TimePix Kintex Platform System Manual

Rick Raffanti

Table of Contents

System Overview 2

Hardware Design

- **ASIC Board** 3
- **Interface Board** 4
- **ZDOK Adapter** 5
- **Voltage Regulator Board** 6
- **ROACH Board** 7

Firmware Design

- **Interface Board** 8
- **ROACH Board** 11

Interfaces

- **TimePix DashBoard GUI** 13
- **C Language Library** 16
- **PC ←→ Roach** 18
- **Roach ←→ Interface Board** 26

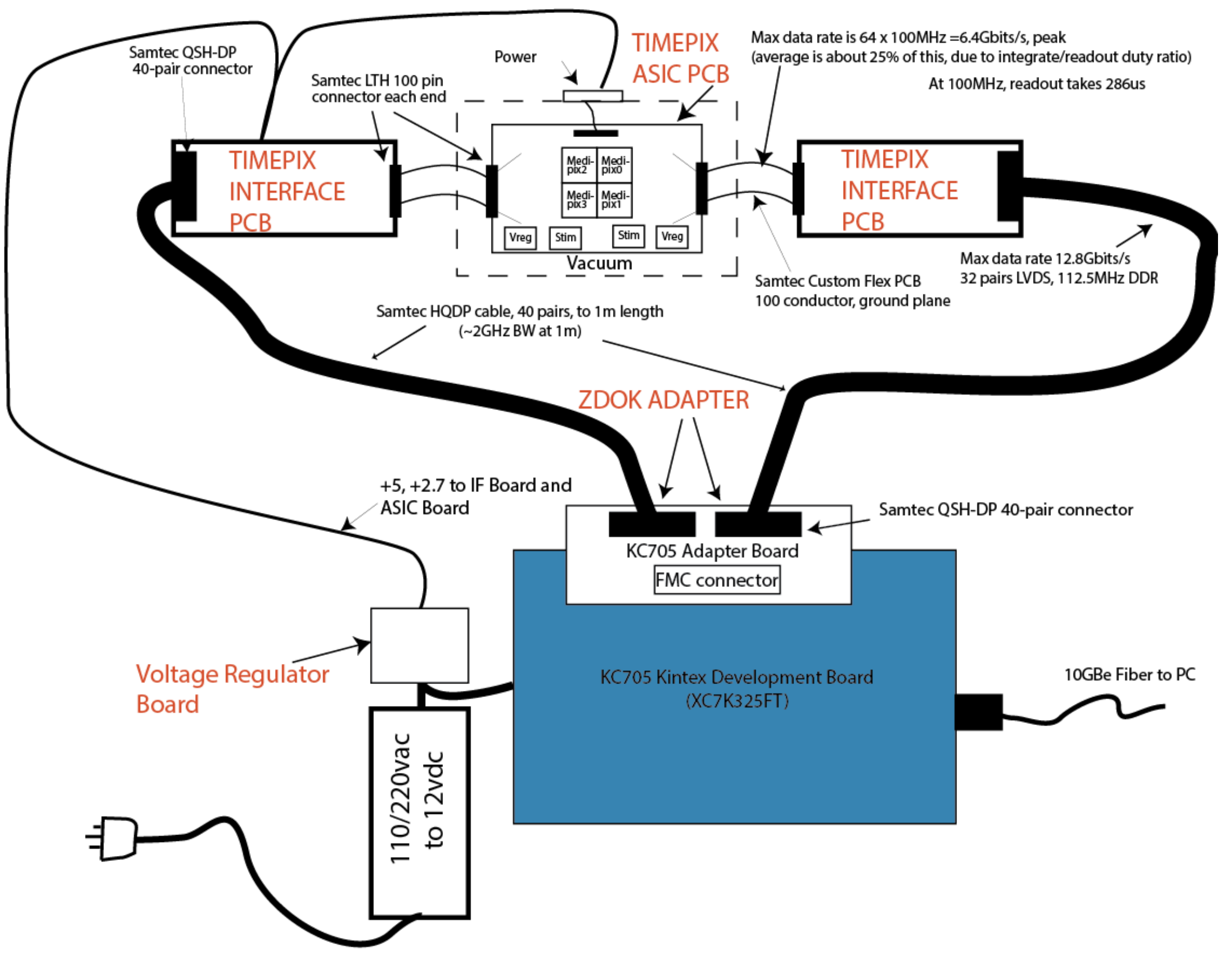
Documents

- **Top Assembly BOM and Drawing tree** 32
- **Schematics** 34
- **Parts Lists** 41

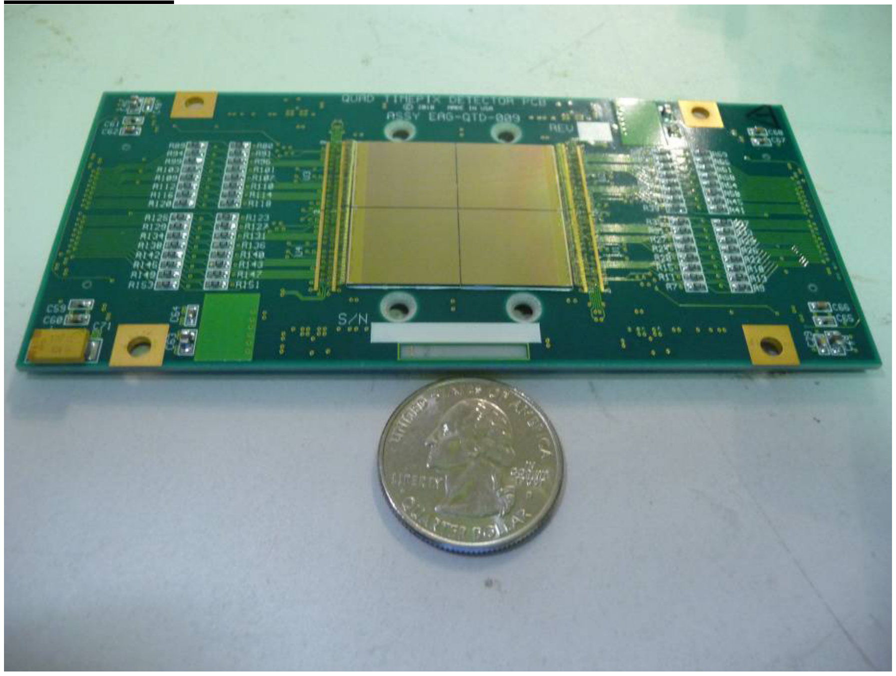
Hardware Fabrication History 44

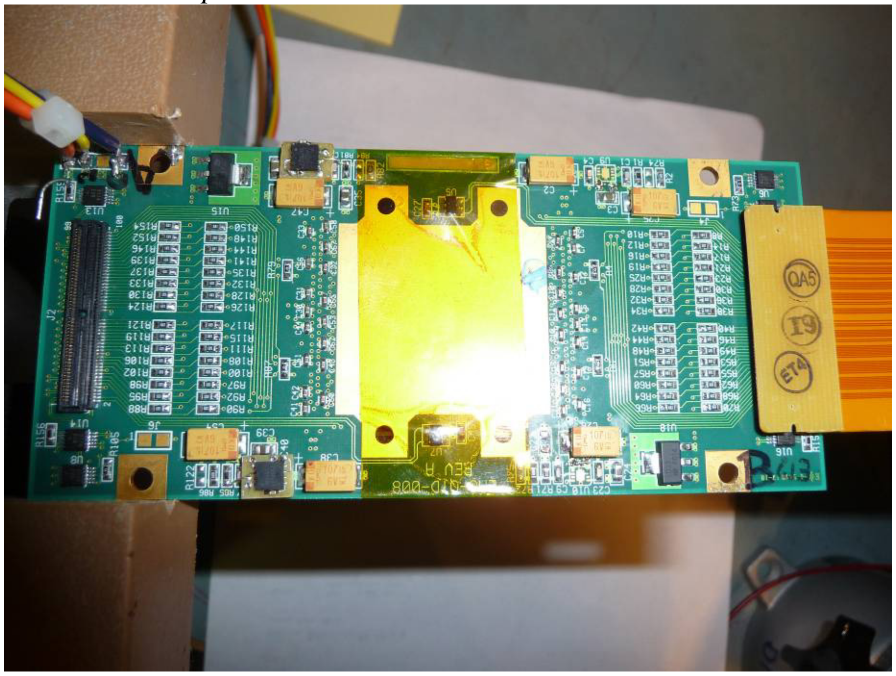
System Overview

The TimePix Fast Readout system is designed to read out four 256 by 256 pixel TimePix ASICs, for a total of 256k pixels. The diagram shows the major system components.


Samtec QSH-DP 40-pair connector
Samtec LTH 100 pin connector each end
Power
TIMEPIX ASIC PCB
Max data rate is 64 x 100MHz =6.4Gbits/s, peak (average is about 25% of this, due to integrate/readout duty ratio)
At 100MHz, readout takes 286us
TIMEPIX INTERFACE PCB
Medi-pix2
Medi-pix0
Medi-pix3
Medi-pix1
Vreg
Stim
Stim
Vreg
Vacuum
TIMEPIX INTERFACE PCB
Max data rate 12.8Gbits/s 32 pairs LVDS, 112.5MHz DDR
Samtec Custom Flex PCB 100 conductor, ground plane
Samtec HQDP cable, 40 pairs, to 1m length (~2GHz BW at 1m)
ZDOK ADAPTER
+5, +2.7 to IF Board and ASIC Board
Samtec QSH-DP 40-pair connector
KC705 Adapter Board
FMC connector
Voltage Regulator Board
KC705 Kintex Development Board (XC7K325FT)
10GBe Fiber to PC
110/220vac to 12vdc

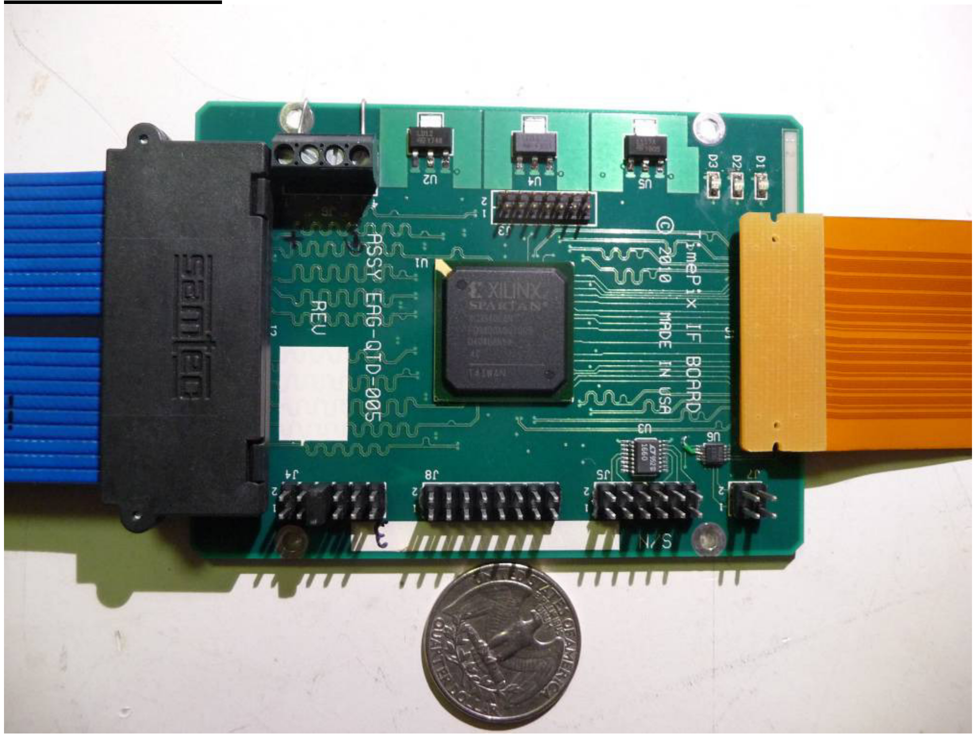
Hardware Design
ASIC Board


ASIC Board- top


ASIC Board- bottom

The ASIC Board, shown above without ASICs, carries four TimePix ASICs, butted together and die-bonded in the central area, top. A 100-pin connector (Samtec LTH-050-01-G-D-A-TR) at each end connects via a custom flex PCB (one of which is shown at bottom) to an Interface Board, each of which serves two of the ASICs. Two temperature sensor chips are located near the ASICs on the other side of the PCB. The ASIC board requires +5 and +2.7v, and locally regulates to the required supply rails. Power dissipation is about 5W. The board mounts in a vacuum housing with the flex cables bonded into slots exiting the housing.

Interface Board



The Interface Board, shown above with a flex PCB installed at the right and a data transmission cable at the left, connects to one end of the ASIC board, servicing two of the ASICs. A Spartan3AN FPGA interfaces to the ASICs, reading out the data and rearranging the bits before conditioning the data for DDR (Double Data Rate) transmission to the Roach board.

A four-channel ADC is provided to read out the voltage of the DAC_OUT terminal of each ASIC; the other two channels are used to monitor the local supply voltages (the 2-by-2 jumper block J7, bottom right, allows access to these two channels for other purposes if desired, 12 bits, 3.3v full scale).

An eight-channel DAC is provided to set the two levels of the TEST signal; a signal generated by the FPGA produces a transition between these two levels when so commanded. The other six DAC channels are made available on the 2-by-6 jumper block J5, bottom, second to right, 2.0v full scale, 10 bits.

The 2-by-8 jumper block J8, second from left, bottom, provides 8 FPGA inputs, and is unused.

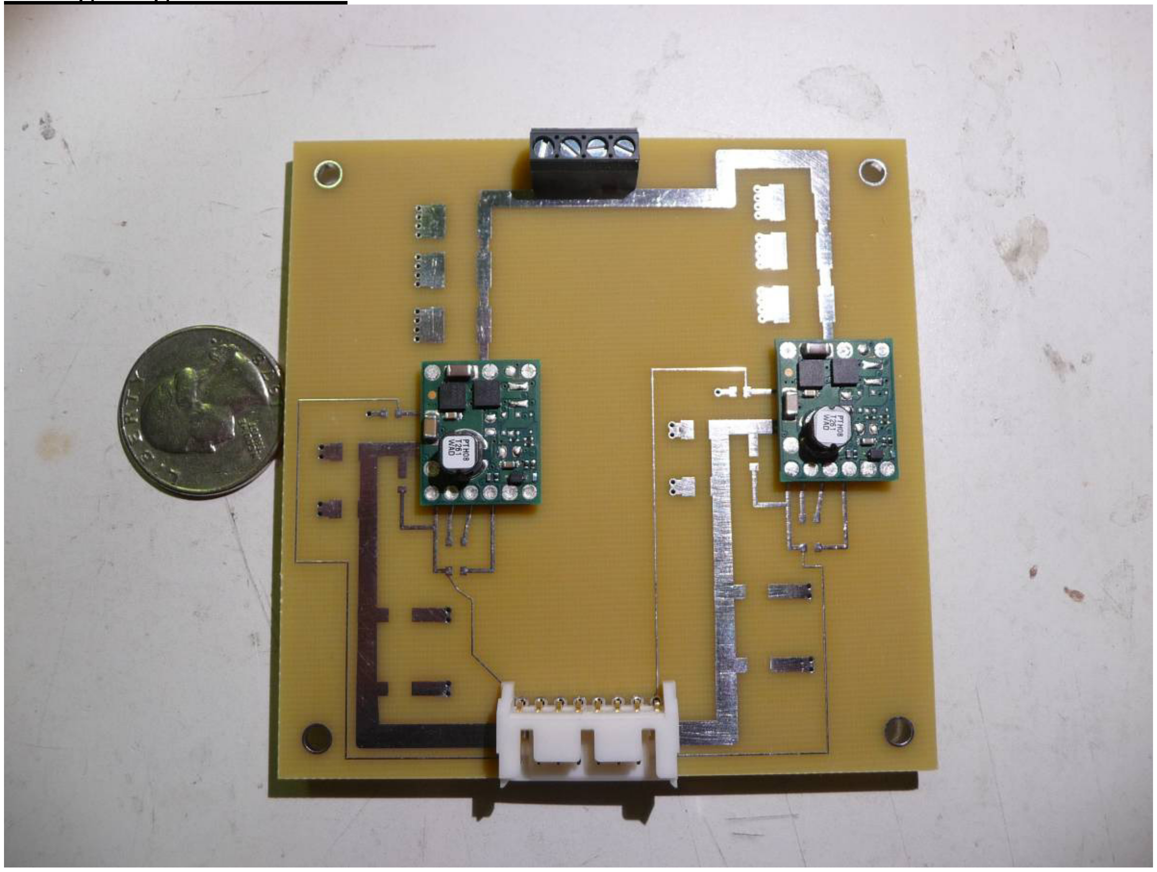
The bottom left 2-by-6 block J4 provides access to 6 FPGA input/output pins. The jumper block **must** be configured as shown (one jumper installed) when power is applied for the FPGA to boot properly (normally we'll leave the jumper block like this, though the jumper can be removed to allow a debug signal to be output on this pin).

The three LEDs at top right are, from right, D1, D2, and D3. D1 is the "FPGA OK" LED: if this LED is not on, nothing will work. The other two are commandable as described below, under "TimePixDashBoard".

The Interface Board is mounted in a housing made by Compac-RF, a standard box with modifications by the manufacturer. It requires +5v, locally regulating to the necessary rails, and dissipates about 2 watts.

KC705 Adapter Board

Voltage Regulator Board

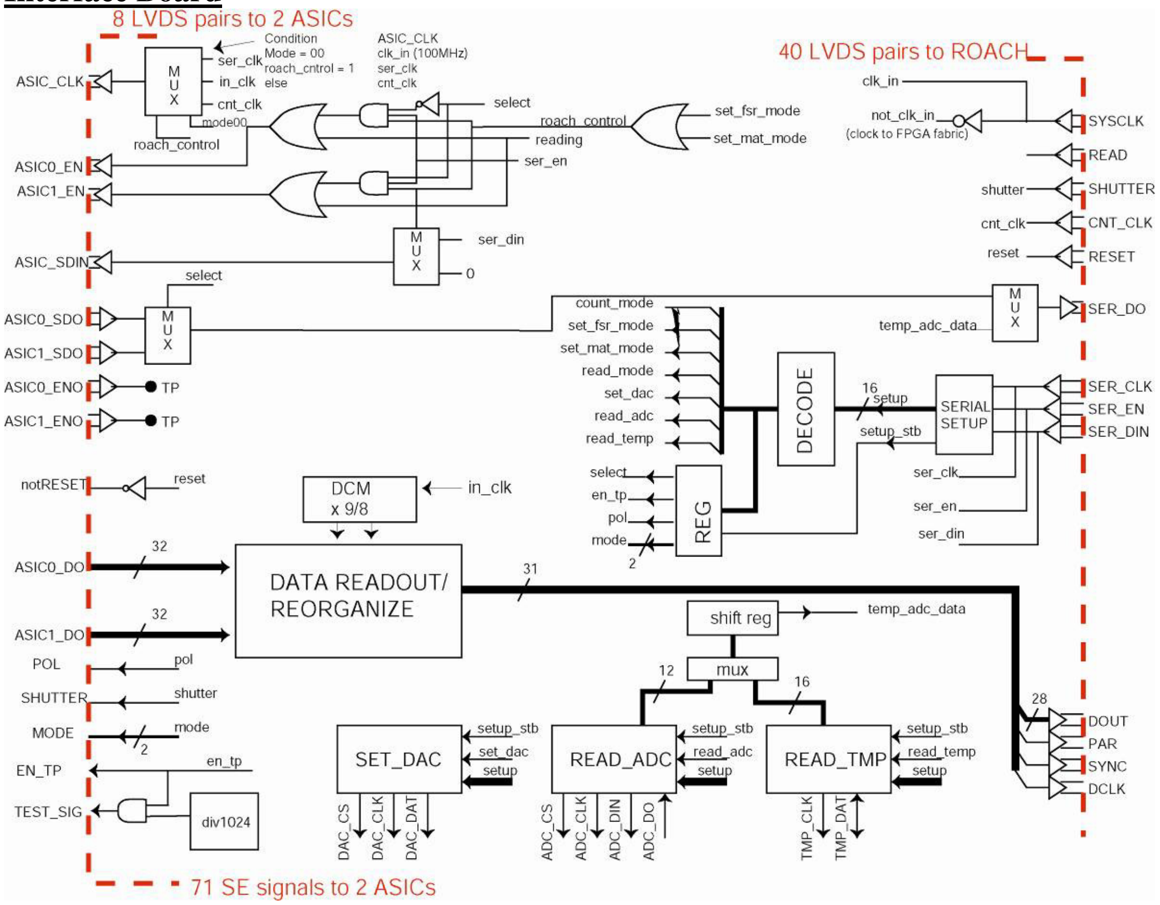


The Voltage Regulator Board, shown above not fully populated, accepts 12v dc input on the screw terminal connector at top and employs two dc/dc converters to produce the +5v and +2.7v required by the ASIC and Interface boards. The voltages supplied are "remotely sensed" (a separate wire is used to sense the voltage at the end of the connecting cable) to remove the voltage drop that occurs in the cable and allow the use of smaller conductors (the board will work without a cable installed, though).

