

DSAT Firmware / Software Specification

E. Hazen – Rev 4 22 July 2005

Should correspond to firmware Rev 09

This document describes the firmware and low-level software interface to the DSAT (DFEA Stand-Alone Tester) module. *N.B.* this document will evolve continuously!

RELEASE NOTES

Rev 09 firmware:

Many changes from Mr Wu

Rev 07 firmware:

Added bits 4-7 in SCL CSR for ISO_IN and ISO_OUT control (DC only)

Rev 06 firmware:

Timing changes on backplane per Wu's request

Rev 05 firmware:

All LVDS transmit/receivers active

SCL works as advertised except that NRZC is only active when running a sequence from the SCL ram

Rev 02 firmware:

Programming interfaces for LVDS transmitters 0-3 only are active.

Transmitters (0,1) send identical data, program as Tx0

Transmitters (2,3) send identical data, program as Tx1 etc

(ultimately this will be fixed using DDR outputs for LVDS)

Sending FX (first crossing) in first frame of first word for SCL ram is unreliable.

Best to leave first frame all 0's and start real data in 2nd frame.

1. Introduction

The DSAT is intended to serve three purposes:

1. A debugging aid for the initial DFEA board checkout
2. A facility for validation of the DFEA firmware using test vectors
3. A production test stand for DFEA modules

As such the firmware and software must support both simple, word-by-word debugging

and also efficient transfers of significant volumes of data to/from the host PC.

The DSAT consists of several logical components, described in detail below. The **EPP interface** provides access to the DSAT from a host PC running Linux. The **R/W bus** provides random read/write access to registers on the DFEA board. The **LVDS transmitters** send simulated drift tube hits to the DFEA from a small memory buffer on the DSAT. The **LVDS** and **SLDB** receivers read data output by the DFEA and store it in a memory on the DFEA. Finally, the **SCL/Timing** block provides simulated accelerator timing signals to control overall operation.

2. Parallel Port Interface

The DSAT connects to a host PC via a standard parallel (printer) port in EPP mode. This provides bidirectional communication with both data and address channels available. The EPP port provides an 8-bit data path, while the registers on the DSAT are typically 32 bits wide. The following low-level protocol is used:

Function	No	7	6	5	4	3	2	1	0
ADDRESS	0	A7	A6	A5	a4	a3	a2	a1	a0
DATA0	1	d7	d6	d5	d4	d3	d2	d1	d0
DATA1	2	d15	d14	d13	d12	d11	d10	d9	d8
DATA2	3	d23	d22	d21	d20	d19	d18	d17	d16
DATA3	4	d31	d30	d29	d28	d27	d26	d25	d24

For each transfer five cycles on the EPP are used. First comes an address byte, which specifies a register address on the DSAT. Following this one or more groups of four data bytes may be transferred in either direction (read or write). An address read operation may occur at any time and will return the last address byte written.

Fewer than 4 data bytes may be read or written if only the lower byte or bytes need to be accessed.

The following functions are provided to access to the DSAT interface at the lowest level:

```
int dsat_open( char *devname, int mode);
int dsat_close( int fd);
int dsat_write_word( int fd, int address, unsigned *data);
int dsat_read_word( int fd, int address, unsigned *data);
int dsat_write_memory( int fd, int address, unsigned *data,
    unsigned count);
int dsat_read_memory( int fd, int address, unsigned *data,
    unsigned count);
```

```
int dsat_set_bit( int fd, int address,
                 int bit_no, int value);
int dsat_read_bit( int fd, int address, int bit_no);
```

dsat_open() must be called before any other operation, and returns a file descriptor to be used on subsequent calls. **dsat_read** and **dsat_write** attempt to read/write a single 32-bit quantity to the specified address. A zero value is returned on success. Any other value indicates a (currently unspecified) error. The memory functions transfer the specified count of 32-bit words to or from a memory buffer as described below. The bit functions operate on single bits. The set_bit function reads the register value, sets or clears the bit using bitwise and/or operations, and writes a new value to the register.

Many DSAT features involve memory, which is implemented using a standard 16k x 32 block in the Xilinx firmware. A standardized interface is provided to simplify software. Each memory buffer is addressed using the following pair of registers:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0	Not Used													Address																		
+1	Data Byte 3				Data Byte 2				Data Byte 1				Data Byte 0																			

Offset 0 contains a 9-bit memory address register, which may be read or written to specify the starting address for any subsequent read/write operations. Data read or written to offset 1 will be transferred to or from the corresponding RAM location. The address register auto-increments after each data read/write operation. The upper 22 bits of the address register are not used.

