

# Electromagnet Testing Using Hardware-Programmed Devices

Marco Enriquez

August 6, 2003



Fermi National Particle Accelerator



Tufts University, ECE Department

## ABSTRACT:

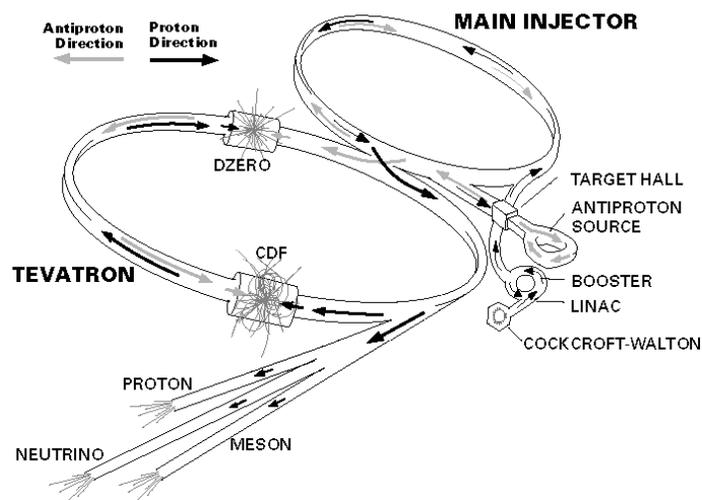
The Extensible Measurement System (EMS), a framework that standardizes electromagnet measurement systems like the Single Stretched-Wire System (SSW), is an important part of particle acceleration at Fermilab. These programs insure that electromagnets have the correct properties before they are used in particle accelerators. The EMS system is consistently being integrated with new programs to improve current measurement systems. One of these new programs is a website that helps test the SSW's positioning system for the Linux platform. This program was successfully implemented using a complex hierarchy of programs written in C++, CGI, HTML and JavaScript.

## INTRODUCTION:

### I. Fermilab and Particle Acceleration

The Fermi National Accelerator Laboratory (Fermilab) is one of the world's leading research facilities for Particle Physics and Astrophysics. Laboratories such as Fermilab have advanced human knowledge of the world and the universe by analyzing various particle-antiparticle collisions and the fundamental matter particles called quarks, which occasionally emerge from them. All known subatomic particles such as protons and neutrons are composed of quarks. Two of the six known quarks in the universe, the bottom and top quark, have been discovered at Fermilab by colliding protons and anti-protons using the world's highest energy particle accelerator, the Tevatron.

The Tevatron acquired its name because it accelerates particles to roughly one trillion electron-Volts (1 TeV). There are, however, numerous processes that need to take place before particles can achieve such a high energy and speed. The particles must first

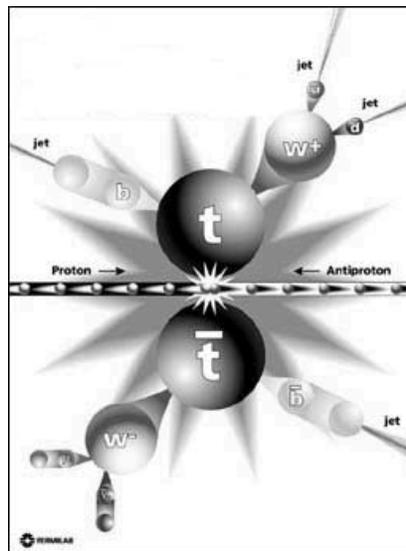


**Figure 1: Fermilab's chain of particle accelerators**

pass four preliminary accelerators before they reach the Tevatron, where they can be collided. These four accelerators are called the Cockcroft-Walton, the Linear Accelerator (or Linac), the Booster and the Main Injector. Figure 1 shows how these particle accelerators are configured in Fermilab.

The Cockcroft-Walton is where protons are produced and preliminarily accelerated. The protons are then further accelerated into the Linear Accelerator, where they gather more energy and speed. The Linear Accelerator then leads protons to a series of two circular accelerators: the booster and the Main Injector. The antiprotons also use primary accelerators such as the Main Injector to gain energy and speed. Combined, these primary accelerators accelerate protons and antiprotons roughly 150 GeV (billion electron-Volts).

