GAMMASPHERE DATA ACQUISITION SYSTEM and SYSTEM TEST

March 8, 1994

1.0 OVERVIEW OF GAMMASPHERE DATA ACQUISITION SYSTEM 2
1.1 Description of the GammaSphere Data Acquisition System 2
1.2 Block Diagram of the GammaSphere Data Acquisition System 3
1.3 Modules Used in the GammaSphere Data Acquisition System 6

2.0 GAMMASPHERE DATA ACQUISITION SYSTEM TEST 7

APPENDIX A: TEST SOFTWARE 12

1.0 OVERVIEW OF GAMMASPHERE DATA ACQUISITION SYSTEM

This section gives a general overview of the GAMMASPHERE Data Acquisition System which was designed and developed by Argonne National Laboratory's ECT Department (Electronics and Computing Technology).

1.1 Description of the GammaSphere Data Acquisition System

The main function of the Data Acquisition System is to collect the digitized data from all of the GeBGO Detectors and transmit it to a number of VME Memory Cards where it may be processed. The normal path that the data takes to these VME Memory Cards is described in detail below.

- Trigger Module detects an event and sends a Trigger signal to the MRM.
- DPR for channel A on the MRM is transferred into pipeline registers. This data can be written by the user and is used as the header data for the event. (4-32 bit words)
- 22V10 on MRM Initiates token which is first sent to the FASTOUT FPGA on MRM.
- Header data in the MRM's pipeline registers is transmitted to the Event Builder via the fiber link. When the Event Builder detects the beginning of an event from the fiber link it begins storing the data into FIFOs while counts the number of words. When the End of an Event is detected the Event Builder transmitting the data out of the FIFOs to the selected Memory Card. The Event Builder inserts the size of the event into the LS half of the first 32 bit word. The Event Builder is also in control of which memory card receives the data and switches memory cards when memory fills up on the selected memory card.
- Token is returned to the 22V10 on MRM.
- MRM removes token to the on board FASTOUT and sends the token to first LRM.

- Each LRM goes through the same sequence of events.
    - LRM relays token to the FASTOUT FPGA on its first GeBGO Modules.
    - Each GeBGO Modules goes though the same sequence of events.
        - If the local RDY/BSYA and/or the RDY/BSYB have not been detected then the token is immediately sent to the next GeBGO Modules. If this is the last GeBGO Modules (determined by a jumper) the token is returned to the LRM.
        - If RDY/BSYA and/or RDY/BSYB have been detected and latched then the appropriate data is transmitted from the pipeline registers to the LRM via the VXI Local Bus. The LRM relays the data to the MRM via a 32 bit differential ribbon cable bus. The LRM also stores the data in a FIFO so it may be read by the local Resource Manager. The MRM receives the data from the LRM and transmits it to the Event Builder via the fiber link. When the GeBGO Module is done transmitting its data it sends the token to the next GeBGO Module. If this is the last GeBGO Module (determined by a jumper) then the token is returned to the LRM.
    - The LRM sends the token to the next LRM, or if this is the last LRM then it returns the token to the MRM.
- The MRM removes the token from the first LRM and sends the token to GeBGO Modules local to its VXI mainframe.
    - The GeBGO Module behave the same as the ones with LRM in their VXI Mainframe. The last GeBGO Module returns the token to the MRM.
- The MRM removes the token form the local GeBGO modules and pulses the EOE signal to all GeBGO boards.

1.2 Block Diagram of the GammaSphere Data Acquisition System

On the following page is a block diagram of the GammaSphere System. The final system will contain seven VXI mainframes in a room above the GammaSphere, and a number of VME mainframes in a control room about 250 feet from the VXI mainframes. The VXI Mainframe will consist of one Master VXI Mainframe and six Slave VXI Mainframes. The VXI mainframes will be configured as shown below.

