GeBGO INTERFACE BOARD

July 28, 1994

1.0 OVERVIEW OF GeBGO BOARD 3
1.1 Description of the GeBGO Board 3
1.2 Block Diagram of the GeBGO Board 3
1.3 Subsections of the GeBGO Board 5
1.4 Memory Map 7

2.0 TEST PROCEDURE 12
2.1 VXI Interface Test 12
2.1.1 Verifying "resman" mapping 12
2.1.2 Testing Internal VXI Registers 15
2.2 Dual Port Ram Test 17
2.3 Scaler Chip Test 18
2.4 SIO and DAC Test 18
2.6 Data Flow Test 23

APPENDIX A: ABEL PROGRAM FOR 22V10 24

APPENDIX B: TEST SOFTWARE 26

HISTORY

Date	Who	What
	John Weizeorick	Original
4-19-94	John Weizeorick	Modified Memory map and other sections to show that SPARE1_A and SPARE1_B have been deleted and DAC_A3 and DAC_B3 have been added.
7-28-94	John Weizeorick	Modified Memory map to show TOKEN_IN added to VXI Status Register.

1.0 OVERVIEW OF GeBGO BOARD

This section gives a brief overview of the portion of the GeBGO BOARD designed and developed by Argonne National Laboratory's Electronics Department, to be used by LBL in the Gamma Sphere Project.

1.1 Description of the GeBGO Board

The GeBGO Board consist of hardware designed by both LBL and ANL's Electronics Department.

LBL is designing the analog portion of the GeBGO Board which interface with the GeBGO detectors. LBL will provide digital data out of ADC, which will interface with the digital circuity designed by the ANL Electronics Department.

There are two main functions that the digital circuitry designed by ANL is to provide. First it provides a VXI interface which allows a VXI Resource Manager to setup, test, and monitor the GeBGO Boards within its crate. Secondly, the digital circuitry controls the data flow from the ADC's into the Local Readout Module.

This document will describe in detail the digital electronics designed by ANL.

1.2 Block Diagram of the GeBGO Board

On the following page is a block diagram of the GeBGO Board. Section 1.3 describes the function of each section shown in the block diagram.

Block diagram goes on this page

1.3 Subsections of the GeBGO Board

The ANL portion of the GeBGO Board consist of the following subsections listed below.

- VXI INTERFACE FPGA
- FASTOUT DATA CONTROLLER FPGA
- SIO FPGA
- SCALER FPGA
- DUAL PORT RAM (DPR)
- PIPELINE REGISTERS
- VXI LOCAL BUS INTERFACE
- MISCELLANEOUS CIRCUITRY
- TEST CIRCUITY

The VXI Interface FPGA allows a VXI Resource Manager to setup, monitor and control everything on the GeBGO VXI INTERFACE BOARD. The VXI Interface FPGA is a VXI Register Based Interface with Dynamic Configuration. For more details on the VXI INTERFACE FPGA see its specification.

The FASTOUT DATA CONTROLLER FPGA monitors when an event occurs on one or both of the GeBGO Modules, and controls the data flow from the GeBGO VXI INTERFACE BOARD into the LOCAL READOUT BOARD. Each GeBGO channel Phas eight 16 bit words to be transferred from the GeBGO VXI INTERFACE BOARD to the LOCAL READOUT BOARD as four 32 bit words. The 32 bit words are transferred to the LOCAL READOUT BOARD over the VXI Local bus with National Semiconductor DS3890 Octal Trapezoidal Drivers. For more details on the FASTOUT DATA CONTROLLER FPGA see its specification.

The SIO FPGA provides an interface between Serial devices and the VXI Interface. The SIO FPGA consists of three channels of 16 bit serial I/O. Two channels have a serial input, a serial output, and a clock signal generated by the FPGA. The other channel contains only a serial output line and a clock signal. The two serial channels with a serial input and output are used to provide off-board serial communications to the GeBGO Detector Modules A and B. The channel with only a serial output is used for on-board communication to serial DACs, which control the various BIAS and Amplitude levels for Module A and B. For more details on the SIO FPGA see its specification.

There are two SCALER FPGAs on the GeBGO Board. Each SCALER FPGA consist of twelve 16 bit scalers. Each of the twelve scaler channels has its own gate (g0 - g11) and clock (c0 - c11) input. Each scaler channel is incremented on the rising edge of its clock input when its gate is enabled (high). For more details on the
SCALER FPGA see its specification.

The Dual Port Ram serves a two functions which depend on the mode of operation. In the normal operating mode, when an event occurs, data comes form the ADCs and flows into both, the Pipeline Registers, and into the Dual Port Ram. In this mode the VXI Resource Manager can sample the raw data before it gets to the VME Memory Cards. In the test mode, when an event occurs, data comes form the Dual Port Ram and flows into the Pipeline Registers. In the test mode the Resource Manager can write test data into the Dual Port Ram and then generate an event. This provides a method of testing the data flow through the system and into the memory cards.

The Pipeline Registers store eight 16 bit words for each of the two modules. The data comes from the ADCs or the Dual Port Ram and goes into the Pipeline Registers as 16 bit words. When a token is passed to a GeBGO Board any data in the Pipeline Registers is transferred over the VXI Local Bus to the Local Event Builder. The data is clocked out of the Pipeline Registers as four 32 bit words for each module.

