The hardware device-driver interface of the CCB

Martin Shepherd, California Institute of Technology

November 27, 2002

This page intentionally left blank.

Abstract

This document details the requirements of the PCI based interface between the FPGA and the Linux device driver used by the Caltech Continuum Backend (CCB).

Contents

List of Figures

1 A definition of terms

The following terms are used throughout this document. It is recommended that in other documents that describe both the hardware and software, these terms be prefixed with the word *hardware* when referring to the FPGA.

- **A sample**
  A single A/D sample.
- **A measurement**
  The group of samples added within a single phase-switch state of a single cycle.
- **A cycle**
  A single phase-switch cycle is composed of between 1 and 32 samples, with each sample individually programmable to have 1 of 4 possible phase switch states.
- **integration period**
  The integer number of cycles during which each integration is accumulated before being passed to the device driver.
- **An integration**
  The data accumulated for a single phase switch state during an integration period.
- **A scan**
  A group of one or more integrations associated with a single set of scan-specific configuration parameters.

2 Data-type conventions

2.1 Integer parameters

Integer valued parameters presented over the PCI bus will be unsigned and stored in little-endian byte order.

2.2 Boolean parameters

Boolean registers presented over the PCI bus will be represented by single bits in a larger data-type. The ordering of the bits within these data-types is defined such that performing a logical shift right by 1 bit, results in the successively next least significant bit appearing on the carry line of the shift register. The meanings of the values of these bits are as follows:

	0	1
Boolean	false	true
Switch state	off	on

Note that when referring to phase-switches, **off** corresponds to when the phase-shifter isn't inserting any phase-shift into the signal path, and **on** means that it is inserting a 180° phase shift.

3 The PCI interface between the computer and the FPGA

3.1 Per scan configuration registers

The following registers can be written at any time, but their contents must be ignored by the FPGA until the next "start scan" command is received.

Description	Qty	Units	Default	Range
Sample interval	1	0.1μs	250	1-65535
Samples per cycle	1	1 sample	1	1-32
The state of phase switch A	32	boolean	0	0 or 1
The state of phase switch B	32	boolean	0	0 or 1
Phase switch update flag	32	boolean	0	0 or 1
Integration period	1	1 cycle	40	1-65535
Phase-switch blanking interval	1	0.1μs	0	1-255
Integrator-reset blanking interval	1	0.1μs	0	1-255
Calibration diode settling time	1	0.1μs	0	1-$2^{32}-1$

3.1.1 Detailed descriptions

- **Sample interval**

  This sets the time taken per A/D sample. Although this is configurable, the analog electronics preceding the ADC will be designed for a sampling interval of 25μs, so longer intervals will likely result in the A/D saturating, and much shorter intervals may result in incomplete phase-switching.

  Note that this interval must always be greater than the sum of the blanking intervals, since this interval subtracts from the start of the sample interval. It is the responsibility of the device driver to enforce this.

- **Samples per cycle**

The states of the phase-switch within each stage of the phase-switch cycle are configured via a 32 stage state-machine, cycled every time a new sample is taken by the A/D. This register specifies how many of these 32 states are actually to be used.

Note that while the direct specification of a 32-state state machine allows a great deal of flexibility, in practice the states will be filled by the device driver from much higher level configuration parameters, such as the number of samples per measurement, the selection of one of 3 phase-switching modes (ie. no phase-switching, 1-diode phase-switching, 2-diode phase-switching), and the specified initial states of the phase switches at the start of each cycle.

- **The state of phase switch A**

  Each bit of this register specifies the state of phase-switch A of both radiometers at one stage of the phase-switching state machine. This is designed to be loaded into a 32-bit shift register at the start of each phase-switch cycle, then right shifted at the start of each new sample, to determine the required state of the phase switches in that sample. The number of shifts required to complete a cycle is determined by the "samples per cycle" register described above. Note that the values in this register are only used to drive the state of the phase switch when the corresponding "phase-switch update flag" is set.

- **The state of phase switch B**

  This register is the equivalent of the "Phase switch A states" register for phase-switch B, and is interpreted identically.

