ARCHON

A HIGH PERFORMANCE MODULAR CCD CONTROLLER

Revision 1.0.1166 – February 23, 2021

1241 Puerta Del Sol, San Clemente, CA 92673
(949) 481-1595 – http://www.sta-inc.net – customerservice@sta-inc.net

Table of Contents

Introduction

Archon is a high performance modular CCD controller developed by Semiconductor Technology Associates, Inc (STA). This manual provides an overview of the controller followed by detailed descriptions of the hardware and software. A top level block diagram is shown in Figure 1.

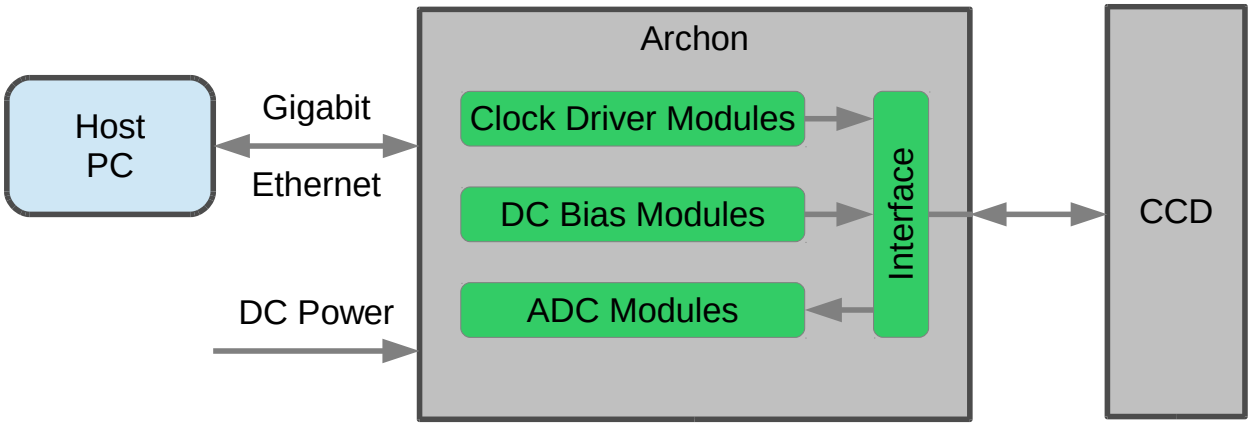


Figure 1: Top Level Block Diagram

An Archon system receives configuration information from and sends status and image data to a host PC via a gigabit Ethernet connection (either copper or fiber). Power is supplied to Archon through a circular connector carrying the DC voltages necessary for a particular system, or through a standard AC power cord for Archon AC. The CCD to be operated is connected to Archon through a custom interface board, built to route signals from the CCD cabling to the internal Archon module connectors.

Archon Feature Summary

- Compact size: Standard chassis is 11.5” x 8” x 4.5” (29.21 x 20.32 x 11.43 cm), Archon AC chassis is 11.5” x 8” x 7”.
- Modular: 12 slots for ADC, clock driver, bias, heater, or other custom modules
- Dense: Up to 4 ADC modules for 16 total CCD outputs
- Low weight: 8.5 lbs (3.9 kg) for a typical 4 channel system, 17.5 lbs (8kg) for a 16 channel Archon AC system
- Low power: 41 W for a typical 4 channel system, 89W for a 16 channel Archon AC system
- High dynamic range: 108 dB at 100 kHz, 98 dB at 1 MHz using 16 or 32 bits per sample
- Easy interfacing: standard gigabit network interface, either copper or fiber SFP module
- On-board frame buffer: 2GB RAM for flexible readout
- Timing core: 100 MHz master clock for 10 ns timing resolution
- ADC Module: 4 fully differential AC-coupled 100 MHz 16 bit channels using digital CDS, and software selectable 1.33 V or 4 V input range
- ADM Module: 18 fully differential DC-coupled 12.5 MHz 18 bit channels using digital CDS, and a 6 V input range

- Clock Driver Module: 8 or 12 channels of 100 MHz 14-bit DACs for generating slew-rate controlled, multi-level clocks from -13.000…+13.000 V
- Low Voltage Bias Module: 30 total biases at -14.000…+14.000 V, with 6 high power channels supplying up to 500 mA each (programmable current limit), and 24 low power channels supplying up to 10 mA each (1 A max total current per module)
- High Voltage Bias Module: 30 total biases at 0.000…+31.000 V, with 6 high power channels supplying up to 250 mA each (programmable current limit), and 24 low power channels supplying up to 10 mA each (1 A max total current per module)
- XV Bias Module: 4 channels at 0.000…+95.000 V and 4 channels at -95.000…0.000, each supplying up to 50 mA
- HeaterX Module: Two 25W heater drivers (intended for 25 Ohm resistors) and three temperature sensor channels, with programmable excitation currents from 25 nA to 1.5 mA
- High Speed Clock Module: 12 channels of LVDS clocks with 1 ns resolution, 12 clock magnitudes from 5.000…14.000V (up to 1A each), 12 clock offsets from -14.000V…+14.000V (10 mA each)
- LVDS Clock Module: 16 channels of LVDS clocks with 10 ns resolution, +3.3V, +/-5V, +/-16V supplies (1A each)
- Digital I/O: General purpose digital I/O lines powered internally or externally are on the LVBias, Heater, HS and LVDS modules (8 lines on LVBias/Heater, 4 lines on HS/LVDS)
- Programmable I/O: Modules with digital I/O lines contain real time 16-bit CPUs that can be user programmed for interface tasks
- All biases monitor current and voltage
- Triggering: opto-isolated BNC input and output
- Synchronization: multiple controllers can be synchronized over dedicated Cat5 cable
- Software: example GUI application provided with source for Windows/Linux/OSX
- Temperature: operating range -20C to +40C

Basic Operation

High level system configuration is accomplished by a 32-bit CPU embedded in the Archon backplane master FPGA. This CPU communicates with an upstream host (typically a data capture computer) via gigabit Ethernet using a human-readable serial text stream of commands and responses. The 32-bit CPU translates high level camera configuration commands from the host (bias voltages, clock timing, etc.) into low level commands for the FPGAs on each installed module (DACs, ADCs, etc).

The clock sequences needed to read out a CCD are generated by timing cores embedded in each FPGA. The backplane supplies a master 100 MHz clock to all timing cores. The timing cores are simple processors/state machines, with an instruction set composed of NOPs, GOTOs, CALLs, and RETURNs. Each instruction executes in a single master clock cycle (10 ns). Instruction execution can be conditional on user-set parameters. The call stack allows subroutines to be nested 16 deep. Each instruction in memory has an associated set of output signals for that clock state specific to a particular module, including clock voltage levels, clamp and sample timing for the ADCs, and triggers.

Clock Driver Module

Each clock driver module has 8 channels (or 12 for DriverX) of 100 MHz 14-bit DACs, with output swings of -13.000 V to +13.000 V (~2 mV resolution). Every clock driver has a target voltage that is set at each 10 ns clock tick. Each clock slews towards its current target at a selectable fast or slow rate. The fast and slow slew rates are configurable per channel, and the slew rate can be changed to fast or slow at each clock tick. This flexibility makes it possible to extract maximum performance from a CCD. For example, a CCD area clock could be programmed to slew slowly from inversion to a voltage just out of inversion, followed by a fast slew to its nominal high level, minimizing spurious charge and maximizing the readout rate. The same clock could then go to a very high voltage during integration to optimize MTF. The clock drivers have a series 50 ohm resistor, which is usually the limiting factor for the maximum slew rate when driving heavy capacitive loads. For light loads, the 10% to 90% time for a 10V swing is typically 22 ns at the maximum slew rate.

DC Bias Modules

Each bias module provides 30 programmable DC biases, 0.000 to +31.000 V for the HVBias module, and -14.000 to +14.000 V for the LVBias module. On each bias module, there are 6 high power (250 mA HV, 500 mA LV) biases and 24 low power (10 mA) biases. The high power biases have programmable current limits. Note that while each high power bias is rated for 250/500 mA, the combined current from all biases on a single module cannot exceed 1A. The current and voltage of each bias is monitored, both for calibration when the bias is set and to assist in debugging system faults.

The XV bias module provides 8 programmable DC biases, with four channels that range from 0.000 to +95.000 V, and four channels that range from -95.000 to 0.000 V. Each channel can supply up to 50 mA. The current and voltage of each bias is monitored, both for calibration when the bias is set and to assist in debugging system faults.

The biases start at 0V. When a “Power On” command is given, the biases rise to their nominal levels. The power-up sequence can be controlled by assigning a step number to each bias. Step 1 biases will go to their target values first. Their levels will be checked, and then step 2 biases will come up, and so on until all biases are at their nominal voltages. During a power down, the sequence is reversed. This is useful for CCDs that require particular power-up sequences to avoid damage. All of the clock drivers and biases are isolated from the CCD by solid state relays. Beyond the relays are weak (100k, 1M on XVBias) resistors to ground. The relays only connect the biases and clocks to the CCD when all Archon power supplies are at nominal levels and the FPGA has commanded the relays to engage. If the FPGA is reset, or one the system power supplies goes bad, the relays are opened and will not close again until the system has been reconfigured.

The LVBias module additionally has 8 general purpose digital I/Os, which can be powered by an internal +3.3V or by an external supply from +1.65V to +5.5V. Each group of 2 I/Os can be configured as inputs or outputs. When an output, each line can be driven high, low, or by the timing core. In addition, the digital I/O lines can be controlled by a dynamically programmable embedded 100 MHz 16-bit CPU for simple interface tasks, such as communication with an RS-232 vacuum gauge or I2C temperature sensor.

Improved versions of the bias modules have been introduced, called the HVX and LVX bias modules. They have identical functionality to the standard bias modules, but add additional buffering that enables much faster background bias polling. Note that the new extended bias modules are longer than standard modules, and can only be installed in slots 3-4 and slots 9-12.

ADC Module

Each ADC module has four 100 MHz 16-bit channels with preamps, designed for CCD pixel clocks of up to 3 MHz. Up to four ADC modules can be installed in an Archon chassis. The preamps are fully differential and AC-coupled. The preamp gain is software selectable, and can be set for either a 4V or 1.33V full scale input. Because the preamps are AC-coupled, an integrated DC-restore clamp must be activated periodically. Clamping is usually done once per line, either during overscan pixels or during the vertical transfer. The clamp level is programmable to map the CCD reset level near the top of the ADC range. Correlated double sampling (CDS) is performed digitally in the FPGA, which allows multiple reset and video levels to be averaged, driving noise down as the pixel clock is reduced. For typical CCD timings at high gain (full scale ≈ 1.33V), noise is 0.79 DN (16 uV RMS) at a 1 MHz pixel clock with a grounded input, and 0.28 DN (5.7 uV RMS) at 100 kHz. At low gain (full scale ≈ 4V), noise is 0.66 DN (40 uV RMS) at a 1 MHz pixel clock with a grounded input, and 0.24 DN (15 uV RMS) at 100 kHz. The previous values assume 16 bits per pixel. Digitizing signals with noise less than the equivalent of 0.5 DN at 16 bits requires additional bits per pixel. 32 bits per pixel can be acquired in the controller's high dynamic range mode. In addition to the normal CDS pixel data, portions of the raw ADC data can be simultaneously captured to a memory buffer. This allows a detailed, low-noise oscilloscope view of the CCD output waveform during device tuning. The raw waveforms simplify the examination of clock feedthrough, reset level stability, and optimal CDS sample points.

ADM Module

Each ADM module has eighteen 12.5 MHz 18-bit channels with preamps, designed for pixel clocks of up to 5 MHz. Up to four ADM modules can be installed in an Archon chassis. The preamps are fully differential and DC-coupled. The input range is 6V (+3V to -3V differential). Correlated double sampling (CDS) is performed digitally in the FPGA, which allows multiple reset and video levels to be averaged, driving noise down as the pixel clock is reduced. For a grounded input with 200kHz timings, noise is 0.13DN (11uV RMS). The ADM modules use Archon's normal 100MHz 16 bit data path to the CDS processor. To accomplish this, an 18 bit sample is truncated to 16 bits, and replicated 8 times. The truncated 16 bits are then dithered during those 8 sample periods so CDS averaging will recover the original 18 bit value. In addition to the normal CDS pixel data, portions of the raw ADC data can be simultaneously captured to a memory buffer. This allows a detailed, low-noise oscilloscope view of the CCD output waveform during device tuning. The raw waveforms simplify the examination of clock feedthrough, reset level stability, and optimal CDS sample points.

High Speed Clock Module

The high speed clock modules are intended to drive external high speed clock buffers. There are 12 high speed channels. Each channel has an LVDS output with 1 ns timing resolution. Each channel also has a clock magnitude programmable from 5V to 14V capable of sourcing 1A, and a clock offset programmable from -14V to + 14V that can source/sink 10 mA. Each clock magnitude and offset is monitored for voltage and current. Clock magnitudes and offsets power on and off at step 1 of the power sequence.

The high speed clock module additionally has 4 general purpose digital I/Os, which can be powered by an internal +3.3V or by an external supply from +1.65V to +5.5V. Each I/O can be configured as an input or an output. When an output, each line can be driven high, low, or by the timing core. In addition, the digital I/O lines can be controlled by a dynamically programmable embedded 100 MHz 16-bit CPU for simple interface tasks, such as communication with an RS-232 vacuum gauge or I2C temperature sensor.

LVDS Clock Module

The LVDS clock modules are intended to drive external high speed clock buffers. There are 16 LVDS channels with 10 ns timing resolution. +3.3V, +/-5V, and +/-16V supplies are also provided. The supplies power on and off at step 1 of the power sequence.

The LVDS clock module additionally has 4 general purpose digital I/Os, which can be powered by an internal supply of +3.3V or by an external supply from +1.65V to +5.5V. Each I/O can be configured as an input or an output. When an output, each line can be driven high, low, or by the timing core. In addition, the digital I/O lines can be controlled by a dynamically programmable embedded 100 MHz 16-bit CPU for simple interface tasks, such as communication with an RS-232 vacuum gauge or I2C temperature sensor.

Heater Module

Each heater module has two output channels, intended to source up to 1A each at up to 25V while driving a heater element (typically 25 ohms). There are also two temperature monitoring channels, which source precisely 10 uA each and are intended to monitor silicon diode temperature sensors via a four wire force/sense interface. A PID loop can be used to drive either heater output based on either temperature input. Heater power is taken from a dedicated line on the Archon power connector for flexibility.

The heater module additionally has 8 general purpose digital I/Os, which can be powered by an internal +3.3V or by an external supply from +1.65V to +5.5V. Each group of 2 I/Os can be configured as inputs or outputs. When an output, each line can be driven high, low, or by the timing core. In addition, the digital I/O lines can be controlled by a dynamically programmable embedded 100 MHz 16-bit CPU for simple interface tasks, such as communication with an RS-232 vacuum gauge or I2C temperature sensor.

HeaterX Module

The HeaterX module is an upgrade to the Heater module. It has the same two 25W heater outputs, but has three temperature monitoring channels with precision current sources programmable from 25 nA to 1.5 mA in 25 nA steps, compatible with silicon diodes or RTDs.

Archon Chassis

The Archon chassis is built from aluminum, with electrically conductive chem film internal surfaces for shielding, and anodized external surfaces for durability. An 80mm fan with filter on the rear face cools the internal components with ambient air, which is channeled past the modules and then exhausted through ducts on the same face as the fan intake. The front face is customized to accommodate the connectors needed for a particular CCD.

Power for the standard chassis is supplied as a set of DC voltages through a circular bayonet connector. The DC supply lines pass through a power conditioning board, which keeps the power lines disconnected from the rest of the system until all voltages are at their nominal values. The voltages necessary for a particular system vary depending on the installed modules. Jumpers on the power conditioning board allow the user to select which voltages should be monitored for a system. Linear or switching external power supplies can be used to generate the system DC voltages, depending on a particular application's need for low noise or high efficiency.

The Archon AC chassis uses a standard IEC AC power cable, and has internal DC-DC converters. The Archon AC chassis also comes with an integrated water block with 3/8" (9.5mm) copper tubing for optional liquid cooling, and the fan can be disabled in software.

Backplane X12

The Archon backplane is responsible for communication with the host system, along with communication and power distribution for the installed modules. The X12 variant can support 12 installed modules, of which 4 can be ADC modules. Processing is done by a 32-bit soft processor embedded in the backplane Kintex 7 FPGA for older systems, and by a 64-bit ARM processor for Rev H backplanes. The CPU has 2 GB of DDR3 RAM, with 512 MB reserved for the processor and three 512 MB frame buffers. 16 MB of flash memory stores firmware and controller configuration data.

Communication

All communication goes through the onboard FPGA. The embedded CPU communicates with an upstream host system through an SFP socket, which can be populated with either a fiber or copper gigabit Ethernet module. Rev H backplanes have a dedicated copper gigabit ethernet port, with an additional SFP socket which supersedes the copper interface when populated with a fiber SFP module. Note that the socket communicates only at a gigabit rate; it is not backward compatible with 10 or 100 megabit networks. The CPU listens for a TCP/IP connection on port 4242 to initiate communication. Rev F and older backplanes can only support a single connection at a time. For this reason, and to avoid network contention, the Archon controller is normally connected to a dedicated network port on the host system. Rev H backplanes currently support up to four simultaneous connections.

Hardware Triggers

The backplane has connections to two BNC connectors on the chassis, Trigger Out and Trigger In. The electronic Trigger Out interface is shown in Figure 2. The output can be optoisolated, or optionally directly driven with a 499 ohm pull-up to 5V on the positive line and the negative line tied to ground. Refer to the FOD817 datasheet for specific performance information for the optoisolator. In general, the output will sink 10 mA with a saturation voltage < 0.5V when on. The Trigger Out signal can be forced high or low, or connected to a timing core clock (the INT control signal, for exposure integration time). This is typically used to drive a shutter or light source.

Rev F and newer versions of the backplane have separate optoisolated and driven Trigger Out circuits that are selected by connecting the Trigger Out cable to the desired header, as seen in Figure 3.

The Trigger In circuit is shown in Figure 4. A 3.3V input at 2 mA is sufficient to trigger the optoisolator. Inputs up to 10V are tolerated. The Trigger In signal can be connected to the timing core reset input. This is typically used to trigger a new frame.

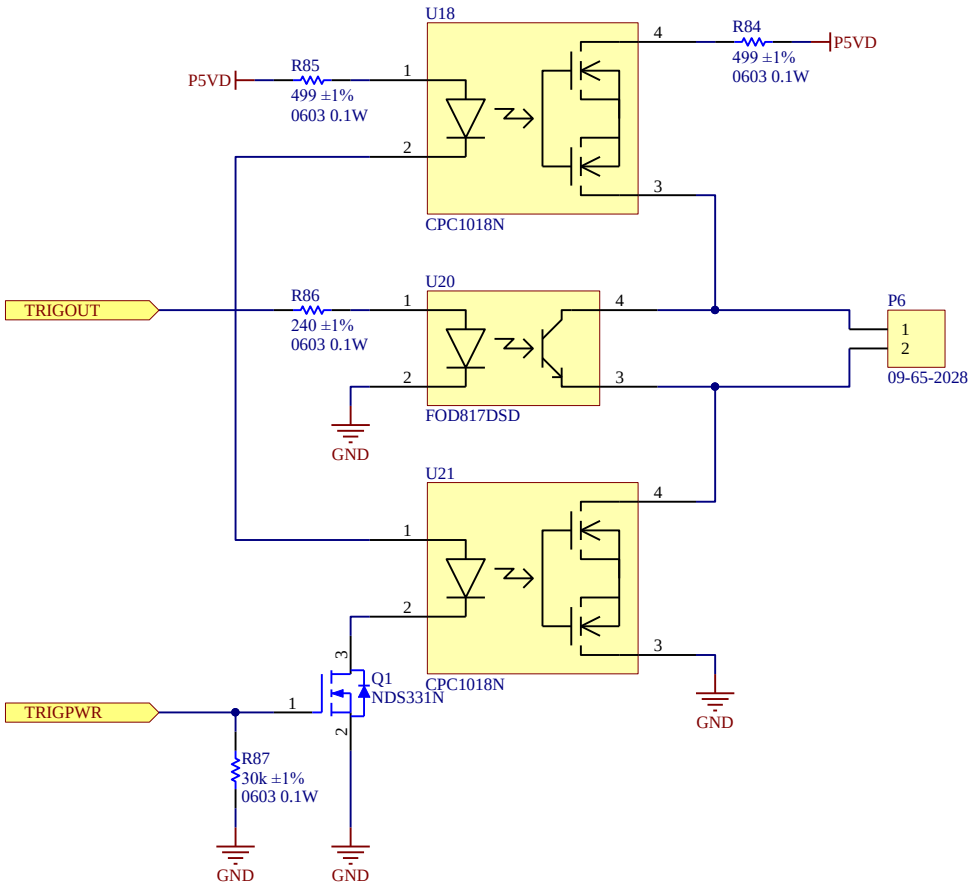


Figure 2: Trigger Out Schematic (Rev E and earlier)

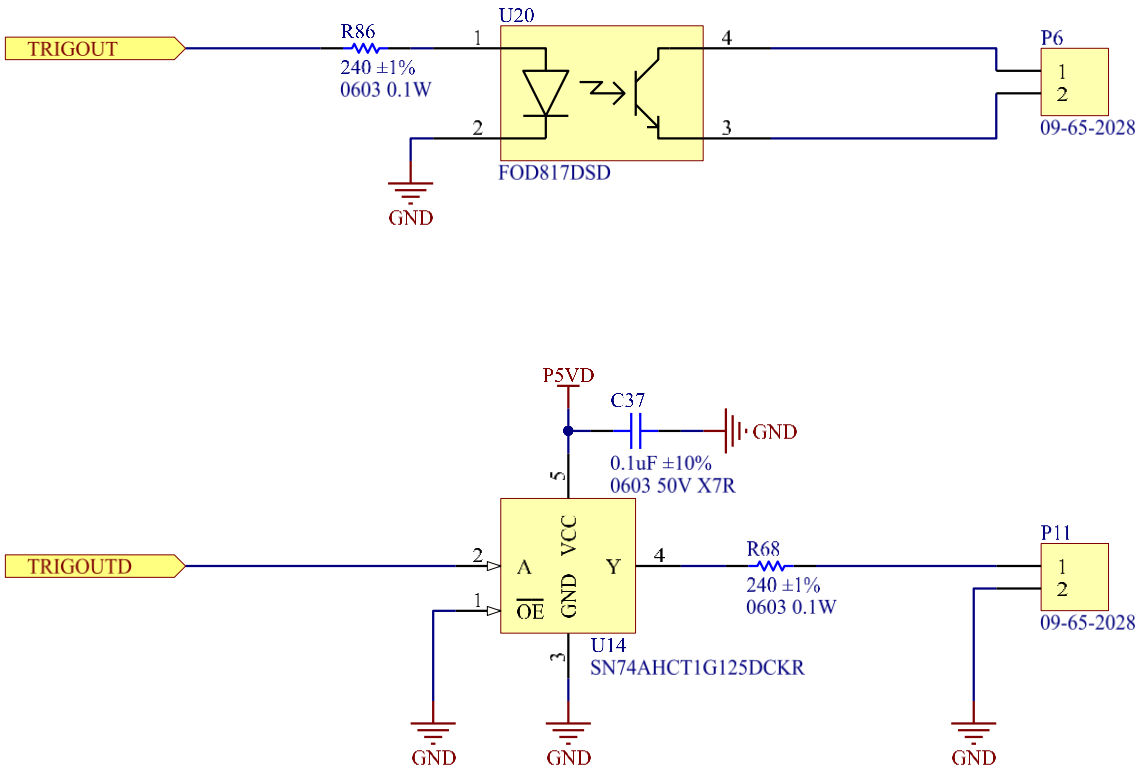


Figure 3: Trigger Out Schematic (Rev F and later)

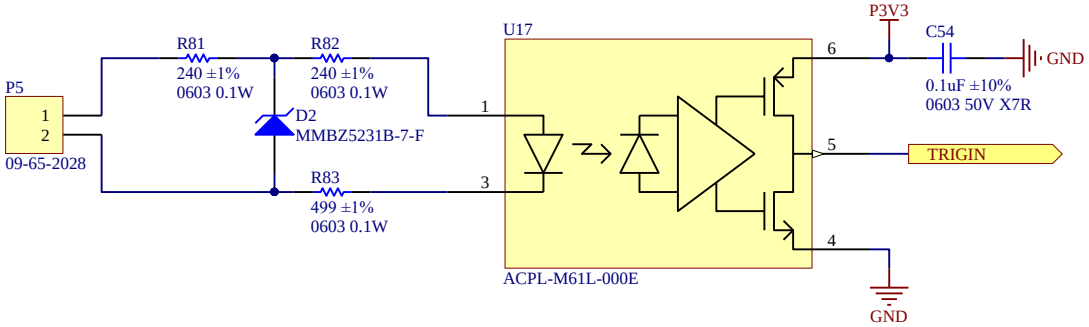


Figure 4: Trigger In Schematic

Synchronization (Backplane Rev. D and earlier)

The Sync RJ-45 jack on the backplane is provided to facilitate the synchronization of multiple Archon controllers. The external synchronization interface consists of 3 LVDS receiver pairs, with 110 Ohm internal termination. The connector pinout is shown in Figure 5. EXTCLK is routed to a PLL which generates the master 100.0 MHz clock from either an internal 25.0 MHz clock or from EXTCLK when in external master clock mode. EXTRESET and EXTLOAD are connected to the RESET and LOAD signals for all timing cores in external master clock mode.

