In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of the College of Science and Engineering, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____ Date _____

# DEVELOPMENT OF AN AUTOMATED TESTING AND CALIBRATION SOFTWARE PACKAGE FOR THE NIFFTE TPC PROJECT

by

Brycen Wendt
wendbryc@isu.edu

Department of Nuclear Engineering
Idaho State University

A thesis
submitted in partial fulfillment
of the requirements for the degree of
Master of Science in the Department of Nuclear Engineering
Idaho State University

Summer 2015

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Brycen Wendt find it satisfactory and recommend that it be accepted.

Eric Burgett
Major Advisor

Tony Hill
Committee Member

David Beard
Graduate Faculty Representative

For Tanya

warm boundless Love
life, laughter, devotion, tears
Mother, Wife, my all

**Acknowledgments**

First, I would like to thank my academic advisor, Dr. Eric Burgett. He provides me with a wide variety of projects and challenges that developed my capabilities as a student, a nuclear engineer, a professional, and a leader. I have a great respect for his comprehensive knowledge. His dedication and work ethic inspire me to forge onward and upward.

Second, Dr. Tony Hill deserves recognition not only for his leadership in the NIFFTE collaboration, but for his very personal involvement in my success. I frequently depended on his explanation of this work's underlying physics and mathematics. His experience and guidance have shaped this project.

I want to recognize the assistance provided by Dr. Keith Jewell, who contributed often with his step-by-step debugging and troubleshooting. His experience with instrumentation prevented me from getting lost in a maze of wires, cabling, and circuitry.

At the project's inception I knew very little about *how* I was to accomplish my objectives. Two men to help me gather inertia. Dr. Mike Heffner was always quick to respond and answer any question I had. Dr. Vincent Riot, the original EtherDaqGUI developer, provided the solution when my initial steps at modification broke everything.

I cannot ignore the lifelong contribution of my parents. I have great respect for their experience, advice, and wisdom. Both have set an example of life-long learning and self-education; from my childhood on they fostered my love of learning. They continued to show their love and support even after I left "the nest" to start my own family.

Finally, I would also like to acknowledge the incredible NEUP fellowship funded by DOE (credits on page 126). I would not be writing this now without the incredible financial assistance the fellowship provided.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| ADC | analog to digital converter xiii, 4, 5, 15, 82, 107 |
| API | application programming interface 43–46, 58, 64, 73, 109 |
| ASIC | application-specific integrated circuit 4 |
| BNC | Bayonet Neill-Concelman xiii, 9, 10, 12, 14, 36 |
| CPU | central processing unit 63 |
| CRTP | curiously recurring template pattern 45–47, 62–64 |
| CSV | comma separated value 41, 72 |
| DAQ | digital acquisition xiii, 2, 11, 24, 90 |
| DBB | digital bus board 11 |
| DOE | Department of Energy v, 90, 126 |
| EMI | electromagnetic interference 9, 11 |
| FPGA | field-programmable gate array 5, 36 |
| GUI | graphical user interface 13, 19, 24, 43, 44, 48, 73, 94 |
| ID | identification 30, 31, 40 |
| IDE | integrated development environment 18 |
| IP | internet protocol 24, 27, 94 |
| ISU | Idaho State University i, ii, 10 |
| IUP | Integrated University Program 126 |
| LANL | Los Alamos National Laboratory 13, 16 |
| LED | light-emitting diode 27 |
| LLNL | Lawrence Livermore National Laboratory 9, 18 |
| MAC | media access control 20, 24, 94 |
| NEUP | Nuclear Energy University Program v |

NIFFTE Neutron Induced Fission Fragment Tracking Experiment ii, v, xiii, xvi, 3–5, 7, 14, 42, 60, 86, 89, 90
NIST National Institute of Standards and Technology 90

OOP object-oriented programming 43–45

PCB printed circuit board xiv, 5, 6
POST power-on self test 20

RC resistor-capacitor 68, 69, 75–78
RMS residual mean square 70

SDK software development kit 27, 28
SN serial number 30, 31, 40
SQL structured query language 89
SRS Stanford Research Systems 10, 13, 27, 103, 104
STL standard template library xv
SVN Apache Subversion 24, 108

TPC time projection chamber ii, xiii, xvi, 2–8, 10–12, 14, 20, 65, 86, 87, 89, 90
TTL transistor-transistor logic 36

URL uniform resource locator 94
USB universal serial bus 27

# Glossary

**.dll** a file containing a shared library of compiled functions which can be loaded and used by an external process

**bank** **1** a BNC connector on the front of the test stand box **2** a bar on the distribution board that splits a pulse into eight separate signals

**bit** the smallest unit of information in a computer, represented by either 0 or 1

**C** a general-purpose programming language developed by Dennis Ritchie at AT&T Bell Labs

**channel** **1** a single charge collection unit in the NIFFTE TPC **2** an isolated amplification circuit on a preamp card, usually connected to a NIFFTE TPC channel

**code** (*see* source code)

**const** a C++ keyword that results in unchanging behavior in the context to which it is applied, usually used to declare fixed-value data member; the website `http://duramecho.com/ComputerInformation/WhyHowCppConst.html` extensively documents the multiple meanings and contexts of use

**C++** a derivative language of C that provides object-oriented and generic programming features

**E/P** EtherDAQ/preamp, i.e. a combined unit containing one EtherDAQ card and one preamp

**EtherDAQ** a term, unique to the NIFFTE collaboration, that means Ethernet digital acquisition (*usually* EtherDAQ card)

**Ethernet** a system for connecting a number of computer systems to form a local area network, with protocols to control the passing of information while avoiding simultaneous transmission by two or more systems

**gain** the quotient of a pulse's voltage amplitude under the corresponding waveform's extracted ADC charge parameter

**H(n,n)H** a purely elastic reaction between a hydrogen nucleus (i.e. a proton) and a neutron

**managed** code that contains extended features, such as security and memory management, which are 'managed' by an additional runtime component that runs transparently in the background (*compare* unmanaged)

**mask** an ordered sequence of bits that are used to toggle on or off entities or actions mapped against the individual bit positions

**measurand** a quantity of interest that is not measured directly, but determined from other actually measured quantities

**mezzanine board** a PCB used simply as a bridge or link between two separate electronic components, often PCBs themselves

**NaN** not a number, a floating-point representation of an undefined value

**packet** a segment of data sent from one network device to another

**pedestal** the flat front-half of a waveform that contains the low data points (*compare* plateau)

**phase** a step of the testing procedure characterized by pulses sent to the test stand input banks in a specific temporal order

**plateau** the sloped back-half of a waveform that contains the high data points (*compare* pedestal)

**$Pu^{239}$** Plutonium 239

**pulse** a voltage signal produced by the pulse generator

**rise** (*see* transition)

**signed** a computer variable that allows number representation in both the positive and negative ranges (*compare* unsigned)

**source** (*see* source code)

**source code** human readable instructions that are translated by a compiler, interpreter, or assembler into object code which can be executed on a computer (*also* code, source)

**subphase** a part of a phase, characterized as a unique group of pulses with a consistent voltage amplitude

**thread** a small sequence of programmed instructions that can be managed independently, but shares the resources of the parent process

**transition** the central portion of a waveform that contains the transition from the pedestal to the plateau (*also* rise)

**unmanaged** code that is compiled directly into machine instructions, enabling faster operation but at the expense of extended features such as security and memory management (*compare* managed)

**unsigned** a computer variable that allows only positive number representation (*compare* signed)

**$U^{235}$** Uranium 235
**$U^{238}$** Uranium 238

**vector** a dynamic array class, part of the the C++ STL
**Visual C♯** an object-oriented programming language, developed by Microsoft as one of the .NET Framework languages, that aims to combine the computing power of C++ with the programming ease of Visual Basic

**waveform** the digital representation of a pulse

**Abstract**

The NIFFTE collaboration seeks to make nuclear data measurements with uncertainties below 1.0% using a time projection chamber (TPC). To achieve this goal, each of the 5952 signal collection and processing channels of the TPC must be routinely calibrated. The calibration constants are fundamental to the TPC data collection effort. The TPC channels are distributed amongst 192 32-channel cards specifically designed for the challenges of the project.

A diagnostic platform has been developed to interrogate the 32-channel cards over their full dynamic range by leveraging existing software, originally designed as a testing interface during card development, and enhancing the functionality to provide an appropriate data collection platform.

An independent analysis framework was implemented to provide the infrastructure required to carry out bulk calibration analyses of the vast amount of channel-by-channel calibration data collected.

# Chapter 1

# Introduction

## 1.1 Background [1]

Reactors, weapons, and nucleosynthesis calculations are all dependent on nuclear physics for cross sections and particle kinematics. These applications are very sensitive to the nuclear physics in the fast neutron energy region and, therefore, have overlaps in nuclear data needs. Computer codes interface to nuclear data through nuclear data libraries, which are a culmination of experimental results and nuclear theory and modeling. Uncertainties in the data contained in these libraries propagate into uncertainties in calculated performance parameters.

The impact of nuclear data uncertainties has been studied in detail for reactor and weapon systems, and sensitivity codes have subsequently been developed that provide nuclear data accuracy requirements. Sensitivity studies have provided specific requirements for uncertainties on many fission cross sections, many of which are beyond the reach of current experimental tools.

The proposed method to obtain high-accuracy precision fission measurements is to employ a TPC[1] and perform neutron energy dependent fission cross measurements relative to H(n,n)H elastic scattering. The TPC is the perfect tool for minimizing most of the systematic errors associated with fission measurements—the technology that has been in use in high-energy physics for over two decades—but must be optimized for the task. This includes miniaturization, design for hydrogen gas, and large dynamic range electronics.

The fission TPC is specifically engineered for delivering fission cross section measurements with uncertainties below 1.0%. The result of these new measurements will be a refined understanding of computational results, thus reducing the liability from nuclear data in the overall uncertainties of calculated integral quantities. Measuring fission cross sections relative to H(n,n)H elastic scattering removes the uncertainties associated with using the $U^{235}$ fission cross section for normalization; a primary goal is to provide the world's best differential measurement of the $U^{235}$ fission cross section. This will impact nearly all fission library data, since the $U^{235}$ fission cross section has been used as a standard in much of the available experimental fission data.

The fission TPC design contains nearly 6000 charge collection pads for recording the events inside the detection volume. The charge signals are individually amplified and converted to a digital waveform representation for later event reconstruction and data analysis. This amplification and digitization is performed by two paired cards, shown in fig. 1.1. The right board serves to amplify the charge signals, while the left digitizes and transmits the signals to the DAQ[2] system. These cards are known respectively as a preamp and EtherDAQ card, and together are

[1] time projection chamber
[2] digital acquisition

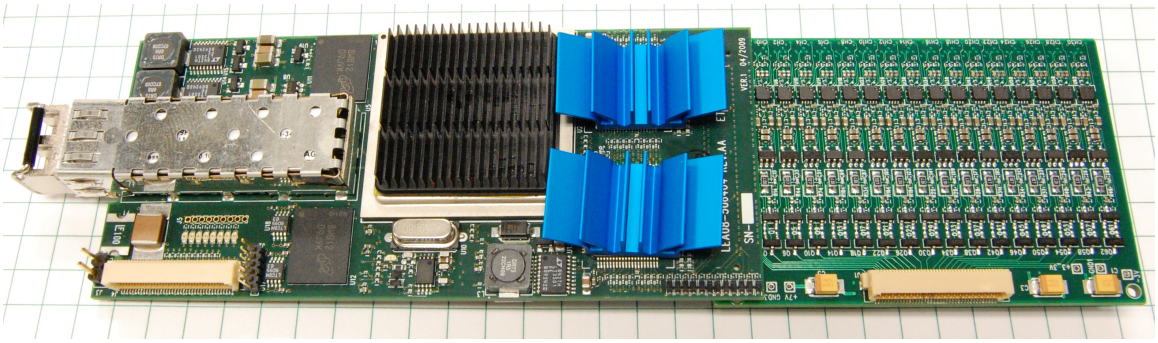


**Figure 1.1** – EtherDAQ/Preamp Card Pair

referred to as a E/P pair. A fully-instrumented TPC contains 192 E/P card pairs. Each E/P pair is instrumented with 32 channels, with one channel providing the readout for a single charge collection pad. Every channel needs to be individually and repeatedly calibrated in order to achieve the measurement goals.

This work describes the approach at developing a framework for performing the required calibration measurements. Existing equipment and software were modified to create a platform for sending charge signals to an E/P card pair, performing an initial health evaluation, and recording the results. A new framework was developed for performing advanced analyses on the calibration data. The framework was demonstrated to be capable of the detailed analyses required for generating the calibration constants needed for the TPC data reconstruction efforts.

## 1.2 The Fission TPC Project

The effort for ultra-precise nuclear data measurements is coordinated under the NIFFTE[3] collaboration.[4] The immediate objective of this effort is to implement a fission cross section measurement program with the goal of providing the most

[3]Neutron Induced Fission Fragment Tracking Experiment
[4]NIFFTE Collaboration homepage: http://nuclear.calpoly.edu/~niffte/

needed measurements with unprecedented precision and accuracy using a TPC. The current focus is on generating precision $Pu^{239}/U^{235}$ and $U^{238}/U^{238}$ fission ratio measurements.

The $U^{235}$/H(n,n)H measurement will provide the best single measurement of the $U^{235}$ fission cross section and will allow the NIFFTE collaboration to convert the initial, and any subsequent, ratio experiments to the world's best absolute measurements. After completion of the $U^{235}$ and $Pu^{239}$ ratio measurements, the collaboration will move on to other measurements, such as minor actinide cross sections and fission fragment distribution. This information will play a crucial role in current and future nuclear research and development campaigns.

## 1.3 The TPC Signal Processing System [2]

The NIFFTE TPC, shown in fig. 1.2, operations under the same principles as an ionization chamber. It consists of a target placed between two pressure chambers, each known as a volume, resulting in $4\pi$ detection capabilities.

Each volume is instrumented at the far end by a pad plane that contains 2976 hexagonal charge collection pads. Each charge collection pad is instrumented by an E/P channel. Figure 1.3 shows the pad plane layout; each plane is split into sextants, each sextant into sixteen sectors, and each sector into 32 pads.[5]

Each channel is instrumented with a preamplifier, an ADC,[6] and a digital readout. The challenge of accommodating a large number of densely packed high-speed channels has been met in the past with relatively expensive custom ASIC[7] chips.

[5]The four triangular sectors on the outside edge of each sextant are instrumented with only 28 pads, which is why the total number of pads for each side is not 3072

[6]analog to digital converter

[7]application-specific integrated circuit

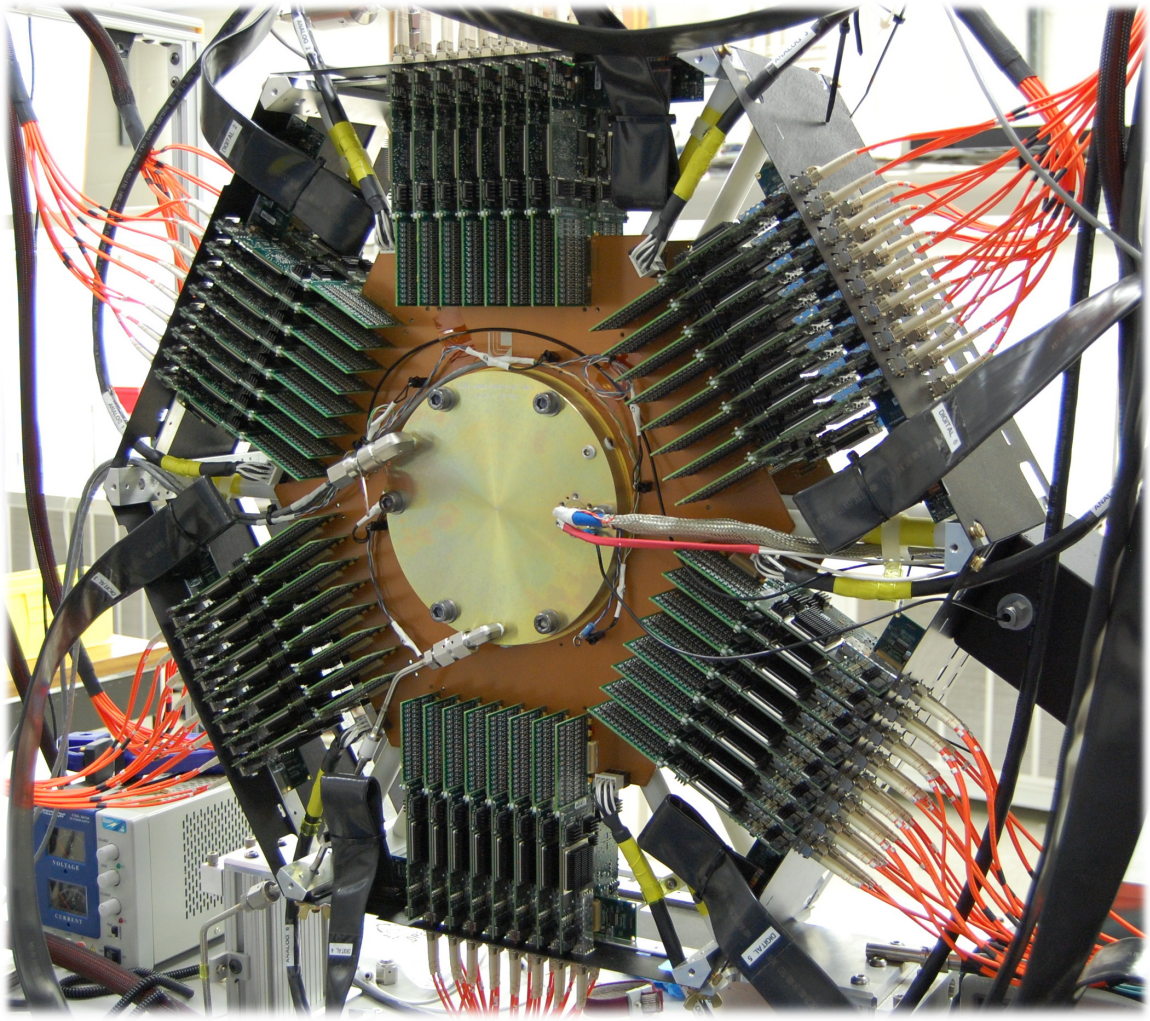


**Figure 1.2** – The NIFFTE TPC with One Instrumented Volume

The technology for ADC and FPGA[8] type devices has improved considerably over the last decade. It is now possible to use off-the-shelf components to accomplish the same task for considerably less development cost, less time to working prototypes, and considerably more flexibility in the final design.

This, combined with PCB[9] manufacturing advancements, has made small-budget custom electronics a viable option. The solution for the NIFFTE TPC was to

[8] field-programmable gate array
[9] printed circuit board

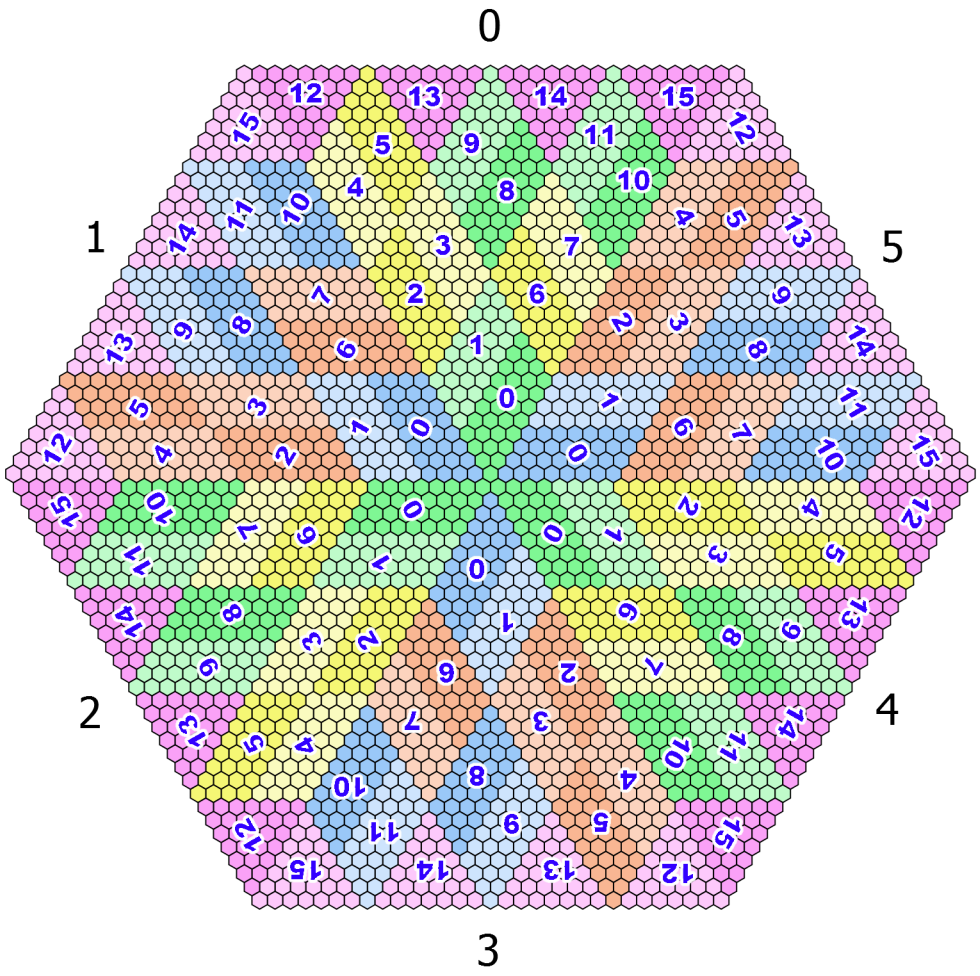



**Figure 1.3** – Pad Plane Layout

create two PCBs intended to be paired together. These are called: 1) the EtherDAQ card, and 2) the preamp card. The actual design and performance of each are detailed elsewhere [3]. The E/P pairs can be identified in fig. 1.2 as the green PCBs arranged around the outer edge of the TPC. The orange cables contain the fiber optics for data transmission.

The card pairings are intended to be permanent, which insures that valid calibration measurements are maintained even between different TPC configurations. Each TPC contains 192 E/P pairs. The primary goal for low data uncertainties requires a very high measurement fidelity. Each card pairing must be individually

tested to ensure it meets the necessary quality requirements. Acceptance criteria will require information regarding the number of dead or noisy channels, the level of crosstalk and the gain linearity

Additionally, minor variations naturally exist in the electronic components between each channel. Thus, each channel must be individually calibrated——a nontrivial task for such a large number of channels. Routine inspection for time-dependent performance is also required. Furthermore, the card pairs are subject to failure due to experimental conditions and are occasionally replaced with fresh counterparts. The individual testing and calibration of each card pair is ill-suited for a manual process due to large time requirements. A manual process for such a large number of cards is also susceptible to human-based error. An automated system is, therefore, the most desirable solution for testing and calibration [4] and is the focus of the work described herein.

## 1.4 Scope of Work

The capability for testing and calibrating the E/P card pairs is a fundamental component of the goal for the success of making precision nuclear data measurements using the NIFFTE TPC. This capability requires testing hardware and a companion software interface for rapid testing of the E/P card pairs, performing initial health checks, and recording data. A robust analysis platform is required to generate the calibration measurements and perform more detailed studies. This work will embody the creation of a comprehensive test stand solution for testing the E/P card pairs. This will include modifications to a hardware testing solution, a software based system for systematic interrogation of the cards, as well as a software analysis package; these are described next.

An existing test stand setup will be adapted to provide the necessary hardware for performing calibration measurements. A crucial element will be the ability to test all 32 channels in an E/P pair simultaneously. The input signals must also be completely reproducible over a large time frame, i.e. the pulse generation equipment must have extremely tight performance and repeatability tolerances.

The current interface software will be modified to be more user friendly; in its current state it can communicate with and configure EtherDAQ cards, but lacks an automated and easy-to-use interface. It will also be modified to configure and operate the pulse generation equipment used to provide input signals to the test stand; the parameters controlling the test must be configurable. It must be developed to include basic functions for preliminary analysis, i.e. inspecting the results of a test for a basic health evaluation. Further, provisions must be made for a user to change the test parameters. Lastly, it will be expanded to save the generated waveform data to an external file for subsequent processing on a more robust analysis platform.

An analysis framework will be developed to carry out detailed user-specified analyses that will result in calibration constants for use in the TPC experiment. It must be capable of fitting the waveforms and extracting the characterizing parameters. The parameter of interest is the gain, which is the primary source of information for channel health and calibration. However, the other fit parameters must also be available if extended analysis needs arise. Further, the framework must be designed to be flexible and adaptable to any future needs.

# Chapter 2

# Calibration Hardware

A test stand is required to provide the necessary hardware for performing calibration measurements. A crucial element is the ability to test all 32 channels in an E/P pair simultaneously. The input signals must also be completely reproducible over a large time frame, i.e. the pulse generation equipment must have extremely tight performance and repeatability tolerances. Finally, the calibration hardware needs to be integrated with a single unified user interface to control the various components. The interface software development is detailed in chapter 3.

## 2.1 Basic Setup

The test stand (see fig. 2.1) was originally designed at LLNL[1] for characterizing and debugging the EtherDAQ and preamp cards. It is constructed of steel paneling so that it functions like a Faraday cage and minimizes EMI.[2] BNC[3] connectors serve to carry signals to the internal components.

[1] Lawrence Livermore National Laboratory
[2] electromagnetic interference
[3] Bayonet Neill-Concelman

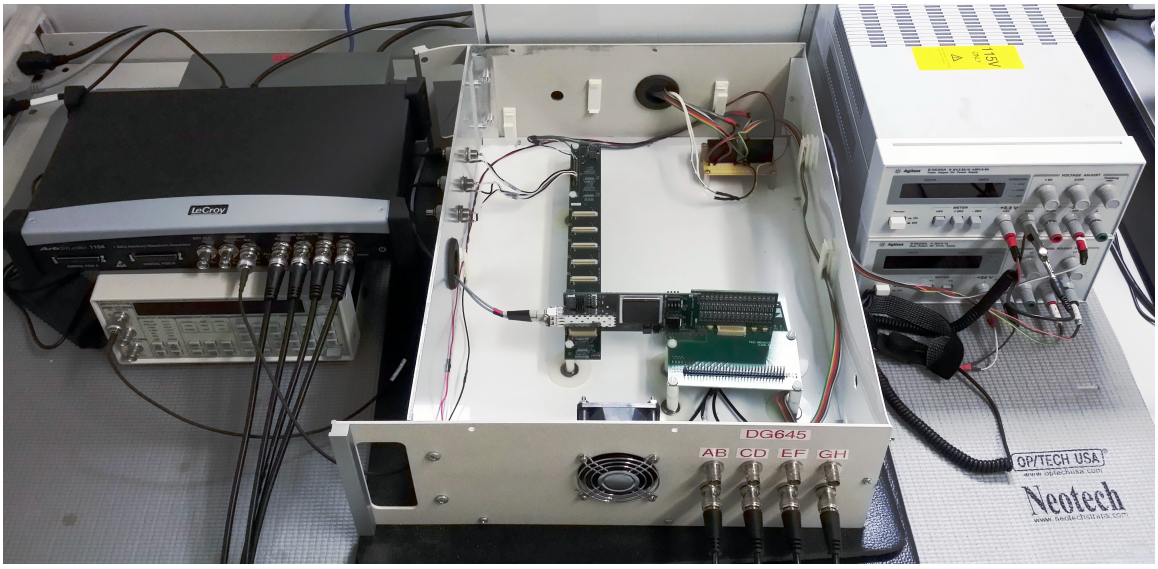



**Figure 2.1** – Test Stand Setup at ISU

The back of the case contains a pass-through for power cables, and the left side contains a second pass-through for the fiber-optic data cable. Clips for holding wires are mounted internally as a preventative measure against accidental tugging or other damage. A cooling fan is also mounted to the front.

### 2.1.1 Power and Clock

Three BNC connectors on the left-side panel provide ports for an external clock signal, an external trigger, and a timestamp reset. An SRS[4] CG635 is used to generate the clock signal required by the EtherDAQ card.[5] Power is supplied via a set of coupled Agilent E3620A and E3630A DC power supplies. Two power supplies are required because of the unique power requirements—the EtherDAQ card requires +24 V, while the preamp requires −3.0 V, +3.3 V, and +7.0 V. The

[4]Stanford Research Systems

[5]No internal clock is built into the EtherDAQ cards. For the TPC application the EtherDAQ cards must be synchronized, which is accomplished by driving all 192 cards with a single clock source [3]—the TPC reconstruction methods disintegrates if the EtherDAQ clocks are not synchronized. This is not required for the test stand's purposes, but the external clock signal must still be provided.

+24 V and clock signal leads are soldered to the DBB,[6] and the preamp power leads are connected to the designated power header pins on the breakout board. An AC adapter is used to power a fan for cooling.

Internally, the test stand is designed replicate the seating of an E/P pair onto the TPC. The setup uses the same DBB for the EtherDAQ cards as is used in the actual TPC. The only difference is that just the first EtherDAQ seat is configured and active since the test stand is designed to evaluate E/P pairs one at a time.

### 2.1.2 Protective Measures

Insulation mats were placed under the test stand. The intent was to prevent acoustic interference as well as static-based EMI.

The timestamp reset was disabled by opening the connection with a 50 Ω resistive load terminator. It is not currently needed due to the 40 day maximum value of the timestamp value. An accidental timestamp reset in the middle of a card testing procedure could even cause an error in the adapted DAQ system; the timestamp reset is reserved primarily for use between long data runs during actual TPC operation.

A interlock relay system (see fig. 2.2) was installed that closes the power supply circuits only when the cooling fan is on. This precautionary measure was implemented to prevent accidental overheating of the EtherDAQ components.

[6] digital bus board

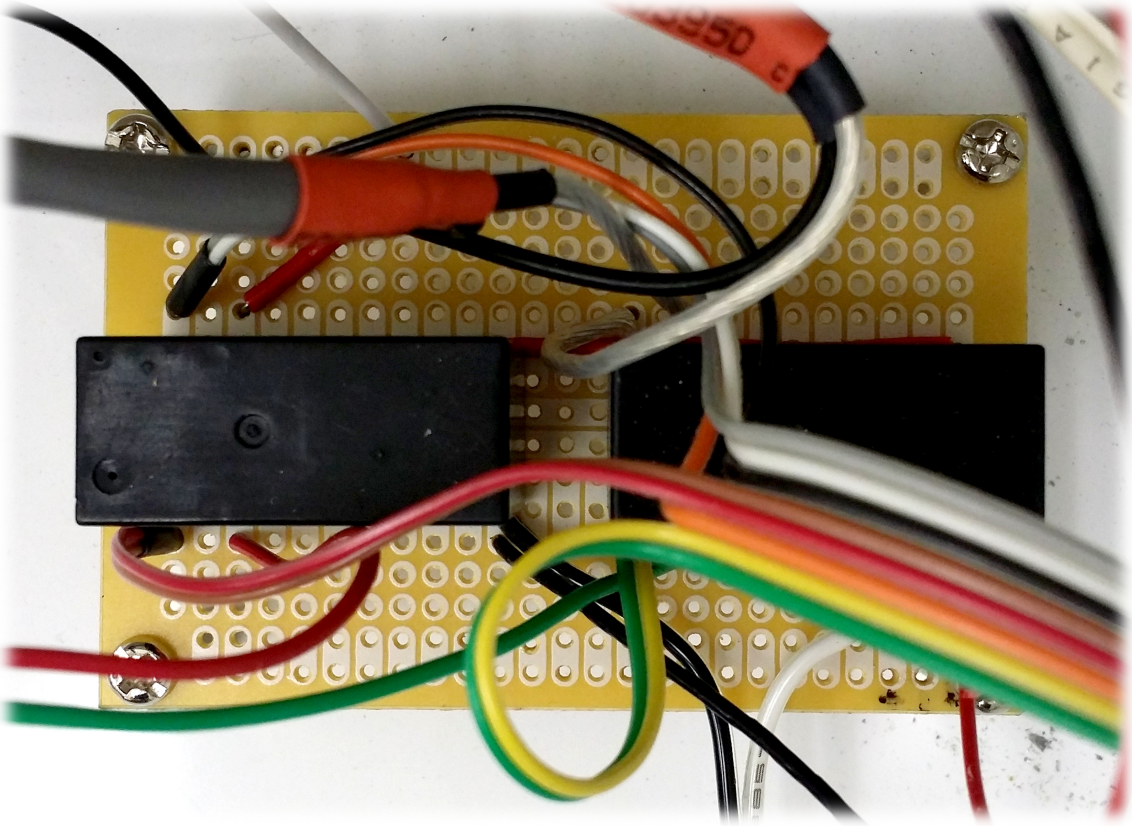
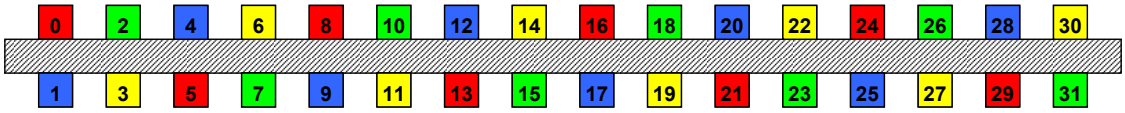


**Figure 2.2** – Interlock Relay Modification

## 2.2 Pulse Distribution System

Function generators are used to the provide voltage pulses that drive the test stand system. The front of the test stand provides 8 BNC ports for supplying the input voltage pulses internally.

Capacitors are used to convert each pulse's voltage into a charge signal, emulating of the charge collection behavior of the TPC detection pads. Mezzanine boards, referred to as breakout boards, are used to deliver the capacitor-generated charge signals to the preamp card through the exact same connector as used on the TPC. The breakout boards are also used to supply power to the preamp card, which is




| Bank Output | Preamp Channel | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| AB | 0 | 5 | 8 | 13 | 16 | 21 | 24 | 29 |
| CD | 1 | 4 | 9 | 12 | 17 | 20 | 25 | 28 |
| EF | 2 | 7 | 10 | 15 | 18 | 23 | 26 | 31 |
| GH | 3 | 6 | 11 | 14 | 19 | 22 | 27 | 30 |

**Figure 2.3** – Bank to Channel Correlation

provided through a set of jumper connectors on the bottom. The locations of the channel and power connections are printed on the breakout boards[7] for clarity.

The channels are configured so that sequentially numbered channels do not neighbor each other on the preamp card. The top-half of fig. 2.3 illustrates this configuration.

### 2.2.1 Pulse Generation

An SRS DG645 generator was used initially at LANL[8] for preliminary testing purposes. However, extended testing with the DG645 yielded inconsistent results. After consulting the spec sheets and contacting customer support, it became apparent that the DG645 performance was a poor match for the testing and calibration requirements—it was designed with a focus on precise timing—whereas the listed

[7]The hardware design, source code, and older GUI interfaces generally employ a 0-based indexing scheme, meaning that the channels are numbered 00 through 31. The GUI interfaces designed during this project (see chapters 3 and 4) use a 1-based indexing scheme so that the channel numbering starts with 1 and goes to 32. A 1-based indexing is employed in these situations because a typical user is not familiar with 0-based indexing.

[8]Los Alamos National Laboratory

amplitude uncertainty was $100\,\text{mV} \pm 5\%$. Evaluation of the DG645 pulses were performed under a variety of environmental conditions, loads, usage duration, and pulse parameters. The observed variations confirmed the documentation specifications.

Consistent pulse amplitude behavior was absolutely necessary for testing and calibrating the preamp cards. The ArbStudio line, manufactured by Teledyne LeCroy, was identified as the best option. The ArbStudio units are arbitrary waveform generators, enabling a wider range of testing conditions than the DG645's square pulse.

A demo ArbStudio 1104 unit was evaluated over the period of a month and compared against the DG645 using the same pulse evaluation tests. Compatibility with the interface software was also verified. Niffte project management determined that the ArbStudio 1104 fulfilled project requirements, actually meeting or improving on the information listed in the spec sheets. The demo unit was shipped back and a new unit purchased for use with the test stand.

