Behavior Research Methods & Instrumentation
1982, Vol. 14 (2), 241-249

Some simple Apple II software for the collection and analysis of observational data

JOHN H. FLOWERS
University of Nebraska, Lincoln, Nebraska 68588

Two general-purpose software packages for collecting and analyzing observational data from a variety of settings are discussed. One package is designed for coding mutually exclusive behavioral states using the Apple's keyboard as an input device. The other is designed to monitor temporally overlapping behaviors, and it makes use of the Apple II's built-in game-control button inputs to indicate up to three behavioral states that may occur simultaneously.

Personal computers can be programmed to provide quite elaborate but highly efficient behavioral recording and analysis routines for dedicated research projects (Bernstein & Livingston, 1982; Hargrove & Martin, 1982). Such specialized programs are highly useful in research projects for which the investigator intends to collect large amounts of data over a long period of time. In contrast, the two categories of programs described in this paper were designed to fit a wide variety of laboratory settings instead of a single dedicated application. The primary intended use of the software was for undergraduate laboratory courses in experimental design, animal behavior, and developmental psychology, although several of these routines have been adapted for advanced research. Because of the need to collect data in very diverse settings (ranging from observation of rats dividing their time between the activities of eating, drinking, and grooming to assessing the interdependence of eye contact and vocalization in a social distance experiment), these programs have been purposefully kept extremely simple. Thus, they essentially represent "bare-bones" illustrations of using a small computer for behavioral observation, and they demonstrate programming principles that can be implemented in more elaborate or specialized routines.

Overlapping vs. Nonoverlapping Behaviors

Two different sets of programs will be described; each set includes both a main data acquisition program and a sample summary and analysis routine. The two sets of programs are each designed for use in slightly different research settings. The first (and simplest) of these packages is designed to study the onsets, offsets, and transitions among several mutually exclusive (temporally nonoverlapping) behavioral states. These states may include a "null" state, representing times when the subject under observation is engaged in a behavior other than one of interest. The second (and slightly more complex) package is designed to allow the investigator to monitor up to three behavioral states, which may occur jointly. These temporally overlapping behaviors may coexist within a single subject (e.g., a subject may be simultaneously talking to and holding eye contact with another person). Alternatively, these behaviors may occur jointly among up to three different individual subjects in a single setting (e.g., three ground squirrels each giving alarm calls in response to a predator). A third application might involve the observation of the same behavior within the same subject by up to three independent judges. The last application can be useful for teaching students the principles of interjudge reliability and demonstrating that certain categories of behavior can be more reliably assessed than others. The data acquisition routines for both the mutually exclusive and temporally overlapping situations were written in APPLESOFT BASIC, with machine code routines for accomplishing the timing functions and initializing the programmable timers used to establish the time base. Before describing each routine in more detail, two design principles common to the program should be mentioned.

Separation of Data Acquisition from Data Analysis and Summary

For both software packages, the programs for data acquisition were kept separate from programs used in data analysis and summary. This principle was followed not only for maintaining program simplicity, but also because widely different types of analysis may be called for in different research applications. By providing a highly general data acquisition program that formats data on disks in a straightforward and well documented manner, individual users are free to create their own experimental analysis and summary routines. While some research applications may require examination of sequential dependencies between different behaviors, for example, many applications may require little more than

This work was supported in part by National Science Foundation Grant SER 80-15123 to the author.

a printout of the onset time and duration of each bout of behavior. Particularly in teaching applications, in which it is often appropriate for students to proceed on their own through the steps involved in computing descriptive statistics and constructing frequency distributions, one may wish to have very little "preprocessing" of the raw data other than to present it in a highly readable form. An additional reason for separating the functions of data acquisition from those of data summary and analysis is that, for some applications, one may wish to move the computer (which is highly portable) to the research site for data acquisition. With the present routines, all that need be transported are the computer, a video monitor, and a single disk drive; printers (which are often bulky, heavy, and delicate) can be left behind for use at a later time.

Interrupt-Based Timing

Both categories of observation programs make use of computer interrupts generated by a programmable timer located in the Apple's backplane slots. The program for monitoring temporally overlapping behaviors comes in two versions; one is designed to use the Mountain Computer Apple clock, and another is designed to use the California Computer System (CCS) Model 7440 programmable timer. Both sets of programs can be adapted for use with other products by slight modification in the software.

As opposed to timing routines that read a free-running real-time clock (a procedure followed in the software described by Bernstein & Livingston, 1982, and by Hargrove & Martin, 1982), the interrupt timing system uses a pair of memory locations in the computer to keep track of elapsed time. This pair of memory locations is incremented upon each interrupt. In order to make use of interrupts for timing purposes, two types of routines (written at the assembly level) are required. One of these routines is a setup or initialization procedure that is specific to the particular type of timer being used. Such routines are usually provided in the documentation obtained from the vendor of the timer card. Initialization routines, which are called at the onset of the period of time for which the timer will be in progress, perform several functions. First, they instruct the computer where to find the subroutine that will be automatically executed upon each interrupt. In an Apple II, this is done by placing the address of the interrupt handling subroutine (in our case, the one that will increment the pair of memory locations serving as the internal "clock") in hexadecimal memory locations $3FE and $3FF. A second function of the initialization routine is to provide the sequence of commands that tell the timing device to generate periodic interrupts and at what rate to do so. Finally, it is necessary for an initialization routine to enable the interrupts (i.e., start the timing process) by providing some type of start signal to the timer and by allowing the Apple to accept interrupts through execution of the clear interrupt (CLI) command. Within the same section of code containing the initialization procedures, it is necessary to provide a "stop" routine, which terminates the process of interrupt generation, effectively stopping the "clock." This consists of a command to the timer that stops its interrupt generation, plus a command to the computer (a set interrupt, SEI) that stops the computer from accepting further interrupts. The stop routine is called from the main program at the termination of the observation session.

