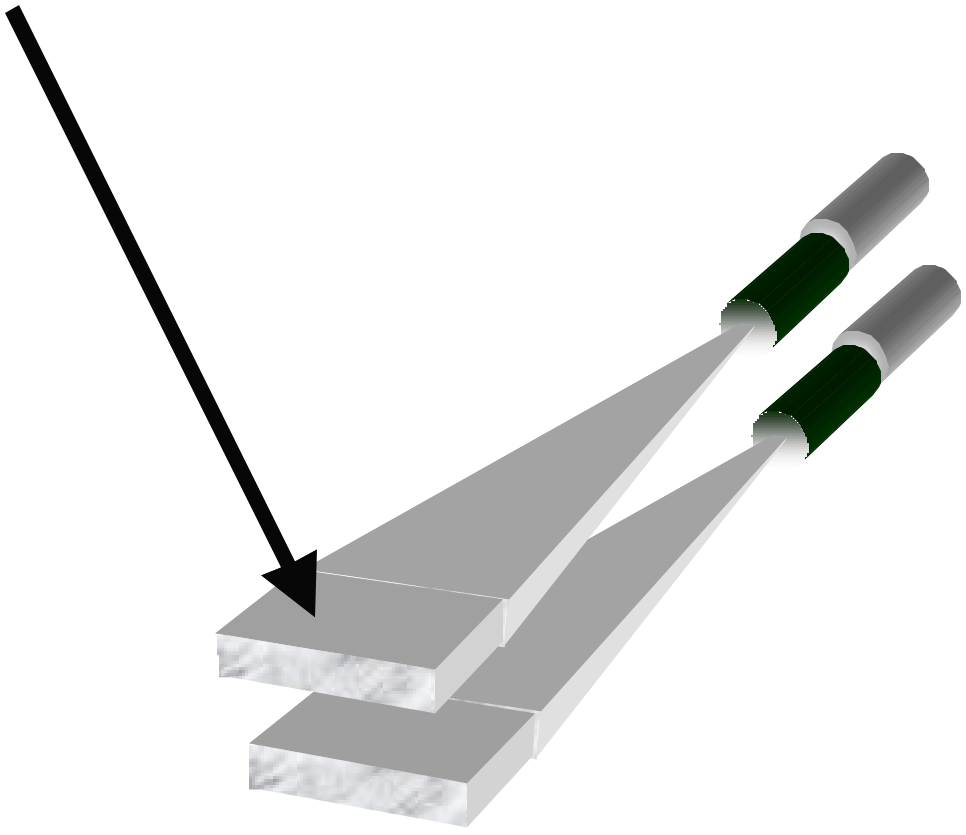
QuarkNet Cosmic Ray Detection System Manual



Presented By Sten Hansen, Electrical Engineer, FermiLab PPD, John Lofgren, Erich Keller
Written By Daniel Balick
Fermi National Accelerator Laboratory

ÓJuly 2001 (Version 1.00)

QuarkNet Contacts

Thomas Jordan
QuarkNet Coordinator
Education Specialist
Fermilab EO
JordanT@FNAL.gov

Prof. Kevin McFarland
Physics Professor
University of Rochester
Ksmcf@pas.rochester.edu

Mark Adams

Project Team

Sten Hansen
Electrical Engineer
FermiLab PPD/EED
Hansen@FNAL.gov

John Lofgren
CO-OP Student
Electrical Engineering Major
Valparaiso University

Erich Keller
CO-OP Student
Electrical Engineering Major
Valparaiso University

Daniel Balick
FermiLab Summer Student
Author of Manual

Table of Contents

QuarkNet Cosmic Ray Detection System Manual

Daniel Balick

I. Introduction

The cosmic ray detection system is designed to allow the user to accurately observe the rate of incoming charged particles, as well as the approximate decay time of each. The vast majority of charged particles that reach the surface of the earth are muons. Muons have a longer lifetime, and can travel farther than other particles that come from the atmosphere with enough energy to reach the detector. Consequently, the detector can be used to determine the average muon decay time, or muon lifetime.

The detectors are designed specifically to provide a reasonably priced detection system on a scale appropriate for classroom or home use. The parts used in this device are standard for particle physics experiments, and are the most appropriate choices in terms of speed, efficiency, and cost. The detectors are scintillator paddles which, when struck by muons, convert the particles' kinetic energy into several photons. These photons excite nearby electrons. As the electrons return to ground state, photons are emitted. The number of photons produced is proportional to thickness of the scintillator. This allows the connected photomultiplier tubes to pick up the light gathered by the light guides and convert it into a signal. This signal is then transmitted to the PC board.

The PC board's default settings require a coincidence, a signal from each paddle within a set amount of time, to process the event and consider it a charged particle hit. The four-digit hexadecimal LED display will show the number of triggers that has occurred since the device was reset. The PC board can transmit a full listing of data. This includes: the port number of the paddle that received the second pulse of the coincidence, the time interval between the two hits involved, the time interval since the last hit (either a coincidence or a single hit), and confirmation that either a hit or a coincidence occurred.

There is a small probability that a low energy muon will enter the scintillator and be stopped. In this case, the muon decays and its decay products will cause a scintillation. The time interval between these two signals is very small. This event is defined as a *double*. The data received by hits such as these can be used to calculate the average muon decay time.

The following manual will serve to provide the user with a basic understanding of the electronics involved in the hardware of the PC board, simple set up instructions, and suggestions for a few possible uses for the apparatus.

II. Description of Parts

A. Scintillator Paddles

The acrylic-based scintillator plastic generate a light flash in response to a charged particle transit. Plastic scintillator was chosen because it is relatively cheap, can be made to fit any shape, and has a fast response time. The paddles are enclosed in a light tight wrapping to exclude ambient light from getting to the phototubes.