The VXI Local Bus Interface consist of bus drivers and receivers used for transferring data over the VXI Local Bus. Four bus drivers are used to transfer the 32 bit words from the Pipeline Registers. Another bus driver is used to drive the TOKEN_OUT, STROBE, and BSY_RDY signals. A bus receiver chip is used to read the TOKEN_IN signal.

The Miscellaneous Circuitry on the GeBGO Board consist of glue logic surrounding the FPGAs and the VXI Interface. The main functions performed and controlled by the Miscellaneous Circuitry are Reset, Done, SYSFAIL, DTACK, BERR. Reset is used by the FPGAs to initiate their configuration. On reset each FPGA drives its CCLK signal and loads its configuration from its serial PROM. When an FPGA is finish loading its configuration it releases the Done signal. When all FPGAs are configured the Done signal should go high. The SYSFAIL signal is an open collector signal on the VXI bus. Any board may pull this line low to indicate that it has failed. The GeBGO board will hold this signal low if the Done signal does not go high or if the VXI-INF FPGA holds the signal low. DTACK is a handshaking signal used by the VXI bus. It is controlled by the VXI-INF chip. BERR is also controlled by the VXI-INF and notifies the VXI Master when there has been a bus error.

The TEST CIRCUITRY is not intended to be used on the final GeBGO, but provides a method for software routines to test most of the circuitry.

1.4 Memory Map

The Ge/BGO modules in the Gammasphere have 512 bytes of address, in A16 address space, per module. The absolute VXI address will be determined dynamically by the Resource Manager. The base addresses of these registers are defined as follows:

A16 ADDRESSING FORMAT
Base Address

11LL LLLH	HH00 0000

L + H = Logical address
L = Bits which are configured during Dynamic Configuration
H = Hard wired (common decode bits for 8 devices/module)

The register definition per Ge/BGO module is:

OFFSET FROM BASE	READ FUNCTION	INITIAL VALUE
00H	ID REGISTER	1111 1111 0000 0000
02H	DEVICE TYPE REGISTER	0000 0000 0000 0000
04H	STATUS REGISTER	0M00 0000 0000 110T M = MODID BIT T = TOKEN_IN BIT
06H	OFFSET REGISTER	0000 0000 0000 1000

OFFSET FROM BASE	WRITE FUNCTION	INITIAL VALUE
00H	LOGICAL ADDRESS REGISTER	xxxx xxxx PPPP Pxxx P = Programmable Bits
02H		
04H		
06H		

Logical Address - used during dynamic configuration to set the Logical Address.
ID Reg - A16 only, For customized device = F00h.
Device Type Reg - 0 bytes of memory, model code = 0.
Status Reg - No A24/A32, Modid active, Ready Passed.
Control Reg - A24/A32 Disabled, Sysfail active, Reset state Off.
Offset Reg - 8 devices (512 bytes) sharing address decoding.

Device	Relative Address	Use	Function	Dev 0 Offset Address	Dev Offset Address
0	000...0ff	VXI Configuration Registers	Read ID; Write Logical Address Read Device Type Read Status; Write Control Read/Write Offset Register	000h 002h 004h 006h	00h 02h 04h 06h
0	010...01f	GeBGO DATA "A"	Read/Write Ram Ge High Resolution Energy Hit Pattern Delta M Delta T Ge Low Resolution Energy Ge Time BGO Energy BGO Time	 010h 012h 014h 016h 018h 01ah 01ch 01eh	 10h 12h 14h 16h 18h 1ah 1ch 1eh
0	020...02f	Miscellaneous "A"	Read Ram; Write Ram & Device Initiate 1uSec Pulse on "Soft A" Write State of "ADC_MEM" Ctrl Sig 3 Bit Latch "A" 0 - Ge_A_Range_0 1 - Ge_A_Range_1 2 - Not Used 3 - Det_A_Off ID Register "A"	 020h 022h 024h 026h	 20h 22h 24h 26h
0	030...03f	Scratch Memory	Read/Write Ram		
1	040...04f	VXI Configuration Registers	Points to same Registers as Device 0		00h
1	050...05f	BGO Discriminators "A"	Read Ram; Write Ram & SIO "BD", Ram, & DAC_A1 Discriminator 1A Bias Discriminator 2A Bias Discriminator 3A Bias Discriminator 4A Bias Discriminator 5A Bias Discriminator 6A Bias Discriminator 7A Bias Discriminator 8A Bias	 050h 052h 054h 056h 058h 05ah 05ch 05eh	 10h 12h 14h 16h 18h 1ah 1ch 1eh
1	060...06f	Bias & Pulser Amplitude "A"	Read Ram; Write Ram & SIO "BD", Ram, & DAC_A2 Ge CFD Bias "A" Ge Pile-up Bias "A" BGO CFD Bias "A" BGO Pulser Amplitude "A"	 060h 062h 064h 066h	 20h 22h 24h 26h
1	070...07f	Spare DAC Interface for channel A	Read Ram; Write Ram & SIO "BD", RAM, & DAC_A3	070h	30h

