Table of Contents

HSIODataFormat2

v4182

Introduction

[This page contains info on "Schema 2" firmware. That is firmware version 0x4000 and upwards. Although it is titled data-format, it also has info on items that have changed while doing this update.]

Communication with the HSIO board is via packets of data transferred as Raw Ethernet. Packets contain one or more **Opcodes**. Each *opcode* consists of an `opcode-id`, a `sequence-number` and `payload-size` field, along with (optional) data payload. In some cases opcodes initiate a dataless operation (Command), in others they write their data to a register (RegisterWrite), a block of registers (RegisterBlockWrite) or control and I2C transaction. When an opcode operation completes, an acknowledge/reply packet is sent to the host, providing data if requested. A second acknowledge (Ack) packet is also sent from the packet handler (the level above opcode) indicating the packet was processed. If an opcode is not processed, a timeout will occur and a packet will be sent back to the host with the opcode as sent (echo).

Ack/Reply packets have the same structure as the received packet. Reply packets contain the sequence number as sent. Multiple opcodes can be loaded into a single network packet to reduce latency.

The HSIO can send data packets at any time, however Network Ack packets have a higher priority. An arbiter takes care of this.

Network Packet Format

Packets are built from 16 bit words. Ethernet restricts the payload to between 64 and 1.5kB in length (i.e. not jumbo). Therefore we must transmit large chunks of data in multiple packets, with the data-format taking care of the ordering.

Minimum packet is 64 bytes (including MAC stuff), so small packets are padded with zeros.

The Ethernet packet has 12 bytes of MAC address (`dest(5:0); source(5:0)`) and a 2 byte type field, which we use a "magic number". The HSIO will send to the source MAC address of the last packet received.

Packet Format	Notes
Magic number	For Ethernet, this uses the Type field
Sequence number	Defined by software, used for reply packets
Packet length	Length in bytes of entire packet (including zero padding in used)
Opcode Count	16bit (rounds header to 16 bytes)
Opcode 1	
...	
Opcode N	
Trailer	16 bit CRC

Magic Numbers	
PC to HSIO	0x8765
HSIO to PC	0x8765

Opcode Format	Notes
OpcodeID	16-bit, described below
Sequence Number	16-bit, increment for each comblock. Normally a reply has matching sequence number (how does this work if extra packets sent in between, maybe each side keeps their own count)
Payload Size	16-bit, length in bytes, 0 means no data
Payload Data	See below
[CRC ?]	Dropped, we have one at the packet level (and Ethernet)

Opcode List

OpcodeID	Name	ShortName	Payload
0x0003	Echo	ECHO	Any data
0x0010	RegisterWrite	REGWRITE	Reg No, Data
0x0014	RegisterBlockWrite	REGBLOCK_WR	32 word (64 byte) block of reg data in sequence
0x0015	RegisterBlockRead	REGBLOCK_RD	none
0x0019	StatusBlockRead	STATREAD	none
0x0030	Command	COMMAND	(16b mask)(optional 16b mask)
0x0050	StreamConfigWrite	STRM_CONF_WR	`(bit mask)(stream-id)(data)[(stream-id)(data)`
0x0051	StreamRequestStatus	STRM_REQ_STATS	`(stream mask 0)...(stream mask 6)`
0x0052	BStreamConfigWrite	BSTRM_CONF_WR	`(stream mask 0)...(stream mask 6)(bit mask)(data)`
0x005c	StreamCommand	STRM_COMMAND	`(stream-id)(command mask)[(stream-id)(co mask)...]`
0x005e	BStreamCommand	BSTRM_COMMAND	`(stream mask 0)...(stream mask 6)(comman mask)`
0x0070	COMPatternLoad	COM_PATTERN	Bit stream to be placed in pattern RAM
0x0074	SendRawSigs	RAWSIG	Bit stream

0x0080	TwoWireInterface	TWOWIRE	`(control word - unused)(cmd 15:8, data 7:0)(cmd, data 1)...`
0x00F0	Reset OCB sub-system	RESET_OCB	Ignored
0x0101	SendRawCOM	RAWCOM	Bit stream

