SESSION XI

APPLICATIONS: SOFTWARE

David A. Eckerman, *President*

Going FORTH in the laboratory

JOHN L. ORR

University of Texas Health Science Center, Dallas, Texas

FORTH is a computer language that is useful for writing programs to control experiments. The language is extensible in that new capabilities may be added to FORTH by using FORTH. This article describes a FORTH application word set for animal psychophysics. The words in the application word set facilitate the translation from state diagrams of behavioral psychophysical procedures to experiment-control programs.

FORTH is a high-level computer language that is uniquely suited to the implementation of experiment-control systems. Programs written in FORTH have the potential to be portable because the language is available on several types of computers and because specialized capabilities for experiment control can be added in FORTH. Tutorials on FORTH are available (Brodie, 1981; McCabe, 1983), as are reference books (Haydon, 1983; Scanlon, 1982) and a bibliography (Hofert, 1983).

An important feature of FORTH is that it is extensible in FORTH itself. New functions may be added to perform input/output (I/O) to the experimental apparatus, implement a state interpreter, or record and analyze data. This approach is different from enhancing BASIC (Gordon, Foree, & Eckerman, 1983; Rayfield, 1982; Rayfield & Carney, 1981) or FORTRAN (Hale, Myerson, & Miezin, 1982; Lebovitz & Nicholes, 1981) by adding assembly language subroutines, because FORTH extensions written in FORTH may be used on other processors. Since FORTH-based systems are potentially portable to other processors, the approach is also different from that in systems that compile state tables for a specific processor or processor family (Butler, 1980; Elsner, 1982; Gilbert & Rice, 1978, 1979; Millenson, 1973; Moody, 1972; Snapper & Kadden, 1973; Snapper, Kadden, & Inglis, 1982; Takigawa & Mino, 1981).

The system described here demonstrates the potential for using FORTH to implement an application word set to describe and operate a behavioral procedure for animal psychophysics on a microcomputer. An application word set is the set of FORTH words required to perform a set of experiment-control and data-handling functions. A similar approach can be used to make an application word set for any particular area of research.

This research was supported by Grant RO1ES02750 to Robert Lebovitz. A set of FORTH screens with the application word set and the operating context is available from the author at: Behavioral Sciences Section, Department of Bioengineering, Southwest Research Institute, P.O. Drawer 28510, 6220 Culebra Road, San Antonio, TX 78284.

FORTH

Lea (1982) has summarized some important features of FORTH for experiment-controlling applications: (1) It is a fast compiled language, (2) it is memory-space efficient, (3) it is interpretative and words may be tested from the command mode, (4) assembly code routines are supported, (5) it is a structured language, (6) postfix notation is used, (7) it is extensible, (8) it is powerful, and (9) it can use its own file-handling system.

A FORTH program module is called a FORTH "word." FORTH words use a postfix notation to describe operations to be performed on arguments on a first-in last-out stack and the results to be left on the stack.

FORTH is an incrementally compiled language. As new FORTH words are defined, they are added to the FORTH dictionary. The time to compile a new FORTH word and to place it in the dictionary occurs when each new word is defined. This is in contrast to other languages in which the entire program must be recompiled and linked before it may be tested. During the process of compilation, FORTH threads together routines and places the compiled word in the dictionary. At run time, the FORTH interpreter executes the FORTH words after looking them up in the FORTH dictionary.

In FORTH, the parameter stack is frequently used to pass arguments from one program module to another. This approach facilitates the interactive debugging available in FORTH. FORTH words are tested by setting up the stack with the arguments for the word, executing the word by typing its name, and checking the outputs and status of the stack. There is no need to load a debugging program or a symbol table to test FORTH words.

AVAILABILITY

The September 1983 issue of *Dr. Dobb's Journal* (ISSN 0278-6508) carried ads from more than a dozen

vendors of FORTH implementations for a wide range of microcomputers. Versions were advertised to run on 8080, Z-80, 8086, 8088, 6502, and 68000 microprocessors. FORTH as implemented for specific machines was advertised for Apple, Atari, HP-85, HP-75, Heath, IBM PC, Osborne, Kaypro, Northstar, TRS-80, Timex, Vic-20, and the Victor 9000. In addition, there were three advertisements for small microcomputers that include FORTH as the resident language.

