A microcomputer interface for control of real-time experiments in cognitive psychology

ROGER RATCLIFF
Dartmouth College, Hanover, New Hampshire 03755

and

W. M. LAYTON
AGS Corporation, Hanover, New Hampshire 03755

A microcomputer system for real-time control of experiments in cognitive psychology is described. The microcomputer serves as an interface that allows a remote timesharing computer to control the timed display of textual material on CRTs and collect response times accurate to 1 msec. It can control two CRT subject stations presenting the same or different experiments and control other devices such as slide projectors and tape recorders. It is argued that such special-purpose microcomputer interfaces provide a real-time laboratory with significantly less effort than does the more traditional laboratory minicomputer.

A major portion of the time of many cognitive psychologists is spent in the development and use of laboratory computer systems. Such systems have often taken the form of dedicated laboratory minicomputers for precise experimental control of stimulus presentation and response recording. The rapid decrease in cost and the rise in power of microcomputers now make it feasible to develop special-purpose microcomputers that act as intelligent controllers of experiments. The system described in this paper uses a microcomputer-based interface to allow a remote timesharing computer to control timed display of textual material and collection of response times. It also gives the timesharing system the ability to control devices (e.g., slide projectors and tape recorders) and to receive input from on-off devices (e.g., extra pushbuttons) or analog input (e.g., potentiometer or GSR levels).

The interface is plugged in between the computer and the terminals. To the computer it is exactly like a terminal; to the terminal the interface is like a computer. Thus, no spacial connections or modifications are needed to the computer or the terminals to install the interface (although extra connections can be added to provide better control of the CRT displays). When the system is first switched on, the interface is in its "transparent" mode and one of the terminals operates as if it were connected directly to the timesharing computer. A sequence of special characters converts the interface into test mode and it begins executing real-time commands. Transparent mode is reentered by pressing the reset button or sending another sequence of special characters.

Stimulus material is prepared on the main computer and sent to the interface, which displays it on the terminals and sends responses and response times back to the main computer. Timing and device control commands are embedded in the stimulus text. For example, in the line "Dog #W500 Cat #R" the #W500 instructs the interface to wait 500 msec between the display of "Dog" and "Cat," and the #R instructs it to wait for the subject to press a key and then send the character and response time back to the main computer. These embedded commands replace all of the complicated programming and special hardware normally associated with real-time stimulus presentation. The result is an easy-to-use system that can be connected to almost any main computer, to alleviate many of the problems the harried cognitive psychologist encounters in setting up a real-time laboratory.

It is useful to note some of the advantages a microcomputer-based interface to a timesharing computer might have over a stand-alone system. First, the system is far cheaper if there is access to a timesharing computer because there are no requirements for mass storage and data reduction, analysis, or transmission software. Second, there is reduced maintenance: The computer center services the timesharing computer, and the simplicity of the interface computer creates few problems. Third, the larger the system, the greater the number of programs available in libraries, and the easier is the programming compared with a stand-alone system.

If a free (or cheap) timesharing computer is unavailable, the interface can be connected easily to a stand-alone microcomputer such as a Terak or Apple and

Roger Ratcliff provided some aid in system design and Mike Layton performed the system implementation. For information on a commercial version of this system, contact W. Michael Layton, AGS, Box 64, Hanover, New Hampshire 03755. We would like to thank Jim Hansen for providing the microcomputer for the prototype version of the system and Gail McKoon for comments on this paper. Reprint requests should be sent to Roger Ratcliff at the Psychology Department, Yale University, Box 11A Yale Station, New Haven, Connecticut 06520.

0005-7878/81/020216-05$00.75/0

on those machines.

Such microcomputer systems have recently been developed or are currently under development within the framework of a specific computer laboratory (e.g., Bailey, Ward, Spear, Leatherman, Waite, & Christian, 1979; Fox, Ward, & Lesgold, 1979). However, as yet, transportable systems have not been developed (see Friendly & Franklin, 1979, for additional discussion of such designs).

THE COMMAND LANGUAGE

The interface was designed to simplify the real-time programming required to control and measure the timing of events in experiments. The commands are sufficiently powerful to present the material to the subject in almost any timing sequence desired, but are simple and very easy to use. In fact, it usually takes ½ h to add and debug the real-time components once the stimulus lists have been constructed and placed in disk files on the timesharing computer.

Table 1
Cognitive Interface Commands

Transmission and Buffering Commands

%0 sends the text to both terminals.
%1 sends the text to Terminal 1 only.
%2 sends the text to Terminal 2 only.
%B signals the interface that it can begin processing and displaying the preceding text and commands. All text and commands between %Bs are stored in the buffer until the %B is received. Thus, the number of characters between %Bs must not exceed the size of the buffer.

