The counter is 12 bits (to match the computer word size) plus overflow, and is driven by a 1 KHz ticker. The clock can function in either an interrogation mode (the computer reads the time-word), or in an interruption mode (the clock triggers an interrupt request when a preset time has elapsed). To facilitate the latter mode, any initial value can be written into the counter. The interrupt is generated on overflow, contingent on the status of "interrupt request enable," which is under program control. After an overflow condition is established, the counter can continue to count. The status of the overflow bit can be checked by the program (so that we can effectively have a 13-bit counter if desired) and separately cleared by the program.

The clock may be stopped and started dependent upon the status of "ticks enable." Thus, we can accumulate lapsed time for a process that starts and stops. This same logic could be used to steer one or another time-mark generator to the counter, thus providing programmed selection of time bases. Moreover, the logic is designed to interrupt, momentarily and automatically, the counter from counting while the computer is effecting a transfer to or from the counter. Things are so arranged that reading on the fly can be accomplished without missing a count.

Much of our previous discussion pointed up the merits of a computer with a multi-channel priority-interrupt capability. This clock is used on a PDP-8 which has but a single interrupt line. The technique used to bypass the single channel constraint is worth noting.

Because our S's response buttons are also tied into the same interrupt line, we have no way of knowing whether an interrupt was caused by the clock or the subject(s). We could set up status flops which could be checked by the program and would serve to answer the question. Instead, a "block data transfer" is executed on every interrupt—we read in the clock counter and all Ss' response devices. Sorting out who or what caused the interrupt is done afterwards, thus preserving timing accuracy. This technique can be extended to other external data sources; the only trick is to pack the information into as few computer words as possible so as to make the block as small as possible.

## CONCLUDING REMARKS

It seems safe to say that the clock with both ticker and counter external to the computer is to be preferred to most of the alternative possibilities considered. A possible exception is the case in which the ticker is external but has a direct portal to the memory location that is serving as the counter. The latter arrangement has the advantage that it is not necessary to effect input-output transfers when setting, reading, or manipulating counter values. Otherwise these two schemes are quite comparable.

The relative acceptability of any particular clock implementation will depend somewhat, of course, on the specifics of the system in which it is to be used. Consider again, for example, the case of an external ticker and an internal programmed counter. This is the scheme in which each tick of the clock causes the ongoing program to be interrupted long enough to execute a service program which simply increments the clock count. This method works very well provided the system has a multi-channel interrupt system so that one of the channels can be devoted exclusively to the ticker. If, however, the interrupt system has only a single channel, this clock implementation is less attractive.

The clock shown in Fig. 5 could be made somewhat more versatile with the addition of one or more addressable alarm registers, which would function in the following way. The contents of each alarm register would be compared with the clock count following each tick. Whenever a match occurred between any one of the alarm registers and the value of the clock counter, an interrupt would be requested. The interrupt request from each alarm register should be gated so that the user could enable or disable each alarm at will. Whether the added versatility and programming ease is worth the additional cost of such features depends on the uses to which a particular system is to be put and on the tradeoffs that the system's users are prepared to make. We have found the clock shown in Fig. 5 to be both adequate and convenient for the computer-controlled experiments undertaken at this facility to date.

## REFERENCES


BORKO, H. (Ed.), *Computer applications in the behavioral sciences.* Englewood Cliff's: Prentice-Hall, 1962.

COULSON, J. E. (Ed.), *Programmed learning and computer-based instruction.* New York: Wiley, 1962.

FLIEGE, S. Digital computers. In J. B. Sidowski (Ed.), *Experimental methods and instrumentation in psychology.* New York: McGraw-Hill, 1966.

GREEN, B. F. Jr. *Digital computers in research.* New York: McGraw-Hill, 1963.


## NOTE


1. This work was supported by the Human Performance Branch of the NASA-Ames Research Center under Contract No. NAS2-2676.


# An on-line computer in a visual perception laboratory[1]


RALPH NORMAN HABER, *DEPARTMENT OF PSYCHOLOGY, UNIVERSITY OF ROCHESTER, Rochester, N.Y. 14627*



*The use of a small data acquisition computer (PDP-8) is described for on-line operations in visual perception experiments. A typical program to control an experiment on short-term visual memory is presented, and the interface to connect the computer to the external stimulating and response devices is described in some detail. A brief description is given of some special purpose interface components and how the computer could be used for experimental situations other than those in visual perception.*


The last 10 to 15 years have shown a rapid growth in the application of electronic components in experimental psychology laboratories. The timers, counters, controlling systems, and measuring devices now in use, although greatly improved in this past decade, are mainly extensions and outgrowths of those designed earlier. However, dramatic changes have occurred since the early 1960's, with the advent of inexpensive digital computers that can be used on-line in an experimental laboratory. This has been a departure from the more customary use of the computers as bulk data processors and analyzers, since this new function is to run experiments which acquire data, and in the process control

external apparatus and provide feedback for the Ss.

This paper describes the use of a small digital computer (a PDP-8) for these purposes, with a detailed example from visual perception research. Within this context I have included some discussion of the design and construction of the interface, and some basic problems of connecting and programming data acquisition computers for psychological experiments. No attempt will be made to describe the details of the research other than their procedures and apparatus.