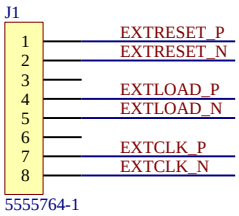


Figure 5: Sync Connector

To synchronize multiple systems, an external 25.0 MHz LVDS clock must be applied to EXTCLK. Hold EXTRESET high, apply the desired system configurations (including configuration key "EXTCLOCK = 1") to all systems, and then release EXTRESET. All controllers will then begin executing their timing scripts synchronously. To synchronously update a timing parameter, issue the "PREPPARAM" command to all systems, and then pulse the EXTLOAD signal. All controllers will remain in lockstep indefinitely under these conditions. Note that removing EXTCLK from a controller while in external master clock mode will cause the system to hang. The controller will reboot if the clock is reconnected, and need to be reconfigured.

Synchronization (Backplane Rev. E and later)

Rev. E and later Backplanes were designed to allow either daisy-chained or star topology controller synchronization. Two RJ-45 connectors (SYNCIN/J1 and SYNCOUT/J2) are provided on the backplane for synchronization of multiple Archon controllers. The synchronization interface consists of 3 LVDS pairs. The SYNCIN receivers have 110 Ohm internal termination. The connector pinouts are shown in Figure 6. EXTCLK is routed to a PLL which generates the master 100.0 MHz clock from either an internal 100.0 MHz clock or from EXTCLK when an external 100.0 MHz clock is detected. EXTRESET and EXTLOAD are connected to the RESET and LOAD signals for all timing cores when an external clock is present.

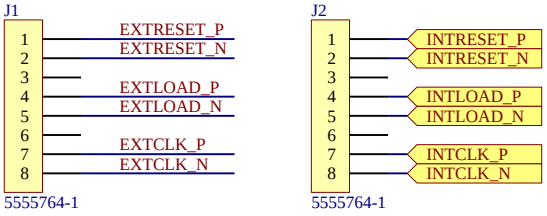


Figure 6: Sync Connector

INTCLK, INTRESET, and INTLOAD are driven by EXTCLK, EXTRESET, and EXTLOAD when an external clock is present, or by the internal 100.0 MHz clock, RESET, and LOAD signals when no external clock is detected. To synchronize multiple systems in a daisy chain, connect standard Cat-5 network cables from the SYNCOUT of each system to the SYNCIN of the next system. The first system in the chain (the master) will have no SYNCIN connection, and the last system in the chain will have no SYNCOUT connection. Keep all timing cores in reset (Give the "HOLDTIMING" command to either the master or all systems). Apply the desired system configurations to all systems. Release the timing cores from reset using "RELEASETIMING". All controllers will then begin executing their timing scripts synchronously. Note that there is a static delay between systems of about 10 ns plus cable propagation delay. To synchronously update a timing parameter, issue the "PREPPARAM" command to all systems, and then give the "LOADPARAM" command (either to the master system or all systems). All controllers will remain in lockstep indefinitely under these conditions. Note that unplugging the sync cables during operation may cause the system to hang. The controller will reboot if the clock is reconnected, and will need to be reconfigured.

If it's necessary to eliminate the static delay associated with the daisy chain topology, it is also possible to build a simple "sync box" that outputs a common 100.0 MHz clock, RESET, and LOAD signal to all controllers in a star topology.

Power

Power is supplied to the backplane through two power connectors. The pinout is shown in Figure 7. The 2.5V and 5V inputs are required to power the backplane. The other voltages need only be applied if installed modules require them. Typically, the power board (described later) monitors and gates the supply voltages and passes them to these connectors once all are at nominal levels. The backplane communicates with the power board via an IDC cable connected to P9, with the pinout shown in Figure 8.

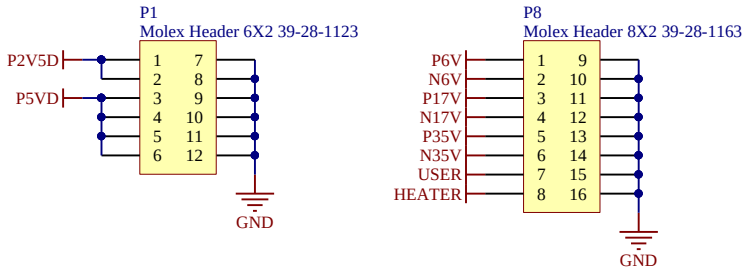


Figure 7: Power Connectors

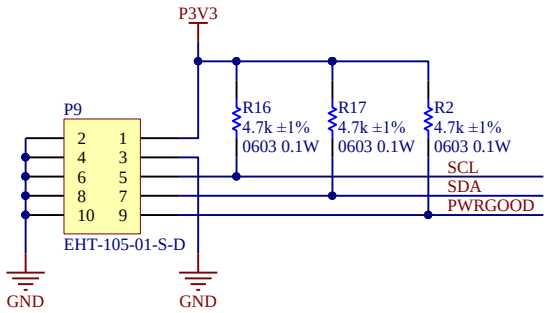


Figure 8: Power Communication

Modules

12 modules can be installed on the backplane. Most modules can be installed in any slot. The HVX Bias, LVX Bias, and HS modules are exceptions, and can only be installed in slots 3-4 or 9-12 because of their extended length. ADC modules can only be installed in the central 4 slots (5-8), which have an additional connector that carries the high speed ADC data. The module connector pinout is shown in Figure 9. A module indicates that it's installed by grounding the PRESENT signal. A module indicates that it is overheating by pulling the OVERHEAT signal low. A module is commanded to power down when the STANDBY input is low. The 100 MHz master clock is supplied to the modules on the MCLK LVDS pair. Timing core resets are commanded by RESET, and timing core parameter loads by LOAD. Serial communication between the module and the backplane is done over the SERFBP and SERTBP LVDS pairs at 6.25 Mbps. The ADC module connector is shown in Figure 10. The ADC modules are given a low jitter 100 MHz clock on the CLK pair, and transmit back a 400 MHz clock on LCLK, a framing clock on ADCLK, and sampled data bits on the OUT pairs.

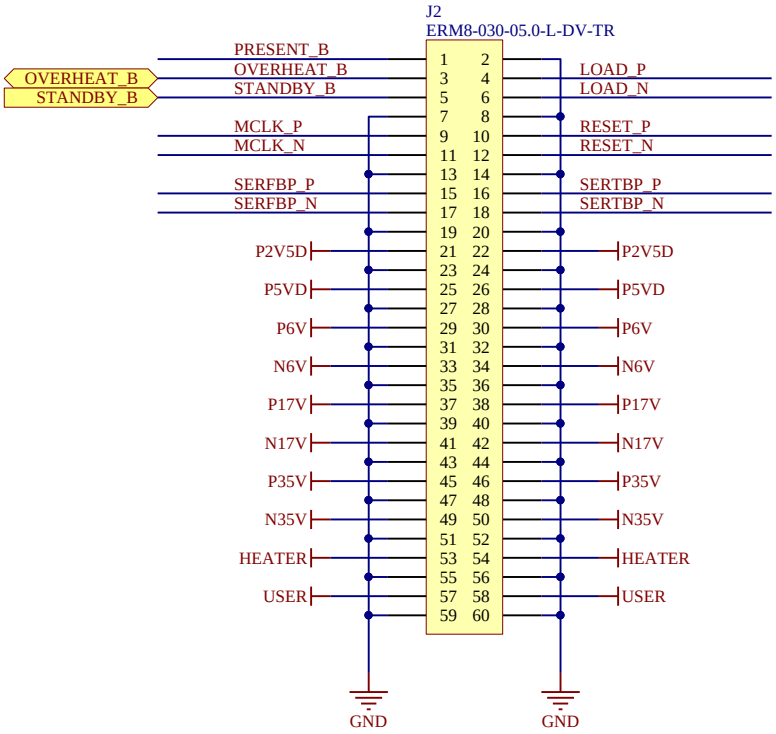


Figure 9: Module Backplane Connector

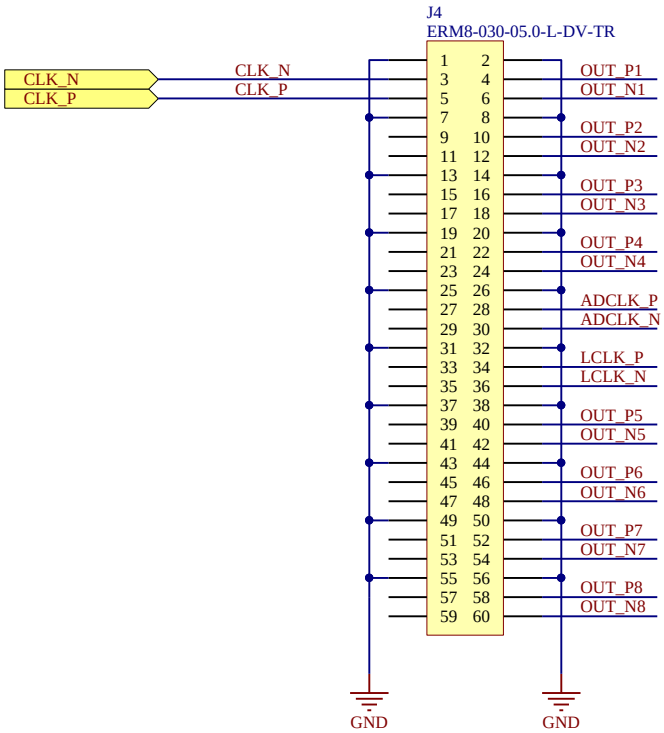


Figure 10: ADC Module Backplane Connector

ADC Module

The ADC module simultaneously samples four fully differential inputs at 16 bits and 100 MHz. The raw samples are transmitted to the backplane for digital CDS processing. The input buffer is shown in Figure 11. By default, R10 is not installed. A single-ended or differential termination resistor could be installed there, or a load resistor for a JFET buffer near the CCD output. The CLAMP_P and CLAMP_N biases are generated by DACs, and are connected through analog switches to reset the DC level of the Q1 JFETs (typically once per line). The clamp levels are normally chosen to set the DC differential input voltage near the top of the ADC input range.

The preamp stages following Q1 have two selectable gain settings, set by resistors. The default gain resistors set the full scale differential input swing to either 1.3V or 4V, but can be modified if necessary for a particular user application. Measured low gain is 65.8 uV/DN (4.31V full scale). Measured high gain is 21.9 uV/DN (1.43V full scale). There is a low-pass filter before the ADC with a time constant of 10 ns (-3dB at 15 MHz).

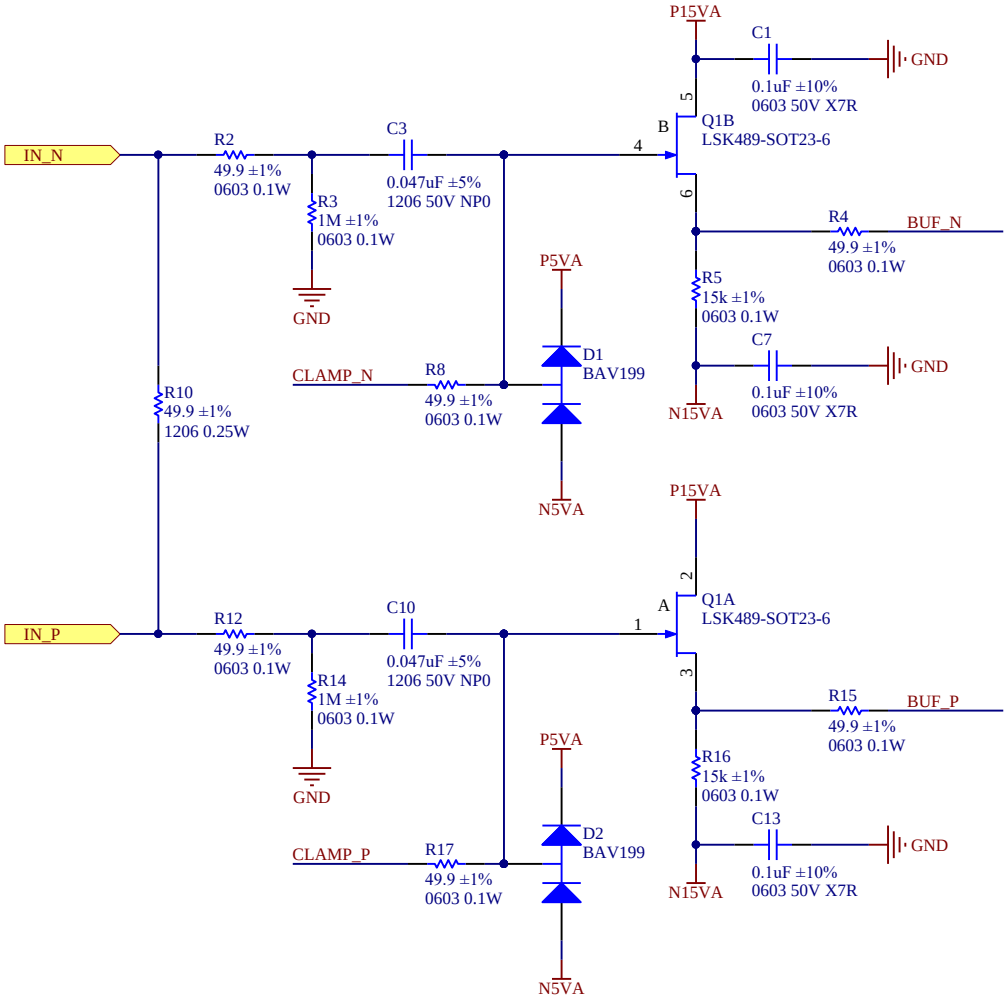


Figure 11: ADC Input Buffer

The ADC module requires the P6V, N6V, P17V and N17V system supplies. The pinout for the ADC module connector is shown in Figure 12.

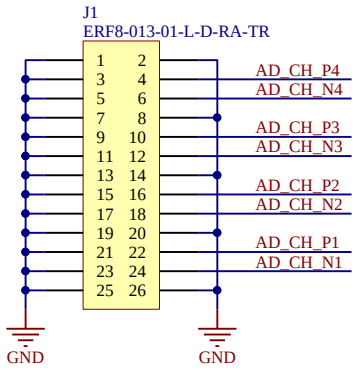


Figure 12: ADC Module Connector

ADC module performance measurements:

- Grounded input noise (high gain, 100 MHz): 2.6 DN
- Grounded input noise (low gain, 100 MHz): 2.2 DN
- Channel to channel crosstalk: -101 dB
  (worst case, measured with 45000 DN signal on one channel, other channels grounded)
- Grounded input FFT:

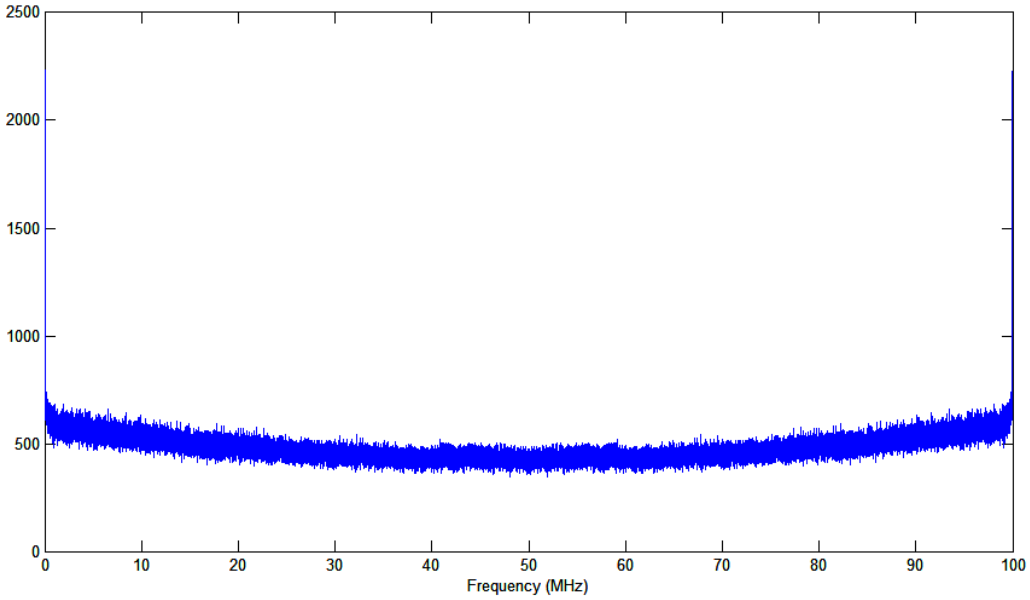


Figure 13: ADC Grounded Input FFT

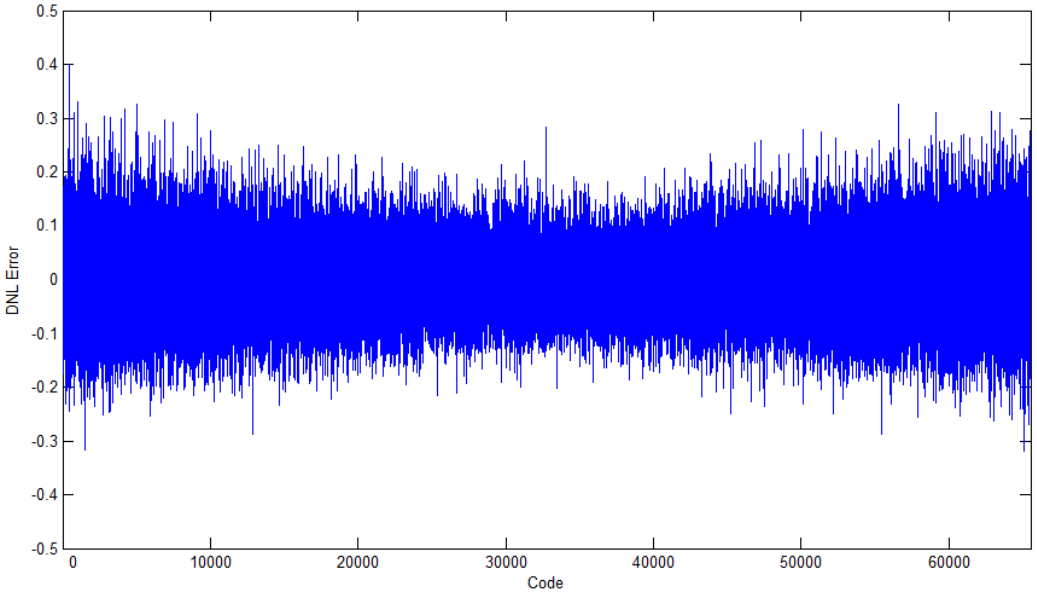


Figure 14: ADC DNL

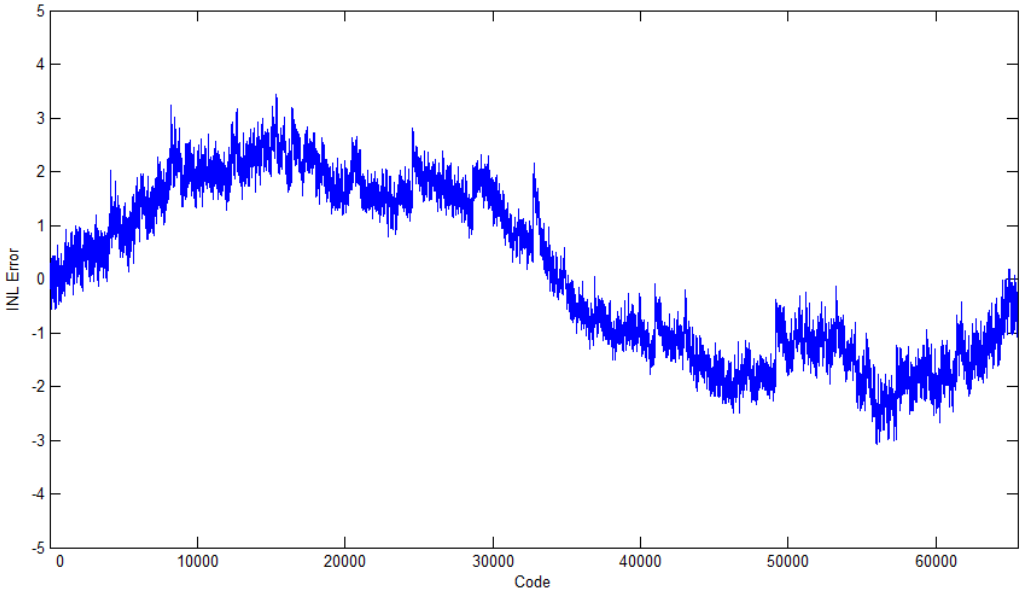


Figure 15: ADC INL

ADM Module

The ADM module simultaneously samples eighteen fully differential inputs at 18 bits and 12.5 MHz. The raw samples are transmitted to the backplane for digital CDS processing. The input buffer is shown in Figure 16. By default, R18 is not installed. A single-ended or differential termination resistor could be installed there. The full scale differential input swing is 6V, but can be modified if necessary for a particular user application. Measured gain is 91.5 uV/DN. There is a low-pass filter before the ADC with a time constant of 4.5 ns (-3dB at 35 MHz).

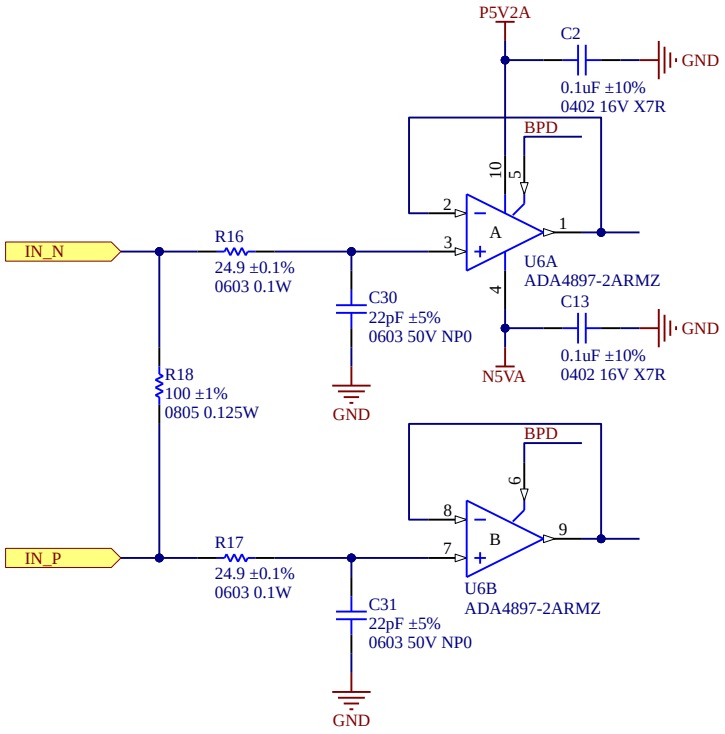


Figure 16: ADM Input Buffer

The ADM module requires the P6V and N6V system supplies. The pinout for the ADM module connector is shown in Figure 17.

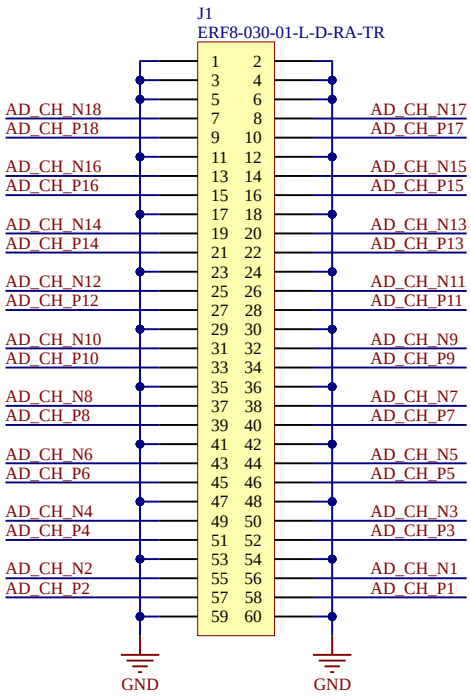


Figure 17: ADM Module Connector

Driver Module

Each clock driver module generates 8 CCD clocks using 14-bit 100 MHz DACs, with outputs ranging from -13.000 V to +13.000 V. Clock levels can be set with a resolution of about 2 mV. The timing core in the clock driver FPGA generates a new target clock level every 10 ns. The DAC outputs linearly slew toward the new target at a software selectable fast or slow rate, yielding clean trapezoidal CCD waveforms.

