Testing Structures and Procedures for the GLAST Tracker Front End Chip

S. Stromberg, M. Hirayama

***Abstract*- The GLAST satellite will use twenty thousand front-end readout chips. An efficient testing procedure was thus needed to check the functions of each chip before its installation into the satellite. This paper describes the hardware, software and procedures developed during the Summer REU program at the Santa Cruz Institute for Particle Physics.**

I. Introduction

GLAST, the Gamma-ray Large Area Space Telescope will be the next generation gamma ray observatory. It will convert high-energy gamma rays to charged particle anti-particle pairs and use silicon strip detectors to observe their trajectories. A calorimeter is also used to observe the energy of pair of particles and thus the gamma-ray energy.

The telescope consists of a five by five array of towers. Each tower has sixteen trays and a calorimeter base. The trays consist of two planes of detector strips oriented perpendicular to each other to create a two-dimensional read out image. The planes are composed of a five by five system of silicon strip detectors, SSD's, all oriented in the same direction. The detectors each have 320 strips and are bonded to the detectors immediately in front of and in back of it such that each strip is connected to the corresponding strip on the next detector. Thus each plane consists of five 320-strip ladders.

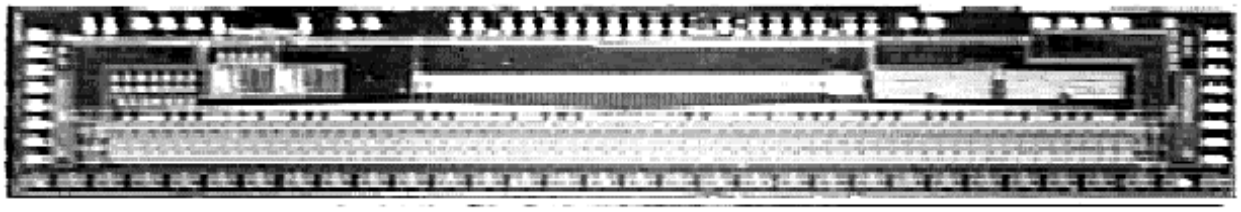
On one end of each plane the ladders are connected to a hybrid electronics board. It is called hybrid because it combines unpacked chips bonded directly to a circuit board. Each 320-strip ladder is directly bonded to five 64 channel front-end chips, the GTFE64 (GLAST Tracker Front End 64 channel).

The GTFE64 chip amplifies the signals from the SSD's, converts the signals to a bit pattern, sends out a trigger, and holds the data until it is called upon to read out the data. Each of the GTFE64's output and input functions are daisy chained, or bussed together with a controller chip at each end of the hybrid board. Two controllers are used for redundancy so that the data can be read out in either direction if there is a malfunction somewhere on the hybrid or on a chip. The controller chips communicate between the front-end chips and the tower control. The controller chips input and output functions are connected to corresponding functions on the chips directly above and below it. Again the connections are done by either daisy chain or bus depending on the function. Kapton, a flexible plastic circuit board material, is used to connect the controller chips and power functions of each plane.

II. The GLAST Tracker Front end Sixty Four Channel Chip

The primary function of the front end chip is to amplify the output of the SSD's and convert it to a digital bit pattern that can be serially read out to a controller chip when required. All of the input and output functions of the chip are sent and received by differential signal in order to cancel out any noise that may be picked up between chips. To convert to a bit pattern there is a discriminator[1] following each of the

[1]A discriminator is a switching device that switches its output between two control voltages depending on the voltage level of the input. It thus creates an on or off type output.



Grayscale image of GTFE64A: actual chip size 2.2mm by 11.7mm.

amplifiers. After the data is converted to a bit pattern it is latched which means that it is saved in memory buffers[2] until the data is called on to be read out.

♦ **Setting the Operating Conditions of the Chip**

Before normal operation several of the functions on the chip need to be preset. First each chip on a plane needs an address so that the controller chip can send command signals along a bus line. Each chip has two command line inputs one for the controller on the right and one for the controller on the left. The controller chip's command line output is connected to all the front end chips in the plane. Since there are 25 front-end chips we assign a five bit binary address to each chip. This way when the controller chip sends a command it first sends a start bit of 1 followed by a five bit address least significant bit first, and finally a three bit command. To assign the chip its address there are five address pads that are hard wired to on or off voltages on the hybrid. In addition to each chip's individual address there is a broadcasting address of 11111 that all chips on the plane will accept commands under.

