SPECIFICATION

Real-time Data Processing System for Processing Time Period Commands

The present invention relates to real-time data processing systems and is more particularly concerned with arrangements for processing "wait-for time period" commands.

In realtime data processing systems used to control for example telecommunication switching systems or the like it is often necessary to suspend a process for a specified time period. For example during call set-up the dialled digit registration process could be suspended after the reception of each digit for a predetermined period (such as the inter-digital pause). Such an operation is achieved by executing a wait-for time period command or macroinstruction in which the command itself specifies the time delay. Many other instances of "wait-for time period" macroinstructions can be envisaged and in a large exchange system using a multiprocessor stored program control the number of suspended processes experienced at any one time can be quite substantial.

It is an aim of the present invention to arrange all the "wait-for time period" commands into an ordered list and to arrange for the data processing system to administer this list so that the process suspended can be restarted when the wait-for time period matures.

According to the invention there is provided a real-time digital data processing system of the type handling processes which include "wait-for time period" commands, each said command specifying one of a plurality of predetermined time periods and all processes which are suspended by commands having the same particular time period are chained to each other in the chronological order in which they are suspended and to a timing chain header which holds information on the real-time at which the wait-for time period of the first process in the chain will mature and all the timing chain headers are linked in list and the system includes a process for searching the list when a predetermined time interval has elapsed to ascertain the process or processes whose wait-for time period has matured.

Also according to the invention there is provided a real-time digital data processing system of the type handling a plurality of processes which include "wait-for time period" commands, each said said command specifying one of a plurality of predetermined time periods, the system including a memory, for storing information relevant to the processes, and at least one processor unit arranged to perform the processes and each process is provided with an information segment in the memory for holding working parameters for the process when the process is suspended and the information segment includes an indication of the time when the wait-for time period is due to mature for that process and information segment linking information which forms the information segments, of all the processes which are suspended by commands having the same particular time period, into a first linked list arranged in the chronological order in which the processes are suspended and the first linked list is also linked to a timing chain header segment stored in the memory and exclusively allocated to the said particular time period, the timing chain header segment storing a wake-up value indicative of the time when the wait-for time period for the first information segment on the first linked list will mature and each timing chain header includes header linking information forming the timing chain header segments into a second list and the system includes a timing chain search process which causes the second list to be searched to ascertain the process or processes whose wait-for time periods have matured and the system also includes means for activating interval timing arrangement conditioned to monitor a time interval and on completion of the interval to activate the timing chain search process.

The invention will now be described with reference to the accompanying drawings in which

Figs. 1a and 1b which should be placed side by side with Fig. 1a on the left show a block diagram of a central processing unit suitable for use in one embodiment of the invention.

Fig. 2 shows the registers incorporated in the CPU of Figs. 1a and 1b.

Fig. 3 shows the store protection (capability) registers incorporated in the CPU of Figs. 1a and 1b.

Fig. 4 shows a process dump stack.

Fig. 5 shows a process base.

Fig. 6 shows the scheduler's running list.

Fig. 7 shows a timing chain according to the invention.

Fig. 8 shows how timing chain headers are linked.

Fig. 9 shows a flow diagram of a processes suspension algorithm, whereas,

Fig. 10 shows a flow diagram of a so-called "usherette" process.

Before embarking on a detailed description of the arrangements of the invention, opportunity will be taken to review briefly a central processor unit which would be suitable for use in the data processing system of the invention. Such a central processing unit is shown in Figs. 1a and 1b which should be placed side by side with Fig. 1a on the left. The central processing unit is connected to a storage module over leads SOHCS, SDH, SIHCS and SIH. Leads SDH and SIH being the output and input data paths respectively of the storage module whereas leads SOHCS and SIHCS are the output and input control signal highways respectively for the storage module.

