Serial Protocols for FEB

This document describes the protocols used to load and configure the FEB for module 0. There are 5 different protocols seen at this time.
Each of these protocols will be supported in an EPLD which I will call the SPAC Interface EPLD. This is not the SPAC EPLD but rather the one which will convert the standard SPAC parallel output to the serial protocols needed by the front end board (FEB).

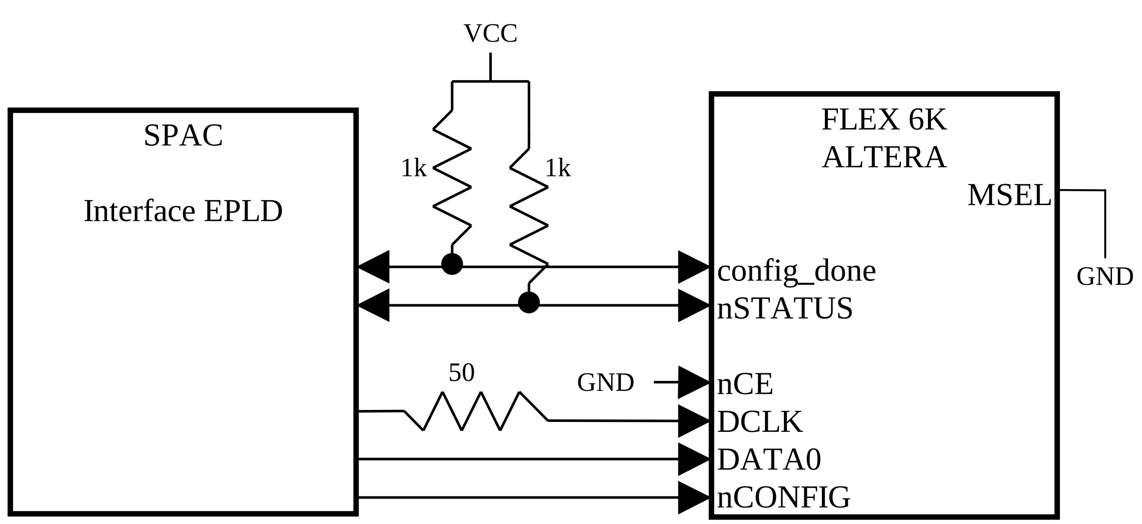
The 5 protocols are,

1) 6K Altera configuration,
2) 4k Xilinx configuration,
3) parameter loading of Xilinx and Altera (common method),
4) I2C loading for the delay line,
5) serial loading for the dac(s).
6) Shaper control lines. (For V2 version)

These will each be dealt with independently.

<u>**6K Altera configuration**</u>:

The following diagram describes how the Altera 6K should be configured. There are two DCLK lines which feed the left/right sides of the board. These will always be driven at the same time.

