



**Programming in LabView:
The 2280 System Ramp and Linearity Tests**

Stacey Nordt

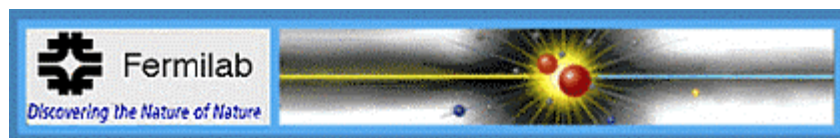
Computer Science

North Carolina A&T State University

August 7, 2001

Project Supervisor: Tim Kasza

Fermi National Accelerator Laboratory



Abstract

The objective of this project was to convert a Visual C++ program into a LabView program. The LabView program was used to test the 2280 system for accurate channel data and develop various charts to output the data received. This paper discusses the process in which this was done.

Table of Contents

Introduction

I.	Project Introduction	3
II.	Test Modules	4
III.	LabView	5

Project 2280

I.	Description	6
II.	Training	7
III.	Pre-Requisites	7
IV.	Data Table	8
V.	Ramp Test	9
VI.	Deviations Test	10
VII.	Diflin Test	11
VIII.	Post-requisites	12
IX.	Problems with Project	12
X.	Future Adjustments	13

Closing

I.	Conclusion	13
II.	Acknowledgements	14
III.	References	14
IV.	Appendix	15

Introduction

In January 2000, the Computing Department of Fermi National Accelerator Laboratory started a project that dealt with the conversion of existing diagnostics into LabView. These diagnostics are used by the Equipment Support Department to test various CAMAC modules. They were originally written in several different programming languages and have occupied the attention of technicians and summer interns in order to convert them. The Equipment Support Department first explored the feasibility of LabView during the summer of 1999. The technicians found that LabView was a more practical programming tool than the various programming languages used prior to this evaluation. The Computing Department did the first alteration consisting of converting PCCDL (Personal Computer CAMAC diagnostic languages) into LabView 5.1 in 2000. In 2001, they converted the tests created in LabView 5.1 to the newer LabView 6i. They also started a directions page in HTML explaining how to use and set up each test developed. The goal for summer 2002 was to complete the diagnostics conversion for CAMAC systems (LRS 2280, 4300, 4290) written in C or C++ programming languages with the most recent version, LabView 6.1. This report describes the methods used for completing the LRS 2280 system project and assisting the Computing Department in accomplishing this goal.

Test Modules

CAMAC

The Computer Automated Measurement And Control (CAMAC) is a modular data handling system that can be used for data and programmable applications. CAMAC crates are controlled by intelligent devices. For this project the Jorway 73A SCSI bus crate controller was utilized. The main function of the CAMAC is to provide communication from the modules to the standardized backplane called a DATAWAY and then the DATAWAY to a computer.

CAMAC Modules

The CAMAC modules required for this test package are:

- LRS 2323 - Programmable gate and delay generator
- LRS 4222 - Gate and delay generator that produces time delays and intervals
- LRS 2280 - The analog to digital conversion system processor
- LRS 1976 - Time to digital and analog to digital conversion tester
- LRS 1987A - 48 channel Charge Fan Out
- Fermi 156 - A digital analog converter that provides a voltage level to the 2280 to determine the overall gain of the system
- LRS 2285A - Analog to Digital Converter being tested for its linearity and data from its 24 channels

NIM

The first Nuclear Instrumentation Module was made in 1964 for the nuclear and high energy physics communities. The purpose of the NIM was to create a system of modules

that could be interchangeable. The NIM module required for this test package is the LRS 428F.

LabView

Laboratory Virtual Instrument Engineering Workbench (LabView), developed by National Instruments, is a powerful analysis programming language. It was first introduced in 1983, when National Instruments was interested in discovering new ways to decrease the time used in programming software languages. The idea was to combine front panel user interfaces with a block diagram programming method allowing for a faster and more efficient, software-based graphical instrumentation. Since 1983, LabView has grown popular in many research laboratories and industries.

LabView is a part of a new graphical programming language, called “G”. It uses terminology, icons and ideas that are familiar to technicians, scientists and engineers. Virtual Instruments (VI’s) created in LabView can generate charts, graphs and customized, user-defined graphics. LabView integrates data acquisition, analysis, and presentation in one system making programming simple and manageable.

Versions of LabView

- LabView 1: ©1986. This version was only created for Apple Macintosh due to its graphical capabilities.
- LabView 2: ©1990. LabView was rewritten and execution speeds were comparable to C programming languages.

- LabView 3: ©1993. Used with Apple Macintosh, Microsoft Windows, and SunOS. Version 3 introduced multi-platform capabilities.
- LabView 4: ©1996. LabView improved on a simplified set of palettes that include only the most used functions.
- LabView 5: ©1998. This version added multithreading, ActiveX Automation, translation tools, instrumentation, data acquisition, and general interface.
- LabView 6: ©2000. Version 6 added Waveform improvements, I/O name controls, DataSocket improvements, cluster and array shortcuts, automatic wiring, better graph and chart set up, better navigation of palettes, and a new dialog box.
- LabView 6.1: ©2001. Most recent version. It is now compatible with PC's running Microsoft Windows, UNIX/Linux, MacOS, and Concurrent PowerMax operating systems.

Project 2280

Description

The main purpose of this project was to convert a previously written Visual C++ program into a LabView program. The C++ code was written in order to test the LRS 2280 CAMAC ADC system for data acquisition and linearity. A data table was created that included numerical information outputted from the system. The modules' performance was evaluated by creating three tests using the data table. The three tests displayed graphs that allowed the user to see the results. They were the Ramp, Deviations, and Difflin graphs. The test could be run for a 12-bit or 15-bit 2280 module. All of these steps had to be reconstructed in LabView.