STATE NOTATION AND STATE TABLES

The use of state notation to describe behavioral procedures (Snapper & Kadden, 1973; Snapper, Knapp, & Kushner, 1970; Snapper et al., 1982) is valuable independently of the particular method used to implement the state tables. State diagrams and state tables are the descriptions and definitions of behavioral procedures. A state diagram is a graph that shows a set of "states of the environment," usually represented as circles, and the requirements for moving from one state to another. The movement from one state to another is a transition. A state set is a set of states connected by transition rules wherein only one state may be active at a time. A state table is the computer program used to implement the state diagram.

The system described here is a small state table system. In this system, when a state set is scanned, the state description word corresponding to current state in the state set is executed. If a transition is generated to a new state, the GOTO word sets up the new state as the current one for the next scan. The state sets are scanned when a signal from an external clock is detected at an input port. Once a scan of the state sets begins, the current state in each state set is executed and any transitions are processed. After all state sets have been scanned, the system waits for the next clock pulse to repeat the cycle. The program described here is written in MVP-FORTH (Haydon, 1983) and runs on an S-100 buss 4-MHz Z-80 microcomputer using the CP/M operating system.

To illustrate the potential of a FORTH approach to experiment control, a series of state diagrams of increasing complexity and the corresponding state tables using the application word set are given in Figures 1-8. The state diagrams are simplified to emphasize program flow. The state words are the definition of the experimental procedure. In these examples, only the "user" words are discussed. To function, these words must be used in a context in which the necessary "internal" words have been defined and the appropriate constants loaded. A listing that has the application word set, the necessary internal words, and the data structures as used on the microcomputer system described above is available from the author.

EXAMPLE 1

The state diagram in Figure 1 describes a simple pro-

Figure 1. State diagram description of a system in which depressing a response lever causes an output to be activated for 1 sec.

```
0 : SOSS0 KEYDWN? IF CTRLITE ON 1 GOTO THEN ;
1 : S1SS0 CLK? IF CTRLITE OFF 0 GOTO THEN ;
```

Figure 2. State table written in FORTH using the experimental application words described in the text to implement the state diagram in Figure 1.

cedure. A depression of the response lever causes the output device to turn on for 1 sec. This is an example of a one-shot, or a nonretriggerable monostable multivibrator.

Figure 2 shows the experimental FORTH words and syntax necessary to implement the simplified state diagram of Figure 1. In this example, two new FORTH words are defined, SOSS0 and S1SS0, which are State 0 and State 1 of State Set 0. Each FORTH word definition begins with a colon and is followed immediately by the name of the word being defined. A FORTH word definition ends with a semicolon. In this system, the name of the FORTH word for a state describes the state and the state set to which it belongs. For example, the name SOSS0 stands for State 0 of State Set 0.

In SOSS0 (Figure 2), the word KEYDWN? tests whether the appropriate input is provided and returns a flag on the stack. The IF clause then takes the appropriate action based on the value of the flag. If the key is down, the center light (CTRLITE) is turned on, the number of the new state, State 1 (the state to be executed on the next scan), is placed on the stack, and the GOTO word is executed. The GOTO word is the key to this state interpreter. The GOTO word updates the current state pointer for the current state set and loads the software clock with the previously defined clock constant for the next state, and then returns to await the next scan. On the next scan, the S1SS0 word is executed and the software clock is decremented and checked for having reached zero by the word CLK?, which leaves a flag on the stack for the IF clause. If the clock has not timed out, no change is made and control is returned to the state interpreter to await the next scan. If the clock has timed out, the light is turned off and the GOTO word is used to set up State 0 as the current state for the next scan.

