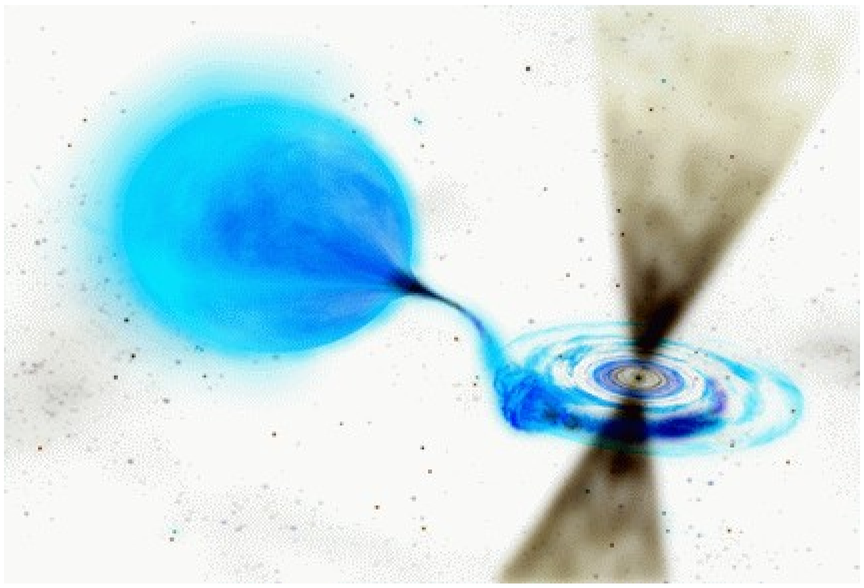

TRANSIENT EVENT

CAPTURE β

CICADA Note 7: User's Guide

Patrick Brandt
Brandon Rumberg

Table of Contents

Index of Tables

Other Figures

0. Change Log

Revision	Log	Author	Date
1.1	Revised for CICADA Note series.	P. Brandt	2007-August-9
1.0	Creation	P. Brandt	2007-July-20

1. Abstract

This document describes the use of the Event Capture β design, although most of the functionality pertains to both the α and β design. The User's Guide explains the parameters used to identify events and initiate event capturing. In this guide, it is assumed that the IBOB is connected to a Linux computer via a dedicated 10 Mbps Ethernet link and a serial cable. The user must be logged on to the computer that has the these connections to the IBOB.

The Event Capture design takes either two or four polarized inputs from a radio telescope and captures short term (~6 µs) time series, based on detection of very short term (~10 ns) transients. Further motivations and applications for this project have been described in CICADA Notes 1 through 4 located at:

https://wikio.nrao.edu/bin/view/CICADA/CicadaNotes

2. Functional Overview

This section provides an overview of the functional design of the EventCap profile to improve understanding of how the options and configuration settings affect the detection and recording of events.

