IM Release # LLNL-AR-741177	Revision: 2.1

Nanosecond Gated CMOS Camera (NSGCC) ICD

LLNL v1 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
2.02	1/8/2021	11.1	BTF	Added five more bits to MISC_SENSOR_CTL register for accumulation mode control and reordered the register.
2.01	11/3/2020	11.1	BTF	Added MISC_SENSOR_CTL register to control miscellaneous Icarus sensor pins.
2.0	10/16/2020	6.3, 11.1, 13, 14, 15, 17	BTF	Added Section 15 mentioning the RS422 USB driver location that is used and primarily tested with the hardware. Added Section 17 to discuss ELM-U references to the board. Rewrote Daedalus sections for bypass Phi Clock and RSL programming in Section 14 . 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 11.1 . Updated Dual Edge Trigger in Section 6.3 . Moved Power Save Mode Section to Icarus Implementation.
1.22	9/22/2020	2, 11.1, 14	BTF	Added the sensor readoff time for Daedalus. Fixed up register map with respect to Daedalus implementation. Updated Daedalus Implementation section.
1.21	8/13/2020	6.3, 12.7, 13, 14	BTF	Will test Section 6.3 comment as well as Section 11 register map in lab. Revised Section 12.7 file names. Revised Sections 13 and 14 for Tables 17 and 19 by removing parenthesis and associated number of ADC5 monitor sub channel.
1.20	5/26/20	5,7,11 13,14	JMH	Sensor-specific details moved to sensor sections and restructured. Removed unused quad_enable registers. Updated Daedalus POT & monitor assignments

Previous change notes may be found in Section 18

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). The Icarus is a 1024 × 512-pixel array with 25 µm spatial resolution containing four frames of storage per pixel and has improved timing generation and distribution components while achieving 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 expanded set of features for a wider variety of applications from interlacing of rows in each frame to configurability of all shutters. [1] See section 0for details regarding the Icarus implementation of the firmware and section 14 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 the Icarus, 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 1.0 Board' was the result of these efforts. It mounts all the components required to fully utilize the Icarus including analog to digital converters to convert pixel data and various system voltages to digital form for readout and analysis, digital potentiometers for remote configuration of critical bias 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.

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 1.0 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 potentiometers (POTs), 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 (approximate, does not include software overhead)

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 thirteen digital POTs and one 8-channel ADC for calibration, monitoring, and tuning of critical bias voltages
- 10 kHz Heartbeat clock on Trigger3 BNC
- Timer/Counter which increments at 1 second intervals while FPGA is running
- Temperature sensor to monitor the system temperature

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

- 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 1.0 Board, including the FPGA, is shown in Figure 1. The image sensor attaches to the Version 1.0 Board via a SamTec SEAF connector.

