A method for investigating the sequential ordering of simple movement patterns: An IBM-based application using the Tecmar Labmaster

IAN MICHAEL FRANKS

The University of British Columbia, Vancouver, British Columbia, Canada

A computer program and hardware interface are described that enable the examination of various phenomena related to the organization of simple movement patterns. Components of the program focus on the tracking of serial stimulus events; the reproduction of varied sequential, rhythmic patterns; and the production of self-initiated rhythmic patterns. The program controls the input from and output to the parallel ports of a multifunction data acquisition expansion board via a custom-built circuit board.

A substantial number of complex motor skills involve the sequential temporal ordering of simple movement patterns. How subjects organize these simple movements has been the focus of a great deal of research in experimental psychology. Questions concerning the encoding, cognitive organization, and execution of movement patterns were manifest following Lashley's (1951) investigations into the general problem of serial order in behavior. Lashley brought into focus such issues as hierarchical and associative organization of behavior, the nature of rhythmic movement, and the central control of motor actions. Since that time, general models and theories have been put forward to explain such diverse phenomena as serial pattern learning (Collard & Povel, 1982; Restle, 1970), the control of rapid movement sequences (Rosenbaum, Hindorff, & Munro, 1987), the internal representation of temporal patterns (Povel, 1981), the perception and production of rhythms (Kolars & Brewster, 1985), and the structure of motor programs (Keele & Summers, 1975).

To facilitate the study of such phenomena, we developed a generalized computer program and hardware interface. This program, SOSMP (an acronym for the sequential ordering of simple movement patterns), has the potential to control several variations in experimental methodology. Simple sound and light patterns can be created by the experimenter to be used as serial event stimuli (through speakers or LEDs), and the subject's responses (depression of pushbutton switches) to these patterns are compared with the stimuli and stored for later analysis. Three general types of experiments can be or-

ganized using this program. The *tracking* experiment requires the subject to respond simultaneously to the onset of each event within the sequence of events. The *reproduction* experiment requires the subject to reproduce a given pattern following a programmable delay. The *production* experiment allows the subject to produce his/her own self-initiated pattern of events.

The SOSMP program was designed to operate in conjunction with the digital input/output (I/O) of the Tecmar Labmaster multifunction data acquisition board that was resident in an IBM XT microcomputer. Although it is possible to use the parallel port and the programmable clock of the IBM computer for this application, the Labmaster was chosen for several reasons. First, the Labmaster has the capability of reliably timing an event to an accuracy of 0.1 msec, whereas the resident clock within the IBM XT increments at only 18.2 Hz. An expansion board programmable clock is another alternative, since it can time to an accuracy of 0.5 msec. However, not all of these expansion boards (especially for IBM compatibles) have the same chip responsible for timekeeping. Thus the software would have to be adapted for different expansion boards, since the port locations and the process by which the numbers are accessed vary. In addition, the crystal responsible for timekeeping on the Labmaster is extremely reliable and designed specifically for experimental applications.

A second reason for using the Labmaster instead of the computer's parallel port relates to the availability of many independently controlled events. Although the specific application described here utilizes 20 possible stimulus output events (10 LEDs linked to 10 speakers) plus 2 warning events and 10 inputs (keypresses), the experimental procedure was initially designed to accommodate a maximum number of 50 distinct visual and auditory stimuli and 50 responses that can be controlled independently, in addition to the warning events. The three

Funding for this project was obtained from the Natural Sciences and Engineering Research Council of Canada (Grant A0434). Requests for reprints should be sent to Ian M. Franks, University of British Columbia, 6081 University Blvd., Vancouver, British Columbia V6T 1W5, Canada.

Figure 1. General overview of the interaction between various components of the system.

8-bit ports of the digital I/O can yield 3×256 independently controlled events, whereas the parallel port of the computer can access 8 bits of data output with only 5 bits of input. Use of the Labmaster, therefore, allows expansion of the present system with a relatively small adaptation in programming and hardware.

