Senior Project 2002-2003

Messiah College

PC to Scoreboard Integration

Team Scoreboard

Andrew Vogel
James Barley
Amy Hall
Timothy Hillner

Advised by Dr. Harold Underwood

May 9, 2003

Abstract:

Our project goal is to interface a scoreboard with a PC in order to simultaneously keep statistics and run a scoreboard for wrestling, volleyball, and basketball while controlling an external scoreboard similar to those found in every sporting arena. The project will also provide for text output of the statistics for a particular event. It consists of a computer program and a specially designed interface between the PC and a prototype scoreboard. Our team consists of James Barley, Amy Hall, Tim Hillner, and Andy Vogel, as well as our advisor Dr. Harold Underwood.

Table of Contents

Acknowledgements:

Earl Swope – Electrical Laboratory Technician

Earl provided valuable technical advice as well as access to the electronics lab in times of need. His helpful suggestions were greatly appreciated by the project team

Dr. Harold Underwood – Project Advisor

Dr. Underwood advised the project from start to finish. He aided us in setting measurable objectives for each week and at each stage of the project. He also provided advice on when it was time to abandon a particular course of action in favor of a more achievable one.

Dr. Don Pratt – Project Supervisor

Dr. Pratt served as the supervisor of this, and all the other, projects. If we were engineers, he was the manager we worked for. In addition to facilitating the overall project, he provided technical support and guidance.

Professor Brian Nejmeh – Database Consultant

Professor Nejmeh served as the database expert in our project. He suggested potential database programs and arrangements as well as helping to set deadlines for various stages of database creation.

Coach Mike Miller – Messiah College Women's Basketball Coach

Coach Miller provided the project team with information concerning the statistics and recording of statistics for the sport of basketball.

Coach Judi Tobias – Messiah College Volleyball Coach

Coach Tobias provided information about the statistical software package currently used by the volleyball team as well as the recorded statistics for volleyball.

1 Introduction

1.1 Description

The project has the goal of automating and streamlining the process of recording statistics and operating the scoreboard for the sports of volleyball, basketball, and wrestling. Ideally, the project will decrease the number of people required to run the scoreboard and keep statistics at athletic contests. The goal was a product that is easy to use, reliable, and efficient for the previously stated purpose. The project consists of two separate pieces which are detailed below.

Computer Program

The computer program is a simple-to-use interface that allows users to keep player/team statistics; it also automatically updates the score of the match for the current sporting event on the scoreboard. The program works in the following manner. The user inputs the player/team statistics into the computer using software that our team has developed. Depending upon the entered input, the scoreboard prototype updates accordingly. If the user input does not require a change in score, then no change in the state of the scoreboard will occur. The user can also change the time and period, allowing full control of the scoreboard prototype.

Once the user makes a statistical change within the program, output from the computer is sent to an interface connecting the PC and the scoreboard. The interface then translates the computer output signal into a readable input for the scoreboard.

An additional piece of the computer program is the statistical database. In addition to recording statistics for a single contest, the program utilizes a MySQL database to keep track of multiple seasons and teams worth of statistics in each sport.

Prototype Scoreboard

Essentially, the interface between the PC and scoreboard connects a PC with a serial output to the prototype scoreboard, which has its own input connection and signal protocols. If the PC program is analogous to the keypad on a production scoreboard controller (see attached diagram and photographs), the interface to the prototype scoreboard can be compared to the back end of the controller that translates the keypad entries into signals readable by the scoreboard itself. Physically speaking, the prototype scoreboard is small in size. That is, it is nowhere near the size of an actual scoreboard and uses typical seven segment and single LEDs for display elements. Its internal circuitry of the scoreboard comprises integrated circuits for various purposes as well as a microcontroller to control the ICs and convert the incoming serial data to a usable form for the other ICs. The ICs in question are designed to translate a digital input into a signal that can control a series of seven segment LEDs. This allows the single serial data stream to control multiple display elements. The main advantage gained through this combination of prototype scoreboard and computer program is that it obviates the need for pencil and paper recording of statistics.

The objectives of the project were as follows:

1. To design a computer program that will enable one to easily record the relevant statistics for wrestling, volleyball, and basketball. The relevant statistics will be determined through interviews with the coaches of the three sports.
2. To develop a testing procedure for the program, interface, and scoreboard signals.
3. To create a technical manual/user's guide for the PC/Scoreboard system.

4. To achieve a degree of reliability that ensures error free reliability for an entire year of athletic contests and saves the data in the program in the unlikely event of power outage or computer malfunction.
5. To complete the project within the allotted budget of $500

1.2 Literature Review

Nevco Scoreboards is the manufacturer of the current scoreboard system in place in Brubaker Auditorium. It is controlled by a microprocessor-based keypad that has special switches to start two separate timers. In its current state, it has a memory to guard against accidental power outages without losing scoreboard information. It does not keep track of individual statistics beyond that which is displayed on the scoreboard itself. Information concerning the Nevco 2550-D scoreboard found in Brubaker Auditorium can be found both in the user's manual as well as the company website www.nevcoscoreboards.com.

Santech, Inc. builds electronic scoreboards that feature the ability to control the scoreboard through an interface box and a computer program. It is unknown at this time if the computer program simply controls the scoreboard or if it can also keep statistics. The project team is in the process of gathering more information about the computer aspect of Santech scoreboards. More information is available at www.scoreboards.com/home.htm.

Automated Display Systems manufactures a line of wireless scoreboards. The main selling feature of their LED scoreboards is the wireless communication aspect of them. They do not appear to be computer controlled, nor do they record statistics. More information can be found at www.adsled.com.

Microsoft is in the process of designing a Tablet PC that will allow for a better connection between the network and the individual. Microsoft wants to revolutionize the actual interface between computer and user. The main objective of this new technology is to be able to access information, such as e-mail and other data, anywhere, regardless of hardware platform, wireless connection, or data medium.

The Tablet PC can be thought of as resembling a laptop-sized Palm Pilot. It will allow the user to draw directly onto the screen, by handwriting sentences or tapping letters on a virtual keyboard. Benefits of the Tablet PC include writing handwritten notes in e-mails, clipping a web page for e-mail, or writing notes in the margins of an electronic book.[1]

The University of Washington's Human Interface Technology (HIT) Laboratory has been involved in the development of advanced computer and virtual interfaces. Currently, the lab's research has included aspects pertaining to various interface venues, notably voice recognition, gestural interfaces, augmented reality, wearable computers, and digital entertainment. One of HIT lab's objective is to sharpen the focus on creating natural interfaces to computers.

Tom Furness director of the HIT Lab comments, "With advances such as Intel's advanced processors, there are more cycles to support the user interface and do speech recognition, gestural interfaces, image processing and emotional determination. You mainly need processing power for this, and we're now getting to the point where it does exist."[2]

StatsNOW is a statistical software package that allows you to record real time stats at courtside or enter data at a later time. This software package is unique in that it is designed for Windows CE & Windows 95/NT, which allows it to be run on a PDA or a PC. More information is available at www.bbhighway.com/talk/coach%20Library/reviews/Software/statsnow_review.asp

Hy-Tek's Meet Manager is a software program for Swim meets. Hy-Tek's Scoreboard interface connects the Hy-Tek Meet Manager for Windows computer directly to most alphanumeric scoreboard systems. Meet Manager sends both start lists (swimmers' names) and results right to the scoreboard display. The Hy-Tek Meet Manager computer connects directly to the scoreboard computer with a standard serial cable. More information is available at www.hy-tekltd.com/swim/mm/

The cumulative sum of these sources shows that no advertised product currently solves the problem as stated by the project team. While a combination of platforms may allow for computerized statistical recording or computerized scoreboard control for various sports, but no product combines the two along with database capabilities.

1.3 Solution

Solving the problem set forth by the project team involved a variety of design decisions. Several of these are given in Section 1.1, but will be described in greater detail here. Components of the systems described in Section 1.2 are used, but the design of the project is unique to the project team and integrates the previously described parts of the project in an innovative fashion. The format of the prototype scoreboard is based on the Nevco 2550-D scoreboard that is currently used Brubaker Auditorium on the Messiah College campus (figure 1 of the appendix). The PC program is inspired by standard statistical programs, but its design is entirely unique to allow for timely entry of statistics in order to keep up with the flow of an athletic contest.

Software User Interface

The user interface was designed to tackle the problem of keeping track of a multitude of statistics for each sport as the event unfolds in real time. This is likely one of the limiting factors that has prevented a product similar to that developed by the project team from being introduced previously. The most similar product found in research by the project team is the StatsNOW software package explained in the Section 1.2. As previously described, this package allows for real-time input of statistics into the program. Though it has no capability for data output to a scoreboard, the concept of real-time input is a common bond between StatsNOW and the graphical user interface developed for the project. The Visual Basic interface created by the project team is designed to allow convenient and efficient entry of statistical data. One can clearly see the need for an efficient user interface due to the fast pace of the three sports addressed by the project. The final design of the graphical user interface for basketball can be seen in figure 2 in the, for volleyball in figure 3, and for wrestling in figure 4 of the appendix. The choice of Visual Basic 6 as a programming language was made due to the ease of creating Microsoft Windows based applications in Visual Basic. As one could reasonably ascertain, the choices made in the design and programming language of the user interface are arbitrary and determined solely by the project team and the statistics necessary in each sport. A different design could have been used, but the project team chose to implement the current design.

In addition to the statistical entry forms, the graphical user interface employs a series of data handling forms. These forms communicate with the database, which is invisible to the user. Each sport features the options to Load, Create, Edit, and Delete a season. Additionally, this menu allows the user to proceed directly to the statistical data entry and scoreboard control screens without being concerned about a specific team or season. Screenshots of these forms are available in the user manual.

One of the more time-consuming pieces of the software design involved the extensive error checking necessary in any software package with a strong emphasis on data entry. The communication between the graphical user interface and the database requires that the user only refer to data that is already in the database for the purpose of loading, deleting, and editing data. Also, the creation of data requires checking to see if data by the desired name already exists within the database. The error checking features ensure that the database and data entry forms will be user friendly and intuitive.

Finally, the software graphical user interface allows the user to print a detailed listing of the statistics for a given event that is suitable for submission to the NCAA as a record of the

event. The format of this document for each sport was determined by observation of the results for Messiah College athletics events as posted on the Messiah College Athletics website (www.messiah.edu/athletics). Examples of the generated text file for each sport can be found in the appendix of this report.

Database

The database forms the backbone of the data collection capabilities of the graphical user interface. It handles the storage and retrieval of data for each of the three sports. However, the user of the program requires no knowledge of the database beyond that required to install the program. The data forms discussed previously communicate with the database for the user. This greatly simplifies the process of data management by not requiring the user to use the database commands unique to the specific database used.

The database application chosen for use in the project is MySQL. MySQL is an open source program that can handle the relatively large amounts of data required by the project. An advantage of open source software in the scope of this project is that the database portion of the project has no cost for purchase or licensing of the software. This helped to keep the development cost of the project to a minimum. Though the database application used in the project is not a Microsoft product, a fairly straightforward procedure for communicating with the database from within Visual Basic exists.

Within the database, the capability exists to create teams, players, and seasons for each of the sports. Modification and deleting of these entries can also be accomplished. Each set of data is distinguishable from the others based on such values as team name, year, player name, player number, et al.

Hardware

For the purposes of the project and designed prototype, all of the hardware is confined to the prototype scoreboard assembly. Serial (RS-232) data comes from the PC running the software developed by the project team and enters the prototype scoreboard, where it is converted and interpreted for output to the various LEDs making up the display of the prototype scoreboard.

The transfer of data from a graphical user interface to an external display formed the basis of hardware design. Several obstacles stood in the way of a successful transfer. Most basically, the signal from the PC serial port is a digital stream of data while the LEDs on the display function when the proper pins are given the proper voltages. The basic issue to solve is how to turn the serial data stream into a recognizable pattern of voltages for the LEDs. Solving this problem required the use of several integrated circuits as well as a microprocessor. The microprocessor used in the current implementation is a Motorola 68HC11. It features extensive ability to communicate with peripheral devices as well as a flexible instruction set. Also, several other ICs were used in the design. The most important of these is the Maxim 7221 chip. Each of these ICs has the capability to turn a serial data stream into the proper control signal for up to eight seven segment LEDs. Additionally, several auxiliary ICs are used, such as the Maxim 692A, which monitors the power of the microprocessor, and the Maxim 232, which converts an RS-232 signal to the TTL signal required by the 68HC11.

Once components were chosen, integrating them with the software provided the next challenge. The software sends RS-232 data out of its serial port. It is then converted by the Maxim 232 chip and sent to the microprocessor. The microprocessor readies it for the Maxim 7221 and sends it to the three 7221 chips on the same circuit board. Finally, the 7221 chips activate the LEDs in accordance with the data sent from the graphical user interface. The 7221 features pins that activate the chosen seven segment LED along with other pins that control each of the segments on the LED.

Analyzing Alternatives

Obviously, the project team could have made different choices in developing a solution. However the project team attempted to make design decisions base on what is the most cost effective, efficient, and practical. The prototype scoreboard, graphical user interface, and database all had characteristics that could have been altered under a different design scheme.

The prototype scoreboard had one designed purpose and function, but many different designs could have resulted in a product performing the same task. For example, different components could have been used to handle the control of the LEDs. However, the components used by the project team were selected for ease of use and low or no cost. Additionally, the division of the scoreboard into two circuit boards reflected the opinion that a single circuit board containing both the integrated circuits and LEDs would be cumbersome to fabricate. A separate hardware issue is the communication between the PC and scoreboard. USB communication could have been used in place of RS-232, but the added complexity of the required code made that alternative unattractive.

The database also had room for alternatives. The alternatives in the database dealt not so much with the actual design of the database as they did with the database application used in the project. Microsoft Access and SQL Server were both considered in the design process. The application used in the project, MySQL, has a major advantage over each of the alternatives. It is a much larger and more flexible application than Access and has no cost in comparison to the prohibitive cost of SQL Server.

The graphical user interface is perhaps the area of the project featuring the most flexibility in design. Any number of different implementations of statistical recording could have been used. The current design for basketball features a separate form for each team's statistics. An alternative design could place the buttons for the statistics of both teams on one form with a listing of the statistics for both teams on a separate form. The choice was made to separate the forms by team rather than function in order to give the user a visual representation of each team in order to allow for editing of statistics in case of an error.

