01 May 1994

Automation of the Electrostatic Aerosol Classifier and the Continuous Flow Cloud Diffusion Chamber

Daron W. Dryer

AUTOMATION OF THE ELECTROSTATIC AEROSOL CLASSIFIER AND THE CONTINUOUS FLOW CLOUD DIFFUSION CHAMBER

DARON W. DRYER
DEPARTMENT OF MECHANICAL ENGINEERING

ABSTRACT

Analyzing cloud condensation nuclei through the use of a Continuous Flow Thermal Diffusion Cloud Chamber (CFD) and an Electrostatic Aerosol Classifier (EAC) can be very time consuming. An alternative to manually monitoring these devices is to automate the equipment, thus freeing personnel for other research as well increasing CCN data collection.

Automation of the EAC will be carried out by digital command through the use of a Quick Basic program. This program will be run from an IBM personal computer outfitted with a digital input/output board.

RS-485 Standard will also be incorporated into the Quick Basic program in order to control the CFD.

INTRODUCTION

This investigation involves automation of two instruments used in aerosol research. Aerosol here means liquid or solid particles suspended in air. The first instrument is called an electrostatic aerosol classifier (EAC). In the EAC, particles carrying one or more electrical charges are passed through an annular region between two cylindrical electrodes which provide an electrical field. At the exit end of the EAC, air passes through a slit in the inner electrode, and this air will contain only particles in a narrow range of electrical mobility. The EAC acts like a filter which passes only particles of a specific mobility. The inlet aerosol is exposed to a device to give the particles an equilibrium distribution of electrical charges, which results in the situation that the aerosol at the outlet of the EAC contains mostly singly charged particles in a narrow size range. Automation of the EAC consists of using a computer to control the voltage of the center electrode. The voltage is supplied by a Bertan model 205C power supply, which is equipped for external control of the output high voltage. The voltage is to be changed in steps, so that with a suitable particle counter at the output of the EAC, the size distribution of the aerosol at the inlet of the EAC can be measured.

The second aerosol instrumentation to be automated is a continuous flow diffusion chamber (CFD). This device exposes an aerosol to a supersaturation, defined as the excess of the relative humidity above 100%. Water drops grow in the CFD on some of the particles, and these drops are counted downstream of the CFD. The supersaturation of the CFD is controlled

by changing the temperature of one part of the CFD, a part called the hot plate. The typical use of the CFD would be to vary this hot plate temperature in steps, in order to record the number of particles as a function of supersaturation. Thus to automate the CFD, a computer is to be used to control the CFD temperature. In this particular case the computer needs to send a set point signal to an Omega model CN6070A controller.

COMPUTER AND SOFTWARE

An IBM compatible computer is used, with DOS 3.3 as the operating system. The programming is done in Quick Basic, version 4.5. For the EAC automation, an I/O board was employed (National Instruments model PC-DIO-96). For the CFD automation a communications port was accessed in the Quick Basic program. Commands were then issued using the RS-232 electrical standard (which Quick Basic requires). A piece of hardware that converts from RS-232 to RS-485 was also employed because the temperature controller requires RS-485, and because eventually more devices will need to be controlled, and RS-485 allows multiple devices to be controlled by one communications port.

PROCEDURE

Controlling the Electrostatic Aerosol Classifier (EAC)

The voltage supply for the Electrostatic Aerosol Classifier is produced by a BERTAN Model 205c high power supply. This instrument can be controlled through 16 bits of digital input. To automate this device, a National Instruments PC-DIO-96 digital I/O board was employed. The PC-DIO-96 was installed in an IBM compatible personal computer. It contains 96 bits of digital input and output, which are arranged into 12 ports containing 8 bits apiece. These ports are labeled as APA, APB, APC, BPA, BPB, BPC, CPA, CPB, CPC, DPA, DPB, and DPC or 0 through 11 for digital I/O functions. Since the voltage supply only requires 16 bits of input, these can be represented by lines 2 through 17 of a standard cable with line 2 being the most significant bit. Only ports APA, APB, and APC were used for cable connection; APA and APB were used for output; and port APC was used for control purposes. Figure 1 illustrates the relationship of the lines from the voltage source to their pin positions in the DIO-96 connector.