A customized circuit board was designed to interface the Labmaster and an array of 10 stimuli and the appropriate warning events (see Figure 1). The SOSMP program,

written in Turbo Pascal, records the event and the time of each response (i.e., the absolute time that the specified key is depressed and the time of release). It does not, however, complete any analysis of these events. Instead, the SOSMP program stores the events in a format that can easily be read by another program, which in turn can be used for analysis. The program we used to analyze these data was Lotus Symphony. One feature of Symphony is the use of *macros*, which are computer programs

Figure 2. A detailed physical layout of the SOSMP circuit board.

Table 1
Switch Debouncing Wiring List

IC1		IC2	
Switch 2	Switch 1	Switch 4	Switch 3
1 Socket1-3,R	8 IC1-12	1 Socket1-9,R	8 IC2-12
2 IC1-6	9 Socket1-2,R	2 IC2-6	9 Socket1-6,R
3 IC1-4(out)	10 IC1-11	3 IC2-4(out)	10 IC2-11
4 IC1-3	11 IC1-10(out)	4 IC2-3	11 IC2-10(out)
5 Socket1-4,R	12 IC1-8	5 Socket1-10,R	12 IC2-8
6 IC1-2	13 Socket1-1,R	6 IC2-2	13 Socket1-5,R
7 GND	14 +5 V	7 GND	14 +5 V

IC3		IC4	
Switch 6	Switch 5	Switch 8	Switch 7
1 Socket2-1,R	8 IC3-12	1 Socket2-5,R	8 IC4-12
2 IC3-6	9 Socket1-12,R	2 IC4-6	9 Socket2-4,R
3 IC3-(out)	10 IC3-11	3 IC4-4(out)	10 IC4-11
4 IC3-3	11 IC3-10(out)	4 IC4-3	11 IC4-10(out)
5 Socket2-2,R	12 IC3-8	5 Socket2-6,R	12 IC4-8
6 IC3-2	13 Socket 1-11, R	6 IC4-2	13 Socket2-3,R
7 GND	14 +5 V	7 GND	14 +5 V

IC5	
Switch 10	Switch 5
1 Socket2-11,R	8 IC5-12
2 IC5-6	9 Socket2-10,R
3 IC5-4(out)	10 IC5-11
4 IC5-3	11 IC5-10(out)
5 Socket2-12,R	12 IC5-8
6 IC5-2	13 Socket2-9,R
7 GND	14 +5 V

Note—The integrated circuits (ICs) are numbered on Figure 2. These ICs (1 through 5) are 7400 quad 2-input NAND gate circuits. The R in the wiring list indicates that the pin should also be connected through a 2.7K resistor to +5 V. The (out) in the wiring list indicates that this pin provides the debounced state of the corresponding switch and is fed to the priority encoding circuit.

that take the place of many Symphony commands. The macro, used in conjunction with SOSMP, loads the stored data onto the spreadsheet, thus allowing the experimenter to view the data and complete various analyses of the data.

SOSMP CIRCUIT BOARD

The circuit board is made up of a number of subcircuits, each of which carries out a specific task (see Figure 2).

Switch sensing is carried out by the left half of the circuit board, which includes the *switch debouncing* circuit and the *priority encoding* circuit. Sockets 1 and 2 are connected to the response switches, and these signals are fed into the switch debouncing circuit. The 10 pushbutton switches are mechanical and thus are susceptible to mechanical switch bouncing. To overcome this, each switch is fed into a switch debouncer that consists of NAND gates configured as “flip-flops” (see Table 1 and Figure 3). These NAND gates are found on the five 7400 chips (note that the switches used are single pole, double throw). The signals from the switch debouncing circuit are then fed into the priority encoding circuit (see Table 2). This circuit has 10 inputs and outputs, and a sin-

gle 4-bit number represents each switch from 1 to 10. If more than one switch is activated simultaneously, only the binary code of the switch with the highest priority appears at the output of this subcircuit, which is made up of two 74148 chips and one 7400 chip. Finally, the signal goes to port A of the Labmaster board via socket 5.