A control register[3] is used in the front-end chip to hold its operational settings while the chip is reading out data. The timing clock for all the front-end chips is located on the controller chip, this saves power and allows all the chips to operate on the same timing cycle. To input new data into a chip's control register a signal must be sent to the command line input. The command for changing the control register's contents consists of a start bit, the chip address, the three bit command 001, and a 207 bit string of data for operational settings. The control register's data consists of three types of masks for each of the 64 channels, two 7 bit settings for digital to analog converters, and one bit for controller direction. The 001 command is unique to the chip in that if the chip is currently receiving its commands from one controller and the other controller sends this command the chip will still respond. This is necessary because it is this command that tells the chip which controller it is operating with. The control direction is set by the last bit of the 207-bit string, a 0 for left and a 1 for right. The default value is 0.

♦ **The Masks**

The three masks are used to test and turn off the functions of channels that are not working properly. The masks are the calibration mask, the channel mask, and the trigger mask. The calibration mask allows a controllable amount of charge to be placed on the masked channels of the chip. If a channel is selected for the calibration mask and the chip receives the 011 calibration strobe command then charge will be placed onto the masked channels. This allows us to test the other functions of the chip without having actual hits on those channels. To calibration mask a channel a one is placed on the corresponding bit of the 207 bit string being input into the control register. The first bit of the pattern corresponds to channel 0 and the 64th bit corresponds to channel 63.

The channel mask is used to ignore the data output of noisy or damaged channels. If the channel is mask value is 0 then the channel can still send triggers to the controller chip but its data output will always register as having no hits. In normal operation

[2] Buffering is what it is called when the output impedance of a stage is lowered so that its output voltage stays virtually constant while its output current can fluctuate. In the memory buffer the voltage level of the data is held constant in much the same way.

[3] The control register is a shift register, a digital device that reads and holds new data by shifting its old data over bit by bit. When the register reads a new bit, the oldest bit can be read at the register output. The read and shift routine takes place at the ticks of a timing clock.

a trigger mask almost always accompanies the channel mask. To set a channel mask a 0 is placed in the corresponding bit between the 65^{th} and 128^{th} bits of the 207-bit string. For the channel mask however the 65^{th} bit corresponds to channel 63, and the 128^{th} bit to channel 0.

The trigger mask is used to ignore the triggers of noisy or damaged channels. The trigger mask's job is to avoid having false triggers sent to the controller chip, yet still allowing the data of trigger masked channels to be collected when other channels trigger a significant event. Trigger masking is done in the same fashion as calibration masking but from bits 129 to 192 of the 207 bit string.

♦ **The Digital to Analog Converters.**

The digital to analog converters, DAC's, are used to set the voltage level of both the discriminator threshold and the calibration pulse. The calibration DAC's seven-bit string is located between bits 193 and 199 of the 207-bit control register string. The voltage value of the calibration pulse is set approximately by the following equation:

$$V=6.2+6.0xDAC.$$

Where the DAC value is the binary value (least significant bit first) of the last six bits of the 7-bit string. If the first bit of the 7-bit string is a 1 then the voltage value is multiplied by 4. If the chip receives the 011 calibration strobe command then the calibration DAC's voltage value is seen on all calibration masked channels.

The discriminators turn their output voltage to on whenever their input voltage goes higher than a preset voltage threshold level. Bits 200 to 206 of the control register input string set the threshold level. The voltage level of the threshold is set in a way similar to the calibration DAC level using the equation:

$$V=5.4+5.5xDAC.$$

Again the first bit of the 7-bit string multiplies the equation by approximately 4 when its value is 1. The threshold value is typically set to a level where noise is rarely seen but SSD hits show up clearly.

♦ **During Satellite Operation**

During operation signals coming in to the channel pads are amplified. If the amplified signal is larger than the threshold level the discriminator will be switched to its on voltage, and a trigger signal will be sent to either the chip on the immediate left or right depending on what control direction the chip was given when it was initialized. The chip will also send a trigger if it received a trigger from the chip immediately before it. Thus each chip has a 65 input fast-or gate, one input for each channel and one input for the previous chip's output. This connection arrangement of each chip's fast-or connected to the fast-or of the next chip is an example of the daisy chain.

The trigger will travel through each chip until it gets to the control chip at the end of the chain. The control chip receives the trigger and sends it on to the tower control. The control chip also takes a reading of how long the trigger pulse was. This is known as the time over threshold or TOT. The TOT can be used to tell us something about the source of the signal. Noise will give a short TOT while a charged particle will commonly give a much larger TOT.