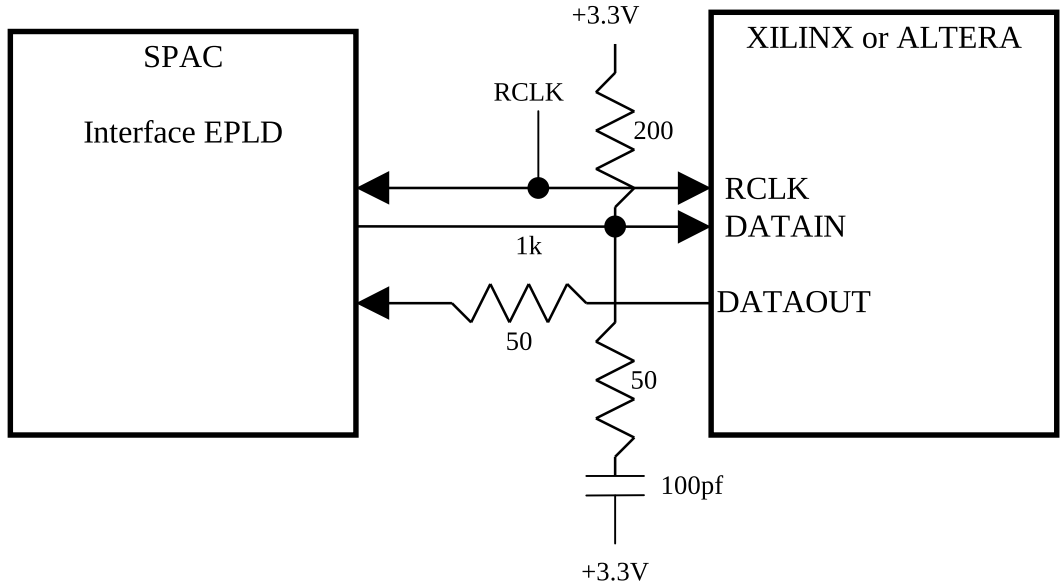


All the signals can be bussed (if all the EPLDs contain the same program).

The timing of the various signals are given in on page 13 of AN 87: "Configuring FLEX 6K Devices".
The basic sequence is :

The nCONFIG line is pulsed initiating the sequence.
Configuration clocks must not occur before 1usec after nSTATUS is released.
The data bits are then set and clocked in.
When finished, the 10K EPLD will let config_done go high. At this point an additional 10 clocks must be issued to enter user mode.
Clock rate must be under 10Mhz.

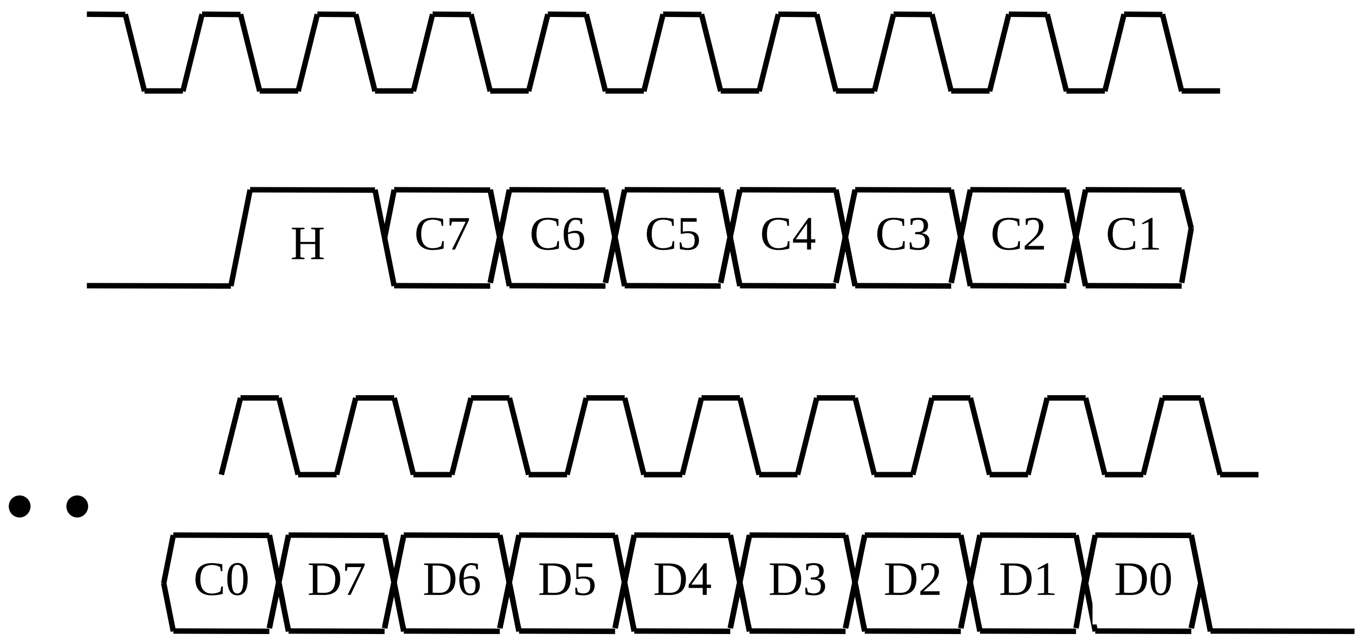
Parameter Loading Configuration:



The capacitor above will not be installed unless it is seen as necessary to terminate the line in 50 ohms to reduce noise.