B. Photomultiplier Tubes

The photomultiplier tube (PMT) picks up the light produced in the scintillator. This device, by means of the photoelectric effect and a current multiplier, converts light into current. The device has a very high gain and allows small traces of light to be picked up and amplified. Unfortunately, it tends to be quite expensive. This device also has the disadvantage of a relatively low quantum efficiency (approximately 15 to 20%).

A photocathode is positioned at the front of the tube to induce the photoelectric effect. Directly after the photocathode is an electrode that focuses the electrons produced into a beam aimed at the multiplier. (See Figure 1) Through the use of a high voltage source, in this case the high voltage base attached to the PMT, a stepladder, so to speak, is created using incremental levels of potential energy. An electron focused into the multiplier is accelerated by the potential difference into the first dynode. A collision occurs, freeing several electrons that, for the most part, continue along the intended path of the original electron. From here, each of these electrons is accelerated towards the next dynode, where the same result occurs on a bigger scale. This process continues down the "ladder" until the electrons reach the anode. At this point they are collected as a current.

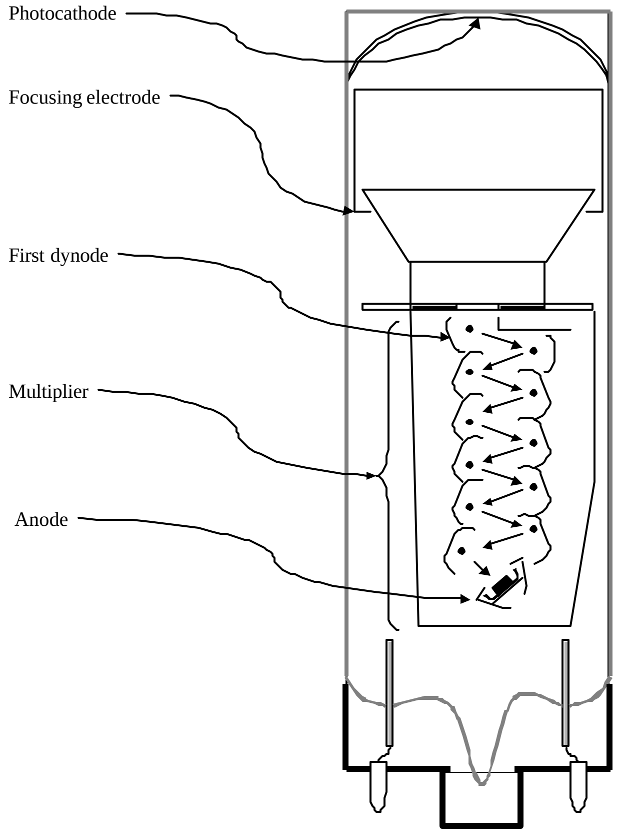


Figure 1: Photomultiplier

C. High Voltage Converter

The attached high voltage base uses a resonant converter and a Cockroft-Walton (CW) chain to create a high voltage source for the PMT safely and efficiently. With the system involved, very little energy is lost to waste heat, and +5V is the only necessary supply voltage. This also allows the user to reach the desired level of energy without the use of a costly high voltage box. This is a great advantage and makes the unit quite cost effective. A brief explanation of the high voltage converter, along with a schematic diagram, appears in Appendixes A and B, respectively.

In addition, the high voltage base serves to relay the signal from the PMT to the PC board with as little interference as possible. As this is done through a simulated coaxial cable built in to the board, the noise level is significantly reduced. The total noise received from the high voltage source is less than 2mV.

For more information on the high voltage base, consult "HV Resonant Converter for Photomultiplier Tube Bases" found in Appendix D.

D. QuarkNet Coincidence Logic Board

By following the signal chain received from the high voltage base and transmitted to the PC board, the reader will gain a better understanding of each individual involved part, as well as their combined use in the device. A schematic diagram for the board can be found in Appendix C. The reader may also refer to Figure 2 below.

i. LEMO Connector

The anode signal cables are attached to the coincidence logic board using a standard LEMO connected. All input channels are terminated to match a 50-ohm cable.

