*Pharmacology Biochemistry & Behavior*, Vol. 17, pp. 583–587, 1982. Printed in the U.S.A.

# A TRS-80-Based System for the Control of Behavioral Experiments[1]

M. W. EMMETT-OGLESBY, D. G. SPENCER, JR. AND D. E. ARNOULT

*Department of Pharmacology, Texas College of Osteopathic Medicine, Fort Worth, TX 76107*

Received 11 February 1982

EMMETT-OGLESBY, M. W., D. G. SPENCER AND D. E. ARNOULT. *A TRS-80-based system for the control of behavioral experiments*. PHARMAC. BIOCHEM. BEHAV. 17(3) 583–587, 1982.—A TRS-80-based system is described for controlling and recording events from operant chambers. The system features a TRS-80 microprocessor with 32 K RAM, a floppy disk drive, an LVB optical interface, a printer, and a user-oriented program (Operant Protocol System, OPS). The LVB interface system is capable of up to 128 independent input/output operations for each of up to 8 boxes. The system uses assembly language routines for real-time control and event recording of up to 8 boxes simultaneously. BASIC is used to obtain user input and deliver data output in an interactive fasion. Advantages include dependable hardware, extensively documented and tested software, relative inexpensiveness, and standard-deviation-based variable schedules.

TRS-80 microcomputer Operant research Behavioral pharmacology Assembly language BASIC Behavior

THE increasing power and flexibility of microcomputers, coupled with their rapidly declining costs, has made them a progressively more attractive alternative to modular electromechanical and minicomputer control systems for behavioral pharmacology laboratories. Although inexpensive, microprocessors are flexible, allowing the user to perform several types of experiments; indeed, they can be adapted to uses such as data analysis or word-processing when not engaged in laboratory functions. Finally, they permit the acquisition and analysis of data which are beyond the capacity of modular equipment. However, microcomputers have limitations; most notably, they must be programmed. A program designed to execute behavioral experiments should be as general purpose as possible and feature easily understood input and output instructions. Moreover, the program should execute rapidly enough to insure the detection of every response made in the operant chambers. This latter constraint necessitates programming in assembly language, which is difficult and tedious. Software considerations aside, microcomputers are in general poorly equipped to directly control external equipment; some type of interfacing device is necessary. An interface should be selected that is compatable with the particular microprocessor in use, can handle real-time jobs such as switch debouncing of inputs and pulse duration setting for outputs, and can conduct a large number of input/output operations.

To date, microcomputer control systems have been reported in which a mixture of higher level languages, such as BASIC, and assembly language are used for real-time experimental control [1, 2, 4, 5]. The use of higher level languages not only slows program execution, it also makes it very difficult to calculate program execution time. Thus, no estimates are provided of the time it takes these programs to execute all functions under the worst possible case, thereby making estimates of the maximum number of operant chambers that can be controlled by a single system difficult. The Operant Protocol System (OPS) described in this paper was designed to overcome these difficulties in two ways: (1) OPS is specifically designed to process up to 8 chambers, and (2) real-time experimental control and data recording is done solely in assembly language. TRS-80 provides precise times of execution for all assembly language commands, and these data have been used to calculate OPS running time.

Although OPS uses assembly language for real-time execution, BASIC has been used to write programs that interact with the experimenter before and after the experimental session in order to receive the experimental parameters (session time, operant schedule, subject number, etc.) and to deliver the session data, respectively. These programs are interactive and require no programming knowledge to perform experiments with OPS. BASIC and assembly-language programming ability is necessary only if modification of the OPS software is desired, and for this purpose, BASIC and assembly language routines have been extensively documented.

## *Hardware*

A TRS-80 Model III microcomputer (Radio Shack) was selected because extensive software is currently available, thereby reducing time spent in user program development. Minimum requirements are 32 K random access memory

[1]Supported in part by ADAMHA Grant 1 RO 3 MH34607 and Faculty Research Grant 34940.

(RAM) and one floppy disk drive. Using TRS-80 bus extensions, the computer's input and output ports are connected to an optically isolated LVB interface (Med Associates, Inc.) which safely separates the low-voltage (5 V DC) computer line from the higher voltage (28 V DC) working lines. The LVB interface comes as a chassis which can hold up to 6 cards. Each slot can hold either an input or an output card, and each card either inputs or outputs 1 byte (8 bits, 1 bit per box) of data simultaneously. OPS uses a card for each different type of event in up to 8 chambers, eg., one input card monitors all left levers in the 8 operant chambers; similarly, one output card controls all houselights in the 8 chambers. The LVB chassis can be connected in series via bus extenders and because the TRS-80 uses 7 bits of an 8-bit byte to address these cards, up to 128 cards could be used. On the input side, the LVB interface pulse-forms (eliminates clatter) switch closures and holds all input data in a buffer which is cleared on reading. On the output side, the LVB can be set to act as a transparent interface (output line high or low until the logic state is modified by the processor) or a controlling interface (output duration controlled by an adjustable one-shot). These features permit maximum flexibility in output configuration to solenoid devices, lights and speakers. If hard copy of data is desired, the TRS-80 can also be connected to a printer via a parallel interface port, and OPS will control data printing. In order to run 8 boxes, the present cost of microcomputer with 32K RAM, 1 disk drive, LVB interface (with 6 cards), inexpensive printer and OPS software is approximately $4,500 or less. Since each additional LVB card costs an average of $200, independent control of additional input or output devices is quite economical, assuming the user has the capability to program in BASIC and assembly language.