Stimulus generation is carried out by the right side of the SOSMP circuit board (refer to Figure 2). Six wires originate from port C of Labmaster and enter socket 5. The 4 least significant bits represent a number from 0 to 11 and are sent to the “1-of-16” decoder (see Table 3), which activates 1 of 11 separate lines. These signals are then sent to the LED driver sections (see Figure 4) and to the quad bilateral switches (through the 7406 inverter). Tones are generated by the *tone generation* subcircuit (see Figure 5), and signals are sent to the quad bilateral switches. Further processing of the signal through the quad bilateral switch enables a decoder to output a “1” on the switch. Bit 5 represents LEDs, and bit 6 represents tones. The signal then goes to the *LED/Tone Ground* subcircuit. Bits 4 and 5 from Part C of the Labmaster indicate whether LEDs, tones, or both should be used for the stimuli. Instead of all LEDs being connected to ground, they are connected to the collector of an NPN transistor, and this transistor's emitter is connected to ground. Hence, the LEDs are not connected to ground unless the base of the NPN transistor is properly biased. Bit 4 is used to bias the base of the transistor through a resistor. If bit 4 is 0 V, then the transistor is off and the LEDs are not connected to ground. Alternatively, if bit 4 is 5 V, then the transistor is on and the LEDs are connected to ground. Similarly, the speakers are also connected to the collector of an NPN transistor and are biased through bit 5 (see Figure 6).

SOSMP PROGRAM

Structure

The SOSMP program is menu driven, and the general flow of the program is shown in Figure 7. The menu consists of five items: Check Digital Input/Output, Edit Cycles, Tracking, Reproduction, and Production. The last three items of this menu list deal with the control and operation of specific experimental methodologies and procedures, and they are grouped in the following description.

Figure 3. A circuit diagram of the switch debouncer.

Table 2
Priority Encoding Wiring List

IC6	
1 IC2-3 (Switch 4)	9 IC8-13
2 IC3-11 (Switch 5)	10 GND
3 IC3-3 (Switch 6)	11 IC1-11 (Switch 1)
4 IC4-11 (Switch 7)	12 IC1-3 (Switch 2)
5 Socket5-7,IC7-15 (bit 3)	13 IC2-11 (Switch 3)
6 IC8-4	14 unused
7 IC8-2	15 unused
8 GND	16 +5 V
IC7	
1 +5 V	9 IC8-12
2 +5 V	10 IC4-3 (Switch 8)
3 +5 V	11 IC5-11 (Switch 9)
4 +5 V	12 IC5-3 (Switch 10)
5 GND	13 +5 V
6 IC8-5	14 unused
7 IC8-1	15 IC6-5
8 GND	16 +5 V
IC8	
1 IC7-7	8 unused
2 IC6-7	9 +5 V
3 Socket5-3 (bit 1)	10 +5 V
4 IC6-6	11 Socket5-1 (bit 0)
5 IC7-6	12 IC7-9
6 Socket5-5 (bit 2)	13 IC6-9
7 GND	14 +5 V

Note—ICs 6 and 7 are 74148 1-line to 3-line priority encoders. IC8 is a 7400 quad 1-input NAND gate. These ICs are wired together to encode 10 lines (the 10 pushbutton switches) into a 4-bit binary code (bits 0 to 3). These 4 bits are connected to socket 5, which is then connected to the Labmaster.

Check Digital Input/Output. This subprogram (see Figure 8) can be used to ensure that the SOSMP circuit board is connected correctly and is working properly. Once this option has been chosen, the array of switches, LEDs, and speakers becomes active. Pressing a switch activates the corresponding LED and speaker. In this manner, all of the switches, LEDs, and speakers can be tested.