SLOT	Master VXI Mainframe	SLOT	Slave VXI Mainframes
0	NI VXIcpu-030	0	NI VXIcpu-030
1	Master Readout Module	1	Local Readout Module
2-x	Optional GeBGO Modules	2-11	GeBGO Modules
x+1	MicroBall Interface Module	12	Local Trigger Module
x+2	Master Trigger Module		

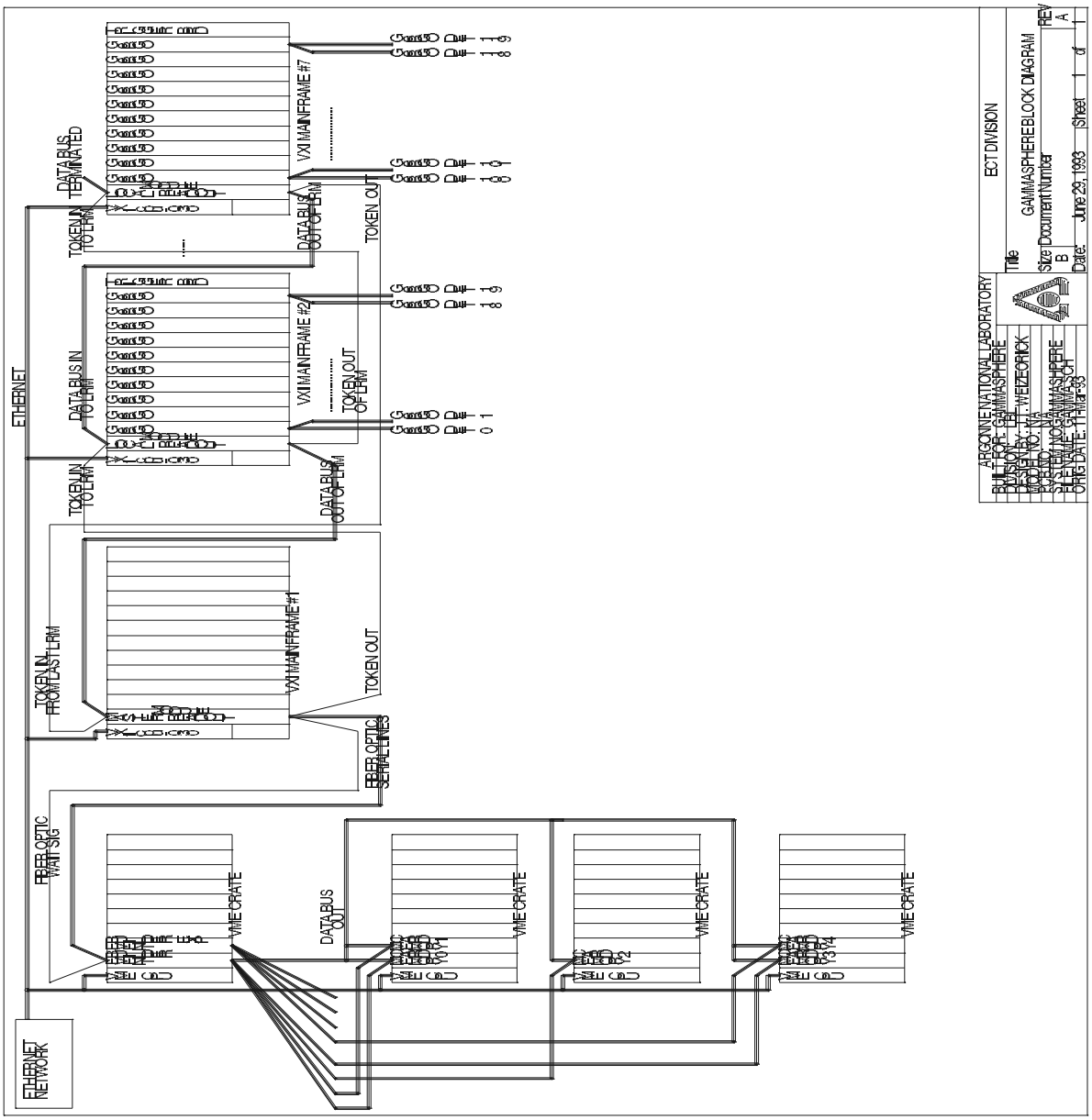
The VXI Mainframes are interfaced together by a 32 bit differential ribbon cable bus which connects the Local Readout Modules to the Master Readout Module. The MRM uses the 32 bit data stream to generate four serial bit streams which are transmitted to the Event Builder Board over four fiber optic lines.

