 GLAST LAT SPECIFICATION	Document # **LAT-DS-00456-D7**	Date Effective 29 April 2002
	Prepared by(s) Daniel Wood	Supersedes None
	Subsystem/Office Calorimeter Subsystem	
Document Title **Calorimeter TEM Simulator Software Design Description**		

Gamma-ray Large Area Space Telescope (GLAST)

Large Area Telescope (LAT)

Calorimeter TEM Simulator Software Design Description

(V0-0-7)

1. OVERVIEW 3

2. COMMANDS 4

COMMAND FORMAT 4
TEM COMMON CONTROLLER COMMANDS 6
TEM CAL CONTROLLER COMMANDS 7
CAL FRONT END ELECTRONICS COMMANDS 8
SIMULATED TEM REGISTERS 9
COMMAND REPLIES 12

3. EVENT DATA 12

EVENT DATA FORMATS 13
CAL DEBUG FORMAT EVENT DATA 15
TEM FORMAT EVENT DATA 18

4. RUNNING THE SIMULATOR SERVER 21

THE UNIX SERVER 21
THE VXWORKS SERVER 21

5. UNIX COMMAND INTERFACE 23

SYMBOLIC NAMES 24
COMMAND DEFINITIONS 25

6. UNIX EVENT DATA INTERFACE 27

1. Overview

The GLAST calorimeter tower electronics module simulator provides a reasonable functional simulation of the operation of the TEM. Much of the detailed TEM hardware functionality is accomplished in software with the simulator. The main limitation of the simulator setup is that only that functionality directly related to the operation of the CAL hardware is currently supported. There are two major goals of the TEM simulator:

- Provide a means of testing the calorimeter front end electronics.
- Provide a basis for GSE software development.

The simulator environment is composed of four major components.

- One or more GSE workstations running Solaris or Linux
- A VME single board computer (SBC) running VxWorks
- A VME front end communications (IOCOM) board installed in the SBC chassis
- A complete or partial set of CAL front end electronics

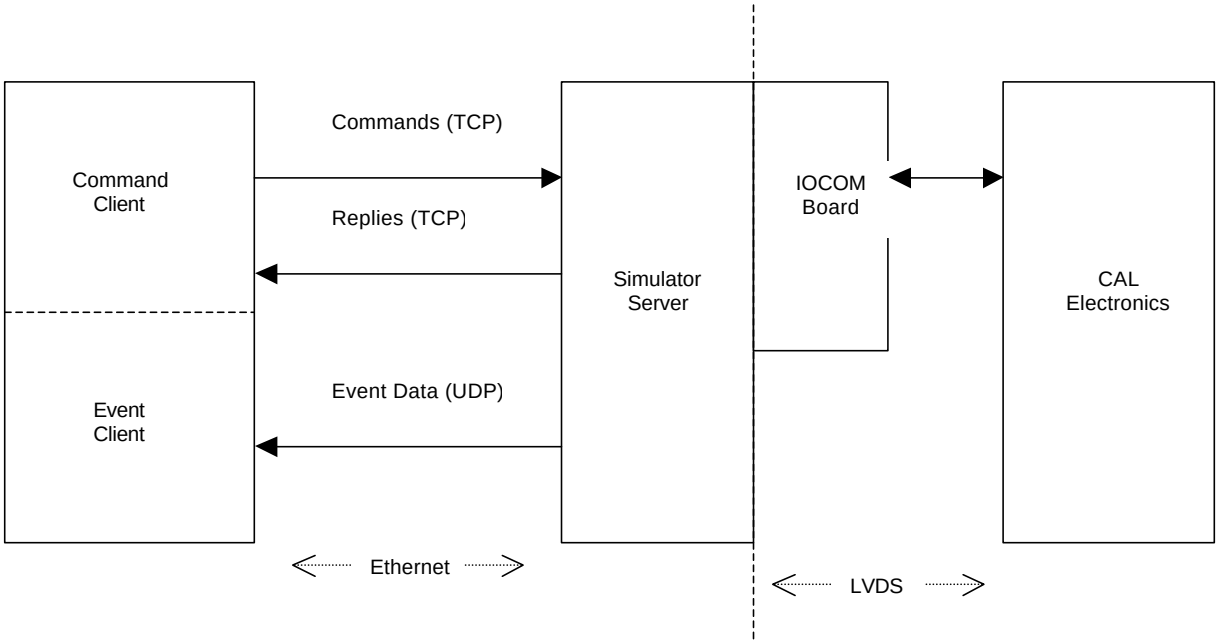
The simulator itself is a VxWorks application which runs on the SBC. Solaris and Linux versions of the simulator will also be available which provide a limited subset of the simulator functionality. The simulator communicates with the GSE workstation across Ethernet using IP protocols. The simulator uses the IOCOM board to communicate with CAL electronics across LVDS serial lines.

The GSE workstation can send commands to the simulator and receive replies back. The intent is that the format of these commands and replies is very close to the command and reply format sent to and from the actual TEM. The simulator will parse the input commands and take appropriate action. In some cases, the commands will specify some action for the TEM. The simulator will mimic as well as possible those actions. In other cases, the commands will specify some action for the CAL front end electronics. The simulator will reformat the command data as necessary and send the bits across the serial link to the CAL hardware. The simulator will send back command replies to the GSE workstation when the command requires such action. The TEM command replies are simulated in some cases, while in other cases, the reply is a reformatted CAL hardware response.