After gathering enough energy and speed in the Main Injector, the protons and anti-protons are sent to the Tevatron in opposing paths. The protons and anti-protons are collided at a detector hall once they gather approximately 1 trillion electron-Volts of energy and are moving at 99.99% the speed of light. These particle-antiparticle collisions are then carefully tracked and recorded by powerful computers located in the detector halls. This data is used to determine if a pair of quarks and antiquarks has been created from the collisions. Figure 2 illustrates a proton-antiproton collision and the quarks that result from the collision.



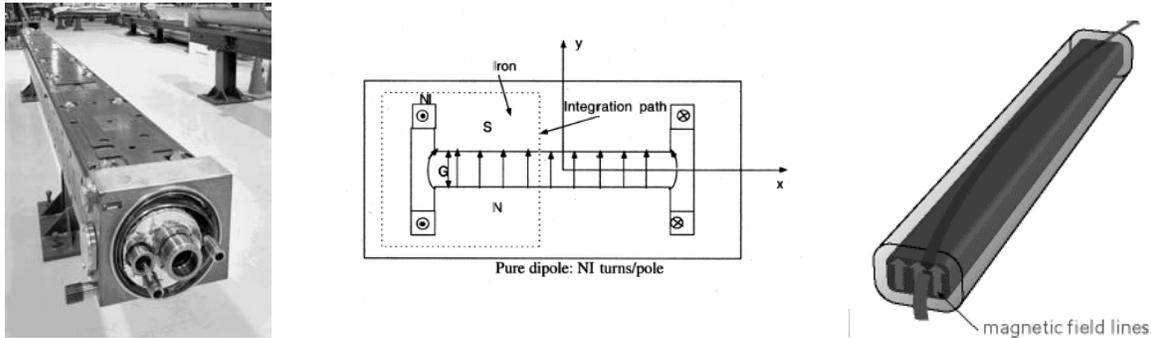
**Figure 2: Graphical depiction of a proton-antiproton collision**

## II. Particle Accelerators and Electromagnets

To better understand the process of particle acceleration, it is also important to understand what particle accelerators are mainly composed of: electromagnets. Particle accelerators are massive devices that create strong electromagnetic fields which alter and

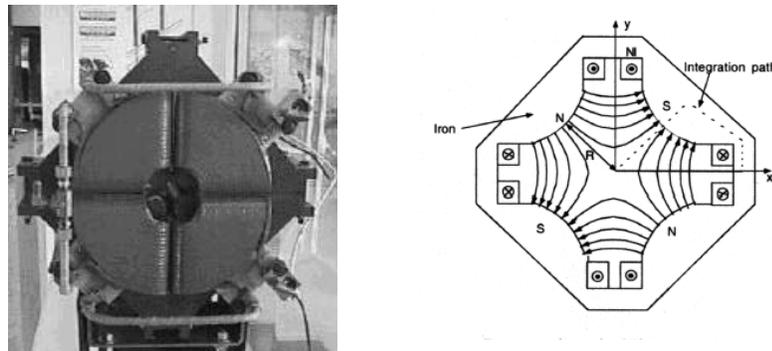
confine various particles' paths. Accelerators such as the Tevatron use many powerful electromagnets to create the strong electromagnetic field needed for particle-antiparticle collisions.

The two most commonly used types of magnets at Fermilab are dipole and quadrupole electromagnets. Dipole electromagnets only have one North and South pole, and are responsible for bending the paths of particle beams inside accelerators. Figure 3a shows a picture of the dipole electromagnet and Figure 3b depicts a cross section of a typical dipole electromagnet. Figure 3c illustrates the bent path particles take in a dipole electromagnet.