- **Phase switch update flag**

  This register is right shifted at the same time as the above two registers with each successive bit and determines both when to instantiate the values in these registers and when to include a phase-switch transition delay.

- **Integration period**

  This is the integer number of complete phase-switch cycles to integrate per phase switch state before presenting the data to the computer. It was decided at the preliminary design review that 1ms is the minimum integration time that needs to be supported. In practice, shorter times may also be supported, depending on the speed of the computer the bandwidth of the PCI bus, and the interface to Ygor.

- **Phase switch blanking interval**

  The sum of this interval and the integrator reset interval specify how long to stop integrating after initiating a phase-switch transition. The choice of this number is determined by the settling time of the phase switches. This interval is subtracted from the beginning of just those sample intervals that follow phase-switch transitions.

- **Integrator reset blanking interval**

  This specifies how much time will be needed to reset the analogue integrators to zero. The choice of this number is determined by the current-handling capacity of both the

capacitors and the FET switches that are used to discharge them, and is anticipated to be around 0.5μs. This interval is subtracted from the start of the A/D acquisition time of all samples.

- **Calibration diode settling time**

  The calibration diodes are switched at the start of integration periods only when the "Cal diode states" configuration register contains values that differ from the current cal-diode states. Whenever this happens, the state machine should command the new cal-diode switch states, then discard the number of A/D samples specified by this register, before starting the first cycle of the next integration period.

  Note that the 32-bit range of this register is bigger than that of any other timer. This is due to uncertainties over how long the diodes will actually take to settle to a level suitable for astronomical measurements, versus what the manufacturer claims.

3.2 Per integration-period configuration registers

The following registers are re-examined at the start of each new integration period, at which point they are copied into the internal working registers that affect the operation of the hardware.

Description	Qty	Units	Default	Range
Cal diode states	2	boolean	0	0 or 1
Cal diode update flag	1	boolean	0	0 or 1

3.2.1 Detailed descriptions

- **Cal diode states**

  The 2 least-significant bits of this register specify the states that the noise diodes should be set to at the start of the next integration period, if the cal-diode-update flag is set. A value of 1 means turn the diode on; a value of 0 means turn it off.

- **Cal diode update flag**

  This tells the hardware both whether to update the cal-diode states to match the current values in the above cal-diode-states register, and whether to accomodate this with a delay equal to the duration specified by the cal-diode-settling-time register.

3.3 Control register

This register is a 32-bit register who's individual bits set flags in the hardware. The first thing that the FPGA must do after its program is loaded from EPROM, is to zero the bits

in this register. It should then wait for the driver to set the enable-interrupts bit in this register before starting its state machine. This allows the driver to initialize the remaining registers to default values before the first scan starts.

Bit	Description	Effect
0	Start scan	Start a new scan at the end of the current sample
1	Stop scan	Start a new scan at the end of the current integration
2	Reload FPGA	Immediately reload the FPGA program from EPROM
3	Enable interrupts	Enable 1-PPS and integration interrupts
4	Drive phase-switch A	Drive phase-switch A's control line
5	Drive phase-switch B	Drive phase-switch B's control line
6	Drive cal-diode A	Drive cal-diode A's control line
7	Drive cal-diode B	Drive cal-diode B's control line

The operations listed in the above table are all asserted by setting the corresponding bits to 1. The response of the hardware to each of these bits is detailed below.

3.3.1 Detailed descriptions

- **Start scan**

  When the current ADC sample has completed, wait for the next rising edge of the 1-PPS signal, then read the current values of the per-scan and per-integration configuration registers into internal configuration registers, issue an integration interrupt to tell the driver that it should now send the integration parameters for the 2nd integration of the scan, then start the first integration of the scan.