The simulator can also gather event data from the CAL hardware in response to calibration commands. The event data is reformatted to match the actual TEM output format as close as possible. The reformatted event data is sent to the GSE workstation for processing.

The schematic below shows the basic setup components.

Figure 1 - CTS Simulator Components



The IOCOM Board and CAL Electronics boards components are only supported with the VxWorks version running on a VME SBC.

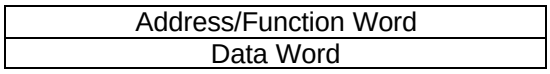
2. Commands

Each command to the TEM simulator is either a register read, a register write, a reset, or a special command. The write and reset commands simply execute on the target. The read commands cause the CTS simulator to produce a command reply which gives the requested data. Special commands, which include the calibrate command, may cause the CTS simulator to generate event data. Each command may be directed towards a simulated TEM component, a single CAL GCRC chip, a single CAL GCFE chip, or broadcast to the GCRC and GCFE chips. The GCRC chips are considered to compose CAL *layers*. Each CAL front end electronics board contains four GCRC chips. These are addressed 0 – 3, with 0 being towards the front of the calorimeter (towards the TKR). Each GCRC chip communicates with 12 GCFE chips which are addressed 0 – 11. Each GCFE chip is responsible for one CAL log end detector and is considered to be in 1 of 12 *columns*. The CAL front end electronics boards are considered to compose *cables*. These are addressed 0 – 3, with 0 being the board perpendicular to the X+ axis. Board 1 faces the Y+ axis, board 2 faces the X- axis, and board 3 faces the Y- axis.

Command Format

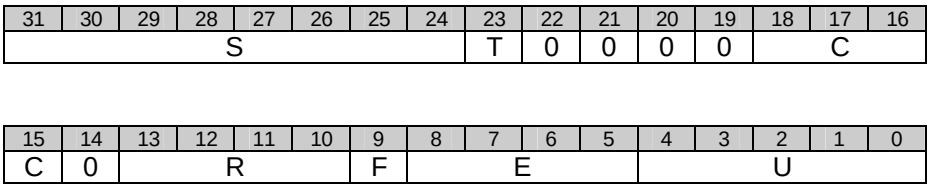
Each CTS command is composed of a 32 bit address/function word followed by a 32 bit data word.

Figure 2 – Command Packet



The address word is formatted as follows:

Figure 3 – Command Address Word



S – Subsystem Address
- 00000000 = TEM Common controller
- 00000010 = TEM CAL controller

T – TEM address flag
- 0 = forward command
- 1 = internal command

C – Cable address
- 0001 = cable 0 (X+ side board)
- 0010 = cable 1 (Y+ side board)
- 0100 = cable 2 (X- side board)
- 1000 = cable 3 (Y- side board)
- 1111 = broadcast

R – GCRC chip address
- 0000 = chip 0
- 0001 = chip 1
- 0010 = chip 2
- 0011 = chip 3
- 1111=broadcast

F – Front end address flag
- 0 = GCRC cmd
- 1 = GCFE cmd

E – GCFE chip address
- 1111 = broadcast

U – Command function word

The CTS software does not support the full decoding of the system address field (S). Essentially, all of the commands are always directed to the TEM on tower #0. The data word is formatted as follows:

Figure 4 – Command Data Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The calorimeter commands never contain more than 16 bits of data, but the extended data field allows a common GLAST format for all of the sub-systems. The 5 function word bits (U) are broken up into a 1 bit read/write flag and a 4 bit function code:

Figure 5 – Command Function Word Bits

4	3	2	1	0
R	F			

R – Read flag
0 = write
1 = read

F – Function code
Indicates specific operation of command.

Each command may be addressed to the simulated TEM common controller, the simulated TEM CAL controller, or output to the actual front end hardware. Most of the command function codes represent writes and reads to and from registers. The function codes for internal TEM accesses and external front end accesses overlap since they can be distinguished by different addresses.

TEM Common Controller Commands

The TEM common controller commands simulate global TEM hardware functionality.

Table 1 – TEM Common Controller Write Functions

Function Code	Description
0	No Operation
4	TEM Reset All
5	TEM Reset Command Counter
10	TEM Reset Event Counter

Table 2 – TEM Common Controller Read Functions

Function Code	Description
11	TEM Command Status Register Read
14	TEM Event Count MSW Register Read
15	TEM Event Count LSW Register Read

TEM CAL Controller Commands

The TEM CAL controller commands simulate TEM calorimeter interface hardware functionality.

Table 3 – TEM CAL Controller Write Functions

Function Code	Description
0	No Operation
1	TEM CAL Layer Enable Register Write
2	TEM CAL TACK Delay Register Write
3	TEM CAL Configuration Register 0 Write
4	TEM CAL Configuration Register 1 Write
5	TEM CAL Configuration Register 2 Write
6	TEM CAL Log Enable Register 0 Write
7	TEM CAL Log Enable Register 1 Write
8	TEM CAL Log Enable Register 2 Write
9	TEM CAL Log Enable Register 3 Write
10	TEM CAL Log Enable Register 4 Write
15	TEM CAL Hardware Reset FE Board

Table 4 TEM CAL Controller Read Functions

Function Code	Description
1	TEM CAL Layer Enable Register Read
2	TEM CAL TACK Delay Register Read
3	TEM CAL Configuration Register 0 Read
4	TEM CAL Configuration Register 1 Read
5	TEM CAL Configuration Register 2 Read
6	TEM CAL Log Enable Register 0 Read
7	TEM CAL Log Enable Register 1 Read
8	TEM CAL Log Enable Register 2 Read
9	TEM CAL Log Enable Register 3 Read
10	TEM CAL Log Enable Register 4 Read
11	TEM CAL Status Register Read