2 Design Process

Traversing from an idea of PC/scoreboard integration to a finished combination of graphical user interface, statistical database, and prototype scoreboard was not always a direct route. Design decisions were made for reasons such as efficiency, cost, and ease of fabrication. The experience, or lack thereof, of the project team contributed to some of the choices made in design. Software design and hardware fabrication are skills that improve with time and experience. Therefore, the process of developing a solution provided many opportunities for learning in all areas of the project. Also, the process of developing a solution resulted in following some dead ends that, while not necessarily advancing the project toward the final solution, resulted in the gain of knowledge that aided in later pieces of the solution.

Reverse Engineering

The original plan was to do extensive reverse engineering in order to develop a solution that allows the user to keep statistics while updating the Nevco 2550-D scoreboard currently installed in Brubaker Auditorium on the Messiah College campus. Ideally, the current control box (figure 5 of the appendix) would be eliminated and replaced by a PC and any additional hardware designed by the project team. Essentially, the main design challenge would be to recreate the various signals coming out of the control box. After observing the output signal and speaking with a representative from Nevco, it was determined that that input signal was a modulated 250 kHz sine wave transmitted on coaxial cable. Capturing and decoding this signal would have proved difficult, if not impossible. The only possibility for the team to decode the signal lay in acquiring some information about the communication between the control box and scoreboard from the manufacturer. However, discussion with a representative from Nevco resulted in discovering that the control box signal was proprietary to Nevco. Therefore, the project team elected to abandon recreating the control box output and sought an alternate method of scoreboard control.

The option then considered by the project team still involved the scoreboard control box. The keypad of the control box (figure 6 of the appendix) sends a signal to the rest of the control box (figure 7 of the appendix) which creates the signal sent to the scoreboard. Upon investigation, it was determined that this signal was a serial digital signal with no modulation at a much lower frequency than the 250 kHz signal coming out of the control box. It was posited that recreating this signal would still allow the team to complete the original goal of interfacing the Nevco scoreboard with a PC. The only difference would be that the control box would be modified, rather than eliminated. The finished product would be a combination of Nevco and newly designed hardware. Significant time and energy was invested in developing a method of capturing the signal for future decoding and recreation. The Motorola 68HC11 microcontroller was used to capture the signal. The signal was a periodic bit stream with what appeared to be low-true logic. While the project team was able to develop a method for capturing portions of the signal, the signal itself was constantly changing. Even with no input from the keypad and the clock not running, the data changed constantly. Examination of the serial data revealed a large number of possible combinations. The amount of time and effort that would have been required to capture and decode the signal for each press of the keypad would have been prohibitive to completion of the project as well as giving no guarantee of ultimate success. Therefore, the project team decided to forego using the Nevco hardware at all. Rather, the team would prove

the concept of PC to scoreboard integration by developing a prototype scoreboard with a communication protocol completely under control of the designers. While this approach has its own share of design challenges, the elimination of the extensive reverse engineering necessary under the initial plan allows for a greater array of design choices and increased the likelihood of the project team achieving its remaining goals.

Final Design

Once the decision to develop a prototype scoreboard was finalized, each of the three main pieces of the project. The user interface, database, and prototype scoreboard each presented a unique challenge. The problem of interfacing all three components of the project together stretched the creativity of the project team the most. The user sees and knows only the graphical user interface, yet expects the database and scoreboard to respond accordingly. Therefore, these two aspects of the project must operate on a mostly automatic basis. In order for this automatic operation to occur, the pieces must seamlessly integrate with one another. Additionally, the user manual authored by the project team must enable the user to install, set up, and use the final product.

User Interface

The user interface was the first part of the project designed, so it had the most flexibility in design. The rest of the project was influenced by the design of the user interface in order to integrate all parts of the project together. Creating a user interface that allowed for efficient and convenient entry of statistics within the flow of the athletic contest. Many statistical programs in use currently are used after the game has finished. A statistical worker will enter handwritten statistics into a computer program for storage in a database. Therefore, the program used by this person need not place a premium on the speed an efficiency with which data is entered. However, for this project, speed was at a premium. A creative solution was required in order to allow for the user to keep up with the pace of a contest in the chosen sport. Several features of the user interface for each sport enhance the efficiency of use. Each sport features a unique button for each statistic. Additionally, a series of radio buttons selects the player to whom the chosen statistic will be applied. For the sport of wrestling, this method provides the necessary efficiency due to the presence of only two athletes on the mat at any given time as well as the fact that scoring in wrestling occurs less regularly than in basketball or volleyball. For those two sports, the project team has implemented an additional feature to speed data entry. The statistical entry forms for basketball and volleyball feature a text entry box in which the user may enter the number of the player for whom he wishes to enter a statistic and then clicks on the button corresponding to the proper statistic. The program recognized which number was typed and updates the proper value. Then, the number entry text box clears itself in anticipation of the next entry.

Error Checking

In order for the application for function efficiently and simplistically, there must some form of error checking to keep the user from entering invalid or incorrect data and crashing the application. Error checking can be done on both the application side and the database side of the

software. However, since the application can read data from the database directly, all error checking for the software is accomplished on the application side. The easiest way to manage errors is to display a popup message to the user saying that he/she has done something wrong. Most error checking will tell the user what went wrong while other error checking will not notify the user of errors but just keep the system stable.

Error checking was used in each data manipulation form to keep the user from inputting invalid information. Drop down menus were used to create easy to read and easy to follow forms that restrict the user to select data that already exists in the database. This kept the user from searching for information that did not exist. If the user somehow was able to manipulate the drop down menus in such a way that the data they represented was invalid, the application would not update the database and return a message stating which data the user supplied was invalid. If a user was creating an entry for the database and needed to type information in manually without the assistance of a drop down menu, the application would search the database first to make sure what the user typed in did not already exist. If the user did supply duplicate information, his operation would be stopped and a message would be returned stating the problem.

If a user wanted to export a match to the database the application would first search for all the players in the database and find each player table to store his/her own statistics in. If the tables did no exist, the application would not export the data and would return a message indicating the problem to the user. If the user attempted to import data from the database and supplied invalid match or team information the application would not allow the importing action and would return a message stating the problem.

Other error checking needed to be done on only the application itself. The application will not keep track of negative statistics or scores. If a user attempts this, a message will be returned stating the problem. Also there are limits to some statistics and scores and games that the application should stay within. For example, a volleyball match can never have more than five games and the fifth game, if played, must only be played until a team reaches the score of 15 and is in the lead by two. All other games are played until a team reaches 30. To solve problems such as this the application was developed to contain the rules of the game and does not allow users to attempt actions that could never arise in an actual game. Errors generated by this type of action do not return messages; they just do not allow the user to do what he/she wants to do.

Database

The initial design of the database involved using 6 different tables. Each table would keep track of different data. The season table would keep track for season year and location, the team table would keep track of the team name, the player table would keep track of the player name, number and position, the match and game tables would keep track of match and game scores and the stats table would keep track of statistics per match for each player in the database. In order for data to be stored into the database, a primary key must be used. Usually a primary key is an auto-generated number that is unique for each entry in a table. However, using auto-generated primary keys would require the user the write down that number for season, teams, players, etc., which would defeat the purpose of the application. Therefore instead of using auto-generated keys, unique attributes of each table would be used. This caused problems because there could exist a team with the same name, but in different seasons, or players with the same number but on different teams. This issue was resolved with a different database design.

The final database design included one main season table that contained the season year, location, team name, player name, number, and position. The primary key of this table was composed of the season year, season location, team name, and player number. These four attributes together would be unique for every single player on any team in any season. With each new player that was added to the database, a new player table would be created. Since tables require a unique name, the name of the table was derived from the primary key of the new entry in the season table. Statistics for individual players would then be stored in their own player table according to match date. Refer to the data dictionary and ER diagram in the appendix for more information.

Prototype Scoreboard

The final piece of the project is the prototype scoreboard. The finished product is a combination of LEDs and integrated circuits. The scoreboard features 23 separate elements that must be controlled simultaneously. These elements include seven segment LEDs and single multicolor LEDs. Controlling each of the 23 elements with a serial data signal proved to be a significant design challenge. A rough initial solution called for a complicated series of shift registers and demultiplexers to ensure that the proper signal was sent to each digit or LED. Fortunately, before much effort was invested in developing such a solution, the project team encountered the Maxim 7221 serially interfaced, eight digit LED driver. This integrated circuit performed many of the tasks described above. By sending a two byte signal to the 24 pin chip, one can control up to eight digits. The first byte sent determines which LED is lit, and the second byte determines what digit will appear on the LED. The 7221 has several other features which can be seen found in the datasheet from Maxim. The integrated circuit is able to control eight separate digits even though it has only 24 pins by constantly refreshing the digits. Once a data is sent to light a certain digit, the digit holds that value until it is changed by another two byte signal. The chip refreshes constantly by cycling through all of the lit digits a rate faster than the human eye can distinguish. Therefore, at any given time, only one digit is on, but it appears to the viewer that all digits are constantly on. Using three 7221 chips in series allows control of the 23 separate display elements from a single data line. The chip supports multiple data input formats, but the project team elected to us the Motorola Serial Peripheral Interface format in order that the Motorola 68HC11 could be used to communicate with both the PC and the Maxim 7221.

The Motorola 68HC11 is a microcontroller with serial communication abilities. While it has many other functions, the project team made only used it for serial communication. The specific version of the microcontroller used was the 68HC711E9. It is a 52 pin, socket mounted chip with both EEPROM and EPROM memory for the loading of programs. The project team made use of the Serial Communications Interface and the Serial Peripheral Interface to accomplish the communication necessary for proper operation.

Serial Communication Interface

PC-to-Serial RS-232

The PC-to-Serial interface involves the PC, the operating system, and the serial communications port on the back of the PC. Visual Basic includes a component called the communications control which allows for easy access and configuration of the serial communications port on the computer. The communications control provides an interface to a standard set of communications commands. It allows you to establish a connection to a serial port, connect to another communications device, issue commands, exchange data, and monitor and respond to various events and errors that may be encountered during a serial communication.

The communications control utilizes a serial process. As data is sent through the serial port from the CPU, byte values are converted to serial bits. On the operating system's side, Microsoft Windows uses a communications driver, comm.drv, to send and receive data using standard windows API functions. When one uses the communications control, the user is issuing API functions, which are interpreted by comm.drv and passed to the device driver.

First, the project team wrote a small program that would output data that the user inputted into textboxes on the PC. Next, an oscilloscope was wired to the serial port in order to study the actual signal that the communications port was creating. This revealed that the communications port uses an RS-232 signal which is inverse logic with amplitude of +12 or –12 volts. When no data is being sent, the line sits at an idle level of -12 volts. Each data packet included a start bit at the beginning of the data and a stop bit at the end. The start bit initiates the port to indicate that a data would be transmitted. After the start bit, the 8 data bits follow with the highest order bit first; these bits are then followed by a stop bit which is identical to an idle bit.

The communications control allows for different packet sizes and either one or two stop bits. Also, the parity bit can only be removed when an 8 bit data word is used. After all the data packets have been transmitted, the line returns to its idle state. The data rate may also be selected from a list of standard serial data rates. Upon comparison of the available bit rates and the specifications of the microcontroller, a bit rat of 9600 bps was chosen.

Because three Maxim 7221 chips were needed, six separate bytes were sent each time the entire scoreboard was refreshed. The first byte of each pair corresponded to the address of an LED, and the second byte held information pertaining to what each digit would display. A Visual Basic function took care of transmitting the proper data in six byte increments. Though the communication port has nine pins, for the purposes of this project, only two pins were needed. One pin transmitted the data, while another served as circuit ground.

RS-232 signal to MAX232 Chip-TTL output

Upon studying the Motorola 68HC11 microcontroller manual it was determined that the serial input line on the chip requires a TTL digital serial format. The RS-232 data format had to be converted into the TTL 0 to 5 volt normal logic format. Consultation with the project advisor revealed that the 68HC11 prototyping boards use a MAX232 IC to do the required conversion. This integrated circuit simply and efficiently makes the conversion.

SCI Receive on 68HC11

After consulting the 68HC11 manual, the project team decided that the Serial communications interface (SCI) would be the best option for receiving the data from the PC. SCI is a full-duplex UART-type asynchronous system that uses standard non-return-to-zero format, which is one start bit, 8 or 9 data bits and a stop bit. An on-chip baud rate generator derives standard baud-rate frequencies from the MCU oscillator. The 68HC11 resynchronizes the receiver bit clock on all 1-to-0 transitions in the bit stream rather than just at the beginning of the start bit time, therefore synchronizing to the PC is not necessary.

Before coding could begin, the project team needed to determine what conditions were required in each of the five SCI control registers. SCI uses the port D pins 0 (RxD) and 1 (TxD) for receiving and transmitting. SCI logic takes control of the pin buffers for the associated port D pins in order to ensure full control over the pins.

In order to achieve the necessary 9600 baud rate on the processor, the prescaler bits were loaded with the number $30. The SCCR2 register contained the SCI enable bit, which was set to $34; this enables the following: the SCI receiver, receive interrupt, and the idle line interrupt systems. The SCCR1 register controls the wake on idle bit and selects what data word size, either 8 or 9 bits. In order to select wake on idle and 8-bit word size, the SCCR1 register must be clear. The SCSR register contains the RDRF flag which is set when the receive data register is full. The SCDR register is where the data is stored after the receiver has completed a reception of the data-packet.

Once set up of the SCI registers was complete, coding could begin. Due to the nature of our incoming signal a counter was employed that would account for the groups of six data packets needed for the display drivers. Each packet would then be received and retrieved from the SCDR and pushed onto the stack for the data to be converted and transmitted to the scoreboard. After each data packet was received the RDRF flag was cleared in order for the SCI to receive the next data packet and then the counter was decremented. When finished receiving the data packet group, the SCI receiver is disabled by clearing the SCCR2 register. The SCI is not enabled again until after the data packet group is sent to the scoreboard. This process is then run forever in an infinite loop in order to constantly update the scoreboard as data is received.

Serial Peripheral Interface

Once the SCI has received the data from the PC, the Serial Peripheral Interface (SPI) of the microcontroller takes responsibility for sending the data to the Maxim 7221 and on the display. The SPI sends data in a format readable by the 7221 in single byte messages. Just as with the SCI, the SPI sends six bytes at a time in order to cover the address and data bytes for each 7221 chip.

To use the SPI as output, bits 3, 4, and 5 of DDRD (Data Direction Register, Port D) must be set for output. Bit 3 of Port D is the PD3/MOSI pin. This pin will contain the signal that is sent to the MAX7221 chips. Bit 4 of Port D is the PD4/SCK pin. This pin contains the clock that controls the MAX7221 chips. The clock timing comes from the value specified in the SPCR (SPI Control Register) in bits 1 (SPR1) and 0 (SPR0). A table is shown below with the clock speeds.