3 Block Diagram

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

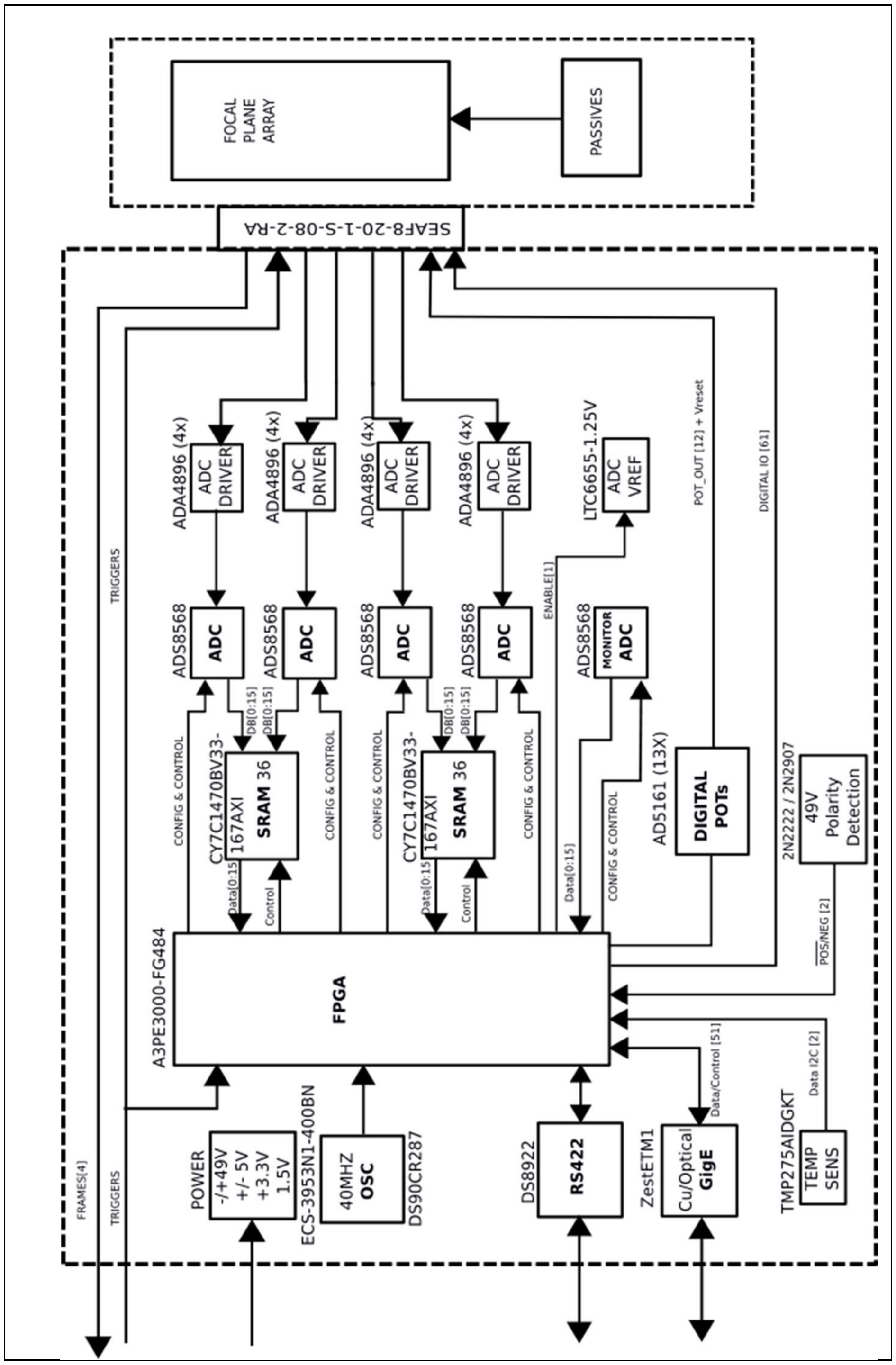

Figure 1: Version 1.0 Board, System Block Diagram

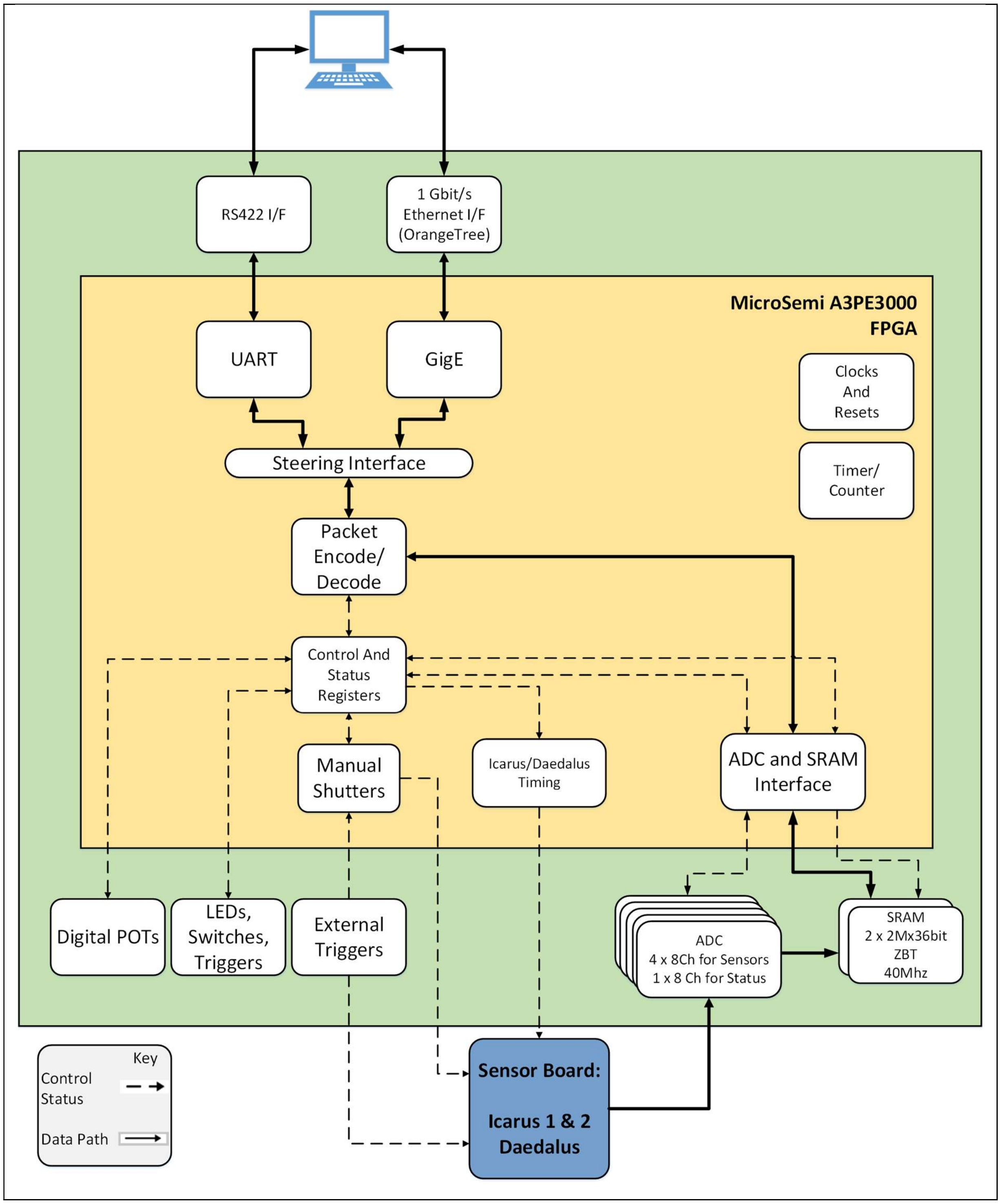

Figure 2: FPGA Block Diagram

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, 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 configuration pins on the FPGA (called *comm_port_sel_i*[1:0]). Table 2 **i**llustrates communications port selection.

com_port_sel_i (input pin) state			**Port Selected**
comm_port_sel_i[1:0] logic level	*JMP1 position*	*JMP2 position*	
00	2-3	2-3	RS422
01	2-3	1-2	N/A
10	1-2	2-3	Gigabit Ethernet
11	1-2	1-2	Reserved

Table 2: Communications Port Selection through 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 five analog-to-digital converters (ADCs) on the board. All five ADCs are Texas Instruments ADS8568SPM, which are eight-channel 16-bit devices. Four of the ADCs are used to convert the image sensor's analog pixel information into 16-bit digital data. The fifth ADC (ADC5) is used to monitor a subset of the board's potentiometers (POTs). See Sections 0and 14 for the mapping of ADC5 inputs for the Icarus and Daedalus sensors, respectively.

The FPGA reads the ADC5 periodically and writes the voltage data into the **ADC5_DATA_1** through **ADC5_DATA_4** registers; these can be read by the user at any time. The time between updates of these registers is controlled by the **ADC5_PPER** register.

By populating and de-populating surface-mount selection resistors, the user can change the connectivity of three of ADC5's channels, numbers 3, 5, and 7. Channels 3 and 5 can be connected to external signals via J32, and channel 7 can be connected to VRST. The schematic snippet in Figure 3 illustrates the resistors that need to be populated/de-populated, and the connectivity to J32.