ii. x10 Amplifier

This amplifier converts the input signal to one that is big enough

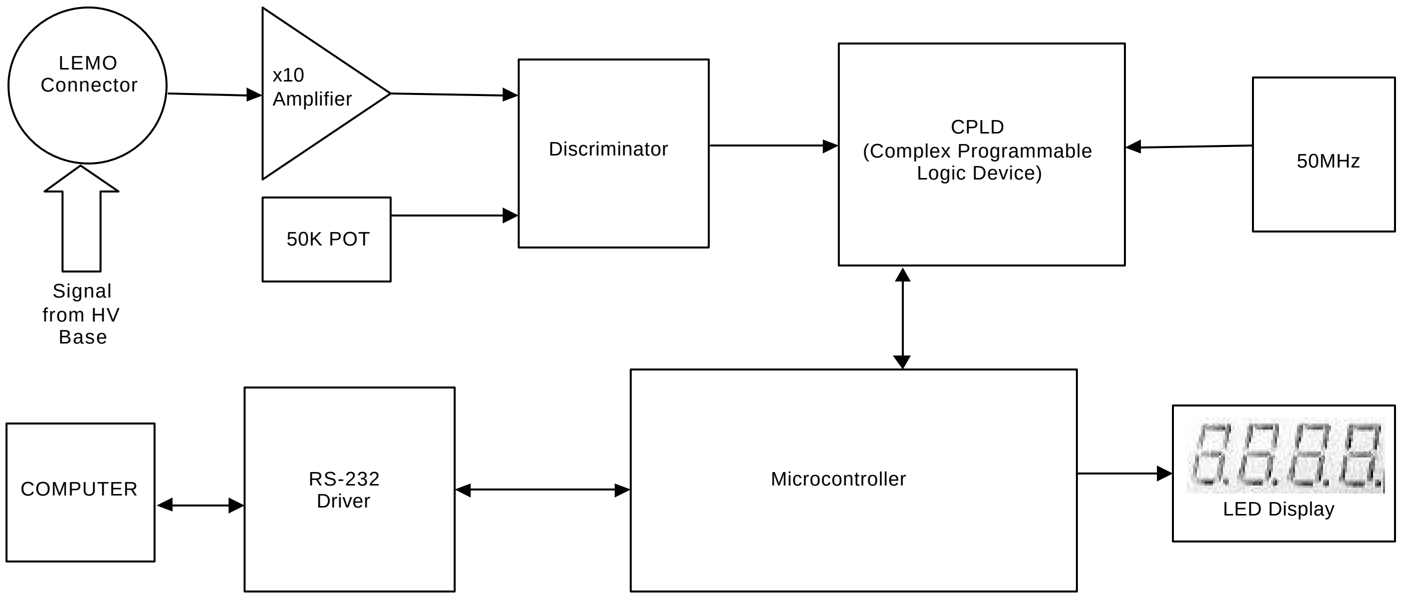


Figure 2: Block diagram for QuarkNet Coincidence Logic Board

to be read by the discriminator. Further motivation in using an amplifier, as opposed to lowering the threshold voltage (see 50K POT for more information), includes the reduced significance of local noise to the signal. Noise at 2 mV is likely to upset a 50mV signal; the same amount of noise will hardly effect a signal at 500mV. Figure 3 below shows signal patterns before and after passing through the x10 amp. Channel 1 shows the input signal and Channel 2 shows the output signal on a scale ten times as large. The amplifier preserves signal fidelity.

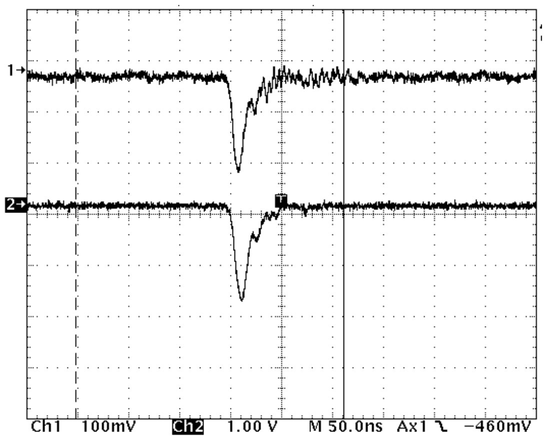


Figure 3: x10 Amplifier

iii. 50K Potentiometer (POT)

The purpose of this potentiometer is to set the threshold voltage for the discriminator. As actual particle hits are expected to be close to -500mV, the threshold level is nominally set at approximately -300mV. The threshold for each channel can be independently tuned from 0mV to 820mV. (See discriminator for more information.)

iv. Discriminator

The discriminator, also commonly known as a comparator, compares the threshold voltage defined by the potentiometer to the voltage level of the input signal. If the input signal exceeds the threshold voltage, the discriminator outputs a logical '1' signal. If the level is not met, it outputs a logical '0' signal. The Figure 4 below shows the signals before and after going through the discriminator. Channel 1 shows the pulse immediately after the x10 amplifier at the negative input of the discriminator. Channel 2 shows a logical '1' signal after the pulse went through the discriminator. The reference channel (R1) shows the threshold signal at the positive input of the discriminator. A capacitor is included in positive feedback to the threshold signal in order to lengthen the pulse signal to 80ns. This ensures that the CPLD will have sufficient time to recognize the pulse.

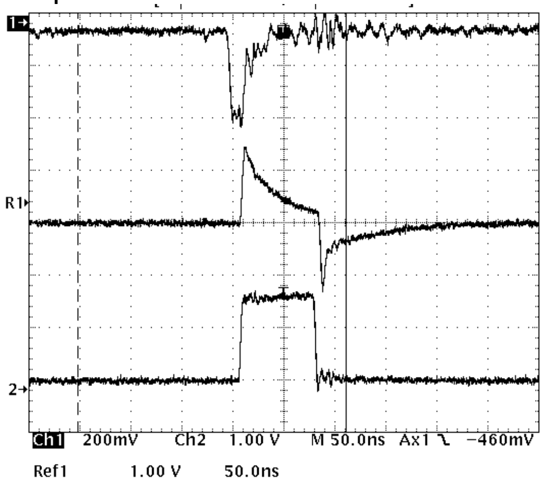


Figure 4: Discriminator

v. Complex Programmable Logic Device (CPLD)

The CPLD contains the coincidence logic of the board. It can be dynamically changed with user-configurable control registers.

Each of the channels can be individually enabled or disabled. The multiplicity level can be set from one to four. (A multiplicity of one requires that a single hit occur on any enabled input channel, whereas a multiplicity of four requires a coincident hit on all four channels.) Also, input Channels 2, 3, and 4 can be set as a 'veto' channel. If a veto is enabled, the CPLD requires that the veto channel *does not* fire to trigger an event. The veto function is useful when dealing with three or more paddles, and can be used to reduce background noise when carrying out muon lifetime experiments.

The CPLD can also be programmed with different 'gate widths'. The gate width is the amount of time allowed between hits considered coincident. The default gate width is one cycle, but can be altered from one to up to six cycles.

A 10-bit Delta T (Δt) counter is included in the CPLD. This measures the time interval between the two pulses of a double. The counter starts each time a coincidence is satisfied. If the counter overflows before a second pulse is received, then the event is reported as a single. If, however, a second pulse is received before the counter overflows, then the counter is stopped and the event is reported as a double. The counter value can be read by the microcontroller when the event is processed. The maximum Delta T window is 20us, or 1000 counts.

vi. 50MHz Crystal Oscillator Clock

This 50MHz oscillator sets the frequency for the CPLD at 20ns/cycle. This is why it is important to lengthen the input signal to 80ns with the capacitor. It ensures that a signal will be recognized by the CPLD. The CPLD also divides the clock frequency by eight to provide a 6.25MHz clock for the microcontroller.

vii. Microcontroller

The microcontroller acts as the hub of the processing system and the main buffer between the user and the electronics. All configuration parameters for the CPLD can be set through the microcontroller. The microcontroller measures trigger event timing using its timer capture/compare module. A 48-bit counter runs continuously. When an interrupt pulse is received from the CPLD, the counter value is recorded. This value is subtracted from the previous event’s recorded counter value to give a time interval between triggers.

