Pixie-Net XL interface with TTCL

January 3, 2023

XIA LLC
2744 East 11th St,
Oakland, CA 94601 USA
Email: support@xia.com
http://www.xia.com/

Table of Contents

1 Introduction

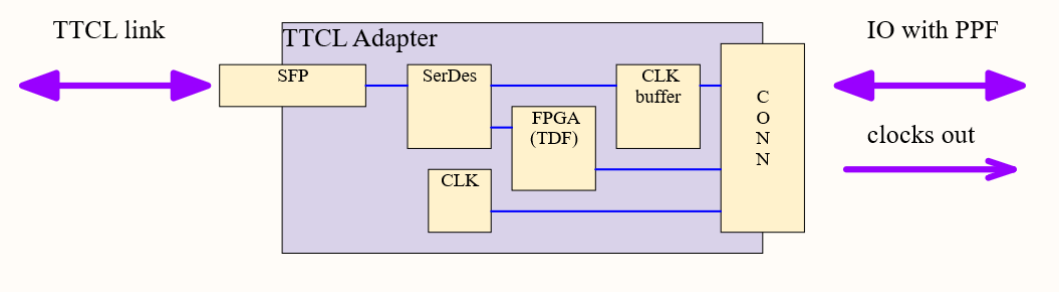
The Pixie-Net XL is designed as a digital detector readout module with high speed Ethernet data outputs. Several methods can be used to synchronize clocks and triggers when operating multiple Pixie-Net XL modules, or when operating a Pixie-Net XL module with other electronics. These methods include dedicated clock/trigger cabling on GPIO and coaxial inputs, White Rabbit IEEE 1588 network time synchronization, and the ability to add interface adapter boards to the system. This document describes the specification and functionality of an adapter board for the Trigger Timing and Control Link (TTCL) used in GRETINA and Digital Gammasphere, and the forthcoming GRETA Trigger, Timing and Control link (GTCL) of the GRETA trigger system[1].

[1] https://wiki.anl.gov/wiki_gsdaq/images/4/44/20160418_trig_command_link.pdf

2 Physical Board Description

The Pixie-Net XL TTCL adapter board is a roughly 2in x 3in circuit board that plugs into the “WRclkDB” connectors on the Pixie-Net XL main board. Through this connector to the Pixie-Net XL, the TTCL adapter board can provide clocks for multiple devices on the Pixie-Net XL main board: SFP Ethernet connection, the analog to digital converters (ADC) digitizing the detector signals, and the FPGA that processes the ADC data (pulse processing FPGA, or **PPF**). The TTCL adapter board also connects to digital I/O pins on the pulse processing FPGA for trigger distribution and control, including an SPI interface.

On the Pixie-Net XL main board, there are two PPFs with associated ADC boards and “WRclkDB” connectors. For the purposes of this document, only the operation of a single adapter board with a single PPF is considered. The second PPF can be operated either in the same way as the first with its own adapter board, or it can share clocks and triggers with the first PPF in a manner yet to be defined. That is outside the scope of this document at this time.



The TTCL adapter board includes the following components as illustrated in the diagram above:

- An SFP connector for the link to the TTCL trigger distribution system. This link carries a continuous stream of data at 1Gbps that contains an endlessly repeating cycle of frames containing trigger and synchronization messages (not Ethernet),
- A serializer/deserializer chip to reformat the TTCL data.
- An FPGA for decoding the TTCL data, extracting the timestamp, commands and event accept messages from the trigger systems of Gammasphere, GRETINA and GRETA (TTCL decoding FPGA, or **TDF**). The TTCL decoding FPGA creates signals to the Pixie-Net XL indicative of timestamp synchronization and event acceptance from external trigger systems
- A clock distribution chip generating clocks for pulse processing FPGA and ADCs from the SerDes recovered clock
- A fixed clock chip for the FPGA Ethernet interface

3 Clock Distribution and Reset

3.1 TTCL clock distribution

The TTCL high speed serial link data link incorporates a clock synchronized to the system master clock. The SerDes chip recovers the clock and feeds it into the programmable clock buffer, which generates

- 50 MHz for the PPF for purposes of TTCL interfacing
- 125 MHZ for the ADC clock distribution tree on the main board (multiplied to 250 or 500 MHz as needed). This tree also clocks PPF pulse processing logic.
- 50 MHz (or other frequency) for the TDF

This ensures all clocks on the Pixie-Net XL relevant for data collection and time stamping are synchronized to the master clock.