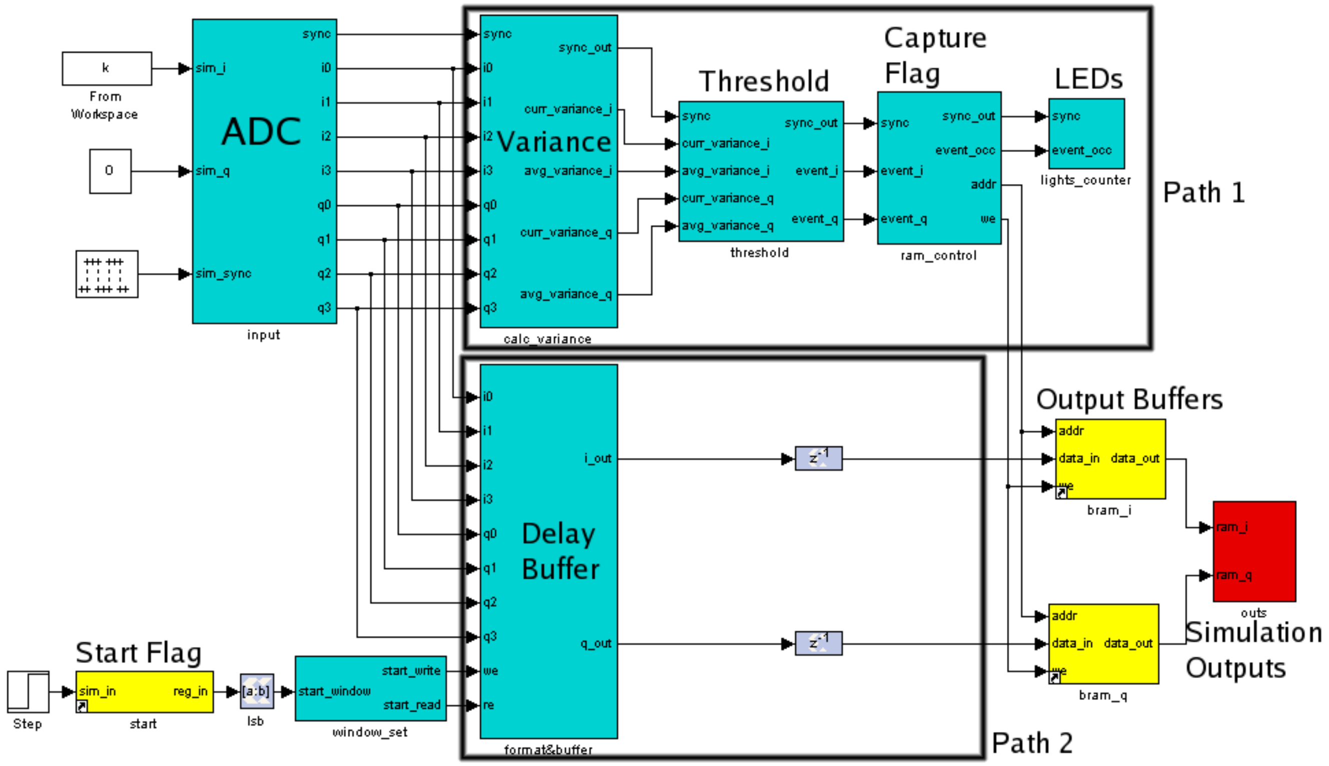

Figure 1: EventCap β overview

The above diagram functionally describes three configurations of the Event Capture β design that provide a different number of input signals and sampling rates; however, the detection and capturing concepts are the same for all of these variations.

Event Capture contains two parallel data paths. The first path, pictured at the top of Figure 1, implements the detection and event processing components. The second implements buffering and storing the data in a more efficient packaged format for transmission to the recording computer. This concatenation is due to the disparity in word size; the ADC digitizes in an 8 bit resolution, but the register and BRAM blocks are 32 bits, thus each memory location holds four data points.

3. Operation

This section provides a brief overview of the setup and use of the Event Capture profile for the IBOB.

3.1 Hardware

Install the Event Capture profile using either the BEE_XPS program in MATLAB (writes to flash memory) or the Impact software (writes to PROM, refer to the IBOB PROM Burning document for more information). Connect the serial/Ethernet cable between the IBOB and the Linux machine with the support software installed. In the test setup two Linux machines were used, radar.gb.nrao.edu (connected via Ethernet) and volans.gb.nrao.edu (connected via serial).

3.2 Clock and Timing Signals

The Event Capture β design requires an external sine wave clock signal for the ADC card, in the range of 400-1000 MHz at -3 to 0 dBm. The absolute time of the captured events is determined by synchronization of the 1 PPS signal with the internal IBOB system clock.

3.3 RF Inputs

The RF inputs to the ADC card should be bandwidth limited, -30 dBm, broadband signals. The input signal bandwidth must be in the range of the clock frequency to twice the clock frequency. Alternatively the input signal could be in the range zero to the clock frequency.

4. IBOB Software

The use of the Event Capture requires first setting the parameters for capturing transients, and then transferring the data to the recording computer. There are two ways to set up the event identification. One way is to log onto the IBOB and set the registers manually using regwrite. The second way is to use a Linux program, ecconf, on the computer that is connected to the IBOB via the serial port.