One area of research on which the PDP-8 has been used has concerned short-term visual storage processes (STVS). These experiments are designed to determine whether any internal representation of stimulation persists after the offset of the physical stimulus. In these studies S is generally presented with a brief visual display, for example, a matrix of letters. Some short time after its offset a second stimulus occurs which indicates which letter is to be reported. The general finding is that for brief intervals (less than 1 sec, usually) an S can give very accurate partial reports. An S is not able to recall all the letters, since all but the first one or two reported "fade away" before he can attend to them. From this it has been concluded that for a second or so the S must have an interval post-stimulus representation of the display which contains more content of the stimulus than is available for normal recall.

Partial report procedures require an adaptation field, a stimulus display field, a field to be on during the interstimulus interval (usually a return of the adaptation field), and another field for the partial report indicator. Usually a three-channel tachistoscope is used for such experiments, although other types of apparatus can be utilized for this design.

A second area of research has concerned the question of how much time is required to process, identify, and make decisions about incoming stimuli. In addition to the use of simultaneous displays, such as with a three-channel tachistoscope, processing time can also be studied with sequential displays. Rather than to present them all at the same time, symbols can be presented one at a time, either in the same location or spread out across the display surface. The following may be varied: area of presentation, type and redundancy of items, location of display, and the relation of the duration of each item to the interval between one item and the next. Several types of apparatus have been used for these designs. One is an alpha-numeric display made from an electroluminescent panel. By programming the individual segments that make up each letter, any sequence can be created with complete control over rate and timing maintained by the computer. Another device currently used is a CRT display, directly interfaced to the computer. Here any symbol or symbols can be displayed either in sequence or simultaneously, in the same location, or across the face of the CRT.

A third area of work is concerned with word recognition processes, especially the effects of repetition of presentation, and the mechanisms which determine the clarity of perception. In most of the experiments many hundreds of words are presented, with careful control needed over the duration and the luminance of the letters as well as on the number of presentations. A two- or a three-channel tachistoscope is generally used with automatic stimulus-changing mechanisms used to change the words from trial to trial. The computer changes the stimuli, manipulates the duration and luminance of presentation, records responses, and analyzes the data.

All of these examples involve vision research. Without going into any detail it should be obvious that there is no reason why a visual perception laboratory is any better for the use of a data acquisition on-line computer than any other instrumentation-oriented laboratory in psychology. For example, a series of Skinner boxes could be interfaced to a computer in which a number of animals could be run simultaneously—each on different schedules, with the computer keeping track of the schedules, providing the reinforcements, the presentation of stimuli, counting responses and latencies, and maintaining the responses of the organism. It is also possible to connect a computer to a set of runways, even ones in which each alley segment is being counted separately. The computer then could operate doorways, record choices, and it could record the outputs of photocells. A computer could be used also to run a multi-channel communications system in a social psychological experiment; and it could be used to handle probability experiments. Processing of electrophysiological data is a natural use, and the computer could also conduct the experiment from which the data were obtained. The programs would each be different, but the interface would be very much the same.

## EXPERIMENTAL PROCEDURE OF AN STVS STUDY WITH AN ON-LINE COMPUTER

To illustrate how an on-line computer can be used to run an experiment, a detailed description is given of a short-term visual storage (STVS) experiment. While it has some specific details both in design and in apparatus that do not occur in other experiments, it is of sufficient generality to provide a useful example.

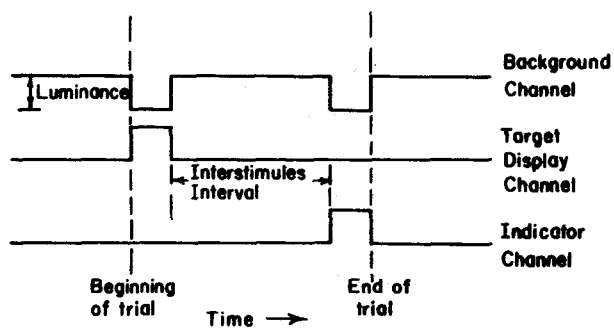


**Fig. 1. A typical trial sequence for a short-term visual storage experiment using three optical channels.**

The S is initially adapted to one field (background) of a three-channel tachistoscope set at a fixed luminance (see Fig. 1). When ready, S depresses a foot switch. The background channel is then replaced by a stimulus channel containing a target display which remains on for a predetermined duration. At its termination, the background channel returns, also for a predetermined duration, serving to produce a delay after the target display. An indicator channel then briefly replaces the background; it contains a pointer informing the S which part of the target to report. After presentation of the indicator, the background channel returns and remains on until the next trial commences. All three channels are set at the same luminance.

The background channel is blank (usually white), with some fixation boundaries drawn on it to help S locate the target. The target and indicator channels have displays which are changed

after each trial is completed. Each of these displays is drawn in advance of the experimental session on a long, fan-folded roll of 8-in. paper which is mounted and advanced by an automated memory drum. Each drum is attached to the rear of the optical path of a channel, so that when illuminated by the light source from that channel, the information on the paper is visible. The positioning of each display is controlled by holes punched in the paper, which are sensed by microswitches. This provides accuracy of positioning to within .005 in. The paper is advanced on signal by motors capable of moving the paper at the rate of 4 in./sec. After a trial ends, the two drums are pulsed to advance their respective displays. Alignment is guaranteed if the holes controlling the stopping positions are aligned. At the end of a trial, the S responds verbally by naming the marked item and the position it occupied in the target. In some cases he has been trained to push response buttons on a typewriter console connected to the computer. It is possible to completely computerize this experiment, so that no experimenter is needed.

