(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 355 243 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **88810591.3**

(51) Int. Cl.4: **G04G 15/00**

(22) Date of filing: **26.08.88**

(43) Date of publication of application:
28.02.90 Bulletin 90/09

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Want, Roy, Dr.**
76 Greystone Road
Cambridge CB1 4DS(GB)
Inventor: **Williamson, Robin, Dr.**
Morgentalstrasse 34
CH-8038 Zürich(CH)

(74) Representative: **Möhlen, Wolfgang C., Dipl.-Ing.**
IBM Corporation Säumerstrasse 4
CH-8803 Rüschlikon(CH)

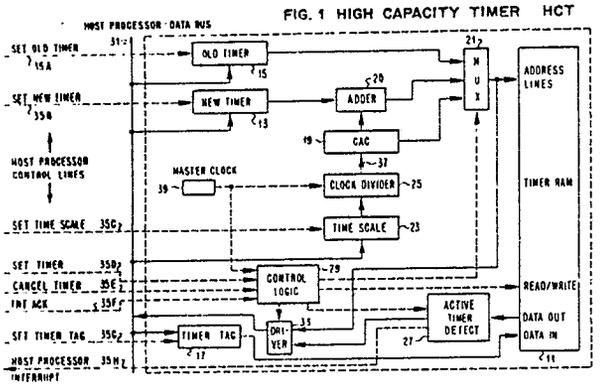
(54) High capacity timer arrangement.

(57) A device for providing a large number of timers is disclosed which comprises a random access storage (11) whose locations are sequentially addressed by a cyclic counter (19). For setting a new timer, the timeout value (13) and a timer tag (17) are furnished to the high capacity timer. The timeout value is then added to the current contents of the cyclic counter to generate an address at which the timer tag is stored thus establishing a new timer.

When during cyclic addressing, it is detected that the location just addressed is not empty, a timer has expired, an interrupt signal is generated, and the stored timer tag is returned.

A specific time scale register (23) and a clock divider (25) are provided so that the resolution of the high capacity timer is easily selectable. To allow existence of several timers expiring at the same time, a special timer service with chainable timer control blocks (T-CB) is provided.

FIG. 1 HIGH CAPACITY TIMER HCT



EP 0 355 243 A1

HIGH CAPACITY TIMER ARRANGEMENT

FIELD OF INVENTION

Present invention relates to an arrangement for implementing a large number of timers, and to a method of operating such timer arrangement. A timer is understood to be a device which can be set to furnish an interrupt or a timeout indication, plus an identification of the timer, at a specific time instant or after a selected time interval. The timer arrangement is particularly useful for communication systems in which protocols require the timed supervision of a very large number of simultaneously occuring tasks such as data transmissions over a network.

BACKGROUND

Timers can be implemented in several ways. Two common approaches are the following:

A register or storage place can be provided for each timer to be activated. The timeout value, i.e. a numeric value representing a time, is stored in each activated timer. The identity of the timer, i.e. the event to which it is related, may be stored in a separate list which also keeps the addresses or names of the respective registers or storage places. All active timers are read out on a regular basis, and their contents is decreased to reflect the elapsed time; simultaneously, a test is made whether the new contents has reached a given value, e.g. zero. If this is the case, the timer is expired and with the aid of the list, the using system can be notified of the timeout, giving the identity of the respective event.

Examples for such timer arrangements were disclosed in the following publications:

- M.D. Schettl, D.J.Thomforde: Centralized Interval Timer and Event Counter. IBM Technical Disclosure Bulletin, Vol.17, No.5, October 1974, pp.1455-1459.
- G.R.Mitchell: Interval Timers Implemented with Virtual Address Translation Apparatus. IBM Technical Disclosure Bulletin, Vol.23, No.2, July 1980, pp.751-756.

Because of the necessity to regularly read out and update the contents of all timers, this solution is not feasable for systems where a very large number of timers in the order of thousands must be active simultaneously.

Another possibilty for implementing timers is to have a list of events for which a timeout is to be supervised, which list is ordered in the same sequence in which the timeouts will occur. For the first item in the list, an actual timer is provided which is temporarily associated with this list item and provides a signal or interrupt when the timeout actually occurs. Thereafter, the next item in the list can be then associated with the timer. There can be provided also a plurality (e.g. 4) of actual timers which are then associated with the respective number of leading items in the list.

Also this solution requires much manipulation of data in the form of items in a waiting queue. New items must be inserted in the correct position. If a timer is to be cancelled which is often desirable, the whole queue must be searched for the respective item to delete it.

Examples of time supervison arrangements are also given in the following patents:

- U.S.Patent 3,451,520: Time-Sharing Arrangement. -Discloses a time-sharing arrangement for a multiprocessing system in which there is provided a time-sharing queue which consists of component tasks from jobs presented to the system for execution and placed on the queue in chronological order. A demand interrupt list is included which includes a list of time slots. A first pointer is provided to indicate the current time slot in the list. A second pointer is provided to indicate a slot in the list which is a chosen multiple of slots removed from the current slot. Further pointers are provided to indicate the oldest slots for tasks to be executed. -This arrangement is suitable for ensuring the timely handling of several tasks in a time-sharing system. It does not provide, however, for timeout interrupts at specific time instants, nor for the storage of a timer identifier which can be presented when a preselected time has elapsed.
- U.S.Patent 4,144,447: Interval Timer. - Discloses a device control circuit which includes the provision of control commands at precise time intervals by gating each command at a time controlled by a continuously running counter to which is added the next event delay value. - The disclosed device provides for the timing of a single (next) event at a time and is not suited for the simultaneous operation of a large number of independent timers. Each next event must be entered into the device after a previous timeout has occured.