4.1 Tinyshell Setup

Start minicom to connect to the IBOB (or telnet if using Ethernet). For basic data collection, the pertinent registers are *bias_rate_i*, *bias_rate_q*, *inhibit*, *length_i*, *length_q*, *reset*, *start*, *variance_i*, *variance_q*, and *window_time*.

4.1.1 bias_rate

calc_variance/bias_rate_<i, q>
The *bias_rate* registers are 16 bits with 15 bits of dedicated fractional precision. These values determine the rate of change in the bias detection. The bias determination and removal are performed automatically. Typical values for this register range from 2^{-10} (32 explicitly in memory) to 2^{-8} (128 explicitly in memory). A lower value causes the bias value to change more slowly, and be less sensitive to short term changes in the input signal (events).

To set the *bias_rate* registers, use the regwrite command:

Example 1: Bias rate register
```
IBOB % regwrite calc_variance/bias_rate_i 32
IBOB %                                          #Produces a bias_rate of 2^(-10)
```

Other registers can be written in a similar fashion by using:

Example 2: Generic register setting
```
IBOB % regwrite <subsystem/><register_name> <value>
```

Note that using the regwrite command does not take into account the interpreted representation (such as binary point location), so to store the number 1 in an 8 bit register with 7 bits of fractional precision, you would enter the value as $(127)_{10}$ or $(7F)_{16}$, assuming signed 2s complement representation (note that this is not exactly 1, but rather $1-2^{-7}$).

4.1.2 inhibit

ram_control/inhibit
The inhibit register is a boolean (0 or 1) value that disables or enables event capture capabilities without resetting memory address counters (see *reset*).

4.1.3 length

calc_variance/length
The *length* registers are 8 bit numbers which indicate the interval between the σ_{calc} samples (L). Increasing this number will make the long average comparison value less sensitive to short term changes in the input signal (events).

4.1.4 reset

ram_control/reset
The reset register is a boolean (0 or 1) value that toggles the reset pins of all BRAM address counters. In order to reset the RAM blocks (and thus the capture of events), simply write a 0 followed by a 1 to the reset register.

Example 3: Reset register
```
IBOB % regwrite ram_control/reset 0
IBOB % regwrite ram_control/reset 1
```

4.1.5 start

start
To commence capturing transient events, set the *start* register to 1. Setting the *start* register to 0 stops the event capture design.

4.1.6 variance

threshold/variance_<i, q>
The formula used to evaluate signals for events can be written as:

$$\sigma_{calc}(m)=\sum_{n=0}^{N-1} X_m(n)^2-(\sum_{n=0}^{N-1} X_m(n))^2 \quad (1)$$

$$\sigma_{current} ?> \frac{\sum_{m=0}^{M-1} \sigma_{calc}(m*L)}{M} * T \quad (2)$$

Figure 2: Variance equations

where:

- *X(n)* represents a vector of *N* data points of the incoming signal
- σ_{calc} represents the variance calculated in the signal
- *M* is the number of σ_{calc} samples to be averaged
- *L* represents the length parameter specified by the *length* register in the *calc_variance* sub-block, which is the time between consecutive σ_{calc} samples (see the following sections)
- *T* represents the variance threshold as entered by the user.

In simpler terms:

$$\sigma_{current} ?> \sigma_{average} * T \quad (3)$$

Figure 3: Simpler variance equation

The *variance* registers are 15 bits with 4 bits of dedicated fractional precision. These values are used to determine the range of events captured. Typical variance values range from 1 (entered and represented in memory as 16) to 10 (entered and represented in memory as 160).

4.1.7 window_time

window_set/window_time
The delay between the detection path and the FIFO buffering path are such that, when an event is detected and the RAM control enabled, the address write enable signal will come *window_time* samples (see the following sections) before the event spur leaves the FIFO buffer.

The *window_time* register determines the size of the data set to be acquired. The number of data points captured is *8*(window_time)* per polarization. For example:

Example 4: Window time register
```
IBOB % regwrite window_set/window_time 200
```

would produce 1600 data points per polarization (~2µsec with an 800MHz ADC clock/200MHz IBOB system clock).