The output amplifier for each clock is a THS3095, rated for 250 mA. There is a series 49.9 Ohm resistor after each THS3095, which is usually the limiting factor for slew rate with heavy capacitive loads. There is also an optoisolator between each clock and the output connector. Initially, the optoisolators are open, and the pin on the output is connected to ground through a 100k resistor. When the system is configured, each clock is calibrated. When the system is commanded to enable power to the CCD, all clocks go to their initial levels, and the optoisolators are closed after the last step of the power sequence.

The system supplies are continuously monitored in hardware by comparators. In the event a system supply leaves its normal operating range, all optoisolators to the CCD are opened.

The driver module requires the P6V, N6V, P17V and N17V system supplies. The driver module pinout is shown in Figure 18.

For firmware > 1090, the clock pattern source for each clock channel can be chosen from the 8 timing core channel outputs. This means that if a CCD clock S1 was on channel 1, and S2 was on channel 2, serial clocking could be reversed by assigning channel 1 to timing core output 2 and vice -versa, with no other timing modifications necessary.

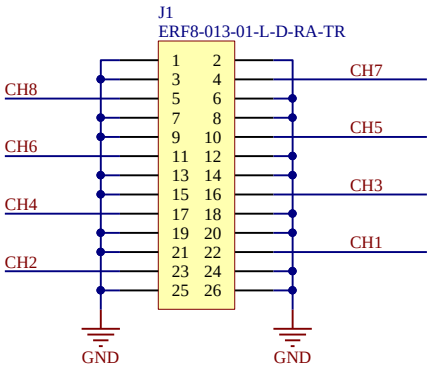


Figure 18: Clock Driver Module Connector

DriverX Module

Each DriverX module generates 12 CCD clocks using 14-bit 100 MHz DACs, with outputs ranging from -13.000 V to +13.000 V. Clock levels can be set with a resolution of about 2 mV. The timing core in the clock driver FPGA generates a new target clock level every 10 ns. The DAC outputs linearly slew toward the new target at a software selectable fast or slow rate, yielding clean trapezoidal CCD waveforms.

The output amplifier for each clock is a THS3095, rated for 250 mA. There is a series 49.9 Ohm resistor after each THS3095, which is usually the limiting factor for slew rate with heavy capacitive loads. There is also an optoisolator between each clock and the output connector. Initially, the optoisolators are open, and the pin on the output is connected to ground through a 100k resistor. When the system is configured, each clock is calibrated. When the system is commanded to enable power to the CCD, all clocks go to their initial levels, and the optoisolators are closed after the last step of the power sequence.

The system supplies are continuously monitored in hardware by comparators. In the event a system supply leaves its normal operating range, all optoisolators to the CCD are opened.

The driver module requires the P6V, N6V, P17V and N17V system supplies. The driver module pinout is shown in Figure 19.

The clock pattern source for each clock channel can be chosen from the 12 timing core channel outputs. This means that if a CCD clock S1 was on channel 1, and S2 was on channel 2, serial clocking could be reversed by assigning channel 1 to timing core output 2 and vice -versa, with no other timing modifications necessary.

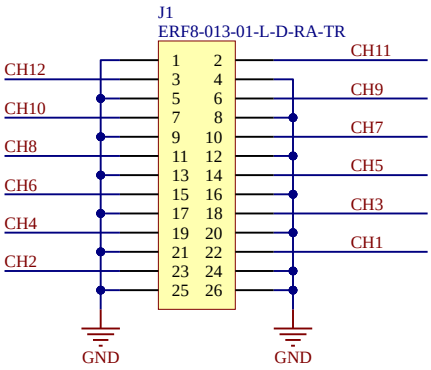


Figure 19: DriverX Module Connector

LVBias/LVXBias Module

The low voltage bias module generates DC biases from -14.000V to +14.000V, with a resolution of about 1 mV. There are 24 low current biases (LV1-LV24, 10 mA max) and 6 high current biases (LV25-LV30, 500 mA max). The high current biases also have a programmable current limit (resolution of about 1 mA, minimum of 4 mA). Total current for the module can't exceed 1A. The voltage and current of each bias is monitored and reported. The capacitive load for each bias should be kept below 1 uF to guarantee amplifier stability.

There are optoisolators between each bias and the output connector. Initially, the optoisolators are open, and the pin on the output is connected to ground through a 100k resistor. When the system is commanded to enable power to the CCD, all biases are set to 0 V. The biases are checked, and then the optoisolators are closed. Next, each bias is set to its operating level in a user programmable sequence. At each step, the bias levels are checked before proceeding to the next step. The power down sequence proceeds in the reverse order.

The system supplies are continuously monitored in hardware by comparators. In the event a system supply leaves its normal operating range, all optoisolators to the CCD are opened.

The module also has 8 general purpose digital I/O lines, along with a digital power line. Each pair of I/O lines can be configured as inputs or outputs. The digital power line (DPWR) can either be driven externally by a 1.65 V to 5.5 V supply, or connected to an internal 3.3 V supply. Current draw from the internal power supply by external devices should be limited to 100 mA. By default, the DPWR line is set to be driven externally, and all I/O lines are configured as inputs. A user-programmable 16-bit 100 MHz CPU is available for simple digital I/O communication tasks. See the VCPU section for details.

The low voltage bias module requires the P17V and N17V system supplies. The low voltage bias module pinout is shown in Figure 20.

The LVXBias module has functionality and pinouts identical to the LVBias module, but is a longer card that only fits slots 3-4/9-12 and has additional buffering that accelerates the bias monitoring functions.

For firmware versions after 833, commands to change a bias can be given by the timing core. Bias changes commanded this way are ignored until the system has been configured and all biases have powered on. One bias command can be given per master clock cycle. It takes about 10 ms for a channel to settle to a new value. If too many bias commands are given too quickly, some will be ignored. Biases commanded this way are not as accurate as no corrections are performed. The output bias value will typically be within +/- 50 mV of the commanded value.

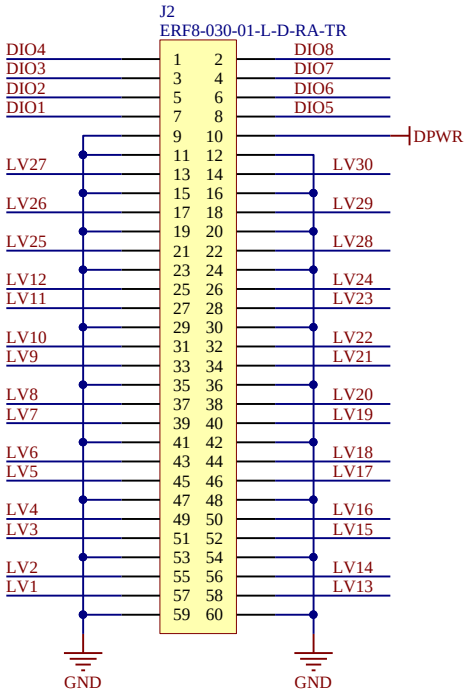


Figure 20: LVBias Module Connector

HVBias/HVXBias Module

The high voltage bias module generates DC biases from 0.000V to +31.000V, with a resolution of about 1 mV. There are 24 low current biases (HV1-HV24, 10 mA max) and 6 high current biases (HV25-HV30, 250 mA max). The high current biases also have a programmable current limit (resolution of about 1 mA, minimum of 4 mA). Total current for the module can't exceed 1A. The voltage and current of each bias is monitored and reported. The capacitive load for each bias should be kept below 1 uF to guarantee amplifier stability.

There are optoisolators between each bias and the output connector. Initially, the optoisolators are open, and the pin on the output is connected to ground through a 100k resistor. When the system is commanded to enable power to the CCD, all biases are set to 0 V. The biases are checked, and then the optoisolators are closed. Next, each bias is set to its operating level in a user programmable sequence. At each step, the bias levels are checked before proceeding to the next step. The power down sequence proceeds in the reverse order.

The system supplies are continuously monitored in hardware by comparators. In the event a system supply leaves its normal operating range, all optoisolators to the CCD are opened.

The high voltage bias module requires the N6V, P17V, N17V and P35V system supplies. The high voltage bias module pinout is shown in Figure 21.

The HVXBias module has functionality and pinouts identical to the HVBias module, but is a longer card that only fits slots 3-4/9-12 and has additional buffering that accelerates the bias monitoring functions.

For firmware versions after 833, commands to change a bias can be given by the timing core. Bias changes commanded this way are ignored until the system has been configured and all biases have powered on. One bias command can be given per master clock cycle. It takes about 10 ms for a channel to settle to a new value. If too many bias commands are given too quickly, some will be ignored. Biases commanded this way are not as accurate as no corrections are performed. The output bias value will typically be within +/- 50 mV of the commanded value.

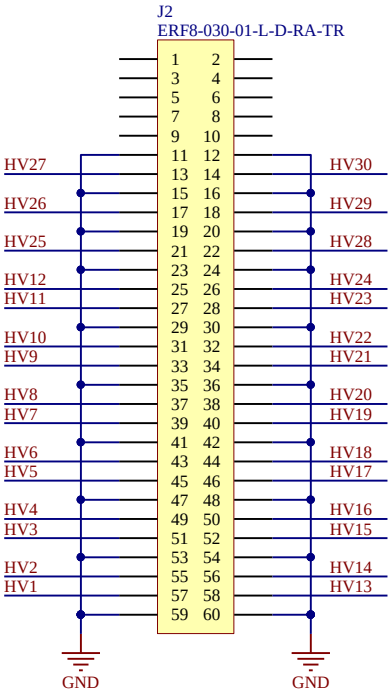


Figure 21: HVBias Module Connector

XVBias Module

The XV bias module generates four DC biases from 0.000V to +91.000V and four DC biases from -95.000V to 0.000V with a resolution of about 2 mV. Each can supply 50 mA. The voltage and current of each bias is monitored and reported. The capacitive load for each bias should be kept below 0.1 uF to guarantee amplifier stability.

There are optoisolators between each bias and the output connector. Initially, the optoisolators are open, and the pin on the output is connected to ground through a 1M resistor. When the system is commanded to enable power to the CCD, all biases are set to 0 V. The biases are checked, and then the optoisolators are closed. Next, each bias is set to its operating level in a user programmable sequence. At each step, the bias levels are checked before proceeding to the next step. The power down sequence proceeds in the reverse order.

The system supplies are continuously monitored in hardware by comparators. In the event a system supply leaves its normal operating range, all optoisolators to the CCD are opened.

The XV bias module requires the P17V, N17V, P100V and N100V system supplies. This requires the use of an XV power supply and XV chassis. The XV bias module pinout is shown in Figure 22.

For firmware versions after 1090, commands to change a bias can be given by the timing core. Bias changes commanded this way are ignored until the system has been configured and all biases have powered on. One bias command can be given per master clock cycle to a P and/or N channel. It takes about 10 ms for a channel to settle to a new value. If too many bias commands are given too quickly, some will be ignored. Biases commanded this way are not as accurate as no corrections are performed. The output bias value will typically be within +/- 50 mV of the commanded value.

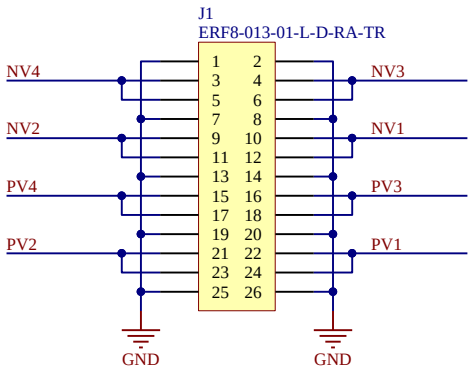


Figure 22: XVBias Module Connector

HS (High Speed) Module

The high speed module outputs 12 high speed LVDS clocks with 1 ns timing resolution. The module also provides clock magnitudes adjustable from 5V to 14V with about 1 mV of resolution and capable of sourcing 1A. There are also clock offset voltages adjustable from -14 to +14V with about 1 mV of resolution and capable of sourcing/sinking 10 mA. The currents and voltages of each magnitude and offset are monitored. Together, each channel is intended to drive an LVDS receiver followed by a high speed clock buffer (such as an EL7457) that's AC-coupled to a CCD clock line. The clock buffer is powered by the channel magnitude. The CCD clock line is tied to the channel offset through a parallel diode and resistor. The diode direction is chosen based on whether the clock is normally high or low. The magnitude and offset are enabled and disabled at step 1 of the power sequence.

The module also has 4 general purpose digital I/O lines, along with a digital power line. Each I/O line can be configured as an input or output. The digital power line (DPWR) can either be driven externally by a 1.65 V to 5.5 V supply, or connected to an internal 3.3 V supply. Current draw from the internal power supply by external devices should be limited to 100 mA. By default, the DPWR line is set to be driven externally, and all I/O lines are configured as inputs. A user-programmable 16-bit 100 MHz CPU is available for simple digital I/O communication tasks. See the VCPU section for details.

The high speed module requires the P17V and N17V system supplies. The high speed module pinout is shown in Figure 23.

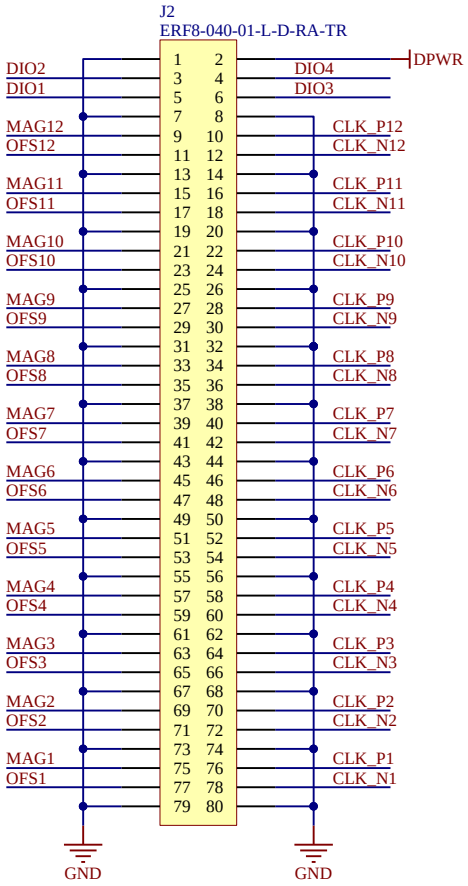


Figure 23: High Speed Module Connector

LVDS Module

The LVDS module outputs 16 LVDS clocks with 10 ns timing resolution. The module also provides +3.3V, +/-5V, and +/-16V supplies (maximum current 1A each). The supplies are enabled and disabled at step 1 of the power sequence.

The module also has 4 general purpose digital I/O lines, along with a digital power line. Each I/O line can be configured as an input or output. The digital power line (DPWR) can either be driven externally by a 1.65 V to 5.5 V supply, or connected to an internal 3.3 V supply. Current draw from the internal power supply by external devices should be limited to 100 mA. By default, the DPWR line is set to be driven externally, and all I/O lines are configured as inputs. A user-programmable 16-bit 100 MHz CPU is available for simple digital I/O communication tasks. See the VCPU section for details.

The LVDS module requires the P6V, N6V, P17V and N17V system supplies. The LVDS module pinout is shown in Figure 24.

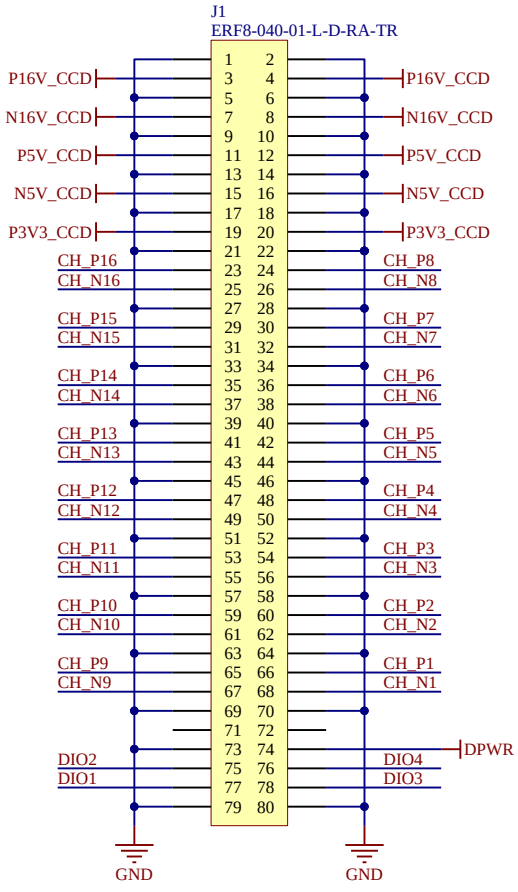


Figure 24 : LVDS Module Connector

Heater Module

The heater module can drive two resistive heater elements (typically up to 25V at 1A each), and reads two temperature sensors. Power for the heater elements is drawn from the HEATER system supply, which should be a minimum of 2V higher than the maximum desired output voltage, but no greater than 36V.

The temperature sensing circuitry consists of two precision 10 uA current sources and two ADCs with high impedance instrumentation amplifier inputs. These are intended to be used with standard temperature sensing diodes such as the Lakeshore DT-670 with a four wire interface. The force lines (TEMPxF_P and TEMPxF_N) and sense lines (TEMPxS_P and TEMPxS_N) should be connected to the temperature sensor with shielded twisted pair. The positive lines (x_P) should be connected to the temperature sensor anode, and the negative lines (x_N) to the cathode. Interfacing with RTDs or other sensor types can be accommodated by customizing the gain and current source resistors on the heater module, and updating the firmware with the appropriate temperature curves. Each heater output can be forced to a constant voltage, or a target temperature can be set with a PID loop controlling the heater output. The heater output can be limited to a specified voltage to limit the output power. Upper and lower limits for valid sensor readings can be set. The heater output is disabled if these limits are exceeded.

Each heater PID loop is configured by setting P, I, D, and I Limit terms, and specifying which temperature sensor to close the loop around. A target temperature is given. A PID loop update time is also defined. Internally, the PID loop calculates an error signal by subtracting the current temperature from the target temperature. Once per update loop, the P term is multiplied by the current error, the I term is multiplied by a running sum of the errors (limited to I Limit), and the D term is multiplied by the difference between this error and the last. The sum of these results is then translated into a linear power output to the heater. The loop update time should be set to a timescale comparable to how long it takes the system to show a response to an input. If ramping is enabled for a heater, the PID loop target begins at the current temperature, and linearly ramps to the final target temperature at the configured rate.

The module also has 8 general purpose digital I/O lines, along with a digital power line. Each pair of I/O lines can be configured as inputs or outputs. The digital power line (DPWR) can either be driven externally by a 1.65 V to 5.5 V supply, or connected to an internal 3.3 V supply. Current draw from the internal power supply by external devices should be limited to 100 mA. By default, the DPWR line is set to be driven externally, and all I/O lines are configured as inputs. A user-programmable 16-bit 100 MHz CPU is available for simple digital I/O communication tasks. See the VCPU section for details.

The heater module requires the P17V and HEATER system supplies. The heater module pinout is shown in Figure 25.

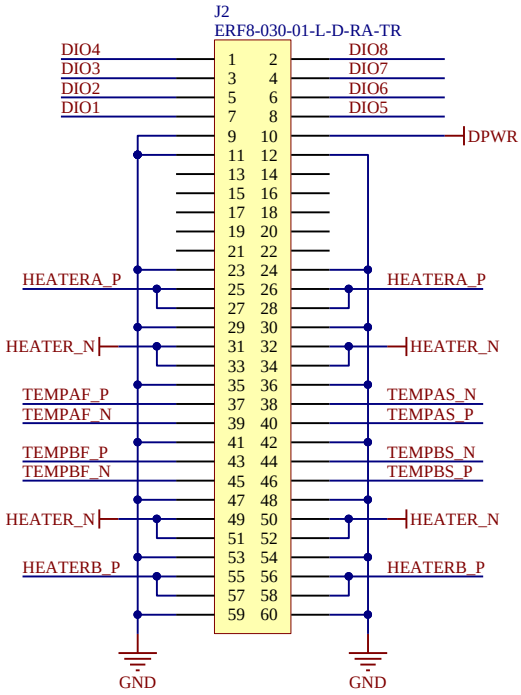


Figure 25: Heater Module Connector

HeaterX Module

The HeaterX module (an upgrade of the standard Heater module) can drive two resistive heater elements (typically up to 25V at 1A each), and reads three temperature sensors. Power for the heater elements is drawn from the HEATER system supply, which should be a minimum of 2V higher than the maximum desired output voltage, but no greater than 36V.

The temperature sensing circuitry consists of three precision current sources programmable from 25 nA to 1.5 mA in 25 nA steps, and three ADCs with high impedance inputs. They are intended to be used with standard temperature sensing diodes such as the Lakeshore DT-670 with a four wire interface, or with standard 100 or 1000 Ohm RTDs. The force lines (TEMPxF_P and TEMPxF_N) and sense lines (TEMPxS_P and TEMPxS_N) should be connected to the temperature sensor with shielded twisted pair. The positive lines (x_P) should be connected to the temperature sensor anode, and the negative lines (x_N) to the cathode. Interfacing with other sensor types can be accommodated by updating the firmware with the appropriate temperature curves. Each heater output can be forced to a constant voltage, or a target temperature can be set with a PID loop controlling the heater output. The heater output can be limited to a specified voltage to limit the output power. Upper and lower limits for valid sensor readings can be set. The heater output is disabled if these limits are exceeded.

Each heater PID loop is configured by setting P, I, D, and I Limit terms, and specifying which temperature sensor to close the loop around. A target temperature is given. A PID loop update time is also defined. Internally, the PID loop calculates an error signal by subtracting the current temperature from the target temperature. Once per update loop, the P term is multiplied by the current error, the I term is multiplied by a running sum of the errors (limited to I Limit), and the D term is multiplied by the difference between this error and the last. The sum of these results is then translated into a linear power output to the heater. The loop update time should be set to a timescale comparable to how long it takes the system to show a response to an input. If ramping is enabled for a heater, the PID loop target begins at the current temperature, and linearly ramps to the final target temperature at the configured rate.

The module also has 8 general purpose digital I/O lines, along with a digital power line. Each pair of I/O lines can be configured as inputs or outputs. The digital power line (DPWR) can either be driven externally by a 1.65 V to 5.5 V supply, or connected to an internal 3.3 V supply. Current draw from the internal power supply by external devices should be limited to 100 mA. By default, the DPWR line is set to be driven externally, and all I/O lines are configured as inputs. A user-programmable 16-bit 100 MHz CPU is available for simple digital I/O communication tasks. See the VCPU section for details.

The HeaterX module requires the P17V, N6V and HEATER system supplies. The heater module pinout is shown in Figure 26.

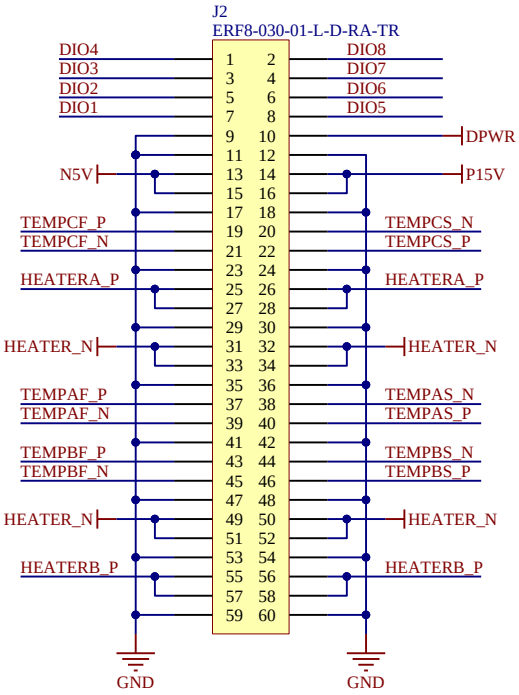


Figure 26: HeaterX Module Connector

System Power

Power is supplied to a standard Archon system through a Souriau 28-pin UT002028PH circular connector (Figure 27).