Real-Time Commands

#R collects a response from the terminal and sends it back.
#Wttt waits for ttt msec and then continues.
#WDn=m waits until Input Line n is low if m = 0, high if m = 1 (used for tape-recorder input or voice key).
#N turns on the screen.
#F turns off (blanks) the screen.
#Dnm sets Output Line n low if m = 0, high if m = 1 (used for control of slide projector).
#I(R=?)<t1><t2> displays the text and executes the commands of t1 if the last response was a "?" and those of t2 if it was not.
#Jnn skips over all text and commands until a #Mnn is encountered.
#Mnn marks a position for the #J command.

Cursor Commands

@nnmm moves the cursor to the nth line and mth column. The screen has 80 columns and 24 lines.
@C clears screen and moves cursor to home.
@B erases to end of line.
@E erases to end of page.

Special Strings

$T displays the last response time for the subject.
$R displays the last response.
$C displays the millisecond clock time.

Macros

$$1sssssss$$ defines macro string "sssssss."
$1 is replaced by its defined macro string wherever it appears in the text. For example, after the definitions $$1cat$$ and $$2dog$$, the text "$1 and $2" is expanded to "cat and dog." Macro strings must be less than 60 characters long.

A Sample Real-Time Program for the Sternberg Paradigm

```
@C to begin press space bar.#R @C #W1000
DOG #W1000 @C
COST #W1000 @C
ACHIEVE # W1000 @C
**** #W1000 @C
COST #R
%B
@C To begin press space bar.#R @C #W1000
ITEM #1000 @C
CLOCK #W1000 @C
COPY #W1000 @C
**** #W1000 @C
FIRST #R
%B
Test ended.%B
```

The ease of programming the interface stems from its system of embedded commands (Table 1). The use of these commands is best introduced by example. Table 2 shows a simple real-time program to present the Sternberg paradigm (Sternberg, 1966). The trial begins by clearing the screen (@C) and presenting a message that asks the subjects to initiate the next trial by pressing a key. (A response time collected by #R and sent to the timesharing computer is ignored in analysis.) The screen is again cleared and after a delay of 1,000 msec (#W1000), the study words DOG, COST, ACHIEVE are presented as the memory set, one at a time for 1,000 msec/word. After the last stimulus item disappears, a row of asterisks is displayed for 1,000 msec to signal the test item. When a response is made, the key and the response time are returned to the controlling (e.g., timesharing) computer and the next trial proceeds. The "%B" is used to separate trials, and the interface begins displaying a trial only when the entire trial (including the "%B") is in the buffer. This guarantees that each trial will be presented without interruption.

The program of Table 2 produces the desired results, but a number of improvements are possible. The first involves the use of macroinstructions (macros) to reduce the number of characters that must be transmitted to the interface. When one becomes accustomed to macros, they also increase the flexibility and ease of programming. The concept is simple: The string $$1xxx$$ defines macro $1 to be xxx; whenever $1 occurs in the text it is automatically expanded to xxx. Thus, after the definition $$2 #W1000@C $$, the line COST$2 is expanded to COST #W1000 @C. Table 3 shows the program of Table 2 shortened by macros.

A second addition to the program positions the text nearer the center of the CRT screen. This is useful because the distortion on most CRTs is least in the center. The command @1010 placed before each word to be displayed positions it at the 10th row and 10th column. If the macros $1 and $2 are used, the @1010 can be placed at the end of each macro and the rest of the program remains unchanged.

Table 3
The Sternberg Paradigm Using Macros

```
$$1 @C To begin press space bar.#R @C #W1000 $$
$$2 #W1000 @C $$
$1DOG$2
COST$2
ACHIEVE$2
****$2
COST#R
%B
$1ITEM$2
CLOCK$2
COPY$2
****$2
FIRST#R
%B
Test ended.%B
```

Another useful modification uses the command language's decision capability, which is based on the subject's last response. In the Sternberg (1966) paradigm, for example, we might try to increase the accuracy of the subject's responses by imposing a 5-sec delay to penalize each wrong answer. This is accomplished by adding

#I(R = Y) <RIGHT #W1000> <WRONG #W5000>

after COST#R and

#I(R = N) <RIGHT #W1000> <WRONG #5000>

after FIRST#R. If the response is a Y in the first case (or an N in the second), the word RIGHT is displayed for 1 sec. If the response was wrong, WRONG is displayed for 5 sec.

A final addition to the program increases the accuracy of the display and response times. The interface communicates with the CRT at 960 characters/sec, but even at this high rate it takes almost 2 sec to fill the entire screen. To allow the presentation of large blocks of text for precise durations, we have modified the CRTs to allow the interface to turn off the electron beam (and thus the image) while the screen is being filled. The commands #N (on) and #F (off) accomplish this in the language. In the example, we could use #F COST #N #R to insure that COST is displayed all at once.

