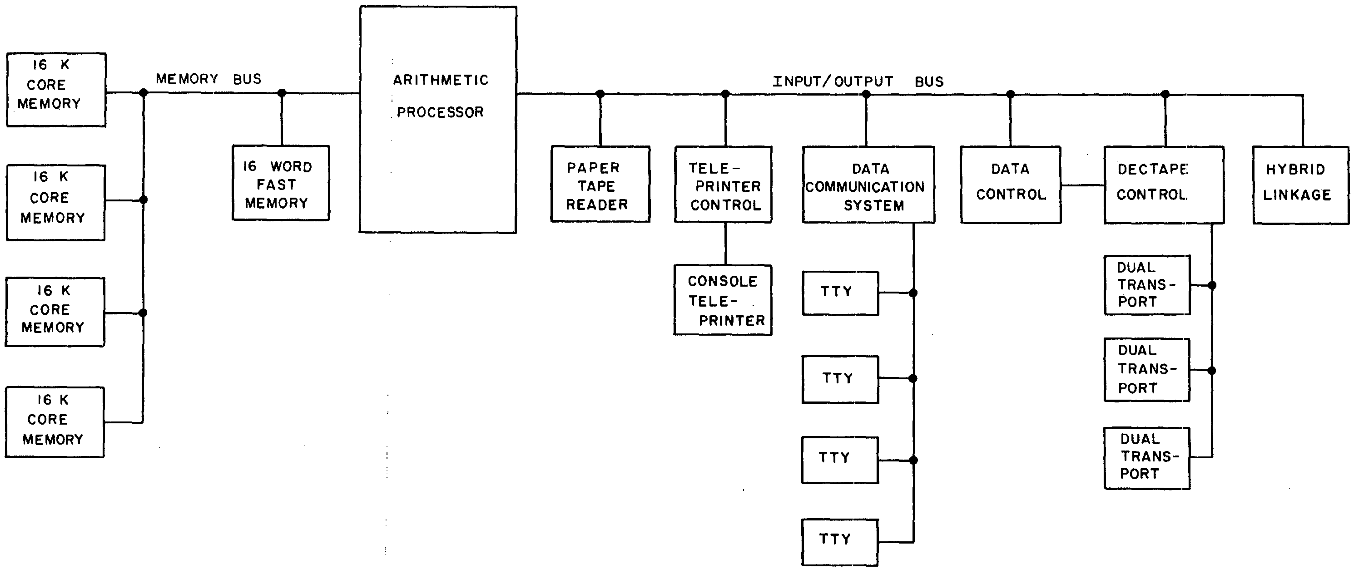

Figure 1. Schematic of digital computer.

reduction in large problem checkout time. Finally, initial and operating costs are low enough so that relatively long periods of time at one of the consoles are still economically feasible.

Basic to this digital system is a 6000-word executive routine which provides overall coordination and control of the total operating system. Additional software packages available include an editor, FORTRAN II (FORTRAN IV is currently under development), a macro-assembler, a relocating linking loader, a desk calculator and a symbolic debugging program.

The present hybrid linkage system contains a Raytheon Multiverter which combines a 48-channel multiplexer with a 14-bit analog to digital converter. Total conversion time is under 40 microseconds. The system also contains 40 Adage Model 4W13 digital to analog converters. These units feature ± 128 volts output and 14-bit accuracy.

Since the system was designed as a multi-user system, it was felt that all users should be able to address particular D/A's or A/D's beginning with address 0. This was accomplished by designing address relocation hardware for the linkage system. Basically, relocation constants are added to A/D and D/A addresses. These constants are predetermined numbers that are functions of how many D/A's and A/D's are assigned to other users. Protection hardware which prevents inadvertant addressing of converters assigned to another user was also designed into the system.

The remainder of this paper is concerned with a detailed description of the hybrid linkage system and modifications to the PDP-6 monitor system that were required to make time-shared, time-dependent computing possible.