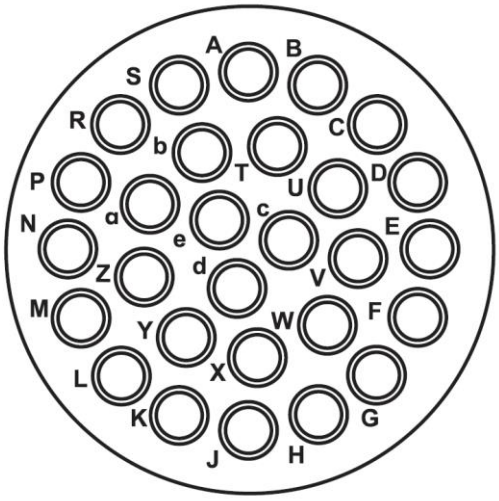


Figure 27: Power Connector

Pin	Voltage	Power Good Range
A, C, T, V	+2.5 V Digital	+2.1 V … +2.9 V
B, D, U, W	+2.5 V Digital Return	
E, G	+5 V Digital	+4.4 V … +5.6 V
F, H	+5 V Digital Return	
J	+6 V Analog	+5.5 V … +6.6 V
K	+6 V / -6 V Analog Return	
L	-6 V Analog	-5.3 V … -6.6 V
M	+17 V Analog	+16.4 V … +17.5 V
N	+17 V / -17 V Analog Return	
P	-17 V Analog	-16.6 V … -17.7 V
R	+35 V Analog	+34.3 V … +36.0 V
b	+35 V / -35 V Analog Return	
S	-35 V Analog	-33.8 V … -35.9 V
X	Heater	+18.0 V … +36.0 V
Y	Heater Return	
Z	User	+18.0 V … +36.0 V
a	User Return	
c	Fan (+12 V)	Directly connected to fan
d	Fan Return	
e	Earth ground	

The XV power supply and XV chassis use a modified pinout, shown below.

Pin	Voltage	Power Good Range
A, C, T, V	+2.5 V Digital	+2.1 V … +2.9 V
B, D, U, W	+2.5 V Digital Return	
E, G	+5 V Digital	+4.4 V … +5.6 V
F, H	+5 V Digital Return	
J	+6 V Analog	+5.5 V … +6.6 V
K	+6 V / -6 V Analog Return	
L	-6 V Analog	-5.3 V … -6.6 V
M	+17 V Analog	+16.4 V … +17.5 V
N	+17 V / -17 V Analog Return	
P	-17 V Analog	-16.6 V … -17.7 V
R	+35 V Analog	+34.3 V … +36.0 V
b	+35 V / -35 V Analog Return	
S	-100 V Analog	-97 V … -103 V
X	Heater	+18.0 V … +36.0 V
Y	Heater Return	
Z	+100 V Analog	+97 V … +103 V
a	+100 V / -100V Analog Return	
c	Fan (+12 V)	Directly connected to fan
d	Fan Return	
e	Earth ground	

The Archon AC chassis only requires a standard IEC AC power cable, and 110/220V AC power.

Only the voltages used by the installed modules need be supplied. The backplane always requires the +2.5 V and +5 V digital supplies. Power from the circular connector is routed to a power board. This board monitors the system supply voltages. When all supply voltages are at their nominal levels, the power is connected to the rest of the system through solid state relays. The allowed nominal levels are set by resistors on the power board, and the default power good ranges are shown in the power cable pinout. Switches must be set on P1 and P7 on the power board to indicate which supply voltages are being used in the system. The power good connector pinout is shown in Figure 28. Set the respective switches to ON to indicate a supply is in use and should be monitored.

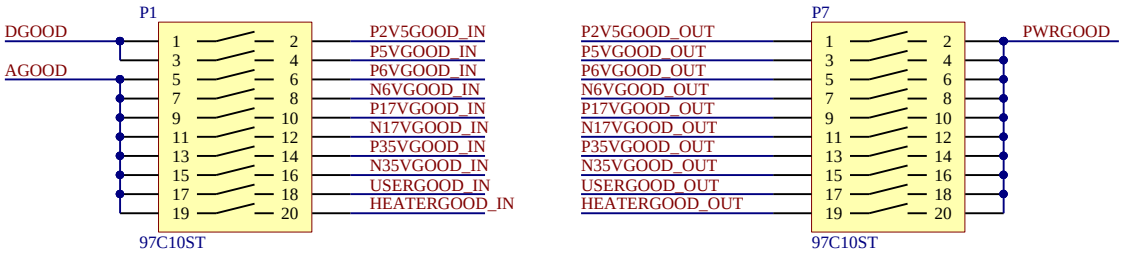


Figure 28: Power Monitor Switches

The XV chassis has a modified set of switches, as shown in Figure 29.

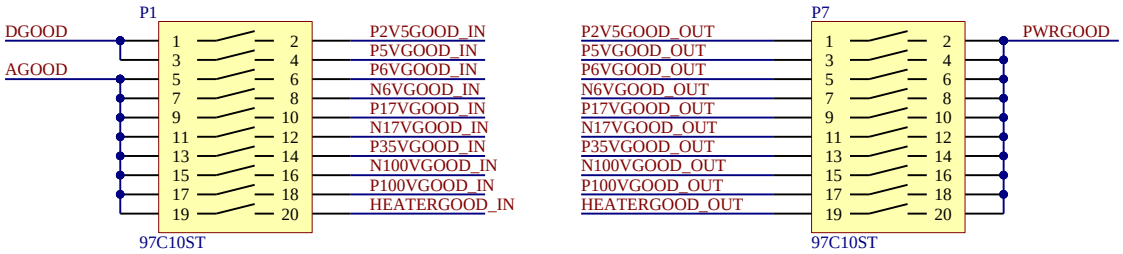


Figure 29: XV Power Monitor Switches

In addition, all of the system supply voltages and currents are monitored and digitized. These values are retrieved by the backplane over a cable connected to P9. This connector also carries the system-wide Power Good signal.

The standard Archon power supply is shown in Figure 30. It uses low noise switching supplies to generate +2.5V, +5V, +6V, -6V, +17V, -17V, and +35V voltages, along with +12V fan and +28V heater voltages. It accepts 100-240VAC at 50/60Hz, and is typically 61% efficient. Its dimensions are 8” x 6” x 4.5” (20.32cm x 15.24cm x 11.43cm), and it weighs 4.5 pounds (2.1 kg). The XV power supply has the same dimensions, but adds +100V and -100V voltages. The Archon AC chassis needs no external power supply, and can optionally have +/-100V supplies installed.

Figure 30: Archon Power Supply

Power Consumption

The following table lists the approximate current required from the system supplies for various configurations. These currents do not include any loads external to the controller (current supplied to CCD output FETs, current required to clock capacitve loads, heater current, etc). Enabled biases are set to 50% of their maximum values, and clocks are set to 0V. The typical systems contain 1 LVBias module, 1 HVBias module, 3 Driver modules, 1 Heater module, and either 1 or 4 AD modules.

Configuration	+2.5V (A)	+5V (A)	+/-6V (A)	+/-17V (A)	+35V (A)
Backplane alone	2.2	0.8			
LVBias, no HC enabled	2.2	1		0.05	
LVBias, all HC enabled	2.2	1		0.1	
HVBias, no HC enabled	2.2	1	0.02	0.03	0.04
HVBias, all HC enabled	2.2	1	0.06	0.03	0.09
Driver, no channels	2.2	1			
Driver, all channels	2.2	1.1	0.06	0.09	
AD, no channels	2.2	1.1	0.03		
AD, all channels	3	1.1	0.12		
Heater	2.2	1			
Typical 4 channel	3.5	2.6	0.33	0.36	0.09
Typical 16 channel	6.3	3.4	0.7	0.39	0.09

The total typical DC power required for the 4 channel system is 41 W. The 16 channel system requires 58 W. The AC power consumption at the wall is the listed power divided by the efficiency of the Archon power supply (~61%). A typical 16 channel Archon AC chassis draws 89W at the wall in full operation.

Communication

All communication between the host system and the Archon controller is over the gigabit Ethernet interface. By default, the controller listens for a TCP/IP connection at address 10.0.0.2 on port 4242 (the host is usually set to 10.0.0.1). To change the IP address, connect to Archon using the GUI. Set the new, desired IP address at the bottom left of the System tab and click "Apply Network Configuration" (there will be error messages because Archon is no longer responding on the old address). Click Disconnect, change the Archon IP address at the top left to the new address, and click Connect. Select System->Flash Active Config to make this change permanent.

The controller only responds to commands, it never initiates a message. Commands to the controller are of the form:

```
>xxCOMMAND
```

The command begins with a greater-than symbol, followed by a two hexadecimal digit reference number, followed by the command itself, and terminated with a newline (\n / ASCII 10 / 0x0A).

Unrecognized commands are ignored. In the event of an error parsing or executing a command, the response from the controller will be:

```
?xx
```

The initial '?' indicates an error occurred processing the command. The two hexadecimal digit reference number matching what was sent with the command follows. The response is terminated with a newline.

On success, the controller will respond:

```
<xxRESPONSE
```

The initial '<' indicates success. The two hexadecimal digit reference number matching what was sent with the command follows. The text of the response (if any) is next, and the response is terminated with a newline.

To conserve bandwidth, certain commands request the return of raw binary data (such as fetching the contents of a frame buffer). In these cases, the response takes the following form:

```
<xx:bbbbb…bbbbb
```

The initial '<' indicates success. The two hexadecimal digit reference number matching what was sent with the command follows. The ':' indicates a binary response. The remainder of the response is 1024 bytes of raw binary data, with no terminating newline.

The list of controller commands follows.

SYSTEM

Reports the system configuration (installed modules, firmware versions, etc). The response is a sequence of KEY=VALUE pairs separated by spaces and terminated with a newline. The list of keys returned follows.

```
BACKPLANE_TYPE=n        ; n = 1 for an X12 backplane, n = 2 for X16
BACKPLANE_REV=n         ; Backplane PCB revision, 0 = A, 1 = B…
BACKPLANE_VERSION=n.n.n ; Backplane firmware, major.minor.build
BACKPLANE_ID=x          ; 16 hexadecimal digit backplane unique ID
MOD_PRESENT=x           ; Hexadecimal bit field: a 1 in the LSB
                        ; indicates a module is present in slot 1
MODn_TYPE=n             ; Reports module type for slots 1…n.
                        ; 0: None
                        ; 1: Driver
                        ; 2: AD
                        ; 3: LVBias
                        ; 4: HVBias
                        ; 5: Heater
                        ; 7: HS
                        ; 8: HVXBias
                        ; 9: LVXBias
                        ; 10: LVDS
                        ; 11: HeaterX
                        ; 12: XVBias
                        ; 13: ADF
                        ; 14: ADX
                        ; 15: ADLN
                        ; 16+: Unknown
MODn_REV=n              ; Module m PCB revision, 0 = A, 1 = B…
MODn_VERSION=n.n.n      ; Module m firmware, major.minor.build
MODn_ID=x               ; 16 hexadecimal digit module m unique ID
```

STATUS

Reports the system status. The response is a sequence of KEY=VALUE pairs separated by spaces and terminated with a newline. The list of keys returned follows.

```
VALID=n                 ; n = 1 if remaining status fields are valid
COUNT=n                 ; Number of times system status has been
                        ; updated
LOG=n                   ; Number of log entries available
POWER=n                 ; Power status.  Possible values:
                        ; 0: Unknown – usually an internal error
                        ; 1: Not Configured – no configuration applied
                        ; 2: Off – power to the CCD is off
                        ; 3: Intermediate – some modules have enabled
                        ;    power to the CCD, some have not
                        ; 4: On – Power to the CCD is on
                        ; 5: Standby – System is in standby
POWERGOOD=n             ; n = 1 when system power supply is good
OVERHEAT=n              ; n = 1 when system is overheating
BACKPLANE_TEMP=f        ; Floating point backplane temperature in C
P2V5_V=f                ; +2.5V system supply voltage in V
P2V5_I=f                ; +2.5V system supply current in A
P5V_V=f                 ; +5V system supply voltage in V
P5V_I=f                 ; +5V system supply current in A
P6V_V=f                 ; +6V system supply voltage in V
P6V_I=f                 ; +6V system supply current in A
N6V_V=f                 ; -6V system supply voltage in V
N6V_I=f                 ; -6V system supply current in A
P17V_V=f                ; +17V system supply voltage in V
P17V_I=f                ; +17V system supply current in A
N17V_V=f                ; -17V system supply voltage in V
N17V_I=f                ; -17V system supply current in A
P35V_V=f                ; +35V system supply voltage in V
P35V_I=f                ; +35V system supply current in A
N35V_V=f                ; -35V system supply voltage in V
N35V_I=f                ; -35V system supply current in A
P100V_V=f               ; +100V system supply voltage in V
P100V_I=f               ; +100V system supply current in A
N100V_V=f               ; -100V system supply voltage in V
N100V_I=f               ; -100V system supply current in A
USER_V=f                ; User system supply voltage in V
USER_I=f                ; User system supply current in A
HEATER_V=f              ; Heater system supply voltage in V
HEATER_I=f              ; Heater system supply current in A
FANTACH=n               ; Fan speed in RPM (Rev F only)
```

```
MODm/TEMP=f             ; Floating point module m temperature in C
MODm/LVLC_Vn=f          ; LV(X)Bias only: Floating point module m low
                        ; voltage low current n voltage reading in V
                        ; n = 1 to 24 maps to LV1 to LV24
MODm/LVLC_In=f          ; LV(X)Bias only: Floating point module m low
                        ; voltage low current n current reading in mA
                        ; n = 1 to 24 maps to LV1 to LV24
MODm/LVHC_Vn=f          ; LV(X)Bias only: Floating point module m low
                        ; voltage high current n voltage reading in V
                        ; n = 1 to 6 maps to LV25 to LV30
MODm/LVHC_In=f          ; LV(X)Bias only: Floating point module m low
                        ; voltage high current n current reading in mA
                        ; n = 1 to 6 maps to LV25 to LV30
MODm/HVLC_Vn=f          ; HV(X)Bias only: Floating point module m high
                        ; voltage low current n voltage reading in V
                        ; n = 1 to 24 maps to HV1 to HV24
MODm/HVLC_In=f          ; HV(X)Bias only: Floating point module m high
                        ; voltage low current n current reading in mA
                        ; n = 1 to 24 maps to HV1 to HV24
MODm/HVHC_Vn=f          ; HV(X)Bias only: Floating point module m high
                        ; voltage high current n voltage reading in V
                        ; n = 1 to 6 maps to HV25 to HV30
MODm/HVHC_In=f          ; HV(X)Bias only: Floating point module m high
                        ; voltage high current n current reading in mA
                        ; n = 1 to 6 maps to HV25 to HV30

MODm/TEMPA=f            ; Heater(X) only: Floating point temperature
                        ; sensor A reading in K
MODm/TEMPB=f            ; Heater(X) only: Floating point temperature
                        ; sensor B reading in K
MODm/TEMPC=f            ; HeaterX only: Floating point temperature
                        ; sensor C reading in K
MODm/HEATERAOUTPUT=f    ; Heater only: Floating point heater A
                        ; output in V
MODm/HEATERBOUTPUT=f    ; Heater only: Floating point heater B
                        ; output in V
MODm/HEATERAP=d         ; Heater only: Heater A P term contribution
                        ; to PID loop (signed integer)
MODm/HEATERAI=d         ; Heater only: Heater A I term contribution
                        ; to PID loop (signed integer)
MODm/HEATERAD=d         ; Heater only: Heater A D term contribution
                        ; to PID loop (signed integer)
MODm/HEATERBP=d         ; Heater only: Heater B P term contribution
                        ; to PID loop (signed integer)
MODm/HEATERBI=d         ; Heater only: Heater B I term contribution
                        ; to PID loop (signed integer)
MODm/HEATERBD=d         ; Heater only: Heater B D term contribution
                        ; to PID loop (signed integer)
```

```
MODm/DINPUTS=bbbbbbbb   ; LV(X)Bias and Heater(X): reports the status
                        ; of DIO1 to DIO8 (each is 0=low or 1=high)
MODm/MAG_Vn=f           ; HS only: Floating point module m magnitude n
                        ; voltage reading in V
MODm/MAG_In=f           ; HS only: Floating point module m magnitude n
                        ; current reading in mA
MODm/OFS_Vn=f           ; HS only: Floating point module m offset n
                        ; voltage reading in V
MODm/OFS_In=f           ; HS only: Floating point module m offset n
                        ; current reading in mA
MODm/DINPUTS=bbbb       ; HS and LVDS: reports the status of
                        ; DIO1 to DIO4 (each is 0=low or 1=high)
MODm/VCPU_OUTREGn=d     ; Modules with DIO: VCPU output register n
                        ; (unsigned 16-bit integer)
```

TIMER

Reports the internal 64-bit timer/counter. One tick of the counter is 10 ns. Use this command as a reference for synchronizing the frame buffer timestamps to universal time on the host. In testing on a Win XP machine with a dedicated network link to Archon, the delta between the fetched Archon timer to the host computer timer had a standard deviation of 5.7us with a worst case of +/- 20us.

```
TIMER=x                 ; Current hexadecimal 64-bit internal timer
```

FRAME

Reports the frame buffer status. The response is a sequence of KEY=VALUE pairs separated by spaces and terminated with a newline. The buffer number is 1 to 3. The list of keys returned follows.

```
TIMER=x                 ; Current hexadecimal 64-bit internal timer
RBUF=d                  ; Current buffer number locked for reading
WBUF=d                  ; Current buffer number locked for writing
BUFnSAMPLE=d            ; Buffer n sample mode, 0: 16 bit, 1: 32 bit
BUFnCOMPLETE=d          ; Buffer n complete, 1: buffer ready to read
BUFnMODE=d              ; Buffer n mode, 0: top, 1: bottom, 2: split
BUFnBASE=d              ; Buffer n base address for fetching
BUFnFRAME=d             ; Buffer n frame number
BUFnWIDTH=d             ; Buffer n width
BUFnHEIGHT=d            ; Buffer n height
BUFnPIXELS=d            ; Buffer n pixel progress
BUFnLINES=d             ; Buffer n line progress
BUFnRAWBLOCKS=d         ; Buffer n raw blocks per line
BUFnRAWLINES=d          ; Buffer n raw lines
BUFnRAWOFFSET=d         ; Buffer n raw offset
BUFnTIMESTAMP=x         ; Buffer n hexadecimal 64-bit time stamp
BUFnRETIMESTAMP=x       ; Buffer n trigger rising edge time stamp
BUFnFETIMESTAMP=x       ; Buffer n trigger falling edge time stamp
BUFnREATIMESTAMP=x      ; Buffer n trigger A rising edge time stamp
BUFnFEATIMESTAMP=x      ; Buffer n trigger A falling edge time stamp
BUFnREBTIMESTAMP=x      ; Buffer n trigger B rising edge time stamp
BUFnFEBTIMESTAMP=x      ; Buffer n trigger B falling edge time stamp
```

FETCHLOG

Fetches the oldest log entry.

LOCKn

Locks frame buffer n for reading, where n is 1 to 3. Use n = 0 to unlock all buffers.

VERIFYMODxxyyyyzzzz

Verifies (reads) the firmware of module xx (hex, 00 = module 1), starting at block address yyyy (hex), and reading zzzz (hex) blocks. The controller replies with one binary response per requested block. Each block is 1024 bytes.

ERASEMODxx

Erases the firmware of module xx (hex, 00 = module 1).

FLASHMODxxyyyyzzz…zzz

Flashes a 1024 byte block of the firmware of module xx (hex, 00 = module 1), starting at block address yyyy (hex), using the 1024 hexadecimal bytes zzz…zzz.

ERASExxxxxxxxyyyyyyyy

Erases yyyyyyyy (hex) bytes of the backplane firmware starting at address xxxxxxxx (hex).

FLASHxxxxyyy…yyy

Flashes a 1024 byte block of the backplane firmware, starting at block address xxxx (hex), using the 1024 hexadecimal bytes yyy…yyy.

VERIFYxxxxyyyy

Verifies (reads) the backplane firmware, starting at block address xxxx (hex), and reading yyyy (hex) blocks. The controller replies with one binary response per requested block. Each block is 1024 bytes.

REBOOT

Reboots the backplane, which forces the backplane and all modules to reset and reread all FPGA firmware from the configuration memories. This will cause the network connection to drop.

WARMBOOT

Forces the backplane processor to restart, without causing the backplane or module FPGAs to reload their firmware. This will cause the network connection to drop.

FETCHxxxxxxxxyyyyyyyy

Fetches (reads) yyyyyyyy (hex) 1024 byte blocks of the backplane RAM starting at address xxxxxxxx (hex). The controller replies with one binary response per requested block. This is usually used to read the frame buffer contents.

WCONFIGxxxxttt…ttt

Write the text ttt…ttt to configuration line xxxx (hex). The maximum number of configuration lines is 16384, and the maximum configuration line length is 2048 characters.

RCONFIGxxxx

Reads the configuration line xxxx (hex).

CLEARCONFIG

Clears the configuration memory.

APPLYALL

Parses and applies the complete system configuration from the configuration memory to the system (excluding network configuration data). Power to the CCD will be off after this operation.

POWERON

Turns power on to the CCD. An APPLYALL is required before this operation.

POWEROFF

Turns power to the CCD off.

LOADTIMING

Parses and compiles the timing script and parameters contained in the configuration memory, and applies them to the system. This resets the timing cores.

LOADPARAMS

Parses the timing parameters contained in the configuration memory, and applies them to the system. This does not reset the timing cores. Note: the parameters are updated system-wide one at a time, starting with the first in the parameter list.

LOADPARAM p

Parses the timing parameters contained in the configuration memory, and applies the parameter named "p" to the system. This does not reset the timing cores.

PREPPARAM p

Parses the timing parameters contained in the configuration memory, and prepares the parameter named "p" to be applied to the system. The new parameter value is loaded when the EXTLOAD signal goes high. This command is intended for use when synchronizing multiple systems. This does not reset the timing cores.

FASTLOADPARAM p d

Immediately loads the parameter named "p" with the value "d", where d is from 0 to 1000000. This does not reset the timing cores. The configuration memory is unchanged, and is not parsed. A configuration defining the parameter "p" must have already been loaded and applied.

FASTPREPPARAM p d

Immediately prepares the parameter named "p" to be loaded with the value "d", where d is from 0 to 1000000. The new parameter value is loaded when the EXTLOAD signal goes high. This command is intended for use when synchronizing multiple systems. This does not reset the timing cores. The configuration memory is unchanged, and is not parsed. A configuration defining the parameter "p" must have already been loaded and applied.

RESETTIMING

Resets the timing cores. This has the effect of starting all timing cores from the first line of the timing script.

HOLDTIMING

Holds all timing cores in reset until released. Timing core outputs will have the values of the first state in the timing script. Release the timing cores from reset (and begin the timing script) using "RELEASETIMING".

RELEASETIMING

Releases the timing cores from reset if they had been held there by the "HOLDTIMING" command.

APPLYMODxx

Parses and applies the configuration for module xx (hex, 00 = module 1) from the configuration memory.

APPLYDIOxx

Parses and applies the DIO and VCPU configuration for module xx (hex, 00 = module 1) from the configuration memory.

APPLYSYSTEM

Parses and applies the backplane-specific system settings (mostly trigger control) from the configuration memory to the system.

APPLYCDS

Parses and applies the deinterlacing and CDS settings from the configuration memory to the system.

FLASHACTIVECONFIG

Store the current configuration into nonvolatile flash memory

ERASESTOREDCONFIG

Erase the configuration data stored in nonvolatile flash memory

APPLYNET

Begin using the IP and port in the current configuration for communication

Configuration

All of the configuration information for the Archon controller is described by a text file of KEY=VALUE pairs. The configuration is typically stored as a text file on the host system and written to Archon over the network interface. The configuration will usually be modified by a program on the host system based on user input (e.g. to set the integration time). The configuration is stored as text in a dedicated, volatile configuration memory in Archon. This memory accommodates 16384 configuration lines, which can each be 2048 characters long. At startup, the configuration memory is typically first cleared by the host program using the CLEARCONFIG command, and then filled with the desired configuration using the WCONFIG command. The host can also read back the configuration memory using the RCONFIG command. Once a complete configuration is written, an APPLYALL command will instruct Archon to parse the configuration and apply it to the system. A snippet of a potential communication is shown below:

```
HOST : >01CLEARCONFIG
ARCHON: <01
HOST : >02WCONFIG0000LINES=99
ARCHON: <02
HOST : >03WCONFIG0001STATES=7
ARCHON: <03
HOST : >04RCONFIG0001
ARCHON: <04STATES=7
```

There is also a section of nonvolatile flash memory dedicated for permanently storing a configuration. The active configuration can be written to this flash storage using the FLASHACTIVECONFIG command, and the flash storage can be cleared using the ERASESTOREDCONFIG command. The configuration stored in the flash memory is copied to the active configuration on power-up (but is not applied). In particular, this stored configuration is used to configure the controller's IP and port number at power up.

The list of configuration keys follows.

IP

IP address for controller communication. Default is 10.0.0.2. This setting is only applied during power up when read from the nonvolatile flash configuration memory, or after receiving the APPLYNET command.

PORT

Port number for controller communication. Default is 4242. This setting is only applied during power up when read from the nonvolatile flash configuration memory, or after receiving the APPLYNET command.

