ERL--A language for implementing evoked response and psychophysiological experiments

JAMES GIPS*, ADOLF PFEFFERBAUM**, and MONTE BUCHSBAUM†
National Institute of Mental Health, Bethesda, Maryland 20014

Evoked Response Language (ERL) is a high-level language for implementing psychophysiological experiments with emphasis on signal averaging techniques. It allows the E to specify, in English-like statements, a sequence of auditory and visual stimuli and to collect or average associated physiological data. The language can be learned quickly without the need for knowledge of machine language programming. Programs can be easily composed and promptly tested. ERL has been running in our laboratory on a classic LINC computer since the spring of 1970. Expanded versions of ERL are being planned for the PDP-12 and SEL 810B computers.

Most psychophysiological experiments require the presentation of sequences of visual or auditory stimuli and the simultaneous sampling of electrophysiological data. When experiments are implemented with hard-wired configurations of timers, relays, oscillators, tape recorders, and special-purpose electronic devices, it often takes weeks or months to set up a new experiment. Switching of hardware to change experimental conditions is often awkward and leads to errors. Further, once an investment in a large special-purpose system has been made, investigators often grow reluctant to change. An alternative to these cumbersome arrangements is the process-control computer which can time intervals, control switches, and present stimuli—in addition to recording physiological data and analyzing results (Uttal, 1968). As new experiments can be specified simply by writing new programs, the time for implementing new experiments is reduced to hours or days. The E, however, often must become proficient in the arcane art of machine language programming. These programs are generally lengthy and not self-explanatory; moreover, debugging process control programs is difficult. To counteract such problems, the use of high-level languages is the obvious next step. With a suitable language, experimental procedures can be implemented in minutes in an easy straightforward manner. Furthermore, the high-level language serves as a detailed documentation of the experimental procedure. A few such languages for specifying experiments now exist, e.g., PSYCHOL (McLean, 1969), a language for psychological experimentation, and SCAT (Stadler, 1969), a commercially available language often used for conditioning experiments.

Evoked Response Language (ERL), the language to be described here, is designed specifically for implementing average evoked response (AER) experiments but could be equally useful for other psychophysiological variables such as heart rate, GSR, etc. The novel features of ERL include direct specification of stimuli in a way natural to the E, high-level commands for signal averaging, and ease of acquisition of consecutive analogue data. Evoked responses are obtained by EEG averaging which is time-locked to the onset or cessation of a stimulus. ERL allows the E to specify the nature of the sequence of stimuli and associated AER. A feature of the ERL compiler system is that it allows the program to be immediately tested and the results displayed (see Fig. 1).

*Unit on Psychophysiology, Laboratory of Psychology, National Institute of Mental Health, Public Health Service, U.S. Department of Health, Education and Welfare, Bethesda, Maryland 20014.

**Adult Psychiatry Branch, National Institute of Mental Health, Public Health Service, U.S. Department of Health, Education and Welfare, Bethesda, Maryland 20014.

†National Institute of Mental Health, Building 10, Room 2N-315, Bethesda, Maryland 20014.

ENVIRONMENT

ERL currently is running on a "classic" LINC computer (Clark & Molner, 1965) with 2,048 12-bit words of memory, of which half can be used for programs. The LINC is the forerunner of the Spear Microlinc and the PDP-12 (Clayton, 1970). All programming was done in LINC assembly language, using the LAP6 assembler (Wilkes, 1970). The ERL compiler is written as an addition to the LAP6 system. LAP6 is used for all text editing and file handling; the ERL compiler is called from LAP6, using the "free" meta command. The compiler converts the current manuscript directly into executable binary code and stores this binary program in the LAP6 "current binary program area" of the tape. The binary program produced by the compiler can be run either on a stand-alone basis or through a special monitor system written for the LINC (Gips, Pfefferbaum, & Buchsbaum, in press).

AN ERL EXAMPLE

A program for the study of the effect of a preceding visual stimulus on an auditory evoked response illustrates the use of ERL. The experimental paradigm might be: flash a dim light; sound a tone, and take an evoked response; flash a bright light; sound the same tone, and take another evoked response. An ERL program for this experiment is given in Table 1. Lines 3-5 are declaration statements; the variables in ERL are AERs and stimuli. Lines 6-20 form the executable part of the program. The experiment will last 10 min (120 loops x 5 sec/loop), and its output will be two AERs of 250 points each. Each AER will be the sum of 120 trials. The

Table 1
Sample ERL Program With LAP6 Line Numbers

```
 1. Experiment contrast
 2. [This experiment tests the effect of the
    intensity of a preceding light on a tone
    AER
 3. AER A, B 250 steps of 2 msec
 4. Light L 500 msec
 5. Tone T 500 msec at 65 dB
 6. DO 120 times
 7. Present L at 10 fc
10. Delay 500 msec
11. Present T with A
12. Delay 1 sec
13. Present L at 250 fc
14. Delay 500 msec
15. Present T with B
16. Delay 1 sec
17. End DO
20. Write
21. End experiment
```