EXAMPLE 2

Figure 3 shows the same function as in Figure 1, but it is implemented using two concurrent state sets to illustrate the process of synchronizing multiple state sets. The synchronization of parallel state sets is accom-

Figure 3. A semaphore is used to synchronize concurrent state sets to perform the same function as the state table in Figure 1.

```

0 : S0SS0 KEYDWN? IF ZONE! 0 GOTO THEN ;
1 : S0SS1 ZONE? IF ZONECL CTRLITE ON 1 GOTO THEN ;
2 : S1SS1 CLK? IF CTRLITE OFF 0 GOTO THEN ;

```

Figure 4. State table written in FORTH using the experimental application words for the state diagram in Figure 3.

plished using semaphores for communication. Semaphores are flag systems that are outputs in one state set and inputs in another. The semaphores perform the same function as Z pulses in SUPERSKED (Snapper et al., 1982) and "synchs" in PASTOR (Elsner, 1982). To emphasize the similarity, semaphores have names starting with a Z in this FORTH system. A semaphore is a variable that is set to a particular value when issued as an output and tested and reset, if necessary, when used as an input. The use of semaphores to synchronize concurrent processes is a way to call a routine that is executed in parallel with the routine that called it.

The state sets defined in Figure 4 perform the same function as the state set in Figure 2. In S0SS0, if the response lever is depressed, the semaphore ZONE is generated as an output. The word ZONE! stores a value of 1 in the variable ZONE. State Set 1 generates the 1-sec output. In S0SS1, the variable ZONE is checked by the word ZONE? and a flag is left on the stack. If ZONE is nonzero, it is cleared to zero, the light is turned on, and State 1 is set up with the GOTO word as the current state for the next scan. After the output is turned on, the system will be in S1SS0 waiting for the clock to time out. When there have been enough scans to count the software clock down to 0, the output is turned off and State 0 is made the current state for State Set 1.

EXAMPLE 3

Figure 5 illustrates the use of semaphores in a more complex application of concurrent state sets. In this example, the states of State Set 1 are used to implement a software astable multivibrator or pulse train generator. The onset and offset of the pulses are controlled by

semaphores generated in State Set 0. This software astable multivibrator always starts with a full-width output pulse. In Figure 5, once the response lever is depressed and the semaphore ZONE is set, State Set 1 oscillates between States 1 and 2, pulsing the output until the lever is released and the semaphore ZTWO puts State Set 1 into State 0 with the output off.

Figure 6 is the state table corresponding to the simplified state diagram shown in Figure 5. In State Set 1 in Figure 6, in State 1 and State 2, there are two potential sources of a transition, either a semaphore or the timing out of the software clock. The semaphore is checked for first, and if it is set, the word EXIT is executed after the GOTO word sets up for the next scan. EXIT causes the second IF clause, which checks the software clock to be skipped. The order of testing the conditions in the state determines the priority of different potential sources of transitions. In this case, the semaphore is the most important and comes first in the table. The number of potential transition sources is determined by the number of conditions tested in the word for a state; the priority is determined by the order in which the conditions are tested.

ANIMAL PSYCHOPHYSICS PROCEDURE

Figure 7 is a simplified state diagram for a system that implements a single-response-lever press-release behavioral procedure for auditory noise burst pulse rate discrimination. State Set 0 controls the overall flow of

Figure 5. Semaphores generated as outputs in SS0 are used as inputs in SS1 to control the operation of a software astable multivibrator that produces a stream of 1-sec output pulses followed by a 1-sec off interval (output pulses are generated starting when the lever is depressed and cease when it is released).

```

0 : S0SS0 KEYDWN? IF ZONE! 1 GOTO THEN ;
1 : S1SS0 KEYUP? IF ZTWO! 0 GOTO THEN ;
2 : S0SS1 ZONE? IF ZONECL CTRLITE ON 1 GOTO THEN ;
3 : S1SS1 ZTWO? IF ZTWOCL CTRLITE OFF 0 GOTO EXIT THEN ;
4 : CLK? IF CTRLITE OFF 2 GOTO THEN ;
5 : S2SS1 ZTWO? IF ZTWOCL 0 GOTO EXIT THEN ;
6 : CLK? IF CTRLITE ON 1 GOTO THEN ;

```

Figure 6. State table in FORTH using the experimental extension words to implement the state diagram of Figure 5.