The central processing unit is microprogram controlled μPROG which exercises control over data transfer gates allowing the passage of a data on and off the highways of the unit. The unit

comprises an arithmetic unit MILL, a comparator COMP, an instruction register IR, an operand register OPREG, a store communication register SDIREG and three register stacks. The unit is organised on the so-called capability register structure which is disclosed in detail in Specification No. 1,329,721. The stacks provide general purpose register storage ACC STK (shown in detail in Fig. 2) and capability register storage divided into base BASE STK and limit T/C LMT STK stacks and these are shown in detail in Fig. 3. The central processor is particularly suitable for use in modular processing systems of the type discussed in "System 250—a fault-tolerant, modular processing system for control applications" pages 26 to 34 of issue 27 of Systems Technology. In such systems the capability structure is arranged such that a master list of all resources is held in a so-called system capability table and each process is provided with a list of pointers to that table indicative of the resources to which the process is to be permitted access. In addition each process is provided with a so-called dump stack (shown in Fig. 4) held in the common memory which is pointed to by the dump stack capability register DCR of the capability register stack in the processor unit executing that process. The dump stack DA is used to store the contents of the general purposes register stack ACC STK when a change process occurs (i.e. when the process is suspended) together with the value of sequence control register SCR and all the working capability registers RSP WCR0 to RSP WCR7. In addition each process dump stack DA includes a RIGHT LINK, a LEFT LINK and an INCREMENT value and these values will be used later in operations required to perform the embodiment of the invention.

Each process is provided with a so-called process base (Fig. 5) PB which is a list of primary pointers, including one to the process dump stack WPDS, a job link block WJLB, a trap flag WFL, a scheduler work space linke WLINK and a work space holding a list of Post and wait-for flags.

The operations performed by the multi-processor system of System 250 are organised by a scheduler process which operates on a running list. This list is shown in Fig. 6 and it consists of three blocks of information having one entry for each processor unit in the system. Fig. 6 shows a running list arrangement for a three processor unit system and the list includes for each processor (i) a pointer to the process base for the process running on a processor, (ii) a pointer to the dump stack for each running process and (iii) a pointer to the dump stack for the process previously running on the processor unit.

Each processor has its own current time block and this block is addressed by a time capability register TCR (Fig. 3). The block pointed to by TCR consists of three words labelled WIT, WITSUP and WETNI. WIT is the interval timer, WETNI is the estimated time at which the next timer interrupt will occur and WITSUP is used to record the interval timer value when one processor interrupts another.

Timing information for WAIT-FOR timing periods, is based on a notional supervisor "current time". In fact, there is no single value; instead, each processor maintains its own version, and external system control arrangements (not shown) are made to keep the processors in reasonable synchronism.

Each version of current time is maintained as follows. The word labelled WETNI is used to hold the estimated real-time at which the next timer interrupt will occur, i.e. the real-time at which WIT will become zero. Current time is given by CT=WETNI—WIT. Whenever WIT is updated at the end of a time interval to start a new internal time-out, WETNI is naturally altered by the same amount. The value WIT is set-up by an interval timer process to a value indicative of the number of 100 μ second periods which are to elapse before the internal timer process is to be reactivated. Each processor in the system is arranged to be interrupted at 100 μ second intervals and the interval timer word WIT is decremented by one and tested for a zero condition. When a zero condition is encountered the interval time-out is complete and reentry into the process waiting for the interval to mature is made.

The actual interval timer process used in this embodiment of the invention is a so-called "usherette" process which will be considered in detail later.

Each CT is normally an increasing value, but there are two possible exceptions to this:—

(a) When any time value is about to overflow the capacity of one word, all the time values are reduced by a fixed amount (typically 3 minutes).

The CTs may be subject to minor corrections when synchronisation is performed.

The use of the word labelled WITSUP is two-fold. When one processor wishes to interrupt another, it sets the latter's WIT to zero, (unless it is already zero or negative). The previous value is WITSUP so that the original interval to be timed can later be re-established. WITSUP is also used to achieve single-threading of operations on the block: the value —1 indicates that the block is "locked".

In the system according to the invention, a process can suspend for an interval of time which it specifies itself. The interval can take any integer value from one upto a fixed maximum value, and no limit is placed on the number of processes which can be simultaneously suspended.

The data structure used to implement the facility has been designed to be efficient for application in which (i) the number of processes simultaneously suspended can be vary large and (ii) the number of different time intervals in force at any instant is relatively small. In a practical example thirty is considered to be a reasonable number, however, it is arranged for a given interval to be in force for a large number of processes which request it at different times, and

which therefore, need to be "woken-up" at different times.