Training

Having no knowledge of LabView prior to arriving at Fermilab, extensive training was necessary. The training exercises consisted of the two tutorials and two demonstration sessions listed below.

- LabView Tutorial Basics: Getting Started, Introduction to LabView, Creating and Debugging a VI, Creating a sub VI, Loops and Charts, Arrays and Graphs, Case and Sequence Structures, Strings and File I/O, and VI options.
- LabView Tutorial Advanced: Planning LabView Applications, LabView Clusters, Local and Global Variables, Attribute Nodes, Advanced File I/O Techniques, and Developing Larger Projects.
- LabView Demonstrations: Conducted by supervisor, Tim Kasza, and veteran intern, Joseph Jaskierny.

Prerequisite to Starting the Project

- Prior to starting this project, it had to be possible for all of the 2280 tests to be accessed upon request. It was essential to develop a main test panel to do so.
- A CAMAC I/O analysis also had to be produced to test the 2280 for valid sub-addresses and function codes to ensure accurate testing ability.

Data Table

The data table was established so that the user could see the data values being read out of the 24 channels of the 2285 A. The table also outputs the following terms:

pC: the charge in Pico Coulombs being outputted from the 1987A

DAC word: digital representation of voltage level applied to charge fan-out module

Residual: the distance of a data point from a calculated slope

Worst Residual: the residual that is the most out of range of the expected absolute value.

Maximum Value: the maximum output value of each channel from the 2285

Gain: counts per Pico Coulomb

Pedestal: the initial data gathered from the 2285 with no input charge

If the absolute value of the residuals for the 12-bit module exceeds 2, the channel will fail. The same results will occur for the 15-bit module if the residuals exceed 15.

The screenshot shows a LabVIEW front panel titled "LRS2280_Ram280c.vi". On the left, there is a "BIT SWITCH" control with a knob set to "12 bit". The main area is a "Data Table" with the following columns: Header, step 16: Data, Residual, step 17: Data, Residual, step 18: Data, Residual, step 19: Data, Residual, Maximum Value, Gain, Pedestal, and Worst Resk. The table contains 24 channels of data, each with a "PASSED" status in a green box. Below the table, there are two input fields: "Dac Word" (value 8912) and "pC" (value 272). A "Return to Main Menu" button is at the bottom.

Header	step 16: Data	Residual	step 17: Data	Residual	step 18: Data	Residual	step 19: Data	Residual	Maximum Value	Gain	Pedestal	Worst Resk
0	0.000	0	0.000	0	0.000	0	0.000	0	0.00	0	0.000	
PASSED Channel 1	3225	-0.098	3396	0.087	3568	-0.728	3739	-0.543	3739	10.07	486	1.608
PASSED Channel 2	3176	-0.092	3348	-0.814	3519	-0.535	3690	-0.257	3690	10.08	436	0.961
PASSED Channel 3	3238	0.148	3409	-0.506	3579	-0.160	3750	-0.814	3750	10.02	513	1.073
PASSED Channel 4	3229	0.666	3402	0.592	3576	-0.483	3749	-0.557	3749	10.17	463	1.187
PASSED Channel 5	3126	-0.741	3297	-0.070	3469	-0.399	3641	-0.729	3641	10.10	379	0.906
PASSED Channel 6	3255	0.312	3426	-0.102	3597	-0.515	3768	-0.929	3768	10.04	526	1.207
PASSED Channel 7	3280	-0.020	3452	-0.223	3624	-0.426	3796	-0.629	3796	10.10	532	1.589
PASSED Channel 8	3340	-0.153	3517	0.169	3695	-0.508	3873	-1.186	3873	10.43	504	1.589
PASSED Channel 9	3412	-0.162	3593	0.083	3775	-0.673	3957	-1.429	3957	10.66	513	1.128
PASSED Channel 10	3353	-0.141	3529	-0.132	3705	-0.123	3881	-0.114	3881	10.35	537	0.823
PASSED Channel 11	3295	-0.046	3469	-0.307	3643	-0.568	3816	0.171	3816	10.22	516	-0.871
PASSED Channel 12	3375	-0.273	3553	0.247	3732	-0.234	3911	-0.714	3911	10.50	519	1.208
PASSED Channel 13	3352	-0.552	3530	-0.106	3709	-0.660	3888	-1.214	3888	10.50	497	1.327
PASSED Channel 14	3234	-0.502	3407	0.575	3583	-1.348	3756	-0.271	3756	10.24	449	1.038
PASSED Channel 15	3244	-0.219	3416	-0.560	3588	-0.902	3759	-0.243	3759	10.10	498	1.171
PASSED Channel 16	3372	-0.450	3549	-0.219	3726	0.012	3905	-1.757	3905	10.42	537	1.935
PASSED Channel 17	3360	-0.469	3537	0.559	3716	-0.414	3895	-1.386	3895	10.47	511	1.336
PASSED Channel 18	3235	-0.032	3406	0.079	3577	0.189	3749	-0.700	3749	10.06	498	1.195
PASSED Channel 19	3279	0.421	3451	-0.053	3623	-0.526	3795	-1.000	3795	10.09	535	-0.947
PASSED Channel 20	3426	-0.805	3605	0.263	3786	-0.668	3967	-1.600	3967	10.59	545	1.126
PASSED Channel 21	3316	-0.289	3493	-0.064	3671	-0.839	3848	-0.614	3848	10.43	480	1.362
PASSED Channel 22	3248	-0.168	3419	0.183	3591	-0.465	3762	-0.114	3762	10.08	506	0.726
PASSED Channel 23	3386	-0.265	3566	-0.467	3746	-0.669	3925	0.129	3925	10.58	510	1.162
PASSED Channel 24	3387	-0.768	3562	-0.079	3738	-0.389	3914	-0.700	3914	10.33	576	1.405
Dac Word	8912		9469		10026		10583					
pC	272		289		306		323					

Figure 1: front panel screen shot of the last few data table columns.