Edit Cycles. This subprogram allows the experimenter to list the cycles stored on diskette and to create, examine, and delete cycles from the diskette. A cycle is simply a pattern of temporal stimulus (tone or light) events that can be repeated a specified number of times to form a larger pattern of events. When a cycle is edited using "procedure edit" (see Figure 8), the file cycle __nm is accessed. This file contains the names of files that have been created using this particular program disk.

Creating a cycle results in two actions. First, a file with the same name as the cycle being created is stored on the disk. This file contains the number of times a stimulus event is to be activated, the specific stimulus event to be activated, the duration of the activation, and the time period between deactivation of one stimulus event and activation of the next. Second, the cycle name is stored in the cycle __nm file so that the program can identify it.

Listing the cycles simply reads the cycle __nm file from the program disk being used and prints out all the names contained in that file. *Examining* a cycle loads the named cycle, and this cycle is generated via LEDs and/or tones. *Deleting* a cycle removes a specified cycle __nm.

Tracking, Reproduction, and Production. These subroutines can be chosen from the main menu, and values for certain administrative details are required (see Figures 9, 10, and 11 for graphical descriptions of these elements of the program). The following values must be stored for each experiment:

1. Tracking Parameters

Warning Event: The warning can be a tone, an LED, or both, and has a duration from 100 msec to 4,000 msec.

Foreperiod: The foreperiod is an unfilled temporal interval that lasts between 100 msec and 4,000 msec.

Stimulus: This includes the name of the cycle to be used as the stimulus pattern, the number of cycles per trial, and the type of stimulus (tone, LED, or both).

Trials: The number of trials to be used must be specified. For each trial (in all three types of experiments) the selected parameters are repeated.

2. Reproduction Parameters

Warning Event: The warning event length can be either variable or constant and can be a tone, an LED, or both. In the variable condition the computer's random generator selects a number between a programmable range of values (minimum of 100 msec and maximum of 4,000 msec).

Foreperiod: This unfilled temporal interval may be either constant or variable.

Presentation: The program presents a pattern of stimulus events that must later be reproduced by the subject.

Sequence Finished Warning Event: This warning informs subjects that the presented pattern has finished. The length of this warning may be from 100 msec to 4,000 msec and may be a tone, an LED, or both.

Stimulus-Response Interval: This is an unfilled temporal interval with possible values that vary from 0 to 120 sec.

Second Warning Event: This acts as a filled variable or a constant foreperiod. The maximum and minimum range for this event is from 0 to 4,000 msec.

Catch Trials: The catch trial is an elongated second warning event (10 sec). Errors are recorded and stored. The experimenter can choose what percentage of catch trials should

Table 3
4 Bits to 1-of-16 Decoder Wiring List

IC9			
1 unused	9 IC10-5 (#8)	17 unused	
2 IC11-13 (#1)	10 IC10-3 (#9)	18 GND	
3 IC10-9 (#2)	11 IC10-1 (#10)	19 GND	
4 IC10-11 (#3)	12 GND	20 Socket5-13 (bit 3)**	
5 IC10-13 (#4)	13 IC11-11 (Warning, #11)	21 Socket5-14 (bit 2)	
6 IC11-5 (#5)	14 unused	22 Socket5-15 (bit 1)	
7 IC11-3 (#6)	15 unused	23 Socket5-16 (bit 0)	
8 IC11-1 (#7)	16 unused	24 +5 V	

IC10		IC11	
1 IC9-11	8 IC12-5 (#2)	1 IC9-8	8 unused
2 IC14-13 (#10)	9 IC9-3	2 IC13-13 (#7)	9 +5 V
3 IC9-10	10 IC12-13 (#3)	3 IC9-7	10 IC12-12 (#11)
4 IC14-5 (#9)	11 IC9-4	4 IC13-12 (#6)	11 IC9-13
5 IC9-9	12 IC13-6 (#4)	5 IC9-6	12 IC12-6 (#1)
6 IC14-6 (#8)	13 IC9-5	6 IC13-5 (#5)	13 IC9-2
7 GND	14 +5 V	7 GND	14 +5 V