♦ **The First in First out Memory Buffer and reading out data.**

If the tower controller registers enough triggers it will send a trigger acknowledge signal. When the trigger acknowledge signal is received by the front-end chip it latches its data and saves it in an 8 event first in first out (FIFO) memory buffer. The FIFO Operates by having two pointers, a read pointer and a write pointer. Every time we send a trigger acknowledge signal the write pointer inputs a 65-bit pattern from the discriminator outputs into the FIFO, 64-bits for the channels and an initial bit of whether the chip received any hits. Then the write pointer moves to the next position in the FIFO. This can be done eight times at which point the FIFO will be full. To read the data in the FIFO the 010 read event command needs to be sent to the chip. When the front-end chip receives this signal the FIFO read event pointer places the oldest line of the FIFO onto a data output shift register and the shift register begins moving the data serially towards the control chip.

The data output of one chip is directly connected to the data input of the next chip in a daisy chain. The controller chip must allow enough clocks to read out the data from the shift registers of all the chips it controls. After enough clocks have been sent for the wanted data to be read out the 111 end read event command is sent to the chip. This command disables the clock to the data output shift register.

If an unwanted line is latched in the FIFO the 100 clear event command can be sent. This command moves the read pointer down one event without reading out the current data line. If we would like to reset the entire FIFO we can do so by sending the 110 command. This command doesn't remove any previous data lines it only sets the read and write pointers to the same data line. This allows the data in the FIFO to be written over, and does not allow the old data to be read. If the need arises to reset the chip including the FIFO and all data registers, 101 the reset chip command can be sent. This returns the chip to its default settings.

III. Testing Procedures for the Front-end Chip

The following are the functions of the front-end chip that require testing.

- The trigger fast or function.
- Data in and out.
- The command functions, including the control register command.
- The chip reset function.
- The addressing of the chip.
- The first in first out memory buffer.
- The chip's zero suppression function.
- The Power consumption of the chip.

Each of these tests had to be performed twice, once for the left decoder and once for the right.

The automatic probing station at the Santa Cruz Institute for Particle Physics was used to interface the chip with exterior testing devices. A probe card was used to make contact between the bonding pads of the chip and the testing hardware. A Hewlett-Packard 16500A logic analyzer generated the input signals to the chip and read its outputs. A PC was used to write and send the control pattern to the logic analyzer via a GPIB cable. The logic analyzer sends TTL[4] logic signals out and the chip reads differential signals of different levels. For this reason a converter board that changed TTL to LVDS[5] signals and an interface board that changed the voltage levels of the LVDS signals had to be built. A parameter analyzer was used as a voltage source for the chip so that the chip's power consumption could be measured.

♦ The Hewlett-Packard 16500A Logic Analysis System

We used the pattern generator and the logic analyzer, the two primary functions of the logic analysis system for the testing. The pattern generator creates output patterns based on input data. Before inputting data the system must be initialized. This consists of naming each of the input and output channels used, setting the clock time, and setting the logic type. The computer program that was created for the testing procedure will initialize the logic analysis system on command. Data input is done by individually setting the output signal (1 or 0), for each channel of the pattern generator being used. The data output must be specified for every clock. Separate data lines are not required if the inputs for all the channels remain constant for more than one clock cycle. It is necessary only to specify for how many clocks the data will be repeated. A maximum of 4092 lines of data can be input into the pattern generator. The test developed uses 3950 lines for testing the functions of each direction and as a result data input and testing must be done separately for each side. The computer program used for testing translates simple commands from a text file into the bit patterns used by the pattern generator. In addition the program will enter the pattern into the pattern generator. Unfortunately the program loads one line at a time into the pattern generator instead of loading

[4] TTL is a form of logic used by bipolar transistor logic devices. The front-end chip is made of CMOS transistors not bipolar.

[5] LVDS stands for low voltage differential signal. LVDS is used both because the chip's functions are differential and differential signals can be sent over long distances without being susceptible to noise.

the pattern in a more efficient block data form. The data transmission for the test pattern of one side takes over an hour, but once loaded into the pattern generator can be saved to a floppy disk in block data form. This way we have two 3950 line patterns that we can switch between in seconds.