3.2 Local clocks

Fixed frequency oscillators provide 156.25 for the PPF GTX pins (for use in the SFP 10G Ethernet link) and 200 MHz for the TDF.

3.3 Clock Counter Reset

Both TDF and PPF use the recovered TTCL clock (50 MHz) to increment a local clock counter. The increment is 2 in each clock cycle to match the TTCL 100 MHz clock scaling. The clock counter value is synchronized with the TTCL master time and used for timestamping events.

Timestamp synchronization in the TDF uses two forms of TTCL sync messages:

1. Every 2 microseconds the trigger systems send the Sync message that informs the receiving system what the timestamp is. This can be thought of as Big Ben tolling the hour. The receiving system may use the timestamp contained within the Sync message to check its own timestamp counter and re-align as needed. For example, if there are intermittent communication errors on the TTCL/GTCL, the timestamp received every 2us can sometimes detect that. If the connection errors are severe enough that the SerDes chip drops out of physical lock for milliseconds at a time the typical clock tree response will be to fall back to the local oscillator that's close but not identical to the TTCL clock. Automatically resynchronizing to the timestamp every Sync can help in this case by minimizing drift in the timestamp counter.
2. The other form of the message is an Imperative Sync. When this is received the receiver must load the timestamp counter with the value in the message and release the counter to count coincident with the last word in the 5-word message frame. This is used at system startup to ensure everybody starts counting at the same time. Typically the receivers reset the clock to zero (GRETINA) but DGS firmware allows for a non-zero start value because of ANL setups where multiple digital systems are slaved to a single "monarch" that is the timing master for all detectors.

The synchronization is forwarded to the PPF as follows:

The TDF implements a set of registers written by the PPF that specify a future time of synchronization T_sync. Control bits in the pulsed control register enable the logic to compare T_sync with the TDF current time (synchronized to TTCL as described above). When the TDF time counter is equal to T_sync, the TDF issues a pulse to the PPF.

When the system is first brought up, the PPF therefore i) reads the current time from the TDF for coarse time matching, ii) adds an appropriate offset to accommodate read/write delays between TDF and PDF to compute T_sync, iii) writes T_sync to the TDF and enables the comparison logic, and iv) upon receipt of the synchronization pulse at T_sync, sets the PPF time counter equal to T_sync.

During operation, the PPF can periodically perform the same procedure for re-synchronization. Alternatively the TDF may implement a comparison of only the lower 16 time counter bits, which will then generate a periodic signal every few microseconds to the PPF.

4 Overview of Trigger Distribution

As background information, section 4.1 and 4.2 give an overview of the trigger concept in the Pixie-Net XL and in a representative of devices designed to be synchronized via TTCL (here: Gammasphere firmware). Section 4.3 and 4.4 describe how the TTCL synchronization works in the Pixie-Net XL for receiving and issuing triggers, respectively

4.1 Pixie-Net XL Trigger Concept

Each DAQ channel of the Pixie-Net XL runs local trigger logic to detect incoming pulses. The logic compares the threshold TH to the difference of two sums with L ADC values each, spaced apart by G samples (L = sum length, G = gap between them). L, G and TH are user programmable parameters. When an over threshold condition occurs, a one-cycle pulse is issued to

- latch the timestamp (= value of local clock counter)
- start a counter for the pileup inspection period, which is the length of the energy filter
- start collecting waveform data in a "waveform FIFO" buffer (the data is delayed by a few microseconds to capture pre-trigger data)
- start a number of optional, parallel processes such as CFD time analysis and capture of short sums along the pulse for pulse shape analysis
- latch other status information, for example the state of an external logic signal (for gating) or the value of a counter incremented by an external clock
- notify other channels/modules via local and external trigger distribution logic. These other channels can use the trigger to start the above processes, or ignore it. This has been used in the past for example to capture data from all channels of a segmented Ge detector based on the trigger in one segment.