Device	Relative Address	Use	Function	Dev 0 Offset Address	Dev Offset Address
2	080...08f	VXI Configuration Registers	Points to same Registers as Device 0		00h
2	090...09f	Detector Module Serial Output "A"	Read Ram; Write Ram & SIO "A"		
			Channel 1 "A"	090h	10h
			Channel 2 "A"	092h	12h
			Channel 3 "A"	094h	14h
			Channel 4 "A"	096h	16h
			Channel 5 "A"	098h	18h
			Channel 6 "A"	09ah	1ah
			Channel 7 "A"	09ch	1ch
			Channel 8 "A"	09eh	1eh
2	0a0...0af	Detector Module Serial Output "A"	Read Ram; Write Ram & SIO "A"		
			Channel 9 "A"	0a0h	20h
			Channel 10 "A"	0a2h	22h
			Channel 11 "A"	0a4h	24h
			Channel 12 "A"	0a6h	26h
			Channel 13 "A"	0a8h	28h
			Channel 14 "A"	0aah	2ah
			Channel 15 "A"	0ach	2ch
			Channel 16 "A"	0aeh	2eh
2	0b0...0bf	Detector Module Serial Input "A"	Read SIO "A"		
			Read Module "A"	0b0h	30h
3	0c0...0cf	VXI Configuration Registers	Points to same Registers as Device 0		00h
3	0d0...0df	Read Scalers "A"	Read Scalers A-0 to A-7		
			Scaler "A" 0	0d0h	10h
			Scaler "A" 1	0d2h	12h
			Scaler "A" 2	0d4h	14h
			Scaler "A" 3	0d6h	16h
			Scaler "A" 4	0d8h	18h
			Scaler "A" 5	0dah	1ah
			Scaler "A" 6	0dch	1ch
			Scaler "A" 7	0deh	1eh
3	0e0...0ef	Read Scalers "A"	Read Scalers A-8 to A-11		
			Scaler "A" 8	0e0h	20h
			Scaler "A" 9	0e2h	22h
			Scaler "A" 10	0e4h	24h
			Scaler "A" 11	0e6h	26h
3	0f0...0ff	Scratch Memory	Read/Write Ram		

Device	Relative Address	Use	Function	Dev 0 Offset Address	Dev Offset Address
4	100...1ff	VXI Configuration Registers	Points to same Registers as Device 0		00h
4	110...11f	GeBGO DATA "B"	Read/Write Ram		
			Ge High Resolution Energy	110h	10h
			Hit Pattern	112h	12h
			Delta M	114h	14h
			Delta T	116h	16h
			Ge Low Resolution Energy	118h	18h
			Ge Time	11ah	1ah
			BGO Energy	11ch	1ch
			BGO Time	11eh	1eh
4	120...12f	Miscellaneous "B"	Read Ram; Write Ram & Device		
			Initiate 1uSec Pulse on "Soft B"	120h	20h
			Write State of "ADC_MEM" Ctrl Sig	122h	22h
			3 Bit Latch "B"	124h	24h
			0 - Ge_B_Range_0		
			1 - Ge_B_Range_1		
			2 - Not Used		
			3 - Det_B_Off		
			ID Register "B"	126h	26h
4	130...13f	Scratch Memory	Read/Write Ram		
5	140...14f	VXI Configuration Registers	Points to same Registers as Device 0		00h
5	150...15f	BGO Discriminators "B"	Read Ram; Write Ram & SIO "BD", Ram, & DAC B1		
			Discriminator 1B Bias	150h	10h
			Discriminator 2B Bias	152h	12h
			Discriminator 3B Bias	154h	14h
			Discriminator 4B Bias	156h	16h
			Discriminator 5B Bias	158h	18h
			Discriminator 6B Bias	15ah	1ah
			Discriminator 7B Bias	15ch	1ch
			Discriminator 8B Bias	15eh	1eh
5	160...16f	Bias & Pulser Amplitude "B"	Read Ram; Write Ram & SIO "BD", Ram, & DAC B2		
			Ge CFD Bias "B"	160h	20h
			Ge Pile-up Bias "B"	162h	22h
			BGO CFD Bias "B"	164h	24h
			BGO Pulser Amplitude "B"	166h	26h
5	170...17f	Spare DAC Interface for channel A	Read Ram; Write Ram & SIO "BD", RAM, & DAC_A3	170h	30h

