INSTRUMENTATION & TECHNIQUES

Interactive computer assistance for biofeedback and psychophysiological research: Pragmatics of development

PETER M. DOLAN and DAVID SHAPIRO

*Department of Psychiatry and Biobehavioral Sciences
University of California, Los Angeles, California 90024*

Key assumptions and decisions influencing the design, programming, and operation of an interactive computer-control system for a biofeedback research laboratory are discussed. Most interactions with the computer are in the form of simple English queries and answers. Laboratory activities for which the computer provides assistance include instrument calibration, time-critical system testing, experiment set-up and instrument hook-up, experiment control, data management, and analysis. Use of structured programming technique is described. Authors and laboratory staff conclude that interactive assistance is a helpful research tool in many situations and that future laboratory-control systems will have more extensive English language interactive capability.

This paper presents a brief description of a computer-based psychophysiology laboratory and discusses a number of practical matters that were considered and resolved during the course of setting up and operating the laboratory. As the commentary indicates, we do not recommend that everyone follow exactly the same procedures we used to build and develop the computer system, but some of the techniques, especially those for assessing performance, may be useful to current and prospective users of similar systems.

DESIGN CONSIDERATIONS

The single most important consideration in designing the laboratory (by consensus of faculty and the laboratory staff) was the requirement that the computer-control system was to be operated by staff and investigators who had little or no experience with computers. The computer, therefore, was to be used in two distinct ways. First, it would provide extensive English language interactions with the experimenter during set-up and calibrations of both analog and digital components of the laboratory. This arrangement was to take the form of interactive computer-generated queries provided to the experimenter or laboratory staff at appropriate times. Second, the computer was to provide management of the experiment in a start-to-finish sequence.

This work was supported in part by Office of Naval Research Contract N00014-75-C-0150, NR201-152. Address reprint requests to David Shapiro, PhD, Department of Psychiatry, University of California, Los Angeles, 760 Westwood Plaza, Los Angeles, California 90024.

Six areas of research support were identified: (1) instrument calibration, (2) time-critical testing of the overall system, (3) experiment set-up and instrument hook-up, (4) experiment control, (5) data acquisition and management, (6) preliminary analysis of data and data reduction.

Decisions regarding areas amenable to computer assistance emerged during the course of the 6-month set-up period for the laboratory.

HARDWARE/SOFTWARE SELECTION

A Digital Equipment Corporation (DEC) GT40 system was purchased with these components: (1) PDP-11/05-type processor with 28K memory (16-bit word), including twin graphic display units (1,000 × 1,000 matrix) with processor, (2) single 1.1-million-word disk cartridge unit, (3) DEC tape unit, (4) eight-channel laboratory peripheral system (programmable clock, eight-channel analog-to-digital converter, Schmitt triggers, etc.), (5) DEC writer console, (6) 80 bits of digital output, and (7) 16 bits of digital input.

DEC RT-11 software included the Version 1 monitor and RT-11 BASIC with laboratory peripheral system and graphic display extensions. These components were selected with the expectation that the speed of the PDP-11 and RT-11 monitor would allow development of experimental support software with a high-level language (BASIC) rather than in assembly language. Our initial expectation was that use of the high-level language would in turn save substantial software development time by allowing English language interactions to be interspersed with time-critical instrument-control routines. (Subse-

Figure 1. Overall system.

quent developments in microprocessors and firmware have made this use of higher level languages even more feasible.)

The following types of analog instruments are employed: Beckman eight-channel Dynagraph; associated and additional amplifiers, couplers, and transducers; Hewlett-Packard instrumentation recorder; oscilloscope; various stimulus displays; and multichannel audio system. Computer interfacing was made through an eight-channel input/output patch panel unit for analog signals. The interface arrangement that allowed independent operation of analog and digital components eventually proved to be extremely useful for checkout of both analog and digital components of the laboratory system. For example, the computer time base could be independently verified on the polygraph or the oscilloscope, and the digitized output of analog instrumentation could be used to check up on instrument functions.

A diagram of the overall system is shown in Figure 1.

INTERACTIVE SOFTWARE APPLICATIONS

Instrument Calibration

The most obvious and simplest first step for providing computer assistance was using the computer interactively during the course of calibrating analog instrumentation. Accordingly, an interlocked sequence of programs that provide English language prompts for the laboratory assistant were developed. The programs are sequential and, once started, will run to completion only if all analog channels meet linearity and alignment specifications. The calibration procedure is guided step by step by the programs, with standard remedial procedures defined. Figure 2 shows some of the interactive portions of this procedure.

An important result of using these programs is that the quality of laboratory system calibration is identical for all investigators, from senior level to novice. In addition, most equipment failures and hook-up errors are

identified during the course of regularly scheduled calibration procedures, rather than during experiments.