KC705 Development Board

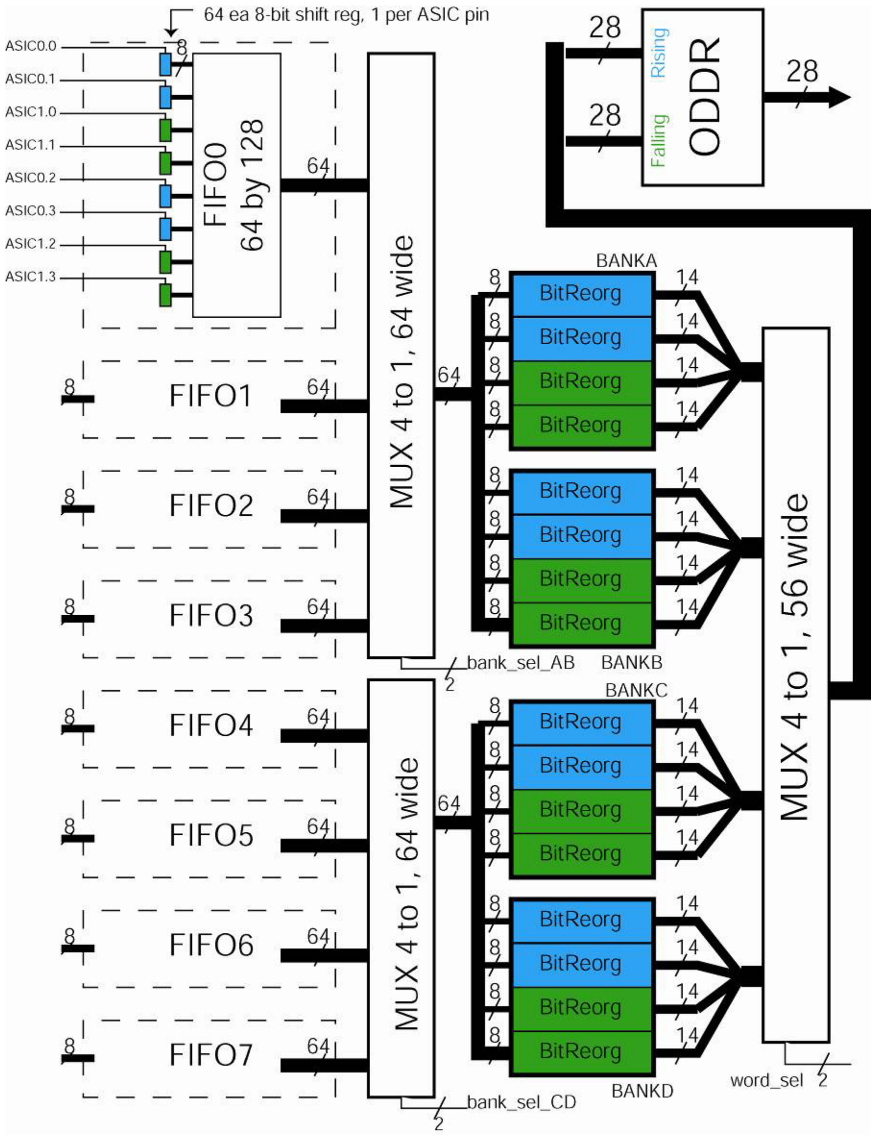
Firmware Design

Interface Board



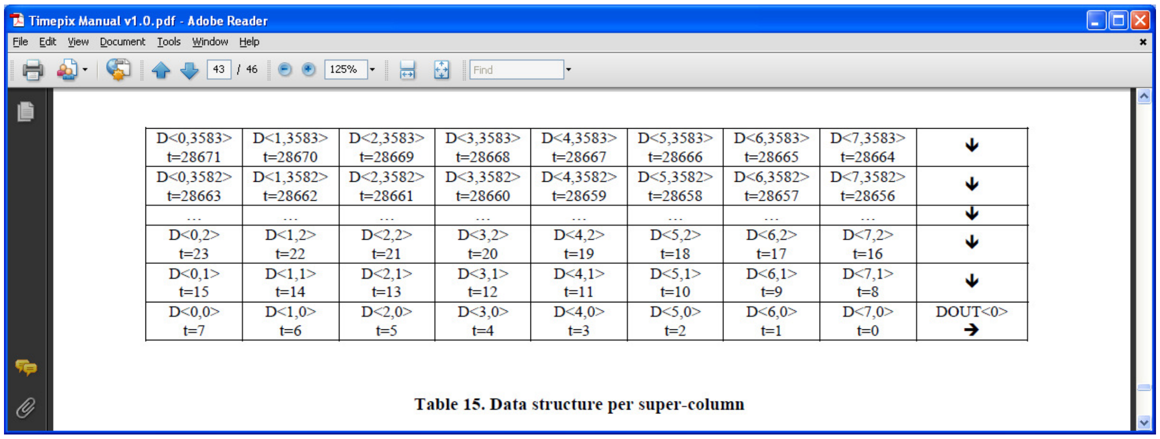
Timepix IF board firmware overview

The block diagram above shows the overall design of the Spartan 3AN firmware of the interface board. At the right, 40 pairs from the Roach board (via the ZDOK adapter) enter. At the left, the flex PCB carries signals to one half of the ASIC board (two ASICs). Two clocks are supplied: the 100MHz sysclk, and the variable-frequency cnt_clk. A 3-pair serial interface accepts 16-bit commands and data for the ASIC Fast Shift Register (256 bits) and Pixel Matrix (917,504 bits). Logic is included for reading out the ADC (for digitizing the DAC_out signals of the ASICs) and the temperature sensors on the ASIC board, and for setting the DACs (for controlling the TEST signal levels). The pixel data is read out from the ASICs by the Data Readout/Reorganize block, and is made available at the output in the form of two 14-bit busses, a parity bit (as a check of data transmission integrity), a sync bit (to indicate the beginning of a readout packet) and a clock signal, in a Double Data Rate (DDR) format at 112.5MHz. This data rate allows the 100 MHz 64 bit data entering the board at 6.4Mbits/sec to be transmitted out at 28 bits * 2 (DDR) * 112.5MHz = 6.3Mbits/sec, with a set of local FIFOs buffering a small amount of data (1/64 * ~2Mbits = 32kbits) during each transmission.



TimePix IF Board Data Readout/Reorganize Module

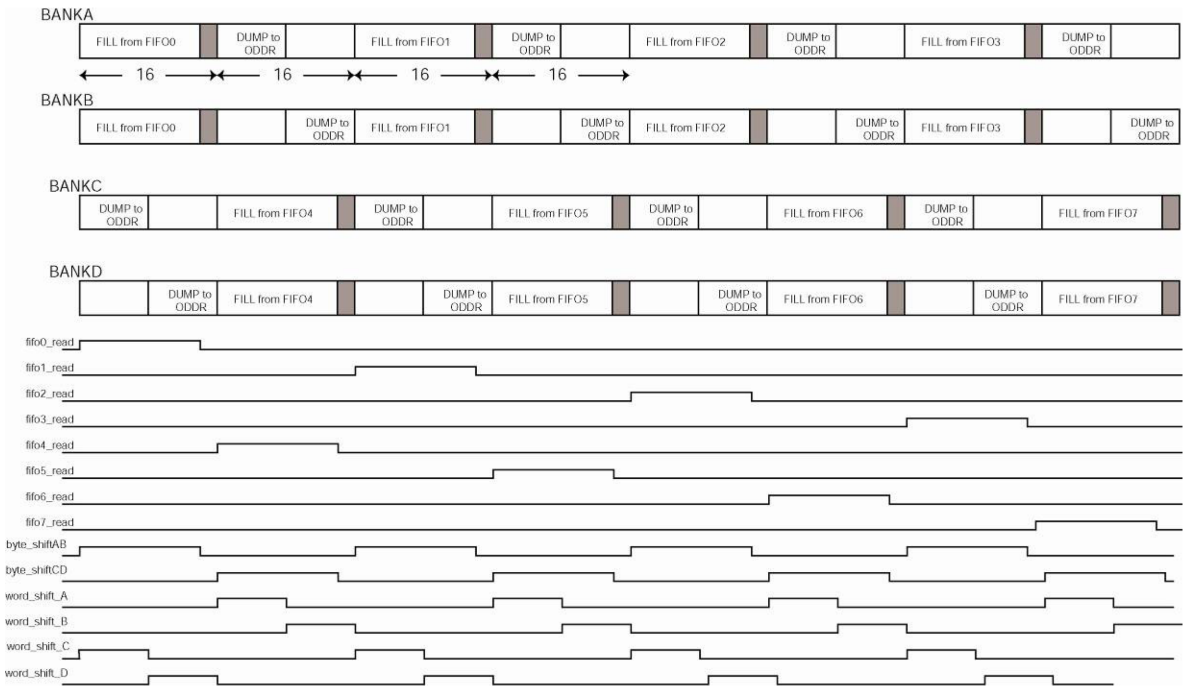
The Data Readout/Reorganize module, shown in block diagram above, receives the pixel data from the ASICs, clocked out at 100MHz. The data is presented in a scrambled form, the result of the TimePix chip's serial/parallel readout scheme. Data is shifted in serially eight bits at a time and buffered in FIFOs. The bytes are sequenced through the Bit Reorganizer blocks, in which 14 bytes are loaded into a register file, then read out serially to the Output Double Data Rate transmitters (ODDRs) according to the timing diagram below. As the data from one Bit Reorganizer are being read out, the next Bit Reorganizer is being filled with bytes from the input shift registers; when the first has been emptied, the next is ready to be dumped. In this way a constant stream of data is made available to the ODDR block. In the diagram above, data from ASIC 0 is processed by the blue blocks and output on the DDR bus rising edges, from ASIC 1 by the green blocks and the falling edges.


Timepix Manual v1.0.pdf - Adobe Reader
File Edit View Document Tools Window Help
43 / 46
125%
Find

D<0,3583> t=28671	D<1,3583> t=28670	D<2,3583> t=28669	D<3,3583> t=28668	D<4,3583> t=28667	D<5,3583> t=28666	D<6,3583> t=28665	D<7,3583> t=28664	↓
D<0,3582> t=28663	D<1,3582> t=28662	D<2,3582> t=28661	D<3,3582> t=28660	D<4,3582> t=28659	D<5,3582> t=28658	D<6,3582> t=28657	D<7,3582> t=28656	↓
…	…	…	…	…	…	…	…	↓
D<0,2> t=23	D<1,2> t=22	D<2,2> t=21	D<3,2> t=20	D<4,2> t=19	D<5,2> t=18	D<6,2> t=17	D<7,2> t=16	↓
D<0,1> t=15	D<1,1> t=14	D<2,1> t=13	D<3,1> t=12	D<4,1> t=11	D<5,1> t=10	D<6,1> t=9	D<7,1> t=8	↓
D<0,0> t=7	D<1,0> t=6	D<2,0> t=5	D<3,0> t=4	D<4,0> t=3	D<5,0> t=2	D<6,0> t=1	D<7,0> t=0	DOUT<0> →

Table 15. Data structure per super-column

TimePix ASIC parallel output data sequence, data bus bit 0.



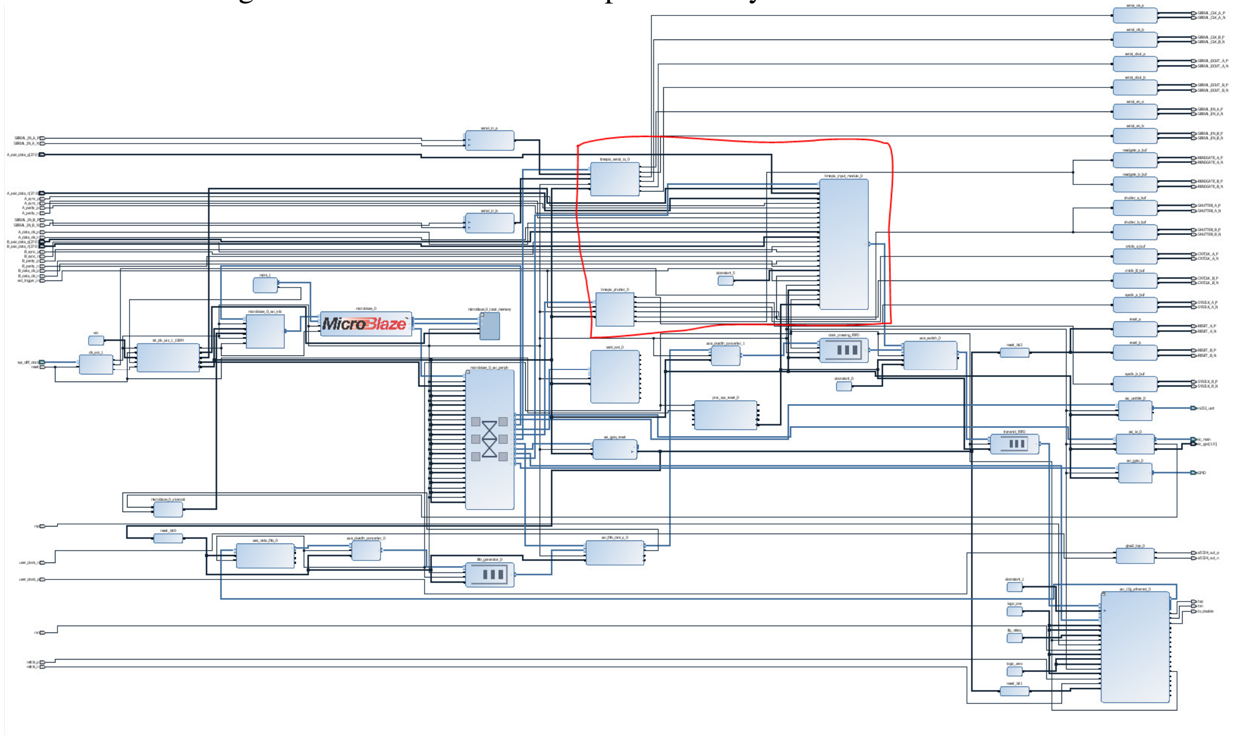
TimePix IF Bit Reorganizer Timing

The Parity bit, not shown in the block diagram, is generated in the IF board and is added to each 28-bit word to guarantee even parity (an even number of 1's in each 29-bit word). This can be checked either in Roach firmware or in software to verify the data transmission integrity from IF board to host (but not including the transmission from ASIC board to IF board).

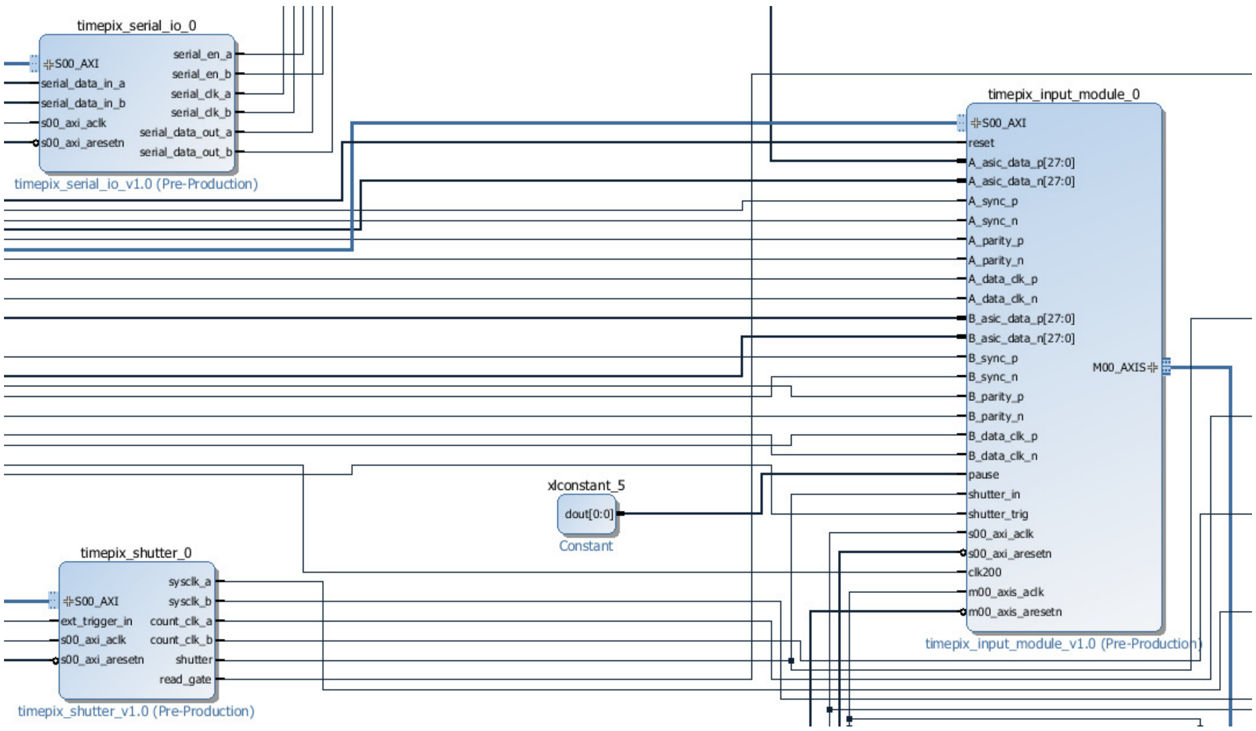
The Sync bit generation is also not shown. A single pulse is transmitted on this pair to indicate the first data word sent in a readout packet.