SPR1	SPR0	E Clock Divided By
0	0	2
0	1	4
1	0	16
1	1	32

Finally, bit 5 of Port D is the PD5/SS pin. This pin will trigger the MAX7221 and tell it when to output its saved bits to the LEDs. A one on this pin tells the chip to output its data, a zero tells the chip to hold on to its data and continue gathering data.

The SPCR sets up the 68HC11 to control the SPI according to the users needs. The SPCR is located at address $1028 and contains the following bits:

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPRI	SPR0
Reset:	0	0	0	0	0	1	U	U

SPIE is the SPI Interrupt Enable bit. A one turns the interrupt service on, a zero turns it off. Since we did not use the SPI Interrupt Service, we set this bit to be zero. The SPE is the SPI Enable bit. A one enables the SPI allowing the user to either read or write to the SPDR. DWOM is the Port D Wired-OR Mode Select Bit. This bit was set to a zero so the Port D outputs would act in a push-pull fashion, sending constant signals to the MAX7221 chip. MSTR is the Master/Slave Mode Select Bit. Since out 68HC11 was not being controlled or sent signals from another IC, this bit was set to one to configure the SPI as a master. CPOL is the Clock Polarity Bit. This bit contains a zero because the MAX7221 chip gathers data when the clock driving it is set high. When the SCK is low, the MAX7221 chip is idle and not saving data. CPHA is the Clock Phase Select Bit. This bit was set to zero because the MAX7221 saves data on the rising edge of the clock. This bit sets the rising edge of the clock in the middle of each bit coming from the MOSI pin. SPR1 and SPR0 have already been discussed.

Once the DDRD and the SPCR was initialized, data can be sent via Port D to the MAX7221 chip. First, the SS pin is set low to allow the MAX7221 chip to gather information. Since the SCI had already stored six bytes of information on the stack, the program pulls the information from the stack one byte at a time. Each time a byte is pulled off the stack, it is written into the SPI Data Register (SPDR). Once the data is written to this register it is sent out through Port D via the MOSI pin or pin PD3. The data is shifted out one bit at a time. A flag is set in the SPI Status Register (SPSR) when the complete byte has been outputted to Port D. The program polls this flag (the SPIF) and waits in a loop until that flag is set. Once this flag is set, the program pulls the next number off of the stack and goes through the same process again until it has sent out six bytes, or forty-eight bits. Once all six bytes have been outputted, the SS pin is set high and the MAX7221 chip outputs its data to the LED displays.

Through the previously described process, data can originate with the graphical user interface in the PC and end up on the display in the proper locations. In essence, the virtual scoreboard seen in the basketball and volleyball user interfaces and the prototype scoreboard display the same information due to the work of the communication port, SCI, SPI, and Maxim 7221. However, in order for the user to make use of this product, he must be trained in its use. The user manual provides one method of training.

User Manual

The user manual designed by the project team provides step by step instructions for using the Scoreboard Statistical Program. Some instructions pertain to creating, deleting, and modifying various sport seasons, teams, and players. Other areas in the manual describe the process of keeping score, recording statistics, and controlling the clock and its features.

Other than providing step by step instructions for each process, a visual aid is displayed to show the user how to utilize each task. This capability allows for the more visual learners to easily comprehend operation of the program.

For the three sports, a diagram of each graphical user interface is given and labeled to show the function of each button. In addition, examples of data entry forms for loading, creating, editing, and deleting a season are provided as a guide for the user to follow.

The user manual also offers the installation procedure for the software programs needed to run the Scoreboard Statistical Program. Again, written instructions are given as well as illustrations pertaining to the main features of the installation process.

3 Implementation

3.1 Construction

Once the design of the prototype scoreboard was finalized, construction began. Inexperience on the part of the project team led to several dead ends in construction, yet also resulted in the development of creative solutions to overcome limitations due to inexperience and the facilities available.

Dual Layer Circuit Board

One of the first difficulties encountered involved the construction of printed circuit boards for both the display elements and the integrated circuits. An original plan involved placing all hardware onto a single printed circuit board. Examination of a potential design revealed several pitfalls that the project team wished to avoid. First, the design involved placing components on both sides of a printed circuit board, which complicates the design. Also, a multiple layer circuit board would be required in order to keep a majority of the traces on the surface of the board, as opposed to above the surface as jumper wires. The printed circuit board fabrication facilities available to the project team allowed for a maximum of two layers to a circuit board. While dual layer circuit boards could be fabricated, the difficulty of soldering components on both sides of the board as well as dealing with the vias in between layers made this option undesirable. As an alternative, the project team determined that using two single layer circuit boards connected by ribbon cables would allow for easier construction. Figure 8 of the appendix shows the dual circuit board layout.

One of the circuit boards holds the integrated circuits and data receiving hardware. This can be seen as Printed Circuit Board 2 in figure 8, and a block diagram showing the significant circuit elements can be seen in the figure 9. This circuit board consists of the integrated circuits as well as miscellaneous hardware such as the power switch. The other circuit board is referred to at Printed Circuit Board 1 in figure 8 and contains all of the display elements as well as ribbon cable headers to connect the board to the other circuit board. A photograph of the display board can be seen in figure 10 of the appendix. The figure shows a display typical of a basketball game because basketball makes use of all the display elements, while wrestling and volleyball omit several.

PCB Layout

Even though the team employed the dual circuit board layout, the large number of traces did not allow the elimination of all jumper wires. In fact, a comparatively large number of wires were required. The project team decided that this large number of wires was acceptable for a prototype scoreboard. In the event of a full size production model, more sophisticated methods of printed circuit board production could be used to eliminate these jumper wires. In construction of the prototype, the circuit board layout tool Ultiboard was used to get an initial layout of the board. Since only one layer was used, the software was unable to complete the design without jumper wires. The project team then etched a board with as many traces as Ultiboard could draw and then used jumper wires to connect the rest of the pins (figure 11 of the appendix). After eliminating any cold solder joints, the display worked just as well as if more

complicated circuit board fabrication had been completed. An admitted pitfall in this design is reliability, but the prototyping aspect of the project allows for some reliability issues that would be solved in a final product. A prototype exists mainly to prove functionality of design.

Construction Issues

As stated previously, the goal was to create two printed circuit boards in order to accommodate all the needed components. However, complications arose in construction. The circuit board containing the microcontroller and other integrated circuits was designed and built using similar techniques to those used on the display board. However, the ribbon cable/header system designed to allow the boards to communicate failed to work properly. The cause of this malfunction has not been thoroughly investigated, but it is believed that either the headers are too short, or the ribbon cables used did not form a good connection with the headers. This has been hypothesized because plugging jumper wires into the ribbon cable worked without issues. In order to have successful operation, the components were placed onto a prototyping breadboard as seen in figure 12 of the appendix. The project team believes the design to be sound, and a full sized scoreboard could use the same method with better components.

3.2 Operation

Hardware

The main testing on the hardware portion of the project involved running the software and ensuring that the proper light came on when they should. Also, upon completion of the display hardware, a connectivity test was performed to ensure that the path from header to digit was complete for every trace. Also, the testing made sure that no pins were connected to each other incorrectly. For all of the hardware, standard practices were used to make sure the construction was done correctly. Most of the testing, however, for the project focused on the software because the hardware only existed to display data sent from the PC and microcontroller.

Software

In order to test the software application, a test plan document was developed to create test cases to properly assess the software application. Once test cases were developed, a test results document was generated to evaluate the functionality of the database and application. Each test case revolves around a requirement listed in the Software Requirements Specification Document. In order to simplify each document, only the sport of volleyball was extensively tested and documented. The similarity of the database for each sport precludes the need to extensively test all three sports to the same degree. All software testing documentation can be found in the appendix. Additionally, the error checking discussed in Section 2 was tested exhaustively during its creation.

4 Schedule

Under the originally proposed schedule, many of the objectives were completed in the time allotted. One aspect that must be clarified is the lateness of the beginning of design for the prototype scoreboard. The reason for this is the slight objective change made by the project team. Until the second semester of the school year, the decision to create a prototype scoreboard had not yet been made. Therefore, design did not begin until later than one would anticipate. The schedule for the prototype as well as the rest of the project can be seen in the Gantt Chart in the appendix.

5 Budget

The Messiah College Engineering Department imposed a limit of $500.00 on the budget of the project. Therefore, creative sources of parts and components were needed. A major portion of the project was completed through the acquisition of many free samples of LEDs and integrated circuits. The main costs incident to the project involved the four digit LEDs and the Motorola 68HC11 processor. Additionally, printing costs for the report played a role in the project cost. In the end, the project used approximately half of the allotted budget. A detailed breakdown of project costs can be seen in a chart in the appendix.

While this project cost approximately half its maximum budget, expanding the prototype into a full sized operational scoreboard would obviously increase the cost substantially. The project team did no research into the cost of a full size scoreboard, instead focusing on the operation of the prototype. One can safely assume, however, that a full size scoreboard would not fit within the $500.00 budget of the current project.

6 Conclusions

The original goal was to interface a PC with the scoreboard in Brubaker Auditorium. By the conclusion of the project, the goal completed by the project team was to interface a PC statistical program with a scoreboard of the project team's design. A photograph of this final design can be seen in figures 13 and 14 of the appendix. This represents a success considering the effort and ingenuity required to take an RS-232 signal from a statistical program, convert it to TTL, translate the TTL signal to the format required by the output chips, and send that output to a display in a predictable and repeatable fashion. Most objectives were met to an acceptable degree. The one goal that the project team did not accomplish was to create a system that would be usable by the athletic department of Messiah College. Rather, the concept of scoreboard control was proved viable, and the project team was able to produce a prototype system on which a production level system could be based.

The objectives of the team were stated in the introduction. These objectives were met with varying degrees of success as follows:

1. To design a computer program that will enable one to easily record the relevant statistics for wrestling, volleyball, and basketball. The relevant statistics will be determined through interviews with the coaches of the three sports.

The graphical user interface efficiently and accurately keeps track of the statistics for each of the three sports dealt with in the project. Interaction with the coaches provided insight into which statistics were important to keep.

2. To develop a testing procedure for the program, interface, and scoreboard signals.

The database and user interface underwent significant testing. The prototype scoreboard also underwent testing. The testing procedures have been described in this report. Unfortunately, no testing was done in an actual game setting.

3. To create a technical manual/user's guide for the PC/Scoreboard system.

The project team created a full-color, indexed user manual that details installation and operation of the software for each of the three sports.

4. To achieve a degree of reliability that ensures error free reliability for an entire year of athletic contests and saves the data in the program in the unlikely event of power outage or computer malfunction.

The software is backed up by the battery of the laptop. Periodic saving of statistics would be a desirable expansion of the user interface/database.

5. To complete the project within the allotted budget of $500.

As described in Section 5, the team completed the project by using approximately half the allotted $500.00.

7 Recommendations for Future Work

The next step in turning the project into a marketable product would be to build and communicate with a full sized scoreboard. The voltage, current, and power consumption capabilities of the design would require an upgrade in order to power a much larger scoreboard. Also a new communications method other than RS-232 would be necessary. RS-232 communication serves the purpose of this project, but its distance limitations preclude its use for a final product scoreboard.

The database was a large part of the project which could use more functions. The database could employ a search and sort of statistics, which would provide information about teams, players, and seasons. These improvements fall more under the category of computer science than engineering, and implementing them would be a minor project compared to the scope of the entire scoreboard project.

The prototype scoreboard efficiency could be improved. The data output of the PC occasionally results in a delay in updating the scoreboard. Increasing the speed of the data output would increase the precision of the scoreboard displayed data, which would allow for increased accuracy and less delay. A more efficient output function within the graphical user interface would help to solve this problem.

A final interesting feature that could be added is internet connectivity. A system which updates a web site with statistics as the game progresses in real time would help to increase interest in a given sport and allow for fans to have a statistical readout quickly and easily similar to the readouts available on such national websites as ESPN.com.

Bibliography

[1] Ames, Benjamin B. "Throw away your keyboard." *Design News* Vol. 57 Issue 16. Boston. Aug. 19, 2002. pp. 31-32.

[2] Rooney, Paula. "University of Washington's HIT laboratory: Research facility pioneers advanced computer interfaces." *Computer Reseller News* Issue 920. Manhasset. Nov. 13, 2000. p. A74.