In a simple version of this study, the amount of STVS as a function of the delay and position of the indicator is investigated. The target is a row of eight randomly chosen letters. Eight different delays (0 to 3200 msec) separate the offset of the target and the onset of the indicator, an arrow pointing to where one of the letters had been. Thus the data fall into 64 cells, eight delays by eight positions. In each session the S is shown 256 trials, four replications of the 64 cells. The order of the 256 trials is randomized, but in the same order for each S. Each S is tested in 10 sessions, each of which used a different order of the 256 trials. Each one of the 2,560 trials has a different eight letter target.

The computer, through its program, is connected by an interface (see Fig. 2) to several aspects of this experiment. The three lights, corresponding to the three channels, are turned on and off by the computer, which also determines their sequence and durations. Luminances are controlled by a 16-step resister tree operated by relays which are set by the computer, which also controls motors advancing the target and indicator displays. The S's start switch and response keys are also connected directly to the computer. The principal communication device is a typewriter on which the E can type the S's answers, and on which the computer provides a listing of all the raw data for each trial.

The program for this experiment has several parts (see Fig. 3),

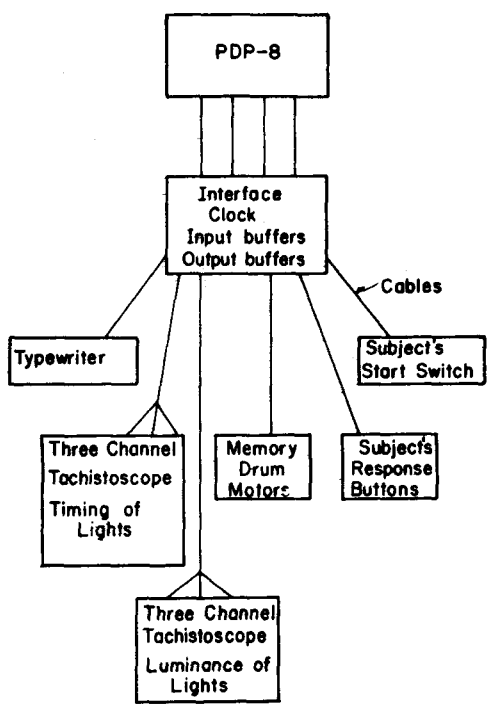


**Fig. 2. Configuration of computer and external equipment as used in short-term visual storage experiment. The computer is located about 6 ft from the interface and about 100 ft from each of the other components, with cables as interconnections.**

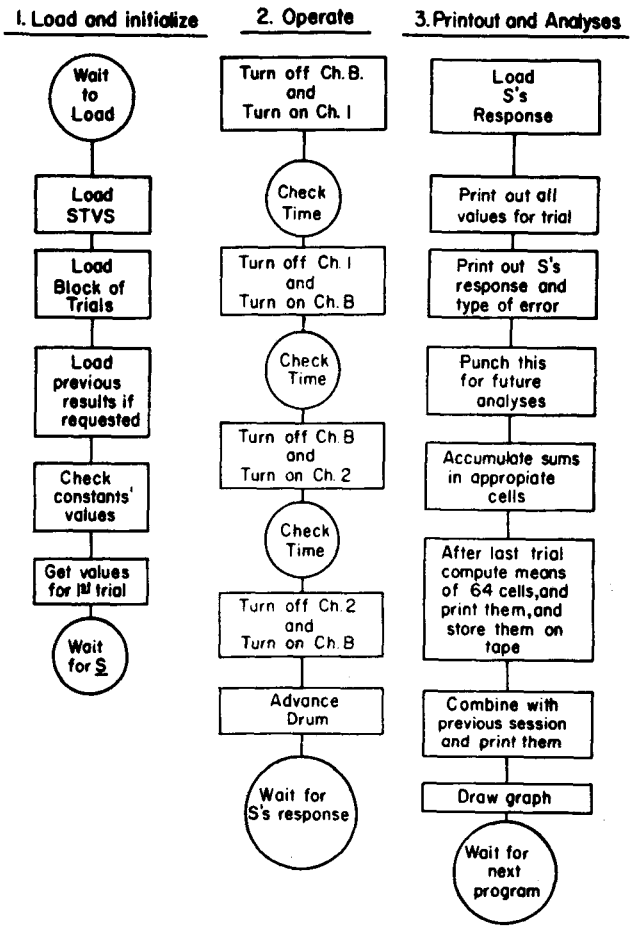


**Fig. 3. Schematic flow chart for the PDP-8 program of the short-term visual storage experiment.**

all of which are on magnetic tape, along with each of the 10 blocks of 256 trials. The first part controls the loading and initializing of values of the program. When the E starts the session with a particular S, he will, after turning on the computer, type the name of the program, the name of the S, the number of the block of 256 trials that will be used, and the trial number within that block which will begin the session. The proper block of 256 trials will then be loaded into the computer core memory from the magnetic tape. For each trial, this includes the trial number, the eight letters in the target, the position of the indicator, the correct answer, and the duration of the delay (the other durations are not varied and hence are not in the trial listing). After the 256 trials are loaded into memory, values from the selected trial

number are placed in the appropriate registers in the operating routine to the program. Then the computer will wait for the S's start switch. Less than 20 sec elapse from the time that the computer is turned on to the first trial–that time is taken up almost entirely with loading information from magnetic tape. The E will have to set the paper tapes for the display and indicator to the appropriate starting points, and have the S properly adapted.

