

 <b>GLAST ACD TECHNICAL DOCUMENT</b>	Document # <b>ACD-PROC-000062</b>	Date Effective <b>08-15-02</b>
	Prepared by: <b>D. A. Sheppard</b>	Supersedes <b>None</b>
	Subsystem <b>ACD Electronics</b>	
Document Title <b>GLAST LAT ACD GARC V1 Design Description and Test Plan</b>		

**Gamma-ray Large Area Space Telescope (GLAST)  
Large Area Telescope (LAT)  
Anti-Coincidence Detector (ACD)**

**GARC ASIC (GLAST ACD Readout Controller)**

**GARC V1 Design Description and Test Plan**

**ACD-PROC-000062**

## GARC Test Plan Table of Contents

1.0	Description of the GARC ASIC .....	4
2.0	Prior to Starting the GARC Functional Test .....	4
3.0	GARC Command Mnemonics and Functions .....	5
4.0	Testing the GARC Digital Logic .....	8
5.0	Initial Reset Test.....	9
6.0	GARC Power Measurements .....	11
6.1	Measurement at the Nominal Power Supply Voltage .....	11
6.2	Measurement at the Minimum Power Supply Voltage .....	11
6.3	Measurement at the Maximum Power Supply Voltage.....	12
6.4	Measurement of the GARC Bias Resistor Voltages .....	13
7.0	GARC Register Read/Write Tests .....	13
7.1	Veto_Delay Register Test .....	14
7.2	HVBS Level Register Test.....	14
7.3	SAA Level Register Test .....	15
7.4	Hold Delay Register Test.....	15
7.5	Veto Width Register Test.....	15
7.6	HitMap Width Register Test .....	15
7.7	HitMap Deadtime Register Test.....	16
7.8	HitMap Delay Register Test.....	16
7.9	PHA En0 Register Test .....	16
7.10	PHA En1 Register Test .....	17
7.11	Veto En0 Register Test .....	17
7.12	Veto En1 Register Test .....	17
7.13	MaxPHA Register Test .....	17
7.14	GARC Mode Register Test .....	18
7.15	PHA Threshold Channel 00 Register Test .....	18
7.16	PHA Threshold Channel 01 Register Test .....	19
7.17	PHA Threshold Channel 02 Register Test .....	19
7.18	PHA Threshold Channel 03 Register Test .....	19
7.19	PHA Threshold Channel 04 Register Test .....	19
7.20	PHA Threshold Channel 05 Register Test .....	20
7.21	PHA Threshold Channel 06 Register Test .....	20
7.22	PHA Threshold Channel 07 Register Test .....	20
7.23	PHA Threshold Channel 08 Register Test .....	21
7.24	PHA Threshold Channel 09 Register Test .....	21
7.25	PHA Threshold Channel 10 Register Test .....	21
7.26	PHA Threshold Channel 11 Register Test .....	21
7.27	PHA Threshold Channel 12 Register Test .....	22
7.28	PHA Threshold Channel 13 Register Test .....	22
7.29	PHA Threshold Channel 14 Register Test .....	22
7.30	PHA Threshold Channel 15 Register Test .....	23
7.31	PHA Threshold Channel 16 Register Test .....	23
7.32	PHA Threshold Channel 17 Register Test .....	23
7.33	ADC TACQ Register Test .....	24
8.0	GARC Test Pin Mux Verification .....	24
9.0	Test of the Hold Delay Operation.....	24
10.0	Test of the AEM VETO Signal Functionality .....	27
10.1	Veto Delay Test.....	28
10.2	Veto Width Test .....	29
10.3	Veto Enable and Disable Test .....	30
10.4	Discriminator Input Minimum Width Test .....	31

10.5	Discriminator Input Extended Width Test .....	32
10.6	Discriminator Input Double Pulse Test .....	33
11.0	Test of the HitMap Functionality .....	34
11.1	HitMap Width Test .....	34
11.2	HitMap Delay Test .....	35
11.3	HitMap Deadtime Stretch Test .....	36
11.4	HitMap Minimum Width Test .....	37
11.5	HitMap Extended Pulse Test .....	37
11.6	HitMap Double Pulse Test .....	37
12.0	Synchronization of the HitMap Latch to the Discriminator Input Pulse .....	38
13.0	Strobe Test .....	39
14.0	Test of the MAX5121 DAC and Associated Control Functions .....	40
14.1	Capture of the DAC Control Signals .....	40
14.2	Test of the GARC DAC Interface Circuitry .....	41
15.0	Capture of the ADC Control Signals .....	41
16.0	ADC TACQ Test .....	41
17.0	Test of the GARC Return Data Parity .....	43
18.0	Test of the HVBS Triple Modular Redundancy Circuitry .....	43
19.0	Test of the Look-At-Me Circuitry .....	45
20.0	Test of the GAFE Parity Command Circuitry .....	45
21.0	Test of the GARC LVDS Circuitry Driver Currents .....	46
22.0	FREE Board ID Circuit Test .....	48
23.0	Maximum PHA Return Test .....	48
24.0	PHA Enable/Disable Test .....	49
25.0	PHA Threshold Verification Test .....	50
26.0	GARC Diagnostics Verification .....	51
26.1	Test of the GARC Diagnostic Status Register .....	51
26.2	AEM Command Parity Error Simulation Test .....	51
26.3	AEM Data Parity Error Simulation Test .....	52
26.4	Command Counter Test .....	52
26.5	GARC Diagnostic State Loop Counter Test .....	52
27.0	GAFE Interface Test .....	53
28.0	Testing the GAFE Logic for ASICs Connected to the GARC .....	53
28.1	Initial GAFE Logic Reset Test .....	53
28.2	GAFE ASICs Mode Register Test .....	54
28.3	GAFE ASIC DAC1 Register Test .....	54
28.4	GAFE ASIC DAC2 Register Test .....	54
28.5	GAFE ASIC DAC3 Register Test .....	55
28.6	GAFE ASIC DAC4 Register Test .....	55
28.7	GAFE ASIC DAC5 Register Test .....	56
28.8	GAFE ASIC Diagnostics Verification .....	56
28.9	GAFE ASIC Broadcast Command Functional Verification .....	56
Appendix 1:	GARC Documentation .....	56
Appendix 2:	GARC Configuration Command Format .....	57
Appendix 3:	GARC Event Data Format: .....	58
Appendix 4:	GARC Configuration Data Readback Format: .....	59
Appendix 5:	GARC Pin List .....	60

## 1.0 Description of the GARC ASIC

GARC is the acronym for the Gamma-Ray Large Area Space Telescope (GLAST) Anti-Coincidence Detector (ACD) Readout Controller Application Specific Integrated Circuit (ASIC). The GARC is the main logical interface for the ACD to the LAT instrument electronics. GARC provides command and data return functions for electronics boards associated with the ACD. There are 12 GARCs in the flight system, one for each of the event electronics cards. The GARC is a +3.3V single supply device and communicates to the LAT digitally via LVDS interfaces. The GARC is packaged in a 208 pin plastic quad flatpack.

This document describes the functions and test plan of the GARC ASIC. The GARC is designed to meet the requirements of the ACD Level IV Electronics Requirements, LAT-SS-00352, and the ACD-LAT Interface Control Document, LAT-SS-00-363.

The design was done utilizing Verilog as the description language, Exemplar Leonardo Spectrum as the synthesis tool targeting the Tanner Agilent 0.5  $\mu\text{m}$  standard cell library, and Tanner L-Edit as the automated place and route layout tool. Core verification was performed via the Tanner LVS tool.

## 2.0 Prior to Starting the GARC Functional Test

This is the nominal functional test used to verify that the GARC meets the specified requirements. Prior to starting this test, the power supply to the GARC test board should be verified to be at the correct voltage, which is nominally +3.3V. Variations of this test may include voltages in the +3.0V to +3.6V range, temperatures in the range of -20C to +60C, and different clock frequencies. Nominally, the ACD clock frequency is to be 20 MHz.

Additionally, prior to starting the functional test, the proper biasing of the GARC circuitry is to be verified. The following table details the expected biases.

<b>GARC Bias Signal</b>	<b>GARC Pin</b>	<b>Nominal Bias Resistor Required</b>	<b>Function of Bias Resistor</b>
HLD_WOR_BIAS	104	7 k $\Omega$ to DGND	HLD Wired-OR Rcvr Bias
BIAS_RCVR	156	84 k $\Omega$ to VDD	LVDS Rcvr Bias Adjust
BIAS_DRV_H	160	7 k $\Omega$ to DGND	LVDS Driver Bias Adjust
BIAS_DRV_L	169	7 k $\Omega$ to DGND	LLDRV Driver Bias Adjust
LVDS_PRESETADJ	184	7 k $\Omega$ to VDD	LVDS Preset Adjust

An additional variation possible to this test is to individually vary the bias resistors by some nominal amount (e.g., 10%) and utilize this functional test to determine variations in performance, if any.

### 3.0 GARC Command Mnemonics and Functions

The following table represents each of the available GARC commands. Additionally, all GAFE commands are passed through the GARC. These command patterns are detailed in the document discussing the GAFE logic. There are two types of GARC commands – trigger commands (4 bits in length) and configuration commands (34 bits in length). The AEM-ACD ICD contains the authoritative formats for all command types, but they are repeated in the appendices of this document for convenience. The table below defines the command mnemonics that will be used in this test.

Note that a GAFE will process a write command either for the address hard-wired to the chip address pins or to an address of 'h1F, the GAFE broadcast address. A GAFE will process a read command only for an address identical to the hard-wired address. It is an operational constraint that, for any given ACD circuit board, each GAFE must have a unique address.

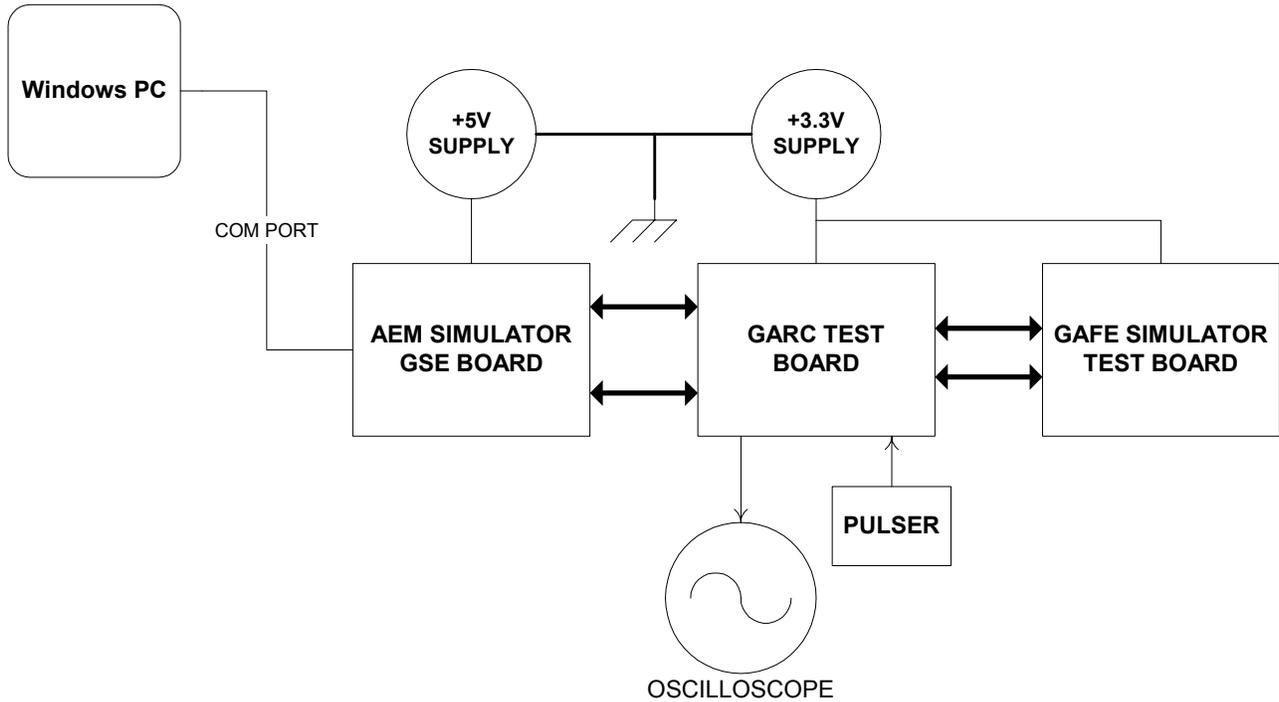
GARC Cmd No.	ACD Command Mnemonic	Rd/Wr Status	Select GARC=0 GAFE=1	Function Block	Register Number	No. of Data Bits	Command Function
1	Trigger_ZS	N/A	N/A	N/A	N/A	N/A	ACD Trigger, Zero-Suppression Enable
2	Trigger_NOZS	N/A	N/A	N/A	N/A	N/A	ACD Trigger, No Zero-Suppression
3	GARC_Reset	W	0	0	1	0	Generates reset for GARC and GAFE registers
4	Veto_Delay_Wr	W	0	0	2	5	Sets Delay from Disc_In to VETO Out
5	Veto_Delay_Rd	R	0	0	2	5	Reads contents of Veto_Delay register
6	GARC_Cal_Strobe	W	0	0	3	0	Sends Calibration Strobe signal to all GAFEs
7	HVBS_Level_Wr	W	0	0	8	12	Sets GARC register value from which HVBS may be commanded in the science mode
8	HVBS_Level_Rd	R	0	0	8	12	Reads contents of HVBS Level register
9	SAA_Level_Wr	W	0	0	9	12	Sets GARC register value from which HVBS may be commanded when in the SAA
10	SAA_Level_Rd	R	0	0	9	12	Reads contents of SAA Level register
11	Use_HV_Normal	W	0	0	10	0	Sends 12 bit value in HVBS Level register to the MAX5121 DAC
12	DAC_HVReg_Rd	R	0	0	10	0	Reads MAX5121 DAC Config Register
13	Use_SAA_Normal	W	0	0	11	0	Sends 12 bit value in SAA Level register to the MAX5121 DAC
14	DAC_SAAReg_Rd	R	0	0	11	0	Reads MAX5121 DAC Config Register
15	Hold_Delay_Wr	W	0	0	12	7	Sets GARC Hold Delay
16	Hold_Delay_Rd	R	0	0	12	7	Reads back value of GARC Hold Delay register
17	Veto_Width_Wr	W	0	0	13	3	Sets width of the VETO signals
18	Veto_Width_Rd	R	0	0	13	3	Reads back width of the VETO width register
19	HitMap_Width_Wr	W	0	0	14	4	Sets width of HitMap pulses
20	HitMap_Width_Rd	R	0	0	14	4	Reads back value in the HitMap width register
21	HitMap_Deptime_Wr	W	0	0	15	3	Sets stretch at end of HitMap pulses
22	HitMap_Deptime_Rd	R	0	0	15	3	Sets stretch at end of HitMap pulses
23	Look_At_Me	W	0	1	4	16	GARC interface selection command (primary/secondary)
24	HitMap_Delay_Wr	W	0	1	8	5	Sets delay from Disc_In to HitMap pulse
25	HitMap_Delay_Rd	R	0	1	8	5	Reads back contents of the HitMap_Delay register
26	PHA_EN0_Wr	W	0	1	9	16	Enables and Disables PHA readout enable, channels 0 -15