Ramp Test

The Ramp Test was built to plot the ADC output (counts) versus charge of each channel of the 2280 system on a graph. There are 20 data points on the graph to represent the number of steps. The steps are calculated by the number of times there was charge inputted into the 1987 A.

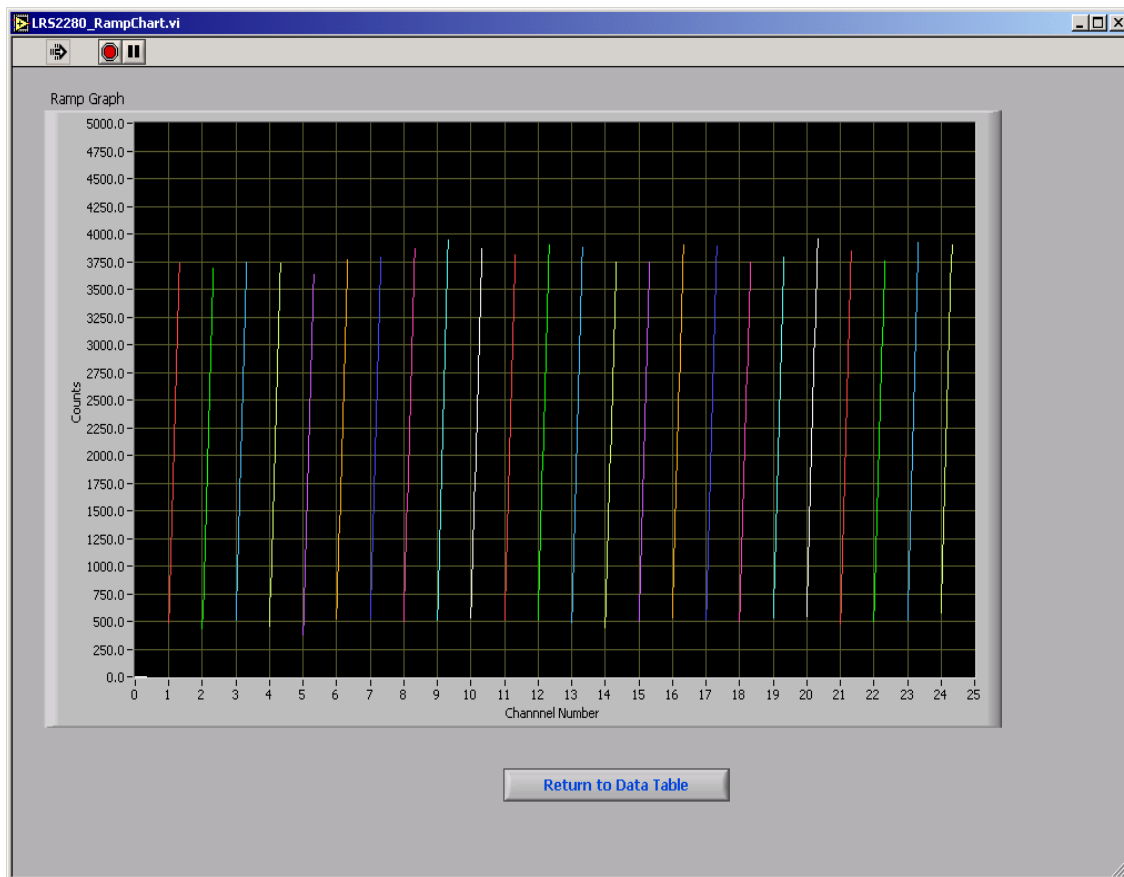


Figure 2: Front panel of the Ramp Graph

Deviations Test

This test outputs a graph that plots the residuals and the pC with the data received from each channel of the 2285 A. The graph allows the user to see if the residuals are within the appropriate range.