The parameter loading of the FPGA chips is done by using the slow, free-running RCLK (5 MHz currently). The command and address as well as a header is transmitted on the DATAIN line. The DATAOUT line is used for the return of data. The DATAIN and DATAOUT lines both change on the falling edge of the clock (RCLK). They are to be clocked on the rising edge of RCLK at the receiver. All reads and writes contain data which is a fixed length of 1 byte.

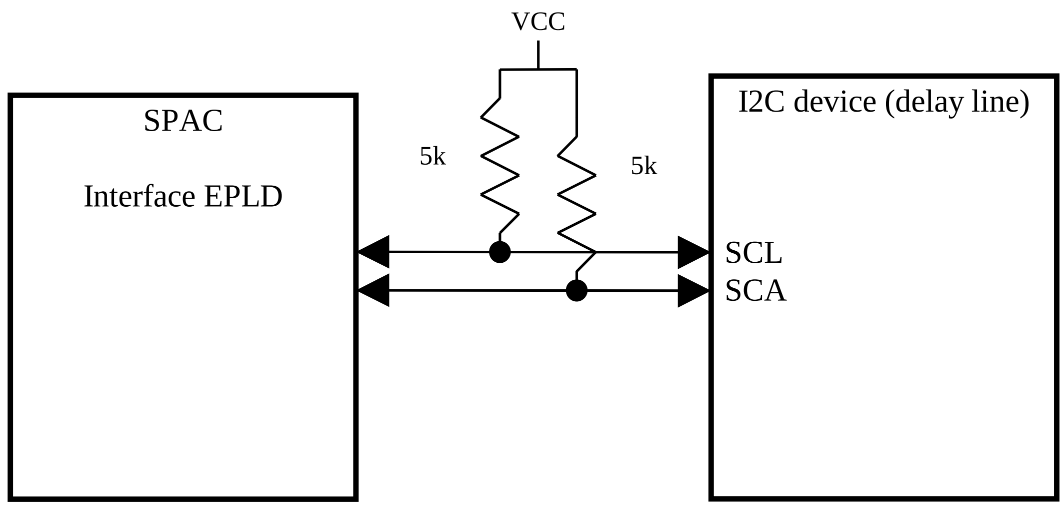
A “write” sequence is defined below,



The sequence above describes a write of “8” bits with command C[7:0]. H0 indicates the header which is a logical high. The serial line will be low when data is not being transferred. The header is used to indicate a data packet is coming.