Time-Critical Test of Overall System

Most of our anticipated experimental uses required collection of one of two types of data: (1) timing of events such as heartbeats, pulse waves, Korotkoff sounds, subject responses, stimulus onset; and (2) periodic sampling of analog signals such as EMG, electrodermal responses, respiration, or other data. While the calibration assistance programs help protect against analog instrument and hook-up faults, a 3-min timing procedure (described in the next section) is used to assess timing accuracy and reliability.

Standardized System Control Software

Because investigators unfamiliar with computers were to conduct experiments in the laboratory, it was necessary to provide consistent, reliable, and secure system software. These requirements were met by the simple procedure of using separate disk cartridges for each experimenter and experiment. Each computer-controlled experiment is based on standard system software (typically the RT-11 monitor and BASIC) that has been found to be operationally error free. Software is copied from a master disk cartridge to one reserved for use by a single experimenter for a single experiment (typically 10 to 50 single subject sessions). Each experimenter's disk cartridge is a self-contained module with system software, all computer programs necessary for a particular experiment, and, eventually, all data files created during the experiment. Because all copies are made from the same master copy, system-level software faults are eliminated. Each experimenter has a known, standard system for control of all computer components and is protected from incursion by other experimenters into his/her own "computer work space." The removable cartridges later serve as permanent archives for all programs and data associated with a particular experiment. Thus, reanalysis

```

RUN
STEP01      BASIC V01-05

BEGIN CALIBRATION SEQUENCE.
WHAT IS CURRENT DATE AND TIME
?3

DYNAGRAPH 382 POWER SUPPLY MODULE:
HAS POWER SWITCH BEEN 'ON' FOR 15 MINUTES (Y/N)??
TURN POWER TO 'ON'.
IS POWER ON NOW (Y/N)?Y

SIGNAL PATCH PANEL:
SIG. OUT TO DIST. BUS JUMPERS CONNECTED (Y/N)?Y
OPTIONAL STEP: DIST BUS TO CHART AND/OR TAPE JUMPERS CONN. (Y/N)?Y
FOR THE RECORD-- JUMPERS CONNECTED TO:(CHART/TAPE/BOTH/NEITHER)?B
BECKMAN PRE-AMP 481 A/B:
GAIN SWITCHES OFF?Y
DYNAGRAPH AMP 482A:
ZERO SUPPRESSION CONTROL OFF (Y/N)?Y
BECKMAN POST-AMP 428 (REAR OF RACK) :
GAIN SWITCHES SET CORRECTLY (Y/N)?Y
MONITOR PANEL:
DO ALL METERS READ ZERO (Y/N)?Y
SIGNAL CALIBRATOR/REFERENCE:
IS REFERENCE METER SWITCH ON (Y/N)?Y
FUNCTION SWITCH TO CAL (Y/N)?Y
SIGNAL '0' TEST:
CHANNEL CALIBRATE SWITCH OFF (Y/N)?Y
TEST RESULTS:
CHANNEL 0 IN SPEC AT ZERO ( 2065.46 )
CHANNEL 1 IN SPEC AT ZERO ( 2054.88 )
*** PROBLEM *** CHANNEL 2 TOO LOW ( 1993.03 )
CHANNEL 3 IN SPEC AT ZERO ( 2063.99 )
CHANNEL 4 IN SPEC AT ZERO ( 2039.16 )
*** PROBLEM *** CHANNEL 5 TOO HIGH ( 2080.6 )
*** PROBLEM *** CHANNEL 6 TOO HIGH ( 2082.82 )
CHANNEL 7 IN SPEC AT ZERO ( 2060.48 )
*** NOTE THAT SOME CHANNELS NEED ADJUSTMENT.
MAKE ADJUSTMENTS AS NEEDED, THEN PRESS <CR> FOR RETEST

```

Figure 2. Example of interactive calibration procedures.

of data or replication of the entire experiment is possible at a later date. The archived disk cartridges also serve as program libraries for implementing similar experiments or variations on the original experiment. Since disk cartridges can be changed in a matter of minutes, it is possible for a variety of experimental efforts to be conducted in a single workday.

Insistence on proven reliability of system software typically forces our laboratory to wait 6-18 months after new software has been released by the manufacturer to insure delivery of a reliable system.

Experiment Set-Up and Instrument Hook-Up

Since experimental designs vary widely and the degree of technical expertise of experimenters differs, computer assistance with experiment set-up and hook-up of instruments is provided at varying levels. Typically, the computer gives a brief display of incoming data for the experimenter's inspection before an experimental session begins. For advanced experimenters, this verification may be as simple as an inspection of indicator lights and control switch options, along with a query, such as: "VERIFY: ALL INSTRUMENTS WORKING

PROPERLY (YES/NO)?" A "YES" response is required before continuing with the experiment. For less experienced experimenters, and for experiments with unique or complex instrumentation, further prompting, reminders, and hook-up instructions are provided.