Figure 7. State diagram of three concurrent processes used to implement a single-response-lever auditory pulse rate discrimination task. (In SS0, S and F refer to the slow and fast rates of pulsation of wide-band noise). States 0, 1, 2, and 6 require the subject to release the lever and not depress it for at least 3 sec for a trial to begin. The trial begins with the onset of slow noise pulsing when State 2 is entered. The trial proceeds when the subject depresses the response lever and causes the transition to State 3. State 3 controls the variable foreperiod and the selection of a signal or nonsignal trial. State 4 is the blank trial state, in which the noise pulse rate does not change. State 5 is the signal trial state, in which the noise pulses faster. After the 2-4-sec foreperiod is completed, a nonsignal trial occurs with $p(1 - \text{PRIOR})$, where PRIOR is a variable containing the prior odds of signal occurrence. If State 4, the nonsignal trial, is selected, the correct response is to continue to depress the key for 2 more seconds and receive a food pellet. Release of the lever during a blank trial is a false alarm and is not followed by food. If a signal trial (State 5) occurs, release of the lever is the correct response and is followed by a food pellet. SS1 is used to generate the stream of noise pulses. SS1 is similar to the example in Figure 5, with the addition of programmable on and off times. The semaphores that control SS1 from SS0 are generated by the FORTH words controlling the rate of pulsation and the end of a trial. SS2 is used to record a histogram of response durations. This data-recording state set is "armed" when a transition occurs from State 2 to State 3 in SS0, and a response duration is recorded in a histogram by the word HIST if a pretrial release or a false alarm occurs. Hits and posttrial releases are kept out of the response duration histogram by the RSTHST word, which "disarms" the data-recording mechanism if a signal trial or a correct rejection occurs.

the procedure. State Set 1 is used to generate the pulses and is similar to State Set 1 in Figure 5, except that the duty cycle of the pulsing is controlled by using variables to store the on and off times. By loading the variables with different values for the number of scans required to cause a time-requirement transition, the state set can control pulse generation at different rates. State Set 2 is used to record response durations.

