FPGA-Based TOF System

E. Hazen
January 18, 2006

This document describes replacement logic for the TOF system built and documented previously by W. Earle [refs]. The DSP and ECL logic is replaced by a Xilinx Spartan-3 Development Board with a companion interface board. The Spartan-3 FPGA generates the pseudo-random bit sequence based on data stored in on-chip RAM, and records TDC hits similarly. The computer interface is an RS-232C serial port provided on the Spartan-3 board.

This initial version contains only an appendix describing the serial protocol. Changes from previous version highlighted.
Appendix A - Serial Protocol

The serial interface operates at 9600 baud, 8 data bits, 1 stop bit, no parity. The protocol is line-oriented ascii, so it may be conveniently tested with a serial terminal program such as Hyperterminal (windows) or minicom (linux).

On power-up the board sends an identification string:

    TOFnn>

Where 'nn' is the current firmware revision (currently '08').

The board sends a '>' prompt when it is ready to accept a command. Each command sent to the board should be terminated with an ascii CR (hex 0x0d). By default, characters sent to the board are not echoed. This may be changed with the 'E' command.

All data is in hexadecimal. The number of hex digits shown in the table must always be sent, so for example to represent a zero address one must send '000'.
**Command Descriptions**

**Single Register Access**

The board has 14 read-write registers and one read-only register. Each register is one byte wide.

- **PW0 nn** Number of cycles for TOF mode (lsb)
- **PW1 nn** Number of cycles for TOF mode (msb)
- **PW2 nn** Number of bits in random sequence (lsb)
- **PW3 nn** Number of bits in random sequence (msb)
- **PW4 nn** Width of one time bin, 250 ns units [1-10]
- **PW5 nn** Number of us in continuous mode (byte 0)
- **PW6 nn** Number of us in continuous mode (byte 1)
- **PW7 nn** Number of us in continuous mode (byte 2)
- **PW8 nn** Number of us in continuous mode (byte 3)
- **PW9 nn** Baud rate divisor (bits 0-7)
- **PWA nn** Baud rate divisor (bit 8)
  - 9600 baud 0x0145 (PW9 45; PWA 01)
  - 19200 baud 0x00A3
  - 38400 baud 0x0051
  - 57600 baud 0x0036
- **PWD nn** Control Register 1 (momentary bits)
  - bit 0 reset* Stop any run in progress; prepare the logic to start a new run.
  - bit 1 start continuous mode run*
  - bit 2 start TOF run *
  - bit 3 update baud rate *
  - bits 4-6 reserved- (set to '0')
  - bit 7 test LED
- **PRE** Status Register (read-only)
  - bit 0 run in progress ('1')
- **PWF nn** Control Register 2 (latched bits)
  - bit 0 '1' Enable loop-back mode.
  - TOF: seq output direct to input
  - Continuous: 1MHz pulser
  - bit 1 Input select
    - '0' for standard ECL input
    - '1' for front-panel BNC input
  - bit 2 TOF buffer select
  - bit 3 readout buffer select
Each individual register may also be read by sending:

\[ \text{PR} r \]

where \( r \) is the register number.

* Bits perform the specified function when a '1' is written. They will always return '0' when read.

**Block RAM Access**

The board contains two blocks of memory. The first is used to contain the pseudo-random sequence, which may be up to 1024 bits. The second contains the TDC data for TOF mode, arranged as up to 1024 words of 16 bits each. The memory is always accessed in bytes, with the lsb in the lower address for 16-bit quantities.

For the pseudo-random sequence, only bit 0 of each byte is used.

The TDC data RAM is read-only.