♦The Converter and Interface Boards

Signals coming out of the pattern generator are immediately converted to LVDS signals by a converter board. The LVDS signals travel approximately one foot via flat cable and are received by the chip interface board. The interface board converts the LVDS signals to the voltage levels the chip uses. The interface board also converts the output signals of the chip back to LVDS. The converter board then turns the signals back into TTL logic to be read by the logic analyzer.

Both the converter board and the interface board had to be custom built. A piece of G10 fiberglass with a grid of aluminum clad holes was used as a base plate for the board. Chip location for the boards was laid out and a power grid was designed to give the chips power. The chips used were all of surface mount technology and had to be soldered to adapters so that they could be placed into the grid. The adapters, like most common IC's contain two parallel lines of pins.

Power was brought to the IC's by laying copper foil between the parallel lines of pins and soldering the VDD connection to the foil. The geometry of the foil gives it a low current density, this gives the power connections a very low inductance. The rest of the board was then covered with the copper foil to create a ground plane. The ground pins of the IC were then soldered to the ground plane. Using a ground plane does more than just reduce the inductance to the chip, it also provides shielding to the transmission lines on the chip. The ground and power to the chip were wired in from an external voltage source and connections were made at the edge of the board. To lower the effect of fast switching to the VDD's 10μF capacitors were placed between the power voltages and ground at the power line connection. Such large capacitors have a large inductance so in addition to the 10μF capacitors, 0.01μF capacitors, having lower inductance, were connected between the VDD's of each chip and ground. All interconnections on the boards were made via wire wrapping, so that they could be easily changed if needed.

♦The Testing Software

The software developed for the testing procedure lets the user write commands to the chip using a text editor such as Windows notepad. The program translates the commands into the data input lines for the pattern generator. A copy of the final text program for the right decoder is attached. The program it is broken up into in the following manner:

```
•Test pattern for right decoder
    Test control register
        Test calibration mask
        Test channel mask
        Test trigger mask
        Test DAC's
    Test stop reading events
    Test FIFO
    Test Trigger right input⇒TRO
    Test address decoding
```

The program for the left decoder is identical accept that the control and trigger directions are reversed.

♦A Detailed Explanation of the Testing Program.

The final text file used for the testing (See Appendix) Starts with the command:

```
placePattern:  ADDR 17
```

this tells the pattern generator to hold the parallel voltages of outputs A0-A5 constant at the binary value 10001. Now the chip will only respond to commands addressed to chip 17 or to the broadcasting address11111.

The following commands are then input into the program:

```
address: 17
commandOn: right
ctrlDir: right
```

These are not commands sent to the pattern generator, they are reference values for the program. The "address: 17," line tells the program to use address 17 whenever commands are sent. "commandOn: right," Tells the program to only send the following

commands to the command right input. The program can allow commands to be sent to either of the command line inputs or to both. The "ctrlDir: right," tells the computer to set the control direction bit to the right value when ever a control register command is sent.

For testing the control register we want to input data into the control register and then read it out. Several more reference values need to be set for the control register. The reference values for the masks are set by:

```
mask: calib 0-63-3
mask: chan 1-63-3
mask: trig 2-36-3
```

The number sequence simply means starting with the first number, ending with the second number, and setting 1 to every third bit, three being the third number of the sequence. A 1 enables the function indicated by the mask. In addition the control register command requires values to be set for the DAC's, so the reference commands are put in:

```
threshDac: 13 high
calibDac: 14 high
```

The program will then input all the values in their correct places into the control register whenever "register:" appears in the program. In the control register test before we use the "register:" command we have the line "placePattern: RESET 1 0." This sends a pulse to the reset pad of the chip to reset the decoder. The command lines that are necessary to input new control register data after the input values are defined are:

```
placePattern: RESET 1 0
register:
*placePattern: CLKS 1*250 0
```

The "*placePattern: CLKS 1*250 0" lets the logic analyzer sample the outputs of the chip for 250 clocks and the * means that it is done at the same time as the command before it. This allows us to read out the output of the control register. At this point all we can read is the default settings of the chip, but if we send the last two commands again we can read out the control register pattern we just placed in the chip. Next we want to see if the reset chip command actually works so we send "resetChip:" to reset all of the values to the default settings. To make sure the default settings were placed in the chip we send the "register:" command again and expect to see the default settings at the output.