Bits	Fiber Line
0-7	Fiber Optic Line 1
8-15	Fiber Optic Line 2
16-23	Fiber Optic Line 3
24-31	Fiber Optic Line 4

If one VME rack is used it could be configured as below. Each Memory Module may be located in its own VME Rack, but the Event Builder and the Event Builder Expansion Board must be located in the same VME Rack with the outer rows of P2 connected.

SLOT	VME Rack	
0	Processor Board Running VxWorks	
1	CES Memory Module 0	
2	CES Memory Module 1	
3	CES Memory Module 2	
4	CES Memory Module 3	
5	CES Memory Module 4	
6	Event Builder	(Interface to memory boards 0-3)
7	Event Builder Expansion Board	(Interface to memory boards 0-3)

The data comes into the Event Builder over four fiber optic lines and is transmitted to the CES Memory Modules over a FARA Bus. The FARA Bus consist of a 32 bit Differential ECL bus and a 12 pin header for control signals. The ECL bus comes from the Event Builder and is connected to all of the Memory Modules. The Event Builder has four 12 pin headers to provide the control signals for the first four Memory Modules. For each Event Builder Expansion Board an additional four Memory Modules may be added up to a maximum of 64 Memory Modules.


ETHERNET
ETHERNET NETWORK
VXI MAINFRAME #7
VXI MAINFRAME #2
VXI MAINFRAME #1
DATA BUS TERMINATED
DATA BUS IN TO LRM
DATA BUS OUT OF LRM
TOKEN IN TO LRM
TOKEN OUT OF LRM
TOKEN IN FROM LAST LRM
TOKEN OUT
FIBER OPTIC SERIAL LINES
FIBER OPTIC WAIT SIG
DATA BUS OUT
VME CRATE
VME CRATE
VME CRATE
VME CRATE
ARGONNE NATIONAL LABORATORY
ECT DIVISION
GAMMASPHERE BLOCK DIAGRAM
Size B
Date: June 29, 1993
Sheet 1 of 1
REV A

1.3 Modules Used in the GammaSphere Data Acquisition System

The GammaSphere Data Acquisition System consist of the following boards listed below.

VXI Modules
- National Instruments VXI-cpu030
- Master Readout Module
- Local Readout Module
- Local Trigger Module
- Master Trigger Module
- GeBGO Module

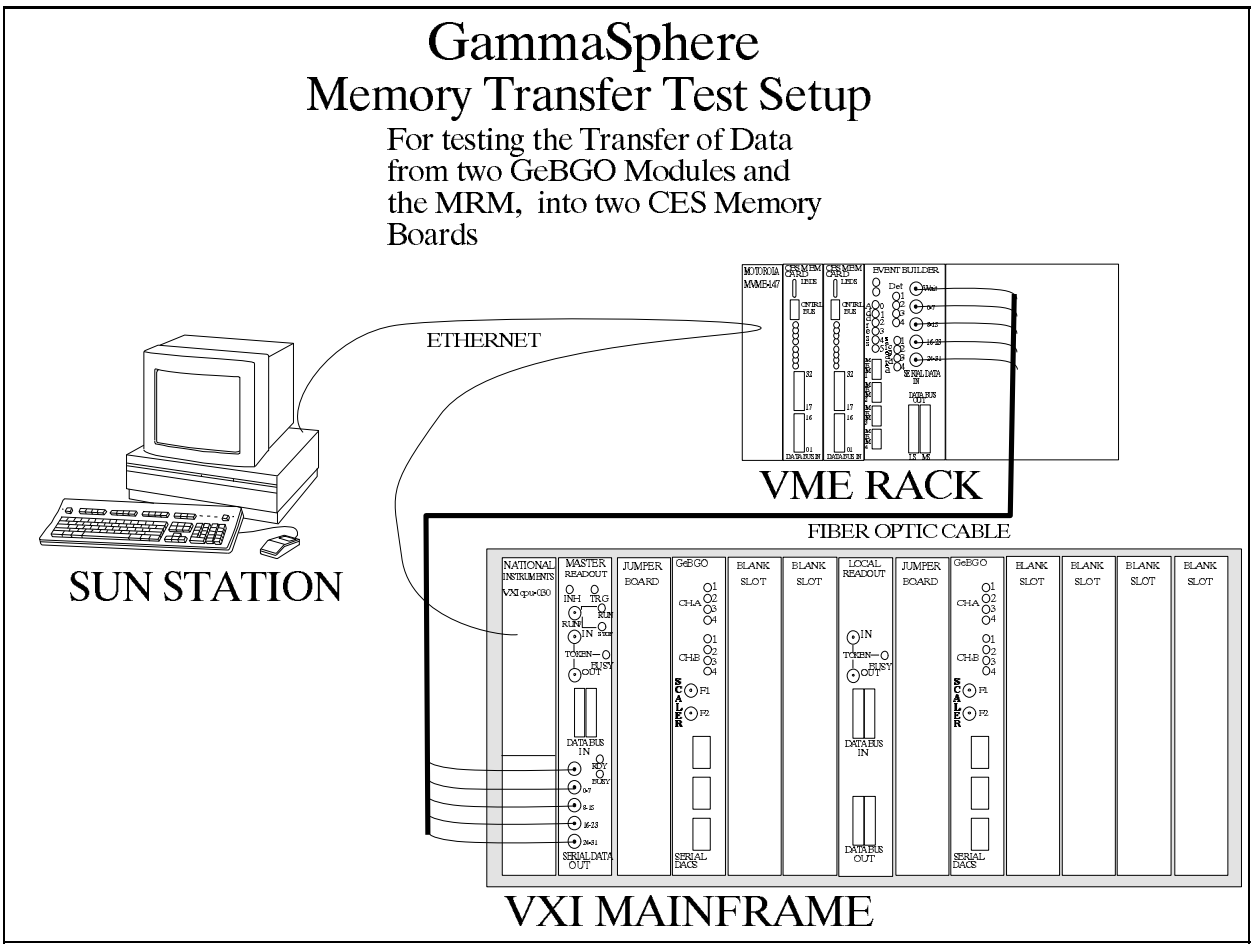
VME Modules
- Processor Cards - MVME147
- Event Builder
- Event Builder Expansion Boards
- CPM Memory Cards