The second type of routine necessary for implementing interrupt-driven timing is the interrupt handler itself. The interrupt handler is a special type of subroutine that is entered automatically every time an interrupt is generated. For behavioral timing, the simplest form of interrupt handler is one that simply increments the pair of memory locations constituting the elapsed time clock and then returns control to the main program. Although a small amount of assembly language programming is necessary to implement an interrupt-driven timing system, there are some advantages to such an approach. Since upcounting the clock occurs automatically, the higher level language program need not bother with special clock reading routines other than a simple pair of peeks to a pair of memory locations. The remainder of the main program can be constructed as a loop that continuously updates information for the experimental observer and looks for special conditions, such as keypresses, and so on, that indicate changes in behavioral states or commands to terminate the session. Programming in the high-level language thus becomes simpler in structure than would be the case if a set of complex commands to a peripheral device, such as a real-time clock, had to be programmed by a routine imbedded within the main program structure. For some applications in which the timing of relatively rapid behavioral events is required, or a level of precision on the order of hundredths of a second is required, attempting to execute the type of commands necessary to read many of the commercially available real-time clocks from a high-level language might be sufficiently slow that timing errors would occur. While the possibility of such timing errors may not be important if behavior is measured in seconds, interrupt-driven timing routines written at the assembly level preclude the problem even if faster time bases are used.

BCODE: KEYBOARD CODING OF MUTUALLY EXCLUSIVE BEHAVIORS

Listings of the BCODE program written in APPLESOFT BASIC, together with the assembly language that accompany it, are presented in the appendix as Listings 1 and 2. This data acquisition program does not include elaborate error-trapping routines or provisions for on-line data analysis; it is thus an example of an ultimately "stripped-down" behavioral recording program.

The BCODE program is designed to record the onsets and offsets of up to 20 or more behavioral states defined by the observer. Each behavior is coded as a simple character on the Apple keyboard. The observer notes the onset of a behavior by pressing the appropriate key; a carriage return is not required. Each key depression causes the program to store, as a string, the character represented by the key, together with the elapsed time since the beginning of the observation session. A transition into a behavior not under observation (i.e., the offset of a behavior of interest) is keyed by striking the space bar. Since leading space characters cannot be read from a disk in APPLESOFT BASIC, the space bar character is converted into an asterisk character (*) before being stored with the elapsed time. The elapsed time is measured in units defined by the observer prior to beginning the session; these units can be any multiple of .1 sec from .1 to 25.5 sec. The experimental session is terminated by pressing "control Q." At that point, the strings are written to disk in a file named by observer for later use by analysis in summary programs.

Timing and Time-Base Selection

The BCODE program makes use of a CCS 7440 timer located in Slot 2 of the Apple backplane. This timer contains a set of three counters, two of which are used in the present application. The initial time base for one of the timers (Timer 1) is taken from the Apple computer's 14-MHz internal clock and is divided into units of .1 sec. The output of this timer (now in .1-sec cycles) is input via a wire jumper into the second timer, which is initially loaded with the number of .1-sec units the observer wishes to define as the time base. Timer 2 generates an interrupt at the point in time at which the number of time units with which it has been loaded have been counted out. The timers are reloaded at each interrupt. Details of the timer commands and instructions for installing the jumper wire are provided in detail in the manual provided with the CCS 7440 timer card. The assembly language code shown in Lines 27-47 of Listing 2 provides the appropriate commands to the timers and clears the two memory locations (hexadecimal addresses, $FA and $FB, or decimal addresses, 250 and 251), which are used as the "clock" to measure the elapsed time. As shown in Listing 2, the timer that generates the interrupts is initially loaded with a value of 10. However, that command can be modified by the APPLESOFT BASIC main program, which changes that initial value loaded to any value between 1 and 255 (see Lines 22-25 of Listing 1). Thus, the observer is able to select a time base ranging from .1 sec/unit to 25.5 sec/unit. For recording relatively fleeting behaviors in a highly active organism (e.g., sexual behavior and aggression in wild mice), one might wish to select a fairly short time base, such as .2 sec. For long-term observation of more slowly changing behavior states (e.g., documenting the activities of a teacher in a classroom), a value of 5-10 sec/"tick" might be appropriate. It is important that the observer remember what time base was selected, since no record of that value is contained in the data stored on the disk.

The interrupt handling routine located at the hexadecimal address $300 (Lines 9-25 of Listing 2) is simply a segment of code that upcounts the pair of memory locations after having temporarily saved the values of the computer's internal registers and then restores those registers to their original values before returning to the main program. Upon a keypress, the two memory locations (250 and 251 decimal) are peeked by the APPLESOFT program. Location 251 contains the number of multiples of 256 ticks (e.g., the overflow from the 8-bit register) and Location 250 will contain the remainder. These two values are then converted into the total number of time units.

Data Analysis and Summary

Data from the BCODE observation program are written to floppy disks in the following format: The first record contains the number of behavioral bouts during the observation session, and the remaining records (the number of which is equal to the number of bouts) contain strings consisting of the letter code of the behavior (including the * for the null behavior) followed by a string of digits indicating the elapsed time since the beginning of the session at the point when the behavior was entered.

