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH3 / MON_TEMP** | | |
| | 23:12 | **MON_CH4 / MON_COL_TOP_IBIAS_IN** | | |
| **0x097** | **ADC5_DATA_3** | V4 | Read-only | --- |
| Monitors 5 and 6 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH5 / MON_HST_OSC_R_BIAS** | | |
| | 23:12 | **MON_CH6 / MON_VAB / MON_CHG** (DACG) | | |
| **0x098** | **ADC5_DATA_4** | V4 | Read-only | --- |
| Monitors 7 and 8 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH7 / MON_HST_RO_NC_IBIAS, MON_HST_RO_IBIAS / MON_CHE** (DACE) – see register 0x029 | | |
| | 11:0 | **MON_CH7 / MON_HST_OSC_CTL / MON_CHE** (DACE) – see register 0x029 | | |
| | 23:12 | **MON_CH8 / MON_VRST / MON_CHH** (DACH) – see register 0x02A | | |
| **0x099** | **ADC6_DATA_1** | V4 | Read-only | --- |
| Monitors 9 and 10 data.<br>Full scale is 0-5 V. | 11:0 | **MON_CH9 / MON_COL_BOT_IBIAS_IN** | | |
| | 23:12 | **MON_CH10 / MON_HST_A_PDELAY / MON_CHA** (DACA) – see register 0x027 | | |

| | | | | | |
|---|---|---|---|---|---|
| See Section 5.1 for ADC channel description. | 23:12 | **MON_CH10 / MON_TSENSE_OUT** | | | |
| **0x09A** | **ADC6_DATA_2** | | V4 | Read-only | --- |
| Monitors 11 and 12 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH11 / MON_HST_B_NDELAY / MON_CHD** (DACD) – see register 0x028 | | | |
| | 11:0 | **MON_CH11 / MON_BGREF** | | | |
| | 23:12 | **MON_CH12** (DOSIMETER) | | | |
| **0x09B** | **ADC6_DATA_3** | | V4 | Read-only | --- |
| Monitors 13 and 14 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH13 / MON_HST_OSC_VREF_IN** | | | |
| | 11:0 | **MON_CH13 / MON_HST_RO_NC_IBIAS** | | | |
| | 23:12 | **MON_CH14 / MON_HST_B_PDELAY / MON_CHC** (DACC) – see register 0x028 | | | |
| | 23:12 | **MON_CH14 / MON_HST_OSC_VREF_IN / MON_CHC** (DACC) – see register 0x028 | | | |
| **0x09C** | **ADC6_DATA_4** | | V4 | Read-only | --- |
| Monitors 15 and 16 data.<br>Full scale is 0-5 V.<br>See Section 5.1 for ADC channel description. | 11:0 | **MON_CH15 / MON_HST_OSC_CTL / MON_CHF** (DACF) – see register 0x029 | | | |
| | 11:0 | **MON_CH15 / MON_COL_TST_IN / MON_CHF** (DACF) – see register 0x029 | | | |
| | 23:12 | **MON_CH16 / MON_HST_A_NDELAY / MON_CHB** (DACB) – see register 0x027 | | | |
| | 23:12 | **MON_CH16 / MON_HST_OSC_PBIAS_PAD** | | | |
| **0x09D** | **ADC_PPER** | | V4 | Read/Write | 0x000_0064 |
| Polling period for ADC 5 and 6 in milliseconds | 27:0 | ADC monitor (5 and 6) polling period. Default is 100 ms → 0x64 | | | |
| **0x09E** | **ADC_RESET** | | V1, V4 | Read/Write | 0x0000_001F |
| Controls reset pins to ADC 1 through 4 | 0 | ADC1 standby / reset | | | |
| | 1 | ADC2 standby / reset | | | |
| | 2 | ADC3 standby / reset | | | |
| | 3 | ADC4 standby / reset | | | |
| **0x120** | **HST_TRIGGER_DELAY_DATA_LO** | | V1, V4 | Read/Write | 0x0000_0000 |
| High-speed timing trigger delay, LSBs | 31:0 | Delay timing pattern bits [31:0] | | | |
| **0x121** | **HST_TRIGGER_DELAY_DATA_HI** | | V1, V4 | Read/Write | 0x0000_0000 |
| High-speed timing trigger delay, MSBs | 7:0 | Delay timing pattern bits [39:32] | | | |
| **0x122** | **HST_PHI_DELAY_DATA_LO** | | V1, V4 | Read/Write | 0x0000_0000 |