27	PHA_EN0_Rd	R	0	1	9	16	Reads back contents of the PHA_EN0 register
28	VETO_EN0_Wr	W	0	1	10	16	Enables and Disables VETO signals, channels 0 -15
29	VETO_EN0_Rd	R	0	1	10	16	Reads back contents of the VETO_EN0 register
30	PHA_EN1_Wr	W	0	1	12	2	Enables and Disables PHA readout enable, channels 16 & 17
31	PHA_EN1_Rd	R	0	1	12	2	Reads back contents of the PHA_EN1 register
32	VETO_EN1_Wr	W	0	1	13	2	Enables and Disables VETO signals, channels 16 & 17
33	VETO_EN1_Rd	R	0	1	13	2	Reads back contents of the VETP_EN1 register
34	Max_PHA_Wr	W	0	1	15	5	Sets the Maximum number (limit) of PHA to be returned in a single event data packet
35	Max_PHA_Rd	R	0	1	15	5	Reads back the contents of the Max_PHA register
36	GARC_Mode_Wr	W	0	2	8	11	Sets values of GARC mode bits
37	GARC_Mode_Rd	R	0	2	8	11	Reads back the value of the GARC mode register
38	GARC_Status	R	0	2	9	6	Reads back the value of the GARC status register
39	GARC_Cmd_Reg	R	0	2	10	16	Reads back the value of the GARC command register
40	GARC_Diagnostic	R	0	2	11	16	Reads back the value of the GARC diagnostic register
41	GARC_Cmd_Rejects	R	0	2	12	8	Reads back the number of rejected commands since reset
42	FREE_Board_ID	R	0	2	13	8	Reads back the FREE board serial number
43	GARC_Version	R	0	2	14	3	Reads back the GARC ASIC version number
44	PHA_Thresh00_Wr	W	0	3	8	12	Writes PHA ZS threshold for channel 00
45	PHA_Thresh00_Rd	R	0	3	8	12	Reads back PHA ZS threshold register for channel 00
46	PHA_Thresh01_Wr	W	0	3	9	12	Writes PHA ZS threshold for channel 01
47	PHA_Thresh01_Rd	R	0	3	9	12	Reads back PHA ZS threshold register for channel 01
48	PHA_Thresh02_Wr	W	0	3	10	12	Writes PHA ZS threshold for channel 02
49	PHA_Thresh02_Rd	R	0	3	10	12	Reads back PHA ZS threshold register for channel 02
50	PHA_Thresh03_Wr	W	0	3	11	12	Writes PHA ZS threshold for channel 03
51	PHA_Thresh03_Rd	R	0	3	11	12	Reads back PHA ZS threshold register for channel 03
52	PHA_Thresh04_Wr	W	0	3	12	12	Writes PHA ZS threshold for channel 04
53	PHA_Thresh04_Rd	R	0	3	12	12	Reads back PHA ZS threshold register for channel 04
54	PHA_Thresh05_Wr	W	0	3	13	12	Writes PHA ZS threshold for channel 05
55	PHA_Thresh05_Rd	R	0	3	13	12	Reads back PHA ZS threshold register for channel 05
56	PHA_Thresh06_Wr	W	0	3	14	12	Writes PHA ZS threshold for channel 06
57	PHA_Thresh06_Rd	R	0	3	14	12	Reads back PHA ZS threshold register for channel 06
58	PHA_Thresh07_Wr	W	0	4	8	12	Writes PHA ZS threshold for channel 07
59	PHA_Thresh07_Rd	R	0	4	8	12	Reads back PHA ZS threshold register for channel 07
60	PHA_Thresh08_Wr	W	0	4	9	12	Writes PHA ZS threshold for channel 08
61	PHA_Thresh08_Rd	R	0	4	9	12	Reads back PHA ZS threshold register for channel 08
62	PHA_Thresh09_Wr	W	0	4	10	12	Writes PHA ZS threshold for channel 09
63	PHA_Thresh09_Rd	R	0	4	10	12	Reads back PHA ZS threshold register

							for channel 09
64	PHA_Thresh10_Wr	W	0	4	11	12	Writes PHA ZS threshold for channel 10
65	PHA_Thresh10_Rd	R	0	4	11	12	Reads back PHA ZS threshold register for channel 10
66	PHA_Thresh11_Wr	W	0	4	12	12	Writes PHA ZS threshold for channel 11
67	PHA_Thresh11_Rd	R	0	4	12	12	Reads back PHA ZS threshold register for channel 11
68	PHA_Thresh12_Wr	W	0	4	13	12	Writes PHA ZS threshold for channel 12
69	PHA_Thresh12_Rd	R	0	4	13	12	Reads back PHA ZS threshold register for channel 12
70	PHA_Thresh13_Wr	W	0	4	14	12	Writes PHA ZS threshold for channel 13
71	PHA_Thresh13_Rd	R	0	4	14	12	Reads back PHA ZS threshold register for channel 13
72	PHA_Thresh14_Wr	W	0	5	8	12	Writes PHA ZS threshold for channel 14
73	PHA_Thresh14_Rd	R	0	5	8	12	Reads back PHA ZS threshold register for channel 14
74	PHA_Thresh15_Wr	W	0	5	9	12	Writes PHA ZS threshold for channel 15
75	PHA_Thresh15_Rd	R	0	5	9	12	Reads back PHA ZS threshold register for channel 15
76	PHA_Thresh16_Wr	W	0	5	10	12	Writes PHA ZS threshold for channel 16
77	PHA_Thresh16_Rd	R	0	5	10	12	Reads back PHA ZS threshold register for channel 16
78	PHA_Thresh17_Wr	W	0	5	11	12	Writes PHA ZS threshold for channel 17
79	PHA_Thresh17_Rd	R	0	5	11	12	Reads back PHA ZS threshold register for channel 17
80	ADC_TACQ_Wr	W	0	5	12	6	Sets ADC Acquisition Time from Hold to Start of Conversion
81	ADC_TACQ_Rd	R	0	5	12	6	Reads back contents of the ADC_TACQ register
82	GAFE_Mode_Wr	W	1	GAFE Addr	0	16	Writes the GAFE mode register for the ASIC addressed
83	GAFE_Mode_Rd	R	1	GAFE Addr	0	16	Reads the GAFE mode register contents for the ASIC addressed
84	GAFE_DAC1_Wr	W	1	GAFE Addr	1	6	Writes the DAC1 register in the GAFE addressed
85	GAFE_DAC1_Rd	R	1	GAFE Addr	1	6	Reads back the contents of the DAC1 register in the addressed GAFE
86	GAFE_DAC2_Wr	W	1	GAFE Addr	2	6	Writes the DAC2 register in the GAFE addressed
87	GAFE_DAC2_Rd	R	1	GAFE Addr	2	6	Reads back the contents of the DAC2 register in the addressed GAFE
88	GAFE_DAC3_Wr	W	1	GAFE Addr	3	6	Writes the DAC3 register in the GAFE addressed
89	GAFE_DAC3_Rd	R	1	GAFE Addr	3	6	Reads back the contents of the DAC3 register in the addressed GAFE
90	GAFE_DAC4_Wr	W	1	GAFE Addr	4	6	Writes the DAC4 register in the GAFE addressed
91	GAFE_DAC4_Rd	R	1	GAFE Addr	4	6	Reads back the contents of the DAC4 register in the addressed GAFE
92	GAFE_DAC5_Wr	W	1	GAFE Addr	5	6	Writes the DAC5 register in the GAFE addressed
93	GAFE_DAC5_Rd	R	1	GAFE Addr	5	6	Reads back the contents of the DAC5 register in the addressed GAFE
94	GAFE_Version	R	1	GAFE Addr	6	3	Reads back the GAFE ASIC version
95	GAFE_Write_Ctr	R	1	GAFE Addr	7	8	Reads back the contents of the GAFE write counter register
96	GAFE_Reject_Ctr	R	1	GAFE Addr	8	8	Reads back the contents of the GAFE command reject register
97	GAFE_Cmd_Ctr	R	1	GAFE Addr	9	8	Reads back the contents of the GAFE command counter
98	GAFE_Chip_Addr	R	1	GAFE Addr	10	5	Reads back the hardwired address of a GAFE ASIC

#### 4.0 Testing the GARC Digital Logic

The GARC logic is based on a command-response protocol and requires an AEM or AEM simulator to access the logic functions. For each of the following commands, the proper GARC and/or GAFE response may be tested. Note that all GAFEs will respond to a broadcast write command (e.g., GAFE address of decimal 31), but a read command requires a unique GAFE address. The following test details the proper sequence for a single GARC ASIC. The steps are numbered for easier reference. The test setup required is detailed in the diagram below.



### **GARC ASIC FUNCTIONAL TEST SETUP**

## 5.0 Initial Reset Test

This section will verify that GARC registers have been properly initialized during a reset command. The following test sequence of commands will perform this verification. This test will also verify that the GARC to AEM command link is functional.

1. Send the GARC\_Reset command.
2. Send the Trigger\_ZS command (this captures the FREE board ID).
3. Send the Veto\_Delay\_Rd command. The data field in the return data stream should be 5.
4. Send the HVBS\_Level\_Rd command. The data field in the return data stream should be 0.
5. Send the SAA\_Level\_Rd command. The data field in the return data stream should be 0.
6. Send the Hold\_Delay\_Rd command. The data field in the return data stream should be 28.
7. Send the Veto\_Width\_Rd command. The data field in the return data stream should be 2.
8. Send the HitMap\_Width\_Rd command. The data field in the return data stream should be 7.
9. Send the HitMap\_Deadtme\_Rd command. The data field in the return data stream should be 0.
10. Send the HitMap\_Delay\_Rd command. The data field in the return data stream should be 16.
11. Send the PHA\_En0\_Rd command. The data field in the return data stream should be 65535.
12. Send the PHA\_En1\_Rd command. The data field in the return data stream should be 3.
13. Send the Veto\_En0\_Rd command. The data field in the return data stream should be 65535.
14. Send the Veto\_En1\_Rd command. The data field in the return data stream should be 3.
15. Send the Max\_PHA\_Rd command. The data field in the return data stream should be 4.
16. Send the GARC\_Mode\_Rd command. The data field in the return data stream should be 768.
17. Send the GARC\_Status command. The data field in the return data stream should be 24.
18. Send the GARC\_Cmd\_Reg command. The data field in the return data stream should be 0.
19. Send the GARC\_Cmd\_Rejects command. The data field in the return data stream should be 0.
20. Send the FREE\_Board\_ID command. The data field in the return data stream should be the same as the FREE board serial number.
21. Send the GARC\_Version command. The data field in the return data stream should be 1 for GARC V1 (e.g., version #1, May 2002).
22. Send the PHA\_Thresh00\_Rd command. The data field in the return data stream should be 1114.
23. Send the PHA\_Thresh01\_Rd command. The data field in the return data stream should be 1114.
24. Send the PHA\_Thresh02\_Rd command. The data field in the return data stream should be 1114.
25. Send the PHA\_Thresh03\_Rd command. The data field in the return data stream should be 1114.
26. Send the PHA\_Thresh04\_Rd command. The data field in the return data stream should be 1114.
27. Send the PHA\_Thresh05\_Rd command. The data field in the return data stream should be 1114.
28. Send the PHA\_Thresh06\_Rd command. The data field in the return data stream should be 1114.
29. Send the PHA\_Thresh07\_Rd command. The data field in the return data stream should be 1114.
30. Send the PHA\_Thresh08\_Rd command. The data field in the return data stream should be 1114.
31. Send the PHA\_Thresh09\_Rd command. The data field in the return data stream should be 1114.

32. Send the PHA\_Thresh10\_Rd command. The data field in the return data stream should be 1114.
33. Send the PHA\_Thresh11\_Rd command. The data field in the return data stream should be 1114.
34. Send the PHA\_Thresh12\_Rd command. The data field in the return data stream should be 1114.
35. Send the PHA\_Thresh13\_Rd command. The data field in the return data stream should be 1114.
36. Send the PHA\_Thresh14\_Rd command. The data field in the return data stream should be 1114.
37. Send the PHA\_Thresh15\_Rd command. The data field in the return data stream should be 1114.
38. Send the PHA\_Thresh16\_Rd command. The data field in the return data stream should be 1114.
39. Send the PHA\_Thresh17\_Rd command. The data field in the return data stream should be 1114.
40. Send the ADC\_TACQ\_Rd command. The data field in the return data stream should be 0.
41. Initiate an external GARC Power On Reset by placing a 0V to 3.3V to 0V pulse approximately 100 usec wide (or longer) on the GARC external reset pin, pin 182.
42. Verify the status of the GARC registers as shown in steps 2 – 40 above.

If all steps above have verified correctly, the GARC reset function has been verified. A summary of the initial reset parameters for GARC and GAFE is detailed below.

Register	Initial Value (decimal)	Initial Value (hex)
Veto_Delay	5	5
HVBS_Level	0	0
SAA_Level	0	0
Hold_Delay	28	1C
Veto_Width	2	2
HitMap_Width	7	7
HitMap_Deptime	3	3
HitMap_Delay	16	10
PHA_EN0	65535	FFFF
PHA_EN1	3	3
VETO_EN0	65535	FFFF
VETO_EN1	3	3
Max_PHA	4	4
GARC_Mode	768	300
GARC_Status	24	18
Command_Register	0	0
GARC_Diagnostic	0	0
Cmd_Reject_Ctr	0	0
FREE_Board_ID	*	*
GARC_Version	1	1
PHA_Threshold_00	1114	45A
PHA_Threshold_01	1114	45A
PHA_Threshold_02	1114	45A
PHA_Threshold_03	1114	45A
PHA_Threshold_04	1114	45A
PHA_Threshold_05	1114	45A
PHA_Threshold_06	1114	45A
PHA_Threshold_07	1114	45A
PHA_Threshold_08	1114	45A
PHA_Threshold_09	1114	45A
PHA_Threshold_10	1114	45A
PHA_Threshold_11	1114	45A

PHA_Threshold_12	1114	45A
PHA_Threshold_13	1114	45A
PHA_Threshold_14	1114	45A
PHA_Threshold_15	1114	45A
PHA_Threshold_16	1114	45A
PHA_Threshold_17	1114	45A
ADC_TACQ	0	0
GAFE_Mode	48	30
GAFE_DAC1	57	39
GAFE_DAC2	38	26
GAFE_DAC3	55	37
GAFE_DAC4	32	20
GAFE_DAC5	0	0
GAFE_Wr_Ctr	0	0
GAFE_Reject_Ctr	0	0

## 6.0 GARC Power Measurements

### 6.1 Measurement at the Nominal Power Supply Voltage

Verify that the GARC power supply is set to +3.30V. After initial power up, measure the +3.30V power supply current to the GARC in the following modes. Record these values in the table below.

1. Send the GARC\_Mode\_Wr command with a data argument of 768. Send the GARC\_Mode\_Rd command to verify. Both the primary and secondary LVDS VETO drivers should be enabled. Record the GARC current in the table below.
2. Send the GARC\_Mode\_Wr command with a data argument of 256. Send the GARC\_Mode\_Rd command to verify. Only the primary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
3. Send the GARC\_Mode\_Wr command with a data argument of 512. Send the GARC\_Mode\_Rd command to verify. Only the secondary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
4. Send the GARC\_Mode\_Wr command with a data argument of 0. Send the GARC\_Mode\_Rd command to verify. None of the LVDS VETO drivers should now be enabled. Record the GARC current in the table below.

GARC Mode	GARC_Mode_Wr Data Argument	+3.3V Current Measured (mA)	+3.3V Current Expected (mA)
LVDS "A" Drivers Enabled LVDS "B" Drivers Enabled	768		
LVDS "A" Drivers Enabled LVDS "B" Drivers Disabled	256		
LVDS "A" Drivers Disabled LVDS "B" Drivers Enabled	512		
LVDS "A" Drivers Disabled LVDS "B" Drivers Disabled	0		

### 6.2 Measurement at the Minimum Power Supply Voltage

Verify that the GARC power supply is set to +3.0V. After initial power up, measure the +3.0V power supply current to the GARC in the following modes. Record these values in the table below.

1. Send the GARC\_Mode\_Wr command with a data argument of 768. Send the GARC\_Mode\_Rd command to verify. Both the primary and secondary LVDS VETO drivers should be enabled. Record the GARC current in the table below.
2. Send the GARC\_Mode\_Wr command with a data argument of 256. Send the GARC\_Mode\_Rd command to verify. Only the primary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
3. Send the GARC\_Mode\_Wr command with a data argument of 512. Send the GARC\_Mode\_Rd command to verify. Only the secondary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
4. Send the GARC\_Mode\_Wr command with a data argument of 0. Send the GARC\_Mode\_Rd command to verify. None of the LVDS VETO drivers should now be enabled. Record the GARC current in the table below.

GARC Mode	GARC_Mode_Wr Data Argument	+3.0V Current Measured (mA)	+3.0V Current Expected (mA)
LVDS "A" Drivers Enabled LVDS "B" Drivers Enabled	768		
LVDS "A" Drivers Enabled LVDS "B" Drivers Disabled	256		
LVDS "A" Drivers Disabled LVDS "B" Drivers Enabled	512		
LVDS "A" Drivers Disabled LVDS "B" Drivers Disabled	0		

### 6.3 Measurement at the Maximum Power Supply Voltage

Verify that the GARC power supply is set to +3.6V. After initial power up, measure the +3.6V power supply current to the GARC in the following modes. Record these values in the table below.

1. Send the GARC\_Mode\_Wr command with a data argument of 768. Send the GARC\_Mode\_Rd command to verify. Both the primary and secondary LVDS VETO drivers should be enabled. Record the GARC current in the table below.
2. Send the GARC\_Mode\_Wr command with a data argument of 256. Send the GARC\_Mode\_Rd command to verify. Only the primary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
3. Send the GARC\_Mode\_Wr command with a data argument of 512. Send the GARC\_Mode\_Rd command to verify. Only the secondary LVDS VETO drivers should now be enabled. Record the GARC current in the table below.
4. Send the GARC\_Mode\_Wr command with a data argument of 0. Send the GARC\_Mode\_Rd command to verify. None of the LVDS VETO drivers should now be enabled. Record the GARC current in the table below.

GARC Mode	GARC_Mode_Wr Data Argument	+3.6V Current Measured (mA)	+3.6V Current Expected (mA)
LVDS "A" Drivers Enabled LVDS "B" Drivers Enabled	768		
LVDS "A" Drivers Enabled LVDS "B" Drivers Disabled	256		
LVDS "A" Drivers Disabled LVDS "B" Drivers Enabled	512		
LVDS "A" Drivers Disabled LVDS "B" Drivers Disabled	0		

5. Reset the GARC power supply to +3.3V.
6. Send the GARC\_Reset command to return the GARC to the initial power-on configuration.

#### 6.4 Measurement of the GARC Bias Resistor Voltages

Verify that the GARC power supply is set to +3.3V. Using the GLAST GARC Test Board (5/2002), measure the following bias voltages on the resistors indicated.