Device	Relative Address	Use	Function	Dev 0 Offset Address	Dev Offset Address
6	180...18f	VXI Configuration Registers	Points to same Registers as Device 0		00h
6	190...19f	Detector Module Serial Output "B"	Read Ram; Write Ram & SIO "B"		
			Channel 1 "B"	190h	10h
			Channel 2 "B"	192h	12h
			Channel 3 "B"	194h	14h
			Channel 4 "B"	196h	16h
			Channel 5 "B"	198h	18h
			Channel 6 "B"	19ah	1ah
			Channel 7 "B"	19ch	1ch
			Channel 8 "B"	19eh	1eh
6	1a0...1af	Detector Module Serial Output "B"	Read Ram; Write Ram & SIO "B"		
			Channel 9 "B"	1a0h	20h
			Channel 10 "B"	1a2h	22h
			Channel 11 "B"	1a4h	24h
			Channel 12 "B"	1a6h	26h
			Channel 13 "B"	1a8h	28h
			Channel 14 "B"	1aah	2ah
			Channel 15 "B"	1ach	2ch
			Channel 16 "B"	1aeh	2eh
6	1b0...1bf	Detector Module Serial Input "B"	Read SIO "B"		
			Read Module "B"	0b0h	030h
7	1c0...1cf	VXI Configuration Registers	Points to same Registers as Device 0		00h
7	1d0...1df	Read Scalers "B"	Read Scalers B-0 to B-7		
			Scaler "B" 0	1d0h	10h
			Scaler "B" 1	1d2h	12h
			Scaler "B" 2	1d4h	14h
			Scaler "B" 3	1d6h	16h
			Scaler "B" 4	1d8h	18h
			Scaler "B" 5	1dah	1ah
			Scaler "B" 6	1dch	1ch
			Scaler "B" 7	1deh	1eh
7	1e0...1ef	Read Scalers "B"	Read Scalers B-8 to B-11		
			Scaler "B" 8	1e0h	20h
			Scaler "B" 9	1e2h	22h
			Scaler "B" 10	1e4h	24h
			Scaler "B" 11	1e6h	26h
7	1f0...1ff	Scratch Memory	Read/Write Ram		

2.0 TEST PROCEDURE

The sections below give a step by step procedures used to test out the GeBGO Interface Board. All of the test procedures require the following setup in the VXI Mainframe.

SLOT 0	National Instruments VXIcpu-030
SLOT 1	Local Readout Module
SLOT 2	GeBGO Prototype Module

The National Instrument board should be running VxWorks and should be a node on a SUN's Ethernet Network. If the VXIcpu-030 board is properly configured on powerup it will load VxWorks from the network and execute a start up routine. The start up routine will execute a program called resman, and load all of the programs needed to test the GeBGO Board.

In the test procedures below, commands which an operator must enter are written in **BOLD**. The computer output is a smaller font and comments are in the normal font. The commands are entered by an operator on a terminal logged into the National Instruments VXIcpu-030 node. This terminal will be running a C shell under VxWorks.

2.1 VXI Interface Test

There are two test used to verify that the VXI Interface is working on both the GeBGO Prototype Module and the Local Readout Module.

- Verify that resman has properly mapped both boards.
- Execute the VXI register test, which test the VXI registers internal to the VXI Interface Xilinx.

2.1.1 Verifying "resman" mapping

To verify that resman has properly mapped the boards, execute the following commands while in the C shell. This will verify that the VXI Interface Chip allowed the Resource Manager to map the two boards into the VXI address space.

->**vxitedit**
VXI Text Resource Editor (Ver 1.0)
(c) 1991, National Instruments

(1) resource manager display
(2) configuration editor
(3) manufacturer name editor
:
:
[VXItedit:]**1**

The two commands above put you into vxitedit resource manager display. When list is typed a list of devices plugged into the VXIbus are printed out. Since the Local Readout Module and the GeBGO Prototype Module each consist of a block of eight devices, Slot 1 should have eight devices and Slot 2 should have eight devices.

[VXItedit:ResMan:]**list**

Index	la	DevName	MfrName	ModelName	Frame	Slot
0	0	ROOT	Nat'l Insts	VXIcpu-030	0	0
1	8				0	1
2	9				0	1
:	:				:	:
8	f				0	1
9	10				0	2
10	11				0	2
:	:				:	:
16	17				0	2

The display command gives detail information about a specified device. By typing *display 0* information about the VXIcpu-030 will be displayed. By typing *display 1* information about the Local Readout Module will be displayed. By typing *display 9* information about the GeBGO Prototype Module will be displayed. Note that the only difference between display 1 and display 9 is the slot number.

[VXItedit:ResMan:]**display 0**

Fd #	Field Name	Field Value
----	----------	----------
Index = 0		
(0)	Log Addr	0
(1)	Dev Name	ROOT
(2)	Cmdr La	ffff
(3)	Frame	0
(4)	Slot	0
(5)	ManId	ff6
(6)	Mfr Name	Nat' Insts
(7)	Model No	f7
(8)	Model Name	VXIcpu-030
(9)	Dev Class	2
(10)	Sub Class	0
(11)	Mem Space	3
(12)	Mem Base	ffffffff
(13)	Mem Size	0
(14)	Irq Level	none.
(14)	Irq Handle	7.
	Servants	
	8	
	9	
	:	
	:	
	17	

[VXItedit:ResMan:] **display 1**

```
Fd #    Field Name          Field Value
----    ----------          -----------
Index = 1
(0)     Log Addr            8
(1)     Dev Name
(2)     Cmdr La
(3)     Frame               0
(4)     Slot                1
(5)     ManId               f00
(6)     Mfr Name
(7)     Model No            0
(8)     Model Name
(9)     Dev Class           3
(10)    Sub Class           0
(11)    Mem Space           3
(12)    Mem Base            ffffffff
(13)    Mem Size            0
(14)    Irq Level           none.
(14)    Irq Handle          none.
            Servants
            none ..
```