Appendix: Photographs and Figures

Figure 1 – Nevco 2550-D Scoreboard

Figure 2 – Graphical User Interface, Basketball

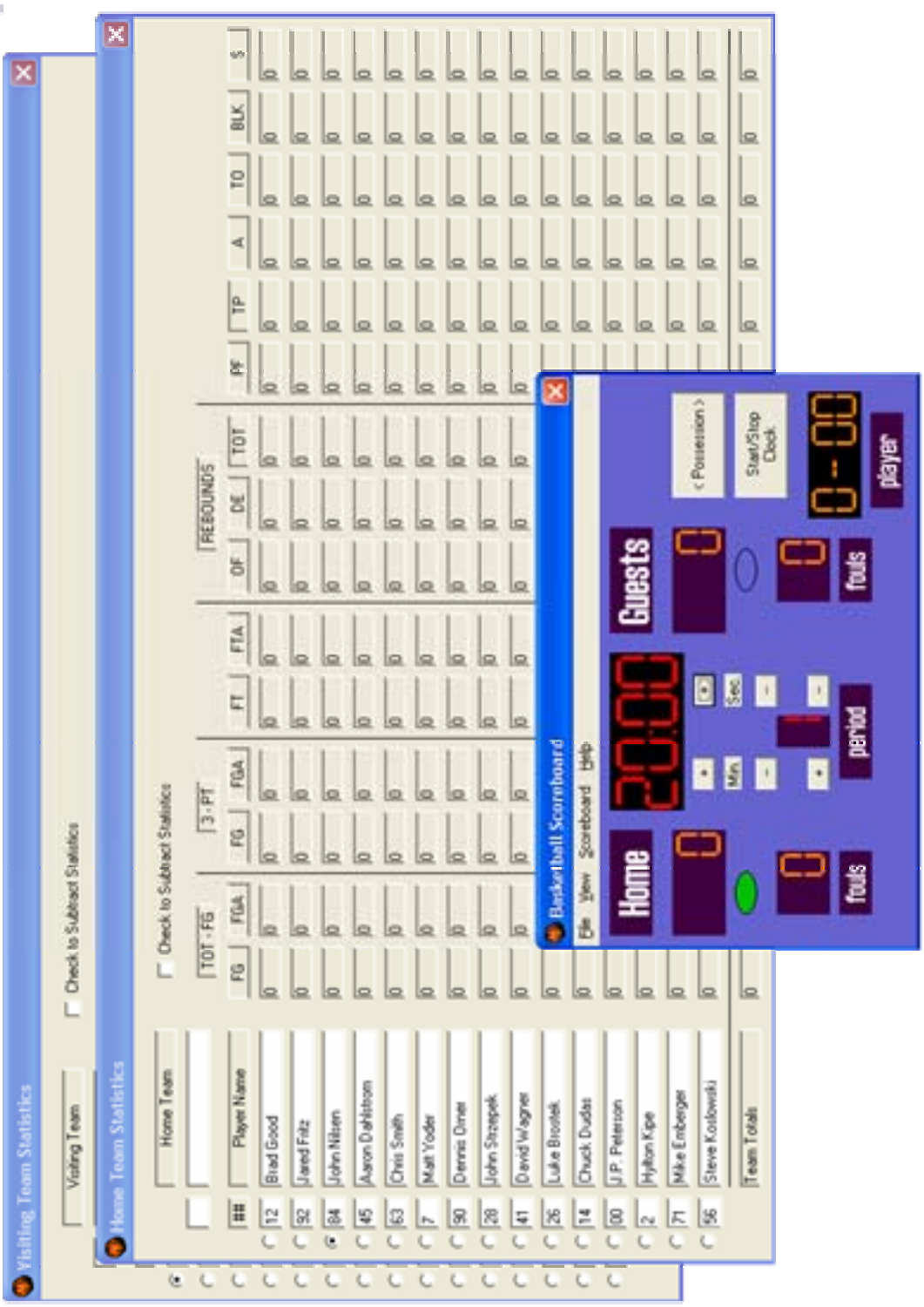


Figure 3 – Graphical User Interface, Volleyball

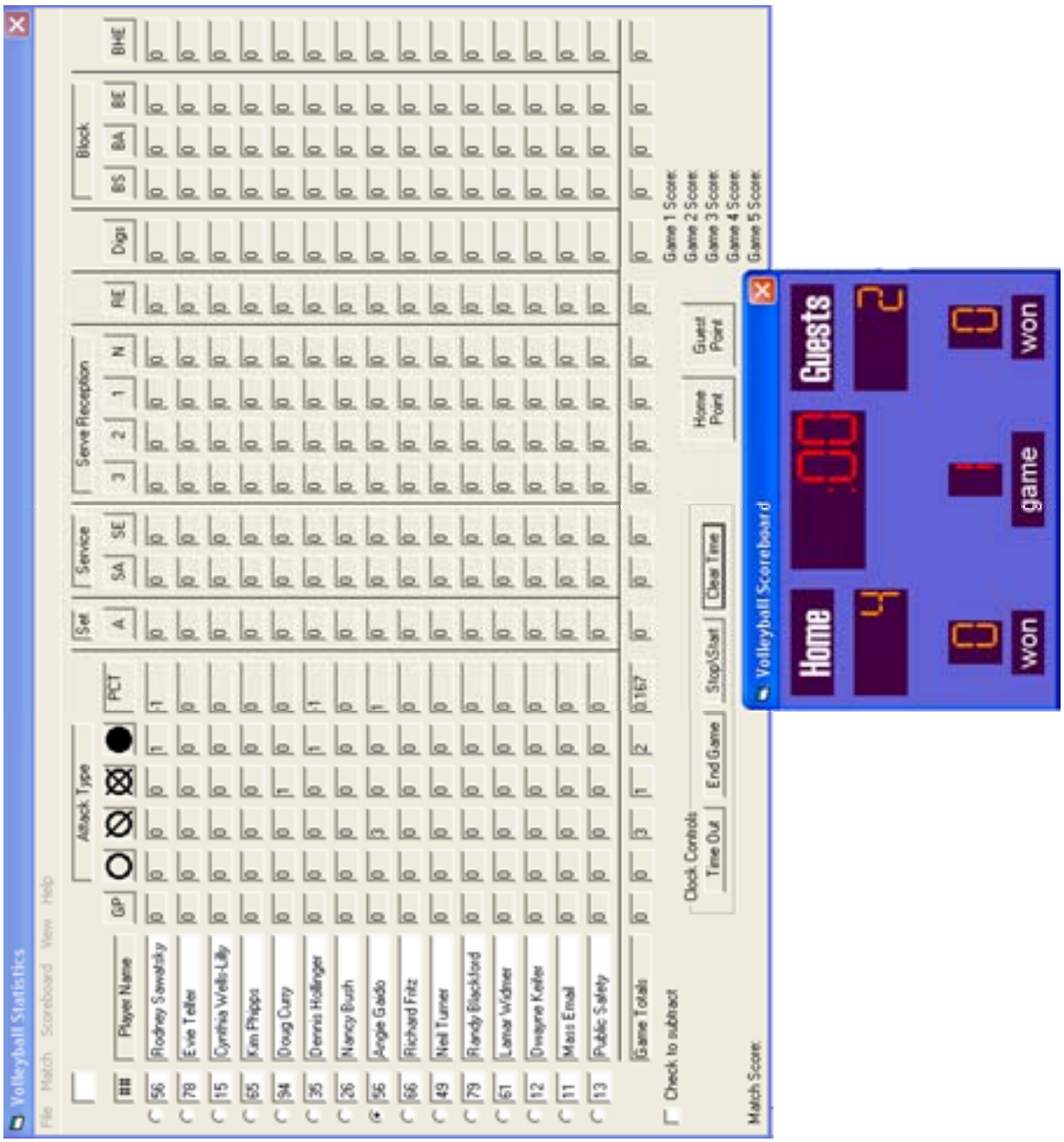


Figure 4 – Graphical User Interface, Wrestling

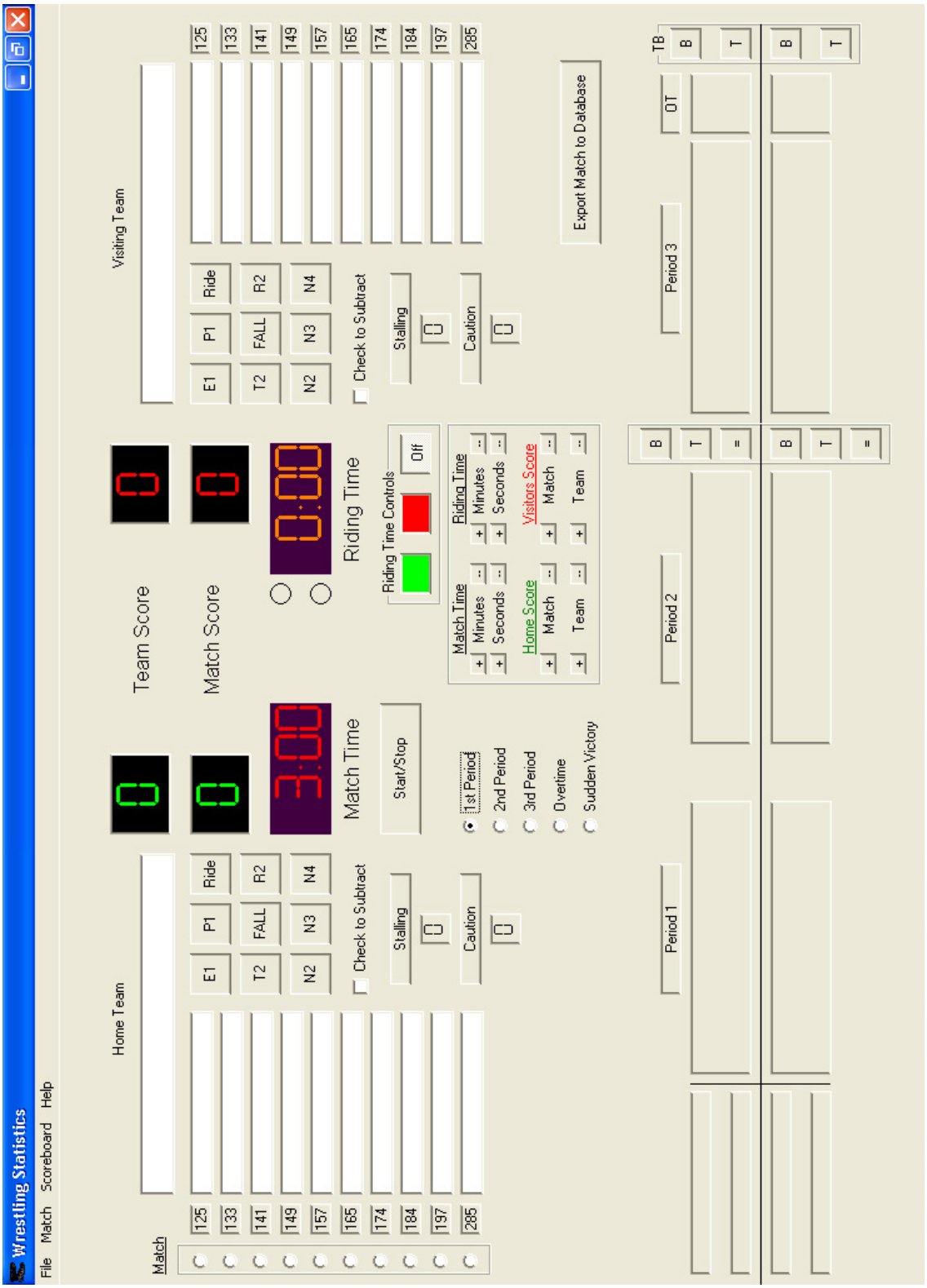


Figure 5 – Nevco Control Box

Figure 6 – Internal View of Control Box Keypad

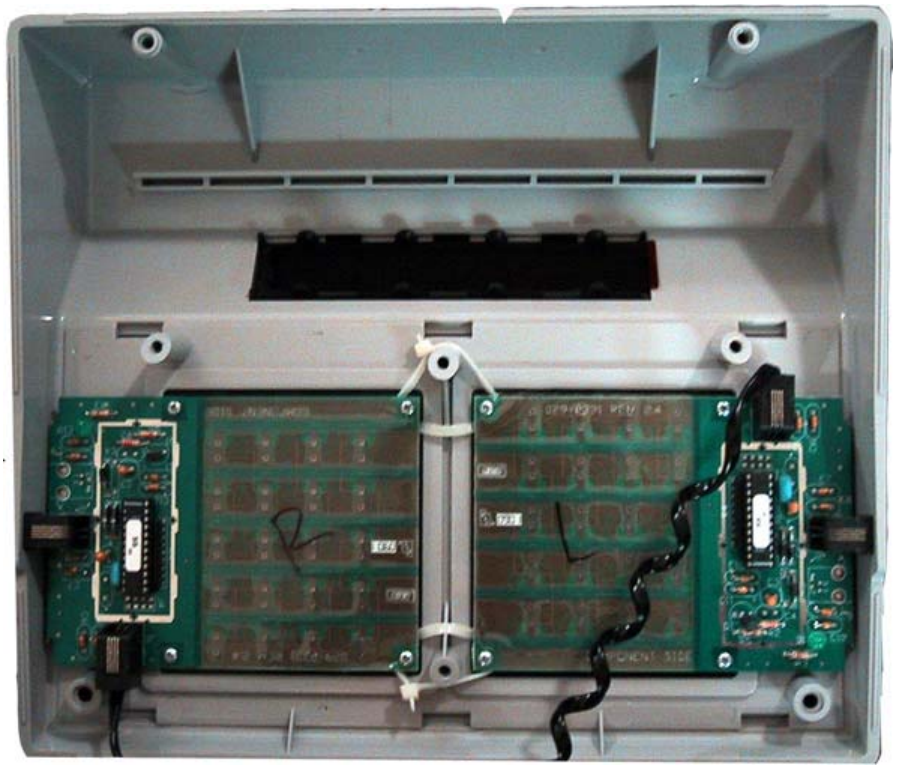


Figure 7 – Internal Circuitry of Control Box

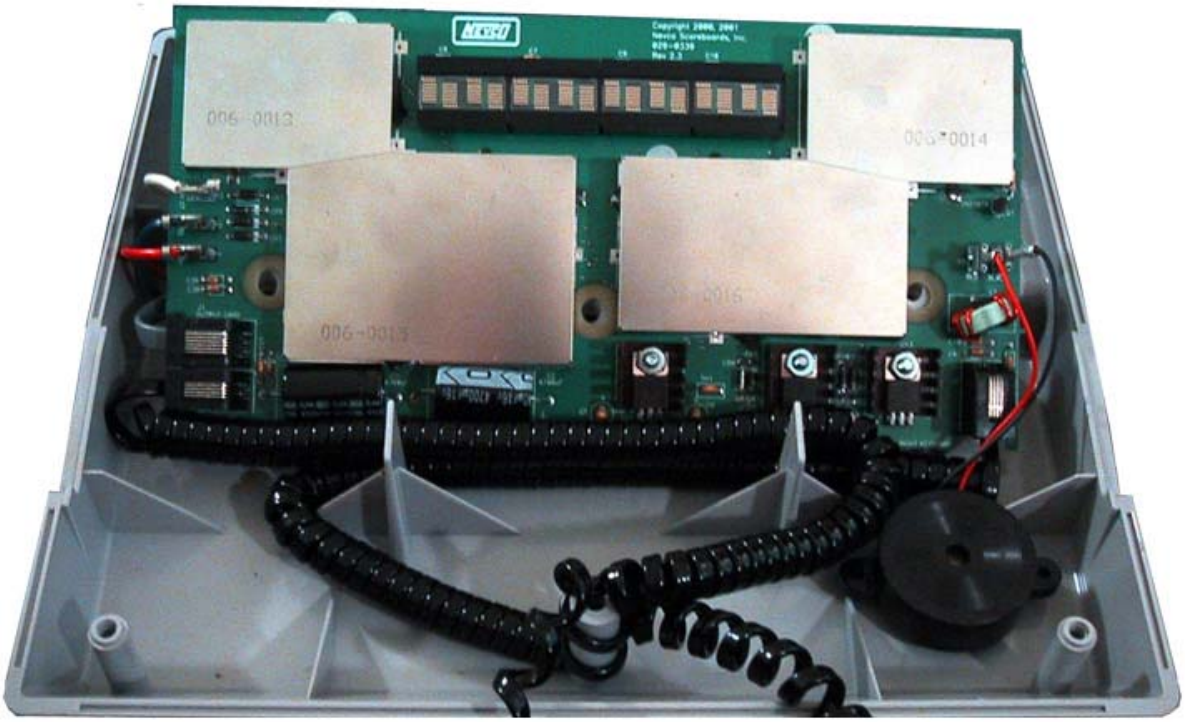


Figure 8 – Dual Circuit Board Layout

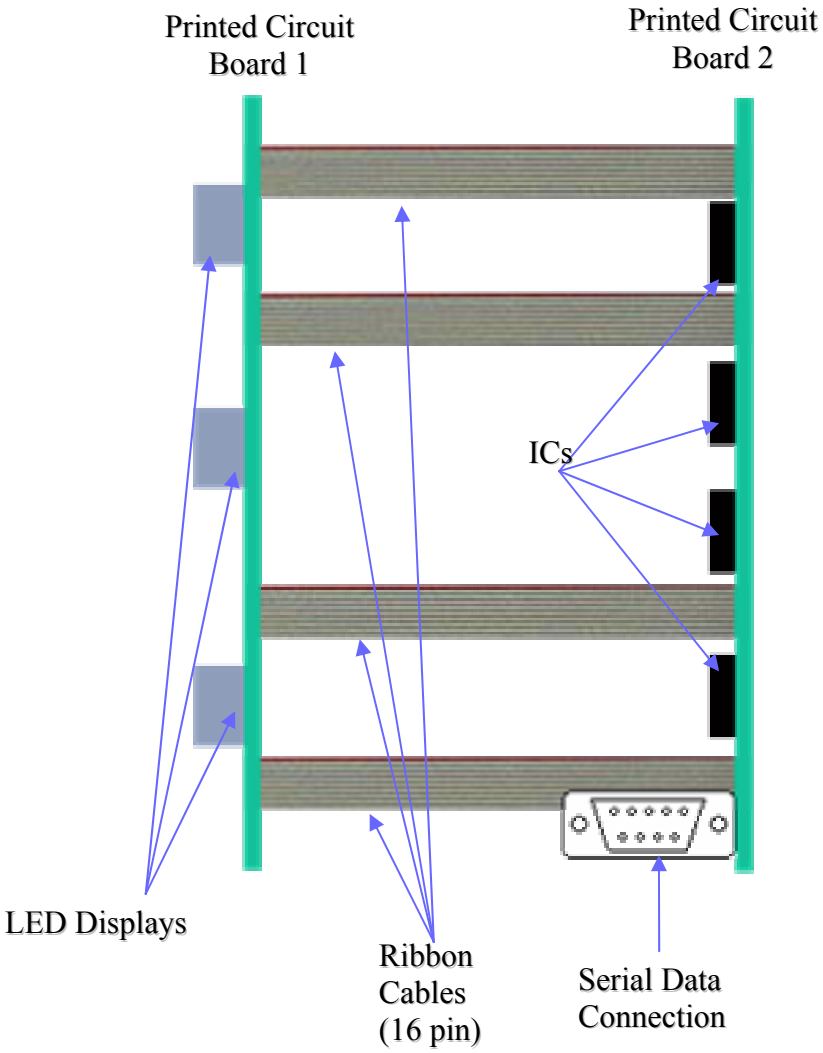


Figure 9 – Block Diagram of Scoreboard Circuit Layout

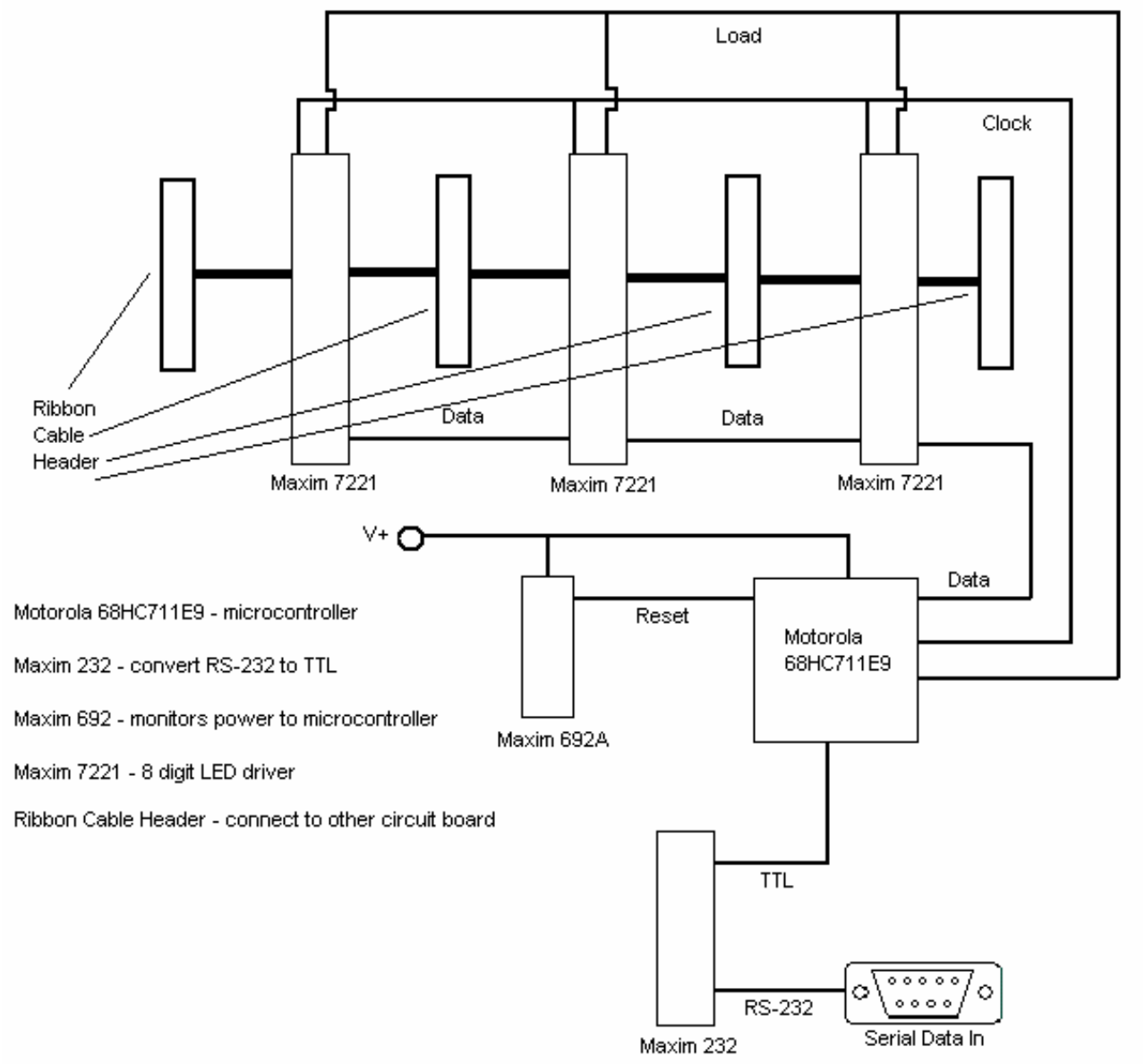


Figure 10 – Photograph of Display

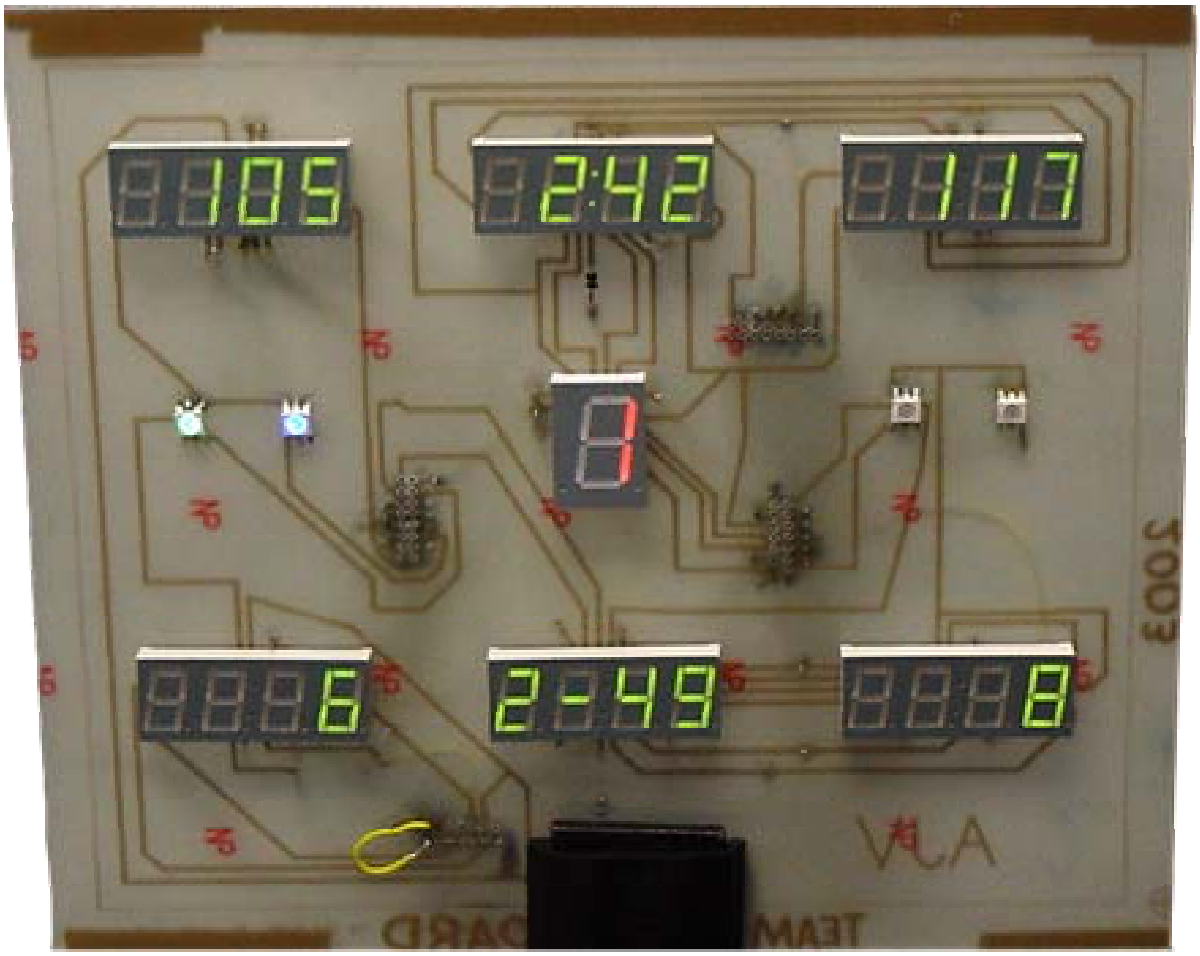


Figure 11 – Photograph of Soldering on Display Board

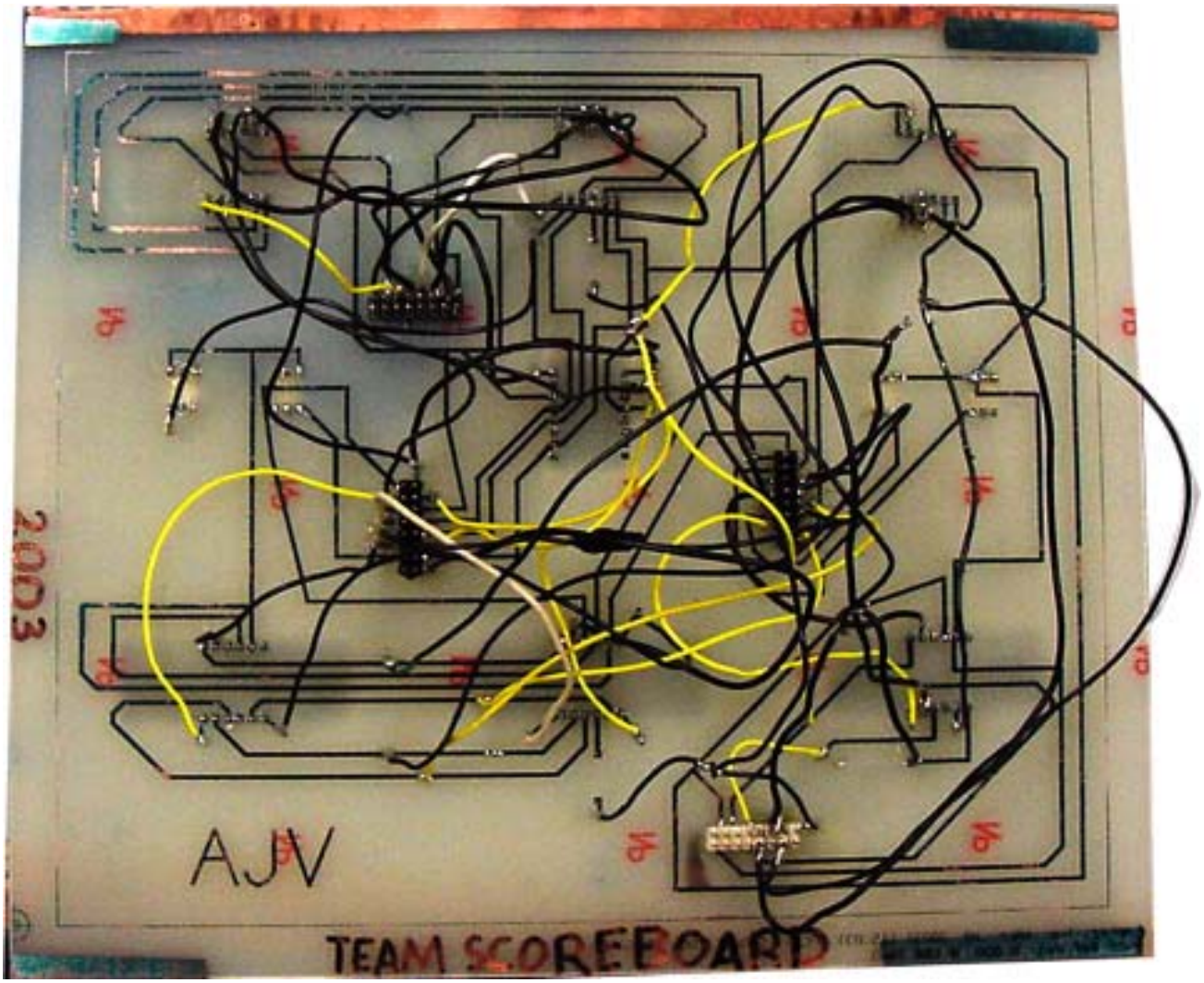


Figure 12 – Rear View of Scoreboard, Circuit Board 2 in Foreground

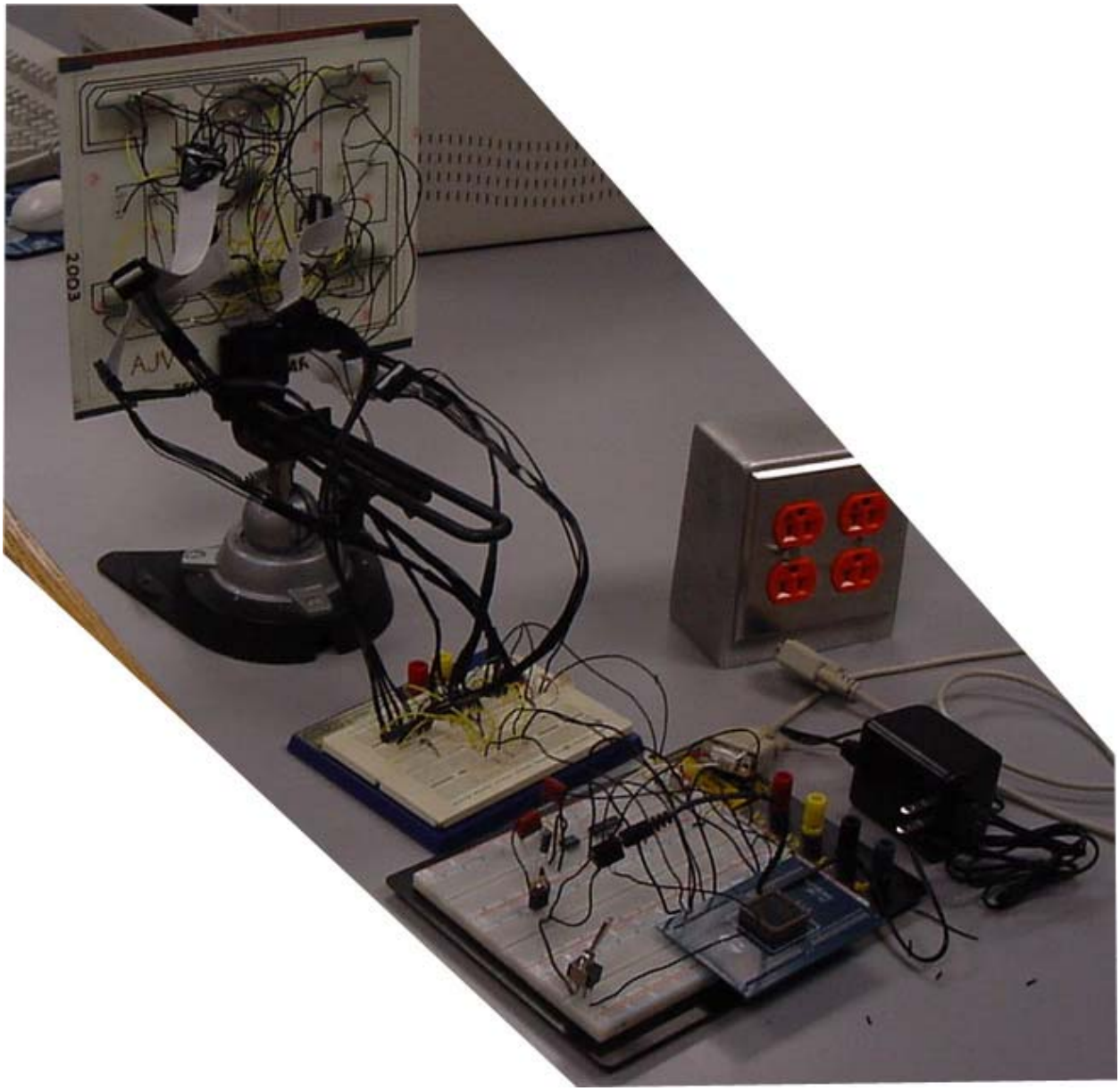


Figure 13 – Final Scoreboard Prototype

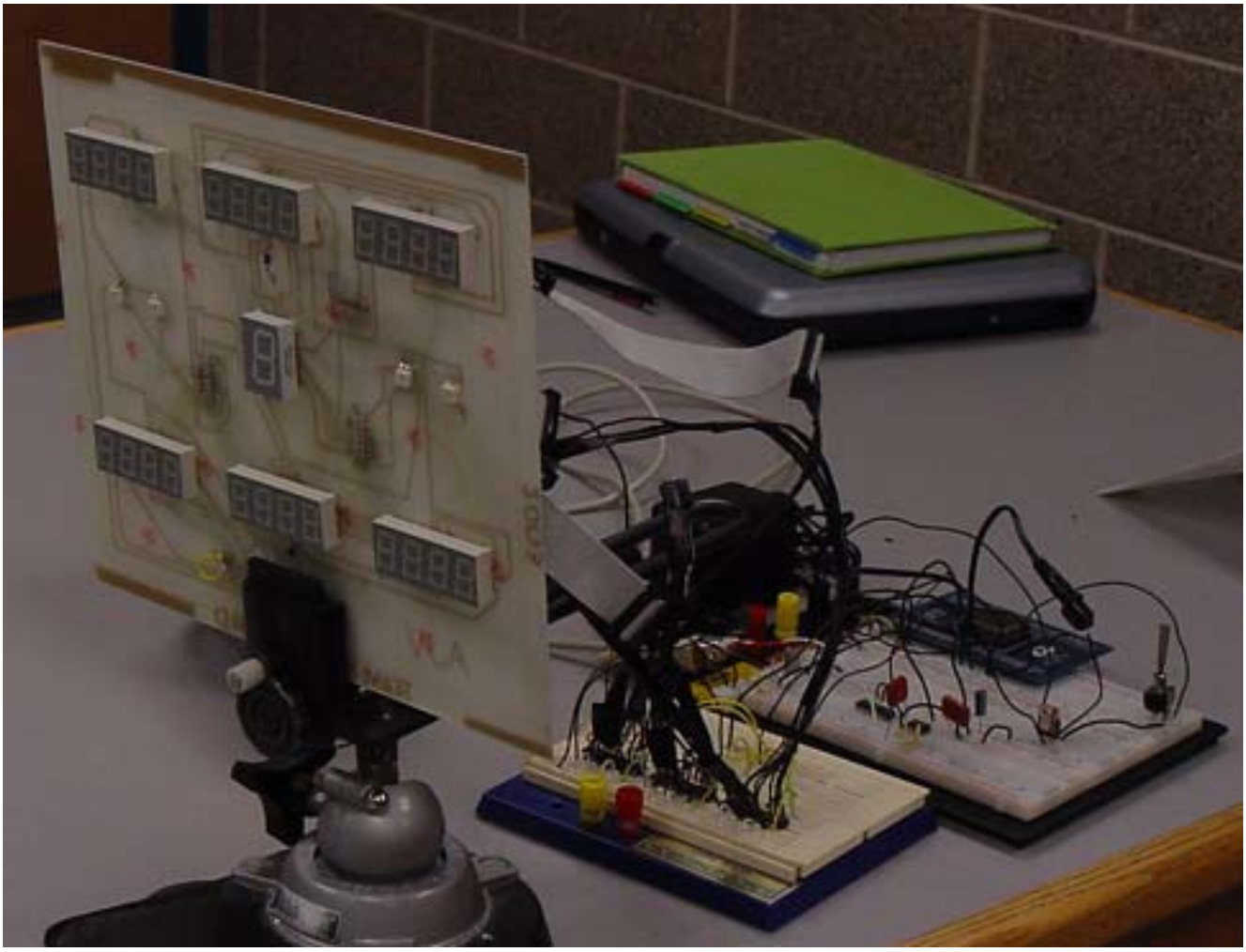


Figure 14 – Final Scoreboard Prototype

Appendix: Supporting Documentation

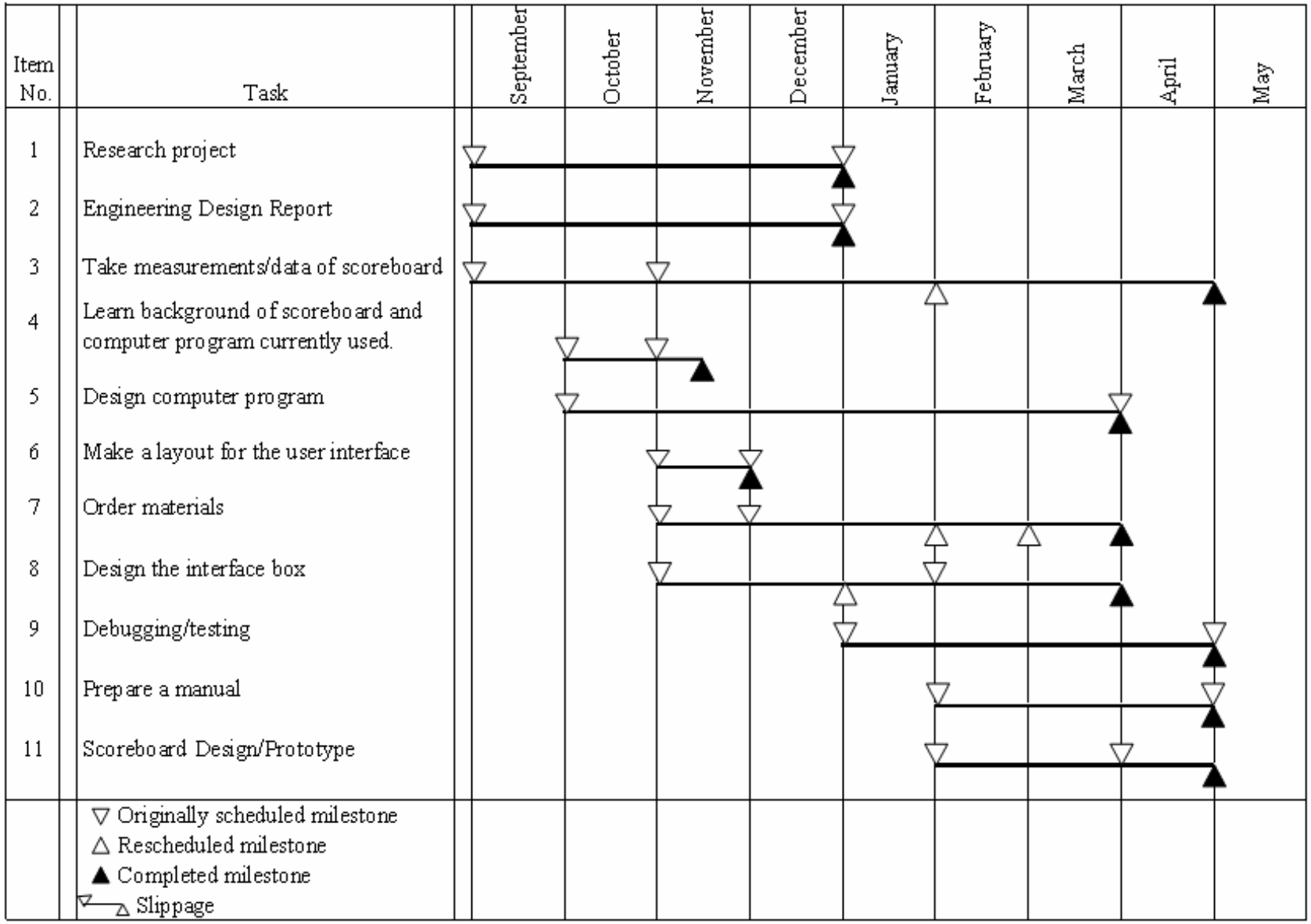
Team Scoreboard Budget

Actual			*Budget =*	$500.00
Price	**Quantity**	**Description**		**Amount ($)**
Electronic Components				$100.00
$7.00	4	5.0 x 6.0 mm Full Color LED Lamps		**($28.00)**
$6.00	8	.56" 4-Digit Green LED Displays		**($48.00)**
$0.98	1	Orange 7-Segment LED Display		$0.00
$3.99	3	Maxium MAX7221, Serially Interfaced, 8-Digit LED Display Drivers		$0.00