THE HYBRID LINKAGE

The possibility of having several hybrid simulations in progress simultaneously on one digital computer creates a need for a rather special hybrid linkage. Over and above the requirements that each analog computer have the appropriate number of accurate, reliable and fast converters are the special requirements imposed by the use of one digital computer to service several analog computers. The solution to this problem was found by using a pool of linkage equipment which could be distributed easily and effectively. The analog to digital converter channels and the digital to analog converters are distributed to the four analog consoles by a patch board located within the PDP-6 hybrid linkage as shown in Fig. 2.

Special I/O Instructions and Analog Assignment

A PDP-6 user cannot normally execute any input/output instructions. Input and output are performed by the monitor on request from a user. Devices must be assigned to a user before he can expect to use them. In order to make the use of

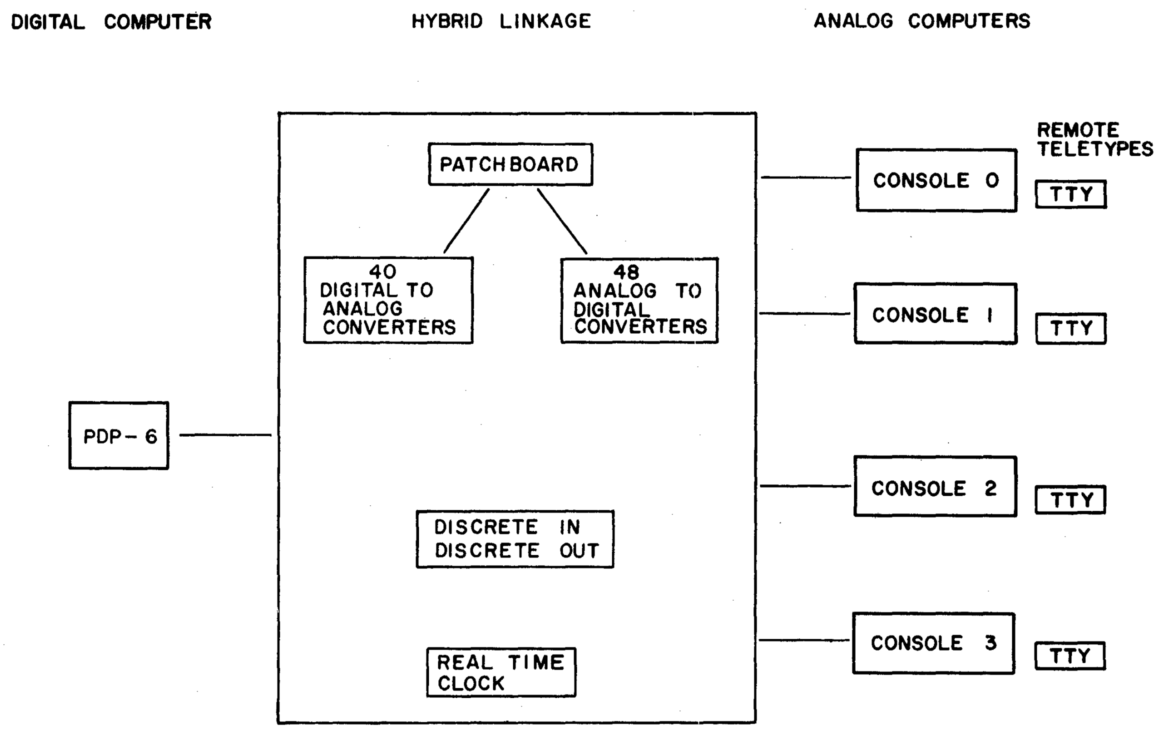

Figure 2. Hybrid linkage.

hybrid equipment as simple and as fast as possible, the in-out instructions on the PDP-6 were divided into two classes; the normal in-out instructions which are illegal for a user and the special instructions which are legal for users. Most instructions for the hybrid linkage are legal for all users.

One user cannot, however, affect another user's hybrid equipment. This protection is accomplished by both software and hardware features. Once a user has been assigned a specific analog console he can use only that portion of the linkage connected to that analog console. The monitor turns on the correct portion of the linkage whenever a job is selected for running. The monitor can actually turn on any one of five analog consoles. These are consoles NONE, 0, 1, 2 and 3. Console NONE is turned on for all users not assigned a specific analog console. A user cannot change analog console assignment except via the monitor commands.

