PROGRAM ABSTRACTS/ALGORITHMS

Computer-controlled experiments: The case for FORTRAN

LESTER H. GILBERT

*University of Cape Town,
Rondebosch 7700, Republic of South Africa*

Many articles and reports have been written on the details of desirable software for on-line real-time control of experiments. Certain minimum functions need to be programmed in assembler, and certain time-critical experiments may also need to be written in assembler. Thereafter, however, a wide variety of control languages has been proposed for use in the psychological laboratory, including ACT (Millenson, 1971), APCOL (Pinkus & Gregg, 1973), CAN-4 (Durell, 1972), DTES (Spear, Overgard, & Christian, 1975), EMPP (Burkhardt, 1975), FOCLAB (Mullen, 1976), HECL (Doll, 1972), PACER (Henry, 1976), PEPL (Getty, 1975), PROSS (Scholz, 1973), PSYCHOL (McLean, 1969), PSYCLE (Creelman, 1971), SCAT (Polson, 1973), SIMPLE (Aaronson & Brauth, 1972), SKED (Snapper, 1975), and SP-12 (Pitz, 1975). Some experimenters have, on the other hand, directed their efforts toward the use of already existing computer languages as vehicles for experimental control: BASIC, FOCAL (e.g., Link, 1975), and FORTRAN (Haber, Barry, & Uhlman, 1970; Kaplan, 1977).

In general, formal terms, a special-purpose control language is better suited to its domain of convenience than is a general-purpose language. The presumed deficiencies of a general-purpose language such as FORTRAN have been the motivating factors behind the creation of many of the languages mentioned above.

There is, in practical terms, very little to be gained, and perhaps much effort to be wasted, in the creation of a special-purpose process-control language when a FORTRAN compiler is already available (which includes the COMMON and CALL statements). My remarks are primarily directed toward the single-user computer system; timeshared, multiuser, or multisubject systems do require special operating systems. Even in these situations, however, modification of the following techniques, along the lines of Kaplan (1977), allows the use of FORTRAN with no loss of adequacy.

This study was supported by a grant from the University of Cape Town to P. D. du Preez and formed part of a thesis submitted to the University of Cape Town by the author for the degree of Master of Science. Requests for reprints should be sent to the author, Department of Psychology, University of Cape Town, Private Bag, Rondebosch 7700, Republic of South Africa. An expanded treatment of this topic, with examples, is available from the author.

Programming in FORTRAN

The computer system on which the following procedures were developed comprises a Hewlett-Packard 2114B minicomputer with 8K main memory (16-bit word length), a paper-tape reader, a paper-tape punch, an ASR 33 Teletypewriter, and an ADDS Consul 520 visual display unit. The process-control hardware comprises a clock (a time-base generator, TBG), a reed-relay output, and a switch-sense input.

In the interests of readability, certain of the syntactic features of FORTRAN are ignored in this paper. In particular, all variables are of type INTEGER.

Definition of Variables. The FORTRAN main program uses three variables: TIME, RELAY, and SWITCH. TIME contains the current time; its value is incremented by 1 every 10 msec by the assembler routine, TIMER, which is described below. TIME is thus a counter of hundredths of a second; its maximum value is 327.67 sec. RELAY contains the desired pattern of reed-relay settings. Each bit in the word corresponds to a relay such that 0 = off and 1 = on. For example, to turn Relay 3 on, the 3rd bit of RELAY must be 1; this is achieved by the statement RELAY = 4 in the program (decimal 4 = 100 in binary). SWITCH contains the pattern of the switch settings in the laboratory. Each bit of SWITCH is 0 or 1, depending upon the off/on state of the corresponding switch. For example, a response key might be connected to Switch 5; when closed, SWITCH will assume a value of 16 (decimal 16 = 10,000 in binary). The use of these variables and the techniques of programming on-line experiments are illustrated in the following example.