The calibration mask is used to simulate the chip getting hits. When the strobe command is sent to the chip all masked channels will receive the charge indicated by the calibration DAC. To test the calibration mask we do three separate tests each one masking every third channel. This way we test every channel's calibration mask. After we set the reference values for the control register and input the pattern into the chip we reset the FIFO. Resetting FIFO does not fill the FIFO with eight lines of 0's, it only re alines the read and write pointers. Resetting the FIFO is done with the commands:

```
resetFifo:
default: 256
```

The default command simply holds the addressing constant and sends no signals for 256 clocks while the control register is loaded. This is done because loading the control register is very noisy. The following commands then appear in the program:

```
strobe:
*placePattern: TACKR 0*12 1 0
*placePattern: CLKS 1*50 0
```

The strobe command puts the charge on the chip while both the logic analyzer samples the outputs and the trigger acknowledge command stores the 64 channel bit pattern into the FIFO. The data sampled during the 50 clocks should show a trigger pulse on the trigger right output. The trigger acknowledge pulse is delayed for 12 clocks to give the DAC and the discriminators more than enough time to register hits. To check that the calibration masks work we use the read event sequence:

```
readEvt:
*placePattern: DRI 0*10 1 0 1*2
   0 1*3 0 1*4 0
*placePattern: CLKR 1*100 0
*placePattern: CLKS 1*100 0
endRead:
```

The read event command places the first line of the FIFO onto the data output shift register. The CLKR sends the shift register its timing clock. While the shift register is outputting the chip's 64+1 channel data output the data right input is reading the DRI pattern. The sampling clock goes for 100 clocks which is more than enough time for the chip to output its data and for us to read our input pattern at its

output. The end read command makes the chip unresponsive to the shift register clock, that way the controller chip can keep its CLKR pulses going without the chip responding. This cycle is repeated three times in order to test every channel's calibration mask. Each test requires new "mask: calib" values, 0-63-3 for the first, 1-63-3 for the second, and 2-63-3 for the third. By examining the output data sampled by the logic analyzer we expect to see a hit on every third bit of the data left output.

The channel mask and trigger mask tests are done in much the same way as the calibration mask test. Each one consists of three different tests, each masking every third channel. For the channel mask test we enable every second and third channel (011011011…) and calibration mask the other channels:

```
mask: trig 0-63
mask: chan 1-63-3 2-63-3
mask: calib 0-63-3
```

When we examine the output data we expect to see triggers indicating that the calibration masks worked, but we should find only one 0 followed by our test pattern. This one 0 is an example of the use of the zero suppression bit, the 65th bit. Whenever a chip receives no hits instead of sending a long string of zeros it only needs to send one. This cuts down on readout time for the chip.

For the trigger mask tests the same method is used, we enable the triggers of all the channels accept for the channels where we enable the calibration mask:

```
mask: trig 1-63-3 2-63-3
mask: chan 0-63
mask: calib 0-63-3
```

The data we read out from this test should show a hit on every third channel followed by our test pattern, however there should be no triggers registered.

The DAC's are tested by first setting the calibration DAC high enough above the threshold DAC so that we read a hit on every channel of the chip. The second DAC test consists of setting the control DAC low enough so that we have no hits on the read out data. The same commands are used as in the above tests to set the values and read out the data.

The next test is the stop read event test. It works simply by placing a hit onto every third channel of the chip. Then the read event sequence is given however only 30 clocks are given to the chip and then the end read event command is sent. After the end read event command the sampling clock and CLKR continue working as before for 70 more clocks. The data that we expect to read out is the first 30 channels and then 70 zeros. After the seventy zeros we reset the FIFO and again read out the data, this time reading out all the data.

The FIFO test is done by inputting 8 events into the FIFO and then reading 8 events out. To distinguish the events from each other each event has a different channel mask pattern:

```
mask: chan 0-63-8
```

for the first event,

```
mask: chan 1-63-8
```

for the second event, and so on. These events are then read out in the usual way. In addition to this test we reset the FIFO, scipp two events, and read out the third event. To scipp an event we use the command:

```
clearFifo:
```

This just moves the read pointer down one event without reading the event.

The trigger right input to trigger right output test was done by simply placing our test pattern on the TRI pad and sampling the output of the TRO pad to make sure we see our pattern.

The addressing functions of the chip were tested by assigning the chip an address then sending the normal test commands to that address. The outputs of the chip were sampled to make sure that the chip responds. After each address was tested commands were sent to address 17 while the chip was still assigned to another address. No response was expected for these commands. Not all 32 of the different address possibilities were checked. All 0's were checked as well as the five combinations of one 1 and four 0's. The broadcasting address was checked by assigning the chip address 17 and sending commands to address 31. We expect to read out a response for these commands when we check the data.