Using NI-DAQ Software

The National Instruments package also contained NI-DAQ software that was essential for the use of the I/O board. This software contained a set of functions that could be incorporated into the Quick Basic program through the use of "quick libraries". Once called up with the Quick Basic program through a batch file, these functions made the output of data to the voltage source possible. In order to employ these functions into a program the programmer must type commands in the syntax listed in the NI-DAQ Function Manual along with the information needed for the function to complete its task. When the function has completed its work it will then send back an integer status code revealing whether or not the operation was successful. Figure 2 shows syntax for utilizing NI-DAQ functions and demonstrates how a block of code used to send digital output would appear, as well as giving an explanation for each step.

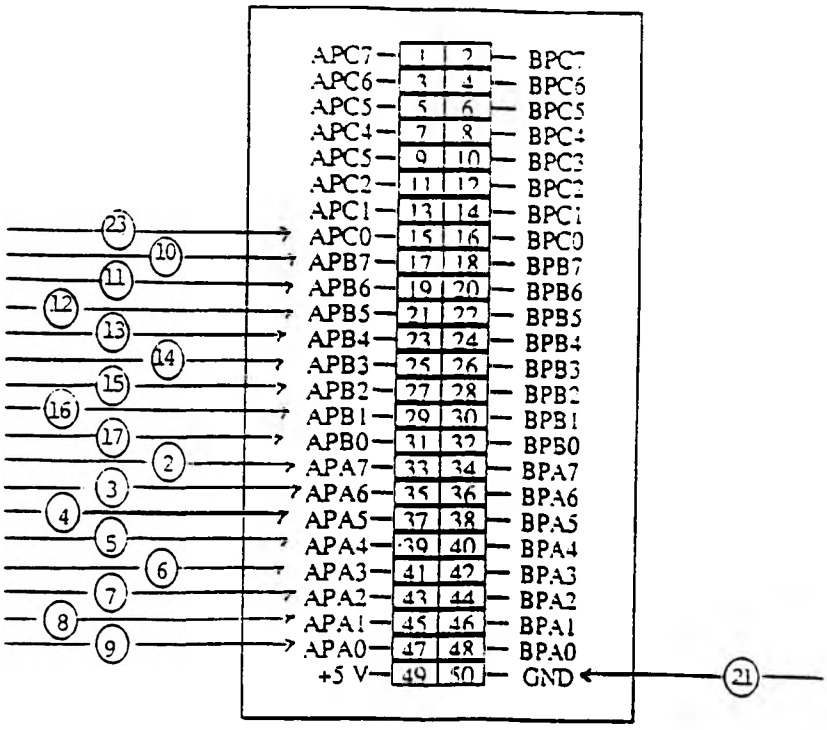

Figure 1. Assembly Connector Pinout for PC-DIO-96 Connector

STATUS% = DIG.OUT.PORT(BOARD,PORT,PATTERN)	Writes digital data to the specified port. BOARD = Integer, represents the I/O board slot or I.D. number. PORT = Inteter, represents the digital I/O port number. PATTERN = Integer, represents the 8-bit digital command.
PRINT STATUS%	Prints the integer status code.

Figure 2. Syntax for Utilizing NI-DAQ Functions

Special Programming Considerations for EAC

Due to equipment requirements, grouping together of ports (treating two 8 bit ports as one 16 bit port) was not possible. This presented a problem that needed to be solved before the writing of Quick Basic could begin. The problem was due to the fact that two 8 bit ports had to be used to send output to an apparatus that required 16 bits. For example, if the user wanted to bring the voltage device to full power the digital command that the device must receive is: $2^0+2^1+.....+2^{14}+2^{15}$ or 65535. Each port can only hold a number of: $2^0+2^1+...+2^7$ (255), any more will over load that port. Therefore, when we enter a command, "255" can be placed in the least significant port, APB, because it contains the lower 8 bits, but "65280" cannot be input in the most significant port, APA, because like APB it is just an 8 bit port.