### *Software*

OPS is an interactive user-oriented package that requires no programming knowledge to execute the operant schedules described below. Because the TRS-80 has an 8-bit word size, OPS is designed to process data from all 8 boxes in parallel, where each bit in a byte represents one box. Thus, lever responses are read and output is updated for all 8 boxes simultaneously. The OPS software currently allows for up to 3 manipulanda inputs; outputs for houselights, feeders and panel lights; and the following operant schedules: fixed and variable ratio (FR and VR), fixed and variable interval (FI and VI), fixed and variable time (i.e., response-independent delivery of reinforcement; FT and VT), differential reinforcement of low and high rates (DRL and DRH), and extinction (EXT). In addition, a drug discrimination testing procedure is available, in which two levers are on a concurrent FR 10 schedule. Any of up to three manipulanda can be defined as correct (reinforceable) for any of the simple schedules.

OPS software consists of BASIC and assembly language routines constructed using technqiues of structured programming [6]. This method emphasizes subroutine-oriented, task-specific programming, resulting in modular program segments for users having assembly language programming skills and wishing to modify OPS. For this purpose, BASIC and assembly language routines have been extensively documented. All BASIC and assembly language programs are loaded into RAM from disk, providing minimal delay in program initiation. The program flow during operation is assigned to three components: user input, real-time processing, and data output. These components are executed sequentially after an initial BASIC load and run command. User input and data output are performed entirely through BASIC routines, and real-time execution is performed solely through assembly language routines. The flow of control between BASIC and assembly language routines is schematized in Fig. 1.

User input is initiated with disk loading and running of a BASIC program which prompts the user to supply the information necessary to perform the experiment. As shown by the prompts in Fig. 2, eight boxes can be run under the same or different schedules. Even when data must be entered for each box individually, typical set-up time is not more than 3 minutes. To facilitate input further, we are developing a system in which user-input data can be stored to disk; thus, all future experiments employing the same parameters could be loaded in a few seconds.

Upon receiving the experimental parameters, the BASIC program inserts the parameters into their appropriate RAM locations, loads the assembly language real-time processing component, and then erases itself from memory (providing 3K additional RAM for data storage). The real-time processing component consists of two assembly language programs which are called from BASIC. The first program runs a test routine which prompts the user to test all response manipulanda and terminates itself upon delivering reinforcement to all chambers. The counters are then reinitialized and the user is prompted to start the session by pressing "S." Upon "S" input, the session is started, a chamber-monitoring display is set up on the CRT and the main assembly language routine which runs the actual experimental session is called. This program uses the internal TRS-80 30 Hz clock to initiate execution every 100 msec. On each execution, inputs are polled from all chambers, outputs delivered, and data processed in 15.5 msec at most. This worst case execution figure was calculated for the following conditions: 8 chambers on a VI schedule; all chambers generate a response on this execution, all chambers receive reinforcement, schedule values are recalculated for all chambers, data are processed and stored, and clocks are updated. Therefore, although the program currently uses a 100 msec interrupt, it could be modified to make full use of the 30 Hz clock and interrupt every 33.3 msec even with all 8 chambers running concurrently. This modification would make OPS easily compatible with research subjects such as pigeons, which are capable of responding in excess of ten times per second. This laboratory has been using OPS to run two sets of 8 operant chambers on drug discrimination paradigms since December, 1981. Although the majority of our experience with OPS has been on FR schedules, all operant schedules have been fully tested and debugged.