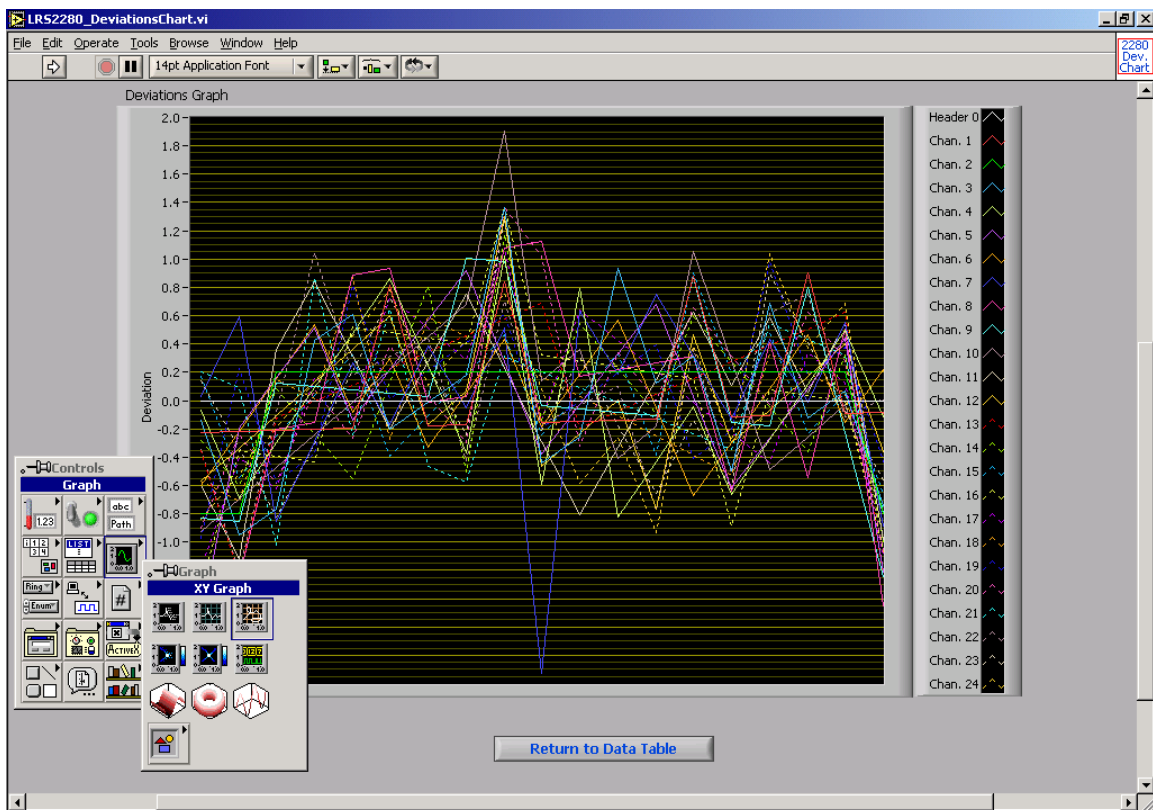


Figure 3: The front panel of the Deviations Graph with graphing menu.

Diflin Test

The Diflin Test was meant to test the differential non-linearity of the 2285 A. The graph displays hexadecimal data for incrementing pC values. The range for these results depends on the resolution (12-bit or 15-bit) of the 2280 module. The test observes whether or not each data bit has been reached and that none of the bits get stuck on or off. This lets the user know that the data is precise.

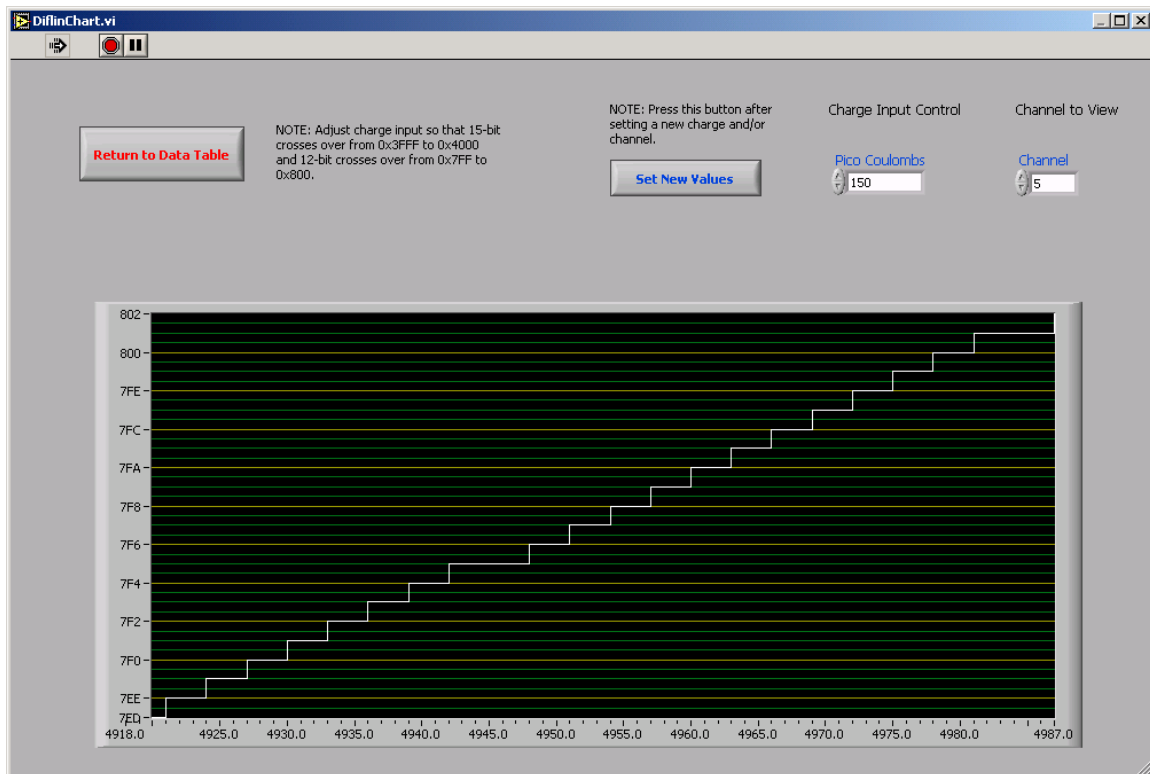


Figure 4: Front panel of the Diflin Graph with accurate data and no missed bits.

* Note: To see an example of the block diagrams for these tests refer to the appendix.

Post-requisite Before Completion

Upon completing each project, it is required that an HTML Directions page be developed explaining the set up and instructions for the project with the intention of making the LabView program more convenient for the user. This page can be found at http://www-esd.fnal.gov/Lvdocs/Lrs2280/Lrs2280_Directions.html. The programmer also has to perform a demonstration of the project for the Computing Department.