Unsolicited opcodes

Opcode	Name	Payload	Description
0xBnnn	Unrecognised Opcode	Echo of RX Opcode, with 0xB in first nibble of opcode-id	If an opcode is not handled by hsio within 1 second, it is grabbed by the echo ocb and returned
0xD0mm	Data	Data - first word contains `(hsio-link-id)(fragment number)=`	`mm is the =mode` detailed below
0xF0mm	Deserialiser Error	2 words only - Data header (as above) and Error Code (see below)	Sent Data is dropped due to timeout on trailer, or full fifo (or both!). See error codes below.
0xFFFF	Fail	Empty	Generic failure

mm - Mode Word		
Value	**Name**	**Description**
0x00	Normal Data	Serial data stream as sent by ASICs
0x01	Alex Histogram Data	
0x02	Sim data in header detect mode	
0x04	Capture Data (no header detect)	
0x06	Sim data in capture mode	
0x08	ABC130 Data	Deserialised packets (with some reformating) from ABC130 ASICs
0x11	Tom Histogram Data (addition of error counters)	

Dealing with Readout Buffer Errors

The stream FIFOs are fixed length, and if they are not read fast enough, they can overflow. Each FIFO can support a complete network packet of data (1.5kB). If the FIFO is nearly full and filling, the data will be *truncated*. Truncated packets can be identified by a lack of trailer. The system will then wait for the trailer of the event before resuming header detection. The system will wait indefinitely for a trailer, but a single Error Packet will be sent after some time. If an event is longer than 1 packet it will be split over 2, as a *fragmented* event. An event can only be a max. 2 packets long - any longer and he event will be truncated.

Error Codes		
Value	**Name**	**Description**
0x01	EMPTY_EVENT	FIFOs too full to send proper event (not currently used)
0x02	TRUNCEV_TRAILER_TO	Truncated Event trailer timeout
0x03	FRAGEV_TRAILER_TO	
0x04	EMPTYEV_TRAILER_TO	
0x05	LEN_FIFO_FULL	

Some Opcode Details

STRM_CONF_WR

StreamConfigWrite (0x50)

Word	Name
0	Bit Mask
1	Stream ID
2	Data
Optional:	
3	*Stream ID 1*
4	*Data 1*
. . .	
N	*Stream ID X*
M	*Data Y*

Reply	
Word	**Name**
0	0xacac

StreamConfig Word

Bit Field	Description
15:9	undefined
8	*reserved for parallel histo enable*
7	Busy enable - FIFO level (half-full)
6	Busy enable - Delta trigger/headers
5:4	deser mode (00b: header/trail det, 01b: capture, 10b: undefined, 11b: Disabled=Histo)
3:1	data source (see below)
0	stream enable

Data Source

Value	Description
0	Input pin
1	*reserved for Input pin +16*
2	*reserved for Input pin +32*
3	*reserved*
4	Data generator (counter) 0
5	Data generator (counter) 1
6	Simulated event generator 0
7	Simulated event generator 1

STRM_REQ_STATS

StreamReqStatus (0x51)

Word	Name	Description
0	Stream Mask 0	Streams 0-15 (top)

1	Stream Mask 1	Streams 16-31 (top)
2	Stream Mask 2	Streams 32-47 (top)
3	Stream Mask 3	Streams 48-63
4	Stream Mask 4	Streams 64-79 (bot)
5	Stream Mask 5	Streams 80-95 (bot)
6	Stream Mask 6	Streams 96-111 (bot)
7	Stream Mask 7	Streams 112-127
8	Stream Mask 8	Streams 128-143 (IDC)

StreamStatus Words

Bit Field	Description
Word 0	
15:0	StreamConfig Word
Word 1	
15:12	dropped headers (aka no packet send in response to header) count
11:10	lengths fifo fullness, in quarters
9:8	data fifo fullness, in quarters
7	busy_fifo
6	busy_delta
5:0	header-trigger delta

STRM_COMMAND

StreamCommand (0x5c)

Word	Name
0	Stream ID
1	Command Mask
Optional:	
2	*Stream ID 1*
3	*Command Mask 1*
. . .	
N	*Stream ID X*
M	*Command Mask Y*

Reply	
Word	**Name**
0	0xacac

StreamCommand Word

Bit Field	Description
15	stream reset
14:1	undefined
0	histo memory readout

BSTRM_CONF_WR/COMMAND

Broadcast StreamConfig/Command (0x52/5e)

Word	Name	Description
0	Stream Mask 0	Streams 0-15 (top)
1	Stream Mask 1	Streams 16-31 (top)
2	Stream Mask 2	Streams 32-47 (top)
3	Stream Mask 3	Streams 48-63
4	Stream Mask 4	Streams 64-79 (bot)
5	Stream Mask 5	Streams 80-95 (bot)
6	Stream Mask 6	Streams 96-111 (bot)
7	Stream Mask 7	Streams 112-127
8	Stream Mask 8	Streams 128-143 (IDC)
9	Data/Mask	The StreamConfigWrite (0x0052) data or StreamCommand (0x005e) mask

TWOWIRE

Send/Recieve data on various 2 wire busses. (0x80).