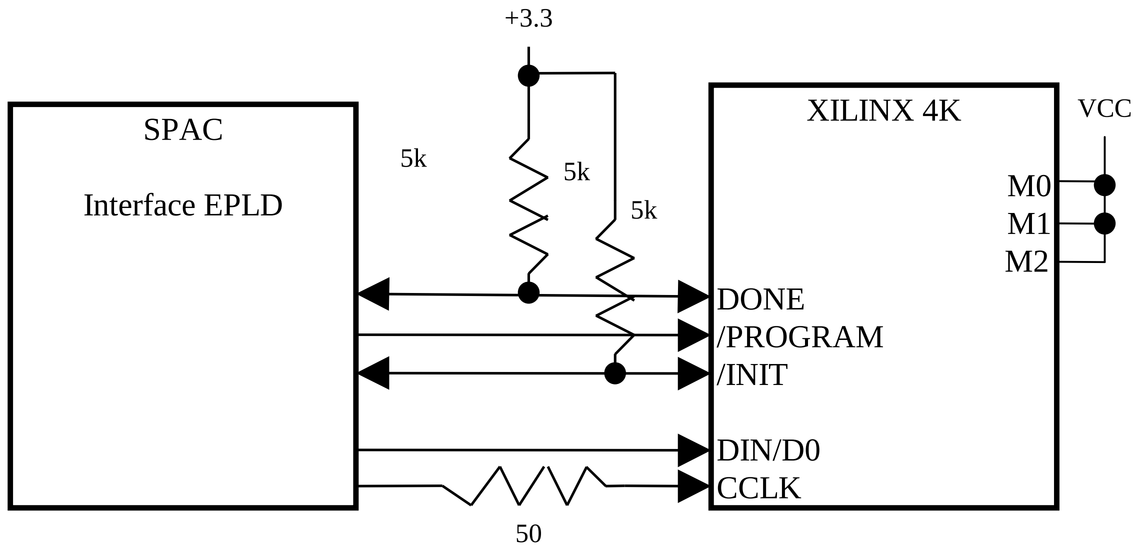
If a command is a “read”, then the last command bit is to be followed by the return data on a separate signal line. The timing of this read data is exactly the same as in the “write” sense except that the read line is driven by the slave device (the Altera device).

I2C Loading Configuration:



The loading of the I2C devices is done as shown above. The I2C protocol which is complicated is supported in the SPAC Interface EPLD. This is described in documents describing I2C such as any of the I2C compatible devices from Philips Semiconductors. See for instance PCF8570C which is a simple SRAM with an I2C interface.

4K Series Xilinx Configuration:



The CCLK line is duplicated for both the left and the right sides of the board. They are driven separately but are always driven with the same data at the same time.

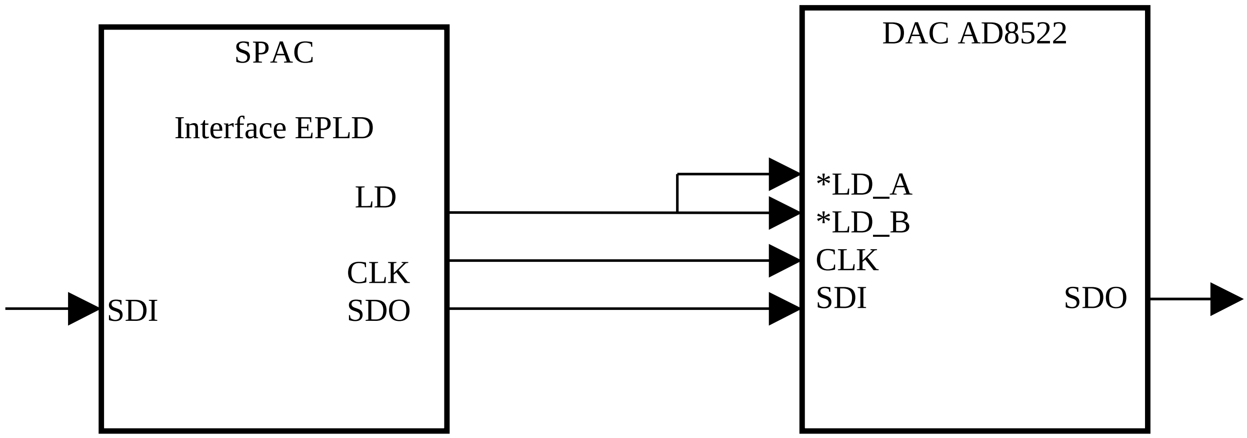
Here the sequence is as follows:

1) /PROGRAM is pulsed high-low-high
2) Wait for a long time = N msec where N = # of frames in device
   The number of frames is 1578 for the XC4028 and 1775 for the XC4036.
3) Clock all the serial data in with DIN/D0 and CCLK
4) Clock the device an additional 1 or 2 times to enter user mode.

The software implementation for this is as follows.