GARC Bias Signal	GARC Pin	Resistor Test Point	Expected Voltage	Measured Voltage
HLD_WOR_BIAS	104	R19		
BIAS_RCVR	156	R16		
BIAS_DRV_H	160	R15		
BIAS_DRV_L	169	R13		
LVDS_PRESET_ADJ	184	R20		

#### 7.0 GARC Register Read/Write Tests

This section tests the proper functioning of each bit of the commandable registers in the GARC. The intent is to toggle each bit in a variety of patterns to ensure that all bits are addressable and that there is no stuck-at-fault condition. The following GARC registers will be tested in this section:

GARC Register Name	Register Width (bits)	Register Type
Veto Delay	5	Read/Write
HVBS Level	12	Read/Write
SAA Level	12	Read/Write
Hold Delay	7	Read/Write
Veto Width	3	Read/Write
HitMap Width	4	Read/Write
HitMap Deadtime	3	Read/Write
HitMap Delay	5	Read/Write
PHA En0	16	Read/Write
PHA En1	2	Read/Write
VETO En0	16	Read/Write
VETO En1	2	Read/Write
Max PHA	5	Read/Write
GARC Mode	11	Read/Write

GARC Status	6	Read Only
Command Register	16	Read Only
GARC Diagnostic	16	Read Only
Cmd Reject Counter	8	Read Only
FREE Board ID	8	Read Only
GARC Version	3	Read Only
PHA Threshold 00	16	Read/Write
PHA Threshold 01	16	Read/Write
PHA Threshold 02	16	Read/Write
PHA Threshold 03	16	Read/Write
PHA Threshold 04	16	Read/Write
PHA Threshold 05	16	Read/Write
PHA Threshold 06	16	Read/Write
PHA Threshold 07	16	Read/Write
PHA Threshold 08	16	Read/Write
PHA Threshold 09	16	Read/Write
PHA Threshold 10	16	Read/Write
PHA Threshold 11	16	Read/Write
PHA Threshold 12	16	Read/Write
PHA Threshold 13	16	Read/Write
PHA Threshold 14	16	Read/Write
PHA Threshold 15	16	Read/Write
PHA Threshold 16	16	Read/Write
PHA Threshold 17	16	Read/Write
ADC TACQ	6	Read/Write

### 7.1 Veto Delay Register Test

1. Send the Veto\_Delay\_Wr command with a data argument of 5'h0 (0). Send the Veto\_Delay\_Rd command and read back this commanded data argument.
2. Send the Veto\_Delay\_Wr command with a data argument of 5'h15 (21). Send the Veto\_Delay\_Rd command and read back this commanded data argument.
3. Send the Veto\_Delay\_Wr command with a data argument of 5'h0A (10). Send the Veto\_Delay\_Rd command and read back this commanded data argument.
4. Send the Veto\_Delay\_Wr command with a data argument of 5'h1F (31). Send the Veto\_Delay\_Rd command and read back this commanded data argument.
5. Send the Veto\_Delay\_Wr command with a data argument of 5'h5 (5). Send the Veto\_Delay\_Rd command and read back this commanded data argument.

### 7.2 HVBS Level Register Test

1. Send the HVBS\_Level\_Wr command with a data argument of 12'h0 (0). Send the HVBS\_Level\_Rd command and read back this commanded data argument.
2. Send the HVBS\_Level\_Wr command with a data argument of 12'h555 (1365). Send the HVBS\_Level\_Rd command and read back this commanded data argument.
3. Send the HVBS\_Level\_Wr command with a data argument of 12'hAAA (2730). Send the HVBS\_Level\_Rd command and read back this commanded data argument.
4. Send the HVBS\_Level\_Wr command with a data argument of 12'hFFF (4095). Send the HVBS\_Level\_Rd command and read back this commanded data argument.

5. Send the HVBS\_Level\_Wr command with a data argument of 12'h0 (0). Send the HVBS\_Level\_Rd command and read back this commanded data argument.

### **7.3 SAA Level Register Test**

1. Send the SAA\_Level\_Wr command with a data argument of 12'h0 (0). Send the SAA\_Level\_Rd command and read back this commanded data argument.
2. Send the SAA\_Level\_Wr command with a data argument of 12'h555 (1365). Send the SAA\_Level\_Rd command and read back this commanded data argument.
3. Send the SAA\_Level\_Wr command with a data argument of 12'hAAA (2730). Send the SAA\_Level\_Rd command and read back this commanded data argument.
4. Send the SAA\_Level\_Wr command with a data argument of 12'hFFF (4095). Send the SAA\_Level\_Rd command and read back this commanded data argument.
5. Send the SAA\_Level\_Wr command with a data argument of 12'h0 (0). Send the SAA\_Level\_Rd command and read back this commanded data argument.

### **7.4 Hold Delay Register Test**

1. Send the Hold\_Delay\_Wr command with a data argument of 7'h0 (0). Send the Hold\_Delay\_Rd command and read back this commanded data argument.
2. Send the Hold\_Delay\_Wr command with a data argument of 7'h55 (85). Send the Hold\_Delay\_Rd command and read back this commanded data argument.
3. Send the Hold\_Delay\_Wr command with a data argument of 7'h2A (42). Send the Hold\_Delay\_Rd command and read back this commanded data argument.
4. Send the Hold\_Delay\_Wr command with a data argument of 7'h7F (127). Send the Hold\_Delay\_Rd command and read back this commanded data argument.
5. Send the Hold\_Delay\_Wr command with a data argument of 7'h1C (28). Send the Hold\_Delay\_Rd command and read back this commanded data argument.

### **7.5 Veto Width Register Test**

1. Send the Veto\_Width\_Wr command with a data argument of 3'h0 (0). Send the Veto\_Width\_Rd command and read back this commanded data argument.
2. Send the Veto\_Width\_Wr command with a data argument of 3'h5 (5). Send the Veto\_Width\_Rd command and read back this commanded data argument.
3. Send the Veto\_Width\_Wr command with a data argument of 3'h7 (7). Send the Veto\_Width\_Rd command and read back this commanded data argument.
4. Send the Veto\_Width\_Wr command with a data argument of 3'h2 (2). Send the Veto\_Width\_Rd command and read back this commanded data argument.

### **7.6 HitMap Width Register Test**

1. Send the HitMap\_Width\_Wr command with a data argument of 4'h0 (0). Send the HitMap\_Width\_Rd command and read back this commanded data argument.

2. Send the HitMap\_Width\_Wr command with a data argument of 4'h5 (5). Send the HitMap\_Width\_Rd command and read back this commanded data argument.
3. Send the HitMap\_Width\_Wr command with a data argument of 4'hA (10). Send the HitMap\_Width\_Rd command and read back this commanded data argument.
4. Send the HitMap\_Width\_Wr command with a data argument of 4'hF (15). Send the HitMap\_Width\_Rd command and read back this commanded data argument.
5. Send the HitMap\_Width\_Wr command with a data argument of 4'h7 (7). Send the HitMap\_Width\_Rd command and read back this commanded data argument.

### **7.7 HitMap Deadtime Register Test**

1. Send the HitMap\_Deadtime\_Wr command with a data argument of 3'h0 (0). Send the HitMap\_Deadtime\_Rd command and read back this commanded data argument.
2. Send the HitMap\_Deadtime\_Wr command with a data argument of 3'h5 (5). Send the HitMap\_Deadtime\_Rd command and read back this commanded data argument.
3. Send the HitMap\_Deadtime\_Wr command with a data argument of 3'h2 (2). Send the HitMap\_Deadtime\_Rd command and read back this commanded data argument.
4. Send the HitMap\_Deadtime\_Wr command with a data argument of 3'h7 (7). Send the HitMap\_Deadtime\_Rd command and read back this commanded data argument.
5. Send the HitMap\_Deadtime\_Wr command with a data argument of 3'h3 (3). Send the HitMap\_Deadtime\_Rd command and read back this commanded data argument.

### **7.8 HitMap Delay Register Test**

1. Send the HitMap\_Delay\_Wr command with a data argument of 5'h0 (0). Send the HitMap\_Delay\_Rd command and read back this commanded data argument.
2. Send the HitMap\_Delay\_Wr command with a data argument of 5'h15 (21). Send the HitMap\_Delay\_Rd command and read back this commanded data argument.
3. Send the HitMap\_Delay\_Wr command with a data argument of 5'h0A (10). Send the HitMap\_Delay\_Rd command and read back this commanded data argument.
4. Send the HitMap\_Delay\_Wr command with a data argument of 5'h1F (31). Send the HitMap\_Delay\_Rd command and read back this commanded data argument.
5. Send the HitMap\_Delay\_Wr command with a data argument of 5'h10 (16). Send the HitMap\_Delay\_Rd command and read back this commanded data argument.

### **7.9 PHA En0 Register Test**

1. Send the PHA\_En0\_Wr command with a data argument of 16'h0 (0). Send the PHA\_En0\_Rd command and read back this commanded data argument.
2. Send the PHA\_En0\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_En0\_Rd command and read back this commanded data argument.
3. Send the PHA\_En0\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_En0\_Rd command and read back this commanded data argument.
4. Send the PHA\_En0\_Wr command with a data argument of 16'hFFFF (65535). Send the PHA\_En0\_Rd command and read back this commanded data argument.

### **7.10 PHA En1 Register Test**

1. Send the PHA\_En1\_Wr command with a data argument of 2'h0 (0). Send the PHA\_En1\_Rd command and read back this commanded data argument.
2. Send the PHA\_En1\_Wr command with a data argument of 2'h1 (1). Send the PHA\_En1\_Rd command and read back this commanded data argument.
3. Send the PHA\_En1\_Wr command with a data argument of 2'h2 (2). Send the PHA\_En1\_Rd command and read back this commanded data argument.
4. Send the PHA\_En1\_Wr command with a data argument of 2'h3 (3). Send the PHA\_En1\_Rd command and read back this commanded data argument.

### **7.11 Veto En0 Register Test**

1. Send the VETO\_En0\_Wr command with a data argument of 16'h0 (0). Send the VETO\_En0\_Rd command and read back this commanded data argument.
2. Send the VETO\_En0\_Wr command with a data argument of 16'h5555 (21845). Send the VETO\_En0\_Rd command and read back this commanded data argument.
3. Send the VETO\_En0\_Wr command with a data argument of 16'hAAAA (43690). Send the VETO\_En0\_Rd command and read back this commanded data argument.
4. Send the VETO\_En0\_Wr command with a data argument of 16'hFFFF (65535). Send the VETO\_En0\_Rd command and read back this commanded data argument.

### **7.12 Veto En1 Register Test**

1. Send the VETO\_En1\_Wr command with a data argument of 2'h0 (0). Send the VETO\_En1\_Rd command and read back this commanded data argument.
2. Send the VETO\_En1\_Wr command with a data argument of 2'h1 (1). Send the VETO\_En1\_Rd command and read back this commanded data argument.
3. Send the VETO\_En1\_Wr command with a data argument of 2'h2 (2). Send the VETO\_En1\_Rd command and read back this commanded data argument.
4. Send the VETO\_En1\_Wr command with a data argument of 2'h3 (3). Send the VETO\_En1\_Rd command and read back this commanded data argument.

### **7.13 MaxPHA Register Test**

1. Send the MaxPHA\_Wr command with a data argument of 5'h0 (0). Send the MaxPHA\_Rd command and read back this commanded data argument.
2. Send the MaxPHA\_Wr command with a data argument of 5'h15 (21). Send the MaxPHA\_Rd command and read back this commanded data argument.
3. Send the MaxPHA\_Wr command with a data argument of 5'h0A (10). Send the MaxPHA\_Rd command and read back this commanded data argument.
4. Send the MaxPHA\_Wr command with a data argument of 5'h1F (31). Send the MaxPHA\_Rd command and read back this commanded data argument.

5. Send the MaxPHA\_Wr command with a data argument of 5'h4 (4). Send the MaxPHA\_Rd command and read back this commanded data argument.

#### **7.14 GARC Mode Register Test**

**\*\* Note this sequence must be followed exactly to preclude placing the GARC into the undesired mode of sending return data with incorrect parity.**

1. Send the GARC\_Mode\_Wr command with a data argument of 11'h2 (2). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
2. Send the GARC\_Mode\_Wr command with a data argument of 11'h4 (4). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
3. Send the GARC\_Mode\_Wr command with a data argument of 11'h8 (8). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
4. Send the GARC\_Mode\_Wr command with a data argument of 11'h10 (16). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
5. Send the GARC\_Mode\_Wr command with a data argument of 11'h20 (32). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
6. Send the GARC\_Mode\_Wr command with a data argument of 11'h40 (64). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
7. Send the GARC\_Mode\_Wr command with a data argument of 11'h80 (128). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
8. Send the GARC\_Mode\_Wr command with a data argument of 11'h100 (256). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
9. Send the GARC\_Mode\_Wr command with a data argument of 11'h200 (512). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
10. Send the GARC\_Mode\_Wr command with a data argument of 11'h400 (1024). Send the GARC\_Mode\_Rd command and read back this commanded data argument.
11. Send the GARC\_Mode\_Wr command with a data argument of 11'h300 (768). Send the GARC\_Mode\_Rd command and read back this commanded data argument.

#### **7.15 PHA Threshold Channel 00 Register Test**

1. Send the PHA\_Thresh00\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh00\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh00\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh00\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh00\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh00\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh00\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh00\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh00\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh00\_Rd command and read back this commanded data argument.

### **7.16 PHA Threshold Channel 01 Register Test**

1. Send the PHA\_Thresh01\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh01\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh01\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh01\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh01\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh01\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh01\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh01\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh01\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh01\_Rd command and read back this commanded data argument.

### **7.17 PHA Threshold Channel 02 Register Test**

1. Send the PHA\_Thresh02\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh02\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh02\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh02\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh02\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh02\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh02\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh02\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh02\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh02\_Rd command and read back this commanded data argument.

### **7.18 PHA Threshold Channel 03 Register Test**

1. Send the PHA\_Thresh03\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh03\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh03\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh03\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh03\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh03\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh03\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh03\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh03\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh03\_Rd command and read back this commanded data argument.

### **7.19 PHA Threshold Channel 04 Register Test**

1. Send the PHA\_Thresh04\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh04\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh04\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh04\_Rd command and read back this commanded data argument.

3. Send the PHA\_Thresh04\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh04\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh04\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh04\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh04\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh04\_Rd command and read back this commanded data argument.

#### **7.20 PHA Threshold Channel 05 Register Test**

1. Send the PHA\_Thresh05\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh05\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh05\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh05\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh05\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh05\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh05\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh05\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh05\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh05\_Rd command and read back this commanded data argument.

#### **7.21 PHA Threshold Channel 06 Register Test**

1. Send the PHA\_Thresh06\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh06\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh06\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh06\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh06\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh06\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh06\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh06\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh06\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh06\_Rd command and read back this commanded data argument.

#### **7.22 PHA Threshold Channel 07 Register Test**

1. Send the PHA\_Thresh07\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh07\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh07\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh07\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh07\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh07\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh07\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh07\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh07\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh07\_Rd command and read back this commanded data argument.

### **7.23 PHA Threshold Channel 08 Register Test**

1. Send the PHA\_Thresh08\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh08\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh08\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh08\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh08\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh08\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh08\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh08\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh08\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh08\_Rd command and read back this commanded data argument.

### **7.24 PHA Threshold Channel 09 Register Test**

1. Send the PHA\_Thresh09\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh09\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh09\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh09\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh09\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh09\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh09\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh09\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh09\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh09\_Rd command and read back this commanded data argument.

### **7.25 PHA Threshold Channel 10 Register Test**

1. Send the PHA\_Thresh10\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh10\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh10\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh10\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh10\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh10\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh10\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh10\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh10\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh10\_Rd command and read back this commanded data argument.

### **7.26 PHA Threshold Channel 11 Register Test**

1. Send the PHA\_Thresh11\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh11\_Rd command and read back this commanded data argument.

2. Send the PHA\_Thresh11\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh11\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh11\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh11\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh11\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh11\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh11\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh11\_Rd command and read back this commanded data argument.

#### **7.27 PHA Threshold Channel 12 Register Test**

1. Send the PHA\_Thresh12\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh12\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh12\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh12\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh12\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh12\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh12\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh12\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh12\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh12\_Rd command and read back this commanded data argument.

#### **7.28 PHA Threshold Channel 13 Register Test**

1. Send the PHA\_Thresh13\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh13\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh13\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh13\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh13\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh13\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh13\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh13\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh13\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh13\_Rd command and read back this commanded data argument.

#### **7.29 PHA Threshold Channel 14 Register Test**

1. Send the PHA\_Thresh14\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh14\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh14\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh14\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh14\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh14\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh14\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh14\_Rd command and read back this commanded data argument.

5. Send the PHA\_Thresh14\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh14\_Rd command and read back this commanded data argument.

### **7.30 PHA Threshold Channel 15 Register Test**

1. Send the PHA\_Thresh15\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh15\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh15\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh15\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh15\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh15\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh15\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh15\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh15\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh15\_Rd command and read back this commanded data argument.

### **7.31 PHA Threshold Channel 16 Register Test**

1. Send the PHA\_Thresh16\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh16\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh16\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh16\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh16\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh16\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh16\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh16\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh16\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh16\_Rd command and read back this commanded data argument.

### **7.32 PHA Threshold Channel 17 Register Test**

1. Send the PHA\_Thresh17\_Wr command with a data argument of 16'h0 (0). Send the PHA\_Thresh17\_Rd command and read back this commanded data argument.
2. Send the PHA\_Thresh17\_Wr command with a data argument of 16'h5555 (21845). Send the PHA\_Thresh17\_Rd command and read back this commanded data argument.
3. Send the PHA\_Thresh17\_Wr command with a data argument of 16'hAAAA (43690). Send the PHA\_Thresh17\_Rd command and read back this commanded data argument.
4. Send the PHA\_Thresh17\_Wr command with a data argument of 16'hAAAA (65536). Send the PHA\_Thresh17\_Rd command and read back this commanded data argument.
5. Send the PHA\_Thresh17\_Wr command with a data argument of 16'h45A (1114). Send the PHA\_Thresh17\_Rd command and read back this commanded data argument.