$15.95	2	Motorola 68HC11 Microprocessor		**($31.90)**
$0.79	10	Crystal 8-MHz, 50ppm Series		**($7.90)**
$1.33	1	Maxim MAX232 RS-232 to TTL 5volt Logic DIP		$0.00
$0.25	2	Germanium Rectifier		**($0.50)**
$0.38	1	7805 Transistor		**($0.38)**
$0.23	2	100nF Capacitor		**($0.46)**
$0.21	2	22pF Capacitor		**($0.42)**
$0.25	1	4.7 µF Capacitor		**($0.25)**
$0.06	1	1/4 Watt 10 MΩ Resistor		**($0.06)**
$0.06	1	1/2 Watt 10 kΩ Resistor		**($0.06)**
$0.59	7	2-position Header Blocks		$0.00
$19.00	*Shipping*			**($19.00)**
$149.08	*Total*		*Budget Amount Left*	**($36.93)**
			Actual Amount Used	$136.93
Interface Box Materials				$40.00
$19.20	2	PCB Circuit Board		($38.40)
$14.00	1	6VDC 1000mA Wall Adapter		($14.00)
$0.00	*Shipping*			$0.00
$52.40	*Total*		*Budget Amount Left*	($12.40)
			Actual Amount Used	$52.40
Phone Calls				$10.00
$0.07	60	Phone Calls Made		($4.20)
$4.20	*Total*		*Budget Amount Left*	$5.80
			Actual Amount Used	$4.20
Other				$350.00
$9.97	1	Color Copier/Photo Paper		($9.97)
$34.97	1	Color Printer Cartridge		($34.97)
$44.94	*Total*		*Budget Amount Left*	$305.06
			Actual Amount Used	$44.94
$205.68	*Total*		*Budget Total*	$500.00
			Actual Total	$238.47
			Budget Amount Left	$261.53

Prototype

Price	Quantity	Description
$7.00	4	5.0 x 6.0 mm Full Color LED Lamps
$6.00	8	.56" 4-Digit Green LED Displays
$0.98	1	Orange 7-Segment LED Display
$3.99	3	Maxim MAX7221, Serially Interfaced, 8-Digit LED Display Drivers+H37
$1.33	1	Maxim MAX232 RS-232 to TTL 5volt Logic DIP
$15.95	1	Motorola 68HC11 Microprocessor MC68HC711E9CFN2