Also, after a trigger event interrupt, the microcontroller reads and buffers the contents of all status registers in the CPLD as well as the 10-bit Delta T counter. The microcontroller then checks the data to see what type of event has occurred. If the event has passed the trigger requirements, then the microcontroller will display the data to the user terminal. Below is an example of the output data.

```
…
0002FBFA 13
0008AB2E 13
0001F712 13
0001C021 53 02 002B
000784B5 13
00022D4E 13
000323FD 13
```

```
0002299E 13
…
```

Each line represents a separate event. All numbers involved are hexadecimal. The first column is the elapsed time interval since the previous event trigger. The second column is the value of the first CPLD status register (QuarkStatA). The third and fourth columns only appear in the event of a double. The third column is the value of the second CPLD status register (QuarkStatB). The fourth column is the value recorded by the Delta T counter. Refer to Appendix E for bit significance of the CPLD status registers.

In addition to the streaming data, the coincidence board also keeps track of some simple statistics on its own. The microcontroller keeps individual scalers for each channel as well as a total trigger scaler. These values can be displayed by issuing commands to the coincidence board with a PC. In addition, for convenience, a manual scaler reset switch and a four-digit hexadecimal visual scaler are provided on the face of the board. With this switch, the LED display, and appropriate default settings on startup, the board can, in theory, be used to do some very simple experiments without the need for a PC.

viii. RS-232 Driver

The RS-232 driver acts as an interpreter between the controlling PC and the microcontroller. It serves to convert logic pulses received from the microcontroller (0V/+3.3V logic) to RS-232 standard, higher voltage pulses (-10V/+10V) that are less likely to be distorted by outside noise or the resistance of the cable used. The RS-232 configuration used is 9600 Baud, 8-bits, no parity, 1 stop bit, and no flow control.

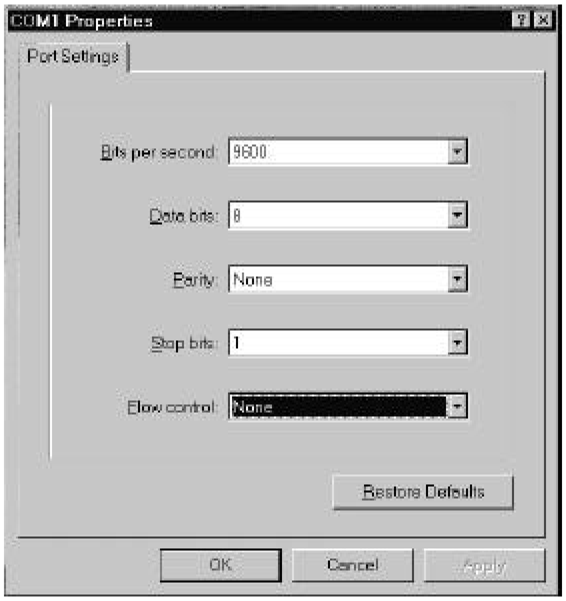
III. Operations

A. Setting Up

The following parts are necessary to fully assemble your QuarkNet Scintillator System: one to four scintillator paddles with connected photomultiplier tubes, one high voltage converter base for each photomultiplier tube, LEMO connector cables for each paddle/tube/base set, one +5V daisy-chained jumper cable for each set, one QuarkNet Coincidence Logic Board, one wall adapter power supply, one RS-232 connector ribbon cable, and a PC from which to run the board.

To get started, attach the high voltage bases to the photomultiplier tubes. Plug the power supply into one power jack on the board and into a local outlet. Connect the high voltage bases to the QuarkNet PC board by plugging one connector of the LEMO cable to the base and the other to the appropriate LEMO jack on the PC board. Plug the daisy-chained jumper cable into the other power jack on the board and one power jack of the first high voltage base. To power the additional bases daisy chain their power jacks using the jumper cables. Plug the RS-232 ribbon cable into the 9-pin connectors on the board and on the PC. (For additional setup help, consult the diagram found in Appendix F)

To set up the computer interface, simply go to your 'Start Menu', select the 'Find' option, and select the 'Find Files or Folders' option. Search your local hard drive for 'HyperTerminal'. When the search is complete, select the 'HyperTerminal' icon with a file size of 1K (it should be a 'Shortcut'). This will take you to the HyperTerminal interface and guide you through the setup procedures. When you reach the 'Port Settings' interface, enter the information asked for as 9600 bits per second, 8 data bits, no parity, 1 stop bit, and no flow control. Figure 5 below illustrates the options that should be chosen.


COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Figure 5: Port Settings

You are now ready to begin.

B. HyperTerminal

The data must be received in a Terminal emulator such as HyperTerminal. This program is included with Windows 95/98/2000/NT and can be accessed easily. You may, however use another terminal emulator if you wish. The user will access the QuarkNet Coincidence Logic Board through this interface. Parameters can be set and data can be displayed directly through HyperTerminal. For a HyperTerminal command list, see Appendix G. The list shown is a copy of the text displayed in the help menu as well as a more detailed explanation of commonly used commands.

C. Possible Uses

There are countless potential uses for the QuarkNet Scintillation System. It is up to the user to design experiments that take advantage of the resources and data provided by the device. This section will outline two possible uses for the system, as well as discuss how to select and analyze the desired data in each case.

i. Basic Rate Measurements

This simple experiment can be used to compare rates of incoming charged particles in different situations. Rates can be compared between the inside and outside of a building, at different heights, at different times of the day, with different materials on top of the paddles, and with different paddle configurations. The paddle configurations can be, but are not limited to, adjustments in angles, distance between paddles, and overlapping area.

a. Working in HyperTerminal

When everything is physically connected, the user should start the HyperTerminal interface. The user should note that the default settings do not allow you to see what you are typing. The 'EC' (enable character echo) command may be typed in to allow the user to see what they are typing. This command can be used as a comfort in most cases, but is not necessary.