Note: The *window_time* register must be set while the *start* register is set to zero. Currently, setting the *window_time* register after setting the *start* register to 1 (or setting the *start* register to 1 and then 0) will usually result in erroneous data or non-centered events. The window is the only parameter which may not be changed once the start command has been issued; all others are variable at any time.

The Event Capture profile also supports the Ethernet LWIP interface, and is currently supported by the alpha program IPCap. IPCap performs the same function as PickIt with no options. Both are described in later sections.

5. Support Software Functionality

This section describes the support software associated with the EventCap profile for the IBOB.

5.1 IBOB Configuration

Configuring the EventCap profile once it has been loaded onto the IBOB can also be accomplished through the *ecconf* program. *ecconf* is a simple routine that connects via the serial port to the IBOB and sets the registers specified by the user.

The *ecconf* program requires a connection to the IBOB via a serial port. In the test setup this computer was volans.gb.nrao.edu.

Example 5: EventCap configurator

```
$ ecconf
Usage: ecconf [option [value]]...
      bias        <value>     Set the bias parameter for both polarizations.
      bias_i      <value>     Set the bias rate parameter for polarization i.
      bias_q      <value>     Set the bias rate parameter for polarization q.
      inhibit     <value>     Enable or disable capturing of events.
      reset                   Resets BRAM address counter logic.
      sim         <value>     Set simulation mode on (1) or off (0).
      start                   Starts event captures.
      stop                    Stops event captures.
      variance    <value>     Set the variance detection level.
      variance_i  <value>     Set the variance detection level for polarization i.
      variance_q  <value>     Set the variance detection level for polarization q.
      window      <value>     Set the window size (# of data points) to capture.
```

5.1.1 bias

The *bias* values are signed 16 bit numbers with15 bits of dedicated fractional precision. The bias values indicate the rate of change of the bias detection, and the bias determination and removal are performed automatically. Typical values are on the order of 2^{-10} to 2^{-8}.

5.1.2 inhibit

The *inhibit* value enables (1) or disables (0) capturing of events, but not reset RAM address counter logic.

5.1.3 reset

The *reset* command is used to reset the BRAM address counter logic, and allows more events to be captured.

5.1.4 sim

The *sim* register is a boolean (0 or 1) register that determines the input of the EventCap design. In simulation mode (1), the data is read from a set of input BRAMs rather than the ADC, and mock signals or events can be loaded into these BRAMs for testing purposes.

5.1.5 start/stop

The *start* and *stop* commands control the operation of the design.

5.1.6 variance

The *variance* values are unsigned 15 bits with 4 bits of dedicated fractional precision which indicate how many "sigma" (σ) an event must be from the RMS noise level to be detected. Typical values range between 1 and 10.

5.1.7 window

The *window* value sets the number of data points collected per event per polarization.

Note: Currently the window must be set before the start command is given or the event data may become erroneous or improperly centered. Setting the window while start is enabled will do nothing, but if start is disabled and re-enabled, the data will be incorrect. The window is the only parameter which may not be changed once the start command has been issued; all others are variable at any time.

Example 6: ecconf – Example of execution

```
$ ecconf start variance 7 sim 0 window 9600
regwrite threshold/variance_i 112
regwrite threshold/variance_q 112
regwrite window_set/window_time 1200
regwrite input/sim 0
regwrite start 1
```

5.2 Data Acquisition

Once the IBOB is configured, it is ready to gather event data to transfer off of the storage machine. To transfer the data to permanent storage, another program called *pickit* is used. PickIt is a polling program that periodically checks the IBOB's state to see if event data is available. If a transient has been captured, PickIt transfers several timing parameters and the voltage measurements from both polarizations into a data file on the storage machine via the serial link.

PickIt requires a connection to the IBOB via a serial port. In the test setup this computer was volans.gb.nrao.edu. The TCP/IP clone of PickIt, called IPCap, uses the 10 Mbps Ethernet connection rather than the serial port to connect and transfer data from the IBOB to the recording computer (volans.gb.nrao.edu in the previous case). IPCap does not have any options. In the test setup, the computer running IPCap was radar.gb.nrao.edu.