KC705 Firmware

The firmware design is based on a Microblaze processor system and three custom IP cores:

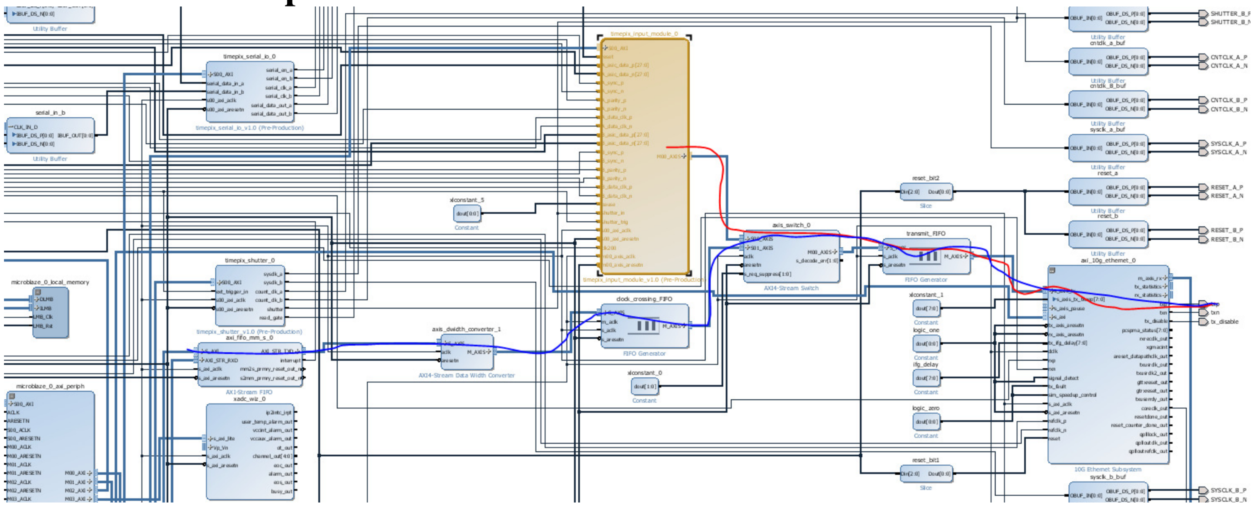


Microblaze system, custom cores indicated



The custom cores: "timepix_input_module", "timepix_shutter", "timepix_serial_io"

10GbE Transmit paths



The timepix_input_module transmits packets directly through the standard Xilinx IP core "10G Ethernet Subsystem"; this is the red path, above. The Microblaze can also transmit packets via the blue path, for housekeeping data. A number of standard IP modules are needed to make this work:

AXI Stream Memory Mapped FIFO, to convert from processor-accessible memory mapped access to stream access

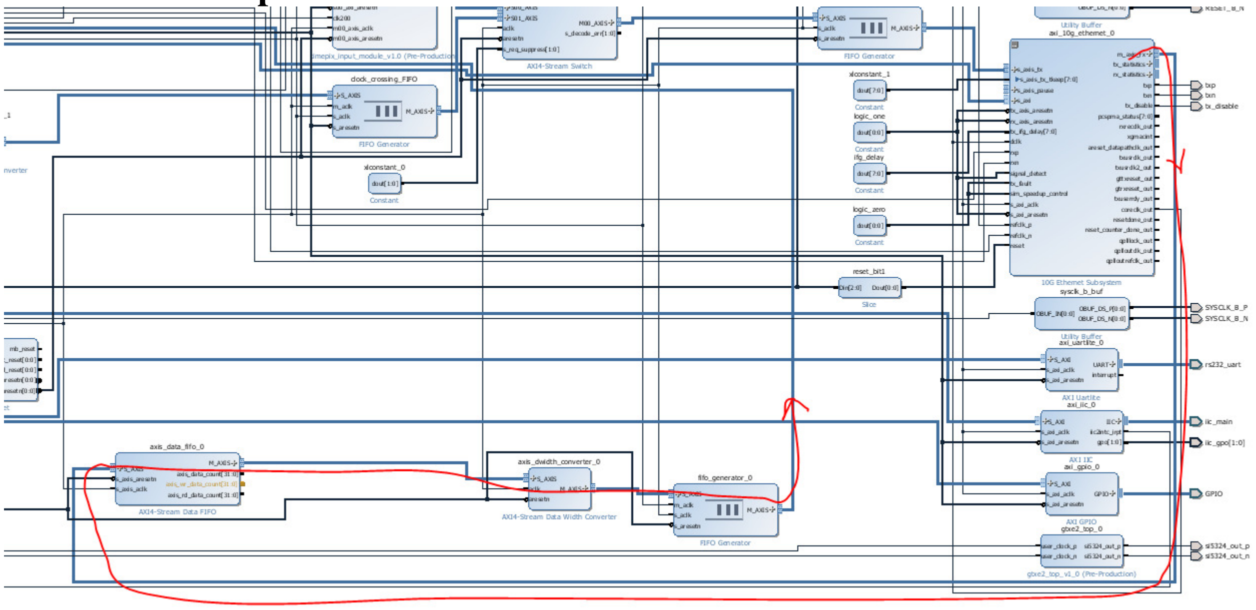
AXI-4 Stream Datawidth Converter, to go from 32-bit processor domain to 64-bit width of the 10G subsystem

Clock Crossing FIFO, to go from the 100MHz processor domain to the 156.25MHz 10G domain

AXIS switch, to combine the two streams

Transmit FIFO, to buffer packets prior to transmission

10GbE Receive path



There is a single receive path, allowing the KC705 to receive commands via the 10GbE interface, as indicated in red, requiring:

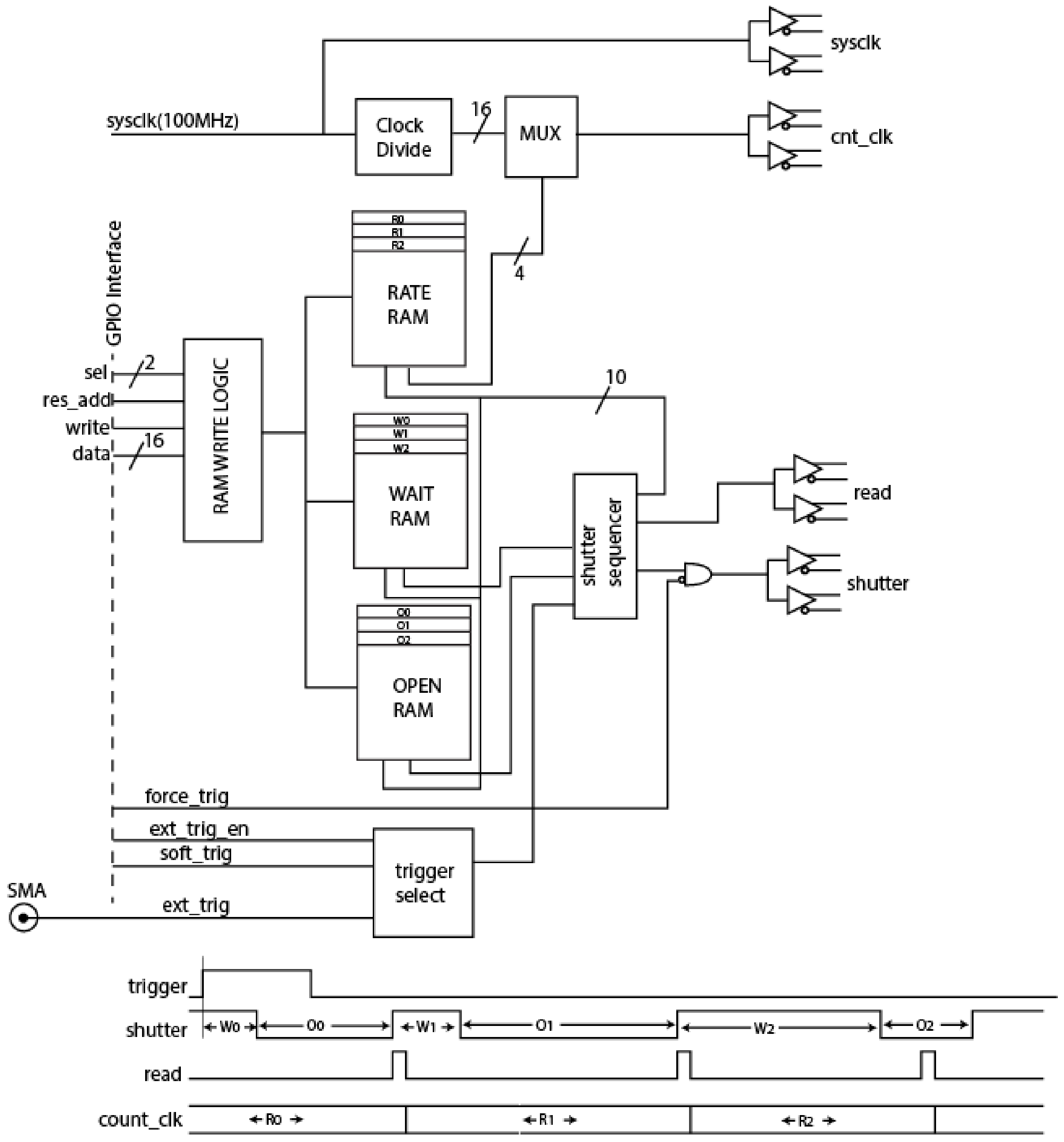
AXI-4 Stream FIFO, for buffering received packets

AXI-4 Stream Datawidth Converter, to go from 64 to 32 bits

Clock Crossing FIFO, to go from the 156.25MHz domain to the 100MHz one.

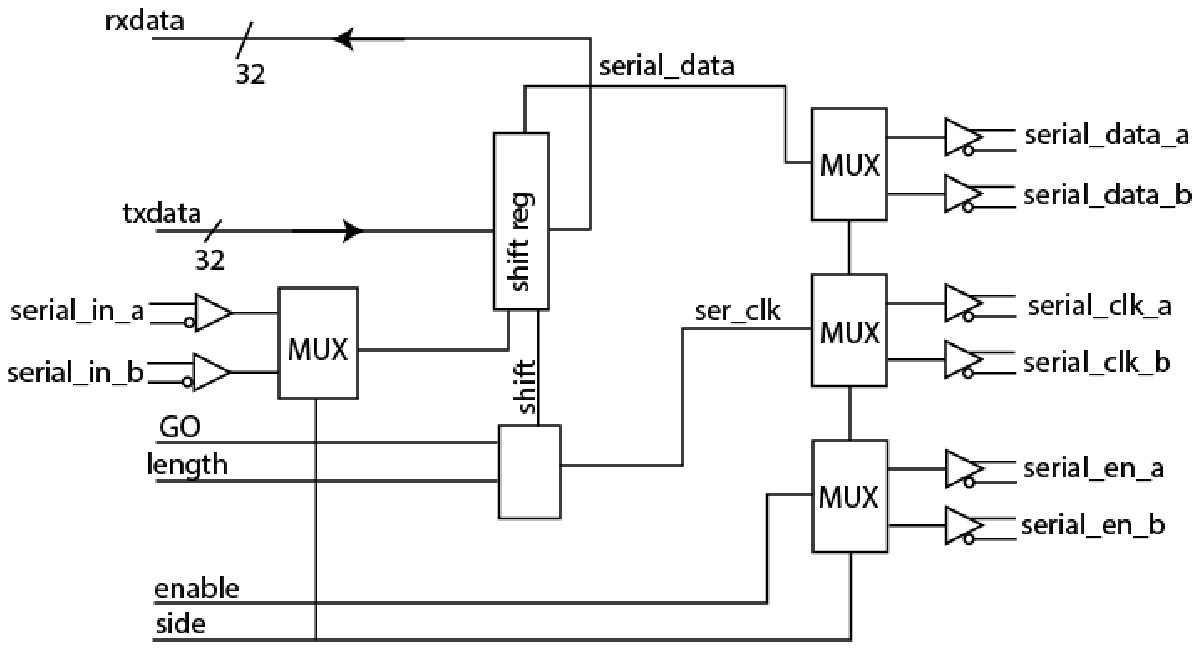
The received data packets are made available to the processor via the **AXI Stream Memory Mapped FIFO**

Shutter Control module



The shutter control module consists of three 512 deep RAMs whose contents can be written under program control. When triggered (either from the external hardware trigger input or from software control) the shutter signal to the ASICs is stepped through the programmed series of Wait and Open periods, settable in 10ns steps, and the count_clock pair is sequenced through the programmed series of frequencies. A Read pulse is emitted at the end of each shutter to tell the IF boards to send back data.

Serial IO module

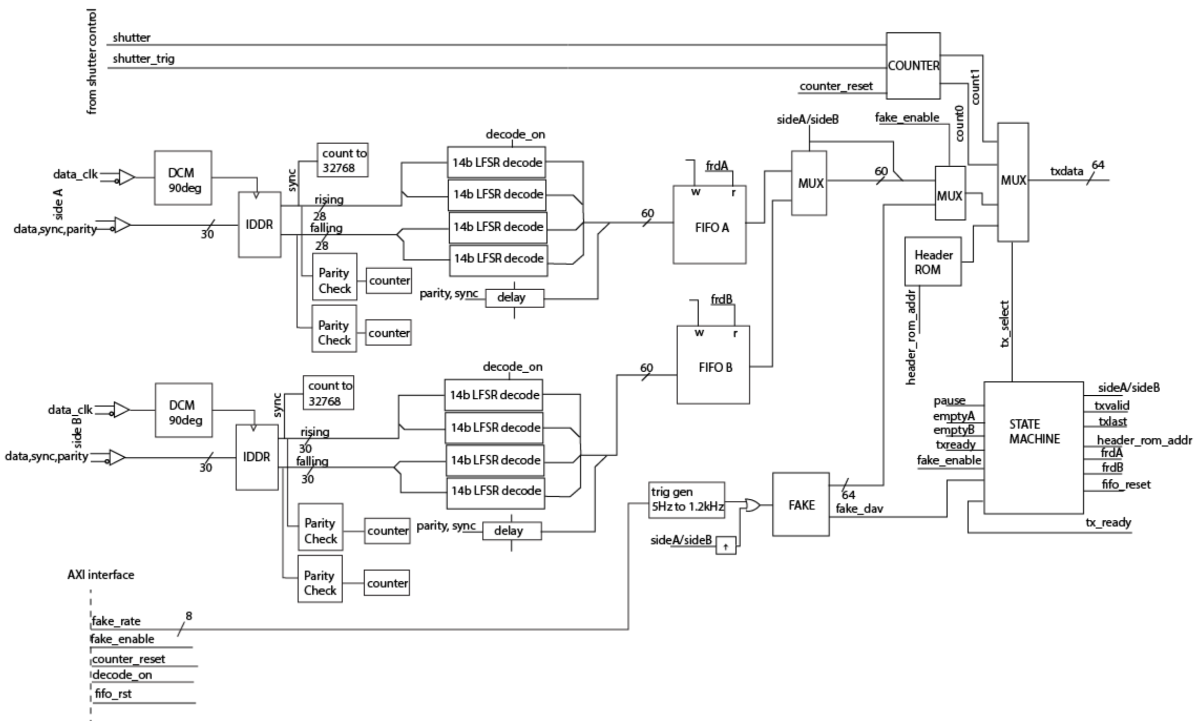


The Serial_IO module allows the shifting out of a 16- or 32-bit word to one or the other IF board (determined by the side signal). Longer sequences can be written by holding the enable line true while writing successive 32-bit words. Data shifted out from the IF boards is then available to be read under processor control. The serial clock is 1/32 of the processor clock, so about 3MHz for a 100MHz processor clock.

Timepix_input_module

This module receives the pixel data from the IF boards and transmits them to the 10GbE interface.

30 data pairs and a clock pair enter from each IF board. The 30 pairs consist of two groups of 14 pixel data pairs, a parity pair, and a sync pair.



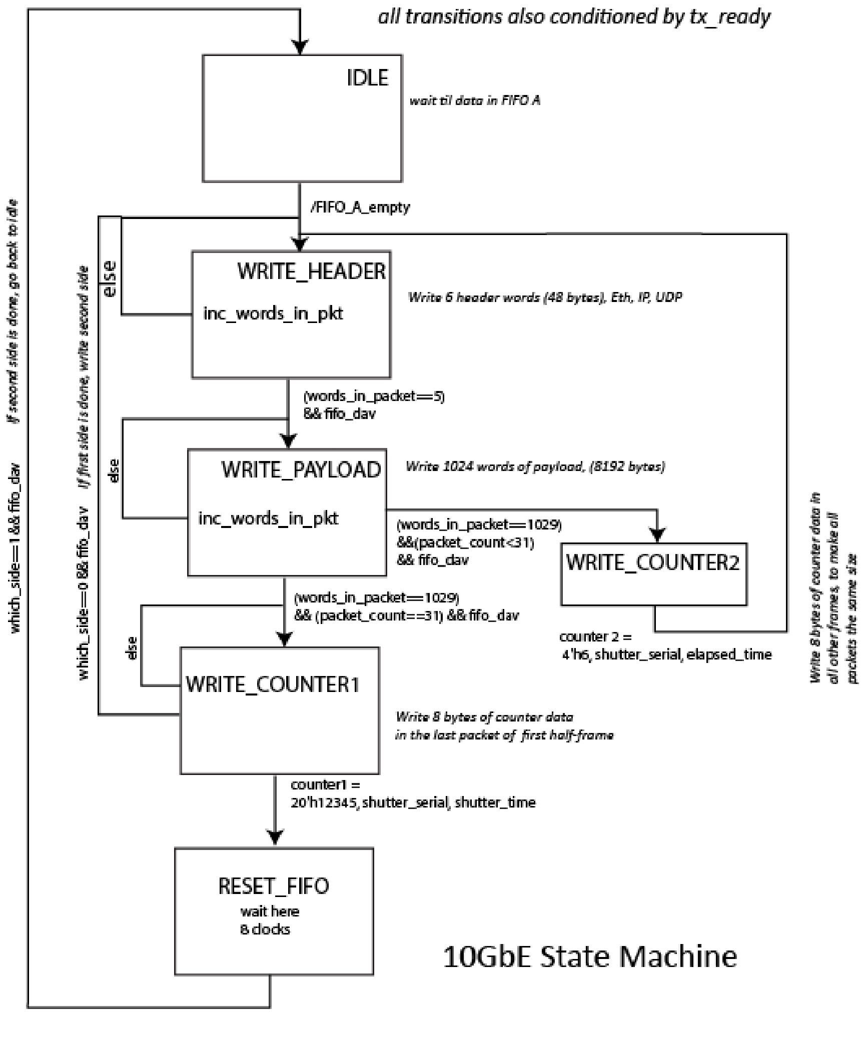
The timepix_input module incorporates two large (60 by 32k) FIFOs to receive data from the two IF boards. Data is written at 112.5MHz DDR; a DCM for each source-sync clock pair is used to center the clock edge in the data eye. Each burst of data is accompanied by a SYNC pulse, asserted during the first word of the burst. The SYNC pulse resets a counter which counts for 32768 clock pulses, filling the FIFO. Both IF boards transmit data simultaneously at the end of the shutter interval.

The data written to the FIFOs is decoded from the LFSR format to binary, if the decode_on is asserted, else it's passed through unchanged. The data is transferred in 64-bit words to the 10GbE Ethernet subsystem under the control of a state machine. Appropriate header words for Ethernet, IP, and UDP formats are added. MAC address, IP address, and UDP port are all hard-coded.

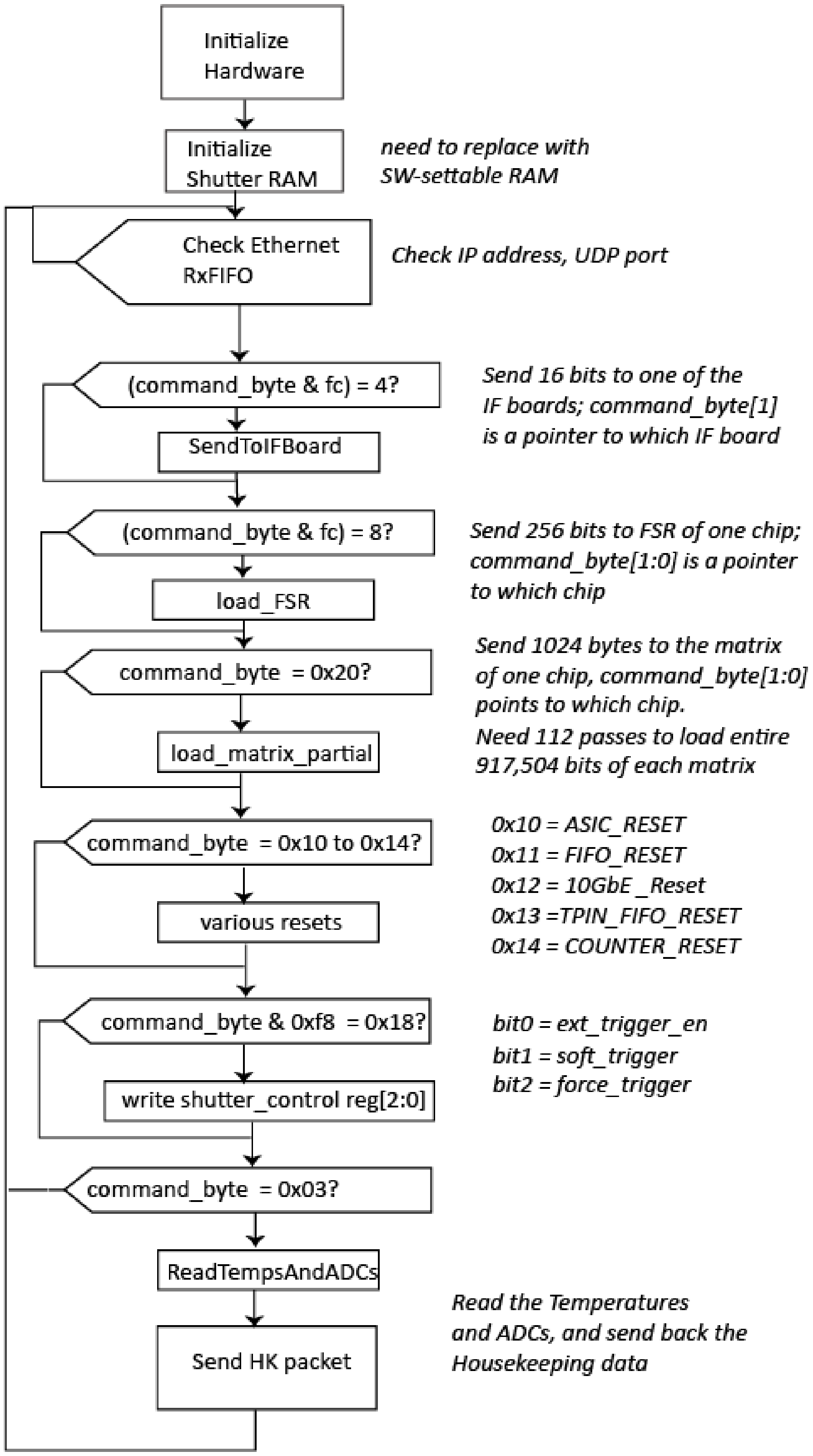
Parity-checking logic looks for and counts parity errors in the input data (each pair of 14-bit parallel words at the input is accompanied by one parity bit).