[VXItedit:ResMan:] **display 9**

```
Fd #    Field Name          Field Value
----  ----------            -----------
Index = 9
(0)     Log Addr            10
(1)     Dev Name
(2)     Cmdr La
(3)     Frame               0
(4)     Slot                2
(5)     ManId               f00
(6)     Mfr Name
(7)     Model No            0
(8)     Model Name
(9)     Dev Class           3
(10)    Sub Class           0
(11)    Mem Space           3
(12)    Mem Base            ffffffff
(13)    Mem Size            0
(14)    Irq Level           none.
(14)    Irq Handle          none.
            Servants
            none ..
```

Get back to the C shell with the following commands:

[VXItedit:ResMan:] **exit**

[VXItedit:] **exit**

value = 0 = 0x0

->

2.1.2 Testing Internal VXI Registers

To test the internal VXI registers on the GeBGO Prototype Board, execute the following routine. Note that *sp regtest,2* spawns a task running regtest. The *,2* specifies which slot to execute the regtest on. Do not worry if the task spawned id and name shown below is different from yours. It is good to remember the task name because this will be needed to deleted the task. This is done by entering the command: *td "task name"*

->**ld < regtest.o**
value = 0 = 0x0
->**sp regtest,2**
task spawned: id = 0x3a9fd0, name = t1
value = 3842000 = 0x3a9fd0
->
This routine will test the VXI_INF registers for the board in:
Slot Number 2
PASSED: Internal VXI_INF registers.

Performing Dynamic Configuration Test.
Number of devices = 17
The LA's of the devices found are 0 8 9 10 11 12 13 14 15 16 17 18
The Number of open address Blocks for the BGO board = 28
The open addresses are:
18 20 28 30 40 48 50 58 60
68 70 78 80 88 90 a0 a8 b0
b8 c0 c8 d0 d8 e0 e8 f0
PASSED: Remapping Test

Resetting back to original address = 10
PASSED: Mapped back to original address

->

The VXI configuration register test should be repeated on the Local Readout Module by executing the same command with the Local Readout slot number.

->**sp regtest,1**
task spawned: id = 0x3a9fd0, name = t2
value = 3842000 = 0x3a9fd0
->
VXI library already initialized
This routine will test the VXI_INF registers for the board in:
Slot Number 1
PASSED: Internal VXI_INF registers.

Performing Dynamic Configuration Test.
Number of devices = 17
The LA's of the devices found are 0 8 9 10 11 12 13 14 15 16 17 18
The Number of open address Blocks for the BGO board = 28
The open addresses are:
18 20 28 30 40 48 50 58 60
68 70 78 80 88 90 a0 a8 b0
b8 c0 c8 d0 d8 e0 e8 f0
PASSED: Remapping Test

Resetting back to original address = 8
PASSED: Mapped back to original address

->

2.2 Dual Port Ram Test

The VXI side of the DPR (Dual Port Ram) on the GeBGO board can be tested with the following command. In order to stop the test the operator must type *td t#* where # is the number of the task. The number of the task can be found by typing *i* which list the tasks and information about them. If this test generates an error the problem could be in either the Dual Port Ram or the VXI interface.

->**sp ge_memtest,2** /* *,2* specifies the slot number */

task spawned: id = 0x3a9fd0, name = t9
value = 3842000 = 0x3a9fd0
-> VXI library already initialized

The Board's Base Address for slot 2 is c400
Memory cleared
Memory cleared verified

TESTING MEMORY .... /* Will keep testing until task is stopped */

td t9

->

YOUR TASK NUMBER WILL PROBABLY BE DIFFERENT THAN THE ONE SHOWN HERE! TO PROPERLY DELETE THE TASK ENTER YOUR TASK NUMBER. TO VIEW ALL TASKS THAT ARE RUNNING ENTER THE COMMAND "*i*".

2.3 Scaler Chip Test

The Scaler FPGAs can be tested with the following program and a Wave Form Generator.
Since the Ge_BGO Prototype Board has no way of clocking any of the scaler channels the scalers will be tested interactively.

The format of the scaler test program is given below.
-> sp ge_scaler,

1. First set up the Wave Form Generator to produce a burst of 500 counts when the single burst button is pressed. The pulses should be going from 0 to 5 Volts and have a width of at least 50 nsec. This signal will be used as the input into the scalers by connecting it to the F1 lemo on the front panel of the GeBGO board.
2. Remove all of the scaler channel jumpers except for channel 1.
3. Run the scaler test program by entering the command below. This program resets all of the scaler channels and then monitors and reports any changes to the scaler inputs.

->sp **ge_scaler,2** /* ,2 specifies the slot number */

4. While the scaler test program is running trigger the Wave Form Generator. The monitor should display that Scaler A-1 has received 500 counts. If it takes longer than 1 sec to count up to 500 you will receive two messages like the ones below.

```
0x3c2fb8 (t3): Scaler A-1 = counting
0x3c2fb8 (t3): Scaler A-1 = 500
```

5. Remove the jumper on channel 1 and move it to channel 2.
6. Trigger the Wave Form Generator again. The monitor should now report that channel 2 has received 500 counts.

```
0x3c2fb8 (t3): Scaler A-2 = counting
0x3c2fb8 (t3): Scaler A-2 = 500
```

7. Repeat the test for all 22 scaler channels by moving the jumper to each channel and triggering the Wave Form Generator.

