Behavior Research Methods & Instrumentation
1978, Vol. 10 (2), 352-355

A microprocessor reaction time laboratory

JOHN L. SANTA and PAUL STREIT
Douglass College, Rutgers University, New Brunswick, New Jersey 08903

The hardware and software required to implement a multiple-subject microprocessor-based laboratory are described. A basic stimulus-probe reaction time paradigm encompasses a wide range of cognitive psychology experiments and requires little or no sophistication to use.

Recent technical innovations have resulted in the widespread availability of microcomputers, which permits a compromise between the general computer installation and the very specialized one-investigator computer. The compromise involves developing a program that requires no sophisticated knowledge, that is, a program that an undergraduate can learn to use with only one or two training sessions. Yet, the program must be sufficiently general to perform a variety of experiments.

In order to realize such a goal, we must first choose a paradigm that will accommodate a reasonable range of psychological experiments. Our choice of paradigm is best described as a basic stimulus-probe situation. Stimuli can consist of any set of alphanumeric material, from a single character up to a full 64-character by 16-line display. The basic paradigm involves presenting the stimulus material for a timed interval (greater than 16 msec) followed by a clearing of the display and a variable-length wait interval. A probe is defined as any set of alphanumeric characters presented for a timed interval followed by a clearing of the screen and a wait period. The probe differs from the stimulus session only in that the onset of the probe is accompanied by the onset of a latency counter. Two alternative responses and latencies for multiple subjects are then recorded for each probe presentation.

With this basic stimulus-probe paradigm, we can perform many of the standard experiments in cognitive psychology. For example, a paired associate or free recall learning experiment requires only the stimulus component, with no responses being recorded. A simple semantic memory experiment, in which subjects respond true or false to assertions such as "A canary is a bird," employs only the probe component of the paradigm, recording latencies and responses to each assertion. A modified semantic memory task might present the first part of the assertion as a stimulus (A canary has) followed by a probe component (wings). The paradigm can also accommodate simultaneous or delayed matching tasks (B followed by B or b or c). Similarly, it is possible to perform search experiments in which a target letter (B) is presented as a stimulus followed by a search set of letters as the probe (AOKBRT). Latency timing begins with the presentation of the search set. A Sternberg task reverses this procedure by presenting a memory set as stimulus, followed by a single letter as probe. It is also possible to perform a vast array of text verification experiments in which the stimulus consists of a probe passage (of any length, but displayed in 16-line sections) followed by a probe question about the passage.

In sum, the standard stimulus-probe paradigm has wide applicability in cognitive psychology. Moreover, by confining ourselves to a general paradigm, it is possible to create a program that requires only a few parameters from the experimenter, and no programming sophistication.

HARDWARE

Our reaction time laboratory is currently implemented on an Altair 8080 computer, although it could be easily adapted to any 8080 machine. The main computer is equipped with a 16k RAM memory, a serial interface card, and a combination parallel-serial interface (3P + S). The computer is connected to six peripherals including: an Icom floppy disk, a SOL video terminal, a Teletype, a modem, a response button interface, and an external kilohertz crystal clock. The SOL is used only as a video display device (although it can be used to control the experiment by itself since the SOL also contains an 8080CPU). The SOL is currently connected to the Altair via a 4,800-baud serial port. The video output of the SOL output is connected to a video monitor, and "daisy chained" to four subject station monitors (SONY 11 in.). The floppy disk is interfaced to the Altair through an s-100 bus card that contains a full disk controller and a monitor in ROM memory.

The response buttons are level-tripped microswitches that are debounced with J-K flip-flops and enter through a single 8-bit parallel port. We currently have four subject stations implemented. The crystal clock is used only to create a stream of 1-msec pulses which enter through a single bit or parallel port.

Total system cost, including four video displays but excluding the Teletype (which is not essential), is about $6,000, with all components purchased, assembled, and tested. The same system can be implemented on a SOL computer with a minifloppy for under $3,000.

SOFTWARE

The reaction time laboratory is built around a general library of assembly language subroutines, together with a master program and a few BASIC language data summarization routines. The entire system is based on a concept of simplicity of function as opposed to elegance of execution. Thus, the entire control system was written in assembly language without interrupt capacity, direct memory access, or higher level language programming. A computer scientist might well shudder at the extravagant way we tie up the central processor. However, most laboratories simply do not need such fast and fancy techniques. In spite of the inelegance, we can accomplish our goals with a minimum of sophisticated hardware and programming.