If the E is planning to accumulate the results of the session just beginning with those of the previous sessions (which is appropriate since each session is a complete replication of the experiment), then he will call for the loading of the means and sums from the previous sessions at this point. These data have been stored on magnetic tape, and at the end of the current session, the accumulated results including this session will replace the prior accumulations.

The first part of the program contains all of the constants and variables in the study. These include the durations of the three lights, the intervals between them, and also their luminances. The same values are used from session to session unless changed at the beginning of each session at the time E loads the program. The only variable that changes from trial to trial is the delay between the target and the indicator. The eight different values that this variable can assume are also available for change at the beginning of a session.

If any of these constants are to be changed (e.g., warmup trials indicated S's threshold had drifted), E can do so at this point by typing the name of the constant and then its new value.

The second part of the program contains the operation routines to run the experimental apparatus and provide the timing. It turns on the lights at the proper luminance, sequence, and duration; it listens for the S's start switch, and then for his answer, and then it advances the drums moving the stimuli.

The third part of the program contains the raw data printout and analysis routines. A common drawback of on-line computer operations is that often E can never recover the raw data, either to check for errors or to perform unanticipated analyses. To overcome this hazard, the raw data for each trial are printed on the typewriter after each trial, so the E can, if he wishes, examine them as the experiment progresses during the session, and he can save the printout for further work. On each trial, a line of data is generated. This includes the block number, trial number, the eight letters in the array, the position indicated, the delay between the display and indicator, the duration of the target, the duration of the indicator, the luminance of the channels, the correct answer, the S's answer (both item and position), and if the S was in error, whether his answer was an item error, position error, or both. To facilitate subsequent analyses of this raw data, a punched paper tape is also generated trial-by-trial which may be fed back into the computer without any intervening steps being needed.

This last part of the program also performs data analyses. These are accumulated during the course of the session and completed at its termination. While many complex analyses were eventually done on these data, only simple descriptive analyses were programmed to be completed on-line. These included descriptive statistics for the accuracy of reporting the letter indicated, broken down by the 64 cells. The type of error was ignored, and the particular letter being indicated or reported was not differentiated at this point. These 64 means were printed out along with the sums making them up. In addition they were punched onto paper tape. If E wishes to have an accumulative record of the performance over the 2,560 trials, he could combine the output of the previous session and then get the accumulated score for each of the 64 cells, summing the previous sessions and the current one. All of the analyses are completed in less than 3 sec, though the printout of the results takes a minute or so. The typewriter was programmed to draw a graph of per cent correct on the ordinate as a function of the eight positions, with eight parameters representing the eight delays.

## INTERFACE DESIGN

Without discussing the details of the experiment further, this section will describe how the computer is connected to external apparatus. The general input-output functions of an on-line data acquisition computer will be described first, and then some specific examples of connections of the PDP-8 to a number of pieces of apparatus. Examples will be chosen with sufficient generality so that other applications of computers can be visualized.

There are four basic programmable interconnections between the computer and external equipment that are utilized in on-line experimental functions. (There are other interconnections that are available on the PDP-8, but are not used for interfacing to experimental equipment. These include high speed channels for very rapid transfers of large blocks of information such as for bulk storage devices as magnetic tapes, drums, discs and the like.) Of the four involved in programmed input-output operations, one is for input to the accumulator, another is output from the accumulator, a third is for input-output device selection, and a fourth is for input-output program control.

The first of these is a 12 conductor cable connecting external equipment to the 12 input bits of the accumulator. These 12 input lines to the accumulator will set the 12 accumulator bits to zeros or ones, depending upon the voltage levels on each of the 12 lines. In this way, with appropriate gating (see below) it is possible to load up to 12 bits of information simultaneously into the accumulator from any one or more external devices.

The second interconnection is an output cable from the accumulator. Each of the 12 conductors in this cable will carry an appropriate voltage level depending upon whether the accumulator bits contain zeros or ones. The accumulator output lines will always follow the content of the accumulator. Therefore in order to select only those times when this output is needed, these lines must be connected to external equipment through gates. In this way, the contents of the accumulator at any given time can be used to determine the state of external equipment.

The third interconnection is for the input-output instruction number, and is used to indicate the particular device number being called for by the program operating at that time. This cable terminates in a device selector which is located, for simplicity, in the interface. By appropriate gating, the device selector decodes the device number and distributes it, via separate connections to the appropriate device being signaled. A program in the PDP-8 can directly address any one of up to 192 separate external devices. While this number would rarely need to be exceeded in practice, it easily can by "multiplexing" any one of these 192 channels.