Initiating packet(let) contains a control word as the first word, this is echoed as the first word of the reply. Each subsequent word will have a corrosponding response word in the reply packet (even if no response was requested). Only single bytes can be written at a time, but 16b words can be read. When writing 2 bytes, 2 operations will be required (examples below). To pre/append protocol specific start or stop signalling (where they exist) use bits 2,3 of the command field.

If we get a timeout waiting for e.g. an Ack from a Slave device, then the operation is aborted and all remaining reply fields filled with 0xF00B. This presumes we will never read 16b data(!).

More than one packetlet (i.e. a set of commands starting with a control word) can be sent in a single opcoed by separating them with 0xFFFF (well 111xxxxxxxxxxxxx actually).

Word	Name	Description
0	Control	Select output channel and clock freq see below)
1:n	Cmd/Data	8b command + 8b data - see below

Control Word

Bits	Name	Description
7:4	Select clock	0 - 100kHz
		1 - 10kHz
		2 - 1kHz
3:0	Output Channel	0 - EOS AD7998 Top 0
		1 - EOS AD7998 Top 1
		2 - EOS AD7998 Top 2
		4 - EOS AD7998 Bottom 0
		5 - EOS AD7998 Bottom 1
		6 - EOS AD7998 Bottom 2
		12 - Aux 0 (P2 IDC) 1:sck 2:sda
		13 - Aux 1 (P2 IDC) 3:sck 4:sda
		14 - Aux 2 (P2 IDC) 5:sck 6:sda

		15 - Aux 3 (P2 IDC) 7:sck 8:sda

Command/Data Word

The command portion of the word is build thus:

7:5	Protocol	0 - I2C
		1 - SHT
		2 - SPi - (not yet)
		7 - Packetlet separator
5	reserved	may be *Enable Special Functions* (e.g. send CONVST) with bits 4:0
4	Append Stop	
3	Prepend Start	
2	Send Byte	Send a byte on the serial bus (always precedes getting bytes)
1:0	Get N Bytes	Get 1 or 2 bytes from the serial bus, 3 is reserved

BUT you may be better off just using the codes given below and stop trying to be too smart ☺

I2C	
0x00nn	Send byte
0x01xx	Get byte
0x02xx	Get 2 bytes
0x0caa	Send Start + Address-chip byte
0x14nn	Send byte + Stop
0x11xx	Get byte + Stop
0x12xx	Get word (2 bytes) + Stop
SHT71	
0x4c06	Send command for write (issue before 0x44nn below)
0x44nn	Send byte (issue after 0x4406 above)
0x4d07	Get/Read Stats (returns 1 byte in LSB)
0x4e03	Get/Read Temp (returns 2 bytes)
0x4e05	Get/Read Humi (returns 2 bytes)

Examples

Write data to AD7998 (I2C)

```
   0: 0x000n - select which I2C destination
   1: 0x0c44 - Address chip for write
   2: 0x04nn - Write to Address pointer register
   3: 0x04nn - Write Data MSB
   4: 0x14nn - Write Data LSB + Stop
```

Read word from AD7998 (I2C)

```
   0: 0x000n - select which I2C destination
   1: 0x0e45 - Address chip and read word
```

Now the biggy - setup and read all temperatures (all values in hex)

```
0000                   # Send on Top ADC 0
0c44 0402 040f 14f8    # Write to config reg, all chans enabled
0c44 0470              # Set addr-pointer = 0x70 (cycle all ch, result reg) (no stop)
0c45 0200 0200 0200 0200 0200 0200 0200 1200    # Read 8 words (stop on last one)
```

Read SHT71 data

```
 0: 0x000c  # send on P2 port (aka Aux)
 1: 0x4e03  # get/read temp
 2: 0x4e05  # get/read humi
 3: 0x4d07  # get/read status
```

Write to SHT71 "status" register:

```
 0: 0x000c  # send on P2 port (aka Aux)
 1: 0x4c06  # send command for status write
 2: 0x44nn  # send new status byte
```

REGWRITE/BLOCK_WR/BLOCK_RD

Reg/Block Read/Write (0x10/14/15)