### 2.2.2 Individual Channel Testing

A special breakout board, shown in fig. 2.4, provided the capability to isolate a single channel and test it independently of the 31 other preamp channels.

Individual testing is accomplished using individual leads, each with an embedded capacitor, soldered to the bnc connector. The embedded capacitors are used to create charge signals from the voltage pulses, which action emulates to the charge collection behavior of a pad on the tpc. The current configurations are summarized in table 2.1. The capacitance affects the formation characteristics of

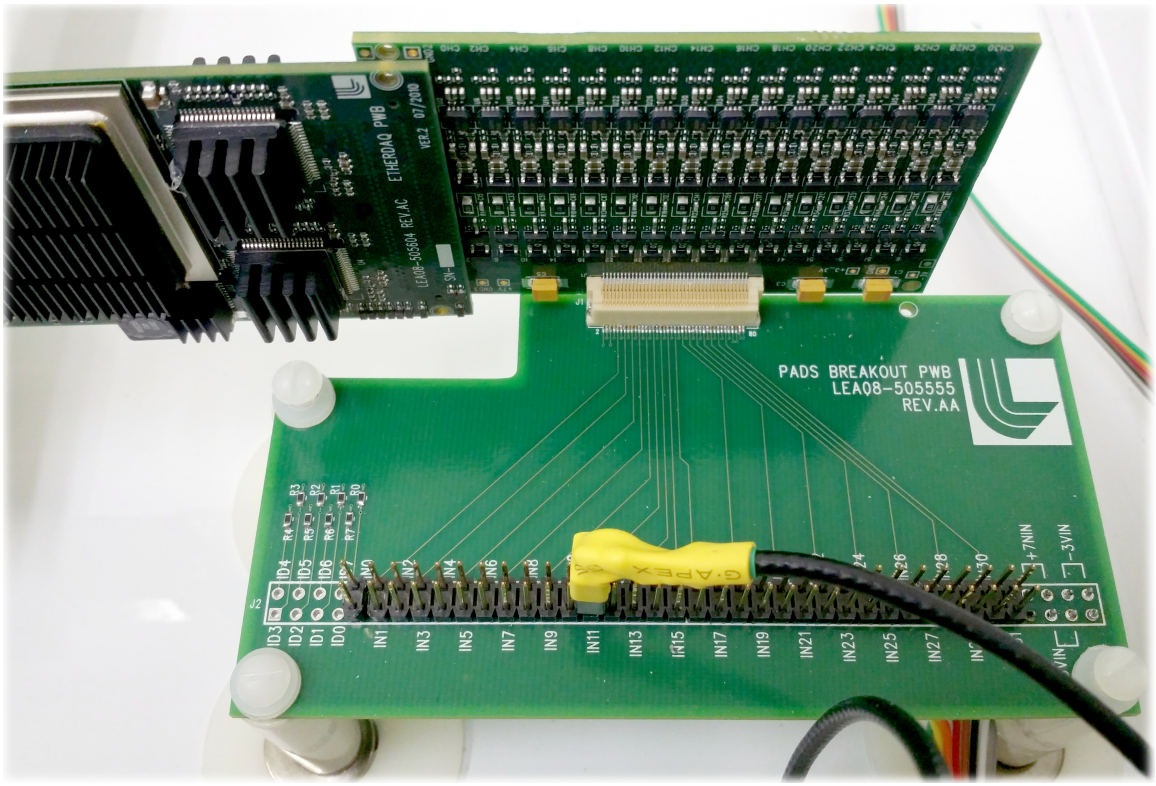


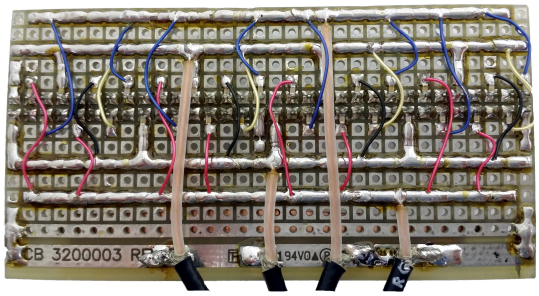

**Figure 2.4** – Individual Channel Testing Setup

**Table 2.1** – Channel Isolation Capacitance Configuration

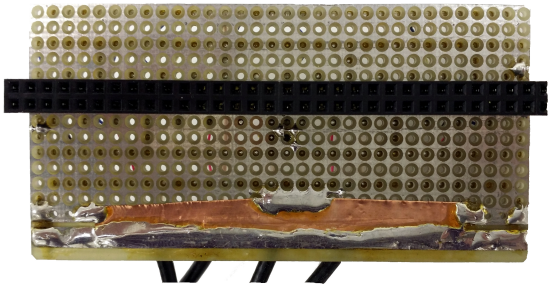
| **Test Stand Connector** | AB | CD | EF | GH |
|---|---|---|---|---|
| **Capacitance** | 0.5 pF | 1.0 pF | *unused* | *unused* |

the charge signal generated from the voltage pulse. Further, the ideal capacitance will generate charge signals within the dynamic range of an E/P pair.

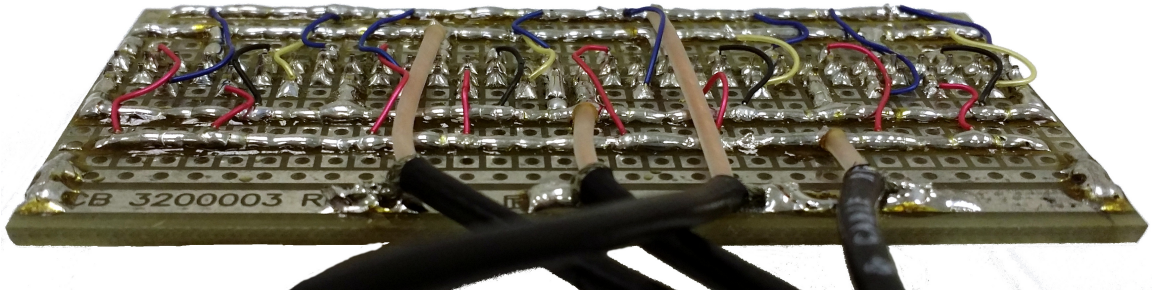
The jumper connector on the end of each lead was given a silver orientation mark on one side. The mark indicates which side must face the channel number printed on the breakout board. Proper orientation is required for the proper charge polarity; an inverted charge signal will not be detected and converted into a waveform by the ADC system.




**(a)** Conversion from Banks to Channels



**(b)** Breakout Board Connector




**(c)** Wiring Profile

**Figure 2.5** – Distribution Board

### 2.2.3 Distribution Board

The distribution board (see fig. 2.5), created at LANL, splits the signals from the four banks into 32 separate pulses. The banks are, ordered from the top/back to the bottom/front: AB, CD, EF, and GH. In figs. 2.5a and 2.5c the channels are numbered 1 to 32 from right to left. The channel ordering is reversed for fig. 2.5b since it shows the flip-side.

Small capacitors, $\approx$0.5 pF in size, are soldered between the pulse splitters and the jumper connection to the preamp. These capacitors can be seen between the colored wires and jumper solder points in fig. 2.5a, with local detail on the channel 07 capacitor in fig. 2.6.

A breakout board (detailed in fig. 2.7), conducts the 32 pulses created by the distribution board to the 32 channels on the preamp. The channels are organized

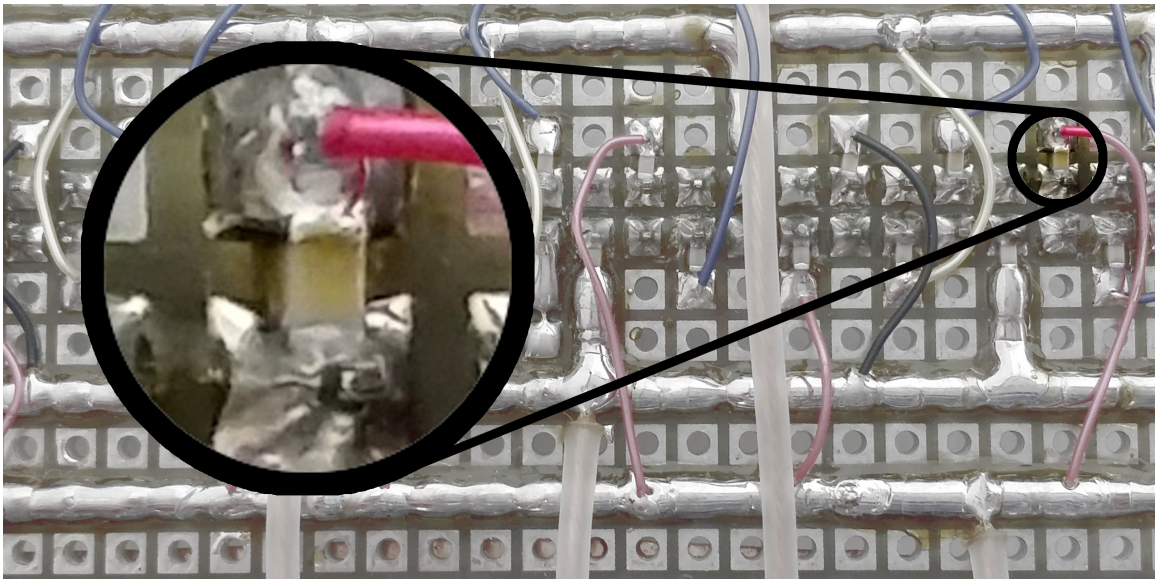
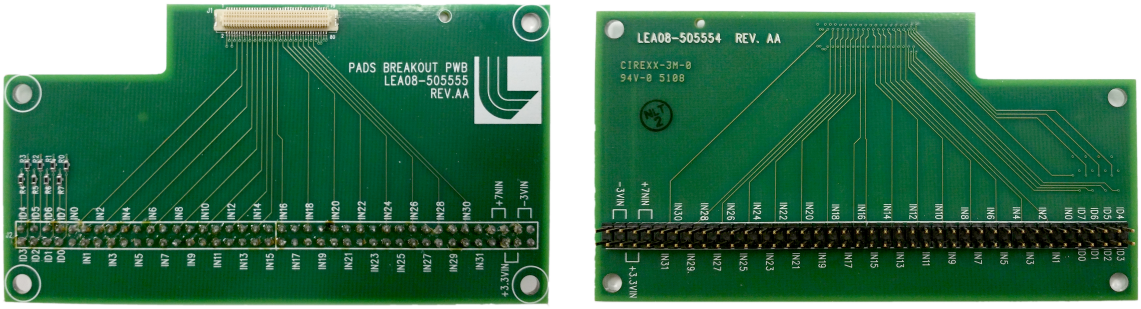


**Figure 2.6** – Detail on the Channel 07 Capacitor




**(a)** Top **(b)** Bottom

**Figure 2.7** – Breakout Board

such that neighboring and opposite-side channels are all fed by different banks. In other words, a channel's source bank is different from its neighboring and opposite-side channel's source banks, which are all themselves different from each other. Figure 2.3 illustrates this separation by using a color-coding scheme to represent the bank to channel correlations.

# Chapter 3

# Interface Software

The name of the interface software for the test stand is EtherDaqGUI, and was developed at LLNL by Dr. Vincet Riot. EtherDaqGUI runs in Windows[1] XP and is coded in C++ using the Visual Studio[2] IDE.[3] Its primary purpose was for the continual testing and debugging of the EtherDAQ and preamp cards throughout the hardware design phase (see chapter 2).

EtherDaqGUI must be modified in order to perform to perform quick, basic health evaluations on the E/P card pairs, provide complete control over the test stand hardware, and save all the hardware configuration and waveform data to a file for subsequent analysis. Specifically, the interface must be modified to be more user friendly; in its original state it could communicate with and configure EtherDAQ cards, but lacked an automated and easy-to-use interface. It must also be modified to configure and operate the pulse generation equipment used to provide input signals to the test stand. The user must be able to modify all the test parameters. EtherDaqGUI must then save the configuration data along with

[1]Windows® operating system is a registered trademark of Microsoft Corporation
[2]Visual Studio® is a registered trademark of Microsoft Corporation
[3]integrated development environment

the generated waveform data to an external file for subsequent processing on a more robust analysis platform. Finally, a full test suite should last only a few minutes, depending on the test parameters. The resulting data should be studied in much greater detail using a newly developed analysis platform, later discussed in chapter 4.

## 3.1 Conversion from a Debugging Tool

It was decided to adapt the existing EtherDaqGUI code by inserting the additional abilities required for the card testing process. The alternative, redeveloping these capabilities from scratch,[4] was not considered to be a time-effective solution. EtherDaqGUI already contained the following basic features required for the card testing procedure:

- network interface
  - scanning
  - card identification
- initialization
- status interrogation
- data collection

The application GUI[5] is organized by dividing the various functions into tabs. A new tab, <Card Test>, was created for the purpose of automated card testing.

[4]The resulting interface from re-development would have been purpose-built, potentially increasing the run-time efficiency. This does not mean that the existing interface had poor performance, but that there was potential for improvement by stripping out unneeded functionality. However, the chance for performance gains was not sufficient justification when compared to the estimated development time.

[5]graphical user interface

Figure 3.1 demonstrates the final repurposed version of EtherDaqGUI with the <Card Test> tab in action.

## 3.2 Automated EtherDAQ Configuration

The EtherDAQ cards first perform a POST[6] when supplied with power. They do not, however, start sending data upon POST completion; the data recording device must first configure the EtherDAQ card(s).

The original framework required the user to perform a multi-step process in order to identify and configure an EtherDAQ card. First, the network must be scanned for MAC[7] addresses that are pre-configured to be associated with the EtherDAQ cards. Next, the operational parameters must be set. These are, in no order of importance: 1) destination MAC address, 2) lookback duration, 3) high and low trigger thresholds, 4) neighbor triggers, and 5) sampling frequency. Finally, the desired channels must be enabled, as they are all disabled by default. This is done by calling CMD_SetSetAnalogChannelEnable().[8][9] Enabling and disabling of individual channels is a necessary part of operating the EtherDAQ cards for the TPC application. This way a channel's output can be suppressed if its electronics fail and begin to introduce erroneous waveforms into the data stream.

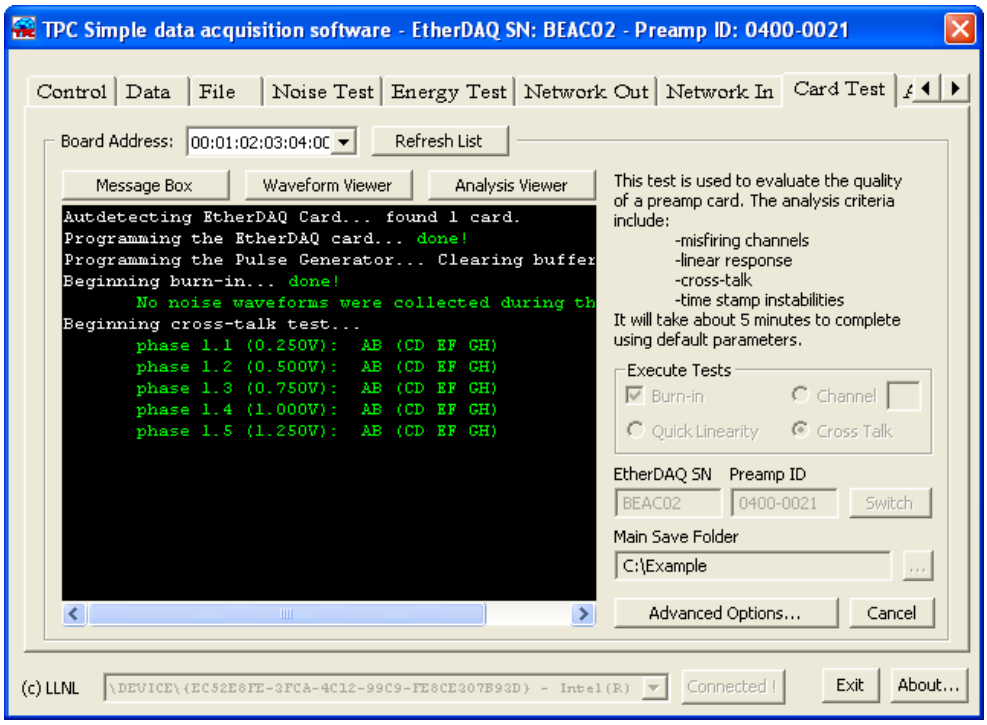
An automated configuration process was designed that simplifies this process. The user is required only to click on the <Refresh List> button whenever a change is made to the system. The network is then scanned for EtherDAQ cards and

[6] power-on self test

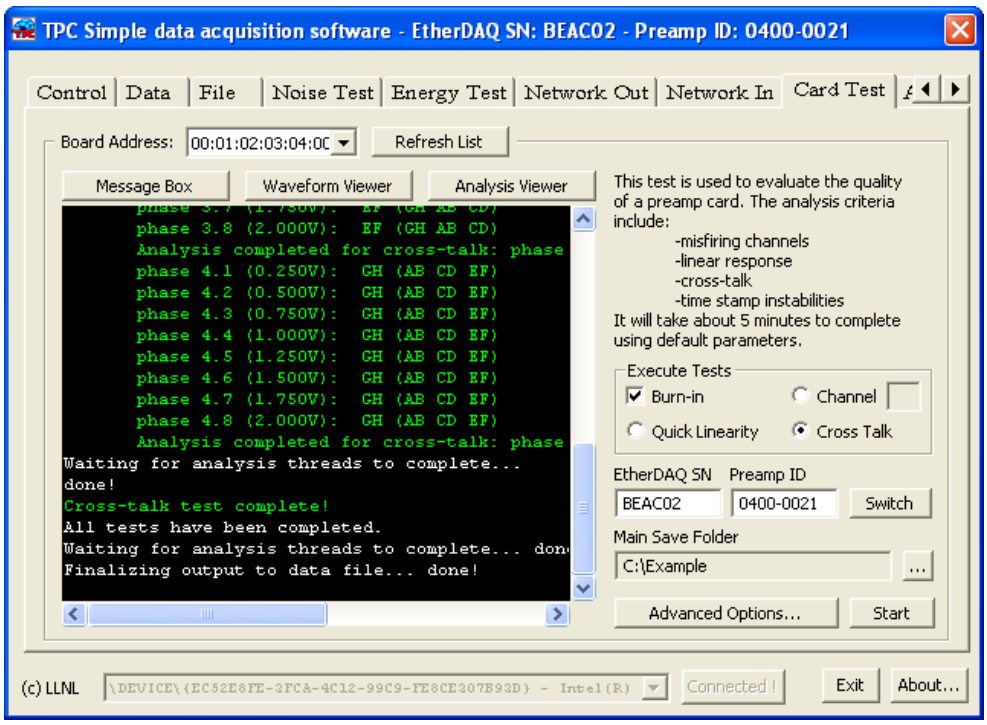
[7] media access control

[8] TPCPacketParser.cpp:1847–1866

[9] This sends a packet containing a single 32-bit mask value. The individual bit positions correlate directly to the channel numbering; the 1st bit maps to channel 1 and the 32nd bit maps to channel 32. A bit value of '1' enables the channel, whereas '0' disables the channel. For example, a '1' for the 1st bit causes channel 1 to be enabled; conversely, a '0' in the same position would disable channel 1.




**(a)** Test Running




**(b)** Test Completed

**Figure 3.1** – EtherDaqGUI Card Test Tab In Action

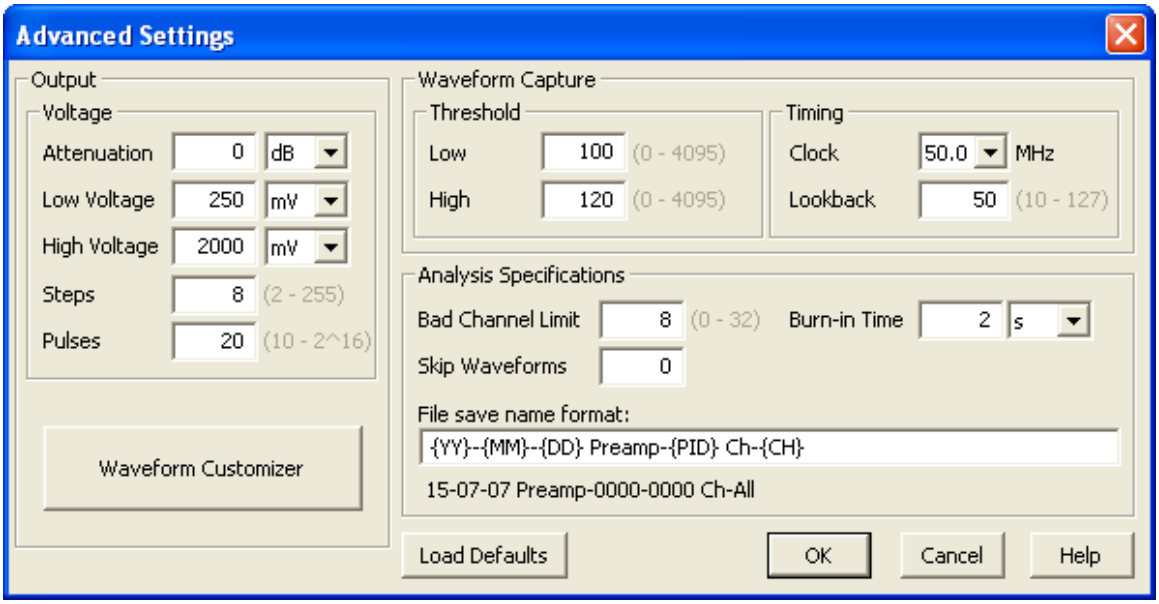



**Figure 3.2** – EtherDaqGUI Advanced Settings Dialog

the configuration steps are performed for each discovered EtherDAQ card. In some cases the changes are not immediately detected. In this instance it may be necessary to click on the <Refresh List> button until the changes are recognized. If nothing is detected after a few refreshes then there is likely an error in the hardware setup; some items to check are: 1) the EtherDAQ card is powered on, 2) the fiber-optic adapter is properly connected, and 3) the optic fibers are seated properly and have no sudden corners.[10]

Default operating parameters are defined in CTSetDefaults().[11] These default parameters are selected from experience to fit most testing needs. The parameters are user-configurable via the <Advanced Settings> dialog, as shown in fig. 3.2. All changes to the operating parameters can be reverted to the default values by clicking on the <Load Defaults> button within the <Advanced Settings> dialog.

[10] If none of these conditions exists the problem may lie in the network communication interface of EtherDaqGUI. The only recourse is to close EtherDaqGUI down, use the Task Manager to ensure the process is closed, then re-open EtherDaqGUI.

[11] CardTest.cpp:1528-1605

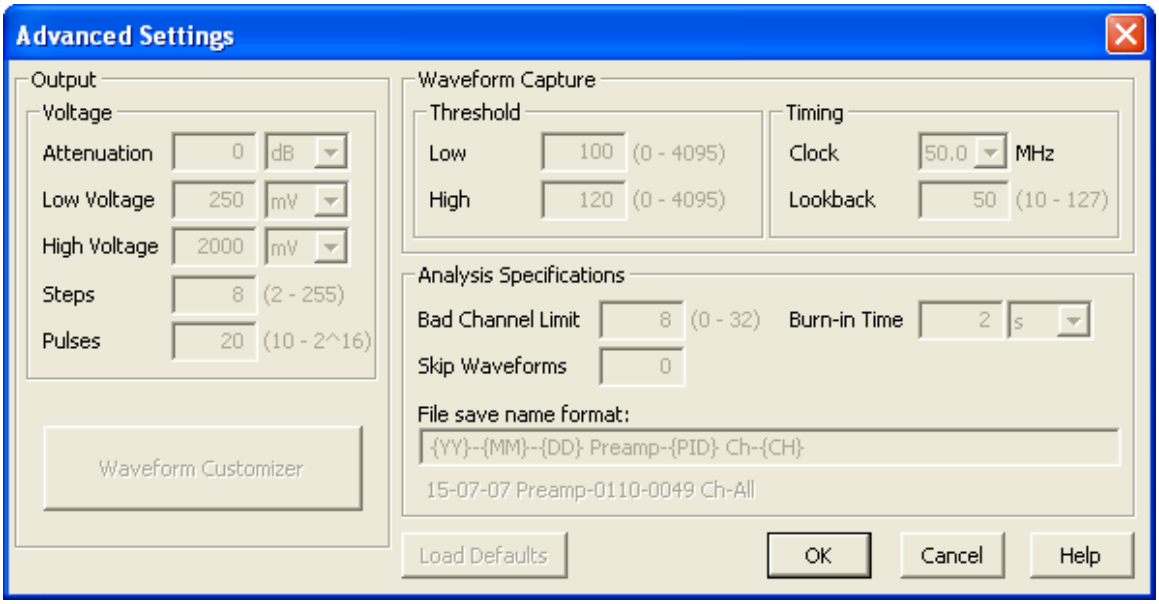



**Figure 3.3** – EtherDaqGUI Advanced Settings Dialog in Disabled State

Help is available by clicking on the <Help> button. This opens up a window with descriptions about each of the items and how they influence the testing procedures. The syntax and substitutions of the <File save name format> box are also fully listed and described.

Modifying the parameters is possible only when EtherDaqGUI is idle. The user can always open the advanced setting dialog to view the settings, but the parameters are grayed out and cannot be modified if the pulse generation equipment is being operated. Figure 3.3 shows the appearance of the <Advanced Settings> dialog with the parameters locked.

## 3.3 Driver Compatibility Issues

EtherDaqGUI depends on ndisprot, a raw Ethernet communications service library, to communicate with the EtherDAQ cards. This allows the communication based on

MAC addresses instead of IP[12] address, enabling more efficient packet transmission and routing.

It was discovered during development that `ndisprot` was not compatible with Windows 7. Although the computer driving the test stand used Windows XP and suffered no compatibility issues, the development computer ran Windows 7. Different potential solutions were investigated to resolve the Windows 7 compatibility, each without success. It was determined that the lack of compatibility was caused by changes to underlying architectural design differences and driver registration restrictions in Windows 7.

The primary development continued on the Windows 7 system. Final testing and debugging was performed on the test stand system. SVN[13] was used to keep the code between the development and test stand computers synchronized. Two tweaks enabled the necessary debugging on the development system.

First, it was necessary to inspect the GUI as the `<Card Test>` tab and associated items were designed and laid out. Originally, EtherDaqGUI exited automatically if `ndisprot` was not detected. This behavior was replaced by a message box warning of the situation. Once the warning was acknowledged the main EtherDaqGUI window opened up.

Second, the validity of the waveform analysis routines needed to be checked. The `PseudoPulseGenerator` class was created which emulated the acquisition of waveforms from the EtherDAQ card. These simulated results were feed directly into the `Card Test` class's DAQ component. This class was configured to be used automatically if no pulse generator was detected upon starting a card test. Again, the user was notified via a warning message.

[12] internet protocol
[13] Apache Subversion

## 3.4 Pulse Generator Operation

A detailed understanding of the inputs to the system is a critical component of performing calibration measurements. This is best achieved if the test system itself specifies the input signal characteristics. Thus, controls for operating the pulse generating equipment were added to EtherDaqGUI. A standard pulse configuration defines the following items:

- voltage amplitude & offset
- frequency
- duty cycle
- number of pulses

Nothing should be assumed about previous operational states—the entire pulse definition sequence should be transmitted to reprogram the generator when a test is begun, with any previous settings or changes overwritten. EtherDaqGUI must completely also maintain control the pulse generating equipment throughout the entire card testing procedure.

### 3.4.1 Remote Programming

The remote programming of the DG645 is achieved via its Ethernet port. Originally, the DG645 was controlled with hand-modified script written in C. An executable program wass produced by compiling the script in a Cygwin environment[14] and linking it against the Windows `WinSock32` libraries.[15] Running this program opened a connection to the DG645, sent all the commands, then closed the connection once all the commands were executed.

[14]Refer to the Cygwin website for more information: `https://www.cygwin.com/`
[15]The `WinSock32` libraries provide the framework for communicating over Ethernet

A template file was been created to simplify this process based on an example script found in the manual [5, pp. 70–72]. Still, compilation or run-time errors often occurred after modifying the template from minor mistakes, e.g. mistyping a single character. Often these were challenging to locate and resolve. The need for frequent modifications due to changing testing requirements compounded the inefficiency of this approach.

Ultimately, the communications protocols were embedded into EtherDaqGUI, which provided the necessary dynamic communication interface. This mirrored much of the work required by the external template file modification, while reducing the potential points of failure. This also enabled near-instantaneous programming of the DG645, limited only by the response time of the generator.

### 3.4.2 Common Control Features

Throughout testing the pulse generator is sent more commands to change the pulse parameters as needed. Trigger events are also sent by EtherDaqGUI to initiate the next burst of pulses. The controls are implemented such that each of the four outputs can be independently defined.

It was determined that compatibility with the DG645 would be retained when support was added for ArbStudio devices (see section 3.4.4). To accomplish this the complementary macros `UseArbstudio`[16] and `UseSrs`[17] were created. The interface for the ArbStudio devices is used if `UseArbstudio` is defined at compile time. Otherwise, the preprocessor defines `UseSrs` and the DG645 interface is used. This provides support for both generators with only a slight decrease in source code legibility.

[16] `CardTest.cpp:8`
[17] `CardTest.cpp:19-21`

### 3.4.3 SRS DG645 Interface

The controls for DG645 were implemented in the `PulseGeneratorEthernetInterface`[18] class. The class handles the initialization procedures, status interrogation, and setting the pulse parameters. EtherDaqGUI constructs an instance of the `PulseGeneratorEthernetInterface` class when launched. Commands are immediately sent to open the connection and give control to the computer. This places the DG645 into remote lock out mode. The LED[19] labeled 'REM' is lit to indicate when the remote lock out mode is active.

For the current setup the DG645 was assigned `192.168.1.6` as its static IP address. This value was manually keyed into the DG645 using the front panel and hard coded into the `PulseGeneratorEthernetInterface` class initializer.[20]

### 3.4.4 ArbStudio 1104 Interface

The ArbStudio line of devices communicate using USB.[21] An SDK[22] for integrating device control was provided by Teledyne LeCroy. The SDK was built and compiled using Visual C♯ from the Microsoft[23] .NET Framework.

#### 3.4.4.1 License Issue Resolution

Forcing the ArbStudio SDK to function properly was a complicated process; the SDK depended on an NI[24] .dll that failed when verifying licensure. The .dll was

[18] `SRSGeneratorInterface.(h/cpp)`
[19] light-emitting diode
[20] `SRSGeneratorInterface.cpp:16`
[21] universal serial bus
[22] software development kit
[23] Microsoft® is a registered trademark of Microsoft Corporation
[24] NI is a trademark of National Instruments. This publication is independent of National Instruments, which is not affiliated with the publisher or the author, and does not authorize, sponsor, endorse or otherwise approve this publication.

critical to device control and utilized to generate Fourier series from arbitrary pulse shape descriptions. The Fourier series coefficients, not the potentially millions of individual discrete data steps, are transmitted to the pulse generator to define the pulse characteristics.

The ArbStudio software engineers were unable to resolve the issue; this issue was completely unknown since the same .dll was successfully used in their own software package. A back-door solution was discovered that required modifying the code of the failing .dll. This was possible only because of the partially-compiled nature of .NET Framework executables and libraries. Using a decompiler, the offending code was commented out and then the .dll regenerated.

#### 3.4.4.2 Implementation

EtherDaqGUI was originally written using unmanaged C code, while the ArbStudio SDK was written using managed Visual C♯ code. Because of the complexities relating to interfacing these two different programming languages and design paradigms, it was necessary to create an separate ArbStudio project within the EtherDaqGUI solution. The new project was written in Visual C♯ and linked against the ArbStudio SDK. The unmanaged to managed code connection was implemented in the EtherDaqGUI project as the `ArbStudioGeneratorInterface`[25] class, and includes the necessary functions for mapping the unmanaged function calls to a managed scope.

In additional to the capabilities of the DG645 interface, the ArbStudio project included the ability to customize the shape of the pulse. Although any arbitrary cyclic function is possible, only a few standard options are currently available. These are square, triangular, sinusoidal, and Erf. The default shape is square,

[25] `ArbStudioGeneratorInterface.(h/cpp)`

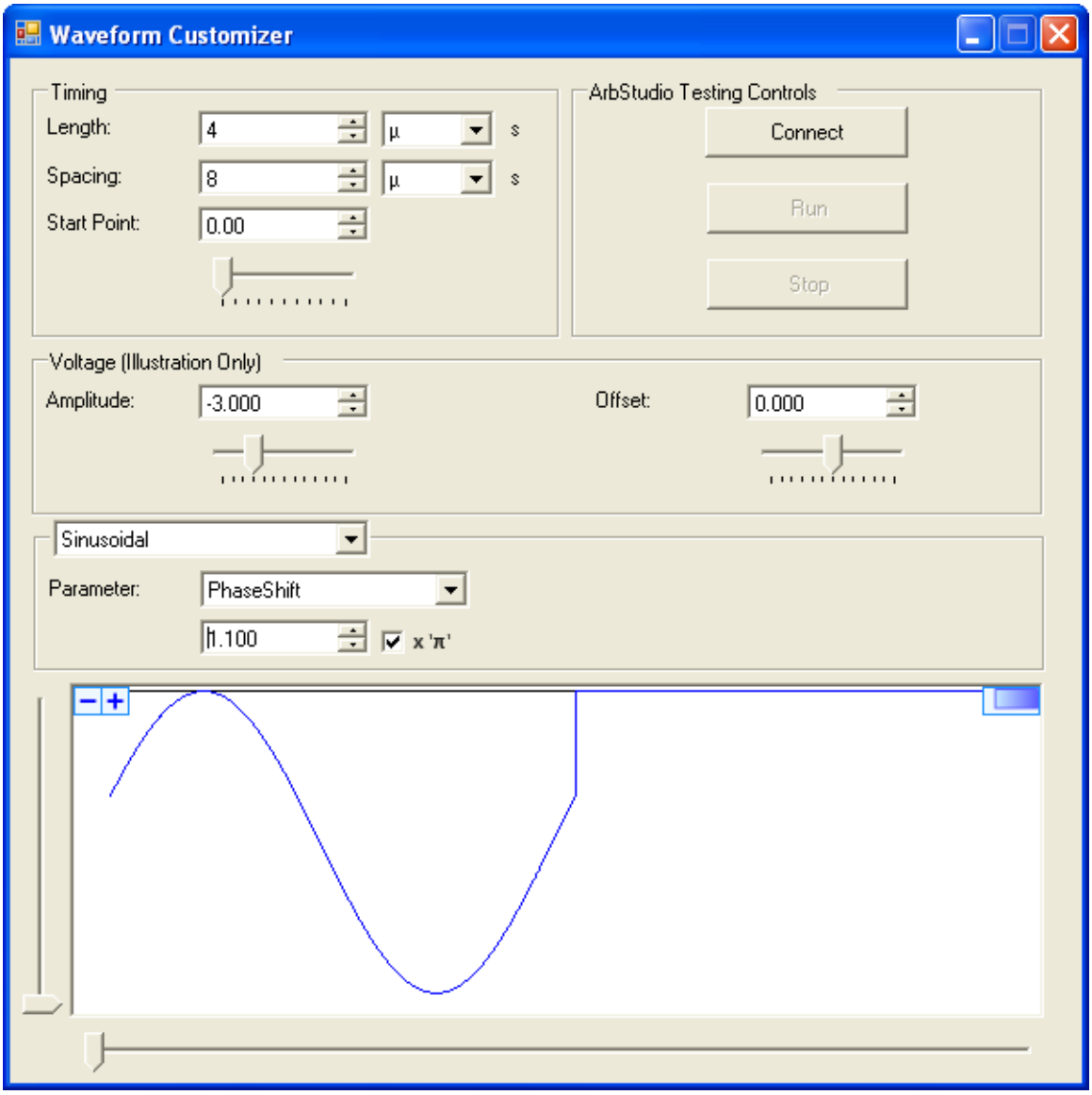
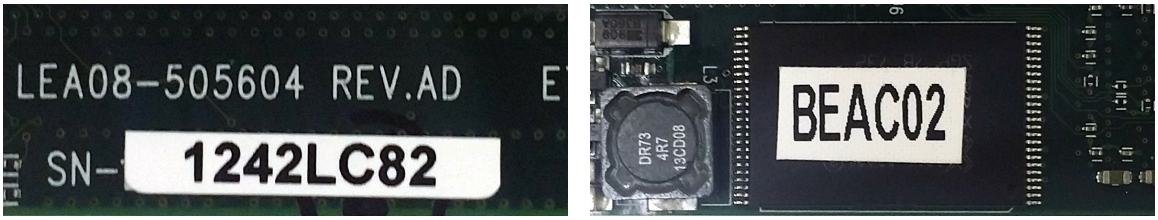



**Figure 3.4** – Waveform Customizer Dialog

which is the most similar to the output of the DG645. The <Waveform Customizer> dialog, shown in fig. 3.4 with parameters selected purely for demonstration, was created to support this functionality. It is accessed through the <Waveform Customizer> button in the <Advanced Settings> dialog.