A pair of counters is used to attach a time-tag to the shutter pulse; the 48-bit elapsed time counter provides a time-stamp relative to a software-controlled reset, while the 32-bit shutter_time counter provides a time-stamp relative to the trigger

signal which initiated the shutter sequence. A shutter serial number, incremented at each shutter and reset at trigger, is also provided.
A fake data generator is available to check data transmission from KC705 to the host. The generator produces a 32768-word-long pseudo-random sequence. The repetition rate for the sequence can be varied from about 5Hz to 1.2kHz.
The data format is
fake_data = {1'b1, LFSR15, 1'b0, LFSR15, 1'b0, LFSR15, 1'b1, LFSR15},
where LFSR15 is the 15-bit LFSR sequence starting at 0x0001, with feedback taken from bits 14 and 13. The LFSR sequence itself is of 32767 values; the last value is repeated once to form the 32768 value sequence.


all transitions also conditioned by tx_ready
IDLE
wait til data in FIFO A
/FIFO_A_empty
WRITE_HEADER
inc_words_in_pkt
Write 6 header words (48 bytes), Eth, IP, UDP
(words_in_packet==5)
&& fifo_dav
WRITE_PAYLOAD
inc_words_in_pkt
Write 1024 words of payload, (8192 bytes)
(words_in_packet==1029)
&&(packet_count<31)
&& fifo_dav
WRITE_COUNTER2
counter 2 =
4'h6, shutter_serial, elapsed_time
Write 8 bytes of counter data in all other frames, to make all packets the same size
(words_in_packet==1029)
&& (packet_count==31) && fifo_dav
WRITE_COUNTER1
Write 8 bytes of counter data
in the last packet of first half-frame
counter1 =
20'h12345, shutter_serial, shutter_time
RESET_FIFO
wait here
8 clocks
else
else
else
which_side==1 && fifo_dav
If second side is done, go back to idle
which_side==0 && fifo_dav
If first side is done, write second side
10GbE State Machine

KC705 MicroBlaze Software


Initialize Hardware
Initialize Shutter RAM
need to replace with SW-settable RAM
Check Ethernet RxFIFO
Check IP address, UDP port
(command_byte & fc) = 4?
SendToIFBoard
Send 16 bits to one of the IF boards; command_byte[1] is a pointer to which IF board
(command_byte & fc) = 8?
load_FSR
Send 256 bits to FSR of one chip; command_byte[1:0] is a pointer to which chip
command_byte = 0x20?
load_matrix_partial
Send 1024 bytes to the matrix of one chip, command_byte[1:0] points to which chip.
Need 112 passes to load entire 917,504 bits of each matrix
command_byte = 0x10 to 0x14?
various resets
0x10 = ASIC_RESET
0x11 = FIFO_RESET
0x12 = 10GbE _Reset
0x13 =TPIN_FIFO_RESET
0x14 = COUNTER_RESET
command_byte & 0xf8 = 0x18?
write shutter_control reg[2:0]
bit0 = ext_trigger_en
bit1 = soft_trigger
bit2 = force_trigger
command_byte = 0x03?
ReadTempsAndADCs
Send HK packet
Read the Temperatures and ADCs, and send back the Housekeeping data

Interfaces

TimePix_FE to TimePix_IF interconnect

TimePix_IF to KC705 Adapter Interconnect

Adapter board pinout is as follows. All signal pairs are inverted by a pair-swap, except for the two highlighted.

January 26, 2015. Rev A. Per PCB layout

	TimePix IF Board			KC705 Adapter					KC705
Index	TimePix_IF FPGA pin	Signal Name	Function	QSH Pin	Which QSH	Adapter Net	Which FMC	FMC pin	FPGA loc
1	W12	LVDS_P[0]	PDATA_P[0]	2	J3	L_LA00P	J2	g6	AD23
2	Y12	LVDS_N[0]	PDATA_N[0]	4	J3	L_LA00N	J2	g7	AE24
3	W10	LVDS_P[1]	PDATA_P[1]	6	J3	L_LA03P	J2	g9	AG20
4	V10	LVDS_N[1]	PDATA_N[1]	8	J3	L_LA03N	J2	g10	AH20
5	Y9	LVDS_P[2]	PDATA_P[2]	10	J3	L_LA05P	J2	d11	AG22
6	W9	LVDS_N[2]	PDATA_N[2]	12	J3	L_LA05N	J2	d12	AH22
7	V9	LVDS_P[3]	PDATA_P[3]	14	J3	L_LA08P	J2	g12	AJ22
8	V8	LVDS_N[3]	PDATA_N[3]	16	J3	L_LA08N	J2	g13	AJ23
9	U5	LVDS_P[4]	PDATA_P[4]	18	J3	L_LA10P	J2	c14	AJ24
10	V5	LVDS_N[4]	PDATA_N[4]	20	J3	L_LA10N	J2	c15	AK25
11	U6	LVDS_P[5]	PDATA_P[5]	22	J3	L_LA12P	J2	g15	AA20
12	T7	LVDS_N[5]	PDATA_N[5]	24	J3	L_LA12N	J2	g16	AB20
13	U9	LVDS_P[6]	PDATA_P[6]	26	J3	L_LA13P	J2	d17	AB24
14	T9	LVDS_N[6]	PDATA_N[6]	28	J3	L_LA13N	J2	d18	AC25
15	V12	LVDS_P[7]	PDATA_P[7]	30	J3	L_LA16P	J2	g18	AC22
16	U11	LVDS_N[7]	PDATA_N[7]	32	J3	L_LA16N	J2	g19	AD22
17	V13	LVDS_P[8]	PDATA_P[8]	34	J3	L_LA17_CCP	J2	d20	AB27
18	U13	LVDS_N[8]	PDATA_N[8]	36	J3	L_LA17_CCN	J2	d21	AC27
19	F9	LVDS_P[9]	PDATA_P[9]	38	J3	L_LA19P	J2	h22	AJ26
20	E9	LVDS_N[9]	PDATA_N[9]	40	J3	L_LA19N	J2	h23	AK26
21	F13	LVDS_P[10]	PDATA_P[10]	42	J3	L_LA23P	J2	d23	AH26
22	E13	LVDS_N[10]	PDATA_N[10]	44	J3	L_LA23N	J2	d24	AH27
23	F7	LVDS_P[11]	PDATA_P[11]	46	J3	L_LA21P	J2	h25	AG27
24	E7	LVDS_N[11]	PDATA_N[11]	48	J3	L_LA21N	J2	h26	AG28
25	D8	LVDS_P[12]	PDATA_P[12]	50	J3	L_LA27P	J2	c26	AJ28
26	C7	LVDS_N[12]	PDATA_N[12]	52	J3	L_LA27N	J2	c27	AJ29
27	B3	LVDS_P[13]	PDATA_P[13]	54	J3	H_LA26P	J1	d26	B18
28	A3	LVDS_N[13]	PDATA_N[13]	56	J3	H_LA26N	J1	d27	A18
29	B2	LVDS_P[14]	PDATA_P[14]	58	J3	L_LA29P	J2	g30	AE28
30	A2	LVDS_N[14]	PDATA_N[14]	60	J3	L_LA29N	J2	g31	AF28
31	B5	LVDS_P[15]	PDATA_P[15]	62	J3	H_LA04P	J1	h10	G28
32	A5	LVDS_N15]	PDATA_N[15]	64	J3	H_LA04N	J1	h11	F28
33	A7	LVDS_P[16]	PDATA_P[16]	66	J3	H_LA23N	J1	d24	A22
34	B7	LVDS_N[16]	PDATA_N[16]	68	J3	H_LA23P	J1	d23	B22
35	C8	LVDS_P[17]	PDATA_P[17]	70	J3	H_HA14P	J1	j15	J16
36	B8	LVDS_N[17]	PDATA_N[17]	72	J3	H_HA14N	J1	j16	H16
37	E10	LVDS_P[18]	PDATA_P[18]	74	J3	H_HA22N	J1	j22	K11

38	D10	LVDS_N[18]	PDATA_N[18]	76	J3	H_HA22P	J1	j21	L11
39	C11	LVDS_P[19]	PDATA_P[19]	78	J3	H_LA02P	J1	h7	H24
40	B11	LVDS_N[19]	PDATA_N[19]	80	J3	H_LA02N	J1	h8	H25
41	W13	LVDS_P[20]	PDATA_P[20]	1	J3	L_LA02P	J2	h7	AF20
42	Y13	LVDS_N[20]	PDATA_N[20]	3	J3	L_LA02N	J2	h8	AF21
43	V11	LVDS_P[21]	PDATA_P[21]	5	J3	L_LA01P	J2	d8	AE23
44	Y11	LVDS_N[21]	PDATA_N[21]	7	J3	L_LA01N	J2	d9	AF23
45	Y7	LVDS_P[22]	PDATA_P[22]	9	J3	L_LA06P	J2	c10	AK20
46	Y6	LVDS_N[22]	PDATA_N[22]	11	J3	L_LA06N	J2	c11	AK21
47	W8	LVDS_P[23]	PDATA_P[23]	13	J3	L_LA04P	J2	h10	AH21
48	V7	LVDS_N[23]	PDATA_N[23]	15	J3	L_LA04N	J2	h11	AJ21
49	Y5	LVDS_P[24]	PDATA_P[24]	17	J3	L_LA07P	J2	h13	AG25
50	Y4	LVDS_N[24]	PDATA_N[24]	19	J3	L_LA07N	J2	h14	AH25
51	W4	LVDS_P[25]	PDATA_P[25]	21	J3	L_LA09P	J2	d14	AK23
52	Y3	LVDS_N[25]	PDATA_N[25]	23	J3	L_LA09N	J2	d15	AK24
53	R7	LVDS_P[26]	PDATA_P[26]	25	J3	L_LA11P	J2	h16	AE25
54	T6	LVDS_N[26]	PDATA_N[26]	27	J3	L_LA11N	J2	h17	AF25
55	T10	LVDS_P[27]	PDATA_P[27]	29	J3	L_LA14P	J2	c18	AD21
56	U10	LVDS_N[27]	PDATA_N[27]	31	J3	L_LA14N	J2	c19	AE21
57	R12	LVDS_P[28]	Parity_P	33	J3	L_LA15P	J2	h19	AC24
58	T12	LVDS_N[28]	Parity_N	35	J3	L_LA15N	J2	h20	AD24
59	R13	LVDS_P[29]	Sync_P	37	J3	L_LA20P	J2	g21	AF26
60	T13	LVDS_N[29]	Sync_N	39	J3	L_LA20N	J2	g22	AF27
61	F12	LVDS_P[30]	DataClk_P	41	J3	L_LA18_CCP	J2	c22	AD27
62	D12	LVDS_N[30]	DataClk_N	43	J3	L_LA18_CCN	J2	c23	AD28
63	F8	LVDS_P[31]	SysClk_P	45	J3	L_LA22P	J2	g24	AJ27
64	E8	LVDS_N[31]	SysClk_N	47	J3	L_LA22N	J2	g25	AK28
65	F6	LVDS_P[32]	Reset_P	49	J3	L_LA26P	J2	d26	AK29
66	E6	LVDS_N[32]	Reset_N	51	J3	L_LA26N	J2	d27	AK30
67	D6	LVDS_P[33]	Read_P	53	J3	L_LA25P	J2	g27	AC26
68	C5	LVDS_N[33]	Read_N	55	J3	L_LA25N	J2	g28	AD26
69	C4	LVDS_P[34]	Shutter_P	57	J3	L_LA24P	J2	h28	AG30
70	A4	LVDS_N[34]	Shutter_N	59	J3	L_LA24N	J2	h29	AH30
71	C6	LVDS_P[35]	CountClk_P	61	J3	L_LA28P	J2	h31	AE30
72	A6	LVDS_N[35]	CountClk_N	63	J3	L_LA28N	J2	h32	AF30
73	A8	LVDS_P[36]	SerEn_P	65	J3	L_LA31P	J2	g33	AD29
74	A9	LVDS_N[36]	SerEn_N	67	J3	L_LA31N	J2	g34	AE29
75	C9	LVDS_P[37]	SerDat2IF_P	69	J3	L_LA30P	J2	h34	AB29
76	B9	LVDS_N[37]	SerDat2IF_N	71	J3	L_LA30N	J2	h35	AB30
77	E11	LVDS_P[38]	SerClk_P	73	J3	L_LA33P	J2	g36	AC29
78	D11	LVDS_N[38]	SerClk_N	75	J3	L_LA33N	J2	g37	AC30
79	C12	LVDS_P[39]	SerDatFrIF_P	77	J3	L_LA32P	J2	h37	Y30
80	B13	LVDS_N[39]	SerDatFrIF_N	79	J3	L_LA32N	J2	h38	AA30
81	W12	LVDS_P[0]	PDATA_P[0]	2	J4	H_HA01_CCP	J1	e2	H14
82	Y12	LVDS_N[0]	PDATA_N[0]	4	J4	H_HA01_CCN	J1	e3	G14
83	W21	LVDS_P[1]	PDATA_P[1]	6	J4	H_HA02P	J1	k7	D11

84	V10	LVDS_N[1]	*PDATA_N[1]*	8	J4	H_HA02N	J1	k8	C11
85	W9	LVDS_P[2]	*PDATA_P[2]*	10	J4	H_HA04P	J1	f7	F11
86	Y9	LVDS_N[2]	*PDATA_N[2]*	12	J4	H_HA04N	J1	f8	E11
87	V9	LVDS_P[3]	*PDATA_P[3]*	14	J4	H_LA03P	J1	g9	H26
88	V8	LVDS_N[3]	*PDATA_N[3]*	16	J4	H_LA03N	J1	g10	H27
89	V5	LVDS_P[4]	*PDATA_P[4]*	18	J4	H_HA06P	J1	k10	D14
90	U5	LVDS_N[4]	*PDATA_N[4]*	20	J4	H_HA06N	J1	k11	C14
91	U6	LVDS_P[5]	*PDATA_P[5]*	22	J4	H_LA08P	J1	g12	E29
92	T7	LVDS_N[5]	*PDATA_N[5]*	24	J4	H_LA08N	J1	g13	E30
93	U9	LVDS_P[6]	*PDATA_P[6]*	26	J4	H_LA12P	J1	g15	C29
94	T9	LVDS_N[6]	*PDATA_N[6]*	28	J4	H_LA12N	J1	g16	B29
95	V12	LVDS_P[7]	*PDATA_P[7]*	30	J4	H_HA10P	J1	k13	A11
96	U11	LVDS_N[7]	*PDATA_N[7]*	32	J4	H_HA10N	J1	k14	A12
97	V13	LVDS_P[8]	*PDATA_P[8]*	34	J4	H_HA15P	J1	f16	H15
98	U13	LVDS_N[8]	*PDATA_N[8]*	36	J4	H_HA15N	J1	f17	G15
99	F9	LVDS_P[9]	*PDATA_P[9]*	38	J4	H_LA11P	J1	h16	G27
100	E9	LVDS_N[9]	*PDATA_N[9]*	40	J4	H_LA11N	J1	h17	F27
101	F13	LVDS_P[10]	*PDATA_P[10]*	42	J4	H_LA13P	J1	d17	A25
102	E13	LVDS_N[10]	*PDATA_N[10]*	44	J4	H_LA13N	J1	d18	A26
103	F7	LVDS_P[11]	*PDATA_P[11]*	46	J4	H_HA21P	J1	k19	J11
104	E7	LVDS_N[11]	*PDATA_N[11]*	48	J4	H_HA21N	J1	k20	J12
105	D8	LVDS_P[12]	*PDATA_P[12]*	50	J4	H_HA19P	J1	f19	H11
106	C7	LVDS_N[12]	*PDATA_N[12]*	52	J4	H_HA19N	J1	f20	H12
107	B3	LVDS_P[13]	*PDATA_P[13]*	54	J4	H_LA20N	J1	g21	E19
108	A3	LVDS_N[13]	*PDATA_N[13]*	56	J4	H_LA20P	J1	g22	D19
109	B2	LVDS_P[14]	*PDATA_P[14]*	58	J4	H_LA25P	J1	g27	G17
110	A2	LVDS_N[14]	*PDATA_N[14]*	60	J4	H_LA25N	J1	g28	F17
111	B5	LVDS_P[15]	*PDATA_P[15]*	62	J4	*H_LA28P*	*J1*	*h31*	*D16*
112	A5	LVDS_N15]	*PDATA_N[15]*	64	J4	*H_LA28N*	*J1*	*h32*	*C16*
113	A7	LVDS_P[16]	*PDATA_P[16]*	66	J4	*H_LA21P*	*J1*	*h25*	*A20*
114	B7	LVDS_N[16]	*PDATA_N[16]*	68	J4	*H_LA21N*	*J1*	*h26*	*A21*
115	C8	LVDS_P[17]	*PDATA_P[17]*	70	J4	*H_LA30P*	*J1*	*h34*	*D22*
116	B8	LVDS_N[17]	*PDATA_N[17]*	72	J4	*H_LA30N*	*J1*	*h35*	*C22*
117	E10	LVDS_P[18]	*PDATA_P[18]*	74	J4	*H_LA24P*	*J1*	*h28*	*A16*
118	D10	LVDS_N[18]	*PDATA_N[18]*	76	J4	*H_LA24N*	*J1*	*h29*	*A17*
119	C11	LVDS_P[19]	*PDATA_P[19]*	78	J4	H_LA33P	J1	g36	H21
120	B11	LVDS_N[19]	*PDATA_N[19]*	80	J4	H_LA33N	J1	g37	H22
121	W13	LVDS_P[20]	*PDATA_P[20]*	1	J4	H_HA03P	J1	j6	C12
122	Y13	LVDS_N[20]	*PDATA_N[20]*	3	J4	H_HA03N	J1	j7	B12
123	V11	LVDS_P[21]	*PDATA_P[21]*	5	J4	H_LA00_CCP	J1	g6	C25
124	Y11	LVDS_N[21]	*PDATA_N[21]*	7	J4	H_LA00_CCN	J1	g7	B25
125	Y7	LVDS_P[22]	*PDATA_P[22]*	9	J4	H_HA07P	J1	j9	B14
126	Y6	LVDS_N[22]	*PDATA_N[22]*	11	J4	H_HA07N	J1	j10	A15
127	W8	LVDS_P[23]	*PDATA_P[23]*	13	J4	H_HA00_CCP	J1	f4	D12

128	V7	LVDS_N[23]	*PDATA_N[23]*	15	J4	H_HA00_CCN	J1	f5	D13
129	Y5	LVDS_P[24]	*PDATA_P[24]*	17	J4	H_HA08P	J1	f10	E14
130	Y4	LVDS_N[24]	*PDATA_N[24]*	19	J4	H_HA08N	J1	f11	E15
131	W4	LVDS_P[25]	*PDATA_P[25]*	21	J4	H_HA11P	J1	j12	B13
132	Y3	LVDS_N[25]	*PDATA_N[25]*	23	J4	H_HA11N	J1	j13	A13
133	R7	LVDS_P[26]	*PDATA_P[26]*	25	J4	H_LA07P	J1	h13	E28
134	T6	LVDS_N[26]	*PDATA_N[26]*	27	J4	H_LA07N	J1	h14	D28
135	T10	LVDS_P[27]	*PDATA_P[27]*	29	J4	H_HA12P	J1	f13	C15
136	U10	LVDS_N[27]	*PDATA_N[27]*	31	J4	H_HA12N	J1	f14	B15
137	R12	LVDS_P[28]	*Parity_P*	33	J4	H_LA09P	J1	d14	B30
138	T12	LVDS_N[28]	*Parity_N*	35	J4	H_LA09N	J1	d15	A30
139	R13	LVDS_P[29]	*Sync_P*	37	J4	H_HA18P	J1	j18	K14
140	T13	LVDS_N[29]	*Sync_N*	39	J4	H_HA18N	J1	j19	J14
141	F12	LVDS_P[30]	*DataClk_P*	41	J4	H_HA17_CCP	J1	k16	G13
142	D12	LVDS_N[30]	*DataClk_N*	43	J4	H_HA17_CCN	J1	k17	F13
143	F8	LVDS_P[31]	*SysClk_P*	45	J4	H_LA15P	J1	h19	C24
144	E8	LVDS_N[31]	*SysClk_N*	47	J4	H_LA15N	J1	h20	B24
145	F6	LVDS_P[32]	*Reset_P*	49	J4	H_LA16P	J1	g18	B27
146	E6	LVDS_N[32]	*Reset_N*	51	J4	H_LA16N	J1	g19	A27
147	D6	LVDS_P[33]	*Read_P*	53	J4	H_LA17_CCP	J1	d20	F20
148	C5	LVDS_N[33]	*Read_N*	55	J4	H_LA17_CCN	J1	d21	E20
149	C4	LVDS_P[34]	*Shutter_P*	57	J4	H_HA23P	J1	k22	L12
150	A4	LVDS_N[34]	*Shutter_N*	59	J4	H_HA23N	J1	k23	L13
151	C6	LVDS_P[35]	*CountClk_P*	61	J4	H_LA19P	J1	h22	G18
152	A6	LVDS_N[35]	*CountClk_N*	63	J4	H_LA19N	J1	h23	F18
153	A8	LVDS_P[36]	*SerEn_P*	65	J4	H_LA22P	J1	g24	C20
154	A9	LVDS_N[36]	*SerEn_N*	67	J4	H_LA22N	J1	g25	B20
155	C9	LVDS_P[37]	*SerDat2IF_P*	69	J4	H_LA29P	J1	g30	C17
156	B9	LVDS_N[37]	*SerDat2IF_N*	71	J4	H_LA29N	J1	g31	B17
157	E11	LVDS_P[38]	*SerClk_P*	73	J4	H_LA31P	J1	g33	G22
158	D11	LVDS_N[38]	*SerClk_N*	75	J4	H_LA31N	J1	g34	F22
159	C12	LVDS_P[39]	*SerDatFrIF_P*	77	J4	*H_LA32P*	*J1*	*h37*	*D21*
160	B13	LVDS_N[39]	*SerDatFrIF_N*	79	J4	*H_LA32N*	*J1*	*h38*	*C21*
						TESTPOINTS			
				2	J5	H_HA05_N	J1	e7	E16
				1	J5	H_HA05_P	J1	e6	F15
				4	J5	H_HA09_N	J1	e10	E13
				3	J5	H_HA09_P	J1	e9	F12
				6	J5	H_HA13_N	J1	e13	K16
				5	J5	H_HA13_P	J1	e12	L16
				8	J5	H_HA16_N	J1	e16	K15
				7	J5	H_HA16_P	J1	e15	L15
				10	J5	H_HA20_N	J1	e19	J13
				9	J5	H_HA20_P	J1	e18	K13
				12	J5	H_LA01__CCN	J1	d9	C26
				11	J5	H_LA01__CCP	J1	d8	D26
				14	J5	H_LA05_N	J1	d12	F30
				13	J5	H_LA05_P	J1	d11	G29
				16	J5	H_LA06_N	J1	c11	G30