Problems with Project

- C++ and Fortran code: The C++ code that was to be converted was written in the summer of 1996. Most of the source code had been misplaced in this six-year period. Fortunately, the code written in C++ had been a conversion from a previously written Fortran code. In addition to rewriting the C++ program, knowledge of the Fortran program was needed to locate the missing code. The majority of the functions written to create the Diflin graph were missing as well. Consequently, it was necessary to use the LRS 4300 system program in order to complete the graph.
- Lack of Space: In LabView programming there is limited space available before the program starts to become illegible. To alleviate this problem sub VI's within VI's needed to be created. This can also be difficult to follow, but it allows for more programming space.

- File Corruption: File corruption occurred on the 14th of June, which prevented opening the LabView program. Assuming that the entire project was lost caused disappointments and delays; however, it was later discovered that the drives are backed up each night and only one day's work was lost.

Possible Future Adjustments

National Instruments upgrades LabView annually. In the past, the Computing Division has had to make changes to programs they wanted to upgrade from 6.i to 6.1. The 2280 System Project may have to undergo some modifications for future upgrades. Also, upon the completion of projects in the Computing Department, the technicians test run the software to check for adjustments that may be needed prior to its use. As a result, there may have to be some minor changes to the 2280 system project.

Conclusion

The 2280 system project has been completed, although minor adjustments may be necessary. The data being outputted is correct when compared with the original Visual C++ code; however, the code had to be altered. The maximum charge requested for the original code had to be decreased due to aged modules. This alteration improved the residual values used for graphing.

I believe that LabView is a worthy addition to the technological advancements in programming languages. It is efficient and manageable for the programmer as well as the user. I appreciate being given the opportunity to work with LabView and with such an outstanding group of engineers.

Acknowledgements

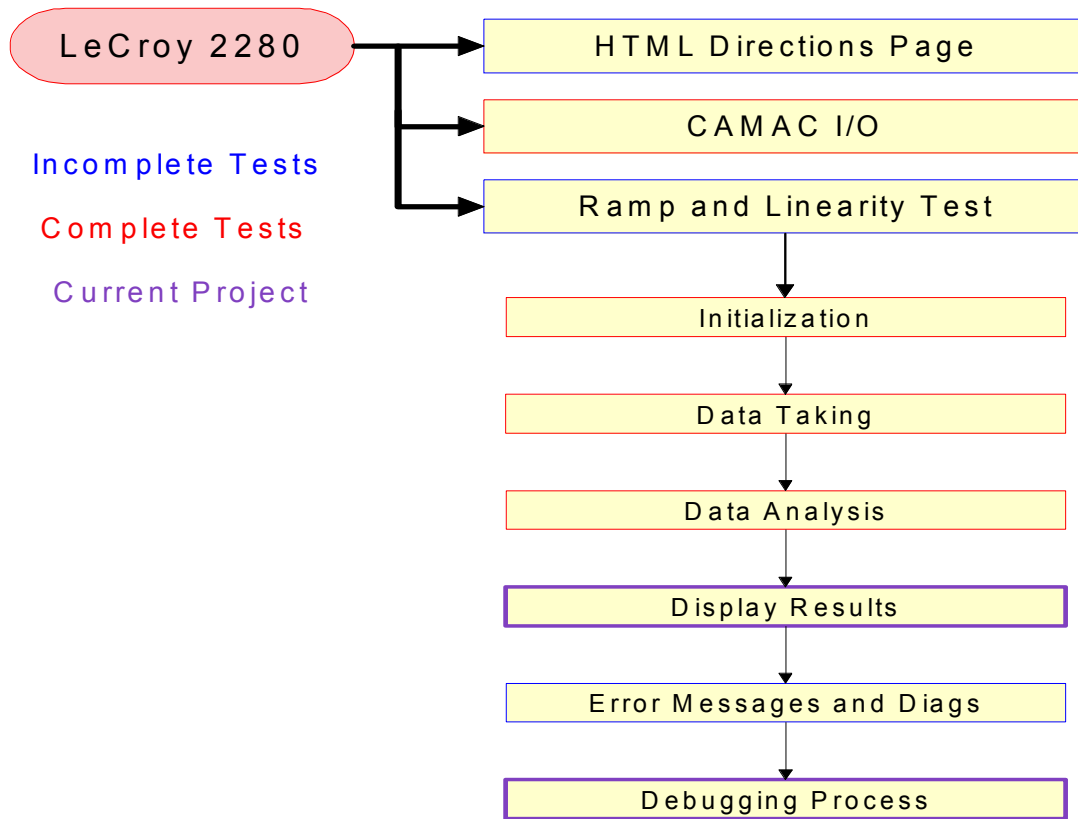
I would like to thank Elliot McCrory and Dianne Engram for allowing me to be apart of the Fermilab family. Thanks to Jim Davenport for giving me wonderful input regarding my paper and to Dave Peterson for being an excellent mentor. To Tim Kasza, Adam Walters, and the entire Computing Department for making sure that I was comfortable and that I had everything I needed. I'd like to give a special thanks to the LabView Team: Mike Behnke, Rick Mahlum, Joseph Wilhelmi, and Joseph Jaskierny. I couldn't have completed this project without you guys. To the SIST interns, whether you like it or not we will always be bonded by this experience. In the case that I have mistakenly forgotten anyone, please know that I appreciate you as well.

Thank you all for making this a pleasurable and memorable summer.