The screen blanking commands are also useful for avoiding a timing problem produced by the raster scan display system used in most CRTs. In this system, the beam sweeps the screen once every 17 msec. This means that each phosphor on the screen glows for a few microseconds once every 17 msec. When very accurate presentation of stimuli is required, the position of the beam is critical. Beam position is also critical when very accurate response times are needed. If, for example, a word is displayed just before the beam reaches that position on the screen, the subject will see the word almost immediately. If, however, the beam has just passed the word's position, the word will not become visible for 17 msec. The #N command solves this problem by always waiting until the beam is at the top of the screen before turning the beam on again. A finite time (17/24 msec per line from top) still elapses before a given line is illuminated after the #N turns on the beam, but this time is constant for stimuli presented in the same position on the screen. Thus, the #F COST #N#R phase also provides less variable response time.

STIMULUS PREPARATION AND RESPONSE ANALYSIS

One of the advantages of the interface system is that a large timesharing system can be used to prepare the stimulus lists and to analyze the responses. Typically, editors are used to generate the stimulus lists and then FORTRAN programs are used to randomize them. Real-time functions are added by minor modification of the WRITE statements, and the final stimulus list is written to disk files. A short BASIC program transfers the disk files to the interface. Because the real-time commands are simple and visible in the text, a careful reading of the prepared file will discover most errors and very little real-time debugging is necessary.

Responses are continuously transmitted back to the main computer and are available immediately at the end of each session. FORTRAN programs are used for analysis. The programs that generate the stimulus lists also produce scoring files that are used to automatically process the data.

Both preparation and analysis are aided by the timesharing system's high-speed printers, plotters, library statistics routines, and other utilities. The computer center also provides automatic back-up in case of system failure and a magnetic tape system that allows economical storage of all programs, stimulus lists, and data.

COMMUNICATION WITH THE CONTROLLING COMPUTER

All communication between the interface and the controlling computer takes place over one standard (RS-232C) computer connection. The rate may vary from 10 to 1,920 characters/sec (we generally use 30 characters/sec). An acoustic coupler allows this connection to be made from any telephone.

Because the interface appears to be a normal terminal, very little software and no special hardware is required on the controlling computer. A short BASIC program is used to transmit the stimulus list from disk to the interface and to store the responses returned by the interface back onto the disk. If the two control terminals are receiving the same stimulus lists, the list need be transmitted only once. If the lists are different, the BASIC program merges them, adding "%1" and "%2" to direct the text to the appropriate terminal. The two subjects run completely independently of each other.

as presently configured and that class involves response-contingent feedback or stimulus construction. Such paradigms can be run on the system if the communication routine (running on the timesharing computer) is incorporated in the list generation program so that responses can be used in further list construction. If a microcomputer such as an Apple or Terak is used, then communication can be at 9,600 baud, thus reducing time delays to a minimum.

INTERFACE HARDWARE

The microcomputer is an M6800 with 8K bytes of random-access memory (RAM), 2K bytes of read-only memory (ROM) (in which the real-time interpreter resides), three serial interfaces, and a parallel interface. The microcomputer connects to the main timesharing computer through one serial interface through an Anderson Jacobson coupler via telephone line at a 300-baud (bit/second) transmission rate. (This rate has been fast enough to maintain two subjects without delays between blocks of trials.) The microcomputer connects to two Datamedia Elite 1520 CRT display terminals, each through one serial interface, and communicates at a 9,600-baud transmission rate. Dummy characters are sent to maintain a 9,600-baud transmission rate (960/sec) and timing intervals are calculated by counting characters, thus giving accuracy of about 1 msec. The parallel interface is designed to serve two functions: (1) to control CRT screen blanking, which allows large texts to be displayed in one 17-msec sweep of the CRT screen and provides response recording consistency to 1 msec and (2) to control other peripheral devices (e.g., slide projectors, tape recorders, and extra switches).

Other interfaces have been designed for this system that provide analog input and output which can be used for such things as reading GSRs and controlling light intensities. The total cost of the hardware was about $1,000 for the basic microcomputer (not including construction or design costs), $1,040 each for the Datamedia terminals, and $240 for the telephone coupler.

INTERFACE SOFTWARE

The following describes the program that is permanently stored in ROMs within the microprocessor. Knowledge of the structure of the program is unnecessary for the day-to-day use of the interface.

The interface program consists of three timeshared processes plus an interrupt process. Two of the three timeshared processes execute commands and control the two terminals. The third handles interaction with the timesharing computer. These timeshared processes are serviced at least once every 1,024 microsec and have no direct interaction with each other. Each process appears processes interfere with each other. The interrupt process is activated every 1,024 microsec, when the first terminal's port is ready to send another character. A character is transmitted to each CRT buffer (a null character if no other character is to be transmitted) and a 10-digit decimal clock is updated.