An example of a simple data summary program, which performs simple summary statistics, is provided as Listing 3 of the appendix. This program, called BSUM, consists of three subroutines, one of which inputs the data generated by the BCODE program, the second of which prints out the behaviors and their onset times in order of their occurrence, and the third of which prints the total number of episodes, cumulative time spent in those episodes, percentage of total session time, and mean and standard deviation of episode length for any single behavior. The data input subroutine asks the experimenter for the name of the data file and then inputs the number of bouts and the string associated with each bout from disk. The subroutine that prints the data by sequential episodes separates each of the strings into the label for the behavior and the elapsed time at which it occurred (storing the latter as a floating-point number for later use). It then prints the letter-code label, followed by elapsed time in its order of occurrence. The subroutine that provides descriptive statistics for individual behaviors involves a search process that selects out those bouts in which the letter code of a behavior matches the code supplied by the experimenter and then calculates the descriptive statistics for that behavior. A sample output from BSUM is shown in Listing 4. One could easily add subroutines to this program to perform operations such as sequential dependencies, or one could delete the descriptive statistic routines for use in undergraduate instruction.

Limitations

As written, the BCODE program will handle up to 500 successive bouts, including null behaviors, and the elapsed time clock will "wrap around" to zero after 16,384 counts. At a 1.0-sec rate of counting, that would amount to over 4.5 h of continuous monitoring. The program will not terminate with the clock overflow, so the researcher can simply correct the data by hand, at the point at which the overflow occurs. Exceeding 500 different behavioral transitions, however, will automatically terminate the program and write the existing data to disk.

MONITORING NONEXCLUSIVE BEHAVIORS: THE TACT3 PACKAGE

The TACT3 program and associated data analysis routines were developed for use in any situation in which the observer is required to monitor the occurrence of up to three separate behaviors, each of which is either present or absent at any point in time and each of which may occur while any of the other behaviors is also present. The behaviors are recorded through the Apple II computer's push-button game input, which can be attached to simple push buttons or to toggle switches to fit the needs of a given application.[1] Two versions of the TACT3 program have been written; one is designed for use with the Mountain Hardware Apple clock, and the other has been adapted for use with the CCS Model 7440 timer. The version shown in Listings 5 and 6 is the Mountain Hardware version.[2]

Principles of Operation

The TACT3 program is, like the BCODE program, simple to operate. After the program is loaded and run, a simple depression of the space bar begins the observation panel. The visual information displayed to the operator is somewhat more elaborate than in the BCODE program, since the observer is required to monitor the behaviors, which may overlap in time. While observation is in progress, the video screen is continuously updated with the following information: elapsed time since the beginning of the session, the state of each of the behaviors (on or off), cumulative time spent since the beginning of the session in each of the behaviors, and the number of discrete episodes of each of the behaviors. Upon session termination, which is signaled by pressing any of the keys on the Apple's keyboard, the onset times and episode lengths for each bout of behavior are stored on disks for subsequent analysis and a preliminary data summary is shown on the video screen. This abbreviated data summary includes the number of episodes of each of the behaviors monitored, the cumulative time spent in each behavior, and the percentage of the session time spent in each of the behaviors. The total session time is also presented at the bottom of the screen. This partial summary allows the experimenter to write down relevant information at the time of the experiment for future reference; this information is also available in hard copy when data analysis routines are run at a later time.

The Interplay Between the APPLESOFT TACT3 Program and the Interrupt Routines

The Mountain Hardware clock and CCS timer versions of the TACT3 program differ only in the assembly language routines that initialize and control the clock or timer. The interrupt processing is considerably more elaborate than in the BCODE program, since in addition to providing an elapsed time clock, the interrupt routine also performs the task of checking the states of the push-button inputs to the Apple used to signal the presence or absence of a given behavior. This checking is done once at each 1-sec interrupt, and the three switch inputs are then scanned in rapid succession. On the Apple, the state of each button is assessed by testing the most significant bit of three successive memory locations, hexadecimal $C061, $C062, $C063. Should one of the bits be set, then that program checks to see whether or not the bit for that memory location was also set on the previous interrupt; if not, it indicates the onset of a new behavioral bout, and a memory location that counts the number of new behavior bouts for that particular behavior is incremented. Also, if the bit indicating the depression of the switch or button is active during an interrupt, a pair of memory locations that serve as a cumulative time clock for that particular behavior is incremented. Hence, the program is time sampling for the presence or absence of each behavior, once every 1 sec.

Unlike the BCODE program, which used a single pair of memory locations simply as an elapsed time clock, the TACT3 program uses four different pairs of memory locations: one for a session elapsed time clock and the remaining three for cumulative time clocks for each of the three behaviors. However, in addition to keeping track of the cumulative time spent in each behavior, the interrupt handling routine uses three blocks of memory, located above the locations used by the APPLESOFT program, for storage of the onset and offset times of each behavioral bout. Thus the interrupt handling routine, written in assembly language, performs the entire task of managing the data as they are obtained. The role of the APPLESOFT main program is exclusively that of updating the information on the video screen while the observation is in progress. For this reason, the assembly language programming is considerably more complex than that used in the BCODE program, making use of several of the 6502 processor's more sophisticated addressing modes and data testing capabilities.