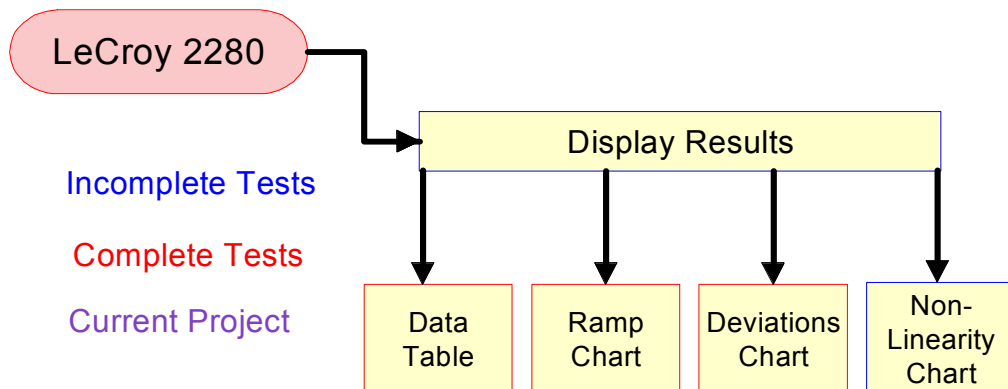
References

1. "Electronics Equipment Pool Catalog" Dec. 2001. FermiLab.
<<http://www-esd.fnal.gov/esd/catalog/>>
2. "LabView Manual" 2002. National Instruments<<http://www.ni.com>>
3. Fortran Power Station Reference. Microsoft Corporation, 1995.
4. Nell Dale, Chip Weens, and Mark Headington. "Programming and Problem Solving with C++". 2nd edition. Sudbury, MA: Jones and Bartlett Publishers, Inc., 2000

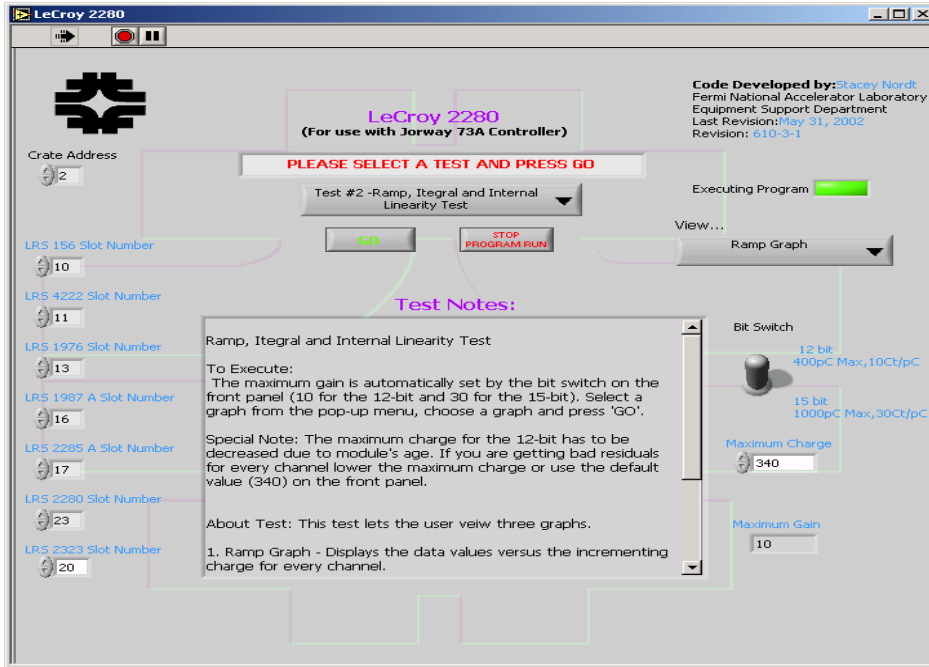
Appendix



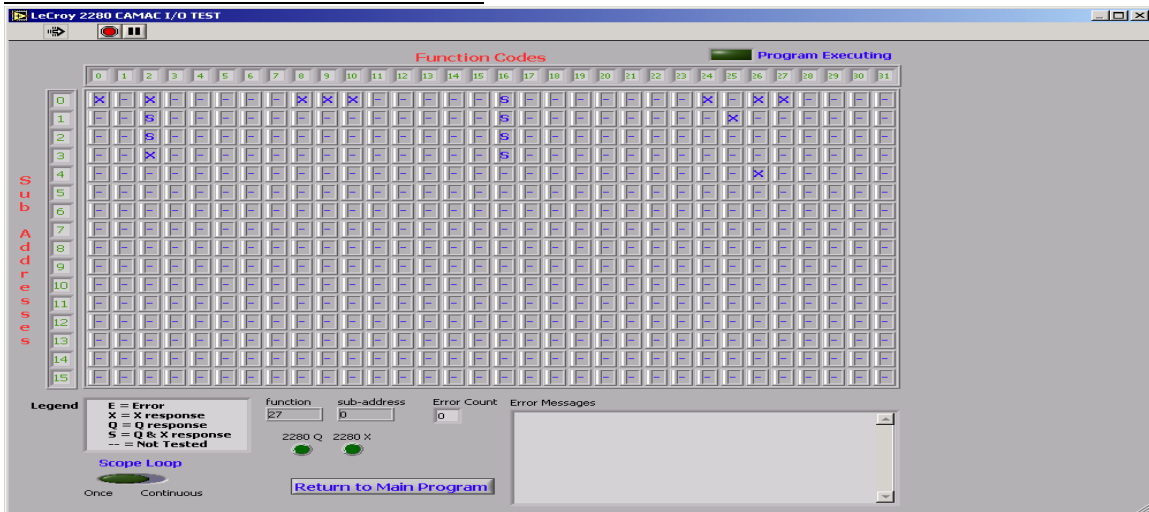
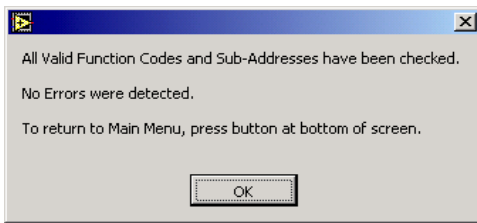
Example Flowcharts completed to display the status of the project in weekly LabView meetings.