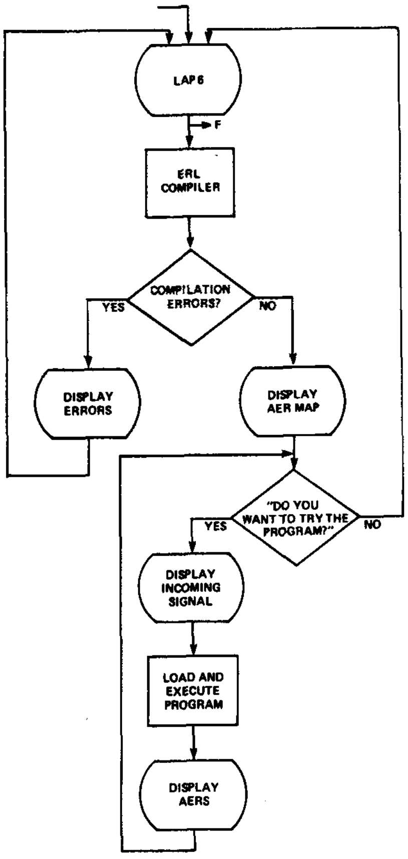

Fig. 1. Flowchart for ERL compiler system.

first AER will be auditory, preceded by a dim flash; the second will be the auditory AER with a preceding bright flash.

ERL SPECIFICATIONS

Every ERL program must begin with the statement: "EXPERIMENT name," where "name" is the name of the experiment, and end with the statement: "END EXPERIMENT." Comments are specified by beginning the line with a left bracket, as noted, for example, in Line 2 of Table 1. Each program has two parts: first, the declaration section; second, the execution section.

Declaration Statements

All AERs and stimuli must be declared at the beginning of the program. Each is given a one-letter name.

In an AER declaration statement, the number of steps (i.e., data points) in each AER and the time between each step must be specified (see Line 3, Table 1). Up to 16 different AERs are allowed, but the total storage space required may not exceed 1,024 data points. Each AER can be assigned a different duration and sampling rate by using multiple AER declaration statements. A technique in which the responses to a given stimulus type are divided randomly and averaged into two AERs, each with half of the trials, has been employed in our laboratory and elsewhere (Buchsbaum & Fedio, 1969). This has been made a feature of the language; by specifying "WITH RANDOM REPLICATES" at the end of the AER declaration statement, responses to individual stimuli are averaged into two separate AERs on a pseudorandom basis.

Three kinds of stimuli are allowed: tones, lights, and clicks (see Lines 4 and 5, Table 1). At the time of execution, each stimulus must have an intensity associated with it, and tones and lights must have a duration. Intensity is expressed in decibels (DB) or footcandles (FC); duration is expressed in MSEC or SEC. The duration and intensity of a stimulus may be assigned in either the declaration or execution section of the program.

Execution Statements

Repeating cycles of stimulus presentation and data collection are specified by DO loops, as noted in Lines 6 and 17 of Table 1. DO loops begin with the statement: "DO N TIMES," where N is an integer less than 1,024, and end with the statement: "END DO." The section of instructions between a DO and the corresponding END DO is executed N times. DO loops may be nested to a depth of eight.

Stimuli are administered by using the PRESENT statement, shown in Lines 7, 11, 13, and 15 of Table 1. The stimulus must have been previously declared; intensity and duration may be specified in this statement. An AER is taken time-locked to stimulus onset if the expression "WITH A," where A is an AER name, is included, as shown in Lines 11 and 15, Table 1.

Periods of time of no stimulus presentation are specified by using the DELAY statement shown in Lines 10, 12, 14, and 16 of Table 1. The amount of time of delay may be expressed in either MSEC or SEC. An AER is taken time-locked to the beginning of the delay if the expression "WITH A," where A is an AER name, is included. Prestimulus AERs and AERs to stimulus cessation are specified in this way.

The WRITE statement (Line 20, Table 1) causes the AERs to be saved in the next available file on the data tape. If WRITE AND CLEAR is specified, the AER data area is cleared; this is useful for single trial experiments or for the collection of physiological data such as GSR or heart rate where averaging is not required.

Error Checking

Extensive error checking is done by the compiler. If an error is found, the compiler skips to the next line of the ERL program and continues the compilation process, thus allowing multiple errors to be discovered. At the end of the compilation, the line number of each error and the associated error type is displayed, as shown in Fig. 1.