Prior to use, the host software must configure the ADCs properly using the **ADC_CTL** and **ADC*X*_CONFIG_DATA** registers (see Section 11.1). 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).

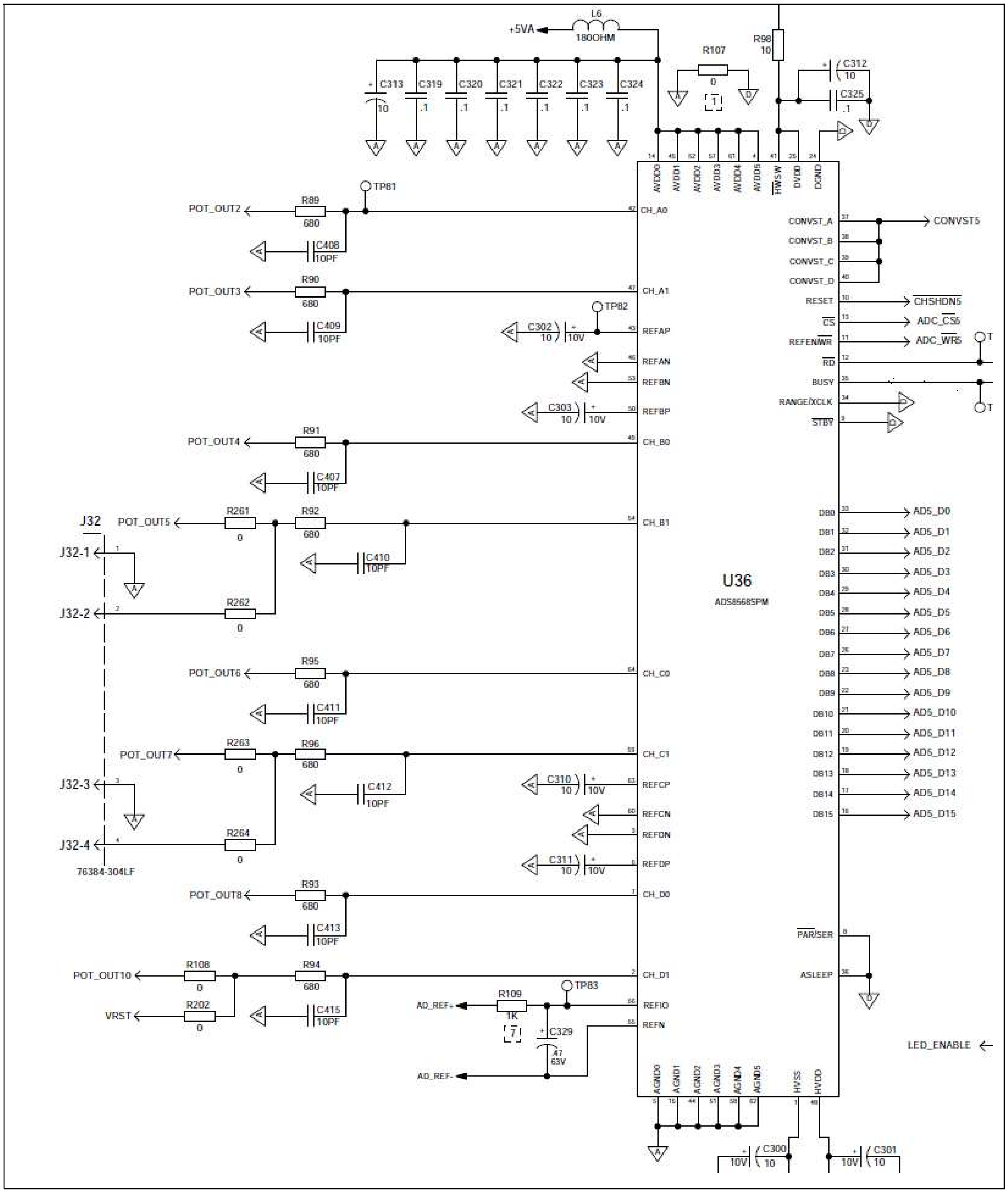

Figure 3: ADC5 – Illustration of Channel 3, 5, and 7 Connectivity Options

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 requested data is read from the sensor and stored in the SRAM. To handle storage of this data, the Version 1.0 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 to commence readout of pixel data from SRAM to the host for storage and processing.

5.3 Automatic Sensor Detection

POT11 assists in the automatic sensor detection process, where the attached sensor type is detected upon power on or reset, and the appropriate FPGA functions are configured to support the detected sensor. At system power on or reset, the FPGA will automatically program POT11 to 3.3 V for 2 seconds, then program it back to its default voltage. While at 3.3 V, the sensor_det_i signals from the sensor are biased such that they can be sampled by the FPGA to determine the sensor type that is attached. The FPGA will then assert bits in the **SENSOR_VOLT_STAT** status register indicating which sensor has been detected (see Table 3).

Sensor Detect (FPGA input pins)		FPGA Control Bits (in SENSOR_VOLT_STAT register)		
sensor_det(1)	*sensor_det(0)*	*icarus_det*	*daedalus_det*	*reserved*
1	1	1	0	0
0	1	0	0	1
0	0	0	1	0
1	0	N/A	N/A	N/A

Table 3: Sensor Detect (sensor_det) Truth Table

5.4 Sensor Power Control

Sensor power can be enabled by either the FPGA under software control, or by PCB-installed hardware circuitry. In Table 4, *bypass_sensor_det* is a bit in the **SENSOR_VOLT_CTL** register, *icarus_det* is a bit in the **SENSOR_VOLT_STAT** register that is generated by the state of the sensor_det[1:0] pins, and *pos_n*, and *neg_p* are signals from PCB circuitry. The last column, *Sensor_power_on*, is an output signal of the FPGA; when asserted, power to the sensor is turned on.