CAL Front End Electronics Commands

The command function codes to access the front end electronics are grouped according to the target chip, GCRC or GCFE:

Table 5 – GCRC Write Functions

Function Code	Description
0	GCRC No Operation
1	GCRC Reset
3	GCRC Calibrate Command
11	GCRC Time Delay Register 1 Write
12	GCRC Time Delay Register 2 Write
13	GCRC Time Delay Register 3 Write
14	GCRC CAL DAC Register Write
15	GCRC Configuration Register Write

Table 6 - GCRC Read Functions

Function Code	Description
0	GCRC Status Register Read
1	GCRC Command Error Register Read
11	GCRC Time Delay Register 1 Read
12	GCRC Time Delay Register 2 Read
13	GCRC Time Delay Register 3 Read
14	GCRC CAL DAC Register Read
15	GCRC Configuration Register Read

Table 7 - GCFE Write Functions

Function Code	Description
8	GCFE Configuration Register 0 Write
9	GCFE Configuration Register 1 Write
10	GCFE FLE DAC Register Write
11	GCFE FHE DAC Register Write
12	GCFE LAC DAC Register Write
13	GCFE ULD DAC Register Write
14	GCFE REF DAC Register Write

Table 8 - GCFE Read Functions

Function Code	Description
0	GCFE Configuration Register 0 Read
1	GCFE Configuration Register 1 Read
2	GCFE FLE DAC Register Read
3	GCFE FHE DAC Register Read
4	GCFE LAC DAC Register Read
5	GCFE ULD DAC Register Read
6	GCFE REF DAC Register Read

Simulated TEM Registers

The CTS software simulates a few of the TEM common controller registers and most of the TEM calorimeter controller registers. In some cases, registers and bits in registers have been added, modified, or removed in order to achieve the desired functionallity. The reference below shows which TEM registers are accessable and the meanings of their bits.

Figure 6 - TEM Common Controller Command Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CCNT														

CCNT – Commands Received Counter

Increments by one for every command packet received across the Ethernet.

Default value = 0x0000.

Figure 7 - TEM Common Controller Event Counter MSW Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECNT_MSW															

ECNT_MSW – Event Counter MSW

The most significant 16 bits of the event counter. Increments by one for every event received across the LVDS.

Default value = 0x0000.

Figure 8 - TEM Common Controller Event Counter LSW Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECNT_LSW															

ECNT_LSW – Event Counter LSW

The least significant 16 bits of the event counter. Increments by one for every event received across the LVDS.

Default value = 0x0000.

Figure 9 - TEM CAL Controller TACK Delay Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	D						

D – Delay Value

The number of clock ticks to delay the TACK command after the calibrate command.

Default value = 0x0000.

Figure 10 - TEM CAL Controller Configuration Register #0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	M		Z	R	E	T	D	C

M – CNO Trigger Mode
- 00 = 1 range readout
- 01 = 1 range readout
- 10 = 4 range readout
- 11 = 1 range readout

Z – Zero Suppression Enable
- 0 = zero suppression disabled
- 1 = zero suppression enabled

R – Read Parity
- 0 = Expect odd parity on LVDS command replies and event data
- 1 = Expect even parity on LVDS command replies and event data

E – Event Data Mode
- 0 = Generate 16 bit log end debug format event data to Ethernet
- 1 = Generate 32 bit TEM format data to Ethernet

T – Trigger Parity
- 0 = Generate odd parity for LVDS trigger prefix bits
- 1 = Generate even parity for LVDS trigger prefix bits

D – Data Parity
- 0 = Generate odd parity for LVDS command data bits
- 1 = Generate even parity for LVDS command data bits

C – Command Parity
- 0 = Generate odd parity for LVDS command address bits
- 1 = Generate even parity for LVDS command address bits

Default value = 0x0020.

Figure 11 - TEM CAL Controller Status Regiser

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	P	T

P – Parity Error
The simulator detected a parity error on an LVDS command reply.

T – Timeout Error
The simulator detected a timeout error on an LVDS command reply.

Default value = 0x0000.

Command Replies

For those commands that instruct the simulator to read a register value, the CTS software will issue a command reply packet. Each CTS reply is composed of a 32 bit address/function word and a 32 bit data/error word.

Figure 12 - Command Reply Packet

Address/Function Word
Data/Error Word

The address/function word is a copy of the corresponding word from the command. The data word is formatted as follows:

Figure 13 - Reply Data Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E

DATA - Command Reply Data

E – Error Summary Flag
0 = No error occurred
1 = Error occurred

The calorimeter replies never contain more than 16 bits of data, but the extended data field allows a common GLAST format for all of the sub-systems. Error reports are signaled by an unused bit in the command reply data word. When the error flag is set, various status registers may be interrogated to determine the exact cause of the condition.

3. Event Data

The CTS event data addressing scheme is the same as for commanding. There are four cables (boards) with four layers each. A layer contains 12 log ends. For the event data, the distinction is

made between a *positive log end* and a *negative log end*. A positive log end is controlled by the CAL electronics board perpendicular to the positive direction of an instrument coordinate axis (for example, the X+ board). A negative log end is defined similarly.