The fourth interconnection contains control lines. The most important of these are input controls to the skip bus and the interrupt bus. Whenever the appropriate voltage is applied to the interrupt bus from an external switch, or some interface process,

the computer program will be stopped wherever it is, the address of the currect instruction is stored and the program will automatically jump to the interrupt routines. Thus, any piece of external equipment can interrupt ongoing activities of the computer by changing effectively a single switch. The skip bus operates in the same way except that voltage applied to the skip bus will have no effect unless a skip instruction is currently being executed by the program.

This completes a very cursory description of the interconnection links between the computer and external equipment. In a few cases, external apparatus can be connected directly to the cables just described. More typically, however, interface components have to be built between the computer and the external equipment. There are two principal reasons for this: the amount of current available to operate components and equipment connected directly to the output-input cables is very small, so that drivers or other amplifiers are needed to increase the power; and more importantly, the duration of pulses on these cables is on the order of 100 nanosec (100 billionths of a sec). Since external equipment generally will not respond to pulses that short, and since the duration of the desired operations of the external equipment is many times longer than the duration of the pulses turning them on, some buffer must be created in the interface to maintain the state of the apparatus beyond the duration of the brief pulse initiating it.

The interface needed for any experimental situation is very simple and relatively inexpensive. One or more of the 12 accumulator output lines is connected to all pieces of equipment that the computer will operate. As a new experiment is designed involving new apparatus, that apparatus is also connected to as many accumulator output lines as are needed. These connections are not direct, but are made through an AND gate for each line involved. An AND gate is a logical device which will provide an output only if all inputs to it are in the same predetermined state. Thus, the accumulator can operate a particular device (see Fig. 4 below) only if the instruction in the program calling for that device number is also executed at the same time. In this way, that device, while connected to the accumulator, will be unaffected by the accumulator unless its own device code is simultanously executed by the program.

The same principle is used for inputs to the accumulator. All devices for sending information into the computer (e.g., a typewriter, buttons, keyboards, feedback signals, sensors and the like) are connected through AND gates to one or more lines in the accumulator input cable. When the computer program wishes to sample information in any of these devices, it issues the appropriate device code which provides the additional pulse necessary to produce an output from the AND gates. When this is done, the accumulator can then be simultaneously loaded with up to 12 bits of information from an external device (see Fig. 5 for an example).

Such interrogation of external equipment is initiated by the program. If the external device wants to interrupt the program, then it also has to be connected to the program interrupt bus. After the interrupt is sensed, the program will then determine which device interrupted it, and react to the interrupt in ways determined by the program. For example, it may load any information from the device, or take some other action, such as to turn off a light.

## OUTPUT FUNCTIONS

In addition to AND gates, several other components are also in the circuit to control output functions. To use a simple example, Fig. 4 shows how a set of four lights—these could be any components—would be interfaced with the accumulator. By changing the bits in the accumulator (via the program), the pattern of the four lights is changed.