15	J5	H_LA06_P	J1	c10	H30
18	J5	H_LA10_N	J1	c15	C30
17	J5	H_LA10_P	J1	c14	D29
20	J5	H_LA14_N	J1	c19	A28
19	J5	H_LA14_P	J1	c18	B28
22	J5	H_LA18__CCN	J1	c23	E21
21	J5	H_LA18__CCP	J1	c22	F21
24	J5	H_LA27_N	J1	c27	B19
23	J5	H_LA27_P	J1	c26	C19

TimePix_IF Board commanding
The IF boards are commandable via a 16-bit command word:

```
//Define a bunch of command signals.  These will change on the cycle following
//  serial_strobe
wire command_count_mode = (setup[15:10] == 6'b0000_00);
wire command_set_matrix_mode = (setup[15:10] == 6'b0000_01);
wire command_set_FSR_mode = (setup[15:10] == 6'b0000_10);
wire command_set_DAC = (setup[15:13] == 3'b001);
wire command_read_ADC = (setup[15:10] == 6'b0000_11);
wire command_read_temp = (setup[15:10] == 6'b0001_00);
wire command_set_LEDs = (setup[15:10] == 6'b0001_01);
wire command_idle = (setup[15] == 1'b1);
```

KC705 Firmware Address Space

AXI Access to Shutter_control module:

One 32-bit register, write only:

Address offset 0

bits	name	function
0	ext_trigger_enable	High to enable external trigger, low for SW trig
1	soft_trigger	Rising edge trigger for shutter sequence
2	force_shutter_low	High to hold shutter open (open = logic 0)
3	force_shutter_high	High to hold shutter closed (closed = logic 1)
8	reset_ram_addr	High to reset RAM address counter, for writing RAM
10:9	ram_select	wait_ram = 0, open_ram=1, rate_ram = 2
11	ram_write	Pulse high to write next RAM location
31:16	ram_data	16-bit RAM data

AXI Access to Serial_io module

Two 32-bit registers, write only:

Address Offset 0

bits	name	function
31:0	Txdata	Data to transmit

Address Offset 4

bits	name	function
0	GO	Rising edge to send serial data
1	side	High for IF Board A, low for B
6	length	High to send 32 bits, low to send 16
8	enable	Set high during data transmission

One 32-bit register, read only:

Address Offset 0

bits	name	function
31:0	Rxdata	Data received back from IF board

AXI Access to Timepix_input_module:

One 32-bit register, write only:

Address offset 0

bits	name	function
0	fifo_reset	Assert to reset data collection FIFOs
1	decode_on	High to convert from LFSR to binary; low to pass through
2	counter_reset	High to reset shutter counters
3	fake_enable	
11:4	fake_dav_rate	
12	dcm_reset	

Host to KC705 Interface Definition:
UDP command packets, all of 1026 bytes payload length, are sent to port 60001. Data payload as follows:

byte offset	contents
0	command_byte
1	junk
2 to 1025	data

command_byte:
0x03 = read temperatures and ADCs
0x04 = send 16 bits to IF board A, data = {byte1,byte2} ("Big Endian")
0x06 = send 16 bits to IF board B
0x08 = send 264 bits to FSR of chip A0, data = {byte1, ..., byte33}
0x09 = send 264 bits to FSR of chip A1
0x0a = send 264 bits to FSR of chip B0
0x0b = send 264 bits to FSR of chip B1
0x10 = send reset pulse to IF boards
0x11 = send reset pulse to AXI FIFOs
0x12 = send reset pulse to 10GbE core
0x13 = send reset pulse to data reception FIFOs
0x14 = send reset pulse to counters and 10GbE state machine
0b0001_1xxx = send xxx to shutter control reg[2:0]
0b0010_00xx = partial load matrix chip A0, state = xx
0b0010_01xx = partial load matrix chip A1, state = xx
0b0010_10xx = partial load matrix chip B0, state = xx
0b0010_11xx = partial load matrix chip B1, state = xx
0b0011_0000 = write 512 values to shutter WAIT RAM
0b0011_0001 = write 512 values to shutter OPEN RAM
0b0011_0010 = write 512 values to shutter RATE RAM
for these three commands, the 16-bit values are contained in bytes 2-1025, in little-endian format.

When command_byte==0x03 an echo response packet is produced, sent to port 60001:

byte offset	contents
0	ser_no[7:0]
1	ser_no[15:8]
2	junk (due to byte/int alignment in MicroBlaze)
3	junk
4	command_byte, echo
5 to 1029	data, echo
1030	tempA[7:0]
1031	tempA[15:8]
1032	tempB[7:0]
1033	tempB[15:8]

1034	adcA0[7:0]
1035	adcA0[15:8]
1036	adcA1[7:0]
1037	adcA1[15:8]
1038	adcA2[7:0]
1039	adcA2[15:8]
1040	adcA3[7:0]
1041	adcA3[15:8]
1042	adcB0[7:0]
1043	adcB0[15:8]
1044	adcB1[7:0]
1045	adcB1[15:8]
1046	adcB2[7:0]
1047	adcB2[15:8]
1048	adcB3[7:0]
1049	adcB3[15:8]
1050	junk
1051	junk

Pixel Data Format

After each shutter or series of shutters a (~4Mbit) frame of data is collected in the KC705 FIFOs and is then sent as a series of 64 UDP packets of 8206 payload bytes each. (8248 total bytes, including 14 Ethernet header, 20 IP header, and 8 UDP header). The first six payload bytes are 0 (padding for the 8-byte MAC interface). The next 8192 bytes of each packet contain pixel data; the last 8 bytes contain other information, as described here. "serno" is a 12-bit shutter serial number which increments at each shutter opening; "shutter_time" is the 32-bit time (10ns ticks) from the (hardware or software) trigger edge to the falling edge of the shutter, and "elapsed_time" is the 48-bit time, in 10ns ticks (wraps in one month). Note that the data in the last 8 bytes of packet 31 is different from that of all other packets.

The data from each pixel is a 14-bit quantity; each value is arranged in two consecutive bytes to facilitate processing in the host. The two extra bits in each 16-bit byte pair are used to indicate the first datum of a frame, and the detector side from which the data come. In the list below, A0(x,y) represents the contents of pixel (x,y) for chip A0 (IF board A, chip 0); [a:b] indicates a bit-select from that value. The data come out in interleaved chunks of 8 values from each of the two chips per IF board. In each group of eight bytes, one bit contains the "side" bit (= 0 for side A, 1 for side B), and one bit contains the sync bit (= 1 for the first value of each half-frame). A parity bit, P, is generated for each pair of values. For instance, P in byte index 9 is 1 if the sum of ones in the pair of 14-bit values A0(8,0) and A0(0,0) is odd, 0 if the sum is even.

UDP packet 0:

Index	Value	
6	A0(0,0)[7:0]	
7	0,1,A0(0,0)[13:8]	bits[7:6] = 01 indicates start of first half
8	A0(8,0)[7:0]	
9	P,0,A0(8,0)[13:8]	
10	A1(0,0) [7:0]	
11	0,0,A1(0,0)[13:8]	
12	A1(8,0) [7:0]	
13	P,0,A1(8,0)[13:8]	
14	A0(1,0)[7:0]	
15	0,0,A0(1,0)[13:8]	
16	A0(9,0)[7:0]	
…		
68	A1(15,0)[7:0]	
69	P,0,A1(15,0)[13:8]	
70	A0(16,0)[7:0]	
71	0,0,A0(16,0)[13:8]	
…		
1028	A1(255,0)[7:0]	
1029	P,0,A1(255,0)[13:8]	
1030	A0(0,1)[7:0]	
1031	0,0,A0(0,1)[13:8]	
…		
8196	A1(255,7)[7:0]	

8197	P,0,A1(255,7)[13:8]
8198	0x6, serno[11:8]
8199	serno[7:0]
8200	elapsed_time[47:40]
8201	elapsed_time[39:32]
8202	elapsed_time[31:24]
8203	elapsed_time[23:16]
8204	elapsed_time[15:8]
8205	elapsed_time[7:0]

UDP packet 1:

6	A0(0,8)[7:0]
7	0,0,A0(0,8)[13:8]
8	A0(8,8)[7:0]
9	P,0,A0(8,8)[13:8]
10	A1(0,8) [7:0]
11	0,0,A1(0,8)[13:8]
…	
8198	0x6, serno[11:8]
8199	serno[7:0]
8200	elapsed_time[47:40]
8201	elapsed_time[39:32]
8202	elapsed_time[31:24]
8203	elapsed_time[23:16]

UDP packet 31:

6	A0(0,248)[7:0]
7	0,0,A0(0,248)[13:8]
8	A0(8,248)[7:0]
9	P,0,A0(8,248)[13:8]
…	
8196	A1(255,255)[7:0]
8197	P,0,A1(255,255)[13:8]
8198	0x12
8199	0x34
8200	0x5, serno[11:8]
8201	serno[7:0]
8202	shutter_time[31:24]
8203	shutter_time[23:16]
8204	shutter_time[15:8]
8205	shutter_time[7:0]

UDP packet 32:

Index	Value	
6	B0(0,0)[7:0]	
7	1,1,B0(0,0)[13:8]	bits[7:6] = 11 indicates start of second half

8	B0(8,0)[7:0]
9	P,0,B0(8,0)[13:8]
10	B1(0,0) [7:0]
11	0,0,B1(0,0)[13:8]
12	B1(8,0) [7:0]
13	P,0,B1(8,0)[13:8]
14	B0(1,0)[7:0]
15	1,0,B0(1,0)[13:8]
16	B0(9,0)[7:0]
…	
68	B1(15,0)[7:0]
69	P,0,B1(15,0)[13:8]
70	B0(16,0)[7:0]
71	0,0,B0(16,0)[13:8]
…	
1028	B1(255,0)[7:0]
1029	P,0,B1(255,0)[13:8]
1030	B0(0,1)[7:0]
1031	0,0,B0(0,1)[13:8]
…	
8196	B1(255,7)[7:0]
8197	P,0,B1(255,7)[13:8]
8198	0x6, serno[11:8]
8199	serno[7:0]
8200	elapsed_time[47:40]
8201	elapsed_time[39:32]
8202	elapsed_time[31:24]
8203	elapsed_time[23:16]
8204	elapsed_time[15:8]
8205	elapsed_time[7:0]

UDP packet 33:

6	B0(0,8)[7:0]
7	0,0,B0(0,8)[13:8]
8	B0(8,8)[7:0]
9	P,0,B0(8,8)[13:8]
10	B1(0,8) [7:0]
11	0,0,B1(0,8)[13:8]
…	
8198	0x6, serno[11:8]
8199	serno[7:0]
8200	elapsed_time[47:40]
8201	elapsed_time[39:32]
8202	elapsed_time[31:24]
8203	elapsed_time[23:16]
8204	elapsed_time[15:8]

8205	elapsed_time[7:0]

UDP packet 63:

6	B0(0,248)[7:0]
7	0,0,B0(0,248)[13:8]
8	B0(8,248)[7:0]
9	P,0,B0(8,248)[13:8]
…	
8196	B1(255,255)[7:0]
8197	P,0,B1(255,255)[13:8]
8198	0x6, serno[11:8]
8199	serno[7:0]
8200	elapsed_time[47:40]
8201	elapsed_time[39:32]
8202	elapsed_time[31:24]
8203	elapsed_time[23:16]
8204	elapsed_time[15:8]
8205	elapsed_time[7:0]

C Language Library

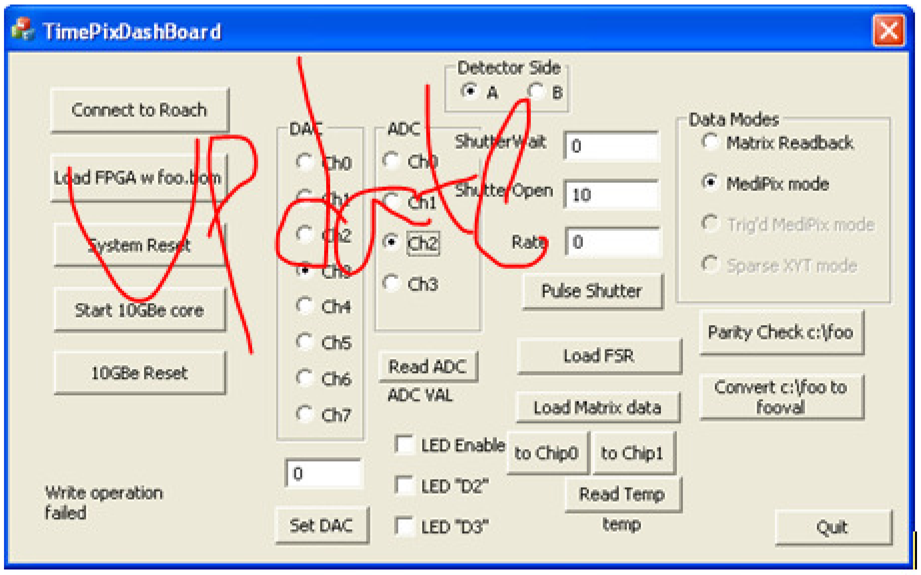
Extract from the TimePixInterface.h header file describing the functions available to the user appears below. The intent is to encapsulate the details of the lower-level interfaces, described later, in this library. Comments below describe the functions.

```
/////////////////////////////////////////////////////////////////////////////
//Functions to be called by user
//
//Starts up the Winsock connection, starts RxHKData thread, creates KC705 Socket
int Connect(const char* pcHost, int nPort);

//Closes the KC705 socket
int Disconnect();

//Read temp sensor from ASIC board (selected by global "which_side" variable)
// returns temp in degrees C
double readtemp();

//Pulse the ASIC RESET line to both IF boards, so resetting all ASICs
bool sys_reset();

//Pulse the 10GbE core RESET line high then low
bool tengbe_reset();

//Pulse the shutter control "GO" bit high then low
bool pulse_shutter();

//Force the shutter open (true) or stop forcing it (false)
bool force_shutter(bool open);

//write a value between 0 and 1023 to a channel between 0 and 7
//returns false if something goes wrong, including if either parameter
// is out of range
bool write_dac(int channel, int value);

//writes a value to location 0 of the shutter timing control BRAM, LS 16 bits
// returns false if anything goes wrong, including if the
//  value is out of  range (0 to 65535)
bool set_wait(int wait);

//writes a value to location 0 of the shutter timing control BRAM, MS 16 bits
// returns false if anything goes wrong, including if the
//  value is out of  range (0 to 65535)
bool set_open(int open);

//writes a value to location 0 of the shutter rate control BRAM, MS 16 bits
// returns false if anything goes wrong, including if the
//  value is out of  range (0 to 15)
bool set_rate(int value);

//Write values to all 256 locations of rate_ram.
bool load_rate_ram(int * rate_array);

//Write values to all 256 locations of rate_ram.
bool load_shutter_ram(unsigned int * open_array, unsigned int * wait_array);
```

```
//writes a value to the packet_wait register, 0 to 255, causing readout
//  data to be slowed by a factor of value
bool set_skip(int value);

//Read a value from one of the 4 ADC channels, 0 to 3.
//  Returns a 12-bit value with approximately 2200mv full scale,
// or a -1 if something goes wrong
int read_adc(int channel);

//Set the "SIDE" bit to 0 or 1, to determine which
// of the two pairs of ASICs is being addressed
//Argument can be 0 or 1 only
bool set_side(int side);

//Set the data mode bits in the control register
bool set_mode(int mode);

//Set the GO bit in the control register
bool set_GO(int GO);

//Start up the 10GBe tap server with IP address IP and port port
bool tap_start(int IP, int port);

//Check the even parity of numwords 32-bit words in raw binary file saved by WireShark
//  returns number of errors.
int parity_check(const char * filename, int numwords);

//Read in the raw binary file (saved by Wireshark).  Convert to 14-bit values (4 per 64
bytes)
//  represented in ascii hex
void convert_to_values(const char * infile, const char * outfile);

//Read in the raw binary file (saved by Wireshark).  Convert to 14 bit values, and
compare
// the data from chip n (0 or 1) to the 256 by 256 matrix argument.
unsigned int * check_matrix(const int * pmatrix, int chip, const char * infile, int
write_log);

//Load the 256*256 14 bit values from int array "data[256][256]", into the serial_data_in
ram
//returns true if all OK
bool load_matrix_ram(const int * data);

//send the data from the serial_data_in ram to chip (0 or 1)
bool load_matrix(int chip);

//Set the ASICs back in count mode.  Do this before taking data or reading back the
matrix,
//  so that the ASIC clock pair is no longer controlled by the ROACH.
//  One command sets both ASICs
bool count_mode(void);

//Load the 256 bit data, from char array "data", into the Fast Shift Register of one chip
//returns the chip ID, and 999 if something went wrong
int load_FSR(const char * data, int chip);

 //Turn the LFSR-to-binary decode function on (1) or off (0)
bool set_decode(int decode);
```

```
//Set or clear the IGNORE bit.  When set, this causes the input FIFOs to
//  ignore any incoming data (but does not clear them)
bool set_ignore(int ignore);

//Check one of the firmware parity counters:
//  0=ZDOK0, chip0; 1=ZDOK0, chip1; 2=ZDOK1, chip0; 3=ZDOK1, chip1
//  returns number of errors
int check_parity_counter(int whichone);
```

<u>TimePix DashBoard GUI</u>



Detector Side: [✓ A] B

DAC: Ch0 Ch1 Ch2 [✓ Ch3] Ch4 Ch5 Ch6 Ch7 — 0

ADC: Ch0 Ch1 [✓ Ch2] Ch3

ShutterWait 0

Shutter Open 10

Rate 0

Data Modes: Matrix Readback [✓ MediPix mode] Trig'd MediPix mode Sparse XYT mode

☐ LED Enable ☐ LED "D2" ☐ LED "D3"

Write operation failed

Connect to Roach
Establishes the socket connection to the Roach

Load FPGA with foo.bof
The compiled Roach firmware is contained in a .bof file, in the /boffiles directory of the Roach operating system. Rename this file "foo.bom" and ensure that the "execute" permission is set; then, this button commands the Roach to load the FPGA file

System Reset
Pulses the RESET bit of the control register, sending a reset pulse to the IF boards.

Start 10GbE core
Issues the "tap-start" command to the Roach

10GbE Reset
Pulses the RS_10G bit of the Control register, resetting the 10GbE core.

Side A/B
The button selects whether the commands are sent to IF board A (controlled from Roach ZDOK 0) or IF board B (ZDOK 1).

Set DAC
Enter a value from 0 to 1023 and select a channel from 0 to 7, then click the button. Channel 0 is the TEST_HI level to the ASIC board, channel1 is the TEST_LO, and the remaining six are available on the IF board.

Read ADC

Select a channel and click the button. 12-bit ADC value is displayed. Channel 0 is the DAC_OUT level from ASIC 0; channel 1 from ASIC 1. Ch 2 is the IF board 2.2v supply (ADC full scale is 3.3v, so Ch 2 should measure about 2.2/3.3 * 4096 = 2730). Ch 3 is the IF board 5v supply divided by 3, so should measure about 2048 counts. (Ch 2 and Ch 3 can be used for other purposes by removing the jumpers on the IF board).