In this rate experiment, it will be beneficial to display all incoming data. Consequently, the 'ES' (enable singles) command will provide the user with more data. For rate data experimentation specifically on low-energy muons, the 'SS' (suppress singles) option can be used instead.

The 'WC NN' (write command) option will allow the user to specify which channels to enable, specify vetoes, and set the coincidence level. See bit diagram in Appendix F for explanation of specific bit choices and their hexadecimal notation. See Appendix H for an explanation of the hexadecimal system. The combination of information given by these appendices will allow the user to set the 'NN' in the command to the appropriate hexadecimal number. As the default settings enable all channels with a veto on Channel 4 and require a coincidence of two (WC DF), a user working with two paddles need not adjust the settings.

The 'WW N' option allows the user to set the gate width (see CPLD) from 1 to 6 simply by replacing 'N' with the number desired. In this experiment, the gate width should be set to 1 as the user need not concern himself with doubles.

b. Capturing Data

Now that everything is set up, the user can begin recording data. This is done by capturing the text in a separate file. (Make sure to save the file in a readily accessible directory.) To start capturing the data, select 'Capture Text' in the 'Transfer' menu. This will begin the data sequence. When you have collected sufficient data, find the 'Capture Text' option in the 'Transfer' menu, which will now have additional options. Selecting stop ends the data collection. The user can now access this data in the designated file.

c. Working in Excel

The data must now be converted to Microsoft Excel format (or another spreadsheet based program) for further analysis. Open Excel and chose the open file option. Open the captured text file. As it is a text file, make sure to select the 'All Files (*.*)' option in the 'Files of type:' box. The 'Text Import Wizard' will appear. On the first screen select the 'Fixed width' option and press 'Next >'. The next screen will allow the user to adjust the position of the column breaks if desired. If adjustment is necessary, simply follow the instructions provided. (If the RS-232 cable is not used directly, the data format may not be uniform and will be harder to analyze.) When this is completed press 'Next >'. The following screen allows you to set the data format for each column. All columns should be set to text format. To do this, select each column individually and click on the 'Text' option. When finished, click the 'Finish' button. The data displayed should be in two columns with an occasional double register consisting of four columns. (For explanations of each column, see Microcontroller above.) If there is no data in Row 1, delete it.

The first step in analyzing the data is to convert the results from hexadecimal to decimal. To do so, an add-in application must be enabled. In the 'Tools' menu, select the 'Add-ins…' option. When the screen appears, check the 'Analysis ToolPak' box and select 'OK'. Now you can select a column to designate to the new decimal numbers. Select the E1 box and click on the '=' button to write a formula to it. (Column E is suggested, but is by no means important, and another column may be chosen if desired. This applies to all mention of specific Columns in this paper.) Type "HEX2DEC(A1)" in the formula space. This will convert the hexadecimal results in box A1, and place the result in E1. To apply this conversion to all of the data, simply click and drag the square in the lower right-hand corner of the E1 box down to the desired point. This step may be repeated for the data in Column D if desired by placing the formula in Column F.

d. Data Analysis

The two crucial factors of rate experimentation are the number of hits and time interval over which they were collected. The exact time can be calculated by summing Column E. As Column E records the elapsed time since the last hit, summing Column E will produce a total time in 20ns units. To sum Column E, click on the button labeled 'E' above the data in the column. This will select the entire column. By pressing the sigma (Σ) button, you can add up the highlighted boxes. The result will be placed under the last full box in that column. If the sigma button is unavailable, select a box and click on the '=' button to write in a formula. Type 'SUM(E1:ENNNN)', where 'NNNN' represents the number of the last data box in Column E. The average frequency can be calculated by dividing the number of hits by the total time. This number can be compared to data taken under different conditions.

ii. Average Muon Decay Time Experiment

This experiment uses doubles, events interpreted as decaying low-energy muons, to determine the average muon decay time.

The data must be collected in HyperTerminal with suppressed singles. Follow directions found in the Working in HyperTerminal section above. Instead of using the 'ES' option, suppress singles by typing 'SS'. The gate width ('WW N' option) should be adjusted accordingly. If you have slow scintillator paddles or phototubes, you may feel it necessary to extend the gate width and allow more time for the second pulse to be picked up. Consequentially, this causes more noise to register and false doubles to be recorded.

Capturing data is explained in the Capturing Data section above. Simply follow the instructions given. To import the data into Excel, follow the first paragraph of the Working in Excel section above. (Please note that in this experiment, all data should be in four-column format.)

For this experiment, the

primary concerns in data collection are the elapsed Delta T time (decay time) and the frequency of events for each bin of Delta T times. Column D includes the Delta T data in hexadecimal and must be converted to decimal format using the method described in the Data Analysis section above. Place the converted numbers in Column E

As the PC board registers all doubles, certain data points must be thrown out. The user should determine to which channel the bottom paddle is connected. Only doubles registered on the bottom paddle are valid. Column C describes, in hexadecimal, which paddle received the double. (01 represents Channel 1, 02 is Channel 2, 04 is Channel 3, and 08 is Channel 4) Delta T values that are impossibly small (8 counts or less) should be thrown out, as well. Column F can be formulized to throw out both types of invalid hits. Select the F1 box, click on the '=' button, and type 'IF(OR(C1="0N",E1<8),0,E1)' where '0N' represents the hexadecimal channel(see above) that is connected to the bottom paddle. This uses logic to throw out the undesirable data. To apply this conversion to all of the data, click and drag the square in the lower right-hand corner of the F1 box down to the desired row.

To get an accurate result, the approximate noise level must be calculated by isolating all false doubles. This can be done by applying the following formula to the G1 box, and, in turn, the entirety of Column G in the manner explained in the above paragraph: 'IF(OR(C1="0X",E1<8),0,E1)' where '0X' represents the hexadecimal channel(see above) that is connected to the top paddle.