IC12	
1 Tone #3 Out	8 Tone #1 In
2 Tone #3 In	9 Tone #1 Out
3 Tone #2 Out	10 Warning Tone In
4 Tone #2 In	11 Warning Tone Out
5 IC10-8 #2 control	12 IC11-10 Warning control
6 IC11-12 #1 control	13 IC10-10 #3 control
7 GND	14 +12 V

IC13	
1 Tone #7 Out	8 Tone #4 In
2 Tone #7 In	9 Tone #4 Out
3 Tone #5 Out	10 Tone #6 In
4 Tone #5 In	11 Tone #6 Out
5 IC11-6 #5 control	12 IC11-4 #6 control
6 IC10-12 #4 control	13 IC11-2 #7 control
7 GND	14 +12 V

IC14	
1 Tone #10 Out	8 Tone #8 In
2 Tone #10 In	9 Tone #8 Out
3 Tone #9 Out	10 unused
4 Tone #9 In	11 unused
5 IC10-4 #9 control	12 unused
6 IC10-6 #8 control	13 IC10-2 #10 control
7 GND	14 +12 V

Note—IC9 is a 74154 4-line to 16-line decoder. It takes 4 bits from socket 5 and puts 1 of 11 output lines low (the device is active low). These output lines are fed into IC10 and IC11, which are 7406 hex invert buffers/drivers with open-collector high-voltage outputs. These chips convert the low signal into a high (12-V) signal (the outputs are connected to 12 V through a 22K resistor). This high signal is then used to drive the base of an LED driver (refer to Figure 4) and also to close one of the quad bilateral switches.

IC12, IC13, and IC14 are CMOS 4066 quad bilateral switches. Pin 14 is at 12 V because the signals passed by the switches are 12-V signals from the tone generation circuitry. If a control pin has 12 V put to it, then whatever is at the input pin will appear at the output pin. There is no TTL equivalent to the CMOS bilateral switch.

The 11 tones are always present on the input pins of the quad bilateral switches; however, only one of the tones is passed to an output pin.

Figure 4. The LED driver circuit. Instead of the LEDs' being driven directly from the open collector inverter outputs, they are driven through an NPN transistor. Figure 6 shows the circuit. The transistors used were FMPS 6515 and MPS 2222.

Figure 5. The 11 tones are continuously generated by the Tone Generation Circuit. These signals are connected to the Quad Bilateral switches as described in the text. Each tone is generated separately through the use of a variable resistor, a capacitor, and a Uni-Junction Transistor (UJT: GE2N4871).

be included in the trials. The program then randomly selects the location of these catch trials throughout the session. If subjects are in error, a message will be displayed on the screen. Reproduction: After the second warning event, the subject reproduces the pattern, and the pushbutton responses can activate tones, LEDs, or both.

Correct Number of Responses Made: This final event informs the subject that he/she has pressed the correct number of keys and, therefore, should stop the trial. It does not, however, imply that the keys have been pressed in the correct order or for the correct length of time.

3. Production Parameter

Warning Event: This event can be a tone, LED, or both and has time options similar to the previous tracking and reproduction methodologies.

Production: After the warning event has finished, the subject should begin producing the self-initiated pattern.

LABMASTER ISSUES

Use of Counters

The Labmaster counters are used extensively by the SOSMP program. The master clock signal on the Labmaster data acquisition board originates from a slower clock signal ($1000\text{KHz} \pm 1\text{KHz}$). All counters are 16-bit counters and can count, therefore, from 0 to 65,536. The SOSMP program utilizes a clock pulse with a period of $100 \mu\text{sec}$ (0.1 msec), which ensures a design specification of at least 1-msec accuracy.