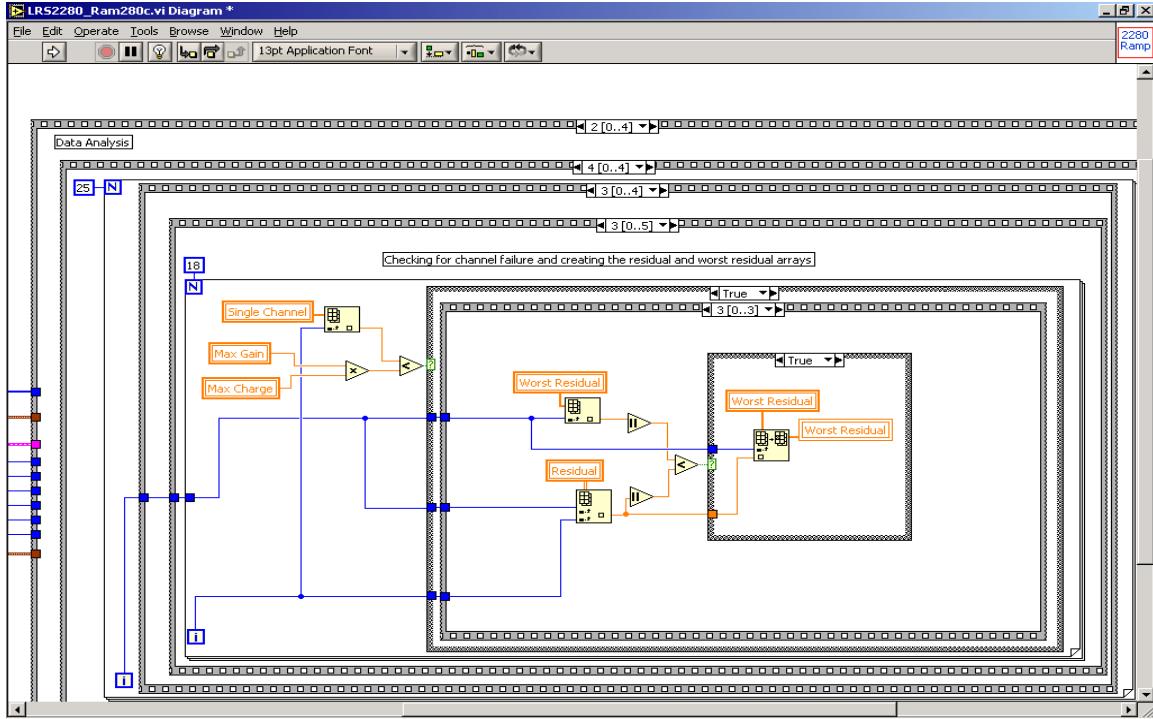
Main test panel



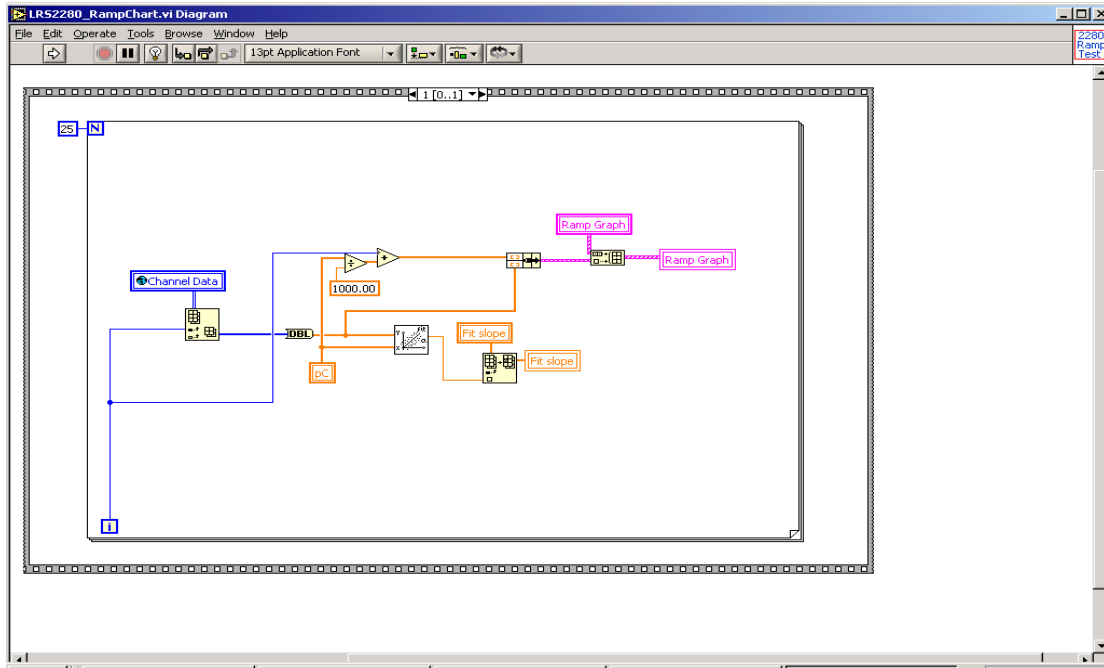
CAMAC I/O front panel and error detector