LED control
Check the enable box to allow IF board LEDs D2 and D3 to be turned on and off via the other two check boxes. If the enable box is unchecked, D2 will flash each time a command is sent to the IF board, and D3 will flash each time a packet is sent back to the ROACH

Shutter Wait/ Shutter Open/ Rate
Enter values for the first location of the shutter_ram and the rate_ram. These values set the width and delay of the shutter pulse, and the count clock frequency during the pulse, as described below. Only the first ram location is accessible.

Pulse Shutter
In Medipix mode, click this button to produce a single shutter pulse.

Data Modes
Only Medipix mode (software control of the shutter) is supported so far.

Load FSR
Send 256 bits to the Fast Shift Register of one ASIC. The data contents and the ASIC select are hard-coded at this point (just for debug).

Load Matrix Data
Take a 256 by 256 array of 14-bit values, transform to the appropriate bit stream, and load into the serial_data_ram. Data contents are hard-coded for debug.

To Chip0/ Chip1
Send the “Set Matrix” command to one or the other ASIC, which causes the data loaded into serial_data_ram to be sent out to the ASIC.

Parity check c:\foo
Check the parity of the binary file foo stored in the root of the C: drive. This would have been stored by the WireSharq “Follow UDP stream” function, in raw binary format.

Convert c:foo to fooval
Convert that binary file to an ASCII text file consisting of four columns of 14-bit values.

Operating Modes

Below are described the several basic system functions and the host←→ Roach interactions necessary to perform them.

Initialization

Host sets all bits of the control register to 0.
Host pulses RESET
Wait 5 sec
Host pulses RS_10G
Wait 5 sec

Set up Matrix

Each of the four sensor chips has a 256-by-256 matrix of 14-bit registers which controls the operating characteristics of each pixel. Eight bits of each 14 are used; the other 6 are ignored. The bit assignments are as in Fig 35, p34 of the Timepix Manual. The arrangement of the 256*256*14 = 917,504 bits required to set each chip up is shown in Table 16, p44. I'll follow the bit-naming convention used there.

Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0400_0000	Send "SetMatrix0" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 10	Set to send long packet
host → Roach	SetupIn <= matrix data	917,512 bits for ASIC A0
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0600_0000	Send "SetMatrix1" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 10	Set to send long packet
host → Roach	SetupIn <= matrix data	917,512 bits for ASIC A1
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
host→ Roach	SIDE <= 1	Select the other IF board
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0400_0000	Send "SetMatrix0" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 10	Set to send long packet
host → Roach	SetupIn <= matrix data	917,512 bits for ASIC B0
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0600_0000	Send "SetMatrix1" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 10	Set to send long packet
host → Roach	SetupIn <= matrix data	917,512 bits for ASIC B1
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	

Set up FSR registers/ read device IDs

Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0800_0000	Send "SetFSR0" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 01	Set to send medium packet
host → Roach	SetupIn <= fsr data	264 bits for ASIC A0
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= device ID A0	
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0A00_0000	Send "SetFSR1" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 01	Set to send medium packet
host → Roach	SetupIn <= fsr data	264 bits for ASIC A1
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= device ID A1	
host→ Roach	SIDE <= 1	Select the other IF board
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0800_0000	Send "SetFSR0" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 01	Set to send medium packet
host → Roach	SetupIn <= fsr data	264 bits for ASIC B0
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= device ID B0	
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'h0A00_0000	Send "SetFSR1" command to IF board
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
host → Roach	LENGTH <= 01	Set to send medium packet
host → Roach	SetupIn <= fsr data	264 bits for ASIC B1
host→ Roach	GO_SET <= 1	Transition sends packet
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= device ID B1	

Set DAC

Send a 10-bit value to one of the 8 DAC channels (channels 0 and 1are used to set the High and Low Test Pulse levels, the other 6 channels are brought out to header J5 on the IF board)

Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'hyyyy_0000	Send "SetDAC" command to IF board
	16-bit field yyyy = {001,DACADD[2:0],DACVAL[9:0]}	
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	

Set IF board LED

Turn on/off D2 and/or D3 (D1 is the "FPGA DONE" LED)

Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32'hyyyy_0000	Send "SetLED" command to IF board

	16-bit field yyyy = {000101, LEDENABLE, LED[1:0], X[6:0]}
	LEDENABLE = 1 gives control of the LED,
	LEDENABLE = 0 gives the LEDs other functions
	X = don't care	

host→ Roach	GO_SET <= 1	Transition sends packet to IF

host→ Roach	GO_SET <= 0	

Read ADC

Read a 12-bit value from one of the 4 ADC channels. Channel 0 is the “DAC_OUT” of ASIC0, channel 1 the “DAC OUT” of ASIC1.
Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32’hyyyy_0000	Send “ReadADC” command to IF board
	16-bit field yyyy = {000011, ADC_CHAN[1:0],X[7:0]}	
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
wait at least 25us		
host → Roach	SetupIn <= 32’hyyyy_0000	Send “ReadADC” command to IF board
	16-bit field yyyy = {000011, ADC_CHAN[1:0],X[7:0]}	
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= ADC value	

(Command is sent twice. First command sets the ADC multiplexer; second reads the value.)

Read Temperature

Read the 16-bit temperature value from one of two temp sensors on the ASIC board, one associated with each IF board
Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	LENGTH <= 00	Set to send short packet
host → Roach	SetupIn <= 32’hyyyy_0000	Send “ReadTemperature” command to IF board
	16-bit field yyyy = {000100, X[10:0]}	
host→ Roach	GO_SET <= 1	Transition sends packet to IF
host→ Roach	GO_SET <= 0	
Roach→ host	SetupOut <= temperature	

Read Back Matrix Data

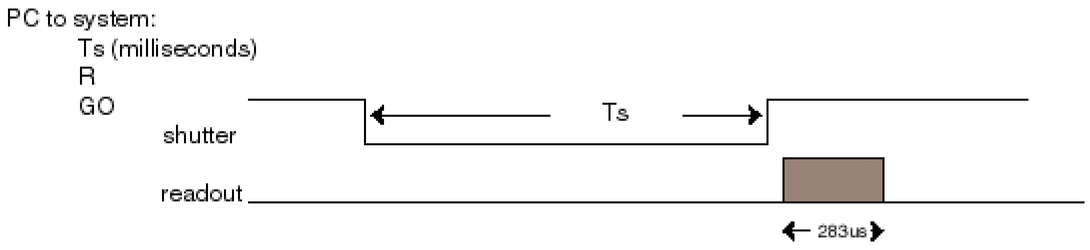
Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	DATAMODE <= 4’h0	Select “Matrix Read” mode
host → Roach	Shutter_time in[0] <= {32’h0}	Flag to indicate end of data
host → Roach	GO <= 1	Initiate counting/readout operation

Matrix data will appear as UDP packets in the pixel data stream

Data acquisition/readout: “Medipix” mode

PC tells the system “Integrate with clock R for Ts ms”, then issues a “GO” command. System opens shutter for the desired time, reads out data, and passes it to the PC.

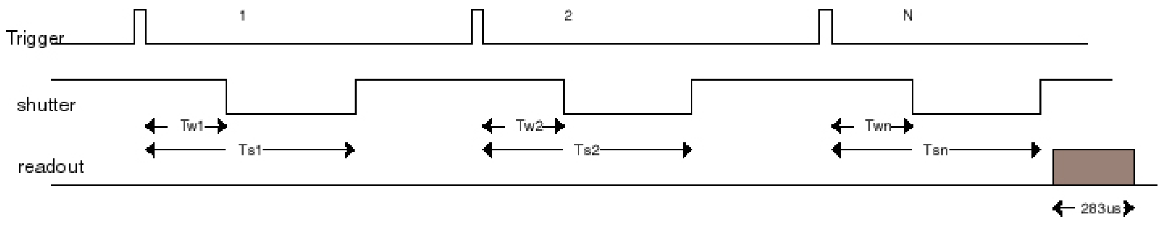


Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	DATAMODE <= 4’h1	Select “Medipix” mode

host → Roach	Shutter_time in[0] <= {Ts(16 bits), 16'h0}	Set up shutter time
host → Roach	Shutter_rate in[0] <= {28'h0, rate}	Set up count rate
host → Roach	Shutter_time in[1] <= {32'h0}	Flag to indicate end of data
host → Roach	GO <= 1	Initiate counting/readout operation

Data acquisition/readout: "Triggered Medipix" mode.
PC tells the system a clock rate, start time, relative to a trigger, a stop time, and a number of shutter openings, then issues a "GO" command. System controls the shutter as commanded, then reads out the data (once only) and passes the data to the PC.

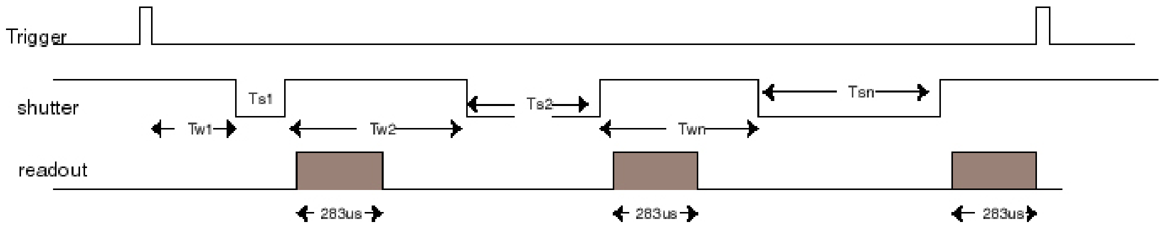


Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	DATA_MODE <= 4'h2	Set to triggered mode
host → Roach	Shutter_time in[0] <= {Ts1, Tw1}	Set up shutter time
host → Roach	Shutter_rate in[0] <= {28'h0, R1}	and count rate for each trigger
host → Roach	Shutter_time in[1] <= {Ts2, Tw2}	
host → Roach	Shutter_rate in[1] <= {28'h0, R2}	
…		
host → Roach	Shutter_time in[n-1] <= {Tsn, Twn}	
host → Roach	Shutter_rate in[n-1] <= {28'h0, Rn}	
host → Roach	Shutter_time in[n] <= {32'h0}	Flag to indicate end of data
host → Roach	GO <= 1	Initiate counting/readout operation

Data acquisition/readout: "Sparsified XYT" mode

PC sends to the system a set of N parameter trios to control the time and clock rate for each shutter opening. The system controls the shutter as commanded, reads out the data following each shutter closure, removes all "0" count data, and passes the data back to the PC, in event-list form X, Y, T for each event. X and Y are the (9-bit) pixel location values, and T is a 15-bit value calculated from the pixel contents and the Bn, En, Rn parameters.



Start with all control bits set to 0.

Data transfer	Roach register	Comment
host → Roach	DATA_MODE <= 4'h3	Set to sparsified mode
host → Roach	EN_SET <= 1	Enable the setup link
host → Roach	SetupIn <= 0x0000_0000	Send "ReadMode" command to IF board
host → Roach	EN_SET <= 0	Falling edge causes command to take effect
host → Roach	Shutter_time in[0] <= {Ts1, Tw1}	Set up shutter time
host → Roach	Shutter_rate in[0] <= {28'h0, R1}	and count rate for each trigger
host → Roach	Shutter_time in[1] <= {Ts2, Tw2}	
host → Roach	Shutter_rate in[1] <= {28'h0, R2}	
…		
host → Roach	Shutter_time in[n-1] <= {Tsn, Twn}	
host → Roach	Shutter_rate in[n-1] <= {28'h0, Rn}	
host → Roach	Shutter_time in[n] <= {32'h0}	Flag to indicate end of data
host → Roach	GO <= 1	Initiate counting/readout operation

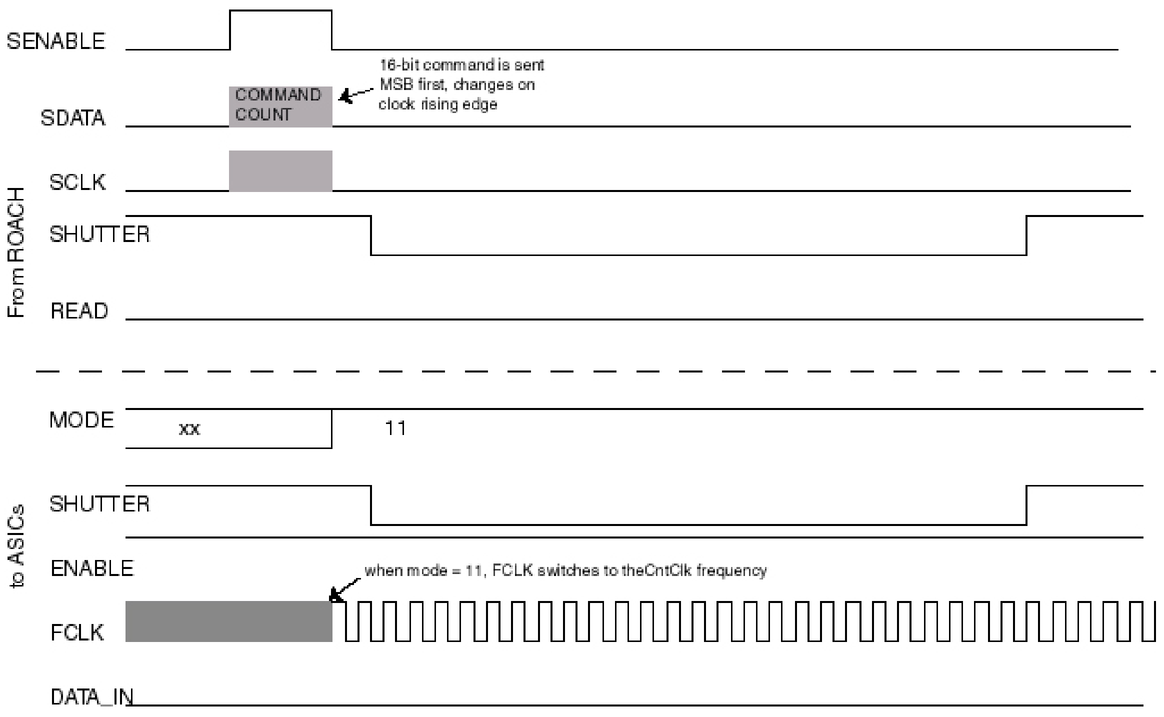
Roach ←→ Interface Board
Electrical interface consists of 40 pairs terminated in a QSH-040-01-F-D-DP connector (Samtec) at both ends; cable is SamtecHQDP-040-length-TBR-SBL-1
Electrical levels are LVDS

Pair	Name	Function	I/O (rel ROACH)
0-27	PData	Pixel Data from ASICs	I
28	Parity	Parity bit for error checking	I
29	Sync	Pixel data sync- marks beginning of dump	I
30	DataClock	Pixel Data Clock	I
31	SysClk	Master system clock (100MHz CW)	O
32	Reset	System Reset	O
33	Read	Pulse high to start pixel data read out	O
34	Shutter	High to integrate	O
35	CntClk	ASIC count clock, programmable freq	O
36	SEnable	High to enable serial (housekeeping) link	O
37	SDataOut	Serial (HK) data to IF board	O
38	Sclk	Serial (HK) clock	O
39	SDataIn	Serial (HK) data from IF board	I

-

Operating modes

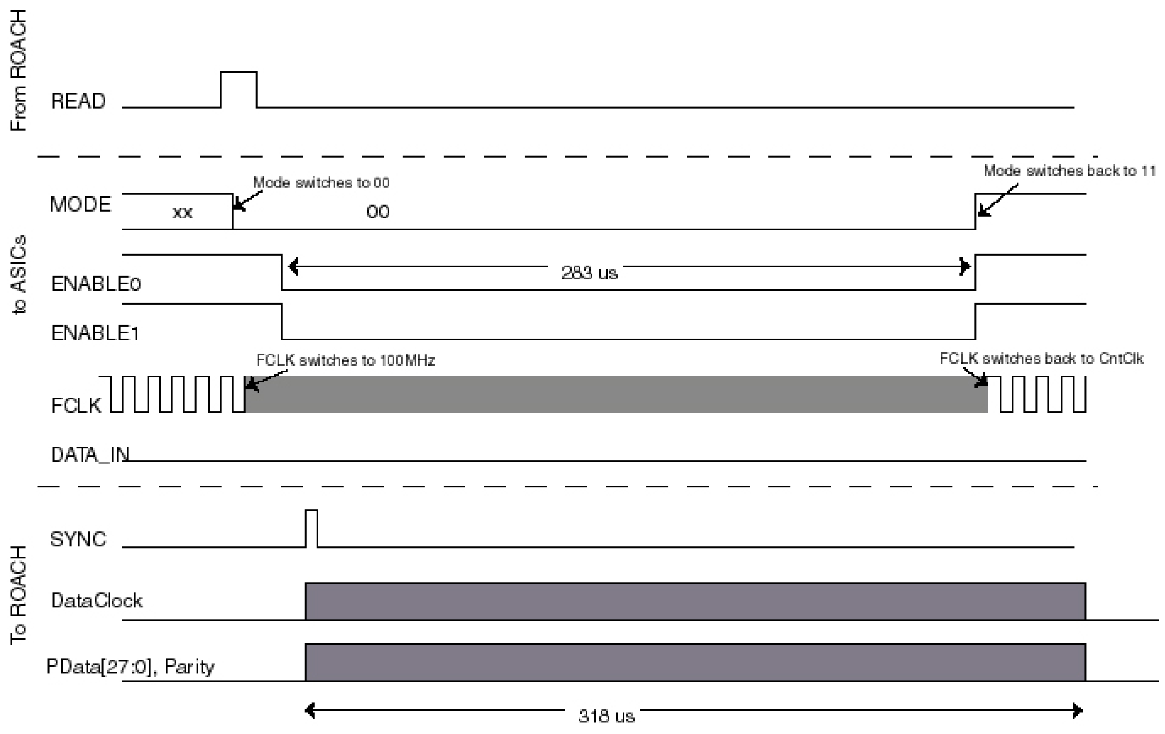
1. Counting mode



SENABLE
SDATA
COMMAND COUNT
16-bit command is sent MSB first, changes on clock rising edge
SCLK
SHUTTER
READ
From ROACH
MODE
xx
11
SHUTTER
ENABLE
when mode = 11, FCLK switches to theCntClk frequency
FCLK
DATA_IN
to ASICs

A “COUNT” command sent over the serial link sets the ASIC mode bits to 11 and switches the FCLK to the CntClk frequency, then the SHUTTER signal enables the ASIC counting function. Other control lines to the ASICs are quiet.

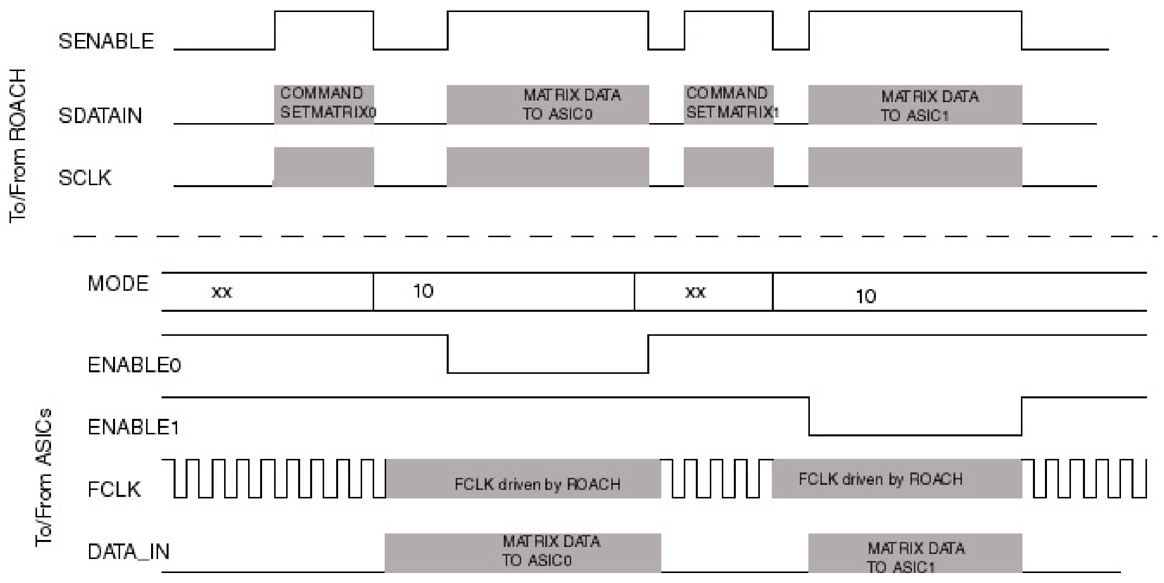
2. Pixel Readout



A single pulse on the READ line starts the readout. ASIC mode bits are forced to 00 and FCLK is switched to 100MHz. The pixel data is read out of the ASICs at 100MHz. At the end of the readout period, FCLK is returned to the CntClk rate, and mode is switched back to 11 (counting mode).