All the processes which are suspended waiting for the expiry of the same time interval are chained to each other in the order in which they made their request and to a "timing chain header". The chaining is achieved by means of the RIGHT LINK (RL) and LEFT LINK (LL) pointers in the process dump stacks as shown in Fig. 7. As Fig. 7 shows a chain is formed which is doubly circularly linked. In the example given three processes, having dump stacks 1, 2 and 3 respectively have "asked" to be suspended for an interval of 15 milli-seconds. The fifteen value is multiplied by ten to convert it to units of 100 μ seconds as the system is arranged to have 100 μ seconds processing slots. The achieved value of 150 is written into the interval INT location in the TIMING CHAIN HEADER. Considering now the example chain shown in Fig. 6 the first process in the chain is due to be woken up at time 1270 and this is written into the WAKE UP location in the header. The dump stack of the first process in the chain has a zero value written into its increment location INC. The second process in the chain requires to be re-enacted at time 1270 plus 20 (i.e. at time 1290) while process 3 should be re-enacted after a further increment of 10 (i.e. at time 1300). The increment value INC for the first process in the chain is always zero and the value NEGSUM in the header is maintained as a signed twenty-two bit number which is the negative sum of the increments of the chain. The NEGSUM is computed by negating the sum of the increments and subtracting one. Hence in the example chosen and the NEGSUM becomes —31 from the increments of 20 and 10. This value is of significance when adding a process to the end of a chain since it enables the INC value for the new process to be determined without traversing the entire chain.

Fig. 7 shows the way in which the process dump stacks for processes are chained for processes which are suspended for a particular timing period (i.e. 15 milli-seconds). In any multi-processor system handling real-time processes, such as are found in the telecommunications art, there will be process suspended for differing intervals. Such a situation is handled by having several timing chains each with a timing chain header.

The headers are then linked as shown in Fig. 8 using the down link WDLINK of the header shown in Fig. 7. The head of the header chain is a timing chain master pointer block having a pointer FIRST HEADER to the head of the timing chain header list. The timing chain headers are not arranged in any particular order. When an individual timing chain becomes empty (i.e. there are no longer any processes on it) the header is said to be "free" and can be re-used for a different interval if required. The pointer FREE in the timing chain master pointer block is used to contain a pointer to a free header and this pointer is set to NULL if no free headers are available.

The timing chain scheme described above involves three major operations in its administration.

1. A process suspends execution until a specified time interval has elapsed (i.e. performs a WAIT FOR time interval instruction) or possibly until some external event (i.e. Flag) occurs, whichever is the sooner. The process has therefore to link itself to an appropriate timing chain.

2. A process in a timing chain is to removed from the chain because of the occurrence of an external event.

3. A process in a timing chain is removed from the chain because its time interval has matured.

The first operation is shown in flow diagram form in Fig. 9 whereas a special process, called the "usherette" process, is responsible searching the timing chain headers for handling the third operation and this is shown in Fig. 10. The second operation is handled using some of the steps of Fig. 10.

Consideration will firstly be given to the operations necessary to suspend a process, in other words place a process on a timing chain.

Process Suspension

As mentioned previously Fig. 9 shows the flow diagram of the operations performed when a WAIT-FOR time period macroinstruction is encountered. The processor unit encountering that instruction will set up, in one of its capability registers, the descriptor (i.e. base and limit value together with the access type code) for the master pointer block (Fig. 8). The processor unit will then perform step PS1 which addresses the FREE location in the master pointer block using the MILL to form the actual required store address in the register SDIREG and addressing the storage module in which the block resides over leads SIH in Figs. 1a and 1b. The contents of the addressed location are returned on leads SDH into say one of the general purpose registers in ACC STK. The processor then performs step PS2 of Fig. 9 where the contents of that register will be tested by the MILL using the arithmetic unit condition signals AUIS.

It will be appreciated by those skilled in the art that all the process operational steps in Figs. 9 and 10 can be arranged to be executed using the equipment of Figs. 1a and 1b using normal data processing operations and accordingly no detailed break-down of the rest of the operation step will be attempted in the rest of this specification.

Step PS3

In this step, which is entered because no free timing header was identified by the FREE location of the master pointer block, the list of timing chain headers is scanned using the down line WDLINK (Fig. 8) of each timing chain header testing each header to see if it is free. If one is found the pointer FREE in the master pointer block is conditioned to point to that header. If no such free header is found a new store block is

obtained from a pool administered by a store manager process and this new block is initialised as a timing chain header and it is added to the end of the timing chain header list.