$19.20	2	PCB Circuit Board
$0.79	1	Crystal 8-MHz, 50ppm Series
$14.00	1	6VDC 1000mA Wall Adapter
$0.25	2	Germanium Rectifier
$0.38	1	7805 Transistor
$0.23	2	100nF Capacitor
$0.21	2	22pF Capacitor
$0.25	1	4.7 µF Capacitor
$0.06	1	1/4 Watt 10 MΩ Resistor
$0.06	1	1/2 Watt 10 kΩ Resistor
$0.59	7	2-position Header Blocks
$19.00	*Shipping*	
$184.68	*Total*	

Gantt Chart


Item No.
Task
September
October
November
December
January
February
March
April
May
1
Research project
2
Engineering Design Report
3
Take measurements/data of scoreboard
4
Learn background of scoreboard and computer program currently used.
5
Design computer program
6
Make a layout for the user interface
7
Order materials
8
Design the interface box
9
Debugging/testing
10
Prepare a manual
11
Scoreboard Design/Prototype
Originally scheduled milestone
Rescheduled milestone
Completed milestone
Slippage

CSC 333 Spring, 2003
Professor Brian Nejmeh

Software Requirements Specification (SRS)

SRS Version Number: 1.2
Date of Publication: February 26, 2003

Scoreboard

James Barley

Messiah College
One College Avenue
Grantham, PA 17027
(717) 766-2511

1. Introduction

1.1. Vision

This project has the vision of automating and streamlining the process of recording statistics and operating the scoreboard for the Messiah College Women's Volleyball team. Ideally, the project will decrease the number of people required to run the scoreboard and keep statistics at athletic contests. This project will create a database and user interface that is easy to use, reliable, and efficient for the previously stated purpose.