LINECOUNT

Number of lines per tap, from 1 to 65535.

LINES
Number of lines in the timing script, from 0 to 2048.

LINEn
Text for timing script line n, where n is from 0 to 2047.

STATES
Number of timing states, from 0 to 2047.

STATEn/NAME
Assign a name to state n, where n is from 0 to 2047.

STATEn/CONTROL
Set state n control clocks.

STATEn/MODi
Set state n clocks and keeps for module i.

PARAMETERS
Set the number of parameter definition lines, from 0 to 255.

PARAMETERn
Text for parameter definition line n.

CONSTANTS
Set the number of constant definition lines, from 0 to 255.

CONSTANTn
Text for constant definition line n.

SHP1
Set the start of the sample and hold period for the reset pedestal.

SHP2
Set the end of the sample and hold period for the reset pedestal.

SHD1
Set the start of the sample and hold period for the video data pedestal.

SHD2
Set the end of the sample and hold period for the video data pedestal.

BIGBUF
Enable (1) for 2x 768MB frame buffers, disable (0) for 3x 512MB frame buffers.

RAWENABLE
Enable (1) or disable (0) raw data capture.

RAWSEL
Select the AD channel for raw data capture, from 0 to 15.

RAWSTARTLINE
Set the first line for which raw data will be captured, from 0 to 65535.

RAWENDLINE
Set the last line for which raw data will be captured, from 0 to 65535.

RAWSTARTPIXEL
Set the first pixel for which raw data will be captured, from 0 to 65535.

RAWSAMPLES
Set the number of raw samples per line to capture, from 0 to 65535.

SAMPLEMODE
Set the sample mode, where 0 is 16-bit and 1 is 32-bit.

PIXELCOUNT
Set the number of pixels captured per tap.

FRAMEMODE
Set the frame deinterlacing mode, where 0 is top first, 1 is bottom first, and 2 is split.

LINESCAN
Backplane Rev. E and later: Enable (1) or disable (0) line scan mode. [1.0.1028]

TAPLINES
Set the number of tap definition lines, from 0 to 63.

TAPLINEn
Text for tap definition line n.

TRIGOUTFORCE
Set to 1 to force the trigger out level to match the TRIGOUTLEVEL setting. Set to 0 to have the trigger out level controlled by the INT control clock.

TRIGOUTLEVEL
The trigger out level is set to this value (0 or 1) when TRIGOUTFORCE is 1.

TRIGOUTINVERT

The trigger out level is inverted when 1, or unchanged when 0. This always affects the trigger out level, whether the trigger level is forced or derived from the INT control clock.

TRIGOUTPOWER

Backplane Rev. E and earlier only: Set to 1 to actively drive the trigger out line using an internal 3.3 V power supply. Set to 0 for fully optoisolated operation. (Rev E and earlier)

TRIGINENABLE

The trigger in line is connected to the timing core reset when 1, or ignored when 0. The trigger in line is always ignored if the system is synchronized to an external clock. [1.0.876]

TRIGININVERT

When TRIGININVERT is set to 0 and TRIGINENABLE is set to 1, the timing core is held in reset as long as the trigger in line is high (>= 3.3V). When TRIGININVERT is set to 1 and TRIGINENABLE is set to 1, the timing core is held in reset as long as the trigger in line is low (0V). [1.0.876]

EXTCLOCK

Backplane Rev. D and earlier only: Set to 1 to switch from an internal 25 MHz clock source to an externally provided 25 MHz clock source (on the EXTCLK pins). When EXTCLOCK is 1, the timing core RESET and LOAD signals are driven by EXTRESET and EXTLOAD, so that multiple controllers (with identical timing scripts) may be synchronized.

FANDISABLE

Backplane Rev. F and later only: Set to 1 to disable the internal cooling fan, or set to 0 to enable the fan. This can be used to eliminate fan vibration when the system is being water cooled.
WARNING: Disabling the fan without water cooling the system will probably cause the system to overheat and shut down, potentially reducing its lifetime.

APPLYALL

Set to 1 to have the controller perform an “Apply All” using the configuration stored in the controller at power up. [1.0.1054]

POWERON

Set to 1 to have the controller perform a “Power On” using the configuration stored in the controller at power up. [1.0.1054]

MODm/LABELi

Driver: Define a text label for clock channel i of module m, with i from 1 to 8.
DriverX: Define a text label for clock channel i of module m, with i from 1 to 12.

MODm/FASTSLEWRATEi

Driver: Define the fast slew rate (floating, in V / us, from 0.001 to 1000.0) for clock channel i of module m, with i from 1 to 8. This can use one of the defined constants.
DriverX: Define the fast slew rate (floating, in V / us, from 0.001 to 1000.0) for clock channel i of module m, with i from 1 to 12. This can use one of the defined constants.

MODm/SLOWSLEWRATEi

Driver: Define the slow slew rate (floating, in V / us, from 0.001 to 1000.0) for clock channel i of module m, with i from 1 to 8. This can use one of the defined constants.
DriverX: Define the slow slew rate (floating, in V / us, from 0.001 to 1000.0) for clock channel i of module m, with i from 1 to 12. This can use one of the defined constants.

MODm/ENABLEi

Driver: Set to 1 to enable clock channel i (1..8) of module m, or 0 to power down and disable.
DriverX: Set to 1 to enable clock channel i (1..12) of module m, or 0 to power down and disable.

MODm/SOURCEi

Driver: Set to desired timing core clock source (1..8) for clock channel i. By default, 1 maps to 1, 2 maps to 2, etc. [1.0.1064]
DriverX: Set to desired timing core clock source (1..12) for clock channel i. By default, 1 maps to 1, 2 maps to 2, etc.

MODm/CLAMPHIGH

ADC Rev C: Set the DC clamp level for the high/positive side of the preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/CLAMPLOW

ADC Rev C: Set the DC clamp level for the low/negative side of the preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/CLAMP1

ADC Rev D: Set the DC clamp level for the channel 1 preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/CLAMP2

ADC Rev D: Set the DC clamp level for the channel 2 preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/CLAMP3

ADC Rev D: Set the DC clamp level for the channel 3 preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/CLAMP4

ADC Rev D: Set the DC clamp level for the channel 4 preamp, from -2.500 V to +2.500 V. This can use one of the defined constants.

MODm/PREAMPGAIN

ADC: Set to 0 to select the low preamp gain (4 V input range), and 1 for the high preamp gain (1.33 V input range). This can use one of the defined constants.

MODm/LVLC_LABELi

LVBias: Define a text label for LVLC channel i (i from 1 to 24).

MODm/LVHC_LABELi

LVBias: Define a text label for LVHC channel i (i from 1 to 6).

MODm/LVLC_Vi

LVBias: Set the power on voltage (from -14.000 to +14.000) for LVLC channel i (i from 1 to 24).

MODm/LVLC_ORDERi

LVBias: Set the power on order (>= 0) for LVLC channel i (i from 1 to 24).

MODm/LVHC_Vi

LVBias: Set the power on voltage (from -14.000 to +14.000) for LVHC channel i (i from 1 to 6).

MODm/LVHC_ORDERi

LVBias: Set the power on order (>= 0) for LVHC channel i (i from 1 to 6).

MODm/LVHC_ENABLEi

LVBias: Set to 1 to enable LVHC channel i of module m, or 0 to power down and disable.

MODm/LVHC_ILi

LVBias: Set the current limit in mA (from 0 to 500) for LVHC channel i (i from 1 to 6).

MODm/HVLC_LABELi

HVBias: Define a text label for HVLC channel i (i from 1 to 24).

MODm/HVHC_LABELi

HVBias: Define a text label for HVHC channel i (i from 1 to 6).

MODm/HVLC_Vi

HVBias: Set the power on voltage (from 0.000 to +31.000) for HVLC channel i (i from 1 to 24).

MODm/HVLC_ORDERi

HVBias: Set the power on order (>= 0) for HVLC channel i (i from 1 to 24).

MODm/HVHC_Vi

HVBias: Set the power on voltage (from 0.000 to +31.000) for HVHC channel i (i from 1 to 6).

MODm/HVHC_ORDERi

HVBias: Set the power on order (>= 0) for HVHC channel i (i from 1 to 6).

MODm/HVHC_ENABLEi

HVBias: Set to 1 to enable HVHC channel i of module m, or 0 to power down and disable.

MODm/HVHC_ILi

HVBias: Set the current limit in mA (from 0 to 250) for HVHC channel i (i from 1 to 6).

MODm/HEATERxENABLE

Heater/HeaterX: Set to 0 to disable or 1 to enable the Heater x (x = A or B) output.

MODm/HEATERxFORCE

Heater/HeaterX: Set to 1 to force the Heater x (x = A or B) output to the HEATERxFORCELEVEL, 0 for normal operation.
WARNING: Forcing the heater output on can cause overheating if the temperature isn't monitored.

MODm/HEATERxFORCELEVEL

Heater/HeaterX: Voltage level for Heater x (x = A or B) output when HEATERxFORCE is 1, from 0.000 to 25.000.

MODm/HEATERxLIMIT

Heater/HeaterX: Maximum voltage level for Heater x (x = A or B) output in PID mode, from 0.000 to 25.000.

MODm/HEATERxTARGET

Heater/HeaterX: Set target temperature for Heater x (x = A or B) control loop in degrees C (from -150.0 to 50.0). [-250 to 50 for 1.0.1087]

MODm/HEATERxSENSOR

Heater/HeaterX: Select temperature sensor used to control Heater x (x = A or B) output (0 is A, 1 is B, 2 is C; C only available on HeaterX modules).

MODm/HEATERxSENSORTYPE

Heater: Select temperature sensor x (x = A or B) type (0 is DT-670, 1 is DT-470, 2 is RTD100, 3 is RTD400). Note that the RTD100 and RTD400 types require custom resistors to be installed on the Rev D Heater modules.

MODm/SENSORxTYPE

HeaterX: Select temperature sensor x (x = A, B or C) type (0 is DT-670, 1 is DT-470, 2 is RTD100, 3 is RTD400, 4 is RTD1000, 5 is RTD2000). [1.0.758, 1.0.1061 for RTD2000]

MODm/SENSORxCURRENT

HeaterX: Select temperature sensor x (x = A, B or C) current in nA. This is used only for RTDs, for the DT-470/DT-670 this defaults to 10 uA. [1.0.758]

MODm/SENSORxLOWERLIMIT

Heater/HeaterX: Select temperature sensor x (x = A, B, or C; C only available on HeaterX modules) lower limit in degrees C (from -150.0 to 50.0). [-250 to 50 for 1.0.1002]

MODm/SENSORxUPPERLIMIT

Heater/HeaterX: Select temperature sensor x (x = A, B, or C; C only available on HeaterX modules) upper limit in degrees C (from -150.0 to 50.0). [-250 to 50 for 1.0.1002]

MODm/SENSORxFILTER

HeaterX: Enable digital filtering to average previous readings (from 0 to 8, where 0 is disabled, 1 averages the last 2 readings, up to 8 which averages the last 256 readings). [1.0.1054]

MODm/HEATERxP

Heater/HeaterX: Set P term for Heater x (x = A or B) control loop (from 0 to 10000).
HeaterX: Fractional P term support added (from 0.000 to 10000.000) [Backplane 1.0.1054]

MODm/HEATERxI

Heater/HeaterX: Set I term for Heater x (x = A or B) control loop (from 0 to 10000). [1.0.478]
HeaterX: Fractional I term support added (from 0.000 to 10000.000) [Backplane 1.0.1054]

MODm/HEATERxIL

Heater/HeaterX: Set I limit term for Heater x (x = A or B) control loop (from 0 to 10000). [1.0.478]

MODm/HEATERxD

Heater/HeaterX: Set D term for Heater x (x = A or B) control loop (from 0 to 10000). [1.0.478]
HeaterX: Fractional D term support added (from 0.000 to 10000.000) [Backplane 1.0.1054]

MODm/HEATERUPDATETIME

Heater: Set update time for control loop (from 1 to 30000, in milliseconds). [1.0.478]

MODm/HEATERxRAMP

Heater/HeaterX: Set to 0 to disable Heater x (x = A or B) ramping, or 1 to enable ramping (the heater target linearly ramps from the current temperature to the set point). [1.0.548]

MODm/HEATERxRAMPRATE

Heater/HeaterX: Set ramp rate for Heater x (x = A or B), in mK / update time (from 1 to 32767). [1.0.548]

MODm/HEATERxLABEL

HeaterX: Define a text label for heater x (x = A or B). [1.0.758]

MODm/SENSORxLABEL

HeaterX: Define a text label for sensor x (x = A, B or C). [1.0.758]

MODm/DIO_LABELi

LVBias/Heater/HeaterX: Define a text label for DIO line i (i from 1 to 8). [1.0.624]
HS/LVDS: Define a text label for DIO line i (i from 1 to 4). [1.0.695]

MODm/DIO_SOURCEi

LVBias/Heater/HeaterX: Select the signal source for DIO line i (i from 1 to 8), where 0 is low, 1 is high, 2 selects the timing core, and 3 selects the VCPU. [1.0.784]
HS/LVDS: Select the signal source for DIO line i (i from 1 to 4), where 0 is low, 1 is high, 2 selects the timing core, and 3 selects the VCPU. [1.0.784]

MODm/DIO_DIRi

LVBias/Heater/HeaterX: Select the direction for DIO lines i (i = 12, 34, 56, or 78), using 0 for input, 1 for output. [1.0.624]
HS/LVDS: Select the direction for DIO lines i (i = 1, 2, 3, 4), using 0 for input, 1 for output. [1.0.695]

MODm/DIO_POWER

LVBias/Heater/HeaterX /HS/LVDS: Set to 0 to use an external voltage to power the digital I/O line buffers, or 1 to the internal +3.3V supply. The +3.3V supply is routed to the DPWR pin when enabled. [1.0.624]

MODm/VCPU_LINES

LVBias/Heater/HeaterX /HS/LVDS: Set the number of VCPU program lines, from 0 to 511. [1.0.784]

MODm/VCPU_LINEi

LVBias/Heater/HeaterX /HS/LVDS: Text for VCPU program line i (i = 0-511). [1.0.784]

MODm/VCPU_INREGi

LVBias/Heater/HeaterX /HS/LVDS: Set VCPU input register i (i = 0-15), from 0 to 65535. [1.0.784]

MODm/HS_LABELi

HS: Define a text label for high speed LVDS channel i (i = 1-12). [1.0.695]

MODm/MAG_LABELi

HS: Define a text label for high speed magnitude channel i (i = 1-12). [1.0.695]

MODm/HS_LABELi

HS: Define a text label for high speed offset channel i (i = 1-12). [1.0.695]

MODm/MAG_Vi

HS: Set the clock magnitude (5.000 to 14.000) for high speed channel i (i from 1 to 12). [1.0.695]

MODm/OFS_Vi

HS: Set the clock offset (-14.000 to 14.000) for high speed channel i (i from 1 to 12). [1.0.695]

MODm/LVDS_LABELi

LVDS: Define a text label for LVDS channel i (i = 1-16). [1.0.741]

MODm/XVP_LABELi

XVBias: Define a text label for positive bias channel i (i = 1-4). [1.0.758]

MODm/XVN_LABELi

XVBias: Define a text label for negative bias channel i (i = 1-4). [1.0.758]

MODm/XVP_Vi

XVBias: Set the power on voltage (from 0.000 to +95.000) for positive channel i (i from 1 to 4). [1.0.758]

MODm/XVN_Vi

XVBias: Set the power on voltage (from -95.000 to 0.000) for negative channel i (i from 1 to 4). [1.0.758]

MODm/XVP_ENABLEi

XVBias: Set to 1 to enable positive channel i (i from 1 to 4) of module m, or 0 to power down and disable. [1.0.758]

MODm/XVN_ENABLEi

XVBias: Set to 1 to enable negative channel i (i from 1 to 4) of module m, or 0 to power down and disable. [1.0.758]

MODm/XVP_ORDERi

XVBias: Set the power on order (>= 0) for positive channel i (i from 1 to 4). [1.0.758]

MODm/XVN_ORDERi

XVBias: Set the power on order (>= 0) for negative channel i (i from 1 to 4). [1.0.758]

Timing Core

Most of the modules have timing cores integrated into their FPGAs. The timing core is essentially a state machine. All timing cores are synchronized by the backplane, which distributes a master clock (MCLK) signal, a reset (RESET) signal, and a load (LOAD) signal to all modules. Each timing core has an associated RAM (with space for 2048 instructions) that is loaded with op-codes and output signal states during system configuration. The same sequence of op-codes is loaded into all timing cores; only the output signal states vary. The RESET signal forces all timing cores to begin from address zero. The outputs of the timing core are a function of the module: the ADC module's timing core outputs a clamp signal, the driver module's timing core outputs the desired clock level for each channel, and so on.

In addition to the instruction memory, there is also a set of 64 parameters that are used by the timing core. These parameters can be used to set loop counts (such as number of pixels per line). They can also be tested (zero / not zero) for conditional jumps, and can optionally be decremented by one by an instruction. A parameter can be changed on the fly – the parameter to change and its new value are set on all the boards, and then the LOAD signal is asserted to synchronously change that parameter on all modules.

Instructions

The simplest timing core instruction is a NOP (no operation). The outputs are set to the state stored with the NOP for one master clock cycle, and the timing core proceeds to the next instruction in memory. All timing core instructions execute in one clock cycle, and must define the signal outputs for the current clock cycle. Each output can be commanded high, low, or to keep its previous value.

A GOTO instruction gives the timing core a new address to begin execution at. The GOTO can be made conditional on a parameter value.

A CALL instruction commands the timing core to push the next execution address and a count value to a stack, and to begin execution at a provided address. The stack allows subroutines to be nested 16 levels deep. The count value can be either a constant or a parameter. If the count is a parameter, the CALL will only be executed if the count is non-zero. The CALL can be made conditional on another parameter value. Counts and parameters are 20-bit values.

A RETURN instruction commands the timing core to decrement the active count value. If the count is non-zero, execution jumps to an address provided with the RETURN instruction. If the count is zero, the next execution address and active count value are popped from the stack.

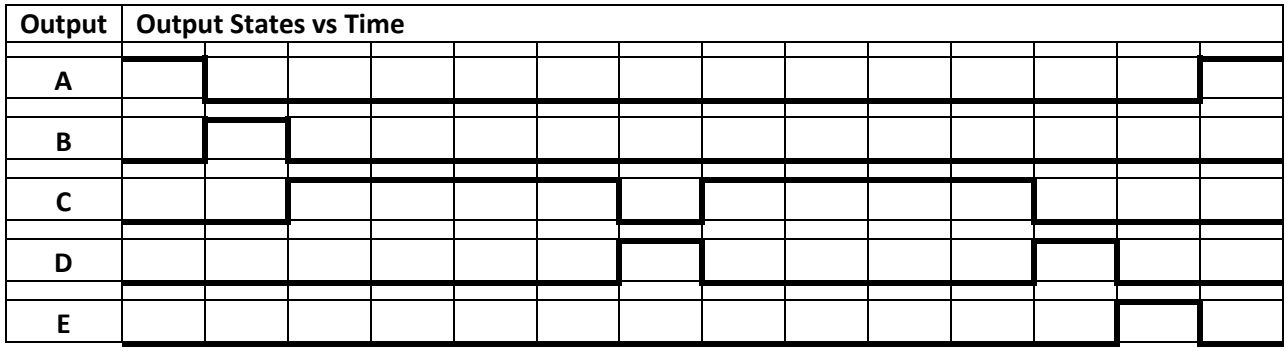
Optionally during any instruction, a parameter can be decremented. This is useful when an external trigger is meant to fire a sequence of one or more frame captures followed by a return to an idle mode. Decrementing a parameter that's already at zero has no effect. The decrement operation occurs before control passes to the next instruction (whether that's the next instruction in the script, or the result of a CALL or GOTO during this clock tick).

Timing Script

The sequence of states the timing core will emit is defined by a timing script. Scripts are basic text with the following statements allowed:

```
# Any line starting with a hash is a comment, and is ignored.  Blank lines are also ignored.

# A name followed by a colon is a label, and is used as a target for GOTO, CALL and
# RETURN instructions
MyLabel:

# For any line that is not a label, the first token expected is the name of an output state.
# These output states are defined outside of the script, and declare the state of the
# timing core outputs for this clock cycle (high, low, or no change).
AllClocksLow

# A jump is declared as follows:
AllClocksLow; GOTO MyLabel

# Parameters are also defined outside the script, and are referenced by name.
# To conditionally jump if a parameter is non-zero:
AllClocksLow; IF MyParameter GOTO MyLabel

# To conditionally jump if a parameter is zero:
AllClocksLow; IF !MyParameter GOTO MyLabel

# To call a subroutine 100 times using a direct value, a declared constant or a parameter:
AllClocksLow; CALL MySubroutine(100)
AllClocksLow; CALL MySubroutine(MyConstant)
# MyParameter would have to be set to 100 for the subroutine to execute 100 times.
# If MyParameter is zero, the call will not execute.  Constants must be non-zero.
AllClocksLow; CALL MySubroutine(MyParameter)
# All of these CALLs push the next execution address and current active count value to
# the stack, set the active count value to the count provided, and jump to the given address.

# A subroutine that has all clocks high for one clock tick would look like this:
MySubroutine:
AllClocksHigh; RETURN MySubroutine
# The RETURN statement tells the core to decrement the active count.  If zero, it pops the
# return address and active count from the stack.  If non-zero, it jumps to the provided
# address (executing MySubroutine again).

# It's often convenient to hold a particular output state for a given number of clock cycles.
# This could be done by defining a subroutine with just one instruction (as in the
# MySubroutine example above) and calling it 100 times.  This can also be accomplished
# as follows:
```

```
AllClocksHigh; AllClocksHigh(99)
# This implicitly generates a subroutine and calls it – all clocks will be high for 100 clock cycles.

# Here is an example script with output.  A timing core with outputs A through E is assumed,
# with a single output high in each state
ExampleStart:
StateA
StateB; CALL ExampleSub(2)
StateE; GOTO ExampleStart
ExampleSub:
StateC; StateC(3)
StateD; RETURN ExampleSub
```




```
# To decrement a parameter, add a directive to the end of a line like this:
AllClocksLow; MyParameter--
AllClocksLow; CALL MySubroutine(100); MyParameter—

# Backplanes with firmware >= 1.0.1105 can also increment a parameter
AllClocksLow; MyParameter++
```

Timing Core States

The states used in a timing script are defined by the configuration file. The STATES key declares the number of timing states defined (0…2047). Each state is then defined by a set of subkeys for a STATEn key (where n is 0…2047). The first subkey is STATEn/NAME, and it defines the name used to reference a state from the timing script. The next subkey is STATEn/CONTROL, which defines the backplane control clocks. This subkey has the form:

```
STATEn/CONTROL=a,b      ; a is the hexadecimal clock state, where:
                        ; Bit 0 = INT
                        ; Bit 1 = FRAME
                        ; Bit 2 = LINE
                        ; Bit 3 = PIXEL
                        ; b is a hexadecimal flag. When a bit is 1, it
                        ; indicates that the corresponding clock should
                        ; keep its previous value.
```

The clock state for each module is defined by a STATEn/MODi subkey. The format of the clock state data varies based on which module is being used. For the ADC module:

```
STATEn/MODi=a,b         ; a is the hexadecimal clock state, where:
                        ; Bit 0 = CLAMP
                        ; b is a hexadecimal flag. When a bit is 1, it
                        ; indicates that the corresponding clock should
                        ; keep its previous value.
```

For the clock driver module:

```
STATEn/MODi=d1level,d1slew,d1keep,d2level,…,d8keep
                        ; dnlevel is the floating point target clock
                        ; level for each channel in Volts.  This can
                        ; also be a constant name.
                        ; dnslew is 0 to select the slow slew rate,
                        ; and 1 to select the fast slew rate.
                        ; dnkeep is 1 to leave the level and slew
                        ; unchanged, and 0 to use this state's values.
```

For the Heater modules (and LVBias/LVXBias modules < 833):

```
STATEn/MODi=d1state,d1keep,d2state,…,d8keep
                        ; dnstate is the DIO state (0 = low, 1 = high).
                        ; dnkeep is 1 to leave the state
                        ; unchanged, and 0 to use this state's value.
```

For LVBias/LVXBias modules (with firmware > 833):

```
STATEn/MODi=d1state,d1keep,d2state,…,d8keep,bcmd,bch,bval
                          ; dnstate is the DIO state (0 = low, 1 = high).
                          ; dnkeep is 1 to leave the state
                          ; unchanged, and 0 to use this state's value.
                          ; bcmd: 1 = generate a bias command, 0 = NOP
                          ; bch: bias channel to command (1-30)
                          ; bval: bias voltage to command (-14.0 to 14.0)
```

For HVBias/HVXBias modules (with firmware > 833):

```
STATEn/MODi=bcmd,bch,bval
                          ; bcmd: 1 = generate a bias command, 0 = NOP
                          ; bch: bias channel to command (1-30)
                          ; bval: bias voltage to command (0.0 to 31.0)
```

For XVBias modules (with firmware > 1090):

```
STATEn/MODi=pcmd,pch,pval,ncmd,nch,nval
                          ; pcmd: 1 = generate a P bias command, 0 = NOP
                          ; pch: P bias channel to command (1-4)
                          ; pval: P bias voltage to command (0.0 to 95.0)
                          ; ncmd: 1 = generate a N bias command, 0 = NOP
                          ; nch: N bias channel to command (1-4)
                          ; nval: N bias voltage to command(0.0 to -95.0)
```

For the HS module:

```
STATEn/MODi=hs1state,hs1keep,…,hs12keep,d1state,d1keep,…,d4keep
                          ; hsnstate is a 10 digit binary sequence for
                          ; hs channel n, where each 0 or 1 in the
                          ; sequence represents the clock state for 1 ns
                          ; of the 10 ns master clock tick.
                          ; hsnkeep is 1 to leave the previous sequence
                          ; unchanged, and 0 to use this state's value.
                          ; dnstate is the DIO state (0 = low, 1 = high).
                          ; dnkeep is 1 to leave the state
                          ; unchanged, and 0 to use this state's value.
```

For the LVDS module:

```
STATEn/MODi=lvds1state,lvds1keep,…,lvds16keep,d1state,d1keep,…,d4keep
                          ; lvdsnstate is the state for LVDS channel n
                          ; (0 = low, 1 = high)
                          ; lvdsnkeep is 1 to leave the previous state
                          ; unchanged, and 0 to use this state's value.
                          ; dnstate is the DIO state (0 = low, 1 = high).
                          ; dnkeep is 1 to leave the state
                          ; unchanged, and 0 to use this state's value.
```

Sampling and Deinterlacing

The CCD outputs are continuously sampled at 100 MHz by the ADC modules. The controller needs to know which samples to use for calculating a pixel value, and where to put those pixels in the image frame buffer. The timing for these operations is controlled by the FRAME, LINE, and PIXEL control clocks. PIXEL goes high for one master clock tick at the start of a pixel. This pulse causes the internal counters and accumulators for performing correlated double sampling (CDS) to reset to zero. CDS is performed by summing the ADC samples when the CDS counter is between SHP1 and SHP2 to get a reset level value, summing the ADC samples when the CDS counter is between SHD1 and SHD2 to get a video level value, normalizing the two values, and then calculating the difference between those two values. This calculated value is then adjusted by the digital gain and offset defined in the tap configuration. The accumulators are 32 bits, and the emitted pixel values can be either 16 or 32 bits. The CDS counters are 16 bit, so the longest pixel sampling time is 655.36 us, or 1.5 kHz.