Equipment Control

Computer control of experiment protocols is extensive. Typically, the experimenter's design is programmed in such a way that once it is initiated, each session is controlled by the computer in a start-to-finish sequence in which the experimenter or laboratory assistant interacts with the computer in English, in a purely supervisory capacity.

Most experiments consist of a series of trials during

which data are collected and appropriate feedback is delivered to the subject. Prior to data collection, instrument hook-up and functioning are verified by the experimenter and control parameters are specified. The computer collects and stores data, provides feedback, and otherwise manages the details of the experiment, providing periodic reports to the experimenter (usually during intertrial intervals).

Data Acquisition and Storage

Data collected during an experiment, along with appropriate labeling information, are permanently stored in files on the disk cartridge. Unique identification codes are used for each subject file, and the experimenter bears the responsibility for keeping records

RUN									
LOGGER BASIC V01B-02									
***** PRINTOUT OF EXPERIMENT LOG FILE *****									
LOG FILE NAME (6 CHARACTERS ONLY)?KEITH1									
LOG FILE NAME (6 CHARACTERS ONLY)?KEITH1									
SUBJ. ID= 1									
SESSION NO= 1									
MONTH/DAY/YEAR/TIME3/20/78/4:00									
FILE KEITH1 PREPARED									
SKIN COND. GAIN= 1									
SC ZERO SUPP= 1.18									
INITIAL CUFF PRESSURE= 124									
CYCLE LENGTH= 60 SECONDS.									
PRETRIAL LENGTH= 15									
EXPT HAS 30 CYCLES									
ELAPSED	OBS	HEART	K+/-	PRES	CH.	CH.	CH.		
TIME	NO.	RATE	TIME	SURE	#4	#5	#6		
>>> PRETRIAL DATA:									
702	1	85.47	0360	00	02543	02073	01269		
1392	2	86.95	0362	00	02504	02076	01271		
2075	3	87.84	0362	00	02594	02075	01253		
2760	4	87.59	0362	00	02621	02076	01233		
3449	5	87.08	0362	00	02618	02076	01233		
4154	6	85.10	0362	00	02517	02075	01246		
4871	7	83.68	0362	00	02372	02074	01258		
5612	8	80.97	0363	00	02487	02076	01267		
6352	9	81.08	0363	00	02479	02076	01274		
7081	10	82.30	0362	00	02480	02077	01274		
7795	11	84.03	0363	00	02537	02075	01269		
8500	12	85.10	0363	00	02549	02073	01270		
9207	13	84.86	0363	00	02522	02073	01275		
9909	14	85.47	0362	00	02513	02073	01281		
10605	15	86.20	0363	00	02582	02075	01286		
11273	16	89.82	0363	00	02627	02074	01289		
11931	17	91.18	0363	00	02656	02073	01292		
12585	18	91.74	0363	00	02625	02072	01293		
13228	19	93.31	0363	00	02586	02073	01295		
13889	20	90.77	0362	00	02582	02073	01299		
14564	21	88.88	0362	00	02643	02074	01302		
>>> TRIAL ON DATA:									
15251	22	87.33	0290	126	02279	02088	01230		
15994	23	80.75	0363	124	02599	02089	01256		
16739	24	80.53	0362	122	02730	02089	01279		
17464	25	82.75	0363	120	02672	02089	01282		
18161	26	86.08	0362	118	02520	02088	01293		
18829	27	89.82	0362	116	02759	02089	01303		
19487	28	91.18	0254	118	02513	02088	01302		
20127	29	93.75	0236	120	01871	02089	01290		
20752	30	96.00	0246	122	01923	02090	01281		
21371	31	96.93	0279	124	02146	02090	01274		
21992	32	96.61	0261	126	02479	02090	01274		
22635	33	93.31	0244	128	02493	02090	01282		
23287	34	92.02	0254	130	02677	02088	01291		
23974	35	87.33	0363	128	02776	02090	01267		
24654	36	88.23	0284	130	02572	02089	01237		
25343	37	87.08	0278	132	02431	02090	01241		

Figure 3. Excerpt of computer log during experiment.

of subject identification. Subject names or other personal information is never entered into the computer system.

Several types of records are maintained during the course of a computer-administered experiment. Since the system console is used for communication between the experimenter and the computer, the printed copy produced by the console serves as an experiment log book or laboratory notebook and can be used to determine that initial set-up procedures were correct. During the course of an experiment, periodic reports produced by the computer provide a positive indication to the experimenter that everything is normal. (Figure 3 shows an excerpt of a computer log generated during an experiment.) Finally, the digital data file produced for each subject usually contains sufficient data to reconstruct the sequence of events for which data were collected and stored. Duplicates of experimental data files are maintained on magnetic tape until publishable results are obtained, at which time the disk cartridge is used as a data archive.