Patch Board Wiring and Converter Addressing

Once set up the hybrid linkage allows each programmer, both digital and analog, to address analog to digital converters and digital to analog converters starting at address 0. Each analog program will use as many A/D's and D/A's as is needed.

The corresponding digital program will address the converters by identical addresses. This addressing scheme applies to all "four" hybrid computers. A particular program will not change from day to day as the converters are distributed to users in different arrangements. Of course the total number of converters is limited. This addressing scheme also protects each user.

In order to properly distribute analog to digital converters and digital to analog converters to the analog computers the hybrid linkage must be "set up." This requires a patch board to be wired and inserted in the patch bay at the hybrid linkage. The number of converters used on each console must also be entered in a toggle switch register. The linkage hardware will then automatically relocate converter addresses so that each user starts addressing with number 0.

Since this addressing is perhaps the most unusual feature of this hybrid linkage a more detailed description is in order. The outputs of all digital to analog converters appear on the patch board in the hybrid linkage. The analog computer patch panel positions that are to be used for D/A outputs are also wired to this patch board. The D/A outputs must be patched to the analog computers in an orderly way. The D/A outputs are used starting at number 0. The required number of converters for console 0 are wired in order. The next consecutive group of D/A converter outputs are wired to analog console 1, starting with position 0 on console 1. The next group is wired to console 2 and the final group to console 3. The number of converters delegated to each analog console is then entered on the toggle switch register. The analog to digital converters are distributed in the same manner.

Figure 3 is a block diagram of the address calculating hardware. The six-bit address comes directly from the programmer's in-out instruction. The console number is selected by the time-shared monitor. The numbers N_0, N_1, N_2 and N_3 are set in the toggle switch registers and correspond to the number of components of this type assigned to the corresponding analog console.

Consider an example. Let console 2 be selected and $N_0 = 10$, $N_1 = 10$ and $N_2 = 4$. The program addresses converter number 3. The output of the subtractor signals an illegal address when the result of the calculation is zero or negative. $4 - 3$ is positive and therefore does not signal an illegal address. The addition circuit will add 3 to $N_0 + N_1$ and will therefore calculate 23. If the patch board wiring is examined at this point, converter number 23 will be found wired to patch panel position 3 on console 2.

As a second example, suppose console NONE is selected and the programmer addresses channel 0. The subtractor signals an illegal address and the converter action is inhibited. Any illegal converter reference would result in this same action.

Summary of Linkage Instructions

In this section an outline of the use of each component in the hybrid linkage is given. This is done without going into a detailed description of the instruction or data formats. These instructions are exactly the same for any of the "four" hybrid computers.

Two instructions are used with the digital to analog converter system. One instruction is used to select a converter address. The other instruction transfers data to the selected converter. Each data transfer instruction will increment the address by one. Consecutive data transfers will load consecutive converters.

Three instructions are associated with the analog to digital converting system. One loads the converter address and starts the converter, another transfers data to the computer. The third instruction is used as a skip instruction for timing purposes. This instruction will skip when conversion is complete. As the data is transferred into the computer the converter address is automatically incremented by one and the next conversion is started.

Discrete inputs can be read from each analog console into the user's digital program. Discrete outputs can be set at each analog console from the user's program. This requires two instructions; one to read and one to set. There are 12 of these discrete inputs and outputs permanently assigned to each analog console.

The hybrid linkage contains a millisecond clock that is used mainly by the real-time monitor for all timing control. Any user can, however, read the clock. This is useful in computing times for program execution within one computing time interval.

TIME-SHARING SOFTWARE

Time-dependent problem solutions using the digital computer combined with dynamic devices, as in hybrid simulation, require execution of the digital program within a repeated, fixed time interval while the problem is operating, i.e., compute mode, in analog terminology. In previous hybrid computation at United Aircraft Research Laboratories the repeated digital program cycle and the execution time were the same. In many instances, the repetition rate, or duty cycle, need not be as short as execution time. With a longer duty cycle and/or a faster computer, a portion of the duty cycle can be shared with program service activities. By minimizing the ratio of time-dependent program, execution time to an acceptable duty cycle, a number of these programs can coexist within a common duty cycle.