Step PS4

This step involves the setting of a value X into a general purpose register in the processor unit. The value X is calculated by adding the WAIT-FOR time interval (I) to the current system time. The value X will be used later to condition the WAKE UP value or the increment value for the process.

Step PS5

In this step the timing chain header list (Fig. 8) is scanned looking for a header having an interval time (INT Fig. 7) equal to the requisite wait-for time (I). If a timing chain involving the wait-for time exists Step PS6 will be completed on the y path whereas if no timing chain already exists path n will be followed.

Step PS7

In this step the increment (INC) for the process dump stack is formed by subtracting a value formed from WAKE UP minus NEGSUM minues one (for the timing chain header identified in Step PS6) from the value X formed in step PS4. This step also causes the NEGSUM in the selected timing chain header to be up-dated by subtracting the new formed increment for the new process.

Step PS8

Upon the completion of step PS7 it is necessary to insert the process into the selected timing chain and this is performed by identifying the last process in the chain by reading the LEFT LINK NL LINK slot in the header. It should be noted that if the chain is empty, the "last process" will be the header itself. Having identified the last process in the list the links between this last process and the header are broken are reformed incorporating the new process as the last process in the chain. The value X is also used in this step to ensure that the usherette process is woken up at or before time X. Upon completion of this operation the suspend process process is completed and the processor unit executing the suspend process process would now enter he process scheduler.

Step PS9

If no timing chain exists having an interval equal to the wait for time of the newly suspended process step PS6 will be ended on the n path. Steps PS9, 10 and 11 are performed in this case to take a new timing chain header into use and to set up the WAKE UP, INT and NEGSUM value (Fig. 9) as well as removing that header from the FREE pointer of the master pointer block (Fig. 8).

From the above it can be seen that to suspend a process for a defined period the process dump stack is added to the end of a chain of such process dump stacks all waiting on the same interval equal to the defined period. The administration of that timing chain is achieved by the WAKE UP, RIGHT LINK, LEFT LINK and NEGSUM values of the timing chain header.

Once a process is suspended by the WAIT FOR time period macro it will remain on the timing chain until (a) the required interval matures, or (b) it is removed by the occurrence of some external event. Both of these operations are achieved in the flow diagram of Fig. 10 which is a so-called USHERETTE Process and they will be described in the order defined above.

Usherette Process

This process is scheduled under the control of an interval timer arrangement in a processor unit which is conditioned to mature and cause entry into the Usherette Process at or before the time at which the shortest timing chain wake-up value is to mature. The way in which the shortest value is identified is defined in the following process. Basically the usherette process is used as a timing chain search process to ascertain the process or processes whose wait-for time periods have matured.

Step UP1

In this step a general purpose register in the processor executing the Usherette Process is used to store the value TIME for the process and this register is set to current system time.

Step UP2

A second general purpose register in the processor is used to store the value NEXT TIME and the contents of that register in this step are set to a very large value, say all ones.

Step UP3

The processor now inspects the first header in the chain. This is achieved by using the FIRST HEADER Pointer (Fig. 8). The operations performed in this step involves the reading of FIRST HEADER into a timing chain list pointer (TCL POINTER) which again is one of the general purpose registers of the processor.

Step UP4

In this step the TCL POINTER is used to read the WAKE UP value (see Fig. 7) of the first timing chain header of the chain.

Step UP5

The processor unit now compares the WAKE UP value just read with the value TIME. Typically one of the two values held in separate registers in the processor may be subtracted from the other by the MILL and the result monitored. If WAKE UP>TIME steps UP6 etc. are performed to step onto the next header in the chain as the wake-up time for the first header is greater than the current system time. If the result of the comparison in the step indicates that TIME is equal to (or greater than) WAKEUP then steps UP11 etc. are performed to extract the first process on the first timing chain and to place it on the "ready to run

list". It will be assumed at this stage that the first header's WAKE UP is greater than TIME.

Step UP6

In this step WAKE UP is compared with NEXT TIME and if WAKE UP is less than NEXT TIME step UP7 is performed. In the situation assumed NEXT TIME has been set in step UP2 to a very large value and hence step UP7 will be performed.