  This will be used by the driver as follows. On receiving a start-scan command from the manager, the driver will wait for the next integration interrupt. On receiving this interrupt, it will write both the configuration parameters of the new scan and those of the first integration of the new scan to the corresponding hardware configuration registers. It won't bother reading the latest integrations on receiving this interrupt, and will continue to discard future integrations until the new scan starts. Next, if the start-scan request from the manager was received more than a second in advance of the particular 1-PPS time that the manager says that it wants the scan to start on, the driver will first wait for the 1-second tick that precedes the desired one. Then it will send a start-scan command to the hardware, to be executed on the next 1-PPS. The hardware will see the start-scan request on finishing the current sample interval, and then start waiting for the receipt of the next 1-PPS. When this 1-PPS signal arrives, the hardware will copy the new per-scan and per-integration parameters into its internal working registers, send an integration interrupt to tell the driver to send the integration parameters for the 2nd integration of the scan, then start the first integration. At the same time, when the driver receives the 1-PPS interrupt that signifies the start of the scan, it will reenable the reading of integrations.

- **Stop scan**

  When the current integration has completed, read the current values of the per-scan and per-integration configuration registers into internal working registers, send an integration interrupt to tell the driver to write the configuration parameters of the 2nd integration of the scan, then start the first integration of the scan.

  Note that unlike the start-scan command, the start of a new scan by this command is not synchronized with the 1-PPS signal. This is because scans started by this command are only used for monitoring, which have no need for time synchronization. Furthermore, since in general users will be using this command for quick experiments with new scan-parameters (via a GUI), they won't want to have to wait for a full second to see the results of each experiment.

- **Reload FGPA**

  This command will be issued by the driver at startup, and can also be used during initial debugging to reload the FPGA program from EPROM after a crash. After a reload, the configuration parameters should all be set to the defaults listed in the above register tables, and the first of a new scan of integration periods should be initiated using these values.

- **Enable interrupts**

  At startup, 1-PPS and integration interrupts will be disabled. This command is used to enable them once the driver is in a fit state to respond to them. Subsequently setting it to zero should simply stop interrupts being generated, without affecting the rest of the state machine.

- **Drive phase-switch A**

  This turns on the output control line which drives phase-switch A. Note that turning this off must not disable the part of the phase-switching state machine which controls phase-switch A. It must continue to switch as though output was on.

  The purpose of this register and the next three registers is both to allow the same backend to be used with instruments with differing number of phase switches and to allow other backends to take control of the frontend without interference from ours.

- **Drive phase-switch B**

  This is the equivalent of the "drive phase-switch A" register for phase switch B.

- **Drive cal-diode A**

  This turns on the output control line which drives calibration diode A. Note that the state machine which controls the signal going to this output must continue to function irrespective of whether the output control line is on or off.

- **Drive cal-diode B**

  This is the equivalent of the "drive cal-diode A" register for calibration diode B.

3.4 Diagnostic monitoring registers

These registers, which will be stored in PCI DMA memory, must be updated by the hardware just before the integration interrupt is sent. The device driver must therefore read them out on receipt of this interrupt. In order not to hold up the start of the next integration, the hardware must initiate measurements of these values sufficiently far advance of the end of the integration that the resulting values can simply be copied from internal cache registers to DMA memory at the end of each integration. To ensure this, the flow diagram at the end of this document initiates new measurements just before starting the first scan of each integration. The A/D converters and associated readout circuits thus have a whole integration period (ie. on the order of around 1ms) to measure and cache new values.

Description	Qty	Units	Range
Power supply voltages	N/A	mv	N/A
A/D converter outputs	16	A/D sample	$0 \cdots 2^{32} - 1$

The ordering of the array of 16 A/D converter outputs will be as follows.

Array Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Detector	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Band	1 . . .		2 . . .		3 . . .		4 . . .		1 . . .		2		3		4	
Radiometer	1								2							

3.4.1 Detailed descriptions

- **Power supply voltage**

  This will record the voltages of all power supplies in the system.

- **A/D converter outputs**

  The last values read from the 16 A/D converters. This will mainly be used to check for failing A/D converters and A/D saturation.

3.5 Data output via DMA

PCI direct memory access (DMA) will be used to transfer integrations to the CPU. These locations should be updated by the FPGA at the end of each new integration, before generating the integration interrupt. The driver will read them ASAP when the interrupt is received.