Since the data are actually collected by machine code routines rather than by an APPLESOFT program, it is necessary for the APPLESOFT main program to convert disk data (stored in binary form in specified memory locations) to a form of data that can be used by the APPLESOFT language for arithmetic manipulation, and which then can be written to disk. One disadvantage of placing a greater burden of the data collection on machine language routines, as opposed to embedding them within a high-level language program, is that the program listings are less readable. For example, the high-level language program (see Listing 5) contains a great many peeks, pokes, and calls; one must consult the assembly language listings to understand the effects of these statements. However, the data sensing by machine language routines may be necessary to avoid timing errors if several different inputs must be scanned in rapid succession. It may be possible to write a program entirely in APPLESOFT BASIC that scans the state of three button inputs and determines whether their state has changed since the last scan, all within a 1-sec period. However, such a program would push the upper limit of the number of tasks that could be completed in that time interval.

Format of Data Generated by TACT3

Once the observation session has been terminated by striking the space bar (or any key on the Apple keyboard), the TACT3 program asks for a file name for the data. Once this has been supplied, the data from the observation session is written to disk for later use by another program. Record 1 contains the total session time. The second record contains the number of activities or behaviors that were monitored during that session. Record 3 contains the total number of separate episodes of the first behavior. The fourth record contains the length of the first episode of the first behavior, and the fifth record contains the onset time of the first episode of the first behavior monitored. From then on, each successive record contains the length of each behavioral bout of the behavior, followed by a record containing the onset time of that bout, until all of the episodes of the first behavior are exhausted. After that, the next successive record contains the number of bouts of the second behavior, followed by pairs of records indicating the length of each bout and its onset time, until all of the bouts of behavior of the second activity have been described. If those behaviors are monitored, the same procedure is followed for recording the data.

A short segment of APPLESOFT BASIC code that retrieves these data from a disk file is provided in the appendix, as Listing 7. This subroutine can be imbedded in any data summary and analysis program the user wishes to create. While we have written several different analysis programs to be used in conjunction with the TACT3 observation program, the programming principles involved essentially parallel those of the BSUM program described earlier and thus will not be presented in detail. Examples of the type of analysis performed on data generated by the TACT3 program are described by LaGuardia (1982).

Limitations of the BCODE Routine

In the version presented here, a maximum of 200 separate episodes or bouts of any single behavior is permitted. If this limit is exceeded, the observation session automatically terminates and the data are sent to disk for storage. These limits may be readjusted by moving the locations of the memory buffers for the data storage if, for example, only two behaviors are to be monitored and there is a need to monitor more than 200 separate bouts of at least one of the behaviors during a session. As with the BCODE program, the "clock" will reset to zero after 16,384 counts.

SUMMARY AND CONCLUSIONS

This paper has described two general-purpose programs acquiring observational data using the Apple II personal computer. These data acquisition programs represent prototypes of routines that can be applied to a wide variety of research and teaching applications. In addition to being designed for slightly different applications (mutually exclusive vs. temporally overlapping behaviors), the BCODE and TACT3 data acquisition programs have employed slightly different programming strategies to accomplish the task of recording the stream of behavior in a naturalistic setting. In the BCODE example, in which only a single behavior could occur at a given time, the data acquisition process was accomplished almost entirely by high-level (APPLESOFT BASIC) language programming, with only the elapsed time generation performed by machine code. Such a programming approach is easier to understand, particularly by novice programmers and students who may have a little experience with machine code or assembly-level programming. However, the need to monitor several events that may occur simultaneously and the need to signal the presence of those events from several remote locations (LaGuardia, 1982) may preclude the use of the keyboard as a data input console. Because of time constraints, exclusive reliance upon a highly interpretative language for data acquisition is not practical for such application. Thus, in the example provided by the TACT3 program, interrupt-driven assembly language subroutines took over the primary responsibility for data input. The role of the high-level language "host" program was dedicated to updating information displayed on the video screen for the researchers' benefit and the task of converting the machine code-generated data into a form suitable for analysis by high-level language summary programs. In using both approaches, an effort was made to format the data in a straightforward manner to be stored on floppy disks for subsequent analysis and summary by other programs. This separation of data acquisition from analysis procedures allows the experimenter to take maximum advantage of the high degree of flexibility and portability offered by today's personal computers.

REFERENCES

Bernstein, D. M., & Livingston, C. An interactive program for observation and analysis of human behavior in a long-term continuous laboratory. *Behavior Research Methods & Instrumentation,* 1982, 14, 231-235.

Hargrove, D. S., & Martin, T. A. Development of a microcomputer system for verbal interaction analysis. *Behavior Research Methods & Instrumentation,* 1982, 14, 236-239.

LaGuardia, R. L. The application of computer systems to research in experimental social psychology. *Behavior Research Methods & Instrumentation,* 1982, 14, 249-252.

NOTES

1. Because of FCC regulations, game buttons are no longer supplied as standard equipment on Apples. However, the connector socket is still available, and instructions on how to use it may be found in the Apple reference manual.