In order to rectify this inconvenience a method had to be devised to transform the desired digital command into two separate numbers acceptable to the two eight bit ports, yet keeping the same positive logic. The algorithm produced achieved this through three steps.

First, it sorted the initial digital command, given the variable name DIG&, in order to determine the values that must be output through each port. A one dimensional array "Port" was created to store the values that would be held in the most significant port APA. Then each of these values were compared to the digital command by a FOR...NEXT loop until the largest possible value that could be stored in APA was found. It and the remainder were then stored as two variables, TOTAL& and NUM, respectively. It should be noted that the first element in the array "Port" would ideally be 2^{15} or 32768, but in order to keep the array single precision, it was entered as 32767. The algorithm described is shown below.

```
K = 14
PORT(1) = 32767
FOR I = 2 TO 8
PORT(I) = 2^K
K = K - 1
NEXT I
TOTAL& = 0
FOR I = 1 TO 8
IF PORT(I) + TOTAL& < = DIG& THEN
TOTAL& = TOTAL& + PORT(I)
ELSE TOTAL& = TOTAL&
ENDIF
NEXT I
NUM = DIG& - TOTAL&
```

Second, the value to be input into the most significant port APA had to be modified for the eight bit port. This was achieved by calculating to what power two was raised, subtracting eight from this number, raising two to this new value, and storing it as the new variable to be placed in port APA. This portion of the program is as follows.

```
N1 = (LOG(TOTAL&))/(LOG(2))
```

N2 = N1 - 8
NEWTOT = 2^N2

The last step consists of outputting these variables, both NEWTOT and NUM, to their respective ports.

EXAMPLE:

Consider a case in which the digital command 300 would be entered. The largest value that could be stored in port APA would be 2^8 or 256. By taking the natural logarithm of this number and dividing it by the natural logarithm or two, the program finds to what value two was raised.

$$\ln(256)/\ln(2) = 8 \qquad 1).$$

By subtracting eight from that number, then raising two to that new value, a digital command is found that can be placed in port APA.

$$2^{(8-8)} = 2^0 = 1 \qquad 2).$$

Notice, as Figure 3 suggests, that the value "one" places a positive logic in the first bit of the port APA. This is the same location it would have been had 256 been placed in the upper half of a 16 bit port.

LINE NUMBER	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
BINARY NUMBER	0 2	1 2	2 2	3 2	4 2	5 2	6 2	7 2	8 2	9 2	10 2	11 2	12 2	13 2	14 2	15 2
POSITIVE LOGIC			+1	+1		+1			+1							

16 BIT PORT WITH DIGITAL COMMAND OF 300 ENTERED.

APB

LINE NUMBER	17	16	15	14	13	12	11	10
BINARY NUM.	0 2	1 2	2 2	3 2	4 2	5 2	6 2	7 2
POSITIVE LOGIC			+1	+1		+1		

APA

9	8	7	6	5	4	3	2
8 2	9 2	10 2	11 2	12 2	13 2	14 2	15 2
+1							

8 BIT PORTS APA AND APB. DIGITAL COMMAND OF 1 ENTERED INTO APA AND COMMAND OF 44 ENTERED INTO APB.

Figure 3. Port Comparison

Automating the Temperature Controller for the CFD

The water temperature for the hot plate of the CFD is controlled by an OMEGA CN6070A series temperature controller. Unlike the voltage source for the EAC, the temperature controller employs RS-485 serial programming. Unfortunately, Quick Basic would usually only be able to program through the use of RS-232, but by using an Omega RS-232 to RS-485 converter it was possible to communicate with the CN6070A successfully. RS-232 and RS-485 are electrical standards that are published by the Electronic Industries Association (EIA). These two electrical standards provide specifications that dictate voltage ranges for data and control signals in order to insures proper transmission. RS-485 differs from the RS-232 do to the fact that the RS-485 interface allows multiple devices to be connected to the same communications port.

