DYNAMIC SUPERVISORS - THEIR DESIGN AND CONSTRUCTION

Mr. D.H.R.Huxtable. Mr. M.T.Warwick. English Electric Computers Ltd., Kidsgrove, England.

The paper demonstrates the technology necessary to bring the facilities of Supervisor construction and modification to the level at which a user can, without a great deal of research and analysis modify his installation's Operating System. The Supervisor is seen to be a set of processes linked by a formalised control mechanism.

1. Basic Structure of a Supervisor

Basically a Supervisor provides processes which are executed in response to requests by users. In the simplest possible situation, where the machine is 'uniprogrammed' and the processes requested are completed before returning to the users program, all that is required is a standard entry process from a user job to Supervisor; consisting of parameter setting and a branch and link to a standard location. This is shown in Fig.1.

If however the Supervisor process involves waiting for an independent autonomous activity (e.g. I/O transfer) to finish, this simple approach is evidently wasteful. The solution is to provide two processes, one to initiate the transfer and one to complete any actions necessary on its termination, provided a signal is available to indicate that the activity has finished, so that the termination process can be entered (an interrupt). Schematically the procedure may be represented as shown in Fig.2.

This introduces the concept of interrupt, and the machine can be regarded as having two processor states, one user state, and one entered on machine interrupt. Even in this simple situation it is seen that the capability of having more than one autonomous activity results in the Supervisor acting as a mechanism for analysing interrupts and entering the appropriate processes. The addition of multiprogramming merely adds to the number of processes which may be executed and introduces the concept of priority which must exist in a Supervisor. The concept of priority implies that processes may not necessarily be 'run to completion', that they may have to be interrupted to allow higher priority processes to be executed. This may be represented as follows when activity 2 has a higher priority than activity 1. It assumes that Supervisor activities are themselves interruptable. This is shown in Fig.3.

Equally, lower priority processes may have to be queued to wait the completion of a higher priority activity. Whilst this concept of priority can be applied to nearly all Supervisor functions there is a small residue of processes which by their nature must not be interrupted, e.g. those which control the sequencing of processes, or those which are time dependant (e.g. pocket select on MICR devices). The class also includes processes to copy the contents of machine condition registers into data areas before further machine interruption can be allowed, (e.g. Secondary I/O Status bytes).

A considerable simplification of logic occurs if the interrupt analysis and process selection of functions of Supervisor are separated from the processes themselves. The machine action diagrams now are represented as 3 levels with the highest level controlling the priority selection of the activities. (Figure 4 shows the operation of a user program requiring two activities where activity 2 is of higher priority than activity 1. The action on the reversing of the order of call is also illustrated). Separation of the functions allows the processes to be written as independant procedures sequenced and multiprogrammed together with the user programs by the analysis and selection "Kernel" or core. The distinction between user programs and Supervisor processes becomes a matter of process priority and privilege.

Summary and Definitions and an example

A Supervisor has two logically distinct functions:

(a) Analysis and Selection.
(b) Activities executed as a result of the analysis and selection.

The Kernel of the Supervisor has four distinct actions:

1. The analysis of interrupts.
2. Identification of the process associated with the interrupt.
3. Marking the process as requiring execution.
4. Selection of the highest priority process requiring execution and passing control to it.

The activities are divided into two classes

(a) Time Dependant or Immediate Service Processes (ISPs)

This class of processes includes those which are

(i) Time dependant
(ii) Cannot be interrupted
(iii) Cannot call another process and wait for its completion.

(b) Interruptable or Tasking Processes (TPs)

These may be

(i) Interrupted for the execution of higher priority processes
(ii) Wait on the completion of a called process
(iii) No restriction can be applied to the depth of nested TP calls in an activity
(iv) The original calling process (e.g. user) must be identifiable at any time in the TP chain.

Any process terminates with a call for a new process or a return to the process which called it.

The relationship between the Supervisor functions, and machine states is described in Section 8.

Figure 5 shows an example of supervisor activity worked out in detail. The example shown is of a User Job calling a Supervisor activity which uses I/O. The activities at the time intervals are as follows (for definitions of TP and ISP see next section).

t_1: User Job 1 is entered having been selected by the Time Slice Scheduler (TSS) in preference to User Job 2.

t_2: User Job 1 calls the service of Loader. Loader is a TP and thus User Job 1 is 'Waited' for Loaders completion. The TSS is entered.

t_3: TSS selects the Loader to enter.

t_4-t_6 Loader issues an I/O request. Three ISPs are entered to fire the I/O.

t_7: TSS is entered to select the TP to enter. Loader is selected.

t_8: Loader is reentered.

t_9: Loader issues WAIT for the completion of the I/O.

t_{10}: TSS is entered. Both Loader and User Job 1 are 'Waited'. It thus selects User Job 2.

t_{11}: User Job is entered.

t_{12}-t_{13} I/O termination interrupt is received. Process 'Unwaits' Loader and fires off further I/O if any further entries have been queued.

t_{14}: TSS selects Loader.

t_{15}: Loader continues operation.

t_{16}: Loader ends and calls End of Process (EOP). EOP deactivates Loader and 'Unwaits' User Job 1.

t_{17}: TSS is entered and User Job 1 is selected.

t_{18}: User Job 1 continues.

2. Supervisor Construction

In order to achieve the distinction made between the Supervisor Kernel and Supervisor Processes, the processes must be described and related to machine conditions in a formal tabular manner.