### **7.33 ADC TACQ Register Test**

1. Send the ADC\_TACQ\_Wr command with a data argument of 6'h0 (0). Send the ADC\_TACQ\_Rd command and read back this commanded data argument.
2. Send the ADC\_TACQ\_Wr command with a data argument of 6'h15 (21). Send the ADC\_TACQ\_Rd command and read back this commanded data argument.
3. Send the ADC\_TACQ\_Wr command with a data argument of 6'h2A (42). Send the ADC\_TACQ\_Rd command and read back this commanded data argument.
4. Send the ADC\_TACQ\_Wr command with a data argument of 6'h3F (63). Send the ADC\_TACQ\_Rd command and read back this commanded data argument.
5. Send the ADC\_TACQ\_Wr command with a data argument of 6'h0 (0). Send the ADC\_TACQ\_Rd command and read back this commanded data argument.
6. Send the GARC\_Reset command to ensure all registers are at the proper initial value.

At the successful conclusion of this section, the GARC commandable register bits have been demonstrated to be functional with all bits toggling as commanded.

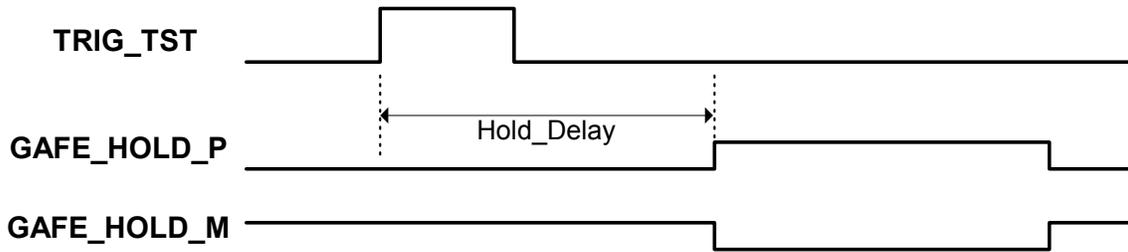
### **8.0 GARC Test Pin Mux Verification**

This section tests the proper functionality, switch the GARC test pin between the “Live” and “HitMap Test” outputs.

1. Send the GARC\_Mode\_Wr command with a data argument of 1792 to switch the GARC test pin mux to view the “live” signal. Send the GARC\_Mode\_Rd command to verify the register data. When configuration commands are sent, the “live” signal should go inactive low during the state machine busy time. This may be verified via the oscilloscope. The GARC test pin is pin 179.
2. Send the GARC\_Mode\_Wr command with a data argument of 768 to switch the GARC test pin mux to view the “HitMap test” signal. Send the GARC\_Mode\_Rd command to verify the register data. When trigger commands are sent, the “HitMap test” signal should go active high during times when there is a delayed VETO signal present at the DISC\_IN input. This may be verified via the oscilloscope. The GARC test pin is pin 179.
3. Send the GARC\_Reset command to ensure all registers are at the proper initial value.

### **9.0 Test of the Hold Delay Operation**

This section tests the proper functioning of the adjustment of the commandable delay function of the trigger to shaping amplifier hold signal. The trigger signal may be monitored on the GARC at pin 180, TRIG\_TST. The Hold signal may be monitored differentially on the GARC, GAFE\_HOLD\_P at pin 160 and GAFE\_HOLD\_M at pin 161, as shown in the diagram below.



### TRIGGER-TO-HOLD MEASUREMENT

1. Send the Hold\_Delay\_Wr command with a data argument of decimal 0 and verify with the Hold\_Delay\_Rd command. The Hold\_Delay is variable with arguments of 0 – 127, representing 250 – 6600 ns.

Hold Delay Command	Measured Hold Delay (ns)	Expected Hold Delay (ns)
0		250
1		300
2		350
3		400
4		450
5		500
6		550
7		600
8		650
9		700
10		750
11		800
12		850
13		900
14		950
15		1000
16		1050
17		1100
18		1150
19		1200
20		1250
21		1300
22		1350
23		1400
24		1450
25		1500
26		1550
27		1600
28		1650
29		1700
30		1750
31		1800
32		1850
33		1900
34		1950
35		2000
36		2050
37		2100

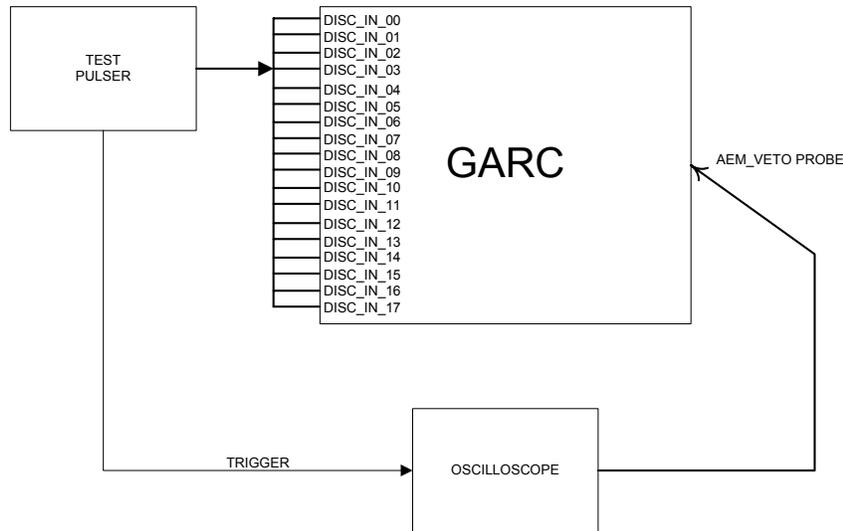
38		2150
39		2200
40		2250
41		2300
42		2350
43		2400
44		2450
45		2500
46		2550
47		2600
48		2650
49		2700
50		2750
51		2800
52		2850
53		2900
54		2950
55		3000
56		3050
57		3100
58		3150
59		3200
60		3250
61		3300
62		3350
63		3400
64		3450
65		3500
66		3550
67		3600
68		3650
69		3700
70		3750
71		3800
72		3850
73		3900
74		3950
75		4000
76		4050
77		4100
78		4150
79		4200
80		4250
81		4300
82		4350
83		4400
84		4450
85		4500
86		4550
87		4600
88		4650
89		4700
90		4750
91		4800
92		4850
93		4900

94		4950
95		5000
96		5050
97		5100
98		5150
99		5200
100		5250
101		5300
102		5350
103		5400
104		5450
105		5500
106		5550
107		5600
108		5650
109		5700
110		5750
111		5800
112		5850
113		5900
114		5950
115		6000
116		6050
117		6100
118		6150
119		6200
120		6250
121		6300
122		6350
123		6400
124		6450
125		6500
126		6550
127		6600

2. Send the HitMap\_Width\_Wr command with a data argument of 0. Verify the command using the HitMap\_Width\_Rd readback.
3. Send the GARC\_Reset command to restore all register values to their initial state.

### **10.0 Test of the AEM VETO Signal Functionality**

This section tests the proper functioning of the commandable functions of the AEM\_VETO signals. This test requires the use of an oscilloscope, a test pulser for input stimulus, and test points to monitor the AEM\_VETOs. The test setup is as shown below.

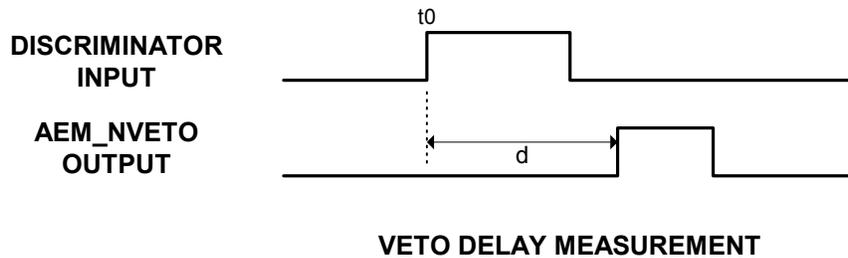


### GARC VETO Test Setup

All 18 of the discriminator inputs are tied together and also to the output of the test pulser for this portion of the test. The input test pulse should be approximately 200 ns wide and the rate should be set at approximately 20 kHz. The oscilloscope probe will be monitoring the 18 AEM\_VETO outputs. Between commands, this probe will be moved manually from test point to test point to perform this test. Setting the scope to acquire on the pulse output, this test will be measuring the delay (time from trigger to VETO leading edge) and width (time between leading and trailing edges) of each of the VETO pulses. Since each input has the same signal, the outputs should also be common in delay and width. In timing measurements, the activation signal from the test pulser will be the relative zero.

#### 10.1 Veto Delay Test

1. Send the Veto\_Delay\_Wr command with a data argument of 0. Confirm this command via the Veto\_Delay\_Rd command.
2. Send the Veto\_Width\_Wr command with a data argument of 0. Confirm this command via the Veto\_Width\_Rd command. Using the oscilloscope, measure the delay from scope trigger to leading edge of VETO for each of the 18 pulses and record the data in the table below. Note that all 18 pulses are designed to have identical timing for this test. The VETO Delay is the time **d** in the diagram below.



3. Repeat the Veto\_Delay\_Wr, Veto\_Delay\_Rd sequence for data argument values of decimal 1 to 31. Record the results in the table below. Note that there is a 50 ns jitter on each of the “expected” numbers.

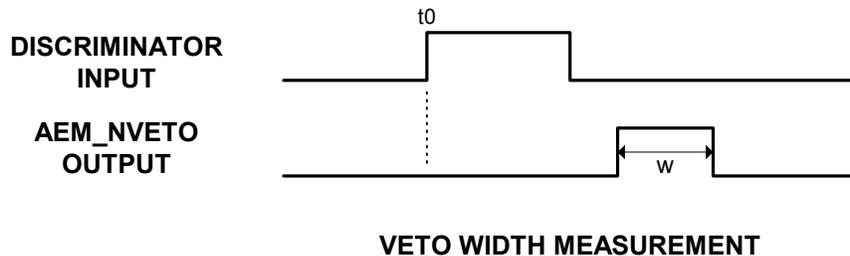
Veto_Delay Setting	Delay from Scope trigger to Leading Edge (ns)	Expected (ns)
0		150

1		200
2		250
3		300
4		350
5		400
6		450
7		500
8		550
9		600
10		650
11		700
12		750
13		800
14		850
15		900
16		950
17		1000
18		1050
19		1100
20		1150
21		1200
22		1250
23		1300
24		1350
25		1400
26		1450
27		1500
28		1550
29		1600
30		1650
31		1700

- Send the Veto\_Delay\_Wr command with a data argument of 0 and a Veto\_Delay\_Rd command to verify this value.

## 10.2 Veto Width Test

- Send the Veto\_Width\_Wr command with a data argument of 0. Confirm this command via the Veto\_Width\_Rd command. Using the oscilloscope, measure the width of each of the 18 VETO pulses and record the data in the table below. Note that the width of each of these pulses is expected to be identical for any given commanded setting. The VETO Width is the “w” parameter in the diagram below.



2. Repeat the Veto\_Width\_Wr, Veto\_Width\_Rd sequence for data argument values of decimal 1 to 7. Record the results in the table below.

Veto_Width	Measured Width of VETO Pulse (ns)	Expected Width (ns)
0		50
1		100
2		150
3		200
4		250
5		300
6		350
7		400

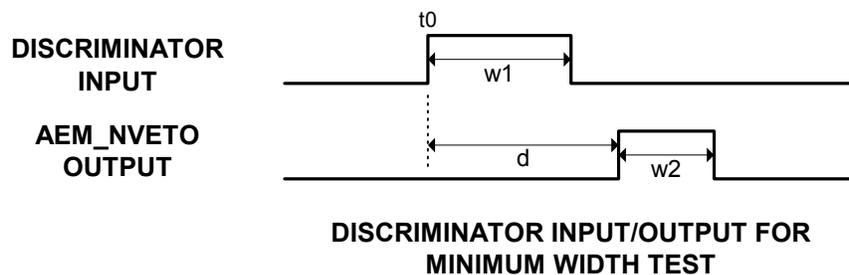
### 10.3 Veto Enable and Disable Test

1. Send the VETO\_En0\_Rd command and verify that the return data has value 'hFFFF (65535). Send the VETO\_En1\_Rd command and verify that the return data has the value 3.
2. Send the VETO\_En0\_Wr command with a data argument 'hFFFE (65534). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 0.
3. Send the VETO\_En0\_Wr command with a data argument 'hFFFD (65533). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 1.
4. Send the VETO\_En0\_Wr command with a data argument 'hFFFB (65531). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 2.
5. Send the VETO\_En0\_Wr command with a data argument 'hFFF7 (65527). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 3.
6. Send the VETO\_En0\_Wr command with a data argument 'hFFEF (65519). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 4.
7. Send the VETO\_En0\_Wr command with a data argument 'hFFDF (65503). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 5.
8. Send the VETO\_En0\_Wr command with a data argument 'hFFBF (65471). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 6.
9. Send the VETO\_En0\_Wr command with a data argument 'hFF7F (65407). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 7.
10. Send the VETO\_En0\_Wr command with a data argument 'hFFEF (65279). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 8.
11. Send the VETO\_En0\_Wr command with a data argument 'hFCFF (64767). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 9.
12. Send the VETO\_En0\_Wr command with a data argument 'hFBFF (64511). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 10.
13. Send the VETO\_En0\_Wr command with a data argument 'hF7FF (63487). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 11.
14. Send the VETO\_En0\_Wr command with a data argument 'hEFFF (61439). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 12.
15. Send the VETO\_En0\_Wr command with a data argument 'hCFFF (53247). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 13.

16. Send the VETO\_En0\_Wr command with a data argument 'hBFFF (49151). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 14.
17. Send the VETO\_En0\_Wr command with a data argument 'h7FFF (32767). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on with the exception of channel 15.
18. Send the VETO\_En0\_Wr command with a data argument 'hFFFF (65535). Verify the command with the VETO\_En0\_Rd command. Verify that all VETOs are on.
19. Send the VETO\_En1\_Wr command with a data argument 2. Verify the command with the VETO\_En1\_Rd command. Verify that all VETOs are on with the exception of channel 16.
20. Send the VETO\_En1\_Wr command with a data argument 1. Verify the command with the VETO\_En1\_Rd command. Verify that all VETOs are on with the exception of channel 17.
21. Send the GARC\_Reset command to restore all register values to their initial state.

#### 10.4 Discriminator Input Minimum Width Test

This test verifies that the VETO processing circuitry meets the specifications detailed in the ICD for the input width. For this test, all discriminator inputs greater than 100 ns in width should be processed, those in the 50 ns to 100 ns range should be sometimes processed (due to jitter), and those inputs less than 50 ns should never be processed.

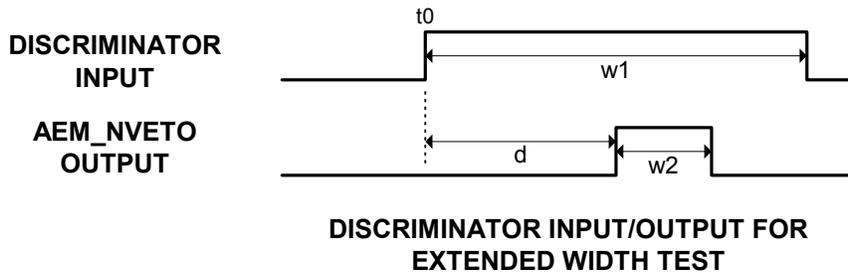


1. Send the VETO\_Delay\_Wr command with a data argument of 0. Verify the command with the VETO\_Delay\_Rd command. This fixes the delay (**d** in the figure) at 150 ns for this test.
2. Send the VETO\_Width\_Wr command with a data argument of 0. Verify the command with the VETO\_Width\_Rd command. This fixes the output width (**w2** in the figure) at 50 ns for this test.
3. Set the discriminator input on channel 00 to be a pulse of greater than 300 ns in width (**w1** in the figure). Verify the ACD\_NVETO\_00 signal out has a pulse output of 50 ns width.
4. Steadily decrease the width (**w1**) of the input pulse down to just greater than 100 ns in width. Verify that no pulses are missed. Adjust the width (**w1**) of the input pulse to between 50 ns and 100 ns in width. Verify that some pulses are there and some are missed due to the jitter in synchronization of the input signal. Adjust the input pulse width (**w1**) to just less than 50 ns and verify that all input pulses are rejected (i.e., no pulses output at NVETO).
5. Repeat this test for channel 01 and verify.
6. Repeat this test for channel 02 and verify.
7. Repeat this test for channel 03 and verify.
8. Repeat this test for channel 04 and verify.
9. Repeat this test for channel 05 and verify.
10. Repeat this test for channel 06 and verify.

11. Repeat this test for channel 07 and verify.
12. Repeat this test for channel 08 and verify.
13. Repeat this test for channel 09 and verify.
14. Repeat this test for channel 10 and verify.
15. Repeat this test for channel 11 and verify.
16. Repeat this test for channel 12 and verify.
17. Repeat this test for channel 13 and verify.
18. Repeat this test for channel 14 and verify.
19. Repeat this test for channel 15 and verify.
20. Repeat this test for channel 16 and verify.
21. Repeat this test for channel 17 and verify.
22. Send the GARC\_Reset command to restore all register values to their initial state.

