SOME GENERAL COMMENTS

As was noted earlier, S-POOL was written (and tested) in sections. Because of this, the search operation was not written as a subroutine. Consequently, there are a number of sections of the program concerned with locating the appropriate student. When the various sections of S-POOL were integrated, it was felt that the addition of such a subroutine would not improve (or reduce the size of) the program enough to warrant the additional programming and debugging time. (The program, including comments, contains 480 cards.)

Ideally, the program should have been written in a language better suited to the accounting function served by S-POOL. A general-purpose language such as PL/1 or a business-oriented language such as COBOL would have minimized the energy expenditure in the task set for the programmer. Fortran IV was chosen at SUNY/B *only* because it is the language with which most of the graduate students in our department were conversant. It had been our hope that any of these students could be placed in charge of S-POOL (i.e., programming and debugging further modifications and running the program). It is, of course, quite a hardship to modify a large, complex program written by another programmer. Furthermore, S-POOL was quickly modified to its present (finalized) form. The author would thus urge that any further efforts involve one of the other languages that is more compatible with the purposes of the program.

S-POOL allocates three storage locations to each student for experiment reference numbers. A "packing" feature allows three experiment reference numbers to be stored and retrieved, in a relatively simple manner, at each location. This feature together with the "overflow" feature are important sections for rendering the program compatible with other systems. They would, for example, prove of value to schools operating under a system where points are added to the final grade for each experiment in which S has participated.

REFERENCE

STARR, B. J. S-POOL: A program for keeping track of participation in psychological experiments. Behavioral Science, 1969, 14, 252 (CPA 328).

NOTES

1. The development of this program was partially supported by National Institute of Dental Research Training Grant 1T1-DE-170. In addition, help and computer time were accorded by the State University of New York at Buffalo Computing Center, which is partially supported by National Institute of Health Grant FR-00126 and by National Science Foundation Grant GP-7318.

2. The author would like to thank Dr. Roy Lachman for helpful suggestions made during the development of the program and Dr. Erwin M. Segal for his critical reading of the manuscript.

Variants of basic black computer interfaces[1]

ROBERT S. McLEAN,[2]
CARNEGIE-MELLON UNIVERSITY, Pittsburgh, Pennsylvania 15213

An input interface for connecting psychological apparatus to a process-control computer is described. It permits connection of response switches to eliminate contact bounce and to provide interrupts on ongoing transitions of any data bit.

Uttal (1968) has described a number of basic interfaces between control computers and psychological experimentation. This note mentions a variation of the basic input interface that has been implemented at the Computer Controlled Psychology Laboratory (CCPL) at Carnegie-Mellon University; the intent is to make available an alternative set of techniques to be considered by the researcher about to install a control computer. The logic symbols employed by Uttal are used here, and the reader is referred to the original paper for discussion.

RESPONSE SWITCHES

Uttal describes a design for an input interface that allows Ss' operations of buttons, switches, and similar manipulanda to be sensed by the computer. One of the basic problems inherent in switch closures is that they often produce "contact bounce"—the momentary generation of rapid on and off indications at the instant that they are being turned on or off. This problem was recognized, and a switch filter was utilized in Uttal's example. The CCPL solution to this was to require that each switch provide both a set and a reset output, and that the input flip-flop be switched accordingly.

Each switch used by the S (or E) is a double-throw type (Fig. 1). The common contact is connected to the logic level that activates direct inputs to flip-flops (reset-set flip-flop). In this way, the flip-flop "follows" the state of the switch. Since the electrical noise produced by most switches consists of rapid making and breaking of the connection due to mechanical contact of the moving parts and not to alteration of the two states of the switch, this provides for a stable switching arrangement. When the switch is thrown "on," the flip-flop assumes the "on" state at the first indication of "on" from the switch, the switch having previously broken the "off" connection early in its mechanical travel from "off" to "on." Successive pulses on the "on" side have no further effect since no "off" pulse has reset the flip-flop in the meantime. Turning the switch "off" results in a similar process: The flip-flop remains "on" until the first indication of the "off" setting from the switch. Further "off" pulses have no further effect since no "on" pulses have occurred in the meantime.

This also has the advantage that it overcomes some of the problems associated with long wires from experimental laboratory to computer. Induced noise and crosstalk among the conductors of the cable carrying signals to the laboratory and computer may cause spurious data. In this input method, one data line is always grounded except during the interval of switching. If crosstalk continues to be a problem, it may be necessary to provide additional pull-up voltage to the input lines. This is usually accomplished by adding a resistor of appropriate value between the flip-flop inputs and the "1" logic voltage, as indicated in Fig. 2.

INPUT INTERRUPTS

The problem of determining the occurrence and timing of events in the experiment via the computer input circuitry is sometimes difficult. A basic decision must be made concerning the conditions under which the computer is to be able to determine that something of interest is happening in the experiment.

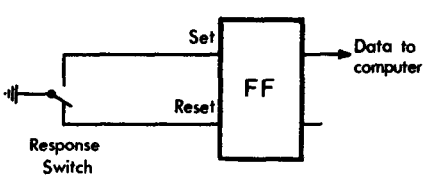

Fig. 1. An input buffer bit and a S's response switch.