Supervisor Tables

(a) Machine Condition/Process Table (MCPT) every condition which can arise must be associated with a process. In practice there will be a homomorphic mapping of many machine conditions on to a single process but if generality is to be preserved there must ultimately be one process per condition, (some processes will thus be dummies or failures).

This table is used by the Kernel to select the process required to service the interrupt conditions.

(b) Process Address Table (PAT)
Each process required a control block (see PCB below). PAT gives the address of the PCB for each process.

(c) Process Control Block Table (PCBT)
Each entry in the table is a Process Control Block (PCB). The PCB holds the linkage and status information and also points to the general process information block (PIB). The PCB can be shown as follows:

A Process Control Block

Process Status	a	Active marker
	b	Conditional Wait
	c	Called for process
Process Information	d	Calling process
	e	Priority in PBT

FIG.6.

Slot a : records whether the process is in use or not. (Zero if not in use; address of next instruction when interrupted or waiting).

Slot b : records the cause of the conditional wait whenever two or more parallel processes require interlocking.

Slot c : identifies the process which was called and upon which the present process is waited.

Slots a,b and c record the activity status of the process and from these three slots it can be found whether the process is in use and whether the process currently requires processor time or whether it is in a held position.

Slot d : is linkage information to enable a wait condition to be released in the calling process when the end process condition arises in the called process.

Slot e : defines the process priority for multiprogramming in the Priority Bit Table (PBT - see below), this priority is unique. It is used when changes to the PBT is made following alteration to slots a,b,c.

Slot f : This slot points to the Process Information Block (PIB) which includes:

Physical location of process

Process control details

privilege status

protection status

type marker (ISP or TP)

alternative process marker

testing marker

Register dump area

Second or Subsequent calls on the process.

(d) Priority Bit Table (PBT)

This table is a priority ordered set of markers indicating whether or not the process of a given priority is available for execution. Look up of this table is defined as producing the priority level number. It is necessary to have an efficient method of selecting the next process to use the processorsas this mechanism is used with a very high frequency. No TP can be entered without this scan being performed and the Supervisor efficiency depends on this table and the operations on it. The name ofthe table suggests its construction i.e. as a bit list. The mechanism for scanning of such a list should not involve an instruction loop. (See Section 8).

(e) Priority/Process Table (PPT)

This table enables the priority level number obtained from a scan of the PBT to be converted to a reference to a PCB and thus define the process which is to be executed. This two stage conversion technique enables the relative priorities of the processes to be changed by redefinition of the PPT.

3. Supervisor Kernel and Associated ISP's

The above five tables hold the information necessary to control the action, operation, and linkage of the processes. The action of calling a process and entering it are seen,in relation to the tables, to be:

1. Machine Condition (perhaps caused by a SVC) defines a process from the MCPT.
2. PCB is found for this process from the PAT.
3. Linkages and activation are established in the PCBT.
4. Summary of revised status conditions inserted in PBT.
5. Multiprogramming action finds next priority to be entered from PBT.
6. Priority Level is converted to process using PPT.

Fig. 7 illustrates the action of the Kernel and its relationship to processes (Line 1 is the division between the processes and the Kernel itself). The machine interrupt is received and is analysed (Block 1). This is not in general a one level activity and the further analysis of the cause of an interrupt may be required.

From the analysis a process is determined using the MCPT (Block 2). The process type is found from the PCB (Block 3). This may be an ISP or TP, if the latter then the information about the TP is passed to a special ISP (TP set up, see below). Thus the Kernel only causes directly the entry to ISPs. Following the operation of an ISP there can be 4 conditions:

(a) A further ISP is requested and a further determination of the process is required before the Kernel loop is reentered (Block 4).

(b) An interrupt has arisen since the ISP was entered and the interrupt analysis (Block 1) is reentered.

(c) No ISP has been named as successor and a TP has been selected for execution.

(d) No ISP has been named as successor and no TP has yet been selected.

(e) A TP has been entered.

Conditions 'a' 'c' & 'd' are identical since the implicit successor ISP in conditions 'c' & 'd' are the TSS and Exit processes respectively.

The Supervisor therefore requires a basic set of ISPs for its working, four are detailed as follows:

(a) TP Set Up. An ISP is required to set up the entry conditions for a TP. This process, working on the PCBT and PBT, activates the process by the insertion of the entry address (PCB Slot 1) into Slot 'a'. If the TP is in use then an addition is made to the list of outstanding calls.

(b) Time Slice Scheduler (TSS). When no further ISP is required this ISP is entered to select the next TP for execution.

(c) Exit Process. Following the selection of a TP an inspection is made of the interrupt register and if no interrupt condition exists the procedure for entering the process is undertaken. This consists of reestablishing, if necessary, the TP registers and setting the conditions for its execution (machine state, interruptable status, interstore protection).

(d) End of Process. This ISP is entered as termination of any TP, and resets the 'Wait' condition in the PCB of the calling process.

4. Kernel/Process Relationships

It is perhaps relevant to comment on some aspects of the relationship of the processes and the Kernel and between processes themselves.

(a) I.S.P. Considerations

(a) The overhead of entering a TP demands that ISPs are linked directly to the Kernel, and executed 'inline' with it. This overhead is discussed further in section 4(b) & 7.

(b) ISPs cannot be interrupted and thus cannot use the interrupt facilities of the machine to call successor processes.

(c) The entry and exit linkages are formalised simply by conventional use of register or store locations, thus preserving the independance to the Kernel from any process.

(b) Timing Overhead. When a TP calls an ISP the critical Supervisor path is only executed once. However when it calls a TP the path is executed twice. The first execution is on the call for the TP the second at the conclusion of its execution when the called process ends by issuing an SVC. It follows them that a choice of a process as a TP instead of an ISP entails an extra overhead of one critical path time.

(c) Register dumping. As an ISP is uninterruptable and must leave itself in such a condition that after its operation, it can be reused there is never any requirement to dump or reload any registers on its account. However the registers of a TP must be preserved on any interrupt. This dump and reload time is part of the critical path function of process entry and is an important factor in the efficiency of Supervisor. It is shown in the section of hardware that a multistate processor helps solve this problem.

(d) TP as a normal program. Every TP is fully interruptable and hence can use every facility of the Supervisor. Supervisor TPs can call other Supervisor TPs to any nested depth, but excluding recursion. Supervisor processes and user job processes only differ in respect of their priority and the privilege they are given to utilise machine facilities and resources from which the normal user is banned (in order to preserve the installation integrity). The protection or privilege is established by the ISP which exits from the Kernel.

(e) Process Independance. Processes interact either via data tables or the Kernel. Providing these tables are indirectly addressed each process can be addressed independent of the Kernel and of every other process. The base address of the list of table addresses can be kept either in a fixed store location or a register. Each process in this way becomes a separate program segment which requires no address linkage outside itself. Thus a process need not be composed with the remainder of Supervisor, but exist as part of a user program. This enables a user program to contain within itself a Supervisor process which it requires, and the mechanism (a process called by a SVC) for entering a new process in the Supervisor tables allows it to be used as a normal Supervisor process.

5. Construction and Development of Supervisor

Initial Testing

The implementation and initial testing of a Supervisor is a relatively simple task within the formal structure described above. Initially a sufficient set of processes must be defined in order to deal with the minimum set of conditions necessary to run the machine i.e. the Kernel, the ISPs to select and enter TPs a set of simple TPs and ISPs to provide I/O and Loading functions, and a set of dummy or failure processes for the undefined activities. Once this framework has been established, the alteration of information in the tables and the provision of new processes is a simple procedure and one which can be automated. Thus the Supervisor can be built using bootstrap techniques.

Examination of the tables and the structure show that the following information is required:

(a) Association of process number with a machine condition (MCPT table)

(b) Definition of Priority (PPT)

(c) Definition of the information to be inserted into the PCB by an ISP, and the PAT entry made. (See Section on Supervisor tables for details).

Under the artificial conditions which hold during Supervisor implementation the ability to add (by redefinition of a dummy) a new process together with Post-Mortem TPs (and other similar processes) gives a sufficient mechanisation for this phase of development. This process can be used to interchange tested processes after the system has been commissioned. For example it may be necessary to replace the Job Step Sequencing process to reflect changes in the User environment on a time of day basis and in this case the insertion ISP would be called from the process which deals with clock interrupts. This ISP must be given the lowest priority of Supervisor process as and if a 'Wait' condition exists in the PCB table, it must exit and recall itself.

6. On-Line Provision of Alternative Processes

Tasking Processes

The above structure is defined around a one to one correspondence between interrupt condition and process with subsequent analysis within a process if required (e.g. I/O end of transfer analysis) and a calling of further processes consequent to this analysis.

The facility of dynamically providing specialist processes for use of privileged users is a simple use of this structure. Such a dynamic process must not involve changes to any of the processes which call the one which is being supplemented by an alternative. A decision process is therefore used to select the required process according to the conditions.

In order to do this the following operations must be performed:

(a) The existing process (if any) must be renumbered

(b) The decision process be inserted in place of the existing process

(c) The alternatives inserted and allocated to spare process numbers

The condition under which various alternative processes are selected are numerous, some examples are: user classification, special peripheral conditions, time of day, operator supplied data. Thus most of the decision processes may have to be specially written by the user.

IS Processes

The mechanism of replacement and renaming can be extended to ISPs. In general it will be a requirement that the decision process itself an ISP and that it does not cause an unacceptable overhead.

If the ISP is exceptionally time critical this method may therefore be unacceptable, in which case the ISP must be written to include all the alternatives.

This apparent lack of flexibility is unlikely to be important since most of the ISPs are not concerned with processes directly involved with a User, but with system oriented functions (e.g. I/O, Time Slice Scheduling etc).

7. On-Line Testing of New Tasking Processes

The method described above for handling alternative processes is obviously well suited for allowing an alternative process to be under test.

Establish of a TP for test

The procedure for inserting alternative processes described above provides the basic mechanism. There are however some important differences.

(a) Both the alternative and the original may have to be executed.

(b) The process under test must operate entirely within a copy of the Supervisors data space.

(c) All processes called by the process under test must also be executed on the data copy.

(d) Only processes which do not change the state of other parts of the system (e.g. issue I/O instructions to a system device) can be allowed to operate on the data copy.

Thus the decision process described above becomes as Test Management Process, which carried out the above functions of change and test.

The conditions of each test must therefore be established so that the correct sequence of operations can be carried out. As with alternative processes a TP must be provided, with the following parameters:

1. Identification of process being 'replaced'.
2. New name for the existing process.
3. Address of the space for data copy (for input to an internally called 'Data Copy' ISP).
4. Marker to indicate if existing process is to be executed, and possibly the address of its parameters (they may be different from the new version!).
5. Specification of processes to be executed on the copy data before execution of the test process (e.g. substituting simulated devices for system devices involves changing the system device table) together with address of their parameters. These processes may be existing Supervisor processes (in which case the specification would be in the form of an SVC) or be supplied by the programmer.
6. Information identifying the process under test (see section 2 Supervisor Construction).
7. Processes which may be automatically called and which are permitted to be executed on copy data.

Execution of process under test conditions

It is evident that a process under test is entered with a different data address and that any normal processes that are called must also operate on the data copy. Further a check must be made that the process is valid for operation on copied data.

The PCB of a test processor must therefore contain a Test Marker, which is transferred as a Temporary Marker to any called process. The existance of either Test Marker causes the Temporary Marker to be set. The Temporary Marker, unlike the Permanent Test Marker, is cancelled when a process is terminated. A marker denoting ability to operate on a data copy must also be provided. For safety all processes are assumed to be in this category unless otherwise specified by the programmer.

Any errors caused by a test process (e.g. invalid operator) must be regarded as a normal user error, including the case when the process is only temporarily under test.

Entry to Test Processes

SVC Process. The Test Process is entered by the normal SVC mechanism. If the original process is required for execution then two sets of parameters must be created, and the two processes will act on separate data copies in an interleaved fashion.

Other Interrupt Processes. Sometimes however the process which is being tested is not entered directly, but either via a higher level process or as a result of a non-program generated interrupt. The first of these is merely a slight logical extension of the direct entry process described above since all processes are on the same control level regardless of the 'logical level' within a series of processes.

The entry from non-program interrupt does require a special facility. This facility must simulate a supervisor interrupt and cause the test process to be entered as though from the Kernel. The simulation consists in establishing within the copy data space the conditions which would have been set had that process been entered normally i.e. it acts as a substitute for

the ISP which always precedes a TP which is dependant on machine hardware registers (e.g. Secondary Bytes of I/O channels).

Testing IS Processes

Since any ISP can be replaced by a TP this problem can, in most circumstances, be reduced to an example of TP testing. However such a replacement may cause the conditions which necessitate the use of an ISP to be invalidated, e.g. the time conditions. When such conditions occur it is impossible to provide a general facility of the kind described for TP testing. The ability to replace processes can be used however to replace the original by one which whilst allowing the timing conditions to be met does allow alternative process to be selected.

With modern machine design, and autonomous peripheral and communications systems the need for time-dependant ISPs is very much reduced. The limitation in testing a new ISP is not therefore likely to prove a serious problem, and in no way diminishes the value of the system described.

8. Hardware and Efficiency Considerations

The success of the design of the Supervisor depends on three main hardware considerations which all influence the timing of the critical path.

a. Multiple Processor States and Registers

It has been illustrated in previous sections that the calling of processes is a nested procedure. Ideally every level of the nest of calls requires its own processor state and registers. This enables the nesting of process calls to operate with no dumping or reloading of state registers. The illustrations demonstrate the requirement for at least three levels, a user level, a Supervisor TP level, and a Supervisor ISP level.

b. Interrupt Analysis

The interrupt system must give the maximum assistance to isolate the condition of the interrupt in hardware. At least the first level analysis should be a hardware function.

c. Time Slice Scheduler and Exit process

These processes should not contain an instruction loop. This imposes a requirement for some type of search instruction and also a single instruction to load registers. The Supervisor design relies on the multiprogramming of nearly all processes including user jobs. Since there are likely to be of the order of a hundred such levels, a fast search mechanism is vital.

System 4

The above three requirements are met on the System 4 range of English Electric Computers. This range is compatible with the R.C.A. SPECTRA 70 which uses the IBM 360 order code, but differs from the IBM 360 in having a different set of privileged instructions and also four (instead of two) processor states.

The use of the four states of this System is summarised in Fig.8.

a. Multiple Processor States and Registers

A summary table of the proportion and usage of a multi-state processor is shown in Fig.8. This shows that either of the following can occur without register manipulation.

(i) The operation of a user job to be suspended while a hardware originated interrupt is serviced by a Supervisor TP or ISP.

(ii) A user job to request the service of a Supervisor TP which itself requires the service of a Supervisor ISP.

In the illustration of the linkage requirements of processes (fig.5) the only register dumping and reloading occurs at times t_{11} and t_{18} when processor state P1 changes its operation from user job 1 to user job 2 and then reverts to user job 1.

b. Interrupt Analysis

The interrupt system causes on interrupt the setting of a flag in a 32 bit register. The hardware also sets a value in a P3 register which allows an immediate determination of the process. If the interrupt is an SVC the call number is also immediately available in a further register.

c. Time Slice Scheduler

In the System 4 instruction code it is possible to scan a bit list for the first zero (or non zero) entry by use of the Translate and Test instruction. Following the operation of this single instruction acting with a specially constructed translation table the position of the zero bit is indicated by the contents of two registers. This technique is used to scan the PBT.

Supervisor Efficiency.

In the description of the Kernel it was established that a Supervisor TP is subject to the overhead of the critical path time. The efficiency of the Supervisor can therefore be adjusted by making very high frequency processes

into ISPs. In figure 5 various I/O processes are shown as ISPs not because they require to run uninterrupted but because of this fact. Very few processes fall into this category in practice apart from I/O operations. Placing such processes into the IS category allows I/O operations to be used by Supervisor TP without register dumping. Clearly the fewer processes made uninterruptable the better control can be placed on the priorities of processes and the easier it is successfully to service time dependant devices.

Microprogramming

Supervisor efficiency will be further enhanced if the invariant processes of the Supervisor are built into the hardware. The paper has demonstrated the possibility of the invariance of the Supervisor Kernel, it can thus be microprogrammed. With the development of slow write/fast read memories it will be possible to place the ISPs into hardware to be used as 'extracodes' (as on the ICT ATLAS Computer) without sacrificing the flexibility of the system.

9. Documentation of Process and Table Usage

Each Supervisor process having been isolated and formalised is potentially replaceable and documentation is required to establish the interrelationship of each process with other processes and tables. Figure 9 represents a tabulation of the interrelationship of P_i. This shows the condition upon which P_i is called, its table usage, and the chain of process calls arising from the execution of P_i. It is possible to represent the total set of interrelations of the Supervisor on a single table. Each process dependency is easily seen and an installation requiring to replace a Supervisor facility can replace just the required process chain without requiring knowledge of the Kernel or other processes.

Such formal documentation can obviously be built in to the Supervisor and used as part of the Test Management Process for guiding programmers.

10. Summary

The Supervisor is only entered by machine interrupt, each of which is seen as a request for the activitation of a Supervisor process. This may itself make a further analysis and request a further activity. The interruption can be hardware originated such as a termination of an input/output operation, or program initiated when a program requests the operation of a Supervisor service. Some Supervisor activities are of high frequency, some are time dependent, some of high priority. Other activities can be multiprogrammed. Internal processes of Supervisor require to use other internal processes.

With these considerations in mind the Supervisor is seen as a set of 'processes' controlled by a Kernel within a formal data and linkage structure. Each process performs a distinct Supervisor activity. The processes are of two types: those which are interruptable and those which can be multiprogrammed. The first type satisfy the considerations of high or immediate priority, and high frequency use. The second type satisfies the requirements of the parallel activities.

The User Job is seen by the Supervisor as an addition to these processes, that is, no distinction is made between the tasks produced by Supervisor processes and those by User Jobs. It is shown that all processes can be entered by the same formalised procedure, and that the formalised linkage mechanism ensures the independence of all processes to the extent that they do not require to be compiled or composed with the Kernel, but can be embedded in a User program.

Formal decision tables and machine interrupt conditions direct the execution of the Supervisor Kernel. The Kernel is seen as being independent of every process, thus enabling the Kernel to become a direct extension of the machine hardware. The Kernel is shown to be ideally suited to the use of microprogramming techniques with no loss of flexibility or generality. With the advent of fast read/slow write memories it is seen to be possible to retain the flexibility, whilst microprogramming frequent used routines and thus improving Supervisor efficiency.

With the modularity and formalisation of Supervisor it is possible to document in tabular form every interrelationship of the Supervisor activities to the extent that such a tabulation can define the total effect of any alteration to the Supervisor.

11. Conclusion

The problem of providing a single operating system for many diverse user requirements within a single environment is seen as principally a requirement to provide a completely flexible Supervisor, in which every process is replaceable, and which allows simple insertion, deletion and replacement of its parts.

The paper has described the design of such a Supervisor and has shown that the following objectives can be met:

(a) Efficient operation

(b) Easy implementation

(c) Formalised and modular structure

(d) A basic structure which allows, by microprogramming, substitution of hardware for software facilities, without losing any flexibility

(e) Alterable by regeneration;

by dynamic addition or replacement of protested processes;

for dynamic testing of processes;

for selected classes of users.

12. Acknowledgement

This paper is presented by permission of English Electric Computers Ltd. The authors wish to acknowledge many hours of discussion with their colleagues and particularly Mr. A.Williams and Mr. R.C.Hutton., who are responsible for the implementation of a Disc Operating System Supervisor for System 4-50/4-70 based on the principles described.

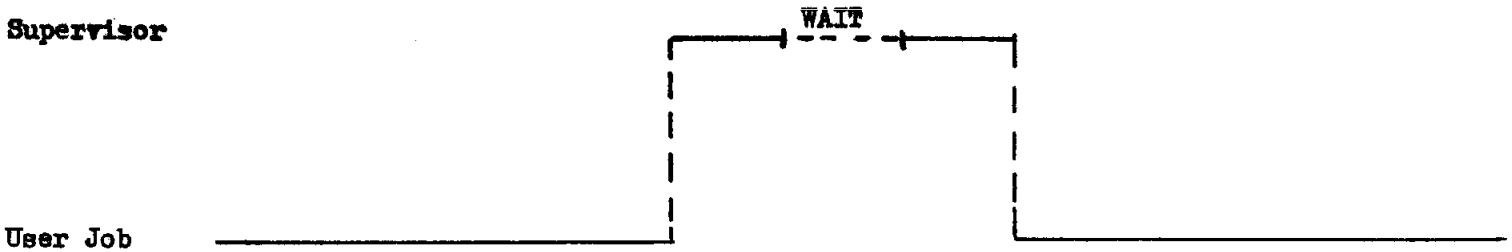

FIG.1

Uniprogram, serial activity

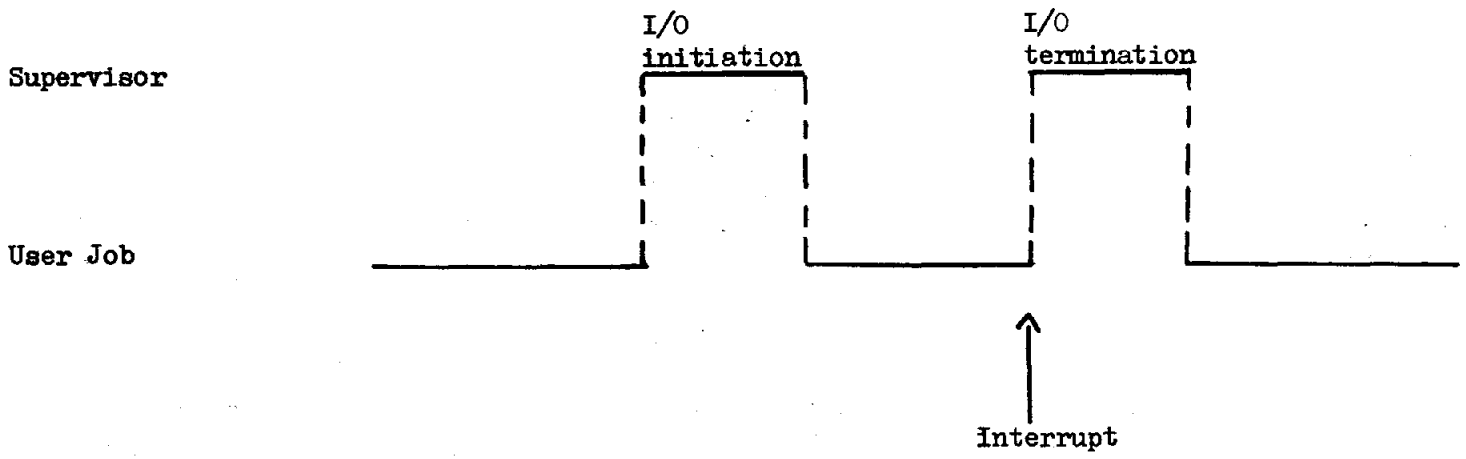

FIG.2

Supervisor with basic interrupt facility

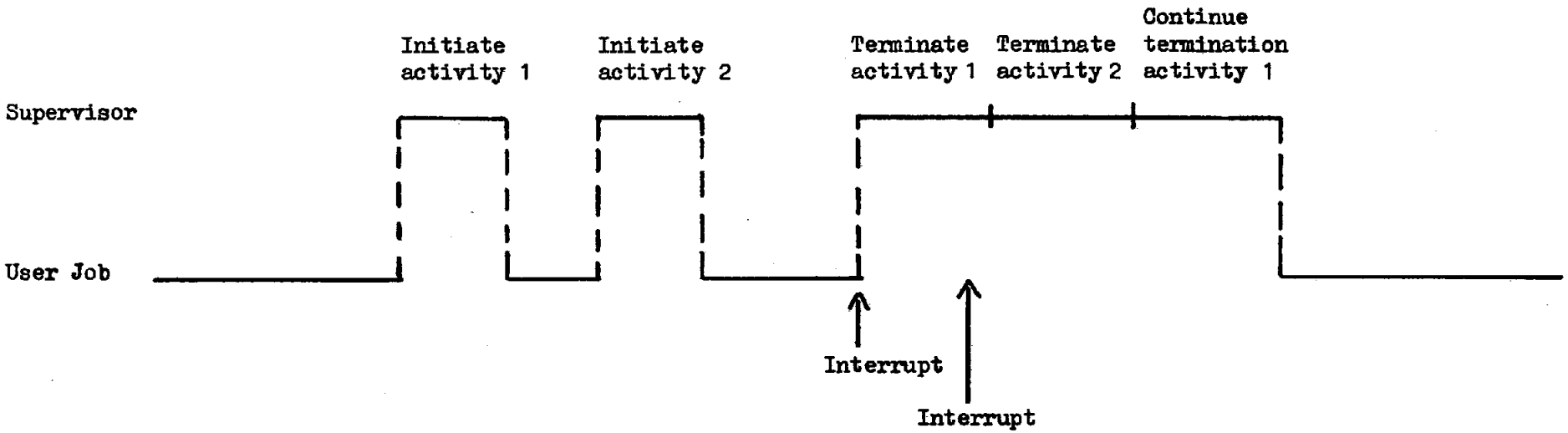

FIG.3

Simple Priority Supervisor.