This data can be separated into bins using the histogram function. The noise level can be graphed next to the doubles data, and subtracted from it if desired. By using an exponential fit line of the subtracted data, the average muon decay time can be calculated. Display the formula of the fit line. It should be in '$y = ke^{xt}$ ' format. The inverse of the t value will give you the average muon decay time of your data. The accepted value for the muon lifetime is 2.19703μs. From this information, the user can calculate the percent error of their calculation.

D. Default Settings

The default settings for the board are mentioned above in the Working in HyperTerminal section.

Suggested Readings

1. Beiser, Fred, Colleen Twitty, and Howard Matis. “Tips to Assemble the Berkeley Lab Cosmic Ray Detector”. Lawrence Berkeley National Laboratory, Version 1.01, 2000.\
2. Leo, William R. *Techniques for Nuclear and Particle Physics Experiments: A How-To Approach*. Springer Verlag, 1994.
3. Matis, Howard, and Colleen Twitty. “Guide to Using the Berkeley Lab Cosmic Ray Detector”. Lawrence Berkeley National Laboratory, Version 1.01, 2000.
4. “The Particle Adventure: the funadamentals of matter and forces”. Particle Data Group, 2000. http://www.particleadventure.org/. 31 July 2001.
5. *Photomultiplier Handbook*. Burle Technologies, Inc., 1980.
6. "QuarkNet". http://quarknet.fnal.gov. 31July 2001.

Appendix E: CPLD Control and Status Registers

All the functions of QuarkNet Coincidence Logic Board are controlled by changing the values of bits (binary digits) that are stored electronically on the board. For example, there is one bit on the board reserved as the *Channel One Input Enable Bit*. If this bit is cleared (assigned a value of 0), the Channel One Input is disabled. If the bit is set (assigned a value of 1), the Channel One Input is enabled. There are also bits reserved for *Channel Two Input Enable*, *Channel Three Input Enable*, *Channel Four Input Enable* and all other configuration options for the board.

There are also bits reserved for recording information about particle triggers. For example, there is one bit reserved as the *Channel One Hit Bit*. The CPLD on the logic board sets this bit when a hit is received on the Channel One Input. It clears the bit after the hit has been acknowledged by the board's microcontroller. There are also bits reserved for *Channel Two Hit, Channel Three Hit*, *Channel Four Hit*, and all other details recorded about particle triggers.

For the sake of efficiency, bits are usually placed together into *registers*. A *register* is a group of bits that must be written and read as a group. In other words, in order to change the value of any bit in a register, the entire register must be written. To check the value of any bit, the entire register must be read. Registers that contain bits used for configuring the function of some electronic unit are typically called *control registers*. Registers that contain feedback information from an electronic unit are typically called *status registers*. Control registers can be written externally to control the unit, and status registers can be read externally to check the unit's status.

In the coincidence logic unit of QuarkNet board, most of the control and status registers are 8 bits wide. However, when using the text-based user interface, nearly all numbers sent to or received from the board are expected to be in hexadecimal format. This includes all the trigger timer intervals and delta-t timer intervals, as well as the contents of all the control and status registers. Therefore, when writing a value to the CPLD control register (using the WC command) one must convert the desired 8-bit binary settings for the register to a 2-digit hexadecimal number.

As an example, if one wants to configure the coincidence logic with no veto, a coincidence level of 2, and channels 1, 2, and 3 enabled, then one can refer to the bit diagram for the QuarkControl register. Based on the bit diagram, the control register must be set with the bit sequence 00010111. Converted to hexadecimal, this is 0x17. Therefore, the user would need to enter the command 'WC 17' into the terminal in order to configure the board as desired.

The same is true for the feedback status registers. For each particle trigger, the value of the QuarkStatA register is displayed to the user terminal. Say, for example, the number displayed is 13. This must be interpreted as a hexadecimal number. 0x13 expands to the 8-bit binary sequence 00010011. Referring to the bit diagram for the QuarkStatA register, one can see that this sequence means a trigger has occurred and channels 1 and 2 were hit.

CPLD Control and Status Registers

Control Registers

QuarkControl

7	6	5	4	3	2	1	0
Veto select bits. Options range from 0 to 3. 0 means no veto and is represented by 00. 1 to 3 select vetoes on channels 2 to 4, respectively. 1 is 01, 2 is 10, 3 is 11. Input 1 cannot be set as a veto.		Set the multiplicity level. Options range from 0 to 3, corresponding with multiplicity ranges of 1 to 4. 0 is represented by 00, 1 is represented by 01, etc...		Enable/disable input channels 1-4. Bit 0 corresponds to input 1, Bit 1 corresponds to input 2. When the bit is 1 the channel is enabled, when it is 0 the channel is disabled.			

QuarkWidth

2	1	0
Defines the number of 20ns CPLD clock ticks in the gate width. Only numbers 0 to 6 are used. 0 turns gate width function off, 1 corresponds to 001, 2 to 010… 6 to 110.		

Status Registers

QuarkStatA

7	6	5	4	3	2	1	0
Not used	Set when double occurs on any channel.	Not used	Set when trigger is satisfied.	Specific channel hit bits for confirmation of satisfied multiplicity requirement. 0 corresponds to channel 1, 1 to channel 2, etc...			

QuarkStatB

3	2	1	0
Specific doubles channel hit bits. 0 corresponds to a double on channel 1, 1 to channel 2, etc...			