No.	Addr	Short Name	Name	Description
0	0x00	IN_ENA	Inputs Enables	see Signals Registers
1	0x01	OUT_ENA	Outputs Enables	see Signals Registers
2	0x02	INT_ENA	Internal Enables	see Signals Registers
3	0x03	IN_INV	Inputs Invert	
4	0x04			
5	0x05			
6	0x06	SPYSIG_CTL	spy signals stream control	configure ABC signals spy block register bits
7	0x07	LEN0	Length field 0	gendata0 len, capt mode len (capt. len is in 16b words & rounded down to nearest 0x10)
8	0x08	LEN1	Length field 1	gendata1 len
9	0x09	IDELAY	idelay control register	register bits
10	0x0a	SG0_CONF_LO	Simulation data generator 0 config word (15:0)	see Simulation data generator info
11	0x0b	SG0_CONF_HI	Simulation data generator 0 config word (31:16)	see Simulation data generator info
12	0x0c	SG1_CONF_LO	Simulation data generator 1 config word (15:0)	see Simulation data generator info
13	0x0d	SG1_CONF_HI	Simulation data generator 1 config word (31:0)	see Simulation data generator info
14	0x0e	SQ_RNDSEEDS	Random sees of SG0/1 `[seed SG1 8b][seed SG0 8b]`	see Simulation data generator info
15	0x0f	TWIN_DELAY	Trigger window delay	
16	0x10	COM_ENA	ABC signals enables and control	register bits
17	0x11	BUSY_DELTA	Busy-delta On (13:8), Off (5:0)	Sets the level of delta trigger-header at which busy is (de)asserted. Global.
18	0x12			
19	0x13			
20	0x14	DISP_SEL	display reg select	Select data source for output on dot-matrix display
21	0x15	LEMO_STRM	lemo 1,2,3 debug stream select	Decides which stream is on the J1-3 LEMOs. 1 = strm even, 2 strm odd, 3 muxed link data (see details)
22	0x16			
23	0x17	CONTROL	hsio control register	register bits

24	0x18	TB_TRIGS	trig_burst(tb) trigs	Number of triggers in a burst
25	0x19	TB_BURSTS	tb bursts	Number of bursts
26	0x1a	TB_PMIN	tb rnd period min	Randomiser trigger period minimum - in 400ns units
27	0x1b	TB_PMAX	tb rnd period max	Randomiser trigger period maximum - in 400ns units
28	0x1c	TB_PDEAD	tb intra-burst deadtime	Time between bursts in 400ns units
29	0x1d	TDELAY	Trigger Delay	Delay in clk40 steps (~25ns)
30	0x1e	BCO_DC	BCO Duty Cycle	Enabled by 23.10
31	0x1f	DL0_DELAY	L0 trigger delay	When using RAWSIG an L0 can be sent after a delay. Enabled by 23.15

SignalsRegisters

Registers 0,1,2: In, Out and Internal Enables

Each bit corrosponds to an enable for a signal source (generator/input) or output.

Bit	Description
0	Trigger
1	BCR
2	ECR

The external trigger input is in LEMO J4.

SPYSIG_CTL

Register 6: Spy Signal Stream Control

Bit Field	Description
7:4	Spy channel select, see below
0	SpySig Enable

Spy Channels	
0	COM
1	L1R
11:2	
12	ABC130 COM
13	ABC130 L0
14	ABC130 L1
15	ABC130 R3

IDELAY

Register 9: IDELAY Control

Bit Field	Description
13:8	Hybrid link number (0-67)
5:0	Delay value, in ~78ps steps

SimulationDataGen

Register 10-13:

Bits	Short	Description
0	Activation	0 - route input to output, 1 - ignore input, generate random events)
1	Send only clock/2	0 - send event, 1 - switch to clk/2
2	Send config readback	0 - send events, 1 - send config
3-7	custom header	if not 0, uses bits as header
8-9	hit prob	Send hit vs empty 0 - hit, 1 - 50/50, 2 - 0.4% hit, 3 - empty
10-11	err prob	Send error flag 0 - normal, 1 - error 0.4%, 2 - error 50%, 3 - error 100 %
12-18	chipnum	How many chips
19-25	hitnum	Cluster has n hits
26-27	hitmap	0 - 011, 1 - 01X, 2 - X1X, 3 - XXX
28-29	cluster size	Hits in cluster
30-31	Link 0 error probability	Bit flip 0 - 0, 1 - 1 in 512, 2 - 1 in 4, 3 - 1 in 512k

Register 14 is the seed, but it's only loaded on reset?