2. A listing of the CCS version is available from the author upon request.

APPENDIX

```
1  REM   **** MUTUALLY EXCLUSIVE BEHAVIOR RECORDING
2  REM  **** COPYRIGHT 1981, JOHN H. FLOWERS
5  DIM R$(500):D$ =  CHR$ (4):Q$ =  CHR$ (17):CL = 250:CH = 251:M = 256
6 KBD =  - 16384
10  HOME : PRINT D$;"BLOAD CLOCK.OBJO"
20  PRINT "CODING ROUTINE FOR EXCLUSIVE BEHAVIORS": PRINT : PRINT "THIS ROUTINE
REQUIRES A CCS 7440 IN S#2."
21  PRINT "JOHN H. FLOWERS -- MAY,1981.": PRINT
22  VTAB (9): PRINT "ENTER TIMEBASE IN 0.1 SEC UNITS."
24  INPUT ">";TB: IF TB < 1 OR TB > 255 THEN  CALL  - 198: PRINT "REENTER!": GOT
O 24
25  POKE 814,TB: PRINT : PRINT "ANY KEY TO BEGIN"
30  GET X$: PRINT X$: HOME : PRINT "A CONTROL-Q WILL STOP THE SESSION."
40  VTAB (5): PRINT "CURRENT BEHAVIOR >"
50  CALL 793: REM   **** START TIMER
60 I = 0: REM  *** BEHAVIOR COUNTER
70 X =  PEEK (KBD): IF X < 127 THEN 70
75  POKE  - 16368,0:X$ =  CHR$ (X - 128): VTAB (5): HTAB (22): PRINT X$
77  IF X$ = " " THEN X$ = "*"
80 I = I + 1:R$(I) = X$ +  STR$ ( PEEK (CL) + M *  PEEK (CH))
90  IF I = 500 THEN  HOME : GOTO 199
100  IF X$ = Q$ THEN  HOME : GOTO 200
110  GOTO 70
199  CALL 850: CALL  - 198: HOME : FLASH : PRINT "MAX. # OF EPISODES EXCEEDED":
NORMAL : GOTO 205
200  CALL 850: HOME
205  VTAB (5): INPUT "DATA FILE NAME>";N$
210  PRINT D$;"OPEN ";N$: PRINT D$;"DELETE ";N$: PRINT D$;"OPEN ";N$: PRINT D$;"
WRITE ";N$
215  PRINT I
220  FOR K = 1 TO I: PRINT R$(K): NEXT K
230  PRINT D$;"CLOSE ";N$
240  END
```

Listing 1. The BCODE program.

```
0000:            ***** CLOCK ROUTINE FOR CCS TIMER
0000:            *** KEEPS TIME IN $FA & $FB IN
0000:            *** UNITS SPECIFIED IN LOCATION $32E
0000:            *** AS THE NUMBER OF TENTHS OF SECONDS
00FA:            CLKL     EQU  $FA
00FB:            CLKH     EQU  $FB
----- NEXT OBJECT FILE NAME IS CLOCK.OBJO
0300:                     ORG  $300
0300:A5 45       TICK:    LDA  $45
0302:48                   PHA
0303:8A                   TXA
0304:48                   PHA
0305:98                   TYA
0306:48                   PHA
0307:20 52 03             JSR  STOP
030A:E6 FA                INC  CLKL       CLKL
030C:D0 02                BNE  LEAVE
030E:E6 FB                INC  CLKH
0310:20 4B 03    LEAVE:   JSR  GO
0313:68                   PLA
0314:A8                   TAY
0315:68                   PLA
0316:AA                   TAX
0317:68                   PLA
0318:40                   RTI
0319:            **** SETUP ROUTINE
0319:A9 00                LDA  #$0
031B:8D FE 03             STA  $3FE
031E:A9 03                LDA  #$03
0320:8D FF 03             STA  $3FF
0323:A9 71                LDA  #$71;     TIMER2 CMD
0325:8D A1 C0             STA  $C0A1
0328:A9 00                LDA  #0
032A:8D A4 C0             STA  $C0A4;    TIMER2 HIDATA
032D:A9 0A                LDA  #$0A;     10 TICKS
032F:8D A5 C0             STA  $C0A5
0332:A9 93                LDA  #$93
0334:8D A0 C0             STA  $C0A0;    TIMER1 CMD
0337:A9 C7                LDA  #$C7
0339:8D A2 C0             STA  $C0A2;    TIMER1HIBYTE
033C:A9 C0                LDA  #$C0
033E:8D A3 C0             STA  $C0A3;    LOBYTE
0341:A9 00                LDA  #0
0343:85 FA                STA  CLKL
0345:85 FB                STA  CLKH;     CLEARS CLOCKS
0347:20 4B 03             JSR  GO
034A:60                   RTS
034B:            *** START ROUTINE
034B:A9 92       GO:      LDA  #$92
034D:8D A0 C0             STA  $C0A0
0350:58                   CLI
0351:60                   RTS
0352:            **** STOP ROUTINE
0352:A9 93       STOP:    LDA  #$93
0354:8D A0 C0             STA  $C0A0
0357:78                   SEI
0358:60                   RTS
```