OBJECTS OF THE INVENTION

It is a primary object of the invention provide an arrangement by which a large number of timers can be simultaneously implemented.

It is a further object of this invention to devise a high-capacity timer arrangement which needs a

minimum in overhead operations requiring processing time of a system.

Another object of the invention is to enable, in a multiple-timer arrangement, simple setting of timers and cancelling of timers.

A further object is a multiple-timer arrangement which can be increased in capacity by simple adding another storage module.

Another object of the invention is a method of operating a high-capacity timer device which simplifies the handling of timers, and which also allows the handling of multiple timers expiring at the same time.

SUMMARY OF THE INVENTION

These objects are achieved by the invented timer arrangement which is defined in Claim 1, and by the method of operating this timer arrangement, which is defined in Claim 8.

The invention allows the implementation of several hundreds or even thousands of timers by providing a single high-capacity timer chip and a relatively simple software timer service in the host using the timer arrangement. With a respective additional feature, the time scale can be selected prior to usage of the timer by simple insertion of a time scale factor. Furthermore, cancelling of active timers can be easily effected. Another advantage is that the total time of the timer can be increased if required easily by adding another module to the timer RAM storage.

Further features and advantages of the invention will become apparent from the following detailed description of a preferred embodiment in connection with the accompanying drawings.

LIST OF DRAWINGS

Fig.1 is a block diagram of a high-capacity timer implementing present invention.

Fig.2 is a schematic representation of the interaction between the high-capacity timer and a user process or application program in the host processor through a software timer service improving the operation of the high-capacity timer.

Fig.3 shows the format of a timer control block used in the software timer service for simplifying operation of the high-capacity timer.

Fig.4 is a schematic representation of the operations in, and data transfers between, the various units involved in the operation of the high-capacity timer.

Fig.5 is a schematic representation similar to the one of Fig.4, including the necessary steps for allowing multiple timers having the same timeout.

Fig.6 illustrates the contens of plural chained timer control blocks for enabling the existence of plural timers having the same timeout instant.

DETAILED DESCRIPTION

(A) TIMER REQUIREMENTS AND PRINCIPLES OF HIGH CAPACITY TIME

For effective operation of a program there should be available a number of timers which can each be set individually to a target time (timeout) and which, when the timer expires, interrupts the respective program and identifies the timer which has just expired.

This is usually done by storing the target time and an identifier in one of several registers used as timers. For each time increment, the then current time value is compared to the target times in all active timers and when a match occurs, an interrupt is effected and the identifier is read out and transferred to the application program.

This is only possible for a resticted number of timers because the frequent and numerous comparisons are very time-consuming. Present invention suggests the following novel principle which allows a large number of timers to be implemented without heavily burdening the processing power of the host processor.

In the novel solution, a random access storage (RAM) is provided in which each location represents a time instant. All locations are accessed in a cyclic sequence, the address being provided by a counter which is sequentially incremented. To set a timer, an identifier is provided as well as the timeout value in terms of the number of time increments of the advancing clock of the cyclic counter; the timer is established by generating an address which is the sum of the current cyclic counter value and the provided timeout value, and by entering the identifier into the thus addressed RAM location. When later during the normal cyclic addressing procedure the respective RAM location is accessed again, an interrupt is generated indicating a timer expiration, and the stored identifier is returned to the host or application program. A more detailed description is given in the next sections.

(B) HIGH CAPACITY TIMER

The High Capacity Timer (termed HCT in the following) is a peripheral unit, i.e. a special hardware unit which can be used to provide a large number of timers to an application program running on an associated processor (host processor). The highlights of the High Capacity Timer architecture

are: (a) minimal processor interaction, (b) flexible number of timers provided, (c) programmable resolution.

In the following sections, first the structure of the HCT will be described and then the operation of this unit will be explained. Thereafter, the capability to manipulate the timer resolution and duration will be described. Finally, an example of how the HCT can be used to provide a general purpose timer service will be given.

1. Structure of High Capacity Timer

The architecture (structure) of the HCT is shown in the block diagram of Fig.1. Its major components are:

- A RAM chip 11 (Timer RAM). The number of storage locations in the RAM defines the maximum duration of the timers.
- A register 13 to hold a new timeout value requested by the user processor (New Timer).
- A register 15 to hold the reference to a timer (i.e., its address) which is to be cancelled (Old Timer).
- A register 17 to hold the tag (i.e. the identification) to be associated with a new timer (Timer Tag).
- A counter 19 which cycles through the RAM address range (CAC = Current Address Counter).
- An adder 20 connected to the New Timer register (13) and to the Current Address Counter (19) for adding the contents of both.
- A multiplexer 21 for selecting one of three possible address sources to be applied to the Timer RAM (Mux).
- A register 23 to hold the value by which the master time clock is to be divided before being applied to the CAC (Time Scale).
- A divider 25 for dividing a master clock signal by the contents of the Time Scale register, for providing local clock pulses (or "ticks") to the CAC (Clock Divider).
- A detecting unit 27 for receiving data read from a RAM location and furnishing respective interrupt and tag signals (Active Timer Detect).
- A logic unit 29 for directing the operation of the various HCT components under control of the master clock and received "external" control signals (Control Logic).