Time-Sharing Control

Software control is provided to accomplish time sharing by the PDP-6 Multiprogramming System.[4] This system includes a resident monitor which handles input/output service and schedules user program execution. Using the PDP-6 priority interrupt channels, the monitor sequences programs based on delayed input/output requests, clocked timing, and user's programmed or teletyped commands. As shown in Fig. 4, service is provided, in order, to input/output device requests, monitor service requests from user software or teletype, and a round-robin sequencing of user programs.

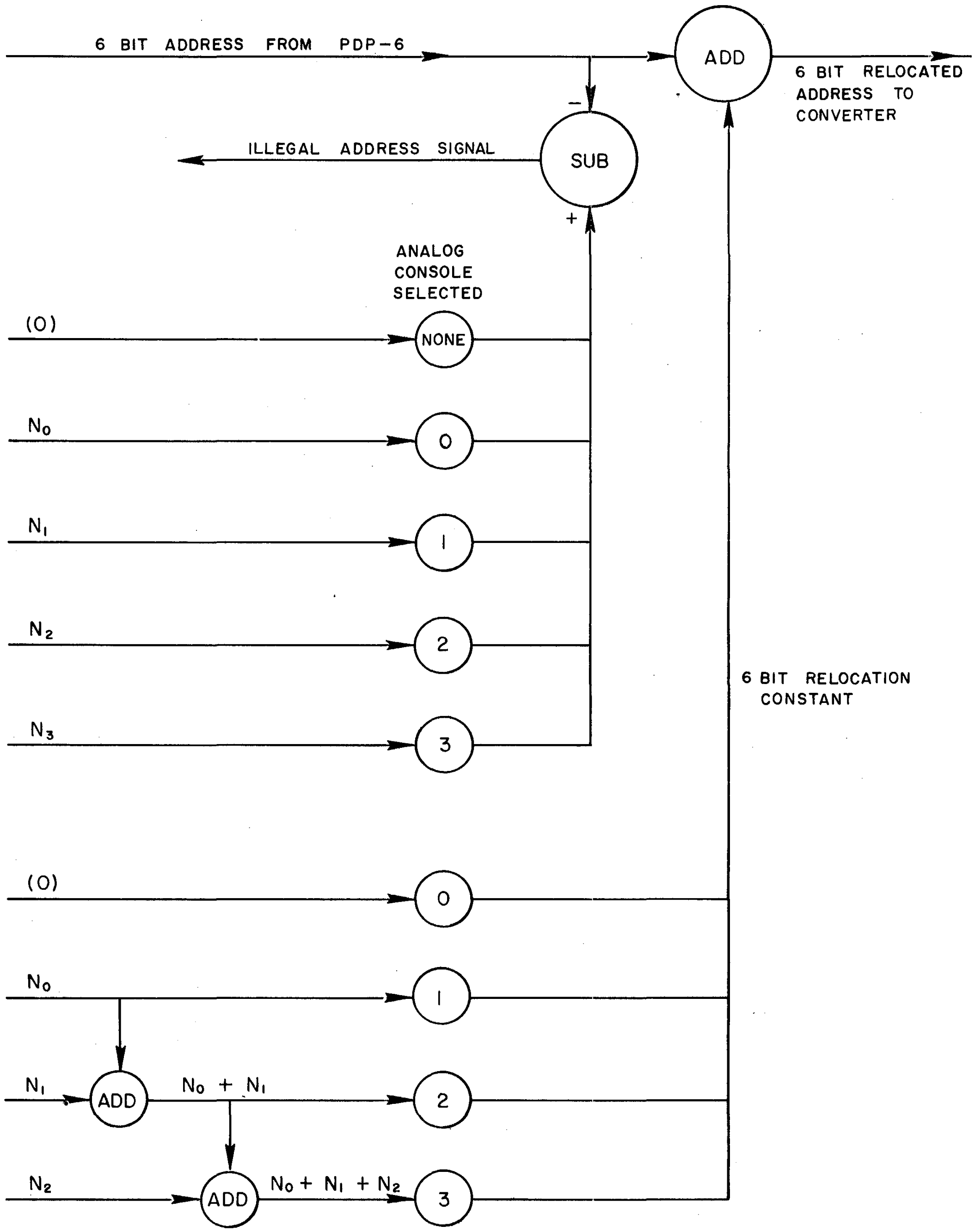

Figure 3. Converter addressing.