2.0 GAMMASPHERE DATA ACQUISITION SYSTEM TEST

The sections below give a step by step procedures used to test out the GammaSphere Data Acquisition System. A figure of the system configuration is given on the next page.

The system test is controlled by a Sun Station on the network which is logged into the VME Rack and the VXI Mainframes. The VME and VXI nodes are running a task under VxWorks, which waits for Remote Procedure Calls (RPCs) from the network. When it receives a command it executes the task and returns the results.

In order to perform this test the operator must generate a configuration file and hook the modules up as defined in the configuration file. Below is an example of a configuration file and how the modules are plugged in to match the configuration file. Test may need to be performed on a number of different configurations to complete test the GammaSphere Data Acquisition.



```
CONFIGURATION FILE
; This is a configuration file to be used as input to the SYSTEM TEST PROGRAM

; Master Readout Module
; MRM     nodeid      slot_number
  MRM     walleye     1

; Local Readout Module
; LRM     nodeid      slot_number
  LRM     walleye     6

; GeBGO Modules */
; The Active_Channels indicate which channels are to be filled with test data
; The possible options are A, B, A&B, or NONE.
; Be sure to place the channels in the order they will receive the token.
; GeBGO   nodeid      slot_number Active_Channels
  GeBGO   walleye     7   A&B
  GeBGO   walleye     4   B

;  Event Builder
;  EB      nodeid      base_address    eb_str_adr  num_cards
   EB      mallard     0xffff9000      0x0         0x2

; CES Memory Cards
; CES     nodeid      base_address    eb_address  overflow_cnt
  CES     mallard     0x11000000      0x0         0xc00
  CES     mallard     0x10000000      0x1         0xc00
```

With the above configuration the following cards must be plugged into the following locations.

VME Rack — Node Name = mallard

- Slot 0 - MVME 147 Processor board running VxWorks
- Slot 1 - HSM-8170 CES Memory Board: Terminated
  Jumpered for Address 0x11000000
- Slot 2 - HSM-8170 CES Memory Board: Unterminated
  Jumpered for Address 0x10000000
- Slot 3 - EB - Event Builder

VXI Mainframe Node Name = walleye

Slot 0 - VXIcpu-030 Processor board running VxWorks
Slot 1 - Master Readout Module
Slot 2 - Jumper Board
Slot 3 - Jumper Board
Slot 4 - GeBGO #2 (Should have the Token Return Jumper inserted)
Slot 5 - Blank
Slot 6 - Local Readout Module (Simulate Second Rack)
Slot 7 - GeBGO #1 (Should have the Token Return Jumper inserted)

Cables and Connections

ECL BUS IN VME RACK:

Two Ribbon Cables between Event Builder and Memory Cards.
Event Builder at one end and Terminated Memory Card at the other end.