The HCT is interconnected with the host processor by:

(a) a data bus 31 (carrying data to New Timer 13, Old Timer 15, Timer Tag 17, Time Scale 23, and from a driver unit 33 which is connected to the outputs of Mux 21 and Active Timer Detect 27); and

(b) control lines 35 (35A ... 35H) (carrying binary control signals to New Timer 13, Old Timer 15, Timer Tag 17, Time Scale 23, Control Logic 29, and from Active Timer Detect 27).

The data bus and its branches are shown as solid lines, weheras the control lines ar shown as dotted lines.

2. Basic Operation of High Capacity Timer

The high capacity timer operates by sequencing through the address space of the RAM, i.e. all storage locations of the Timer RAM are accessed sequentially. When a location with non-zero contents is acessed, a timeout has occured and the host processor is interrupted and an identification (Timer Tag) is returned to the host processor. When the addressed location holds a zero word, then no timer has expired. Hence, setting a timer is performed by setting the contents of some particular storage location to a specific value. This value is the Timer Tag which is contained in the respective register 17 when a new timer is requested. The most significant bit of each timer RAM word should always be set to a logical '1' in order to prevent an error when the Timer Tag was selected to have the value zero. Thus, the number of possible tag values is equal to $2^{(N-1)}$ where N is the number of bit positions in each Timer RAM word (storage location).

When the HCT interrupts the host processor by activating the Host Processor Interrupt control line 35H to indicate that a timer has expired, incrementing of the CAC is suspended. The user processor responds to the interrupt by activating the IntAck control line (35F). This causes the contents of the currently adressed Timer RAM location, i.e. the tag of the expired timer, to be gated onto the the processor data bus. At the end of the interrupt acknowledge procedure, the contents of the respective Timer RAM location are reset to zero.

The host processor can request the setting of a new timer and the cancelling of an existing (running) timer by two commands: SET TIMER and CANCEL TIMER. In order to update, i.e. to change the target timeout, of a timer, the user processor must explicitly call the HCT twice: first to cancel the previous value and then to set the new value.

SET TIMER

In order to set a timer, the user processor first loads the tag (identifier) to be associated with the timer into the Timer Tag register 17 by furnishing this tag to the data bus 31 and activating the Set Timer Tag control line 35G.It then loads the timeout value (i.e. a number indicating after how many time units or increments the timer should

expire) into the New Timer register 13 by furnishing this value to the data bus 31 and activating the Set New Time control line 35B, and then activates the Set Timer control line (35D). This causes the address of the currently accessed RAM location (i.e. the CAC contents) to be added to the value held in the New Timer register. The contents of the Timer Tag register is then written into Timer RAM at the location addressed by the resulting sum. Hence, the value furnished by the CAC can be seen as representing the current time. Each Timer RAM address/location can be seen as representing a future instant of time which is ahead of the current time by as many time units/increments as the current CAC contents and the respective Timer RAM address are apart.

In order to give the host processor a reference to the timer which has just been set, the HCT also gates the address of the new timer onto the host processor data bus 31 via the driver circuit 33. This value will be required if the user wishes to utilize the Cancel Timer facility.

It is possible that, when the HCT accesses a Timer RAM location to set a timer, it finds that the respective location already has a non-zero contents (i.e. the location is already in use for another timer which was set earlier but should have the same expiration time). In this case, a special procedure (for a "multiple timer") takes place which is described later in section B3.

The normal cycling/sequencing of the CAC contents is of course suspended during the setting of a new timer. The cycle during which a new timer is set is interleaved into the normal incrementing operation of the CAC. Because the contents of the CAC is not altered by the Set Timer operation, the cycling through the Timer RAM addresses resumes at the point at which it was suspended. Hence, no timers are lost during a Set Timer operation.

CANCEL TIMER

In order to cancel a timer, the host processor must load the reference value received from the HCT at the time of setting the timer, into the Old Timer register 15 by furnishing this value to the data bus 31 and activating the Set Old Timer control line 35A. Then the host processor activates the Cancel Timer command line (35E). The HCT then uses the value stored in the Old Timer register as address to access a location of the Timer RAM. The contents of this location is set to zero, thereby cancelling the timer.

3. Timer Resolution and Maximum Duration

The resolution and maximum duration of the HCT are interrelated and depend on the frequency of the clock signal which is used to increment the Current Address Counter (CAC) and which is furnished on output line 37 of the Clock Divider 25, and on the depth (number of storage locations) of the Timer RAM.

The resolution is defined as the period of the CAC clock signal on line 37. In Fig.1 it can be seen that the duration of the CAC clock period is derived from a master clock which is divided by the contents of the Time Scale register 23. (The master clock is either furnished by a master clock generator 39 in the HCT, or it is provided on a separate clock line from the host processor) . For changing the time scale, a respective value is furnished to the data bus 31 and then the Set Time Scale control line 35C is activated. Using this facility, the host processor can define the length of each timer tick (the value of the time increment) prior to using the HCT. In addition, the user may utilize this function to "slow down" or "speed up" time.