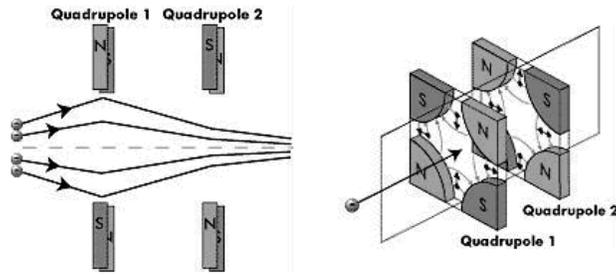


**Figure 3a (Left): A dipole magnet**  
**Figure 3b (Middle): A cross-Section of a dipole magnet**  
**Figure 3c (Right): A bent particle path due to a dipole magnet**

Quadrupole electromagnets have two North poles and two South poles and are used to focus and defocus particle beams inside the accelerator. Figure 4a shows a quadrupole electromagnet and Figure 4b illustrates its cross section. This focusing and defocusing keeps the particle beams from diverging to any other direction, hence confining them in the particle accelerator. Figure 4c and 4d exemplify particle confinement using focusing and defocusing quadrupoles.

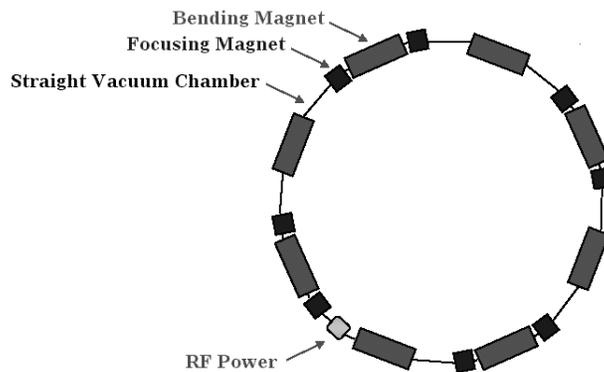


**Figure 4a (Left): A quadrupole magnet**  
**Figure 4b (Right): A cross-section of a quadrupole magnet**



**Figure 4c (Left): Particle beam focusing using quadrupole magnets**  
**Figure 4d (Right): Particle confinement from quadrupole magnets**

Circular accelerators such as the Tevatron are comprised of dipole and quadrupole electromagnets carefully placed in a circular formation and connected by straight vacuum tube pipes. Figure 5 shows a possible configuration for a circular particle accelerator.



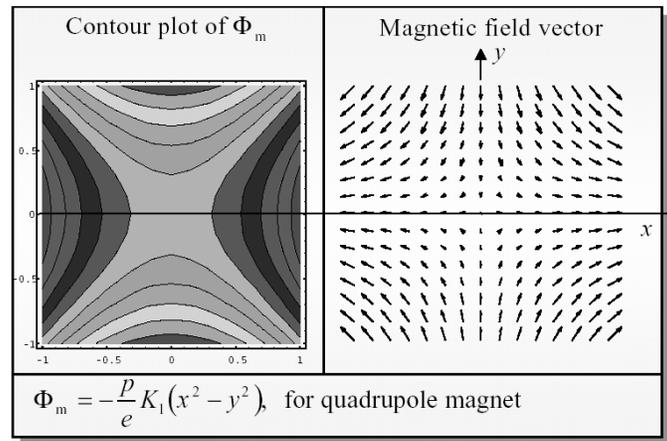
**Figure 5: A circular accelerator constructed from focusing and bending electromagnets**

Before electromagnets can be placed inside a particle accelerator, however, they must undergo rigorous testing. Even the dipole and quadrupole electromagnets already inside the accelerators are continuously replaced and tested to ensure its proper operation.

### III. Electromagnet Testing and the Single Stretched-Wire System

Since Fermilab constructs most of the dipole and quadrupole magnets for their accelerators, extensive testing of these magnets need to be done before they can be used. The three fundamental tests for electromagnets are the tests to find their field shape, magnetic strength and magnetic center.

Finding an electromagnet’s field shape is important because an electromagnet, whether it is a dipole or a quadrupole, needs to have the proper field shape in order to accomplish its task. If a dipole magnet has an overly skewed or non-uniform field, it may not bend a particle’s path properly. If a quadrupole magnet has an improper field shape, it may not focus or defocus a particle beam properly, leading to beam loss. Figure 6 shows the ideal field shape for a quadrupole magnet, and its contour graph.