ECL BUS CONTROL SIGNALS:

Two sperate 12 pin ribbon cables.
1 between Event Builder Memory Card 0 connector and the terminated memory card.
1 between Event Builder Memory Card 1 connector and the unterminated memory card.

DIFFERENTIAL RIBBON CABLE BUS:

Two Ribbon Cables between the MRM and the LRM. The differential input of the LRM should have the Termination Plug connected to it.

TOKEN

The Token Out Lemo connector form the MRM should go to the Token In Lemo connector on the LRM. The Token Out from the LRM should go to the Token In connector on the MRM.

If the proper startup files are used for the VME and VXI nodes all of the programs needed to run the SYSTEM test will be loaded in on bootup and the RPC will be started. The

The VME startup program for our system is listed below.

```
shellPromptSet("mallard:vxworks-> ")
hostAdd ("llama", "130.202.21.102")
hostAdd ("walleye", "130.202.23.29")
hostAdd ("pike", "130.202.21.126")
cd "cougar:/home/cougar/WEIZEOR/gamma"
ld < vme_ma32d32.o
ld < vme_ma16d16.o
cd "cougar:/home/cougar/WEIZEOR/gamma_new"
ld < vme_init_mem.o
ld < vme_init_eb.o
ld < tst_data_ver.o
ld < vme_stubs.o
ld < /home/cougar/WEIZEOR/gamma_rpc/svc.o
sp gen_svc
```

The VXI startup program for our system is listed below.

```
shellPromptSet ("walleye:vxworks-> ")
hostAdd ("llama", "130.202.21.102")
hostAdd ("mallard", "130.202.21.98")
hostAdd ("pike", "130.202.21.126")
nfsMount ("cougar","/home/cougar","/home/cougar/")
nfsAuthUnixSet ("cougar",2100,2100,0)
VXIinit
ld < /home/cougar/WEIZEOR/walleye/boot/resman.o
resman
cd "/home/cougar/WEIZEOR/gamma"
ld < vxi_error.o
ld < ma16d16.o
ld < rca16d16.o
ld < wca16d16.o
ld < regtest.o
ld < dump_fifo.o
ld < ge_memtest.o
ld < ge_lrm.o
ld < ge_scaler.o
ld < ge_dac.o
ld < ge_dac_saw.o
cd "/home/cougar/WEIZEOR/gamma_new"
ld < init_gebgo.o
ld < init_lrm.o
ld < init_mrm.o
ld < tst_data_wt.o
ld < trg_ge.o
ld < trg_mrm.o
ld < vxi_stubs.o
ld < /home/cougar/WEIZEOR/gamma_rpc/svc.o
sp gen_svc
```

After bootup both the VME and VXI nodes will be running their RPC server tasks and are now ready to accept commands form the SUN. To execute the SYSTEM TEST enter the following command from the SUN.

%**ges2mem_tst config_file_name**

ges2mem_tst performs the following RPC once to set up all of the modules.

vme_init_mem (0x11000000, 0x0, 0xcc0)
vme_init_mem (0x10000000, 0x1, 0x230)
vme_init_eb (0xffff9000, 0x0, 0x2)
init_mrm (0x1)
init_lrm (0x6)
init_gebgo (0x7, A&B)

Then ges2mem_tst goes into a loop where it writes test data to all of the GeBGO Modules, generates a rdy\bsy trigger for all of the GeBGO Modules, generates a trigger on the MRM, and then verifies the data has been transferred into the CES Memory Card. This loop is repeated until a one is shifted though every bit.

tst_data_wt (1, A, 0x0, 0x1, F)
tst_data_wt (7, A, 0x0, 0x2, M)
tst_data_wt (7, B, 0x0, 0x4, M)
tst_data_wt (4, B, 0x0, 0x8, L)
trg_ge (1 A 7 A&B 4 B)
trg_mrm (0x1)
tst_data_ver(0x11000000, 0xcc0, 0x5, 0x14, 0x0, 0x1)