8. To verify that the Scalers meet their specification this test may be repeated with different pulse widths and time between pulses.

2.4 SIO and DAC Test

The following test will use the SIO FPGA to control the onboard and offboard DAC's. Because there is no way for the GeBGO Board to read the analog voltage being produced by the DACs this test will be interactive.

Listed below are the DAC MODULES which will be tested.
(Note: This test could be modified to include test for the DAC_A3 and DAC_B3 which were added later)

DAC MODULE 1 = DAC_A1 On board, Serial Input, 8 Channel, 8 Bit DAC
DAC MODULE 2 = DAC_A2 On board, Serial Input, 8 Channel, 8 Bit DAC
DAC MODULE 3 = DAC_B1 On board, Serial Input, 8 Channel, 8 Bit DAC
DAC MODULE 4 = DAC_B2 On board, Serial Input, 8 Channel, 8 Bit DAC
DAC MODULE 5 = Detector Module A Serial Out
Connected to a Serial Input, 8 Channel, 8 Bit DAC on the GeBGO Test Board. While data is being written to the DAC, data from the ADAT dip switches is being stored and may be read.
DAC MODULE 6 = Detector Module B Serial Out
Connected to a Serial Input, 8 Channel, 8 Bit DAC on the GeBGO Test Board. While data is being written to the DAC, data from the BDAT dip switches is being stored and may be read.

There are two routines which assist the operator in testing the DAC's and SIO interface. The first test is a static test and the second test is a dynamic test. The static test uses a command entered by the operator to modify the output of one channel on a specified dac module. The dynamic test outputs a saw tooth pattern on eight channels of a specified dac module.

The format for the static dac command is given below:
->sp ge_dac, slot_number, dac_module, dac_channel, dac_value

Where:
slot_number = The slot number of GeBGO Board being tested
dac_module = DAC Module to be written to (1 -> 6)
dac_channel = DAC Channel to be written
(0 -> 7) for DAC MODULES 1 -> 4
(0 -> 15) for DAC MODULES 5 & 6
dac_value = DAC Value to be written (hex number 0 -> 0xFF)

The format for the dynamic dac command is given below:
->sp ge_dac_saw, slot_number, dac_module

Where:
slot_number = The slot number of GeBGO Board being tested
dac_module = DAC Module to be written to (1 -> 6)

In order to reduce test circuitry a single PMI DAC8800 was used for DAC A1, DAC A2, DAC B1 and DAC B2. A jumper is used to determine which of the four dacs is being addressed.

There are two other DAC modules connected to the Detector Modules Serial I/O. These two DAC's provide a way of testing the off board serial communication to Detector Module A and B.

For the following command place the jumper in the DAC A1 position.
->**sp ge_dac_saw,2,1** ;This command will output the sawtooth pattern on DAC A1 ;module.

Set a scope up to trigger on the falling edge of channel 1 on DAC A1. Then while looking at DAC channel 1 on the scope use another channel of the scope to view the other DAC channels one at a time. As the scope is moved form channel 2 to 8 the operator should notice a phase shift in the saw tooth. The phase shift between each channel should be 1/8 the cycle time for the whole wave form. This test should be sufficient in verifying that a DAC is working properly. If further testing is needed the operator may make use of the *ge_dac* command described above.

This test will run until the task is deleted. To delete the task enter the "*td*" command followed by the task number that "*ge_dac_saw*" is running under. The task number can be viewed by entering the "*i*" command. The command to delete the task will look like the one below but with a different task number.

->**td t6**

The above commands should be repeat for DAC A2, DAC B1 and DAC B2. Before the command is enter switch the DAC jumper to the correct position. Listed below are the commands for each of the DACs.

SWITCH DAC JUMPER TO DAC A2.
->**sp ge_dac_saw,2,2** ;Outputs sawtooth pattern on DAC A2 module.
;Test waveform with scope.
->**td t7**

SWITCH DAC JUMPER TO DAC B1.
->**sp ge_dac_saw,2,3** ;Outputs sawtooth pattern on DAC B1 module.
->**td t8**

SWITCH DAC JUMPER TO DAC B2.
->**sp ge_dac_saw,2,4** ;Outputs sawtooth pattern on DAC B2 module.
->**td t9**

The same command is used to test the serial output to Detector Module A and B.

->**sp ge_dac_saw,2,5** ;Outputs sawtooth pattern to Detector Module's A Serial output.
;Test wave form with scope.
;NOTE: Only lower 8 channels have saw tooth pattern
->**td t10**

->**sp ge_dac_saw,2,6** ;Outputs sawtooth pattern to Detector Module's B Serial output.
;Test wave form with scope.
->**td t10**

The Test Circuitry provides two 8 bit DIP SWITCHES to test the serial input from the Detector Module A and B. The DIP switches provide the upper eight bits of a 16 bit word. The DIP switches provide the input into the DIP Switch Latch. The DIP Switch Latch is shifted into the SIO FPGA during a write to the associated DAC. The contents of the DIP SWITCHES is latched at the end of the write.