1.2. System Goals

- To obtain a working statistical database for the Messiah College Women's Volleyball team that includes all of the current statistics.
- Have the database be controlled by an easy-to-use user interface on a Messiah College laptop.
- Allow statistics to be sorted by player, team, game, match, season, and specific statistic(s).
- Statistics must be able to be outputted to a text file that follows the NCAA statistics submission guidelines.

1.3. System Concept of Operations

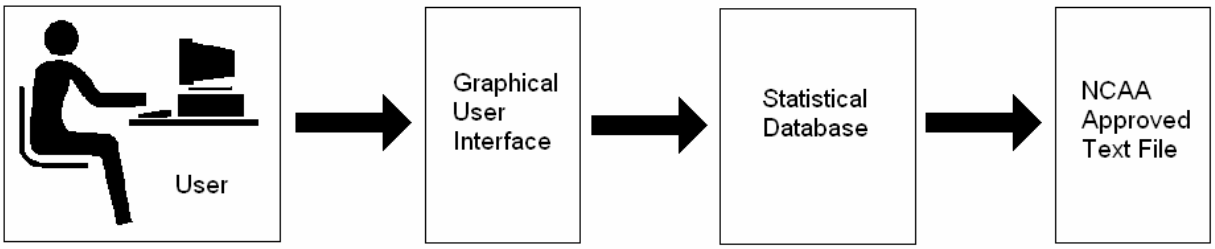


Figure 1-1

There will be one and only one user at a time accessing the designed database. The user will interact with only the graphical user interface (GUI). It will be the interfaces job to communicate with the database and tell the database what the desired outputs are. The database will then output the desired file which was originally initialized by the user.

1.4. System Overview

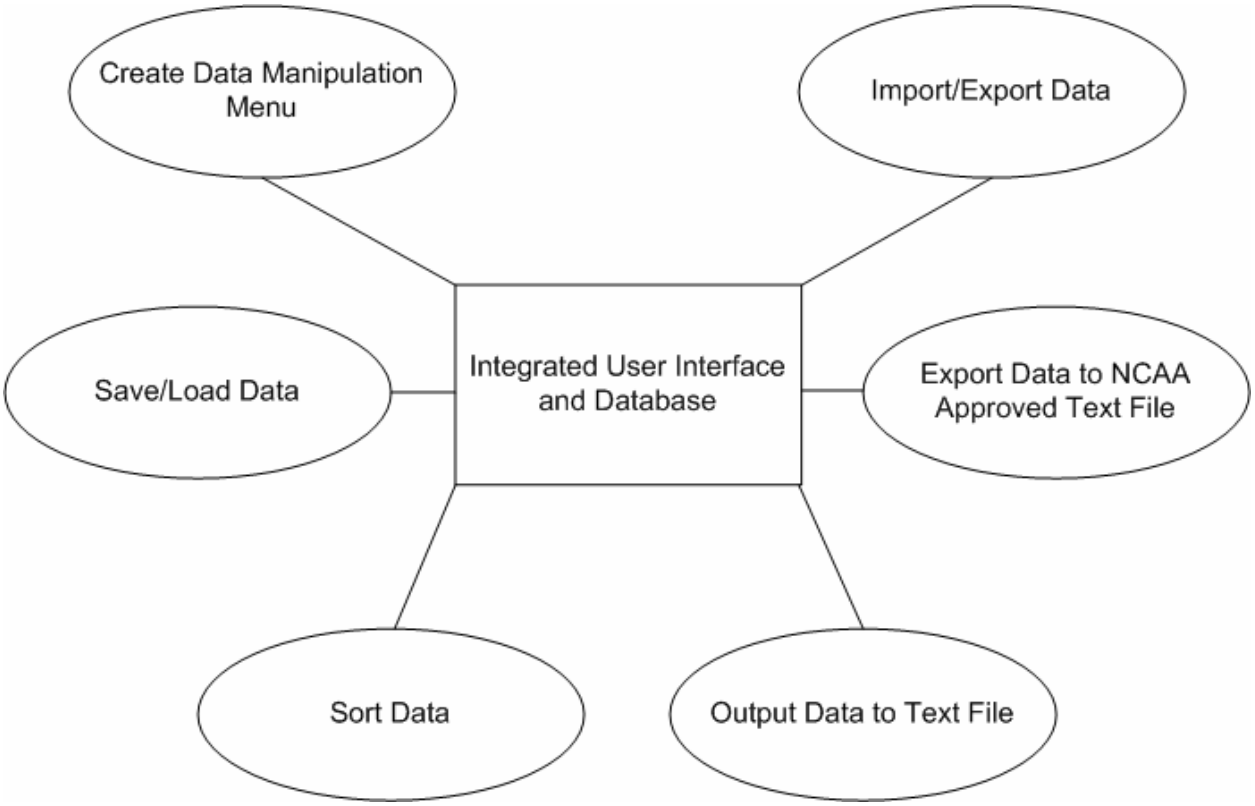


Figure 1-2

The integrated user interface and database will control all aspects of the working program. The first feature of the program will be to manipulate data within the database. The user will have the ability to create, edit, delete and load seasons. Next, the user will be able to save and load data to and from the database. Next, the program will be able to sort data by player, team, match, or season. The next feature will be to output the statistical data or a match displayed by the GUI to a text file. With these past two features accomplished, an NCAA approved text file will be able to be generated. The next step in engineering the software is to be able to import data from previous seasons using only a given NCAA text file. The user will also be able to export sorted data into text formats as well.

1.5. User Types

There will be only one user for this database. Since this database application will be operating on a stand-alone laptop, there is no need or use for multiple users. The user type will be classified as Scoreboard Operator. This is very close to a power user in that he will be able to create, edit, and delete seasons and team statistical information. However, the user will not have administrator rights in that he will be unable to modify the data structure and create new tables, attributes, etc. There is no need for an administrator since the data structure has no need to change for the database to be operational.

1.6. Constraints and Assumptions

- The database application will be able to keep, save and load statistics for the current volleyball season as well as load statistics from previous seasons. It will be run only by the user using the laptop so system security among multiple users is non-existent.
- The database application will be able to sort the data by player, team, game, match, season, and specific statistic(s).
- Data will be managed so that the user cannot input data faster than the database can process and organize it.
- The database application can only be executed on a laptop running Microsoft Windows 2000/XP operating system.

2. Software Requirements

2.1. Requirements Overview of Feature Areas

The database is only going to be managed through the user interface by the user. Because there will be no database administrator, the user interface will be limited to what it can do with low risk of security issues. The following are steps to creating the final software distribution.

- The interface will prompt the user to choose between creating, loading, editing, and deleting a season. Once the user chooses his objective, the GUI will act accordingly giving the user the proper permissions and control.
- The user will be able to save and load to and from the database, allowing seasons to be stored and updated.
- The user will have the option to sort the data by player, team, match, and season.
- The user can output and print a simple text document containing the results of a match to be viewed by the general public.
- The user will be able to output a NCAA approved text file that displays the statistical results of a match.
- Finally, data from previous seasons will be able to be imported from previous NCAA text files and export sorted data to a non-NCAA text document.

2.2. Functional Requirements

The functional requirements in Section 2.2.x will be structured based on the feature areas as defined in Section 2.1.

2.2.1. Data Manipulation Menu (Priority: High)

The data manipulation menu will allow the user to select which type of data manipulation he will use before running the main part of the user interface.

2.2.1.1. Create New Season

If the user chooses to create a new season, the database will give him the proper permission to do so. The user will input the required fields of season year, season location, and team name in a form before the new season can be created.

2.2.1.1.1. Add Player

A player can be added to a newly created season using the same form as the Create New Season form. The user will have to input the season year, season location and team name that the new player will be added to. The user will then have to input the new player's first and last name, player number and player position.

2.2.1.1.2. Add Team

A team can be added to a newly created season using the same form as the Create New Season form. The user will have to input the season year and season location that the new team will be added to. The user will then have to input the team's name.

2.2.1.2. Edit a Season

If the user chooses to edit a season, the user will have to supply the season year and location of the season to be edited. The user will then have proper permission to edit any

team, or player information. This includes team name, player first and last name, player position and player number.

2.2.1.2.1. Edit Player

Player data can be modified to accommodate name changes, player number changes, and player position changes. The user will have to input the current information of the player, and then what they want that information changed to.

2.2.1.2.2. Edit Team

Team data can be modified to accommodate team name changes. The user will have to input the current information of the team, and then what they want that information changed to.

2.2.1.3. Delete a Season

If the user chooses to delete a season, the database will give him the proper permission to do so once the season year and season location of the season to be deleted are provided.

2.2.1.3.1. Delete Player

A player can be deleted from a team by selecting the Delete Player option in the Delete a Season form. The season year, season location, and team name of that player must be provided before that player can be deleted.

2.2.1.3.2. Delete Team

A team can be deleted from a season by selecting the Delete Team option in the Delete a Season form. The season year and season location of the team must be provided before that team can be deleted.

2.2.1.4. Load a Season

If the user chooses to load a season, he must supply the season year, season location and team name of the season to be loaded. Once this is done, the GUI will load allowing the user to manipulate statistics for a new match.

2.2.2. Save/Load to Database Option (Priority: Med)

The user will have the option of saving data to the database or loading previously created data into the database. This will be done simply by allowing the user to choose save or load from a drop down menu.

2.2.2.1. Save Match

The user will be able to save the progress of a match or the completed match in the database. The user can save the match to a local hard disk or other secondary storage device. Saving the match will update the entire season database accordingly, making changes to player statistics.

2.2.2.2. Load Match

The user will be able to load a previously saved match in order to make correction and/or complete and unfinished match. The match can be loaded from a local hard disk or other secondary storage device. Loading matches will not affect the data contained within the database.

2.2.3. Sort Option (Priority: Low)

The user will have the ability to sort the data by player, team, match, and season. The user will use a form to complete this task. From this form, the user will select his sorting method from a drop down menu and then input information such as player ID, team name, match ID, and season year and location depending upon what data is to be sorted.

2.2.3.1. Player Sort

The player sort will display statistics from a chosen match, or season for the desired player. The user will select the player whose statistics will be viewed and select sort by match or season.

2.2.3.2. Team Sort

The team sort will display statistics from a chosen match, or season for the desired team. The user will select the team whose statistics will be viewed and select sort by match or season.

2.2.3.3. Match Sort

The match sort will display statistics from a chosen player or team for the desired match. The user will select the match which statistics will be viewed and select sort by player or team.

2.2.3.4. Season Sort

The season sort will display statistics from a chosen player or team for the desired team. The user will select the season which statistics will be viewed and select sort by player or team.

2.2.4. Export and Print Viewed Data (Priority: Low)

The user will have the option of outputting the statistical results displayed by the GUI of a volleyball match that has just been played or has been played in the past to a text file.

2.2.4.1. Export and Print

Exporting to a text format can be used for studying the results of a volleyball match and/or allowing players to see how well they competed during a match. This outputted text file can be printed to a local printer and will not be an official NCAA document. This shall be done by selecting "Export to text file" from a drop down menu.

2.2.5. Output to NCAA Approved Text File (Priority: High)

The user will have the option of outputting the data that is currently being displayed in the GUI to an NCAA approved text file that can be submitted to the NCAA via email.

2.2.5.1. Output Displayed Data

The data that is currently being displayed is the statistical results of a volleyball match that has just been played or has been played in the past. This shall be done by selecting "Export to NCAA text file" from a drop down menu.

2.2.5.2. Send Information to NCAA

The exported text file will be saved on the local hard disk or other secondary storage device. The user will then load his email application of choice and attach the NCAA text file and send it to the NCAA.

2.2.6. Choice of Import or Export (Priority: Med)

The user will have the option of importing data to a high storage device such as a local hard disk, or a smaller storage device such as a floppy, zip, or CD-R/RW disk, or exporting data from the same type of locations as mentioned above.

2.2.6.1. Importing Data

Importing data will allow the user to load data from a previous season given only the NCAA approved text file mentioned in section 2.2.5. This will be done by allowing the user to choose the import location such as a hard disk or floppy disk and then choose the file name.

2.2.6.2. Exporting Data

Exporting data will do the exact same function as 2.2.4, however the sorted data as discussed in section 2.2.3 will be able to be exported in txt file format to either a local hard disk or external floppy, zip or CD-R/RW disk. This user will select which sort he wishes to export and the location he wishes to save that file to.

3. Increments

3.1. Working User Interface

This increment will contain a fully function user interface that simply performs the desired math required for team statistic keeping. The interface will contain other desired forms for user input.

3.2. Data Manipulation with Save and Load

This increment will allow the user to create new seasons, edit seasons, delete seasons, load seasons, save seasons with statistical data and load season data. This increment incorporates requirements 2.2.1 and 2.2.2.

3.3. Sort and View Data

This increment will allow the user to sort data and save data currently being displayed by the interface in a separate text file to be view at the user's convenience. This increment incorporates requirements 2.2.3 and 2.2.4.

3.4. NCAA File with Input/Output

This increment will allow the user to save data to an NCAA approved text file as well as input and output from and to other designated locations. This increment incorporates requirements 2.2.5 and 2.2.6.