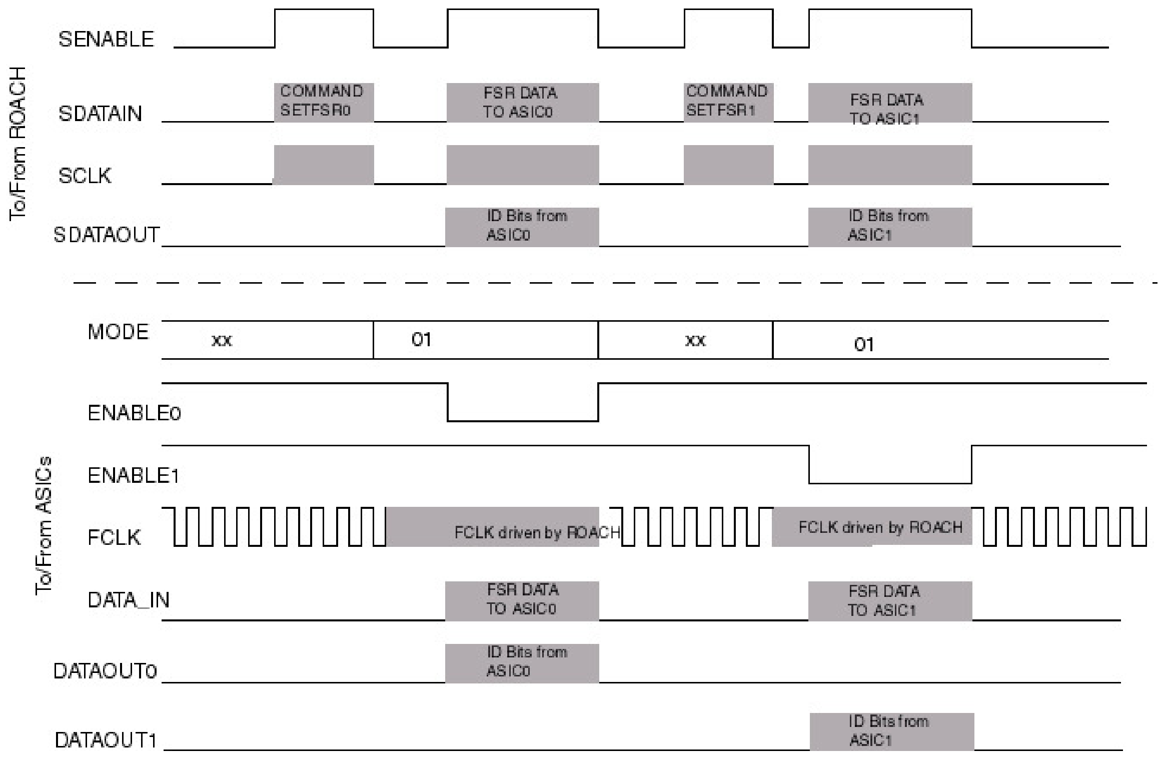
Data is clocked into the ROACH at 112.5 MHz double data rate, so 225Mb/s. This allows the data transfer to almost keep up with the input data from the two ASICs: 2 chips * 32 bits * 100MHz = 6.4Gb/s input rate. A sync bit and a parity bit are added to each pair of (14-bit) counters, so the output rate is 15/14 of this, or 6.857Gb/s. Transmitted over 30 pairs, this requires a clock of ½ * 6857/30 = 114.28MHz. Clocking at 112.5 MHz causes the FIFO to fill a little bit (there is not enough memory in the Spartan to buffer the entire approx 2Mbits of pixel matrix data).

3. Setting the matrix



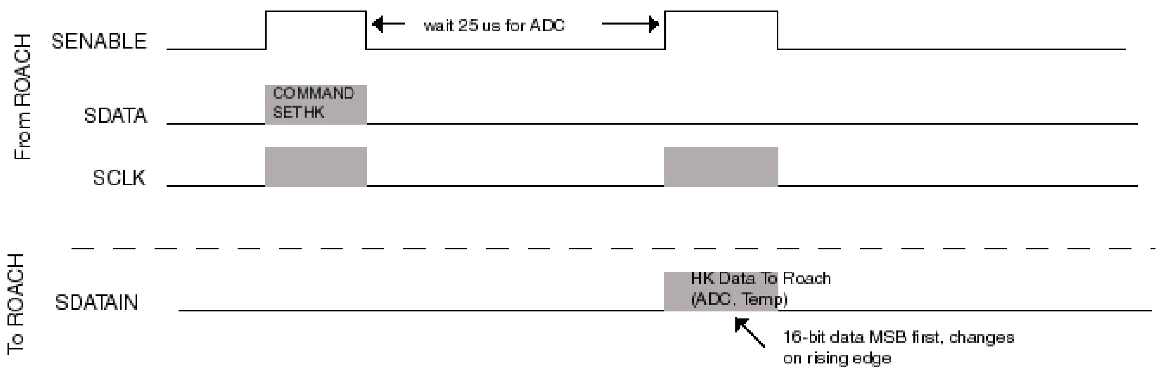
The SETMATRIX0 command is sent via the 3-wire serial link, which causes the FCLK, ENIN0, and DATA_IN signals to the ASIC to be driven from the SCLK, SENABLE, and SDATA lines. The 917,504 bits of matrix setup data are shifted out to ASIC0, then the process is repeated with the SETMATRIX1 command, to set up ASIC1.

4. Setting the FSR register and read back the device ID



As above, but the mode lines are set to 01. The identity bits are read out of each ASIC

5. Set/Read housekeeping data


From ROACH
SENABLE
SDATA
SCLK
wait 25 us for ADC
COMMAND SETHK
To ROACH
SDATAIN
HK Data To Roach (ADC, Temp)
16-bit data MSB first, changes on rising edge

This is used to set up the DAC values (which control the levels of the test pulser), the ENABLE_TESTPULSE control to the ASICs, and any other function we might need. Also to read back the ADC and temp sensor data. One of the SET_HK commands is sent via the 3-wire link; in the case of the ADC or temp sensor, the output data is sent back via the SDATAIN line.

Serial commands (16 bits each):

```
0000_00ab_xxxx_xxxx      Count Mode, set POL to a, ENABLE_TP to
b
0000_010x_xxxx_xxxx      Set Matrix0
0000_011x_xxxx_xxxx      Set Matrix1
0000_100x_xxxx_xxxx      Set FSR0
0000_101x_xxxx_xxxx      Set FSR1
0000_11aa_xxxx_xxxx      Read ADC channel aa
0001_00xx_xxxx_xxxx      Read Temp
0001_01ea_bxxx_xxxx      Set IF board LEDs:
                         e = enable, set to 1 to command LEDs
                         (else they have soft functions)
                         a = LED D2 (1 = on)
                         b = LED D3 (1 = on)
001a_ aabb_bbbb_bbbb     Set DAC channel aaa to value
                         bb_bbbb_bbbb
```

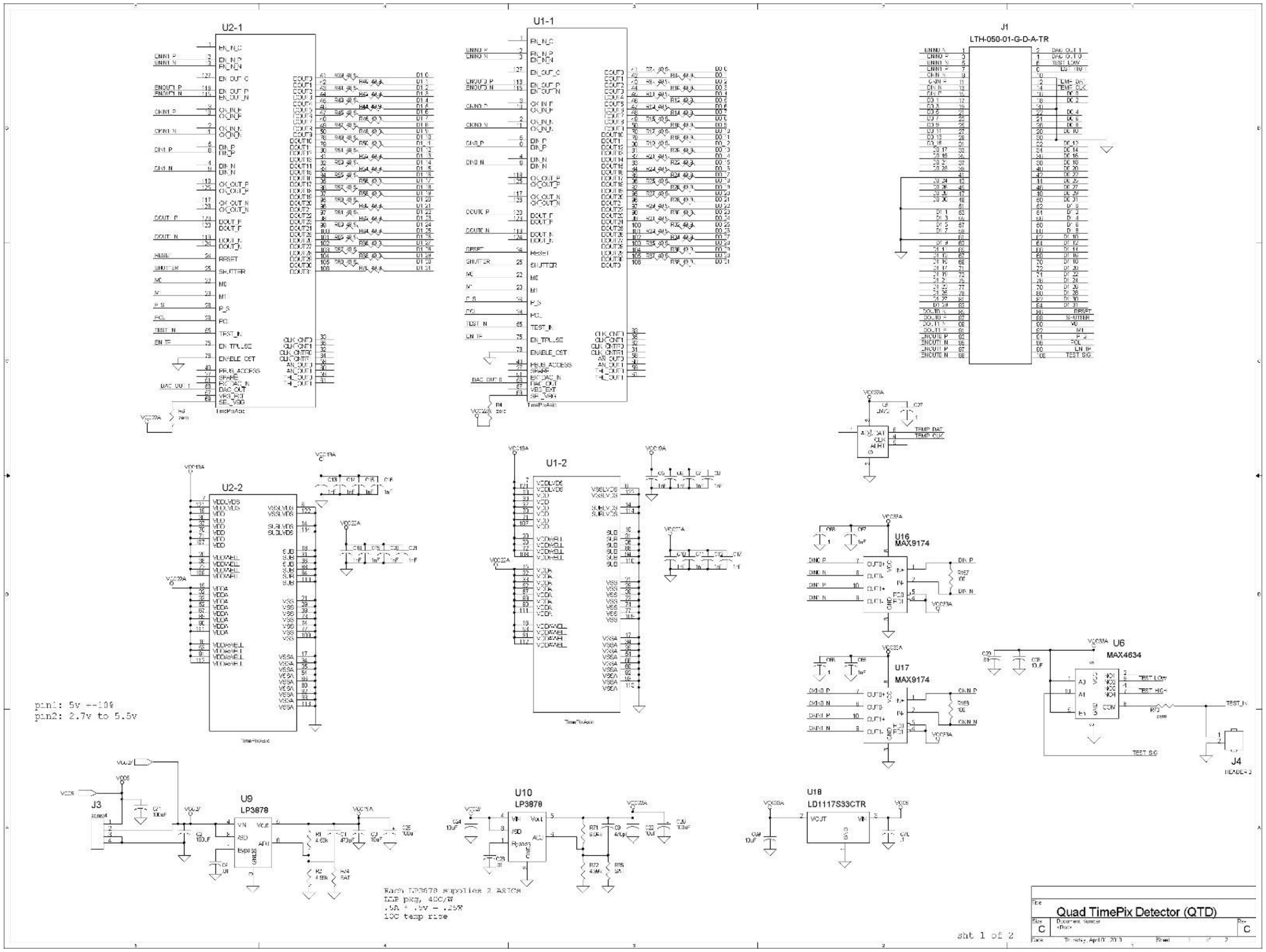
Documents

Top Assembly BOM and Drawing tree (EAG-QTD-013):

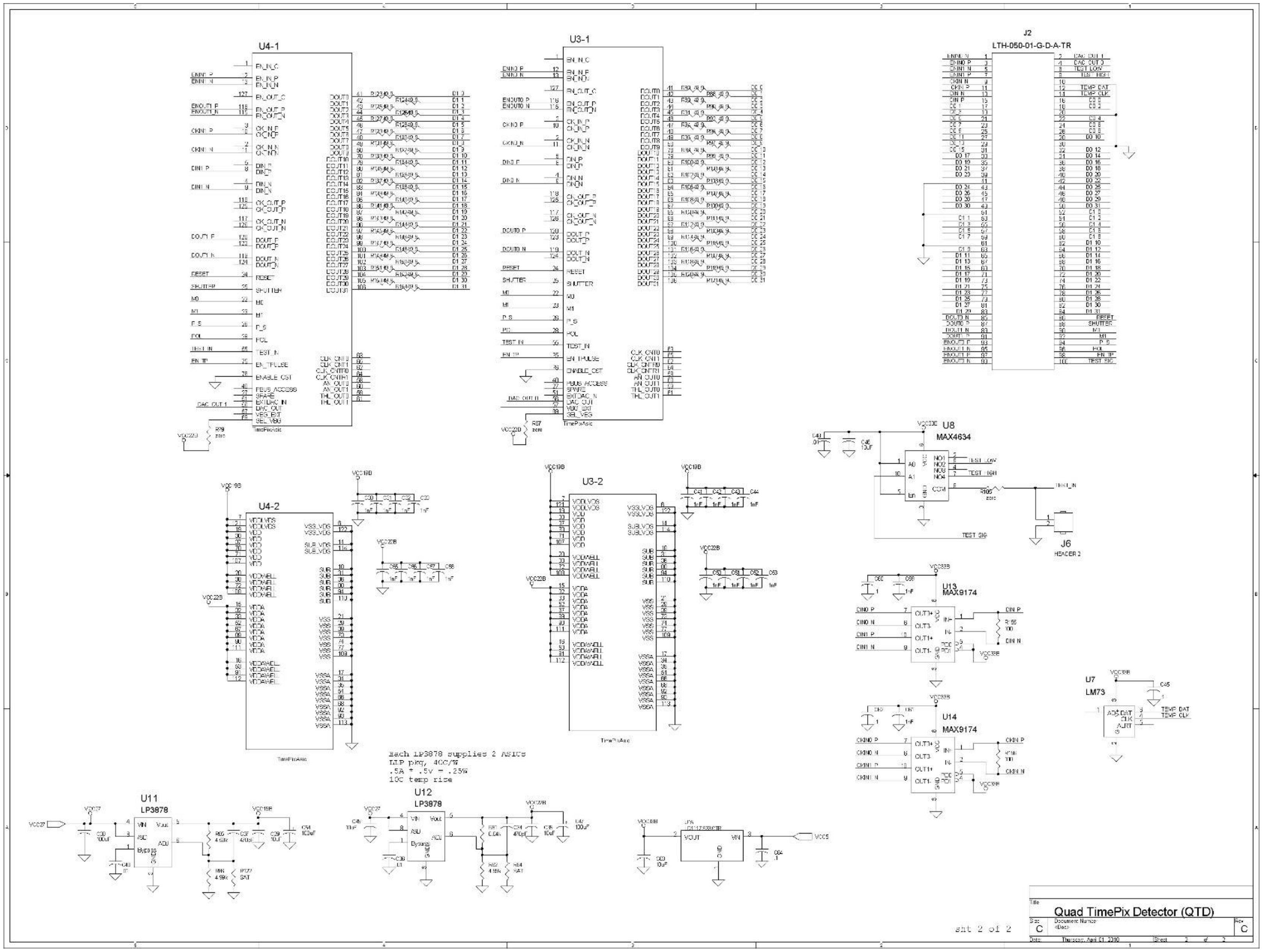
Quad TimePix Detector System					
Top Assembly Bill of Materials					
Qty	Description	Source	Part or Dwg Number	Dwg Type	Notes
1	**ASIC PCB**	build	EAG_QTD_007.xls	BOM	
ref		Evenstar	EAG-QTD-008.zip	PCB fab dwg and files	This PCB has a major problem with the layout of the voltage regulators. We'll probably respin rather than build more like this.
ref			EAG-QTD-009.zip	PCB assy dwg & files	
ref			EAG-QTD-006.pdf	schematic, 2 shts	
2	**IF PCB**	build	EAG_QTD_003.xls	BOM	See assembly notes on worksheet 2 of the xls. These two items should be addressed when more PCBs need to be fabbed.
ref		Evenstar	EAG-QTD-004.zip	PCB fab dwg and files	
ref			EAG-QTD-005.zip	PCB assy dwg & files	
ref			EAG-QTD-002.pdf	schematic, 2 shts	
2	**Enclosure for IF Board**	Compac-RF	TimePixIFBox.pdf	sketch	This needed mods- review before reordering.
2	**Flex PCB**	Samtec	SCDL-101368-3-LSH-LSH-1	Samtec Source Control Dwg	I believe this is a unique SCD number, assigned by Samtec when I ordered this custom flex PCB. But it would be good to check before ordering more.
2	**Cables**	Samtec	HQDP-040-40.00-TBR-SBL-1		Standard Samtec PN
2	**ZDOK Adapter**	Sunstone	TimePix Adapter.123	PCB design file	This is a proprietary design format from Sunstone. No schematic exists
			ZDOK_adapter.pdf	Schematic	
1	**Voltage Regulator Board**	build	TIMEPIX_POWER.BOM.xls	BOM	
		expressPCB	PowerPCB.pcb	Express PCB design file	Proprietary ExpressPCB format
			TimePix Power PCB.pdf	schematic	

1	**Power Cable**	build	TimePixPowerCable.pdf	schematic	
1	**Roach Board**	Digicom	"Roach Board Assembly"		We supplied the FPGA to Digicom
1	**FPGA for Roach**	Xilinx	XC5VSX95T-1136		Xilinx donated 2 of these
1	**Enclosure for Roach**	ProtoCase	roach_case_rev1.cas	Enclosure design file	A proprietary design format from ProtoCase. This should be revisited before buying more- there were some mods to be done.
1	**Power supply**	Digikey	237-1307-ND		12vdc 150W supply. Nothing special
1	**Power supply**	MiniBox	PicoPSU-120		