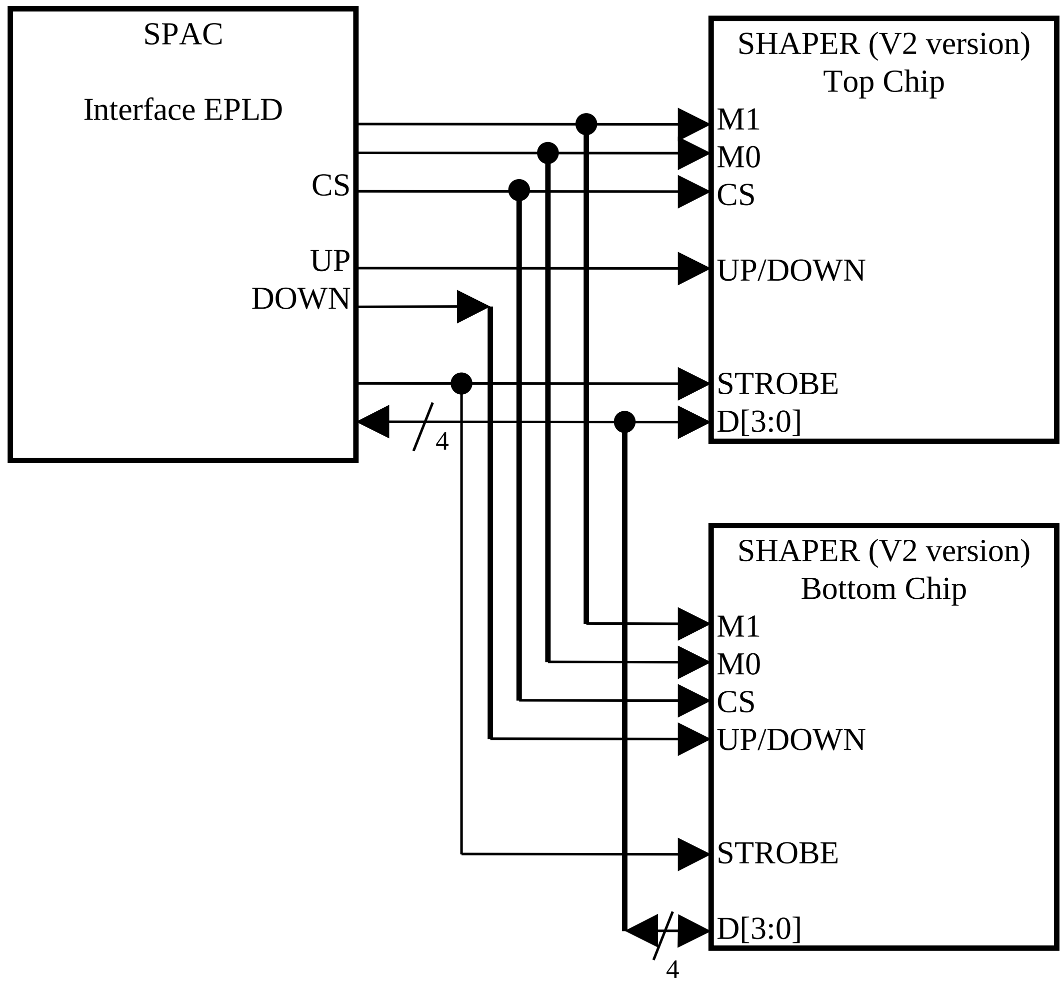
1) Through software pull /PROGRAM low (This will be a SPAC command).
2) Through software pull /PROGRAM high (Similar SPAC command).
3) Wait in a loop on the host computer for the delay specified above.
4) Send down configuration bytes through SPAC.
5) Send command to SPAC to send additional 4 CCLKs.

DAC Loading Configuration:



The loading of the DAC is done in a serial pipeline fashion. All the DACs are connected with their SDO lines to their neighbors SDI lines. The last line returns to the SPAC interface for reading back.
The serial bits are defined in the data sheet.

