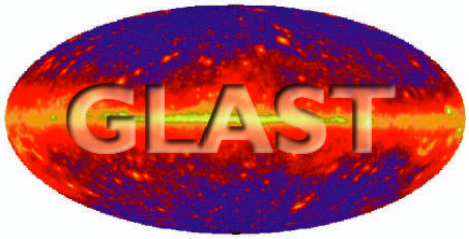

LAT Flight Software

TPG Developer's Guide

Type: Document Template
Version: V1-0-0
Author: J. Swain
Created: 24 September 2003
Updated: 19 November 2003
Printed: 22 September 2004

A user's guide for the Trigger Pattern Generator software

Contents

0 Introduction....2

1 Trigger Input Signals....2

2 Initialisation....3

3 Pattern Creation....3

3.0 ACD FREE VETO....3

3.1 ACD CNO....4

3.2 TEM....4

3.3 Sticky Word....5

3.4 Clearing Pattern Buffers....5

4 Pattern Playback....6

0 Introduction

The Trigger Pattern Generator (TPG) is a combination of hardware and software that provides trigger input signals to the GEM. The TPG simulates one quarter of the LAT, 3 ACD FREE boards and four TEMs, using five VME COMM boards.

1 Trigger Input Signals

Each TEM produces four input signals:

1. Tracker three-in-a-row (TIAR)
2. Calorimeter low energy (CALLO)
3. Calorimeter high energy (CALHI)
4. Busy signal

Each FREE produces nineteen trigger input signals:

1. Eighteen veto signals (VETO00 … VETO17)
2. CNO signal

Each signal can be specified as asserted (on) or de-asserted (off) for each tick of the system clock (a period of fifty nanoseconds for a twenty megahertz clock).

2 Initialisation

```
    address     = malloc(20);

    TPGmsgInit();

    address[ 0] = 0x08000000;
    address[ 4] = 0x08800000;
    address[ 8] = 0x08880000;
    address[12] = 0x0f000000;
    address[16] = 0x0ff00000;

    tgpHandle = TPGnew(address);
```

The code fragment initialises five COMM boards with the addresses shown, creating the five pattern buffers, and starts the MSG printf processor to display error messages.

3 Pattern Creation

Each type of COMM board (FREE, TEM, CNO) has its own pattern manipulation functions.

3.0 ACD FREE VETO

```
    TPGsetFREE (TPG*      tpgHandle,
                unsigned  free_id,   unsigned  veto_id,
                unsigned  start_bit,
                unsigned  pattern,    unsigned pattern_length,
                unsigned  repetions);

    TPGsetFREEv(TPG*      tpgHandle,
                unsigned  free_id,   unsigned  veto_id,
                unsigned  start_bit,
                unsigned* pattern,    unsigned pattern_length,
                unsigned  repetions);
```

With the acceptable ranges on the parameters being

- free_id 0…2
- veto_id 0…17

- startbit 0…32767
- pattern_length 1…32 or 1…32767 for the vector version
- repetitions 1…32767 or 0 meaning “as many as will fit in FIFO”

For example, the following code set the pattern for the VETO2 of FREE0 to 10101010…

```
    TPGsetFREE(tpgHandle, 0, 2, 0, 0x1, 2, 0);
```

3.1 ACD CNO

```
    TPGsetCNO (TPG*      tpgHandle,
               unsigned  free_id,
               unsigned  start_bit,
               unsigned  pattern,    unsigned pattern_length,
               unsigned  repetions);

    TPGsetCNOv(TPG*      tpgHandle,
               unsigned  free_id,
               unsigned  start_bit,
               unsigned* pattern,    unsigned pattern_length,
                unsigned  repetions);
```

With the acceptable ranges on the parameters being

- free_id 0…2
- veto_id 0…17
- startbit 0…32767
- pattern_length 1…32 or 1…32767 for the vector version
- repetitions 1…32767 or 0 meaning “as many as will fit in FIFO”

For example, the following code sets the pattern for CNO of FREE1 to 110110110

```
    TPGsetCNO(tpgHandle, 1, 0, 0x3, 3, 0);
```

3.2 TEM

```
    TPGsetTEM (TPG*      tpgHandle,
               unsigned  tem_id,     temTrg   trg,
               unsigned  start_bit,
               unsigned  pattern,    unsigned pattern_length,
               unsigned  repetions);

    TPGsetCNOv(TPG*      tpgHandle,
               unsigned  tem_id,     temTrg   trg,
               unsigned  start_bit,
               unsigned* pattern,    unsigned pattern_length,
                unsigned  repetions);
```

With the acceptable ranges on the parameters being

- tem_id 0…3
- temTrg TPG_BUSY, TPG_TIAR, TPG_CALHI, TPG_CALLO

- startbit 0…32767
- pattern_length 1…32 or 1…32767 for the vector version
- repetitions 1…32767 or 0 meaning “as many as will fit in FIFO”

For example, the following code sets the patter for the three-in-a row trigger input for TEM1 to 1101011010000000000000…

```
    TPGsetTEM(tpgHandle, 1, TPG_TIAR, 0xb, 5, 2);
```

3.3 Sticky Word

The last word of the playback FIFO sticks on the output, so that TPG software ensures that the last word of each pattern buffer is zero. This means that only 1023 repetitions of a 32-bit pattern will fit into the buffer, even if the last bit of the pattern in 0 (the software doesn’t try to do anything fancy!). Thus, it is quite likely that the 31 LSBs of the final word of the pattern will have to be entered “by hand” as it were. To make life a little easier, there is a global variable called TPG_LASTWORD that gives the start bit of the last word of a pattern. So, to finish off a pattern of all ones for the TEM

```
    TPGsetTEM(tpgHandle, 1, TPG_TIAR, TPG_LASTWORD, 0xfffffffe, 31, 1);
```

To facilitate some forms of debugging the sticky word can be set a channel at a time using for example,

```
    TPGsetTEMsticky(tpgHandle, 1, TPG_TIAR)
```

The line corresponding to the TIAR trigger for TEM1 would remain asserted until the playback FIFO is reloaded and rerun.

3.4 Clearing Pattern Buffers

```
    TPGclear   (TPG* tpgHandle);

    TPGclearBuf(TPG* tpgHandle, unsigned bufferId);
```

With the acceptable ranges on the parameters being

- Buffered 0…4

These functions either set all five pattern buffers to zero, or the pattern buffer indicated by the bufferId.

4 Pattern Playback

```
    clkDelay = malloc(5);

    clkDelay[0] = 0;
    clkDelay[1] = 0;
    clkDelay[2] = 0;
    clkDelay[3] = 0;
    clkDelay[4] = 0;

    TPGstart(tpgHandle, clkdDelay)
```

The code fragment above creates an array of clock delays, which will need to be determined empirically from the clock skews of the five COMM boards, fills the COMM board FIFOs from the pattern buffers and starts the playback.