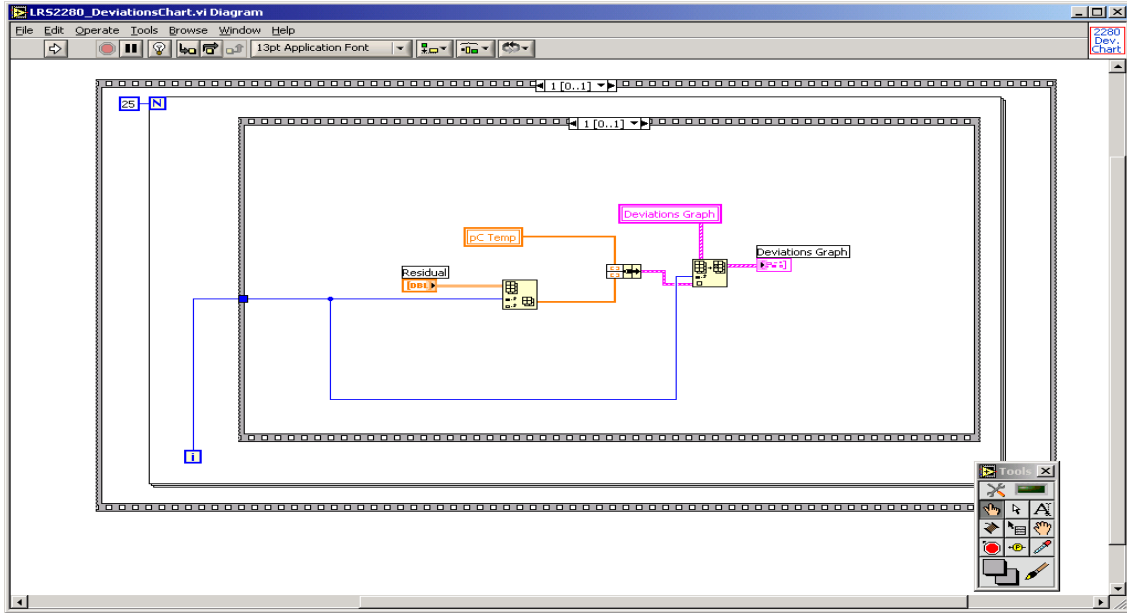
Block Diagram for the Data Table



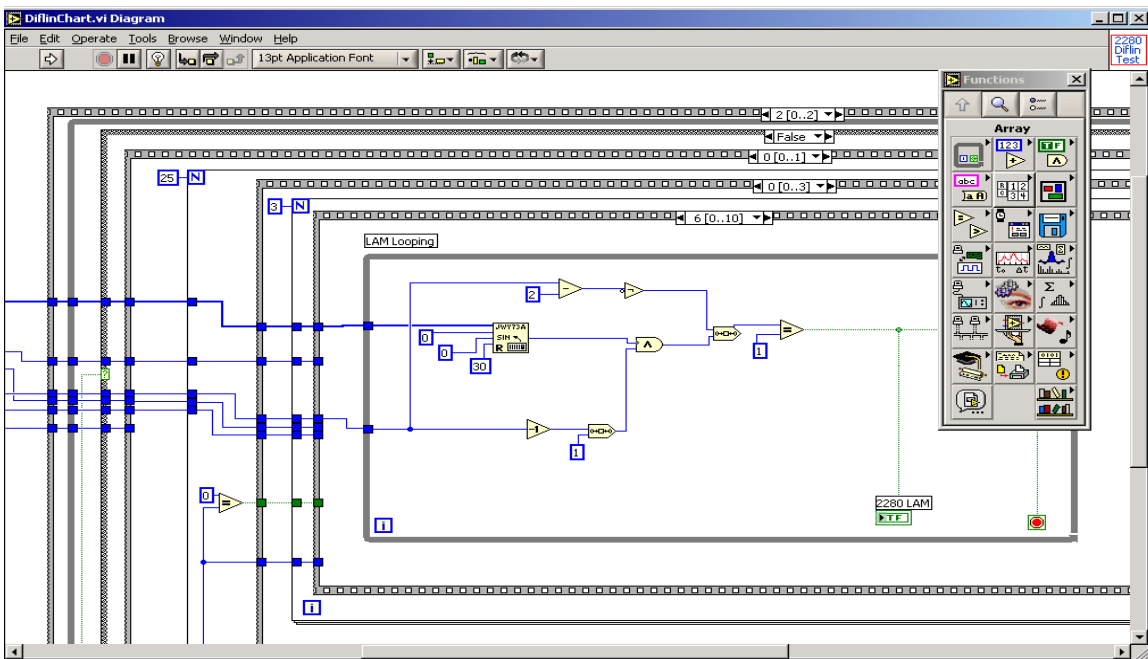
Block Diagram of the Ramp Graph



Block diagram of the Deviations Graph with a sample of the tool bar



Block diagram of the Difflin Graph with functions menu



Crate Setup



Module Set Up from left to right: 399 backplane display, 428F ([with nimcam adapter]– anywhere), 156 (slot 10), 4222 (slot 11), 1976 (slot 13), 1987A (slot 16), 2285A (slot 17), 2323 (slot 20), 2280 (slot 19), and crate controller.

Colors:

7 Lemo Cables – green, 1 50-pin charge fan out – red, 2 Lemo Terminators – yellow, 1 50-pin Terminator - tan

Cabling:

1976 ZERO TO + 20V to 1987 VDAC
2323 DLY chan A to 4222 IN TRIG (3 nSecs)
4222 OUT1 to 2280 CLEAR (3 nSecs)
4222 OUT3 to 428F IN (1 nSec)
428F OUT to 1987 GATE (under 1976 label)
428F OUT to 2280 GATE IN (3 nSecs)
2280 REQ to 2280 GRANT IN (1 nSec)
1987 50-pin charge fan out to adc inputs of the 2285 using coax cables
50-pin terminator (51 ohm resistors across unused 1987 outputs)

Lemo Terminators:

4222 unused IN TRIG input
2280 INH OUT (2285 only)