IMPLEMENTATION

The implementation of a real-time language on a small computer imposes severe restrictions of time and space. Stimulus generation and electrophysiological sampling in ERL programs are done on a millisecond basis. All averaging and internal bookkeeping must be done in time periods of less than a millisecond; the program must be ready for the next stroke of the millisecond clock. As there is no time to swap segments of program into core from DECtape, the entire process control program and all current data must be resident in the 2K of core.

Note that these restrictions apply to the compiled (object) program, not the compiler. There are no serious restrictions for the time of compilation of a program. The compiler can be made arbitrarily long by dividing it into 256 word segments and swapping in and out of core from the DECtape. The ERL compiler is about 3,000 words long and, because of tape shuffling, can take 30 sec or more to compile an ERL program.

The emphasis in the ERL implementation is in generating an extremely short, relatively efficient object code. The 2,048 words of core are allocated by the compiled program in the following way: 256 words for the experiment's main program, 256 words for stimulus and AER parameter tables, 256 words for standard stimulus generation subroutines, 256 words for a resident monitor system and alphanumeric identification, and 1,024 words for the AER data to be collected.

General-purpose subroutines for simultaneously presenting stimuli and sampling electrophysiological data are the heart of the compiled program. These subroutines are standard for all ERL programs. The main program is built around calls to these subroutines. The compiler uses the declaration statements to construct tables containing stimulus and AER parameter information used during execution of the compiled program. Each PRESENT or DELAY statement is compiled in the main object program as a separate call to one of the stimulus subroutines. A DELAY is considered a stimulus of zero intensity. Included in these subroutine calls are the appropriate stimulus and sampling parameters, e.g., the duration and intensity of the stimulus and the duration and rate of sampling (if any).

Stimulus Generation

The stimuli are generated entirely by the subroutines and the digital-to-analogue (D/A) converter; no external oscillators or pulse generators are used. For lights, a voltage corresponding to the intensity of the light is generated through the D/A converter, dc amplified, and supplied directly to the constant-current driver of an Iconix photostimulator. For tones, a 500-Hz square wave of the desired intensity is generated through the D/A converter, smoothed by a RC circuit, and fed directly to an audio amplifier. Each tone is begun and terminated with a program-generated 10-msec ramp to prevent speaker distortion.

Resident Monitor System

The compiled program can be run either on a stand-alone basis or through a monitor system (Gips et al, in press). The monitor system automates many of the tasks of the E. For example, upon normal termination of an ERL program, the program jumps to a location in the block reserved for the monitor system. When run on a stand-alone basis, the block will be empty and the machine halts. If the monitor system is present, a display program will be automatically read in from tape and the results displayed. The compiled program is dependent on the monitor system for information on data storage. When a WRITE command is reached, the compiled program expects to find the number of the first available data tape block in a certain location in storage. If there is no monitor system, and the E uses the ERL WRITE statement, the data will be written beginning in Tape Block 0.

Abort

Because it is often necessary to abort a given run, especially when testing hospital patients, an abort feature has been built into the language. The E is given a special button which is monitored each time a stimulus subroutine is entered. If the abort button has been pushed, the program jumps to a location in the resident monitor. If there is no monitor, the program halts with all collected data intact.

PLANNED IMPROVEMENTS

Because of the severe restrictions imposed by the LINC's small memory (2K core) and relatively slow cycle time (16 μsec for an ADD), desirable features had to be excluded from the language. For example, one would like to be able to easily specify pseudorandom time intervals and pseudorandom intensity levels. Features such as these are just not feasible on the LINC. Expanded versions of ERL are planned for the PDP-12 and SEL 810B computers.

REFERENCES

BUCHSBAUM, M., & FEDIO, P. Visual information and evoked responses from the left and right hemispheres. Electroencephalography & Clinical Neurophysiology, 1969, 26, 266-272.

CLARK, W. A., & MOLNAR, C. A description of the LINC. In R. W. Stacy and B. Waxman (Eds.), *Computers in biomedical research.* Vol. 2. New York: Academic Press, 1965. Pp. 35-66.

CLAYTON, R. J. Comparison of the LINC, LINC-8, and PDP-12 computers. Behavior Research Methods & Instrumentation, 1970, 2, 76.

GIPS, J., PFEFFERBAUM, D., & BUCHSBAUM, M. Use of a small process control computer in a psychophysiological laboratory. Psychophysiology, in press.

McLEAN, R. S. PSYCHOL: A computer language for experimentation. Behavior Research Methods & Instrumentation, 1969, 1, 323-328.

STADLER, S. On the varieties of computer experience. Behavior Research Methods & Instrumentation, 1969, 1, 267-269.

UTTAL, W. R. *Real-time computers—Technique and applications in the psychological sciences.* New York: Harper & Row, 1968.

WILKES, M. A. Conversational access to a 2,048-word machine. Communications of the ACM, 1970, 7, 407-414.