Counter 1 is used for the length of time that a tone or LED should be activated, and Counter 2 is used for the interval between stimulus offset and stimulus onset. Because of the hardware design in the Labmaster, the fol-

Figure 6. The NPN transistor, biased through BIT4 and BIT5. The transistor used was MPS 2222A and the formula used was given by $(5-0.7)/R \cong 1/10 I_{\text{max}}$.

Figure 7. The general flow diagram describing the SOSMP program.

Figure 8. A flow diagram of the Check Digital Input/Output and the Edit Cycles subprograms.

lowing steps were taken: (1) the load registers were set to 0, (2) the alarm registers were set to 1, and (3) the counters were loaded with the complement of the binary representation of how long they should count. The coun-

ters operate in such a way that they approach the value 65,536 and, having done so, are loaded with whatever is in their respective load registers, incremented by 1, and instructed to stop counting. The value in the alarm register is continuously compared with a comparator. If the values are equal, then the alarm bit is set to 1. Thus, when a counter has counted its count length, the alarm bit will be set to 1 and will be in a steady state. It is this alarm bit, which is checked by the SOSMP program, that determines the completion time of the event.

Counters 3 and 4 are used as running times to time the keypresses. SOSMP commands Labmaster that counter 3 is to be used as a count signal to counter 4. This means that when counter 3 counts from 65,536 to 0, counter 4 will be incremented by 1 (counter 3 was set to run continuously so that it does not stop at 65,536+1). Conceptually, this arrangement can be looked upon as having one large 32-bit counter. These counters are reset to 0 at the start of each trial. Whenever a key is pressed, the binary values of these two counters are stored in 8-bit segments in an array. Byte 1 represents from 0 to 25.5 msec, byte 2 from 25.6 to 6,553.5 msec, byte 3 from 6,553.6 to 1,677,721.5 msec, and byte 4 from 1,677,721.6 to 429,496,729.6 msec. This means that counters 3 and 4 are being used as a 0-h to 119-h timer, accurate to within 100 μ sec.

Figure 9. A flow diagram describing the Tracking option.

Figure 10. A flow diagram describing the Reproduction option.

When a cycle is loaded from disk, its values are given in decimal representation. These decimal representations must be converted to binary complement representations so that counters 1 and 2 can use them. This conversion is accomplished by the *convert ms* procedure and is done while the cycle is being loaded from disk. The array *cyc register* is used to store these values. This is a 100×5 array, and can hold one cycle of 100 "activations," the $[n,1]$ element indicates which of the 1 out of 10 LEDs/tones will be active, the $[n,2]$ element is the most significant bit (msb) for the length of activation, $[n,3]$ is the least significant bit (lsb) for the length of activation, $[n,4]$ is the msb for the deactivation length, and $[n,5]$ is the lsb for the deactivation length. Thus the *cyc register* array contains the binary values of the times used by counters 1 and 2 for presenting cycles.

The procedure *check key* continuously checks the pushbutton switches to see if their state has changed (by reading a PORT). A state change is one of: no keys pressed—a key being pressed; a key being pressed—no key being pressed; a key being pressed—a different key being pressed. If one of these state changes has occurred, then a call is made to either the procedure *store stop* or the procedure *store start*, which store the stop and start times, respectively. The first action that takes place in these procedures is to store counters 3 and 4 into their respective hold registers. These are 16-bit hardware registers contained on the Labmaster itself. Next, the data is transferred to the memory of the PC, where it is stored in an array. This transfer is done in 8-bit segments since the bus of the PC is an 8-bit bus (an IBM AT can do 16-bit

transfers). The array in which these times are stored is called *key pressed* and is a $6,016 \times 9$ byte array. *Keys pressed* $[n,1]$ contains the value of the key pressed (1 ... 10); $[n,2]$... $[n,5]$ contain the start time; $[n,6]$... $[n,9]$ contain the stop time.