### 10.5 Discriminator Input Extended Width Test

This test verifies that the VETO processing circuitry meets the specifications detailed in the ICD for extended input widths. For this test, discriminator inputs greater than

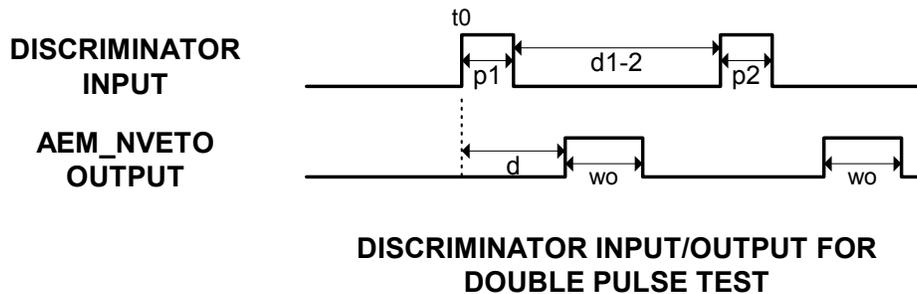


1. Send the Veto\_Delay\_Wr command with a data argument of 0. Verify with the Veto\_Delay\_Rd command.
2. Send the Veto\_Width\_Wr command with a data argument of 1. Verify with the Veto\_Width\_Rd command.
3. Connect the oscilloscope to monitor the discriminator input and the AEM\_NVETO\_00 output as shown in the diagram above. Start with the width of the input discriminator at approximately 150 ns. Verify that the Veto\_Delay is 150 ns and the Veto\_Width is 100 ns.
4. Increase the discriminator input from 150 ns to 10  $\mu$ s. Note that the output from the AEM\_NVETO\_00 signal does not change.
5. Repeat this test for channel 01 and verify.
6. Repeat this test for channel 02 and verify.
7. Repeat this test for channel 03 and verify.
8. Repeat this test for channel 04 and verify.
9. Repeat this test for channel 05 and verify.
10. Repeat this test for channel 06 and verify.

11. Repeat this test for channel 07 and verify.
12. Repeat this test for channel 08 and verify.
13. Repeat this test for channel 09 and verify.
14. Repeat this test for channel 10 and verify.
15. Repeat this test for channel 11 and verify.
16. Repeat this test for channel 12 and verify.
17. Repeat this test for channel 13 and verify.
18. Repeat this test for channel 14 and verify.
19. Repeat this test for channel 15 and verify.
20. Repeat this test for channel 16 and verify.
21. Repeat this test for channel 17 and verify.
22. Send the GARC\_Reset command to restore all register values to their initial state.

### 10.6 Discriminator Input Double Pulse Test

This test verifies that the VETO processing circuitry meets the specifications detailed in the ICD for extended input widths. For this test, multiple discriminator inputs that cause the VETO outputs to pile-up are verified.

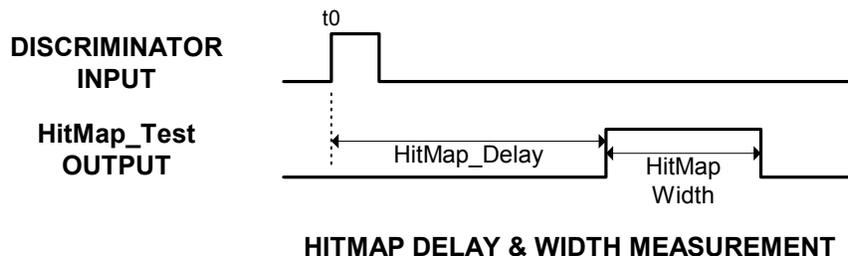


1. Send the Veto\_Delay\_Wr command with a data argument of 0. Verify with the Veto\_Delay\_Rd command.
2. Send the Veto\_Width\_Wr command with a data argument of 3. Verify with the Veto\_Width\_Rd command.
3. Using the test pulser in the double pulse mode, start with the width of the input pulses at 150 ns and the distance, d1-2, at 1  $\mu$ sec. With two pulses input, there should be two pulses output, each 200 ns in width (**wo**) at the AEM\_NVETO\_00 output.
4. Move the input pulses closer until the two AEM\_NVETO\_00 pulses touch. At this point, they should become one extended pulse. As the input pulses move closer, the tail of the extended pulse should become shorter. As the input pulses are separated again by greater than 200 ns, the AEM\_NVETO\_00 should again show two pulses.
5. Repeat this test for channel 01 and verify.
6. Repeat this test for channel 02 and verify.
7. Repeat this test for channel 03 and verify.
8. Repeat this test for channel 04 and verify.

9. Repeat this test for channel 05 and verify.
10. Repeat this test for channel 06 and verify.
11. Repeat this test for channel 07 and verify.
12. Repeat this test for channel 08 and verify.
13. Repeat this test for channel 09 and verify.
14. Repeat this test for channel 10 and verify.
15. Repeat this test for channel 11 and verify.
16. Repeat this test for channel 12 and verify.
17. Repeat this test for channel 13 and verify.
18. Repeat this test for channel 14 and verify.
19. Repeat this test for channel 15 and verify.
20. Repeat this test for channel 16 and verify.
21. Repeat this test for channel 17 and verify.
22. Send the GARC\_Reset command to restore all register values to their initial state.

### 11.0 Test of the HitMap Functionality

This section tests the proper functioning of the commandable functions of the HitMap test point. This test requires the use of an oscilloscope, a test pulser for input stimulus, and test points to monitor the AEM\_VETOs. The test setup is identical to the VETO test setup shown above. Only channel 00 is available on the test pin for HitMap verification. The diagram below details the measurement points to use with the digitizing oscilloscope.



#### 11.1 HitMap Width Test

1. Send the GARC\_Mode\_Wr command with a data argument of decimal 768 to ensure the test pin multiplexer is set to the HitMap\_Test output. Place an oscilloscope probe on this test point.
2. Send the HitMap\_Width\_Wr command with a data argument of 0. Verify the command using the HitMap\_Width\_Rd readback.
3. Send the HitMap\_Delay\_Wr command with a data argument of 0. Verify the command using the HitMap\_Delay\_Rd readback.
4. Send the HitMap\_Deadtime\_Wr command with a data argument of 0. Verify the command using the HitMap\_Deadtime\_Rd readback.

- Set up the test pulser to stimulate the discriminator input for channel 0. Monitor the HitMap Test output with the oscilloscope triggered from the test pulser. Using the HitMap\_Width\_Wr command with data arguments of 0 – 15, verify the HitMap Width varies as expected and record the data in the table below.

HitMap Width Command	Measured Width (ns)	Expected Width (ns)
0		150
1		200
2		250
3		300
4		350
5		400
6		450
7		500
8		550
9		600
10		650
11		700
12		750
13		800
14		850
15		900

- Send the HitMap\_Width\_Wr command with a data argument of 0 to reset the HitMap Width. Verify the command with the HitMap\_Width\_Rd command.

## 11.2 HitMap Delay Test

- Using the HitMap\_Delay\_Wr command with data arguments of 0 – 31, verify the HitMap Delay varies as expected and record the data in the table below.

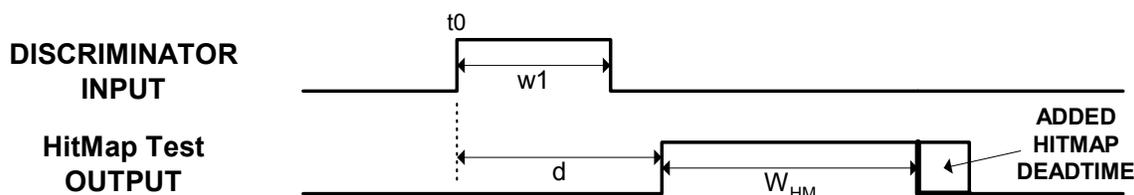
HitMap Delay Command	Measured Delay (ns)	Expected Delay (ns)
0		850
1		900
2		950
3		1000
4		1050
5		1100
6		1150
7		1200
8		1250
9		1300
10		1350
11		1400
12		1450
13		1500
14		1550
15		1600
16		1650

17		1700
18		1750
19		1800
20		1850
21		1900
22		1950
23		2000
24		2050
25		2100
26		2150
27		2200
28		2250
29		2300
30		2350
31		2400

- Send the HitMap\_Delay\_Wr command with a data argument of 0 to reset the HitMap Delay. Verify the command with the HitMap\_Delay\_Rd command.

### 11.3 HitMap Deadtime Stretch Test

This test will monitor the function of the HitMap\_Deadtime adjustment. This Deadtime register adds a time (0 to 350 ns) to the end of the HitMap pulse, starting at the end of the HitMap Width calculation. The diagram below illustrates the position of the added time window.



**HITMAP DEADTIME TEST**

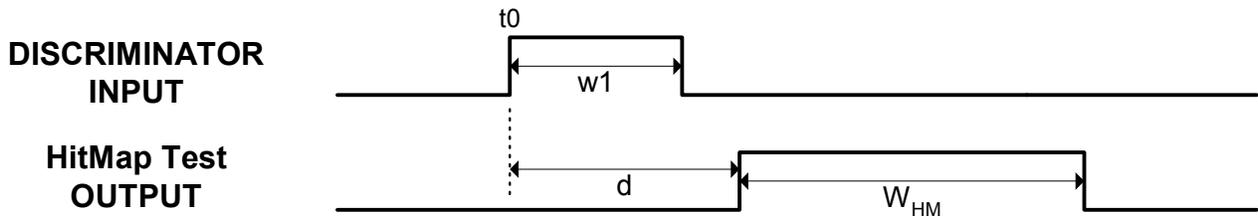
- Using the HitMap\_Deadtime\_Wr command with data arguments of 0 – 7, verify the HitMap Deadtime extends the trailing edge of the HitMap test pulse as expected and record the data in the table below.

HitMap Deadtime Command	Measured Deadtime Pulse Extension - (ns)	Expected Deadtime Pulse Extension -(ns)
0		0
1		50
2		100
3		150
4		200
5		250
6		300
7		350

- Send the GARC\_Reset command to restore all register values to their initial state.

### 11.4 HitMap Minimum Width Test

This test characterizes the performance of the HitMap functionality for a minimum width discriminator input.

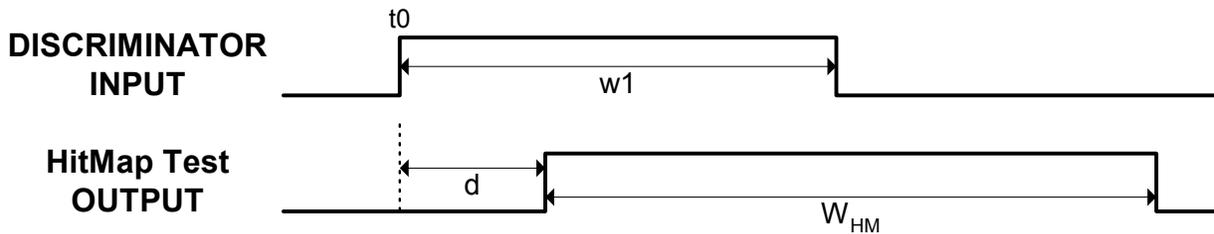


#### HITMAP MINIMUM WIDTH TEST

1. Send the HitMap\_Delay\_Wr command with a data argument of 0. Verify the command with the HitMap\_Delay\_Rd command. This fixes the delay ( $d$  in the figure) at 850 ns for this test.
2. Send the HitMap\_Width\_Wr command with a data argument of 0. Verify the command with the HitMap\_Width\_Rd command. This fixes the output width ( $W_{HM}$  in the figure) at 150 ns for this test.
3. Starting with the discriminator input width at 200 ns, note that the HitMap output is shown as above. Decrease the width of the discriminator input until the HitMap output starts to disappear. Verify that the HitMap\_Test is stable for all discriminator inputs greater than 50 ns.

### 11.5 HitMap Extended Pulse Test

This test characterizes the performance of the HitMap circuitry with an extended discriminator input pulse.

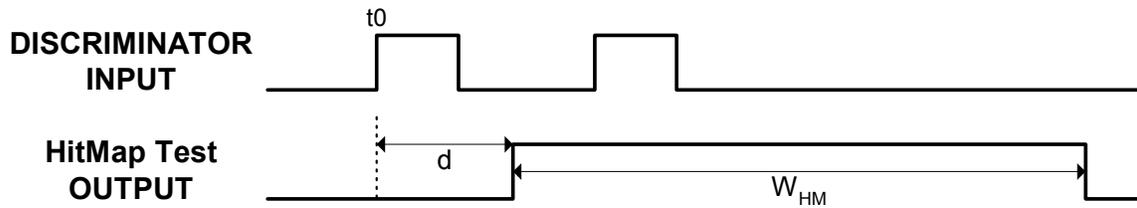


#### HITMAP EXTENDED PULSE TEST

1. Send the HitMap\_Delay\_Wr command with a data argument of 0. Verify the command with the HitMap\_Delay\_Rd command. This fixes the delay ( $d$  in the figure) at 850 ns for this test.
2. Send the HitMap\_Width\_Wr command with a data argument of 0. Verify the command with the HitMap\_Width\_Rd command. This fixes the output width ( $W_{HM}$  in the figure) at 150 ns for this test.
3. Starting with the discriminator input width at about 500 ns, increase the discriminator width up to approximately 10  $\mu$ sec. Verify that the HitMap\_Width increases with the discriminator input width.

### 11.6 HitMap Double Pulse Test

This test characterizes the performance of the HitMap circuitry with a double pulse at the discriminator input.



### HITMAP DOUBLE PULSE TEST

1. Send the HitMap\_Delay\_Wr command with a data argument of 0. Verify the command with the HitMap\_Delay\_Rd command. This fixes the delay (**d** in the figure) at 850 ns for this test.
2. Send the HitMap\_Width\_Wr command with a data argument of 0. Verify the command with the HitMap\_Width\_Rd command. This fixes the output width (**W<sub>HM</sub>** in the figure) at 150 ns for this test.
3. Send the HitMap\_Deadtime\_Wr command with a data argument of 0. Verify the command with the HitMap\_Deadtime\_Rd command.
4. Starting with the discriminator input pulses at about 150 ns in width and 10  $\mu$ s apart, note that there are two HitMap output pulses each 850 ns wide.
5. Decrease the distance between the discriminator input pulses until the two HitMap output pulses merge to one extended pulse. Verify that this merge time is approximately  $150+850 = 1000$  ns. Verify that the length of the pulse is extended 850 ns from the leading edge of the second pulse.

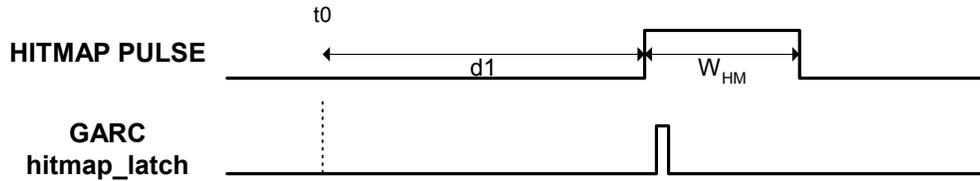
## 12.0 Synchronization of the HitMap Latch to the Discriminator Input Pulse

This section verifies the proper functionality of the HitMap latch function. The HitMap is latched within 50 ns of the moment that a trigger command is detected. The HitMap Delay must be adjusted such that the delayed Veto pulse coincides with the time from the incident particle (i.e.,  $t_0$ ) to the receipt of the trigger.

The HitMap pulse width is adjustable to allow for jitter in the LAT global triggering algorithm. The HitMap “deadtime” is also adjustable; this parameter allows for the GAFE analog baseline sufficient time to return to zero following large pulses.

This test will check for the minimum delay and maximum delay for which the HitMap bits are set active high. Using the discriminator input setup detailed above (all 18 discriminator inputs driven by the pulser), perform the following test.

1. Adjust the HitMap Width to be 150ns by sending the HitMap\_Width\_Wr command with a data argument of 0. Verify this command with the HitMap\_Width\_Rd command.
2. Set the HitMap deadtime to be 0 with the HitMap\_Deadtime\_Wr command with a data argument of 0. Verify this command with the HitMap\_Deadtime\_Rd command.
3. Start with the HitMap\_Delay at a value of 2400 ns by sending the HitMap\_Delay\_Wr command with a data argument of 31. Verify this command with the HitMap\_Delay\_Rd command.
4. Send the Hold\_Delay\_Wr command with a data argument value of 25. Verify with the Hold\_Delay\_Rd command. This sets the Trigger-to-Hold time to be 1500 ns.
5. Send the Trigger\_NOZS command and note the contents of the HitMap. Decrement the value of the HitMap delay and verify, again using the HitMap\_Delay\_Wr and HitMap\_Delay\_Rd commands. Repeat until the HitMap bits in the event data word change from 0 (inactive) to 1 (active).

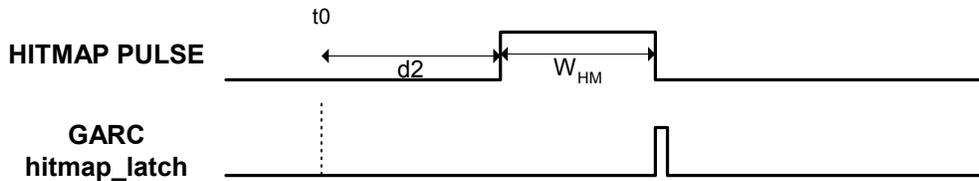


**HITMAP LATCH ON LEADING EDGE**

In the diagram above, this is equivalent to decreasing delay **d1** until the leading edge of the HitMap pulse is captured by the internal GARC latch signal, which is fixed by the Hold\_Delay. Record this delay setting below (the expected setting is 13 = 1500 ns)

**HitMap Delay (d1, leading edge of HitMap):**

6. Send the Trigger\_NOZS command and note the contents of the HitMap. Decrement the value of the HitMap delay and verify, again using the HitMap\_Delay\_Wr and HitMap\_Delay\_Rd commands. Repeat until the HitMap bits in the event data word change from 1 (active) to 0 (inactive).