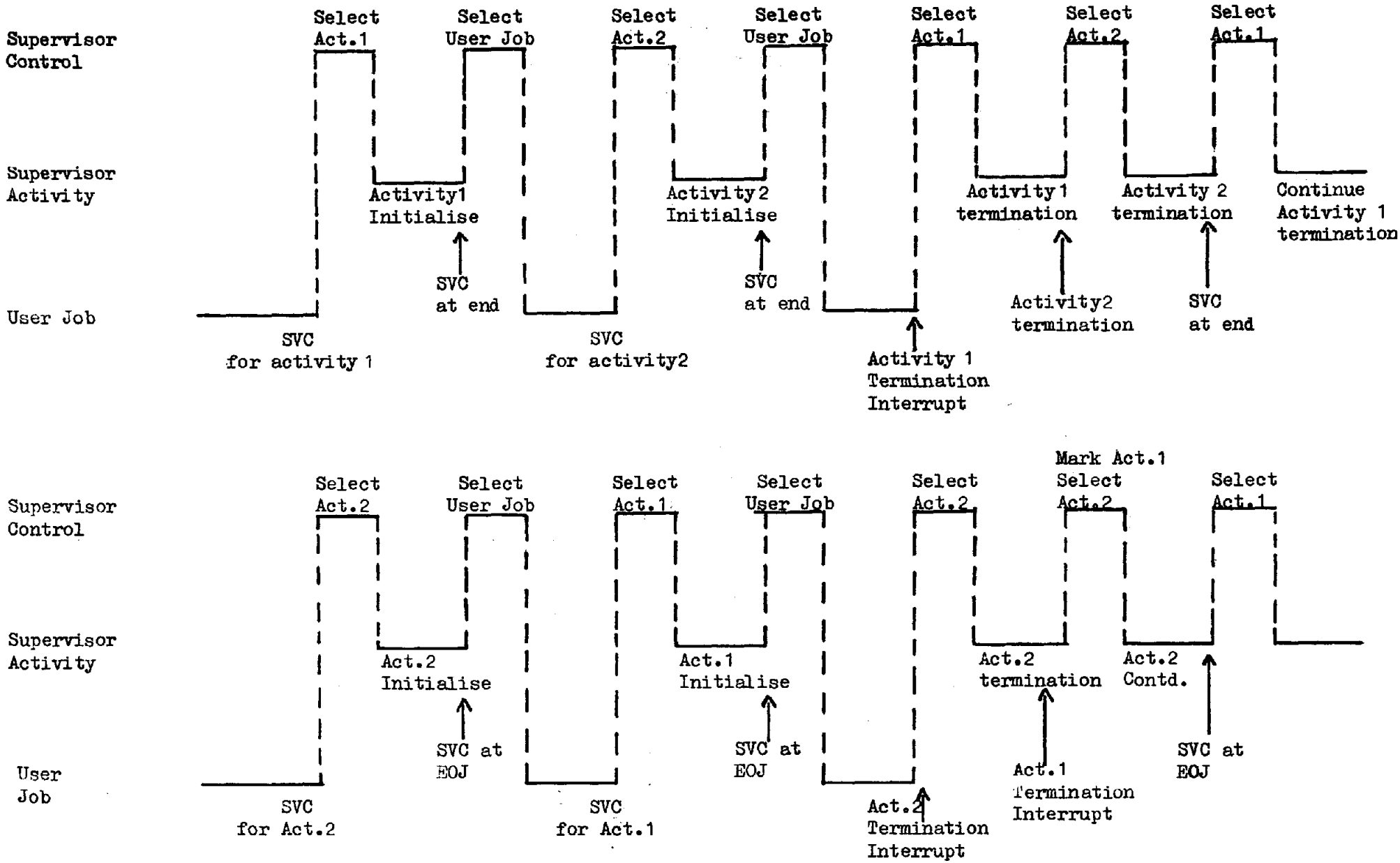

FIG.4

Multiprogramming, Supervisor, Priority controlled.