The two processes associated with the terminals are identical and execute the same instructions but use different data areas. One routine (GET) is responsible for getting the next character to be processed. Characters come either directly from the buffer or are retrieved from the macro area as part of a macro expansion. If the buffer is empty, that is, does not contain a "%B," the GET routine waits until the input process puts one there. Another routine (PUT) simply gives a character to the interrupt process and waits until it is accepted. The remaining routines in the terminal processes use GET and PUT to process the commands.

There are three first-in first-out (FIFO) buffers, two for characters waiting to be processed and one for characters waiting to be sent to the timesharing computer. The size of the buffers can be changed to optimize the transmission rates, but are normally set for the input buffers to hold 2,200 characters and the output buffer to hold 1,912. In addition, two areas holding 1,280 characters are used for storing macros.

The original program required 2-3 months to write by a skilled assembly language programmer and has been in use for 2 years with only minor modification to add new capabilities.

The cost of the software is probably three or more times that of developing software on a minicomputer capable of testing one subject. However, it is common to see a year wasted in the development of a testing station for a single subject. If microcomputer systems such as this are marketed, these systems can be made available at a fraction of the development cost. Once one system has been developed, the time required to produce additional systems is reduced to that of building the hardware (a few days) and burning in new ROMs.

RELIABILITY, EASE OF OPERATION, AND ADVANTAGES OVER A STAND-ALONE SYSTEM

In the 22 months the microcomputer system has been operational, 80 experiments using an average of 32 1-h subject sessions have been performed. The microcomputer was out of action for half a day with a malfunctioning semiconductor chip. Since then there have been 20 months of trouble-free performance. The timesharing computer is generally very reliable, but we expect to lose 2%-10% of the subject data as a result of timesharing problems.

The ease of operation is truly remarkable. Students with no prior knowledge of the system have had the real-time program debugged within 2 h of beginning to learn the system (note that the stimulus list program

stimulus lists are constructed on the timesharing computer and data are immediately available on the timesharing computer means that there is little software development required, and maintenance of that end is in the hands of the computer center. The price of the microcomputer ($1,000) is relatively small, so it would certainly be possible to purchase one or two back-up systems for use in case of hardware problems. Thus, reliability is assured.

A further advantage of this microcomputer system is that it can be interfaced to any laboratory minicomputer (e.g., PDP-11) or microcomputer (e.g., Apple). Thus the laboratory computer would provide mass storage and analysis capabilities while the microcomputer interface system could take care of the real-time operations. This is a more sensible approach than spending months developing real-time routines for the laboratory computer that would be able to run only one subject station.

It is of interest to discuss further reasons for designing a system that interfaces to a timesharing computer rather than designing a stand-alone system. First, at the end of each subject's session the data is immediately available on the timesharing computer for analysis and storage on magnetic tape. This saves time that is often used in saving data in a temporary form on floppy disk, magnetic or paper tape, and in transporting and saving the data in a more permanent form. Second, the construction of stimulus lists (together with real-time control instructions) is accomplished on the main timesharing computer in our application so that literally the only operations that involve the microcomputer are turning it on and off. It is not necessary to write a new stand-alone real-time program for the microcomputer for each new experiment, because the microcomputer interprets and performs all the real-time operations that are specified in the stimulus list. Thus the programming is a one-stage process on the main timesharing computer.

CONCLUSION

The microcomputer-based system has several major advantages over most other experimental systems. First, it is extremely easy to use; adding and debugging the real-time commands usually takes ½ h and the system is very easy to learn. Second, the system is relatively inexpensive and may be interfaced to most large timesharing systems, as well as dedicated minicomputers and microcomputers. This has the advantage of requiring little or no real-time software development. Third, stimulus list construction and data analysis can be carried out on the timesharing computer without any data transfer problems, which makes it very easy to set up and perform pilot experiments. Finally, it is easy to move the interface from one computer system to another and to exchange experiments with colleagues.

REFERENCES

BAILEY, D. E., WARD, W. H., SPEAR, T. L., LEATHERMAN, R. L., WAITE, J. L., & CHRISTIAN, T. W. The CLIPR remote laboratory microcomputer system: Development and applications. *Behavior Research Methods & Instrumentation,* 1979, **11,** 281-292.

FOX, J. L., WARD, R. V., & LESGOLD, A. M. Microcomputer uses in a timeshared facility. *Behavior Research Methods & Instrumentation,* 1979, **11,** 271-280.

FRIENDLY, M., & FRANKLIN, P. Computer control of memory experiments on a large-scale timesharing system. *Behavior Research Methods & Instrumentation,* 1979, **11,** 212-217.

STERNBERG, S. High speed scanning in human memory. *Science,* 1966, **153,** 652-654.