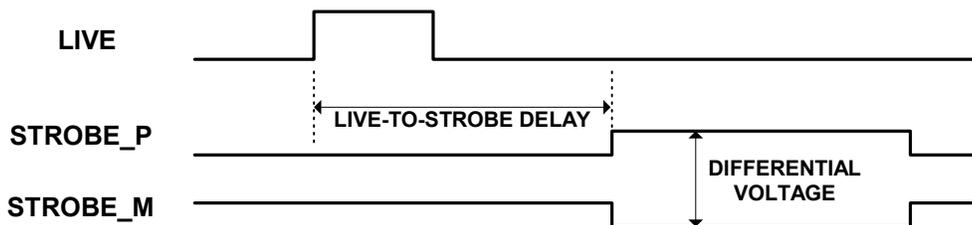
**HITMAP LATCH ON TRAILING EDGE**

In the diagram above, this is equivalent to decreasing delay **d2** until the trailing edge of the HitMap pulse is just past the internal GARC latch signal, which is fixed by the Hold\_Delay. Record this delay setting below (the expected setting is 17 = 1700 ns). Record this delay setting below.

**HitMap Delay (d2, trailing edge of HitMap):**

**13.0 Strobe Test**

This section tests the proper functioning of the commandable strobe pulse to the GAFEs from the GARC. This signal is used as a level command to the GAFE(s) to indicate the timing of the test charge injection to the amplifier front end. Using the oscilloscope monitoring Live on GARC pin 179, GAFE\_STROBEP on GARC pin 163 and GAFE\_STROBEM on GARC pin 164, perform the following (as shown in the diagram below):



**STROBE COMMAND TEST**

1. Send the GARC\_Mode\_Wr command with data argument 1792 to switch the test pin multiplexer to the Live signal.

2. Set the scope to acquire on the posedge of Live. Send the GARC\_Cal\_Strobe command (this is a dataless command so the data argument is 0).
3. Verify the STROBE command has executed by noting the differential signal on the two oscilloscope channels. It is expected that the voltage amplitude of each side of the differential is greater than 50 mV and that the pulse duration is approximately 10  $\mu$ sec.

**STROBEP to STROBEM differential voltage:**

**Live-to-STROBE delay:**

**STROBE pulse duration ( $\mu$ sec):**

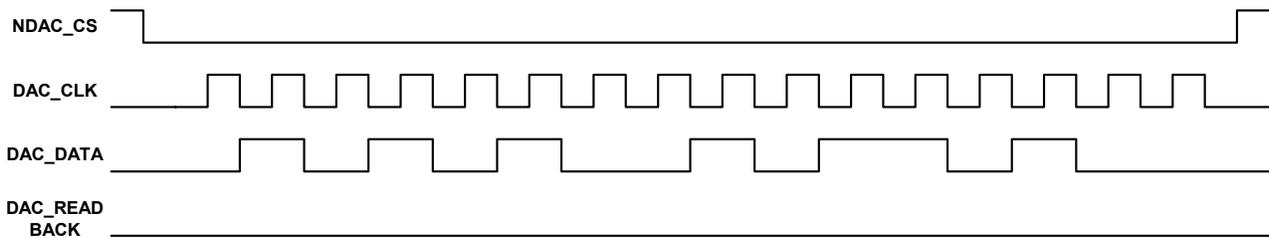
4. Send the GARC\_Reset command to restore all register values to their initial state.

#### **14.0 Test of the MAX5121 DAC and Associated Control Functions**

This section tests the proper commanding of the MAX5121 DAC. A voltmeter will be used to monitor the DAC output on the board under test.

##### **14.1 Capture of the DAC Control Signals**

1. Connect the probe on channel 1 to NDAC\_CS. Set the scope to trigger on the negedge of this signal.
2. Connect the probe on channel 2 to DAC\_CLK.
3. Connect the probe on channel 3 to DAC\_DATA.
4. Connect the probe on channel 4 to DAC\_READBACK.
5. Send the HVBS\_Level\_Wr command with a data argument of hex A5A (decimal 2650). Send the HVBS\_Level\_Rd command to verify.
6. Check that the scope is armed and then send the Use\_HV\_Nominal\_Wr command. The scope should trigger and the following waveform should be displayed. Verify that the DAC clock is nominally 5 MHz.
- 7.



DAC DATA WRITE WITH DATA PATTERN HEX A5A

8. Check that the scope is armed and then send the Use\_HV\_Nominal\_Rd command. The software readback should capture the following value in the 16 bit return word: hex F4B4 (decimal 31348). This value may also be seen on the DAC DATA READBACK scope trace.

9. Move the probe on channel 1 to NDAC\_CLR. Verify the scope is armed and send the GARC\_Reset command. The scope should trigger and a valid active low clear signal should be observed on the scope.

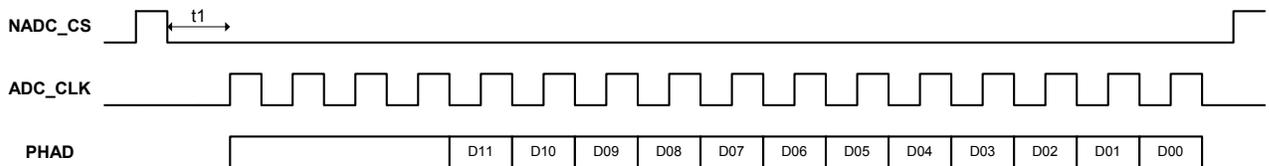
## 14.2 Test of the GARC DAC Interface Circuitry

This section tests the proper functioning of the circuitry that commands the MAX5121 DAC.

1. Send the HVBS\_Level\_Wr command with a data argument of decimal 2048. Send the HVBS\_Level\_Rd command to verify the write command.
2. Send the SAA\_Level\_Wr command with a data argument of decimal 1024. Send the SAA\_Level\_Rd command to verify.
3. Using a multimeter on the DAC output, verify that the MAX5121 output is at 0V.
4. Send the Use\_HV\_Nominal command. Verify that the MAX5121 output goes to half scale, e.g., 0.625 V.
5. Send the DAC\_HVReg\_Rd command and verify the return data pattern from the MAX5121 DAC is decimal 59392.
6. Send the Use\_SAA\_Level command. Verify that the MAX5121 output goes to 1/4 scale, e.g., 0.312 V.
7. Send the DAC\_SAAREg\_Rd command and verify the return data pattern from the MAX5121 DAC is decimal 58368.
8. Send the GARC\_Reset command. Verify that the MAX5121 output goes to 0 V.

## 15.0 Capture of the ADC Control Signals

1. Connect the probe on channel 1 to NADC\_CS. Set the scope to trigger on the negedge of this signal.
2. Connect the probe on channel 2 to ADC\_CLK.
3. Connect the probe on channel 3 to PHAD00.
4. Send the TRIG\_ZS command to initiate an analog-to-digital conversion. The scope should trigger and the following waveform should be displayed. Verify that the ADC clock is nominally 2 MHz. The time t1 represents the total ADC TACQ (ADC acquisition time).

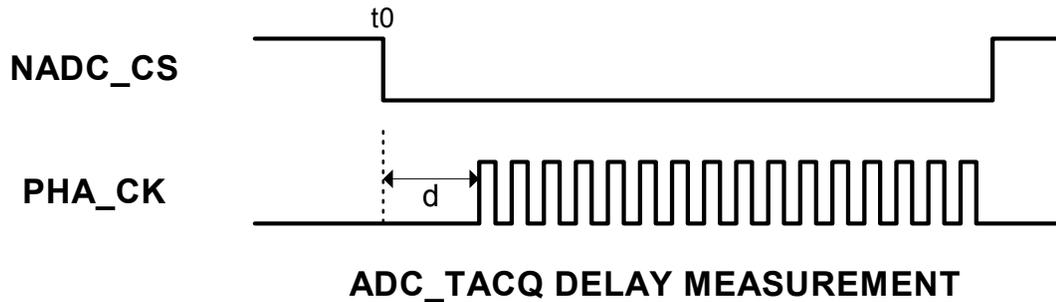


MAX145 ADC CONVERSION CYCLE

## 16.0 ADC TACQ Test

1. The next parameter to be measured will be the ADC time to acquisition (essentially the settling time allowed for the analog signal). This is a command with a 6 bit data argument. This alters the time between the ADC chip select transitioning active low and the start of the ADC clock. The register is set with the ADC\_TACQ\_Wr command and verified with the ADC\_TACQ\_Rd command. For each value of ADC\_TACQ in the table below, record the time from the falling edge of the ADC\_CS\_N

and the rising edge of the first ADC clock. Nominally, with an ADC\_TACQ register setting of 0, there will be a CS → PHA clock delay time of 2500 ns. There is the potential for a small amount of synchronization jitter (~few clocks) within the state machine on this parameter.



For each ACD\_TACQ step, measure the delay, **d**, as shown in the diagram above. For the table below, the important parameter is the even 50 ns spacing on each step from the original zero point.

ADC TACQ Register	Measured Time CS → PHA Clock (ns)	Expected Time CS → PHA Clock (ns)
0		2500
1		2550
2		2600
3		2650
4		2700
5		2750
6		2800
7		2850
8		2900
9		2950
10		3000
11		3050
12		3100
13		3150
14		3200
15		3250
16		3300
17		3350
18		3400
19		3450
20		3500
21		3550
22		3600
23		3650
24		3700
25		3750
26		3800
27		3850
28		3900
29		3950
30		4000
31		4050
32		4100
33		4150
34		4200

35		4250
36		4300
37		4350
38		4400
39		4450
40		4500
41		4550
42		4600
43		4650
44		4700
45		4750
46		4800
47		4850
48		4900
49		4950
50		5000
51		5050
52		5100
53		5150
54		5200
55		5250
56		5300
57		5350
58		5400
59		5450
60		5500
61		5550
62		5600
63		5650

2. Send the GARC\_Reset command to restore all register values to their initial state.

### **17.0 Test of the GARC Return Data Parity**

This section tests the proper functioning of the GARC return data parity select bit.

1. Send the GARC\_Status command. The data field in the return data stream should be decimal 24.
2. Send the GARC\_Mode command with a data argument of decimal 769, a command to return event data with even parity.
3. Send the GARC\_Status command. The data field in the return data stream should show a parity error.
4. Send the GARC\_Mode command with a data argument of decimal 768, a command back to odd parity, the nominal mode.
5. Send the GARC\_Status command. The data field in the return data stream should be decimal 24.

### **18.0 Test of the HVBS Triple Modular Redundancy Circuitry**

This section tests the proper functioning of the GARC HVBS enable circuitry. Each pattern in the TMR logic will be tested to verify proper recovery from a single event upset. The GARC HV\_ENABLE\_1 pin is 185 and the HV\_ENABLE\_2 pins is 186. A truth table for proper TMR circuitry function is detailed below.

<b>Enable Bit A</b>	<b>Enable Bit B</b>	<b>Enable Bit C</b>	<b>HV Enable Output</b>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1. Send the GARC\_Mode\_Wr command with a data argument of 768 and verify the data with the GARC\_Mode\_Rd command.
2. Send the GARC\_Status command. The data field in the return data stream should show that bits 1 and 2 are 0, indicating that HVBS 1 and 2, respectively, are disabled. Verify that GARC pins 185 and 186 are at 0V.
3. Send the GARC\_Mode\_Wr command with decimal argument 770 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 and 186 are at 0V.
4. Send the GARC\_Mode\_Wr command with decimal argument 772 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 and 186 are at 0V.
5. Send the GARC\_Mode\_Wr command with decimal argument 774 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is enabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 3.3V and pin 186 is at 0V.
6. Send the GARC\_Mode\_Wr command with decimal argument 776 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 0V and pin 186 is at 0V.
7. Send the GARC\_Mode\_Wr command with decimal argument 778 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is enabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 3.3V and pin 186 is at 0V.
8. Send the GARC\_Mode\_Wr command with decimal argument 780 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is enabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 3.3V and pin 186 is at 0V.
9. Send the GARC\_Mode\_Wr command with decimal argument 782 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is enabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 3.3V and pin 186 is at 0V.
10. Send the GARC\_Mode\_Wr command with decimal argument 784 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 0V and pin 186 is at 0V.
11. Send the GARC\_Mode\_Wr command with decimal argument 800 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 0V and pin 186 is at 0V.
12. Send the GARC\_Mode\_Wr command with decimal argument 816 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is enabled. Verify that GARC pins 185 is at 0V and pin 186 is at 3.3V.

13. Send the GARC\_Mode\_Wr command with decimal argument 832 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is disabled. Verify that GARC pins 185 is at 0V and pin 186 is at 0V.
14. Send the GARC\_Mode\_Wr command with decimal argument 848 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is enabled. Verify that GARC pins 185 is at 0V and pin 186 is at 3.3V.
15. Send the GARC\_Mode\_Wr command with decimal argument 864 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is enabled. Verify that GARC pins 185 is at 0V and pin 186 is at 3.3V.
16. Send the GARC\_Mode\_Wr command with decimal argument 880 and verify the data with the GARC\_Mode\_Rd command. Send the GARC\_Status command and verify that HVBS 1 is disabled and HVBS 2 is enabled. Verify that GARC pins 185 is at 0V and pin 186 is at 3.3V.
17. Send the GARC\_Reset command. Verify that GARC pins 185 is at 0V and pin 186 is at 0V.

This completes the test of the HVBS enable/disable TMR circuitry.

## **19.0 Test of the Look-At-Me Circuitry**

This section tests the proper functioning of the GARC Look-At-Me circuitry. The GARC has the capability to receive commands from either the primary side or secondary side. The selection of which set of receivers to listen to is controlled by the Look-At-Me circuitry. This circuitry toggles the status of the receiving side based upon receipt of a special command pattern (e.g., **34'h24153D721**).

1. Verify that the AEM (or AEM simulator) is connected to the GARC primary side interface (e.g., the “A” side clock and data). Send the GARC\_Status command. The data field in the return data stream should be decimal 24. Bit 0 (LSB) of the Status register represents the Look-At-Me status (0 = A, 1 = B).
2. Disconnect the interface from the primary side and move the interface to the secondary side. Send the GARC\_Status command. Observe that there is no response from the GARC.
3. Send the Look\_At\_Me command to the GARC from the secondary side interface. Send the GARC\_Status command. The data field in the return data should be decimal 25 (with the LSB = 1, indicating the Look-At-Me status is B side).
4. Disconnect the interface from the secondary side and move the interface to the primary side. Send the GARC\_Status command. Observe that there is no response from the GARC.
5. Send the Look\_At\_Me command to the GARC from the primary side interface. Send the GARC\_Status command. The data field in the return data should be decimal 24 (with the LSB = 0, indicating the Look-At-Me status is A side).

This concludes the test of the GARC Look-At-Me circuitry.

## **20.0 Test of the GAFE Parity Command Circuitry**

This section tests the proper functioning of the GAFE parity command circuitry. At least one GAFE ASIC needs to be connected to the GARC under test to perform this section. GAFE commands must be sent to the same address as the hardwired GAFE address. The GAFE logic nominally expects odd parity. When the GARC command to change the GAFE command message to even parity is executed, the GAFE should no

longer respond to messages from the GARC. When the GAFE parity is again commanded to be odd, nominal response should resume.

1. Send the GARC\_Mode\_Wr command with a decimal data argument of 768 (0x300). Verify this command with the GARC\_Mode\_Rd command.
2. Send the GAFE command GAFE\_Mode\_Rd and verify that the GAFE logic responds with the contents of the mode register.
3. Send the GARC\_Mode\_Wr command with a decimal data argument of 896 (0x380). Verify this command with the GARC\_Mode\_Rd command.
4. Send the GAFE command GAFE\_Mode\_Rd and verify that the GAFE logic does not respond to this command (e.g., the command is rejected due to incorrect parity).
5. Send the GARC\_Mode\_Wr command with a decimal data argument of 768 (0x300). Verify this command with the GARC\_Mode\_Rd command.
6. Send the GAFE command GAFE\_Mode\_Rd and verify that the GAFE logic responds with the contents of the mode register.

### 21.0 Test of the GARC LVDS Circuitry Driver Currents

This section tests the proper functioning of the GARC LVDS Driver circuitry. It is required that all LVDS drivers be terminated at the receiver with a 100 ohm resistor. The ACD-to-AEM nominal drive current is 3.5 mA across the 100 ohms for a voltage differential of 350 mV.