At the end of the pileup inspection, the energy filter sums are also latched, together with a flag indication if piled up or not (i.e. was there a second trigger within the pileup inspection period. When the optional other processes are done also (usually they take less time than the energy filter, but not always), all this information is assembled in a "header FIFO" buffer. By this time also any return values from the trigger distribution logic can be included. The contents of the header and waveform FIFOs are then packaged, buffered, and sent out via the Ethernet interface. If desired, those pulses not matching certain accept criteria (pileup) are not sent out.

The pileup inspection period is typically a few microseconds (e.g. ~6 us energy filter for Ge detectors) and processes such as the CFD time analysis takes a few hundred ns. That gives the trigger distribution logic sufficient time to send around triggers, look for multiplicity or certain channel patterns within a time window, and send the result(s) back to the channels. These results are historically called "validation triggers". In case more time is needed, delays can be added before or after the trigger logic.

4.2 Gammasphere Trigger Concept

In the Gammasphere firmware, there are two energy summation buffers M1 and M2, separated by short buffers D3, D2, D, K and K0 that as a group span the rise time. The ADC samples on both sides of 'D' are used with a bit of filtering for the discriminator in leading-edge mode, or if the user wants CFD, the threshold discriminator moves across 'K' to pre-arm the CFD, and the CFD uses 'D' as the delay factor. When the discriminator fires, the sums across M1 and M2 are stored in a header for the energy data and software external to the firmware does the pole-zero correction based upon other values in the header including the delta-T since the last discriminator hit and other running sums. Similar to the Pixie-Net XL firmware, waveform data is held in a 20us long "trigger delay buffer" and a list of the discriminator timestamps of the hits in the buffer is held in parallel to the buffer. When the trigger sends an accept, numerical subtraction of timestamps of discriminators relative to the timestamp value in the accept message, with two registers to define a window, selects which hit(s) get(s) read out. Pileup recognition is similarly based upon the size of M1/M2. Most people set M1/M2 to around 4-6us.

4.3 Receiving TTCL Triggers in the Pixie-Net XL

When the TDF is decoding the TTCL data and finds an "accept" message, it issues a signal to the PPF as follows:

When the decoder in the TDF sees the trigger decision from the TTCL, it will add a fixed value (say, 256 counts that at 10ns per LSB corresponds to 2.56us) to the timestamp value in the decision message to compensate for the trigger formation and transmission latency and load that value into a small first-word fall-through FIFO. When the local clock counter of the TDF matches the value at the output of the FIFO, a '1' is entered into a 64K x 1bit block RAM delay buffer and the FIFO is advanced to the next word.

The block RAM is set up as a simple delay buffer, with length controlled by a register. A reset bit will reset the block RAM address pointers whenever the length value is changed. The block RAM delay can be chosen to compensate for the Pixie-Net XL energy integration and pileup times. That way the two delay factors are separate and the one that gets controlled through the PPF is related solely to Pixie delay factors.

When the '1' finishes the delay through the block RAM, the TDF issues a pulse to the PPF. The PPF then uses that pulse for validation of any triggers occurring in the offset time before.

4.4 Issuing TTCL Triggers from the Pixie-Net XL

When the PPF is detecting triggers, it can distribute it to the TTCL master as follows:

The PPF tests for multiplicity among the local channels within that FPGA by stretching the one-cycle trigger pulses for N cycles to form a coincidence window. It continuously builds an OR of the stretched trigger, and if the result is logic 1, it will set high one of the digital lines to the TDF. The TDF, upon the rising edge of the line, latches the time stamp of the local clock counter and creates a message to the TTCL system.