Shaper Control:



The connections between the shaper and SPAC interface chip is logical here. All other cases have been also physical but here we have very mismatched levels since the Shaper needs -3 and 0 as it's digital levels. The Shaper control for the FE board is described above. The two shapers shown in the above diagram are for a pair (one on the top and one on the bottom of the board). The other pairs of shapers are connected in a similar way. The UP and DOWN signal lines can be bussed to all the shaper pairs. The signals M0,M1,STROBE, and D[3:0] can be bussed to all individual shapers. The CS line must be different for each shaper pair.

The total number of control lines for the shaper on the 128 channel FE Board is therefore,

Signal	# of signals
M0	1
M1	1
STROBE	1
UP	1
DOWN	1
CS	16
D[3:0]	4
Total	**25**

The mode bits M0 and M1 are defined as follows,

M0	M1	Function
0	0	Read
1	0	Write
0	1	Set all
1	1	Clear all

SPAC COMMANDS FOR FEB

DAC Loading Commands			
Command	Bits Used	R/W	Description of command:
0x04	D[2..0]	W	This will set the bits (as shown below). These bits are the control and data lines for the DAC. All sequencing is done in software.
0x04	D[3..0]	R	This will read the current value of the bits. The bit definitions are here:

Bit #	Function
0	dac_sdo
1	dac_ld
2	dac_clk
3	dac_sdi (read only)

;

Altera Parameter Loading			
Command	Bits Used	R/W	Description of command:
0x08	D[7..0]	W	This will load the data register
0x09	D[7..0]	W	This will load the command register. This will also send the serial string to the altera chips. This command can also be a read! If this is the case, the data will be read serially and left in a register. This is accessed with the following command
0x08	D[7..0]	R	This returns with the data which was read with the most recent command.
0x09	D[0]	R	This returns the “busy still executing” (high true)

Xilinx Parameter Loading (left side)			
Command	Bits Used	R/W	Description of command:
0x10	D[1..0]	W	This will set the bits (as shown below). These bits are the strobe and the data. All sequencing is done in software.
0x10	D[2..0]	R	This will read the current value of the bits. The bit definitions are here:

Bit #	Function
0	Dataout (to Xilinx)
1	Strobe
2	Datain (from Xilinx) (read only)

;

Xilinx Parameter Loading (right side)			
Command	Bits Used	R/W	Description of command:
0x14	D[1..0]	W	This will set the bits (as shown below). These bits are the strobe and the data. All sequencing is done in software.
0x14	D[2..0]	R	This will read the current value of the bits. The bit definitions are here:

Bit #	Function
0	Dataout (to Xilinx)
1	Strobe
2	Datain (from Xilinx) (read only)

;

Altera Configuration (booting)				
Command	Bits Used	R/W	Description of command:	
0x20	D[7..0]	W	This will send out 8 data bits D[7..0] with clocks to the Altera chip .	
0x21	X	W	This will send a nCONFIG pulse.	
0x22	D[3..0]	W	This send out N config clocks where N=D[3..0]+1 to the Altera chips.	
0x23	D[0]	W	This sets the cnf_A_dout line to the value of D[0]	
0x20	D[2..0]	R	Bit #	Function
			0	CNF_A_Config_done signal
			1	CNF_A_nstatus signal
			2	Busy_A_config which indicates that this unit is busy

0X24 IS SET FOR DELAY LINE

Xilinx Configuration (booting)				
Command	Bits Used	R/W	Description of command:	
0x28	D[7..0]	W	This will send out 8 data bits D[7..0] with clocks to the Xilinx chips .	
0x29	X	W	This will send a PROGRAM pulse.	
0x2A	D[3..0]	W	This send out N config clocks where N=D[3..0]+1 to the Altera chips.	
0x2B	D[0]	W	This sets the cnf_X_dout line to the value of D[0]	
0x28	D[2..0]	R	Bit #	Function
			0	Busy_X_cnf signal which indicates that this module is busy.
			1	CNF_X_DONE signal from Xilinx chip
			2	CNF_X_/INIT signal from Xilinx chip