COM_ENA

Register 16: ABCN Signals Enable and Control

Bit	Description
15	com shift 180
14	com shift 90
13	dclk enable
12	bco enable
11	invert reset
10	~~swap bco and dclk outputs~~
9	com into J38-IDC reset en
8	com into J37-IDC reset en
7	dclk invert
6	bco invert
5	com into J38-IDC L1 enable
4	com into J37-IDC L1 enable
3	com into stave L1R enable
2	stave com enable
1	J38-IDC com enable
0	J37-IDC com enable

Register settings to enable clock and com through IDCs connectors:

- 0x1003 (bits 0, 1 and 12 set)

LEMO_STRM

Register 21 (0x15): Debug-output Streams Select

Value in the register points to the **Hybrid LINK** number that is demuxed to become 2 streams i.e. double the link number for even stream number, add an extra 1 for odd stream. The front-panel Lemo outputs contail all 3 of these signals:

Lemo (Jn)	Description

1	Stream (even)
2	Stream+1 (odd)
3	Link containing both streams

CONTROL

Register 23: Control

Bit	Description
15	
14	DCLK40 mode. Samples data at half rate, sends 40MHz DCLK to IDCs
13	com into *eos noise* enable *
12	Trigger destination 1=L1R, 0=COM
11	Use trigger (not com_start) as capture "go"
10	Enable BCO Duty-Cycle tuning
9	~~Re-arrange ST outputs for use with EOS porch~~
8	Enable pattern playback on trigger
7	enable HSIO on-board debug outputs e.g. P4, P5 and Lemo
6	Enable ABC130 ABC format (else HCC format
5	Enable ABC130 data generators (disable=reset)
4	histo test data en (data=addr)
3	
2	
1	
0	Soft-Busy

* Was at bit 6 until v415d

Register settings for 40/80 MHz dclock through IDC and debug outputs enabled:

- *80 MHz data clock*: 0x00A0 (bits 5 and 7 set)
- *40 MHz data clock*: 0x40A0 (bits 14, 5 and 7 set)

STATREAD

Status Word List (0x19)

No.	Addr	Short Name	Bit Range	Default Val	Description
0	0x00	HW_ID	15:0	0x0C02	Hardware ID
1	0x01	SANITY	15:0	0xA510	Sanity check = HSIO
2	0x02	VERSION	15:0		
3	0x03	MODULES_EN	15:0		Total number of modules in the build
4	0x04	TB_TCOUNT	15:0		Burster trigger count down (of current burst)
5	0x05	TB_BCOUNT	15:0		Burster burst number count down
6	0x06	TB_FLAGS	15:0		Burster Flags see TB_FLAGS
7	0x07	BCID_L1A	11:0	0x0000	BCID Value at last L1A, *reset with BCR (see Command Opcode)*
8	0x08	L1ID _LO	15:0	0x0000	L1A/Trigger Counter Lo, *reset with ECR (see Command Opcode)*
9	0x09	L1ID _HI	23:16	0x0000	L1A/Trigger Counter Hi, *reset with ECR (see Command Opcode)*

10	0x0a	SFP0 net interface status word	15:0		
11	0x0b	SFP0 net interface status word	31:16		
12	0x0c	SFP1 net interface status word	15:0		
13	0x0d	SFP1 net interface status word	31:16		
14	0x0e	CU net interface status word	15:0		
15	0x0f				
16	0x10	TMODULES_EN	11:0		top modules included (bitmap, 1 per module)
17	0x11	THISTOS_EN	11:0		top module histos included
18	0x12	BMODULES_EN	11:0		bottom modules included in build
19	0x13	BHISTOS_EN	11:0		bottom module histos included in build
20	0x14	PPHISTMOD_EN	1:0/1:0		IDC modules/histos included in build `[histos 8b][modules 8b]`
21	0x15				
22	0x16	TIMESTAMP_LO	31:16		Timestamp in seconds since the epoc
23	0x17	TIMESTAMP_HI	15:0		
24	0x18				
25	0x19				
26	0x1a				
27	0x1b				
28	0x1c				
29	0x1d				
30	0x1e				
31	0x1f				

TB_FLAGS

Status 6: Trigger Burster (Sequencer) Flags

Bit	Description
15:4	Unused
3	
2	SEQ_FINISHED
1	SEQ_RUNNING
0	SEQ_READY

COMMAND

Command (0x30)

Send commands (pulses) as specified. Comprised 2 words, the second one is optional.