Example 7: PickIt

```
$ pickit -help
Usage: pickit [-b] [-p] [-s <time>] [-u <time>] [-help]
      -b          Obtain background plots every minute (not precise).
      -p          Plot output file using Gnu Plot.
                  (Persistent windows, overwhelming with many events)
      -s <time>   Set the sleep time between polls. (seconds)
      -u <time>   Set the sleep time between polls. (microseconds)
      -help       Display this usage information.
```

The *-b* option for pickit enables capturing of "background plots" which contain only ambient noise. These plots can be used to see what kind of RFI may be present in actual transients to differentiate between the interference and true signals.

The *-p* option simply plots the incoming signals using Gnu Plot. Note that this uses persistent windows, and frequent periodic sources or high levels of RFI will cause your screen to be coated with graphs.

The *-s* and *-u* options can be used to increase the poll period and decrease system load if the events are expected to be farther apart in time.

The unmodified cycle time of PickIt depends primarily on the number of data points the EventCap profile is acquiring (see above regarding *window* in ecconf and the *window_time* register). At 4800 data points per polarization (6µs of data with an 800MHz ADC clock/200MHz IBOB clock), the cycle time is approximately 0.5 seconds when using the serial link.

IPCap was able to achieve a cycle time of ~60 ms with an effective data rate of ~1.3 Mbps.

There are two versions of both pickit and ipcap, one for the two input designs (both one and two ADC cards), and another for the four input design. The pickit designs are named pickit and pickit4p, and the ipcap designs are named ipcap and ipcap4p.

6. IBOB LED Panel

Table 1: Interpreting the LEDs

	Far Left	Left	Right	Far Right
Top	*Not used in β version.*	*event_flash:* Lights when an event has occurred and is waiting to be read.	*thresh_sat_i:* The brightness of this light shows how close polarization I is to the threshold.	*thresh_sat_q:* The brightness of this light shows how close polarization Q is to the threshold.
Bottom	*flash (sync):* Flashes with the sync pulse.	*Not used in β version.*	*adc_sat_i:* The brightness of this light shows how close polarization I is to saturating the ADC.	*adc_sat_q:* The brightness of this light shows how close polarization Q is to saturating the ADC.