Description	Qty	Units	Range
An integrated data value	64	–	$0 \cdots 2^{32} - 1$
Overflow mask	64	boolean	0 or 1

The overflow mask uses 64 bits of a double-word aligned array of 8 bytes to record which of the 64 data values suffered overflows during integration. Bits will be numbered starting at zero with the least significant bit of the highest memory location, and ending at 63 with the most significant bit of the lowest memory location.

The array of integrations will be treated as an array of double-word aligned values, numbered, starting at zero, with the element with the lowest memory location.

The order of assignment of the 64 values and their corresponding overflow bits to array indexes and bit indexes will be as follows.

Array index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Integration	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Detector	1				2				1				2				1	
Band	1								2								3	
Radiometer	1																	

3.6 Interrupts

The following interrupts will share a single interrupt line.

Description	When to send the interrupt
Integration interrupt	Just before reading the per-integration configuration registers
1-PPS interrupt	On the rising edges of the 1-PPS signal

3.6.1 Detailed descriptions

- **Integration interrupt**

  This interrupt serves two purposes.

  1. Except for the first interrupt of a scan, it tells the driver that the results of the last integration period have just been written to the DMA memory area, and that the driver should be read them ASAP.
  2. It also tells the driver that the PCI per-integration configuration registers for the integration that is now being started, have been safely copied into internal working registers, and that the driver should now overwrite these PCI registers with the configuration of the integration that will follow this one.

Note that because integration configuration parameters are loaded one integration period in advance, the driver will be expected to update these parameters at the start of a new scan, so when the scan is starting, the FPGA must send an integration interrupt after reading these parameters but before having done a single integration, so that the driver is told when to send the parameters for the second integration of the scan. See the state machine diagram at the end of this document to see how this comes for free with judicious ordering of states.

In more detail, at the end of each integration period the integrations, overflow registers and monitoring data should be transfered to the DMA memory area. Next the per-integration configuration options of the integration that is about to be started, should be loaded from the PCI registers into internal working registers. Next the integration interrupt-sent register should be set to 1, and finally an interrupt should be generated to tell the device driver to read-out the data and update the integration configuration parameters for the integration that follows the one that is being started.

Note the importance of sending the interrupt *after* reading the integration configuration registers, to avoid a race condition between the driver and the hardware.

- **1-PPS interrupt**

  Whenever a rising edge of the 1-PPS signal is seen, the 1-PPS interrupt-sent register should be set to 1, and an interrupt should be generated to tell the device driver to synchronize its clock.

3.7 Interrupt registers

Description	Function
Integration interrupt-sent	Tells the driver that an integration interrupt was sent
1-PPS interrupt-sent	Tells the driver that a 1-PPS interrupt was sent

3.7.1 Detailed descriptions

- **Integration interrupt-sent**

  Whenever an integration interrupt is generated, this register must be set to non-zero. The device-driver's interrupt handler will reset it to zero. The function of this register is both to provide a way to identify and ignore false interrupts, and also a way, in combination with the 1-PPS interrupt-sent register, to allow sharing of a single interrupt line.

- **1-PPS interrupt-sent**

  Whenever a 1-PPS interrupt is generated, this register must be set to non-zero. It will be reset to zero by the device-driver's interrupt handler.

4 State diagrams

The following flow diagrams illustrate the textual specification written above. They indicate how the hardware must appear to behave, as seen at the device-driver and receiver interfaces. The designer of the FPGA is of course free to implement it as he sees fit, provided that the result acts like a black box who's external interfaces mimic this behavior. The need for such a rigorous specification at this point is dictated by the involvement of multiple groups, all needing to design to a single consistent set of interfaces. Presenting such a detailed specification also facilitates the removal of bugs and the resolution of misunderstandings in the intended behavior before any hardware or software is designed. To be useful for this purpose, it is essential that this specification remain as static as possible, so once all parties have agreed to this specification, it will be frozen, and thereafter, changes to it will be discouraged and require formal justification, agreed upon by all parties. Without such agreement, any departures from this specification, or any failures of the device driver or receiver to work within the constraints of this behavior, will be considered as flaws to be fixed by their respective implementors.

Note that in terms of operation ordering and timing, the diagrams are more rigorous than the explanatory text in the rest of this document, and for this purpose the diagrams should be taken as the primary reference. External data representations on the other hand, are described more rigorously in the text.