SUBROUTINES

The subroutines in the laboratory library can be divided into four sections as follows: disk routines, timing routines, display routines, and stimulus-probe paradigm routines. The primary disk routines are externally provided by the ICOM system monitor. The master program is executed with a RUNGO, infile, outfile statement. Such a statement causes the monitor to set up disk pointers so that successive bytes are read from the input file and written to the output file with each call to the disk read or write routines. At the end of the session all files are closed by calling an "update" routine that is also provided by the ICOM monitor.

The timing routines have been designed to create an internal six-digit decimal millisecond clock. The clock is generated from a stream of millisecond pulses applied to a single bit of an input port. The clock routine checks the pulse bit looking for a transition from high to low. When a transition occurs, the internal clock is incremented by one. Thus, all of our response service routines must be performed within 1 msec in order to return in time to detect each transition. Although this method of timing is inelegant, it is effective. With the 8080 cycle time we are able to perform up to 300 instructions before returning to check the clock pulse. Consequently, we have found no need for sophisticated timing latches or interrupt devices.

The second timing service is a simple "wait" routine that registers a six-digit decimal time in milliseconds and then compares the master clock to this value. Control is returned to the main program when the master clock reaches the preset wait time. The final timing routine is used to reset the master clock to zero.

The display subroutines allow the user to position the cursor, write an ASCII character, clear the screen, blank the screen while writing, and display the full screen. The cursor position routine accepts a two-digit decimal x address (0-63), followed by a two-digit y address (0-15). The cursor is moved to the specified location and writing commences from this point.

Many standard experiments can be performed by simply writing characters serially onto the screen. At 4,800 baud a five-letter word is written in approximately 10 msec, which in many situations is adequate. However, when a long text must be written, or presentation time is critical, it is possible to call the blanking routine. The blanking routine outputs a bit to a parallel port that is hardware connected to a video gating line in the SOL terminal. Thus, characters can be written, but they will not be displayed. After the entire stimulus has been written, the display routine is called to change the parallel bit and enable the video composite. The full screen appears within a single raster scan.

The stimulus-probe paradigm routines include three major components: the main control program, the stimulus routine, and the probe-response routine. The main control program is basically a simple interpreter that reads characters from disk and checks for paradigm instructions. The full instruction set is listed in Table 1. The main program begins by asking for the number of subjects in the current session. After receiving this information from the operator, the program proceeds to read successive characters from the input file. Each character is initially compared to a table of primary control characters (#, @, %, *, /). If one of the five control characters is encountered, control is transferred to a subroutine associated with that character. The # sign is used to denote a time parameter. The occurrence of a # causes a second level of comparison for one of six characters following the # sign (#, L, S, I, P, T). A second # sign denotes the end of the session. Any of the other five letters would be followed by a six-digit decimal number to set a time parameter. (See Table 1 for a description of each parameter.) Once a time parameter is set, it remains in effect until changed. Therefore, it is possible to set the parameters only once at the beginning of the session, but it is also possible to change them as often as necessary.

The @ character is used to control the cursor. It is always followed by a four-digit x-y coordinate that determines the starting location of both stimuli and probes. The @ can also be used within a stimulus or probe text. If the @ appears within text, it will not be displayed but will relocate the cursor. This feature makes it possible to create complex spatially distributed displays.

The % character is followed by six ASCII characters and is used to provide condition coding information as a header for the response table. The six characters following the % sign are simply written onto the disk immediately in front of the response-latency table.

A pair of * characters are used to enclose a stimulus display. The first occurrence of an * calls the stimulus display routine, which immediately clears the screen, blanks the screen, and writes successive characters starting at the default cursor location. When the second * is encountered, the video display is enabled and the processor goes into a wait state for the stimulus time. At