**Figure 6: Field shape and contour of quadrupole magnets**

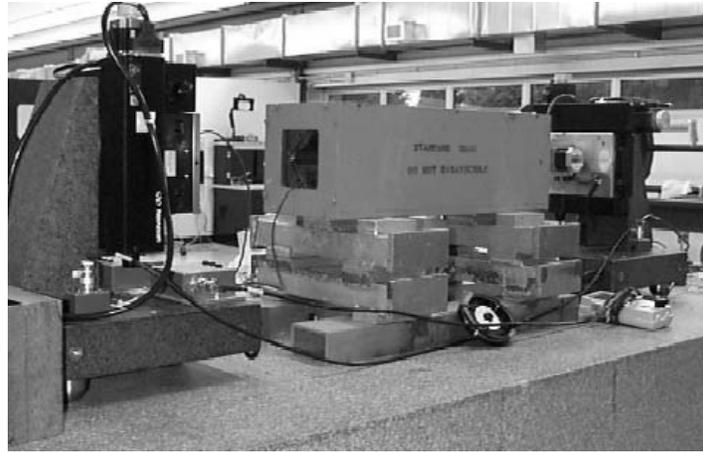
Similarly, the magnetic strength of the electromagnets needs to be known because improper magnetic strength will lead to beam loss in the particle accelerator. If a dipole is too strong, it will over-bend the particle’s path and if the dipole is too weak it will not bend the particles enough. Quadrupole magnets that are either overly strong or weak will improperly focus particles.

Tests that find the magnetic center are mostly used for quadrupole magnets, since a dipole magnet has a simple “North to South” field. Magnetic centers are important for quadrupole magnets because they act as a consistent reference point, which helps engineers align them properly in the particle accelerators. Magnet alignment is just as important as magnet field shape or strength, since misaligned quadrupole magnets also focus particle beams improperly.

To perform these calculations and tests, the engineers at the Technical Division created the Single Stretched-Wire (SSW) system. The SSW system consists of 100 um CuBe wire (chosen for its non-magnetic property and good weight to strength ratio), precision motor stages to hold and control the wire, a universal Motion Controller to control the motor stages, and a measurement machines connected to the stages.

To begin electromagnet testing, the CuBe wire is stretched through the electromagnet, placed on the motor stages and then given current. As the wire is moved around by the motor stages, properties such as flux (incurred by the wire) will change and

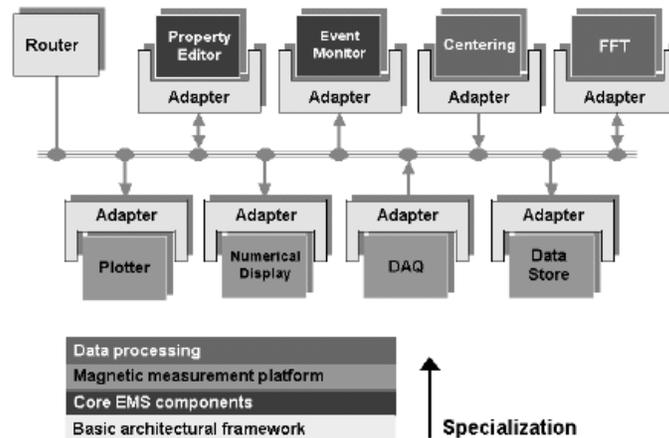
the various measurement machines connected to the motor stages, such as a digital integrator or a multimeter, will detect these changes. The data from these machines are then recorded and analyzed. Figure 7 shows a typical SSW system configuration.