Event Data Formats

The CTS software support two different calorimeter event data formats. An internal 16-bit format is provided for debugging. In this format, each CAL log end is represented by one 16 bit data word. The CTS software also provides a 32-bit TEM format which simulates the actual event data output of the TEM hardware as nearly as possible.

Each CTS CAL debug format event data packet is composed of a 32 bit TEM event data header, a variable number of 16 bit event data words, and possibly, a variable number of 16 bit event error report words.

Figure 14 - CAL Debug Event Data Packet

TEM Event Header Word	
CAL Data Word #0	CAL Data Word #1
CAL Data Word #2	CAL Data Word #3
CAL Data Word #4	CAL Data Word #5
…	
CAL Data Word #n-1	CAL Data Word #n
CAL Error Word #0	CAL Error Word #1
…	
CAL Error Word #m-1	CAL Error Word #m

Each CTS TEM format event data packet is composed of a 32 bit TEM event data header, a 32 bit calorimeter data header, and possibly, a variable number of 32 bit calorimeter event data words.

Figure 15 – TEM Event Data Packet

TEM Event Header Word	
CAL Event Data Header Word	
CAL Event Data Word #0	
CAL Event Data Word #1	
…	
CAL Event Data Word #n	
CAL Error Word #0	CAL Error Word #1
…	
CAL Error Word #m-1	CAL Error Word #m

TEM Event Header

The event header is identical in both event data formats. The Event Header Summary word indicates which subsystems contributed to the event data and the total size of the event data packet.

Figure 16 – TEM Event Header Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A		0	0	0	0	T	C	N							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						0	0	0	0	0	0	0	0	E	0

A – Event Sequence Tag
The lower 2 bits of the 16 bit event trigger counter.

T – Trigger Type Flag
0 – The CAL was readout with a single range trigger command.
1 – The CAL was readout with a four range readout command.

C – Calorimeter Event Mode Flag
0 – The CAL data is in the 16-bit debug format
1 – The CAL data is in the 32-bit TEM format

N – Event Sequence Number
The upper 14 bits of the 16 bit event trigger counter.

E – Error Flag
0 – No error occurred.
1 – An error has occurred.

The CTS software does not fully support the complete event data format. Essentially, every simulator event will indicate that the source of the data is the TEM on tower #0 and that the calorimeter data is present in the event without any other module.

Event Error Reports

Both CTS event data formats will report errors encountered during the event data readout process. Error reports are appended to the normal event data when the Event Header *Error Flag* bit is set. Each error is reported in a 16 bit event error message word.

Figure 17 - CAL Event Data Error Report Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W30	W31	W20	W21	W10	W11	W00	W01	T		0	0	C			

W30 – Layer 3, Wire 0
When set, this bit indicates the error occurred with the layer #3 GCRC chip on data wire #0.

W31 – Layer 3, Wire 1
When set, this bit indicates the error occurred with the layer #3 GCRC chip on data wire #1.

W20 – Layer 2, Wire 0
When set, this bit indicates the error occurred with the layer #2 GCRC chip on data wire #0.

W21 – Layer 2, Wire 1
When set, this bit indicates the error occurred with the layer #2 GCRC chip on data wire #1.

W10 – Layer 1, Wire 0
When set, this bit indicates the error occurred with the layer #1 GCRC chip on data wire #0.

W11 – Layer 1, Wire 1
When set, this bit indicates the error occurred with the layer #1 GCRC chip on data wire #1.

W00 – Layer 0, Wire 0
When set, this bit indicates the error occurred with the layer #0 GCRC chip on data wire #0.

W01 – Layer 0, Wire 1
When set, this bit indicates the error occurred with the layer #0 GCRC chip on data wire #1.

T – Error Type
00 = No error
01 = Start bit timeout
10 = Data parity error

C – Cable Address
1000 = cable 0 (X+ side board)
1001 = cable 1 (Y+ side board)
1010 = cable 2 (X- side board)
1011 = cable 3 (Y- side board)

The error report portion of the event data packet is padded with an extra 16 bit error word in order to make the total event packet size an integral number of 32 bit words. If necessary, the extra error pad word will have the Error Type field set to indicate no actual error.

CAL Debug Format Event Data

In the 16-bit CAL debug event data format, each 16 bit data word following the TEM header represents one gain range readout of a CAL log end detector. The data words are formatted as follows:

Figure 18 - CAL Event Data Log Debug Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A	R		D											

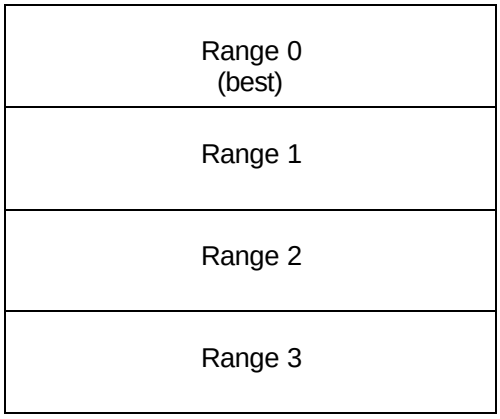
(A) – Log End Accept Bit
0 – Log was not over threshold
1 – Log was over threshold

R – Range Code for Log End
00 – LEX8
01 – LEX1
10 – HEX8
11 – HEX1

D – ADC Value for Log End

For a given gain range block, the log end data words are always all present and they always appear in the same order. Thus, it is possible to obtain the address of the log end by using the data word's relative position in the event data stream. The Trigger Type Flag in the TEM header word will indicate whether the event contains one or four gain ranges. At the outermost level, the data words are ordered by gain range.