Preliminary Analysis of Data/Data Reduction

Overall development strategy thus far has included minimum efforts to develop general-purpose data analysis routines, since widely available packages such as BMD and SPSS are adequate for most research. A series of "data-snooping" routines has been written for retrieval, display, and summary of subject data files using the graphic display unit. These routines are used for inductive analyses and to explore unexpected research results.¹ These analysis routines are intended to provide the experimenter with descriptive statistics sufficient to assess the ongoing condition of research. Since the computer system stores data for each subject on individual files, the experimenter may run preliminary analysis programs immediately after the experiment session has ended. The ability to obtain preliminary analyses for single or multiple subjects prior to completion of the entire research project helps the investigator to preview the effectiveness of the experiment protocol and to identify factors or problems that were not apparent at the time the design was formulated.

PROGRAMMING EXPERIMENT PROTOCOLS

As indicated, typical biofeedback experiments are based on experimenter-defined protocols written in the BASIC language running under RT-11 system software (currently V03). Note that software provided by the manufacturer usually consists of procedures oriented toward controlling the computer system and peripheral devices *per se* and is typically not intended to perform the substantive tasks desired by experimenters. This means that manufacturer-provided software should be regarded as a set of tools from which the experimenter can construct experiment protocols in the form of programs or algorithms. Translating an experiment protocol into a functional algorithm for computer

control is entirely up to the experimenter; computer equipment manufacturers do not usually provide this service.

A fundamental decision that must be made prior to experimentation is the level of computer system control software to be used in research.

Given an operating system of proven reliability, the next choice typically involves selecting one or more higher level languages for building algorithms to control the experiment and data. A number of commercially available products are suitable for this task, including (but not limited to) PASCAL, FORTRAN, and BASIC. Each of these alternatives has intrinsic advantages and disadvantages in terms of speed of program execution, ease in programming, amount of memory required, and other features. Without specific knowledge of the type of experimentation to be done and performance characteristics required of the computer and persons using the computer, it is difficult to determine which language would be most appropriate and/or most economical. A general guideline might be: the faster the response or data acquisition requirements of the research, the more primitive the programming language must be. Alternatively, faster computers allow the use of higher level programming languages. There is a direct tradeoff between ease of programming and operating speed of the computer for the simple reason that increased programming convenience usually means that the computer is doing more work and, hence, takes more time to function. The restrictions of this speed/complexity tradeoff will probably ease as more systems convert to multiple microprocessors for performing separate tasks.

Our experiences have demonstrated that the choice of one particular language over another seems to be less important than how the language is used to obtain desired research results. In this area of software choice, the best position seems to be a pragmatic one: (1) Select the simplest one that will do an adequate job, and (2) make sure that performance specifications are verified by testing before becoming involved with a particular system. (Any vendor should be willing to allow you to benchmark test a system before purchasing.)

Our initial experience with programming research protocols indicated that the desirable end result of our "start-to-finish" approach toward English language interactions was one or a series of programs that guided the researcher in equipment set-up and checkout prior to the experiment, administered the protocol, gathered data and stored them, and gave periodic reports to the experimenter on the overall status of the subject and computer system. In a typical experiment of 30 45-sec "trials," the computer processes several million data items and permanently stores 10,000-100,000 data items per subject. The overall program sequences usually require about 1 h of elapsed time, including computer-assisted set-up, "trials," intertrial intervals and status reports, and updating of data files.

Attempting to code, debug, and test 1-h supervisory

program administering a research protocol proved to be a difficult task. As a result of our early (time-consuming and frustrating) attempts, we now use a structured approach in programming experiment protocols. By structured programming, we mean that we use a series of algorithms (subroutines) that have been designed to perform various tasks during the course of conducting experiments. This repeated use of well-defined subroutines simplifies the programming and revision of an experiment protocol.

Almost all of our experiments are handled by the same overall program structure, shown in Figure 4. Most of the routines designated in Figure 4 are also themselves composed of subroutines. As is evident in Figure 4, with the exception of the first (set-up and test) routine, the entire experiment-control program can be described as a single loop. Entry into this control loop is made only through the set-up and test routine, and exit from the loop occurs when the experiment has been successfully completed. This logical control structure is machine independent and language independent.

Typical programming time for an experiment is 4-20 h. Typical development time for a well-defined experiment (includes programming, prototype testing, and overall testing) is 5 to 10 working days.

Implementing this overall algorithmic approach in the laboratory has the effect of imposing rather rigid requirements on the way in which research is carried out. A simple but understated characterization of this

effect is that each experimental session is handled the same way insofar as instrumentation hook-up and data collection are concerned. While this approach frees the experimenter from many of the mundane tasks necessary during an experiment, it is intolerant of unscheduled events and incorrectly connected instrumentation. For example, once the computer has been programmed to look for an EKG on Channel 1, the EKG must occur on Channel 1 or the experiment will not proceed. Similarly, if an electrode falls off, or an amplifier is set at the wrong sensitivity, or a required signal is missing, everything grinds to a halt.