When *bypass_sensor_det* is low (logic 0), sensor power is dependent only on the state of *icarus_det*, *pos_n*, and *neg_n*. When it is high (logic 1), sensor power is enabled independent of any other signal.

bypass_ sensor_ det	Icarus_det	Pos_not	Neg_p	Sensor_ power_ on
0	1	0	0	1
0	0	1	1	1
1	X	X	X	1

Table 4: Sensor Power Enable Truth Table

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 single 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 200 ns pulse width as measured on the board
- For Icarus sensor operation, a minimum of 11.5 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 the '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
- To initiate the trigger sequence, the external Fine Trigger shall be driven low for a minimum of 11.5 µs, and then returned to the high state. For Daedalus sensor operation, the Fine Trigger must be held low at least 16 µs.
- The FPGA shall detect the falling edge of the external Fine Trigger and treat this event internally as the Coarse Trigger
- The FPGA shall detect the rising edge of the external Fine Trigger and treat this event internally as the Fine Trigger.

Note that a successful edge detect (w0_top_l_edge1) assertion after the assertion of the Fine Trigger is not possible on the Icarus. Please see the 'UXI_Icarus_FPA' document for more details regarding the edge detect signal from the Icarus sensor.[5] For Daedalus operation, a successful edge detect of SH2_fall_UR (the diagnostic one-shot falling edge of shutter two) is asserted when dual edge trigger is performed twice.[6] The first trigger is used to generate the coarse trigger and the second execution of the same trigger is used to generate the fine trigger. The first trigger pulse must be asserted high for at most 16 µs. The second can be asserted for an arbitrary pulse.

7 Digital Potentiometers

There are thirteen digital potentiometers (POTs) on the board available to bias miscellaneous image sensor functions. Each of these devices is an Analog Devices AD5161; configuration is performed via the POT data registers (**POT_REG4_TO_1**, **POT_REG8TO_5**, **POT_REG12_TO_9**, and **POT_REG13**), and the POT control register (**POT_CTL**). To ensure that the POTs 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 POTs to the default values contained in their respective POT data registers. See sections 0and 14 for the Icarus and Daedalus POT assignments, respectively.

When the user writes to the **POT_CTL** register, a POT write command is initiated. It takes some time for the command to be generated, serialized, sent to the POT, and for the POT interface module to then terminate the process. If a POT write is initiated whilst a previous POT write is in progress, the second POT 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.

There is a similar delay required due to the special Anti-Bloom function associated with **POT11**. Since it takes some time for the Anti-Bloom function to execute (see Section 5.3), the interface software must not execute a POT write command for at least 50 µs following a power cycle or a board reset.

8 Temperature Sensor

A TI275 temperature sensor is used to monitor sensor temperature; it is placed on the board adjacent to the sensor connector and provides ±0.5 °C accuracy. After the system is powered on, the FPGA will continually read the TI275 at an interval determined by the **TEMP_SENSE_PPER** register; the temperature is stored in the **TEMP_SENS_DATA** register, where it can be read by software using the Read Single command.

The value programmed into the **TEMP_SENSE_PPER** register represents the number of system clock cycles between the end of one TI275 temperature read cycle and the beginning of the next. Therefore, the TI275 read frequency is a function of **TEMP_SENSE_PPER** and the time it takes to read off the TI275. The frequency at which the TI275 is read (and the **TEMP_SENS_DATA** register is updated), use the following formula:

$$f_{\text{read}} = f_{\text{system\_clock}} / (n_{\text{read}} + TSP)$$

where n_{read} is the number of clock cycles to read the TI275 = 2917, $f_{\text{system\_clock}} = 40{,}000{,}000$, and TSP is the contents of the **TEMP_SENSE_PPER** register.

9 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.

9.1 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 5.

16 bits	*4 bits*	*12 bits*	*32 bits*	*16 bits*
Preamble	Command	Address	Data	CRC16

Table 5: 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 0x1: Read Single 0x2: Read Burst (i.e., Read Pixels) 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 6: Command Packet Fields

9.2 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 7: 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. 0x8: Write Single 0x9: Read Single 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. 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. Bit[0]: CRC error Bit[1]: Invalid Command – command not executed 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 8: Response Packet Fields

9.3 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 9: 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. 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 10: Burst Response Fields

10 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 1.0 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 1.0 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.

11 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.

11.1 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		`0x8100_0301`
Product Number of the FPGA design Eight-character sequence: 1: Board developer 2: Board revision number 3-5: unused 6: Communication interfaces 7: Radiation tolerance 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 (e.g., `0x29` for the 29th day of the month)				
	15:8	Month of FPGA code release (e.g., `0x12` for the month of December)				
	23:16	Year of FPGA code release (e.g., `0x18` for the year 2018)				
	27:24	Unused				

	31:28	Board version (e.g., `0x1` for v1 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 goes low for 10 µs, then 11.5 µs after the coarse trigger goes high, the fine trigger goes low for 10 µs.		
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)		
0x024	**STAT_REG**	V1, V4	Read-only	---
Status Register. (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_0002`

Control Register	0	Unused		
	1	**LED_EN** – Enable user-managed LEDs		
	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.		
0x026	**POT_CTL**	V1	Read/Write	`0x0000_0000`
Pot configuration control for specific channels	0	POT_CONFIG. When written with a '1', the POT selected by POT_SEL will be configured with the value in its corresponding register. This bit is self-clearing.		
	4:1	POT_SEL[3:0]. Selects the potentiometer to configure. Selection is literal, i.e.: "`0001`" = select POT1, "`0010`" = select POT2 … "`1101`" = select POT13		
0x027	**POT_REG4_TO_1**	V1	Read/Write	---
POT 1 through POT 4 configuration data. See Section 7 for each pot description.	7:0	**POT1 / COL_BOT_IBIAS_IN** – Control voltage to bottom side of sensor's column bias.[5]		0 V
	15:8	**POT2 / HST_A_PDELAY**– see register `0x096.` Control voltage to respective sensor pin for the A side p transistor delay buffer voltage. Decrease in this voltage increases delay of side n transistor.[5]		0 V
	23:16	**POT3 / HST_B_NDELAY** – see register `0x096.` Control voltage to respective sensor pin for the B side n transistor delay buffer voltage. Decrease in this voltage increases delay of side n transistor.[5]		3.3 V
	31:24	**POT4 / HST_RO_IBIAS** – see register `0x097.` Control voltage to the ring with capacitors oscillator (selected by register 0x047). Decrease in this voltage increases the speed of oscillator.[5]		2.5 V