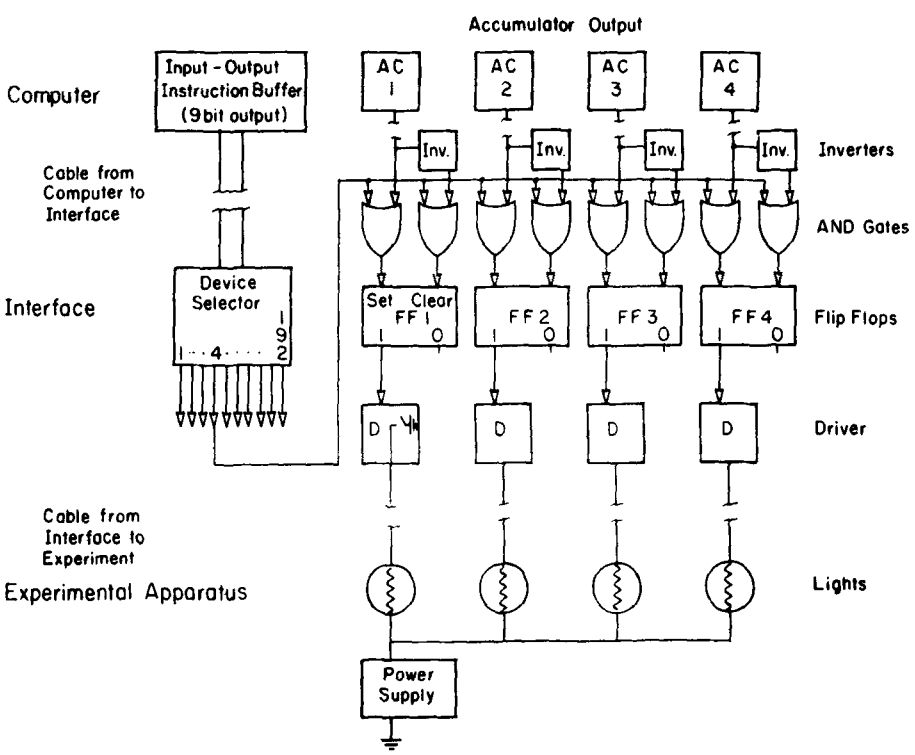


**Fig. 4. A 4 bit output buffer from the accumulator to a number of external devices.**

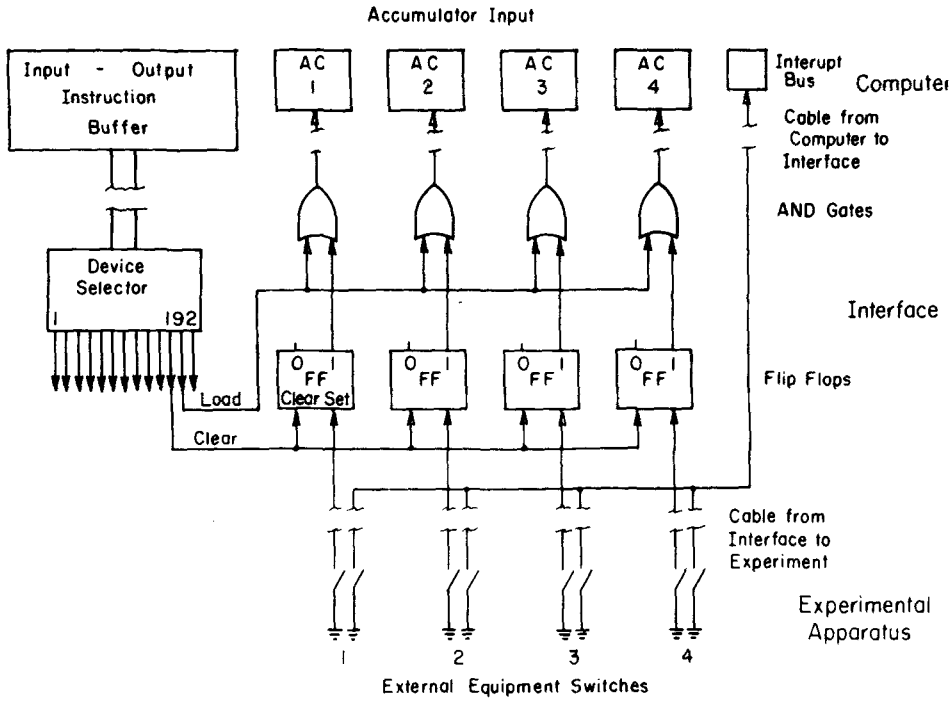


**Fig. 5. A 4 bit input buffer to the accumulator from a number of external devices.**

Each bit from the accumulator is connected in two ways. One of these is directly through an AND gate. The output of each of the four AND gates goes to a flip-flop (which can be thought of as a solid state latching relay). Whenever an output is received from the AND gate to the "set" input of the flip-flop, it will be turned to the "on" state. This will send an output from the on side of the flip-flop to a driver. Whenever the driver is turned on by its appropriate flip-flop, then the output of the driver will complete a circuit to turn on the light bulb. The driver to be used depends upon the power requirements for the device. Drivers are needed because the output of the flip-flop rarely exceeds 30 mA, and this is not enough to operate most electromechanical devices.

Whenever there are bits in the accumulator there will be outputs sent to the AND gates. However, the flip-flops will not necessarily be operated because it takes two inputs to an AND gate to get an output. Consequently, this set of four lights will not ordinarily be changed even though the content of the accumulator is continually changing. But if a particular trial calls for turning on just the first two lights, then ones are loaded into the first two bits of the accumulator, while the next two bits of the accumulator remain at zero. The next instruction in the program must provide a pulse to the four AND gates. This will be the input-output instruction giving the device code for the four lights. Let us call this "device 4." The four flip-flops will follow ones in the accumulator whenever the instruction for device code 4 is issued. In the program, this would be issued only after the accumulator has been loaded with a code which will turn on the desired lights.

A flip-flop is thus set by a one appearing in the accumulator at the time the appropriate input-output instruction is issued. Since the program must eventually turn the light off, it is necessary to provide some ways of clearing as well as setting. That is done by the other output from each of the four accumulator bits. Those four outputs are first inverted so that they take on the opposite sign. Each of these inverted outputs is then connected into an AND gate, the output of which is connected to the "clear" side of the flip-flop. Since device code 4 activates the same four AND gates, a flip-flop will be set or cleared depending on whether it has a zero or a one in the respective bit in the accumulator.

For example, consider only the first AC (accumulator) bit. If it is a one, it will apply a positive voltage level to the AND gate connected to the "set" side of the first flip-flop. The positive level from the AC will also be changed to negative by the inverter, so the AND gate connected to the "clear" side will have a negative level. Now, if device code 4 is issued by the program, a positive level is applied to both AND gates. However, only the one connected to the "set side" will generate an output, so the flip-flop will be set. If the AC bit is now changed to zero, a negative level will be applied to the "set" AND gate and a positive level to the "clear" AND gate. When device code 4 is issued, only the "clear" AND gate will have two positive inputs, so it will generate an output which will clear the flip-flop. In this way, the same device code instruction can change the state of a device back and forth, depending upon the number in the accumulator.

The great advantage of this type of interface is that while only one device code is needed, actually up to 12 separate external circuits can be operated. Further, the different drivers do not have to operate the same kinds of apparatus. Some could be lights, some buzzers, others could be solenoids, and so forth. Of course, it is possible to activate any one of them without affecting the others merely by leaving unchanged the bits in the accumulator for those devices to be left in their previous state.