As might be expected, there is a diversity of opinion among the laboratory staff on the relative merits of an "algorithmic approach" to conducting research. This topic is covered further in the concluding paragraphs.

DATA MANAGEMENT

Experimental data are handled in two distinct ways. The first, and most difficult to plan, is on-line data collection and storage during a time-critical experiment. This phase of data collection and storage inevitably requires the most staff time for programming and testing. The net result of this effort is production of a data file for each experimental session. This file contains all data saved for a single subject. The second manner of processing data involves providing various sorts of summaries from data stored on one or more subject files. Since producing summaries is not typically time critical, the design and production of summary programs is a straightforward programming task requiring testing for accuracy and completeness of data retrieval from files.

Although our particular hardware and software choices will allow about 16,000 data items to be stored in high-speed memory, we typically collect only a few thousand items during a "trial-on" period of an experiment; then during an intertrial period of 2 to 30 sec, we move the data to a file and update the experiment log with brief summaries of what happened during the trial. The diagram in Figure 5 shows a typical time-critical experiment protocol from the perspective of data management.

The data stored on files is usually a combination of raw data, such as EKG R-R interval times, and summaries of data collected on relatively slow-moving signals, such as skin conductance. The slower signals are sampled at some constant frequency, usually between 5 and 100 Hz, and averages are computed for each interval of timed data (e.g., for each R-R interval).

Postexperiment processing of data files most commonly takes the form of production of abstract or summary files, which are transmitted to larger computer systems for analysis with statistical utility packages, such as the BMD-P series. A typical summary of a single subject file would provide 5-sec averages of events (R-R intervals) or signal amplitudes in unit record for-

Figure 4. General structure of experiment control program.

Figure 5. Data collection and management during a typical experiment.

```

RUN
CRUNCH    BASIC V01B-02
HOW MANY FILES TO CRUNCH (UP TO 10)?1
INPUT THOSE NAMES
BP34LI.DAT
CREATE A SEQUENTIAL FILE FOR EACH SUBJECT (Y/N)?Y

DATA CRUNCH (VER. 7/18/78) FROM 'BP1' FOR:

*** BP34LI ***

*GROUP (1=INCREASE BP, 2=DECREASE)      RESP PEAK CRIT: 283
*ORDER (1= RIGHT LITE, 2= LEFT LITE)
**SUBJECT #
**TRIAL #
*EVENT (1=HP, 2=BP, 3=SK, 4=EMG,
5=HP PRETR AVG, TR AVG, PRETR MED, TR MED,
BP PRETR AVG, TR AVG, PRETR MED, TR MED,
SK PRETR AVG, TR AVG, EMG PRETR AVG, TR AVG
6=PRETR RESP: # RESP PERIODS, X INSP MS, S.D. INSP MS,
X INSP ADC, S.D. INSP ADC, X EXP MS, S.D. EXP MS,
X EXP ADC, S.D. EXP ADC, X PERIOD MS, S.D. PERIOD MS
7=TRIAL RESP: SAME AS PRETR)

5 SEC BLOCKS
*****
1234011  1057 1125 1147 1067 1123 1065 1028 1059 1023 1056 1060 981
1234012  125 127 124 123 129 124 128 124 123 123 121 126
1234013  1644 1658 1686 1721 1736 1774 1783 1817 1837 1855 1875 1896
1234014  1592 2202 1935 1554 1548 149 1232 0 798 1272 1918 1230
1234015  1064 1062 125 125 1775 1213
1234016
1234017  16 1981 524.4 587 88.6 1600 291.5 600 124.1 3575 382.1

1234021  959 1045 1035 1107 1165 1114 1144 1105 1072 1055 1125 1110
1234022  126 120 122 128 128 123 126 129 122 121 125 123
1234023  1165 1197 1220 1235 1258 1291 1321 1350 1386 1424 1451 1486
1234024  782 2079 1800 2086 2223 2314 2139 2078 2334 2335 2019 1864
1234025  1082 1085 124 125 1312 1974
1234026
1234027  12 33253097.5 513 126.5 1323 344.4 551 222.9 46583077.3

```

Figure 6. Portion of subject file summary (card image format).

mat, typically card images, which are easily processed by standard programs. Figure 6 shows a portion of such an abstract.