This functionality likely is not implemented in the first round of development

5 Firmware Specifications

5.1 I/O definitions for the TDF

```
entity TriggerInterface is
Port (
--=================================================================
-- Clock signals into design
--=================================================================
    LOCAL_FPGA_OSCN         : in std_logic;        --free running onboard oscillator
    LOCAL_FPGA_OSCP         : in std_logic;        --free running onboard oscillator
    LOCAL_FPGA_CLKN         : in std_logic;        --Jitter cleaned clock from Si5324.
    LOCAL_FPGA_CLKP         : in std_logic;        --Jitter cleaned clock from Si5324.
--=================================================================
-- Interface to/from host (Pixie system)
--        Signals are listed in pin number order of 40-pin Pixie connector.
--=================================================================
    --Pixie connector pin 1 is +3.3V power for board.
    --Pixie connector pin 2 is ground.
    --Pixie connector pin 3 is +3.3V power for board.
    --Pixie connector pin 4 is GTXa_CLKP, from oscillator, no connection to FPGA.
    WR_CLK20_VCXO            : in std_logic;        --pin 5 of Pixie connector.  Clock capable input, bank 2 of
FPGA.
In WR usage, this is an input to the PPF, on a clock capable pin.
    --Pixie connector pin 6 is GTXa_CLKN, from oscillator, no connection to FPGA.
    --Pixie connector pin 7 is ground.
    --Pixie connector pin 8 is ground.
    DECODED_TRIG_FLAG       : out std_logic;    --pin 9 of Pixie connector, WR_PLLDAC_SYNC*.  Output of FPGA to
Pixie.
    --Pixie connector pin 10 is PIXIE_FPGA_CLKP, AC coupled output of Si5324 clock generator, no connection to
FPGA.
    FPGA_XTRA_IO_5              : in std_logic;          --pin 11 of Pixie connector, WR_PLLDAC_SCLK.  Extra
connection, clock capable pin, bank 0.
    --Pixie connector pin 12 is PIXIE_FPGA_CLKN, AC coupled output of Si5324 clock generator, no connection to
FPGA.
    FPGA_XTRA_IO_4          : in std_logic;       --pin 13 of Pixie connector, WR_PLLDAC_DIN.  Extra connection,
clock capable pin, bank 0.
    --Pixie connector pin 14 is ground.
    FPGA_RSTn               : in std_logic;        --pin 15 of Pixie connector, WR_PLLDAC_CLR*.  Used as active
LOW reset to FPGA logic.
    FPGA_SCLK               : in std_logic;        --pin 16 of Pixie connector, WR_PLL_SCLK.  SPI control SCLK
to FPGA.  Clock capable pin, bank 3.
    FPGA_XTRA_IO_3              : in std_logic;          --pin 17 of Pixie connector, WR_PLLDAC_LDAC*.  Extra
connection, clock capable pin, bank 2.
    DECODED_SYNC_FLAG         : out std_logic;    --pin 18 of Pixie connector, WR_PLL_SYNC.  Output of FPGA to
Pixie.
    --Pixie connector pin 19 is SDA to EEPROM, no connection to FPGA.
    FPGA_CS                    : in std_logic;        --Pin 20 of Pixie connector, WR_PLL_CS.  SPI chip select
to FPGA.
    --Pixie connector pin 21 is SCL to EEPROM, no connection to FPGA.
    FPGA_MOSI               : in std_logic;       --Pin 22 of Pixie connector, WR_PLL_SDO.  Serial control data
TO the FPGA from Pixie.
    FPGA_XTRA_IO_2             : in std_logic;         --Pin 23 of Pixie connector, WR_ONE_WIRE. Unused by FPGA,
but connected as spare.
    FPGA_MISO               : out std_logic;    --Pin 24 of Pixie connector, WR_PLL_SDIO.  Serial control data
FROM the FPGA to Pixie.
    --Pixie connector pin 25 is ground.
    --Pixie connector pin 26 is ground.
    DECODED_SM_LOCKED       : out std_logic;    --Pin 27 of Pixie connector, WR_EXTRA.  Decoded output from SFP,
driven by FPGA to Pixie.
    --Pixie connector pin 28 is GTX_CLKP, from oscillator, no connection to FPGA.
    FPGA_XTRA_IO_6          : in std_logic;       --Pin 29 of Pixie connector, "IO29".  Extra spare connection.
    --Pixie connector pin 30 is GTX_CLKP, from oscillator, no connection to FPGA.
    --Pixie connector pin 31 is ground.
    --Pixie connector pin 32 is ground.
    --Pixie connector pin 33 is SCL to clock generators and SFP, no connection to FPGA.
    --Pixie connector pin 34 is ADC_CLKP, DC coupled output from Si5324, no connection to FPGA.
    --Pixie connector pin 35 is SDA to clock generators and SFP, no connection to FPGA.
    --Pixie connector pin 36 is ADC_CLKN, DC coupled output from Si5324, no connection to FPGA.
    FPGA_XTRA_IO_1          : in std_logic;       --Pin 37 of Pixie connector, "IO37".  Extra spare connection.
    --Pixie connector pin 38 is ground.
    --Pixie connector pin 39 is ground.
```

```