The ges2mem_tst program shifts the test bit and repeats.
tst_data_wt (1, A, 0x0, 0x2, F)
tst_data_wt (7, A, 0x0, 0x4, M)
tst_data_wt (7, B, 0x0, 0x8, M)
tst_data_wt (4, B, 0x0, 0x10, L)
trg_ge (1 A 7 A&B)
trg_mrm (0x1)
tst_data_ver(0x11000050, 0xcc0, 0x5, 0x14, 0x0, 0x2)

The seven RPC are repeated with changing parameters until a set bit has been shifted through all four 32 bit words in the first channel on the first GeBGO module. If an error occurs the RPC will return a nonzero number and the SUN will report the error and stop executing the test. An additional error message may be found on the node that the error occurred.

APPENDIX A: TEST SOFTWARE

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

SYSTEM TEST PROGRAMS

SUN:

ges2mem_tst.c - Main Routine which controls the test flow. It reads in a configuration file and uses this information to control the test flow and parameters sent to the VME and VXI nodes through RPC (Remote Procedure Calls)

clnt_sub.c - Routine that sets up to make the RPC.
gen_clnt.c - Generated by rpcgen. Routine is the local svc_prg_1 routine.
gen_xdr.c - Generated by rpcgen. Uses gen.h include file
gamma_ab_ab.cfg -Sample Configuration file.
gamma0.h - Include file for ges2mem_tst.c Note: name was changed from gamma.h to prevent conflict.
gen.x - Used by rpcgen
gen.h - Generated by rpcgen

VXI & VME: - RPC ROUTINE

svc_prg_1.c - Routine which accepts the pass_info structure, which is a list of up to 9 ascii words, looks for the program name and executes it.

gen_svc.c - Created by rpcgen. Had to modify gen.h to make gen_vx.h in gen_svc.c replace gen.h with gen_vx.h

gen_xdr_vx.c- Modified from gen_xdr.c. Makes the RPC work with VxWorks. See Build.hlp for details.

gen_vx.h - Include file for generated server files to run under VxWorks.

vxi_stub.c - In order for svc_prg_1 to be executable on VME mainframe the vxi routines must have some kind of a routine. The vxi_stub.c returns an error message for any VXI routines that are asked to be executed by a VME mainframe.

vme_stub.c - Same as above except for the VXI mainframe.

VXI routines called by svc_prg_1:

init_gebgo.c - Used to initialize the GeBGO Modules

init_lrm.c - Used to Initialize the Local Readout Modules

init_mrm.c - Used to Initialize the Master Readout Modules

tst_data_wt.c - Writes test data into the GeBGO Modules or the Master Readout Module.

trg_ge.c - Generates RDY/BSYA and/or RDY/BSYB on the GeBGO Modules. This moves the test data into the pipeline registers and sets up the GeBGO Modules to transfer the data when they receive the TOKEN.

trg_mrm.c - Generates a Trigger signal in the Master Readout Module. This starts the TOKEN passing scheme.

vxi_stubs.c - Described above

VME routines called by svc_prg_1:

init_mem.c - Initializes the CES Memory Modules

init_eb.c - Initializes the Event Builder

data_ver.c - Verifies that the test data has been received
(Calculates Event size and masks out start & stop bits)

vme_stubs.c - Described above

HELP FILES:

build.hlp - Describes the procedure used to build the various routines used in the system test programs. This procedure could probably be automated at some time.

UTILITYS

SCRIPTS:

c - Used to compile c programs to be executed on the VxWORKS Nodes.
ccgen_tst - Used to build ges2mem_tst.c which runs on the SUN.
csvc - Used to build svc_prg_1.c which will run down on the VxWorks Nodes.

BOOT UP FILES:

startup1.cmd - VME Boot Up File
startup0.cmd - VXI Boot Up File

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_num<enter> = Modify the data at the current address and reads it back.
^<enter> = Read the previous address.
.<enter> = Exit the program.

ma32d32.c Same as ma16d16 except it uses a 32 bit address and addresses 32 bit Data.
vme_ma16d16.c Same as ma16d16 except it was written for the MVME147 card.
vme_ma32d32.c Same as ma32d32 except it was written for the MVEM147 card.

wca16d16.c 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 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.