**Figure 7: The Single Stretched-Wire System**

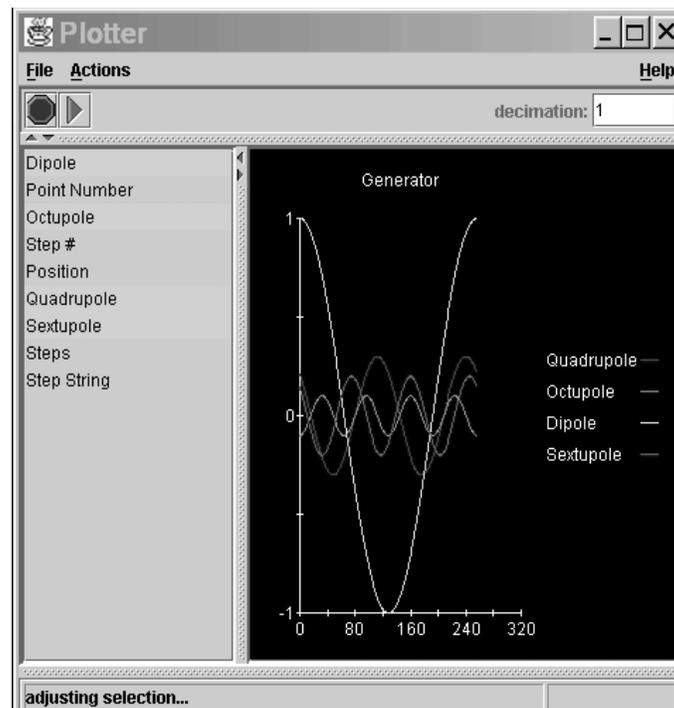
#### IV. Hardware Programming and Electromagnet Testing

For practical reasons, a standardized method of measurement and data acquisition for systems like the SSW must exist. Data discrepancies would corrupt electromagnet tests if a common method of measurement did not exist. To perform this standardization, the software engineers at Fermilab created the Extensible Measurement System (EMS).



**Figure 8: Framework for the EMS program**

The EMS is a framework with a configurable sets of components designed to acquire, process, store and retrieve data. Figure 8 shows a schematic of the EMS framework architecture. It uses many advanced algorithms to perform the appropriate magnet tests and is designed to be compatible with various platforms such as the PC, UNIX, and Linux. The EMS also has plotting software to graphically present acquired data and is capable of performing Fast Fourier Transforms (FFT) in order to properly analyze graphs with noise. Figure 9 shows an EMS graph of flux (in Volt-seconds) versus data points for different types of electromagnets.



**Figure 9:** Flux graph of various magnets from the EMS program

The most important feature of the Extensible Measurement System, however, is its flexibility and dynamic growth. Adding other testing programs will not disrupt the EMS' framework in any way; it is possible to configure the EMS so that it contains all the pertinent programs needed. Hence, the EMS can be fully customized and upgraded whenever needed.

Currently, the Technical Division is appending software to the EMS that can communicate with the motion controller through an IEEE488, or GPIB, connection. This software will reduce operation errors from the SSW system by creating a user interface that is easy to use. It is important to reduce these errors because the motion controller is responsible for moving the motor stages in the SSW system; an error in configuring and commanding the motion controller will probably lead to faulty data. This software will

also be an improvement to the current prototype-level programs that use the slower and problematic serial connection.<sup>1</sup>

The goal of this project is to implement this motion controller program for the Linux platform. This program, which will be mainly used for testing and debugging the SSW's positioning system, will be aimed to make communication with the motion controller simple and error-free by properly initializing the motion controller, automatically establishing virtual connections, and creating an interface with easily accessible commands and on-line help.

## **EXPERIMENT SETUP:**

### I. Hardware Needed

Since the Linux systems at Fermilab and the motion controller both have GPIB ports, the only extra hardware needed for this project was a GPIB cable.

### II. Code Hierarchy

The main component of this project is the complex hierarchy of computer code, consisting of C++, CGI, HTML and JavaScript. This hierarchy is separated in the four categories: the lowest level, the middle level, the higher level and the highest level. At the lowest and middle level of the hierarchy are C++ programs that hold various functions for the motion controller. The higher level consists of HTML and JavaScript, creating a website for the user to use. The highest level is made up of Fermilab's webserver, which handles this website, and the user accessing the website. Figure 10 shows this hierarchy of computer code and how the various levels interact with each other.