Listing 2. Timing routine used by BCODE (2 days).

```
5  DIM R$(500),BX$(200),T(200)
10 D$ =  CHR$ (4):SP$ = " ":Q$ =  CHR$ (17)
15 K$ =  CHR$ (12)
20  GOSUB 1000
30  GOSUB 2000: REM  *** PRINT BY EPISODES
40  GOSUB 4000: REM   **** SUMMARY FOR INDIV. BEHAVIORS
50  HOME : END
1000  HOME : PRINT "OBSERVATIONAL DATA CODING"
1002  PRINT "FOR DATA GENERATED BY BCODE.": PRINT : PRINT "JOHN H. FLOWERS, MAY,
1981."
1005  VTAB (9): INPUT "DATA FILE NAME>";N$
1010  PRINT D$;"OPEN ";N$: PRINT D$;"READ ";N$
1020  INPUT N
1030  FOR I = 1 TO N: INPUT R$(I): NEXT I
1040  PRINT D$;"CLOSE ";N$
1050  RETURN
2000  PRINT D$;"PR#1"
2005  PRINT K$: PRINT "BEHAVIOR IN ORDER OF APPEARANCE IN": PRINT "DATA FILE ";N
$
2007  PRINT
2008  PRINT "CODE            ONSET"
2010  FOR I = 1 TO N
2020 B$ =  LEFT$ (R$(I),1):T$ =  RIGHT$ (R$(I), LEN (R$(I)) - 1)
2022 BX$(I) = B$:T(I) =  VAL (T$)
2025  IF B$ =  CHR$ (17) THEN B$ = "#"
2027  IF B$ = " " THEN B$ = "*"
2030  PRINT B$,T$
2040  NEXT I
2042  PRINT : PRINT "****************************************"
2045  PRINT K$
2050  PRINT D$;"PR#0"
2060  RETURN
4000  HOME : PRINT "INPUT THE LETTER CODE OF A BEHAVIOR": PRINT "WHICH YOU WISH
TO SUMMARIZE."
4005  PRINT : PRINT "(A 'O' WILL QUIT THIS ROUTINE)"
4010  INPUT ">";V$
4015  IF V$ = "O" THEN 4900
4020  PRINT : PRINT "DESCRIPTIVE LABEL FOR THAT BEHAVIOR?": INPUT ">";W$
4030  PRINT D$;"PR#1": PRINT K$
4040  PRINT "SUMMARY FOR BEHAVIOR ";V$;" - ";W$
4050  PRINT
4055  PRINT "EPISODE  ONSET     LENGTH"
4060 S = 0:S2 = 0:NE = 0
4070  FOR I = 1 TO N
4080  IF BX$(I) <  > V$ THEN 4500
4090 NE = NE + 1:P = T(I + 1) - T(I):S = S + P:S2 = S2 + P ^ 2
4100  PRINT NE;: POKE 36,10: PRINT T(I);: POKE 36,20: PRINT P
4500  NEXT I
4510  IF NE = 0 THEN  PRINT "THIS BEHAVIOR DID NOT OCCUR": GOTO 4640
4550  PRINT
4560  PRINT "# EPISODES   CUM. TIME   % SESSION TIME"
4570  POKE 36,2: PRINT NE;: POKE 36,15: PRINT S;: POKE 36,29: PRINT 100 * S / T(
N)
4580  PRINT
4600  IF NE < 2 THEN  PRINT "TOO FEW EPISODES FOR STATISTICS.": GOTO 4640
4610  PRINT "MEAN LENGTH          STD. DEV."
4620 MN = S / NE:V = S2 - S ^ 2 / NE:V =  SQR (V / NE)
4630  PRINT  TAB( 3);MN; TAB( 18);V
4640  PRINT : PRINT "****************************************"
4650  PRINT K$: PRINT D$;"PR#0"
4700  GOTO 4000
4900  RETURN
```

Listing 3. BSUM program: Summarizes data generated by BCODE.

```
BEHAVIOR IN ORDER OF APPEARANCE IN
DATA FILE RAT1

CODE          ONSET
S              22
R              35
*              48
S              61
*              73
G              76
R              91
*              108
S              118
*              122
E              123
*              134
D              141
*              163
R              178
*              181
E              220
*              227
E              234
*              266
D              269
G              301
*              341
S              346
R              350
S              363
#              379

****************************************
```

```
SUMMARY FOR BEHAVIOR D - DRINKING

EPISODE  ONSET     LENGTH
1          141      22
2          269      32

# EPISODES   CUM. TIME   % SESSION TIME
  2             54          14.2480211

MEAN LENGTH          STD. DEV.
  27           4.99999995

****************************************


SUMMARY FOR BEHAVIOR E - EATING

EPISODE  ONSET     LENGTH
1          123      11
2          220      7
3          234      32

# EPISODES   CUM. TIME   % SESSION TIME
  3             50          13.1926121

MEAN LENGTH          STD. DEV.
  16.6666667    10.9645895

****************************************
```

```
SUMMARY FOR BEHAVIOR S - SNIFFING

EPISODE  ONSET     LENGTH
1          22       13
2          61       12
3          118      4
4          346      4
5          363      16

# EPISODES   CUM. TIME   % SESSION TIME
  5             49          12.9287599

MEAN LENGTH          STD. DEV.
  9.8          4.91528229

****************************************
```