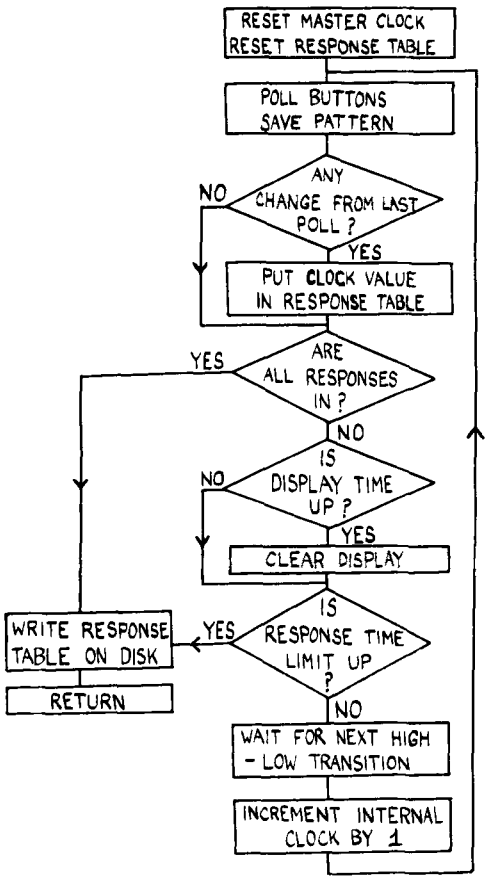

Figure 1. Flowchart analysis of the response polling routine.

the end of the display interval, the screen is cleared and the processor waits for the interstimulus period.

A pair of / characters are used to enclose a probe display, for example, /A CANARY IS A BIRD/. The occurrence of the first / character causes the screen to be cleared and blanked. Characters are then written into the display memory. The next / sign encountered enables the video composite signal and calls the response routine.

The response routine is the single most complicated subroutine in the laboratory library. Figure 1 provides a flowchart analysis of the response routine. Entry to the routine immediately resets the master clock and clears the response latch together with the internal memory image. A polling procedure is then begun in which the response latch is compared to the current memory image of the response pattern. The program is looking for a change in the response pattern between the current latch and the last time the response buttons were polled. If a change has occurred, the current master clock value is stored in the response table. The response table is organized as a set of eight adjacent 3-byte memory locations. Thus, each of the eight table sections can accommodate a six-digit BCD number, and each of the eight sections is associated with a unique external pushbutton. At this point the program checks to see if all subjects have responded; if they have, the entire response table and condition code header are written onto the disk. If all subjects have not yet responded, the program checks to see if the probe display time has been reached; if so, the screen is cleared. A check is then made to determine if the response time limit has been reached. If the limit has been met, the response table is written onto the disk; if not, the program returns to monitor the external clock pulses, waiting for a high-to-low transition. The transition causes the internal clock to be incremented by one and another polling cycle is begun. All polling and comparisons must take place within a single millisecond. This presents no difficulty since a worst-case analysis of the program reveals that the time between clock checks is on the order of 650 microsec.

CONCLUSIONS

The program provides a useful paradigm for a variety of cognitive psychology tasks. Moreover, it has the virtue of being extremely simple to use. A person need not master a programming language. The user need only learn to use a text editor to type in an input file. The file controls the entire experiment via the commands shown in Table 1.

Table 1
Control Characters and Their Functions

Command Character	Function	Examples	
*	Stimulus Display	*STIMULUS*	
/	Probe Display	/PROBE/	
%	Coding Information	%_____	Using any six alphanumeric characters
@	Cursor Position	@XXYY	XX is X position (00-63) YY is Y position (00-15)
#	Time Parameters	#SXXXXXX	Stimulus display time limit
		#IXXXXXX	Interstimulus time limit
		#PXXXXXX	Probe display time limit
		#LXXXXXX	Latency time limit
		#TXXXXXX	Intertrial time limit
		XXXXXX	Six-digit number (000000-999.999) seconds
		##	End of experiment

Table 2
Sample Experiment File

```
#S010000      ; 10-sec stimulus time
#I005000      ; 5-sec interstimulus time
#P000500      ; ½-sec probe display
#L040000      ; 4-sec latency default time
#T120000      ; 12-sec intertrial time
@1005         ; set cursor to 10th character and 5th line
%T01001
*A CANARY HAS*
/WINGS/
%F02001
*A HORSE HAS A*
/BILL/
%T03002 *AN ANIMAL HAS*/LUNGS/
##            ; end of file
```

An example of a short stimulus file is illustrated in Table 2. The first five lines of the file are used to set the time parameters in milliseconds. The sixth line sets the cursor position to the point where the stimuli and probes will begin to be written. Note that it is perfectly possible to have extra comments, linefeeds, and spaces in the files since all extraneous characters are ignored by the program. As the example illustrates, the coding information, stimulus, and probe materials can be written on single or separate lines.

In sum, the equipment is inexpensive, the paradigm is flexible, and the program is easy to use. Hopefully, we have a fancy memory drum/T-scope and Hunter Klockounter in a simpler, more versatile package.