Programming the CN6070A

A communications port had to first be opened in order for the computer to communicate with the temperature controller. The CN6070A responds to ASCII 7 bit code with one start bit, one stop bit, and odd parity. This information had to follow the OPEN "COM1" statement in order to successfully open communications with the CN6070A. A helpful example of this process can be found on pp. 133-134 of the text Programming in Quick Basic by Microsoft Corp.

Next came the task of programming using the syntax required by the CN6070A. The commands required to control the CN6070A are very rigid in regard to the positioning of the characters. Therefore, if an exact syntax was not followed the CN6070A would not respond. The command that must be sent is in the form of a string. This string consists of a start mark, controller number, command letter, parameter number, and parameter to be changed, as well as its units if needed. The start mark consists of a symbol used to inform the temperature controller where the command string begins; for this case it was a "#". The controller number indicates the number assigned to the CN6070A by the operator's manual; this number was 05. The command letter informed the CN6070A of what kind of task it was to perform. The parameter to be changed in this case was temperature, with its units of Celsius. Figure 4 gives a clearer example of how this string command must appear, as well as showing the CN6070A's response. Further information can be found on pages 15-16 and 27-33 in the CN6070A's operator's manual.

Special Programming Considerations For CFD

Future plans for the automation of this system consist of incorporating equipment that count CCN particles produced by the CFD. In order to determine when the particle counter should begin counting CCN particles, the Quick Basic program had to wait until the newly set temperature stabilized. This was accomplished by executing a programmed temperature check every 2 seconds until the set temperature matched the actual temperature of the water. When communicating with the CN6070A it had to be kept in mind that the Quick Basic program not only had to send commands to the CN6070A, but it had to wait for the CN6070A to send responses back. Quick Basic is so fast that it would check its input port before the CN6070A