Step UP7

In this step NEXT TIME will be overwritten by the WAKE UP value read from the header addressed in step UP4. This operation controls the search "down" the timing chain headers setting NEXT TIME to the lowest value encountered until of course the NEXT TIME encountered equals the current system time given by TIME.

Step UP8

The down link WDLINK (Figs. 7 and 8) in the header addressed in step UP4 is read in this step into one of the general purpose registers of the processor.

Step UP9

In this step the down link is tested to see if it is NULL indicating the end of the chain. It will be assumed that the down link is not NULL. It should be pointed out, however, that such a situation could occur because a process could be placed on a timing chain awaiting a time out or an external event and if the external event occurs before the time out the process will be removed from the chain and the Usherette process could therefore have been woken-up to respond to that time-out. The operations necessary to remove a process from a timing chain are related to steps UP11 to UP18 inclusive and will be discussed later.

Step UP10

In this step the down link WDLINK read in step UP8 will be written into the general purpose register used as the timing chain list pointer TCL POINTER to allow steps UP4 and UP5 to be performed again using the WAKE UP value from the next header in the chain. Steps UP6 to UP10 will be performed in a repeated manner until the header having a WAKE UP value which equates to TIME is found or the complete header chain has been traversed. When the condition WAKE UP equals TIME in step UP5 is found step UP11 is performed which starts the chain of steps which remove the suspended process from the timing chain as its wake-up period has matured.

Step UP11

In this step the right link R LINK (Fig. 7) of the timing chain header of the timing list having a WAKE UP value equal to the current system time is read into one of the processor's general purpose registers. This operation identifies the first process dump stack on the timing chain served by the header.

Step UP12

In this step the right link read in step UP11 is tested to see if the chain is empty. Again such a situation could occur if the timing chain contained a dump stack waiting for a time out or the occurrence of an external event and the latter had occurred causing the removal of the process from the timing chain before the WAKE UP for the particular timing chain matured. It will be assumed that the timing chain "entered" in steps UP5 and UP11 has a process dump stack waiting to be processed at this point in time.

Step UP13

In this step a general purpose register defined in the flow diagram of Fig. 10 as X is used to read the increment value of the right hand neighbour of the timing chain.

Step UP14

In this step the increment value (INC) of the right hand neighbour of the timing chain is set to zero as this process is about to become the first process in the chain since the current first process is to be placed on the ready to run list.

Step UP15

In this step the X value is used to adjust the WAKE UP and NEGSUM values in the timing chain header to accommodate the removal of the reactivated process.

Step UP16

In this step the right (1R LINK) and left (NL LINK) links in the timing chain header are reformed to point now to what was process 2 in the chain as previously constituted.

Step UP17

The processor unit now caused the process, which was identified in steps UP5 to UP12 as being ready to be woken-up, to be placed onto the scheduler's "ready to run" list.

The usherette process will then execute steps UP8, UP9, UP10, UP4 and UP5 searching for any other process which should be woken-up at the current system time given by time. Ultimately step UP9 will encounter the then end of the header chain and step UP19 will be performed.

Step UP19

In this step the final value of NEXT TIME, which will be the lowest WAKE UP value encountered while interrogating the headers which did not equate to the current system time, will be written into the HARDWARE TIMER system to cause the Usherette Process to be reactivated at that time. The Usherette Process then suspends itself waiting for the current system time to reach the HARDWARE TIMER value using the interval timer arrangement of one of the processors.

To remove a process from a timing chain it is simply necessary to perform steps UP11, UP12, UP13, UP14, UP15, UP16 and UP17. Point α in Fig. 10 indicates the point of entry into the

Usherette Process which is "forced" by the occurrence of the particular external event.

The timing chain scheme described above has several advantages over other prior art schemes. Typical of these advantages are:

1. The addition of a new process to a chain is always at the end, and is an easy operation due to the chaining and the NEGSUM value. Traversal of the entire chain is not necessary, and so the operation is efficient no matter how long the chain is.

2. Removal of a process from any part of the chain is an easy and efficient operation not requiring traversal of the chain.