		31:24	**POT4 / HST_OSC_CTL** – see register `0x097`. Control voltage to the 500 MHz oscillator (selected by register 0x047). Decrease in voltage increases the speed of oscillator.[5]		1.0 V
0x028	**POT_REG8_TO_5**		V1	Read/Write	---
POT 5 through POT 8 configuration data. See Section 7 for each pot description.		7:0	**POT5 / HST_OSC_VREF_IN** – see register `0x097`. Reference voltage for relaxation oscillator.[5]		2.9 V
		7:0	**POT5 / HST_RO_NC_IBIAS** – see register `0x097`. Control voltage to the ring without capacitors oscillator (selected by register 0x047). Decrease in voltage increases the speed of oscillator.[6]		1.0 V
		15:8	**POT6 / HST_B_PDELAY** – see register `0x098`. 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
		15:8	**POT6 / HST_OSC_VREF_IN** – see register `0x098`. Reference voltage for 500 MHz oscillator.[6]		1.0 V
		23:16	**POT7 / HST_OSC_CTL** – see register `0x098`. Control voltage to the relaxation oscillator (selected by register 0x047). Decrease in voltage increases the speed of oscillator.[5]		1.45 V
		31:24	**POT8 / HST_A_NDELAY** – see register `0x099`. 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
0x029	**POT_REG12_TO_9**		V1	Read/Write	`0xF527_0000`
POT 9 through POT 12 configuration data. See Section 7 for each pot description.		7:0	**POT9 / COL_TOP_IBIAS_IN** – Control voltage to top side of sensor's column bias.[5]		0.025 V
		15:8	**POT10 / HST_OSC_R_BIAS** – Control voltage to current sink of ring oscillators with and without capacitors.[5]		0.0135 V
		23:16	**POT11 / VAB** – Controls the pixel anti-bloom transistor voltage.[5, 6]		0.5 V
		31:24	**POT12 / HST_RO_NC_IBIAS** – Control voltage to the ring without capacitors oscillator determined by register 0x047. Decrease in voltage increases the speed of oscillator.[5]		2.5 V
0x02A	**POT_REG13**		V1	Read/Write	`0x0000_003A`
POT 13 configuration data. See Section 7 for each pot description.		7:0	**POT13 / VRST** – see register `0x099`. Controls the pixel reset voltage.[5]		0 V

0x02B	**LED_GP**		V1, V4	Read/Write	`0x0000_0000`
General purpose LED control.	7:0	LED7 … LED0 - Bits [7:0] will light up LED_IO-8 through LED_IO-1 when written with a ‘1’, respectively.			
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. 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_POTSCONFIGURED** – all POTs 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 'pots configured' AND **TRIGGER_CTL**[0] AND 'HST_Configured' AND NOT 'coarse trigger detected' AND NOT 'fine trigger detected'. 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)			
		27:16	**STAT_TEMP** - Temperature Sensor [11:0]			
		31:28	**STAT_PRESS** - Pressure Sensor (future use)			
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** (`0x031`) . 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. 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 12 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.			
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 be used to determine 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 be used to determine if an error occurred at the TX FIFO output in the readout pipeline during SRAM readoff. 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.		
0x03A	**TRIGGER_CTL**	V1, V4	Read/Write	`0x0000_0000`
Trigger control	0	**'HW_TRIG_EN'** - When '1', 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 falling edge initiates an internal coarse trigger of 10 µs. The second rising 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	---
Request SRAM Readoff	0	**READ_SRAM** - Request SRAM Data when '1'. This bit is self-clearing. When using RS422, software should send no additional command packets after setting this until after 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 during pixel initialization, in 25 ns increments		
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	7:0	**VRESET_HIGH** - *Icarus 1 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. POT13 programmed value during 2-frame Icarus recovery period. `0xFF` is maximum voltage; `0x00` is minimum voltage.			

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
	0	**MANSHUT_MODE**			

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	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.			
0x082	**SENSOR_VOLT_STAT**		V1	Read-only	---
Status of sensor voltage-related signals	0	**SENSOR_POSN** - *pos_n*			
	1	**SENSOR_NEGP** - *neg_p*			
	2	**ICARUS_DET** - Derived from the sensor_det[1:0] bits from the PCB			
	3	**DAEDALUS_DET** - Derived from the sensor_det[1:0] bits from the PCB			
	4	**HORUS_DET** - Derived from the sensor_det[1:0] bits from the PCB			
	5	**SENSOR_POWER** - Asserted according to the truth table in Table 4			