    PIXIE_25MHZ                : in std_logic;            --wclk        Pin 40 of Pixie connector,25MHz clock
from Pixie system.  Connected to clock capable, bank 2.
    --============================================
    --    Connections to the DS92LV18 SERDES chip
    --============================================
    DS92LV18_DEN        : out std_logic;    --SERDES Driver Enable
    DS92LV18_REN        : out std_logic;    --SERDES Receiver Enable
    DS92LV18_LINE_LE    : out std_logic;    --SERDES Line Loopback Enable
    DS92LV18_LOCAL_LE    : out std_logic;    --SERDES Local Loopback Enable
    DS92LV18_SYNC        : out std_logic;    --SERDES Sync pattern control
    DS92LV18_TPWRDN        : out std_logic;    --SERDES Transmitter power-down (Active Low)
    DS92LV18_RPWRDN        : out std_logic;    --SERDES Receiver power-down (Active Low)
    DS92LV18_LOCK        : in std_logic;        --SERDES Lock status (Low = Locked)
    DS92LV18_TXCLK        : out std_logic;    --Multiplex clock output to drive SERDES chip TXCLK
    DS92LV18_REFCLK        : out std_logic;    --FPGA output to drive SERDES chip REFCLK
    --There is no direct connection from the DS92LV18 RCLK to the FPGA.
    --The LOCAL_FPGA_CLKP/LOCAL_FPGA_CLKN differential clock from the Si5324 is
    --driven from RCLK.  This should be the clock used as the IOB clock for DS92LV18_RXDATA.
    --
    --    Note that LOCAL_FPGA_CLKP/LOCAL_FPGA_CLKN comes in as a bank 3 clock, so this means it will
    --    have to be BUFG'd and can't be a local-to-IOB clock.
    --
    DS92LV18_RXDATA        : in std_logic_vector(17 downto 0);        --data received from SERDES (bank 0)
    DS92LV18_TXDATA        : out std_logic_vector(17 downto 0)    --data transmitted to SERDES
    --============================================
    --    Connections to the Si5394A clock generator.
    --============================================
    SI5394A_RST            : out std_logic;        --chip reset
    SI5394A_LOS_XAXB    : in std_logic;            --loss of sync status
    SI5394A_LOLB        : in std_logic;            --loss of lock status
    SI5394A_INTR        : in std_logic;            --interrupt request
    --============================================
    --    LED indicators
    --============================================
    SFPLED1                : out std_logic;        --blue LED of CLMVC-FKA-CL1D1L71BB7C3C3 RGB part.
    SFPLED2                : out std_logic;        --green LED of CLMVC-FKA-CL1D1L71BB7C3C3 RGB part.
    SFPLED3                : out std_logic;        --red LED of CLMVC-FKA-CL1D1L71BB7C3C3 RGB part.
    SFP_SPD_LED            : out std_logic;        --red LED for SFP status.
    --============================================
    --    SFP status/control
    --============================================
    TDIS_SFP            : out std_logic;        --pull low to disable transciever; pulled up on board.
    LOS_SFP                : in std_logic;            --pulled low by SFP if no photons arriving at RX detector.
Pulled up on PCB.
    MOD_PRESENT_SFP        : in std_logic;            --pulled low by SFP if SFP is plugged into board.  Pulled
up on PCB.
    --============================================
    --    GPIO pins on header J5
    --
    --    Yes, these could be a bus, but over time likely some will be 'in' and
    --    some will be 'out'.
    --============================================
    GP_DIAG1            : inout std_logic;        --clock capable input of FPGA, part of bank 0.
    GP_DIAG2            : inout std_logic;        --clock capable input of FPGA, part of bank 1.
    GP_DIAG3            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    GP_DIAG4            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    GP_DIAG5            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    GP_DIAG6            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    GP_DIAG7            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    GP_DIAG8            : inout std_logic;        --standard I/O of FPGA.  Part of bank 1.
    --============================================
    --    Discrete test points.
    --============================================
    TEST_POINT1            : inout std_logic;        --clock capable input of FPGA, part of bank 0.
    TEST_POINT2            : inout std_logic;        --clock capable input of FPGA, part of bank 0.
    TEST_POINT3            : inout std_logic        --clock capable input of FPGA, part of bank 1.
);

end TriggerInterface;
```