		WORD 0
Bit	Short Name	Description
15	TWMEM_GO	TWOWIRE OCB memory playback
14	RCPATT_GO	RAWCOM pattern bank playback
13	SS_RST	Seq/Sink Reset

12	SS_GO	Seq/Sink Go
11		
10		
9		
8	TB_START	Start trigger burst
7		
6	L1ID_RST	HSIO L1ID counter reset
5	BCID_RST	HSIO BCID counter reset
4		
3		
2	ECR	Send ECR on COM (`1010100`) (will also reset HSIO BCID counter)
1	BCR	Send BCR on COM (`1010010`) (will also reset HSIO BCID counter)
0	L1 Trigger	Send Trigger on COM (`110`) or L1R (`10`). See reg 23, bit 10
		WORD 1 (Resets)
15:7	Unused	
6	RST_NETRX	Reset Network RX sub-system
5	RST_NETTX	Reset Network TX sub-system
4	RST_DISP	Reset Display
3	RST_FEO	Reset FE Output sub-systems
2	~~RST_OCB~~	~~Reset all Opcode blocks~~ See RESET_OCB opcode
1	RST_TRIG	Reset Trigger sub-system
0	RST_RO	Reset Readout sub-system

More on the v4000+ schema

The update to a new schema was required to allow the number of streams to grow without needing a large number of bits reserved in a more traditional register space map. In this version each stream is communicated-with independently. New opcodes allow **StreamConfig** and **StreamCommand** words to be sent to individual streams (identified by ID) The **StreamConfig** word holds static values (like *enable*), and the **StreamCommand** word is used for sending pulses (like *reset*). (In the future maskable broadcasts of these operations will also be possible.)

A third new opcode is used to request any number of streams to send a status packet each: **StreamRequestStatus**. This is *request* as multiple streams can repond, and each sends a packet with a sequence number NOT matching the requester. Currently this reply packet only contains the StreamConfig register value, but will grow in the future to contain status words (such as FIFO levels, event counters etc.)

Idelay control is unchanged and as this isn't actually a stream operation - (idelay operates on stream pairs, aka links), Operation is exactly as before (via reg 9). (value readback will soon be implemented.)

HSIODebugPins

Some non data-format info about schema 2:

The firmware in the SVN trunk (https://svnweb.cern.ch/cern/wsvn/atlasupstrip/firmware/hsio/trunk). All the `hsio_c01/c02` etc have been tidied - we now build `hsio`:

- `hsio/src/hsio_top_struct.vhd` is the top level
- =hsio/src/pkg_hsio_globals.vhd = Set up enabled streams
- `ise/hsio` is the ISE project directory (load the .xise file)

The display shows the version number for aprox 8s before switching to another register.

The left most digit of the display shows SFP link status:

- top left: link-up left SFP*
- top right: link-up right SFP*
- bottom left: RX packet
- bottom right: TX packet

* When using an RJ45 SFP, a link is established even when no cable is plugged in. This is because this kind of SFP is actually a bridge between copper and serial and a link is setup between the SFP and FPGA.

Stream Allocations

No.	Name
0-47	Stave Top
64-111	Stave Bottom
128-131	IDC J37
132-135	IDC J38

Not all streams will be included in all builds, check status words for number of top, bottom and IDC streams in the build and if they have histogrammers.

Internal Busy System

Each stream is able to generate a busy. As the ABCN chip has no mechanism (or buffer) to allow flow-control of incoming data, the only way to reduce the flow is to reduce the trigger rate. Although the new ABC130 chip will (arguably) have flow-control, it's far better to assume we need to handle anything the front-ends throw at us. This means we need to generate a busy long before we are actually full as there could still be plenty of data progressing through the system.

Each stream has a "delta triggers" counter - incremented by a trigger, decrimented when a header is seen in the returning data. When this value exceeds 15, `busy_delta` is asserted. The value of 15 is hardwired to save FPGA resources, but could be programmable if needed.

Each stream also monitors it's FIFO level, and when it is half-fill, `busy_fifo` is asserted.

To enable a busy from a stream, set the `busy_delta` and/or `busy_fifo` bits in the StreamConfig Word.

When busy is asserted external triggers will be vetoed and the burst system will pause.

Major updates:
-- MattWarren - 12-Jul-2011

Responsible: warren_40hep_2eucl_2eac_2euk
Last reviewed by: **Never reviewed**

This topic: Atlas > HSIODataFormat2
Topic revision: r81 - 12-Mar-2013 - MattWarren

Copyright &© by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback