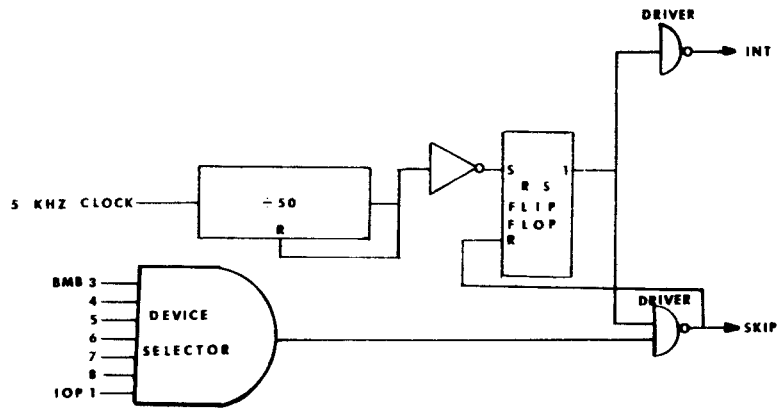

Fig. 3. Simplified diagram of the clock interface.

50 kHz. This selection translates into 100 or 1,000 Hz. The switch can also open the circuit between the clock and the divider. This effectively disables the interrupt so that other programs (i.e., FOCAL, etc.) may be run.

THE BUS-ALL APPROACH

The quad size input, output, and clock cards are placed in a H933 mounting panel. The signals, as they come from a PDP-8/L or the KA8-A, are bussed across the H933. The device code is wired on the input, output, and clock cards. The input and output cards may be placed in any slot. The clock card has resistors to terminate the IOP pulses, therefore it must be placed in the last slot, furthest from the cables.

REFERENCES

Snapper, A. G., & Kadden, R. M. Time-sharing in a small computer based on the use of a behavioral notation system. In B. Weiss (Ed.), *Digital computers in the behavior laboratory*. New York: Appleton-Century-Crofts, 1973. Pp. 41-97.

Snapper, A. G., Knapp, J. Z., Kushner, H. K., & Kadden, R. M. A notation system and computer program for behavioral experiments. Paper presented at the meeting of the Digital Equipment Computer Users Society, New York, June 1967.

Snapper, A. G., & Walker, A. The SKED software system. DECUS Program Library, DECUS 8-465, 1971.

Walker, A., & Snapper, A. G. Improvements to the SKED processor central software system. In DECUS Proceedings, Spring 1971. Pp. 7-12.

Behavior Research Methods & Instrumentation
1974, Vol. 6, No. 2, 173-176

Programming special functions in the SKED system

ARTHUR SNAPPER
Western Michigan University, Kalamazoo, Michigan 49001

and

BRUCE HAMILTON
American University, Washington, D.C. 20016

Machine language subroutines can be integrated with the SKED system. These subroutines can shorten lengthy programs that could otherwise be handled by SKED, and can provide complex decision functions, data recording schemes, and software for new peripheral devices. Rules and examples for each function will be presented.

SKED is a higher level, general language used to simplify programming the PDP-8 for experimental control and data recording. An unfortunate restriction shared by many user-oriented languages is their inflexibility when faced with a specific requirement unforeseen by the creators of the system. Some user finds his application needs 8.5K of core in an 8K machine, or that it is impossible to apply a program that was originally designed for schedules of reinforcement to verbal behavior experiments or to automated psychoanalysis.

Although SKED has more generality than most process control languages, for some applications the system will be either inefficient or impossible to use. For this reason, the Run Time System (RTS) has a built-in method for referencing machine language subroutines, so that the special-purpose programs can be readily accessed at the appropriate time or by the specified response in the state program.

The special-purpose program can be written (by an experienced programmer) in machine language, assembled by PAL III, and loaded along with the RTS in

one of two ways (described in a later section). The special routine, or F3, can be inserted in unused portions of any field and protected from being overlaid by state tables. If the F3 is written with some care, it can then be used by any or all stations concurrently to obtain the special function.