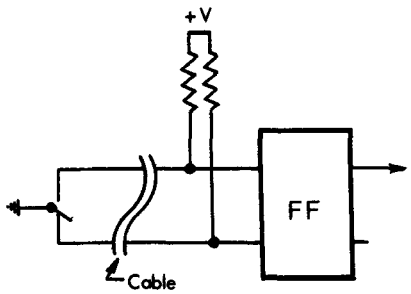

Fig. 2. Input buffer bit with pull-up resistors enabling a long cable to the experiment.

Usually the information sources for the computer include the input bus and an interrupt line. The former is usually a set of wires, one per bit of the accumulator, to which are connected all of the peripheral devices capable of generating input data to the computer. This information is "gated" onto the input bus by appropriate "addressing" of the device having the desired information. The interrupt line (or sometimes lines) indicates that something that might be of interest to the computer has happened in the periphery; this is typically a one-bit indication and conveys no information about what it was that happened.

The problem is that the laboratory designer must decide what conditions warrant requesting the attention of the computer program. In the CCPL, it was decided that any individual bit from the experimental room that changed from a "0" to a "1" state was worthy of notice. A total of 16 bits of input information was available from each experimental room, so a method of detecting 0-1 transitions on any of these bits was required. The basic circuit is indicated in Fig. 3. A negative-going transition on any input (caused by the input buffer flip-flop changing state from 0 to 1) causes a pulse to be generated at the output of the first gate; this pulse can be inverted and used to set an interrupt flip-flop.

The resulting configuration provides information to the computer whenever any new "on" conditions appear in the experiment, regardless of the state of the remaining inputs. If it is desired to detect both transitions (0 to 1 and 1 to 0), this can be done by tying the switch to two bit inputs, reversing the connections to one of the two; thus, throwing the switch "off" turns on one bit and turns off the other. If both transitions are always desired, then the transition-detection circuit can be doubled, connecting the inputs of the second detector to the set sides of the appropriate bits. Alternative circuits for the detector might also be designed to detect the two transitions from a single input.

The CCPL input scheme is similar to that used by Haber (1968) but combines the data and interrupt functions in a single switch pole for each bit, eliminating the possibility of generating an interrupt prior to providing data. No reset from the computer is necessary, and the state of the input buffer is exactly that of the S's device.

Some caution is advised in programming/using this hardware interrupt technique, in that interrupts must be serviced soon after they occur, since the data bits "follow" the state of S's switches. If interrupts are disabled for long periods of time (e.g., 10 msec), an interrupt may appear to have been generated by data that have disappeared. The CCPL software monitor system assures that interrupts are never disabled for longer than ½ msec, obviating this problem. It is also advisable to reset the interrupt just prior to inputting the data from the interface to remove the possiblity of data occurring after the interface has been read but before the interrupt has been reset. The hardware could be arranged to reset the interrupt automatically when data is read, if desired.

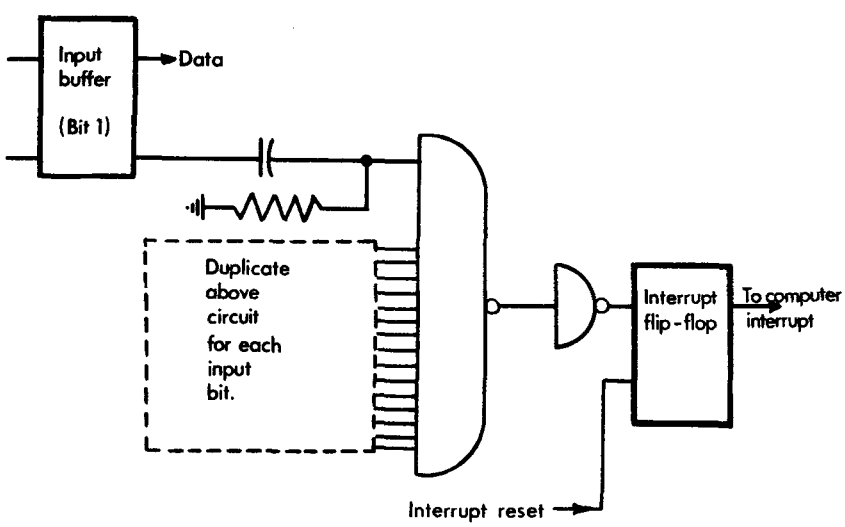

Fig. 3. A circuit for causing an interrupt wherever any input becomes a "1."

REFERENCES

HABER, R. N. An on-line computer in a visual perception laboratory. Behavior Research Methods & Instrumentation, 1968, 1, 86-93.

UTTAL, W. R. "Basic black" in computer interfaces for psychological research. Behavior Research Methods & Instrumentation, 1968, 1, 35-40.

NOTES

1. The work reported here was supported by a grant from the Public Health Service, National Institute of Mental Health, Research Grant MH07722.

2. Now at The Ontario Institute for Studies in Education, Department of Computer Applications.