**(a)** Mezzanine Board **(b)** Chip

**Figure 3.5** – EtherDAQ SN Placement

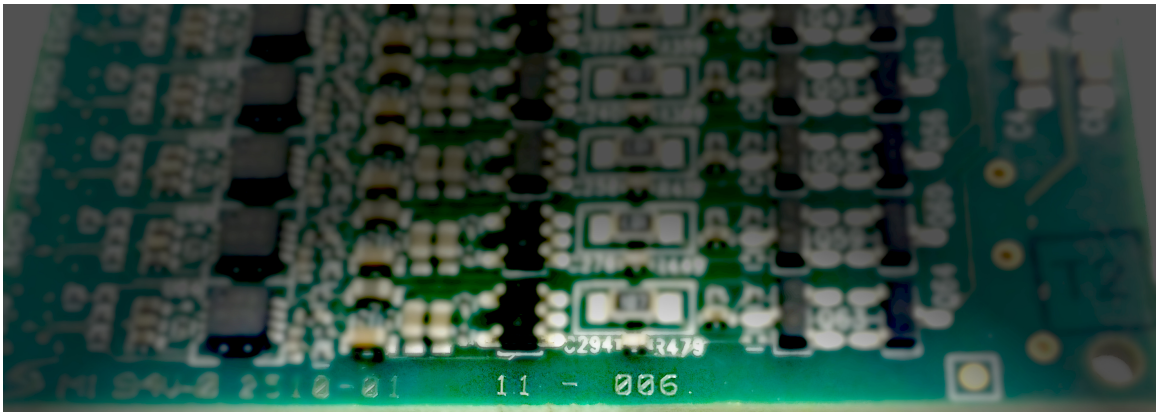



**Figure 3.6** – Preamp ID Location

## 3.5 Card Testing

Card testing can be performed using a few different procedures, depending on the analysis needs. Each test is associated with the date and unique E/P card identifiers. The E/P pair is identified using both the EtherDAQ SN[26] and the preamp ID,[27] which must be entered by the user. The EtherDAQ SN can usually be found on a printed label affixed to the mezzanine board (fig. 3.5a) or a chip (fig. 3.5b). The preamp ID can be found on the backside of the preamp card, as shown in fig. 3.6. It is of the form XXXX-XXXX, where 'X' represents a digit. Occasionally a blank space is used instead of a digit; this is replaced by '0' for consistency. For example, the preamp ID shown in fig. 3.6 should be read as '0110-0006.'

[26] serial number
[27] identification

The <Start> button is enabled when all the parameters are set and an EtherDAQ card has been found. Once the testing is started the <Start> button is replaced by a <Cancel> button. When the <Cancel> button is clicked 1) the current testing operation will terminate, and 2) the equipment will be returned to an idle state.

### 3.5.1 Sanity Test

A sanity test is performed once card testing has been initiated and before any actual testing begins. First, the output path for the data is checked to ensure validity. Second, the EtherDAQ SN and preamp ID are checked against an internal history. If the same pair has already been tested then the user notified and asked if they wish to stop or continue. Third, the connection to the pulse generator is tested and verified. If no pulse generator is found then the user is warned and the testing proceeds using an internal waveform simulation framework. Fourth, the network conditions are checked as described in section 3.5.1.1. Fifth, the (user-defined) test parameters are examined. Any errors are corrected with valid values and the user is notified of the changes. Finally, the internal data buffers are reset and prepared for the new data.

#### 3.5.1.1 Network Conditions

The network is sniffed briefly to verify that the packet transmission rate is zero. If network activity is detected then a soft reset of the EtherDAQ card is attempted.

This check is performed to counter the behavior of the EtherDAQ card under certain situations. Fundamentally, packets are stored in the buffer before being transmitted over the network. When a data packet is transmitted an acknowledgement from EtherDaqGUI is expected in return. If no acknowledgement is received

then the data packet is retransmitted at a regular interval until an acknowledgement is received. This behavior was implemented to prevent any data loss and performs admirably under most conditions.[28]

Occasionally and for reasons still unknown, the EtherDAQ card and EtherDaqGUI fall out of sync with each other: the EtherDAQ card fails to recognize the acknowledgements sent from EtherDaqGUI. As a result the network becomes saturated as each tries unsuccessfully to communicate with the other. This condition is colloquially known as a "10 to the 4th" error, taken from the magnitude of the number of packets being transmitted every second.

#### 3.5.1.2 Soft EtherDAQ Reset

All the channels are disabled, the configurations settings are retransmitted, the internal buffer is flushed with six housekeeping[29] requests, and then the desired channels are re-enabled. Receipt of all six housekeeping packets is checked once the network returns to a silent state. If any packets are found missing then the system is still out of sync and the process is repeated.

The card testing is terminated if too many sequential re-sync attempts fail. At this point the only current recourse is to power-cycle the EtherDAQ card and restart EtherDaqGUI. Restarting the computer is not necessary.

[28] This does not guarantee data preservation—if a packet sits in the EtherDAQ buffer too long it may be overwritten by newer data

[29] "Housekeeping" is another term for using the `GetAnalogChannelStatus()` function. This request causes a large packet, with information for each of the requested channels (supplied by a 32-bit argument bit-mask), to be placed into the buffer. Six of these requests for all 32 channels will fill two buffer blocks, hopefully precipitating a complete buffer flush.

### 3.5.2 Preliminary Analysis

EtherDaqGUI contains preliminary analysis capabilities to check the health of an E/P card pair. The <Advanced Settings> dialog can be used to change the sensitivity of the preliminary analysis to bad channels or waveforms.

Capabilities for both automated and manual preliminary analysis are available. Fundamentally, both depend on the estimated pedestal and plateau values for each waveform. The pedestal is the average of the first ten data points; similarly the plateau is the average of the last ten data points. Finally, the waveform amplitude is the difference between the pedestal and plateau.

#### 3.5.2.1 Automated Preliminary Analysis

The automated preliminary analysis was designed to provide a quick method of evaluating the performance of an E/P based on the pulse amplitude. Two different health metrics are used: channel response and linearity.

**Channel Response**

The channel response analysis requires a simple check of the amplitude. A channel passes if the response is above the given threshold. If the amplitude is below the threshold the channel is flagged for closer investigation. A channel that fails the response test is likely dead and will not be able to collect data at any pulse voltage.

**Linearity**

Linearity is an analysis performed using the combined results of the waveform amplitude at each voltage set during a phase. Simple linear regression is used to analyze the amplitude as a function of the voltage. The 'R' value is used to

determine the statistical quality of the fit. A bad statistical fit is indicative of a channel that is not linear over the range of input voltages. Manual analysis should be done to determine the cause and decide whether the channel is usable.

#### 3.5.2.2 Manual Preliminary Analysis

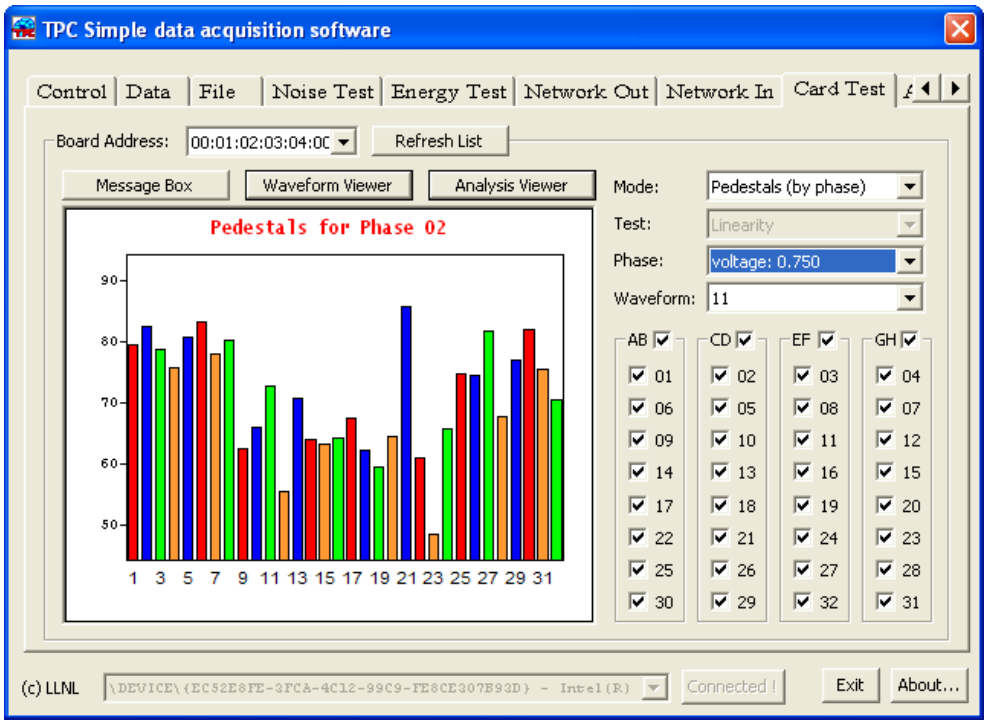
The manual analysis tool collection is designed to aid in further investigating the health of a card. The tools are activated by clicking on the <Analysis Viewer> button under the <Card Test> tab. The resulting perspective is shown in fig. 3.7. On the right are the controls for selecting the analysis tools and data ranges; on the left is the <display pane> which shows the output.

Two sets of controls are used to operate the manual analysis. The first is the <Mode> drop-down menu. The second set are the bank and channel selection controls. The mode controls 1) how the analysis results are displayed in the <display pane>, and 2) the available bank and channel selection controls. Each mode is described in appendix C.

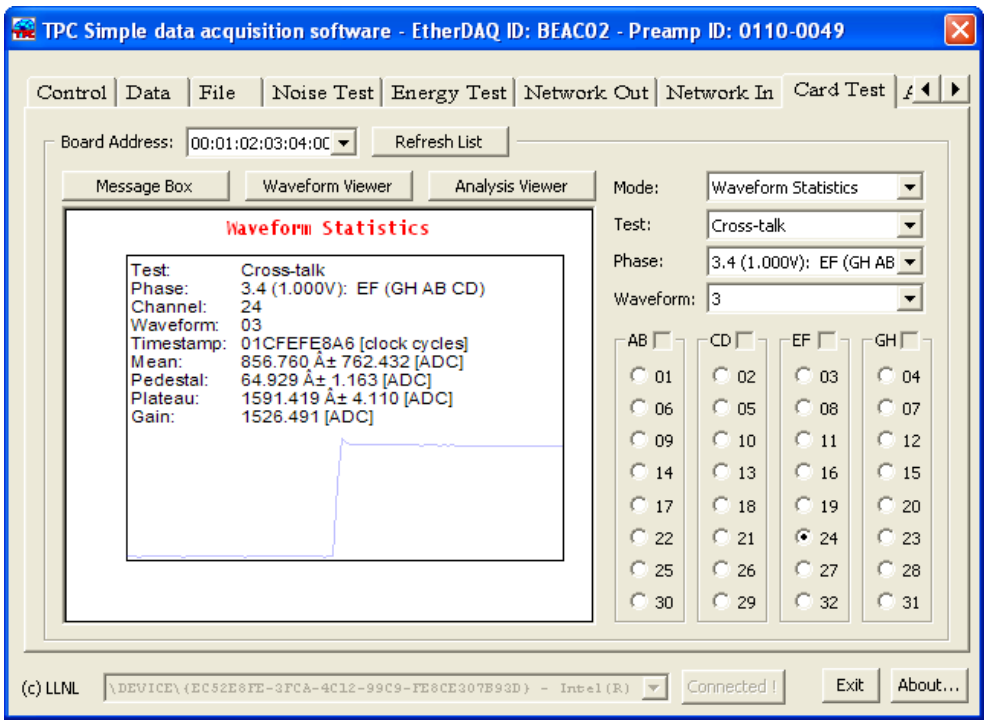
### 3.5.3 Card Testing Procedures

One and only one of the three primary procedures must be selected: 1) Quick Linearity, 2) Single Channel, and 3) Cross Talk. Burn-in, the fourth selection, is optional and can be selected along with with one of the primary procedures.

These procedures determine the how the pulse generator is configured, which data are recorded, and if a preliminary analysis method is performed.




**(a)** Pedestals Histogram




**(b)** Waveform Statistics

**Figure 3.7** – Waveform Analysis Tools Perspective

#### 3.5.3.1 Single Channel

EtherDaqGUI is set to test only one channel. This is used in conjunction with the single channel testing equipment described in section 2.2.2. EtherDaqGUI configures the first output on the pulse generator as the pulse source.

The desired dynamic range determines which bank on the test stand to use; refer to table 2.1 for the available capacitances and connecting banks. The number of the channel being tested[30] must be entered into the edit box located to the right of the <Single Channel> radio button.

#### 3.5.3.2 Cross Talk

The cross talk procedure fulfills the primary role of the automated card testing—it is the most definitive and comprehensive test performed. The data generated from the cross talk test is used extensively by the analysis framework (see chapter 4). The distribution board is used for this test.

The test names comes from the focus of the test parameters to investigate the system for cross talk. Voltage pulses are sent to only one bank (and its 8 corresponding channels) at a time—the other three banks and 24 corresponding channels are idle. The sync output of the pulse generator is connected to the force trigger BNC connector on the test stand. A TTL[31] pulse is sent to the sync output coincident with the leading edge of the voltage pulse. This forced trigger causes the EtherDAQ FPGA to record data for all 32 channel, regardless of whether the filtering algorithms detect a signal or not. Any cross talk existing in the system can then be detected and/or recorded by the idle channels.

[30] 1-based
[31] transistor-transistor logic

The procedure is divided up into four phases. Each phase pulses only one bank, and the phases each pulse a different bank. Each phase contains a number of subphases equal to the number of voltage steps. The first subphase sends pulses at the minimum voltage. Amplitudes for the subsequent subphases increase linearly, with the final subphase voltage at the maximum voltage. Separately pulsing each bank fulfills the cross talk measurement requirements.

Performing each phase with a number of subphases provides data for two additional quality checks. First, it allows a linearity test to be performed on each of the channels. As described previously, linearity is a required characteristic for the intended use of the preamp cards. Second, it allows for the discovery of non-linear cross talk. The calibration and analysis is based on the assumption of linear cross talk. The presence of non-linearity would indicate either a flaw in the analysis assumptions or a bad channel; either would require further detailed interrogation.

#### 3.5.3.3 Quick Linearity

Quick linearity was designed to be a fast test that can be used to verify that each preamp channel exhibits a linear response. In consists of only a single phase with all four banks pulsed simultaneously. Cross talk cannot be evaluated using this test data, although the distribution board is used.[32] The linearity test results can be viewed with the <Linearity> analysis tool described in table C.1.

[32]It is assumed that the cross talk scales linearly with the pulse amplitude, and thus can be overlooked in this context

#### 3.5.3.4 Burn-in

This test is optional, and can be enabled by selecting the <Burn-in> check-box on the main <Card Test> tab. The burn-in can be used with any of the primary tests, and is always the first operation when enabled.

The pulse generator is idle for a specified time, which allows the electronics to reach a steady-state condition. Additionally, EtherDaqGUI does not send any communications to any device. It has been observed that altering settings on both the pulse generator and EtherDAQ will occasionally cause an unexpected waveform event, especially with the DG645 pulse generator.

This idle time is also used to identify poor or noisy channels on the preamp card. Since no pulses are being introduced into system, the system logs any spurious waveforms that may be generated by the preamp card. A channel is flagged if the count exceeds the given threshold parameter.[33] The entire card is considered faulty if a preamp card has too many flagged channels; the testing is terminated and the user notified.

### 3.5.4 Data Export Files

EtherDaqGUI generates all output in text form. All output files are saved into the directory displayed in edit box below the Main Save Folder label. The path is selected using a folder browser which is opened by clicking on the <...> button.

The generated output depends on the testing procedure. All of the test procedures record data to the Waveform Data and Analysis Information files. The single channel test generates an additional Single Channel file.

[33] Waveforms collected from flagged channels are ignored during the primary test

All files are written in text-based output and formatted so that they are human-readable. Only the layout of the Waveform Data is explicitly defined below; it is designed to be parsed by the analysis framework (see chapter 4) and must follow a strict formatting structure. The contents of the two other file types are self-explanatory. The headers and footers of the files all contain the same information.

#### 3.5.4.1 Waveform Data

**Naming:** `*.txt`
**Purpose:** **Stores the measurement values of the collected waveforms**

The Waveform Data file is the primary method of transferring the data collected by EtherDaqGUI to the analysis framework. The file format, shown in Table 3.1, is well defined so that the analysis framework can parse it correctly. Any changes to the format made in EtherDaqGUI will need to be also implemented in analysis framework—no time was invested into maintaining an option for backwards-compatibility with different formats since the additions to EtherDaqGUI were essentially finalized when development on the analysis framework was begun.

Lines containing waveform data begin with 'Sample.' All other lines are either empty or begin with a '#' symbol. This syntax removes the need to specify the number of waveform data entries in each data block; parsing of the waveform data 'Sample' lines continues until the '#' character is encountered. The remainder of that line is then parsed to determine whether to read another data block or the file footer.

#### 3.5.4.2 Analysis Information

**Naming:** `*.ana`
**Purpose:** **Contains a quick overview of the analysis parameters and results**

The first data block contains all the test parameters from the <Advanced Settings> dialog, which govern the data collection and preliminary analysis behaviors. This

**Table 3.1** – Waveform Data File Contents

| Line | Description | Format |
|---|---|---|
| | | *Header* |
| B1 | comment | `# ⟨S\|purpose of file\|⟩` |
| B2 | EtherDAQ serial number | `# EtherDAQ ID: ⟨S⟩` |
| B3 | preamp identification | `# Preamp ID: ⟨NNNN⟩-⟨NNNN⟩` |
| B4 | original save directory | `# File save directory: ⟨S\|path\|⟩` |
| B5 | save name | `# File save name: ⟨S\|name\|⟩` |
| B6 | test start date | `# Start Date: ⟨X\|month\|⟩-⟨X\|day\|⟩-⟨X\|year\|⟩` |
| B7 | test start time | `# Start Time: ⟨X\|24hour\|⟩:⟨X\|min\|⟩:⟨X\|sec\|⟩` |
| | | *Data blocks, repeated, one per subphase* |
| R1 | phase voltage | `#### voltage: ⟨F⟩ ####` |
| R2 | initial temperature | `#### Temperature: ⟨F⟩ ####` |
| R3 | waveform data | `Sample Time:⟨H⟩ Ch⟨NN\|channel\|⟩ ⟨X\|values\|⟩...` |
| ... | | |
| R⟨X⟩ | waveform data | `Sample Time:⟨H⟩ Ch⟨NN\|channel\|⟩ ⟨X\|values\|⟩...` |
| | | *Footer* |
| E2 | test end date | `# End Date: ⟨X\|month\|⟩-⟨X\|day\|⟩-⟨X\|year\|⟩` |
| E1 | test end time | `# End Time: ⟨X\|24hour\|⟩:⟨X\|min\|⟩:⟨X\|sec\|⟩` |

This table uses substitutions; see table A.2 for more information
'B' represents a line relative to the beginning of the file
'E' represents a line relative to the end of the file
'R' represents a line relative to the beginning of a repeated block

information is available be parsed by the analysis framework in addition to the `Waveform Data` file.

If the optional Burn-in procedure was selected then a `Burn-in` section is included in the output. The Burn-in section contains a list of all the spurious waveforms detected during the Burn-in for each channel.

A `Waveforms` block follows, with sections for each of the selected test procedures. This block contains tabulated data of the average waveform amplitude in each subphase for each channel. The total number of waveforms collected is then

reported. Lastly, the configuration details of each subphase are reported, as well as the EtherDAQ temperature at the beginning of that subphase.

#### 3.5.4.3 Single Channel

**Naming:** `*.csv`
**Purpose:** **Reports the statistics for the channel over all the voltage phases**

The `Single Channel` file is designed to provide more in-depth analysis information than the `Analysis Information` file, but is generated only by the Single Channel test procedure. Further, it's structure is somewhat different from the other two files. It is written as a csv[34] file, which eases importing the data into a spreadsheet application when quick-turnaround analysis is required—most spreadsheet applications have the functionality for quickly importing and plotting csv-formatted data. An additional advantage is that no further development was required, in either the analysis framework or in creating a new program, to work with this new data type.

The main focus of the file is tabulated analysis data for each of the pulses in each subphase. The gains are reported, as well as the pedestal and plateau values and uncertainties.

[34] comma separated value

# Chapter 4

# Analysis Framework

An analysis framework is needed to perform the data analyses required for the routine calibration of the E/P card pairs. The primary requirement is a flexible design that allows for a variety of analysis methods and tools, adaptable for future needs and developments. The framework must also be able to load data generated by the test stand and provide powerful tools for visualization and interrogation. Finally, it must be capable of generating the calibration constants for use in the NIFFTE reconstruction software.

The name of the analysis framework is Preana, for preamp analysis. Preana is an extensible, intuitive platform for interfacing with the test stand data to perform calibration analysis. Preana loads the `Waveform Data` files generated by EtherDaqGUI, which data are then fully available to the user for plotting and analysis. The `Minuit2` minimization library is built into Preana, and provides an extensive suite of tools for for fitting the test stand generated waveforms, extracting fitted parameters, and performing statistical analyses. The framework provides users with an initial example set of fit functions that can be easily modified or

replaced for more exacting analyses. These analyses are required to obtain the parameters used for the calibration measurements.

Preana also contains a dynamic and resizable interface. The capability for a larger window size is especially useful when dealing large analyses or multiple data sets. Another feature is that Preana is task-specific, which allows for optimization of the analysis speed and efficiency. The code framework is designed to enable development of the required tasks while allowing for flexibility in future developments.

## 4.1 Design

Preana is written in C++, compiled using Microsoft Visual Studio against the Windows native API[1], and contains extensive application of OOP[2] concepts and meta-programming techniques.

The GUI is designed to be clean and intuitive. A main menu provides access to all the functionality. A group of panes across the top, collectively known as the <Data Browser>, is used to navigate the data sets currently loaded into memory. <Progress Manager> windows display the progress to completion for all running jobs, and automatically hide themselves when no jobs are left. The remaining space is reserved for the <Analysis Display>, which is used to display the <Waveform Display> and <Histogram Display> windows. The <Analysis Display> always contains a <Preview Pane> window, which is identified by the light gray 'WAVEFORM PREVIEW' text. It is used to quickly preview a waveform selected in the <Waveforms> pane.

[1] application programming interface
[2] object-oriented programming

Internally, the test data are stored hierarchically. This provides intuitive access for the user and simpler storage controls for the programmer. This also reduces the amount of redundant information involved in uniquely identifying or categorizing a waveform during analysis.

A few external packages are incorporated into Preana to support its work. The primary external package is `Minuit2`, a package for performing minimization. `Minuit2` is used to fit representative functions to the waveform data and extract the best-fit function parameters and fit uncertainties. Preana also has the ability to export the results and statistics of these fitting analyses for external validation and verification by the `ROOT Data Analysis Framework`.[3] The other packages, `libpng` and `libjpeg`,[4] provide the ability to save any visualizations as images.

Preana also contains a detailed job-scheduling system—many of the analysis tasks can be run in parallel. Modern computing systems often contain multiple processing units, whether logical or physical. Scheduling multiple jobs harnesses this available computing power to speed up the analysis process. Included with the job scheduling system is an event-based signaling interface.

### 4.1.1 GUI Framework

Although the extensively documented Windows API is modernized for use with C++, its design is not inherently OOP-friendly. The Windows API is based on a message-passing system to function. These messages are used to control and interact with the GUI elements. GUI elements are entities such as windows, dialog boxes, menus, and buttons.

[3] More information on ROOT can be found at the project website: `https://root.cern.ch/`

[4] `libpng` depends on the package `zlib`, which is also included but not used directly by Preana.

A specialized function, called a message handler, is used to process the messages and perform the required action.[5] All of these message handlers must exist as static functions. Otherwise, the compiler is unable to match the function signature to the form required by the Windows API. This process is fairly simple as long as there is only one window per message handler. The process becomes more complicated if multiple elements based on the same message handler are required. The process becomes even more complicated when developing in a pure OOP fashion.

#### 4.1.1.1 Base Window Class

The template `WinProcClass`[6] leverages CRTP[7] capabilities to overcome the difficulty of meshing the Windows API requirements with OOP design ideals. It is used to pair a window to a controlling singleton class. An example is the `AboutWindow`[8] class.

`WinProcClass` contains the necessary static function, `WinProcClassStatic()`,[9] to be compiled properly against the Windows API. The CRTP is what allows a unique static function to be created for each derived class. There are also two pure virtual function that must be implemented by any derived class: `MessageHandler()`[10] and `OpenWindowSpecific()`.[11] These two functions are declared as protected so that only a derived class has access.

[5] Behaviors are usually well-defined and understood for elements such as buttons. A defined message handler is not required—the Windows API uses an internally-defined message handler to provide the default behavior.

[6] `WinProc Class.h`

[7] curiously recurring template pattern

[8] `About Window.(h/cpp)`

[9] `WinProc Class.h:28-31`

[10] `WinProc Class.h:49-52`

[11] `WinProc Class.h:53`

MessageHandler() is a non-static member function that processes the messages as desired. One of three values must be returned back to WinProcClass, depending on the result of the processing. These are: 1) WinProc_Message_Completed, 2) WinProc_Message_Needs_More, or 3) WinProc_Message_Not_Processed.

OpenWindowSpecific() must actually create the window. This is typically performed with the Windows api CreateWindowEx() function. The derived class can perform any necessary initializations for its child windows in this step. The handle to the newly created window much be returned.

Three other virtual functions are optional and needed only if the window processes commands in a non-default manner: CommandAccelerator(),[12] CommandControl(),[13] and CommandMenu().[14] Any implementation of these functions must also return one of three values back to WinProcClass. These are: 1) WinProc_Command_Processed, 2) WinProc_Command_Terminate, or 3) WinProc_Command_Not_Processed.

#### 4.1.1.2 Dynamic Sizing

WinProcClassDynamicSizing[15] builds on the foundation of WinProcClass to add dynamic resizing capabilities. It is itself another crtp. A class derived from WinProcClassDynamicSizing automatically contains built-in functionality to react to size changes in the parent window. Additionally, it contains itself to a designated area within the parent window. Examples of derived classes are DataBrowser[16] and ProgressManagerCRTP.[17][18]

[12] WinProc Class.h:42
[13] WinProc Class.h:44-46
[14] WinProc Class.h:47-48
[15] WinProc Class Dynamic Sizing.h
[16] Data Browser.(h/cpp)
[17] Progress Manager CRTP.h
[18] The ProgressManagerCRTP class is discussed further in section 4.1.4.

This capability requires the use of the CM_Window_Parent_Resized notification message. A dynamic resizing window will only query the status of its parent window, and resize if needed, upon receipt of this message. A window containing dynamically-sizing child windows must send this message only after it has completed resizing. Otherwise the information a child window retrieves about its parent window may be outdated and incorrect. The result of a mistimed notification will cause undefined behavior.

The MessageHandler() interface is changed by WinProcClassDynamicSizing—it is replaced with DynamicWindowMessageHandler(),[19] another pure virtual function. The expected return values are identical.

Finally, an additional virtual function ResizeAndRelocateWindow()[20] is available. It provides default behavior for resizing upon receipt of the CM_Window_Parent_Resized message. It can be overridden if further actions are required. In some situations it is desirable to both maintain default behaviors as well as provide specific functionality; the preferred solution is to use scoping[21] instead of copying and pasting code.

#### 4.1.1.3 Managing Multiple Windows

Support for multiple windows is implemented with two layers of abstraction. The first, DisplayBase,[22] is a standard C++ abstract class. The second, DisplayCRTP,[23] is essentially a CRTP wrapper around DisplayBase. The CRTP usage causes the compiler

[19] WinProc Class Dynamic Sizing.h:42-45

[20] WinProc Class Dynamic Sizing.h:39-41

[21] For an example of usage, see the overriding function ResizeAndRelocateWindow(), Analysis Display Organizer.cpp:468-504

[22] Display Base.(h/cpp)

[23] Display CRTP.h

to create a unique static message handler for each derived class. Thus, classes that are designed to support multiple windows must inherit directly from DisplayCRTP.

Multiple windows are managed by storing each class instance pointer as a value in the user data[24] of the associated window. The static message handler retrieves the user data using the window handle accompanying a message. The result is cast to a DisplayBase[25] pointer, which is then used to call the class-specific message handler. DisplayProc()[26] provides the required static message handler, and MessageHandler()[27] is the pure virtual message handler that must be implemented in the derived classes.

All derived classes are responsible for painting both the background and foreground of the window. This mandate is enforced by the pure virtual function PaintForeground()[28] and PaintBackground().[29] An additional pure virtual function, PostPaint(),[30] provides a hook for performing operations after the painting process is completed.

#### 4.1.1.4 Main Window

The MainWindow[31] class is the core of Preana. It inherits directly from WinProcClass, contains Preana's main message handler, controls the GUI, and is responsible for the management of the top-level data. It also is responsible for controlling data access permissions between other objects.

[24] The user data can be retrieved and set using GetWindowLongPtr() and SetWindowLongPtr(), with GWLP_USERDATA as the parameter name.

[25] Display Base.cpp:123

[26] Display Base.h:48-51

[27] Display Base.h:53-56

[28] Display Base.h:62-63

[29] Display Base.h:57-58

[30] Display Base.h:69-70

[31] Main Window.(h/cpp)

The main window, operated by MainWindow, contains a number of child windows. These are the <Data Browser>, the <Analysis Display>, and a few progress managers. Each of the child windows are based on WinProcClassDynamicSizing so that they can react to resizing events.

MainWindow also manages the data analysis. Analyses are requested by the user through either the main menu or the <Data Browser>. A message with the necessary information is dispatched to MainWindow, which then creates the operation and queues it with the appropriate progress manager.

### 4.1.2 Data Storage

Tests are loaded into Preana using the main menu, and are expected to include data from all 32 channels. The menu navigation for loading test data is: <File> ▷ Load Preamp Card ▷ (Select test type). A <File Preview> pane is displayed at the bottom of the dialog box when selecting files. This makes it easy to visually verify the file contents that are being loaded.

The internal data are frozen once they have been imported by declaring them as const data types, i.e. the values cannot be changed. The test data are organized in a hierarchical tree-like structure. Each instance of a data storage class uses a vector to stores instances of the next-level-down data class.

#### 4.1.2.1 Loading Data Sets

A cross talk procedure data set is assumed to have a complete set contained in one file, while a single channel test is inherently split up over 32 files. Loading data from a single channel procedure requires the selection of the corresponding file for each channel. Two methods, manual or automatic (see fig. 4.1a), are available

for the single channel file selection. The automatic mode allows the user to select a directory (see fig. 4.1b) to scan for matching files.[32] The directory scan results are transferred into the file selection dialog (see fig. 4.1c). The user can then add or correct the results of the scan.

The file selection dialog is also used for the manual mode. However, it is not pre-filled with any file names. It is highly recommended that the single channel test files be appropriately named and grouped in directories sorted by E/P pairing to take advantage of the time-saving automatic mode. Manually selecting all 32 files should be done only in dire circumstances.

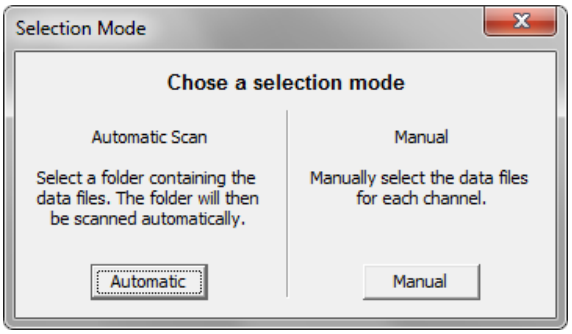
#### 4.1.2.2 Classes

**Test**

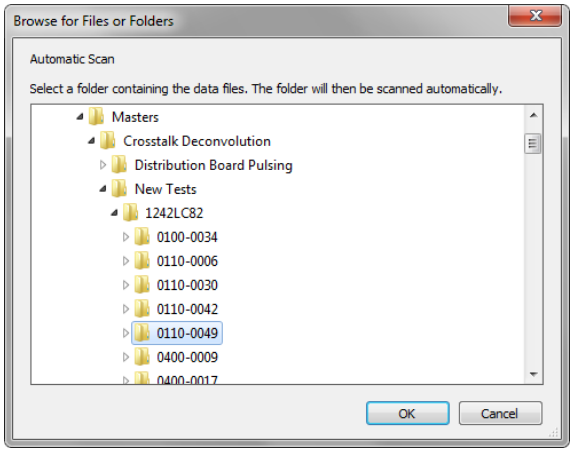
The test type is based on the origin of the data. Three test types are currently implemented, and are listed in table 4.1. All test types derive from the `TestBase`[33] class, which contains the top-level functions for loading data sets, analysis, and exporting results. It also declares three pure virtual methods, which must be implemented by derived classes. These are `Analyze()`,[34] `CalculateDiagonal()`,[35] and `FillExportData()`.[36]

`Test Base` contains `LoadFile()`,[37] a universal function for opening data files. A derived class is responsible for calling `LoadFile()` for each file that needs processed. `LoadFile()` first prepares the class instance for receiving the data and then queues

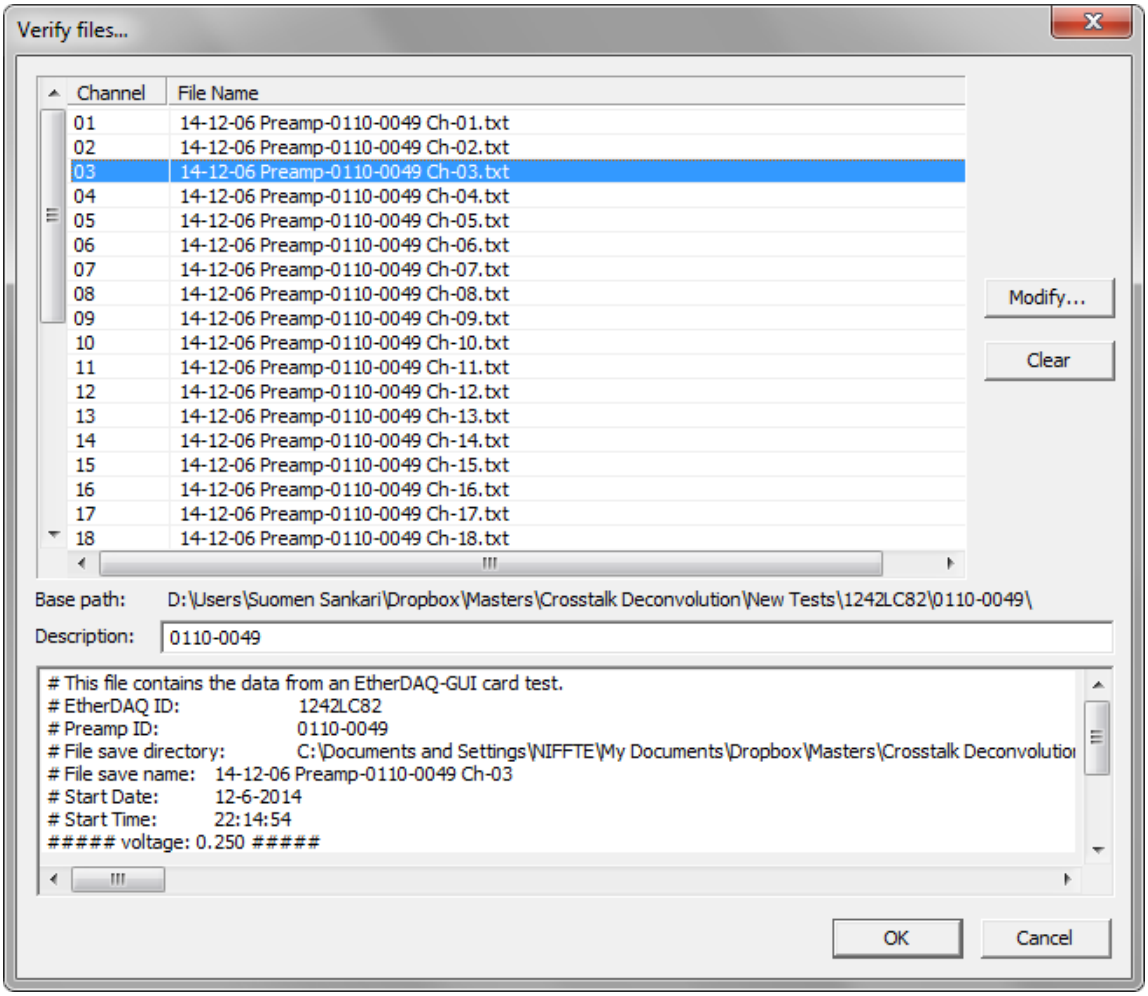
[32] The directory scan examines the file names for a pattern matching 'Ch⟨XX⟩'
[33] `Test Base.(h/cpp)`
[34] `Test Base.h:58`
[35] `Test Base.h:59`
[36] `Test Base.h:63-64`
[37] `Test Base.h:66-67`




**(a)** Selection Mode




**(b)** Folder Browser




**(c)** File Selection

**Figure 4.2** – Single Channel File Selection

**Table 4.1** – Test Types

| Name | Contents |
|---|---|
| **Distribution** | Cross talk test data, performed using the breakout board to simultaneously pulse all 32 channels. |
| **Single Channel** | Single channel test data, performed using the single channel approach to individually test each channel. |
| **Comparison** | Results of an analytical comparison results between two or more tests of any test type. |

the process with `FileLoadingProgressManager`.[38] Another function, `QueueAnalysis()`,[39] queues the analysis for a channel with `AnalysisProgressManager`.[40] Derived classes use this function to perform the actual analyses when implementing `Analyze()`.