FORMAT

The format of the F3 routine has been designed to permit the passage of arguments from the state table to the special-purpose subroutine (F3). This allows for the use of variable parameters, which is essential for the generality of these routines. For example, an F3 has been written to provide a sequence of variable intervals or ratios. The actual values of the intervals or ratios can be specified by arguments in the F3 command that calls for the subroutine. However, if the subroutine had been developed to serve only in the original experiment that required variable intervals, it would have been written without arguments and the particular intervals would have been listed within the subroutine. However, including arguments in the calling statement permits the use of this routine in any experiment requiring a variable, but prespecified, set of values.

The format of an F3 statement which permits the passage of arguments is F3 (Oxxxx, Arg1, Arg2, Arg3, . . . ArgN), where Oxxxx is the octal address of the entry point of the subroutine and the optional arguments are used to set parameters of the routine when needed. For example, an F3 has been developed to transfer control from the state table (which may be located in any field) to an F3 stored in any field of memory, although F3s are presumed to be located in Field Ø. If it is necessary to locate them on some other field, this routine can be used to transfer control and still provide access to the arguments located in the state table. To access the F3 located in a nonzero field, the user simply calls the field F3 with a statement of the following format: F3 (OField, New Field, Address, Arg1, Arg2, . . . ArgN), where OField is the octal address in Field Ø of Field 3, New Field is the octal field number where the desired F3 is located, Address is the octal address of the F3 in the nonzero field, and Arg1, etc., are the optional arguments required by the subroutine. This format exemplifies the generality of a good F3. It can be used to pass control to another F3 in any field of the computer. Very little extra machine language programming was necessary to increase the general utility of the F3 routine.

FUNCTIONS OF THE F3

Five different types of F3s have been developed, and they seem to encompass the major functions for which special-purpose subroutines are needed. The five functions are: (1) economizing on the amount of core required by programs for some state diagrams; (2) implementation of functions that can be notated in state diagrams but which have not yet been incorporated in the original software; (3) control of peripheral equipment of the computer from the state table; (4) programming of complex reinforcement contingencies that require on-line mathematical treatment of data for establishing the contingencies; and (5) analysis of data as it is acquired by the use of more complex sorting and other mathematical operations than is provided for in the RTS.

Examples of each of these categories may clarify the functions of F3s.

Efficiency

A common requirement of reinforcement schedules is the use of a prespecified set of variable intervals or variable ratios. The following state table is a program of VI reinforcement with six different variable intervals.

```
S.S.1,
S1,
      Z1------S2/Interval time out
S2,
      R1:ON1----S3/START REINFORCEMENT
S3,
      4":OFF1;Z2--S1/END REINFORCEMENT
S.S.2,
S1
      30":Z1----S2/FIRST INTERVAL IS 30"
S2,
      Z2------S3/WAIT HERE UNTIL REINF. IS OBTAINED
S3,
      15":Z1----S4/2ND INTERVAL
S4,
      Z2-------S5
S5,
      5":Z1----S6/3RD INTERVAL
S6,
      Z2------S7
S7,
      10":Z1-----S8/4TH INTERVAL
S8,
      Z2------S9
S9,
      25":Z1----S10/5TH INTERVAL
S10,
      Z2------S11
S11,
      20":Z1----S12/6TH INTERVAL
S12,
      Z2-------S1/RECYCLE INTERVALS
$
```

This program requires 357 decimal core locations per station. An F3 has been incorporated in the software to reduce the amount of space required for the variable list function. Each station using the F3 routine requires only N + 6 locations for the variable interval, where N is the number of intervals in the list. The subroutine to handle this function (LISTY) takes less than 64 locations to service all 10 stations, each using one or more variable lists. The format of this routine is slightly cumbersome. The actual format is