Figure 19 - CAL Event Debug Range Addressing



‘

In this figure, the range number referrers to the readout order from the CAL front end. Range 0 is the first range read out and range 3 is the last. The gain range codes in the actual data words will give the electronics gain range for that data. In single range mode, only the first gain range data block will be present. The accept bits only appear in the range 0 data words. At the next level within each gain range block, the CAL debug data words are ordered outermost to innermost by cable address, layer address, then log end address. The cables are ordered as follows:

Figure 20 - CAL Event Debug Cable Addressing

X+ Cable (0)
Y+ Cable (1)
X- Cable (2)
Y- Cable (3)

Within each cable block, the log end data words are organized as follows:

Figure 21 - CAL Event Debug Layer and Log Addressing

Layer 0	Log End 0
	Log End 1
	Log End 2
	Log End 3
	Log End 4
	Log End 5
	Log End 6
	Log End 7
	Log End 8
	Log End 9
	Log End 10
	Log End 11
Layer 1	Log End 0
	Log End 1
	Log End 2
	Log End 3
	Log End 4
	Log End 5
	Log End 6
	Log End 7
	Log End 8
	Log End 9
	Log End 10
	Log End 11
Layer 2	Log End 0
	Log End 1
	Log End 2
	Log End 3
	Log End 4
	Log End 5
	Log End 6
	Log End 7

	Log End 8
	Log End 9
	Log End 10
	Log End 11
Layer 3	Log End 0
	Log End 1
	Log End 2
	Log End 3
	Log End 4
	Log End 5
	Log End 6
	Log End 7
	Log End 8
	Log End 9
	Log End 10
	Log End 11

TEM Format Event Data

In the 32-bit TEM event data format, the TEM header is followed by one 32-bit CAL event header word. After the CAL header word, there are a variable number (0 – 96) 32-bit data words. Each of the CAL event data words corresponds to a log hit, which occurs whenever one of the log end detectors for that log is over threshold. The CAL Event Data Header Word is formatted as follows.

Figure 22 - CAL Event Data Header Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Y3				Y2				X3				X2			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y1				Y0				X1				X0			

Y3 – Layer Count Y3
The number of data words from layer Y3.

Y2 – Layer Count Y2
The number of data words from layer Y2.

X3 – Layer Count X3
The number of data words from layer X3.

X2 – Layer Count X2
The number of data words from layer X2.

Y1 – Layer Count Y1
The number of data words from layer Y1.

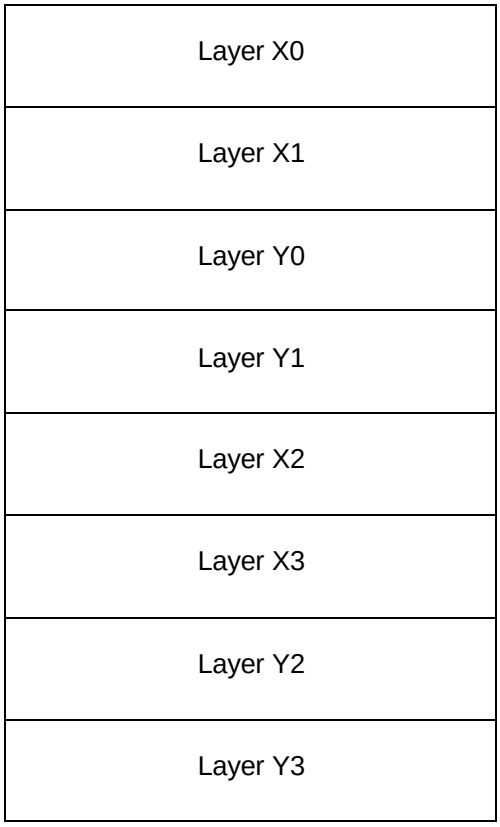
Y0 – Layer Count Y0
The number of data words from layer Y0.

X1 – Layer Count X1
The number of data words from layer X1.

X0 – Layer Count X0
The number of data words from layer X0.

The layer counts in the CAL Event Data Header allow the user to parse the remaining CAL Event Data Words. Each data word corresponds to the readout values from a pair of CAL log end detectors from opposite faces of the calorimeter. Depending on how the CAL front end boards are attached to the CTS hardware, the simulator may have to ignore one of the log end data values. The data words are organized such that the words from the lowest address layer all come together, followed by the next lowest address layer and so on.

Figure 23 - CAL Event TEM Layer Addressing



This allows the user to obtain an implicit layer address for each log data word by using the counts for each layer even though each of the layer blocks shown above is variable length (or perhaps not present). Each log data word is formatted as follows.

Figure 24 - CAL Event Data Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A				RN		DN									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		RP		DP											

A – Log Address

RN – Range Code for Negative Log End Data
- 00 – LEX8
- 01 – LEX1
- 10 – HEX8
- 11 – HEX1

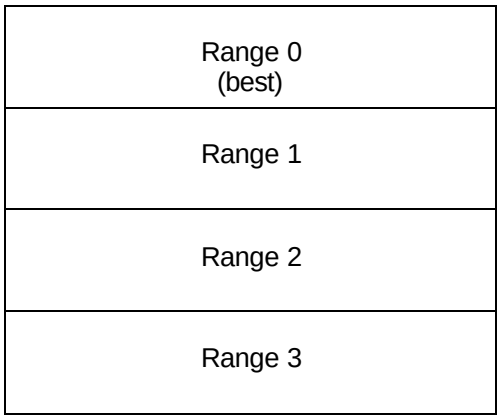
DN – ADC Value for Negative Log End Data

RP – Range Code for Positive Log End Data
- 00 – LEX8
- 01 – LEX1
- 10 – HEX8
- 11 – HEX1

DP – ADC Value for Positive Log End Data

In the TEM format, when the event data is contains four gain ranges, the ranges still need to be addressed implicitly. The organization is similar to the debug format:

Figure 25 - CAL Event TEM Range Addressing



If the TEM simulator is configured to produce only one range for the readout data, then only the range 0 data block will be present. As with the debug event data format, the actual gain range codes for the data values are embedded in the data itself. The range blocks simply denote the conversion order chosen by the GCFE chips.