Figure 8 is the state table corresponding to Figure 7. This is an example of the coding required to implement an application in an extended FORTH environment. The state table for State Set 0 includes the data recording words CNT, HIST, and RSTHST. The word CNT increments the counter whose number is on the stack. HIST increments the bin in a histogram (specified on the stack) corresponding to the time since the state was

```

SCR #66
0 ( INTERTRIAL INTERVAL )
1 : S0SS0
2 KEYUP? IF 1 CNT RSTHST FOREPERIOD 3 CLKCON 12 CNT
3 HIST 1 GOTO EXIT THEN
4 CLK? IF ." S0SS0 " RLDCLK THEN ;
5 : S1SS0
6 KEYDOWN? IF 13 CNT 6 HIST 6 GOTO EXIT THEN
7 CLK? IF SPULSE 2 GOTO THEN ;
8 : S6SS0
9 KEYUP? IF 14 CNT 7 HIST 1 GOTO EXIT THEN
10 CLK? IF ." S6SS0 " RLDCLK THEN ;
SCR #67
0 ( FOREPERIOD AND TRIAL SELECTION )
1 : S2SS0 KEYDOWN? IF 2 CNT 0 HIST 3 GOTO EXIT THEN
2 CLK? IF ." S2SS0 " RLDCLK THEN ;
3 : S3SS0 KEYUP? IF 3 CNT 1 HIST NOPULSE 1 GOTO EXIT THEN
4 CLK? IF PRIOR 0 WITHP IF FPULSE RSTHST 4 CNT 4 GOTO ELSE
5 5 CNT 5 GOTO THEN THEN ;
SCR #68
0 ( SIGNAL AND BLANK TRIAL STATES )
1 : S4SS0
2 KEYUP? IF 6 CNT NOPULSE 0 CNT FOOD BEEP 2 HIST 0 GOTO EXIT
3 THEN CLK? IF 7 CNT NOPULSE 0 GOTO THEN ;
4 : S5SS0
5 KEYUP? IF 8 CNT NOPULSE 4 HIST 0 GOTO EXIT THEN
6 CLK? IF 9 CNT NOPULSE 0 CNT FOOD BEEP 0 GOTO THEN ;
SCR #73
0 ( FLASHER CONTROLLER )
1 : ZONE? ZONE @ ; : ZTW0? ZTW0 @ ;
2 : ZONECL 0 ZONE ' ; : ZTW0CL 0 ZTW0 ! ;
3 : S0SS1
4 ZONE? IF ZONECL NO-NOISE ON 1 GOTO THEN ;
5 : S1SS1
6 ZTW0? IF ZTW0CL 0 GOTO EXIT THEN
7 CLK? IF NO-NOISE OFF 2 GOTO THEN ;
8 : S2SS1
9 ZTW0? IF ZTW0CL NO-NOISE ON 0 GOTO EXIT THEN
10 CLK? IF NO-NOISE ON 1 GOTO THEN ;
SCR #76
0 ( BROKEN HOLDS AND FALSE ALARMS IRT HISTOGRAMMER )
1 : ZTHREE? ZTHREE @ ; : ZTHREELCL 0 ZTHREE ! ;
2 : S0SS2 KEYDOWN? IF CSNA @ 3 = IF 1 GOTO EXIT THEN THEN ;
3 : S1SS2 CLK? IF ." S1SS2 " THEN
4 ZTHREE? IF ZTHREELCL 0 GOTO EXIT THEN
5 KEYUP? IF 5 HIST 10 CNT 0 GOTO EXIT THEN ;

```

Figure 8. State table in FORTH using the experimental extension words to implement the state diagram in Figure 7.

entered. HIST is used to produce reaction time distributions. The word RSTHST is used to reset the data-recording State Set 2 so that only the desired response durations are recorded.

The words SPULSE and FPULSE in State Set 0 control the onset and rate of pulsation produced by State Set 1 by loading the appropriate variables and using a semaphore to start the pulsation. NOPULSE sets a semaphore to stop the pulsation.

The word FOREPERIOD constructs the duration of the foreperiod with a fixed and random component and stores it in the clock constant for State 3. A clock constant is the value loaded into the software clock for the current state of a state set by the GOTO word. When a transition is made to State 3, it is the variable foreperiod that is loaded into the software clock for this state by the word GOTO.

State Set 2 is used to measure response durations. A depression of the response lever, which puts State Set 0 into State 3, arms the data-recording State Set 2 by putting it in State 2. If the lever is released before a RSTHST is executed, the time since lever depression is stored by the HIST word. If there is a signal trial or the lever is held down through a blank trial, the RSTHST word disarms the recorder and no response duration is recorded. Thus, the histogram contains the response durations that end with a release of the lever in the presence of the slow pulse rate, during the foreperiod and blank trials.

DISCUSSION

Since FORTH programs are constructed by building upon previously defined words, there is a definite order to FORTH programs. There are low-level words that are not intended to be used in programming an application. The high-level functions, the application words, are built from lower level words. In FORTH, as in other languages, the convenience of using high-level functions is obtained at the cost of writing the appropriate words.

The resources of pseudoparallel processing software may be allocated as the experiment demands. The small system described here is able to handle three concurrent state sets of modest complexity, as shown in Figure 8, with a scan rate of 100 Hz. The three concurrent state sets illustrate the use of pseudoparallel processing capability to control the experimental sequence, stimulus generation equipment, and data recording. With this same hardware and an appropriate FORTH application word set, an experimenter could make a state table to test different subjects simultaneously on an operant reinforcement schedule, by using a state set to control the contingencies for each subject.