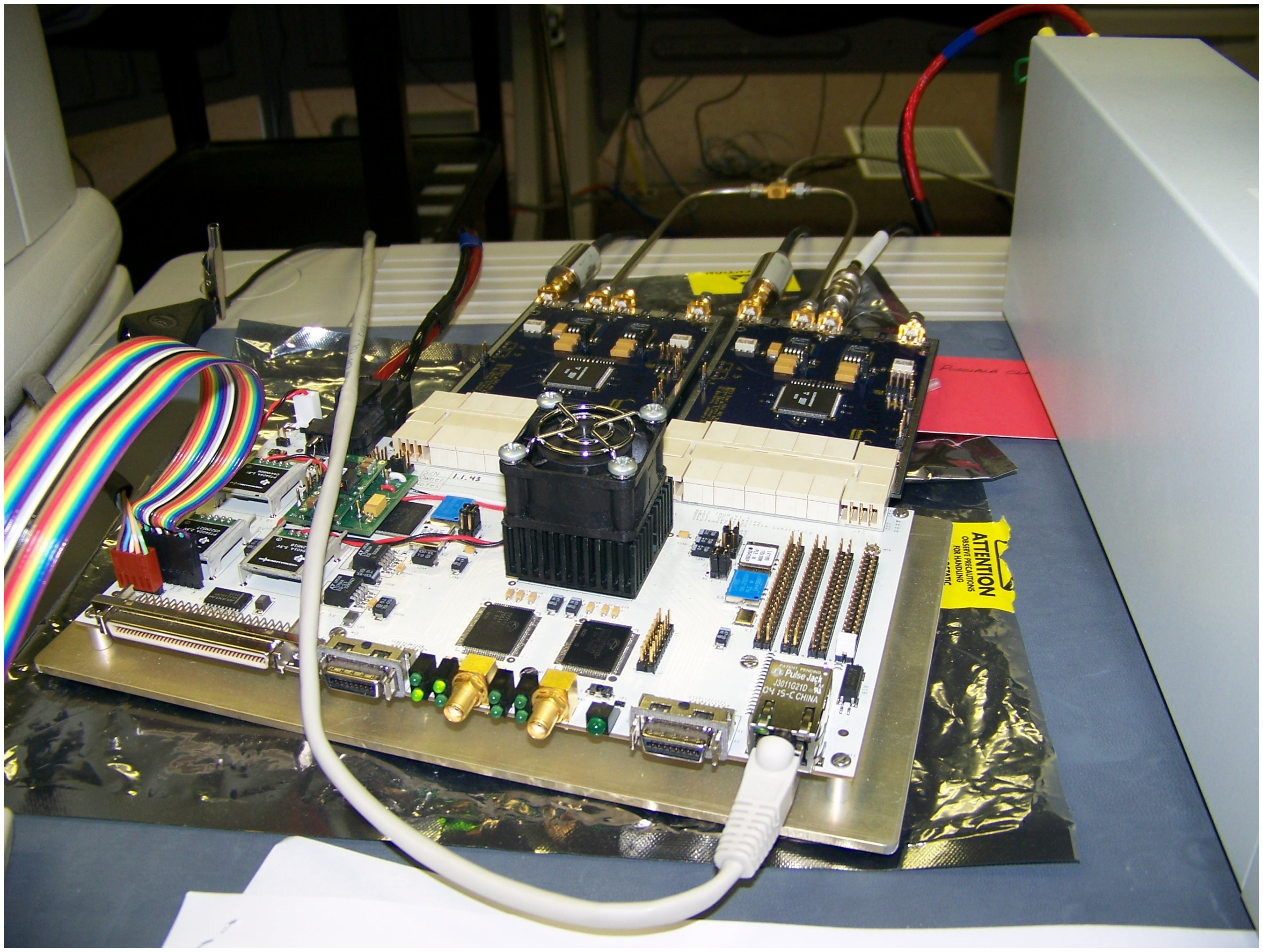
Figure 4: IBOB with 2 ADCs, Ethernet, and serial cable connected

7. Output Format

The output files generated by either pickit or ipcap are in an ASCII format shown below:

```
# SYN: 10651 790248783 RT: 10655 4242723491 EVT: 10651 901037099
# DATE: 2007-08-06T12:20:03.41
0         0.023438   -0.031250
1         0.109375    0.046875
2        -0.078125   -0.039062
3        -0.101562   -0.054688
4         0.062500    0.023438
5         0.023438   -0.101562
6         0.000000   -0.039062
7         0.046875    0.039062
8        -0.046875   -0.085938
9        -0.031250    0.046875
...
```

The tags at the top of each file represent the timing parameters associated with the event in the file. `SYN` is the IBOB clock sample of the last latched 1 PPS pulse, `RT` is the clock pulse latched when the data dump was initiated, and `EVT` is the clock pulse latched when the event was detected (corresponding the midpoint of the data file). The date is generated by the Linux machine at the time of writing the data file. The interval between samples is $1/f_{clock}$ or $1/(2*f_{clock})$ for one ADC card or two ADC cards, respectively (both for 2 input channels). The other data format for the four input design would feature four data columns rather than the two shown above.

8. Data Display

The data files can be visualized simply using gnuplot or eventFft. eventFft is described in further detail in CICADA Note 4. Using the gnuplot subcommand *plot* and *splot* (respectively), the following figures were produced:

These example plots were generated with data obtained from the 43m telescope attached to the IBOB via two ADCs in the two input mode. The input clock was set to 800 MHz. The input IF signal was centered at 1250 MHz with a bandwidth of 550 MHz. The first data column is sample number. The second data column was H polarization and the third data column was V polarization. The strong event detected was from aircraft radar altimeter signals at ~1090 MHz.