Both `...ProgressManager` classes are job schedulers, and described in further detail in section 4.1.4.

### Card Pair

`CardPair`[41] exists primarily because of the data set segmentation—splitting over multiple files—of the single channel test type. An instance of `CardPair` stores the extracted data from just one file. Thus, a distribution-type test will store only one `CardPair` instance, while a single channel-type test will store thirty-two.

Although `TestBase` is responsible for adding tasks to the queue, `CardPair` is the class that actually contains the functions for performing the data parsing and analysis tasks. These are, respectively, `LoadData()`[42] and `RunAnalysis()`.[43] Both

[38] `File Loading Progress Manager.h`
[39] `Test Base.h:70-72`
[40] `Analysis Progress Manager.h`
[41] `CardPair.(h/cpp)`
[42] `Card Pair.h:44`
[43] `Card Pair.h:53`

functions are written for use in a threading context, which is the core of the parallelization capabilities of Preana.

Thread-contextual functions must be static and can have only one argument to match the required function signature. CardPair is equipped to work with this restriction. Two functions, MakeLoadingProgressBarParameters()[44] and MakeAnalysis-ProgressBarParameters(),[45] each create a structure that contains all the necessary parameters. The pointer to a structure is passed in as the single argument. The function is then able to extract the information inside the structure and perform the job correctly.

### Channel

The Channel[46] class is very simple. It exists primarily to provide a logical structure to the data.

### Phase Set

Much like Channel, the PhaseSet[47] class is primarily an organizational entity. However, it does have a unique identifying function: IsPulsedPhase().[48] This function returns true if its class's data correspond to a primary pulse generated during a test procedure.

[44] Card Pair.h:48-50
[45] Card Pair.h:45-47
[46] Channel.(h/cpp)
[47] Phase Set.(h/cpp)
[48] Phase Set.h:38

#### Voltage Set

The data stored by the VoltageSet[49] class maps to the data collected during a subphase. A subphase is identified by the voltage of the driving pulses, hence the class name.

VoltageSet, in additional to its organizational role in the data hierarchy, is responsible for aggregating the analysis results from the waveforms it contains.

#### Waveform

Waveform[50] is the foundation of the data storage and analysis for Preana. Each instance holds the data values and analysis results for a single waveform.

The waveform values are stored in a const vector. The dynamic array capabilities of the vector avoids restrictions or requirements on the length of the waveform, i.e. how many data values is contains. The const qualifier prevents any future modification of the values. This allows a reference or pointer to the vector to be passed around without fear of accidental or intentional modification.

### 4.1.3 Data Exploration and Interrogation

The <Data Browser> is designed to provide intuitive access to the data, from picking a test for analysis to selecting a waveform to view. The <Data Browser> contains five panes for browsing the data. The order of the panes is nearly identical to the internal data hierarchy (see section 4.1.2.2).

The <Data Browser> is linked closely to the <Analysis Display> area, where all the display windows are shown. Together, the two provide the primary interface of Preana (shown in fig. 4.3).

[49] Voltage Set(.h/cpp)
[50] Waveform.(h/cpp)

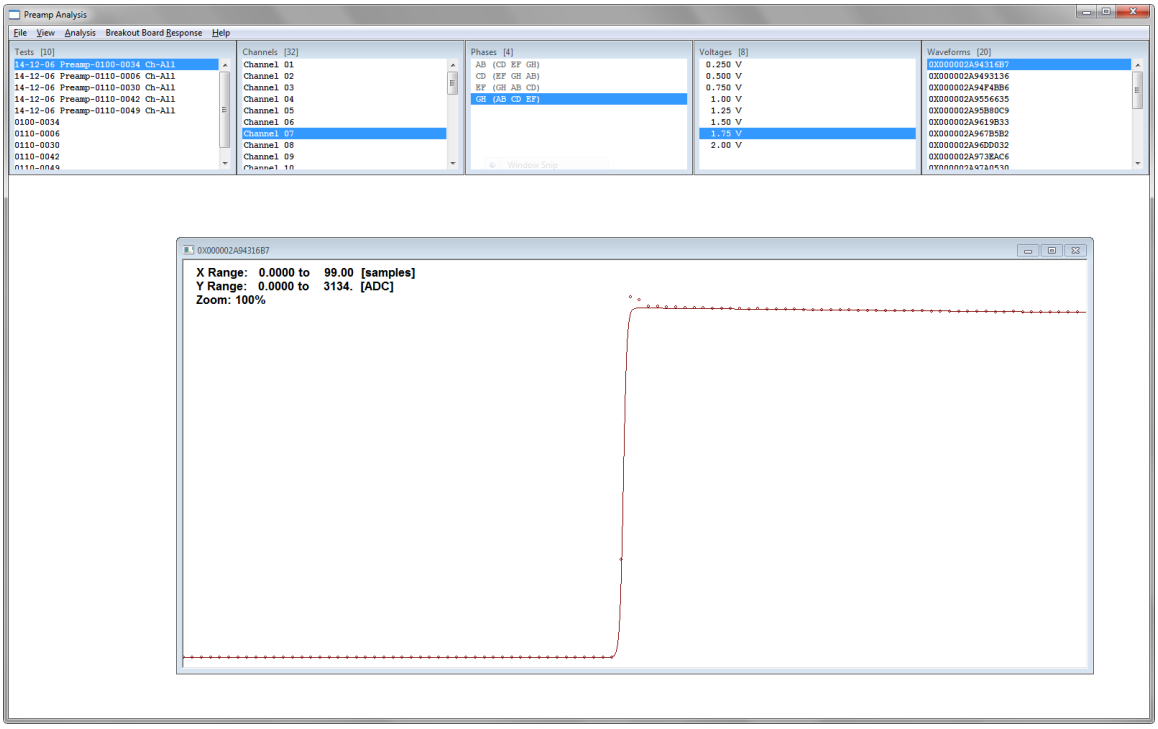



**Figure 4.3** – Data Browser and Analysis Display

#### 4.1.3.1 Data Browser

**Tests**

First, the <Tests> pane lists the loaded test data sets. Only Distribution and Single Channel tests (see section 4.1.2.2) are listed. Left-clicking on a test will highlight it and permit the selection of an item in the <Channels> pane. A right-click displays a pop-up menu with analysis options. The pop-up menu also contains a <Delete test> item, the only current mechanism for deleting a loaded test from memory.

Text and highlighting colors are used to represent the current status of a test. A selected test will be highlighted using the system-defined color, typically blue. A test with an error will be highlighted with orange.[51] Normal text is displayed

[51] RGB value of 0xFF9900

using the system-defined color, typically black. Gray text indicates that the test is currently being loaded from data files. Blue text means the test is currently being analyzed. Green text means that the analysis results are being exported. Red text means the test is being deleted from memory.

#### Channels

Next is the <Channels> pane, which always displays items for all 32 channels. However, the list items are displayed in black and selectable only when a valid test is selected. Otherwise, the items are displayed in a gray text and cannot be selected.

#### Phases

The third pane, <Phases>, lists all the recorded phases for the selected channel. In the event of a single channel test only one phase is listed. A cross talk test will have four, with the primary phase in black and the three secondary phases in gray—all the phases are selectable, the coloration is provided only for identification purposes. A right-click on an item displays a pop-up menu with analysis and plotting options.

#### Voltages

Fourth, the <Voltages> pane list the subphases. A right-click on an item displays a menu with analysis and plotting options.

#### Waveforms

Fifth, the <Waveforms> lists all the waveforms in the selected subphase. Special mouse actions are available in this pane. A single left-click displays the waveform

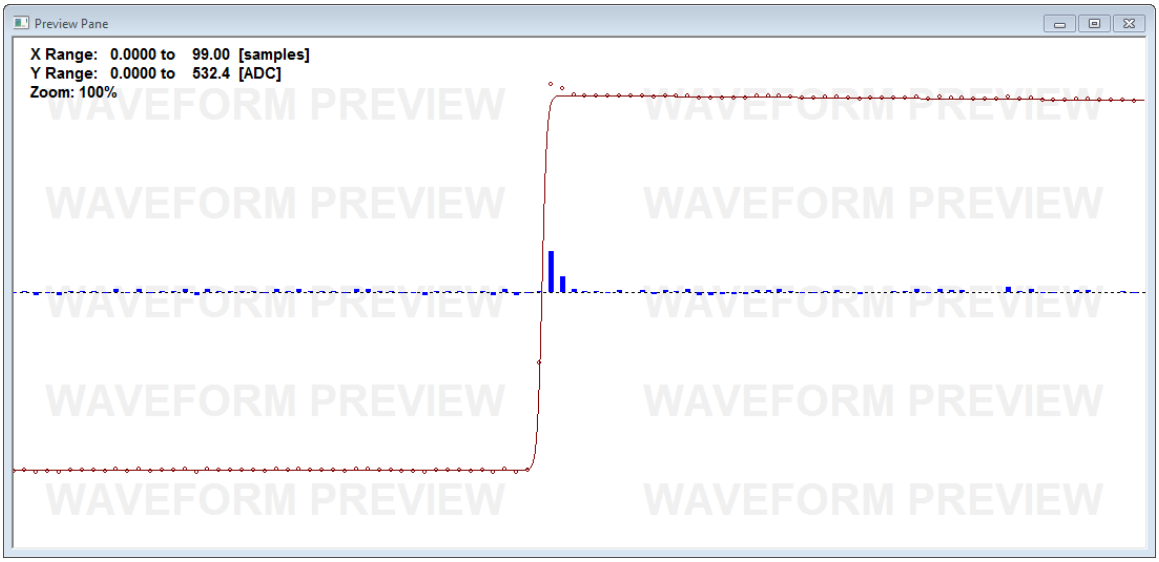



**Figure 4.4** – Preview Pane

automatically in the <Preview Pane>, shown in fig. 4.4. A double left-click opens the waveform up in a new <Waveform Display> window. A Ctrl+left-click adds the waveform to the top-most <Waveform Display> window. A right-click displays a pop-up menu with extended options for analysis or plotting.

#### 4.1.3.2 Analysis Display

The <Analysis Display> window is managed by the AnalysisDisplayOrganizer[52] class, and covers roughly 80% of the main window. Initially <Analysis Display> contains only an empty <Preview Pane>. It can support child windows that inherit from DisplayCRTP (see section 4.1.1.3); currently only WaveformDisplay[53] and HistogramDisplay[54] are implemented.

All classes that inherit from DisplayBase have a built-in pop-up window. The option to export the contents of the window to an image file is usually available.

[52] Analysis Display Organizer.(h/cpp)
[53] Waveform Display.(h/cpp)
[54] Histogram Display.(h/cpp)

A derived class can add more options by overriding ProcessContextMenu().[55] An example is shown in WaveformDisplay::ProcessContextMenu().[56]

AnalysisDisplayOrganizer is derived from WinProcClassDynamicSizing, although the child windows it contains do not. As such, AnalysisDisplayOrganizer dynamically resizes any child windows as its size changes. This prevents a child window from becoming lost in the virtual coordinate space that exists beyond the window viewport.[57]

**Waveform Display**

WaveformDisplay is designed around the need to display one or more waveforms in an accessible manner. Each waveform is drawn as a line with a unique color and pattern. Eight colors and four patterns are available, which allows up to 32 waveforms to be simultaneously and uniquely displayed. Figure 4.5 shows a WaveformDisplay plotting together one waveform from each subphase of a phase. Note that the window title includes the number of waveforms it contains.

A number of navigation controls assist in inspecting the waveforms, and are summarized in table 4.2.

The Preview Pane is a specialized application of WaveformDisplay. It has a permanent watermark in the background and can only display one waveform at a time. If a fit has been performed on the displayed waveform then residual bars are drawn, showing the difference between the fit and the actual values at three times the waveform scale. Blue bars are normal, while red bars indicate that the scaled value

[55] Display Base.h:64-65

[56] Waveform Display.cpp:786-815

[57] The Windows API uses the top-left had corner the origin, with positive directions being down and left. However, nothing prevents any part or all of a child window from moving outside of the edges that define the parent window's borders. As such, anything that causes a child window to exist beyond these edges would effectively hide it from the user—it would become lost and unusable—all the while continuing to exist in memory and consuming processing resources.

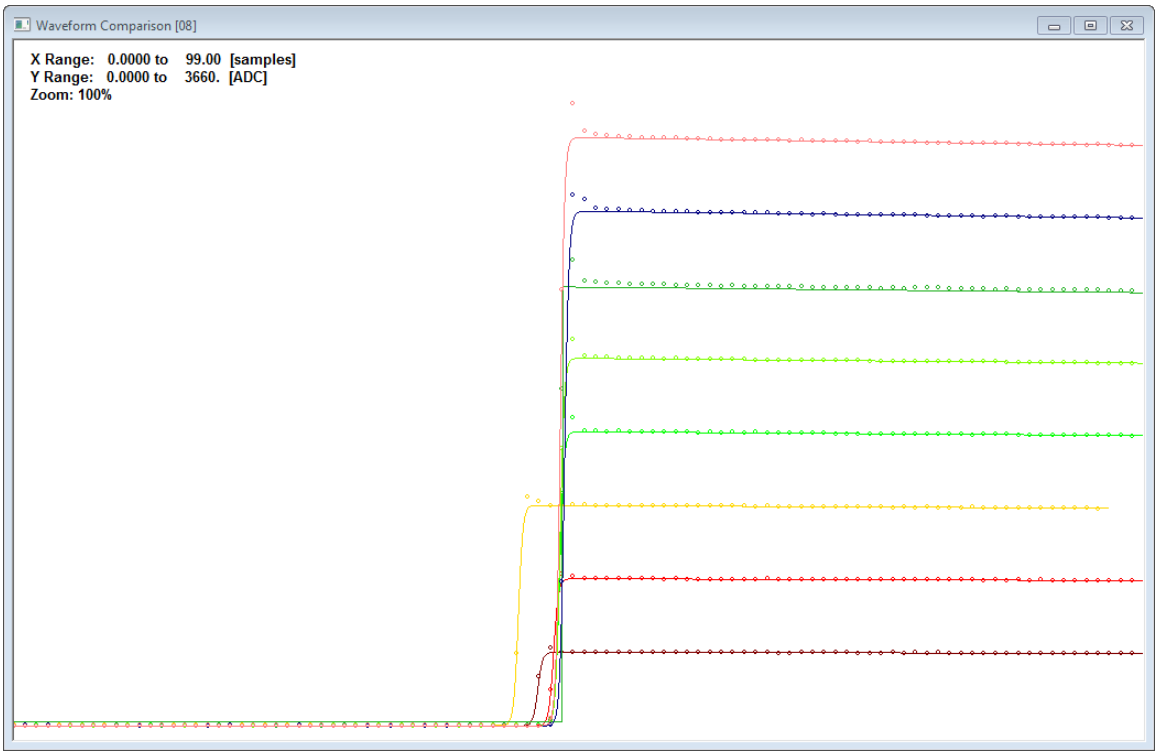



**Figure 4.5** – Display of Multiple Waveforms

**Table 4.2** – Waveform Display Controls

| Mouse Event | Action |
|---|---|
| Double Left-click | Zooms in on the location of the cursor using the current aspect ratio |
| Left-click + Drag | Moves the view with the mouse cursor |
| Right-click | Opens a pop-up menu with options for saving the current view as an image or changing the zoom level |
| Right-click + Drag | Draws a zoom box, which becomes the new view when the button is released |
| Scroll Wheel | Zooms in or out using predefined increments centered on the mouse cursor; resets the aspect ratio |

exceeds the view. Finally, the right-click pop-up menu contains only an option for exporting the contents to an image file.

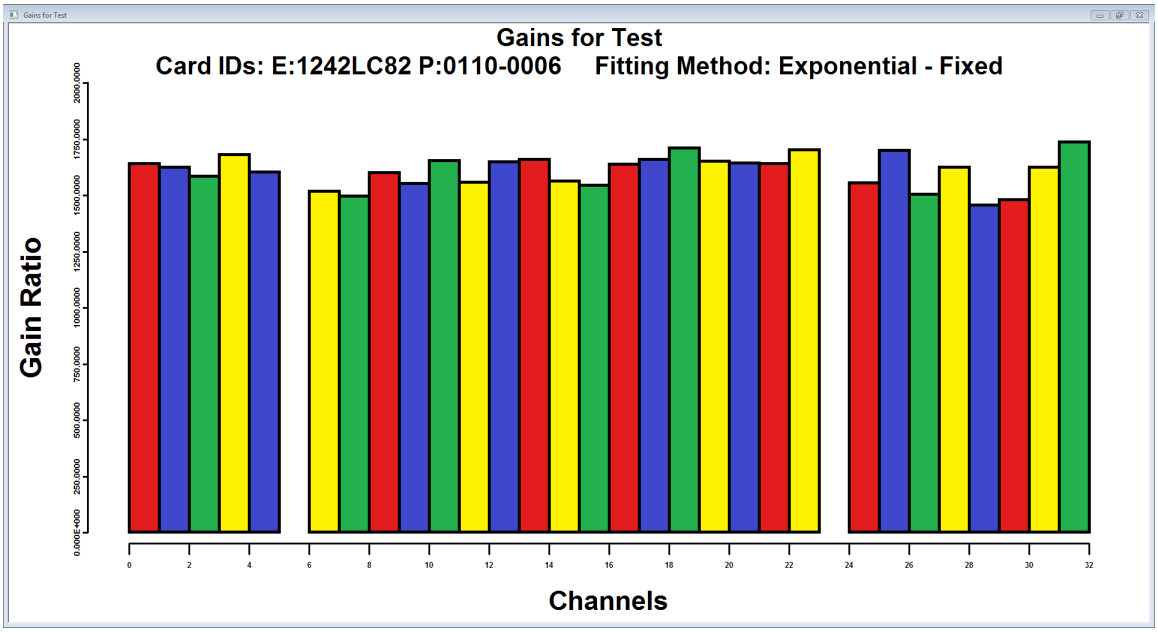
### Histogram Display

`HistogramDisplay` is a custom-built 1D histogram viewer. It behavior is customizable, and can include any of the following items:

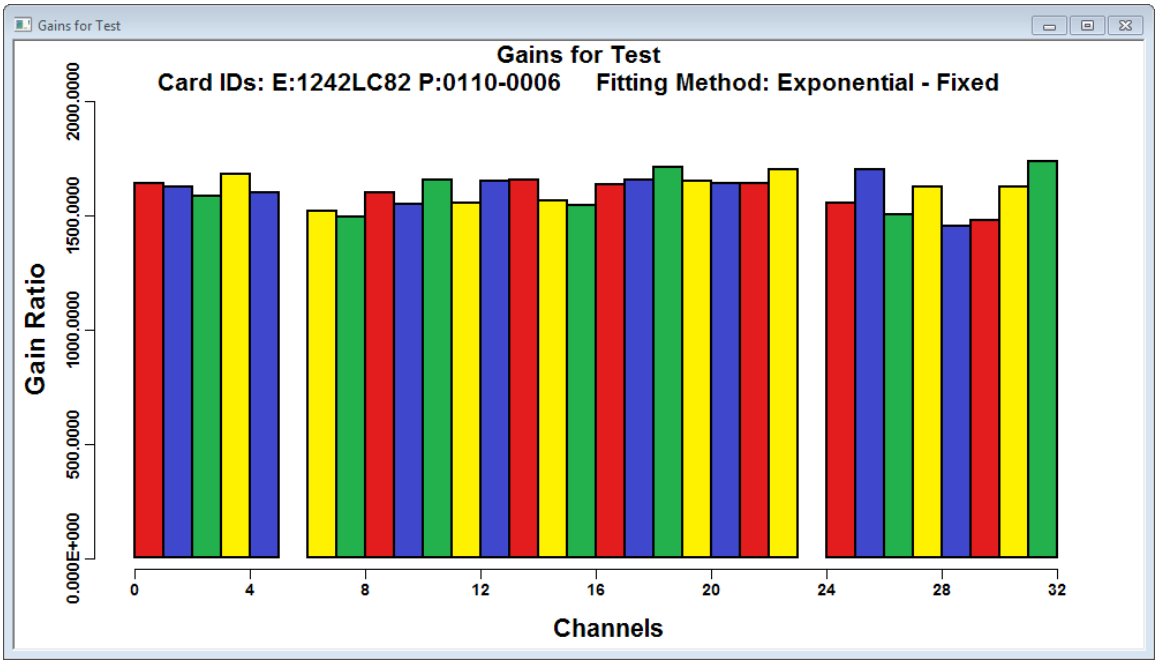
- displaying different data sources
  - dynamically generated bin edges and values from a vector of data
  - pre-calculated bin edges and values
- auto-sizing text labels based on window size (title, axes, bins)
- error bins on either side for out-of-range values
- support for programmer-defined bin colors

An example is shown in fig. 4.6, which exhibits the text auto-sizing and bin color features. Both subplots display the same histogram. Only the window size is different. Figure 4.6a is maximized to fill the entire `<Analysis Display>`, and fig. 4.6b is 75% smaller. The relative sizes of the text are essentially equivalent, while the number of axis tick marks are different. Differences are also seen in the size of the windows' title bars. The colors correspond to the bank colors established within the NIFFTE collaboration (see fig. 2.3).

Typically the histograms are used to plot aggregated results from analyses. However, the `HistogramDisplay` is also used to display the results of a comparison test.




**(a)** Full Window Size




**(b)** 25% Full Window Size

**Figure 4.6** – Text Auto-sizing

### 4.1.4 Job Scheduling

Threading and job scheduling are necessary components of any heavy-duty analysis system. Any long or resource-intensive tasks should be performed on a thread separate from the message queue, especially because of the way Windows interacts with programs—performing a long operation within the message queue causes the application to appear unresponsive or frozen.

Threading is a powerful tool for performing parallelizable tasks. In Preana, the tasks for the loading and the analysis of tests are both extremely parallelizable. These characteristics harness the available system resources and reduce the overall duration of an analysis.

#### 4.1.4.1 Thread Management

The core of Preana's thread scheduling and management is the CRTP `ThreadManager`.[58] A thread manager is necessary to prevent a large number of simultaneous threads from monopolizing resources and choking system-wide responsiveness. For example, the analysis of a test spawns 32 threads, one for each channel. An average workstation would bog down the intensive workload if all 32 threads were running simultaneously. Instead, `ThreadManager` queues all the threads, then runs a limited number of threads concurrently. The next thread in the queue is started once a running thread ends, until the queue is emptied.

A instantiating class of `ThreadManager` provides two arguments to the constructor. The first, the refresh rate interval, specifies the frequency at which the managed threads statuses are polled. The second, the maximum number of concurrent threads, limits the number of threads the manager can have executing simultane-

[58] `Thread Manager.h`

ously. The implementing class is permitted to provide the limit, which should be based on an understanding of the underlying resource loads and system hardware capabilities. For example, three derived classes use the number of CPUs[59] available to determine this limit.[60]

ThreadManager uses a template class ThreadTable[61] to store the information about queued and running threads. All threads require the creation of dynamically allocated data structures to store the execution information and parameters. ThreadTable assumes ownership of these dynamically allocated data, and deletes them when the associated thread is removed.

The thread management involves transitioning threads between events based on the thread status. For example, a thread process must have ended before its associated data can be deleted. Premature deletion of data will cause a program error and possible termination. The AdvanceFromThreadEvents()[62] is primarily responsible for managing changes in thread statuses.

#### 4.1.4.2 Progress Visualization

Most tasks that are parallelizable are lengthy process. Also, they are intrinsically capable of reporting progress to completion. Finally, the user experience is greatly enhanced by a visualization that implies work is being done.

This functionality is implemented by ProgressManagerCRTP, which extends both ThreadManager and WinProcDynamicSizing, in order to manage jobs and display progress. The CRTP form allows for specialization based on the task. Currently, three classes that provide concrete implementation: FileLoadingProgressManager, Analysis-

[59] central processing units

[60] The three classes that used knowledge of the system hardware are: Analysis Progress Manager.cpp:32, File Loading Progress Manager.cpp:28, and Simple Function Thread Controller.cpp:17

[61] Thread Table.h

[62] Thread Manager:135-208

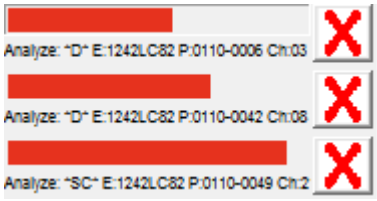



**Figure 4.7** – Progress Manager

ProgressManager, and ChainedProgressThreadManager.[63] Figure 4.7 shows an example of a progress window displaying the progress of three analysis threads.

ProgressManagerCRTP is designed such that any derived class needs only to define a four static variables. Two variables control the appearance of the progress bars by specifying the color and text descriptors, and the other two work with the Windows API to generate a unique window instance.

#### 4.1.4.3 Thread Parameters

Preana includes a suit of CRTP modules that simplify the creation and organization of the thread parameter structures. Each module is created for a specific task and stores the required data. There are modules currently for animation, sequence locked execution, progress reporting, self identification, and window support.

A base class, BasicThreadParameters,[64] provides the common data that all threads require. This, combined with the aforementioned modules, create the thread parameter classes: AnimateWindowThreadParameters,[65] ChainedProgressThreadParameters,[66] ManagerThreadParameters,[67] ProgressBarThreadParameters,[68] and SimpleThreadParameters.[69]

[63] Chained Progress Thread Manager.h
[64] Basic Thread Parameters.(h/cpp)
[65] Animate Window Thread Parameters.(h/cpp)
[66] Chained Progress Thread Parameters.(h/cpp)
[67] Manager Thread Parameters.(h/cpp)
[68] Progress Bar Thread Parameters.(h/cpp)
[69] Simple Thread Parameters.(h/cpp)

One of the required template parameters for ProgressManagerCRTP and ThreadManager is the thread parameters class type. As a result, a thread manager class implementation is usually created for each of the thread types. This is by design so that each manager can perform specialized tasks as needed.

### 4.1.5 Data Fitting

The Minuit2 minimization library is the core of Preana's analysis capabilities. Minuit2 provides the extensive suite of tools for fitting the waveforms contained in the test stand data. These fit parameters are used to generate the calibration measurements required to provide precise TPC nuclear data measurements.

Minuit2 is a C++ adaptation of the Fortran-based Minuit minimization routines. It is not a program or application unto itself, but rather source code that can be downloaded and integrated directly into a program.[70]

#### 4.1.5.1 Fitting

Minuit2 minimization requires, at a minimum, the implementation of two classes: FCNBase[71] and MnUserParameters.[72] Each has its own unique requirements and pure virtual methods. A single class in Preana, Fitting, uses multiple inheritance to implement both classes. FCNBase is inherited as public, while MnUserParameters is inherited as protected. This is because of the methods in the former require external visibility, while it is safer to hide the data and a most methods of the latter.

[70] The source is available from the project website: http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/

[71] FCNBase.h

[72] MnUserParameters.(h/cxx)

Fitting is designed to be applicable to all current and any future fitting needs in Preana. Although the current application fits only waveforms, other applications are possible. For example, it could be used to fit a cumulative distribution of fitting results. The only requirements are that it be provided with a handle to the function for fitting (refer to section 4.1.5.2), a vector of data to fit, and parameter seeds.[73] A companion vector of data uncertainties may also be provided, but is not required.

Methods for controlling the minimization process are also provided. These include limit setting of valid parameter ranges, fixing a parameter, and providing parameter uncertainties. The operator()[74] function provides the $\chi^2$ value calculation.

The actual fitting is started by calling Fit().[75] The fitting uses the MnMigrad[76] minimizer by default. During the fitting Minuit2 performs a search to find the parameter values that minimize the $\chi^2$ value provided by the Fitting class instance. Minuit2 also inherently performs its own error calculations on the parameter values as it converges on a solution.

When the fitting is completed all unnecessary data are erased to preserve memory.[77] The resulting fit parameters and calculated errors are stored locally and accessible via the getter functions. Finally, a $\chi^2$ value without parameter weighting is calculated and stored.

[73] Fitting.h:26-29

[74] Fitting.cpp:225-231

[75] Fitting.h:32-34, Fitting.cpp:77-120

[76] MnMigrad.h

[77] The FunctionMinimum class, which stores the minimization results, is very large. Each Waveform contains an instance of Fitting, which contains an instance of FunctionMinimum. These quickly add up and consume system memory during an analysis.

#### 4.1.5.2 Base Function Class

The base function class is AbstractFunction.[78] The number of parameters is fixed upon initialization, so a derived class must provide the information either as an integer or a vector of initial parameter values. The Fitting class depends on concrete implementations of AbstractFunction to perform the fitting.

The core of AbstractFunction are the pure virtual functions SetParameters()[79] and operator().[80] SetParameters() sets the function parameters. These are used in operator(), along with a position argument, to compute and return the evaluated function value.

Two more pure virtual function must be implemented by derived classes: GetParameterName()[81] and IsValueImportant().[82] The implementation of GetParameterName() takes a parameter number argument and returns the parameter name as a string. IsValueImportant() is used to determine whether a particular location or value is important to the physical representation of the function.

#### 4.1.5.3 Fitting Selection

The fitting is controlled by a variable of enumeration type FittingMethod.[83] These values correspond to combinations of the fitting functions and minimization approaches. Table 4.3 contains the current list of options. The qualifiers, if more than one are used, are delimited by a '_' character. The values are additive, e.g. Exp_FunctionFit_Fix contains the Exp, FunctionFit, and Fix qualifiers for a total value of 31.

[78] Abstract Function.(h/cpp)
[79] Abstract Function.h:34
[80] Abstract Function.h:33
[81] Abstract Function.h:27
[82] Abstract Function.h:31-32
[83] Fundamentals.h:188-207

**Table 4.3** – Fitting Methods

| Qualifier | Value | Meaning |
|---|---|---|
| ESTIMATED[⋆,†] | 1 | Use the first and mid-plateau datum to estimate the waveform properties |
| COARSE[†] | 2 | Scan the data for the transition, then average the data before and after to calculate the pedestal and plateau, respectively |
| EXP | 20 | Use an exponential function to fit the RC decay |
| LINEAR | 40 | Fit the RC decay with a simple linear function |
| ADAPTEDFIT[‡] | 0 | Fit each waveform with parameters seeded by the previously fitted waveform |
| FUNCTIONFIT | 10 | Independently fit each waveform |
| FIX | 1 | Fix the RC time constant so that it is not included in the minimization |

[⋆]Cannot be requested by the user; calculated only when first loading the data
[†]A test fit with this method cannot be used for generating a comparison test
[‡]Result of an attempt to optimize the minimization, no quantifiable effect has been found (see section 4.2.3)

The FittingMethod values are used internally by the data storage classes (see section 4.1.2) to determine which analysis approach to use. Specifically, the Waveform class requires a FittingMethod-type argument for its Fit()[84] function. Waveform then uses this value set up the fit configuration before using its instance of Fitting to perform the fitting.

#### 4.1.5.4 Implementation Example: Fermi Function Fit

Waveforms are best represented by a Fermi function with an added RC[85] decay component. This function fit implemented in the class WaveformFunction,[86] which inherits from AbstractFunction. The Fermi function is used in the form shown by eq. (4.1). Parameter $A$ represents the pedestal height, $B$ is the charge, $C$ is the onset

[84] Waveform.h:43
[85] resistor-capacitor
[86] Waveform Function.(h/cpp)

of the transition point, $D$ is the rise curvature, and $g(t)$ is the RC decay applied after the rise.

$$f(t) = \begin{cases} A + \dfrac{B}{\exp\left(\frac{C-t}{D}\right) + 1} & \text{when } C < t \\ A + \dfrac{B \times g(t)}{\exp\left(\frac{C-t}{D}\right) + 1} & \text{when } C \geq t \end{cases} \tag{4.1}$$

The RC decay is applied via the pure virtual function `GetDecayMultiplier()`.[87] Two forms are provided for the RC decay: the exact exponential form and an approximate linear form. Equation (4.2) is the form of exponential decay, and eq. (4.3) the form of linear decay. These are implemented in `WaveformFunctionExponential`[88] and `WaveformFunctionLinear`,[89] respectively. Parameter $C$ is the onset of the transition point as before, and $E$ is the decay constant.

$$g(t) = \exp\left(\frac{C-t}{E}\right) \tag{4.2}$$

$$g(t) = 1 - E \times (C - t) \tag{4.3}$$

`WaveformFunction` uses the `IsValueImportant()`[90] to indicate that the values surrounding the transition point are less significant. This is because of the presence of overshoot immediately after the transition, which otherwise will incorrectly increase the charge parameter.

### 4.1.6 Statistics Template Class

An important part of the analysis process is calculating the statistical properties of distributions, such as a collection of channel gains. The preferred solution is a

[87] `Waveform Function.h:53`
[88] `Waveform Function - Exponential.(h/cpp)`
[89] `Waveform Function - Linear.(h/cpp)`
[90] `Waveform Function.cpp:140-178`

generalized statistics class that can be used for all numeric types. `Statistics`,[91] a template class, was developed and included in Preana to satisfy this need.