Use of Ports

A port on the PC can be thought of as a byte of memory. Thus each port has a unique address and can be written to and read from. The main difference between a port and a byte of memory is that a port can be physically connected to the outside world. The SOSMP circuit board hardware connects the bank of pushbutton switches to one of the PC's ports via the Labmaster. This port is read (similar to the way a byte of memory is read) to determine whether the state of the switches has changed. Similarly, another port connected to the SOSMP circuit board via the Labmaster is used to turn on a certain LED or tone. The SOSMP program writes a certain binary code to this port, this value is read by the SOSMP board, and a specific LED or tone is output. The addresses of ports are given in the constant definitions section of the SOSMP program. For example, the constant *KEY_PORT* is the address of the port that contains the binary code for the state of the pushbutton switches. Port addresses are used

Figure 11. A flow diagram describing the Production option.

to control the Labmaster hardware. Thus writing to specific ports, in effect, is writing to specific registers/memory on Labmaster, which in turn tells the Labmaster what to do.

PROGRAM CONTROL OF THE SOSMP HARDWARE BOARD

Input

As stated above, the SOSMP program sends/receives data to/from the SOSMP circuit board through ports. Channel A of the Labmaster is the port (address 1820) used for sensing the state of the switches. Only the 4 least significant bits are connected to the SOSMP board since 4 bits can represent 16 different states and are, therefore, sufficient to represent 10 pushbutton switches. The remaining 4 bits must be connected to ground. Thus, when the program reads this port, it should see a value between 1 and 10.

Output

Channel C of the Labmaster is the port (address 1822) used for sending control data to the SOSMP board. Each of the 4 least significant bits of this port will contain a number between 0 and 11. A 0 represents the state in which none of the LEDs/tones will be active. A number between 1 and 11 represents one of the LEDs/tones (11 is the warning). The 5th bit indicates that LEDs should be the type of stimulus, and the 6th bit indicates that tones should be used. Thus, to output a certain stimulus, a value is written to this port. For example, to turn on the warning LED and tone, the value $11 + 16(\text{LED}) + 32(\text{tone}) = 59$ must be written to the port. To turn the warning off, a 0 is written to the port.

Error Handling

Turbo Pascal and most other implementations of Pascal do not contain direct error handling control, such as the "onerror goto" instruction found in BASIC. Because of this, error handling must be handled explicitly within the program. All of the procedures starting with "io__," such as io__wrt, perform simple disk I/O tasks but do so with error handling by using (\$I-) and (\$I+). (Refer to a Turbo Pascal manual for more details.)

REFERENCES

- COLLARD, R. F. A., & POVEL, D. J. (1982). Theory of serial pattern production: Tree traversals. *Psychological Review*, **89**, 693-707.
- KEELE, S. W., & SUMMERS, J. J. (1975). The structure of motor programs. In G. E. Stelmach (Ed.), *Motor control: Issues and trends* (pp. 109-142). New York: Academic Press.
- KOLERS, P. A., & BREWSTER, J. M. (1985). Rhythms and responses. *Journal of Experimental Psychology: Human Perception & Performance*, **11**, 150-168.
- LASHLEY, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior* (pp. 000-000). New York: Wiley.
- POVEL, D. J. (1981). Internal representation of simple temporal patterns. *Journal of Experimental Psychology: Human Perception & Performance*, **7**, 3-18.
- RESTLE, F. (1970). Theory of serial pattern learning. *Psychological Review*, **77**, 481-495.
- ROSENBAUM, D. A., HINDORFF, V., & MUNRO, E. M. (1987). Scheduling and programming of rapid finger sequences: Tests and elaborations of the hierarchical editor model. *Journal of Experimental Psychology: Human Perception & Performance*, **13**, 193-204.

(Manuscript received September 7, 1987;
revision accepted for publication January 4, 1988.)