3. If, as in real time system, absolute time values must occasionally be adjusted so that overflow does not occur, it is only necessary to visit each header and adjust one value (WAKE UP) in it. The time values in the individual process dump stacks are relative and do not need to be adjusted.

4. The set of INTERVAL values can easily be adapted to suit the needs of the moment because when an individual chain becomes empty (i.e. there are no more processes on it) the header is said to be "free" and can be re-used for a different INTERVAL.

Claims

1. A real-time digital data processing system of the type handling processes which include "wait-for time period" commands, each said command specifying one of a plurality of predetermined time periods and all processes which are suspended by commands having a particular time period are chained to each other in the chronological order in which they are suspended and to a timing chain header which holds information on the real-time at which the wait-for time period of the first process in the chain will mature and all the timing chain headers are linked in a list and the system includes a process for searching the list when a predetermined time interval has elapsed to ascertain the process or processes whose wait-for time period has matured.

2. A real-time digital data processing system of the type handling a plurality of processes which include "wait-for time period" commands, each said command specifying one of a plurality of predetermined time periods, the system including a memory, for storing information relevant to the processes, and at least one processor unit arranged to perform the processes and each process is provided with an information segment in the memory for holding working parameters for the process when the process is suspended and the information segment includes an indication of the time when the wait-for time period is due to mature for that process and information segment linking information forming the information segments, of all the processes which are suspended by commands having the same particular time period, into a first linked list arranged in the chronological order in which the processes are suspended and the first linked list is also linked to a timing chain header segment stored in the memory and exclusively allocated to the said particular time period, the timing chain header segment storing a wake-up value indicative of the time when the wait-for time period for the first information segment on the first linked list will mature and each timing chain header segment includes header linking information forming the timing chain header segments into a second list and the system includes a timing chain search process which causes the second list to be searched to ascertain the process or processes whose wait-for time periods have matured and the system also includes means for activating interval timing arrangements conditioned to monitor a time interval and on completion of the interval to activate the timing chain search process.

3. A system according to claim 2 in which each information segment includes a first link pointer and a second link pointer and the first link pointer connects the information segment to the immediately preceding information segment in the first list whereas the second link pointer connects the information segment to the immediately succeeding information segment in the first list.

4. A system according to claim 2 or 3 in which each information segment includes an increment value which defines a time period in the form of the difference between (i) the time at which the wait-for time period will mature for the process provided with the information segment which immediately precedes the said information segment and (ii) the time at which the wait-for time period will mature (for the process provided with the said information segment.

5. A system according to claim 2, 3 or 4 in which each timing chain header segment includes a first chain link pointer and a second chain link pointer and the first chain link pointer points to the information segment at the head of the first linked list whereas the second chain link pointer points to the information segment at the end of the first linked list.

6. A system according to claim 5 in which each timing chain header segment also includes a summation value relative to the sum of the increment values of all the information segments in the first list linked to the timing chain header segment.

7. A system according to any one of claims 2 to 6 in which the system includes a timing chain pointer block which is associated with the timing chain search process and the pointer block includes a first header pointer pointing to the first timing chain header segment in the second list.

8. A system according to any one of claims 2 to 7 in which the timing chain search process includes the steps of

a) reading the wake-up values in each of the timing chain header segments,

b) comparing the wake-up values read with the

Usherette Process which is "forced" by the occurrence of the particular external event.

The timing chain scheme described above has several advantages over other prior art schemes. Typical of these advantages are:

1. The addition of a new process to a chain is always at the end, and is an easy operation due to the chaining and the NEGSUM value. Traversal of the entire chain is not necessary, and so the operation is efficient no matter how long the chain is.

2. Removal of a process from any part of the chain is an easy and efficient operation not requiring traversal of the chain.

3. If, as in real time system, absolute time values must occasionally be adjusted so that overflow does not occur, it is only necessary to visit each header and adjust one value (WAKE UP) in it. The time values in the individual process dump stacks are relative and do not need to be adjusted.

4. The set of INTERVAL values can easily be adapted to suit the needs of the moment because when an individual chain becomes empty (i.e. there are no more processes on it) the header is said to be "free" and can be re-used for a different INTERVAL.