Listing 4. Sample BSUM printout.

```
2  REM  **** COPYRIGHT 1981, JOHN H. FLOWERS
5  HIMEM: 32767
6  HOME : PRINT "TACT3: BEHAVIOR MONITORING FOR UP TO": PRINT "THREE ACTIVITIES
OR BEHAVIOR STATES."
8  PRINT : PRINT "THIS VERSION USES A MOUNTAIN HARDWARE": PRINT "CLOCK IN SLOT #
4.": PRINT : PRINT "JOHN H. FLOWERS -- MAY, 1981."
9  PRINT : PRINT
10  DIM PX(3,200),TX(3,200),CL(3),CH(3),EL(3),EH(3),BF(3)
15  DIM F(3),NE(3)
20  FOR I = 0 TO 2
25 F(I) = 251 + I: POKE F(I),0
30 BF(I) = 32768 + 800 * I:CL(I) = 950 + I:CH(I) = 953 + I:EL(I) = 958 + I:EH(I)
 = 961 + I
32  NEXT I
35  REM  **** ESTABLISHED ADRESSES OF BUFFERS AND CLOCKS, ETC.
40  FOR I = 0 TO 2
45 H =  INT ((BF(I)) / 256):L = BF(I) - 256 * H: POKE 944 + I,H: POKE 947 + I,L:
 NEXT I
47  REM  *** POKE IN BUFFER ADDRESSES
50  FOR I = 950 TO 963: POKE I,0: NEXT I
52  REM  *** CLEAR CLOCKS AND COUNTERS
55 D$ =  CHR$ (4):M = 256:K1 =  - 16384:K2 =  - 16368
57 TL = 956:TH = 957
60  PRINT D$;"BLOAD ACT3.OBJO": REM              ***MACHINE PROGRAM FOR MHC
     CLOCK
70  INPUT "# OF ACTIVITIES YOU WISH TO OBSERVE>";NACT
72  IF NACT < 1 OR NACT > 3 THEN  CALL  - 198: GOTO 70
75  POKE 863,NACT: REM   *** LOOP TERM.
80  PRINT : PRINT "ANY KEY TO BEGIN": GET X$: PRINT X$: GOSUB 5000: CALL 875
90 Q = 0
100  VTAB (2): HTAB 10: PRINT  PEEK (TL) + M *  PEEK (TH)
110  FOR I = 0 TO NACT - 1
115 J = 10 * (I + 1)
120  VTAB (5): HTAB (J): IF  PEEK (F(I)) THEN  INVERSE : PRINT "ON ": NORMAL : G
OTO 130
125  PRINT "OFF"
130  VTAB (8): HTAB (J): PRINT  PEEK (CL(I)) + M *  PEEK (CH(I))
140  VTAB (11): HTAB (J): PRINT  PEEK (EL(I)) + M *  PEEK (EH(I))
145  IF  PEEK (EL(I)) = 200 THEN Q = 1: GOTO 200
147  NEXT I

150  IF  PEEK (K1) > 127 THEN 200
160  GOTO 100
200  CALL 896: POKE K2,0: REM   INT OFF ETC.
202  IF Q THEN  CALL  - 198: PRINT "***** SESSION TERMINATED DUE TO": PRINT "TOO
 MANY EPISODES OF ACTIVITY ";I: PRINT "RUN 205 TO CONTINUE": STOP
205  REM  **** GET DATA
210 TT =  PEEK (TL) + M *  PEEK (TH): REM  *** SESSION TIME
220  FOR I = 0 TO NACT - 1
221 F(I) =  PEEK (F(I))
225 NE(I) =  PEEK (EL(I)): REM  ** CAN'T HAVE MORE THAN 200
227 CT(I) =  PEEK (CL(I)) + M *  PEEK (CH(I))
230  FOR K = 1 TO NE(I):J = BF(I) + 4 * (K - 1)
235 TX(I,K) =  PEEK (J) + M *  PEEK (J + 1)
240  IF K = NE(I) AND F(I) THEN Z = TT: GOTO 250
245 Z =  PEEK (J + 2) + M *  PEEK (J + 3)
250 PX(I,K) = Z - TX(I,K)
260  NEXT K: NEXT I
299  PRINT : PRINT
300  INPUT "NAME FOR DATA FILE>";N$
310  PRINT D$;"OPEN ";N$
320  PRINT D$;"DELETE ";N$
330  PRINT D$;"OPEN ";N$
340  PRINT D$;"WRITE ";N$
350  PRINT TT: PRINT NACT
360  FOR I = 0 TO NACT - 1: PRINT NE(I)
370  PRINT CT(I)
380  FOR J = 1 TO NE(I)
390  PRINT PX(I,J): PRINT TX(I,J)
400  NEXT J: NEXT I
410  PRINT D$;"CLOSE ";N$

500  HOME
510  PRINT "***** SUMMARY OF ACTIVITIES **"
520  VTAB (4): PRINT "#EPS. ";: FOR I = 0 TO NACT - 1: PRINT NE(I),: NEXT I: PRI
NT
530  VTAB (8): PRINT "CUM.  ";: FOR I = 0 TO NACT - 1: PRINT CT(I),: NEXT I: PRI
NT
540  VTAB (12): PRINT "% TM. ";: FOR I = 0 TO NACT - 1
542 PT = CT(I) / TT * 1000:PT =  INT (PT):PT = PT / 10: PRINT PT,: NEXT I: PRINT

550  VTAB 16: PRINT "SESSION TIME = ";TT
600  END
5000  HOME
5005  VTAB (2): PRINT "T="
5010  VTAB (5): PRINT "ACT"
5020  VTAB (8): PRINT "CUM"
5030  VTAB (11): PRINT "#EP"
5040  RETURN
```