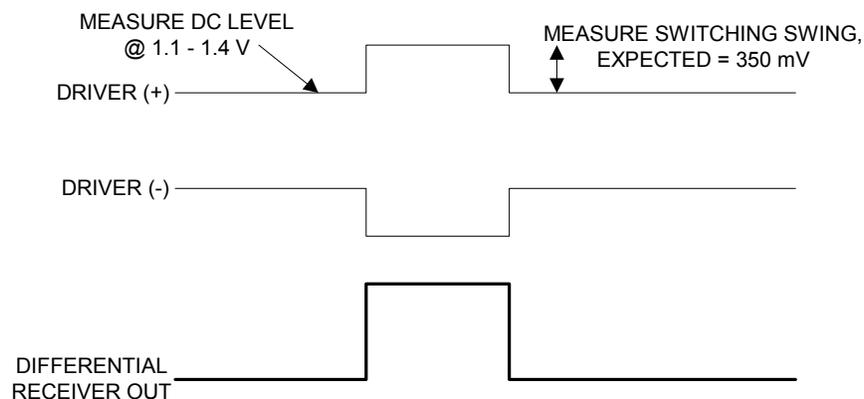
There are 6 LVDS receiver input pairs to the GARC:

- a) ACD\_CLK\_A, ACD\_CLK\_B
- b) ACD\_CMD\_A, ACD\_CMD\_B
- c) ACD\_RESET\_A, ACD\_RESET\_B

There are 40 LVDS driver output pairs from the GARC:

- d) ACD\_NSDATA\_A, ACD\_NSDATA\_B
- e) ACD\_NVETO\_nnA, ACD\_NVETO\_nnB (where nn is 00 – 17)
- f) ACD\_CNO\_A, ACD\_CNO\_B

The “A” side interface will be examined first. Verify that an AEM (or AEM simulator) is connected to the GARC “A” side interface. Using a digitizing oscilloscope, measure the DC baseline and switching differential voltage on each of the driver/receiver pairs. Two scope probes may be used, one on each side of the differential receiver inputs. The DC level and switching voltages may be measured as diagrammed below.



Record the information for each pin on the Primary side interface in the table below.

<b>GARC Signal</b>	<b>GARC Pin (+)</b>	<b>GARC Pin (-)</b>	<b>DC Level (V)</b>	<b>Switch (+)</b>	<b>Switch (-)</b>
ACD_CLK_A	2	3			
ACD_CMD_A	6	7			
ACD_RESET_A	10	11			
ACD_NSDATA_A	14	15			
ACD_NVETO_00A	22	23			
ACD_NVETO_01A	28	29			
ACD_NVETO_02A	32	33			
ACD_NVETO_03A	36	37			
ACD_NVETO_04A	40	41			
ACD_NVETO_05A	44	45			
ACD_NVETO_06A	48	49			
ACD_NVETO_07A	54	55			
ACD_NVETO_08A	58	59			
ACD_NVETO_09A	62	63			
ACD_NVETO_10A	66	67			
ACD_NVETO_11A	70	71			
ACD_NVETO_12A	74	75			
ACD_NVETO_13A	204	205			
ACD_NVETO_14A	200	201			
ACD_NVETO_15A	196	197			
ACD_NVETO_16A	192	193			
ACD_NVETO_17A	188	189			
ACD_CNO_A	18	19			

Switching control of the AEM interface to the Secondary side interface, record the information below.

<b>GARC Signal</b>	<b>GARC Pin (+)</b>	<b>GARC Pin (-)</b>	<b>DC Level (V)</b>	<b>Switch (+)</b>	<b>Switch (-)</b>
ACD_CLK_B	4	5			
ACD_CMD_B	8	9			
ACD_RESET_B	12	13			
ACD_NSDATA_B	16	17			
ACD_NVETO_00B	25	26			
ACD_NVETO_01B	30	31			
ACD_NVETO_02B	34	35			
ACD_NVETO_03B	38	39			
ACD_NVETO_04B	42	43			
ACD_NVETO_05B	46	47			
ACD_NVETO_06B	50	51			
ACD_NVETO_07B	56	57			
ACD_NVETO_08B	60	61			
ACD_NVETO_09B	64	65			
ACD_NVETO_10B	68	69			
ACD_NVETO_11B	72	73			
ACD_NVETO_12B	76	77			
ACD_NVETO_13B	206	207			
ACD_NVETO_14B	202	203			
ACD_NVETO_15B	198	199			
ACD_NVETO_16B	194	195			
ACD_NVETO_17B	190	191			
ACD_CNO_B	20	21			

This completes the testing of the GARC LVDS driver circuitry.

## **22.0 FREE Board ID Circuit Test**

This section tests the proper capture of the FREE circuit card serial number identification by the GARC logic. The FREE circuit ID is operated during a PHA conversion, which may be initiated via a trigger command.

1. Set the hardwired FREE board address to decimal 0. Send the Trigger\_NOZS command to the GARC to initiate a PHA conversion. Send the FREE\_Board\_ID command and look for the address value to be returned in the GARC readback data.
2. Set the hardwired FREE board address to decimal 255. Send the Trigger\_NOZS command to the GARC to initiate a PHA conversion. Send the FREE\_Board\_ID command and look for the address value to be returned in the GARC readback data.
3. Set the hardwired FREE board address to decimal 85. Send the Trigger\_NOZS command to the GARC to initiate a PHA conversion. Send the FREE\_Board\_ID command and look for the address value to be returned in the GARC readback data.
4. Set the hardwired FREE board address to decimal 170. Send the Trigger\_NOZS command to the GARC to initiate a PHA conversion. Send the FREE\_Board\_ID command and look for the address value to be returned in the GARC readback data.

This completes the test of the FREE board ID circuitry.

## **23.0 Maximum PHA Return Test**

This section tests the proper functioning of the event data processor as it handles the Maximum Number of PHA words command.

1. Send the Max\_PHA\_Wr command with a data argument of 18. Send the Max\_PHA\_Rd command and verify that the data value has been set. Send the Trigger\_NOZS command and verify that the event data processor sends back 18 PHA words.
2. Repeat the previous step with a data argument of 17.
3. Repeat the previous step with a data argument of 16.
4. Repeat the previous step with a data argument of 15.
5. Repeat the previous step with a data argument of 14.
6. Repeat the previous step with a data argument of 13.
7. Repeat the previous step with a data argument of 12.
8. Repeat the previous step with a data argument of 11.
9. Repeat the previous step with a data argument of 10.
10. Repeat the previous step with a data argument of 9.
11. Repeat the previous step with a data argument of 8.
12. Repeat the previous step with a data argument of 7.
13. Repeat the previous step with a data argument of 6.

14. Repeat the previous step with a data argument of 5.
15. Repeat the previous step with a data argument of 4.
16. Repeat the previous step with a data argument of 3.
17. Repeat the previous step with a data argument of 2.
18. Repeat the previous step with a data argument of 1.
19. Repeat the previous step with a data argument of 0.
20. Send the GARC\_Reset command to restore the Max\_PHA number to the default. This completes the test of the GARC MAX\_PHA function.

#### **24.0 PHA Enable/Disable Test**

This section tests the proper functioning of the event data processor as it handles the PHA enables and disabled in selection of PHA words for transmission. For this section, the GAFE simulator shall be programmed such that each of the 18 simulated PHA channels has a uniquely identifying PHA value (e.g., for example, related to the GAFE channel number)

1. Send the PHA\_En0\_Wr command with a data argument of decimal 'hFFFF (65535). Verify this value with the PHA\_En0\_Rd command. This enables channels 0 – 15.
2. Send the PHA\_En1\_Wr command with a data argument of decimal 3. Verify this value with the PHA\_En1\_Rd command. This enables channels 16 and 17.
3. Send the Trigger\_NOZS command. The event data processor should respond with 18 PHA words during event readout.
4. Send the PHA\_En0\_Wr command with a data argument of 'hFFFD (65534). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 0.
5. Send the PHA\_En0\_Wr command with a data argument of 'hFFFB (65533). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 1.
6. Send the PHA\_En0\_Wr command with a data argument of 'hFFF7 (65531). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 2.
7. Send the PHA\_En0\_Wr command with a data argument of 'hFFF7 (65527). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 3.
8. Send the PHA\_En0\_Wr command with a data argument of 'hFFEF (65519). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 4.
9. Send the PHA\_En0\_Wr command with a data argument of 'hFFDF (65503). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 5.
10. Send the PHA\_En0\_Wr command with a data argument of 'hFFBF (65471). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 6.

11. Send the PHA\_En0\_Wr command with a data argument of 'hFF7F (65407). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 7.
12. Send the PHA\_En0\_Wr command with a data argument of 'hFFEF (65279). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 8.
13. Send the PHA\_En0\_Wr command with a data argument of 'hFCFF (64767). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 9.
14. Send the PHA\_En0\_Wr command with a data argument of 'hFBFF (64511). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 10.
15. Send the PHA\_En0\_Wr command with a data argument of 'hF7FF (63487). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 11.
16. Send the PHA\_En0\_Wr command with a data argument of 'hEFFF (61439). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 12.
17. Send the PHA\_En0\_Wr command with a data argument of 'hCFFF (53247). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 13.
18. Send the PHA\_En0\_Wr command with a data argument of 'hBFFF (49151). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 14.
19. Send the PHA\_En0\_Wr command with a data argument of 'h7FFF (32767). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 15.
20. Send the PHA\_En0\_Wr command with a data argument of 'hFFFF (65535). Verify this value with the PHA\_En0\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words.
21. Send the PHA\_En1\_Wr command with a data argument of 2. Verify this value with the PHA\_En1\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 16.
22. Send the PHA\_En1\_Wr command with a data argument of 1. Verify this value with the PHA\_En1\_Rd command. Send the Trigger\_NOZS command. The event processor should respond with all PHA words except channel 17.
23. Send the GARC\_Reset command to reinitialize all GARC registers. This completes the testing of the PHA enable and disable register bits.

## **25.0 PHA Threshold Verification Test**

This section tests the proper functioning of each of the 18 PHA thresholding circuits. The intent of the circuits is such that the ZS Map bit is set active high for a given PHA word if either the range bit is high or the 12 bit PHA word is greater than the PHA threshold. This test requires a GAFE simulator test board that is programmed to emulate each of the possible PHA patterns, decimal 0 through 8191 (range bit + 12 bits ADC conversion). This test needs to be run in an automated format due to the very large number of

possibilities. A suggestion for the proper functionality is detailed below. The Max\_PHA should be set to 18 so that none of the PHA words are suppressed by that function.

```
for (pha_word = 0; pha_word < 8192 ; pha_word++)
  for (pha_threshold = 0; pha_threshold < 65536; pha_threshold++)
    for (GAFE_channel = 0; GAFE_channel < 18; GAFE_channel++)
      {
        Send Trigger_ZS command;
        Capture the 18 bit HitMap word;
        If HitMap[GAFE_channel] != range_bit | (pha_word > pha_threshold)
          GARC_Error;
      }
```

During this test, the proper PHA values are checked for return in the event data. For the duration of this test, the PHA value input to each channel up to 17 should be an increment of 1 more than the previous channel so that each channel has a unique PHA identifier (for the first 17 pha\_words, some channels will have a zero value multiple times). It is important to have a unique identifier for PHA words to verify proper position in the event data stream. The ZS Map needs to be compared to the PHA in the event data stream to ensure a match.

At the conclusion of this section of the test, send the GARC\_Reset command to initialize the GARC registers. This completes the PHA Threshold Verification section of the procedure.

## **26.0 GARC Diagnostics Verification**

This section tests the proper functioning of the diagnostics commands in the GARC logic.

### **26.1 Test of the GARC Diagnostic Status Register**

1. Send the GARC\_Reset command to initialize the GARC registers.
2. Send the GARC\_Mode\_Rd command. The data field in the return data stream should be decimal 768.
3. Send the GARC\_Status command. The data field in the return data stream should be decimal 24.
4. Send the GARC\_Cmd\_Reg command. The data field in the return data stream should be decimal 0.
5. Send the GARC\_Diagnostic command. The most significant four bits in the return data stream should be decimal 0. The remaining 12 bits are state machine loop and command counters and the total should not be zero.

### **26.2 AEM Command Parity Error Simulation Test**

This test generates simulated AEM command word errors and verifies that the GARC responds in the appropriate manner. In this section of the test, the AEM (or AEM simulator) must send a GARC command with an error in the command addressing portion of the command word. For example,

A correct (odd parity) Calib command has the format: 34'h2404E0001.

A Calib command with even command parity has the format: 34'h2404C0001.

1. Send the GARC\_Reset command to reinitialize all GARC registers.
2. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b0000.

3. Send the GARC\_Cal\_Strobe command, 34'h2404E0001.
4. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b0000.
5. Send the Command\_Register command and verify that the return data is 16'h0000.
6. Send the even command parity (i.e., error) command: 34'h2404C0001.
7. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b1101.
8. Send the Command\_Register command and verify that the return data is 16'h404E.

### **26.3 AEM Data Parity Error Simulation Test**

This test generates simulated AEM data word errors and verifies that the GARC responds in the appropriate manner. In this section of the test, the AEM (or AEM simulator) must send a GARC command with an error in the command addressing portion of the command word. For example,

A correct (odd parity) Calib command has the format: 34'h2404E0001.

A Calib command with even data parity could have the format: 34'h2404E00081.

1. Send the GARC\_Reset command to reinitialize all GARC registers.
2. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b0000.
3. Send the GARC\_Cal\_Strobe command, 34'h2404E0001.
4. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b0000.
5. Send the Command\_Register command and verify that the return data is 16'h0000.
6. Send the even command parity (i.e., error) command: 34'h2404E00081.
7. Send the GARC\_Diagnostic command and verify that bits 15:12 in the return data word are 4'b1011.
8. Send the Command\_Register command and verify that the return data is 16'h0008.

### **26.4 Command Counter Test**

1. Send the GARC\_Reset command to initialize the GARC registers.
2. Send the GARC\_Diagnostic command. The value in bits 7:0 of the returned data word should have the value 1.
3. Send the GARC\_Version command ten times.
4. Send the GARC\_Diagnostic command. The value in bits 7:0 of the returned data word should have the value 11.

### **26.5 GARC Diagnostic State Loop Counter Test**

1. Send the GARC\_Reset command to initialize the GARC registers.
2. Send the GARC\_Diagnostic command. The value in bits 11:8 of the returned data word should have the value 1.
3. Send the GARC\_Version command ten times.

4. Send the GARC\_Diagnostic command. The value in bits 11:8 of the returned data word should have the value 11.

## **27.0 GAFE Interface Test**

This section tests the proper interface of the GARC to the GAFE ASIC. At least one GAFE ASIC must be present on the board being tested. This section will test the GAFE Hold, Strobe, Command Data, and Return Data.

1. Send the Trig\_NOZS command. Monitor the GAFE\_HOLD signal differentially on the GARC (GAFE\_HOLDP on pin 161 and GAFE\_HOLDM on pin 162) and verify TBD voltage across the termination resistor.
2. Send the GARC\_Cal\_Strobe command. Monitor the STROBE signal to the GAFE differentially on the GARC (GAFE\_STROBEP on pin 163 and GAFE\_STROBEM on pin 164). Verify TBD voltage across the termination resistor.
3. Send the GAFE\_Version command to a valid GAFE address. The data in the readback should indicate GAFE version 2. This verifies the GAFE command and return data paths.
4. Send the GARC\_Reset command to reinitialize the GARC ASIC. Additional testing of the GAFE logic functions may be performed during the GAFE functional test.

## **28.0 Testing the GAFE Logic for ASICs Connected to the GARC**

At this point in the GARC functional test, functional testing of the logic for any GAFEs connected to the GARC may be performed. A minimum of one GAFE ASIC (or the GAFE logic simulator) must be connected for this test. If multiple GAFE logic cores are to be tested concurrently, such as the nominal condition for a populated FREE circuit card, then each address may be tested in sequence, replicating the steps detailed in the GAFE logic test procedure. Each GAFE is independent and the commands for each test may be performed either in series or in parallel up to a total of 18 GAFE ASICs.

### **28.1 Initial GAFE Logic Reset Test**

After initial power up or a GARC Reset command is sent, the GAFE registers should be initialized. This is tested by the following command sequence (sent to each GAFE being tested at the unique five bit address for that GAFE).

1. Send the GARC\_Reset command.
2. Send the GAFE\_Mode\_Read command. The data field in the return data stream should be 16'h30 (48).
3. Send the GAFE\_DAC1\_Read command. The data field in the return data stream should be 16'h39 (57).
4. Send the GAFE\_DAC2\_Read command. The data field in the return data stream should be 16'h26 (38).
5. Send the GAFE\_DAC3\_Read command. The data field in the return data stream should be 16'h37 (55).
6. Send the GAFE\_DAC4\_Read command. The data field in the return data stream should be 16'h20 (32).

7. Send the GAFE\_DAC5\_Read command. The data field in the return data stream should be 16'h00 (0).
8. Send the GAFE\_Version command. The data field in the return data stream should be 16'h02 (2).
9. Send the GAFE\_Write\_Ctr command. The data field in the return data stream should be 16'h00 (0).
10. Send the GAFE\_Reject\_Ctr command. The data field in the return data stream should be 16'h00 (0).
11. Send the GAFE\_Cmd\_Ctr command. The data field in the return data stream should be 16'h0A (10).
12. Send the GAFE\_Chip\_Address command. The data field in the return data stream should be the address of the chip that was commanded.

### **28.2 GAFE ASICs Mode Register Test**

This section tests the proper functioning of the GAFE mode register. Repeat for all GAFE ASICs present using valid GAFE addressing.

1. Send the GAFE\_Mode\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_Mode\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_Mode\_Write command to the GAFE being tested with a data field of 16'hFF (255).
4. Send the GAFE\_Mode\_Read command. The data field in the return data stream should be 16'hFF (255).
5. Send the GAFE\_Mode\_Write command to the GAFE being tested with a data field of 16'h30 (48).
6. Send the GAFE\_Mode\_Read command. The data field in the return data stream should be 16'h30 (48).

### **28.3 GAFE ASIC DAC1 Register Test**

This section tests the proper functioning of the GAFE DAC #1 register. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_DAC1\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_DAC1\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_DAC1\_Write command to the GAFE being tested with a data field of 16'h3F (63).
4. Send the GAFE\_DAC1\_Read command. The data field in the return data stream should be 16'h3F (63).
5. Send the GAFE\_DAC1\_Write command to the GAFE being tested with a data field of 16'h39 (57).
6. Send the GAFE\_DAC1\_Read command. The data field in the return data stream should be 16'h39 (57).