0x083	SENSOR_VOLT_CTL		V1	Read/Write	0x0000_0000
Control of Sensor Voltage Supply		0	*bypass_sensor_det*. When '0', *sensor_power_on* is enabled when *pos_n, neg_p*, and *icarus_det* follow the truth table in Table 4. When '1', *sensor_power_on* is enabled regardless of the state of *pos_n, neg_p*, and *icarus_det*		
0x090	**ADC_CTL**		V1, V4	Write; Self-clearing	---
Control of TI8548 ADCs		0	Configure ADC 1		
		1	Configure ADC 2		
		2	Configure ADC 3		
		3	Configure ADC 4		
		4	Configure ADC 5		
0x091	**ADC1_CONFIG_DATA**		V1, V4	Read/Write	0x81A8_83FF
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	0x81A8_83FF
Configuration data to be manage ADC 2		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		
0x093	**ADC3_CONFIG_DATA**		V1, V4	Read/Write	0x81A8_83FF
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	0x81A8_83FF
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_CONFIG_DATA**	V1, V4	Read/Write	0x81A8_83FF
Configuration data to be written to ADC 5 (see Texas instruments document SBAS543A for details)	9:0	**ADC5_VREF** – Internal reference DAC setting (1 LSB = internal Vref / 1024)		1111111111
	13	**ADC5_VREF3** – Internal reference voltage (‘0’ = 2.5 V, ‘1’ = 3 V)		1
	15	**ADC5_INT** – Internal reference enable		1
	24:19	**ADC5_MULT** – Voltage multiplier controls		110101
0x096	**ADC5_DATA_1**	V1, V4	Read-only	---
POT 2 and 3 data. Full scale is 0-5 V. See Section 7 for each pot description.	15:0	**MON_CH2 / MON_HST_A_PDELAY** – see register 0x027		
	15:0	**MON_CH2 / MON_TSENSEOUT**		
	31:16	**MON_CH3 / MON_HST_B_NDELAY** – see register 0x027		
	31:16	**MON_CH3 / MON_BGREF**		
0x097	**ADC5_DATA_2**	V1, V4	Read-only	---
POT 4 and 5 data. Full scale is 0-5 V. See Section 7 for each pot description.	15:0	**MON_CH4 / MON_HST_RO_IBIAS** – see register 0x027		
	15:0	**MON_CH4 / MON_HST_OSC_CTL** – see register 0x027		
	31:16	**MON_CH5 / MON_HST_OSC_VREF_IN** – see register 0x028		
	31:16	**MON_CH5 / MON_HST_RO_NC_IBIAS**– see register 0x028		
0x098	**ADC5_DATA_3**	V1, V4	Read-only	---
POT 6 and 7 data. Full scale is 0-5 V. See Section 7 for each pot description.	15:0	**MON_CH6 / MON_HST_B_PDELAY** – see register 0x028		
	15:0	**MON_CH6 / MON_HST_OSC_VREF_IN** – see register 0x028		
	31:16	**MON_CH7 / MON_HST_OSC_CTL** – see register 0x028		
	31:16	**MON_CH7 / MON_COL_TST_IN**		
0x099	**ADC5_DATA_4**	V1, V4	Read-only	---
POT 8 and VRST data. Full scale is 0-5 V. See Section 7 for each pot description.	15:0	**MON_CH8 / MON_HST_A_NDELAY** – see register 0x028		
	15:0	**MON_CH8 / MON_HST_OSC_PBIAS_PAD**		
	31:16	**MON_VRST** - Note this is the output of the POT 13 conditioning circuit (TP19) that is connected to the sensor. The direct output of POT 13 (TP24) is not connected to ADC5. See register 0x02A		
0x09A	**ADC5_PPER**	V1, V4	Read/Write	0x001E_8480
Polling period for ADC5, in 20MHz clock cycles.	27:0	ADC5 polling period. Default is 100ms = 2,000,000 = 0x1E_8480		
0x09B	**ADC_STANDBY** (version < rev AD)	V1, V4	Read/Write	0x0000_001F

	ADC_RESET (versions rev AD to present)				
Controls standby pins to ADC 1 through 5 (boards before rev AD) Controls reset pins to ADC 1 through 5 (for versions of AD to present)	0	ADC1 standby / reset			
	1	ADC2 standby / reset			
	2	ADC3 standby / reset			
	3	ADC4 standby / reset			
	4	ADC5 standby / reset			
0x0A0	**TEMP_SENSE_PPER**		V1, V4	Read/Write	0x001E_8480
Polling period for TI TMP275 temperature sensor, in 40MHz clock cycles.	27:0	Temperature sensor polling period. Default is 50ms = 2,000,000 = 0x1E_8480			
0x0A1	**TEMP_SENSE_DATA**		V1, V4	Read-only	---
Data read from TI TMP275 temperature sensor	11:0	Temperature sensor data read out during most recent poll			
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
Phi clock programming, LSBs	31:0	Delay timing pattern bits for programming Phi clock [31:0]			
0x123	**HST_PHI_DELAY_DATA_HI**		V1, V4	Read/Write	0x0000_0000
Phi clock programming, MSBs	7:0	Delay timing pattern bits for programming Phi clock [39:32]			
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
Initiates HST generator	0	Initiates HST generator[6]		
0x131	**HST_DELAY_EN**	V1, V4	Read/Write	0x0000_0000
Enables trigger delay cell for HST generator	0	Enables trigger delay cell for HST generator[6]		
0x132	**HST_TEST_PHI_EN**	V1, V4	Read/Write	0x0000_0000
Enables the external Phi input to bypass the entire HST generator.	0	Enables 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. When ‘1’, the following occurs: 1. The sensor’s test source followers are enabled on each column current source. 2. The sensor’s RDcdEn pin (named RowDcdEn_o in the FPGA) is driven low by the FPGA to disable it 3. The sensor’s nQuenchEn pin (named quench_en_n_o in the FPGA) is driven high by the FPGA to disable it		

		4. The user can then use VRST to drive an analog voltage to the sensor's ColTstIn pin, since VRST is jumpered to ColTstIn. 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			

12 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 are 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.

12.1 Software Reset

A software reset has been implemented which resets all logic within the FPGA. To initiate a 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.

12.2 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 transmit RS422 section (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 receive RS422 section (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.

12.3 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.

To enable the host to read the counter value, it has been connected to 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).

12.4 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.

12.5 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.

12.6 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.

12.7 Resets

Due to the complexity of the radiation tolerant version of the FPGA, numerous resets are generated and used for different purposes. Table 11 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 Rs422.vhd 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 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 packet_enc_dec.vhd

Table 11: Device Resets

13 Icarus implementation

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

13.1 ADC Interface and Control

The FPGA controls the five 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 is used to monitor potentiometer voltages. All five ADCs are Texas Instruments ADS8568SPM, 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 12.

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 12: Sensor Channel to ADC Device Channel Connectivity

13.2 ADC-Potentiometer Mapping

The Icarus-configured board POT and monitor assignments are shown in Table 13. The A0-D1 designation for channel numbers is that used in the manufacturer's data sheet.