Frequently, calculating the width of a distribution requires two passes over the data. The first pass calculates the mean of the distribution, and the second pass uses the mean to calculate the standard deviation or RMS.[92] `Statistics` includes an algorithm for calculating both the mean and standard deviation in real time, i.e. while data are being added. No second pass is needed.

The algorithm requires calculating three parameters after each added value, using the formulas shown in eq. (4.4). $n$ represents the total number of values, $x$ is the new value, and $\overline{x}$ is the running mean. These formulas have been thoroughly analyzed and found to be very robust, even when considering the effects of floating-point round-off errors. The zeroth values are all 0, and it is not necessary to preserve the old values.

$$\begin{aligned} \Delta_n &= x - \overline{x}_{n-1} \\ \overline{x}_n &= \overline{x}_{n-1} + \frac{\Delta_n}{n} \\ M_n &= M_{n-1} + \Delta_n \times (x - \overline{x}_n) \end{aligned} \tag{4.4}$$

The standard deviation $\sigma^2$ can be extracted at any time using the formula in eq. (4.5). At least two values are required for valid calculations, otherwise a NaN will be generated.

$$\sigma_n^2 = \frac{M_n}{n-1} \tag{4.5}$$

Even though it is not necessary to preserve old values for calculating the distribution properties, `Statistics` is able to store the values for later recall. The default action is to store the values. This behavior is set when constructing an instance

[91] `Statistics.h`

[92] residual mean square

of Statistics and toggled by calling the SetStoreData()[93] function. All previously recorded data are immediately discarded if the storage action is disabled.

An extension class, StatisticsWithExternalUncertainty,[94] provides for propagating uncertainty [6] when evaluating distributions composed of quantities with inherent uncertainties. A basic assumption of uncertainty propagation is that the measurement equation, which accounts for all sources of variability in the measurement, is of the form shown in eq. (4.6). Parameter $Y$ is the measurand, $f$ is the relational function, $N$ is the number of measurement quantities, and the $X_i$ are the variability-corrected measurement quantities. The uncorrected output estimate is shown in eq. (4.7). Parameter $y$ is the output estimate, and the $x_i$ are the directly measured quantities.

$$Y = f(X_1, X_2, \ldots, X_N) \tag{4.6}$$

$$y = f(x_1, x_2, \ldots, x_N) \tag{4.7}$$

Uncertainty propagation is evaluated using the law of propagation of uncertainty, shown in eq. (4.8). Parameter $u^2$ is the standard uncertainty, the subscript $t$ denotes a total value, and all other parameters are as defined previously.

$$u_t^2(y) = \sum_{i=1}^{N} \left( \frac{\partial f}{\partial x_i} \right)^2 u^2(x_i) + 2 \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} u(x_i, x_j) \tag{4.8}$$

The simplified combined standard uncertainty form, as shown in eq. (4.9), can be used if the input estimates $x_i$ are uncorrelated.

$$u_t^2(y) = \sum_{i=1}^{N} u^2(x_i) \tag{4.9}$$

StatisticsWithExternalUncertainty is a generalized implementation—the origination of the uncertainties and correlation between quantities are unknown—thus

[93] Statistics.h:39-40

[94] Statistics With External Uncertainty.h

it is assumed that the input estimates $x_i$ are uncorrelated and eq. (4.9) is used to propagate uncertainties. The standard deviation $\sigma^2$ is used for the standard uncertainty $u^2$ value.

Individual value uncertainties are retained if data storage is enabled. Further, templated functions are provided that allow for the bulk addition of data and uncertainties from another `Statistics`-based class. This provides a quick and useful mechanism for aggregating data, e.g. the analysis results from multiple waveforms, subphases, phases, and/or channels.

## 4.2 Examples of Fitting Capabilities

Preana has the ability to export the results and statistics of a fitting analysis for external validation and verification, which allows for the results of multiple analyses to be aggregated together and interpreted in a global manner.

An external package that is used and trusted by physicists world-wide, `ROOT` contains a powerful toolset for quickly performing operations on data. `ROOT` is used not for plotting results or calibration analysis, but rather as a validation tool for Preana's analysis capabilities. Internal validation tools could conceivably be built into Preana; however, this is not considered a necessary component of the framework's final design requirements for performing calibration measurements.

The functionality for collecting all the data for export is built into the data storage classes (see section 4.1.2). The export process appends everything if the file already exists, otherwise it creates the file. This allowed the data from multiple minimizations to be combined into one file for easier comparison.

The file is generated as a csv with one line for each waveform's analysis results. Aggregate results for each subphase, phase, and channel are also included. The

file is saved with .prf as the extension. The first line of the file contains the item headers, providing information about the order in which the values are presented. The names and interpretations are documented in Variable Documentation.txt, and the less-intuitive values are summarized in table 4.4.

The .prf file is converted into a ROOT-formatted TTree and saved into a .root file using the PRF_to_TTree.C script. The documentation file Instructions.txt describes how to perform the conversion operation. Information is also given on working with ROOT's TTree to generate plot data. Finally, PlotGenerator.C and PlotsVsVoltage.C are provided for generating a set of standard plots from the .root file.

Initially, the option of adding the ROOT API to Preana was considered. This would have provided the possibility of internally performing the validations described previously. However, to do so would typically require ROOT to be compiled on each computer running Preana. ROOT is a very large package. Experience shows that the pre-build configuration is not trivial, particularly on a Windows system. The two-step file output and conversion approach is the preferred method instead—especially since the use of ROOT is limited to the Preana validation scenarios, not the core components.

The exporting is queued using the ChainedProgressThreadManager. One thread is created for each channel, and execution occurs in sequential order. This is done to prevent a collision from multiple simultaneous write operations on the same file; the export format does not require that the data be ordered by increasing channel number. Also, the threading allows the GUI to remain responsive while the exporting is occurring.

**Table 4.4** – Data Export Item Specifications

| Item # | Name | Definition in Code | |
|---|---|---|---|
| | | Value | Interpretation |
| 1 | Category | Export Operations.h:19-26 (Categories) | |
| | | 0 | waveform |
| | | 1 | subphase |
| | | 2 | phase |
| | | 3 | channel |
| 2 | Source Type | Fundamentals.h:389-399 (TestTypes) | |
| | | 0 | single channel test |
| | | 1 | distribution test |
| 5 | Phase* | Fundamentals.h:284-324 (PhaseTypes) | |
| | | 0 | Unknown or undefined |
| | | 1 to 32 | channel number |
| | | 256 | bank AB |
| | | 512 | bank CD |
| | | 1024 | bank EF |
| | | 2048 | bank GH |
| 6 | Preamp Proximity† | Fundamentals.cpp:250-322 | |
| | | 255‡ | same side, channel # - 2 |
| | | 0 | pulsed channel |
| | | 1 | same side, channel # + 2 |
| | | 2 | opposite |
| | | 3 | kiddy-corner (edge case) |
| | | 4 | Unknown or undefined |
| 7 | Distribution Board Proximity§ | Fundamentals.cpp:173-248 | |
| | | 0 | pulsed bank |
| | | -3 to 3 | location offset of pulsed bank |
| 8 | Fitting Method | *see table 4.3* | |

*Provides the source of the primary pulse for the entity
†Represents the relative location of the nearest pulsed channel on the preamp card
‡Hexadecimal value of 0xFF, which is interpreted as -1 in a signed context
§Represents the relative location of the nearest pulsed bank on the distribution board

### 4.2.1 $\chi^2$ Distribution

$\chi^2$ distributions are used to understand the goodness of fit and are fundamental to anyone developing a specific calibration procedure. The $\chi^2$ data from all fits are properly stored, and the resulting distributions can be compared.

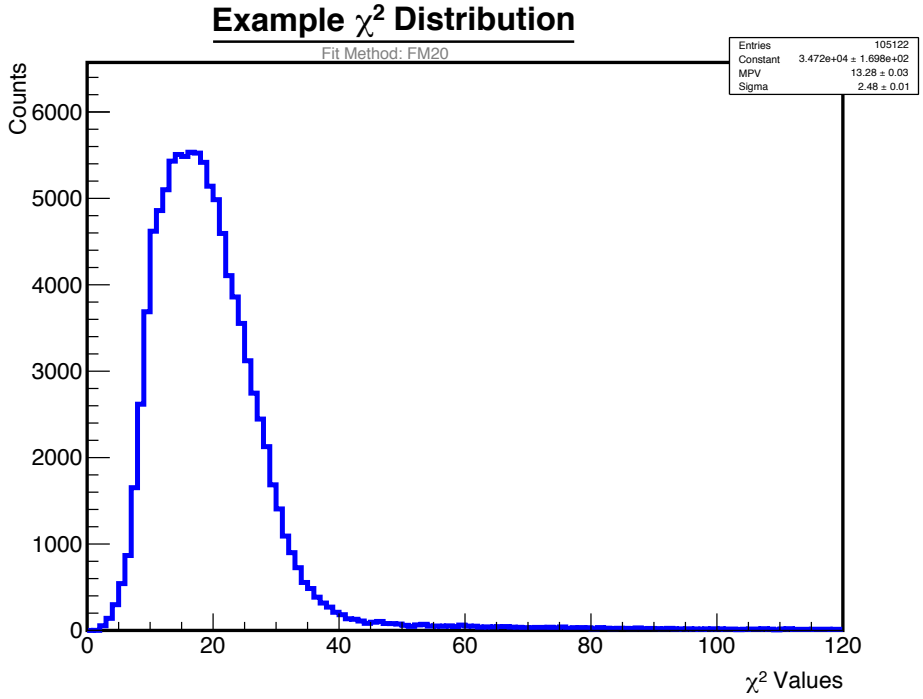
Two distributions from different fitting methods are shown in fig. 4.8. Figure 4.8b is the FIX equivalent of fig. 4.8a, although the FIX distribution has both a higher mean value and uncertainty. The reason for fixing the decay constant is shown fig. 4.9, where it is demonstrated that the minimization procedure infrequently discovers a local minimum that does not accurately describe the waveform.[95]

Fixing the RC constant forces the fit of the plateau to more closely match the data, providing a much better parametrization of the gain. The trade-off is that there are slight differences between the RC decay values, and the fixed value is only an estimate of the actual channels-specific values. However, the RC errors due to these differences is much less than the error contributed by the false minima, and the result is an overall improvement in the fit accuracy. This is a simple example of what a physicist might do in developing the correct calibration procedure.

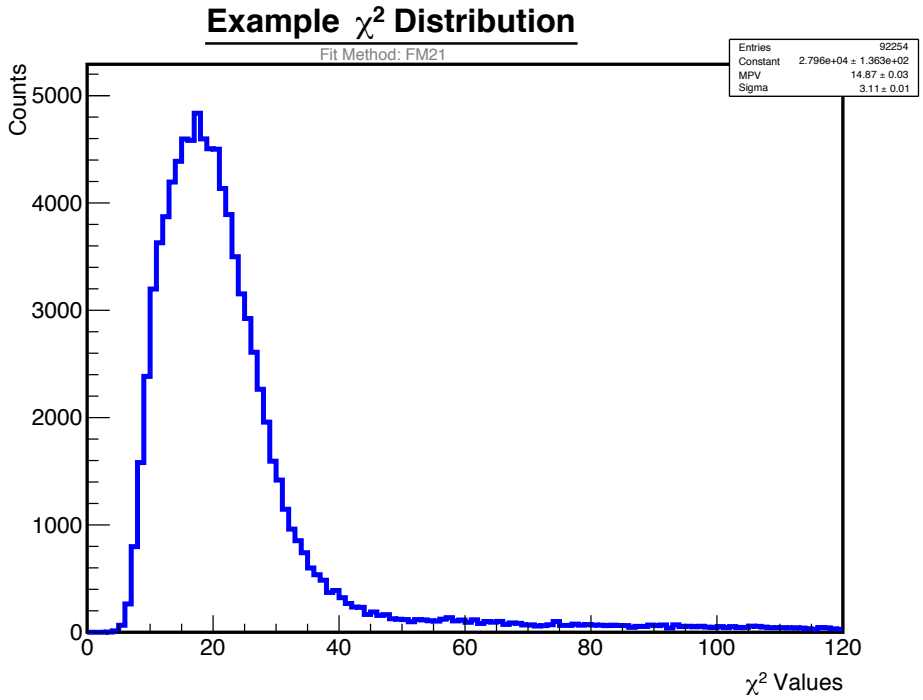
### 4.2.2 Gain Distribution

Gain distributions can be used to understand the accuracy of a fitting a specific function to a set waveforms. The linear performance of a channel can also appear in the width and/or skewdness of the gain distribution. The gain distributions shown herein contain aggregated subphase gain analysis data taken from one

[95] Differences between the fit and data are most easily identified in the <Preview Pane>, which plots the relative difference using colored bars extending from the center of the graph. See fig. 4.9 for an example.




**(a)** Typical $\chi^2$ Distribution




**(b)** $\chi^2$ Distribution with Fixed RC Constant

**Figure 4.8** – Minimization $\chi^2$ Distributions

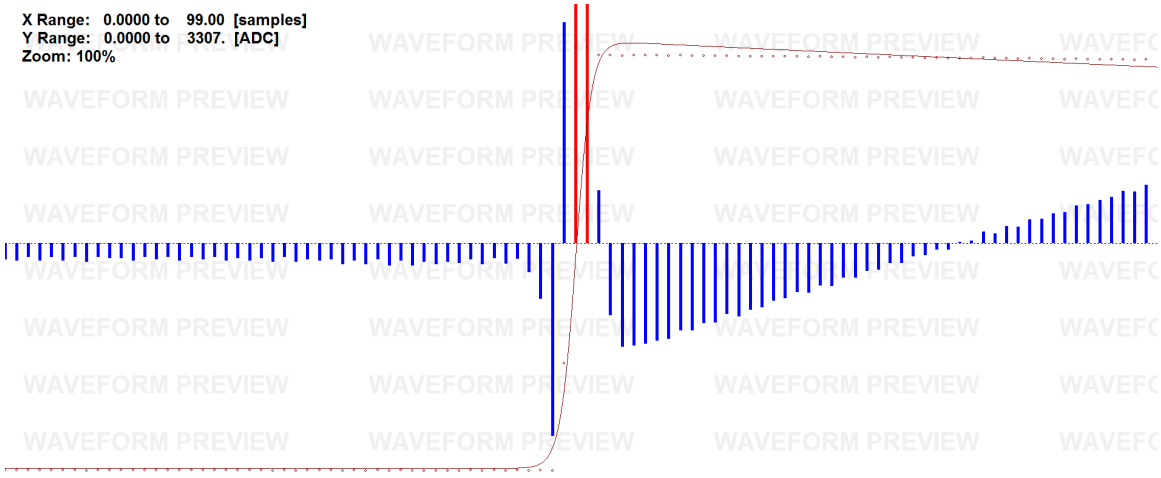


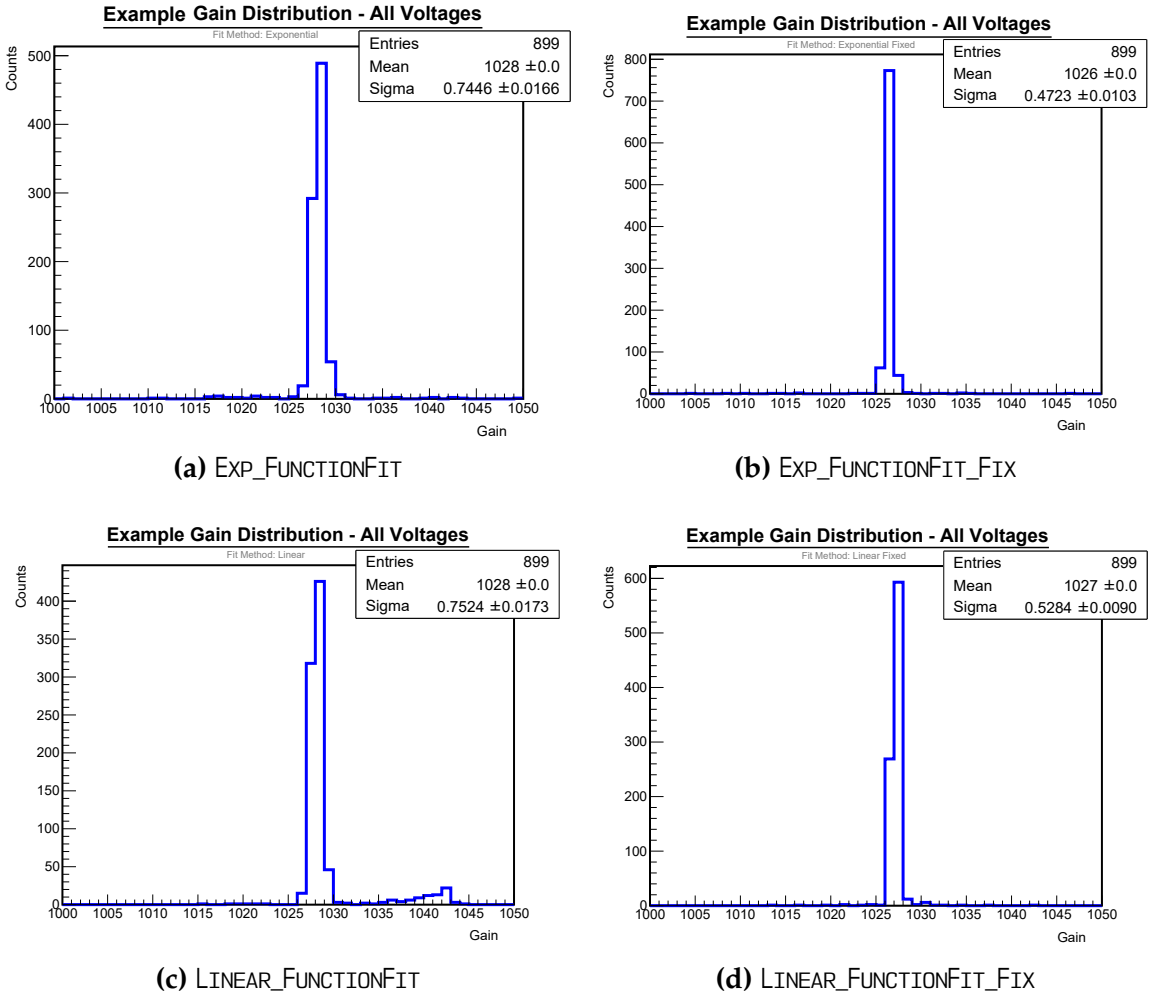

**Figure 4.9** – Local Minima Example

single channel test. Distributions for different fitting methods on the same data are shown.

The improved minimization effect of fixing the RC time constant is further demonstrated by investigating the gains directly. Figure 4.10 shows the results of different fitting methods on the same data set. The mean and sigma values are the parameters of a normal distribution fitted to the data.

The mean values of each distribution are within 0.2% of each other. These distributions globally verify the performance of the E/P card pairs; the cards are shown to have a very linear response over the tested dynamic range, based on the tightness and shape of the distributions. More in-depth analysis on the results of each subphase could be used if more detail on the actual performance is needed.

A marked improvement in the distribution is seen after fixing the RC time constant. This is contrasted most dynamically by EXP_FUNCTIONFIT (fig. 4.10a) with the highest uncertainty, while its fixed counterpart EXP_FUNCTIONFIT_FIX (fig. 4.10b) has the lowest uncertainty.




**(a)** Exp_FunctionFit **(b)** Exp_FunctionFit_Fix

**(c)** Linear_FunctionFit **(d)** Linear_FunctionFit_Fix

**Figure 4.10** – Gain Distribution Comparison of Different Fitting Methods

This improvement is expected since the exponential decay component increases the complexity of the system—thus the possibilities for locating local minima—with a resulting higher uncertainty. However, once the RC time constant is fixed then the exponential form most accurately represents the physical behavior of the test system.

This demonstrates that Preana is capable of performing robust analyses on data, which can be verified externally. Again, this is just an example of what might

be done in determining the correct calibration procedure to be implemented in Preana for production-level analysis.

### 4.2.3 Optimization Attempts

The duration of a full-test analysis ranges, on average, from 30 s to 3 min. Although Preana's performance is sufficient, it is always desirable to improve the overall performance as much as possible. Two approaches are implemented in pursuit of this goal: LINEAR- and ADAPTEDFIT-type fitting methods.

The performance gains from each approach are negligible. As such, only anecdotal information is provided. However, this information is useful when implementing the Preana analysis framework for production-level calibration analyses—the actual form of the fitting function in an AbstractFunction-based class has very minor impact on the overall speed of the Minuit2 fitting routines. Thus, developmental resources should ideally be spent on determining the correct calibration procedure rather than trying to improve efficiency.

#### 4.2.3.1 Linear-type Fitting Methods

The WaveformFunctionLinear class exists due to an attempt to decrease the minimization duration. This corresponds to the LINEAR-type fitting methods. Computationally, exponential functions are more complex to evaluate than addition and multiplication. In theory, replacing an exponential function with a linear function would leverage this computational characteristic to boost the speed.

No differences are currently observable between the LINEAR- and EXPONENTIAL-type fitting methods. It is assumed that this is due to significant improvements in modern computing, a marginal amount of time spent actually computing the

function during the minimization, and a decrease in the physical accuracy of a linear decay function in the plateau region.

#### 4.2.3.2 AdaptedFit-type Fitting Methods

The AdaptedFit exists due to an attempt to improve the performance of each individual minimization. The underlying concept is based on the assumption that all the waveforms in a subphase are identical. If so, the fitted parameters for each should likewise be identical. Thus, each minimization would be assisted by seeding the fit parameters with the results of the previous successful minimization.

Again, the performance improvements are negligible. For a typical FunctionFit analysis lasting a few minutes, the corresponding AdaptedFit offers no improvement. A comparison of the gain distributions and uncertainties associated with each method reveal that the accuracies are likewise comparable.

## 4.3 Cross Talk Module

Initial hardware design considerations attempted to minimize crosstalk between channels. Unfortunately, the distribution board itself introduces cross talk into the pulse distribution system. Thus, all distribution-type tests are subject to cross talk interference.

### 4.3.1 Identification of Cross Talk Origination

The pattern of cross talk from the distribution board is irregular, and doesn't exhibit an immediately apparent relationship between channels. For example, a primary pulse on channel 6 produces a strong response on both channels 4 and 5; however, only a primary pulse on channel 4 elicits a strong response on channel 6—

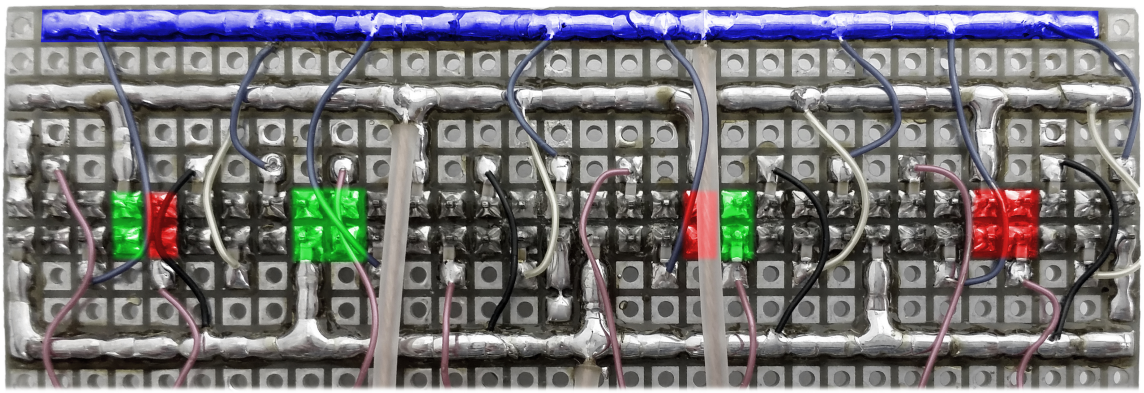


**Figure 4.11** – Correlation of Wiring Proximity to Cross Talk

no pulse from channel 5 has a significant effecton channel 6. Further, the cross talk does not follow any discernible pattern related to the position of the banks relative to each other on the distribution board. Data documenting this behavior, using ten preamp and two EtherDAQ cards to demonstrate repeatability, are collected into tables E.1 to E.9 of appendix E.

The actual cause of the cross talk is due to charge sharing that originates from the unshielded wiring on the bottom of the distribution board. The charge sharing occurs primarily where a wire carrying the primary pulse passes near another channel. The proximity of the wire to another channel has a direct correlation to the amount of charge measured by that channel. Figure 4.11, a modified version of fig. 2.5a, illustrates this correlation. The image enhancement represents a primary pulse sent to bank AB, highlighted by the blue bar. The colors of the charge sharing locations represent the magnitude of the resulting cross talk: red is large, and green is small. Charge sharing locations with negligible results are left unmarked.

This theory of the charge sharing locations was confirmed by modification of the distribution board. The shielding on the two bank bar feeds that extend over the jumper connections was extended, and the wiring was pulled back from the board as much as possible (see fig. 4.12). Data was recorded using the modified

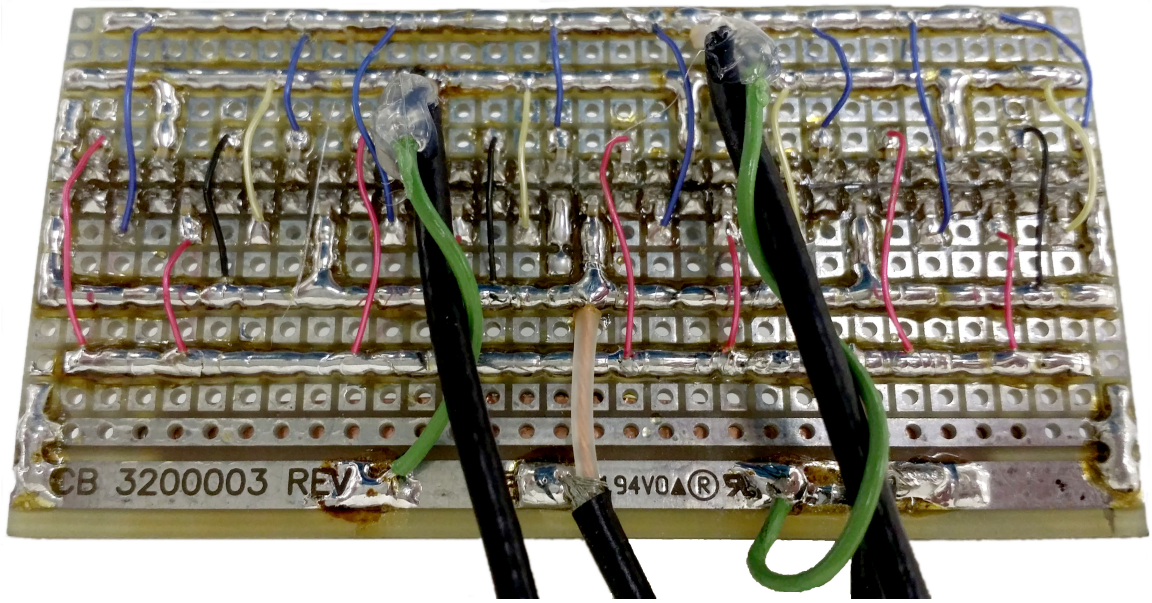



**Figure 4.12** – Modification of the Distribution Board

distribution board and is compiled into table E.10. The modifications caused a reduction of cross talk ADC value, slashed by as much as 90%, in the channels that exhibited significant cross talk.

### 4.3.2 Deconvolution

Cross talk deconvolution allows the distribution test data, which is significantly simpler and faster to collect, to be used for the calibration measurements. Development of a deconvolution matrix requires at least one data set each from the distribution and single channel test types. More than one of each type can be used, and is actually recommended since it will improve the statistical validity of the measurement. Additionally, not all E/P pairs have 100% of their channels operation, so more than one test of each from different E/P pairs helps to ensure a complete deconvolution matrix for all 32 channel.

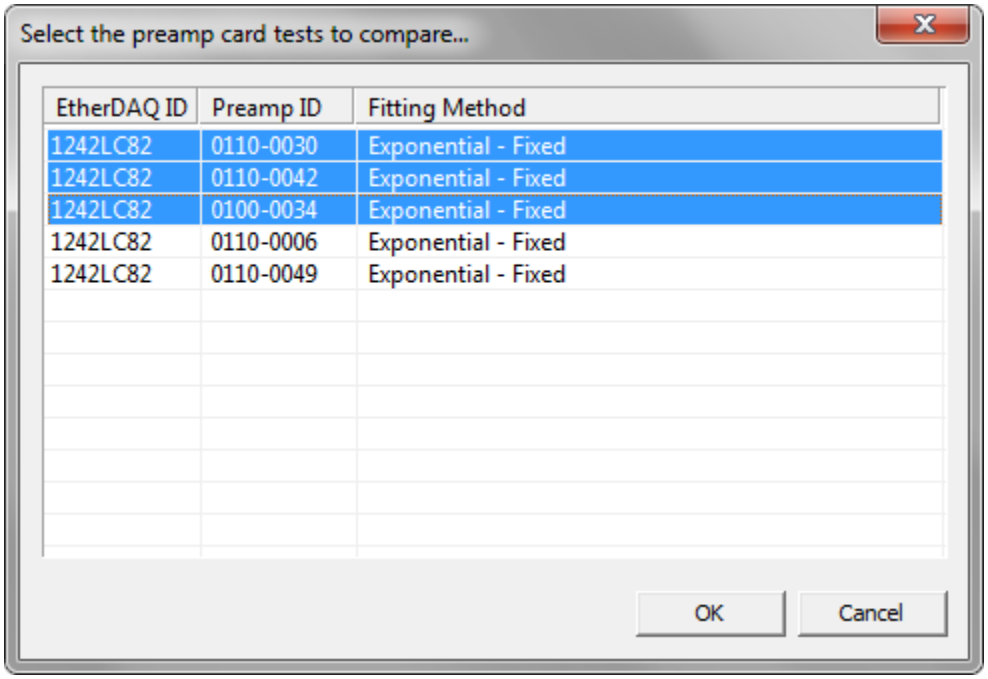



**Figure 4.13** – Test Comparison Selection

Validation of a deconvolution matrix requires two separate groups of data. The two groups must not share data from E/P pairs, i.e. no test data from an E/P may be used in both. The first group is used the generate the response matrices, and the second to verify the deconvolution. This ensures that the results are unbiased.

The menu command <Analysis> ▷ Compare Test Types... is used to generate a response matrix for a single E/P pair. A selection dialog, shown in fig. 4.13, displays the tests that can be used. Only E/P pairs that have both distribution and single channel test data loaded are displayed. Multiple tests can be selected for simultaneous analysis by pressing CTRL while clicking.

A deconvolution matrix can be generated, as soon as one or more response matrices are available, using the menu command <Breakout Board Response> ▷ Generate.... The menu command <Breakout Board Response> ▷ Apply & Predict... is then

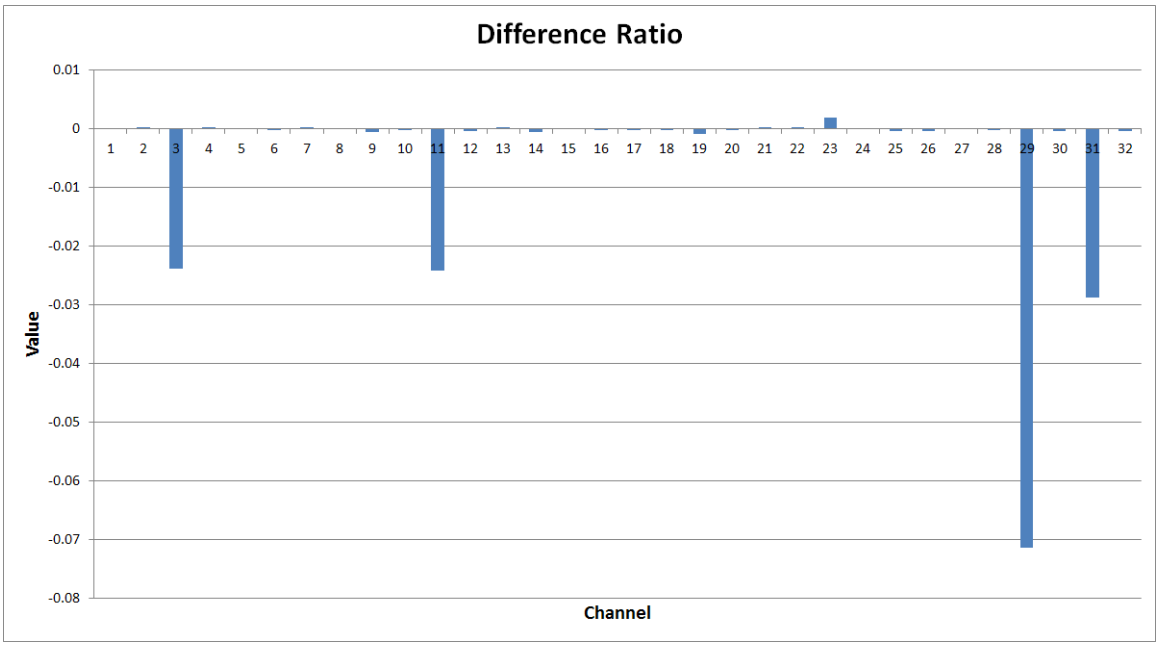
used to apply the deconvolution matrix to a test and predict the outcome of the resulting test type, whether distribution or single channel.

This method of deconvolution is limited by one important assumption: the cross talk and charge sharing must be linear. The accuracy is limited by the accuracy of the data. The deconvolution matrix itself is a simple channel-by-channel ratio of the gain between the distribution and the single channel test results. Conversion between forms is simple linear scaling operation.

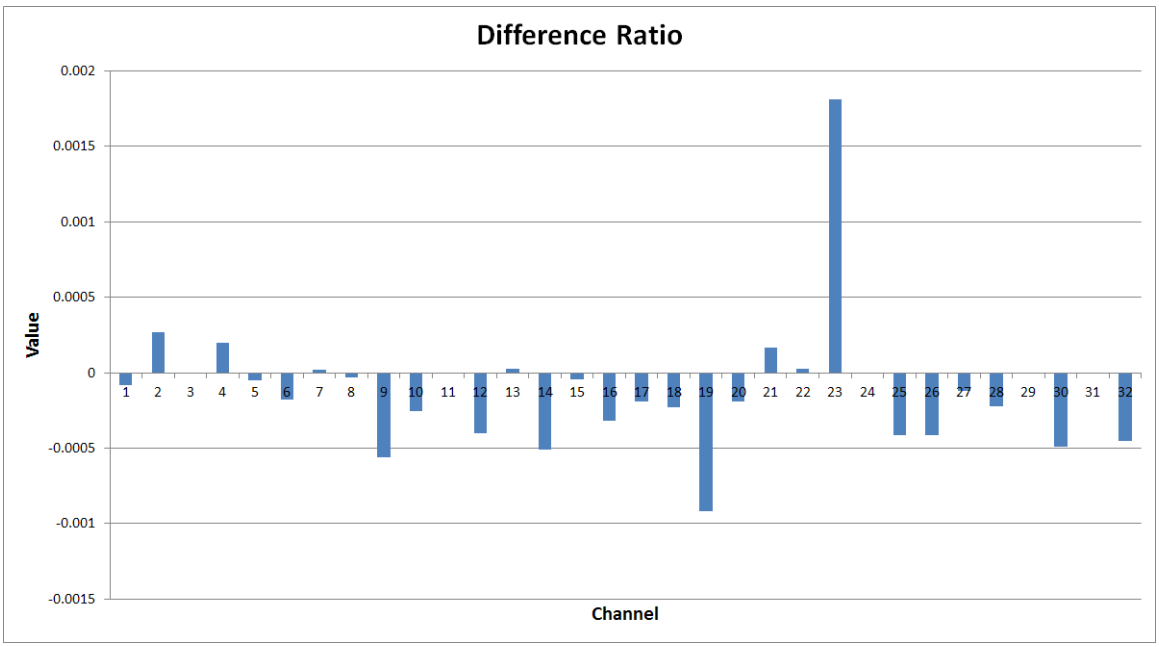
The results of an example deconvolution are contained in appendix F. The deconvolution matrix, shown in fig. F.1, is generated from four response matrices. It is then applied to the results of distribution test analysis, shown in fig. F.2. The predicted response results are compared against the actual single channels test data, respectively figs. F.3 and F.4.