The maximum duration of any timer is given by the product $Dmax = (Nram - 1) \times (clock_period)$, where Nram is the number of addressable locations in the Timer RAM and clock_period is the tick duration of the CAC clock signal on line 37. Therefore, the maximum duration is linearly dependent on the value stored in the Time Scale register 23. The maximum timeout is proportional to Nram - 1 and not Nram because the Timer RAM location with address zero is not used (as was briefly explained above).

4. Example

As an example of the capacity of a High Capacity Timer implementation, a Timer RAM comprising 64 k word locations is considered.

The number of independent timers which the HCT can support is defined by the number of storage locations in the Timer RAM. In this example up to 65,535 independent timers can be supported.

If the period of the master clock is 0.5 ms and the Time Scale register holds the value one, indicating no division of the master clock, the maximum duration of the HCT is ca. 32 seconds. The minimum time between timeouts is the period of the CAC clock signal, i.e. 0.5 ms in this case.

The maximum duration can be increased, along with an increase in the minimum time between timeouts, by increasing the value in the Time Scale register 23. By setting its contents e.g. to 16, the maximum timeout duration rises to ca. 8 minutes. In this case, the minimum time between timeouts is 8 ms.

(C) IMPLEMENTATION DETAILS WITH SOFTWARE TIMER SERVICE

1. Principles

To achieve the best possible utilization of the High Capacity Timer HCT, a software Timer Service (STS) is provided as interface between the application program in the host processor and the HCT. The respective configuration is schematically shown in Fig.2.

The control signals shown on the control lines in Fig.1 are now exchanged between the HCT and the Software Timer Service STS. Furthermore, data are exchanged between the HCT and the STS via the data bus 31. It is assumed that the width of this data bus as well as the width of the host processor data bus is the same as the width of the storage locations of the HCT (i.e., the control or data words can be exchanged in parallel).

The application program in the host processor can issue the two basic commands (which were introduced above) to the STS in the following form:

(1) SET_TIMER (timer_id, ticks, notification)

(2) CANCEL_TIMER (timer_ref)

The three arguments in the SET_TIMER command are the following:

timer_id = The name (identifier) which the application program wishes to associate with the timer

ticks = The duration of the timer (expressed in ticks where one tick is the duration of the CAC clock signal cycle)

notification = The way in which the application program wants to be notified when the timer has expired (This could be e.g. an event which should be posted for the application program)

Upon completion of the SET_TIMER command, the STS returns to the application program a reference. This is to be used should the application program want to cancel the timer.

The argument in the CANCEL_TIMER command is the following:

timer_ref = The reference returned by the STS when the timer was started (This may be the address of the Timer RAM location in which the data of the respective timer are stored)

Essentially, the Software Timer Service STS provides a Timer Control Block (T-CB) for each timer which is active in the HCT. The format of the T-CB is shown in Fig.3.. It comprises six fields as follows:

Timer_id = as defined above

Notification = as defined above

HCT_reference = The reference to be used when the timer is to be cancelled (This is the Timer RAM address of the respective timer)

Activity = A bit indicating whether the status of the respective timer is active or inactive

Next_CB = STS address of the next T-CB if more than one timer is associated with the same expiration time, i.e. with the same Timer RAM location

Chained = A bit indicating whether the status of the respective T-CB is chained to another one

2. Normal Operation

Operation of the Software Timer Service STS is schematically illustrated in Fig.4 and will be explained below:

First (a), the application program issues the SET_TIMER command to the STS, including the three parameters timer_id (T-ID), ticks, and notifier. Thereupon (b), the STS allocates a Timer Control Block T-CB and enters the timer-id and the notification into the two respective fields of the new T-CB.

Then (c) the STS issues in turn a SET_TIMER command to the HCT. It sends to the HCT the ticks value as the new_timer value (NEW-T) and the address of the new T-CB, i.e. ADR(T-CB) as Timer Tag. These two values are entered into the respective registers 13 and 17 of the HCT and are used for establishing (setting) the requested timer (d). Once the timer has been set, the HCT returns (e) to the STS the address of the Timer RAM location which has just been used for establishing the timer, ADR(T-RAM). This address is inserted (f) by the STS into the "HCT_reference" field of the T-CB, and the activity bit in the T-CB is set to "1". The SET_TIMER operation terminates when the STS returns (g) a timer_ref (T-REF) to the application program. The value of this timer_ref is the address of the Timer Control Block, ADR(T-CB), in the Software Timer Service. The application program stores (h) this timer reference for subsequent utilization (if necessary).

When the HCT detects that a timer has expired by determining that an addressed Timer RAM location is not empty, it interrupts the STS (i). During the interrupt acknowledge procedure (j) which was described above already, the STS receives (k) from the HCT the timer_tag stored in the Timer RAM; this is the address ADR(T-CB) of the Timer Control Block associated with the expired timer. The STS then accesses the T-CB (l) and notifies the application program that a timer has expired and returns (m) to the application program the timer_id (T-ID) from the T-CB (and a notifier if such was stored). The T-CB in the STS is then deallocated (n).