Pot	Monitor Channel	ADC5 Channel : Pin	Function	Description	Sensor Pin	Nominal Voltages (V)
POT1	---	---	*COL_BOT_IBIAS_IN*	Column bias for bottom hemisphere of ICARUS sensor.	153	0
POT2	**MON_HST_A_PDELAY**	A0: 42	*HST_A_PDELAY*	Delay voltage for A side pixel array. Increase to add delay.	78	0
POT3	**MON_HST_B_NDELAY**	A1: 47	*HST_B_NDELAY*	Delay voltage for B side pixel array. Decrease to add delay.	86	3.3
POT4	**MON_HST_RO_IBIAS**	B0: 49	*HST_RO_IBIAS/ HST_RO_NC_IBIAS*	Ring oscillator with capacitors bias voltage.	46	2.5
POT5	**MON_HST_OSC_VREF_IN**	B1: 54	*HST_OSC_VREF*	Oscillator voltage reference.	87	2.9
POT6	**MON_HST_B_PDELAY**	C0: 64	*HST_B_PDELAY*	Delay voltage for B side pixel array. Increase to add delay.	94	0
POT7	**MON_HST_OSC_CTL**	C1: 59	*HST_OSC_CTL*	Relaxation oscillator bias control voltage.	48	1.45
POT8	**MON_HST_A_NDELAY**	D0: 7	*HST_A_NDELAY*	Delay voltage for A side pixel array. Decrease to add delay.	70	3.3
POT9	---	---	*COL_TOP_IBIAS_IN*	Column bias for top hemisphere of ICARUS sensor.	8	0.025
POT10	---	D1: 2	*HST_OSC_R_BIAS*	Current sink set for ring oscillator with capacitors.	16	0.0135
POT11	---	---	*VAB*	Gate voltage of all anti-bloom transistors in the ICARUS pixel array.	28	0.5
POT12	---	---	*HST_RO_NC_IBIAS*	Ring oscillator without capacitors bias voltage.	53	2.5
POT13	**MON_VRST**	D1: 2	*VRST*	Resets the voltage for all pixels.	61, 69, 77	0.3

Table 13: ICARUS Pot and Monitor Assignments. See the 'UXI_Icarus_FPA' document for more details[5]
*Assignment determined by resistors (see section 5.1)

13.3 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 1.0 Board, VAB is controlled by the **VAB** subregister (aka **POT11**), which can be set using the setPotV command.

13.4 Sensor Power Control

Sensor power can be enabled by either the FPGA under software control, or by PCB-installed hardware circuitry. In Table 14, *bypass_sensor_det* is a bit in the **SENSOR_VOLT_CTL** register, *icarus_det* is a bit in the **SENSOR_VOLT_STAT** register that is generated by the state of the sensor_det[1:0] pins, and *pos_n*, and *neg_p* are signals from PCB circuitry. The last column, *Sensor_power_on*, is an output signal of the FPGA; when asserted, power to the sensor is turned on.

bypass_ sensor_ det	Icarus_det	Pos_not	Neg_p	Sensor_ power_ on
0	1	0	0	1
0	0	1	1	1
1	X	X	X	1

Table 14: Sensor Power Enable Truth Table

When *bypass_sensor_det* is low (logic 0), sensor power is dependent only on the state of *icarus_det*, *pos_n*, and *neg_n*. When it is high (logic 1), sensor power is enabled independent of any other signal. Currently the state of sensor_det[1:0] is only applicable to Icarus in the firmware.

13.5 High-Speed Timing (HST) Control

The High-Speed Timing Control function is discussed in detail in *UXI ICARUS – Focal Plane Array Interface Document,* so this document will not repeat the information contained in that interface document. 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.

13.6 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 resolution of the manual shutter is coarse when compared with that of 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 4. Manual Shutter Control is discussed in detail in *UXI ICARUS – Focal Plane Array Interface Document,* so this document will not repeat the information contained there.

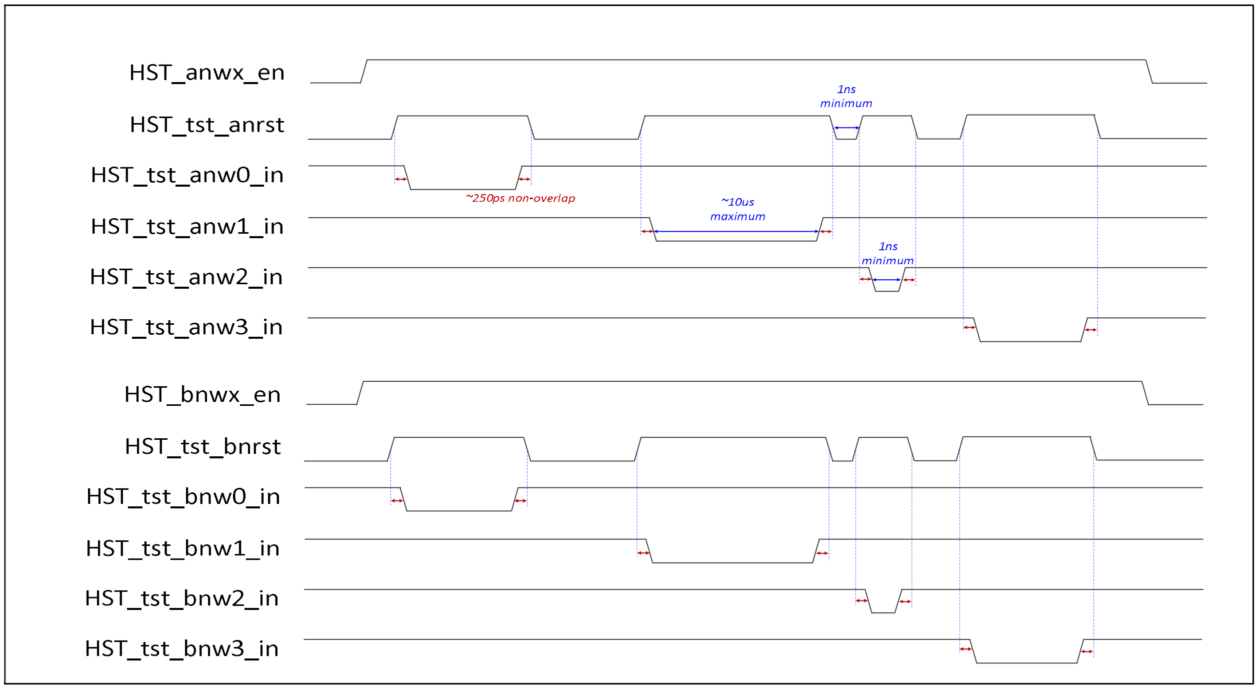

Figure 4. Manual Shutter Timing, Icarus Sensor (A and B sides are here set to the same timing)

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 **WX_INTEGRATION** and ***WY*_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 4 is initiated when the FPGA detects the rising edge of the Fine Trigger.