Ratios of the differences between the predicted and actual values to the actual values are shown in fig. 4.14a. The ratios of the significantly different channels are removed in fig. 4.14b, in which the remaining ratios are less than 0.2%.

Something as simple as an accidental nudging of the wires with a finger alters the distribution board responses; the ratios corresponding to channels 3, 11, 29, and 31 are characteristic of changes in the wiring configuration. These results further establish that 1) the wiring on the distribution board as the source of cross talk in the distribution tests, and 2) that the cross talk levels are extremely sensitive to the wire proximities to the connectors for the breakout board.




**(a)** All Channels




**(b)** Channels 3, 11, 29, and 31 Removed

**Figure 4.14** – Ratios of Deconvolution Result Differences

# Chapter 5

# Conclusions

The framework is developed to support automated E/P card testing and routine calibration measurements.

A test stand is used to emulate the connections and charge signals an E/P pair would receive on the NIFFTE TPC, a fundamental aspect of performing calibration measurements. The pulse generating equipment exhibits extraordinary performance, which ensures the quality of the charge signals. A distribution board provides a method for simultaneously testing all 32 channels on a E/P pair.

EtherDaqGUI interfaces with the test stand hardware to interrogate E/P pairs and collect test data. It has been enhanced to be more user-friendly and operate pulse generation equipment to provide automated testing capabilities. The testing parameters are fully customizable by the user; default parameters comprehensively test an E/P pair in under 5 min, and the resulting data can be used to evaluate channel linearity, check for cross talk, and perform calibration measurements. An initial health evaluation is also performed, which reports channels behaving outside acceptable tolerances. All collected data is saved for later detailed analysis and calibration measurements in the analysis framework.

Preana, the analysis framework, is capable of performing complex and detailed analysis using the embedded Minuit2 library. The Minuit2 minimization routines are used to fit representative functions to the waveform data and extract the characteristic parameters, primarily the gain. Preana is designed to be extremely functional, as demonstrated by the example analyses performed for establishing its capabilities. A flexible design provides for future development as needs arise, and functionality for exporting analysis results enables external validation and verification using ROOT. Finally, Preana boasts a clean and intuitive user interface with functionality for browsing imported data, displaying waveforms with fit residuals, and plotting distributions of analysis result.

## 5.1 Summary

Calibration measurements of the E/P card pairs are central to attaining the sub-1% measurement uncertainty using the TPC (chapter 1). The E/P pairs are tested using the test stand hardware (chapter 2). The test stand equipment is controlled by EtherDaqGUI, which performs and initial health check on the E/P pairs and also records the test results for subsequent detailed analysis (chapter 3). Preana, a user-friendly and flexible analysis framework, is designed to support the detailed analysis requirements for generating calibration measurements (chapter 4).

## 5.2 Future Work

The following items are needed to satisfy the global objective of performing calibration measurements, which is beyond the scope of this work.

### 5.2.1 Distribution Board Resolution

In its current state the distribution board is still changeable, i.e. a accidental or intentional alteration of the wiring will change the cross talk behavior. The test stand will be incapable of performing any meaningful calibration measurements as long as this is the case. Two solutions are proposed to eliminate this Achilles heel:

1. replace the distribution board by another containing no movable/alterable components
2. stabilize the wiring configuration on the back
   (a) hot glue: provide for future alterations
   (b) epoxy: inhibit any future alterations

Developing a new distribution board will likely eliminate all sources of measurable cross talk in the system. Conversely, the cross talk deconvolution module in Preana has been shown to be perfectly capable of removing the effects of the distribution board. Both solutions are equally valid; however, any changes or modifications will need to be thoroughly checked. The level of cross talk present in the system will need to be re-evaluated against the pure single channel tests.

### 5.2.2 Baseline Response Matrix

Once the distribution board issue is resolved, it well necessary to generate the response matrix correlating the distribution board to the single channel board. It will be necessary to save this master response matrix to a file, which can then be quickly loaded when starting a new analysis session. The menu commands to export and import a response matrix are already available in anticipation of this need, but the functionality must still be implemented in the code.

### 5.2.3 Database Integration

The final step is to make the calibration measurements available through NIFFTE's central SQL[1] reconstruction database. Preana must be expanded to include SQL database communications. The calibration constant can then be properly formatted and uploaded as analyses are performed.

The NIFFTE fission TPC project will best accomplish its goal—generating nuclear data measurements with sub-1% uncertainties—through the application of these calibration measurements in data reconstruction efforts.

[1] structured query language

# References


[1] NIFFTE Collaboration. Annual Technical Report Fission Time Projection Chamber Project. Annual report, DOE Collaborative Research, October 2012.

[2] M. Heffner, D. Asner, R. Baker, J. Baker, S. Barrett, C. Brune, J. Bundgaard, E. Burgett, D. Carter, M. Cunningham, J. Deaven, D. Duke, U. Greife, S. Grimes, U. Hager, N. Hertel, T. Hill, D. Isenhower, K. Jewell, J. King, J. Klay, V. Kleinrath, N. Kornilov, R. Kudo, A. Laptev, M. Leonard, W. Loveland, T. Massey, C. McGrath, R. Meharchand, L. Montoya, N. Pickle, H. Qu, V. Riot, J. Ruz, S. Sangiorgio, B. Seilhan, S. Sharma, L. Snyder, S. Stave, G. Tatishvili, R. Thornton, F. Tovesson, D. Towell, R. Towell, S. Watson, B. Wendt, L. Wood, and L. Yao. A time projection chamber for high accuracy and precision fission cross-section measurements. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 759(0):50–64, 2014.

[3] M. Heffner, V. Riot, and L. Fabris. A Compact, Flexible, High Channel Count DAQ Built From Off-the-Shelf Components. *Nuclear Science, IEEE Transactions on*, 60(3):2196–2202, June 2013.

[4] B. Wendt and E. Burgett. Development of a Semi-Automated Testing and Calibration System for Custom-Built TPC DAQ Hardware. In *ANS Transactions*, June 2014.

[5] Stanford Research Systems. DG645 Digital Delay Generator User Manual. http://www.thinksrs.com/downloads/PDFs/Manuals/DG645m.pdf, September 2008.

[6] National Institute of Standards and Technology. Combining uncertainty components. http://physics.nist.gov/cuu/Uncertainty/combination.html, July 2015.

[7] Agilent Technologies. E3620A and E3630A Non-Programmable DC Power Supplies. http://cp.literature.agilent.com/litweb/pdf/5968-9727EN.pdf, January 2014.

[8] Teledyne LeCroy. ArbStudio Arbitrary Waveform Generators. http://cdn.teledynelecroy.com/files/pdf/lecroy_arbstudio_datasheet.pdf, October 2012.

[9] Stanford Research Systems. CG635 Synthesized Clock Generator. http://www.thinksrs.com/downloads/PDFs/Catalog/CG635c.pdf, March 2012.

[10] Stanford Research Systems. DG645 Digital Delay/Pulse Generator. http://www.thinksrs.com/downloads/PDFs/Catalog/DG645c.pdf, February 2013.

# Appendices

# Appendix A

# Formatting

A few different typefaces and formats are used throughout this thesis, each with a unique meaning.

This thesis contains hyperlinks; clicking on a hyperlink will navigate to, or open up, the linked resource when this document is viewed with a supporting document reader. Most document viewers indicate when the mouse is over a hyperlink by changing the cursor. If any hyperlink-type items are used in a special section, such as a title or caption, they will be typeset uniformly with the surrounding type. Nevertheless, the hyperlinks will still be active.

**Table A.1** – Formatting

| Typeface | Format | Example | Meaning |
|---|---|---|---|
| serif | normal | Lorem ipsum | The normal text throughout the document |
| serif | small caps | Lorem ipsum | An acronym, the first occurrence will include a definition in the footnotes; hyperlink |
| sans serif | normal | Lorem ipsum | A term found in the main glossary; hyperlink |
| monospace | normal | `Lorem ipsum` | A computer-related item, may be one of the following:<br>1. Name of a file, class, or function<br>2. A software library<br>3. An IP or MAC address<br>4. An internet URL (e.g. `www.isu.edu`); hyperlink |
| monospace | <text> | <`Lorem ipsum`> | Name of an interactive item within a program's GUI |
| monospace | small caps | `Lorem ipsum` | A macro, parameter, or enumerated value in the code |

It can be easier to represent some concept using substitutions. Any occurrence of substitution uses the conventions below.

**Table A.2** – Substitution Conventions

| **Appearance** | **Meaning** |
|---|---|
| ⟨ ⟩ | contains an element that requires substitution |
| \| \| | used inside ⟨ ⟩, contains information about that substitution |
| ... | a repeating pattern |
| F | a base-10 floating point decimal |
| H | a base-16 (hexadecimal) integer |
| N | a single numeral in the range 0–9 |
| S | a string of one or more characters |
| X | a base-10 integer |

# Appendix B

# Equipment Spec Sheets

**Table B.1** – Spec Sheet: Agilent E3620A and E3630A [7]

**(a)** 1 of 2

| | E3620A | E3630A |
|---|---|---|
| **Features** | Isolated dual outputs, 10 turn pots CV, CL | Tracking, CV, CL (±20 V) CV, CF (+6 V) |
| **Number of outputs** | 2 | 3 |
| **Number of Output Ranges** | 1 | 1 |
| **DC Output Rating** | 25 V, 1 A<br>25 V, 1 A | +6 V, 2.5 A<br>+20 V, 0.5 A<br>−20 V, 0.5 A |
| **Load and Line Regulation** | < 0.01% + 2 mV | |
| **Ripple and Noise** (20 Hz to 20 MHz) | | |
| Normal mode voltage | < 350 µVrms, < 1.5 mVpp | |
| Normal mode current | – | |
| Common mode current | < 1 µArms | |
| **Transient Response Time** | < 50 µsec following a change in output current from full load to half load for output to recover within: | |
| | 15 mV | |
| **Meter Accuracy** | ±0.5% + 2 counts at 25 °C ±5 °C | |
| **Meter Resolution** | | |
| Voltage | 10 mV (0–20 V), 100 mV (>20 V) | 10 mV |
| Current | 1 mA | 10 mA |
| **Isolation** | 240 Vdc | |

**(b)** 2 of 2

# Supplemental characteristics

| | E3620A | E3630A |
|---|---|---|
| **Control Mode** | CV/CL | CV/CL (±20 V)<br>CV/CF (+6 V) |
| **Temperature Coefficient per °C** | | |
| Voltage | < 0.02% + 1 mV | |
| Current | – | |
| **Output Drift** | | |
| Voltage | Less than 0.1% + 5 mV total drift for 8 hours after initial warm-up of 30 minutes | |
| Current | N/A | |
| **Temperature Range** | | |
| | Derate output current 50% between 40 °C to 55 °C | |
| **Cooling** | Convection cooling | |
| **Isolation** | ±240 Vdc | |
| **AC Input** | 100 Vac ±10%, 47–63 Hz (opt. 0E9)<br>115 Vac ±10%, 47–63 Hz (std)<br>230 Vac ±10%, 47–63 Hz (0E3) | |
| **Weight** | 5.0 kg (11.0 lbs) net, 6.25 kg (13.8 lbs) shipping | 3.8 kg (8.4 lbs) net, 5.1 kg (11.3 lbs) shipping |
| **Size** | 88.1 mm H x 212.3 mm W x 392.4 mm D<br>3.5″ H x 8.4″ W x 15.4″ D | 88.1 mm H x 212.3 mm W x 318.4 mm D<br>3.5″ H x 8.4″ W x 12.5″ D |
| **Warranty** | Three year for E3600 Series power supplies<br>Three months for standard shipped accessories | |
| **Product Regulation** | Certified to CSA 22.2 No. 231; conforms to IEC 1010-1; carries CE mark; complies with CISPR-11, Group 1, Class A,<br>KC South Korean EMC Mark, Canadian ICES/NMB-001, Australian C Tick Mark | |

**Table B.2** – Spec Sheet: Arbstudio Waveform Generator [8]

**(a)** 1 of 4

| | ArbStudio 1102 | ArbStudio 1102D | Arb Studio 1104 | ArbStudio 1104D |
|---|---|---|---|---|
| Channels | 2 | 2 | 4 | 4 |
| Digital Pattern Generator | NA | 18 Channels | NA | 36 Channels |
| Waveforms | Sine, Cosine, Triangle, Rectangle, Sawtooth, Ramp, Pulse, Sinc, Exponential, Sweep, DC, Noise, From File, Arbitrary | | | |
| **Waveform Characteristics** | | | | |
| Sine | | | | |
| Frequency Range (Arbitrary) | 2 µHz to 125 MHz | | | |
| Frequency Range @ Max Sample Rate (DDS) | 3.7 mHz to 110 MHz | | | |
| Amplitude Flatness (1 $V_{p-p}$, Typical) | | | | |
| DC to 110 MHz (DDS) | < ±0.1 dB | | | |
| DC to 125 MHz (Arbitrary) | < ±0.1 dB | | | |
| Harmonics Distortion (1 $V_{p-p}$, Typical) | | | | |
| ≤ 1 MHz | < -66 dBc | | | |
| 1 MHz to 5 MHz | < -63 dBc | | | |
| 5 MHz to 10 MHz | < -59 dBc | | | |
| 10 MHz to 25 MHz | < -53 dBc | | | |
| 25 MHz to 75 MHz | < -38 dBc | | | |
| 75 MHz to 110 MHz (DDS) | < -31 dBc | | | |
| 75 MHz to 125 MHz (Arbitrary) | < -28 dBc | | | |
| Non Harmonic Distortion (1 $V_{p-p}$, Typical) | | | | |
| ≤ 1 MHz to 10 MHz | < -71 dBc | | | |
| 10 MHz to 25 MHz | < -66 dBc | | | |
| 25 MHz to 75 MHz | < -53 dBc | | | |
| 75 MHz to 125 MHz (Arbitrary) | < -47 dBc | | | |
| 75 MHz to 100 MHz (DDS) | < -61 dBc | | | |
| 100 MHz to 110MHz (DDS) | < -30 dBc | | | |
| THD | | | | |
| (100 kHz, 1 $V_{p-p}$, Typical) | < 0.15% | | | |
| Phase Noise (20 MHz, 1 $V_{p-p}$, Typical) | | | | |
| 10 kHz Offset | -106 dBc / Hz | | | |
| 100 kHz Offset | -113 dBc / Hz | | | |
| 1 MHz Offset | -128 dBc / Hz | | | |
| Analog Bandwidth | | | | |
| Arbitrary / DDS | 125 MHz / 110 MHz | | | |
| Square Wave, Pulse (1 $V_{p-p}$) | | | | |
| Frequency Range | 2 µHz to 62.5 MHz | | | |
| Duty Cycle Range | 1% to 99% | | | |
| Rise / Fall Time (Typical) | < 3.5 ns | | | |
| Overshoot (Typical) | < 5.5% | | | |
| Random Jitter (rms, Typical) | < 20 ps | | | |
| Triangle / Ramp | | | | |
| Frequency Range | 2 µHz to 31.25 MHz | | | |
| Start Phase Range | 0 to 360° | | | |
| Sinc (Sin(x)/x) | | | | |
| Frequency Range | 2 µHz to 15.5 MHz | | | |
| Minimum Lobe Width | 8 ns | | | |

**(b)** 2 of 4

| | ArbStudio 1102 | ArbStudio 1102D | Arb Studio 1104 | ArbStudio 1104D |
|---|---|---|---|---|
| **Waveform Characteristics (cont'd)** | | | | |
| Waveform Sequencing | | | | |
| Waveforms | All, From File, Arbitrary | | | |
| Waveform Repetitions | 1 to (2^33 – 1) | | | |
| Start Source | Software, Internal, External | | | |
| No. of Waveforms | 1 to 511 | | | |
| **Common Characteristics** | | | | |
| Arbitrary | | | | |
| Sample Rate Real Time | 4 S/s to 250 MS/s | | | |
| Vertical Resolution | 16-bit | | | |
| Waveform Memory | 2 Mpts / Ch | | | |
| Minimum Waveform Length | 8 points | | | |
| Waveform Resolution | 2 points | | | |
| Noise Bandwidth (-3 dB Gaussian Noise), Typical | 100 MHz | | | |
| Run Modes | Single, Continuous, Stepped, Burst | | | |
| Direct Digital Synthesis (DDS) | | | | |
| Sample Rate Real Time | 125 MS/s to 250 MS/s | | | |
| Run Modes | Single, Continuous, Burst | | | |
| Carrier Waveform Memory | 2048 Samples / Ch | | | |
| Amplitude, 50 Ω Load (1 kHz) | 0 V to +12 $V_{p-p}$ | | | |
| Amplitude, Open Circuit | 0 V to +24 $V_{p-p}$ | | | |
| Amplitude Resolution | < 1 mV | | | |
| DC Accuracy, Open Circuit (±12 V Range) | ±0.25% of amplitude range (within ±10 °C of calibration temperature T=25 °C, Humidity ≤ 80%)<br>±0.3% of amplitude range (0 to 50 °C) | | | |
| DC Accuracy, 50 Ω Load (±6 V Range) | ±0.25% of amplitude range (within ±10 °C of calibration temperature T=25 °C, Humidity ≤ 80%)<br>±0.3% of amplitude range (0 to 50 °C) | | | |
| AC Accuracy, Open circuit (0 $V_{p-p}$ to +24 $V_{p-p}$ range, 1 kHz Sine Wave) | ±0.25% of amplitude range (within ±10 °C of calibration temperature T=25 °C, Humidity ≤ 80%)<br>±0.3% of amplitude range (0 to 50 °C) | | | |
| AC Accuracy, 50 Ω Load (0 $V_{p-p}$ to +12 $V_{p-p}$ range, 1 kHz Sine Wave) | ±0.25% of amplitude range (within ±10 °C of calibration temperature T=25 °C, Humidity ≤ 80%)<br>±0.3% of amplitude range (0 to 50 °C) | | | |
| Output Impedance | Selectable: 50 Ω, Low or High Impedance | | | |
| Short Circuit Protection | Signal outputs are robust against permanent shorts against floating ground | | | |
| Frequency accuracy | | | | |
| Stability | < ±5 ppm | | | |
| Aging | < ± 2 ppm / year | | | |
| Max Interpolated Sample Rate | 1 GS/s (4x interpolation) | | | |
| Interpolation Factors | 1x, 2x, 4x | | | |
| Sampling Frequency Resolution | 15 digits limited by 1 nHz | | | |
| Multi Channel Specifications | | | | |
| Sampling Rate Tuning | Programmable per channel couple (Ch 1-2) | | Programmable per channel couple (Ch 1-2, Ch 3-4) | |
| Skew Between Channels (at Common Sample Rate) | | | | |
| Average (Typical) | < 300 ps | | | |
| Standard Deviation (Typical) | < 35 ps | | | |
| Math | Sum, Difference, Multiply between the two channels (Ch 1-2) | | | |

| | ArbStudio 1102 | ArbStudio 1102D | Arb Studio 1104 | ArbStudio 1104D |
|---|---|---|---|---|
| **Modulation** | | | | |
| Amplitude Modulation | | | | |
| Modulation Type | | Arbitrary AM, ASK | | |
| Carrier Waveform | | All, From File, Arbitrary | | |
| Modulating Waveforms | | All, From File, Arbitrary | | |
| Modulating Source | | Internal | | |
| Modulating Waveform Sample Clock at Max. Sampling Rate | | 0.46 S/s to 125 MS/s | | |
| Memory Size | | 2047 entries | | |
| Phase / Frequency Modulation | | | | |
| Modulation Type | | Arbitrary FM/PM, FSK, PSK | | |
| Carrier Waveform | | All, From File, Arbitrary | | |
| Modulating Waveforms | | All, From File, Arbitrary | | |
| Modulating Source | | Internal | | |
| Carrier Frequency at Max. Sample Rate | | | | |
| Sine Wave | | 3.7mHz to 110 MHz | | |
| Square | | 3.7mHz to 62.5 MHz | | |
| Triangle / Ramp | | 3.7mHz to 31.25 MHz | | |
| Modulating Waveform Sample Clock at Max. Sample Rate | | From 119.2S/s to 125 MS/s (per sample programmable) | | |
| Memory Size | | 511 entries | | |
| Frequency Resolution at 125 MS/s Sample Rate | | 0.0019 Hz (FSK)<br>2.15E-5° (PSK) | | |
| Frequency Resolution at 250 MS/s Sample Rate | | 0.0037 Hz (FSK)<br>4.30E-5° (PSK) | | |
| Pulse Width Modulation | | | | |
| Carrier Waveform | | Pulse | | |
| Carrier Frequency | | 100 mHz to 20 MHz | | |
| Duty Cycle Modulating Waveform | | Sine, Triangle, Ramp, Noise, Manual | | |
| Duty Cycle Modulating Frequency | | 10 µHz to 6.67 MHz | | |
| Source | | Internal | | |
| Duty Cycle Deviation | | 0 % to 100 % of pulse period | | |
| Frequency Sweep | | | | |
| Carrier Waveform | | All, From File, Arbitrary | | |
| Sweep Type | | All waveforms | | |
| Sweep Direction | | Up or Down | | |
| Sweep Range at Max. Sample Rate | | | | |
| Sine Wave | | 3.7 mHz to 110 MHz | | |
| Square | | 3.7 mHz to 62.5 MHz | | |
| Triangle / Ramp | | 3.7 mHz to 31.25 MHz | | |
| Sweep Time at Max. Sample Rate | | 100 ns to 4.2 s | | |
| **Pattern Generator Characteristics** | | | | |
| Number of Channels | N/A | 18 | N/A | 18 / 36 |
| Vector Memory Depth | N/A | 1 Mpts / Ch (per Ch programmable direction) | N/A | 1 Mpts / Ch (per Ch programmable direction) |
| Acquisition Memory Depth | N/A | 2 Mpts / Ch | N/A | 2 Mpts / Ch |
| Update Frequency | N/A | 125 MS/s (per Ch programmable direction) | N/A | 125 MS/s (per Ch programmable direction) |
| Sampling Frequency | N/A | 250 MS/s | N/A | 250 MS/s |
| Direction Control | N/A | Per Ch programmable | N/A | Per Ch programmable |
| Output Voltage Level | N/A | 1.2 V to 3.6 V | N/A | 1.2 V to 3.6 V |
| Trigger Levels | N/A | 31 | N/A | 31 |
| Operating Modes | N/A | 18 Ch Digital or<br>2 Ch Analog | N/A | 36 Ch Digital or<br>4 Ch Analog<br>or 18 Ch Digital<br>plus 2 Ch Analog |

**(d)** 4 of 4

| | ArbStudio 1102 | ArbStudio 1102D | Arb Studio 1104 | ArbStudio 1104D |
|---|---|---|---|---|
| **Multi-Instrument Synchronization** | | | | |
| Max Number of Instruments | N/A | N/A | Up to 8 units with AS-SYNC Cable | |
| Synchronization Accuracy | N/A | N/A | < 300 ps | |
| **Auxiliary Inputs/Outputs** | | | | |
| Analog Outputs | | | | |
| Output Connector | Front panel BNC | | | |
| Output Impedance | 50 Ω, Low or High Impedance | | | |
| External Trigger Output | | | | |
| Output Connector | Front panel BNC | | | |
| Output Level | TTL compatible into > 1 KΩ | | | |
| Output Impedance | 50 Ω nominal | | | |
| External Trigger Input | | | | |
| Input Connector | Front panel BNC | | | |
| Frequency Range | DC to 125 MHz | | | |
| Threshold Level | VILmax = 0.8 V, VIHmin = 2 V | | | |
| Voltage Range | -0.5 V to 4 V | | | |
| Damage Level | VINmax < 6 V, VINmin > -2 V | | | |
| Slope | Rising Edge or Falling | | | |
| External Clock | | | | |
| Input Connector | Front panel BNC | | | |
| Frequency Range | 0 MHz to 125 MHz | | | |
| Min. Input Voltage Swing | ΔVINmin > 2 V | | | |
| Damage Level | VINmax < 5 V, VINmin > -5 V | | | |
| Digital I/O | | | | |
| Connector | 50 pin high density (1.27 mm) SCSI connector | | | |
| Connector count | 1 | | 2 | |
| **General Characteristics** | | | | |
| Power Supply Voltage Range | 100 ±10% to 240 ±10% VAC | | | |
| Power Consumption | 35 W max. | | | |
| Power Frequency Range | 50 / 60 Hz ± 5% | | | |
| PC Interface | USB 2.0 | | | |
| **Physical Characteristics** | | | | |
| External Dimensions (HWD) | 2.4" x 12.8" x 7.2" (62 x 326 x 182 mm) | | | |
| Weight | 2.8 lbs (1.3 kg) | | | |
| **Environmental Characteristics** | | | | |
| Temperature (Operating) | Main equipment: 0 to 50 °C, Power adapter: 0 to 40 °C | | | |
| Temperature (Non-Operating) | Main equipment: -40 to 71°C, Power adapter: -25 to 71°C | | | |
| Humidity (Operating) | 5% to 80% RH (non-condensing) at ≤ 30 °C, 50% max RH (non-condensing) at 40 °C | | | |
| Humidity (Non-Operating) | 5% to 95% max RH (non-condensing) | | | |
| Altitude (Operating) | Up to 3,048 m (10,000 ft) at ≤ 30°C | | | |
| Altitude (Non-Operating) | Up to 12,192 m (40,000 ft) | | | |
| **Minimum PC Requirements** | | | | |
| Operative System | Microsoft Windows® 2000 / XP SP2 / Vista / 7 32-bit Editions | | | |
| Processor | Intel® Pentium® III processor, or better | | | |
| Memory | 512 MB RAM | | | |
| Hard Disk | 150 MB available free space | | | |
| Display Resolution | 800 x 600 or better | | | |
| Connectivity | USB 2.0 or 1.1 | | | |

**Table B.3** – Spec Sheet: SRS CG635 Clock Generator [9]

### Frequency

| | |
|---|---|
| Range | DC, 1 µHz to 2.05 GHz |
| Resolution | 16 digits (f ≥ 10 kHz), 1 pHz (f < 10 kHz) |
| Accuracy | $\Delta f < \pm(2 \times 10^{-19} + \text{timebase error}) \times f$ |
| Settling time | <30 ms |

### Timebase *(+20 °C to +30 °C ambient)*

| | |
|---|---|
| Stability | <5 ppm (std. timebase)<br><0.01 ppm (Opt. 02 OCXO)<br><0.0001 ppm (Opt. 03 Rb timebase) |
| Aging | <5 ppm/yr. (std. timebase)<br><0.2 ppm/yr. (Opt. 02 OCXO)<br><0.0005 ppm/yr. (Opt. 03 Rb timebase) |
| External input | 10 MHz ± 10 ppm, sine >0.5 Vpp, 1 kΩ |
| Output | 10 MHz, 1.41 Vpp sine into 50 Ω |

### Phase Noise *(at 622.08 MHz)*

| | |
|---|---|
| 100 Hz offset | <–90 dBc/Hz |
| 1 kHz offset | <–100 dBc/Hz |
| 10 kHz offset | <–100 dBc/Hz |
| 100 kHz offset | <–110 dBc/Hz |

### Jitter and Wander

| | |
|---|---|
| Jitter (rms) | <1 ps (1 kHz to 5 MHz bandwidth) |
| Wander (p-p) | <20 ps (10 s persistence) |

### Time Modulation *(rear-panel input, 1 kΩ)*

| | |
|---|---|
| Sensitivity | 1 ns/V, ±5 % |
| Range | ±5 ns |
| Bandwidth | DC to greater than 10 kHz |

### Phase Setting

| | |
|---|---|
| Range | ±720° (max. step size ±360°) |
| Resolution | <14 ps |
| Slew time | <300 ms |

### Q and $\overline{Q}$ Outputs

| | |
|---|---|
| Outputs | Front-panel BNC connectors |
| Frequency range | DC to 2.05 GHz |
| High level | $-2.00\,V \le V_{HIGH} \le +5.00\,V$ |
| Amplitude | $200\,mV \le V_{AMPL} \le 1.00\,V$<br>$(V_{AMPL} \equiv V_{HIGH} - V_{LOW})$ |
| Level resolution | 10 mV |
| Level error | <1 % + 10 mV |
| Transition time | <100 ps (20 % to 80 %) |
| Symmetry | <100 ps departure from nominal 50 % |
| Source impedance | 50 Ω (±1 %) |
| Load impedance | 50 Ω to ground on both outputs |
| Pre-programmed levels | PECL, LVDS, +7 dBm, ECL |

### CMOS Output

| | |
|---|---|
| Output | Front-panel BNC |
| Frequency range | DC to 250 MHz |
| Low level | $-1.00\,V \le V_{LOW} \le +1.00\,V$ |
| Amplitude | $500\,mV \le V_{AMPL} \le 6.00\,V$<br>$(V_{AMPL} \equiv V_{HIGH} - V_{LOW})$ |
| Level resolution | 10 mV |
| Level error | <2 % of $V_{AMPL}$ + 20 mV |
| Transition time | <1 ns (20 % to 80 %) |
| Symmetry | <500 ps departure from nominal 50 % |
| Source impedance | 50 Ω (reverse terminates cable reflection) |
| Load impedance | Unterminated 50 Ω cable of any length |
| Attenuation (50 Ω load) | Output levels are divided by 2 |
| Pre-programmed levels | 1.2 V, 1.8 V, 2.5 V, 3.3 V or 5.0 V |

### RS-485 Output

| | |
|---|---|
| Output | Rear-panel RJ-45 |
| Frequency range | DC to 105 MHz |
| Transition time | <800 ps (20 % to 80 %) |
| Clock output | Pin 7 and pin 8 drive twisted pair |
| Source impedance | 100 Ω between pin 7 and pin 8 |
| Load impedance | 100 Ω between pin 7 and pin 8 |
| Logic levels | $V_{LOW} = +0.8\,V$, $V_{HIGH} = +2.5\,V$ |
| Recommended cable | Straight-through Category-6 |

### LVDS Output *(EIA/TIA-644)*

| | |
|---|---|
| Output | Rear-panel RJ-45 |
| Frequency range | DC to 2.05 GHz |
| Transition time | <100 ps (20 % to 80 %) |
| Clock output | Pin 1 and pin 2 to drive twisted pair |
| Source impedance | 100 Ω between pin 1 and pin 2 |
| Load impedance | 100 Ω between pin 1 and pin 2 |
| Logic levels | $V_{LOW} = +0.96\,V$, $V_{HIGH} = +1.34\,V$ |
| Recommended cable | Straight-through Category-6 |

### PRBS (Opt. 01) *(EIA/TIA-644)*

| | |
|---|---|
| Outputs | PRBS, –PRBS, CLK and –CLK |
| Frequency range | DC to 1.55 GHz |
| Level | LVDS on rear-panel SMA jacks |
| PRBS generator | $x^7 + x^6 + 1$ for a length of $2^7 - 1$ bits |
| Transition time | <100 ps (20 % to 80 %) |
| Load impedance | 50 Ω to ground on all outputs |

### General

| | |
|---|---|
| Computer interfaces | GPIB and RS-232 std. All functions can be controlled through either interface. |
| Non-volatile memory | Ten sets of instrument configurations can be stored and recalled. |
| Power | 90 to 264 VAC, 47 to 63 Hz, 50 W |
| Dimensions, weight | 8.5" × 3.5" × 13" (WHD), 9 lbs. |
| Warranty | One year parts and labor on defects in materials and workmanship |

**Table B.4** – Spec Sheet: SRS DG645 Pulse Generator [10]

### Delays

| | |
|---|---|
| Channels | 4 independent pulses controlled in position and width. 8 delay channels available as an option (see *Output Options*). |
| Range | 0 to 2000 s |
| Resolution | 5 ps |
| Accuracy | 1 ns + (timebase error × delay) |
| Jitter (rms) | |
| Ext. trig. to any output | 25 ps + (timebase jitter × delay) |
| $T_0$ to any output | 15 ps + (timebase jitter × delay) |
| Trigger delay | 85 ns (ext. trig. to $T_0$ output) |

### Timebases

| Model # | Type | Jitter *(s/s)* | Stability *(20 to 30 °C)* | Aging *(ppm/yr)* |
|---|---|---|---|---|
| Std. | crystal | $10^{-8}$ | $2 \times 10^{-6}$ | 5 |
| Opt. 4 | OCXO | $10^{-11}$ | $1 \times 10^{-9}$ | 0.2 |
| Opt. 5 | Rb | $10^{-11}$ | $1 \times 10^{-10}$ | 0.0005 |

| | |
|---|---|
| External input | 10 MHz ± 10 ppm, sine >0.5 Vpp, 1 kΩ impedance |
| Output | 10 MHz, 2 Vpp sine into 50 Ω |

### External Trigger

| | |
|---|---|
| Rate | DC to 1/(100 ns + longest delay) (maximum of 10 MHz) |
| Threshold | ±3.50 VDC |
| Slope | Trigger on rising or falling edge |
| Impedance | 1 MΩ + 15 pF |

### Internal Rate Generator

| | |
|---|---|
| Trigger modes | Continuous, line or single shot |
| Rate | 100 µHz to 10 MHz |
| Resolution | 1 µHz |
| Accuracy | Same as timebase |
| Jitter (rms) | <25 ps (10 MHz/N trigger rate)<br><100 ps (other trigger rates) |

### Burst Generator

| | |
|---|---|
| Trigger to first $T_0$ | |
| Range | 0 to 2000 s |
| Resolution | 5 ps |
| Period between pulses | |
| Range | 100 ns to 42.9 s |
| Resolution | 10 ns |
| Delay cycles per burst | 1 to $2^{32} - 1$ |

### Outputs ($T_0$, AB, CD, EF, and GH)

| | |
|---|---|
| Source impedance | 50 Ω |
| Transition time | <2 ns |
| Overshoot | <100 mV + 10 % of pulse amplitude |
| Offset | ±2 V |
| Amplitude | 0.5 to 5.0 V (level + offset <6.0 V) |
| Accuracy | 100 mV + 5 % of pulse amplitude |

### General

| | |
|---|---|
| Computer interfaces | GPIB (IEEE-488.2), RS-232, and Ethernet. All instrument functions can be controlled through the interfaces. |
| Non-volatile memory | Nine sets of instrument configurations can be stored and recalled. |
| Power | <100 W, 90 to 264 VAC, 47 Hz to 63 Hz |
| Dimensions | 8.5" × 3.5" × 13" (WHD) |
| Weight | 9 lbs. |
| Warranty | One year parts and labor on defects in materials & workmanship |

## Output Options

### Option 01 (8 Delay Outputs on Rear Panel)

| | |
|---|---|
| Outputs (BNC) | $T_0$, A, B, C, D, E, F, G and H |
| Source impedance | 50 Ω |
| Transition time | <1 ns |
| Overshoot | <100 mV |
| Level | +5 V CMOS logic |
| Pulse characteristics | |
| Rising edge | At programmed delay |
| Falling edge | 25 ns after longest delay |

### Option 02 (8 High-Voltage Delay Outputs on Rear Panel)

| | |
|---|---|
| Outputs (BNC) | $T_0$, A, B, C, D, E, F, G and H |
| Source impedance | 50 Ω |
| Transition time | <5 ns |
| Levels | 0 to 30 V into high impedance<br>0 to 15 V into 50 Ω<br>(amplitude decreases by 1 %/kHz) |
| Pulse Characteristics | |
| Rising Edge | At programmed delay |
| Falling Edge | 100 ns after the rising edge |

### Option 03 (Combinatorial Outputs on Rear Panel)

| | |
|---|---|
| Outputs (BNC) | $T_0$, AB, CD, EF, GH, (AB + CD), (EF + GH), (AB + CD + EF), (AB + CD + EF + GH) |
| Source impedance | 50 Ω |
| Transition time | <1 ns |
| Overshoot | <100 mV + 10 % of pulse amplitude |
| Pulse characteristics | |
| $T_0$, AB, CD, EF, GH | Logic high for time between delays |
| (AB + CD), (EF + GH) | Two pulses created by the logic OR of the given channels |
| (AB + CD + EF) | Three pulses created by the logic OR of the given channels |
| (AB + CD + EF + GH) | Four pulses created by the logic OR of the given channels |

### Option SRD1 (Fast Rise Time Module)

| | |
|---|---|
| Rise time | <100 ps |
| Fall time | <3 ns |
| Offset | 0.8 V to 1.1 V |
| Amplitude | 0.5 V to 5.0 V |
| Load | 50 Ω |

# Appendix C

# Manual Analysis Mode Options

**Table C.1** – Manual Analysis Mode Options

| Mode | Display Type | Description |
|---|---|---|
| Waveform Comparison | Plot | Allows for the direct visual comparison of any waveforms generated from the same pulse set |
| Waveform Statistics | Report | A summary of the waveform analysis results including data uncertainties |
| Linearity | Plot | Shows the gain as a function of the pulse voltage, overlays the best-fit simple linear regression line, and prints the function and R value generated from the fitting |
| Gain Histogram | Histogram | Displays the average gain for each selected channel |
| Y-Intercept Histogram | Histogram | Displays the y-intercept value from the simple linear regression analysis (can be used to check for correlations between groupings) |
| Cross Talk Higher | Plot | Shows the cross talk between a channel and the higher-numbered neighbor channel on the same side of the preamp |
| Cross Talk Lower | Plot | Shows the cross talk between a channel and the lower-numbered neighbor channel on the same side of the preamp |
| Cross Talk Over | Plot | Shows the cross talk between a channel and the channel physically located on the opposite side of the preamp card |
| Cross Talk on a Channel | Plot | Shows the cross talk the selected channel exhibits by pulses on another bank |
| Saturation Points$^{\star}$ | Histogram | Shows the saturation point, if any, based on the detection algorithm of the simple linear regression analysis |
| Timestamp Consistency by Single Pulse | Report | Prints the consistency of the timestamps collected from a single pulse set |
| Timestamp Consistency by Phase | Report | Prints the average consistency of all timestamps collected from all the pulse sets during a complete phase |
| Pedestals | Histogram | Displays the average pedestal values |
| Pedestal Widths | Histogram | Displays the uncertainty associated with the average pedestal values |

*continued . . .*

*. . . continued*

| Mode | Display Type | Description |
|---|---|---|
| Calculated Pedestal Widths | Histogram | Displays the corrected pedestal uncertainties based on system characterization |
| Pedestals vs. Test | Plot | Shows the trend of the pedestal values as a function of the test and phases |
| Slope Comparison | Histogram | Displays the slope values from the simple linear regression analysis |

*Not implemented, i.e. doesn't display anything, due to a decision to keep the pulse voltages within the preamp and ADC dynamic range

# Appendix D

# Source Code Information

This document references the source in committed revision 3721 of the SVN repository located at `nuclear.calpoly.edu/home/niffte/SVN/trunk`

**Table D.1** – Notable EtherDaqGUI Revisions

| Revision | Significance |
|---|---|
| **1575** | First commit |
| **1622** | Software-based waveform simulation |
| **1674** | Waveform viewer |
| **1721** | Image saving |
| **1781** | Major bug fixes |
| **2013** | Ability to disconnect Ethernet interface |
| **2252** | ArbStudio interface |
| **2262** | Waveform customizer |
| **3721** | Current version |

**Table D.2** – Notable Preana Revisions

| Revision | Significance |
|---|---|
| **2166** | First commit |
| **2176** | Interface windows singleton for Windows API |
| **2210** | `Minuit2` and waveform fitting |
| **2578** | Thread manager template |
| **2614** | Data browser toolbar |
| **2681** | Event-based threaded framework |
| **2930** | Histogram class |
| **2975** | Test and analysis multiselect capability |
| **3098** | Dynamic font sizing |
| **3163** | Image saving |
| **3721** | Current version |

# Appendix E

# Cross Talk Data

The data contained herein is extracted from the .ana files generated by EtherDaq-GUI. Thus, the parameter uncertainties are inherently much larger than those produced by a more robust analysis in Preana.