If the application program wants to cancel the timer, it issues (p) the CANCEL_TIMER command to the STS, furnishing the timer_ref (T-REF) which

is the address of the T-CB. Thereupon, the STS can read (q) HCT_reference (which is the Timer RAM address of the respective timer) from the T-CB and it issues (r) in turn a CANCEL_TIMER command to the HCT, furnishing this address ADR-(T-RAM) as the argument. The respective Timer RAM location is set to zero, thereby cancelling the timer (t). The T-CB is also deallocated (s) in the STS.

3. Operation With Multiple Timers Having the Same Expiration Time (Chaining Timer Control Blocks)

The STS operation occuring when multiple timers have the same expiration time is schematically illustrated in Fig.5.

Setting: It is possible that when the Software Timer Service STS issues the SET_TIMER command to the HCT, the Timer RAM location which is to be used to store the timer_tag is already in use by another timer (having the same target or expiration time); i.e., contents is detected to be non-zero (d′). In this case, the HCT will return (e′) to the STS a ZERO instead of a Timer RAM address. The STS then activates the IntAck control line (e″) as if it were interrupted due to a timer expiration, and will receive thereupon from the HCT (e‴) the timer_tag stored at the RAM address just used, which is actually the RAM address ADR(T-RAM) of the existing older timer which is due to expire at the same time as the new timer which is just to be set. The STS now chains (f′) the two control blocks, using the "Next_CB" field in the T-CB of the older timer and setting the "chained" flag in both T-CBs.

Expiration: When later the Timer RAM location is accessed during the normal sequencing procedure and the HCT detects non-zero contents (i), it interrupts the STS thus indicating that the first timer has expired. Upon interrupt acknowledgement (j), STS receives from HCT (k) the address ADR(T-CB) of the first T-CB associated with this expiration time. The STS then extraxts (l) the timer_id (and a notifier if stored) from this first T-CB and from the second, chained T-CB and notifies the application program of the expiration of both timers by returning (m′) the two timer_ids T-ID (and any notifier if available).

The chain of Timer Control Blocks can be extended as far as necessary, i.e. any number of timers can be associated with the same expiration time.

Cancelling: To cancel a timer whose Timer Control Block T-CB is chained is very efficient. The activity status indicator bit of the T-CB identified in the CANCEL_TIMER command (p) is simply reset (q, q′) to indicate inactive status. When later the chained timers expire and the STS steps down the T-CB chain notifying the application program about each timer which has expired, it ignores those which have been marked inactive. The "chained" flag in the T-CBs prevents the last T-CB in the chain from being interpreted as a single, unchained Timer Control Block.

Fig.6 illustrates the contents of three Timer Control Blocks T-CB which are chained. The three control blocks T-CB1, T-CB2, and T-CB3 are stored at locations 026, 084, and 113, respectively.

The first box in Fig.6 (above the dashed line) shows the initial contents of T-CB1, i.e. when it represents a single (unchained) timer. It contains a timer_id "abc", a notifier, the T_RAM address 207 (pointing to the RAM location where its own address 026 is stored as timer_tag), and a "1" in the activity status field. The other two fields contain zeros indicating "no chaining".

The three boxes below the dashed line in Fig.6 represent the chained or concatenated T-CBs. As can be seen, T-CB1 now contains a Next_CB address 084 pointing to T-CB2, and indicates a chained status by a "1" in the last field. T-CB2 and T-CB3, besides storing individual T-REFs and notifiers, have a HCT_ref of zero (because only the first T-CB of a chain stores the associated T-RAM address). Both indicate "active" status and "chained" status. T-CB2 has as Next_CB the address 113 of T-CB3, whereas T-CB3 contains a zero as Next_CB indicating that it is the last in the chain.

If one timer is to be cancelled, e.g. the one represented by T-CB2, only the activity status bit is set to zero (as indicated above the box) but nothing else is changed so that the whole T-CB chain is maintained (as described above already).

Claims

1. Timer arrangement for furnishing a time interval expiration indication for each of plural individually identified timers, comprising

- random access (RAM) timer storage means (11) having plural storage locations each for holding at least one timer tag identifying a timer;
- means (19, 23, 25) for sequentially and cyclically addressing storage locations of said timer storage means and for giving a timer expiration indication when an addressed timer storage location contains a timer tag;
- means (13, 17) for receiving in connection with a timer request, a timer tag and a timeout value representing a desired timeout interval,
- means (20) for deriving, from the present timer storage location address held in said cyclic addressing means (19, 23, 25) and from said timeout

value (17), a future address which will be accessed by said cyclical addressing means after a time interval represented by said timeout value; and
- means (29) for storing said timer tag in the timer storage location having said future address.

2. Timer arrangement in accordance with Claim 1, characterized by
- means (27, 33) for furnishing the stored timer tag from a cyclically addressed timer storage location for which said timer expiration indication was given, to a host data bus (31).

3. Timer arrangement in accordance with Claim 1, characterized in that
- said cyclic addressing means (19, 23, 25) includes a cyclic counter (19) which is advanced at regular intervals by a timer clock signal (37),
- said timeout value being a numeric value which represents the number of timer clock intervals corresponding to the desired timeout interval.