An attractive feature of OPS is the capability of user-set standard deviations for variable ratio, interval, and time schedules. During user-supplied parameter inputs, mean schedule values and standard deviations are obtained. During the experimental session, an assembly language subroutine multiples this standard deviation by a Z-table frequency distribution to obtain random numbers used to offset the mean schedule value. For example, if a user specified a variable interval 20-second schedule with a standard deviation of 5, two-thirds of all the schedule values would fall between 15 and 25 seconds in half-second bins. Although a table look up is used to calculate the offset for the mean schedule value, the decision to add or subtract this offset is

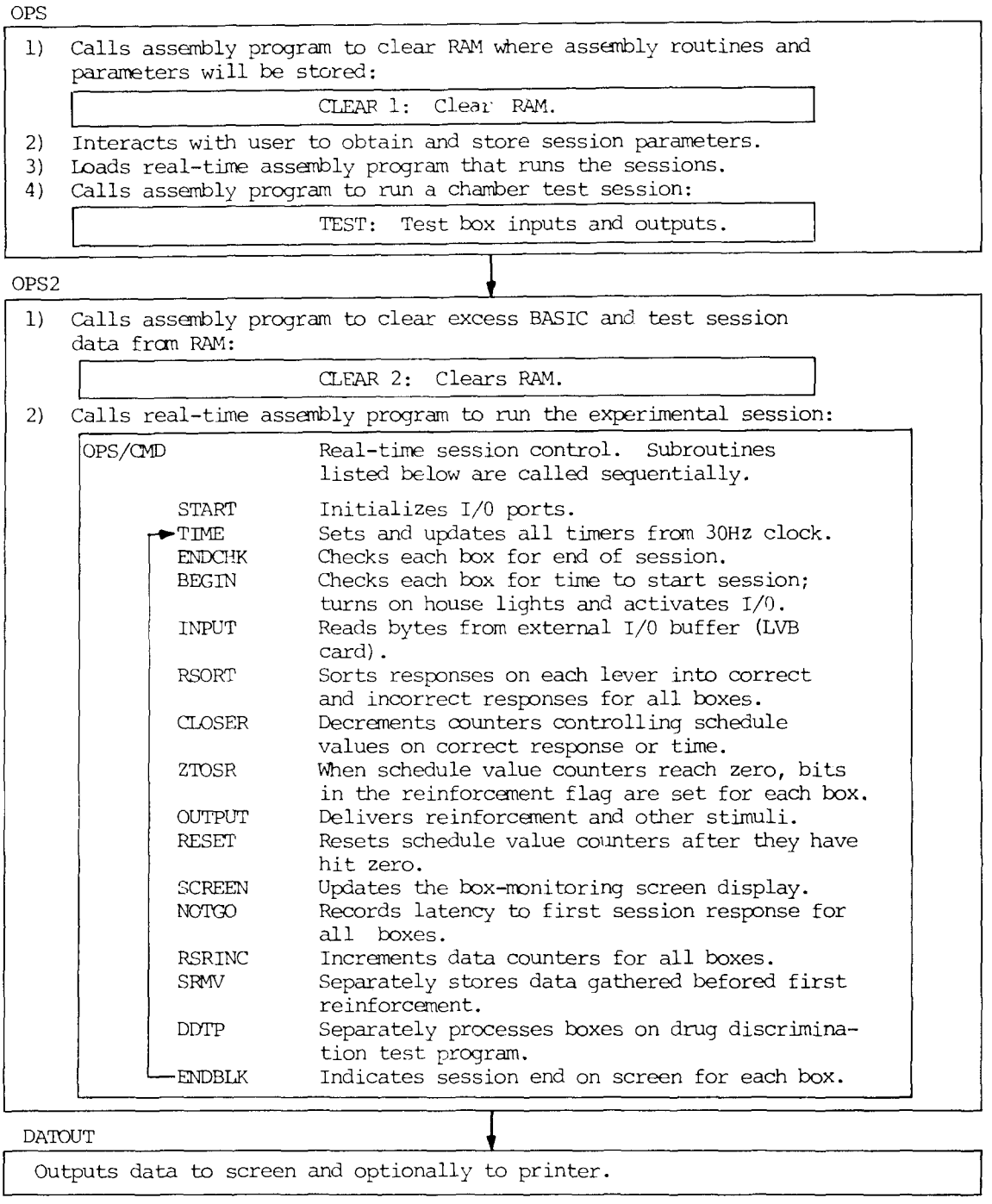


FIG. 1. Program descriptions and flow of control for experimental set-up and execution. BASIC program and sequential functions are schematized in the outer boxes and assembly programs they call are shown in the inner boxes. Once an assembly routine is entered, the computer does not return to BASIC until the assembly execution is completed. When the user wishes to run a session, OPS is loaded, and the flow of program-to-program transfer of control is carried out automatically until the end of DATOUT is reached. If the user answers "Y" to the query, "Do you wish to run more animals?" OPS is automatically reloaded and new session parameters can be entered.

```
TURN ON LVB
    (interface is turned on by user)
ENTER NAME IN 16 CHARACTERS OR LESS.
    DAVID SPENCER
ENTER BRIEF DESCRIPTION OF EXPERIMENT, NO MORE THAN ONE LINE.
    ADENOSINE DRUG DISCRIMINATION
IS THE SESSION TIME THE SAME FOR ALL BOXES (Y OR N)?
    N
DO YOU WANT THE BOXES TO START SIMULTANEOUSLY (Y OR N)?
    N
DO YOU WANT THE DELAY BETWEEN STARTS TO BE OTHER THAN 10 SEC (Y OR N)?
    Y
DELAY BETWEEN STARTS (IN SEC) = ?
    15
ENTER THE NUMBER OF BOXES TO BE RUN (1-8)?
    8
ARE THE ANIMALS ON THE SAME PROGRAM (Y OR N)?
    N
PROGRAMS ARE:  FR=1, FI=2, FT=3, VR=4, VI=5, VT=6, DRL=7, DRH=8, EXT=9, DDT=10.
THE SCHEDULE FOR BOX 1 = ?
    1
RAT IN BOX 1 IS NUMBER?
    23
FOR BOX 1, THE SESSION TIME IN MIN = ?
    10
INDICATE CORRECT LEVER(S):  LEFT, CENTER, RIGHT (Y OR N) IN BOX 1?
    Y,N,N
RESPONSE RATIO IN BOX 1 = ?
    10
SHOULD THE SESSION END UPON NUMBER OF REINFORCEMENTS IN BOX 1 (Y OR N)?
    Y
REINFORCEMENT NUMBER, BOX 1?
    50
PANEL LIGHTS (Y OR N)?
    N
PROGRAMS ARE:  FR=1, FI=2, FT=3, VR=4, VI=5, VT=6, DRL=7, DRH=8, EXT=9, DDT=10.
THE SCHEDULE FOR BOX 2 = ?
    4
RAT IN BOX 2 IS NUMBER?
    49
    .
    .
    .
```

FIG. 2. Sample interaction between user and OPS in the BASIC experimental information input stage. For clarity, user replies to prompts are indented; computer prompts are not. As shown above; schedules, schedule values, and session durations can be declared individually or collectively by the user, but subject numbers and correct/incorrect levers are always specified box-by-box. All input parameters are free to vary between boxes.