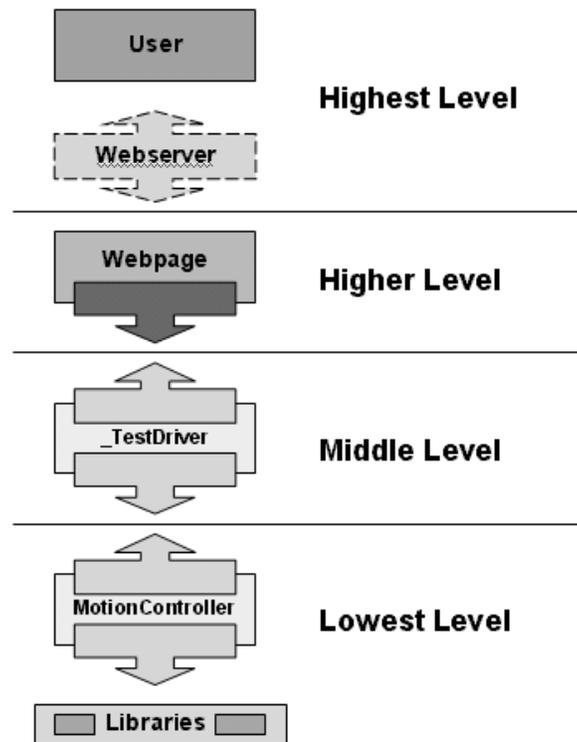
The lowest level of the hierarchy is comprised of three components: fundamental program utilities, a C++ program called MotionController.cc and a header file called MotionController.h. The program utilities are created C++ libraries used for specialized functions such as sending messages via the GPIB port, string initialization, or sending special return types. MotionController.cc then uses these special functions to create a virtual connection with the motion controller, implement functions that hold seventy-six motion controller commands and report specific errors. MotionController.h allows the

---

<sup>1</sup> Though it was originally used to communicate with the motion controller, the serial port produced numerous problems. The first problem with the serial port was its excessive parameters: there was a parameter for parity bits, echoing, etc. These parameters often produced many errors that needed a complex solution. Second, every device and platform had its own set of parameters, which made it difficult to develop cross-compatible software for different platforms.

The simplest and most efficient way to eliminate these errors was to use a different connection device. It was logical to use the GPIB connection since the GPIB ports were already being used in the Technical Division and had software developed for Linux. Also, the GPIB port does not have any of the problematic parameters that the serial port does.

functions created in MotionController.cc to be used by other programs, which in this case is the middle level of the hierarchy.



**Figure 10:** The project's code hierarchy

The middle level of the code hierarchy only consists of a C++ program called MotionController\_TestDiver.cc, which is the heart of the software. This program uses CGI code to retrieve unparsed information from the website (on the next level of the hierarchy) and extracts information from that data using many algorithms. Once the data has been unparsed, MotionController\_TestDriver.cc uses the functions from MotionController.cc (via MotionController.h) to send the proper command to the motion controller. It retrieves the status of the command and then updates the webpage through HTML manipulation.

The higher level of the hierarchy is composed of numerous integrated websites called Top.html, Bottom.html and Frames.html. Top.html is the webpage that outputs command status and Bottom.html contains all the commands needed for motion controller calibration and setup. Frames.html integrates these two websites to output in one screen. The commands in this website and are sorted by functionality and the inputs required. The website also features help buttons that invokes a popup help window to appear when clicked.

The highest level of the hierarchy contains Fermilab's webserver and a user. The webserver is a program that exists in the FNAL domain which fetches the HTML code from the network and sends it to a user's computer. The user's internet browser then interprets the HTML code and creates the webpage for the user. The user, who must be privileged and in the FNAL domain, interacts with this website and uses it to send the motion controller commands.

## **PROBLEMS ENCOUNTERED:**

### I. Newport's GPIB Port

Unfortunately, motion controller's GPIB port is underdeveloped and has interfacing problems. The most persistent interfacing problem is that the motion controller does not automatically reset communication address on the bus after a write command, which is an improper data protocol. This leads to different data retrieval methods, which stops the motion controller from interpreting commands properly. This then leads to bus contention and time-out errors.