♦How to Check the Data

The program reads the sampled data and saves it to a text file. A sample of one page of a text file

can be found in the appendix. It lists the number of the clock sample and the sampled values for trigger right output, trigger left output, the differential signal outputs for the data left and right outputs, and the control register output. Sixty-five pages of data are printed out listing the outputs of 3949 lines of data. Without an efficient way to check the data for the twenty-thousand chips the test would be somewhat useless.

To quickly check the output files a template file was created. The template file is a text file similar to the output files of the chips, except that it is perfect and has X's for output values that don't matter. The testing program will compare the template file to the output files. The comparison makes sure that the output file has 1's and 0's in the same positions as the template file. Every line that doesn't match the template is marked and saved to a text file to be examined. Sample pages of the template and error files can also be found in the appendix.

IV. Testing Results

Thirty-eight chips were tested from two different prototypes, the GTFE64A and the GTFE64B. Of the thirty-eight chips only four were found to have problems, and one chip was abused so much during testing that we recommended it not be used. Several of the chips had minor scratches that didn't have any effect on their performance. We found that contact between the probe tips and the bonding pads is crucial to the testing results. The tips may appear to be making contact but the results may indicate that the chip needs to be repositioned. Repositioning the chip is the first thing we do when we find bad results. Other typical errors were trigger noise, indicated by one or more zeros in a trigger pulse. When trigger noise is found we immediately retest the chip and see if the noise disappears. It normally would. The results of the testing are summarized in the following table.

GLAST Front End Chip Test Results

Chip #	AIDD mA @5V	AIDD2 mA @2V	DIDD µA @3V	Power mW	Works?	Comments
A2	1.10mA	$9*10^{-3}$	>2000	>11	No	This chip was crushed with the probe tips.
A6					Yes*	*This is the chip we used to debug the system, and should be avoided.
A10	1.29	1.47	>2000	>15	No	The Power drain is too large.
A16	1.23	1.59	400	10	Yes	Scratches on ch.57 pad and near ch39.
A17	1.35	1.40	440	11	Yes	Spot on line to FIFO and int. pad 25.
A19	1.30	1.45	430	11	Yes	Scratch on ch.46 and in free space by DAC.
A20	1.26	1.53	410	11	Yes	Scratch on ch.58
A21	1.34	1.41	440	11	Yes	
A22	1.31	1.53	460	11	Yes	
A23	1.34	1.38	430	11	Yes	Spot found near discriminator and free space.
A24	1.27	1.61	400	11	No	Tested several times and found to be bad.

A25	1.30	1.55	430	11	Yes	Scratch around power array on DAC.
A26	1.25	1.52	430	11	Yes	Scratch on ch. pads 4 and 11 and near FIFO.
A27	1.29	1.53	400	11	Yes	
B1	1.35	1.49	420	11	Yes	
B2	1.32	1.53	440	11	Yes	Scratch on ch. Pad 3
B3	1.32	1.50	430	11	Yes	
B4	1.33	1.50	430	11	Yes	
B5	1.32	1.50	430	11	Yes	Originally looked bad, but more contact gave good results.
B6	1.35	1.47	430	11	No	Many scratches, edge destroyed.
B10	1.30	1.51	440	11	Yes	
B11	1.31	1.50	440	11	Yes	
B12	1.31	1.52	450	11	Yes	
B13	1.34	1.50	480	11	Yes	Trigger noise. Not too unusual.
B14	1.34	1.51	430	11	Yes	
B15	1.33	1.47	450	11	Yes	There are some marks on TROP pad.
B16	1.35	1.50	430	11	Yes	Trigger noise.
B17	1.35	1.50	440	11	Yes	Trigger noise.
B18	1.33	1.51	420	11	Yes	
B19	1.33	1.46	440	11	Yes	
B20	1.32	1.51	440	11	Yes	
B21	1.35	1.48	430	11	Yes	Scratch on ch. Pad 60
B22	1.31	1.49	450	11	Yes	Trigger noise.
B23	1.31	1.51	440	11	Yes	
B24	1.35	1.46	430	11	Yes	
B25	1.32	1.51	440	11	Yes	Trigger noise.
B26	1.29	1.50	420	11	Yes	
B27	1.33	1.49	420	11	Yes	