Programs in the round-robin include the users software as well as library routines called by the user such as editor, FORTRAN, etc.

Exact, repeated interval control is not a function of the basic monitor. Clock control is used to terminate programs in order to prevent long operation of a single user without looking at the service requirements of others. However, clock control is at low priority and is often deferred for monitor or input/output functions.

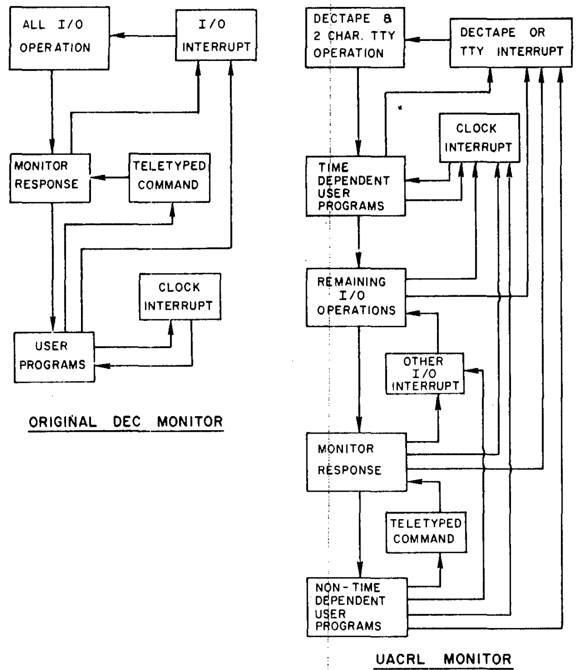

Figure 4. Monitor priority.

Exact timing of time-dependent users has been designed into the monitor using the UACRL clock at high priority. The time-dependent programs are sequenced within an accurate, repeated time interval (the duty cycle) preset to a desired value in milliseconds. Each time-dependent user receives a preassigned position and time interval within the duty cycle.

Dectape and teletype reading and writing operations can interrupt time-dependent service. Any of the routines used at this higher priority is completed well within one millisecond. Their combined use over the duty cycle should never exceed 10% of the duty cycle.

The current time-sharing at UACRL permits up to 95% of the established duty cycle for time-dependent use. Time-dependent users are commutated within the duty cycle in a queue separate from the round-robin of non-time-dependent users. The remaining 5% of the duty cycle is reserved for monitor functions and the round-robin. While the percentage for non-time-dependent use seems small, time-dependent users often operate in compute mode for reasonably short, infrequent intervals. When compute mode does not exist, most of this user's time is returned to the system for lower priority service.

In order to implement this form of time-sharing, a number of new features were added to the monitor. These functions, described below and summarized in Table 1, include control of:

1. Duty cycle
2. User time
3. Time synchronization
4. Hybrid linkage assignment

The last is provided for the assignment of proper linkage hardware to each user.

Duty Cycle

The duty cycle can be set by a teletyped command TSET x, where x is the desired duty cycle in milliseconds. This command is accepted by the monitor only when all time-dependent jobs are absent since the duty cycle affects all time-dependent users. If a user attempts to set the duty cycle when it is protected, he receives a typed return indicating the number of a job still in time-dependent status and the original duty cycle is retained.

A command TCLEAR can be typed by a resident time-dependent user to permit a duty cycle change while retaining his original memory and equipment. He must reinstate his timed operation by a TJOB command, described below, once the new duty cycle is set.

One command, TREAD, can be entered from teletype or from the user's program to access timing information. As a teletype entry, TREAD will yield a printout of the duty cycle, time-dependent job numbers and their reserved time intervals, and time left for additional reservation (balance of 0.95 duty cycle in the present system).