5.2 Register Interface in TDF

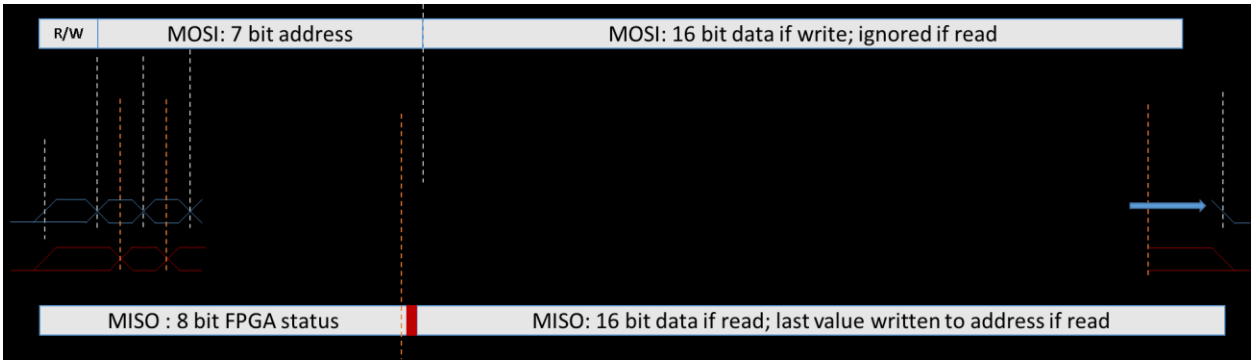
The TDF implements a serial interface that is controlled by the PPF. Serial data is presented on the FPGA_MOSI pin on every falling edge of the FPGA_SCLK signal, and the TDF latches data in on the rising edges of FPGA_SCLK. The TDF asserts data back to the PPF on the rising edges of FPGA_SCLK

and the PPF latches the data from the TDF on the falling edges of FPGA_SCLK. The FPGA_SCLK is discontinuous, needing only to be asserted when a transaction is in progress. The TDF begins a transaction by asserting the FPGA_CS line LOW, then asserting the first bit of data onto FPGA_MOSI, then starts the first bit of data transfer by issuing a rising edge on FPGA_SCLK. FPGA_SCLK is low when a transaction is not in progress. FPGA_CLK is a slow clock, typically 5MHz or less, to allow FPGA logic that runs at typically 100MHz to perform operations between edges of FPGA_SCLK.