Other DSAT functions are controlled by individual 32-bit registers. It is suggested that the software which controls the DSAT be built on two classes of low-level operations: memory operations which read/write a specified number of words from a starting address, and control register operations, which modify a specific set of bits in an individual 32-bit register.

3. R/W Bus

The R/W bus is a simple synchronous bus with a 16-bit address and 16-bit data transfer. Address is by slot number in the real DFEA crate.

The upper six bits of the address specifies the slot number and the lower ten bits specify the register or memory address on the DFE board:



2004-04-30a

Thus each slot in the backplane may have up to 1024 registers, although in reality only a few registers be used. Broadcast writes are supported by setting all of the ADDR[15..10] bits. (DFEA boards should not respond with DTACK when broadcast write operations occur.)

On the DSAT, there is only one slot. Various slot numbers may be simulated by writing to the a register which simulates the hard-wired slot ID on the backplane. To access the slot register, perform a normal backplane write with address bit [15..10] set to "11110". The slot number register is initialized to "00000" at power-up.

The R/W bus is controlled by the following registers:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
+0	Unused																Address																	
+1	Data [2]																Data																	
+2	CSR (Control/Status Register)																																	
+3	Unused																																	

The address register contains the slot number and DFEA register number as described above. The data register causes a 16-bit word to be read or written to the backplane when accessed.

Notes:

[1] in the current implementation (firmware rev 3 and below) a backplane read does not occur when the data register is read. One must first write a '1' to bit 1 of the CSR to trigger the read, then read the data register.

[2] for writes to the FPGA configuration registers on the DFEA2 (addresses 4,5,6,7) the upper two bytes of data are used in a second write cycle. This increases the effective bandwidth of the interface. In other cases only the low 16 bits are used.

The CSR contains the following bits:

<i>Bit</i>	<i>Name</i>	<i>R/W</i>	<i>Description</i>
0	bus_error	R	'1' if there was no DTACK on the last cycle
1	bus_read	W	'1' to trigger backplane read to data register

The following functions are provided for access to the R/W bus:

```
int dfe_read( int fd, int slot_number, int register,
             int number_of_words,
             unsigned short int data_array[]);

int dfe_write( int fd, int slot_number, int register,
              int number_of_words,
              unsigned short int data_array[]);
```

These functions read or write the specified number of *16 bit* words to the specified register on the DFEA.

4. System Timing / SCL Emulation

A 53MHz master clock is delivered to each DFE board differentially using LVDS. Control bits are multiplexed over a single LVDS pair and sent to each board in the crate.

There are seven RF clock ticks per 132ns crossing. Thus seven bits are sent in a control bit “frame”, shown below:

<i>Frame bit</i>	<i>RAM bit</i>	<i>Description</i>
1	-	Start (always set)
2	**	FX First Crossing
3	**	SG Sync Gap
4	0	L1A Level 1 Accept
5	1	SCL Init
6	-	reserved
7	-	Odd parity

The FX and SG bits are generated automatically by the DSAT using counter, to match the tevatron accelerator timing. The DSAT can be programmed to send a (possibly repeating) pattern of up to 256 frames. For each frame, the L1A, SCL_init, ISO_IN0 and ISO_IN1 bits are specified in the SCL buffer.

The system timing is controlled by the following registers:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0	Unused																						Address									
+1	Unused																						SCL									
+2	Device ID = 0xD5A1										Firmware Rev					CSR																
+3	Unused																						Word Count									

Offsets 0 and 1 provide a standard memory buffer interface. Each buffer memory location corresponds to one 7-clock frame, with only the low 4 bits used.

The DSAT CSR (control/status register) contains the following bits:

<i>Bit</i>	<i>Name</i>	<i>R/W</i>	<i>Description</i>
0	dsat_enable	W	Overall enable.
1	dsat_busy	R	Timing cycle(s) in progress
2	dsat_global_trig	W	'1' to trigger one timing cycle
3	dsat_tick_trig	R/W	trigger on specified Tick enable bit
4	dsat_link_bypass	R/W	bypass Rx ports to Tx ports for BER test
5	dsat_link_test	R/W	send DFEA Link pattern
6		R	-not used-
7	dsat_reset	R	system reset
8--15	dsat_tick_no	W	Tick number to trigger on
16--31	dsat_rpt_count	W	Repeat count (0=once, n=every n*7 clocks)

When dsat_enable is set to '1' and dsat_global_trig is written with '1', a timing cycle will begin with the sequence of frames stored at memory address 0. When the number of words specified in the word count register is reached, the sequence will stop (if dsat_rpt_count=0) or repeat (if dsat_rpt_count <> 0). Note that empty frames containing only a 'start' bit are sent continuously whenever the DSAT is powered.