4. Running the Simulator Server

The CTS simulator server is run as an application named *ctssim*. It is available as a Solaris, Linux, and VxWorks executable, but only the VxWorks version supports communications with the CAL electronics through the IOCOM LVDS interface.

The UNIX Server

The basic syntax of the UNIX server startup is shown below:

```
ctssim  <evt_addr>,   <cmd_port>,  <evt_port>
```

evt_addr – The IP address of the machine to receive the UDP event data packets. It may either be a registered host name string or a numerical dot address string.
cmd_port - The TCP port number to listen on for connections from the command client machine.
evt_port - The UDP port number to send the event data packet to.

To start the server, log in to the machine which will run the application. In a terminal window, type the following lines:

Example 1 - Running the UNIX Simulator Server

```
machine% cmx login
machine% ctssim localhost 7000 7001
```

This example shows the server sending event data to the same machine on port 7001. The server listens for command connections on port 7000. The last three parameters are ignored.

The VxWorks Server

The basic syntax of the VxWorks server startup is shown below:

```
ctssim  <evt_addr>,   <cmd_port>,  <evt_port>,  <iocom_addr>,
<tack_delay>, <intr_level>,  <intr_vec>
```

evt_addr – The IP address of the machine to receive the UDP event data packets. It may either be a registered host name string or a numerical dot address string.
cmd_port - The TCP port number to listen on for connections from the command client machine.
evt_port - The UDP port number to send the event data packet to.
iocom_addr – The VME bus address of IOCOM board.
tack_delay – The base delay in 20 MHz ticks from the start of the calibrate command to the start of the TACK command. Longer delays are possible by setting the TEM CAL controller TACK delay register.
intr_level - The VME bus interrupt level for the IOCOM board (2 – 5).
intr_vec – The interrupt vector number for the IOCOM board (0 – 255).

The server must be started from a Sun/Solaris machine which has the Tornado/VxWorks software installed. To start the server, log in to the Solaris machine, and in a terminal window type the following lines:

Example 2 – VxWorks CTS Server Load

```
machine% cmx login
machine% ctssim_start target_machine target_type

ld < /glast/flight/BFS/binary/MV2X/V0-1-3/mv2303/libmv2x_extra.o
Loading /glast/flight/BFS/binary/MV2X/V0-1-3/mv2303/libmv2x_extra.o
|
value = 433272 = 0x69c78
ld < /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/libcts_cmd.o
Loading /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/libcts_cmd.o
value = 417408 = 0x65e80
ld < /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/libcts_evt.o
Loading /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/libcts_evt.o |
value = 418792 = 0x663e8
ld < /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/ctssim.exe
Loading /glast/flight/CAL/binary/CTS/V0-0-7/mv2303/ctssim.exe |
value = 426328 = 0x68158
```

In this example, *target_machine* is the host name or numerical dot address of the VxWorks SBC. The *target_type* parameter gives the specific board type the server is running on and should be either `mv2303` or `mv2304`. The output from the shell script will show the versions of the modules being loaded. After loading, the terminal window should display the Tornado WindSh banner and prompt. At the prompt, the user may invoke the CTS server manually by spawning the startup function *ctssim()*. It may be more useful to place the startup function call inside of a personal script and then source that script.

Example 3 – VxWorks CTS Server User Startup Script

```
-> < /home/user/my_cts_server_script
sp ctssim, "goose", 7000, 7001, 0x08000000, 65, 2, 0xf0
task spawned: id = 1e06ca0, name = s1u0
value = 31485088 = 0x1e06ca0
->
```

At this point, the server is running. To stop the server from running, issue the *exit* command from the WindSh prompt.

Example 4 - CTS Server Shutdown

```
-> exit
ctssim_kill
ctssim: exiting
value = 0 = 0x0
machine%
```

This will stop both the server from running on the target and the WindSh application from running on the Solaris machine.

The CTS software uses a strict assignment scheme for the IOCOM input and output channels. The following tables show the signal assignments.

Table 9 - IOCOM Output Signals

IOCOM Output Channel	Description
Out0	System clock
Out1	Command
Out2	Hardware Reset

Table 10 - IOCOM Input Signals

IOCOM Output Channel	Description
In0	Layer 0 GCRC data 0
In1	Layer 0 GCRC data 1
In2	Layer 1 GCRC data 0
In3	Layer 1 GCRC data 1
In4	Layer 2 GCRC data 0
In5	Layer 2 GCRC data 1
In6	Layer 3 GCRC data 0
In7	Layer 3 GCRC data 1

Each IOCOM board is thus able to support one CAL electronics cable and front end electronics board.

5. UNIX Command Interface

The *ctscmd* application provides a simple command line interface for generating commands for the TEM simulator and displaying the reply values from the read commands. The *ctscmd* client is a command line application. It resides as a CMX constituent in project *CAL*, package *PEM*. The *ctscmd* client will run under both Solaris and Linux.