Listing 5. TACT3 program.

```
ACT3               THREE CATEGORY SAMPLE

0000:               2 *** TIME SAMPLING PROGRAM
0000:               3 *** USES MOUNTAIN HARDWARE CLOCK
0000:               4 *** FOR USE WITH TACT3
0045:               5 TEMPA   EQU  $45
00F9:               6 PTRL    EQU  $F9
00FA:               7 PTRH    EQU  $FA
00FB:               8 FLG     EQU  $FB
C061:               9 BUTTON  EQU  $C061
03B3:              10 PL      EQU  $3B3
03B0:              11 PH      EQU  $3B0
03B6:              12 CLKL    EQU  $3B6
03B9:              13 CLKH    EQU  $3B9
03BC:              14 TIMEL   EQU  $3BC
03BD:              15 TIMEH   EQU  $3BD
03BE:              16 EPL     EQU  $3BE
03C1:              17 EPH     EQU  $3C1
----- NEXT OBJECT FILE NAME IS ACT3.OBJ0


0300:              18         ORG  $300
0300:              19 ***
0300:              20 ***
0300:              21 ***
0300:              22 *** INTERRUPT HANDLER
0300:A5 45         23 INTH:   LDA  TEMPA;    SAVE REG'S
0302:48            24         PHA
0303:8A            25         TXA
0304:48            26         PHA
0305:98            27         TYA
0306:48            28         PHA
0307:20 80 03      29         JSR  STOP
030A:EE BC 03      30         INC  TIMEL
030D:D0 03         31         BNE  L1
030F:EE BD 03      32         INC  TIMEH
0312:A2 00         33 L1:     LDX  #00
0314:BD B3 03      34 LOOK:   LDA  PL,X
0317:85 F9         35         STA  PTRL;     INDEX FOR BUFFER
0319:BD B0 03      36         LDA  PH,X
031C:85 FA         37         STA  PTRH
031E:BD 61 C0      38         LDA  BUTTON,X; BUTTON ON?
0321:30 0B         39         BMI  ON;       YEP,GOTO ON
0323:B5 FB         40         LDA  FLG,X;    NOPE - SEE IF OFF LAST TIME
0325:F0 35         41         BEQ  NXTB;     IF SO, CHECK NEXT BUTTON
0327:A9 00         42         LDA  #0;       ELSE CLEAR FLAG
0329:95 FB         43         STA  FLG,X
032B:4C 44 03      44         JMP  STORE;    AND STORE OFFSET
032E:FE B6 03      45 ON:     INC  CLKL,X;   INCR TIME
0331:D0 03         46         BNE  L2
0333:FE B9 03      47         INC  CLKH,X
0336:B5 FB         48 L2:     LDA  FLG,X;    ON LAST TIME?
0338:D0 22         49         BNE  NXTB
033A:F6 FB         50         INC  FLG,X
033C:FE BE 03      51         INC  EPL,X;    NEW EPISODE
033F:D0 03         52         BNE  STORE
0341:FE C1 03      53         INC  EPH,X
0344:A0 00         54 STORE:  LDY  #0;       STORE ONSET OR OFFSET
0346:AD BC 03      55         LDA  TIMEL
0349:91 F9         56         STA  (PTRL),Y
034B:C8            57         INY
034C:AD BD 03      58         LDA  TIMEH
034F:91 F9         59         STA  (PTRL),Y
0351:FE B3 03      60         INC  PL,X
0354:FE B3 03      61         INC  PL,X;     BUMP TWICE!
0357:D0 03         62         BNE  NXTB
0359:FE B0 03      63         INC  PH,X
035C:E8            64 NXTB:   INX
035D:8A            65         TXA
035E:C9 03         66         CMP  #3;       DONE YET?
0360:              67 *** MODIFIABLE BY BASIC POKE
0360:D0 B2         68         BNE  LOOK
0362:20 79 03      69         JSR  GO;       ENABLE INTERRUPTS
0365:68            70         PLA
0366:A8            71         TAY
0367:68            72         PLA
0368:AA            73         TAX
0369:68            74         PLA
036A:40            75         RTI
036B:              76 **** NOW SETUP ROUTINE
036B:A9 00         77 SETUP:  LDA  #0
036D:8D FE 03      78         STA  $3FE
0370:A9 03         79         LDA  #$03
0372:8D FF 03      80         STA  $3FF;     INT. ADDR
0375:20 79 03      81         JSR  GO
0378:60            82         RTS
0379:              83 *** START ROUTINE
0379:A9 01         84 GO:     LDA  #$01
037B:8D C9 C0      85         STA  $C0C9
037E:58            86         CLI
037F:60            87         RTS
0380:              88 **** STOP ROUTINE
0380:78            89 STOP:   SEI
0381:A9 00         90         LDA  #$00
0383:8D C9 C0      91         STA  $C0C9
0386:AD C7 C0      92         LDA  $C0C7
0389:AD C8 C0      93         LDA  $C0C8
038C:60            94         RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Listing 6. Timing and button-sensing routines used by TACT3.

```
9000  REM  *** THIS ROUTINE READS DATA GENERATED BY TACT3 FROM DISK.
9005  REM  *** IT CAN BE INSERTED IN ANY USER'S SUMMARY PROGRAM.
9007  REM  **** MAIN PROGRAM MUST DIMESION PX AND TX TO (3,200)
9010  PRINT : INPUT "DATA FILE NAME>";N$
9020 D$ =  CHR$ (4): PRINT D$;"OPEN ";N$: PRINT D$;"READ ";N$
9030  INPUT TT: INPUT NACT
9040  FOR I = 0 TO NACT - 1: INPUT NE(I)
9050  INPUT CT(I)
9060  FOR J = 1 TO NE(I)
9070  INPUT PX(I,J): INPUT TX(I,J)
9080  NEXT J: NEXT I
9090  PRINT D$;"CLOSE ";N$
9100  RETURN
```

Listing 7. Sample subroutine for reading data generated by TACT3.