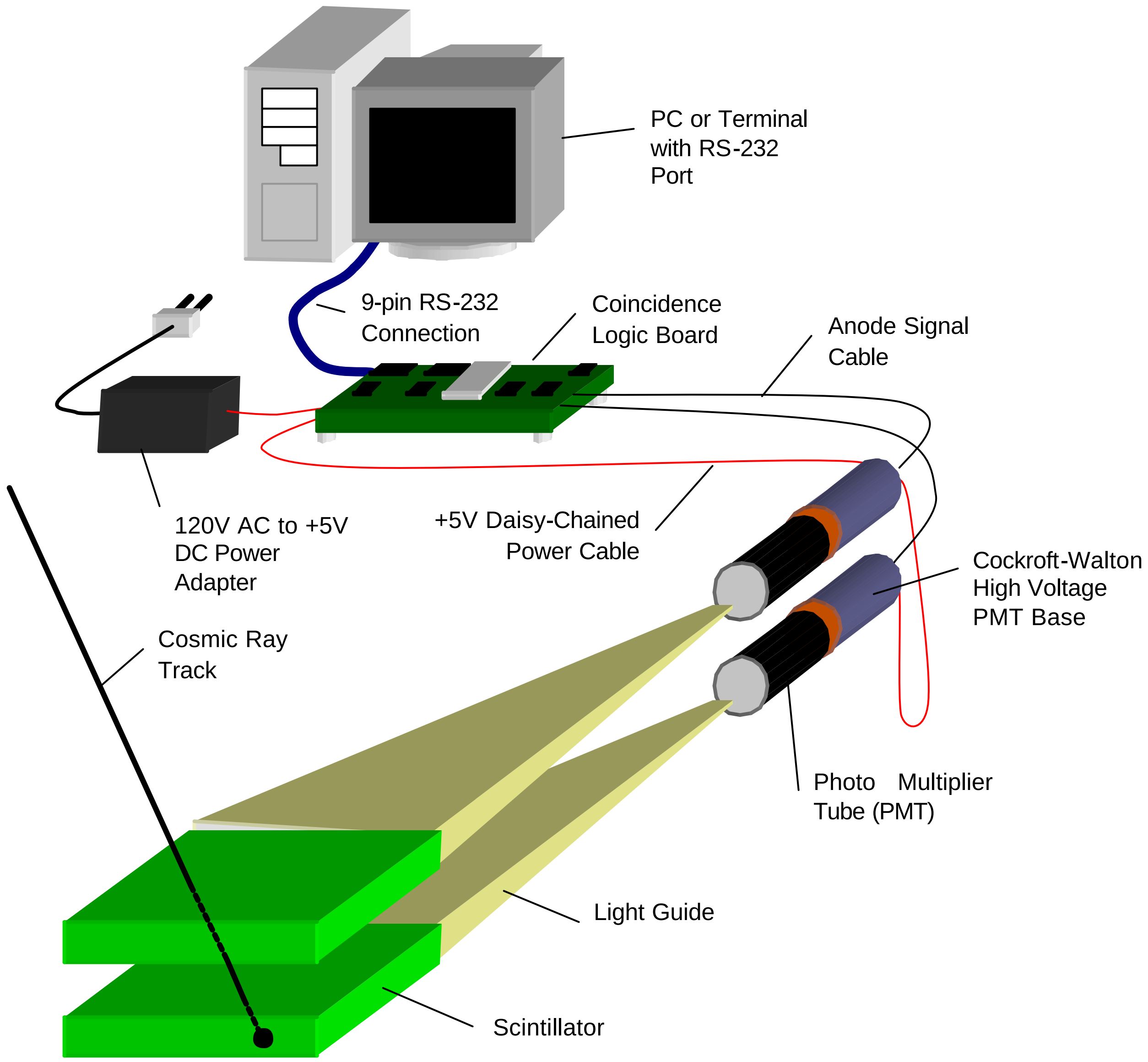
Delta T count (low)

7	6	5	4	3	2	1	0
The lower 8 bits used in the 10-bit Delta T counter. It counts at 50MHz. Used for measuring the interval between the two hits of a double.							

Delta T count (high)

1	0
The upper 2 bits of the Delta T counter.	

Appendix F: QuarkNet Scintillation System Assembly Diagram


PC or Terminal with RS-232 Port
9-pin RS-232 Connection
Coincidence Logic Board
Anode Signal Cable
120V AC to +5V DC Power Adapter
+5V Daisy-Chained Power Cable
Cockroft-Walton High Voltage PMT Base
Cosmic Ray Track
Photo Multiplier Tube (PMT)
Light Guide
Scintillator

Appendix G: COMMAND LIST
(as seen in QuarkNet help interface)

A list of available commands is given below:

NOTE: Leading zeroes are not needed for address or data specifications.

1. HE - Display this help menu.

2. ID - Display the date on which the current code was assembled.

3. DD or DI MMMM (NNNN) - Display data/io memory from location MMMM to NNNN.

4. WD MMMM DDDD - Write 16 bit data word DDDD to location MMMM.

5. WB or WI MMMM DD - Write 8 bit data/io word DD to location MMMM.

6. CL - Clear terminal screen.

7. DC - Display current contents of the quark control register.

8. WC NN - Write the value NN to the quark control register.

9. DS - Display current scalar counter values.

10. RS - Reset scalar counters.

11. DT - Display current trigger counter value.

12. EC - Turn character echo ON.

13. NE - Turn character echo OFF.

14. DW - Display current gate width value.

15. WW - Write gate width register (in the range of 1..6).

16. ES - Enable display of single triggers.

17. SS - Suppress display of single triggers.

18. TP NN - Timer Prescaler. Set the trigger timer interval to be prescaled by NN bits, where NN is a hex number from 0x0 to 0x10.

19. BS - Bin Size display. Displays the current bin size (in hex, units of ns) for the trigger timer and the Delta T timer. The trigger timer bin size is dependent on the prescaler value set using the TP command.

Appendix H: Binary and Hexadecimal Numbers

John Lofgren, 7/23/01

Introduction

When numbers are written down anywhere, they are written using a base. Most of us normally think and work in base-10 or *decimal* numbers, where we use 10 distinct digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. However, most computers (or any digital electronic device, including the QuarkNet Coincidence Logic Board) are only able to do calculations in base-2 or *binary* numbers, with only two distinct digits: 0 and 1. Computers can store a single ‘binary digit,’ or ‘bit’ by either storing charge in a physical location or not storing any charge in that location. The computer can interpret stored charge as a 1 and no stored charge as a 0.