Each time the operator enters a "*ge_dac*" command to the Detector Module's the following sequence of events occur.

- The DAC value is written to the specified DAC Module
- At the same time the contents of the DIP Switch Latch is shifted into the SIO FPGA where it may be read by the Resource Manager.
- The DIP switch Latch is updated with the current contents of the DIP Switches.

The following steps may be followed to test the Serial Input interface from the Detector Modules.

SET DETECTOR MODULE A DIP SWITCHES TO 0x81
NOTE: THE TOP SWITCH IS THE LEAST SIGNIFICANT SWITCH.

->**sp ge_dac,2,5,1,0x80** ;Output data to Detector Module A
task spawned: id = 0x3a7b08, name = t15
value = 3832584 = 0x3a7b08
->
Data from DAC MODULE A = ??00 ;Data on first read is unknown

sp ge_dac,2,5,1,0x80 ;Output data to Detector Module A again
task spawned: id = 0x3a7b08, name = t16
value = 3832584 = 0x3a7b08
->
Data from DAC MODULE A = 8100 ;MS half of word should match DIP Switches on
;second read

Repeat the two commands above for DIP SWITCH settings of 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, and 0x80.

Then repeat the whole test over for Detector Module B.

2.6 Data Flow Test

The program ge_lrm.c generates soft triggers, and verifies that data flows properly for a number of different trigger methods.

DATA FLOW:

GeBGO Prom	->	GeBGO's DPR, and Local Readout's FIFOs'
GeBGO DPR	->	Local Readout's FIFOs'

Each of the transfer methods listed above are triggered Three different ways.

Soft_A by itself
Soft_B by itself
Soft_A and Soft_B together

The ge_lrm.c generates its own TOKEN_IN provided the user pulls up the TOKEN_IN input on the Local Readout Module.

Before the test is run make sure the following conditions are met:

- Proper Boards plugged into the proper slots as described in the beginning of this section.
- JM28 and JM29 are set so that the U31 and U32 PROMS are enabled by the U30 22v10.
- The TOKEN_IN lemo connector on the Local Readout Module is pulled to +5 Volts.

```
->sp ge_lrm,1,2,0xfff      /* 1      = Slot number of LRM board */
                           /* 2      = Slot number of GeBGO board */
                           /*    0xfff = number of passes on test */
task spawned: id = 0x3a9fd0, name = t4
value = 384200 = 0x3a9fd0
->
This program transfers data in the following modes.
    GeBGO Prom and ID Latch  ->   GeBGO DPR and LRM FIFO.
    GeBGO Dual Port Ram      ->   LRM FIFO
The data transfers are triggered in three ways:
    Soft A by itself
    Soft B by itself
    Soft A and Soft B together before token_in.

National Instruments RM:   0
Local Readout:             1
GeBGO Test Module:         2

PULL TOKEN_IN HIGH ON THE LOCAL READ OUT MODULE HIGH!!!

 Testing .....................
.....................
PASSED: ge_lrm has finished     511 passes
```