The FORTH system described here provides a way to achieve some of the benefits of a state notation/state table system on any microprocessor with FORTH.

REFERENCES

- BRODIE, L. (1981). *Starting FORTH*. Englewood Cliffs, NJ: Prentice-Hall.
- BUTLER, F. E. (1980). MicroSKED. *Behavior Research Methods & Instrumentation*, 12, 152-154.
- WIGGINS, R. (Ed.) (1983). Annual FORTH Issue. *Dr. Dobbs Journal*, 8(9).
- ELSNER, J. (1982). PASTOR: A new schedule programming language. *Behavior Research Methods & Instrumentation*, 14, 254-263.
- GILBERT, S. G., & RICE, D. C. (1978). NOVA SKED: A behavioral notation system for Data General minicomputers. *Behavior Research Methods & Instrumentation*, 10, 705-709.
- GILBERT, S. G., & RICE, D. C. (1979). NOVA SKED II: A behavioral notation language using the Data General Corporation real-time disk operating system. *Behavior Research Methods & Instrumentation*, 11, 71-73.
- GORDON, W. A., FOREE, D., & ECKERMAN, D. A. (1983). Using the Apple II microcomputer for real-time control in a behavioral laboratory. *Behavior Research Methods & Instrumentation*, 15, 158-166.
- HALE, S. S., MYERSON, J., & MIEZIN, F. M. (1982). Top-down design for a system to control operant choice experiments. *Behavior Research Methods & Instrumentation*, 14, 272-275.
- HAYDON, G. B. (1983). *All about FORTH* (2nd ed.). Mountain View, CA: Mountain View Press.
- HOFERT, D. K. (Ed.) (1983). *A bibliography of FORTH references*. Rochester, NY: Institute for Applied Forth Research.
- LEA, D. (1982). Using the FORTH language in real-time computer applications. *Behavior Research Methods & Instrumentation*, 14, 29-31.
- LEBOVITZ, R. M., & NICHOLS, N. A. (1981). Computer-based operant behavioral system for microwave bioeffects research.

- IEEE Transactions on Biomedical Engineering*, **28**, 536-543.
- McCABE, C. K. (1983). *FORTH fundamentals*. Beaverton, OR: Dilithium Press.
- MILLENSON, J. R. (1973). On-line sequential control of experiments by an automated contingency translator. In B. Weiss (Ed.), *Digital computers in the behavioral laboratory*. New York: Appleton-Century-Crofts.
- MOODY, D. B. (1972). SNAP: A computer language for control of psychoacoustic experiments. *Computers and Biomedical Research*, **5**, 673-683.
- RAYFIELD, F. (1982). Experimental control and data acquisition with BASIC in the Apple computer. *Behavior Research Methods & Instrumentation*, **14**, 409-411.
- RAYFIELD, F., & CARNEY, J. (1981). Controlling behavior experiments with BASIC on 6502-based microcomputers. *Behavior Research Methods & Instrumentation*, **13**, 735-740.
- SCANLON, L. J. (1982). *FORTH programming*. Indianapolis: Sams.
- SNAPPER, A. G., & KADDEN, R. M. (1973). Time-sharing in a small computer based on a behavioral notation system. In B. Weiss (Ed.), *Digital computers in the behavioral laboratory*. New York: Appleton-Century-Crofts.
- SNAPPER, A. G., KADDEN, R. M., & INGLIS, G. B. (1982). State notation of behavioral procedures. *Behavior Research Methods & Instrumentation*, **14**, 329-342.
- SNAPPER, A. G., KNAPP, J. Z., & KUSHNER, H. K. (1970). Mathematical description of schedules of reinforcement. In W. N. Schoenfeld (Ed.), *The theory of reinforcement schedules*. New York: Appleton-Century-Crofts.
- TAKIGAWA, T., & MINO, T. (1981). TYMES: A high-level language for process control and data manipulation in the behavior laboratory. *Behavior Research Methods & Instrumentation*, **13**, 741-746.