EVALUATING AND IMPROVING SYSTEM PERFORMANCE

As noted previously, when we refer to evaluation of system performance, we mean performance of the overall laboratory system, including both analog instrumentation and computer hardware and software. Within this context, evaluation is made of performance of the laboratory system as we intend to use it to perform a substantive research task. Since our general design goal is the simplest hardware/software combination that will do a specified job within fixed time and cost restraints, the configuration first proposed to handle a particular research chore is usually the simplest and quickest one that we think will work. A key point to remember is that under our current operational "ground rules," changes to the system are made only in response to specific substantive research objectives. The process of change and improvement of the system is an evolutionary one, as can be seen in the following example.

A primary task in our research is on-line timing of human heart rates and delivery of feedback based on rate or heart period (interbeat interval). Heart rate is defined as the rate in beats per minute (bpm), computed on the basis of the interval between successive R-wave segments of a cross-body electrocardiogram signal. When the laboratory was completed, this task was the first research priority. Since timing accuracy prior to this point was based on the cardiometer, our initial accuracy specification was ± 1 bpm. Our first efforts at heart period timing and feedback with the computer were based on a simple program that detected the R-wave, read the elapsed time from the previous R-wave, computed the heart rate, and delivered feedback. Software available at the time delivered ± 4 -msec accuracy, which became our operational standard. Six months later, a new version of system software permitted ± 2 -msec precision. Additional instruments and alternative programming techniques currently allow R-R timing accuracy of ± 0.5 msec, which is our present operational standard. We could raise this standard to ± 0.05 msec by altering the programmable clock speed, but we have not yet done so.

There are a variety of testing tasks that are critical to acceptable performance within substantive research tasks. Perhaps the most fundamental of these is a test for time stability in a working algorithm. When a program is running in a "loop," for example, one that is designed to recognize a particular signal or to collect data at specified time intervals, it is important to know at what speed the program scans input and if there are any time irregularities in the working program. When such program loops are implemented in binary code, there is very little risk that algorithmic cycle time will vary signifi-

cantly. If, however, such programs are written in higher level languages or run under a monitor or supervisory program, there is the possibility that internal functions of the system supervisory program might cause the cycle time or the program to vary.

A simple way to provide benchmark timing for the operational program is to insert timing pulses (for example, to a TTL-compatible digital output) before and after time-critical operations. For example, if the program should be scanning a given input channel every 5 msec, the algorithm might be:

```

    → Read Channel n
    → Wait 5 msec
  
```

To verify that this is happening, insert a marking pulse before and after the scheduled read operation:

```

    → Pulse on
      Read Channel n
      Pulse off
    → Wait 5 msec
  
```

An oscilloscope or polygraph can be used to monitor the "pulse-on" and "pulse-off" times.

The benchmark pulse technique also provides a way to physically time segments of operational programs. Once again, primitive binary program loops should not show significant time variability, but timing variability may occur in operational subroutines within multilevel software systems, for example: (1) real-time monitor and high-level language and experiment-control program; (2) operating system and assembly language routines; (3) any foreground/background, timesharing, or "cycle-stealing" programming arrangement. To obtain execution speed timing information on a program segment, first do benchmark timing of a "pulse-on/pulse-off" loop. This establishes how long it takes to turn the pulse on and off. Then insert a "pulse on" at the beginning of the program segment to be timed and a "pulse off" at the end of the segment. Overall pulse-on/pulse-off time minus the time it takes to turn the pulse on will give a reasonably accurate characterization of execution time. If the pulse-on/off output is to a polygraph, any irregularities in execution time will be recorded.

The above method is routinely used to provide performance information on computer programs related to both substantive and procedural operations, such as measuring disk-head movements during file access operations, execution timing of single lines of code, program overlay processes, subroutine execution times, and system response times.

Since most vendors of software packages will allow testing and evaluation by prospective purchasers, bench-

Figure 7. Simulated cardiac cycle at 150 bpm (2.5-Hz ramp signal with four intercepts).

mark timings can be used to evaluate system software critical to research before the software is purchased and before programming resources are committed. A common use of the technique in the laboratory is for comparison of new software versions before the new software is used in research. Performance testing prior to research has saved our laboratory considerable time and money.

In addition to benchmark timings, we find it useful to perform multichannel timing tests for the accuracy and repeatability of measurements made by both analog and digital equipment. The following procedure is our most frequently used multichannel test.

Since our most frequently used observational unit is the cardiac cycle, we use a ramp-shaped 2.5-Hz signal across all input channels simultaneously. This effectively simulates a heart rate of 150 bpm. Level detectors are set to intercept the ramp signal at points approximating the time relationships (for example) of the R-wave, brachial pulse, radial pulse, and Korotkoff sound. Figure 7 shows a ramp signal with typical intercept levels on a time scale.

Output pulses generated by the level detectors are input to the computer digital input register (DIR), which, in turn, causes a programmable crystal-controlled clock to be read. Each input pulse to the DIR generates a computer interrupt, which causes the elapsed time on the clock counter to be stored in memory along with the DIR image. A program loop scans the incoming data and branches to appropriate service routines, depending on which (simulated) event appears in the DIR.