To start the command client, log in to the machine which will run the application. In a terminal window, type the following lines:

Example 5 - Running the CTS Command Client

```
Machine% cmx login
Machine% ctscmd
CTS>
```

The command client is now ready to accept commands. If the CTS server is not up and running, then it is still possible to issue dummy commands just to explore the syntax and check the command packet formatting. In this test mode, the command replies always show a value of '0' for the returned data. Otherwise if the server is running, the *connect* command must be the first issued in order to actually send commands to the server machine.

Symbolic Names

Since most of the available commands read and write target registers, the *ctscmd* application allows users to specify register name symbolically. The following register names are recognized.

Table 11 - Symbolic Register Names

Register Name	Description
tcom_ecnt_msw	TEM common controller event count MSW register
tcom_ecnt_lsw	TEM common controller event count LSW register
tcom_cmd_stat	TEM common controller command status register
tcal_lay_en	TEM CAL controller layer enable register
tcal_tack	TEM CAL controller TACK delay register
tcal_config_0	TEM CAL controller configuration register #0
tcal_config_1	TEM CAL controller configuration register #1
tcal_config_2	TEM CAL controller configuration register #2
tcal_log_en_0	TEM CAL controller log enable register #0
tcal_log_en_1	TEM CAL controller log enable register #1
tcal_log_en_2	TEM CAL controller log enable register #2
tcal_log_en_3	TEM CAL controller log enable register #3
tcal_log_en_4	TEM CAL controller log enable register #4
tcal_stat	TEM CAL controller status register
gcrc_delay_1	GCRC time delay register #1
gcrc_delay_2	GCRC time delay register #2
gcrc_delay_3	GCRC time delay register #3
gcrc_cal	GCRC calibration DAC register
gcrc_config	GCRC configuration register
gcrc_stat	GCRC status register
gcrc_err	GCRC command error register
gcfe_config_0	GCFE configuration register #0
gcfe_config_1	GCFE configuration register #1
gcfe_fle	GCFE FLE DAC register
gcfe_fhe	GCFE FHE FAC register
gcfe_lac	GCFE log accept DAC register
gcfe_uld	GCFE ULD DAC register
gcfe_ref	GCFE reference DAC register

For some commands, it is more convenient to indicate the target by DAC name. The following symbolic DAC name are recognized.

Table 12 - Symbolic DAC Names

DAC Name	Description
lac	GCFE log accept DAC
fle	GCFE fast low energy trigger DAC
fhe	GCFE fast high energy trigger DAC
uld	GCFE upper limit DAC
ref	GCFE reference DAC
cal	GCRC calibration DAC

Command Definitions

The following commands are recognized by the *ctscmd* application.

Connect

Syntax

```
connect  <sim_ip>  <sim_port>
```

Description

Connects the ctscmd application to the TEM simulator. The *sim_ip* parameter gives the IP address at which the simulator is listening for command connections. The IP address may be given either as a recognized host name or as a numerical dot address. The *sim_port* number should match the port number given at the startup of the simulator for the command TCP port number.

Disconnect

Syntax

```
disconnect
```

Description

Disconnects the *ctscmd* application from the TEM simulator.

Poke

Syntax

```
poke  <reg_name>  <reg_value>  [--cable=<n>]  [--layer=<n>]
[--log=<n>]
```

Description

Writes *reg_value* into the register identified by *reg_name*. The *reg_value* parameter should be given as a hex number of no more than four digits. The optional *cable*, *layer*, and *log* qualifiers allow the user to address a register in a particular front end chip. If the

address qualifiers are not given, or if they are partially specified, then the command will be broadcast as far as possible.

Peek

Syntax

```
peek  <reg_name>  [--cable=<n>]  [--layer=<n>]  [--log=<n>]
```

Description

Reads a value from the register specified by *reg_name*. The optional *cable*, *layer*, and *log* qualifiers allow the user to address a register in a particular front end chip. These address qualifiers are not optional when the target register is in the front end electronics.

Reset

Syntax

```
reset  tem  [ccnt | ecnt | all]
reset  cal  [--cable=<n>]
reset  gcrc  [--cable=<n>]  [--layer=<n>]
```

Description

Resets the indicated module.
If the module is *tem,* and no additional parameter is given, then all of the simulated TEM registers are reset. Otherwise, the *ccnt* parameter specifies that the command counter should be reset, and the *ecnt* parameter specifies that the event counter should be reset. If the module is *cal*, then a hardware reset signal is sent to the front end electronics. The optional qualifier *cable* allows the user to address a particular front end board. If the address qualifier is not given, then the reset signal will be delivered to all front end boards. If the module is *gcrc*, a soft reset command is sent to one or more GCRC chips. The optional qualifiers *cable* and *layer* allow the user to address particular GRCR chips. If the address qualifiers are not given, then the reset command will be broadcast as far as specified.

Dac

Syntax

```
dac  <dac_name>  <dac_value>  [--cable=<n>]  [--layer=<n>]
[--log=<n>]
```

Description

Sets the value of the DAC indicated by *dac_name* to *dac_value*. *dac_value* is given as a decimal number. The optional *cable*, *layer*, and *log* qualifiers allow the user to address a register in a particular front end chip. If the address qualifiers are not given, or if they are partially specified, then the command will be broadcast as far as possible.