While it is conceivable that the experimental apparatus can be located next to the computer, so that no cable is needed between the drivers and the apparatus, in most operations these are in separate locations. Consequently a cable is needed. There is no theoretical upper limit to cable length—in fact telephone lines could be used for several thousand miles. In the experiment described, most of the equipment is about 100 ft away, with twisted-pair or coaxial cable providing the interconnection.

While it is impossible to provide an exact estimate of the cost

of this type of interface, since it depends upon the particular computer being used and the details of the application, a rough estimate is possible. Flip-flops cost between $6.00 and $10.00 each, AND gates about $2.00, inverters about $2.00, and drivers from $1.00 to $20.00 depending on their output. Therefore it is possible for this entire four bit interface to cost less than $60.00. There are many sub-varieties of an interface such as the one described, but this is a basic design used with the PDP-8. Different computers require slightly different designs, but the basic principle is similar.

The output buffer described in Fig. 4 is primarily for operating discrete "binary" devices–those that have two states (on and off). It can be used for more complex devices, however. For example, to avoid analog conversion for the luminance control, a 4-bit resistor ladder has been set up for each light source. The 4 bits allow choice of any one of 16 resistances, thus providing 16 programmable intensities. The selection of resistors is controlled by relays which themselves are operated by a 4-bit buffer like that in Fig. 4. A fifth bit would have provided 32 programmable steps.

The motors on the memory drums require only a brief pulse to start, and then internal holding relays keep them on until a hole is sensed in the paper tape on which the stimuli are drawn. They then shut themselves off. Therefore 1 bit is used to turn the motors on.

It should be noted that if the lights cannot be turned on and off, then mechanical devices such as solenoids or stepping motors would be needed to position wedges, filters, or shutters. These devices can be operated easily from the interface just described. For example, a stepping motor moves a prescribed number of degrees for each pulse received. The duration of the pulse is critical, as is the rate at which they are delivered. Such motors can be pulsed from this type of buffer, 1 motor per bit. Precise control over the motor's position can be maintained without feedback by counting the number of pulses delivered, if the computer also is given the starting position. Usually, to protect against error, a feedback device such as a shaft encoder will be used. This operation is described in the next section covering input functions.

## INPUT FUNCTIONS

There are two basic input functions in on-line operation. One is to load information into the computer. This can be conceived as a mirror-image of the output buffer just described, although it is generally smaller, and simpler in scope. The other function occurs when external equipment is used to signal the program to recognize a change of state. This function is performed through the interrupt and skip busses, rather than through the accumulator directly.

An input buffer is needed for the complement of the reasons requiring an output buffer. Whereas computer pulses are short, and an output buffer is needed to store them for the appropriate duration needed to operate the equipment, pulses from external equipment are long, and generally very noisy and ragged; therefore they are not suitable for operating solid state components. Consequently, these pulses have to be conditioned and shaped before they can be inputted directly into the computer or used to operate standard interface components. Further, the program may not be ready to load information from external devices at the right time; sometimes that information may have to be temporarily stored. Except for conditioning, signals for interrupting programs basically require only a switch closure which completes a circuit to the interrupt bus.

To load information into the computer, lines from external devices can be connected directly to the accumulator input. Since the information to be loaded into the accumulator would come in continuously if the lines were direct, it is necessary to place some gating in the circuit. Figure 5 presents a simple interface example. This example uses a 4 bit input buffer which perhaps could be connected to 4 buttons, one or more of which the S might press after each trial; or it could be from a 4 bit shaft encoder, or some other device. As an example, assume that at the end of the trial, the S presses double-pole switches numbers one and two, but not three and four. Coincident with the switch closures (by one pole on each switch), the computer is informed via the interrupt bus that the S has responded. The program would then jump to a routine that would be designed to load information into the accumulator. When the S closes the two switches, this sets the first two flip-flops. The remaining two flip-flops remain in the "clear" state. The output of the first two flip-flops then goes to their AND gates. However, the accumulator will not yet be affected because there will be no output from any of the 4 AND gates. The first step in the computer information loading routine after the interrupt is sensed is to issue the device code "load." When this is done, accumulator bits one and two will change from a zero to a one, the remaining two bits staying at zero. Following the successful loading, the next instruction in the program will be to issue device code "clear." This will reset all of the flip-flops to a clear state so that the next response can be recorded. If there is some danger that the S might inadvertently press one of the switches before it is appropriate, then the "clear" code could be reserved until just before S's response is required. In this way, inappropriate spurious responses would be ignored, because they would be erased before counted. It is obvious that this 4 bit buffer could be expanded by adding further components.

The flip-flop buffer in this input interface is not always necessary. If the buttons being used are of the type that, when pressed, will remain closed until released mechanically, then the storage function of the flip-flops is not needed. Or, if S is pressing only one button at a time and the poles on the button can be arranged so that the signal to the interrupt bus occurs simultaneously with the signal to the accumulator, then no flip-flops are needed to store the result. From the time the interrupt is sensed until the "load" instruction is issued, no more than about 8 to 10 $\mu$sec need elapse. Since the S will be unable to remove his finger from the button in that time, the button switch will still be closed at the time the "load" instruction is issued. Thus, the information will be loaded directly into the accumulator through only an AND gate, without need of a flip-flop to temporarily store the result.