The ADM modules sample at 12.5MHz. They sample after every 8th tick of the 100MHz clock, and emit an 18 bit sample. The 18 bit sample is truncated to 16 bits, and that value is repeated 8 times to feed the CDS processor at the normal 100MHz rate. Dithering is added to the sample during those 8 ticks such that averaging the 8 values recovers the original 18 bit sample. It's therefore best (but not required) to make the SHP/SHD values a multiple of 8 when using an ADM channel. The system determines when to trigger an ADM sample using a counter that is reset by PCLK.

The pixel values for each CCD output (tap) are sent to the deinterlacing engine, which calculates the destination frame buffer memory address for each arriving pixel and writes the pixels into the frame buffer. The deinterlacing engine maintains internal pixel, line and frame counters, and keeps track of which frame buffers are locked for writing or reading. If the FRAME clock is high when the PIXEL clock goes high, the line and pixel counters are reset and the frame counter is incremented. If the LINE clock is high when the PIXEL clock goes high, the pixel counter is reset and the line counter is incremented. The PIXELCOUNT and LINECOUNT configuration keys specify how many pixels and lines the deinterlacing engine waits for per tap before marking a frame complete and locking the next unused frame buffer for writing.

For line scan applications, set the LINESCAN configuration key to 1. The CCD timing should be written so that the FRAME signal goes high at the beginning of each line. The deinterlacing engine will then acquire lines into the current frame buffer until LINECOUNT lines have arrived, and then the frame will be marked complete and the next frame buffer started. In this way, a large number of lines that arrive quickly can be aggregated into a frame buffer for asynchronous readout by the host PC. Under the hood, the FRAME signal starts a frame as usual when the first line arrives, and then is ignored until the current frame buffer is complete.

Internally, the deinterlacing engine decides where to put each tap's pixels using a pointer that has an initial value at the start of a frame, and is adjusted using specified deltas after each pixel and line. A CCD with four outputs, one in each corner, would have four taps defined that initially point to the four corners of the frame buffer. The deinterlacing pattern is defined by the TAPLINES, TAPLINEn, PIXELCOUNT, LINECOUNT, FRAMEMODE and SAMPLEMODE configuration keys. FRAMEMODE is set to 0

(top) if the first sampled pixels should be written to the top of the frame buffer. FRAMEMODE is set to 1 (bottom) if the first sampled pixels should be written to the bottom of the frame buffer. FRAMEMODE is set to 2 (split) if the first sampled pixels from the first half of the taps should be written to the top of the frame buffer, and the first sampled pixels from the second half of the taps should be written to the bottom of the frame buffer. SAMPLEMODE is set to 0 for 16 bit pixels, and 1 for 32 bit pixels. The order and readout direction of each tap, along with the digital gain and offset to apply to each tap, are specified in a list of taps. The number of tap configuration lines is defined by the TAPLINES key. Each TAPLINEn key has the format:

```
TAPLINEn=tap,gain,offset
```

"tap" is a string of the form ADnd, where n is 1 to 16, and d is 'L' or 'R'. The AD channel for a tap (n) is 1 for the first channel from an ADC module in backplane slot 5, 2 for the second channel from backplane slot 5, up to 16 for the fourth channel from an ADC module in backplane slot 8. The readout direction (d) is 'L' if the first pixel values should be written at the left edge of a tap's portion of the frame buffer, and 'R' to start at the right edge. "gain" is a floating point gain such as "1.0". By default, the reset level is expected to be greater than the video level (pixel value = reset level – video level). If "gain" is negative, the reset level is assumed to be less than the video level (pixel value = video level – reset level). "offset" is an integer to add to the pixel values, which is useful to keep black pixels at some value above zero so that RMS noise can be accurately measured.

When using ADM modules, "tap" is of the form AMnd, where n is 1 to 72 and d is 'L' or 'R' as before. ADM channels 1 to 18 map to slot 5, channels 19 to 36 map to slot 6, and so on.

Raw Samples

Tuning a CCD is aided by viewing the raw CCD output waveform, as it allows for the direct inspection of characteristics such as settling times, optimal sampling windows, reset level stability, and clock feed through. The Archon controller allows some of the raw 16 bit 100 MHz ADC samples for a single channel to be acquired simultaneously with the normal frame data. Due to the finite frame buffer size (512 MB), typically only a portion of the frame can be acquired in this way.

To enable the capture of raw data, set the RAWENABLE key to 1. Set the RAWSEL key to the desired AD channel (0 for channel 1 of the ADC module in slot 5 through 15 for channel 4 of the ADC module in slot 8). Set the RAWSTARTLINE and RAWENDLINE to the first and last lines of raw data to capture. Set the RAWSTARTPIXEL key to the pixel number when raw data should start being captured for each line. Set RAWSAMPLES to the number of raw samples to store from each line. The number of samples will be rounded up to the next even block size (a multiple of 1024). Note that raw captures can quickly become very large: capturing 1000 lines of 1000 pixels with a 100 kHz pixel clock amounts to 1 gigasamples, or 2 gigabytes.

Frame Buffers

There are three 512 MByte frame buffers, occupying the upper 1.5 GBytes of the backplane's 2 GBytes of DDR3 RAM. There is usually one frame locked for reading (by the host), and another locked for writing (by the deinterlacing engine). When the deinterlacing engine starts a new frame (FRAME clock is high when PIXEL clock goes high), it locks the next unlocked frame buffer, updates the frame buffer's info (time stamp, frame number, etc.), and begins filling the buffer with data. It sets the BUFnCOMPLETE flag after the frame is complete. The FRAME command reports the status of the three frame buffers, including the width, height, sample size, timestamp, how many pixels or lines are complete, and whether the frame is complete. A host program can reduce frame capture latency by fetching partial frame data in advance using the BUFnLINES status to determine what data is valid now.

Typically, a host program will issue the FRAME command, and check the returned status data to see if there are any frame buffers holding completed frames with a frame number greater than what the program has already captured. If a new frame exists, the host program issues a LOCK command to prevent that buffer from being overwritten, and then fetches the pixel data using a FETCH command. The number of bytes to fetch for a complete frame is the frame width times the frame height times 2 (for 16-bit samples) or 4 (for 32-bit samples). Divide the number of bytes by 1024 and round up to determine the number of blocks to fetch. The starting address is 0xA0000000 for frame buffer 1, 0xC0000000 for frame buffer 2, and 0xE0000000 for frame buffer 3. Smaller portions of the frame data can be fetched at will using FETCH commands with appropriate offsets and lengths. Raw data in a frame buffer is retrieved using the same FETCH command, but after adding the frame buffer offset given by BUFnRAWOFFSET to the base frame buffer address.

If the BIGBUF configuration key is set to 1, the 1.5 GB of frame buffer memory space is divided into two larger 768MB frame buffers. The FRAMEnBASE keys from the frame status command report the new base addresses of the frame buffers (0xA0000000 and 0xD0000000). FRAME3 is unused in this mode.

GUI

Archon is provided with an example GUI application that implements the necessary functions to communicate with the controller over its network interface, apply a configuration, and read back captured images. The GUI is built on the Qt framework (http://qt-project.org) for cross-platform operation. It also uses the qwt library (qwt.sourceforge.net) for plotting functions. As of version 1.0.786, the GUI can be started with the command line parameter "-small" to add scroll bars for use on smaller monitors.

The next two sections will describe the GUI in conjunction with an example Archon controller system as seen in Figure 31.

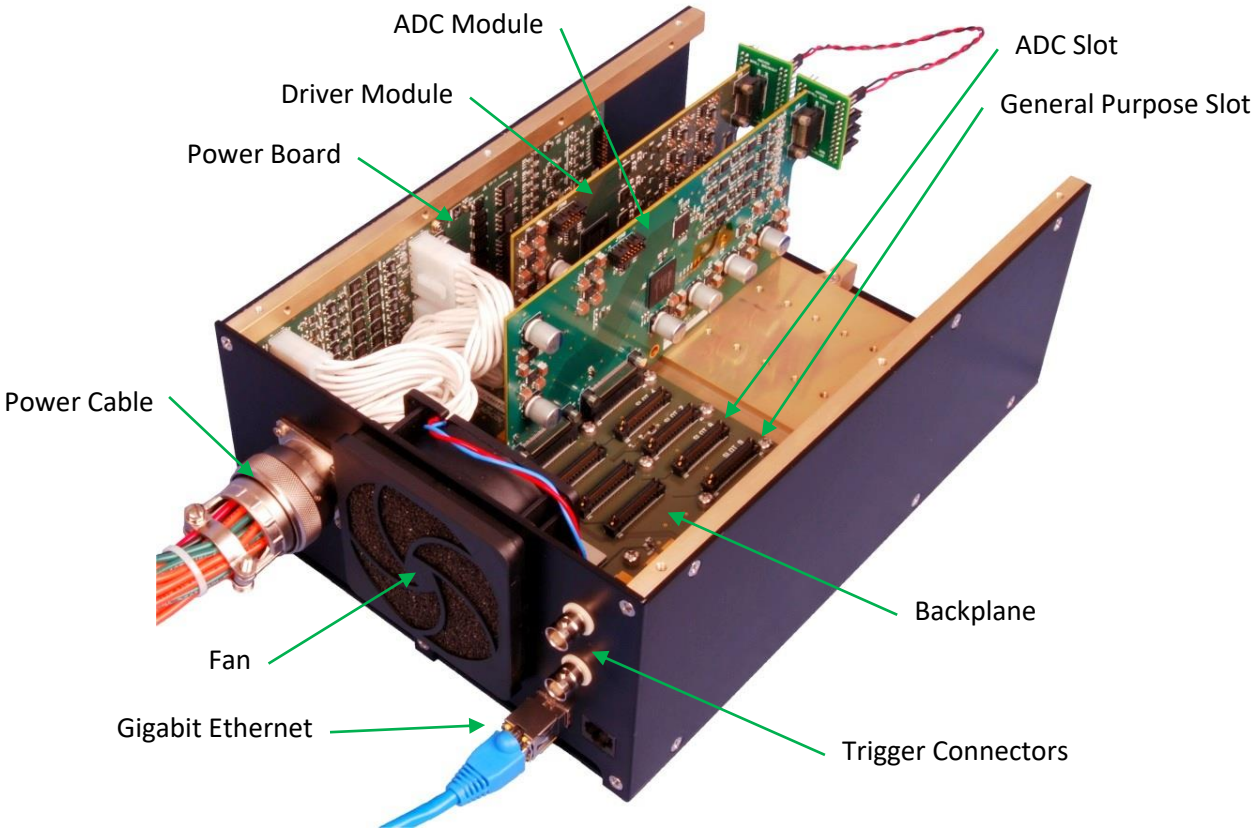


Figure 31: Example Archon Controller

The example controller has a driver module in slot 3, and an ADC module in slot 5. A jumper wire connects channel 1 of the driver to channel 1 of the ADC. The other three ADC channels are grounded. Powering on Archon, launching the GUI, and clicking "Connect" yields something similar to Figure 32.

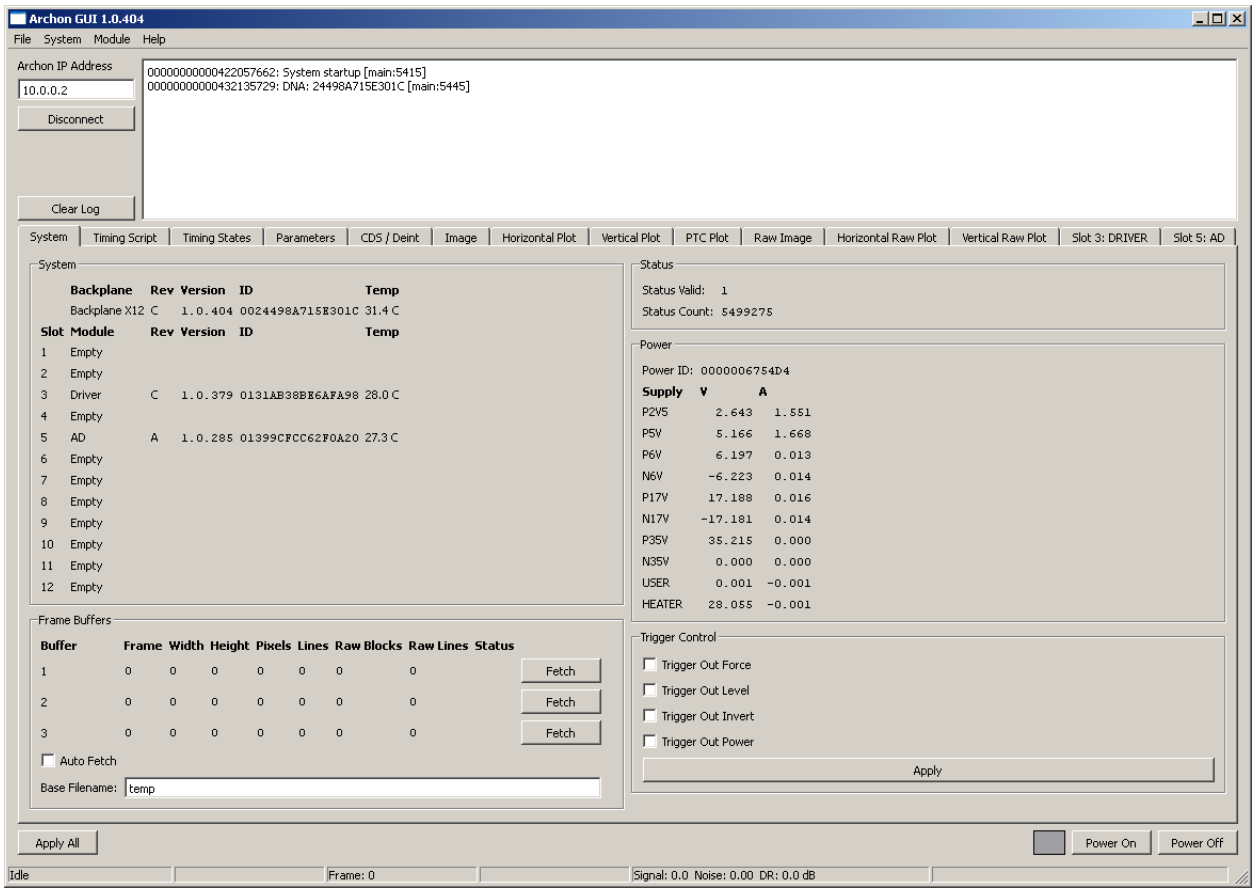


Figure 32: Archon GUI

The top portion of the GUI is dedicated to Archon communication, with the controller's IP address on the left, and a log of informational messages from the GUI and controller on the right.

The File menu allows an Archon configuration file (*.acf) to be loaded or saved. The configuration file is stored in the Windows INI format, with two sections. The first section is [SYSTEM], which stores the output of the SYSTEM network command. This allows the GUI to display and edit a system configuration even without a controller attached. The second INI section is [CONFIG], which contains all of the configuration key/value pairs that are sent to the controller.

A configuration can also be loaded or saved in the (*.ncf) format. This format modifies the normal key/value pairs so that lists (such as the timing script lines) do not have line numbers. This allows for easy configuration file comparisons using modern source control tools.

The System and Module menus allow the firmware of the connected controller to be updated or verified. For the system, the backplane firmware, microprocessor code, and/or camera configuration can be flashed depending on which items are checked in the menu. For the modules, select the target module and then select Flash or Verify. In either case, the GUI will prompt for a firmware file (*.mcs) to open.

The main buttons at the bottom of the GUI include “Apply All”, which writes the camera configuration specified in the rest of the GUI to the controller, and “Power On” and “Power Off” buttons to enable or disable power to the CCD. The box next to the power buttons is gray for an unconfigured system or unknown power state, red when power to the CCD is off, and green when power is on.

The status bar at the bottom of the GUI displays the current operation and its progress, the most recent captured frame number, and statistics from the image window.

The remainder of the GUI is broken into tabs, with several system-wide tabs, and individual tabs for each installed module.

System Tab

The system tab is divided into several boxes. The System box lists the PCB and firmware version, unique ID, and temperature of the backplane and all installed modules.

The Frame Buffers box shows the status of the three 512 MByte frame buffers in the controller. This includes the frame number stored in each buffer, along with the frame’s width, height, completed pixels, completed lines, additional raw mode data size, and status (read/write/complete). An individual buffer can be copied to the GUI’s image buffer by clicking the “Fetch” button next to the desired frame. Alternatively, the “Auto Fetch” option can be selected, and the GUI will constantly retrieve the most recently completed frame. In addition, a base filename can be specified for frames saved to disk.

The GUI continuously fetches status data from the controller while connected, including temperatures, voltages, etc (see the STATUS and FRAME network commands). The Status Box reports whether the status data retrieved is valid, and how many times the controller has updated its internal status registers.

The Power box displays the power board’s unique ID, along with voltage and current readings for each of the system power supplies.

The Trigger Control box allows the controller’s trigger output to be configured. Select the Force checkbox to force the trigger output high or low (depending on the state of the Level checkbox), or deselect it to have the trigger controlled by the timing core. Select the Invert checkbox to invert the sense of the trigger (active is defined as a high voltage by default). Select the Power checkbox to locally power the trigger output, or deselect it to optoisolate the output. Click the “Apply” button to apply the trigger configuration.

Timing Script Tab

Most of the Timing Script tab is dedicated to a textual timing script entry area. The format of the timing script is defined in the Timing Script section. The “Load Timing” button downloads the timing script, timing states, and timing parameters to the controller and resets the timing cores. The “Timing Reset” button only resets the timing cores without changing any other configuration data.

Timing States Tab

The individual timing core states are defined in the Timing States tab, as seen in Figure 33. The list of states is on the left. Add a new state with the "Add" button. Rename it by double-clicking the new state name. Delete a selected state with the "Delete" button. Duplicate a state with the "Duplicate" button. Move selected states up and down in the list with the "Move Up" and "Move Down" buttons. The levels of all system signals during a state are displayed in the tabs to the right of the state list when a state is selected. The Control tab shows system-wide control signals (the Integrate, Frame, Line, and Pixel signals). Each installed module with configurable timing is given its own tab. Simple signals can be high (checkbox checked) or low (unchecked), and can have Keep on or off. More complex signals such as a clock driver channel will have other data, such as slew rate and the desired clock voltage. In either case, if Keep is checked, the signal state will remain unchanged from its previous state. If Keep is unchecked, the signal will assume the new values defined for this state when it executes.

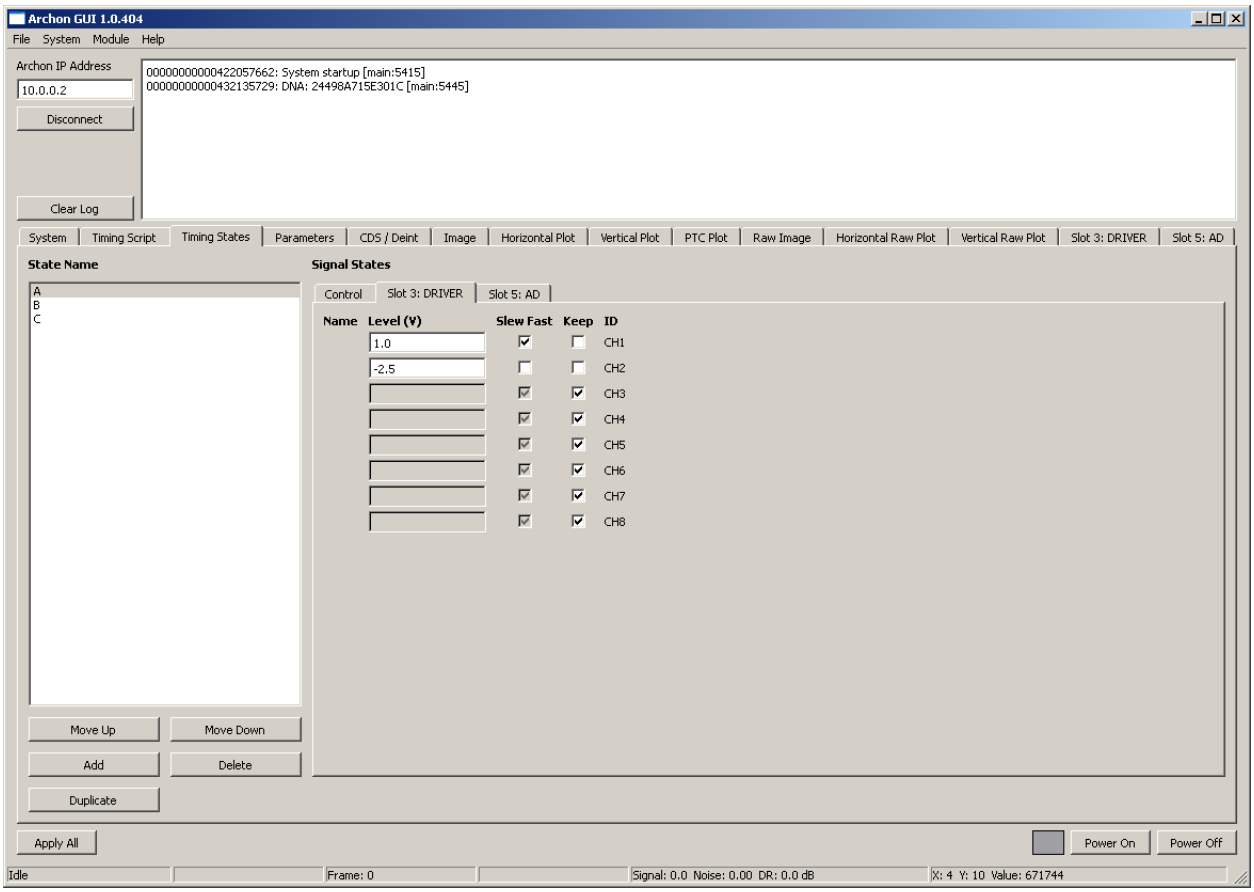


Figure 33: Timing States Entry

Parameters Tab

The Parameters tab is split into two halves. The left half is a list of up to 64 timing core parameters, in NAME=VALUE form. These are used in the timing script, and can be changed on the fly by clicking the "Apply" button. The right half of the Parameters tab is a list of named constants. These can be used in

many places in the GUI (in the timing script, as clock voltage levels, etc.), but changed values only take effect after reconfiguration (via an "Apply" button, reloading the timing, etc.).