**Table E.1** – Cross Talk Data: Bank AB, EtherDAQ ID BEAC02

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 2 634.615 | 3 246.779 | 3 167.003 | 3 180.517 | 3 202.968 | 3 504.023 | 159.942 | 3 359.884 | 3 577.953 | 3 313.133 |
| **2** | 10.093 | 10.876 | 9.164 | 10.360 | 9.716 | 9.640 | 16.998 | 10.044 | 10.140 | 9.642 |
| **3** | 8.220 | 8.379 | 7.910 | 7.713 | 8.572 | 7.819 | 35.258 | 8.219 | 7.272 | 7.436 |
| **4** | 102.974 | 102.141 | 99.579 | 103.590 | 99.339 | 102.115 | 103.424 | 101.760 | 102.672 | 100.105 |
| **5** | 166.475 | 167.573 | 164.524 | 167.531 | 162.440 | 162.924 | 165.625 | 165.995 | 160.857 | 163.788 |
| **6** | 3 205.246 | 100.092 | 3 165.172 | 3 177.021 | 3 250.730 | 3 259.573 | 3 153.961 | 3 127.715 | 3 161.655 | 3 176.045 |
| **7** | 10.303 | 10.507 | 9.232 | 9.641 | 9.723 | 10.764 | 4.309 | 9.602 | 9.373 | 10.504 |
| **8** | 11.835 | 12.276 | 11.025 | 12.197 | 12.331 | 12.838 | 12.575 | 12.198 | 12.387 | 11.102 |
| **9** | 3 162.830 | 3 191.463 | 3 371.791 | 3 170.876 | 3 183.353 | 3 105.175 | 3 214.566 | 3 050.363 | 3 150.405 | 3 231.862 |
| **10** | 10.398 | 9.989 | 10.631 | 10.163 | 10.311 | 10.538 | 10.348 | 9.891 | 11.625 | 10.055 |
| **11** | 13.068 | 12.581 | 12.267 | 12.557 | 11.657 | 12.989 | 11.918 | 12.625 | 0.000 | 11.823 |
| **12** | 44.246 | 44.552 | 49.402 | 45.774 | 43.600 | 44.197 | 44.798 | 46.056 | 44.555 | 44.485 |
| **13** | 105.440 | 109.404 | 110.138 | 106.940 | 116.185 | 93.856 | 116.195 | 109.503 | 103.650 | 105.038 |
| **14** | 3 268.438 | 3 317.357 | 3 194.293 | 3 300.715 | 3 166.217 | 3 454.954 | 3 572.378 | 3 482.743 | 3 236.183 | 3 171.912 |
| **15** | 15.258 | 15.942 | 14.744 | 19.607 | 16.017 | 5.982 | 15.666 | 15.945 | 16.774 | 16.535 |
| **16** | 9.739 | 9.229 | 9.253 | 8.727 | 9.249 | 10.354 | 8.622 | 8.896 | 10.603 | 8.970 |
| **17** | 3 257.520 | 3 221.677 | 3 129.174 | 0.077 | 3 156.860 | 3 250.374 | 3 271.430 | 3 221.458 | 3 030.602 | 3 169.709 |
| **18** | 7.111 | 6.963 | 6.826 | 6.797 | 6.881 | 7.236 | 6.845 | 6.791 | 6.830 | 6.573 |
| **19** | 2.617 | 2.023 | 2.825 | 5.975 | 2.433 | 2.258 | 1.853 | 2.524 | 2.385 | 1.372 |
| **20** | 8.867 | 8.684 | 7.886 | 7.816 | 8.404 | 13.429 | 7.870 | 8.477 | 8.441 | 8.529 |
| **21** | 0.191 | 2.964 | 5.298 | 3.675 | 3.375 | 0.748 | 0.105 | 2.947 | 2.928 | 2.830 |
| **22** | 3 417.741 | 3 220.346 | 3 213.666 | 3 175.147 | 3 062.709 | 2 478.502 | 3 159.598 | 3 259.005 | 3 185.102 | 3 211.566 |
| **23** | 3.481 | 40.233 | 84.011 | 45.585 | 40.539 | 3.581 | 41.026 | 39.747 | 40.242 | 39.727 |
| **24** | 25.079 | 24.394 | 30.654 | 1.386 | 24.489 | 31.833 | 23.852 | 24.933 | 24.625 | 24.535 |
| **25** | 3 175.019 | 3 086.971 | 0.019 | 3 115.183 | 3 363.304 | 3 178.590 | 3 223.833 | 3 151.377 | 3 192.682 | 3 111.058 |
| **26** | 9.250 | 9.104 | 19.404 | 9.508 | 9.720 | 9.418 | 10.148 | 9.864 | 10.297 | 9.023 |
| **27** | 9.083 | 8.940 | 58.572 | 9.052 | 9.190 | 9.241 | 9.760 | 9.111 | 9.396 | 9.804 |
| **28** | 135.446 | 135.862 | 138.723 | 135.105 | 0.228 | 139.492 | 135.268 | 134.168 | 140.122 | 136.908 |
| **29** | 39.639 | 39.706 | −0.141 | 37.235 | 39.102 | 37.644 | 11.102 | 39.436 | 39.680 | 39.622 |
| **30** | 3 157.945 | 2 941.908 | 2 862.659 | 2 938.223 | 2 936.102 | 2 977.334 | 2 977.802 | 2 970.923 | 2 937.256 | 2 872.906 |
| **31** | 3.046 | 2.794 | 4.582 | 3.011 | 3.352 | −0.022 | 0.017 | 2.972 | 3.016 | 2.667 |
| **32** | 8.275 | 8.713 | 7.867 | 1.721 | 8.283 | 8.771 | 9.502 | 9.279 | 8.620 | 9.129 |

**Table E.2** – Cross Talk Data: Bank AB, EtherDAQ ID 1242LC82

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 2 657.898 | 3 253.502 | 3 172.892 | 3 185.829 | 3 210.834 | 3 510.929 | 152.885 | 3 366.517 | 3 584.643 | 3 319.244 |
| **2** | 9.805 | 11.252 | 9.415 | 10.486 | 10.139 | 9.611 | 17.212 | 10.437 | 10.167 | 9.944 |
| **3** | 8.351 | 8.509 | 8.171 | 7.519 | 8.984 | 7.813 | 35.500 | 8.167 | 7.323 | 7.380 |
| **4** | 103.588 | 102.039 | 99.645 | 102.980 | 99.410 | 102.453 | 103.917 | 101.863 | 102.977 | 100.295 |
| **5** | 166.644 | 167.611 | 163.712 | 166.424 | 162.621 | 163.013 | 165.868 | 166.248 | 160.893 | 163.767 |
| **6** | 3 208.442 | 838.754 | 3 169.975 | 3 181.391 | 3 256.024 | 3 263.936 | 3 157.643 | 3 130.917 | 3 165.484 | 3 179.529 |
| **7** | 10.488 | 10.657 | 9.671 | 9.641 | 10.046 | 10.747 | 4.266 | 10.005 | 9.396 | 10.499 |
| **8** | 11.956 | 12.775 | 11.264 | 12.276 | 12.573 | 12.878 | 12.653 | 12.654 | 12.537 | 11.254 |
| **9** | 3 145.300 | 3 174.934 | 3 354.272 | 3 153.561 | 3 166.883 | 3 088.080 | 3 196.655 | 3 033.658 | 3 132.781 | 3 213.587 |
| **10** | 9.834 | 9.757 | 10.610 | 10.193 | 10.451 | 10.550 | 10.073 | 9.805 | 10.945 | 9.925 |
| **11** | 13.111 | 12.214 | 12.000 | 12.237 | 11.488 | 13.216 | 11.654 | 12.691 | 0.000 | 11.591 |
| **12** | 43.625 | 44.076 | 48.348 | 44.433 | 43.049 | 44.226 | 44.410 | 45.455 | 43.502 | 43.586 |
| **13** | 106.358 | 108.707 | 105.647 | 104.493 | 115.629 | 96.232 | 114.980 | 108.466 | 103.016 | 103.638 |
| **14** | 3 241.341 | 3 290.872 | 3 168.857 | 3 273.550 | 3 141.227 | 3 426.619 | 3 542.499 | 3 454.331 | 3 209.495 | 3 145.463 |
| **15** | 14.909 | 15.626 | 14.668 | 19.399 | 15.937 | 6.570 | 15.751 | 15.669 | 16.058 | 16.180 |
| **16** | 9.523 | 8.918 | 9.099 | 8.366 | 9.323 | 10.208 | 8.456 | 8.846 | 10.093 | 8.884 |
| **17** | 3 286.107 | 3 251.267 | 3 157.076 | −0.025 | 3 186.149 | 3 281.104 | 3 300.818 | 3 250.481 | 3 057.617 | 3 198.044 |
| **18** | 6.969 | 6.821 | 7.012 | 6.577 | 6.788 | 7.713 | 7.143 | 7.101 | 6.995 | 7.141 |
| **19** | 2.778 | 1.721 | 3.107 | 6.118 | 2.568 | 2.977 | 2.048 | 2.511 | 2.514 | 1.835 |
| **20** | 8.851 | 8.472 | 8.295 | 7.687 | 8.276 | 13.436 | 8.350 | 8.698 | 8.552 | 9.004 |
| **21** | 0.060 | 2.829 | 5.416 | 3.588 | 3.218 | 0.909 | 0.005 | 2.852 | 3.000 | 3.158 |
| **22** | 3 449.612 | 3 252.626 | 3 245.874 | 3 206.596 | 3 093.677 | 2 547.770 | 3 190.468 | 3 290.467 | 3 215.822 | 3 242.390 |
| **23** | 3.524 | 40.255 | 83.658 | 44.617 | 40.618 | 3.828 | 41.666 | 40.361 | 40.569 | 39.843 |
| **24** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **25** | 3 170.607 | 3 084.613 | 0.002 | 3 111.693 | 3 360.142 | 3 175.267 | 3 218.368 | 3 148.517 | 3 188.600 | 3 106.831 |
| **26** | 9.189 | 9.259 | 19.509 | 9.402 | 9.672 | 9.624 | 9.646 | 9.492 | 10.306 | 9.075 |
| **27** | 9.212 | 8.640 | 58.563 | 9.065 | 9.264 | 9.227 | 9.391 | 9.029 | 9.511 | 9.750 |
| **28** | 135.455 | 135.502 | 138.672 | 134.455 | 0.380 | 138.490 | 135.636 | 134.048 | 139.788 | 136.627 |
| **29** | 38.820 | 39.735 | −0.129 | 37.309 | 39.091 | 38.531 | 12.165 | 39.113 | 39.576 | 39.451 |
| **30** | 3 153.123 | 2 938.660 | 2 860.240 | 2 935.470 | 2 933.068 | 2 973.706 | 2 974.055 | 2 967.886 | 2 933.126 | 2 868.412 |
| **31** | 3.164 | 2.977 | 4.631 | 2.975 | 3.307 | −0.051 | 0.111 | 3.115 | 3.112 | 2.984 |
| **32** | 8.268 | 8.336 | 8.599 | 1.706 | 8.474 | 8.853 | 8.814 | 8.673 | 9.287 | 8.294 |

**Table E.3** – Cross Talk Data: Bank CD, EtherDAQ ID BEAC02

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 180.745 | 172.981 | 173.723 | 177.941 | 194.795 | 22.113 | 186.697 | 199.091 | 184.050 |
| **2** | 3 103.648 | 3 218.613 | 3 053.520 | 3 271.354 | 3 178.635 | 3 063.400 | 3 077.155 | 3 191.162 | 3 254.966 | 3 100.046 |
| **3** | 14.718 | 14.158 | 15.008 | 14.221 | 16.363 | 16.051 | 15.335 | 14.565 | 14.521 | 14.540 |
| **4** | 6.253 | 5.774 | 6.185 | 5.784 | 5.601 | 5.809 | 6.201 | 6.538 | 6.295 | 6.128 |
| **5** | 3 180.004 | 3 177.820 | 3 128.640 | 3 184.923 | 3 104.012 | 3 117.490 | 3 132.954 | 3 177.858 | 3 075.423 | 3 130.841 |
| **6** | 4.807 | 2.128 | 4.519 | 4.648 | 4.742 | 4.440 | 4.328 | 4.363 | 4.515 | 4.496 |
| **7** | 13.845 | 13.549 | 13.171 | 13.575 | 13.561 | 13.792 | 5.224 | 12.634 | 13.145 | 13.765 |
| **8** | 69.325 | 64.715 | 66.361 | 66.611 | 67.432 | 68.361 | 67.561 | 66.624 | 66.495 | 64.907 |
| **9** | 87.914 | 88.001 | 93.540 | 87.755 | 87.961 | 85.684 | 88.864 | 84.350 | 87.177 | 89.472 |
| **10** | 3 107.507 | 3 100.293 | 3 368.680 | 3 161.494 | 3 232.110 | 3 183.998 | 3 257.353 | 3 111.767 | 3 402.300 | 3 188.978 |
| **11** | 11.931 | 11.802 | 10.853 | 11.821 | 10.415 | 11.245 | 10.508 | 10.487 | 0.000 | 11.570 |
| **12** | 6.142 | 5.651 | 6.487 | 5.559 | 4.982 | 5.472 | 5.626 | 6.399 | 5.238 | 5.630 |
| **13** | 3 193.159 | 3 299.381 | 3 195.607 | 3 151.250 | 3 502.268 | 3 232.617 | 3 505.778 | 3 298.458 | 3 125.986 | 3 172.843 |
| **14** | 3.430 | 3.484 | 3.177 | 3.354 | 3.005 | 3.403 | 3.406 | 3.576 | 3.301 | 3.005 |
| **15** | 10.219 | 10.366 | 10.351 | 10.672 | 9.741 | 3.733 | 11.096 | 9.685 | 10.989 | 11.089 |
| **16** | 4.385 | 4.106 | 4.300 | 3.904 | 4.323 | 4.936 | 3.091 | 3.693 | 3.965 | 4.166 |
| **17** | 122.193 | 120.294 | 116.848 | 0.228 | 117.398 | 113.718 | 122.585 | 120.154 | 113.048 | 118.409 |
| **18** | 3 166.521 | 3 261.697 | 3 282.400 | 3 246.543 | 3 238.038 | 3 345.598 | 3 242.810 | 3 322.873 | 3 256.501 | 3 229.639 |
| **19** | 19.595 | 15.966 | 15.100 | 15.137 | 14.820 | 22.596 | 32.515 | 15.673 | 15.761 | 14.731 |
| **20** | 21.278 | 20.870 | 22.371 | 21.212 | 20.060 | 24.601 | 27.286 | 20.867 | 20.516 | 21.055 |
| **21** | 0.118 | 3 223.755 | 3 246.605 | 3 218.083 | 3 584.481 | 2 162.775 | 4.756 | 3 140.701 | 3 207.929 | 3 280.294 |
| **22** | 84.809 | 79.664 | 81.894 | 80.635 | 76.450 | 10.542 | 87.764 | 81.363 | 78.933 | 80.245 |
| **23** | 1.154 | 10.109 | 12.249 | 11.113 | 10.735 | 1.929 | 37.875 | 10.064 | 10.165 | 10.575 |
| **24** | 8.408 | 7.831 | 8.793 | 0.351 | 8.838 | 8.658 | 8.433 | 9.079 | 8.346 | 8.750 |
| **25** | 19.205 | 17.794 | −0.114 | 18.034 | 19.314 | 19.233 | 17.907 | 18.632 | 18.283 | 17.937 |
| **26** | 3 360.736 | 3 380.259 | 3 423.895 | 3 370.154 | 3 382.651 | 3 428.027 | 3 478.645 | 3 427.509 | 3 682.412 | 3 311.663 |
| **27** | 188.073 | 182.877 | 257.082 | 186.910 | 194.121 | 185.847 | 227.829 | 189.856 | 179.797 | 201.756 |
| **28** | 5.542 | 5.203 | 13.497 | 5.164 | 1.654 | 4.444 | 11.474 | 5.011 | 4.979 | 4.739 |
| **29** | 2 872.377 | 2 952.004 | −0.957 | 3 159.496 | 3 086.466 | 3 222.277 | 202.515 | 2 971.752 | 3 060.761 | 2 877.592 |
| **30** | 3.499 | 2.970 | 14.202 | 2.857 | 2.975 | 2.788 | 12.848 | 3.227 | 2.702 | 2.876 |
| **31** | 6.577 | 6.308 | 77.730 | 6.503 | 7.420 | 0.147 | 0.123 | 6.353 | 6.128 | 6.445 |
| **32** | 1.708 | 1.770 | 1.334 | −0.161 | 1.667 | 1.910 | 2.381 | 1.987 | 1.935 | 1.340 |

**Table E.4** – Cross Talk Data: Bank CD, EtherDAQ ID 1242LC82

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 180.728 | 176.427 | 177.040 | 178.462 | 195.076 | 8.799 | 187.276 | 199.133 | 184.592 |
| **2** | 3 108.339 | 3 224.319 | 3 058.221 | 3 276.969 | 3 185.125 | 3 068.312 | 3 082.901 | 3 196.697 | 3 260.456 | 3 104.605 |
| **3** | 14.583 | 14.432 | 15.117 | 14.342 | 16.767 | 16.253 | 15.592 | 15.003 | 14.417 | 14.397 |
| **4** | 6.300 | 5.632 | 5.906 | 6.027 | 5.985 | 5.851 | 6.430 | 6.503 | 6.684 | 6.115 |
| **5** | 3 179.756 | 3 178.674 | 3 128.238 | 3 185.200 | 3 105.252 | 3 117.425 | 3 133.507 | 3 178.445 | 3 075.499 | 3 130.287 |
| **6** | 4.526 | 1.978 | 4.297 | 4.648 | 4.641 | 4.576 | 4.569 | 4.595 | 4.301 | 4.349 |
| **7** | 13.901 | 13.791 | 13.075 | 13.773 | 13.940 | 14.002 | 4.980 | 13.055 | 13.117 | 13.490 |
| **8** | 69.114 | 64.930 | 66.015 | 66.701 | 67.468 | 68.834 | 68.034 | 66.972 | 66.747 | 64.904 |
| **9** | 86.975 | 88.089 | 92.938 | 87.418 | 87.542 | 85.649 | 88.885 | 83.615 | 86.743 | 89.247 |
| **10** | 3 085.711 | 3 080.427 | 3 345.647 | 3 139.976 | 3 210.865 | 3 162.009 | 3 235.724 | 3 091.505 | 3 378.445 | 3 166.542 |
| **11** | 11.846 | 11.680 | 10.768 | 11.711 | 10.035 | 11.294 | 10.348 | 10.702 | 0.000 | 11.675 |
| **12** | 5.911 | 5.696 | 6.240 | 5.414 | 5.296 | 5.915 | 5.498 | 6.798 | 5.024 | 5.382 |
| **13** | 3 166.209 | 3 272.960 | 3 168.581 | 3 124.987 | 3 473.516 | 3 207.767 | 3 476.662 | 3 271.706 | 3 099.124 | 3 145.579 |
| **14** | 3.106 | 3.338 | 2.922 | 2.937 | 2.932 | 3.548 | 3.599 | 3.414 | 3.054 | 3.273 |
| **15** | 10.001 | 10.103 | 9.858 | 10.581 | 9.638 | 4.048 | 11.034 | 9.763 | 10.317 | 10.861 |
| **16** | 4.197 | 4.405 | 4.114 | 3.828 | 4.206 | 4.890 | 2.981 | 3.524 | 3.629 | 4.457 |
| **17** | 123.334 | 121.302 | 117.948 | 0.060 | 118.371 | 116.662 | 123.793 | 121.080 | 114.026 | 119.503 |
| **18** | 3 193.358 | 3 291.077 | 3 310.053 | 3 274.407 | 3 266.568 | 3 373.402 | 3 270.600 | 3 351.587 | 3 283.434 | 3 256.570 |
| **19** | 19.684 | 16.209 | 15.593 | 14.998 | 14.973 | 23.210 | 33.337 | 15.970 | 15.920 | 15.454 |
| **20** | 21.470 | 20.782 | 22.160 | 20.633 | 20.139 | 25.418 | 27.953 | 21.242 | 20.568 | 22.133 |
| **21** | −0.005 | 3 260.290 | 3 276.155 | 3 248.509 | 3 624.277 | 2 232.914 | 4.725 | 3 175.311 | 3 242.405 | 3 315.263 |
| **22** | 86.610 | 80.079 | 81.235 | 80.300 | 76.210 | 10.892 | 89.104 | 81.829 | 80.056 | 81.180 |
| **23** | 1.129 | 10.282 | 12.420 | 11.333 | 10.774 | 1.930 | 38.145 | 10.011 | 10.212 | 10.827 |
| **24** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **25** | 19.004 | 17.851 | −0.189 | 17.666 | 19.374 | 19.731 | 17.145 | 18.263 | 18.895 | 17.942 |
| **26** | 3 353.317 | 3 374.226 | 3 416.427 | 3 363.130 | 3 376.252 | 3 420.865 | 3 471.861 | 3 421.922 | 3 675.134 | 3 304.491 |
| **27** | 186.847 | 182.196 | 255.375 | 185.311 | 194.046 | 186.016 | 226.918 | 189.743 | 179.402 | 201.332 |
| **28** | 5.450 | 4.859 | 12.987 | 5.036 | 1.528 | 4.423 | 11.170 | 5.137 | 5.006 | 4.848 |
| **29** | 3 307.070 | 2 890.376 | −0.883 | 3 164.376 | 2 843.901 | 2 787.415 | 196.754 | 2 816.492 | 2 866.261 | 2 880.093 |
| **30** | 3.079 | 2.891 | 14.006 | 2.677 | 3.055 | 3.273 | 12.861 | 3.159 | 3.175 | 2.858 |
| **31** | 6.292 | 6.592 | 77.485 | 6.278 | 7.986 | 0.125 | −0.059 | 6.601 | 6.558 | 6.234 |
| **32** | 2.054 | 2.142 | 2.084 | −0.069 | 1.768 | 1.970 | 1.742 | 0.980 | 2.014 | 1.559 |

**Table E.5** – Cross Talk Data: Bank EF, EtherDAQ ID BEAC02

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 58.552 | 56.252 | 56.314 | 57.059 | 63.033 | 17.050 | 60.074 | 64.635 | 59.657 |
| **2** | 97.926 | 101.356 | 96.362 | 103.123 | 99.907 | 96.204 | 96.664 | 100.046 | 102.294 | 97.621 |
| **3** | 3 153.411 | 3 138.816 | 3 231.913 | 3 193.183 | 3 476.727 | 3 419.910 | 3 209.635 | 3 183.472 | 3 219.505 | 3 147.812 |
| **4** | 12.433 | 12.225 | 12.229 | 12.215 | 12.182 | 12.058 | 11.949 | 11.946 | 12.574 | 11.774 |
| **5** | 1.874 | 2.272 | 1.822 | 2.031 | 2.017 | 1.385 | 2.342 | 1.809 | 2.323 | 2.104 |
| **6** | 11.553 | 5.345 | 11.426 | 10.943 | 11.459 | 10.967 | 10.431 | 10.527 | 10.623 | 11.878 |
| **7** | 4.441 | 4.545 | 4.326 | 4.420 | 4.511 | 4.390 | 1.526 | 4.133 | 4.561 | 4.455 |
| **8** | 3 131.703 | 2 968.779 | 3 043.358 | 3 018.694 | 3 065.174 | 3 108.578 | 3 096.832 | 3 037.638 | 3 046.132 | 2 943.185 |
| **9** | 32.114 | 31.045 | 33.027 | 31.617 | 32.345 | 30.493 | 32.404 | 29.562 | 32.397 | 32.344 |
| **10** | 83.954 | 84.633 | 91.550 | 85.436 | 87.407 | 86.009 | 88.451 | 84.295 | 92.566 | 86.368 |
| **11** | 3 555.554 | 3 300.247 | 3 263.524 | 3 342.051 | 3 196.664 | 3 333.519 | 3 231.828 | 3 269.903 | 2 893.253 | 3 331.480 |
| **12** | 12.599 | 12.481 | 13.783 | 12.311 | 12.364 | 12.712 | 12.666 | 12.751 | 12.755 | 12.343 |
| **13** | 1.384 | 1.320 | 1.356 | 0.789 | 2.178 | 0.935 | 1.871 | 1.802 | 1.907 | 1.200 |
| **14** | 8.469 | 8.307 | 7.982 | 8.407 | 7.872 | 8.334 | 9.091 | 8.451 | 7.850 | 8.159 |
| **15** | 2.735 | 2.546 | 2.415 | 2.549 | 2.744 | 0.944 | 2.616 | 2.557 | 2.642 | 2.534 |
| **16** | 3 168.389 | 3 088.866 | 3 122.298 | 3 084.578 | 3 124.214 | 3 358.862 | 3 012.014 | 3 106.557 | 3 211.309 | 3 060.236 |
| **17** | 3.724 | 4.362 | 3.465 | 0.210 | 3.872 | 2.567 | 4.191 | 2.990 | 3.097 | 4.321 |
| **18** | 87.544 | 90.501 | 90.623 | 90.196 | 89.496 | 92.104 | 90.543 | 92.126 | 90.220 | 88.981 |
| **19** | 3 270.949 | 3 359.944 | 3 259.608 | 3 307.781 | 3 360.349 | 3 307.304 | 3 338.179 | 3 355.597 | 3 375.639 | 3 312.886 |
| **20** | 15.479 | 14.778 | 15.350 | 15.871 | 14.749 | 15.217 | 15.522 | 14.843 | 14.964 | 15.391 |
| **21** | 0.133 | 2.012 | 2.354 | 2.828 | 3.364 | 0.260 | 0.025 | 2.402 | 2.235 | 2.888 |
| **22** | 6.343 | 5.601 | 5.943 | 9.729 | 6.093 | 0.718 | 6.253 | 6.168 | 5.635 | 5.810 |
| **23** | 0.110 | 0.727 | 0.691 | 0.573 | 0.909 | 0.130 | 0.897 | 0.513 | 0.870 | 0.561 |
| **24** | 2 975.379 | 2 988.703 | 2 991.649 | 0.089 | 3 047.316 | 3 090.320 | 2 933.833 | 3 127.500 | 3 027.176 | 3 026.991 |
| **25** | 5.667 | 5.638 | −0.017 | 5.368 | 6.439 | 5.821 | 3.955 | 5.949 | 4.893 | 5.536 |
| **26** | 15.735 | 16.133 | 15.453 | 18.557 | 15.464 | 15.806 | 16.480 | 15.692 | 16.867 | 15.010 |
| **27** | 3 080.799 | 2 993.742 | 3 084.583 | 3 059.856 | 3 190.969 | 3 048.956 | 2 924.649 | 3 106.334 | 2 961.304 | 3 302.555 |
| **28** | 68.251 | 68.134 | 69.641 | 71.212 | −0.019 | 69.699 | 67.972 | 67.257 | 69.957 | 69.252 |
| **29** | 5.747 | 5.239 | 0.119 | 5.257 | 5.705 | 5.136 | 2.134 | 4.957 | 6.018 | 4.985 |
| **30** | 6.650 | 5.164 | 5.226 | 35.137 | 6.125 | 5.546 | 5.158 | 5.412 | 5.529 | 5.403 |
| **31** | 1.099 | 1.105 | 0.915 | 9.792 | 1.334 | 0.036 | 0.084 | 1.246 | 1.096 | 1.203 |
| **32** | 3 336.889 | 3 449.384 | 3 640.165 | −11.625 | 3 406.865 | 3 390.829 | 3 468.277 | 3 473.185 | 3 481.640 | 3 365.268 |

**Table E.6** – Cross Talk Data: Bank EF, EtherDAQ ID 1242LC82

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 58.742 | 56.168 | 57.293 | 57.522 | 62.496 | 12.339 | 59.936 | 64.855 | 59.819 |
| **2** | 97.648 | 101.418 | 95.942 | 102.952 | 100.027 | 95.906 | 96.833 | 100.100 | 102.250 | 97.567 |
| **3** | 3 158.847 | 3 145.769 | 3 238.596 | 3 199.589 | 3 484.872 | 3 426.374 | 3 216.642 | 3 189.822 | 3 226.123 | 3 153.037 |
| **4** | 12.096 | 12.303 | 12.143 | 12.189 | 12.123 | 12.222 | 12.303 | 12.797 | 12.248 | 12.196 |
| **5** | 1.856 | 2.668 | 1.726 | 2.260 | 1.871 | 1.128 | 2.306 | 2.105 | 2.481 | 2.166 |
| **6** | 11.285 | 5.566 | 11.599 | 11.235 | 11.233 | 10.765 | 10.745 | 11.061 | 10.767 | 11.941 |
| **7** | 4.232 | 4.589 | 4.528 | 4.414 | 4.384 | 4.431 | 1.501 | 4.544 | 4.461 | 4.464 |
| **8** | 3 132.372 | 2 970.878 | 3 045.113 | 3 020.147 | 3 067.981 | 3 109.982 | 3 098.996 | 3 039.328 | 3 047.992 | 2 944.086 |
| **9** | 31.046 | 31.233 | 32.619 | 31.494 | 32.155 | 30.927 | 32.241 | 29.257 | 31.733 | 32.199 |
| **10** | 82.894 | 83.990 | 90.801 | 84.643 | 86.635 | 85.546 | 87.739 | 83.684 | 91.704 | 85.509 |
| **11** | 3 535.896 | 3 282.932 | 3 245.380 | 3 323.354 | 3 179.648 | 3 314.883 | 3 214.372 | 3 252.190 | 2 879.609 | 3 311.838 |
| **12** | 12.490 | 12.426 | 14.175 | 12.378 | 11.952 | 12.328 | 12.586 | 12.845 | 12.407 | 12.578 |
| **13** | 1.130 | 1.269 | 1.379 | 0.789 | 2.147 | 1.164 | 1.698 | 2.063 | 1.711 | 1.063 |
| **14** | 7.798 | 8.237 | 7.523 | 8.316 | 7.477 | 8.433 | 8.963 | 8.436 | 7.483 | 7.897 |
| **15** | 2.402 | 2.456 | 2.512 | 2.440 | 2.513 | 1.052 | 2.549 | 2.459 | 2.523 | 2.590 |
| **16** | 3 141.027 | 3 063.226 | 3 095.488 | 3 057.895 | 3 098.004 | 3 329.721 | 2 986.516 | 3 080.229 | 3 183.025 | 3 032.785 |
| **17** | 2.870 | 3.643 | 3.044 | 0.151 | 2.844 | 1.855 | 3.758 | 2.075 | 2.396 | 3.682 |
| **18** | 87.171 | 90.218 | 90.368 | 90.945 | 89.335 | 91.998 | 90.290 | 91.690 | 89.966 | 88.807 |
| **19** | 3 303.397 | 3 394.666 | 3 291.332 | 3 340.524 | 3 394.951 | 3 339.825 | 3 371.567 | 3 389.717 | 3 408.727 | 3 344.798 |
| **20** | 14.629 | 14.049 | 15.265 | 15.359 | 13.680 | 14.502 | 15.016 | 14.261 | 14.246 | 14.886 |
| **21** | −0.626 | 1.165 | 1.828 | 2.730 | 2.299 | −0.543 | −0.627 | 1.650 | 1.689 | 2.235 |
| **22** | 5.487 | 5.006 | 5.395 | 9.763 | 5.171 | −0.034 | 5.956 | 5.374 | 5.072 | 5.465 |
| **23** | −0.737 | −0.201 | 0.297 | 0.694 | 0.140 | −0.693 | 0.247 | −0.372 | −0.245 | −0.152 |
| **24** | 716.115 | 932.619 | 894.099 | 0.000 | 650.261 | 939.640 | 927.040 | 929.755 | 926.425 | 917.488 |
| **25** | 5.886 | 5.551 | 0.074 | 5.465 | 6.113 | 5.878 | 3.619 | 5.810 | 5.202 | 5.298 |
| **26** | 16.006 | 15.839 | 15.540 | 18.687 | 14.975 | 15.826 | 16.082 | 15.535 | 17.163 | 15.025 |
| **27** | 3 073.528 | 2 988.044 | 3 077.457 | 3 052.666 | 3 184.479 | 3 041.623 | 2 918.746 | 3 100.406 | 2 954.117 | 3 294.408 |
| **28** | 68.038 | 67.993 | 69.794 | 70.581 | −0.057 | 68.973 | 67.799 | 67.080 | 69.027 | 68.401 |
| **29** | 5.634 | 5.203 | −0.055 | 5.570 | 5.499 | 5.243 | 2.003 | 5.102 | 5.957 | 4.886 |
| **30** | 6.648 | 5.165 | 5.454 | 35.729 | 5.659 | 5.435 | 5.298 | 5.259 | 5.686 | 5.134 |
| **31** | 1.172 | 1.165 | 1.446 | 9.940 | 1.086 | −0.046 | 0.012 | 1.223 | 1.336 | 1.009 |
| **32** | 3 337.035 | 3 448.145 | 3 645.171 | −12.636 | 3 407.028 | 3 390.014 | 3 468.777 | 3 474.155 | 3 480.624 | 3 363.837 |