Example. A program fragment that will measure the reaction time of a subject to a light may be constructed as follows. Assuming that no other switch closures are expected except that of the response key, and that the light is connected to Relay 3, the reaction time (RT) may be found as follows.

	RELAY = 4	Select Relay 3.
	TIME = 0	Set counter to zero.
130	CONTINUE	Wait for the switch closure.
	IF (SWITCH) 130, 130, 140	
140	RT = TIME	Immediately capture value of TIME.
	WRITE (2, 13) RT	
13	FORMAT (" RT = ", I5)	

Use and Advantages. Two statements are required at the start of the FORTRAN program. These are explained below.

```
COMMON TIME, RELAY, SWITCH
CALL TIMER
```

ASMB,R,B		
NAM TIMER		
ENT TIMER		
COM TIME,RELAY,SWITCH		
TIMER NOP		The initiator section entry point.
ISZ TIMER		Establish the return address to the main program
LDA MS10		Obtain the 10 ms timing interval and set the clock.
OTA TBG		
STC TBG,C		Start the clock
LDA IJSB		Establish the interrupt linkage.
STA TBG		Store the pointer to the interrupt handler
JMP TIMER,I		Return to the calling program.
ORB		The interrupt linkage. A pointer is established to the interrupt handler, and is automatically activated when the clock interrupts.
IJSB JSB LINK,I		
LINK DEF CONT		
ORR		
CONT NOP		Interrupt handler entry point.
STC TBG,C		Start the clock immediately.
STA SAVEA		Save the contents of the A register.
LDA RELAY		Obtain the on/off pattern of relay settings, and transmit this to the reed relay interface.
OTA ROUT		
LIA SWIN		Obtain the on/off pattern of the switches, and store.
STA SWITCH		
ISZ TIME		Increment the value of the time counter. Guard against a zero value.
NOP		
LDA SAVEA		Restore the content of the A register.
JMP CONT,I		Return to the interrupted program.
TBG EQU 10B		The clock is located in I/O slot 10, highest priority. The other interface boards are located as shown.
ROUT EQU 15B		
SWIN EQU 14B		
MS10 OCT 2		
SAVEA BSS 1		
END		

Figure 1. Program listing of the assembler routine TIMER.

The advantages of the techniques illustrated in the above example are that total control over the laboratory environment may be achieved through the use of only three variables, and this control may be exercised entirely within the FORTRAN program to virtually any degree of complexity. The user does not need special function calls or transfer of control to other subroutines, and the mechanism by which the process-control functions are carried out is entirely transparent.

Mechanism of Process-Control Functions

The three process-control functions are controlled by a transparent assembler program, which, once written, does not need modification for every FORTRAN program. The assembler routine, called **TIMER**, is activated by the hardware clock every 10 msec via the interrupt system. The clock or TBG generates an interrupt, and control is immediately passed to **TIMER** at the end of every 10-msec time period. **TIMER** then performs the three functions: The value of **TIME** is incremented by 1, the data word in **RELAY** is transmitted to the reed-relay output interface board, and the data word from the switch-sense input interface board is placed in **SWITCH**. **TIMER** then returns control to wherever the interrupt occurred.

This mechanism is quite independent of any other I/O that might occur and of the I/O software that is used by the FORTRAN program. The requirements are simply that (1) a hardware clock be able to (2) generate an I/O interrupt that is (3) vectored with (4) the highest priority to the **TIMER** interrupt handler, where (5) the state of the switches in the environment may be sensed, and (6) the state of the relays may be modified.

The Hewlett-Packard programmable TBG provides intervals of 100 microsec, 1 msec, 10 msec, . . . , up to 1,000 sec; such flexibility is useful but not essential to the main concepts involved. The 10-msec interval has been chosen as the most useful for most applications; it involves an overhead of about 50 microsec in the **TIMER** interrupt handler, less than 1% of available computer time.