4. Timer arrangement in accordance with Claim 3, characterized in that
- said future address deriving means (20) comprises an adder connected to the outputs of said cyclic counter (19) and of a register (13) holding said timeout value.

5. Timer arrangement in accordance with Claim 3, characterized in that
- said cyclic addressing means (19, 23, 25) comprises a loadable time scale register (23), and divider means (25) connected to said time scale register and to a source (39) of a master clock signal,
- said divider means scaling down the master clock pulses at a rate given by the contents of the time scale register and furnishing the scaled-down pulse sequence as timer clock signal (37) to said cyclic counter (19).

6. Timer arrangement in accordance with Claim 1, characterized in that
- means (21, 33) are provided for returning a future address which is used to establish a new timer, as timer reference to a host data bus (31), that
- an additional register (15) is provided for holding a received old timer reference which represents the address of a timer storage location assigned to an active timer and containing a timer tag, and that
- means (21, 29) are provided for addressing, in response to a cancel timer signal (35F), the timer storage location identified by said old timer reference, for deleting the contents of said addressed timer storage location to thus cancel the respective timer.

7. Timer arrangement in accordance with Claim 1, characterized in that
- active timer detecting means (27) are provided which are connected to the timer storage means for receiving the contents of an addressed timer storage location, for detecting whether said contents is different from zero, for furnishing an interrupt signal to a host signalling line (35H), and for furnishing the contents of the addressed location which constitutes a timer tag via respective transfer means (33) to a host data bus (31).

8. Method of operating the timer arrangement of Claim 1, comprising the following steps:
- providing a timer service as interface between the timer arrangement and a user,
- for setting a timer, sending a timer request with a timer identifier and a timeout value to said timer service,
- establishing a timer control block (T-CB) in said timer service and storing therein said timer identifier,
- sending from said timer service a timer request to said timer arrangement, with the address of said timer control block as timer tag and with said timeout value,
- in said timer arrangement, storing the address of the timer control block as timer tag in a selected timer storage location, and returning the address of said selected timer storage location as intermediate reference to said timer service, and
- in said timer service, storing the received timer storage location address as intermediate reference in the assigned timer control block.

9. Method in accordance with Claim 8, comprising the following additional step:
- returning from the timer service, the address of the respective timer control block as timer reference to the user.

10. Method in accordance with Claim 9, comprising the following additional steps for cancelling a timer:
- sending the timer reference for the respective timer to the timer service,
- in the timer service, accessing the timer control block addressed by said timer reference and sending the timer storage location address which is the intermediate reference contained therein together with a cancelling request to the timer arrangment, and then deleting the contents of said timer control block, and
- in said timer arrangement, upon receiving the timer storage location address which is the intermediate reference and the cancelling request, deleting the contents of the respective timer storage location.

11. Method in accordance with Claim 8, comprising the following additional steps:
- in said timer arrangement when, for setting a new timer, a timer storage location is accessed which is already occupied: returning to the timer service, a chaining indication and also the timer tag which is stored in the respective timer storage location and which represents the timer control block of an older timer,

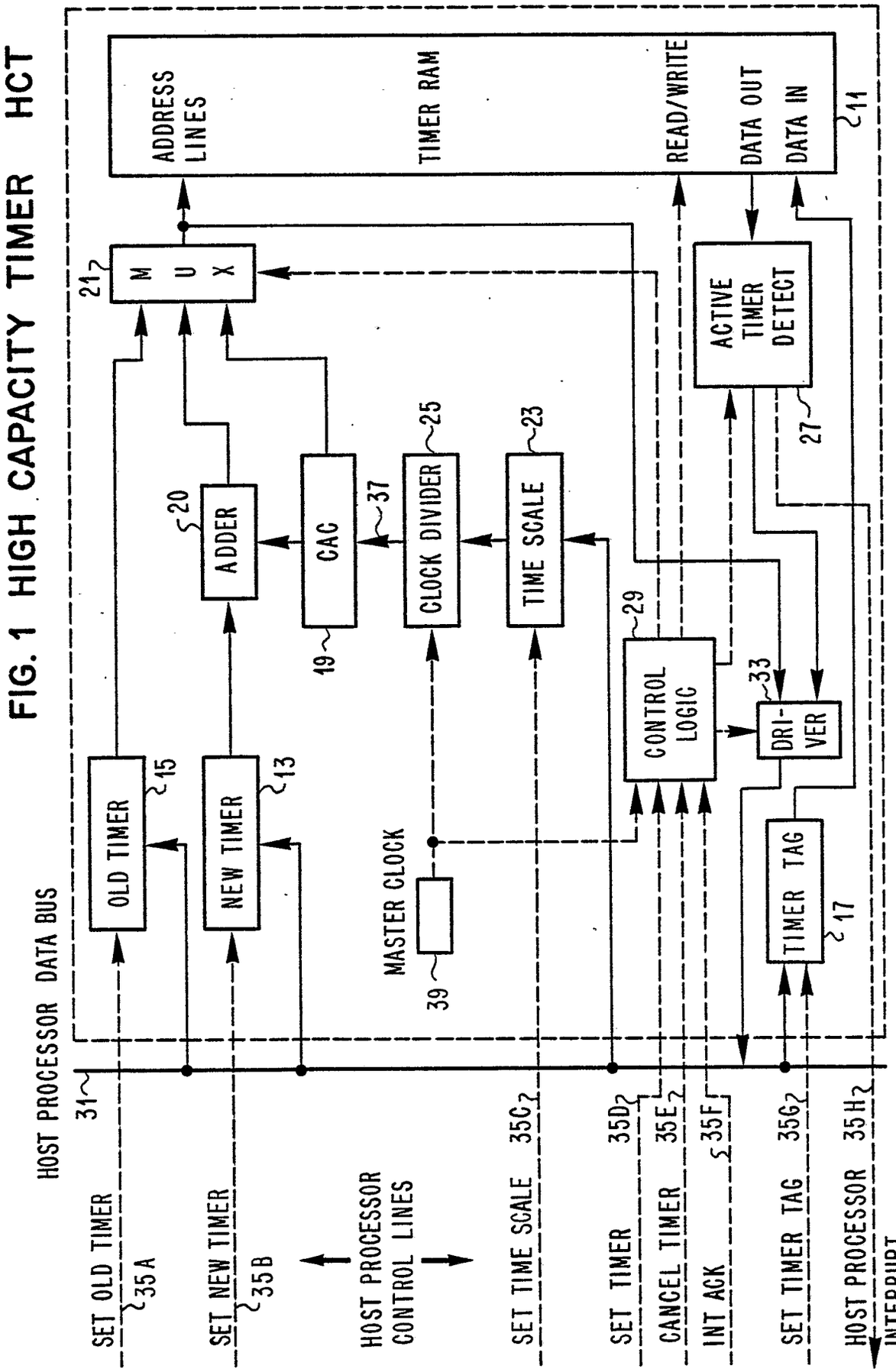
- in the timer service, concatenating the newly established timer control block (T-CB2) and the older timer control block (T-CB1) identified by said returned timer tag, without entering a timer storage location address as an intermediate reference into the newly established control block.