APPENDIX A: ABEL PROGRAM FOR 22V10

```
module TESTMUX
title   'FAKE DATA-PROM ADDRESS DRIVER FOR GEBGOTST CIRCUIT'

        Testmux DEVICE  'P22V10';

        EA0             PIN 1;  "ENABLE A0 INPUT
        EA1             PIN 2;  "ENABLE A1 INPUT
        EA2             PIN 3;  "ENABLE A2 INPUT
        EA3             PIN 4;  "ENABLE A3 INPUT
        EA4             PIN 5;  "ENABLE A4 INPUT
        EA5             PIN 6;  "ENABLE A5 INPUT
        EA6             PIN 7;  "ENABLE A6 INPUT
        EA7             PIN 8;  "ENABLE A7 INPUT
        EB0             PIN 9;  "ENABLE B0 INPUT
        EB1             PIN 10; "ENABLE B1 INPUT
        EB2             PIN 11; "ENABLE B2 INPUT
        EB3             PIN 13; "ENABLE B3 INPUT
        EB4             PIN 14; "ENABLE B4 INPUT
        EB5             PIN 15; "ENABLE B5 INPUT
        EB6             PIN 16; "ENABLE B6 INPUT
        EB7             PIN 17; "ENABLE B7 INPUT

        CEl             PIN 18; "PROM CHIP ENABLE/ FOR LS BYTE
        CEm              PIN 19; "PROM CHIP ENABLE/ FOR MS BYTE
        A3              PIN 20; "PROM ADDRESS BIT 3
        A2              PIN 21; "PROM ADDRESS BIT 2
        A1              PIN 22; "PROM ADDRESS BIT 1
        A0              PIN 23; "PROM ADDRESS BIT 0

        H,L,X,Z,C = 1,0,.X.,.Z.,.C.;
        EA  = [!EA7,!EA6,!EA5,!EA4,!EA3,!EA2,!EA1,!EA0];
        EB  = [!EB7,!EB6,!EB5,!EB4,!EB3,!EB2,!EB1,!EB0];
        EAL = [!EA7,!EA6,!EA5,!EA4,!EA3,!EA2,!EA1,!EA0];
        EAM = [!EA7,!EA6,!EA5,!EA4,!EA3,!EA2,!EA1];
        EBL = [!EB7,!EB6,!EB5,!EB4,!EB3,!EB2,!EB1,!EB0];
        EBM = [!EB7,!EB6,!EB5,!EB4,!EB3,!EB2,!EB1];

        ADD = [A3,A2,A1,A0];
```

```
Equations
        A0      =   !EA1 # !EA3 # !EA5 # !EA7 #
                    !EB1 # !EB3 # !EB5 # !EB7;
        A0.OE   =   H;

        A1      =   !EA2 # !EA3 # !EA6 # !EA7 #
                    !EB2 # !EB3 # !EB6 # !EB7;
        A1.OE   =   H;

        A2      =   !EA4 # !EA5 # !EA6 # !EA7 #
                    !EB4 # !EB5 # !EB6 # !EB7;
        A2.OE   =   H;

        A3      =   (EB != 0);
        A3.OE   =   H;

        !CEl    =   (EAL != 0) # (EBL != 0);
        CEl.OE  =   H;

        !CEm    =   (EAM != 0) # (EBM != 0);
        CEm.OE  =   H;

Test_Vectors
([EA,   EB]     ->      [ADD,   CEm,    CEl]);
[0,     0]      ->      [0,     H,      H];
[1,     0]      ->      [0,     H,      L];
[2,     0]      ->      [1,     L,      L];
[4,     0]      ->      [2,     L,      L];
[8,     0]      ->      [3,     L,      L];
[16,    0]      ->      [4,     L,      L];
[32,    0]      ->      [5,     L,      L];
[64,    0]      ->      [6,     L,      L];
[128,   0]      ->      [7,     L,      L];
[0,     1]      ->      [8,     H,      L];
[0,     2]      ->      [9,     L,      L];
[0,     4]      ->      [10,    L,      L];
[0,     8]      ->      [11,    L,      L];
[0,     16]     ->      [12,    L,      L];
[0,     32]     ->      [13,    L,      L];
[0,     64]     ->      [14,    L,      L];
[0,     128]    ->      [15,    L,      L];

end TESTMUX
```

APPENDIX B: TEST SOFTWARE

This appendix contains all of the source code used to test the GeBGO board. It also includes the start up script which is executed when the VXI crate is powered up. Listed below is all of the files included in this appendix.

File Name	Description
vxi_error.c	Global subroutines used by many of the following routines.
regtest.c	sp regtest, slot_number slot_number = The slot number of the GeBGO Board being tested. Routine test registers internal to the VXI_INF chip.
ge_memtest.c	sp ge_memtest, slot_number slot_number = The slot number of the GeBGO Board being tested. Routine test the VXI side of the Dual Port Ram on a Ge_BGO Board.
ge_scaler.c	sp ge_scaler, slot_number slot_number = The slot number of the GeBGO Board being tested. Routine continuously monitors all of the scaler values of the specified Ge_BGO Board. Any changes are reported on the terminal.
ge_dac.c	sp ge_dac, slot_number, dac_module, dac_channel, dac_value slot_number = The slot number of the GeBGO Board being tested. dac_module = DAC Module to be written to (1 -> 6). dac_channel = DAC Channel to be written to (0 -> 7). dac_value = DAC Value to be written (hex number 0 -> 0xFF). Routine allows the programmer to change the output of the selected DAC.
ge_dac_saw.c	sp ge_dac_saw, slot_number, dac_module slot_number = The slot number of the GeBGO Board being tested. dac_module = DAC Module to be written to (1 -> 6). Routine outputs a saw tooth wave form on all eight of the DAC channels an the selected DAC Module.
ge_lrm.c	sp ge_lrm, slot_num_lrm, slot_num_ge, num_passes slot_num_lrm = The slot number of the Local Readout Module being tested. slot_num_ge = The slot number of the Ge_BGO Board being tested. num_passes = The number of passes to make on the test.

UTILITY ROUTINES

ma16d16.c ma16d16, hex_address

hex_address = 16 Bit hex address to read and possibly write.

Routine provides a way to read and write A16, D16 address on the VXI Bus. While running this routine accepts the following commands.

<enter>	=	Read the next address.
hex_number <enter>	=	Modify the current address and reread the same address.
^<enter>	=	Read the previous address.
.<enter>	=	Exit the program.

wca16d16.c sp wca16d16, hex_address, hex_data

hex_address = 16 Bit hex address to write continuously to.
hex_data = 16 Bit hex data to write continuously.

Routine continuously writes the given data to the given address.

rca16d16.c sp rca16d16, hex_address

hex_address = 16 Bit hex address to read continuously to.

Routine continuously reads the given address. It reports the value read the first time and will report if the data ever changes.

dump_fifo.c sp dump_fifo, slot_number

slot_number = The slot number of the Local Readout Module with the FIFO's to be Read.

Routine provides a way to read the FIFO's on the Local Readout Module.