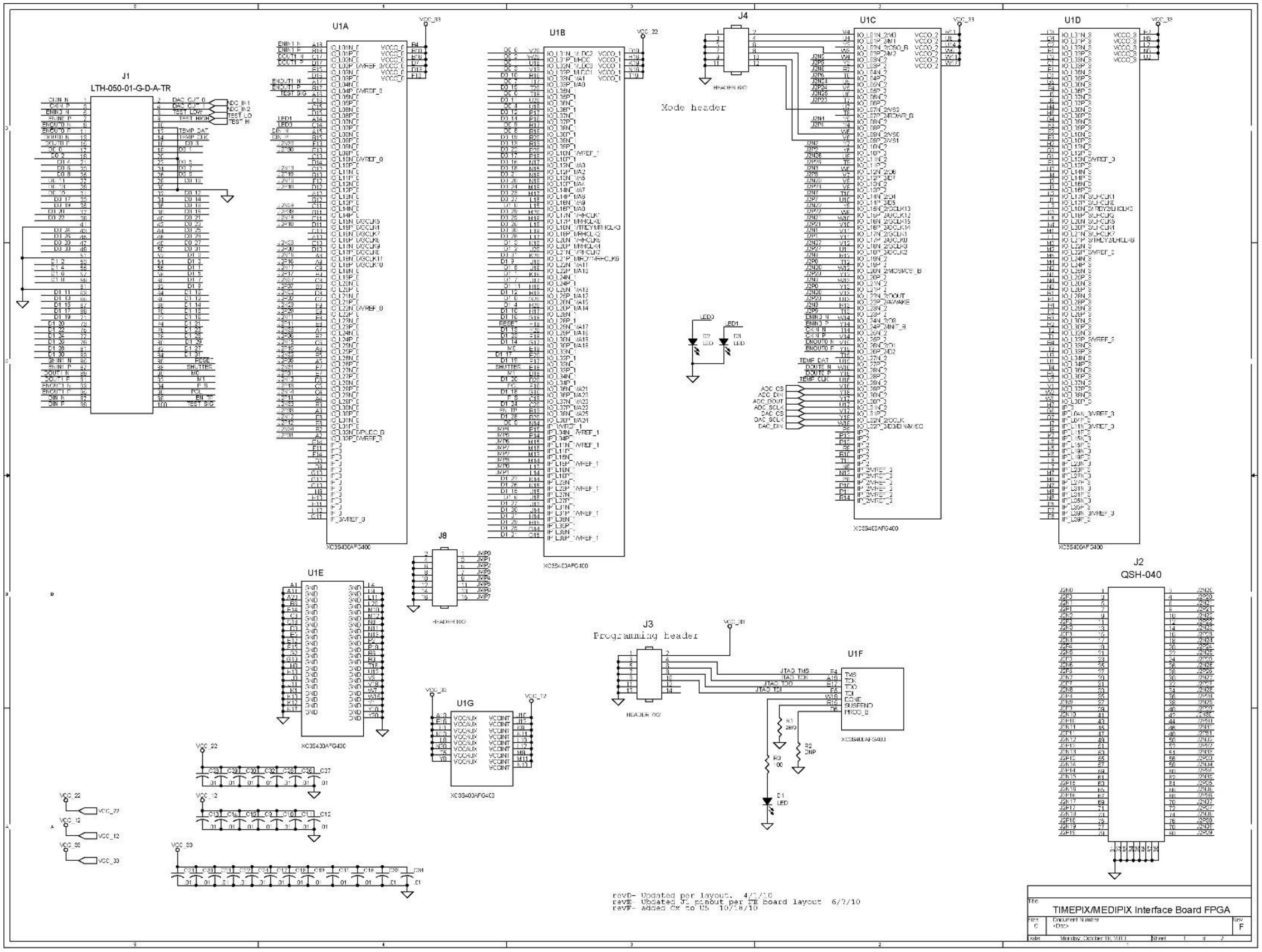
Schematics



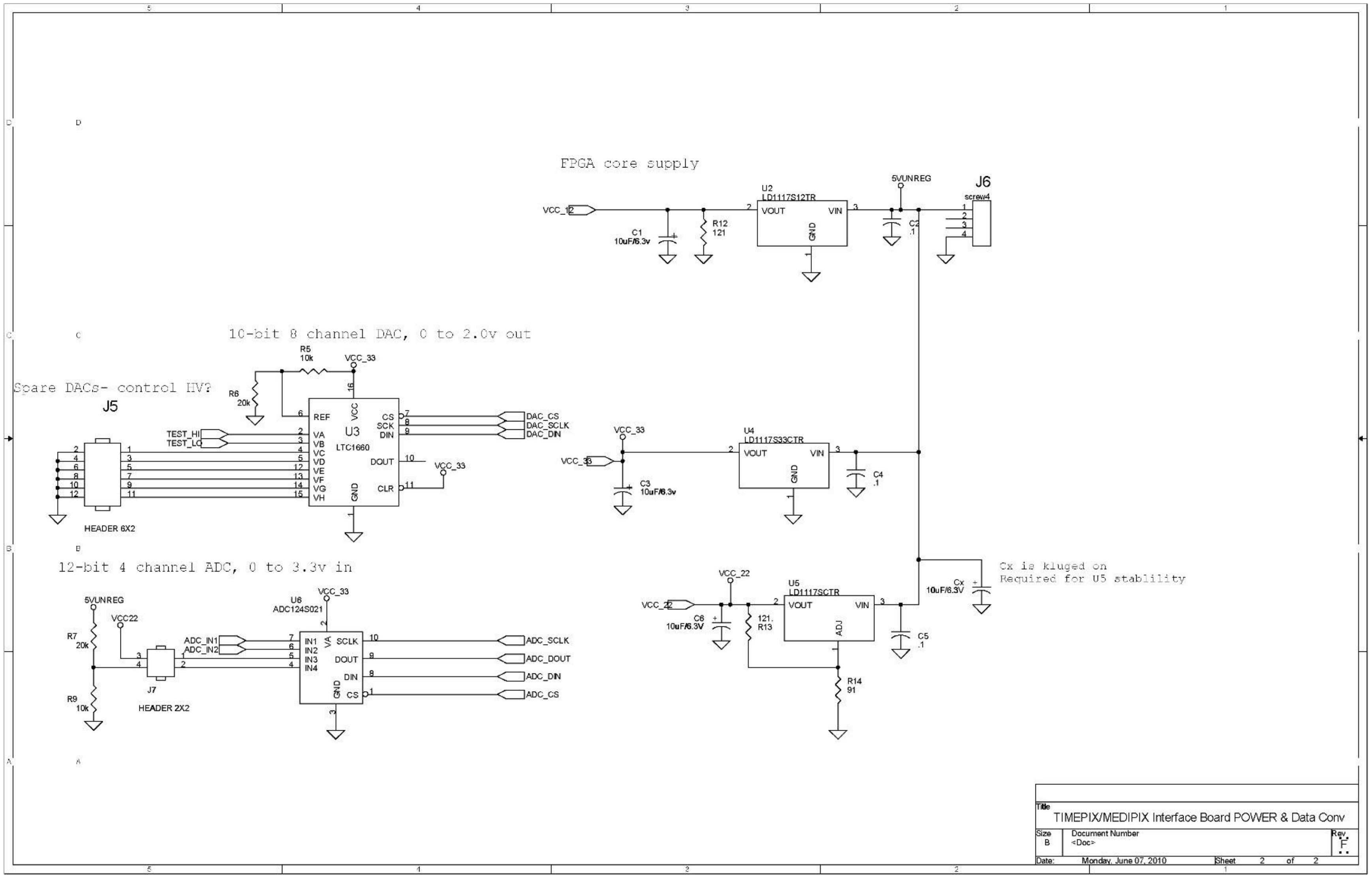
ASIC Board, sht 1 of 2



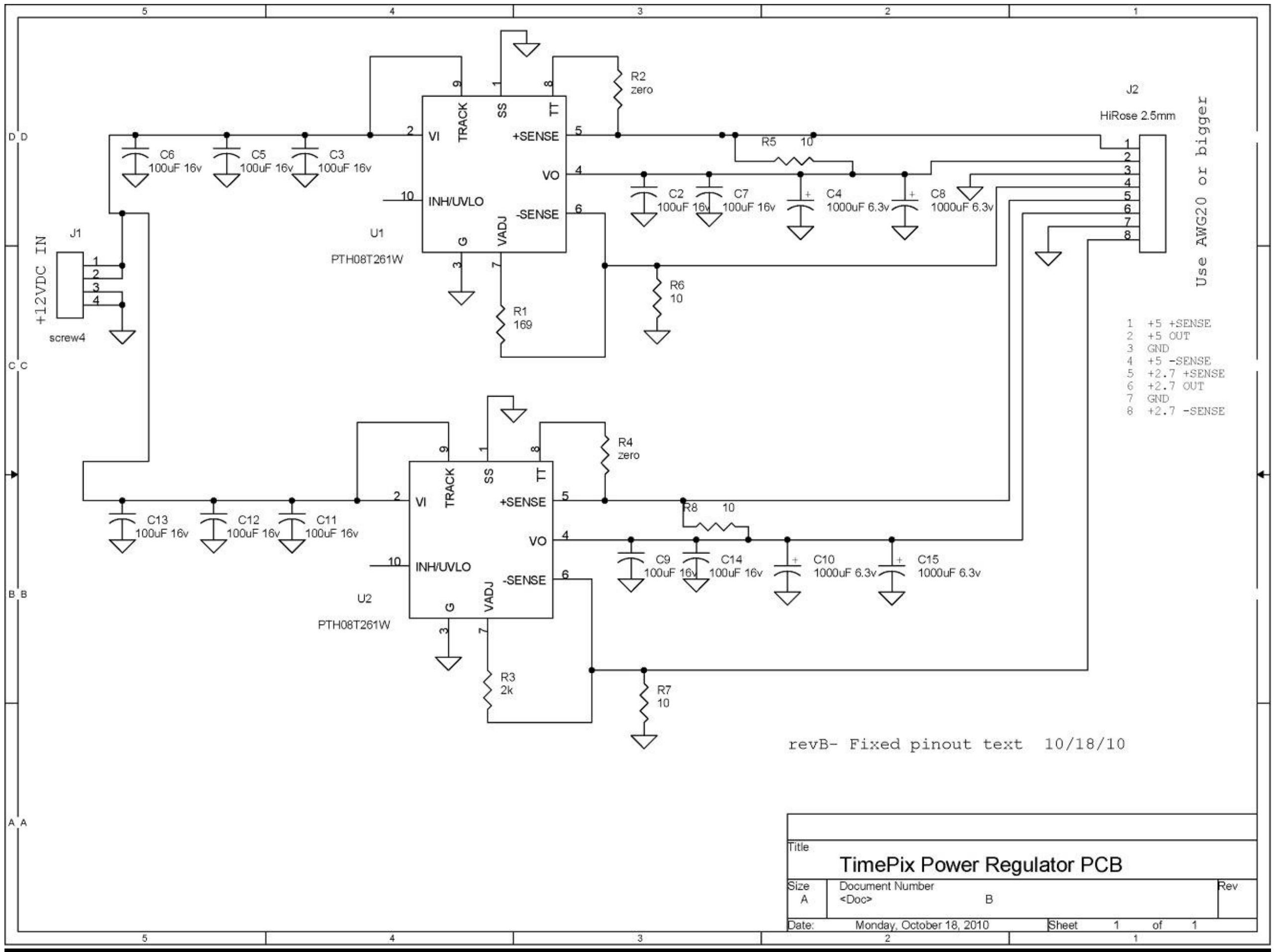
ASIC Board, sht 2 of 2


J1
LTH-050-01-G-D-A-TR
U1A
U1B
J4
Mode header
U1C
U1D
U1E
J8
J3
Programming header
U1F
U1G
J2
QSH-040
revD- Updated per layout. 4/1/10
revE- Updated J1 pinout per TE board layout 6/7/10
revF- added Cx to U5 10/18/10
TIMEPIX/MEDIPIX Interface Board FPGA

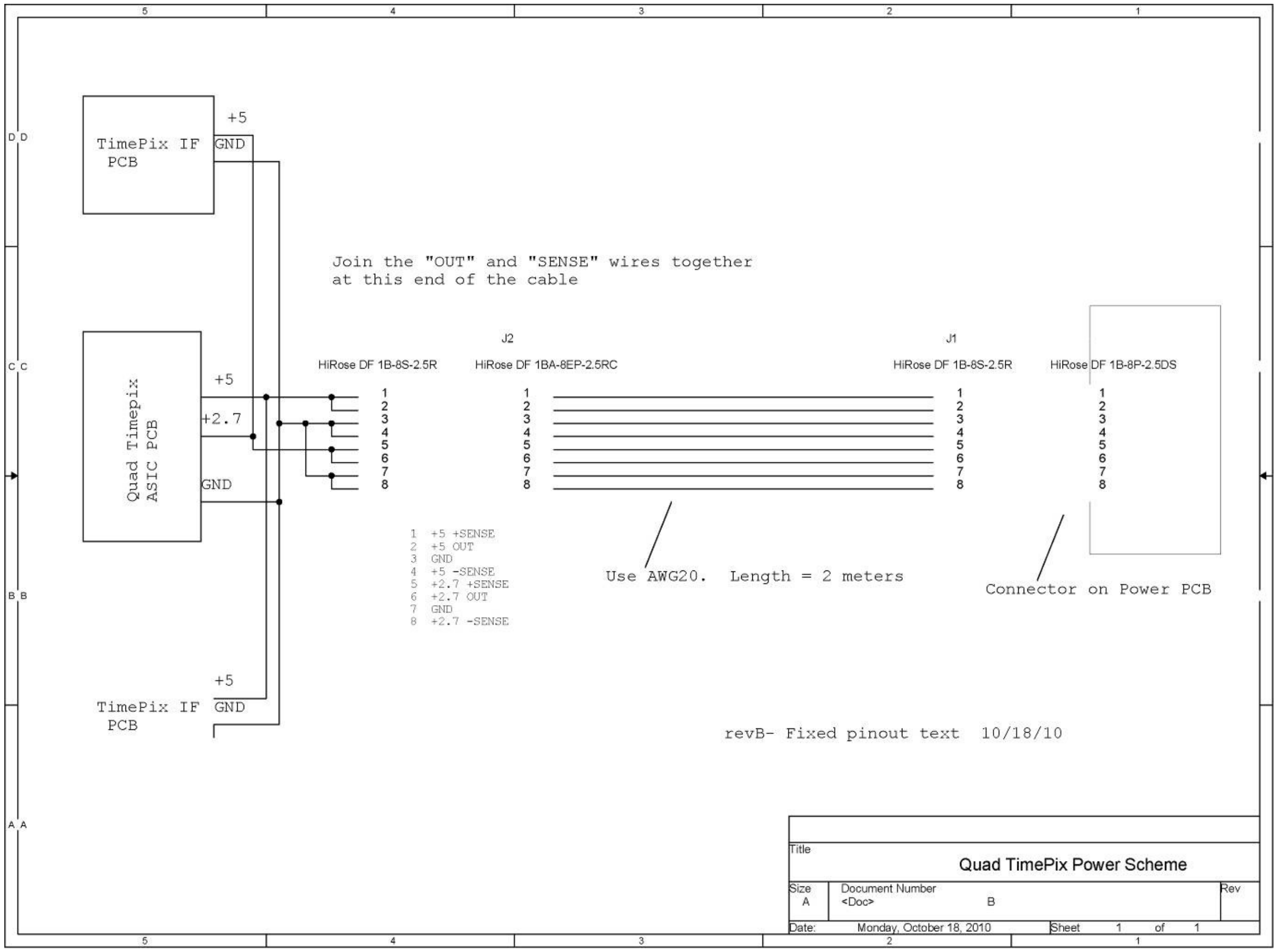
TimePix IF Board, sht 1 of 2



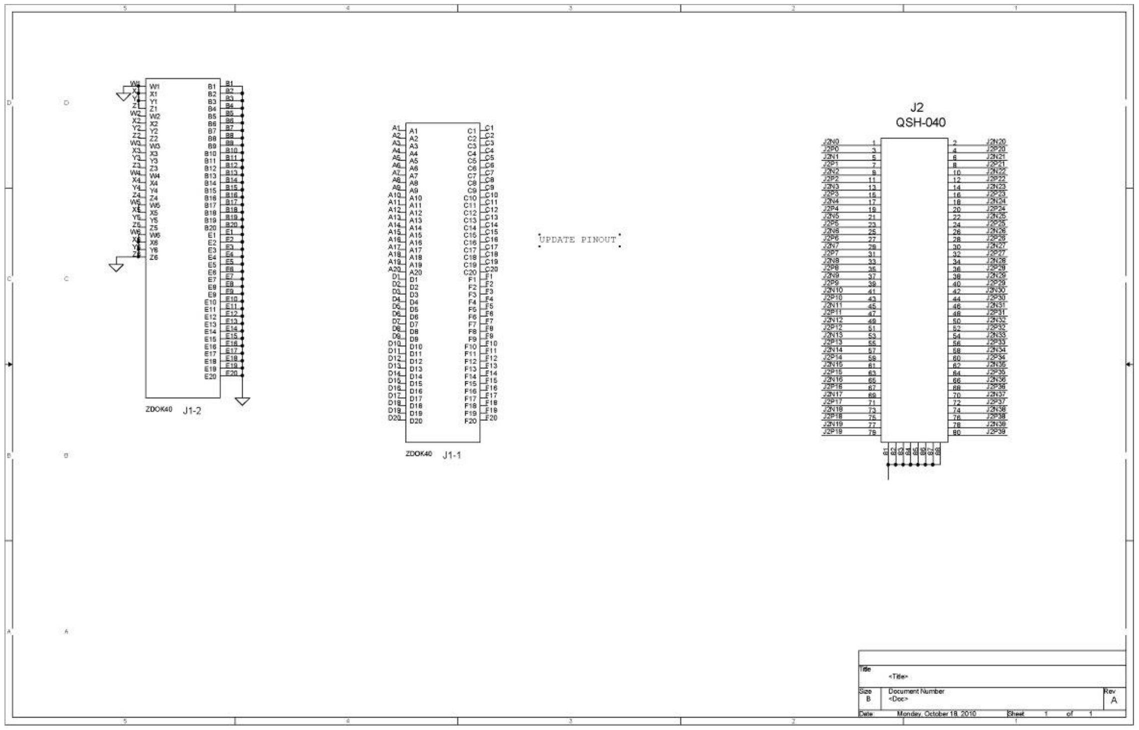
TimePix IF Board, sht 2 of 2



TimePix Voltage Regulator PCB



TimePix Power Cable



ZDOK Adapter schematic

Parts Lists

Quad TimePix Detector (QTD) Revised: 4/7/10					
	Qty	Comp Designator	Value	Pkg	Order#
	4	C1,C9,C34,C37	470pF	0603	478-3720-1
	7	C2,C25,C26,C38,C47,C54,C71	100uF tantalum	7343-20	495-1593-1
	10	C3,C22,C24,C28,C35,C39,C46,C49,C63,C69	10uF ceramic	0805	399-3138-1
	6	C4,C23,C29,C36,C40,C48	0.01	0603	
	36	C5,C6,C7,C8,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20, C21,C30,C31,C32,C33,C41,C42,C43,C44,C50,C51,C52,C53,C55, C56,C57,C58,C59,C61,C65,C67	1nF	0402	445-4924-1
	8	C27,C45,C60,C62,C64,C66,C68,C70	0.1	0603	
	2	J1,J2	LTH-050-01-G-D-A-TR	like IF board	
	1	J3	four SM pads		
	2	J4,J6	two SM pads		
	2	R1,R85	4.53k	0603	P4.53KHCT
	4	R2,R72,R82,R86	4.99k	0603	P4.99KHCT
	6	R4,R6,R73,R79,R87,R105	zero	0603	P0.0GCT
	128	R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20,R21 ,R22,R23,R24,R25,R26,R27,R28,R29,R30,R31,R32,R33,R34,R35 ,R36,R37,R38,R39,R40,R41,R42,R43,R44,R45,R46,R47,R48,R49 ,R50,R51,R52,R53,R54,R55,R56,R57,R58,R59,R60,R61,R62,R63 ,R64,R65,R66,R67,R68,R69,R70,R8	49.9	0603	RMCF1/1649.9FR
	2	R71,R81	6.04k	0603	P6.04KHCT
	4	R74,R75,R84,R122	SAT	0603	
	4	R155,R156,R157,R158	100	0603	P100HCT
	4	U1,U2,U3,U4	TimePixAsic	chip on board	
	2	U5,U7	LM73	SOT23	LM73CIMK-1CT
	2	U6,U8	MAX4634	MSOP10	MAX4634EUB+
	4	U9,U10,U11,U12	LP3878	LLP	LP3878MR-ADJ
	4	U13,U14,U16,U17	MAX9174	MSOP10	MAX9174EUB+
	2	U15,U18	LD1117S33CTR	SOT223	LD1117S33CTR

ASIC Board

TimePix Interface Board BOM				
RR April 8, 2010				
SEE ASSEMBLY NOTE ATTACHED				
	Qty	RefDes	Value	Pkg	
1	3	C1,C3,C6	10uF/6.3V	0805	511-1488-1
2	3	C2,C4,C5	0.1	0603	
3	24	C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32	0.01	0603	
4	1	D1,D2,D3	LED	0805	160-1176-1
6	1	J1	LTH-050-01-G-D-A-TR	see datasheet	
7	1	J2	QSH-040	see datasheet	QSH-040-01-F-D-DP-A
8	1	J3	HEADER 7X2	2mm spacing	H1813
9	2	J4,J5	HEADER 6X2	.1" spacing	part of S2012e-36
10	1	J6	screw4	see DS	ED1516
11	1	J7	HEADER 2X2	.1" spacing	part of S2012e-36
12	1	J8	HEADER 8X2	.1" spacing	part of S2012e-36
13	1	R1	zero	0805	p0.0ACT
14	1	R2	DNP	0806	
15	1	R3	100	0807	P100ACT
16	2	R5,R9	10k	0808	P10.0KCCT
17	2	R6,R7	20k	0809	P20.0KCCT
18	1	R12	121	0810	P121CCT
19	1	R13	121	0811	P121CCT
20	1	R14	91	0812	P91CCT
21	1	U1	XC3S400AN-4FGG400C	FG400	122-1554-ND
22	1	U2	LD1117S12TR	SOT223	497-6974-1
23	1	U3	LTC1660	SSOP16	LTC1660CGN#PBF
24	1	U4	LD1117S33CTR	SOT223	497-1241-1
25	1	U5	LD1117SCTR	SOT223	497-1229-1
26	1	U6	ADC124S021	MSOP10	ADC124S021CIMMCT

Interface Board

ZDOK Adapter BOM

QTY	Desc	Source	PN
1	PCB	Sunstone	TimePix Adapter.123
1	ZDOK	Tyco	6367555-3
1	Samtec	Samtec	QSH-040-01-F-D-DP-A

ZDOK Adapter

TimePix Power Regulator PCB Revised: Thursday, May 20, 2010				
Item	Quantity	Reference	Part	DK Order Number
1	10	C2,C3,C5,C6,C7,C9,C11,C12,C13,C14	100uF 16v	445-3485-1
2	4	C4,C8,C10,C15	1000uF 6.3v	493-3774-1
3	1	J1	screw4	ed1516
4	1	J2	HiRose 2.5mm	H3638
5	1	R1	169	
6	2	R2,R4	zero	
7	1	R3	2k	
8	4	R5,R6,R7,R8	10	
9	2	U1,U2	PTH08T261W	296-21539
		Receptacle for power		H3787
		contacts for above		H3830
		InLine plug (use at detector)		H3689
		contacts for above		H3829

Voltage Regulator Board

Hardware Fabrication History

ASIC Boards
We fabbed 25 PCBs April 2010, and assembled and wirebonded four, I think. These have the voltage regulator layout problem, and we retrofitted two of the boards with piggybacked voltage regulators, which was a not-very-satisfactory fix.

IF Boards
We fabbed 20 of these PCBs and assembled 5. One at least has a problem with shorted pairs on the QSH connector.

ZDOK adapter
Fabbed 12 PCBs, assembled 4. One has a shorted-pair problem.

Voltage regulator PCBs
Fabbed 6 PCBs, assembled one.

Flex PCBs
Purchased 27 of these (Samtec lot charge)

Roach Enclosures
Fabbed two. These needed some mods to make things line up.

IF Board Enclosures
Fabbed 8 of these. Also needed some mods.