determined by the odd or even status of the 30 Hz clock, an essentially random event with respect to the time at which a response is received. Thus, even if only one box is run on a variable schedule, a different pattern of intervals or ratios is obtained each session. Execution time for this segment of the program is less than on other systems since assembly rather than higher level languages is used for standard deviation calculations. Actual time to recalculate 8 VI-20-second schedules and store them in appropriate RAM locations is 4.8 msec.

When the session has terminated for all boxes, the assembly language program returns to the calling BASIC program which loads a BASIC data output program from disk. The data output program displays response and reinforcement data to the CRT and controls printing. This routine also includes options for obtaining a measure of the latency to first response from the start of the session and a summary of responses on correct and incorrect levers until the first reinforcement was obtained. The latter feature is particularly useful for drug discrimination experiments, in which only one of two manipulanda will generate reinforcement and the correct manipulandum is determined by the injection conditions [3]. Finally, subroutines allowing for breakdowns of inter-response times, post-reinforcement pause times, post-reinforcement response frequencies, and temporal analyses of response and reinforcement data are being developed. The

last feature should be useful for monitoring the time course of psychoactive drug effects.

In summary, OPS includes the following features: An inexpensive, reliable hardware system and software package to control up to 8 operant chambers; the ability to program all chambers for different reinforcement schedules, starting times, and ending times; the capability of specifying the range of values to be used in variable schedules; and extensive documentation of all BASIC and assembly language subroutines to facilitate further adaption to individual needs. For more information, write to the first author.

## REFERENCES

1. Carroll, M. E., P. A. Santi and R. L. Rudell. A microcomputer system for the control of behavioral experiments. *Pharmac. Biochem. Behav.* **14:** 415–417, 1981.
2. Dillon, R. F., B. Millman, J. W. Tombough, W. R. Ferguson and W. R. Bezanson. A microcomputer-controlled laboratory: Hardware. *Behav. Res. Meth. Instrum.* **11:** 293–300, 1979.
3. Lal, H. and G. T. Shearman. Interoceptive discriminative stimuli in the development of CNS drugs and a case of an animal model of anxiety. *An. Rep. Mednl Chem.* **15:** 51–58, 1980.
4. Thompson, G. C. Behavioral programming with the APPLE II computer. *Behav. Res. Meth. Insturm.* **11:** 585–588, 1979.
5. Tombough, J. W., R. F. Dillon, B. Millman and W. R. Bezanson. A microcomputer-controlled laboratory: Software. *Behav. Res. Meth. Insturm.* **11:** 301–310, 1979.
6. Yourdan, E. *Technqiues of Program Structure and Design.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.