Introduction
In this note we describe the planned tests on the TSF prototype. This includes test of the individual board components, interface test and firmware tests. Throughout the document we use the following abbreviations:

FOD: Fast Control / Operation Control / Data Formatter FPGA
ENx : Engine FPGA number x
DIN: Data Input Formatter FPGA

We have produced a test board that plugs into the back of the TSF to provide power and as much I/O as possible. This board allows:

1) Loopback of neighbour data, so that Neighbour I/O can be tested from DataIn chip
2) ECL drive for Fast and Data input signals. It should be possible to drive these signals from either an FPGA on the TSF, eg an engine, via the Mictor and the breakout boards, or via a commercial development board (eg Xilinx spartan board)
3) LVDS receivers for ZPD data. These can be connected as for (2)
4) Headers for remaining I/O pins (can be connected as (2))

This should be useful for initial testing, long term repair work in Manchester, and 'Standalone' bench tests at SLAC.

Basic tests

• All chips hold all static I/Os in a safe condition.
• All chips output slow clock to header.
• All chips test local memory, and toggle an error bit to header (one for each bank on engines). Fast clock.
• FOD outputs Clk4 and fast clock to header.
• FOD drives clock fanouts with Clk60, Clk4, Clk8, and Clk16
• ENx & DIN output Clks from fanouts to header
• FOD drives comm links to others with known pattern, eg 0xFACE each clk 4 packet, aligned to Clk4
• ENx & DIN output data from inward comm link to header, Nibble wide, Clk16 rate (and should check data and toggle an error bit to header).
• ENx & DIN to bounce back data from inward comm link to both outward comm links
• FOD to check data returning from remote FPGAs and set error bits on header (latched or toggle, toggle probably best)
• DIN to send known pattern (simple,known,predictable) to ENx on datalines
• ENx to check pattern arriving on data lines, toggle error on header.
• ENx to send known pattern (as above) to FOD on ZPD & BLT data lines
• FOD to check arriving pattern from engines, and toggle an error bit per engine
• FOD drives (very slow, human speed) patterns to LEDs

All these tests need only a logic analyser on the headers. In addition we want to be able test the external I/O connections. These tests will not be at speed, but will confirm if eg a driver chip has died. These tests should be on an FPGA by FPGA basis, ie each test concerns one chip only.

**External Tests**

These tests are verifying the communication between the TSF board and the outside world. Naturally they are scheduled after the basic board tests.

• FOD to output bitstream on fast data link, and check this is echoed back on fast data in (1 clk60 cycle delay may not be sufficient, off board loopback)
• FOD to send pattern on zpd outward lines (inc clock and frame) and check the correct data is returned via the mictor.

Because this fills up the FOD mictor, only these two tests would be possible (only two spare pins for the error outputs). See attached sheet for proposed pinout info.

The FOD chip also needs to drive loops for the assorted 'extra' I/O - things like the monitoring/slow control lines. These will be checked by connecting up the lines as a big chain of pcb traces interconnected by the test card and special 'bridge headers' in the TSF sockets - the tests will be made by boundary scan alone.

• DIN to send (simple, known, predictable) pattern on neighbour output and check same data is returned on neighbour input.

• Either ECL inward data derived from neighbour output or ECL inward data taken from outputs on the Mictor.

Both are easy to implement in hardware. We prefer the former, in which case the mictor should be used to mirror the output pattern.
There are a number of tests on the TSF-TSFi interface
- Produce known patterns on ZPD and BLT out
- Input data from TSFi and bounce back to TSFi.

**Firmware Test**

The correctness of the vhdl will first be established using a simulator and Monte Carlo. The synthesis process provides timing information for the simulator. We plan to focus on those parts of the firmware where the timing can be critical:

- Data input formatter neighbour data
- Access to the LUT from the engines.
- Best segment finding and time stretching in the engines
- Multiple L1Accepts in a short time

We plan to do the following test

- Read/Write to all memories and check with error count (FOD).
- Neighbour data test, feed TSF with MC data, feed the neighbour data out back in. Verify correctness and timing (DIN)
- Data routing, feed MC data and check routing to engines (DIN) Output the input data to engines.
- Use MC data, output look-ups from LUT with timing information. (DIN + EN)
- Test entire chain using MC data, with and without multiple L1Accepts. Output BLT and ZPD data
Appendix A: Pin-Out for Mictor

cav   n22
dav   n23
strobe n17
d0    n21
d1    n24
d2    n18
d3    n19
d4    n14
d5    n15
d6    n13
d7    n16
d8    n10
d9    n11
d10   n9
d11   n12
d12   n6
d13   n7
d14   n5
d15   n8
d16   n2
d17   n3
d18   n1
d19   n4