- **BRW0** aaa dd
  - Write single byte to pseudo-random sequence data
  - aaa memory address (0-7ff)
  - dd data (0-ff)

- **BRR0** aaa
  - Read single byte from pseudo-random data

- **BBR0** aaa
  - Read 256 bytes from pseudo-random data

- **BRR1** aaa
  - Read single byte from TDC data

- **BBR1** aaa
  - Read 256 bytes from TDC data

There are two additional commands which can be used to load the pseudo-random sequence more efficiently:

- **ST**
  - Start a new sequence
  - This will also zero the memory (like ZR0)

- **SD** aa bb cc dd
  - Send up to 4 bytes of sequence data
  - aa hex value of bits 0-7
  - bb hex value of bits 8-15
  - cc hex value of bits 16-23
  - dd hex value of bits 24-31
  - The lowest bit (bit 0) is written first

If the sequence data is read back, only the low bit is used.

**Miscellaneous Commands**

- **E 1**
  - Enable echo of characters sent to board

- **E 0**
  - Disable echo of characters sent to board (default)

- **ZP**
  - Zero all parameters

- **ZR0**
  - Zero block RAM 0 (random sequence data)
**Notes on Operation**

Parameters and RAM contents should only be modified when the board is not running, otherwise the results will be undefined. To stop a run in progress, issue the command 'PWD 01'. To test if a run has completed, issue the command 'PRE'. If bit 0 = '0', then the run is completed. If bit 0 = '1', then the run is not completed.

To start a new run, use the following sequence:

```
PWD 01       reset
(set new parameter values if desired)
```

```
PWD 04       start new TOF run
```

**TOF Example**

To start a run in TOF mode of 1000 cycles using 500ns time bins, using a 10-bit sequence of alternating 1's and 0's:

```
PWD 01 Reset, stop any run in progress
PW0 e8 Set lsb of cycle count (1000 = 0x3e8)
PW1 03 Set msb of cycle count
PW2 0a Set sequence length lsb
PW3 00 Set sequence length msb
PW4 02 Set bin width to 500ns
ST Start of sequence
SD 55 01 Sequence data (two bytes)
PWD 04 Start run
```

After the run has started, you can test if it has finished by sending the command "PRE" which will return the contents of the status register. The least-significant bit will be '0' when the run is complete.

**TOF Double-Buffer Example**

This shows how to use the 'ping-pong' double-buffer in TOF mode. The first part of the initialization is just as for normal mode:

```
PWD 01 Reset, stop any run in progress
PW0 e8 Set lsb of cycle count (1000 = 0x3e8)
PW1 03 Set msb of cycle count
PW2 0a Set sequence length lsb
```
Then you must select one of two buffer configurations. The two choices are '04' and '08', stored in register 'F'. For the first run you must wait until it is finished, and do not read out the data.

Continue in this fashion. As long as the run takes longer than the readout time, no time at all is wasted.

**Continuous Mode**

*Continuous mode is activated by setting bit 1 of the control register.* The only other parameter which must be set is the number of microseconds, as PW9, A, B, C. Note that the longest possible run is $2^{32}-1$ us which is a rather long time (71 minutes).

*Bit 0 of the status register indicates that the run is in progress, just as for TOF mode.*
Continuous mode data appears in the first 4 locations of block RAM 1.

**Continuous Mode Example**

To do a continuous mode run of 1000 us:

```plaintext
PWD 01  Reset, stop any run in progress
PW5 e8  Set byte 0 of us (1000 = 0x3e8)
PW6 03  Set byte 1
PW7 00  Set byte 2
PW8 00  Set byte 3
PWD 02  Start run
```

After the run has started, you can test if it has finished by sending the command "PRE" which will return the contents of the status register. The least-significant bit will be '0' when the run is complete.

To read the continuous mode data, issue the command 'BBR1 000'. The first four bytes returned will contain the number of counts.

**Changing Baud Rate Example**

The default baud rate is 9600. To change it to (for example) 38400 baud, issue the following commands (at 9600 baud, of course!)

```plaintext
PW9 51  Baud rate divisor = 0051 for 38400
PWA 00
PWD 08  Update baud rate
```

Now you must change the baud rate to 38400 on the computer port to continue communicating with the board.