F3(031Ø1,–ARG1,N,–ARG1,–ARG2,–ARG3, . . . –ARGN,–N),

where 31ϕ1 is the octal address of LISTY (O indicates to the compiler that 31ϕ1 is an octal number), the arguments are the successive values of the list and N is the number of arguments in the list. The use of LISTY is exemplified in the following program, which serves the same function as the previous state table.

```
S.S.1,
S1,
      Z1--------S2/INTERVAL TIMES OUT
S2,
      R1:ON1----S3/START REINFORCEMENT
S3,
      4";OFF1;Z2---S1/END REINFORCEMENT
S.S.2,
S1,
      5":Z3;F3(03101,-6,6,-6,-3,-1,-2,-5,-4,-6);Z1---S2/5" UNIT
      Z3-----S1/THIS RESETS THE 5" CLOCK
S2,
      Z2---S1/START NEXT INTERVAL AFTER
           REINFORCEMENT
```

The state table that incorporates LISTY requires 157 decimal locations, thus saving 200 locations per station minus the 64 locations for LISTY. Of course, the savings would increase as the number of intervals increased.

Implementing Notatable Functions

Some straightforward notational formats have not been incorporated as a standard feature of SKED software due to complexity and anticipations of infrequent use. However, it is sometimes necessary to recover these functions in specific applications. An example of this type is the setting of a recording counter to a specified value. In state notation, this would simply be C5=10, for example. The SKED system would demand that Counter 5 (C5) be incremented 10 times to achieve this effect. Since it is often useful to set a particular recording counter by the state table to a specific number to indicate trial number when data is being generated on each trial, an F3 (CTRSET) has been written for this purpose. The format of this F3 is

F3(OCTRSET,ARG1,ARG2),

where ARG1 is the recording counter to be set to the value contained in ARG2. Another F3 has been developed to provide a function omitted from the software. The F1 function used to increment the value of a tagged location does not work properly on a stimulus output. This results from the arrangement of the stimulus word on a bit position scheme instead of as an octal number. If the F1 is used on a stimulus output, a binary incrementation results instead of the desired bit rotation. A set of F3s has been developed to handle this situation.

Control of Peripheral Devices

An example of an F3 for controlling peripheral devices is incorporated in the RTS for recording data on paper tape (through either the low- or high-speed paper-tape punch) under control of events in the state table. This F3 (DODUM) is simply called from the state table without arguments. It will then record the current data for the calling station. It also utilizes a data buffer for the counters so that new data may be acquired while the old data is being punched on the appropriate device. It transfers the original data to the buffer area for the calling station, and then zeros the recording counters immediately, so that new incoming data can be recorded. The data buffer is reserved by requesting twice the number of recording counters used by the state table during compilation.

Complex Contingencies

SKED has been written primarily to generate stimuli contingent either on elapsed time or on the emission of a prespecified number of responses after state entry. Although the notation usually is adequate for reinforcement schedules, some Es recently have become interested in the effects of consequating complex dimensions of the behavior stream which require mathematical analysis of performance to generate reinforcement. For example, an F3 has been written to reinforce fixed-interval behavior only when a S displays a good temporal discrimination, as measured by the quarter-life of the response pattern in the interval. The quarter-life is the percentage of the interval length at which point 25% of the total responses in that interval have been generated. To calculate the quarter-life, the entire response rate obtained in the interval must be obtained in successive subintervals of the fixed interval. The F3 establishes by arguments the first counter of the distribution, the number of counters in the distribution, the required value of the quarter-life for reinforcement, and whether the behavior should exceed or be less than the criterion. This F3 is called by a unit timer, usually 1% of the fixed interval. If the reinforcement criterion is not achieved once the interval has timed out, the distribution is shifted, erasing the first subinterval. A second F3 then is entered by each response to check for a switch that indicates whether the criterion has been met.