When numbers are written that have to do with the operation of computers it is most efficient to write them in a base that is most directly applicable to the computer. Therefore, decimal numbers are no longer as useful when talking about computers. Binary numbers are more useful, but even so, writing large numbers using only 0’s and 1’s can be rather tiresome and hard to read. For this reason, base-16 or *hexadecimal* (or simply *hex*) numbers are a great benefit. Hexadecimal uses 16 distinct digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. As will be shown, it is very simple to convert numbers between binary and hexadecimal representations.

When working with number in various bases, it is important to be careful to indicate the base of each number. So, we add prefixes or suffixes to numbers to be clear about which base we are using. Different computer languages and component manufacturer manuals use various syntax for base prefixes and suffixes. Some common prefixes and suffixes are as follows:

Base	***Prefixes***	Suffixes
Binary	b^, b’, %b	B
Decimal	d^, d’, %d	D
Hexadecimal	h^, h’, %h, 0x	H

For this discussion we will use the prefixes ‘b^’ and ‘d^’ for binary and decimal numbers and the prefix ‘0x’ for hexadecimal numbers.

Binary and Hexadecimal in Relation to Decimal

To begin, let us return to decimal numbers, where we are most comfortable. When we put some decimal digits together to form a decimal number, like **d^43210**, the columns of the number carry different significance.

d^10000’s column	**d^1000’s column**	**d^100’s column**	**d^10’s column**	**d^1’s column**
4	3	2	1	0

Because there are ten choices of digits for each column, each successive column is ten times as significant as the previous one. To find the total value of the number, we multiply the digit in each column by its significance and find the total sum.

(d^4 x d^10000) + (d^3 x d^1000) + (d^2 x d^100) + (d^1 x d^10) + (d^0 x d^1) = d^43210

The same pattern follows for binary numbers. If we put some binary digits together to form a binary number, like **b^10110**, the columns of the number carry different significance. To relate back to decimal format, we will represent the significance of each column as a decimal number.

d^16's column	**d^8's column**	**d^4's column**	**d^2's column**	**d^1's column**
1	0	1	1	0

In the case of binary, there are only two choices of digits for each column, so each column is only twice as significant as the previous one. Again to find the total value of the number, we multiply the digit in each column by its significance and find the total sum.

(b^1 x d^16) + (b^0 x d^8) + (b^1 x d^4) + (b^1 x d^2) + (b^0 x d^1) = d^22

Therefore, the numbers b^10110 and d^22 have the same value. They are equal, but are represented using different bases.

Again, this can be extended to hexadecimal numbers. Let us use the example hexadecimal number **0x5AB0F**.

d^65536's column	**d^4096's column**	**d^256's column**	**d^16's column**	**d^1's column**
5	A	B	0	F

Using hexadecimal, each successive column is sixteen times as significant as the previous. When finding the total value of the number in decimal format, we follow the same pattern as before.

(0x5 x d^65536) + (0xA x d^4096) + (0xB x d^256) + (0x0 x d^16) + (0xF x d^1) = ???

Here is where things start to get a little tricky for our brains that are used to thinking in decimal. We aren't used to multiplying using the digits A, B, C, D, E, and F. Therefore, it can help our brains if we quickly convert these alphabetic digits to their two digit decimal equivalent: 0xA = d^10, 0xB = d^11, 0xC = d^12, 0xD = d^13, 0xE = d^14, 0xF = d^15. (The numeric digits, 0-9, are equivalent in either base.) Then the multiplication and summing looks more straightforward.

(d^5 x d^65536) + (d^10 x d^4096) + (d^11 x d^256) + (d^0 x d^16) + (d^15 x d^1) = d^371471

Therefore, the numbers 0x5AB0F and d^371471 are equal.

Conversion Between Binary and Hexadecimal

When talking about numbers stored somewhere in a computer, it is most direct to represent these numbers as they are stored in the computer: in binary. However, it can be very tiresome to write large binary numbers. For example, the hexadecimal number 0x5AB0F used in the example above is written in binary as b^1011010101100001111. This is very inefficient and difficult to read. For these reasons, hexadecimal format is often used when talking about numbers in a computer.

It is very simple to convert between binary and hexadecimal representation. Because 16 is an even 4th power of 2 (2^4 = 16), each unique sequence of four binary digits corresponds to a single unique hexadecimal digit, and visa-versa. The conversion of a four digit binary number to hexadecimal can be done just as above using column significance, multiplication, and summing. For example, take the binary number **b^1010**.

d^8's column	**d^4's column**	**d^2's column**	**d^1's column**
1	0	1	0

Then the decimal value is:

(b^1 x d^8) + (b^0 x d^4) + (b^1 x d^2) + (b^0 x d^1) = d^10

And switching this decimal value to hexadecimal is simply a matter of converting any two-digit decimal results to their single digit alphabetic hexadecimal equivalent (d^10 = 0xA, d^11 = 0xB, etc.). A conversion table can therefore be generated:

Binary	**Hexadecimal**
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8

1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

The thing that makes converting to hexadecimal so easy and convenient is that this 4-bit grouping pattern can be followed for converting the entire length of a binary number. Each group of four binary digits can be converted to one hexadecimal digit. Using the same example number as above, b^1011010101100001111, we can see how this works. The binary number is split into groups of four digits. (You may like to imagine 0's filled in on the left to make a even multiple of four digits.) Each group of four binary digits can then be compressed to one hexadecimal digit.

Binary	101	1010	1011	0000	1111
Hexadecimal	5	A	B	0	F

Therefore b^1011010101100001111 = 0x5AB0F. This process is easily reversed for converting from hexadecimal to binary, in which case each single hexadecimal digit is expanded directly to four binary digits.