| | | | | | |
|---|---|---|---|---|---|
| High-speed timing trigger delay, MSBs | 7:0 | Delay timing pattern bits [39:32] | | | |
| **0x123** | **HST_PHI_DELAY_DATA_HI** | | V1, V4 | Read/Write | 0x0000_0000 |
| Phi clock programming, LSBs | 31:0 | Delay timing pattern bits for programming Phi clock [31:0] | | | |
| **0x125** | **HST_TRIG_DELAY_READBACK_LO** | | V1, V4 | Read Only | --- |
| High-speed timing trigger delay, LSBs | 31:0 | Delay timing pattern bits [31:0] | | | |
| **0x126** | **HST_TRIG_DELAY_READBACK_HI** | | V1, V4 | Read Only | --- |
| High-speed timing trigger delay, MSBs | 7:0 | Delay timing pattern bits [39:32] | | | |
| **0x127** | **HST_PHI_DELAY_READBACK_LO** | | V1, V4 | Read Only | --- |
| Phi clock programming, LSBs | 31:0 | Delay timing pattern bits for programming Phi clock [31:0] | | | |
| **0x128** | **HST_PHI_DELAY_READBACK_HI** | | V1, V4 | Read Only | --- |
| Phi clock programming, MSBs | 7:0 | Delay timing pattern bits for programming Phi clock [39:32] | | | |
| **0x130** | **HST_COUNT_TRIG** | | V1, V4 | Read/Write | 0x0000_0000 |
| Initiate HST generator | 0 | Initiate HST generator. When asserted, the HST_DIV_CLK is halted[6] | | | |
| **0x131** | **HST_DELAY_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Enable trigger delay cell for HST generator | 0 | Enable trigger delay cell for HST generator[6] | | | |
| **0x132** | **HST_TEST_PHI_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Enable the external Phi input | 0 | Enable the external Phi input to bypass the entire HST generator[6] | | | |
| **0x133** | **RSL_HFW_MODE_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Enable High Full Well mode | 0 | **HFW** – '1' enables HFW mode | | | |
| **0x135** | **RSL_ZDT_MODE_R_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Enable Zero Dead Time mode for right hemisphere | 0 | **ZDT_R** – '1' enables ZDT mode for right hemisphere | | | |
| **0x136** | **RSL_ZDT_MODE_L_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Enable Zero Dead Time mode for left hemisphere | 0 | **ZDT_L** – '1' enables ZDT mode for left hemisphere | | | |
| **0x137** | **BGTRIMA** | | V1, V4 | Read/Write | 0x0000_0000 |
| Drives the BGTRIMA input pins to the Daedalus sensor | 2:0 | BGTRIMA (Bandgap digital trim bit A) pins to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x138** | **BGTRIMB** | | V1, V4 | Read/Write | 0x0000_0000 |
| Drives the BGTRIMB input pins to the Daedalus sensor | 3:0 | BGTRIMB (Bandgap digital trim bit B) pins to the sensor | | | |
| **0x139** | **COLUMN_TEST_EN** | | V1, V4 | Read/Write | 0x0000_0000 |
| Column Test Enable bit | 0 | Drives the ColTstEn pin to the sensor (named col_test_en_o in the FPGA), which controls the global column current source test mode.<br>When '1', the following occurs:<br>1. The sensor's test source followers are enabled on each column current source.<br>2. The sensor's RDcdEn pin (named RowDcdEn_o in the FPGA) is driven low by the FPGA to disable it<br>3. The sensor's nQuenchEn pin (named quench_en_n_o in the FPGA) is driven high by the FPGA to disable it<br>4. The user can then use VRST to drive an analog voltage to the sensor's ColTstIn pin, since VRST is jumpered to ColTstIn.<br>When '0', the sensor and FPGA operate in normal mode | | | |
| **0x140** | **RSL_CONFIG_DATA_R0** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[31:0] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x141** | **RSL_CONFIG_DATA_R1** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[63:32] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x142** | **RSL_CONFIG_DATA_R2** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[95:64] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x143** | **RSL_CONFIG_DATA_R3** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[127:96] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x144** | **RSL_CONFIG_DATA_R4** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[159:128] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x145** | **RSL_CONFIG_DATA_R5** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[191:160] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x146** | **RSL_CONFIG_DATA_R6** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[223:192] for the RSL interlacing shift register input (right side) to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x147** | **RSL_CONFIG_DATA_R7** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[255:224] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x148** | **RSL_CONFIG_DATA_R8** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[287:256] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x149** | **RSL_CONFIG_DATA_R9** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[319:288] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14A** | **RSL_CONFIG_DATA_R10** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[351:320] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14B** | **RSL_CONFIG_DATA_R11** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[383:352] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14C** | **RSL_CONFIG_DATA_R12** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[415:384] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14D** | **RSL_CONFIG_DATA_R13** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[447:416] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14E** | **RSL_CONFIG_DATA_R14** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[479:448] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x14F** | **RSL_CONFIG_DATA_R15** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[511:480] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x150** | **RSL_CONFIG_DATA_R16** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[543:512] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x151** | **RSL_CONFIG_DATA_R17** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[575:544] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x152** | **RSL_CONFIG_DATA_R18** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[607:576] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x153** | **RSL_CONFIG_DATA_R19** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[639:608] for the RSL interlacing shift register input (right side) to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x154** | **RSL_CONFIG_DATA_R20** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[671:640] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x155** | **RSL_CONFIG_DATA_R21** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[703:672] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x156** | **RSL_CONFIG_DATA_R22** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[735:704] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x157** | **RSL_CONFIG_DATA_R23** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[767:736] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x158** | **RSL_CONFIG_DATA_R24** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[799:768] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x159** | **RSL_CONFIG_DATA_R25** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[831:800] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15A** | **RSL_CONFIG_DATA_R26** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[863:832] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15B** | **RSL_CONFIG_DATA_R27** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[895:864] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15C** | **RSL_CONFIG_DATA_R28** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[927:896] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15D** | **RSL_CONFIG_DATA_R29** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[959:928] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15E** | **RSL_CONFIG_DATA_R30** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[991:960] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x15F** | **RSL_CONFIG_DATA_R31** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInR input to the sensor | 31:0 | Data[1023:992] for the RSL interlacing shift register input (right side) to the sensor | | | |
| **0x160** | **RSL_CONFIG_DATA_L0** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[31:0] for the RSL interlacing shift register input (left side) to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x161** | **RSL_CONFIG_DATA_L1** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[63:32] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x162** | **RSL_CONFIG_DATA_L2** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[95:64] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x163** | **RSL_CONFIG_DATA_L3** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[127:96] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x164** | **RSL_CONFIG_DATA_L4** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[159:128] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x165** | **RSL_CONFIG_DATA_L5** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[191:160] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x166** | **RSL_CONFIG_DATA_L6** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[223:192] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x167** | **RSL_CONFIG_DATA_L7** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[255:224] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x168** | **RSL_CONFIG_DATA_L8** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[287:256] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x169** | **RSL_CONFIG_DATA_L9** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[319:288] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x16A** | **RSL_CONFIG_DATA_L10** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[351:320] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x16B** | **RSL_CONFIG_DATA_L11** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[383:352] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x16C** | **RSL_CONFIG_DATA_L12** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[415:384] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x16D** | **RSL_CONFIG_DATA_L13** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[447:416] for the RSL interlacing shift register input (left side) to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x16E** | **RSL_CONFIG_DATA_L14** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[479:448] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x16F** | **RSL_CONFIG_DATA_L15** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[511:480] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x170** | **RSL_CONFIG_DATA_L16** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[543:512] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x171** | **RSL_CONFIG_DATA_L17** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[575:544] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x172** | **RSL_CONFIG_DATA_L18** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[607:576] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x173** | **RSL_CONFIG_DATA_L19** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[639:608] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x174** | **RSL_CONFIG_DATA_L20** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[671:640] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x175** | **RSL_CONFIG_DATA_L21** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[703:672] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x176** | **RSL_CONFIG_DATA_L22** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[735:704] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x177** | **RSL_CONFIG_DATA_L23** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[767:736] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x178** | **RSL_CONFIG_DATA_L24** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[799:768] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x179** | **RSL_CONFIG_DATA_L25** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[831:800] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x17A** | **RSL_CONFIG_DATA_L26** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[863:832] for the RSL interlacing shift register input (left side) to the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x17B** | **RSL_CONFIG_DATA_L27** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[895:864] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x17C** | **RSL_CONFIG_DATA_L28** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[927:896] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x17D** | **RSL_CONFIG_DATA_L29** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[959:928] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x17E** | **RSL_CONFIG_DATA_L30** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[991:960] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x17F** | **RSL_CONFIG_DATA_L31** | | V1, V4 | Read/Write | 0x00000000 |
| RSLScanInL input to the sensor | 31:0 | Data[1023:992] for the RSL interlacing shift register input (left side) to the sensor | | | |
| **0x180** | **RSL_READ_BACK_R0** | | V1, V4 | Read Only | 0x00000000 |
| RSLSanOutR output from the sensor | 31:0 | Data[31:0] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x181** | **RSL_READ_BACK_R1** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[63:32] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x182** | **RSL_READ_BACK_R2** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[95:64] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x183** | **RSL_READ_BACK_R3** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[127:96] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x184** | **RSL_READ_BACK_R4** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[159:128] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x185** | **RSL_READ_BACK_R5** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[191:160] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x186** | **RSL_READ_BACK_R6** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[223:192] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x187** | **RSL_READ_BACK_R7** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[255:224] for the RSL interlacing shift register output (right side) from the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x188** | **RSL_READ_BACK_R8** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[287:256] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x189** | **RSL_READ_BACK_R9** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[319:288] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18A** | **RSL_READ_BACK_R10** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[351:320] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18B** | **RSL_READ_BACK_R11** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[383:352] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18C** | **RSL_READ_BACK_R12** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[415:384] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18D** | **RSL_READ_BACK_R13** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[447:416] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18E** | **RSL_READ_BACK_R14** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[479:448] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x18F** | **RSL_READ_BACK_R15** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[511:480] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x190** | **RSL_READ_BACK_R16** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[543:512] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x191** | **RSL_READ_BACK_R17** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[575:544] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x192** | **RSL_READ_BACK_R18** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[607:576] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x193** | **RSL_READ_BACK_R19** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[639:608] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x194** | **RSL_READ_BACK_R20** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[671:640] for the RSL interlacing shift register output (right side) from the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x195** | **RSL_READ_BACK_R21** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[703:672] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x196** | **RSL_READ_BACK_R22** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[735:704] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x197** | **RSL_READ_BACK_R23** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[767:736] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x198** | **RSL_READ_BACK_R24** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[799:768] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x199** | **RSL_READ_BACK_R25** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[831:800] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19A** | **RSL_READ_BACK_R26** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[863:832] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19B** | **RSL_READ_BACK_R27** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[895:864] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19C** | **RSL_READ_BACK_R28** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[927:896] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19D** | **RSL_READ_BACK_R29** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[959:928] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19E** | **RSL_READ_BACK_R30** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[991:960] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x19F** | **RSL_READ_BACK_R31** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutR output from the sensor | 31:0 | Data[1023:992] for the RSL interlacing shift register output (right side) from the sensor | | | |
| **0x1A0** | **RSL_READ_BACK_L0** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[31:0] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A1** | **RSL_READ_BACK_L1** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[63:32] for the RSL interlacing shift register output (left side) from the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x1A2** | **RSL_READ_BACK_L2** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[95:64] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A3** | **RSL_READ_BACK_L3** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[127:96] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A4** | **RSL_READ_BACK_L4** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[159:128] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A5** | **RSL_READ_BACK_L5** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[191:160] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A6** | **RSL_READ_BACK_L6** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[223:192] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A7** | **RSL_READ_BACK_L7** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[255:224] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A8** | **RSL_READ_BACK_L8** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[287:256] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1A9** | **RSL_READ_BACK_L9** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[319:288] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1AA** | **RSL_READ_BACK_L10** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[351:320] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1AB** | **RSL_READ_BACK_L11** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[383:352] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1AC** | **RSL_READ_BACK_L12** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[415:384] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1AD** | **RSL_READ_BACK_L13** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[447:416] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1AE** | **RSL_READ_BACK_L14** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[479:448] for the RSL interlacing shift register output (left side) from the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x1AF** | **RSL_READ_BACK_L15** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[511:480] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B0** | **RSL_READ_BACK_L16** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[543:512] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B1** | **RSL_READ_BACK_L17** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[575:544] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B2** | **RSL_READ_BACK_L18** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[607:576] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B3** | **RSL_READ_BACK_L19** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[639:608] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B4** | **RSL_READ_BACK_L20** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[671:640] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B5** | **RSL_READ_BACK_L21** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[703:672] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B6** | **RSL_READ_BACK_L22** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[735:704] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B7** | **RSL_READ_BACK_L23** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[767:736] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B8** | **RSL_READ_BACK_L24** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[799:768] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1B9** | **RSL_READ_BACK_L25** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[831:800] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1BA** | **RSL_READ_BACK_L26** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[863:832] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1BB** | **RSL_READ_BACK_L27** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[895:864] for the RSL interlacing shift register output (left side) from the sensor | | | |