### **28.4 GAFE ASIC DAC2 Register Test**

This section tests the proper functioning of the GAFE DAC #2 register. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_DAC2\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_DAC2\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_DAC2\_Write command to the GAFE being tested with a data field of 16'h3F (63).
4. Send the GAFE\_DAC2\_Read command. The data field in the return data stream should be 16'h3F (63).
5. Send the GAFE\_DAC2\_Write command to the GAFE being tested with a data field of 16'h26 (38).
6. Send the GAFE\_DAC2\_Read command. The data field in the return data stream should be 16'h26 (38).

### **28.5 GAFE ASIC DAC3 Register Test**

This section tests the proper functioning of the GAFE DAC #3 register. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_DAC3\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_DAC3\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_DAC3\_Write command to the GAFE being tested with a data field of 16'h3F (63).
4. Send the GAFE\_DAC3\_Read command. The data field in the return data stream should be 16'h3F (63).
5. Send the GAFE\_DAC3\_Write command to the GAFE being tested with a data field of 16'h37 (55).
6. Send the GAFE\_DAC3\_Read command. The data field in the return data stream should be 16'h37 (55).

### **28.6 GAFE ASIC DAC4 Register Test**

This section tests the proper functioning of the GAFE DAC #4 register. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_DAC4\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_DAC4\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_DAC4\_Write command to the GAFE being tested with a data field of 16'h3F (63).
4. Send the GAFE\_DAC4\_Read command. The data field in the return data stream should be 16'h3F (63).
5. Send the GAFE\_DAC4\_Write command to the GAFE being tested with a data field of 16'h20 (32).
6. Send the GAFE\_DAC4\_Read command. The data field in the return data stream should be 16'h20 (32).

### **28.7 GAFE ASIC DAC5 Register Test**

This section tests the proper functioning of the GAFE DAC #5 register. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_DAC5\_Write command to the GAFE being tested with a data field of 16'h00 (0).
2. Send the GAFE\_DAC5\_Read command. The data field in the return data stream should be 16'h00 (0).
3. Send the GAFE\_DAC5\_Write command to the GAFE being tested with a data field of 16'h3F (63).
4. Send the GAFE\_DAC5\_Read command. The data field in the return data stream should be 16'h3F (63).
5. Send the GAFE\_DAC5\_Write command to the GAFE being tested with a data field of 16'h00 (0).
6. Send the GAFE\_DAC5\_Read command. The data field in the return data stream should be 16'h00 (0).

### **28.8 GAFE ASIC Diagnostics Verification**

This section tests the proper functioning of the diagnostics command counters in the GAFE. Repeat for all GAFE ASICs present, using valid GAFE addressing.

1. Send the GAFE\_Reject\_Ctr command to the GAFE being tested. The data field in the return data stream should be 16'h00 (0).
2. Send the GAFE\_Write\_Ctr command. The data field in the return data stream should be 'h12 (18).
3. Send the GAFE\_Cmd\_Ctr command. The data field in the return data stream should be 'h32 (50).

### **28.9 GAFE ASIC Broadcast Command Functional Verification**

This section tests the proper response to broadcast commands (e.g., address 31) by the GAFEs.

1. Send the GAFE\_Mode\_Read command with a broadcast address, i.e., address 'h1F (31) , to all the GAFEs being tested. There should be no data returned from any GAFE.
2. Send the GAFE\_Mode\_Write command with a broadcast address and a data field of 'hAA (170) to the GAFEs.
3. Send a GAFE\_Mode\_Read address command to each GAFE under test. The return data field from each addressed GAFE should be 'hAA (170).
4. Send a GARC\_Reset command to reinitialize each of the GAFE logic cores.

## **Appendix 1: GARC Documentation**

A more complete set of documentation (such as the Verilog, EDIF, layout, and wirebonding diagram) is available on the LHEA ACD electronics web page at:

<http://lhea-glast.gsfc.nasa.gov/acd/electronics/>

## Appendix 2: GARC Configuration Command Format

The GARC logic core responds only to properly structured commands. There are two possible command types – Trigger Commands and Configuration Commands. Trigger commands initiate an Event Data cycle, causing a GAFE Hold, an analog-to-digital conversion, and the return of an event data packet. Configuration commands are used to either read or write GARC or GAFE registers.

The format for Trigger Commands is detailed in the table below.

Bit(s)	Bit Description	Value
3	Start Bit	1
2:1	Trigger Type Bits	10 for ZS Enabled Trigger 01 for Send All PHA Trigger
0	Parity Bit	Odd parity bit over previous two bits

Therefore, a 1100 is a ZS Enabled Trigger command and a 1010 is a Send All PHA Trigger command.

This format for GARC Configuration Commands is detailed in the table below.

Field	# bits	Function
Start	1	1 for start
CMD Type	2	00 for command
CMD Type Parity	1	Odd Parity over previous 2 bits (without Start bit)
GAFE/GARC Select	1	0 for GARC 1 for GAFE
GAFE/GARC Address	5	GAFE: Select which GAFE, 0x1F for all GAFE GARC: Select which function block
Read/Write	1	0 for write, 1 for read
Data/Dataless	1	0 for dataless, 1 for data, always 1 for ACD
register/function number	4	Which register/function in the function block
CMD Parity	1	Odd parity bit over previous 15 bits
Data	16	Data Field
Data Parity	1	Odd parity bit over previous 16 bits

### Appendix 3: GARC Event Data Format:

The GARC core returns an event data packet when a valid trigger command is received. The event data format is detailed in the table below.

<u>Field</u>	<u># bits</u>	<u>Function</u>
Start Bit	1	1 for start
Hit Map Bits	18	Bits 17-0 for channels 0-17, bit set if hit in channel
Zero Suppression Bits	18	Bits 17-0 for channels 0-17, bit set if PHA above threshold
CMD/Data ERROR	1	Error in command parity detected
Header Parity	1	Odd parity bit over previous 37 bits
PHA Words (quantity 0-18) Order: Channel 0 to channel 17	15	Bit 14: 1 if another PHA word follows this one, 0 if this is last one Bit 13: 1 for high range, 0 for low range Bits 12-1: the PHA value, 0 to 4095 Bit 0: Odd parity over last 14 bits

#### Appendix 4: GARC Configuration Data Readback Format:

The GARC core returns an register configuration data packet when a valid configuration readback command is received. The configuration readback data format is detailed in the table below.

<u>Field</u>	<u># bits</u>	<u>Function</u>
Start	1	1 for start
GAFE/GARC Select	1	Copy of write command field (0 for GARC, 1 for GAFE)
GAFE/GARC Address	5	Copy of write command field (Select which GAFE)
Read/Write	1	1 for read
Data/Dataless	1	always 1
Register/function number	4	Copy of write command field (which register/function in the function block)
CMD Parity	1	Odd parity bit over previous 12 bits
Data	16	Data, MSB first
CMD/DATA ERROR	1	Error in parity detected
Parity	1	Odd parity bit over previous 17 bits

### Appendix 5: GARC Pin List

Tanner IO Cell	MOSIS Bond Pad Name	GARC Package Pin Number	GARC Signal Name
PADGnd	L1	1	DGND
PADAREF	L2	2	ACD_CLK_AP
PADAREF	L3	3	ACD_CLK_AM
PADAREF	L4	4	ACD_CLK_BP
PADAREF	L5	5	ACD_CLK_BM
PADAREF	L6	6	ACD_NSCMD_AP
PADAREF	L7	7	ACD_NSCMD_AM
PADAREF	K8	8	ACD_NSCMD_BP
PADAREF	L9	9	ACD_NSCMD_BM
PADAREF	L10	10	ACD_NRST_AP
PADAREF	L11	11	ACD_NRST_AM
PADAREF	L12	12	ACD_NRST_BP
PADAREF	L13	13	ACD_NRST_BM
PADAREF	L14	14	ACD_NSDATA_AP
PADAREF	L15	15	ACD_NSDATA_AM
PADAREF	L16	16	ACD_NSDATA_BP
PADAREF	L17	17	ACD_NSDATA_BM
PADAREF	L18	18	ACD_NCNO_AP
PADAREF	L19	19	ACD_NCNO_AM
PADAREF	L20	20	ACD_NCNO_BP
PADAREF	L21	21	ACD_NCNO_BM
PADAREF	L22	22	ACD_NVETO_00AP
PADAREF	L23	23	ACD_NVETO_00AM
PADVdd	L24	24	DVDD
PADAREF	L25	25	ACD_NVETO_00BP
PADAREF	L26	26	ACD_NVETO_00BM
PADGnd	L27	27	DGND
PADAREF	L28	28	ACD_NVETO_01AP
PADAREF	L29	29	ACD_NVETO_01AM
PADAREF	L30	30	ACD_NVETO_01BP
PADAREF	L31	31	ACD_NVETO_01BM
PADAREF	L32	32	ACD_NVETO_02AP
PADAREF	L33	33	ACD_NVETO_02AM
PADAREF	L34	34	ACD_NVETO_02BP
PADAREF	L35	35	ACD_NVETO_02BM
PADAREF	L36	36	ACD_NVETO_03AP
PADAREF	L37	37	ACD_NVETO_03AM
PADAREF	L38	38	ACD_NVETO_03BP
PADAREF	L39	39	ACD_NVETO_03BM
PADAREF	L40	40	ACD_NVETO_04AP
PADAREF	L41	41	ACD_NVETO_04AM
PADAREF	L42	42	ACD_NVETO_04BP
PADAREF	L43	43	ACD_NVETO_04BM
PADAREF	L44	44	ACD_NVETO_05AP
PADAREF	L45	45	ACD_NVETO_05AM
PADAREF	L46	46	ACD_NVETO_05BP
PADAREF	L47	47	ACD_NVETO_05BM
PADAREF	L48	48	ACD_NVETO_06AP
PADAREF	L49	49	ACD_NVETO_06AM
PADAREF	L50	50	ACD_NVETO_06BP
PADAREF	L51	51	ACD_NVETO_06BM
PADVdd	L52	52	DVDD
PADGnd	B1	53	DGND
PADAREF	B2	54	ACD_NVETO_07AP
PADAREF	B3	55	ACD_NVETO_07AM

PADAREF	B4	56	ACD_NVETO_07BP
PADAREF	B5	57	ACD_NVETO_07BM
PADAREF	B6	58	ACD_NVETO_08AP
PADAREF	B7	59	ACD_NVETO_08AM
PADAREF	B8	60	ACD_NVETO_08BP
PADAREF	B9	61	ACD_NVETO_08BM
PADAREF	B10	62	ACD_NVETO_09AP
PADAREF	B11	63	ACD_NVETO_09AM
PADAREF	B12	64	ACD_NVETO_09BP
PADAREF	B13	65	ACD_NVETO_09BM
PADAREF	B14	66	ACD_NVETO_10AP
PADAREF	B15	67	ACD_NVETO_10AM
PADAREF	B16	68	ACD_NVETO_10BP
PADAREF	B17	69	ACD_NVETO_10BM
PADAREF	B18	70	ACD_NVETO_11AP
PADAREF	B19	71	ACD_NVETO_11AM
PADAREF	B20	72	ACD_NVETO_11BP
PADAREF	B21	73	ACD_NVETO_11BM
PADAREF	B22	74	ACD_NVETO_12AP
PADAREF	B23	75	ACD_NVETO_12AM
PADAREF	B24	76	ACD_NVETO_12BP
PADAREF	B25	77	ACD_NVETO_12BM
PADVdd	B26	78	DVDD
PADAREF	B27	79	CHID_17
PADGnd	B28	80	DGND
PADAREF	B29	81	DISC_17
PADAREF	B30	82	IRTN_17
PADAREF	B31	83	CHID_16
PADAREF	B32	84	DISC_16
PADAREF	B33	85	IRTN_16
PADAREF	B34	86	CHID_15
PADAREF	B35	87	DISC_15
PADAREF	B36	88	IRTN_15
PADAREF	B37	89	CHID_14
PADAREF	B38	90	DISC_14
PADAREF	B39	91	IRTN_14
PADAREF	B40	92	CHID_13
PADAREF	B41	93	DISC_13
PADAREF	B42	94	IRTN_13
PADAREF	B43	95	CHID_12
PADAREF	B44	96	DISC_12
PADAREF	B45	97	IRTN_12
PADAREF	B46	98	CHID_11
PADAREF	B47	99	DISC_11
PADAREF	B48	100	IRTN_11
PADAREF	B49	101	CHID_10
PADAREF	B50	102	DISC_10
PADAREF	B51	103	IRTN_10
PADAREF	B52	104	HLD_WOR_BIAS
PADAREF	R52	105	CHID_09
PADAREF	R51	106	DISC_09
PADAREF	R50	107	IRTN_09
PADAREF	R49	108	CHID_08
PADAREF	R48	109	DISC_08
PADAREF	R47	110	IRTN_08
PADAREF	R46	111	CHID_07
PADAREF	R45	112	DISC_07
PADAREF	R44	113	IRTN_07
PADAREF	R43	114	CHID_06
PADAREF	R42	115	DISC_06
PADAREF	R41	116	IRTN_06
PADAREF	R40	117	CHID_05

PADAREF	R39	118	DISC_05
PADAREF	R38	119	IRTN_05
PADAREF	R37	120	CHID_04
PADAREF	R36	121	DISC_04
PADAREF	R35	122	IRTN_04
PADAREF	R34	123	CHID_03
PADAREF	R33	124	DISC_03
PADAREF	R32	125	IRTN_03
PADAREF	R31	126	CHID_02
PADAREF	R30	127	DISC_02
PADAREF	R29	128	IRTN_02
PADAREF	R28	129	CHID_01
PADAREF	R27	130	DISC_01
PADAREF	R26	131	IRTN_01
PADVdd	R25	132	DVDD
PADAREF	R24	133	CHID_00
PADAREF	R23	134	DISC_00
PADAREF	R22	135	IRTN_00
PADGnd	R21	136	DGND
PADInC	R20	137	PHAD_17
PADInC	R19	138	PHAD_16
PADInC	R18	139	PHAD_15
PADInC	R17	140	PHAD_14
PADInC	R16	141	PHAD_13
PADInC	R15	142	PHAD_12
PADInC	R14	143	PHAD_11
PADInC	R13	144	PHAD_10
PADInC	R12	145	PHAD_09
PADInC	R11	146	PHAD_08
PADInC	R10	147	PHAD_07
PADInC	R9	148	PHAD_06
PADInC	R8	149	PHAD_05
PADInC	R7	150	PHAD_04
PADInC	R6	151	PHAD_03
PADInC	R5	152	PHAD_02
PADInC	R4	153	PHAD_01
PADInC	R3	154	PHAD_00
PADVdd	R2	155	DVDD
PADAREF	R1	156	BIAS_RCVR
PADGnd	T52	157	DGND
PADAREF	T51	158	IRTN_HL_DISC
PADAREF	T50	159	OR_HL_DISC
PADAREF	T49	160	BIAS_DRV_H
PADAREF	T48	161	GAFE_HOLDP
PADAREF	T47	162	GAFE_HOLDM
PADAREF	T46	163	GAFE_STROBEP
PADAREF	T45	164	GAFE_STROBEM
PADAREF	T44	165	GAFE_DATP
PADAREF	T43	166	GAFE_DATM
PADAREF	T42	167	GAFE_CLKP
PADAREF	T41	168	GAFE_CLKM
PADAREF	T40	169	BIAS_DRV_L
PADInC	T39	170	GAFE_RET_DATA
PadOut	T38	171	GAFE_RST
PADOut	T37	172	ADC_CLK
PADOut	T36	173	NADC_CS
PADOut	T35	174	DAC_CLK
PADOut	T34	175	DAC_DATA
PADInC	T33	176	DAC_READBACK
PADOut	T32	177	NDAC_CS
PADOut	T31	178	NDAC_CLR
PADOut	T30	179	HITMAP_TEST

PADOut	T29	180	TRIG_TST
PADInC	T28	181	FREE_ID
PADInC	T27	182	PWR_ON_RST
PADOut	T26	183	VETO_ENA_TST
PADAREF	T25	184	LVDS_PRESETADJ
PADOut	T24	185	HV_ENABLE_1
PADOut	T23	186	HV_ENABLE_2
PADGnd	T22	187	DGND
PADAREF	T21	188	ACD_NVETO_17AP
PADAREF	T20	189	ACD_NVETO_17AM
PADAREF	T19	190	ACD_NVETO_17BP
PADAREF	T18	191	ACD_NVETO_17BM
PADAREF	T17	192	ACD_NVETO_16AP
PADAREF	T16	193	ACD_NVETO_16AM
PADAREF	T15	194	ACD_NVETO_16BP
PADAREF	T14	195	ACD_NVETO_16BM
PADAREF	T13	196	ACD_NVETO_15AP
PADAREF	T12	197	ACD_NVETO_15AM
PADAREF	T11	198	ACD_NVETO_15BP
PADAREF	T10	199	ACD_NVETO_15BM
PADAREF	T9	200	ACD_NVETO_14AP
PADAREF	T8	201	ACD_NVETO_14AM
PADAREF	T7	202	ACD_NVETO_14BP
PADAREF	T6	203	ACD_NVETO_14BM
PADAREF	T5	204	ACD_NVETO_13AP
PADAREF	T4	205	ACD_NVETO_13AM
PADAREF	T3	206	ACD_NVETO_13BP
PADAREF	T2	207	ACD_NVETO_13BM
PADVdd	T1	208	DVDD