Besides providing information on the accuracy of time measurements, this procedure is useful in testing the overall accuracy of logic within an experiment management program. By connecting the output of computer-controlled feedback devices to a polygraph, feedback latency (computer processing time) can be assessed.

The test procedure outlined above can be used for long-term stability tests of analog and digital equipment by programming the computer to make frequency

counts of DIR images and elapsed times read from the clock. The cumulative summary of elapsed times counted should match elapsed "wall clock" time for the duration of the test. Although we have occasionally performed 1-h tests of this sort on our system in order to find system faults occurring at irregular time intervals, a 3-min test is usually sufficient.

Assessing stability of the overall (analog plus digital) system is currently done on an ad hoc basis. Computer-assisted calibration (described above) is performed periodically, insuring that overall system performance is nominal at the time of calibration. If, prior to research, it appears that some irregularity or instability exists in the system, a procedure such as the one described below is used to investigate.

First, the system is set up as it would normally be used during research, including all analog and digital components. Sensing equipment such as strain gauges or pressure transducers is left in place and amplification equipment is set to a 0-V baseline. Electrodes are not attached.

A simple data acquisition program (annotated version shown in Figure 8A) is run, and the results are printed (Figure 8B). In its present state, our A/D system has a voltage input range of +2.5 to -2.5 that is converted to a digital value in the range 0-4,095, with 2,048 = 0 V. Thus Line 4 of the program (Figure 8A) merely increments a counter of A/D point deviation from the nominal input of 0 V (2,048 on the 0-4,095 digital scale). The resulting printout (Figure 8B) in this instance shows slight spiking on Channel 3, which might have resulted from a pre-amp failure.

Occasionally, an on-line display of incoming data via the graphic unit is done for the purpose of assisting the researcher in electrode placement. The procedure is similar to the above test, with the results of one or two channels of A/D data displayed on the graphic unit (1,000 X 1000 dot matrix CRT) with a sample rate of about 100 Hz. Since the graphic device used for biofeedback is located immediately in front of the subject, it is easy for the researcher to assess the effect of electrode

```

LIST
TEST3      BASIC V01B-02
1 DIM D(4095)\$REM --- RESERVE SPACE FOR 4095 A/D VALUES
2 FOR K=1 TO 5000\$REM --- BEGIN 5000 CYCLE LOOP
3 CALL "ADC"(3,X)\$REM --- SAMPLE CHANNEL 3, STORE IN VARIABLE X
4 D(K)=D(K)+1\$REM --- INCREMENT COUNTER FOR THIS SAMPLE VALUE
5 NEXT K\$REM --- END LOOP
6 PRINT "RESULT OF 5000 SAMPLE TEST ON CHANNEL 3:"
7 PRINT "A/D VALUE", "FREQUENCY"
8 FOR K=1 TO 4095\$REM --- BEGIN LOOP TO CHECK ALL POSSIBLE A/D VALUES
9 IF D(K)=0 THEN 11\$REM --- SKIP A/D VALUES WITH FREQUENCY=0
10 PRINT K, D(K)
11 NEXT K\$REM --- END LOOP
12 END

READY

```

A

```

RUN
TEST3      BASIC V01B-02
RESULT OF 5000 SAMPLE TEST ON CHANNEL 3:
A/D VALUE  FREQUENCY
2041         4
2042        118
2043       1092
2044        645
2045        459
2046        361
2047         76
2048       1081
2049        175
2050          3
2051          2
2052          2
2053          7
2054          1
2055          6
2056          2
2057          2
2058          1
2059          9
2060         12
2061         44
2062         19
2063         29
2064          4
2065         38
2066          7
2067        100
2068         57
2069        188
2070        153
2071        222
2072         49
2073         31
2074          1

READY

```

B

Figure 8. (A) Simple data acquisition program. (B) Result of test procedure: A/D values with frequencies for test period (3 min, approximately).

placement or of using various sensing devices by reference to the on-line display.

COMMENTS

Several levels of evaluation are necessary to characterize the reaction of staff and investigators to the operational strengths and weaknesses of the current system. These factors include effectiveness of the overall system as a research tool, effects on productivity of the labora-

tory, and influences of the system on the way in which research is conducted.

Overall, the interactive system appears to be an effective research tool, primarily because of the standardization of calibration and experiment management procedures. Given appropriate computer-generated experiment log entries, the experimenter is able to verify the timely occurrence of every event during an experiment, as well as to detect the presence of unscheduled events. When the laboratory is engaged in a series of

several related investigations, the computer system is most effective. When, however, several unrelated experiments are being developed simultaneously, or in the case of an isolated or unique type of experiment, the debugging and testing required for computer management does not always provide significant advantages over manual procedures.