The use of TREAD as a programmed operator results in the transfer of the duty cycle value to a designated machine accumulator. This information is useful in calculating data affected by the time interval as a problem initializing operation.

User Time

With one time-dependent user in the system, execution of the program at the start of each duty cycle would provide accurate cycling. Program execution time varies due to data values and program branching and interruptions, but accurate time cycling of one user is possible by converting analog information at the start of the cycled program.

With more than one time-dependent user, the second and later users must be assured of similar

Table 1. Summary of Commands Added to the Monitor

Influences	Source	Command	Action	Errors	Teletype Response
Duty Cycle	Teletype	TSET x	Set duty cycle to x in milliseconds if none of the jobs are currently in time-dependent status.	x >8191 or x = 0	x MSEC CAN'T BE SET
				Duty cycle protected by existing time-dependent job.	j IS A TD JOB
				None	Carriage return/line feed
	Teletype	TCLEAR	Release duty cycle protection for initiating job.	None	Carriage return/line feed
	Teletype	TREAD	Type value of duty cycle, reserved user intervals, and remaining time available for time-dependent use.	None	n MSEC DUTY CYCLE TIME JOB NUMBER m_1 j_1 ⋮ ⋮ m_L j_L x MSEC AVAIL
	Program	TREAD	Transfer value of repetition time to specified accumulator.	None	None
User Time	Teletype	TJOB y	Establish a reserved time interval, y, in milliseconds, if available, and set duty cycle protection for the initiating job.	y > (0.95 Duty Cycle less sum of existing reserved intervals)	z MSEC AVAIL
				None	Carriage return/line feed
	Teletype	TJOB 0	Release the execution time interval (or set zero time) for the initiating job. Duty cycle protection is set.	None	Carriage return/line feed
Time Syncronization	Program	TSYNC	Return control to the monitor for job switching and signal the execution of the succeeding instructions in the initiating job when its next execution interval begins.	Execution when job does not have non-zero reserved time interval.	TSYNC FROM NON TD JOB
				TSYNC instruction not executed before clock interrupts time-dependent job.	TD JOB TIMED OUT
				None	None
Hybrid Linkage	Teletype	ENABLE ANAx	Reserve analog console x linkage for the initiating job, if available. The physical hardware assignment exists only during this job's operation. Only one console may be assigned to a job.	Device previously enabled by another job.	ANAx ENABLED FOR JOB j
				Nonexistent console number.	NO SUCH DEVICE
				Illegal device name	DEVICE CAN'T BE ENABLED
				None	Carriage return/line feed
	Teletype	DISABLE ANAx or DISABLE	Release analog assignment for the initiating job.	None - note, wrong address is ignored and any console that was assigned is released.	Carriage return/line feed

cyclic scheduling. This is accomplished by requiring all time-dependent users to establish a reserved interval within which they must complete their operation. Any time not needed by the assigned user, within his time interval, is allotted to the queues of non-time-dependent operations.

Time-dependent programs operate at high priority following each interrupt for successive reserved intervals until all job intervals are exhausted. The remainder of the duty cycle is used for non-time-dependent job functions. If all job operations are exhausted, the remainder of the duty cycle is spent in counting this dead time (the null job). The null job data is a useful indicator of computer utilization.

Figure 5 illustrates the subdivision of the duty cycle. The monitor schedules all these operations within separate queues with high priority given to the time-cycled queue. The information supplied to the clock comes from a ring table in sequence. This table contains the reserved job intervals and the balance of the duty cycle. Thus, with two time-dependent users active, the executive sequences three time values to the clock which determine the interrupts. When the executive responds to an interrupt, it sets the clock interval for the next operation. This will always occur within the first clock count after interrupt was signaled. Should the interrupts occur during execution of a time-dependent program, the job is reduced to round-robin operation and is informed of the error via teletype.

To establish an interval for the user a command TJOB *y* is typed into the system, where *y* is the interval in milliseconds to be reserved for the initiating job. A time slot is then entered at the end of the time-cycled queue for the user. The time requested should represent maximum program execution time plus a buffer time for higher priority input/output and program swapping. This buffer is recommended as 10% of execution time or one millisecond, whichever is largest.