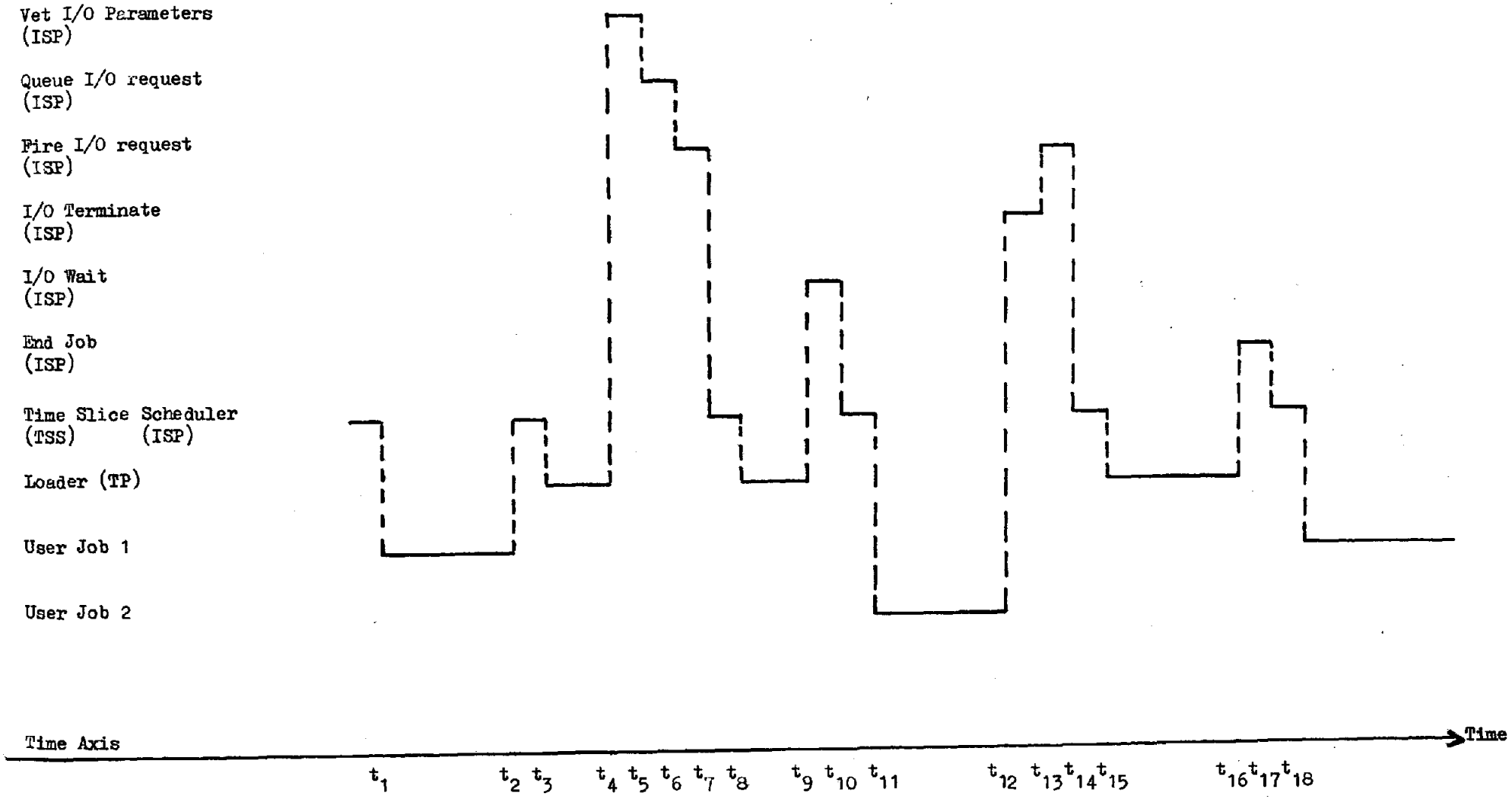

FIG.5

Supervisor handling I/O request.

Processor State	Status	Use	Entry
P1	Fully interruptable	User Job Processes	Programmed change of processor state
P2	Fully interruptable	Supervisor TP	Programmed change of processor state
P3	Uninterruptable except for machine failure	Supervisor Kernel and ISPs	Normal machine interrupt
P4	Uninterruptable	Machine failure conditions	Machine failure interrupt

FIG.8.

Use of Supervisor States

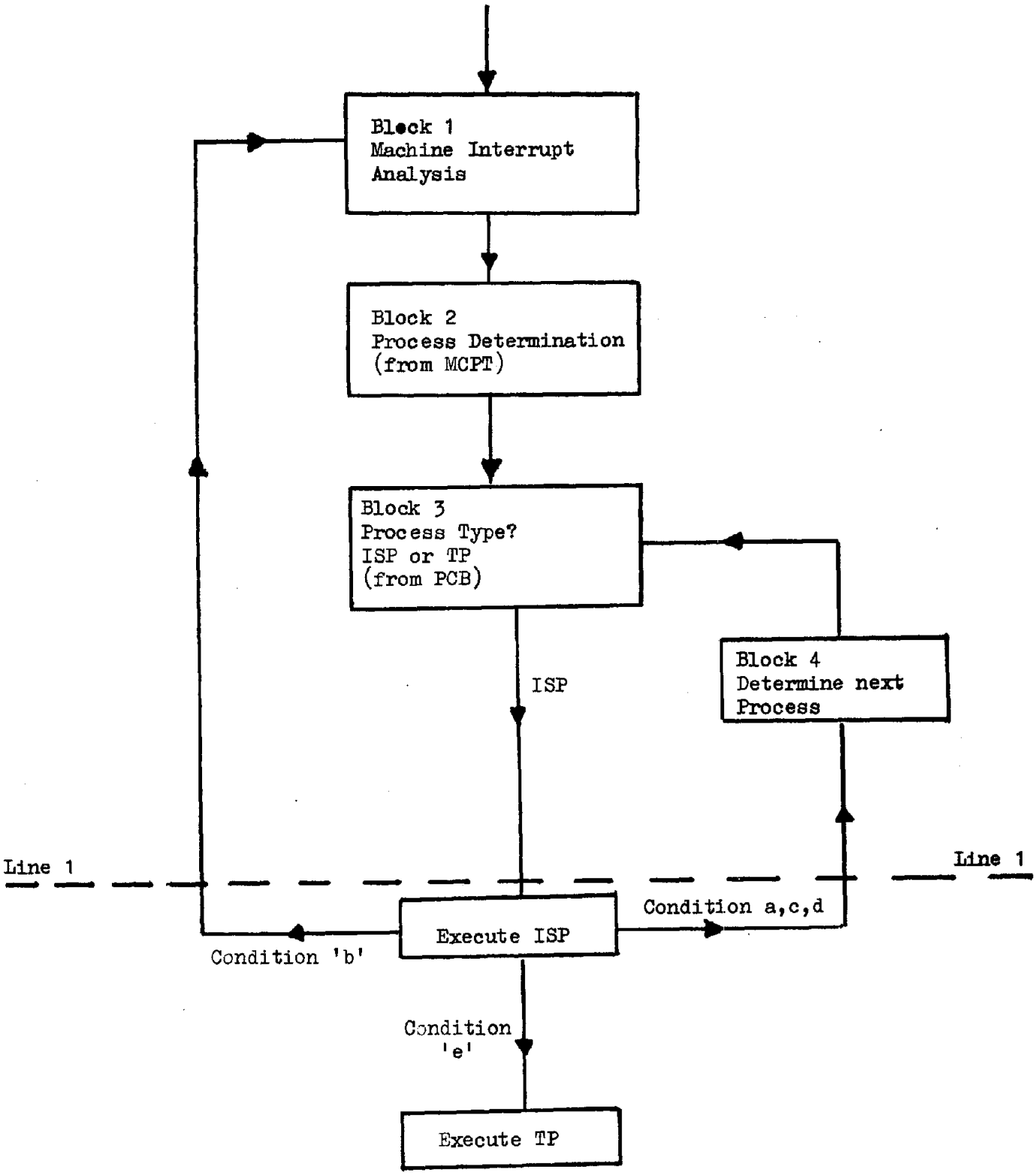

FIG.7

Supervisor Kernel, flow diagram.

Process	Directly		Indirectly			Called by	Uses Tables
P_i	calls	P_1	P_5	P_6	P_7	MCPT entry 4	T_3 (R/W)
	uses	T_4(W)		T_3 (R/W)	T_4 (R)		

FIG.9

Description of P_i