There are a number of other input devices that we are using with an interface similar to the one described in Fig. 5. For example, to read the position of a shaft or arm, either a shaft encoder is needed when digital values are being used, or an analog-to-digital converter must be used to translate analog values. A shaft encoder divides 360 deg of rotation into a binary code of a specific number of bits. A 5 bit code would operate from 5 brushes, each one of which is in contact with a common ground for one-half the circumference. The first bit would be at ground (generate a 1) for 0 deg to 180 deg, and open (generate a 0) for 181 deg to 360 deg. The second bit would generate a 1 for

0 deg to 90 deg, and at 181 deg to 270 deg, and 0 for 91 deg to 180 deg, and 271 deg to 360 deg. The third bit would be a 1 for 0 deg to 45 deg, 91 deg to 135 deg, 180 deg to 225 deg, and 275 deg to 315 deg, and 0 for the rest, and similarly for the remaining bits. Obviously the more brushes (bits), the greater the accuracy of positionings. In this example the 5 bits would be connected directly to 5 AND gates. No flip-flops would be needed. Whenever the "load" instruction is issued, the exact position of the shaft at that instant would be loaded into the accumulator. The duration of the read-in would be 100 nanosec. Hence, even if the shaft is rotating at high speed, the "on the fly" reading would be quite precise and unchanging.

## TIMERS

Timing is a most important function for experiments that are run by computers. This is not just because of a computer's great accuracy, but also because it can handle a large number of timing functions simultaneously. Normally, separate timers have to be used and synchronized in experiments such as the one described. Some small computers come equipped with clocks as part of their hardware. Most, however, have clocks or counters as accessories, or as items that can be added in the interface. Since the PDP-8 falls in the latter category, a brief description will be given as how timing functions are controlled from a single clock and how economical and powerful this procedure is. (Refer to Markowitz & Nickerson, *Behav. res. meth. Instr.*, this issue, for a more detailed discussion.)

The simplest clock is merely a device which provides a brief pulse once every preset unit of time. In this way, a 1 msec clock would provide a brief pulse once a millisecond. This pulse would be fed directly into the program interrupt, and therefore would interrupt the computer once every millisecond to inform it that a millisecond has passed. While this seems very frequent, in fact the PDP-8 can perform 667 different instructions every millisecond, so that for many program steps, the computer will end up waiting for the interrupt. With this kind of interrupting clock, the program then would keep track of the amounts of time that have elapsed by using a core memory location that is set aside by the program as a counting register. No further interface is needed, except for a flip-flop which is used to turn the clock on and off. With this way of handling timing operations, the computer could theoretically keep track of 50 to 100 different timing steps simultaneously. In practice, no difficulty is encountered in timing several lights, reaction time, delays, and duration of pulses to equipment, all in the same program, overlapping in time, with only this one clock (which costs between $25 and $100).

While no description of standard commercial accessories to computers has been given, these are available for all small data acquisition computers. These would include a printer, card punch, card reader, plotter, CRT, light pen, magnetic disc, magnetic drum, magnetic tape, and other such devices. These are made available as standard options by almost all companies selling computers. In many cases accessories can be purchased separately and interfaced without buying the package from the computer manufacturer.

## PROGRAMMING A DATA ACQUISITION COMPUTER FOR ON-LINE OPERATIONS

Since programming languages differed greatly from computer to computer, it is not possible to describe, in the abstract, how to program data acquisition computers. Most of their operations have to be programmed in a language that is only one or two steps removed from basic machine language. While almost all such computers will accept scientific computing languages such as FORTRAN, these are more useful for analyses of data than for running experiments. Small computers differ in their ease of programming; each is usually negatively correlated with ease of interfacing.

For most on-line operations it is nearly impossible to use a complete program written by someone in another laboratory, though many of the subroutines are standard and can be taken from such sources. It is not difficult to program small computers, though the writing of programs can be moderately time-consuming. It is possible to train programmers, but four or five months of on-the-job experience is needed to develop proficiency. Typical programs for visual perception experiments of the type outlined earlier, use about 2000 words of computer memory on the PDP-8.

## SUMMARY

This paper is not intended as a complete guide to the use of a small data acquisition computer in an experiment. However, it has attempted to show the basic principles of such an interconnection and to illustrate how the computer would be used to perform operations. The decision as to whether a computer is economically and experimentally feasible rests with the experimenter, and within the confines of the experimental problem. Programming has not been discussed in detail, since this differs so much from computer to computer. However, we have indicated how the various sections of one type of program are organized. Finally, since this article was primarily designed for individuals who are relatively unacquainted with on-line computer operations, detailed attention has been given to some of the interface components and interconnections.


### NOTE

1. The research program described briefly has been supported by grants from the National Institue of Mental Health MH10753 and MH03244, from the National Science Foundation GB2909 and GB5901, and from the National Institute of Neurological Diseases and Blindness (NB00628). The computer, its auxiliary equipment, and basic interface components, were purchased with the aid of a grant from the National Science Foundation GB 4547 to the Center for Visual Science at the University of Rochester. Most of the programming that we have used on the computer so far has been written by Mr. Larry T. Gell.