On-Line Manipulation of Data

F3s in this category resemble those in the previous category in that they perform complex data treatment during acquisition, but they do this only for recording purposes rather than for establishing contingencies. Although complete recording of the occurrence of each response in time is feasible with some sort of mass-storage device, so that further data analysis could then be conducted after the end of the experiment, most of us do not have the appropriate data storage devices. Therefore, it is often useful to summarize and sort data as it is being acquired to reduce the quantity of it and to permit rapid comprehension of trends in data as it is generated. One F3 of this sort records distributions, depending upon the length of a preceding IRT. This

routine can be used to assess sequential dependencies between successive IRTs and to relate response durations to the IRTs that precede them. Another F3 of this sort records all possible combinations of two events in a sequence of six occurrences of either event.

One SKED user (Frank Butler, personal communication, 1973) has developed a set of mathematical operations that can be applied to record counter values. This set of subroutines even permits the calculation of means and standard deviations on-line through the use of these operators. Unfortunately, this package was developed for a rather specific and unusual configuration of hardware, but the packages could be easily rewritten for use on most machines with only a minimum of effort.

LOADING F3s

Since the loader in the RTS tries to enter state tables into core unoccupied by the RTS monitor, it is necessary to protect any F3 subroutine from being overwritten by state tables. There are two methods for loading F3s. The first of these is to request space for the F3 by means of the dialogue when establishing the system. This serves well for F3s located in Field 0. For longer F3s, or for F3s that are to be used only infrequently, it is often more convenient to load the assembled F3 on top of the RTS. The required protection can be obtained by storing the F3 at the end of one of thefields of memory and by then adjusting one parameter in the RTS to protect this area.

Documentation of some subroutines within SKED, as well as of some parameters of the current state table that will be of use to programmers, has been prepared. Some differences exist between the 4K and the multiple field systems that must be taken into account when writing new F3s. A library and documentation of F3s has been prepared and is available from the SKED users' group.

Behavior Research Methods & Instrumentation
1974, Vol. 6, No. 2, 176-180

FOCAL, FORTRAN, and BASIC programs for reformatting and analyzing data collected by the SKED program*

ARTHUR SNAPPER, DENNIS LEE, LEONARD BURCZYK, and JOSE C. SIMOES-FONTES
Western Michigan University, Kalamazoo, Michigan 49001

Several programs have been written in the FOCAL, FORTRAN, and BASIC languages for reformatting and analyzing SKED data. These programs include selection and explicit labeling of sets of recording counters representing distributions and/or total counts of events, several general manipulations of distributional data, and standard statistical treatment of distributions.

In the behavioral sciences, the amount and complexity of data analysis varies widely, even within the same laboratory. Some experimental questions are amenable to simple comparisons of response rates. Other questions deal with the microanalysis of the behavior stream and require extensive manipulations of data. Most research falls between these extremes.

SKED was designed primarily for the purpose of data acquisition and experimental control. However, some simpler forms of data reduction are easily notated and programmed with the SKED software. The addition of software subroutines, of course, can provide sophisticated data treatment on-line. Limitations on the available amount of core storage and limitations on the time within which the system must respond to new inputs are the two major parameters of an on-line system. Limited core storage restricts the quantity of stored data and thus limits the amount of data that can be acquired. Temporal limitations, on the other hand, reduce the amount of mathematical operations that can be performed on data on-line if the system is to respond to inputs quickly enough. Otherwise, summary statistics might be obtained during the data collection period so that large quantities of data need not be retained in the computer. A balance between these two restraints is achieved by performing simple sorting and summing operations while acquiring data. More extensive data treatment is then left to further analysis off-line, either on the same computer or on a second system.

One other commonly used technique to obtain raw data without forcing it into a prespecified data reduction format is to store new data on an external medium such as magnetic tape or disk, so that minimal core is required for storage, and data analyses is not accomplished during the acquisition period. Even with this system, it is usually desirable to accumulate some minimal amount of data in core before recording it on an external device, since such devices are most efficient when they have fixed-length formats that demand a certain number of characters on each data transfer. Of course, with this scheme, it is necessary to process the data off-line by the

*This work was supported in part by Research Scientist Development Award K2-MH-70483 from the National Institute of Mental Health to the senior author.