13.7 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. Power consumed by the sensor's digital power supply is reduced by 3.3% with Power Save Mode enabled; the sensor's analog power requirements are reduced by 0.6%. The average current and power consumption are listed in Table 15.

	Sensor +3.3 VDD	Sensor +3.3 VA

	Power Save OFF	Power Save ON	Power Save OFF	Power Save ON
Average Current (mA)	37	35.7	5.02	5.05
Average Power (mW)	122	118	16.56	16.66

Table 15: Power Save Mode Results

14 Daedalus Implementation

Currently, there are various Daedalus features that are not functional. 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

14.1 ADC Interface and Control

The reader should look to Section 13.1 for details of the image readoff ADC. Table 16 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
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

Table 16: Daedalus Sensor Channel to ADC Device/Channel Connectivity

14.2 ADC-Potentiometer Mapping

The Daedalus-configured board POT and monitor assignments are shown in Table 17. The A0-D1 designation for channel numbers is that used in the manufacturer's data sheet.

Pot	Monitor Channel	ADC5 Channel: Pin	Function	Description	Nominal Voltages (V)
4	**MON_HST_OSC_CTL**	B0: 49	**HST_OSC_CTL**	Relaxation oscillator bias control voltage.	FPGA dependent
5	**MON_HST_RO_NC_IBIAS**	B1: 54	**HST_RO_NC_IBIAS**	Ring oscillator without capacitors bias control voltage.	FPGA dependent
6	**MON_HST_OSC_VREF_IN**	C0: 64	**HST_OSC_VREF_IN**	Oscillator voltage reference.	FPGA-dependent
---	**MON_ TSENSEOUT**	A0: 42	TSenseOut	Temperature sense analog output with respect to sensor.	Temp. dependent
---	**MON_BGREF**	A1: 47	**BGREF**	Bandgap voltage reference output with respect to sensor.	1.0
---	**MON_COL_TST_IN**	C1: 59	colTstIn	Global column current source test pin.	0
---	**MON_HST_OSC_PBIAS_PAD**	D0: 7	**HST_OSC_PBIAS_PAD**	Current mirror node for oscillator bias.	2.0

Table 17 Daedalus Potentiometer and Monitor Assignments.
See the 'UXI_DAEDALUS_HDD' Document for more details[6]

14.3 Anti-Bloom Circuit Control

See Section 13.3 (Daedalus and Icarus behaviors are identical).

14.4 High-Speed Timing (HST) Control

See Section 13.4 (Daedalus and Icarus behaviors are identical).

14.5 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]

14.6 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]

14.7 External Phi Clock Bypass

Note: Currently, the behavior of this feature is uncertain with respect to the FPGA. An external phi clock sensor pin (HstExtPhi), can be bypassed by the FPGA. To bypass the 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]

14.8 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 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 happen simultaneously for all rows of the pixel array (the default state) or to happen 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. The trade-off of interlacing comes at the expense of vertical resolution. For example, six 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]

14.9 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. The 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 14.8 [6]

14.10 Zero Dead Timing (ZDT) Programming

This feature interlaces pixel rows both spatially and temporally allowing continuous data collection. The mode is described in *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]

15 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 .

16 Software Support

The 'nsCamera' python package provides a software driver and user interface for operating the NSG 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.

17 ELM-U References

LLNL's Enterprise Lifecycle Management – Unclassified (ELM-U) stores various references for the LLNL V1.0 board. Specifically, the revisions of the board are documented. The references include schematic, Gerber files, CAD drawings, etc. The ELM number for the board is *1000189775*.

18 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 Change to 12-bit register addresses 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 Expanded FPGA_NUM definitions

1.07	6/26/18	-	JMH	Spinoff of sensor-specific implementation into separate document. Added subregister names to register list
1.08	7/31/18	All	JMH	Register and subregister definitions added Reintegration of sensor-specific details
1.09	9/26/18	-	JMH	Miscellaneous; synced to nsCamera software release 2.0.5
1.10	12/5/18		JMH	Enter details for all RSLScan in and out registers, and BGTRIMA/B registers. Enter updated description for the FPGA_REV register
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 ICARUS and DAEDALUS configured boards. Pointed the register map descriptions to the appropriate section to describe the pots. Minor edits to all sections. 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	10/25/19	7	BTF	VAB on v1 board Daedalus was not listed on the Pot voltage control. VAB cannot be monitored, but can be set.
1.15	11/13/19	6, 7, 11.1, 13, 16	BTF	Resolved comments made by Jeremy. Updated default pot 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	1/3/20	11	BTF	Resolved definitions of ADC monitor of pots between Icarus and Daedalus. Changed FPA_FRAME_ORDER_SEL to FRAME_ORDER_SEL register.
1.17	2/14/20	5, 7, 11	BTF	Converted SENSOR_VOLT_STAT and SENSOR_VOLT_CTL registers as Icarus firmware specific registers. Removed September 2019 TRIGGER_CTL setting for the trigger modes.
1.18	3/3/20	6	JMH	Trigger controls finalized. Synced to nsCamera 2.0.9
1.19	5/4/20	3, 6, 11, 14	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. Synchronized to nsCamera 2.1

19 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.