**Table E.7** – Cross Talk Data: Bank GH, EtherDAQ ID BEAC02

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 1.586 | 1.464 | 1.814 | 1.411 | 1.325 | 0.167 | 1.289 | 1.820 | 1.423 |
| **2** | 4.424 | 5.341 | 5.151 | 5.283 | 5.411 | 5.085 | 4.243 | 4.502 | 4.198 | 5.110 |
| **3** | 3.669 | 3.826 | 3.708 | 3.834 | 3.874 | 4.024 | 3.750 | 3.577 | 3.757 | 3.640 |
| **4** | 3 400.705 | 3 327.704 | 3 330.393 | 3 406.499 | 3 302.385 | 3 387.980 | 3 414.609 | 3 359.718 | 3 413.578 | 3 329.436 |
| **5** | 66.594 | 67.403 | 66.829 | 67.664 | 66.257 | 66.593 | 66.702 | 68.195 | 65.853 | 66.704 |
| **6** | 129.014 | 58.240 | 128.533 | 128.260 | 131.035 | 131.649 | 127.143 | 125.939 | 127.967 | 128.584 |
| **7** | 3 041.517 | 3 006.487 | 3 123.777 | 3 056.989 | 3 082.647 | 3 167.436 | 2 927.577 | 2 988.902 | 3 045.731 | 3 086.588 |
| **8** | 6.056 | 6.009 | 5.807 | 5.964 | 6.026 | 6.247 | 6.129 | 5.823 | 5.930 | 5.919 |
| **9** | 1.354 | 2.059 | 1.874 | 1.634 | 2.136 | 1.363 | 1.693 | 1.792 | 2.119 | 1.089 |
| **10** | 3.737 | 4.305 | 3.375 | 4.151 | 4.502 | 4.254 | 4.046 | 2.934 | 4.596 | 4.218 |
| **11** | 2.971 | 2.842 | 2.753 | 3.112 | 2.637 | 2.945 | 2.693 | 2.817 | 0.000 | 2.689 |
| **12** | 3 094.367 | 3 112.275 | 3 418.962 | 3 121.906 | 3 060.953 | 3 088.658 | 3 131.082 | 3 203.343 | 3 078.660 | 3 116.489 |
| **13** | 1.718 | 1.241 | 1.151 | 0.598 | 2.406 | 1.304 | 1.085 | 1.829 | 1.287 | 0.769 |
| **14** | 7.811 | 7.360 | 7.072 | 7.195 | 7.697 | 7.944 | 7.905 | 7.414 | 7.794 | 7.559 |
| **15** | 3 075.739 | 3 121.313 | 3 035.152 | 3 120.498 | 3 092.918 | 3 017.490 | 3 306.349 | 3 114.619 | 3 115.412 | 3 179.517 |
| **16** | 53.165 | 51.625 | 52.161 | 51.530 | 52.363 | 56.250 | 50.271 | 51.752 | 53.520 | 51.036 |
| **17** | 23.544 | 22.143 | 21.932 | 0.087 | 21.270 | 18.399 | 22.063 | 21.637 | 20.237 | 21.157 |
| **18** | 2.556 | 2.739 | 2.238 | 3.128 | 2.508 | 3.144 | 3.316 | 2.351 | 3.026 | 2.059 |
| **19** | 4.185 | 3.032 | 2.860 | 2.872 | 2.753 | 4.312 | 3.180 | 2.797 | 3.173 | 2.788 |
| **20** | 3 307.210 | 3 248.179 | 3 392.913 | 3 252.500 | 3 188.857 | 3 369.358 | 3 332.626 | 3 297.829 | 3 306.636 | 3 352.454 |
| **21** | 0.084 | 3.589 | 2.296 | 2.640 | 3.015 | 2.787 | 0.187 | 2.868 | 2.293 | 2.487 |
| **22** | 21.140 | 13.562 | 15.190 | 15.440 | 13.346 | 3.724 | 14.071 | 13.640 | 13.804 | 13.510 |
| **23** | 1 352.548 | 3 345.885 | 3 463.426 | 3 689.899 | 3 391.925 | 1 486.883 | 3 433.506 | 3 361.115 | 3 356.580 | 3 344.047 |
| **24** | 22.139 | 14.023 | 13.587 | 0.136 | 14.751 | 21.602 | 13.670 | 14.444 | 13.908 | 14.029 |
| **25** | 32.687 | 2.620 | −0.129 | 2.447 | 3.341 | 31.647 | 3.365 | 3.243 | 2.191 | 3.498 |
| **26** | 2.755 | 2.323 | 2.648 | 2.769 | 5.939 | 3.397 | 3.509 | 2.648 | 2.452 | 2.484 |
| **27** | 3.035 | 0.764 | 1.417 | 0.893 | 0.728 | 7.826 | 7.437 | 1.261 | 0.839 | 0.850 |
| **28** | 3 234.303 | 3 228.420 | 3 296.862 | 3 205.800 | −0.091 | 3 290.918 | 3 220.864 | 3 194.189 | 3 316.964 | 3 260.367 |
| **29** | 18.883 | 18.050 | 0.237 | 16.695 | 17.301 | 91.425 | 27.250 | 17.940 | 18.118 | 18.289 |
| **30** | 134.533 | 126.266 | 122.216 | 124.234 | 130.300 | 135.644 | 136.492 | 127.570 | 125.997 | 123.839 |
| **31** | 3 350.700 | 3 227.125 | 3 231.852 | 3 210.598 | 3 582.724 | −1.175 | −1.064 | 3 253.597 | 3 278.060 | 3 270.952 |
| **32** | 2.632 | 2.684 | 1.769 | 0.036 | 3.354 | 14.261 | 14.604 | 2.755 | 2.683 | 2.530 |

**Table E.8** – Cross Talk Data: Bank GH, EtherDAQ ID 1242LC82

| | Preamp ID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **0100-0034** | **0110-0006** | **0110-0030** | **0110-0042** | **0110-0049** | **0400-0009** | **0400-0017** | **0400-0021** | **0400-0036** | **0400-0047** |
| **1** | 0.000 | 1.613 | 1.706 | 1.415 | 1.582 | 0.832 | 0.293 | 1.685 | 1.754 | 1.452 |
| **2** | 4.218 | 5.515 | 5.017 | 5.569 | 5.725 | 4.978 | 4.163 | 5.012 | 4.497 | 5.113 |
| **3** | 3.573 | 3.660 | 3.941 | 3.831 | 4.311 | 3.695 | 3.796 | 3.859 | 3.688 | 3.473 |
| **4** | 3 405.654 | 3 333.088 | 3 334.350 | 3 412.983 | 3 307.260 | 3 393.255 | 3 419.795 | 3 364.517 | 3 418.093 | 3 333.304 |
| **5** | 67.057 | 67.273 | 67.250 | 67.330 | 66.675 | 66.496 | 66.994 | 67.984 | 65.729 | 66.662 |
| **6** | 129.338 | 59.278 | 128.541 | 128.226 | 131.744 | 132.016 | 127.612 | 126.001 | 127.916 | 128.720 |
| **7** | 3 046.007 | 3 011.798 | 3 128.276 | 3 062.925 | 3 087.768 | 3 172.524 | 2 929.195 | 2 993.393 | 3 050.591 | 3 090.844 |
| **8** | 5.995 | 6.038 | 5.809 | 5.988 | 6.228 | 5.994 | 6.073 | 6.124 | 5.979 | 5.665 |
| **9** | 1.467 | 2.073 | 1.870 | 1.828 | 2.231 | 1.564 | 1.839 | 1.738 | 1.905 | 1.194 |
| **10** | 3.600 | 3.947 | 3.028 | 4.101 | 4.510 | 4.484 | 3.804 | 2.876 | 4.551 | 4.140 |
| **11** | 3.105 | 2.770 | 2.620 | 2.919 | 2.924 | 3.147 | 2.646 | 2.730 | 0.000 | 2.723 |
| **12** | 3 071.256 | 3 090.223 | 3 393.973 | 3 100.040 | 3 038.871 | 3 066.206 | 3 108.119 | 3 180.302 | 3 054.795 | 3 091.982 |
| **13** | 2.017 | 1.151 | 1.262 | 0.982 | 2.400 | 1.342 | 0.963 | 1.868 | 1.084 | 0.722 |
| **14** | 7.780 | 7.307 | 7.018 | 7.081 | 7.626 | 8.203 | 7.816 | 7.020 | 7.486 | 7.658 |
| **15** | 3 057.189 | 3 103.699 | 3 016.915 | 3 102.801 | 3 074.725 | 3 006.548 | 3 286.888 | 3 096.484 | 3 096.088 | 3 159.296 |
| **16** | 52.711 | 51.127 | 51.667 | 50.909 | 52.036 | 55.604 | 49.945 | 51.389 | 53.088 | 50.501 |
| **17** | 23.800 | 22.064 | 22.350 | 0.045 | 21.168 | 19.681 | 23.015 | 22.061 | 20.679 | 21.667 |
| **18** | 2.661 | 2.530 | 2.256 | 2.919 | 2.587 | 3.874 | 3.665 | 2.607 | 3.125 | 2.618 |
| **19** | 4.441 | 2.914 | 2.935 | 2.839 | 2.591 | 4.665 | 3.626 | 2.762 | 2.911 | 3.054 |
| **20** | 3 333.649 | 3 274.698 | 3 419.838 | 3 278.892 | 3 214.172 | 3 395.932 | 3 360.197 | 3 324.267 | 3 331.898 | 3 377.891 |
| **21** | 0.025 | 3.187 | 2.066 | 2.919 | 2.696 | 2.645 | 0.153 | 3.018 | 2.404 | 2.334 |
| **22** | 21.216 | 13.614 | 14.468 | 14.471 | 13.411 | 4.094 | 14.891 | 13.903 | 13.873 | 14.021 |
| **23** | 1 464.651 | 3 377.717 | 3 492.956 | 3 721.758 | 3 422.436 | 1 577.323 | 3 466.196 | 3 392.559 | 3 386.430 | 3 373.558 |
| **24** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **25** | 31.250 | 2.444 | 0.002 | 2.365 | 3.149 | 30.672 | 3.179 | 2.634 | 2.411 | 3.235 |
| **26** | 2.560 | 2.411 | 2.564 | 2.824 | 6.231 | 3.114 | 3.151 | 2.224 | 2.449 | 2.522 |
| **27** | 2.878 | 0.851 | 1.151 | 0.877 | 0.857 | 7.710 | 6.938 | 0.839 | 0.801 | 0.986 |
| **28** | 3 226.153 | 3 221.532 | 3 288.515 | 3 198.706 | 0.070 | 3 282.325 | 3 213.398 | 3 187.117 | 3 308.305 | 3 251.509 |
| **29** | 18.616 | 17.826 | 0.264 | 16.824 | 17.584 | 91.416 | 28.069 | 17.621 | 18.347 | 18.200 |
| **30** | 134.620 | 126.141 | 123.501 | 125.040 | 130.592 | 135.476 | 136.247 | 127.338 | 125.716 | 123.241 |
| **31** | 3 344.929 | 3 222.880 | 3 226.365 | 3 206.577 | 3 575.997 | −1.076 | −1.200 | 3 249.211 | 3 272.002 | 3 264.636 |
| **32** | 2.906 | 2.693 | 2.886 | 0.091 | 3.617 | 14.607 | 14.499 | 3.023 | 3.152 | 2.642 |

**Table E.9** – Cross Talk Data: Aggregate Statistics

| | Bank AB | | | Bank CD | | | Bank EF | | | Bank GH | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **Mean** | **Std. Dev.** | **Relative** | **Mean** | **Std. Dev.** | **Relative** | **Mean** | **Std. Dev.** | **Relative** | **Mean** | **Std. Dev.** | **Relative** |
| **1** | 3 325.550 | 153.250 | 5.000% | 184.300 | 8.700 | 5.000% | 59.530 | 2.890 | 5.000% | 1.510 | 0.250 | 16.000% |
| **2** | 10.140 | 0.550 | 5.000% | 3 153.920 | 79.500 | 3.000% | 99.110 | 2.510 | 3.000% | 4.930 | 0.500 | 10.000% |
| **3** | 8.020 | 0.560 | 7.000% | 15.020 | 0.790 | 5.000% | 3 240.700 | 113.330 | 3.000% | 3.770 | 0.180 | 5.000% |
| **4** | 101.920 | 1.610 | 2.000% | 6.100 | 0.310 | 5.000% | 12.210 | 0.220 | 2.000% | 3 369.770 | 41.400 | 1.000% |
| **5** | 164.680 | 2.170 | 1.000% | 3 141.110 | 36.590 | 1.000% | 2.030 | 0.360 | 18.000% | 66.910 | 0.630 | 1.000% |
| **6** | 3 190.370 | 44.560 | 1.000% | 4.520 | 0.150 | 3.000% | 11.140 | 0.450 | 4.000% | 128.790 | 1.800 | 1.000% |
| **7** | 10.130 | 0.490 | 5.000% | 13.510 | 0.390 | 3.000% | 4.430 | 0.120 | 3.000% | 3 055.050 | 67.100 | 2.000% |
| **8** | 12.280 | 0.600 | 5.000% | 66.910 | 1.400 | 2.000% | 3 046.850 | 57.650 | 2.000% | 5.990 | 0.140 | 2.000% |
| **9** | 3 165.970 | 84.330 | 3.000% | 87.890 | 2.380 | 3.000% | 31.610 | 1.000 | 3.000% | 1.740 | 0.320 | 18.000% |
| **10** | 10.210 | 0.410 | 4.000% | 3 200.570 | 103.000 | 3.000% | 86.690 | 2.920 | 3.000% | 3.960 | 0.540 | 14.000% |
| **11** | 12.240 | 0.640 | 5.000% | 11.150 | 0.630 | 6.000% | 3 304.740 | 100.080 | 3.000% | 2.840 | 0.170 | 6.000% |
| **12** | 44.470 | 1.520 | 3.000% | 5.720 | 0.490 | 9.000% | 12.650 | 0.500 | 4.000% | 3 131.120 | 102.060 | 3.000% |
| **13** | 106.720 | 5.720 | 5.000% | 3 254.220 | 133.400 | 4.000% | 1.460 | 0.440 | 30.000% | 1.360 | 0.520 | 38.000% |
| **14** | 3 289.430 | 139.940 | 4.000% | 3.260 | 0.230 | 7.000% | 8.170 | 0.440 | 5.000% | 7.540 | 0.340 | 5.000% |
| **15** | 16.020 | 1.360 | 9.000% | 10.350 | 0.510 | 5.000% | 2.540 | 0.100 | 4.000% | 3 108.980 | 78.430 | 3.000% |
| **16** | 9.170 | 0.620 | 7.000% | 4.060 | 0.500 | 12.000% | 3 120.260 | 94.300 | 3.000% | 52.130 | 1.610 | 3.000% |
| **17** | 3 218.740 | 77.850 | 2.000% | 118.930 | 3.260 | 3.000% | 3.260 | 0.750 | 23.000% | 21.600 | 1.310 | 6.000% |
| **18** | 7.030 | 0.300 | 4.000% | 3 273.180 | 50.800 | 2.000% | 90.160 | 1.350 | 1.000% | 2.800 | 0.470 | 17.000% |
| **19** | 2.820 | 1.250 | 44.000% | 16.740 | 2.650 | 16.000% | 3 341.390 | 42.490 | 1.000% | 3.230 | 0.640 | 20.000% |
| **20** | 8.960 | 1.610 | 18.000% | 22.130 | 2.320 | 10.000% | 14.900 | 0.560 | 4.000% | 3 318.000 | 61.820 | 2.000% |
| **21** | 3.440 | 0.910 | 26.000% | 3 288.860 | 140.840 | 4.000% | 2.260 | 0.580 | 26.000% | 2.700 | 0.390 | 15.000% |
| **22** | 3 243.060 | 94.980 | 3.000% | 81.580 | 3.480 | 4.000% | 6.130 | 1.380 | 22.000% | 14.870 | 2.370 | 16.000% |
| **23** | 41.130 | 1.630 | 4.000% | 10.780 | 0.780 | 7.000% | 0.530 | 0.290 | 56.000% | 3 438.750 | 113.480 | 3.000% |
| **24** | — | — | — | — | 0.360 | 4.000% | 3 023.210 | 59.640 | 2.000% | 15.790 | 3.470 | 22.000% |
| **25** | 3 173.850 | 82.060 | 3.000% | 18.460 | 0.730 | 4.000% | 5.450 | 0.700 | 13.000% | 2.870 | 0.460 | 16.000% |
| **26** | 9.520 | 0.360 | 4.000% | 3 421.180 | 98.930 | 3.000% | 16.090 | 1.030 | 6.000% | 2.710 | 0.360 | 13.000% |
| **27** | 9.230 | 0.310 | 3.000% | 199.070 | 23.650 | 12.000% | 3 071.960 | 107.160 | 3.000% | 1.190 | 0.720 | 60.000% |
| **28** | 136.520 | 2.020 | 1.000% | 4.990 | 0.320 | 6.000% | 68.840 | 1.140 | 2.000% | 3 245.900 | 42.020 | 1.000% |
| **29** | 38.950 | 0.770 | 2.000% | 2 984.920 | 160.330 | 5.000% | 5.380 | 0.360 | 7.000% | 19.100 | 3.400 | 18.000% |
| **30** | 2 953.770 | 80.420 | 3.000% | 3.000 | 0.220 | 7.000% | 5.550 | 0.470 | 8.000% | 128.750 | 5.000 | 4.000% |
| **31** | 3.280 | 0.560 | 17.000% | 6.590 | 0.510 | 8.000% | 1.170 | 0.140 | 12.000% | 3 298.010 | 117.410 | 4.000% |
| **32** | 8.620 | 0.330 | 4.000% | 1.850 | 0.280 | 15.000% | 3 445.960 | 87.000 | 3.000% | 2.810 | 0.430 | 15.000% |

**Table E.10** – Cross Talk Data: Modified Board Results

| Channel | Bank AB | Bank CD | Bank EF | Bank GH |
|---|---|---|---|---|
| **1** | 3 241.826 | 40.201 | 7.845 | 1.000 |
| **2** | 9.625 | 3 214.404 | 18.705 | 3.552 |
| **3** | 5.331 | 19.042 | 3 493.259 | 2.090 |
| **4** | 13.185 | 7.540 | 24.160 | 3 310.190 |
| **5** | 15.777 | 3 134.993 | 8.294 | 6.556 |
| **6** | 3 280.738 | 3.159 | 10.955 | 15.331 |
| **7** | 14.676 | 7.768 | 1.517 | 3 095.440 |
| **8** | 15.941 | 6.096 | 3 065.624 | 21.328 |
| **9** | 3 202.793 | 17.761 | 5.619 | 9.554 |
| **10** | 9.847 | 3 265.013 | 18.734 | 5.331 |
| **11** | 4.446 | 20.237 | 3 203.013 | 1.174 |
| **12** | 16.568 | 7.984 | 32.645 | 3 063.196 |
| **13** | 49.316 | 3 516.598 | 6.722 | 4.744 |
| **14** | 3 161.708 | 2.520 | 8.872 | 14.219 |
| **15** | 10.796 | 7.715 | 2.010 | 3 104.320 |
| **16** | 9.734 | 1.348 | 3 136.324 | 16.591 |
| **17** | 3 179.260 | 17.079 | 4.164 | 8.314 |
| **18** | 7.795 | 3 256.037 | 13.742 | 3.150 |
| **19** | 1.738 | 23.824 | 3 352.600 | 0.841 |
| **20** | 9.596 | 7.447 | 21.840 | 3 188.090 |
| **21** | 9.415 | 3 476.768 | 4.522 | 5.369 |
| **22** | 3 072.912 | 2.370 | 7.403 | 22.507 |
| **23** | 10.765 | 7.010 | 0.935 | 3 370.507 |
| **24** | 15.379 | 7.999 | 3 052.505 | 17.382 |
| **25** | 3 360.186 | 25.524 | 6.969 | 4.670 |
| **26** | 8.142 | 3 342.201 | 18.971 | 6.531 |
| **27** | 5.215 | 27.440 | 3 212.986 | 0.804 |
| **28** | 0.083 | 1.740 | −0.363 | −0.119 |
| **29** | 26.099 | 3 295.390 | 6.568 | 9.184 |
| **30** | 2 968.426 | 2.725 | 7.283 | 29.576 |
| **31** | 5.334 | 5.782 | 0.883 | 3 589.614 |
| **32** | 10.674 | 2.009 | 3 414.933 | 7.278 |

# Appendix F

# Cross Talk Deconvolution

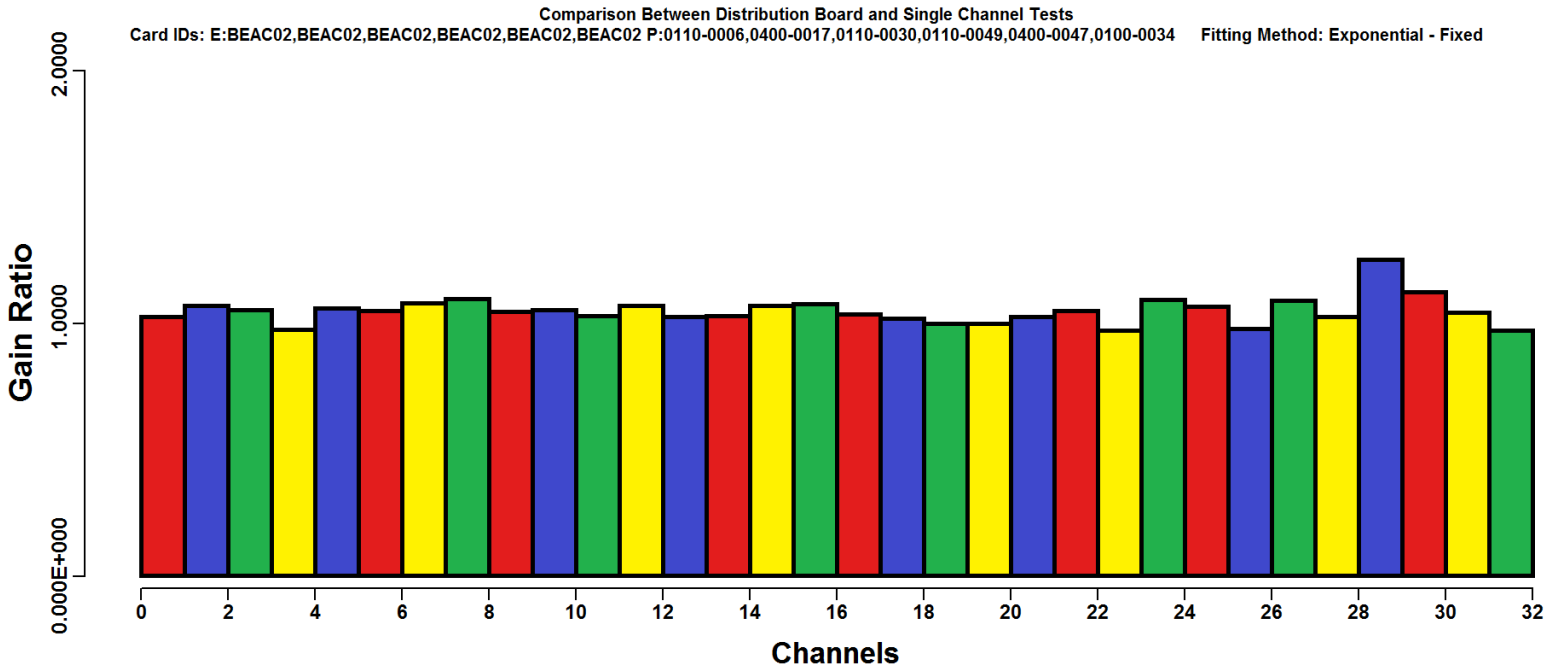



**Figure F.1** – Generated Deconvolution Matrix

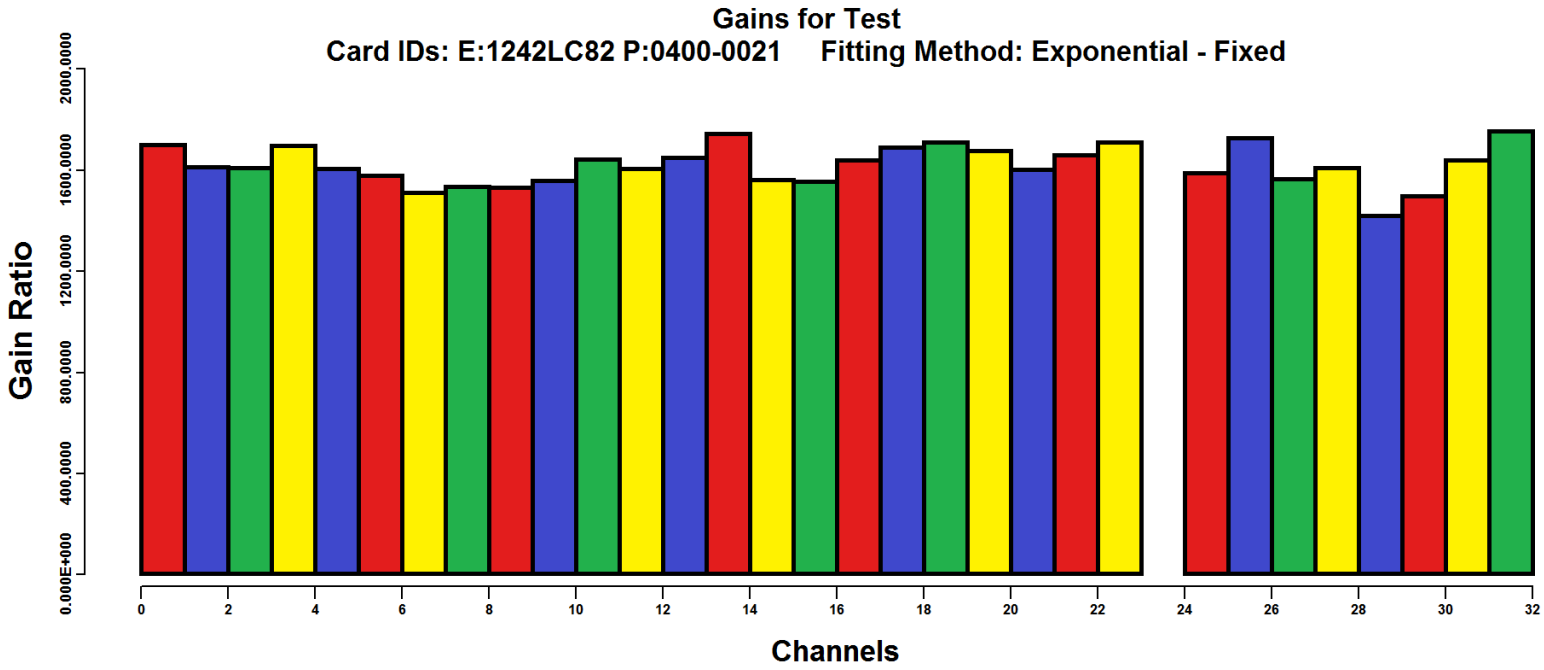



**Figure F.2** – Distribution Test Results

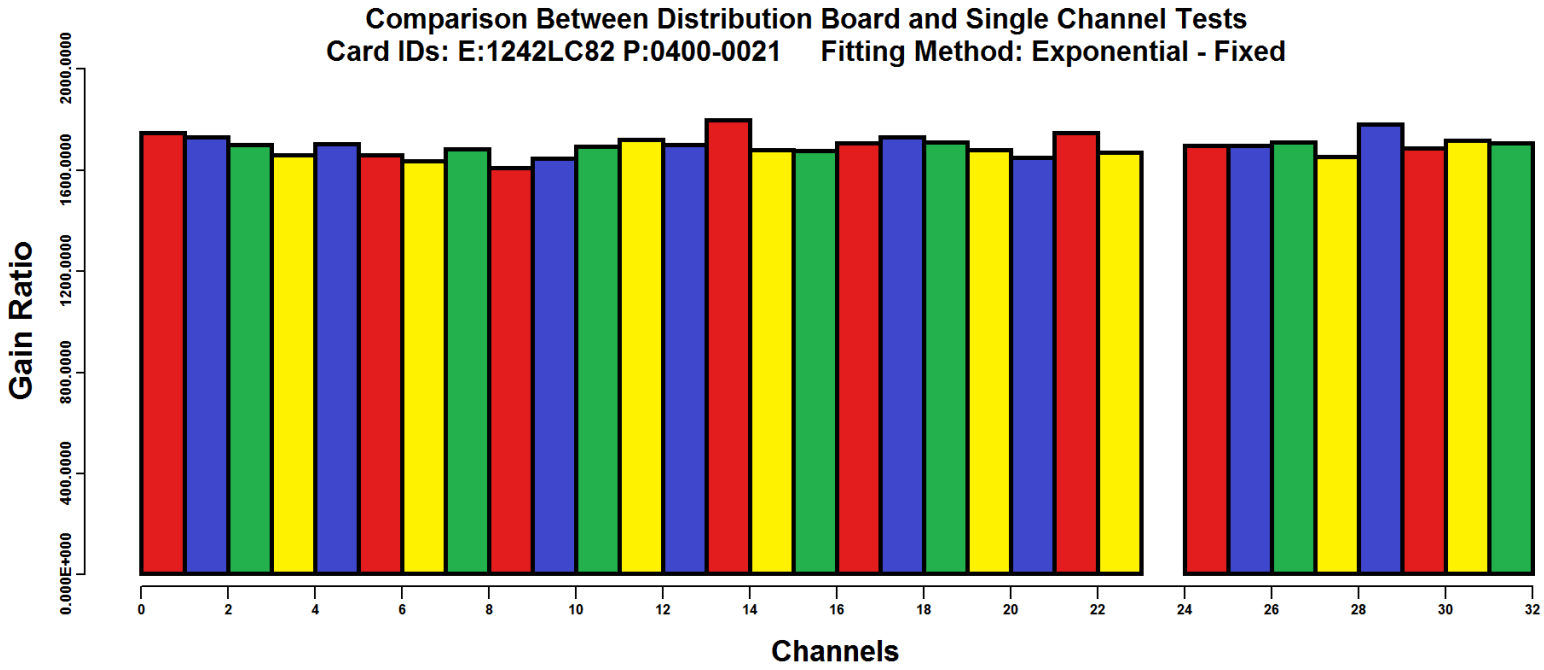



**Figure F.3** – Predicted Single Channel Test Results

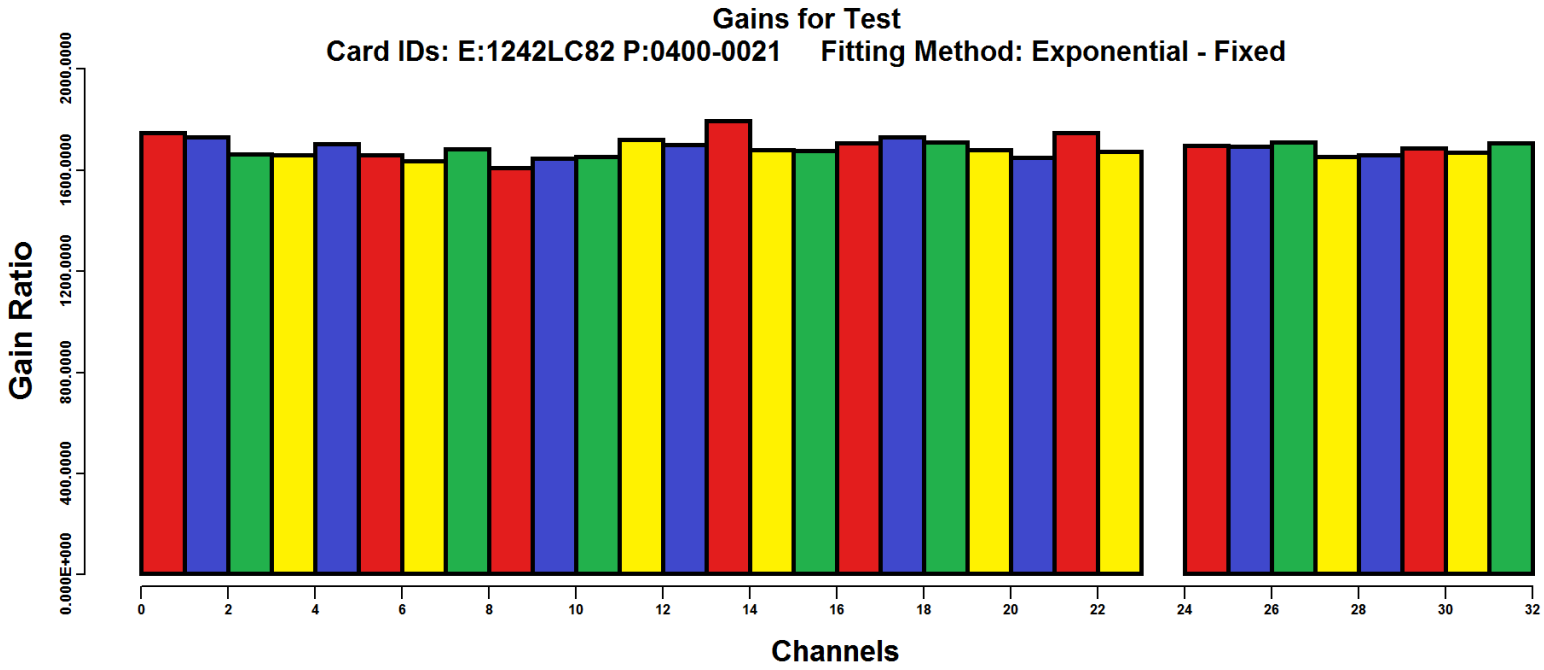
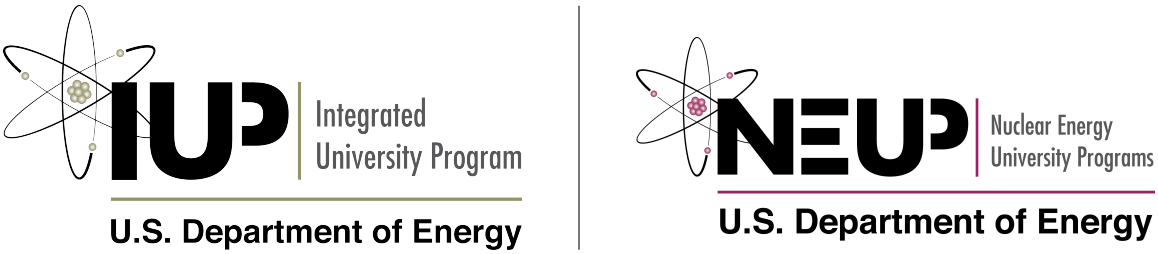



**Figure F.4** – Actual Single Channel Test Results




This research is being performed using funding received from the DOE IUP