Calibrate

Syntax

```
calibrate  [num_trig]  [--cable=<n>]
```

Description

Sends the calibrate command to the GCRC front end chips for the VxWorks platform. For UNIX platforms, this command sends an event packet of test pattern data. The optional *num_trig* parameter allows the users to specify the number of times to issue the command. The default is to run one time. The optional cable address qualifier allows the user to target a specific front end electronics board.

Help

Syntax

```
help
```

Description

Prints a summary of the commands.

Exit

Syntax

```
exit
```

Description

Quits the *ctscmd* application.

6. UNIX Event Data Interface

The *ctsdump* application is a UDP/IP client which provides a simple event data interface to the CTS server. It resides as a CMX constituent in project *CAL*, package *CTS*. The *ctsdump* client will run under both Solaris and Linux.

To start the event client, log in to the machine which will run the application. In a terminal window, type the following lines:

Example 6 - Running the CTS Event Client

```
machine% cmx login
machine% ctsdump –f my_data_dir 7001
Listening on 0.0.0.0:7001
Writing to file 1010086653.tsd
```

In this example, 7001 is the UDP port number to which the CTS server will send event data. This should match the value given as a startup parameter to the server. Also, the server should have been started with the parameter giving the IP address of the machine running the *pemdump* application.

The instructions above start the *ctsdump* application in record mode. The *–f* flag indicates that the application should record the event data to file. The application will always open a new event data file in the directory *my_data_dir*. The *ctsdump* application always automatically generates the event data file name. It is of the format: `<unix_time>.tsd`, where *unix_time* is the current UNIX time. The UNIX time is the number of seconds since Jan 1, 1970 00:00:00 GMT. This convention ensures that the data file name are unique and that they are always ordered by time of capture. The application will print the file name it has chosen at startup.

The *ctsdump* application supports an optional verbose mode of operation:

```
machine% ctsdump –f my_data_dir -v 7001
Listening on 0.0.0.0:7001
Writing to file 1010086709.tsd
```

The *–v* flag indicates that the application should display the event data contents as they are received. The output of the *ctsdump* application differs depending on the event data mode the simulator is currently running. The example below shows the output when the simulator is running in the CAL log end debug mode.

Example 7 - CTS Debug Format Event Display

```
EVENT 1
------------------------------------------------------------
                        RANGE 0
------------------------------------------------------------
      0    1    2    3    4    5    6    7    8    9    10   11
------------------------------------------------------------
CABLE 0
------------------------------------------------------------
L0: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L1: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L2: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L3: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
------------------------------------------------------------
CABLE 1
------------------------------------------------------------
L0: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L1: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L2: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L3: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
------------------------------------------------------------
CABLE 2
------------------------------------------------------------
L0: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L1: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L2: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L3: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
------------------------------------------------------------
CABLE 3
```

```
------------------------------------------------------------
L0: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L1: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L2: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
L3: 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa 4aaa
```

The first line gives the event sequence number. In this example, the simulator is running in one range mode, so the banner shows that the display is for the range 0 block. The second line of the banner gives the log end addresses. For each of the four cables and for each of the four layers per cable, the 16-bit log end data word is displayed.

The output of the ctsdump application is significantly different when the simulator is running in TEM output mode:

Example 8 - CTS TEM Format Event Display

```
EVENT 2
------------------------------------------------------------------
     X3     X2     X1     X0     Y3     Y2     Y1     Y0
------------------------------------------------------------------
     12     12     12     12     12     12     12     12

------------------------------------------------------------------
RANGE 0
------------------------------------------------------------------
Layer 0X  (R0):
------------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 0Y  (R0):
------------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
```

```
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 1X  (R0):
---------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 1Y  (R0):
---------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 2X  (R0):
---------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 2Y  (R0):
```

```
-----------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 3X  (R0):
-----------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)

Layer 3Y  (R0):
-----------------------------------------------------------------
 0 -  2730 (0)     2730 (0)
 1 -  2730 (0)     2730 (0)
 2 -  2730 (0)     2730 (0)
 3 -  2730 (0)     2730 (0)
 4 -  2730 (0)     2730 (0)
 5 -  2730 (0)     2730 (0)
 6 -  2730 (0)     2730 (0)
 7 -  2730 (0)     2730 (0)
 8 -  2730 (0)     2730 (0)
 9 -  2730 (0)     2730 (0)
10 -  2730 (0)     2730 (0)
11 -  2730 (0)     2730 (0)
```

The first line again gives the event sequence number. The next two lines show the accept counts for each of the 8 paired CAL layers. In this example, the simulator is running in one range mode, so the banner shows that the display is for the range 0 block. For each of the 8 paired CAL layers, the accepted log end pairs are displayed. The log address is shown at the far left. The left column shows the positive log end values, and the right column shows the negative log end

values. The first number in each column is the ADC value for that log end displayed in base decimal. The second number in parenthesis is the gain range code value for that log end.

A third possible output mode for the *ctsdump* application is to write the data to a UNIX pipe file on the same machine as which it is running. This is achieved as follows:

```
machine% mkfifo /tmp/ctsdump-pipe
machine% ctsdump –p /tmp/ctsdump-pipe 7001
Listening on 0.0.0.0:7001
Writing to pipe /tmp/ctsdump-pipe
```

The pipe must already have been created before the application is started in this mode. The expected name of the pipe in the example is `/tmp/ctsdump-pipe`. The pipe output mode may be useful for running other GSE applications which are designed for accessing data from a file rather than from an IP network. The verbose and file modes also work when the pipe output option is specified.