COMPUTER TECHNOLOGY

A 64-station computer-assisted teaching and research facility

KARL W. SCHÖLZ and BEATRICE HOLLY

University of Illinois, Champaign, Illinois 61820

A 64-subject computer-controlled laboratory for use in research and as a teaching aid is described. The laboratory hardware includes an 8K minicomputer, a 64-channel input multiplexor, and a video display generator. The software includes a complete operating system and an easy-to-use programming language. The entire laboratory, including the computer, was built for under \$12,000.

Research in many areas of experimental psychology, especially in the area of human verbal learning and cognitive psychology, often requires the use of large numbers of subjects in an effort to evaluate some particularly elusive phenomenon. In some cases, a typical classroom setting provides the appropriate environment for data acquisition from many subjects simultaneously. However, an increasing number of paradigms require the use of an automated laboratory facility such as a computer-controlled television display and a keyboard entry device in order to provide accurate reaction time measurement or response-contingent stimulus display. Most laboratory automation systems that permit this type of experimentation are limited to a small number of simultaneously run subjects. The present paper describes a laboratory that was designed to retain the obvious advantages of computer-controlled data acquisition but permit research on classroom-sized groups. This laboratory, now operational at the University of Illinois, permits research on up to 64 simultaneous subjects, and was built for a cost of under \$12,000, including the cost of the computer.

This paper is divided into two sections. The first discusses the choice of hardware, including the computer system on which the laboratory design was based. The second provides an overview of the stand-alone operating system with which an experimenter may prepare and execute experiments.

THE LABORATORY HARDWARE DESIGN

The primary consideration in the design of the laboratory was to create as effective a research tool as possible with the least expense. Since many subjects were to be run simultaneously, an inexpensive

response device was essential, so we rejected the idea of having individual alphanumeric keyboards for each subject in favor of discrete push-button response boxes. Sixty-four, seven-button response boxes were made at a cost of about \$25 apiece. Each box also included diodes to encode the seven switch closures into 3 bits, permitting the use of four-conductor cable (3 data bits and one common line) to interconnect the boxes.

After surveying the minicomputer market, the Texas Instrument (TI) 960-A process control computer was chosen as providing the architecture most appropriate to the project for the least price. The computer configuration included 8K of 16-bit memory, a real-time clock, and appropriate serial and parallel digital interfaces.

The majority of the hardware design effort was directed toward the creation of a device which would interface the 64 3-bit responses to the computer. The device, a 64-channel polling input multiplexor, is described in detail next.

The multiplexor can be logically divided into two units, an address control and an input port select unit. The address control unit, constructed on a single card (referred to hereafter as the ADC), processes all data lines to and from the computer. The ADC receives nine lines from the computer, six of which specify a port address and three of which specify or control operation modes. The ADC uses 10 lines to transmit information to the computer. Six of these lines specify a port address, three specify the bit configuration (e.g., subject response) for that port, and one signals an interrupt request.

The circuit of the ADC is illustrated in Figure 1. All lines from the computer enter the ADC logic on the left side of the figure, and all lines from the ADC to the computer exit from the right side. The operation mode of the ADC (and hence of the multiplexor as a whole) is determined by the lines labeled AUTOPOLL

K. W. Scholz is now at the IBM Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.

Figure 1. Circuit diagram of the address control card.

and SEQINT. If both of these lines are low (logical zero), the ADC will not interrupt the computer. Rather, the bit configuration for the subject whose port address is presented to address lines IN0 through IN5 will be presented to the computer on lines BIT0, BIT1, and BIT2. If both AUTOPOLL and SEQINT are high (logical one), the ADC will sequentially cycle through all 64 port addresses, interrupting the computer for each address. When each interrupt occurs, the computer can input the response for that port, and identify the address of that port from the six input address lines. The sampling frequency is determined by the frequency of the clock (an SN74123). The COUNT INHIBIT line is used to inhibit counting while an interrupt is being serviced by the computer.

In normal use, a third mode of operation is initiated by setting the AUTOPOLL line high and the SEQINT line low. In this mode, the 64 ports are also sampled sequentially, but an interrupt to the computer will be generated only if one or more of the 3 bits at a given port address are nonzero. Again, sampling rate is determined by the clock frequency, and the COUNT INHIBIT line is used to suspend polling during interrupt servicing.