The **TIMER Routine.** The program listing of the routine is given in Figure 1. The **TIMER** communicates with the FORTRAN main program via the **COMMON** statement, where the addresses of **TIME**, **RELAY**, and **SWITCH** are established. The routine is called only once at the start of the main program; the initiator section of **TIMER** then starts the clock running, sets up the interrupt linkage so that the clock interrupt will be properly vectored to the interrupt handler section, and returns control to the FORTRAN program, where no further action need be taken. The interrupt handler section is then activated every 10 msec: The clock is restarted to time the next 10-msec interval,

the contents of **RELAY** are transmitted to the reed-relay output interface, the switch-sense input interface is interrogated and its data word is placed in **SWITCH**, and the value of **TIME** is incremented by 1.

REFERENCES

- AARONSON, D., & BRAUTH, S. SIMPLE guidelines for developing a computer-based laboratory. *Behavior Research Methods & Instrumentation*, 1972, **4**, 257-264.
- BURKHARDT, K. S. EMPP: An extensible multiprogramming system for experimental psychology. *Behavior Research Methods & Instrumentation*, 1975, **8**, 239-244.
- CREEFMAN, D. D. Rapid response and flexible experimental control with a small on-line computer: PSYCLE. *Behavior Research Methods & Instrumentation*, 1971, **3**, 265-267.
- DOLL, T. J. A 4k computer language for experimentation with human subjects. *Behavior Research Methods & Instrumentation*, 1972, **4**, 27-31.
- DURELL, A. B. Painless computer-controlled experimentation. *Behavior Research Methods & Instrumentation*, 1972, **4**, 165-166.
- GETTY, D. J. The PEPL system for control of experiments by a PDP-8 computer. *Behavior Research Methods & Instrumentation*, 1975, **7**, 131-136.
- HABER, R. N., BARRY, S. H., & UHLMAN, T. On-line FORTRAN for the PDP-8. *Behavior Research Methods & Instrumentation*, 1970, **2**, 248-252.
- HENRY, R. B. Implementation of experimental procedures in the PACER system of on-line control. *Behavior Research Methods & Instrumentation*, 1976, **8**, 508-512.
- KAPLAN, H. L. Clock-driven FORTRAN task scheduling in a multiprogramming environment. *Behavior Research Methods & Instrumentation*, 1977, **9**, 176-183.
- LINK, S. W. 4k laboratory FOCAL. *Behavior Research Methods & Instrumentation*, 1975, **7**, 137-142.
- MCLEAN, R. S. PSYCHOL: A computer language for experimentation. *Behavior Research Methods & Instrumentation*, 1969, **1**, 323-328.
- MILLENSON, J. R. A programming language for on-line control of psychological experiments. *Behavioural Science*, 1971, **16**, 248-256.
- MULLEN, S. FOCLAB: A language for computer-controlled psychology research. *Behavior Research Methods & Instrumentation*, 1976, **8**, 229-232.
- PINKUS, A. L., & GREGG, L. W. APCOL: A programming system for computer-controlled psychological laboratories. *Behavior Research Methods & Instrumentation*, 1973, **5**, 165-172.
- PITZ, G. F. Building a programming language for a small computer: Reinventing the wheel. *Behavior Research Methods & Instrumentation*, 1975, **7**, 42-46.
- POLSON, P. G. SCAT: Design criteria. *Behavior Research Methods & Instrumentation*, 1973, **5**, 242-244.
- SCHOLZ, K. W. PROSS: A processing control programming language. *Behavior Research Methods & Instrumentation*, 1973, **5**, 245-247.
- SNAPPER, A. G. A new OS/8 SKED. *Behavior Research Methods & Instrumentation*, 1975, **7**, 233-238.
- SPEAR, T. L., OVERGARD, D., & CHRISTIAN, T. W. The CLIPR display terminal experiment system. *Behavior Research Methods & Instrumentation*, 1975, **7**, 107-112.

(Accepted for publication July 16, 1979.)