Singals are delayed to ensure that L1A, SCL_init, ISO_IN signals in the first SCL buffer word are properly aligned with Link data of the first event in the LVDS transmitter buffer.

5. LVDS Transmitter

The DSAT has 8 LVDS serial transmitters, which continuously send 28-bit frames. The data to be transmitted is stored in a standard memory buffer. Transmission of data always starts at address 0 and proceeds until the number of words specified in the word count register has been transmitted. When no data is being transmitted, the LVDS transmitters send idle words containing all zeroes.

The transmitter may be triggered by the following events:

- When a '1' is written to the **local_trig** bit of the individual transmitter CSR
- When a '1' is written to the **global_trig** bit of the global CSR, and the **global_enable** bit in the individual CSR is set.
- When an SCL frame with the **first_crossing** (FX) bit set is transmitted and the **FX_enable** bit in the individual CSR is set.

Each transmitter is controller by the following set of registers:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0	Unused																Address															
+1	Unused		Transmitter Data																													
+2	CSR (Control/Status Register)																															
+3	Unused																Word Count															

Offsets 0 and 1 control the memory buffer as described on page 3. The CSR contains the following bits:

<i>Bit</i>	<i>Name</i>	<i>R/W</i>	<i>Description</i>
0	tx_enable	R/W	Transmitter enable
1	tx_busy	R	'1' if transmitter is busy (<i>not implemented</i>)
2	tx_trig	W	Writing '1' starts transmission
3	tx_global_trig	R/W	Enable start on global_trig
4	tx_fx_enable	R/W	Enable start on first crossing
5	tx_repeat	R/W	Repeated trigger mode

6. LVDS and SLDB Receivers

The DSAT contains four LVDS receivers and two SLDB receivers. Each is controlled by the following set of registers:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0	Unused																Address															
+1	Received Data																															
+2	LVDS Data (15:0) spy															Unused								CSR								
+3	Unused					Write Pointer										Unused								Delay								

Offsets 0 and 1 control the memory buffer as described on page 3. The receiver may be

triggered by writing to the rx_trig bit in the CSR, or after a programmable delay (in events) from first crossing. Data received is always stored in the buffer starting at address zero and ending at the last buffer address (hex 0x1fff).

The memory contains the received data as follows. *For the LVDS receivers:*

Bit 31 ISO_OUT0 (rx0, rx1) or ISO_OUT1 (rx2, rx3)
 Bit 30 -unused-
 Bit 29 ISO_OUTx before synchronization
 Bit 28 Data Valid
 Bits 27..0 Received LVDS data

For the SLDB receivers:

Bit 31 Data Valid (generated by DSAT)
 Bits 30..18 -unused-
 Bit 17 DAV from SLDB
 Bit 16 Parity Error signal from SLDB
 Bits 15..0 Received SLDB data

The CSR contains the following bits:

<i>Bit</i>	<i>Name</i>	<i>R/W</i>	<i>Description</i>
0	rx_enable	R/W	RX Enable (set to '0' when accessing memory)
1	rx_busy	R	'1' when receiver is busy (<i>not implemented</i>)
2	rx_trig	W	'1' to trigger receiver
3	rx_global_trig	R/W	Enable on global_trig
4	rx_fx_enable	R/W	Enable on first crossing

The delay register sets the delay in clock ticks between the global_trig or first_crossing and the receiver going active. Bit 0 must be set to '1' to enable receiver. Then a trigger must occur (either user writes '1' to bit 2, or bit 3 or 4 is set to '1' and the corresponding event occurs).

The stored data is automatically aligned to the CTOC header (LVDS) or first muon track word (SLDB) using the documented L1CTT data format.

The upper 16 bits of the CSR are a read-only snapshot of the last word received from the LVDS receiver. This can be useful for debugging.

7. Register Layout

The overall layout of the registers in the DFEA address space is given below. The device number (Dev) is simply the upper 4 bits of the address.

<i>Dev</i>	<i>Addr</i>	<i>Description</i>
0	0x00	DFEA Timing and Control
1	0x10	R/W Bus
2	0x20	LVDS Transmitter 0
3	0x30	LVDS Transmitter 1
4	0x40	LVDS Transmitter 2
5	0x50	LVDS Transmitter 3
6	0x60	LVDS Transmitter 4
7	0x70	LVDS Transmitter 5
8	0x80	LVDS Transmitter 6
9	0x90	LVDS Transmitter 7
10	0xa0	LVDS Receiver 0
11	0xb0	LVDS Receiver 1
12	0xc0	LVDS Receiver 2
13	0xd0	LVDS Receiver 3
14	0xe0	SLDB Receiver 0
15	0xf0	SLDB Receiver 1