| | | | | | |
|---|---|---|---|---|---|
| **0x1BC** | **RSL_READ_BACK_L28** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[927:896] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1BD** | **RSL_READ_BACK_L29** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[959:928] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1BE** | **RSL_READ_BACK_L30** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[991:960] for the RSL interlacing shift register output (left side) from the sensor | | | |
| **0x1BF** | **RSL_READ_BACK_L31** | | V1, V4 | Read Only | 0x00000000 |
| RSLScanOutL output from the sensor | 31:0 | Data[1023:992] for the RSL interlacing shift register output (left side) from the sensor | | | |

# 13 Error Recovery and Diagnostics

Numerous design and development techniques have been utilized in the NSGCC FPGA to ensure that it operates as reliably as possible; however, due to the harsh operational environment that the system is targeted for, logic failures are unavoidable and will occur. In this case, the FPGA's error recovery and diagnostic features become critical, enabling the FPGA to recover from an error and allowing the user to diagnose the root cause so that corrections can be made. These features include

- Status registers, counters, and timers in various parts of the sensor and SRAM readout pipelines to detect when logic modules and state machines have "hung" and to assist in determining the root causes
- Automatic resets in the RS422 logic section, which ensure that communications can be re-established in case of an error in the RS422 modules
- A software reset to enable the user to return the board to its default configuration when desired.

The following sub-sections describe the implemented error recovery and diagnostic features.

## 13.1 Software Reset

A software reset has been implemented which resets all logic within the FPGA. To activate the software reset, write a '1' to the **RESET** subregister. This bit will self-clear after being written. Note that all internal FPGA logic will be reset to their default values, including the control registers.

## 13.2 Automatic Resets on RS422 Transmit and Receive

To ensure that serial communications are maintained in the event of Single Event Upsets in the RS422 logic, automatic resets in both the RS422 transmit and receive sections have been implemented. In the event of a detected "hang" in the RS422 logic, all modules related to reception of command packets and transmission of response packets are reset automatically. The modules which are reset are the RS422, Steering, and Packet Encode/Decode modules. Note that a "hang" condition is determined when an operation takes much longer than under normal operation.

The RS422 transmit module (which generates response packets to the host) is determined to be hung when a byte transfer takes 1 ms, which is much longer than is expected under normal conditions. The RS422 receive module (where the FPGA receives a command packet) is determined to be hung when a byte transfer takes longer than 100 ms (to account for host system performance).

When a "hang" in either the transmit or receive RS422 sections is detected, the appropriate bits in the **STAT_REG2_SRC** and **STAT_REG2** registers are asserted.

## 13.3 ADC to SRAM module Timeout

A counter exists in the ADC to SRAM module which can be useful in determining the root cause of errors which occur during image sensor read off. This will count the number of pixels (in bytes) that have been read out from the sensor and written into the SRAMs. During normal operation, this counter will count to the final pixel/byte and reset itself to zero. In the event of an FPGA hang, the value held by this counter could be helpful in diagnosing a fault. If an error or FPGA hang occurs, and RS422 communications have

been maintained, a non-zero value in this counter indicates that the error occurred during image sensor readout, while the value indicates the last pixel that was read out successfully. If the counter contains a value of zero, then the root cause of the error likely resides elsewhere.

The host can access the read-only status register **ADC_BYTECOUNTER**. Note that this register cannot be cleared by software; to clear the register, the counter itself must be cleared by returning the ADC-to-SRAM module to its default state (which may require resetting the FPGA).

## 13.4 FPA Interface Module Timeout

Additional indicators of an error occurring during image sensor readoff reside in the FPA Interface Module, which controls image sensor timing during image readoff.

To control sensor readoff, the FPA Interface module sends control signals to the ADC to SRAM module. Two of the more useful ones for error detection are *adcReadWriteStart* (which initiates an 8-channel ADC read) and *readOffDone* (which indicates that all desired frames/pixels have been read out from the sensor). To determine if the state machine in this module has hung, the time between these signals is monitored. Since the time between *adcReadWriteStart* pulses is normally 2.875 µs, and the time between the last *adcReadWriteStart* pulse and *readOffDone* is 925 ns, a counter has been implemented that will assert an error bit if the time for either of these events exceeds 2 × 2.875 µs, or 5.75 µs. This error bit is called *fpa_if_to*, and it resides in the **STAT_REG2** and **STAT_REG2_SRC** registers.

## 13.5 SRAM Readoff Module Timeout

To assist in detecting the root cause of errors occurring during SRAM readoff, a diagnostic counter has been implemented to monitor timing in the SRAM Readoff Row module which controls SRAM readoff on a per-row basis.

To determine an error in this module, the signal *dataOutEn_n* is monitored. This signal is low during the time when row data is being read from the SRAM; in between rows, it goes high for a small number of clock cycles. If *dataOutEn_n* has been monitored as low for an abnormally long period of time, an error bit is asserted (this is the *sram_ro_to* bit in the **STAT_REG2** and **STAT_REG2_SRC** registers).

During normal operation, *dataOutEn_n* is low for 12.8 µs; the error bit is not asserted unless the signal has been low for 25.6 µs.

## 13.6 Read Burst Processing Module Timeouts

A timeout counter has been implemented to determine the overall length of time that it takes for the entire image sensor readoff and SRAM readout processes to occur. If the length of time has been found to be much longer than normal, then the logic decides that an error has occurred, and two operations occur: (1) control of the transmit data pipeline within the FPGA reverts from Burst Response (i.e., pixel) data back to Response (i.e., status) data to re-enable communications with the host, and (2) the *pixelrd_timeout_err* bit in the **STAT_REG2** and **STAT_REG2_SRC** registers is asserted.

## 13.7 Resets

Due to the complexity of the radiation tolerant version of the FPGA, numerous resets are generated and used for different purposes. Table 9 lists all the resets used within the device, their modules of origin, purpose, and the logic they affect.

| Reset signal | Driver (origin) module | Purpose | Connected modules |
|---|---|---|---|
| *sys_rst_n_i* | N/A - PCB reset | System reset | clocks_and_resets.vhd (used as input to other reset signals, then distributed to appropriate logic) |
| *sw_rst* | Ctl_and_status_ regs.vhd | Software-controlled system reset | clocks_and_resets.vhd (used as input to other reset signals, then distributed to appropriate logic) |
| *cns_reg_rst* | Clocks_and_ resets | Reset signal dedicated to resetting the control and status module. Is not asserted under any condition except by sys_rst_n_i. Preserves register state in case any other reset occurs. | ctl_and_status_regs.vhd |
| *sys_rst / sysrst* | Clocks_and_ resets | Asserted when sys_rst_n_i or sw_rst are asserted | timer_counter.vhd<br>rs422.vhd<br>dummy_adc_rd.vhd |
| *seu_rec_rst* | Seu_recovery .vhd | Asserted if fpa interface times out due to SEU occur | Connected to seu_sys_rst_bl, shot_sys_rst, and shot_sys_rst_bl signals |
| *seu_sys_ rst_bl* | Nsgcc_top.vhd | OR function of *sys_rst* and *seu_rec_rst*. Used specifically to reset logic in fpa_interface.vhd that does NOT need to be held in a particular state during the shot. | fpa_interface.vhd – used to reset all logic (and sub-modules) except for the state machine and related logic. |
| *rt_shot_ hold* | Clocks_and_ resets.vhd | Hold rad-susceptible logic in reset for 20us following fine trigger | fpa_interface.vhd – holds state machine in WAIT_FOR_FINE_ TRIG state for 20us following fine trigger. If an SEU occurs that corrupts the state machine, it will transition back to WAIT_FOR_FINE_TRIG |
| *shot_sys_rst* | Nsgcc_top.vhd | OR function of sys_rst, rt_shot_hold, and seu_rec_rst | sw_trigger_ctl.vhd |
| *shot_sys_rst_bl* | Nsgcc_top.vhd | Boolean version of shot_sys_rst | sram_readoff_top.vhd<br>adc_to_sram_read_control |
| *timeout_rec_rst* | Rs422_top.vhd | Asserted when timeouts occur during an rs422 RX or TX transaction. | Connected to *shot_comms_rst* signal |
| *shot_comms_rst* | Nsgcc_top.vhd | OR function of *timeout_rec_rst* and *shot_sys_rst*. Resets affected modules if an rs422 timeout occurs | steering.vhd<br>packet_enc_dec.vhd |

Table 9: Device Resets

# 14 Icarus implementation

Information regarding the Icarus sensor interface can be found in the UXI ICARUS – Focal Plane Array Interface Document.

## 14.1 ADC Interface and Control

The FPGA controls the six analog-to-digital converters (ADCs) on the board. Four of the ADCs are used to convert the image sensor's analog pixel information into 16-bit digital data, while the fifth and sixth are used to monitor DAC channel voltages and specific sensor pin voltages. The four ADCs used to convert the image sensor's analog pixel information to digital data are Texas Instruments 'ADS8568SPM's, which are eight-channel 16-bit devices. Since the Icarus sensor is a 32-channel device, four of these 8-channel devices (numbered ADC1 through ADC4) are required to read out all sensor channels simultaneously. The sensor quadrant and column number connectivity to ADC device number/channel number is shown in Table 10. The two ADCs used for ADC monitoring are Texas Instruments 'ADC128S102's, which are eight-channel 12-bit devices.

| Sensor Quadrant | Sensor Column | ADC Device | ADC Channel |
|---|---|---|---|
| **Top Left** | 0-31 | 4 | 2 |
| **Top Left** | 32-63 | 4 | 3 |
| **Top Left** | 64-95 | 3 | 6 |
| **Top Left** | 96-127 | 3 | 5 |
| **Top Left** | 128-159 | 4 | 0 |
| **Top Left** | 160-191 | 4 | 1 |
| **Top Left** | 192-223 | 4 | 5 |
| **Top Left** | 224-255 | 3 | 1 |
| **Top Right** | 256-287 | 3 | 4 |
| **Top Right** | 288-319 | 3 | 7 |
| **Top Right** | 320-351 | 4 | 4 |
| **Top Right** | 352-383 | 4 | 6 |
| **Top Right** | 384-415 | 4 | 7 |
| **Top Right** | 416-447 | 3 | 0 |
| **Top Right** | 448-479 | 3 | 2 |
| **Top Right** | 480-511 | 3 | 3 |
| **Bottom Left** | 0-31 | 2 | 6 |
| **Bottom Left** | 32-63 | 2 | 0 |
| **Bottom Left** | 64-95 | 2 | 2 |
| **Bottom Left** | 96-127 | 2 | 1 |
| **Bottom Left** | 128-159 | 2 | 3 |
| **Bottom Left** | 160-191 | 2 | 7 |
| **Bottom Left** | 192-223 | 2 | 4 |
| **Bottom Left** | 224-255 | 1 | 4 |
| **Bottom Right** | 256-287 | 1 | 6 |
| **Bottom Right** | 288-319 | 1 | 5 |
| **Bottom Right** | 320-351 | 1 | 7 |
| **Bottom Right** | 352-383 | 1 | 3 |
| **Bottom Right** | 384-415 | 1 | 2 |
| **Bottom Right** | 416-447 | 1 | 1 |
| **Bottom Right** | 448-479 | 1 | 0 |
| **Bottom Right** | 480-511 | 2 | 5 |

Table 10: Icarus Sensor Channel to ADC Device/Channel Connectivity

## 14.2 ADC and monitor assignments

The Icarus-configured board ADC, DAC, and monitor assignments are shown in Table 11.

| DAC | Monitor Channel | ADC : Pin | Function | Description | Sens. Pin | Nominal Voltage (V) |
|---|---|---|---|---|---|---|
| **A** | **MON_HST_A_PDELAY** | 6:5 | *HST_A_PDELAY* | Delay voltage for A side pixel array. Increase to add delay. | 78 | 0 |
| **B** | **MON_HST_A_NDELAY** | 6:11 | *HST_A_NDELAY* | Delay voltage for A side pixel array. Decrease to add delay. | 70 | 3.3 |
| **C** | **MON_HST_B_PDELAY** | 6:9 | *HST_B_PDELAY* | Delay voltage for B side pixel array. Increase to add delay. | 94 | 0 |
| **D** | **MON_HST_B_NDELAY** | 6:6 | *HST_B_NDELAY* | Delay voltage for B side pixel array. Decrease to add delay. | 86 | 3.3 |
| **E** | **MON_HST_RO_IBIAS / MON_HST_RO_NC_ IBIAS** | 5:10 | *HST_RO_IBIAS/ HST_RO_NC_ IBIAS* | Ring oscillator with capacitors bias control voltage. | 46 | 2.5 |
| **F** | **MON_HST_OSC_CTL** | 6:10 | *HST_OSC_CTL* | Relaxation oscillator bias control voltage. | 48 | 1.45 |
| **G** | **MON_VAB** | 5:9 | *VAB* | Gate voltage of all anti-bloom transistors in the ICARUS pixel array. | 28 | 0.5 |
| **H** | **MON_VRST** | 5:11 | *VRST* | Resets the voltage for all pixels. | 61, 69, 77 | 0.3 |
| --- | **MON_PRES_MINUS** | 5:4 | --- | Mems pressure sensor, negative output pin (see Section 9 ) | --- | --- |
| --- | **MON_PRES_PLUS** | 5:5 | --- | Mems pressure sensor, positive output pin (see Section 9 ) | --- | --- |
| --- | **MON_TEMP** | 5:6 | --- | Temperature transducer voltage (see Section 8 ) | --- | --- |
| --- | **MON_COL_TOP_ IBIAS_IN** | 5:7 | --- | Column bias for top hemisphere of ICARUS sensor. | 8 | High Z |
| --- | **MON_HST_OSC_ R_BIAS** | 5:8 | --- | Current sink set for ring oscillator with capacitors. | 16 | High Z |
| --- | **MON_COL_BOT_ IBIAS_IN** | 6:4 | --- | Column bias for bottom hemisphere of ICARUS sensor. | 153 | High Z |
| --- | **DOSIMETER** | 6:7 | --- | Voltage of dosimeter | --- | --- |
| --- | **MON_HST_OSC_ VREF_IN** | 6:8 | --- | Oscillator voltage reference, biased by a voltage divider to a specific voltage of 2.9 V. | 87 | 2.9 |

Table 11 ICARUS DAC and Monitor Assignments; see the 'UXI_Icarus_FPA' document for more details[5]

## 14.3 Anti-Bloom Circuit Control

Each Icarus pixel contains an anti-bloom transistor that is designed to shunt photocurrents greater than full-well, which protects the pixel circuitry from large photo diode signal fluctuations. The gate voltage of the transistor controls the $V_{DS}$ at which the transistor starts conducting current; the VAB pin on the ICARUS sensor is connected to the gate of the anti-bloom transistors and enables the user to apply a voltage to it. On the Version 4.0 Board, VAB is controlled by the **VAB** subregister (set using **DACG**).[5]

## 14.4 High-Speed Timing (HST) Control

The High-Speed Timing Control function is discussed in detail in the *UXI ICARUS – Focal Plane Array Interface Document,* so this document will not repeat the information contained in that interface document.[5] However, some important things to note regarding this FPGA's implementation of the HST programming sequences are:

1. High Speed Timing must be configured prior to initiating the programming sequence using the **HS_TIMING_DATA_ALO**, **HS_TIMING_DATA_AHI**, **HS_TIMING_DATA_BLO**, and **HS_TIMING_ DATA_BHI** registers. These four registers define the 80 bits of A and B side timing described in Figure 6 through 12 of the *UXI ICARUS – Focal Plane Array Interface Document.*
2. The HST programming sequence is initiated when the FPGA detects the rising edge of the Coarse Trigger. The programming sequence is also initiated when the Power-On Image Reset Mode is implemented (Section 14.9.2 ).

## 14.5 Manual Shutter Control

The user can override the HST function of the Icarus sensor and utilize a Manual Shutter Control function instead. Although the timing granularity is coarse compared with the HST mechanism (25 ns vs 1 ns), the user can set integration times between 25 ns and 26.8 seconds for each of the frames, as shown in Figure 3. Manual Shutter Control is discussed in detail in the *UXI ICARUS – Focal Plane Array Interface Document,* so this document will not repeat the information contained there.

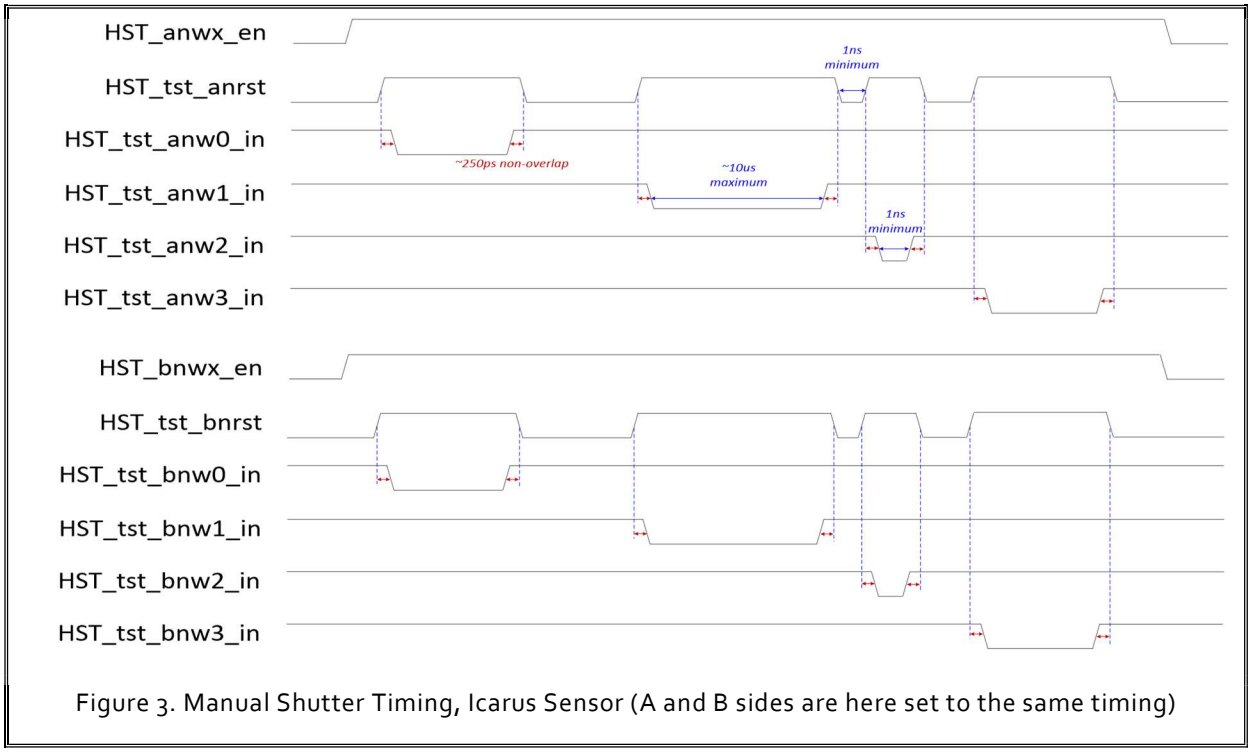


Figure 3. Manual Shutter Timing, Icarus Sensor (A and B sides are here set to the same timing)

Some things to note when using the Manual Shutter Control function of the Icarus sensor:

1. To configure the FPGA and sensor for Manual Shutter control, write a '1' to the **MANSHUT_MODE** subregister.
2. The **W*X*_INTEGRATION** and **W*Y*_INTERFRAME** (X ∈ {0,1,2,3}, Y ∈ {0,1,2}) registers must be programmed to define the timing in 25 ns steps.
3. After the registers are set up properly, the timing sequence in Figure 3 is initiated when the FPGA detects the rising edge of the Fine Trigger.

## 14.6 Power Save Mode

Power Save Mode is enabled by asserting bit 3 of the **CTRL_REG** register. When asserted, sensor power saving is achieved by controlling the Icarus sensor's *hst_osc_bias_en* signal. During power save mode, *hst_osc_bias_en* is asserted when the coarse trigger is detected; it is de-asserted once the sensor readout is complete. The effects of the power save mode on LLNL V4 board operation has not been measured.

In addition, internally to the FPGA, power is saved by asserting *hst_data_nRst* for one clock cycle after readoff of image data from FPA to SRAM is complete. This resets the sensor, reducing the current sourced to the board.

## 14.7 Photodiode Bias Control and Status

The Photodiode Bias (the onboard 49 V switching regulator) can be turned off on boards with firmware revisions AC and later. The FPGA can shut down the regulator at the respective shutdown pin by setting the, **PDBIAS_LOW** subregister high. The feature allows for applications such as sensor hot-swapping and taking data when the photodiode is not biased for comparison.

The status of the photodiode bias can be read from the **PDBIAS_UNREADY** subregister. A 'False' value indicates the bias is functional (exceeding 10 V). 'True' indicates a faulty or non-functional bias below 10 V. Boards with firmware version AC or later can differentiate between a working or non-working bias; boards with earlier versions of the firmware will always indicate a functional bias .

## 14.8 Delaying Image Readout

The image readout from an Icarus sensor can be delayed by the firmware. A counter is implemented in the FPGA before readoff from the sensor is initiated. The delay time is set by the **TIME_ROW_DCD** register (1 count ≙ 25 ns)[5], up to a maximum of 6.7 seconds. This counter is incremented by the rising edge of the FPGA system clock. The FPGA asserts the row decode ensable sensor pin (row_dcd_en) for the duration of the counter.

## 14.9 Radiation-Tolerant Modes

The radiation-tolerant features are currently Icarus-specific in terms of the firmware. They attempt to provide protection from Single-Event Upsets (SEUs) that might strike sensitive nodes on the circuitry or upset the firmware in certain areas. Current Icarus-specific firmware uses both Triple Modular Redundancy (TMR) and Hamming Encoding in crucial areas of the FPGA to provide extra protection. Radiation-protection features are intended for use with the RS422 communication protocol.

### 14.9.1 Suspend Mode

This mode holds various modules in reset for 100 µs after the assertion (or reassertion, for dual-edge triggering) of the fine trigger. This mode is a precaution to hold off sensitive FPGA controls until the passing of the radiation event.

### 14.9.2 Power-On Image Reset Mode

A power-on image reset mode is hardcoded allowing the user to extract reasonable images from the SRAMs on power-up or reset. This mode invokes a software trigger described in Section 6.2 . This means a coarse and fine trigger are generated internal to the FPGA to set the high-speed timing of the sensor followed by image readoff to the SRAMs. The mode is a precaution if either image readoff fails due to a communications or FPGA upset.

### 14.9.3 Blink Mode

If intense radiation exposure is expected, critical radiation-susceptible hardware can be turned off by putting the device into blink mode to prevent hardware upset. The external AUX input can be used to put the board into blink mode. The mode will initiate a power-down of the following hardware components on the camera board without turning off power to the sensor:

- Image readoff ADCs
- DAC
- External reference for image readoff ADCs
- FPGA
- Crystal oscillator for FPGA
- SRAMs
- Pressure sensor
- Temperature sensor
- Push-button reset
- GigE
- RS422
- ADC monitors

Figure 4 describes the recommended timing of blink mode, which should only be used with hardware triggers. This mode is fully functional for board revision AC.

1. The external Coarse Trigger must be asserted for at least 60 ns to allow the FPGA to detect the trigger.
2. The external AUX Trigger should be asserted no sooner than 11.5 µs after the rising edge of the Coarse Trigger to allow time for the FPGA to program the high-speed timing of the sensor.
3. The AUX must be asserted for at least 200 µs before the Fine Trigger is issued in order to fully power down the FPGA.
4. When the AUX input is deasserted, the hardware components are turned back on and various system parameters are reset.
5. The FPGA waits 40 ms after powerup before automatically initiating image readoff from the sensor to the SRAM as a precaution against artifacts or noise caused using by the AUX trigger. This delay is configurable using the **TIME_ROW_DCD** register.
6. Sensor readoff to SRAM takes about 178.59 ms for a 4-frame capture and 89.29 ms for a 2-frame capture.
7. Once readoff is complete, the FPGA sets the **SRAM_READY** bit of the **STAT_REG** register to "1".
8. The software monitors this **SRAM_READY** bit to know when to download image data from SRAM to the host.

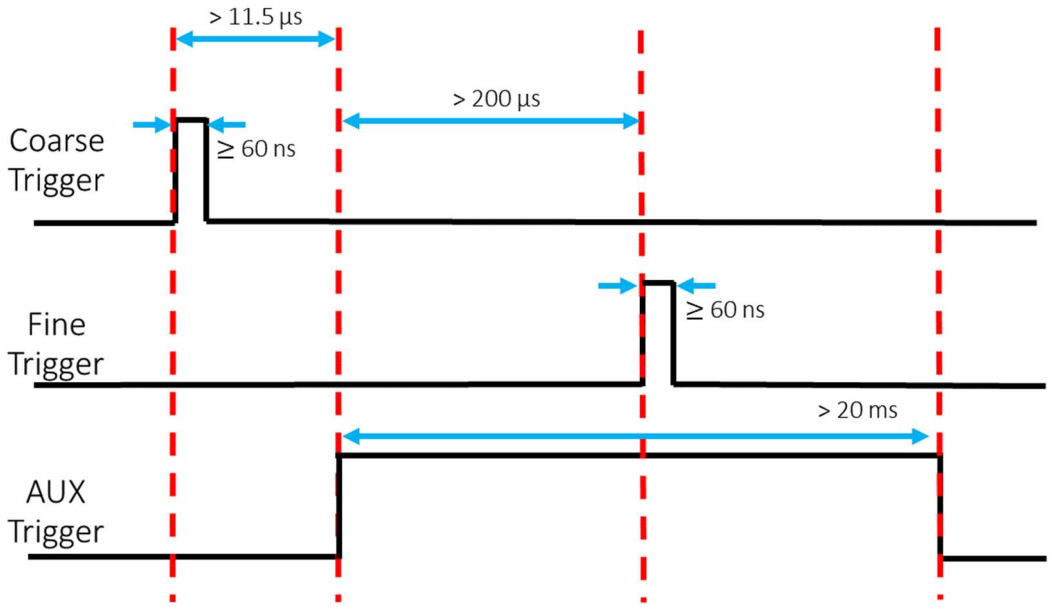


Figure 4: Recommended Timing for Blink Mode

# 15 Daedalus Implementation

Currently, there are various Daedalus features that are not functional. Current Daedalus features that have been tested and functionally perform as per the ICD are[6]:

- High Full Well (HFW)
- Zero-Dead Timing (ZDT)
- Frame order extraction of six different frame orders
- Slow image readoff by a factor of two and three options

Features that currently do not either operate correctly or have not been tested are:

- On-board sensor temperature sensor (TSenseOut)
- Phi clock programming
- Row Shutter Logic (RSL) programming
- External Phi Clock bypass option
- Interlacing
- Trigger delay

## 15.1 ADC Interface and Control

The reader should look to Section 14.1 for details of the image readoff ADC. Table 12 shows the Daedalus sensor channel to ADC connectivity. Note, two of the image readoff ADCs are used for the sensor.

| Sensor Quadrant | Sensor Column | ADC Device | ADC Channel |
|---|---|---|---|
| **Left** | 0-31 | 4 | 2 |
| **Left** | 32-63 | 4 | 3 |
| **Left** | 64-95 | 3 | 6 |
| **Left** | 96-127 | 3 | 5 |
| **Left** | 128-159 | 4 | 0 |
| **Left** | 160-191 | 4 | 1 |
| **Left** | 192-223 | 4 | 5 |
| **Left** | 224-255 | 3 | 1 |
| **Right** | 256-287 | 3 | 4 |
| **Right** | 288-319 | 3 | 7 |
| **Right** | 320-351 | 4 | 4 |
| **Right** | 352-383 | 4 | 6 |
| **Right** | 384-415 | 4 | 7 |
| **Right** | 416-447 | 3 | 0 |
| **Right** | 448-479 | 3 | 2 |
| **Right** | 480-511 | 3 | 3 |

Table 12: Daedalus Sensor Channel to ADC Device/Channel Connectivity

## 15.2 DAC and monitor assignments

The Daedalus-configured board DAC, and monitor assignments are shown in Table 13.

| DAC | Monitor Channel | ADC: Pin | Function | Description | Sensor Pin | Nominal Voltage (V) |
|---|---|---|---|---|---|---|
| **C** | **MON_HST_OSC_ VREF_IN** | *ADC6: 9* | *HST_OSC_ VREF_IN* | Oscillator voltage reference. | 94 | 1.0 |
| **E** | **MON_HST_ OSC_CTL** | *ADC5: 10* | *HST_OSC_CTL* | Relaxation oscillator bias control voltage. | 46 | 1.0 |
| **F** | **MON_COL_ TST_IN** | *ADC6: 10* | *colTstIn* | Global column current source test pin. | 48 | 0 |
| **G** | **MON_VAB** | *ADC5: 9* | *VAB* | Gate voltage of all anti-bloom transistors in the pixel array. | 28 | 0.5 |
| --- | **MON_PRES_ MINUS** | ADC5: 4 | --- | Mems pressure sensor, negative output pin (see Section 9 ) | --- | --- |
| --- | **MON_PRES_ PLUS** | ADC5: 5 | --- | Mems pressure sensor, positive output pin (see Section 9 ) | --- | --- |
| --- | **MON_TEMP** | ADC5: 6 | --- | Temperature transducer voltage (see Section 8 ) | --- | --- |

| | | | | | | |
|---|---|---|---|---|---|---|
| --- | **MON_TSENSE_ OUT** | *ADC6: 5* | *TSenseOut* | Monitor the temperature of the Daedalus sensor. | 78 | --- |
| --- | **MON_BGREF** | *ADC6: 6* | --- | Bandgap voltage reference. | 86 | 1.0 |
| --- | **DOSIMETER** | ADC6: 7 | --- | Voltage of dosimeter | --- | --- |
| --- | **MON_HST_RO_ NC_IBIAS** | *ADC6: 8* | HST_RO_NC_ IBIAS | Ring oscillator biased by a voltage divider to a specific voltage of ~1.0 V | 87 | 1.0 |
| --- | **MON_HST_OSC_ PBIAS_PAD** | *ADC6: 11* | HST_OSC_ PBIAS_PAD | Current mirror node for oscillator bias | 70 | 2.0 |

Table 13: Daedalus DAC and Monitor Assignments; see the 'UXI_Daedalus_HDD" document for more details[7]

## 15.3 Anti-Bloom Circuit Control

See Section 14.3 (Daedalus and Icarus behaviors are identical).

## 15.4 High-Speed Timing (HST) Control

See Section 14.4 (Daedalus and Icarus behaviors are identical).

## 15.5 Trigger Delay

*Note: Currently, the behavior of this feature is uncertain with respect to the FPGA.* Trigger Delay is implemented as described in the *UXI Daedalus – HDD* document. The registers **HST_TRIGGER_DELAY_DATA_LO** and **HST_TRIGGER_DELAY_DATA_HI** are used to program the 40-bit shift register with a fixed delay of ~150 ps for each bit. A maximum of 38 delay blocks can be implemented allowing a trigger delay up to 5.7 ns. The register **HST_DELAY_EN** then can be used to delay the signal HST_OSC_EN when the Fine Trigger is asserted based on the programmed delay. The **HST_TRIG_DELAY_READBACK_LO** and **HST_TRIG_DELAY_READBACK_HI** registers can be used to read back the requested delay written to the sensor.[6]

## 15.6 Phi Clock Programming

*Note: Currently, the behavior of this feature is uncertain with respect to the FPGA.* The internal Phi clock can be programmed to set the shutter timing registers for both left and right hemispheres of the sensor. The programming is described in the *UXI Daedalus – HDD* document. The registers **HST_PHI_DELAY_ DATA_LO** and **HST_PHI_DELAY_DATA_HI** are used to program the left and right hemisphere 40-bit shift registers. The **HST_PHI_DELAY_READBACK_LO** and **HST_PHI_DELAY_READBACK_HI** registers can be used to read back the requested delay written to the sensor.[6]

## 15.7 External Phi Clock Bypass

*Note: Currently, the behavior of this feature is uncertain with respect to the FPGA.* To bypass the sensor's internal phi clock, enable the HstTestPhiEn sensor pin and set the **OSC_SELECT** subregister to "11". This allows the sensor to substitute the on-chip generation of the bit stream with an off-chip shutter and inter-frame time. The off-chip phi clock can be observed on the sensor pin, HstPhiOutPad.[6]

## 15.8 Row Shutter Logic (RSL) Programming

*Note: Currently, the behavior of this feature is uncertain with respect to the FPGA.* The programming is described in more depth in the *UXI Daedalus – HDD* document. RSL is programmed using 1024 bits each for both left and right hemispheres. The bits are used to deserialize the shutter and inter-frame timing information from the phi clock signal. The extraction of each frame's shutter timing and inter-frame time can be set to apply simultaneously for all rows of the pixel array (the default state) or to apply sequentially for interlaced subsets of rows (e.g., even rows first, then odd rows). Interlacing rows of pixels multiplies the number of frames of image data that the Daedalus FPA can capture, but this comes at the expense of vertical resolution. For example, twice the frames of data can be captured at half the vertical resolution (i.e., six 512×512-pixel frames rather than the original three 1024×512-pixel frames).[6]

## 15.9 High Full-Well (HFW) Programming

This feature allows a single frame to be captured that exhibits a full-well capability that is three times that of a normal readout. This mode allows for use of the sensor in experiments with massive fluences or for detection of high energy X-rays. This mode is described in the *UXI Daedalus – HDD* document. Setting the subregister **HFW** to '1' asserts both sensor pins, RSLHFWModeL and RSLHFWModeR, in addition to HSTSingleShotMode after programming the RSL Interlacing shift registers as described in Section 15.8 .[6]

## 15.10 Zero Dead Timing (ZDT) Programming

This feature interlaces pixel rows both spatially and temporally permitting continuous data collection. This mode is described in the *UXI Daedalus – HDD* document. The mode can be set for both left and right hemispheres independently through subregisters **ZDT_L** and **ZDT_R**, respectively. Six 512×512-pixel frames are generated if ZDT is implemented. Even row shutters are 'open' according to the conventional HST scheme. Odd rows have inverted timing ('open' while the even rows are 'closed' and vice-versa).[6]

# 16 RS422 USB Cable

The RS422 USB cable that has been tested for the camera board is the ACCESS I/O Products, Inc. USB-422-IND cable. The datasheet and drivers for this product can be accessed at https://accesio.com/?p=/usb/usb-232-422485-IND.html .

# 17 Software Support

The 'nsCamera' python package provides a software driver and user interface for operating the NSGC camera. The software runs under Windows, MacOS and Linux. Python 3 is recommended; Python 2 is supported but deprecated.

The software is distributed as a compressed archive or is available as part of a complete pre-packaged python environment. Instructions for installation and use are given in the project's README.md file. The software is also available as a MicroManager plugin for use in imageJ.

A Jupyter notebook tutorial *nsCameraTutorial.ipynb* that introduces the software and demonstrates many of the board's features can be found in the *nsCamera/docs* directory. This directory also contains detailed code documentation, as well as the sample script *nsCameraExample.py* which demonstrates the operation of the camera and may be used as a skeleton for the development of user applications. The test script *testSuite.py* is also included which performs an extensive sequence of tests to verify the proper operation of the hardware and software.

# 18 ELM-U References

LLNL's Enterprise Lifecycle Management – Unclassified (ELM-U) stores various references for the LLNL V4.0 board. Specifically, the revisions of the board are documented and divided into two boards: FPGA and ADC board. The references include schematics, Gerber files, CAD drawings, etc. The ELM numbers for the FPGA and ADC boards are *1002464594* and *1002550381* respectively.

# 19 Change note history

See the first page of this document for more recent changes.

| Rev. | Date | Section Edits | Eng. | Description of Change |
|---|---|---|---|---|
| **1.0** | 10/19/17 | N/A | CCM | Initial Release |
| **1.01** | 12/27/17 | 13 | CCM | Fix dual_edge_Trig_en description – s.b. in trigger_ctl register |
| **1.02** | 1/22/18 | 13 | CCM | Fix typos in register table entries for register 0x99 ADC5_DATA_4. |
| **1.03** | 2/8/18 | 14.7 | CCM | Add table listing all FPGA resets for rad hard version. |
| **1.04** | 3/7/18 | 11-14, 16 | JMH | Divergence of Icarus & Daedalus ICD versions<br>Change to 12-bit register addresses<br>Removed specifications for Sandia Rev C board |
| **1.05** | 3/20/18 | 13 | JMH | Removed obsolete registers |
| **1.06** | 4/10/18 | - | JMH | Divergence of LLNLv1 and LLNLv4 versions<br>Expanded FPGA_NUM definitions |
| **1.07** | 6/26/18 | - | JMH | Spinoff of sensor-specific implementation into separate document.<br>Added subregister names to register list |
| | | 1.1-12 | BTF | Revised ADC monitor registers, removed temperature sensor data and pper registers, revised DAC registers, replaced POT with DAC channel, revised POT11 to DAC Channel G, ADC_CTL register and ADC_RESET register now controls only 4 of the 16-bit ADCs (ADS8568) |
| **1.08** | 7/31/18 | All | JMH | Register and subregister definitions added<br>Reintegration of sensor-specific details |
| **1.09** | 9/26/18 | 1.1-12 | JMH<br>BTF | Revised ADC monitor registers, removed temperature sensor data and pper registers, revised DAC registers, replaced POT with DAC channel, revised POT11 to DAC Channel G, ADC_CTL register and ADC_RESET register now controls only 4 of the 16-bit ADCs (ADS8568) |
| **1.10** | 12/5/18 | | JMH | Updated default DAC values, VRESET_HIGH is 16 bits |
| **1.11** | 6/4/19 | 13 | JMH | Added SW_TRIG_EN, Daedalus mode details |

| | | | | |
|---|---|---|---|---|
| **1.12** | 6/24/19 | All | BTF | Edited the description of each pot for an ICARUS configured board. Pointed the register map descriptions to the appropriate section to describe the pot/dac channel. Removed sensor power control section 6.4. Edited automatic sensor detection is unavailable currently. Minor edits throughout the sections.<br>Synced to nsCamera 2.0.8 |
| **1.13** | 9/30/19 | 5, 7, 13.1 | BTF | More detailed information on the Daedalus-configured board for pot values. Added TRIGGER_CTL register info for previous firmware version vs. after of 9/19. |
| **1.14** | 11/8/19 | 3 | BTF<br>JMH | Updated block diagram to include BJT dosimeter.<br>Updated ADC_PPER description, trigger operations |
| **1.15** | 11/13/19 | 2, 6, 7, 11.1, 13, 16 | BTF | Removed italicization and modified ADC monitor table for Daedalus to be 1.0 and 2.0 instead of 1 and 2 V. Resolved comments made by Jeremy. Updated default DAC channel voltages in Register Map section. Need input on if we want to describe pot voltage values in the register map as it is redundant from Section 6 and 7. Need update on CTL_REG in Section 13 for reverse readoff and slow readoff. |
| **1.16** | 1/5/20 | 9, 11 | BTF | Reworded Temperature Sensor section for the scale from 12-bits to 7-bits. Revised register definitions to match LLNL_v4.py board file script with respect to monitor ADC names. |
| **1.17** | 2/14/20 | 11 | BTF | Removed SENSOR_VOLT_STAT and SENSOR_VOLT_CTL registers. Removed September 2019 TRIGGER_CTL setting for the trigger modes. |
| **1.18** | 2/29/20 | 5.1 6, 12.1 | BTF | Described in more detail on Daedalus sensor pins that are disconnected by DAC channels, yet able to be monitored by ADC monitors. Moved tables 3 and 4 in order (MS Word issue possibly). Edited dual edge trigger section.<br>Synced to nsCamera 2.0.9 |
| **1.19** | 5/4/20 | 3, 6, 11, 14.1, 15 | BTF | Attempted to resolve comments: resolved figure with Daedalus timing block, explained more details on dual edge trigger, described details for Daedalus registers, and added what features work for Daedalus and what features do not. Edited section 14.1 to accommodate v4 ADC context. |
| **1.20** | 5/24/20 | 5,7,11 13,14 | JMH | Sensor-specific details moved to sensor sections and restructured. Removed unused quad_enable registers. Updated Daedalus DAC & monitor assignments |
| **1.21** | 8/13/20 | 12, 14, 15 | BTF | Added V4 specific register and sub registers to accommodate PDBIAS status, control of PDBIAS, and control of ROW_DCD_ENABLE. Edited Section 14.2 and 15.1 tables. |

# 20 References

1) Sandia National Laboratories, et al. “UXI ICARUS – Focal Plane Array Interface Document.” Revision 6, 8/23/16
2) Claus, L., et al. “An overview of the Ultra-Fast X-ray Imager (UXI) program at Sandia Labs.” Proceedings Volume 9591, *Target Diagnostics Physics and Engineering for Inertial Confinement Fusion IV*; 95910P (2015); doi: 10.1117/12.2188336
3) Claus, Liam D., et al. “The Ultrafast X-ray Imager (UXI) Program.” No. SAND2016-7045PE. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 2016.
4) Claus, L., et al. "Design and characterization of an improved 2 ns multi-frame imager for the ultra-fast x-ray imager (UXI) program at Sandia National Laboratories." Proc. SPIE 10390, *Target Diagnostics Physics and Engineering for Inertial Confinement Fusion VI*, 103900A (24 August 2017); doi: 10.1117/12.2275293
5) Sanchez, Marcos. “UXI_Icarus_FPA-2.” Version 6. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), August 23, 2016.
6) Sanchez, Marcos. “UXI_Daedalus_HDD.” Version 9. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), July 15, 2019.