Shaper Loading				
Command	Bits Used	R/W	Description of command:	
0x30	D[7..0]	W	This will set the lower 8 cs bits on the shapers	
0x31	D[7..0]	W	This will set the upper 8 cs bits on the shapers	
0x32	D[7..0]	W	This will set the bits as follows: then send a pulse.	
			Bit #	Function
			D[1..0]	Shaper_M[1..0]
			D2	Shaper_up
			D3	Shaper_down
			D[7..4]	Shaper_dout[3..0]
0x33	D[7:0]	W	Same as command 0x32 but without the pulse. This is used for reading.	
0x30	D[7:0]	R	This will return the lower 8 cs bits	
0x31	D[7:0]	R	This will return the upper 8 cs bits	
0x32	D[7:0]	R	This will return the bit as defined in 0x32	
0x33	D[3:0]	R	This will return the 4 bits of read data	

I2C loading for delay line				
Command	Bits Used	R/W	Description of command:	
0x18	D[1..0]	W	This will set the bits as shown below: All sequencing is in software.	
0x18	D[3..0]	R	Bit #	Function
			0	I2C_dat (which we are sending)
			1	I2C_clk (which we are sending)
			2	I2C_dat (from ext pin)
			3	I2C_clk (from ext pin)

Reset loading				
Command	Bits Used	R/W	Description of command:	
0x38	D[3..0]	W	This will set the bits as shown below: All sequencing is in software.	
0x38	D[3..0]	R	Bit #	Function
			0	Altera_reset (low true, reset value)
			1	Xilinx_reset (low true, reset value)
			2	Spac_soft_reset (high true)
			3	Enable for overtemp interrupt (high true)

Temperature sensor loading				
Command	Bits Used	R/W	Description of command:	
0x1c	D[1..0]	W	This will set the bits as shown below: All sequencing is in software.	
0x1c	D[4..0]	R	Bit #	Function
			0	Temp_SDA (which we are sending)
			1	Temp_SCL (which we are sending)
			2	Temp_SDA (from ext pin)(read only)
			3	Temp_SCL (from ext pin) (read only)
			4	Temp_/interrupt (read only)

Pulser for Shaper DAC loading				
Command	Bits Used	R/W	Description of command:	
0x3c	D[1..0]	W	This will set the bits as shown below: All sequencing is in software.	
0x3c	D[1..0]	R	Bit #	Function
			0	PDAC_data
			1	PDAC_ld
			2	PDAC_clk

Altera loading sequence				
precursor	First 8 bits out (highest bits first)			Data (highest bits first)
1	W/R	ADD[2:0]	CMD[3:0]	D[7:0]

Commands which Altera accepts (highest bits should be shifted out first)			
Command	Bits Used	R/W	Description of command:
0xf	x	W	This will send a test pulse
0x1	D[7:0]	W/R	ID[7..0] lower bits for header ID
0x2	D[6:0]	W/R	ID[14..8] upper bits for header ID
	D[7]	W/R	AUTO
0x3	D[7:0]	W/R	UL[7:0] lower bits for upper threshold
0x4	D[7:6]	W/R	NG[1:0]
	D[5:4]	W/R	GA[1:0]
	D[3:0]	W/R	UL[11:8]
0x5	D[7:0]	W/R	LL[7:0] lower bits for lower threshold
0x6	D[7:6]	W/R	GC[1:0]
	D[5:4]	W/R	GB[1:0]
	D[3:0]	W/R	LL[11:8]
0x7	D[7:0]	W/R	TD[7:0]
0x8	D[3:0]	W/R	TD[11:8]
	D[6]	W/R	TMODE (low true indicates Xilinx is ignored)
	D[7]	W/R	TEST