To fix this problem, the motion controller was forced to reset its communication address by sending a read command after every write command. Sending a read command is a standard way to reset communication addresses. Hence, all commands that did not automatically read data were appended with a command-status instruction. If there was an error encountered, the command-status instruction will yield a non-zero number, which was passed to another function so that the error information gets outputted to the user.

Not only does this solution insure a write/read pair for every instruction, it also acts as an error check for all commands that are sent to the motion controller by the software.

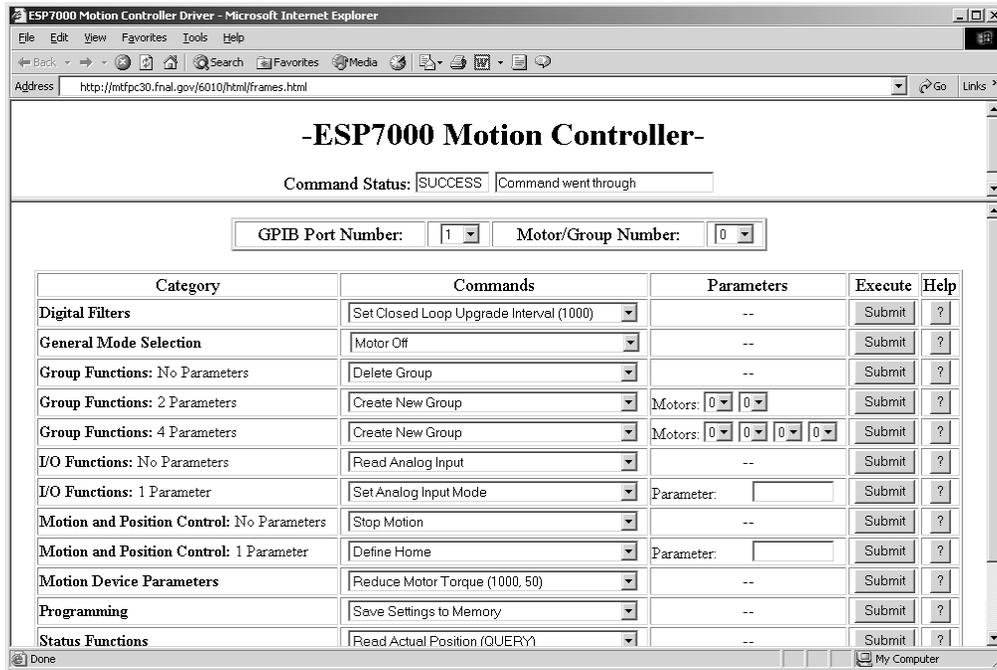
### II. GPIB Bus Noise

Another reason why the GPIB port is easy to use is because it sends and receives data as simple strings. Unfortunately, during transmission or reception, noise gets appended to these strings rendering them unrecognizable by the motion controller.

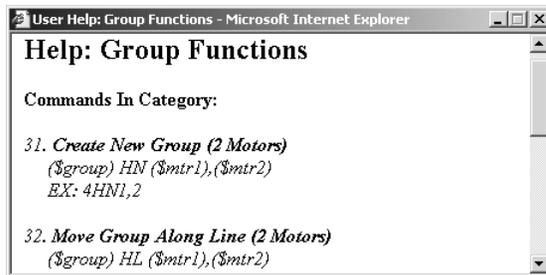
The solution to this problem was a simple coding fix, in which a terminating null (a character that indicates where the string ends) was force-placed at the end of each string. Since there are standard C++ functions that return the string length, it is easy to tell how many characters there are in a string is and where the terminating null should be placed. Though the GPIB bus noise was not eliminated with this solution, it was completely ignored by the software – which is equally effective.