The gating in the top portion of Figure 1 is used to determine the source of the port address which is delivered to the port select unit. If AUTOPOLL is low, the source of the port address is the six lines IN0 to IN5 from the computer, and if AUTOPOLL is high, the port address is taken from the 6-bit address counter in the center of the figure. In either case, the

four least significant bits of the address are passed directly to the port select unit via lines A0 through A3. The four NAND gates and two inverters in the upper portion of the figure are used to decode the two most significant bits into the four control lines, SEL0* through SEL3*, which are also passed to the port select unit. The address counter in the center of the figure is a simple 6-bit binary ripple counter constructed from J-K flip-flops. The remaining logic on the ADC is used to process the interrupt signal for the various operating modes.

The port select unit is constructed on six cards. Each card is a 32-port, 1-bit-wide multiplexor (see Figure 2). The six cards are interconnected, as shown in Figure 3, to form the complete 64-port, 3-bit-wide multiplexor. For any 6-bit port address, the 4 least significant bits are presented simultaneously to all six IMUX boards. The appropriate 16-bit group is selected by the four lines SEL0* through SEL3*, which were decoded from the 2 most significant bits of the port address. Finally, the six lines IMUX1 through IMUX6 are gated on the ADC to produce the 3-bit response for a given port address.

The entire multiplexor is constructed from readily available TTL logic and requires only a single 5-V power supply. The total cost for parts is under \$500, including cabinet and power supply.

Stimuli were presented to the subjects using four 21-in. video monitors driven by video output from an ADDS (Applied Digital Data Systems) Consul 580 CRT terminal. A screen format of 24 lines by 80 characters per line was written using a serial interface operating at 960 characters/sec. The four monitors were placed so that they could be easily viewed by all 64 subjects. Naturally, all subjects viewed the same stimulus information at the same time.

Figure 2. Circuit diagram of a single multiplexor card.

THE LABORATORY SOFTWARE SYSTEM

The major consideration in the design of the software was that it present as simple an interface to the user as possible. Thus, the commands with which a user directed the operation of the system were carefully selected to permit full utilization of the laboratory's capabilities without the necessity for a major learning effort on the part of the user. In the following section, we will first describe the general architecture of the system, and then consider the details of the source language and the nature of its implementation.

The choice of system architecture was limited by several considerations. First of all, budgetary considerations precluded the purchase of any form of secondary storage for the TI 960, so the system had to be entirely storage resident (the alternative of providing paper-tape overlays was considered far too cumbersome). Secondly, the entire project was designed to use by intelligent nonprogrammers (the faculty and graduate students of the psychology department), and as such had to be as simple as possible to utilize. Finally, the nature of the input/output devices (seven-button response boxes and television monitors) defined a research domain for which many aspects of complex programming languages would be inappropriate. These considerations were borne in mind throughout the design and implementation of the system.

Overview of the System

In general, the system consisted of three major components, the device handlers, the translator, and the interpreter. The device handlers included routines to field interrupts and to supervise data transmission to and from all the input/output devices, including a real-time clock, a Teletype, the video character generator (ADDS 580), and the input multiplexor described above. The techniques for interrupt management and queue handling have been described elsewhere (Scholz, 1972).

A second component of the system, the translator, provided the major interface between the system and the user. This collection of routines served both as a limited text editor and as a source language translator. Since this journal is already replete with detailed descriptions of new programming languages (a situation to which the first author confesses contributing), the source language itself will not be described in detail. The structure of the language was the familiar tag-operand form, which is, in our experience, easily mastered by newcomers.

In an effort to ease the programming burden for the user, a novel approach to array handling was incorporated. From a user's point of view, an array was simply a string of characters of arbitrary length. A string cursor which was invisible to the user was associated with each string. This cursor could be

Figure 3. Block diagram of the 64-port multiplexor.

either utilized or manipulated, depending on the context in which the string was referenced. Whenever a string was referenced by a string processing tag (DISPLAY, ACCEPT, COMPARE), the cursor would simply be utilized to delimit substrings. For example, the statement SHOW MESSAGE 6 would display six characters from the string MESSAGE, beginning at the current cursor position, and would then increment the cursor by six. On the other hand, whenever a string was referenced by a numeric variable processing tag (ADD, SET, SUBTRACT), only the string's cursor would be manipulated. For example, the statement SET MESSAGE 0 would position the cursor associated with the string MESSAGE to the left of the first character.