The sum of reserved intervals must be less than

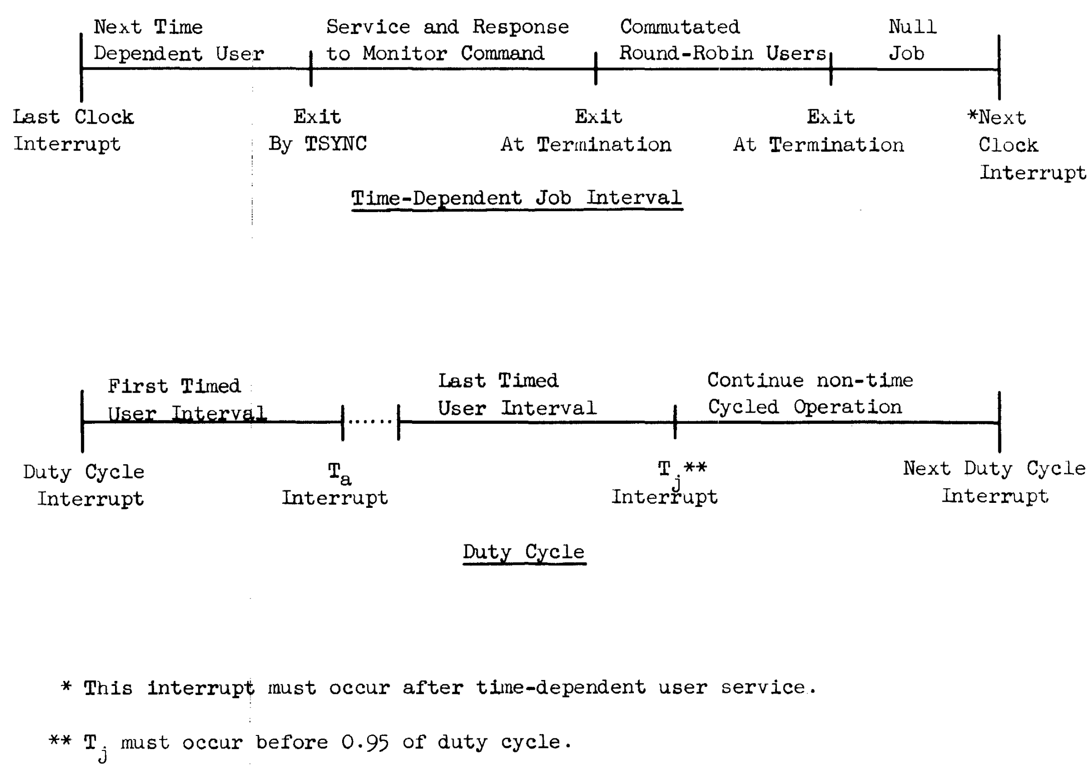

Figure 5. **Time sequences. *This interrupt must occur after time-dependent user service. **T_j must occur before 0.95 of duty cycle.**

0.95 of the duty cycle. Should an excessive interval be specified, an error response will be typed and the user denied entry to the time-cycled queue.

A command TJOB O can be used to release the user's reserved interval but retains protection of the duty cycle. This allows for additional sharing of cyclic operation by two or more cooperating users. Since they require time cycling only during active analog linked computing in simulation, an often small percentage of total use of the computer, users may successfully alternate time-dependent and round-robin operation.

Time Synchronization

While the TJOB command reserves an interval for the user, it does not actually enter his program into cyclic operation. A programmed operator TSYNC must be executed within the user's program to accomplish the transfer. Prior to execution of this instruction, the reserved time interval was used for non-time-dependent functions.

The TSYNC command will result in an error return if it is encountered without a reserved interval being previously set. This includes the condition of zero time reserved with TJOB 0.

When TSYNC is correctly executed, the user's operation is terminated for the remainder of the current duty cycle whether he was operating in time-cycled queue or round-robin. When this user's interval is entered during the next duty cycle, the program begins execution at the instruction following the TSYNC. Thus, the TSYNC command provides for initiating and maintaining time-cycled operation. Termination of timed operation occurs when nonhybrid input/output is attempted or the user signals an exit from program branch or teletype.

While input/output service results in termination of cyclic operation, this monitor controlled service proceeds legally at its lower priority. Automatic return to cyclic execution is possible if the user provides a TSYNC after completing the input/output programmed operator sequence (Table 1).

Hybrid Linkage Assignment

The monitor provides for user assignment and protection of hybrid linkage as provided for any other input/output device. The teletyped command, ENABLE ANAx, allows a user to reserve analog console x for his needs. Once assigned to a specific user, only that job is permitted to access the actual equipment. Only one console can be assigned to a job. A subsequent analog ENABLE, calling for a different console, will result in the release of the prior console as well as enabling the newly specified console.

Hardware assignment of the hybrid linkage takes place whenever the job is selected for operation regardless of which queue contains the job. In this way, round-robin operated debugging can include analog equipment.

The reserved equipment can be released by a corresponding DISABLE ANAx, or just DISABLE, as well as the job canceling KJOB.[4]

CURRENT STATUS

The entire system which includes the digital computer as delivered by Digital Equipment Corporation, the hybrid linkage as designed and assembled by United Aircraft, and the modifications to the standard PDP-6 monitor as implemented by United Aircraft has been operating satisfactorily as described since February 1, 1966.

REFERENCES

1. R. Belluardo, R. E. Gocht and G. A. Paquette, "The Hybrid Computation Facility at United Aircraft Corporation Research Laboratories," *Proceedings*, DECUS (Digital Equipment Computer Users Society), 1963, Maynard, Mass., pp. 261–269 (1964).

2. G. A. Paquette, "Progress of Hybrid Computation at United Aircraft Research Laboratories" *Proceedings*, 1964 Fall Joint Computer Conference.

3. "Programmed Data Processor—6 Handbook," (F-65), Digital Equipment Corporation, Maynard, Mass.

4. "PDP-6 Multiprogramming System Manual," (ACTOO), Digital Equipment Corporation, Maynard, Mass.

HYBRID SIMULATION OF A FREE PISTON ENGINE

R. E. Gagne
Mechanical Engineering Division, National Research Council
Ottawa, Ontario, Canada
and
E. J. Wright
Free Piston Development Company
Kingston, Ontario, Canada

INTRODUCTION

The free piston engine principle is one which has intrigued mechanical engineers for decades. Indeed, the original gas engine of Otto and Langden employed a piston assembly which did not contain the now conventional connecting rod and camshaft arrangement[1] and hence may be considered as an early implementation of the free piston principle. Today, however, the free piston principle is normally applied to a highly supercharged two-stroke compression ignition engine in which two opposed reciprocating pistons are used in a diesel cylinder. These pistons do not transmit mechanical energy but instead pneumatic energy is delivered in the form of high-pressure, high-temperature exhaust gas to a separate power turbine to obtain shaft power.

Such an engine has great performance potential in that it combines the superior thermal efficiency of the diesel cycle and the flexibility of a free turbine drive with a basic mechanical design simplicity in which highly stressed rotating members are absent. Further, unlike the conventional engine arrangement, complete dynamic balance of the pistons is obtained so that the engine is vibrationless, thus eliminating any need for complicated and expensive vibration isolation mountings.

Unfortunately, however, the lack of any mechanical connection to the pistons, other than a simple rack and pinion system to maintain an out-of-phase relationship between the pistons, presents a piston control problem which does not exist in a crankshaft engine. This problem is one of ensuring that the pistons do not exceed certain mechanical design limits irrespective of engine loading. It was principally to investigate various piston control schemes that this simulation of a specific engine was undertaken. The ability to implement any control action without the expense of tooling special components and without the risk of mechanical damage has shown the hybrid simulation to be a valuable design tool.

A number of attempts have been made to simulate the free piston engine in the past[2-4] from which we concluded that neither the analog nor digital computer simulations alone provided the most suitable means of handling the problem. It was felt that the ability of the analog to handle the system dynamics and to allow intimate operator contact with the problem should be teamed with the digital computer's ability to perform the thermodynamic calculations involved; and this hybrid approach seemed a most reasonable way to proceed.

In this discussion we will describe our hybrid