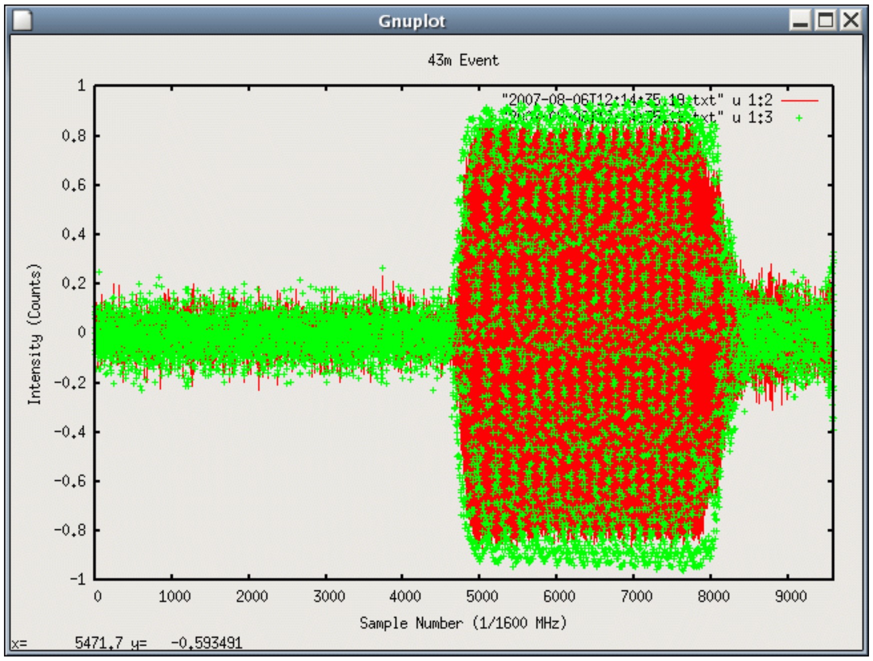

Figure 5 : 2D plot of a captured event; X axis is sample number, Y axis is voltage

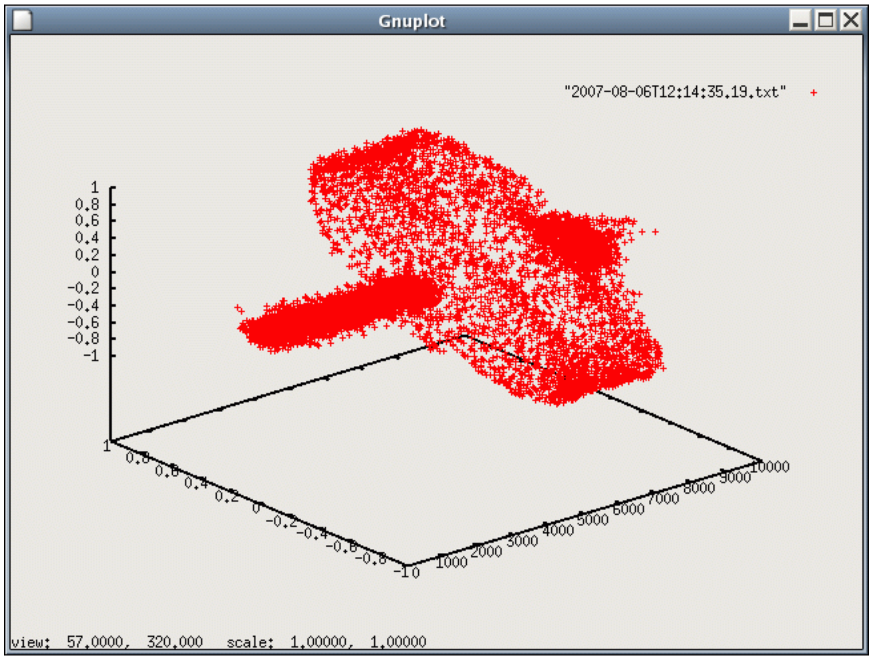

Figure 6: 3D plot of the same data as above