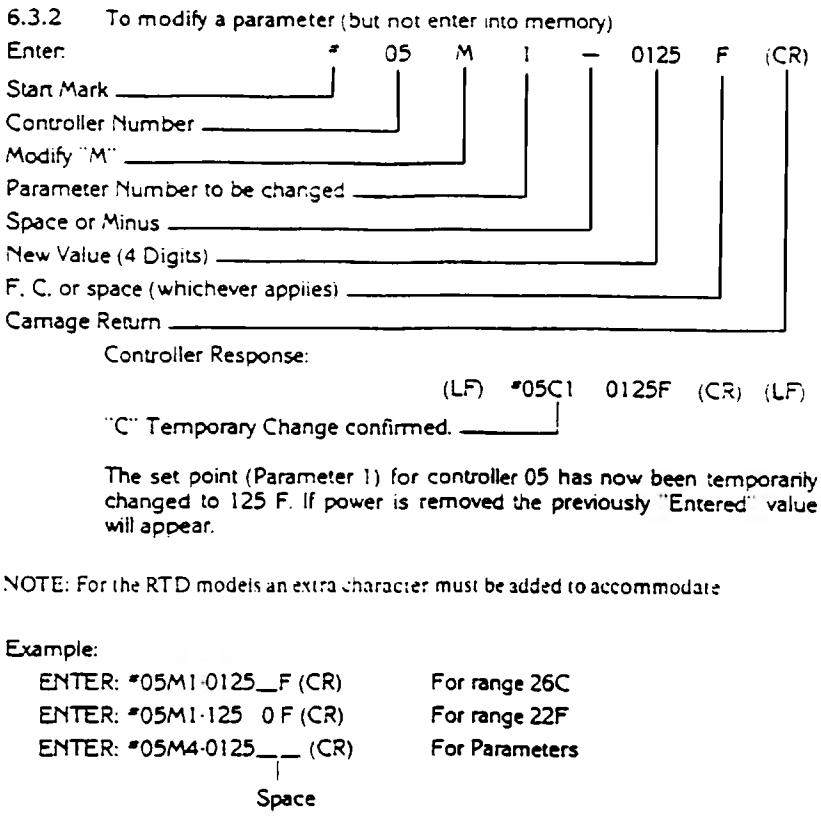

Figure 4. Example of a CN6070A String Command

had a chance to respond. Therefore, when the temperature check algorithm was run the Quick Basic program would always read a "0" for the actual water temperature because no data was yet present. This resulted in the particle counter never starting its count. To avoid this problem a two second pause was incorporated into the code before the section that checked the modem. This two second timer allowed the CN6070A proper time to respond.

Another aspect of this type of programming that deserves to be noted is the transition from integer to string and string to integer that occurred often with the variables. As stated earlier, the CN6070A's syntax required string commands, and sent string responses. In order to transform these variables back and forth into something Quick Basic utilizes, the "VAR" and the "STR$" functions became very valuable. The "LTRIM" statement was also used in order to trim the space that Quick Basic inserts in front of all numbers. This was essential in meeting the CN6070A syntax.

RESULTS

The Quick Basic programs have performed exceptionally well. Through the use of the PC-DIO-96 I/O board and the NI-DAQ software, the Quick Basic program can currently control the Electrostatic Aerosol Classifier's voltage supply with less than one percent error in most cases. The only instance in which the error is greater is if a command of less than five volts is input. This can be attributed to the resolution of 16 bit control and should be of no consequence to the future experiments. The only aspect of this portion of the project remaining is to put a step function into the Quick Basic code that will systematically step the voltage by the square root of two. This can be done during the final phase of combining all of the Quick Basic programs.

The Omega temperature controller responds to its' inputs flawlessly. When given a starting and ending temperature, the program will cause the CN6070A to set to the beginning temperature. It will then check the actual temperature every 2 seconds until it is within 0.2 degrees of the set point. At this time it will pause waiting for the counting process to complete, then step to the next temperature.

CONCLUSIONS

Although the process of collecting experimental data is usually a very time consuming and labor intensive process, through this venture it is shown that automation can be an effective alternative. The use of the PC-DIO-96 I/O board, Quick Basic, RS-485, and other programming techniques, successfully controlled the Electrostatic Aerosol Classifier as well as the temperature controller for the Continuous Flow Cloud Diffusion Chamber. Future studies are required, however. A method to collect and retrieve the data produced by the EAC and the CFD is needed in order to render the system completely automated. This can be accomplished with just a few hours of simple programming and should be of no hinderance to the overall goal of producing a completely automated system.

ACKNOWLEDGMENTS

I would like to thank Dr. D.J. Alofs of the Mechanical Engineering Department at the University of Missouri Rolla for giving me the opportunity to work on this project.

I would also like to thank Mr. Ray Hopkins and Mr. Max Trueblood of the University of Missouri Rolla Cloud Sciences Department for taking time out of their busy schedules to answer my many questions.

BIBLIOGRAPHY

Alofs, Darryl J., "Performance of a Dual-Range Cloud Nucleus Counter," Journal of Applied Meteorology 17 (Sept. 1978): 1286-1296.

Microsoft Corporation. Microsoft Quick Basic, Programming in Basic Version 4.5 for IBM Personal Computers and Compatibles. Microsoft Corporation, 1988.

National Instruments Corporation. NI-DAQ Software Reference Manual for DOS/Windows/Lab Windows Version 4.3. National Instruments Corporation, March 1992.

National Instruments Corporation. NI-DAQ Function Reference Manual for DOS/Windows/Lab Windows Version 4.3. National Instruments Corporation, August 1992.

National Instruments Corporation. PC-DIO-96 User Manual. National Instruments Corporation, June 1992.