Claims

1. A real-time digital data processing system of the type handling processes which include "wait-for time period" commands, each said command specifying one of a plurality of predetermined time periods and all processes which are suspended by commands having a particular time period are chained to each other in the chronological order in which they are suspended and to a timing chain header which holds information on the real-time at which the wait-for time period of the first process in the chain will mature and all the timing chain headers are linked in a list and the system includes a process for searching the list when a predetermined time interval has elapsed to ascertain the process or processes whose wait-for time period has matured.

2. A real-time digital data processing system of the type handling a plurality of processes which include "wait-for time period" commands, each said command specifying one of a plurality of predetermined time periods, the system including a memory, for storing information relevant to the processes, and at least one processor unit arranged to perform the processes and each process is provided with an information segment in the memory for holding working parameters for the process when the process is suspended and the information segment includes an indication of the time when the wait-for time period is due to mature for that process and information segment linking information forming the information segments, of all the processes which are suspended by commands having the same particular time period, into a first linked list arranged in the chronological order in which the processes are suspended and the first linked list is also linked to a timing chain header segment stored in the memory and exclusively allocated to the said particular time period, the timing chain header segment storing a wake-up value indicative of the time when the wait-for time period for the first information segment on the first linked list will mature and each timing chain header segment includes header linking information forming the timing chain header segments into a second list and the system includes a timing chain search process which causes the second list to be searched to ascertain the process or processes whose wait-for time periods have matured and the system also includes means for activating interval timing arrangements conditioned to monitor a time interval and on completion of the interval to activate the timing chain search process.

3. A system according to claim 2 in which each information segment includes a first link pointer and a second link pointer and the first link pointer connects the information segment to the immediately preceding information segment in the first list whereas the second link pointer connects the information segment to the immediately succeeding information segment in the first list.

4. A system according to claim 2 or 3 in which each information segment includes an increment value which defines a time period in the form of the difference between (i) the time at which the wait-for time period will mature for the process provided with the information segment which immediately precedes the said information segment and (ii) the time at which the wait-for time period will mature (for the process provided with the said information segment.

5. A system according to claim 2, 3 or 4 in which each timing chain header segment includes a first chain link pointer and a second chain link pointer and the first chain link pointer points to the information segment at the head of the first linked list whereas the second chain link pointer points to the information segment at the end of the first linked list.

6. A system according to claim 5 in which each timing chain header segment also includes a summation value relative to the sum of the increment values of all the information segments in the first list linked to the timing chain header segment.

7. A system according to any one of claims 2 to 6 in which the system includes a timing chain pointer block which is associated with the timing chain search process and the pointer block includes a first header pointer pointing to the first timing chain header segment in the second list.

8. A system according to any one of claims 2 to 7 in which the timing chain search process includes the steps of

a) reading the wake-up values in each of the timing chain header segments,

b) comparing the wake-up values read with th

time at which the timing chain search process is run,

c) transferring the identity of the processes whose wait-for time periods have matured to a system process scheduler's running list,

d) adjusting the first link lists by removing the information segments from the respective first linked lists which are allocated to the said processes and reforming the said respective first linked lists and

e) computing the time at which the timing chain search process should run.

9. A system according to claim 8 in which step d) includes the operations of cancelling the first and second link pointers in the said information segments and adjusting the second link pointer of the succeeding information segment of each of the adjusted first link list to point to the timing chain header segment and adjusting the timing chain header segment of each adjusted first link list to point to the succeeding information segment.

10. A system according to claim 8 or 9 in which step d) includes the operations of adjusting the wake-up value in each of the timing chain header segments of the said respective first linked lists in accordance with the increment value of the said succeeding information segment.

11. A system according to any one of claims 2 to 10 in which the information segment is a dump stack for the particular process.

12. A system according to any one of claims 8 to 11 in which each processor unit includes interval timing arrangements adapted to be set to a predetermined real-time value and arranged to cause the timing chain search process to run on the processor unit when the predetermined real-time value occurs.

13. A system according to claim 12 in which step e) includes the operation of conditioning the interval timing arrangements of the processor unit executing the timing chain search process with a time value in accordance with the said time at which the timing chain search process should run.

14. A real-time data processing system for processing wait-for time period commands substantially as described with reference to and as shown in the accompanying drawings.

Printed for Her Majesty's Stationery Office by the Courier Press, Leamington Spa, 1982. Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.