A serial transaction between PPF and TDF consists of a 24-bit serial sequence. The first 8 bits asserted on FPGA_MOSI define the direction of the transaction and the internal register address, and the latter 16 bits are the data to write, if the transaction is a write.

Read/Write Command	Device Address							Data Bytes													
R/W	A6	A5	A4	A3	A2	A1	A0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

The R/W bit is defined as 1=R, 0=W. The transaction timing is provided in Figure x.



On writes, the TDF waits for the falling edge of FPGA_CE to indicate “start of transaction”. The first rising edge of FPGA_SCLK is used to latch the R/W* bit. The TDF saves this bit to control what happens at clock #8 of the sequence. The next 7 clocks are used to transmit the address of the register to be read or written. During the transmittal of the R/W* and address, an eight bit “FPGA status word” is clocked out on MISO that provides general, non-address specific status information about the state of the TDF.

After latching the least significant bit of the address with the 8th rising edge of FPGA_SCLK, the TDF flips an internal multiplexer that is the source of MISO from internal status to the register addressed by the data provided in the previous FPGA_SCLKs. In the time interval *after* the 8th rising edge of FPGA_SCLK but *before* the 8th falling edge the value of MISO may be indeterminate.

- If the transaction is a READ, on the following 16 FPGA_SCLKs the current value of the register selected is output on MISO, most significant bit first. The data presented on MOSI is ignored.
- If the transaction is a WRITE, the current value of the register is also provided on MISO during the 16 data clocks, but after the rising edge of the 24th clock the selected register is overwritten with the 16 bits of data that have been sent on MOSI.
    - If verification of a write is required a 2nd READ transaction is required after the WRITE, as the data presented during the WRITE is the *previous value* the register had *prior* to the WRITE command.

5.3 Registers Implemented in TDF

Few of the 128 possible addresses available are implemented in the TDF. The register map at present is

- Address 0 is the *pulsed control* register. Writing 1s to the different bits of this register creates internal pulses for resets or state machine enables. The register self-clears after each write and will always read back zero.
- Address 1 is the SerDes Control register. Bits in this register directly control the DS92LV18 SerDes chip operation.
- Address 2 is the LED register. Bits in this register control the LED indicators of the board.
- Address 3 is the Diagnostic Control register. Bits in this register control the operation of diagnostic features such as what signals are routed to test points or signals connected to any internal diagnostics (e.g. Xilinx "ILA" blocks).
- Addresses 4, 5 and 6 provide the low, middle and high 16 bits of the current timestamp within the TDF FPGA. As serial transactions would span many timestamp counts, these registers are *latched* by writing a 1 to one of the bits of the *pulsed control* register so that the 48-bit value is captured all at once.
    - These registers are read-only.
- Address 7 is the *timestamp offset* register that allows control over the timestamp offset to be applied to any event accept messages from the external trigger system.
- Address 8 is the *accept message delay* register that sets the length of the RAM delay buffer applied to any decoded event accept messages.
    - The value written to this register becomes the new delay when one of the bits in the *pulsed control* register is written, so that the TDF may perform sequenced resets of the delay logic.
- TS error counter 9
- Addresses 10 through 125 are reserved.
- Address 126 is the *code date* register. This reads back a number, in hexadecimal, that may be interpreted as the date associated with the firmware within the TDF. For example, reading 0x20220417 would be a code date of April 17, 2022.
- Address 127 is the *code revision* register. This reads back a simple version number from 0 to 65535 indicative of firmware revision.

5.4 Control Options Implemented in PPF

- TTCL_APPR_WINDOW
  the length (in 8ns clock cycles) that a TTCL trigger approves events in the PPF
  If zero, all events are approved
  Kintex Sys register 43
- TTCL_CLK_OUT
  output TTCL derived clock to LMK PLL (=1) or local clock (=0)
  Kintex CSR bit 14
- TTCL_CLK_SL
  chose Spartan 6 (=0) or Jitter cleaner (=1) as source for TTCL derived clock
  Kintex CSR bit 15