Contrary to our initial expectations, the contribution of the computer-assisted control system to overall productivity of the laboratory appears to be qualitative rather than quantitative. While we are able to rapidly carry out complex experiment protocols repeatedly and with a high degree of standardization, our lag time from project inception to published result does not seem to have been affected by the system. It seems likely, however, that less laboratory staffing is necessary to supply the careful supervision that step-by-step computer assistance can provide. As was the case before using the interactive system, those investigators with well-defined protocols move swiftly from inception to result; those with vaguely defined protocols require more time and have more difficulty in the laboratory.²

The interactive system has influenced the conduct of research in several ways. First, staff and investigators have developed a dependence on computer-generated log entries to supervise the progress of an experimental session.³ First versions of status reports quickly graduated from vague messages such as "Cycle No. 1 complete" to multifunction reports of simple statistical abstracts, such as pre- and posttrial heart rate and blood pressure means, modes, and standard deviations. While the first versions of computer log entries were regarded as helpful adjuncts, we now regard them as necessary positive indications that everything is going smoothly.

The capacity to intelligently scan incoming data has become increasingly important in our experimentation. Computer messages regarding status have expanded to the extent that some protocols now rely on automatic computer adjustment of selected equipment (e.g., blood pressure cuff) and automatic assessment and adjustment of subject performance vis-à-vis performance criteria for heart rate and blood pressure control.⁴ The computer system has made it possible to easily design and implement multiple-feedback research (i.e., providing simultaneous feedback on more than one function). We have not yet systematically explored this area.

The computer system has had a noticeable effect on the manner in which research is designed and implemented. The capacity for "instant analysis" allows us to formulate and test tentative experimental designs and protocols in order to eliminate unworkable versions, or to empirically determine information that is otherwise unpublished. For example, trial time, subject reactions to various stimuli, and conditioning effects of computer-generated stimuli can be quickly determined by using prototype programs and can be used to increase the effectiveness of an experiment design before full-scale research begins.

In order to program a computer-managed protocol, it

is necessary to carefully and precisely define a number of time and data management parameters. This has the effect of forcing the specification of a set of control parameters and assumptions at an early stage of research. The transition from theoretical formulation to operational protocol is facilitated by this process, in the sense that a given research program must take specific form to be prototype tested.

FUTURE DIRECTIONS

The present system is treated by laboratory staff as a research tool that has the ability to communicate in simple English. The volume of English communications for control purposes has gradually increased and will no doubt continue to grow. As computer peripheral equipment increases in operating speed and decreases in cost, this trend will probably continue. Since the computer system can be programmed for English communications, a probable end result of this trend will be a standard English language vocabulary for set-up and management of experiments and data manipulation.

A significant burden has been placed on the programming staff by the increasing use of simple English as a communications medium. Once staff and investigators perceived that the computer-control system accepted limited English communications, that became an accepted operational standard, and reverting to numeric codes or more primitive interactions became very difficult. We are presently working on developing a first-version vocabulary for an English language command set for controlling the system. It seems reasonable to assume that such a standardized vocabulary will be an integral part of future laboratory systems and that the instructions will be in some permanent storage medium, such as read-only memory, along with appropriate operating software.

Although our present ability to control the system with a limited English language vocabulary carries the implication of standardization of commands and syntax requirements, we are still in the early stages of developing uniform external characteristics. We hope to implement standard control vocabularies in future systems. At the present time, interactions are programmed to adequately describe and perform a series of particular tasks rather than to satisfy a uniform syntax. Although laboratory staff and investigators readily agree on such task descriptions, it is difficult to reach a semantic consensus for a more general control vocabulary.

NOTES

1. These analysis routines were designed for use on the system described herein. For further information, contact the authors.

2. Subsequent to completion of this paper, a radically different type of computer-managed experiment was successfully tried. A series of approximately 12 separate (but related) research protocols were linked in a continuous 3-h computer-controlled "superprotocol." Procedurally, this technique of "piggybacking" related experiments allowed us to explore 12

operationally defined working hypotheses simultaneously. First interpretations of our costs of programming and protocol administration indicate that our overhead was increased two to three times over that of a single experiment, indicating the potential for a 6:1 or 4:1 increase in performance vs. cost, and this has begun to affect the manner in which we schedule and use our laboratory.

3. Occasionally, this situation results in a humorous but serious situation; for example, when a novice experimenter watches the computer rather than the subject, the subject might fall asleep, only to be awakened by a stimulus signal.

4. The potential and actuality of automated software-controlled adjustments of experimental equipment used in conjunction with human subjects has resulted in the need to develop a critical set of "fail-safe" standards for overall hardware/software performance. Although interesting, this topic is too broad to be discussed here.

(Received for publication February 26, 1981;
revision accepted May 27, 1981.)