4. Open Issues

4.1. Allowing MySQL to communicate with Visual Basic 6.0 may prove to be cumbersome.

4.2. Table structure may get complicated with having games, matches, seasons and statistics being related to a player.

5. References

N/A

6. Glossary of Terms

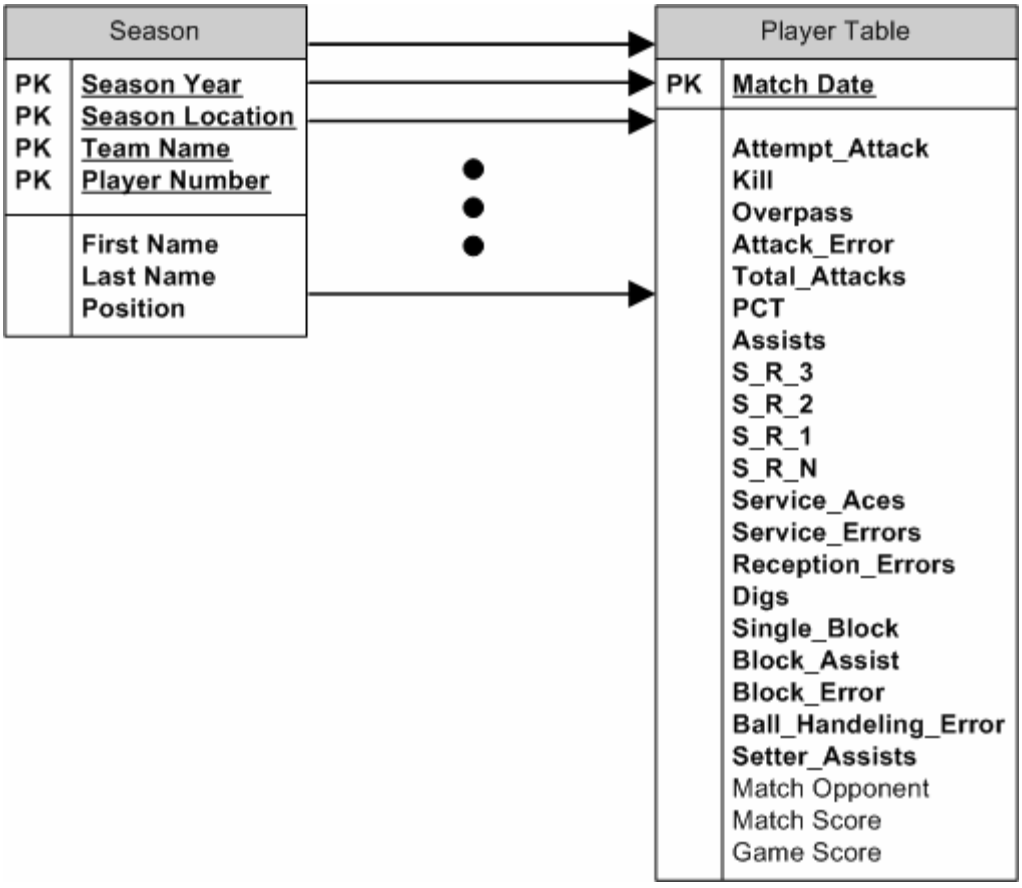
6.1. NCAA approved text file

This is the text file that is emailed to the NCAA in order to keep track of Messiah College's Women's Volleyball team. This text file is used to update the NCAA national database.

Sports Data Dictionary

Type Name	Type (value)	Description
Season	Entity	Data type that defines the entire volleyball season
Season_Year	Attribute (integer)	The year of the season uniquely identifies the season along with the season location
Season_Location	Attribute (text)	The location of the season uniquely identifies the season along with the season year
Team_Name	Attribute (text)	The name of the team
First_Name	Attribute (text)	First name of the player
Last_Name	Attribute (text)	Last name of the player
Player_Num	Attribute (integer)	The players number, unique identifier to identify players
Position	Attribute (text)	Position of the player
Match_Date	Attribute (text)	The date in which a match took place
Match_Opponent	Attribute (text)	The opponent of the match
Match_Score	Attribute (text)	The final score of the match
Game_Score	Attribute (text)	The final score of a game
Attempt_Attack	Attribute (integer)	Number of attack attempts
Kill	Attribute (integer)	Number of kills
Overpass	Attribute (integer)	Number of overpasses
Attack_Error	Attribute (integer)	Number of attack errors
Total_Attacks	Derived Attribute (integer)	Number of total attacks
PCT	Derived Attribute (decimal(2))	Percent total found by dividing the difference of attack errors from kills by the total attacks
Assists	Attribute (integer)	Number of assist
S_R_3	Attribute (integer)	Number of serve receptions of value 3
S_R_2	Attribute (integer)	Number of serve receptions of value 2
S_R_1	Attribute (integer)	Number of serve receptions of value 1
S_R_N	Attribute (integer)	Number of serve reception that result in an overpass
Service_Ace	Attribute (integer)	Number of service aces
Service_Error	Attribute (integer)	Number of service errors
Reception_Error	Attribute (integer)	Number of reception errors
Digs	Attribute (integer)	Number of digs
Single_Block	Attribute (integer)	Number of single blocks
Block_Assist	Attribute (integer)	Number of block assists
Block_Error	Attribute (integer)	Number of block errors
Ball_Handling_Error	Attribute (integer)	Number of ball handling errors

Database ER Diagram



Test Plan

1. Test Plan Overview

    1.1. The test plan will consist of 10 primary tests. Each test will with a different aspect of the program and the database. Most tests will deal with loading data from the database into forms and saving data from the forms into the database. Other tests will make sure data can be effectively manipulated in the database. These tests will involve creating, deleting and updating data.

2. Test Cases

    2.1. Create Seasons, Teams and Players Test

    Input: The volleyball database will be fully populated with 10 seasons each of which will include 10 teams. Each team will then include 10 players. This will be done using the Create New Season form.

    Description: The test will verify that an entry for each player will be placed into the main season table and that an empty table will be created for each new player. This will be done using the Create New Season form.

    Expected Output: A new entry will be added to the season table and a uniquely named table will be created for each player added to the volleyball database.

    SRS Requirements Tested: 2.2.1.1 (Create New Season), 2.2.1.1.1 (Add Player), 2.2.1.1.2 (Add Team).

    2.2. Edit Seasons and Teams Test

    Input: 10 different seasons and team names will be edited to include different years, locations and names.

Description: The test will verify that season and team information can be updated in the season table and the table name for each player can be renamed according to the edited information.

Expected Output: Season and team information in the season table will be updated to reflect the changes made by the user. Tables representing players affected by the changes should be renamed without modifying entries within those tables. The season, with its corresponding teams and players changes will be reflected from the drop down menus once they are changed.

SRS Requirements Tested: 2.2.1.2 (Edit Season), 2.2.1.2.2 (Edit Team).

2.3. Edit Player Test

Input: 10 different players will be edited to include different first and last names, positions and player numbers.

Description: The test will verify that player information can be updated in the season table and the table name for each player can be renamed according to the edited information.

Expected Output: Player information in the season table will be updated to reflect the changes made by the user. Tables representing players affected by the changes should be renamed without modifying entries within those tables. The player changes will be reflected in the drop down menus once the player is edited.

SRS Requirements Tested: 2.2.1.2.1 (Edit Player).

2.4. Delete Season Test

Input: 10 different seasons, each with a team and 10 players will be created and then deleted using the Delete Season form.

Description: The test will verify that seasons can be deleted; this includes deleting the teams and players within those deleted seasons.

Expected Output: All entries in the season table that correspond to the season selected by the user will be deleted. Also, all player tables for players who reside in that the deleted season will be deleted. The deleted season, with its corresponding teams and players will not be available from the drop down menus once the seasons are deleted.

SRS Requirements Tested: 2.2.1.3 (Delete Season)

2.5. Delete Team Test

Input: 10 different teams each with 10 players will be created and then deleted using the Delete Season form.

Description: The test will verify that teams can be deleted; this includes deleting the players within those deleted teams. Seasons will not be affected.

Expected Output: All entries in the season table that correspond to the team selected by the user will be deleted. Also, all player tables for players who reside in that the deleted team should be deleted. The deleted team, with its corresponding players will not be available from the drop down menus once the teams are deleted.

SRS Requirements Tested: 2.2.1.3.2 (Delete Team)

2.6. Delete Player Test

Input: 10 different players will be created and then deleted using the Delete Season form.

Description: The test will verify that only players can be deleted. Seasons and teams will not be affected.

Expected Output: All entries in the season table that correspond to the player selected by the user will be deleted. Also, all player tables that relate to the selected players will be deleted. The deleted players will not be available from the drop down menu once the players are deleted.

SRS Requirements Tested: 2.2.1.3.1 (Delete Player)

2.7. Load Team Test

Input: A team will loaded into the main statistical interface using the Load Season form.

Description: The test will verify that a team of up to 15 players can be loaded into the main statistical form and display the name and number of each player within that team in the main statistical form.

Expected Output: No tables or database information will be created, updated or deleted in this test. Once the season and team are chosen by the user, the players associated with that season and team will be loaded and displayed into the main statistical form.

SRS Requirements Tested: 2.2.4.1 (Load Season)

2.8. Input Invalid Data Test

Input: 5 seasons, teams and players will be created using the Create Season form.

Description: The test will verify that a user cannot attempt to manipulate entries from the database that do not exist. There is the possibility of manipulating the drop-down menus to load invalid data.

Expected Output: Once the user attempts to load, delete, or edit invalid data, a message will appear indicating which information is invalid. No data will be manipulated.

SRS Requirements Tested: 2.2.1.1 (Create Season), 2.2.1.1.1 (Add Player), 2.2.1.1.2 (Add Team), 2.2.1.2 (Edit Season), 2.2.1.2.1 (Edit Player), 2.2.1.2.2 (Edit Team), 2.2.1.3 (Delete Player), 2.2.1.3.1 (Delete Player), 2.2.1.3.2 (Delete Team), 2.2.1.4 (Load a Season).

2.9. Save to Database Test

Input: 5 teams will be created and loaded into the main statistical form. Once the team is loaded, a mock game will be played to update all player statistics. At the end of the game, the data will be saved to the database via the Export Data to Database menu option. The date and the opponent will then be typed in by the user. This will be done for each of the five teams.

Description: The test will verify that player statistical data can be stored into the volleyball database in each player table. It will be done simply by loading, playing a mock game and exporting to the database.

Expected Output: Once the user supplies the date and opponent, all player tables associated with the players on the team will be updated to include their own match statistics.

SRS Requirements Tested: 2.2.2.1 (Save Match).

2.10. Load from Database Test

Input: 1 team will be created with 10 players. The team will be loaded into the main statistical form and 10 mock games will be played. Each game will be

saved to the volleyball database. Then, each match will be loaded into the main statistical form from the volleyball database. The user will then input the date of the match that is to be loaded.

Description: The test will verify that previously saved matches can be loaded into the main statistical from the volleyball database.

Expected Output: Each player table will be read and the proper match will be loaded into the main statistical form without errors. The matches will then be updated and inserted back into the database with a different date.

SRS Requirements Tested: 2.2.2.2 (Load Match)

System Test Results

Test Case 2.1. Create Seasons, Teams and Players Test

Test Results:

Input: The volleyball database was to be fully populated with 10 seasons each of which was to include 10 teams. Each team was then to include 10 players. This was to be done using the Create New Season form.

Output: A new entry was added to the season table and a uniquely named table was created for each player added to the volleyball database.

Pass

Test Case 2.2. Edit Seasons and Teams Test

Test Results:

Input: 10 different seasons and team names were to be edited to include different years, locations and names.

Output: Season and team information in the season table was updated to reflect the changes made by the user. Tables representing players affected by the changes were renamed without modifying entries within those tables. The season, with its corresponding teams and players, changes were reflected from the drop down menus once they were changed.

Pass

Test Case 2.3. Edit Player Test

Test Results:

Input: 10 different players were to be edited to include different first and last names, positions and player numbers.

Output: Player information in the season table was updated to reflect the changes made by the user. Tables representing players affected by the changes were renamed without modifying entries within those tables. The player changes were reflected in the drop down menus once the player was edited.

Pass

Test Case 2.4. Delete Season Test

Test Results:

Input: 10 different seasons, each with a team and 10 players were to be created and then deleted using the Delete Season form.

Output: All entries in the season table that correspond to the season selected by the user were deleted. Also, all player tables for players who reside in that the deleted season were deleted. The deleted season, with its corresponding teams and players were not available from the drop down menus once the seasons were deleted.

Pass

Test Case 2.5. Delete Team Test

Test Results:

Input: 10 different teams each with 10 players were to be created and then deleted using the Delete Season form.

Output: All entries in the season table that correspond to the team selected by the user were deleted. Also, all player tables for players who reside in that the deleted team were deleted. The deleted team, with its corresponding players was not available from the drop down menus once the teams were deleted.

Pass

Test Case 2.6. Delete Player Test

Test Results:

Input: 10 different players were to be created and then deleted using the Delete Season form.

Output: All entries in the season table that correspond to the player selected by the user were deleted. Also, all player tables that relate to the selected players were deleted. The deleted players were not available from the drop down menu once the players were deleted.

Pass

Test Case 2.7. Load Team Test

Test Results:

Input: A team was to be loaded into the main statistical interface using the Load Season form.

Output: No tables or data entries were created, updated or delete. Once the season and team was chosen by the user, the players associated with that season and team were loaded and displayed into the main statistical form.

Pass

Test Case 2.8. Input Invalid Data Test

Test Results:

Input: 5 seasons, teams and players were to be created using the Create Season form.

Output: Once the user attempted to load, delete, or edit invalid data, a message appeared indicating which information was invalid. No data was manipulated.

Pass

Test Case 2.9. Save to Database Test

Test Results:

Input: 5 teams were to be created and loaded into the main statistical form. Once the team was loaded, a mock game was to be played to update all player statistics. At the end of the game, the data was to be saved to the database via the Export Data to Database menu option. The date and the opponent were to be typed in by the user. This would be done for each of the five teams.

Output: Once the user supplied the date and opponent, all player tables associated with the players on the team were updated to include their own match statistics.

Pass

Test Case 2.10. Load From Database Test

Test Results:

Input: 1 team was to be created with 10 players. The team was then to be loaded into the main statistical form and 10 mock games were to be played. Each game was to be saved to the volleyball database. Then, each match was to be loaded into the main statistical form from the volleyball database. The user would then input the date of the match that is to be loaded.

Output: Each player table was read and the proper match was loaded into the main statistical form without errors. The matches were then updated and inserted back into the database with a different date.

Pass