VCPU Tab

The VCPU tab contains tabs for each installed module with a VCPU. A VCPU program can be entered for each module. In addition, there are fields to set the 16 16-bit VCPU input registers, and to read the current values of the 16 16-bit VCPU output registers. The "Apply" button loads and applies the VCPU program, input registers, and DIO settings for the currently selected module. See the VCPU chapter for the description of the VCPU programming language.

CDS/Deint Tab

The CDS/Deint tab contains the settings for the digital correlated double sampling filter and the deinterlacing engine. The First/Last Reset Sample and First/Last Video Sample fields control which 100 MHz ADC samples after a pixel clock are summed for the reset/reference level, and which are summed for the video level. The CDS output is the reference level minus the video level. The CDS accumulators are 32 bits, and are normalized based on the number of summed reset or video level samples. The Sample Mode setting controls whether 16 bits (Normal) or 32 bits (HDR) are passed to the deinterlacing engine.

The remaining settings configure the deinterlacing engine. The Pixels Per Tap and Lines Per Tap settings define the number of pixel and line clocks per frame per CCD output. The engine uses these values to determine when a line or frame is finished. The Frame Mode defines whether the first pixel should be written to the top of the frame (Top), the bottom of the frame (Bottom), or if the first half of the outputs should be written to the top and the second half to the bottom (Split). Big Buffers enables two 768MB frame buffers instead of the standard three 512MB buffers.

If raw/oscilloscope data for one channel should be captured in addition to normal frame data, check the Raw Enable checkbox. The Raw Channel Select field selects the raw capture channel. 1 – 4 selects a channel from the first ADC slot, 5 – 8 from the second ADC slot, etc. The Raw Start Line and Raw End Line indicate during which portion of a frame raw data should be captured. Raw Start Pixel defines when in a line raw data capture should start. Raw Samples configures how many 16 bit 100 MHz samples should be acquired per line. This is rounded up to a multiple of 1024.

The Tap Order box to the right is filled with the CCD output order, as described in the deinterlacing section.

Click the "Apply" button to load and apply the CDS/deinterlacing settings to the controller.

Image Tab

Captured images are displayed in the image tab. Move the cursor over the image to display X, Y, and pixel values in the lower right corner of the GUI. Left click to select the X and Y plot locations (visible under the Horizontal Plot and Vertical Plot tabs). Right click and drag to draw a signal statistics box, and middle click and drag to draw a noise statistics box. Statistics are displayed at the bottom of the GUI. Zoom in and out with the Zoom buttons. Adjust the image brightness and contrast with the sliders at

the bottom of the image tab, and reset to the defaults with the “Reset LUT” button. Save and load raw image data with the “Save Frame” and “Load Frame” buttons.

Plot Tabs

Left click and drag to zoom on a plot. Middle click to restore the display to the full plot. Right click and drag to pan. Check the “Average over signal area” checkbox to display an average plot of the image rows or columns encompassed by the green signal box. Save a text file of the plot data (named “hplot.txt” or “vplot.txt”) by clicking the “Save Plot” button. The PTC Plot can be used to manually acquire points for a photon transfer curve. Click the “Snap PTC” button while the controller is taking a sequence of images with constant illumination to add a signal/variance point to the PTC plot. The GUI calculates the PTC signal as the signal box mean minus the noise box mean, and the PTC variance as the variance of the difference of the last two frames. Change “PTC frames to average” to a number greater than 1 to have the GUI average the variance of multiple sets of images. The PTC plot point will be added after the requisite number of frames have been captured. Clear the PTC plot with the “Reset PTC” button.

Raw Tabs

The Raw Image, Raw Horizontal Plot, and Raw Vertical Plot tabs operate similarly to the Image and Plot tabs, but display the captured raw data from a frame if available.

ADC Tab

An ADC tab will be added to the GUI for each installed ADC module. The “Clamp High” and “Clamp Low” settings define the DC restore voltages for the positive and negative sides of the AC-coupled differential inputs. The best noise performance occurs when these are both zero, but they can be offset to use more of the ADC dynamic range. Raw ADC data should be inspected to verify that the CCD reset level is not exceeding the ADC input range. The hardware preamp gain is selected with the “Preamp Gain” control, with high gain corresponding to a 1.3 V full scale, and low gain giving a 4 V full scale. The “Apply” button applies the changes made for a specific module to the controller.

Driver Tab

A Driver tab will be added to the GUI for each installed clock driver module. Each channel can be assigned a label for user reference. The labels will also appear in the Timing States tab for the corresponding timing signals. Fast and slow slew rates are defined for each channel in volts per microsecond. Each channel can also be enabled or disabled. Disabled channels use significantly less power. The “Apply” button applies the changes made for a specific module to the controller.

Bias Tabs

An LVBias or HVBias tab will be added to the GUI for each installed bias module. Each channel can be assigned a label for user reference. The desired bias levels when CCD power is on are entered in the “Command” boxes. A current limit can be entered for the six high power channels. The measured voltage and current are displayed. When power to the CCD is off, all channels should be at about zero volts. The “Order” fields define the power up and down sequence of the biases. Order 0 biases power on first and off last, Order 1 biases power on second, etc. The high power biases can additionally be

enabled or disabled. Disabled biases draw significantly less power. The “Apply” button applies the changes made for a specific module to the controller.

Basic Examples

This section walks through some examples using the system shown in Figure 31. A single clock driver channel is connected to an ADC channel. The clock driver will simulate a CCD output, and the resulting simulated images will be captured.

To begin, turn Archon on, and click the “Connect” button to connect to the controller and fetch the system configuration. The result is a screen similar to Figure 32. The next step is to define the states for the timing core in the “Timing States” tab. Click the “Add” button. A new state named “new” appears in the “State Name” list. All of the signal states (under the “Control”, “Driver”, and “AD” tabs) default to “Keep”. In a typical script, most states will only change one or a few of the clocks at a time, and leave the rest unchanged. However, the first state should define all of the signals as a starting point. Rename the “new” state to “Reset” by double-clicking on “new” and typing “Reset”. Clear all of the “Keep” checks under the “Control” tab and leave the “INT”, “FRAME”, “LINE”, and “PIXEL” checkboxes cleared. Under the “Driver” tab, clear the “Keep” checkbox for channel 1 and enter a level of “0.0”. Under the “AD” tab clear the “Keep” checkbox and set the “Clamp” checkbox.

Continue adding states. Leave the “Keep” checkboxes set except for the signals defined in the table below.

State Name	Frame	Line	Pixel	Driver Ch 1	Clamp
Reset	Off	Off	Off	0.0	On
Idle				0.0	Off
Frame	On				
Line		On			
Pixel			On		
Clamp					On
A	Off	Off	Off	0.0	
B	Off	Off	Off	-0.25	
C	Off	Off	Off	-0.75	
D	Off	Off	Off	-1.5	
X					

A simple test script to exercise the clocks is listed below. Enter it in the “Timing Script” tab. Also set the fast slew rate for channel 1 under the “Driver” tab to 100 V / us.

```
Start:
A; X(99)
B; X(99)
C; X(99)
D; X(98)
X; GOTO Start
```

Click “Apply All” followed by “Power On”. If everything is correct, the controller will be outputting a waveform from clock driver channel 1 that goes from 0 V, to -0.25 V, to -0.75 V, to -1.5 V. It will sit at each level for 1 us (100 x 10 ns), and slew between levels at 100 V / us. An oscilloscope capture of the waveform is shown in Figure 34. Experiment with the state durations and slew rate. Changing the first line to “A: X(199)” and setting the slew rate to 5 V / us gives Figure 35.

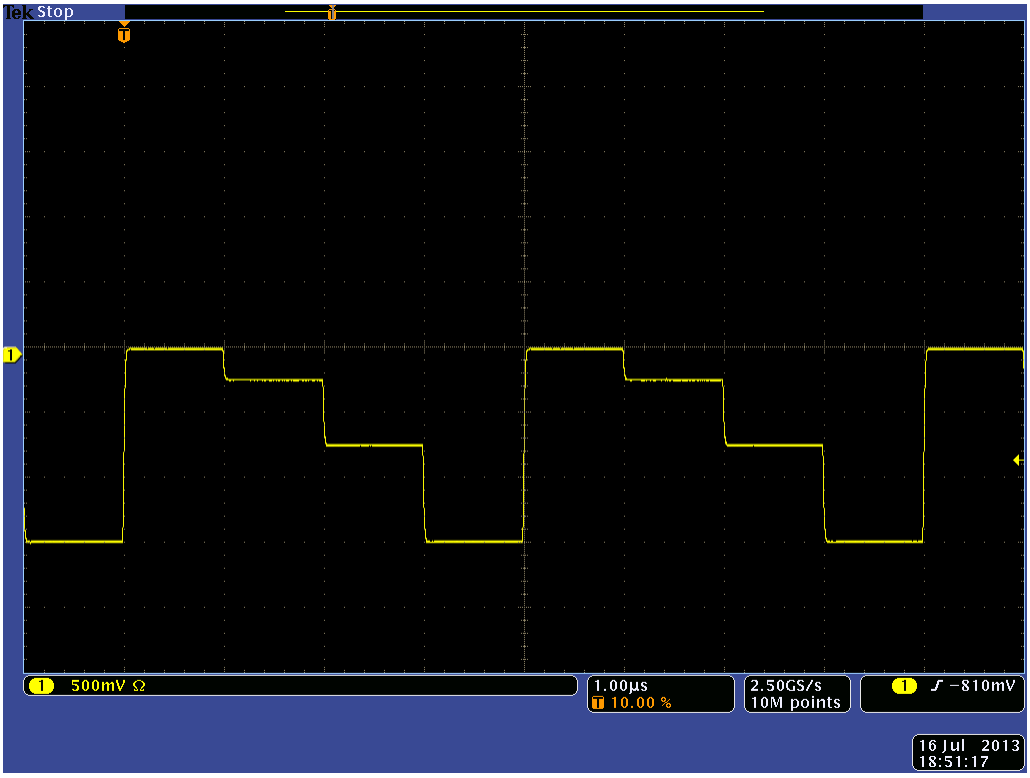


Figure 34: Test Clock Example

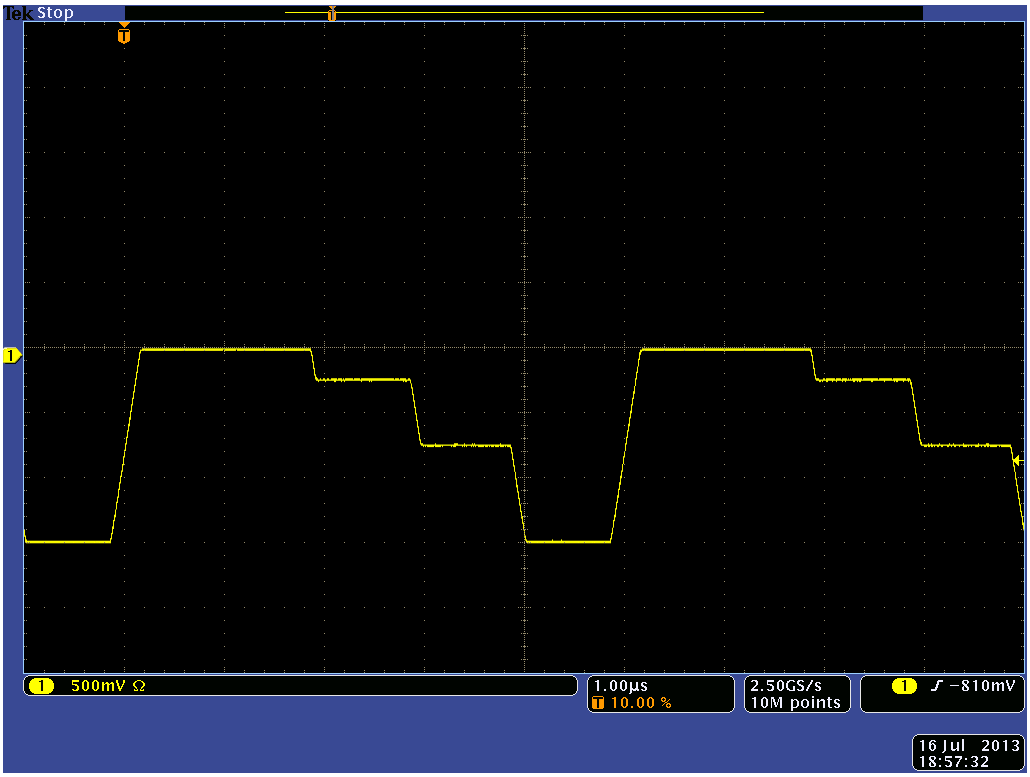


Figure 35: Modified Test Clock Example

The next script uses the previously defined states to generate a simulated 2 x 2 image.

```
Reset

# Wait for frames to be requested (Count greater than 0).
# Clamp ADC inputs while idle.
Start:
Idle; X(100)
Clamp; X(2000)
Idle; X(100)
Idle; IF !Count GOTO Start

# Start a frame
Frame

# First line
Line
Idle; CALL BlackPixel
Idle; CALL DarkPixel
Idle; X(100)
Clamp; X(2000)
Idle; X(100)
```

```
# Second line
Line
Idle; CALL MedPixel
Idle; CALL BrightPixel
Idle; X(100)
Clamp; X(2000)
Idle; X(100)

# Decrement the number of frames and return to the main loop
Idle; GOTO Start; Count--

# Individual pixel timing subroutines with various intensities
BlackPixel:
Pixel
A; X(1020)
X; RETURN BlackPixel

DarkPixel:
Pixel
A; X(509)
B; X(510)
X; RETURN DarkPixel

MedPixel:
Pixel
A; X(509)
C; X(510)
X; RETURN MedPixel

BrightPixel:
Pixel
A; X(509)
D; X(510)
X; RETURN BrightPixel
```

The initial reset sets all of the signals to known values, and leaves the DC restore clamp on. During system configuration, the timing core is held in reset, and outputs the signal levels defined by the first state in the script. Keeping the clamp on during this time prevents the AC-coupled preamps from drifting. The “Start” label is the beginning of this script’s main loop. If the “Count” parameter is zero, the script just clamps the preamps. Once “Count” is greater than zero (when the user has requested a frame), a frame is started. The system is notified that a frame is starting when the Frame signal is high at the same time as the Pixel signal (the Frame and Line signals are only sampled by the system when the Pixel signal is high). This is accomplished by the “Frame” state, which sets the Frame signal high. All of the states between “Frame” and “Pixel” have the Frame signal set to “Keep”, so it remains high. The states after the “Pixel” state all clear the Frame, Line, and Pixel signals, so subsequent states don’t unintentionally start a new line or frame.

The script next generates two lines. The first line generates a black pixel (the reset and video levels are both 0 V), followed by a low level pixel (reset level of 0 V, signal level of -0.25 V). The two pixel sequences are encoded in subroutines. In this script, each pixel takes exactly 1024 clock cycles to complete. The initial "Idle; CALL" is the first cycle, the start of pixel flag "Pixel" is the second, and the subsequent clock level states and final "X; RETURN" add the remainder to get to exactly 1024. After the two pixels per line, the preamp is clamped again. The second line emits a mid-level pixel followed by a bright pixel. After the second line is complete, the "Count" parameter is decremented, and execution jumps back to the main loop.

Before executing this script, some other settings must be adjusted. The "Count" parameter must be defined in the "Parameters" box of the "Parameters" tab, by entering "Count=1". In the "CDS / Deint" tab, set the first reset sample to "100" and the last reset sample to "400". Set the first video sample to "600" and the last video sample to "900". Leave the sample mode at "Normal" (16 bits per pixel), set "Pixels Per Tap" to "2", and "Lines Per Tap" to "2". Leave "Frame Mode" at "Top", and check the "Raw Enable" checkbox. Set "Raw Channel Select" to "1", "Raw Start Line" to "0", and "Raw End Line" to "1". Set "Raw Start Pixel" to "0", and "Raw Samples" to "2048". The effect of all of this is that the system will expect two lines of two pixels each, and will average 300 samples during the reset level and 300 samples during the video level. The system will also store two lines of raw data, which will encompass the entire image since each pixel has a duration of 1024 samples. Under "Tap Order", enter "AD1L, 1.0, 100" (only a single AD channel, starting at the left, with a digital gain of 1 and an offset of 100). Channel 1 of the clock driver should already be set to 100 V / us and enabled under the "Driver" tab. In the "AD" tab, set both clamp levels to 0 V, and select the low gain mode.

The complete configuration has now been entered. Click "Apply All", followed by "Power On". Check the "Auto Fetch" checkbox in the "System" tab. In the "Parameters" tab, click "Apply". This sets the "Count" parameter back to 1. The running script (which has been idling since "Apply All") will see Count change, emit a frame, decrement "Count", and go back to idling. The GUI will detect the newly completed frame and fetch the data. The captured image (zoomed in to show the four pixels) is shown in Figure 36. The pixel values in DN are 100, 3571, 10537, and 21003. A plot of the first raw line is shown in Figure 37. The first 1536 samples average 32768 DN, and the remainder of the line (ignoring the time for slewing) is at 29297 DN. A plot of the second raw line is in Figure 38. The first 512 samples are at 32768 DN, the next 512 samples are at 22331 DN, the next 512 samples are at 32768 DN, and the final 512 samples are at 11865 DN (again ignoring the short slewing intervals). Archon calculated the pixel values as follows:

Pixel 1: [(Reset level = average of samples 100 to 400 of line 1 = 32768) – (video level = average of samples 600 to 900 of line 1 = 32768)] x (CDS gain = 1.0) + (CDS offset = 100) = **100**.

Pixel 2: [(Reset level = average of samples 1124 to 1424 of line 1 = 32768) – (video level = average of samples 1624 to 1924 of line 1 = 29297)] x (CDS gain = 1.0) + (CDS offset = 100) = **3571**.

Pixel 3: [(Reset level = average of samples 100 to 400 of line 2 = 32768) – (video level = average of samples 600 to 900 of line 2 = 22331)] x (CDS gain = 1.0) + (CDS offset = 100) = **10537**.

Pixel 4: [(Reset level = average of samples 1124 to 1424 of line 2 = 32768) – (video level = average of samples 1624 to 1924 of line 2 = 11865)] x (CDS gain = 1.0) + (CDS offset = 100) = **21003**.

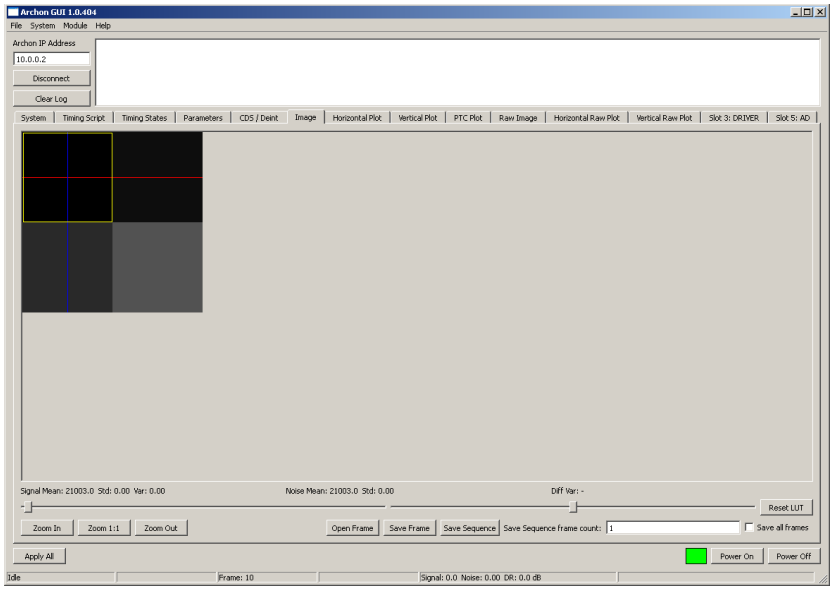


Figure 36: Example Image

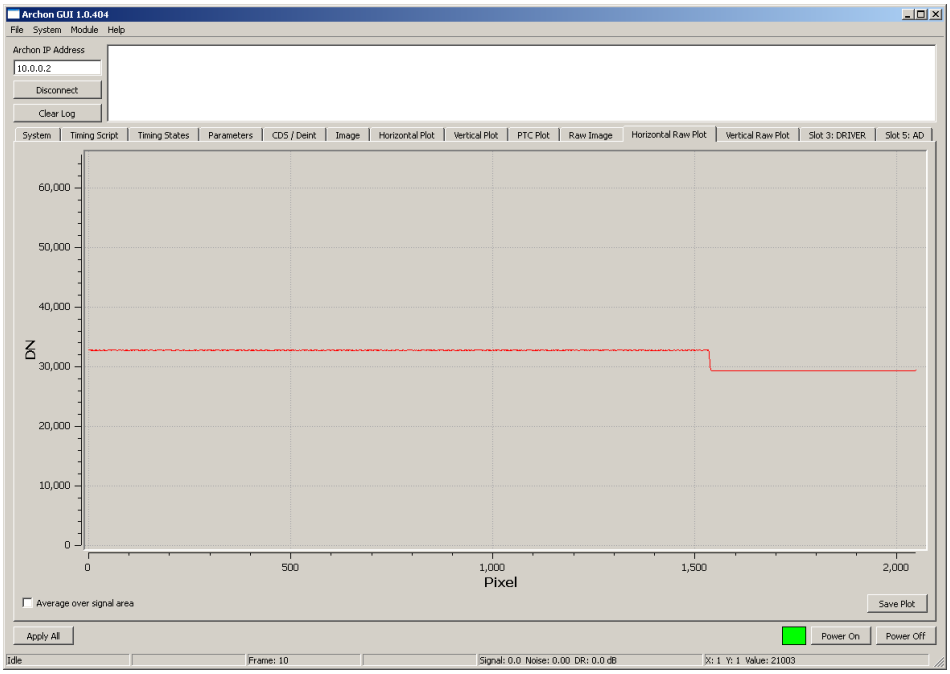


Figure 37: Raw Samples for Line 1

Figure 38: Raw Samples for Line 2

Grounding

Extracting maximum performance from low noise, scientific CCDs requires extra attention to a system's grounding configuration. Archon is designed to accommodate various topologies. By default, the chassis is connected to the earth ground pin on the power connector, and signal ground (the ground planes of the backplane and all modules) is isolated from the chassis. Typically, signal ground and earth ground are tied together at a single point to avoid loops. If signal ground and the chassis are tied together at the CCD (somewhere in the dewar, perhaps), no further action is necessary. Often, signal ground and chassis ground are tied together on a customized interface board (a custom interface board plugs into the Archon module connectors and routes all of the signals to a set of user-defined connectors for connection to the CCD). If the chassis is already earth grounded (perhaps because it's mounted to an earth grounded telescope), the Archon earth ground/chassis connection can be broken by disconnecting the green earth ground wire that goes from the power connector to the corner of the backplane. Alternatively, if there is no other connection between signal ground and earth ground, the nylon washer between the green earth ground wire and the backplane can be removed, so that the corner of the backplane becomes the single point ground.

The cabling chosen to connect to the CCD should have continuous shielding that has a low impedance connection to both the Archon chassis and the dewar, and should be isolated from signal ground. Using properly balanced and terminated differential amplifiers and cabling from the CCD outputs to the Archon AD module inputs will further improve the system's ability to reject external noise sources.

VCPU

Each module with digital I/O lines contains an embedded 100 MHz real time 16-bit CPU, called a VCPU (virtual CPU). The VCPU is programmed by including a simple textual VCPU assembly language program in the Archon configuration. Each module can have a different VCPU program loaded. The Archon controller compiles the program internally, and runs it on the VCPU. Each VCPU instruction executes deterministically in 10 ns. The VCPU is useful for implementing low level communication routines with auxiliary hardware, such as for reading an RS-232 vacuum gauge or monitoring an I2C temperature sensor.

Each digital I/O line can be configured as an input or an output, and can be controlled either by a constant in the configuration file, a timing core output, or the VCPU. Any additional necessary digital I/O interface hardware (such as an RS-232 translator, high voltage relay, etc.) is typically placed on a custom Archon interface board.

Each VCPU communicates with the higher level system through a set of 16-bit registers. There are 16 input registers that the VCPU can read from, and 16 output registers the VCPU can write to. For example, the VCPU might use an output register to store a temperature reading, or use an input register to control the pulse width of a square wave it's generating. The VCPU input registers are set using the VCPU_INREGn keys for each module. The VCPU output registers are reported by the STATUS command. The VCPU input registers and program code are loaded when an APPLYALL, APPLYMOD, or APPLYDIO command is given. The VCPU is held in reset while it's being configured, and then begins running from address 0 after the apply command completes.