Each statement entered into the translator would be compiled into an intermediate code which occupied two consecutive words of memory. All statements were limited syntactically to a maximum of two operands which could be constants, variable names, or string names. These two operands were represented, respectively, in the left 12 bits of the two words of intermediate code. Constants were stored directly into the appropriate 12-bit field as literals. Strings and numeric variables were represented as pointers to the relative memory address of the associated entity. The remaining bits of each word of intermediate code were used to identify the tag (e.g., statement type) and to indicate whether each argument field contained a literal or a pointer.

The final component of the system, the interpreter, constituted the monitor which was in control of the system whenever an actual experiment was being conducted. Upon entry, the interpreter would initialize a software "program counter" to the first word pair of intermediate code produced by the translator. The fields of the intermediate code would then be decoded and control would pass to a routine which would execute the operation specified by the tag field. After the execution of each command, the

program counter would be either incremented to the next word pair or would be appropriately reset if the command required transfer of control. The entire interpretation process ran as a background task, while the interrupt-driven device handlers occupied the foreground.

One problem which frequently confronts the designer of a process control system is the storage and recovery of data generated during an experiment. In the present system, data management was greatly simplified by providing a single command in the source language and a single command to the monitor for data management. In the source language for an experiment, the user codes a SAVE statement each time he wishes to save any type of data, be it a string, a numeric variable, a vector of up to 64 subject responses, or a vector of up to 64 reaction times. The data referenced by a save statement is accumulated in the computer's memory, using all available storage if necessary (typically approximately 6,000 words). Each data entity is prefixed with indicators which represent its type and its size. At the completion of an experimental session, a single monitor command can be used to produce a formatted labeled dump of the accumulated data onto either a Teletype printer or, using phone lines, directly to a large timesharing computer for subsequent analysis. Thus the user is completely freed of the burden of formatting or indexing his data. The data recovery system also permitted the spooling of data to either a Teletype printer or to the phone lines while an experiment was in operation. Thus the total volume of data retained from an experiment was not limited by the size of available memory buffers.

Advantages of the System Design

One major advantage of the translation/interpretation architecture used in the present system is the ease with which it may be expanded. As explained above, the compiler produces an intermediate code which is interpreted at the time a program is executed. The interpretation process only reads the intermediate code—it never modifies it. Thus the system can be easily expanded to permit executing the same program independently for multiple subjects. If, for example, the system is expanded to allow individual displays for each subject (such as an array of stimulus lights or even an alphanumeric display), it is possible to allow each subject to work through an experimental session independently. Such a system can be implemented by providing a small variable block for each subject. This block would contain sufficient space for numeric and string variables referenced by the program, and would contain a single "program counter" which indexes that subject's position in the program. The interpreter would then simply poll from block to block, executing the appropriate next instruction for each subject in turn. It should be

emphasized that only a single copy of the program is maintained in memory; its multiple independent execution is implemented by creating the multiple variable blocks, one for each subject (see Kieras, 1973, for discussion of a similar system).

SUMMARY

The system described in the present paper was implemented in approximately 6 months. The most time-consuming chore was the preparation of the site, including building the 64 response boxes and installing the cabling from each box to the interface. The system was installed in a classroom that had permanently mounted desks, and the response boxes were designed to be easily removable in order to prevent theft (thus they could be removed during normal classroom use).

The system can be used as an educational tool and for research. As a teaching aid, the system can be used to provide on-line administration of multiple choice examinations. Exam questions can be distributed in a conventional exam booklet. The number of each question can be displayed on the television screens, and the student required to indicate his choice by a button press. The student can then be furnished with his score as soon as he completes the exam. The system can also be used to provide a lecturer with feedback during a lecture. Students might indicate confusion or need for clarification by button press, yet remain anonymous. In conventional classroom settings, the lack of anonymity often inhibits student participation.

As a research tool, the system permits the rapid completion of experiments requiring many subjects using a variety of paradigms. Since the system is essentially a multiple subject "memory drum," many paradigms in traditional human verbal learning can be easily implemented. Any experiments in perception or psychophysics which require judgmental responses can be easily conducted. The system also readily supports research in sentence or paragraph memory due to the high-speed alphanumeric display capability.

REFERENCES

- KIERAS, D. A general experimental programming system for the IBM 1800. *Behavior Research Methods & Instrumentation*, 1973, 5, 235-239.
- SCHOLZ, K. W. Computerized process control in behavioral science research. *Behavior Research Methods & Instrumentation*, 1972, 4, 203-208.