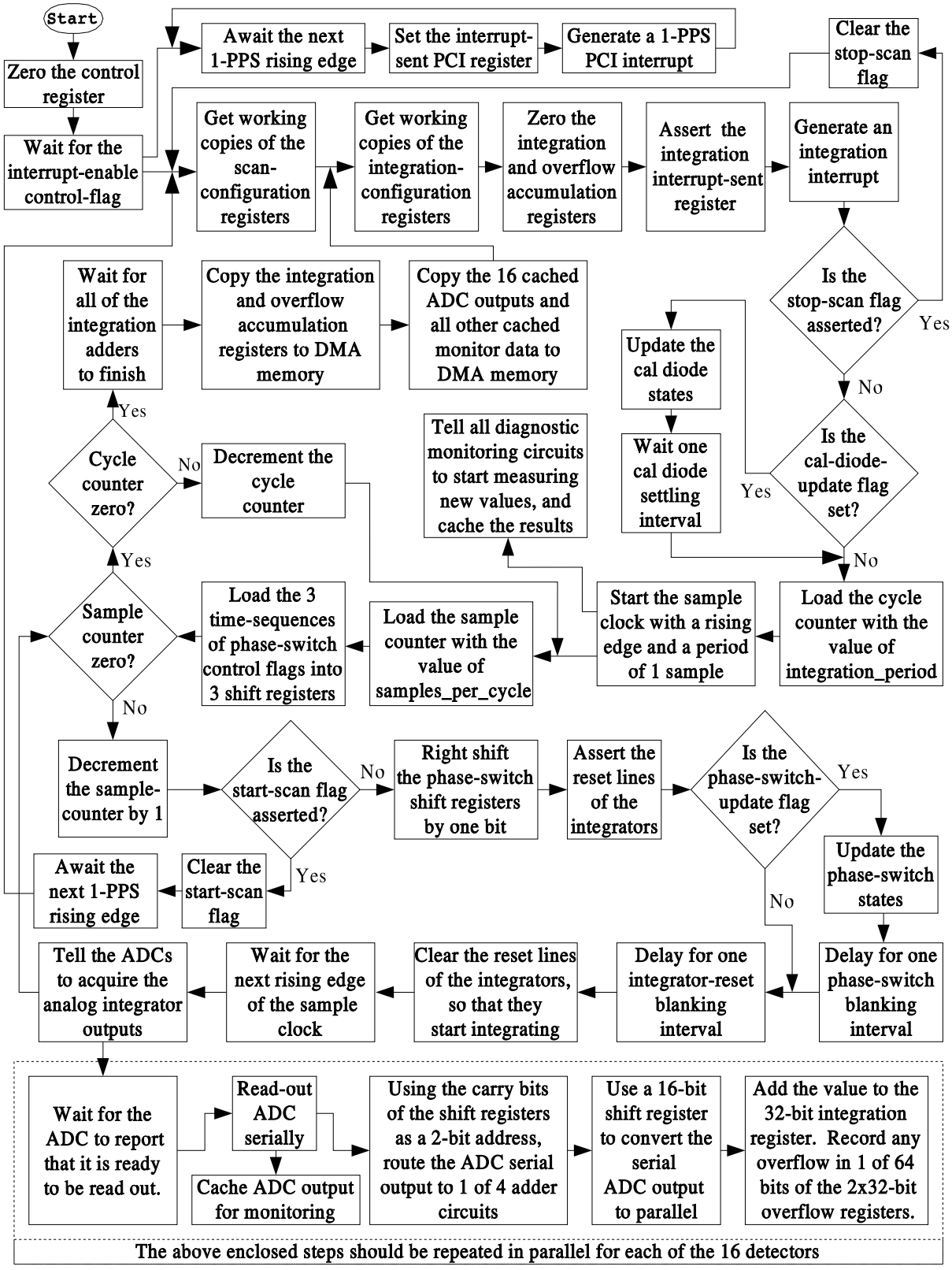

Figure 1: The life cycle of a scan in the hardware