Each VCPU contains 32 internal 16-bit registers for performing computations, named r0 to r1F. Programs can be 512 instructions long. Each VCPU also has a 64 entry deep call stack. Each VCPU is connected to an input and an output bus, which allows the processor to read or write to the digital I/O lines, a simple UART, or to the input and output registers described above. Zero (Z) and Carry (C) flags are set based on the results of some instructions, and can be used to conditionally execute instructions.

VCPU Programs

VCPU programs are entered under the VCPU tab, within the tab for each module. Program lines can contain the following:

- Blank lines (ignored)
- Labels: an alphanumeric label followed by a ":"
- Constants: an alphanumeric constant name followed by an "=" and then a value
- Register aliases: "ALIAS" followed by an alphanumeric alias followed by a register (r0 – r1F)
- Instructions: An instruction of the form [conditional] instruction [destination][, source]
- Comments: lines beginning with a ";"
- Inline comments: an instruction can be followed by a ";" and then a comment

An example instruction is:

```
IF Z LOAD r0, 7
```

This will load the source value of 7 into register r0 if the zero flag is set. Registers are identified as r[hex number], which can range from "r0" to "r1F". Registers can also be given more readable aliases, as shown below:

```
ALIAS r1C LoopCounter
ALIAS r1C AnotherName
LOAD r1C, 8             ; Sets r1C to 8
LOAD LoopCounter, 8     ; Sets r1C to 8
LOAD AnotherName, 8     ; Sets r1C to 8
```

Register aliases, constant names, and labels must be unique. When an instruction uses a constant, you can specify it in decimal, hexadecimal, ASCII, or as a named constant or label. Everything but ASCII constants are case-insensitive. GOTO and CALL instructions can use contants, labels, or register values as a destination, which enables the use of jump tables.

```
MyLabel:
MyConstant = 100
LOAD r0, 100            ; Sets r0 to 100
LOAD r0, 0x64           ; Also sets r0 to 100
LOAD r0, 'd'            ; Also sets r0 to 100
LOAD r0, '\n'           ; Sets r0 to newline (0xA)
LOAD r0, MyLabel        ; Sets r0 to the address of MyLabel
LOAD r0, MyConstant     ; Back to setting r0 to 100
```

VCPU Op-codes

NOP	No operation. Delays for one clock cycle.
LOAD rX, rY	Load destination rX with the value in rY.
LOAD rX, K	Load destination rX with K.
AND rX, rY	Load rX with rX AND rY, set Z if result is zero.
AND rX, K	Load rX with rX AND K, set Z if result is zero.
OR rX, rY	Load rX with rX OR rY, set Z if result is zero.
OR rX, K	Load rX with rX OR K, set Z if result is zero.
XOR rX, rY	Load rX with rX XOR rY, set Z if result is zero.
XOR rX, K	Load rX with rX XOR K, set Z if result is zero.
ADD rX, rY	Load rX with rX + rY, set Z if result is zero, set C if result > FFFF.
ADD rX, K	Load rX with rX + K, set Z if result is zero, set C if result > FFFF.

ADDCY rX, rY **ADDCY rX, K**	Load rX with rX + rY / K + C, set Z if result is zero and last Z was set, set C if result > FFFF.
SUB rX, rY	Load rX with rX - rY, set Z if result is zero, set C if result < 0.
SUB rX, K	Load rX with rX - K, set Z if result is zero, set C if result < 0.
SUBCY rX, rY **SUBCY rX, K**	Load rX with rX – rY / K - C, set Z if result is zero and last Z was set, set C if result < 0.
SL0 rX	Load rX(0) with 0, load rX(15-1) with rX(14-0), load C with rX(15), and set Z if result is zero.
SL1 rX	Load rX(0) with 1, load rX(15-1) with rX(14-0), load C with rX(15), and clear Z.
SLX rX	Load rX(15-1) with rX(14-0), load C with rX(15), and set Z if result is zero.
SLA rX	Load rX(0) with C, load rX(15-1) with rX(14-0), load C with rX(15), and set Z if result is zero.
SLB rX	Load rX(0) with B, load rX(15-1) with rX(14-0), load C with rX(15), and set Z if result is zero.
RL rX	Load rX(0) with rX(15), load rX(15-1) with rX(14-0), load C with rX(15), and set Z if result is zero.
SR0 rX	Load rX(15) with 0, load rX(14-0) with rX(15-1), load C with rX(0), and set Z if result is zero.
SR1 rX	Load rX(15) with 1, load rX(14-0) with rX(15-1), load C with rX(0), and clear Z.
SRX rX	Load rX(14-0) with rX(15-1), load C with rX(0), and set Z if result is zero.
SRA rX	Load rX(15) with C, load rX(14-0) with rX(15-1), load C with rX(0), and set Z if result is zero.
SRB rX	Load rX(15) with B, load rX(14-0) with rX(15-1), load C with rX(0), and set Z if result is zero.
RR rX	Load rX(15) with rX(0), load rX(14-0) with rX(15-1), load C with rX(0), and set Z if result is zero.
BRL rX, s	Barrel rotate rX left s times, where s is 0 to 15.
BRR rX, s	Barrel rotate rX right s times, where s is 0 to 15.

INPUT rX, rY	Load rX with the data read from input port rY.
INPUT rX, K	Load rX with the data read from input port K.
OUTPUT rX, rY	Output the value of rY to port rX.
OUTPUT rX, K	Output the value K to port rX.
GOTO rX	Execution jumps to address rX.
GOTO K	Execution jumps to address K.
CALL rX	Execution jumps to address rX, return address pushed to stack.
CALL K	Execution jumps to address K, return address pushed to stack.
RETURN	Execution jumps to address popped from stack.
LOADRETURN rX, K	Load rX with K and execution jumps to address popped from stack.
TEST rX, rY	Set Z if rX and rY is zero, set C if rX and rY has an odd number of bits set.
TEST rX, K	Set Z if rX and K is zero, set C if rX and K has an odd number of bits set.
TESTCY rX, rY	Set Z if rX and rY is zero and last Z is set, set C if rX and rY with C has an odd number of bits set.
TESTCY rX, K	Set Z if rX and K is zero and last Z is set, set C if rX and K with C has an odd number of bits set.
COMPARE rX, rY	Set Z if rX – rY is zero, set C if rX – rY < 0.
COMPARE rX, K	Set Z if rX – K is zero, set C if rX – K < 0.
COMPARECY rX, rY	Set Z if rX – rY – C is zero and last Z is set, set C if rX – rY – C < 0.
COMPARECY rX, K	Set Z if rX – K – C is zero and last Z is set, set C if rX – K – C < 0.
SETCY	Set C.
CLEARCY	Clear C.

VCPU I/O

Each VCPU communicates with the rest of the system through I/O ports using the INPUT and OUTPUT commands. There is also an input bit (IN_BIT) that can be selected for use with the SLB and SRB commands or an IF B conditional to simplify converting a serial input stream into a word. The input and output port addresses and functions follow.

Output Address	Function
0x01bb	Write to the digital outputs masked by "bb". Bit 0 of the output data is written to Digital Output 1 if bit 0 of bb is set. For example: LOAD r0, 0x0103 OUTPUT r0, 2 sets Digital Output 1 low and Digital Output 2 high.
0x020b	Write to the VCPU output register "b", which is reported in the VCPU_OUTREG fields of the STATUS command.
0x0400	Select IN_BIT. 0 = Digital Input 1, 1 = Digital Input 2, etc.
0x0800	Write to UART divider to select baud rate. 0 = 115200 baud, 1 = 57600 baud, etc.
0x1000	Write a character to the UART. The character is ignored if TX Full is set.
0x2000	Write to the UART control word. Bit 15 = UART reset. Bits 7-4 = select digital output for serial transmit. Bits 3-0 = select digital input for serial receive.

Input Address	Function
0x0000	Read digital inputs, bit 0 = Digital Input 1, etc.
0x0001	Read UART status. Bit 0 = TX Empty Bit 1 = TX Full Bit 2 = RX Empty Bit 3 = RX Full
0x0002	Read a character from the UART. The result is undefined if RX Empty is set.
0x001b	Read VCPU_INREG "b".

VCPU Sample Program

The following code is an example of a VCPU program. In this implementation, a Maxim DS18S20 I2C temperature sensor is connected to an LVXBias module. One LV bias line is set to 5V, and used to power the DS18S20 Vdd pin and to drive the LVXBias DPWR pin. The DQ pin of the DS18S20 is pulled up to 5V through a 4.7k resistor, and is connected to DIO3 which is set to be an input. DIO2 is set to be an output, and drives a MOSFET that can pull DQ to ground when DIO2 is set high.

The program sets DIO2 high to drive the I2C bus low by writing a 2 to port 0x0102. Setting DIO2 low (writing a 0 to 0x0102) allows the I2C bus to idle high. Delay subroutines are used to meet the I2C bus timing requirements specified in the DS18S20 datasheet. The program resets the I2C bus, and then commands a temperature conversion to start. After a 750 ms delay, the program reads the 16 bit result by shifting in a bit at a time from DIO3 (bit 2 of input port 0x0000). Finally, the acquired temperature data is written to VCPU output register 0 by writing to port 0x0200, so that it will be reported by the next STATUS command the system receives. The default GUI will display the raw VCPU output register data (for example, a value of 51 would correspond to a measured temperature of 25.5C). A custom GUI or other program would need to be used to properly interpret and display the results of the VCPU program.

```
; Example DS18S20 temperature polling VCPU program

ALIAS r1 Port
ALIAS r2 OutReg
ALIAS r3 Timer
ALIAS r4 Timer2
ALIAS r5 Data
ALIAS r6 Count

DOUT_PORT = 0x0102
DOUT_HIGH = 0
DOUT_LOW = 2
DIN_PORT = 0x0000
DIN_BIT = 4
OUTREG_PORT = 0x0200

Init:
LOAD Port, DOUT_PORT
OUTPUT Port, DOUT_HIGH
LOAD OutReg, OUTREG_PORT

MainLoop:

; Start conversion
CALL Reset
LOAD Data, 0xCC
CALL Write
LOAD Data, 0x44
CALL Write

; Wait for conversion to complete (750 ms)
LOAD Timer2, 750
CALL BigSleepLoop
```

```
; Fetch result
CALL Reset
LOAD Data, 0xCC
CALL Write
LOAD Data, 0xBE
CALL Write
LOAD Count, 16
ReadLoop:
SR0 Data
OUTPUT Port, DOUT_HIGH
CALL Sleep10us
OUTPUT Port, DOUT_LOW
CALL Sleep2us
OUTPUT Port, DOUT_HIGH
CALL Sleep10us
INPUT r0, DIN_PORT
TEST r0, DIN_BIT
IF NZ OR Data, 0x8000
CALL Sleep100us
SUB Count, 1
IF NZ GOTO ReadLoop

; Report result
OUTPUT OutReg, Data

GOTO MainLoop

Reset:
; Reset DS18S20
OUTPUT Port, DOUT_LOW
CALL Sleep500us
OUTPUT Port, DOUT_HIGH
CALL Sleep500us
RETURN
```

```
Write:
; Write one byte to DS18S20
LOAD Count, 8
WriteLoop:
OUTPUT Port, DOUT_HIGH
CALL Sleep10us
OUTPUT Port, DOUT_LOW
CALL Sleep2us
SR0 Data
IF C OUTPUT Port, DOUT_HIGH
IF NC OUTPUT Port, DOUT_LOW
CALL Sleep100us
OUTPUT Port, DOUT_HIGH
SUB Count, 1
IF NZ GOTO WriteLoop
RETURN

; Sleep subroutines
Sleep2us:
LOAD Timer, 99
GOTO SleepLoop

Sleep10us:
LOAD Timer, 499
GOTO SleepLoop

Sleep100us:
LOAD Timer, 4999
GOTO SleepLoop

Sleep500us:
LOAD Timer, 24999
GOTO SleepLoop

Sleep1ms:
LOAD Timer, 49999

; Shared sleep loop
SleepLoop:
SUB Timer, 1
IF NZ GOTO SleepLoop
RETURN

; Big sleep loop (1ms per loop)
BigSleepLoop:
CALL Sleep1ms
SUB Timer2, 1
IF NZ GOTO BigSleepLoop
RETURN
```

Appendix A: Test Clock Configuration File

This is a complete listing of the GUI configuration file for the test clock example.

```
[SYSTEM]
BACKPLANE_ID=0024498A715E301C
BACKPLANE_REV=2
BACKPLANE_TYPE=1
BACKPLANE_VERSION=1.0.408
MOD1_ID=0000000000000000
MOD1_REV=0
MOD1_TYPE=0
MOD1_VERSION=0.0.0
MOD2_ID=0000000000000000
MOD2_REV=0
MOD2_TYPE=0
MOD2_VERSION=0.0.0
MOD3_ID=0131AB38BE6AFA98
MOD3_REV=2
MOD3_TYPE=1
MOD3_VERSION=1.0.379
MOD4_ID=0000000000000000
MOD4_REV=0
MOD4_TYPE=0
MOD4_VERSION=0.0.0
MOD5_ID=01399CFCC62F0A20
MOD5_REV=0
MOD5_TYPE=2
MOD5_VERSION=1.0.285
MOD6_ID=0000000000000000
MOD6_REV=0
MOD6_TYPE=0
MOD6_VERSION=0.0.0
MOD7_ID=0000000000000000
MOD7_REV=0
MOD7_TYPE=0
MOD7_VERSION=0.0.0
MOD8_ID=0000000000000000
MOD8_REV=0
MOD8_TYPE=0
MOD8_VERSION=0.0.0
MOD9_ID=0000000000000000
MOD9_REV=0
MOD9_TYPE=0
MOD9_VERSION=0.0.0
MOD10_ID=0000000000000000
MOD10_REV=0
MOD10_TYPE=0
MOD10_VERSION=0.0.0
MOD11_ID=0000000000000000
```

```
MOD11_REV=0
MOD11_TYPE=0
MOD11_VERSION=0.0.0
MOD12_ID=0000000000000000
MOD12_REV=0
MOD12_TYPE=0
MOD12_VERSION=0.0.0
MOD_PRESENT=14
POWER_ID=0000006754D4

[CONFIG]
CONSTANT0=
CONSTANTS=1
FRAMEMODE=0
LINE0=Start:
LINE1="A; X(99)"
LINE2="B; X(99)"
LINE3="C; X(99)"
LINE4="D; X(98)"
LINE5="X; GOTO Start"
LINE6=
LINECOUNT=1
LINES=7
MOD3\ENABLE1=1
MOD3\ENABLE2=0
MOD3\ENABLE3=0
MOD3\ENABLE4=0
MOD3\ENABLE5=0
MOD3\ENABLE6=0
MOD3\ENABLE7=0
MOD3\ENABLE8=0
MOD3\FASTSLEWRATE1=100
MOD3\FASTSLEWRATE2=1
MOD3\FASTSLEWRATE3=1
MOD3\FASTSLEWRATE4=1
MOD3\FASTSLEWRATE5=1
MOD3\FASTSLEWRATE6=1
MOD3\FASTSLEWRATE7=1
MOD3\FASTSLEWRATE8=1
MOD3\LABEL1=
MOD3\LABEL2=
MOD3\LABEL3=
MOD3\LABEL4=
MOD3\LABEL5=
MOD3\LABEL6=
MOD3\LABEL7=
MOD3\LABEL8=
MOD3\SLOWSLEWRATE1=1
MOD3\SLOWSLEWRATE2=1
MOD3\SLOWSLEWRATE3=1
```

```
MOD3\SLOWSLEWRATE4=1
MOD3\SLOWSLEWRATE5=1
MOD3\SLOWSLEWRATE6=1
MOD3\SLOWSLEWRATE7=1
MOD3\SLOWSLEWRATE8=1
MOD5\CLAMPHIGH=0.0
MOD5\CLAMPLOW=0.0
MOD5\PREAMPGAIN=0
PARAMETERS=0
PIXELCOUNT=1
RAWENABLE=0
RAWENDLINE=0
RAWSAMPLES=0
RAWSEL=0
RAWSTARTLINE=0
RAWSTARTPIXEL=0
SAMPLEMODE=0
SHD1=0
SHD2=0
SHP1=0
SHP2=0
STATE0\CONTROL="0,0"
STATE0\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE0\MOD5="1,0"
STATE0\NAME=Reset
STATE1\CONTROL="0,F"
STATE1\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE1\MOD5="0,0"
STATE1\NAME=Idle
STATE2\CONTROL="2,D"
STATE2\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE2\MOD5="0,1"
STATE2\NAME=Frame
STATE3\CONTROL="4,B"
STATE3\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE3\MOD5="0,1"
STATE3\NAME=Line
STATE4\CONTROL="8,7"
STATE4\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE4\MOD5="0,1"
STATE4\NAME=Pixel
STATE5\CONTROL="0,F"
STATE5\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE5\MOD5="1,0"
STATE5\NAME=Clamp
STATE6\CONTROL="0,1"
STATE6\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE6\MOD5="0,1"
STATE6\NAME=A
STATE7\CONTROL="0,1"
```

```
STATE7\MOD3="-0.25,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE7\MOD5="0,1"
STATE7\NAME=B
STATE8\CONTROL="0,1"
STATE8\MOD3="-0.75,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE8\MOD5="0,1"
STATE8\NAME=C
STATE9\CONTROL="0,1"
STATE9\MOD3="-1.5,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE9\MOD5="0,1"
STATE9\NAME=D
STATE10\CONTROL="0,F"
STATE10\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE10\MOD5="0,1"
STATE10\NAME=X
STATES=11
TAPLINES=0
TRIGOUTFORCE=0
TRIGOUTINVERT=0
TRIGOUTLEVEL=0
TRIGOUTPOWER=0
```

Appendix B: Basic Example Configuration File

This is a complete listing of the configuration file used to generate the sample 2 x 2 images.

```
[SYSTEM]
BACKPLANE_ID=0024498A715E301C
BACKPLANE_REV=2
BACKPLANE_TYPE=1
BACKPLANE_VERSION=1.0.408
MOD1_ID=0000000000000000
MOD1_REV=0
MOD1_TYPE=0
MOD1_VERSION=0.0.0
MOD2_ID=0000000000000000
MOD2_REV=0
MOD2_TYPE=0
MOD2_VERSION=0.0.0
MOD3_ID=0131AB38BE6AFA98
MOD3_REV=2
MOD3_TYPE=1
MOD3_VERSION=1.0.379
MOD4_ID=0000000000000000
MOD4_REV=0
MOD4_TYPE=0
MOD4_VERSION=0.0.0
MOD5_ID=01399CFCC62F0A20
MOD5_REV=0
MOD5_TYPE=2
MOD5_VERSION=1.0.285
MOD6_ID=0000000000000000
MOD6_REV=0
MOD6_TYPE=0
MOD6_VERSION=0.0.0
MOD7_ID=0000000000000000
MOD7_REV=0
MOD7_TYPE=0
MOD7_VERSION=0.0.0
MOD8_ID=0000000000000000
MOD8_REV=0
MOD8_TYPE=0
MOD8_VERSION=0.0.0
MOD9_ID=0000000000000000
MOD9_REV=0
MOD9_TYPE=0
MOD9_VERSION=0.0.0
MOD10_ID=0000000000000000
MOD10_REV=0
MOD10_TYPE=0
MOD10_VERSION=0.0.0
MOD11_ID=0000000000000000
```

```
MOD11_REV=0
MOD11_TYPE=0
MOD11_VERSION=0.0.0
MOD12_ID=0000000000000000
MOD12_REV=0
MOD12_TYPE=0
MOD12_VERSION=0.0.0
MOD_PRESENT=14
POWER_ID=0000006754D4

[CONFIG]
CONSTANT0=
CONSTANTS=1
FRAMEMODE=0
LINE0=Reset
LINE1=
LINE2=# Wait for frames to be requested (Count greater than 0).  Clamp
ADC inputs while idle.
LINE3=Start:
LINE4="Idle; X(100)"
LINE5="Clamp; X(2000)"
LINE6="Idle; X(100)"
LINE7="Idle; IF !Count GOTO Start"
LINE8=
LINE9=# Start a frame
LINE10=Frame
LINE11=
LINE12=# First line
LINE13=Line
LINE14="Idle; CALL BlackPixel"
LINE15="Idle; CALL DarkPixel"
LINE16="Idle; X(100)"
LINE17="Clamp; X(2000)"
LINE18="Idle; X(100)"
LINE19=
LINE20=# Second line
LINE21=Line
LINE22="Idle; CALL MedPixel"
LINE23="Idle; CALL BrightPixel"
LINE24="Idle; X(100)"
LINE25="Clamp; X(2000)"
LINE26="Idle; X(100)"
LINE27=
LINE28=# Decrement the number of frames and return to the main loop
LINE29="Idle; GOTO Start; Count--"
LINE30=
LINE31=# Individual pixel timing subroutines with various intensities
LINE32=BlackPixel:
LINE33=Pixel
LINE34="A; X(1020)"
```

```
LINE35="X; RETURN BlackPixel"
LINE36=
LINE37=DarkPixel:
LINE38=Pixel
LINE39="A; X(509)"
LINE40="B; X(510)"
LINE41="X; RETURN DarkPixel"
LINE42=
LINE43=MedPixel:
LINE44=Pixel
LINE45="A; X(509)"
LINE46="C; X(510)"
LINE47="X; RETURN MedPixel"
LINE48=
LINE49=BrightPixel:
LINE50=Pixel
LINE51="A; X(509)"
LINE52="D; X(510)"
LINE53="X; RETURN BrightPixel"
LINE54=
LINECOUNT=2
LINES=55
MOD3\ENABLE1=1
MOD3\ENABLE2=0
MOD3\ENABLE3=0
MOD3\ENABLE4=0
MOD3\ENABLE5=0
MOD3\ENABLE6=0
MOD3\ENABLE7=0
MOD3\ENABLE8=0
MOD3\FASTSLEWRATE1=100
MOD3\FASTSLEWRATE2=1
MOD3\FASTSLEWRATE3=1
MOD3\FASTSLEWRATE4=1
MOD3\FASTSLEWRATE5=1
MOD3\FASTSLEWRATE6=1
MOD3\FASTSLEWRATE7=1
MOD3\FASTSLEWRATE8=1
MOD3\LABEL1=
MOD3\LABEL2=
MOD3\LABEL3=
MOD3\LABEL4=
MOD3\LABEL5=
MOD3\LABEL6=
MOD3\LABEL7=
MOD3\LABEL8=
MOD3\SLOWSLEWRATE1=1
MOD3\SLOWSLEWRATE2=1
MOD3\SLOWSLEWRATE3=1
MOD3\SLOWSLEWRATE4=1
```

```
MOD3\SLOWSLEWRATE5=1
MOD3\SLOWSLEWRATE6=1
MOD3\SLOWSLEWRATE7=1
MOD3\SLOWSLEWRATE8=1
MOD5\CLAMPHIGH=0.0
MOD5\CLAMPLOW=0.0
MOD5\PREAMPGAIN=0
PARAMETER0="Count=1"
PARAMETERS=1
PIXELCOUNT=2
RAWENABLE=1
RAWENDLINE=1
RAWSAMPLES=2048
RAWSEL=0
RAWSTARTLINE=0
RAWSTARTPIXEL=0
SAMPLEMODE=0
SHD1=600
SHD2=900
SHP1=100
SHP2=400
STATE0\CONTROL="0,0"
STATE0\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE0\MOD5="1,0"
STATE0\NAME=Reset
STATE1\CONTROL="0,F"
STATE1\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE1\MOD5="0,0"
STATE1\NAME=Idle
STATE2\CONTROL="2,D"
STATE2\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE2\MOD5="0,1"
STATE2\NAME=Frame
STATE3\CONTROL="4,B"
STATE3\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE3\MOD5="0,1"
STATE3\NAME=Line
STATE4\CONTROL="8,7"
STATE4\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE4\MOD5="0,1"
STATE4\NAME=Pixel
STATE5\CONTROL="0,F"
STATE5\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE5\MOD5="1,0"
STATE5\NAME=Clamp
STATE6\CONTROL="0,1"
STATE6\MOD3="0.0,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE6\MOD5="0,1"
STATE6\NAME=A
STATE7\CONTROL="0,1"
```

```
STATE7\MOD3="-0.25,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE7\MOD5="0,1"
STATE7\NAME=B
STATE8\CONTROL="0,1"
STATE8\MOD3="-0.75,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE8\MOD5="0,1"
STATE8\NAME=C
STATE9\CONTROL="0,1"
STATE9\MOD3="-1.5,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE9\MOD5="0,1"
STATE9\NAME=D
STATE10\NAME=X
STATE10\CONTROL="0,F"
STATE10\MOD3=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE10\MOD5="0,1"
STATES=11
TAPLINE0="AD1L, 1.0, 100"
TAPLINES=1
TRIGOUTFORCE=0
TRIGOUTINVERT=0
TRIGOUTLEVEL=0
TRIGOUTPOWER=0
```