**FINAL RESULT:**

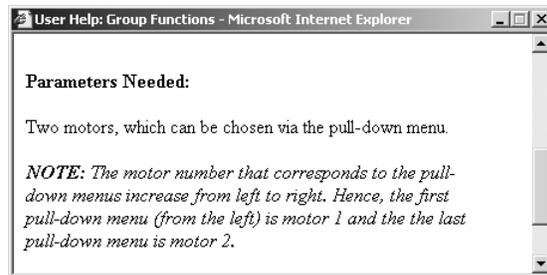
After implementing all levels of the code’s structure and fixing the unexpected problems that occurred, a simple and functional program was developed. The Motion Controller website organizes all the commands needed for the SSW system and reduces mistakes by automating initialization processes and having fixed commands that the user cannot alter. Also, the useful help windows show the command’s format and gives advice on how to properly send parameters. Figure 11 shows the website for the Motion Controller and Figures 12a and 12b show a typical help box.



**Figure 11:** The Motion Controller website



**Figure 12a (Left):** The user help window showing command formats



**Figure 12b (Right):** Extra notes from the user help window

## **CONCLUSION:**

This project was a successful attempt at creating a convenient method of communicating with the motion controller. This software conceals its own complex code hierarchy while processing numerous tasks, allowing the user to focus on performing tests and motion controls rather than the program itself.

Though this project was a success, it is only the first version of the software. There are a few more tasks that need to be accomplished before the final version of this program can be released. First, this project's code should be further optimized. Some of the algorithms can be combined and a few fixed blocks of memory allocated can be allotted dynamically to save system resources. Second, the website's visual structure can be improved by adding or changing the HTML code and using more JavaScript.

Once this project has been finalized, similar versions need to be implemented for the PC and UNIX platforms. The entire software that will be added to the EMS should ultimately integrate the Linux, PC, and UNIX implementations and automatically detect which platform is being used to further minimize setup.

Though relatively unnoticed by the physicists that accelerate particles at Fermilab, projects such as the EMS are vital to particle acceleration. Projects such as the EMS insure that Fermilab's accelerators are in perfect condition for particle colliding – a condition that should not be taken for granted. The proper operation of Fermilab's particle accelerators is the result of many hours of work from many different people; most what we understand from the universe is the result of their efforts.

## REFERENCES:

Jowett, John “Introduction, Types of Accelerator, Limits on Energy” 1996  
<<http://jowett.home.cern.ch/jowett/SummerStudents/lecture1.pdf>>

Jowett, John “The LEP Collider I: Sub-systems of a Large Accelerator” 1996  
<<http://jowett.home.cern.ch/jowett/SummerStudents/lecture5.pdf>>

J. DiMarco “Dipole and Gradient Magnet Measurements with the Single Stretch Wire System” (1996):1-8

J. DiMarco, J. Kryzwinski “MTF Single Stretched Wire System” (1996): 1-51

J. DiMarco “Alignment of IR Quads for the LHC Using a Single Stretched Wire System” (1999): 1-23

J.M. Nogiec, J. DiMarco, H. Glass, J. Sim, K. Trombly-Freytag, G. Velez, D. Walbridge  
“A Flexible and Configurable System to Test Accelerator Magnets” (2001):1-3.

J.M. Nogiec, K. Trombly-Freytag, E. Desavouret, D. Walbridge “An XML Driven Framework for Test Control and Data Analysis” (2001):1-3

## **ACKNOWLEDGEMENTS:**

My sincere thanks go to the following people:

- The SIST Committee for allowing me to work in an amazing place such as Fermilab. It was wonderful to be a part of this intellectual community, even if it was just for twelve weeks.
- Jerzy Nogiec for giving me a project that interested me, for taking time to help and listen to me, and for not treating like an intern but a member of the team. I have learned a lot about software engineering and real-world engineering from you.
- Kelley Trombly-Freytag, Dana Walbridge, Gene Desavouret, Sergey Kotelnikov, Pennie Hall and Ping Wang for their never-ending help, patience and entertainment value. I will miss our staff excursion-lunches and the interesting conversations.
- Joe DiMarco for explaining the SSW system and for sharing the Motion Controller with me over the summer. I hope my project helps the SSW system.
- Dr. Davenport for listening to my concerns, reading my paper and being a wonderful person. It was a pleasure knowing you and discussing my project with you.

## APPENDIX

1. MotionController.cc
2. MotionController.h
3. MotionController\_TestDriver.cc
4. Top.HTML
5. Bottom.HTML
6. Frames.HTML