12. Method in accordance with Claim 11, comprising the following additional steps:
- setting, in each timer control block when it is established, an activity indicator to active status,
- for cancelling a timer, sending the respective timer reference and a cancelling request from the user to the timer service, and
- in the timer service, testing in the respective timer control block whether it has a chained status, and if yes, only changing the activity indication in said timer control block from active to inactive.

13. Method in accordance with Claim 8, comprising the following additional steps:
- addressing the timer storage locations cyclically and sequentially at regular intervals, and when an addressed location contains non-zero data, sending an expiration indication and the contents of said addressed timer storage location as timer tag to the timer service, and deleting the contents of the respective timer storage location, and
- in the timer service, accessing the timer control block designated by said timer tag, sending an expiration indication and the timer identifier stored in said timer control block to the user, and thereafter deleting said timer control block.

14. Method in accordance with claims 11 and 12, comprising the following additional steps:
- in the timer service, when receiving an expiration indication and a timer tag from the timer arrangement, testing in the respective timer control block whether it is chained, and if it is chained, accessing all chained timer control blocks, returning a stored timer identifier to the user only from those timer control blocks in which the activity status is set to active, and thereafter deleting all chained timer control blocks.

FIG. 1 HIGH CAPACITY TIMER HCT



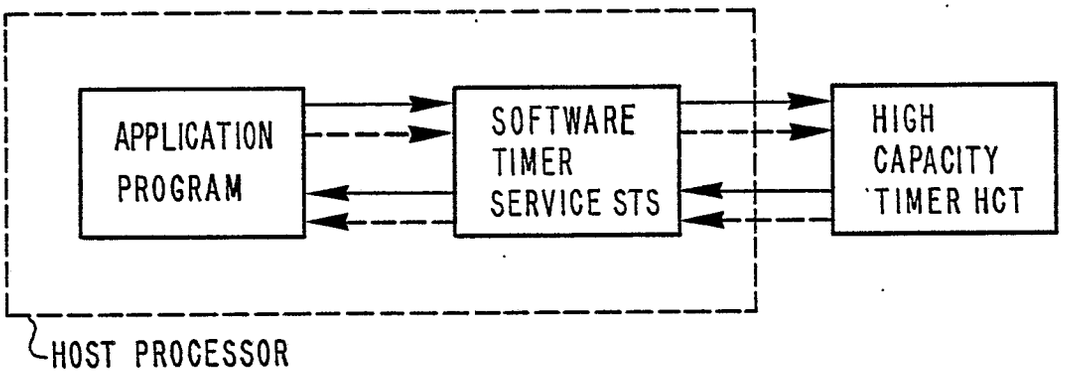


FIG. 2 HCT INTERACTION

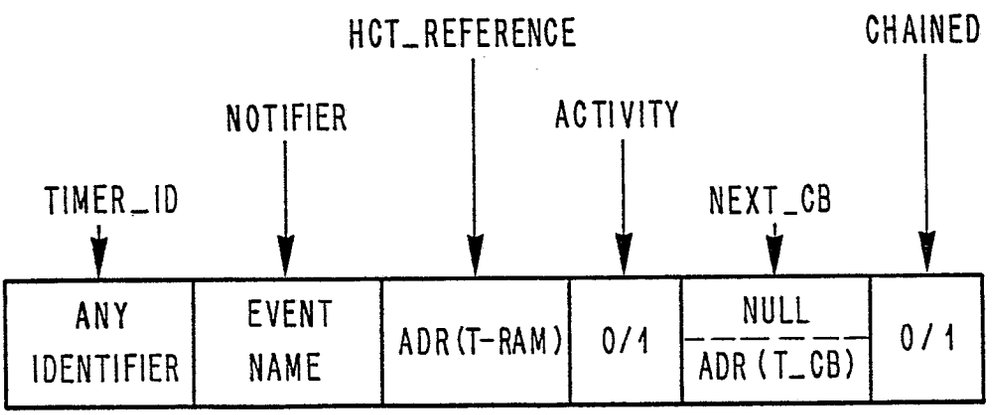


FIG. 3 TIMER CONTROL BLOCK T-CB

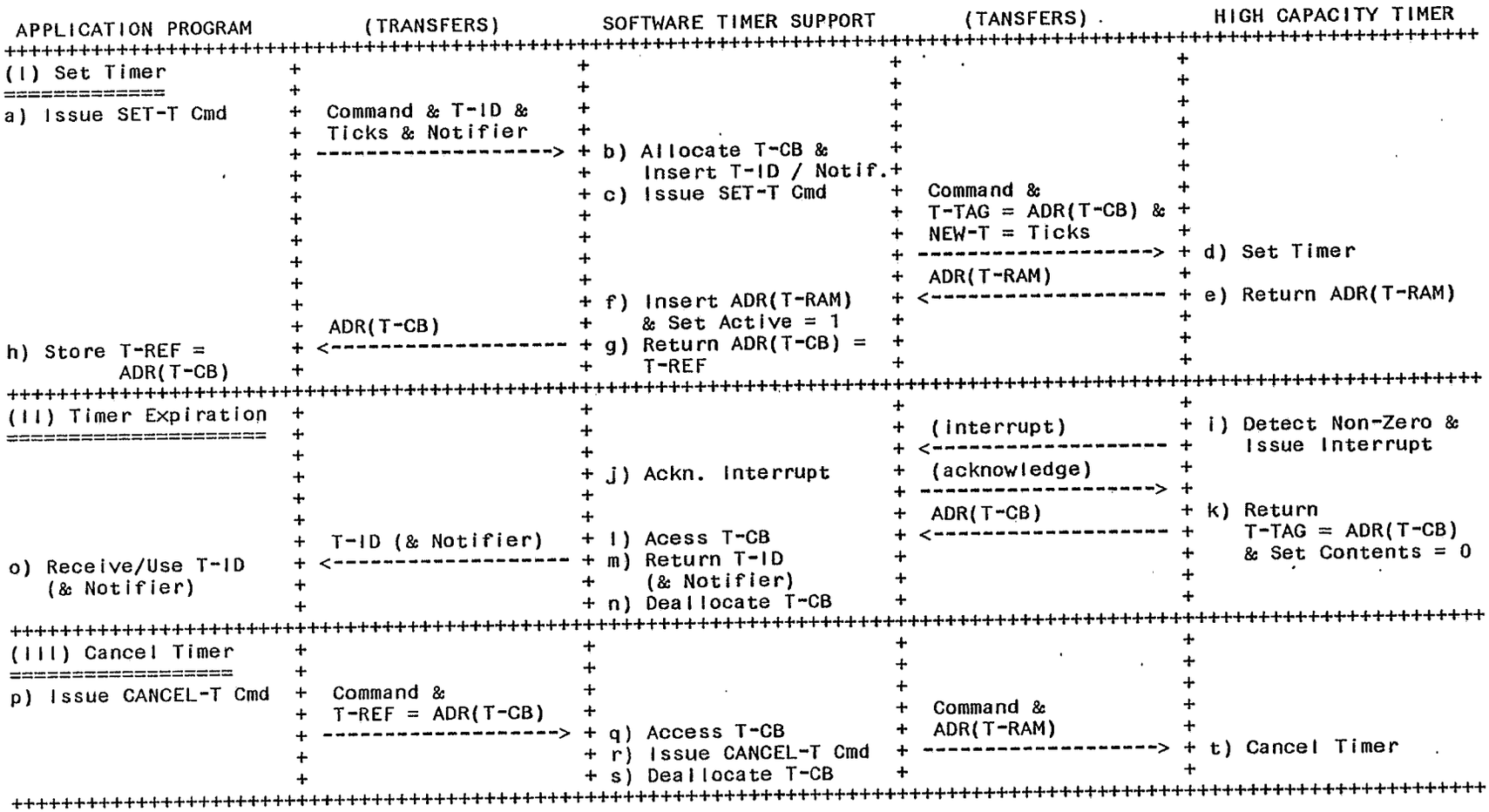


FIG. 4

Commands

SET-T = Set_Timer
CANCEL-T = Cancel_Timer

Entities

T-CB = Timer Control Block
T-RAM = Timer RAM Storage

Parameters

T-ID = timer_identifier
T-REF = timer_reference
T-TAG = timer_tag

NEW-T = new_timer (ticks)
ADR(xx) = address of xx

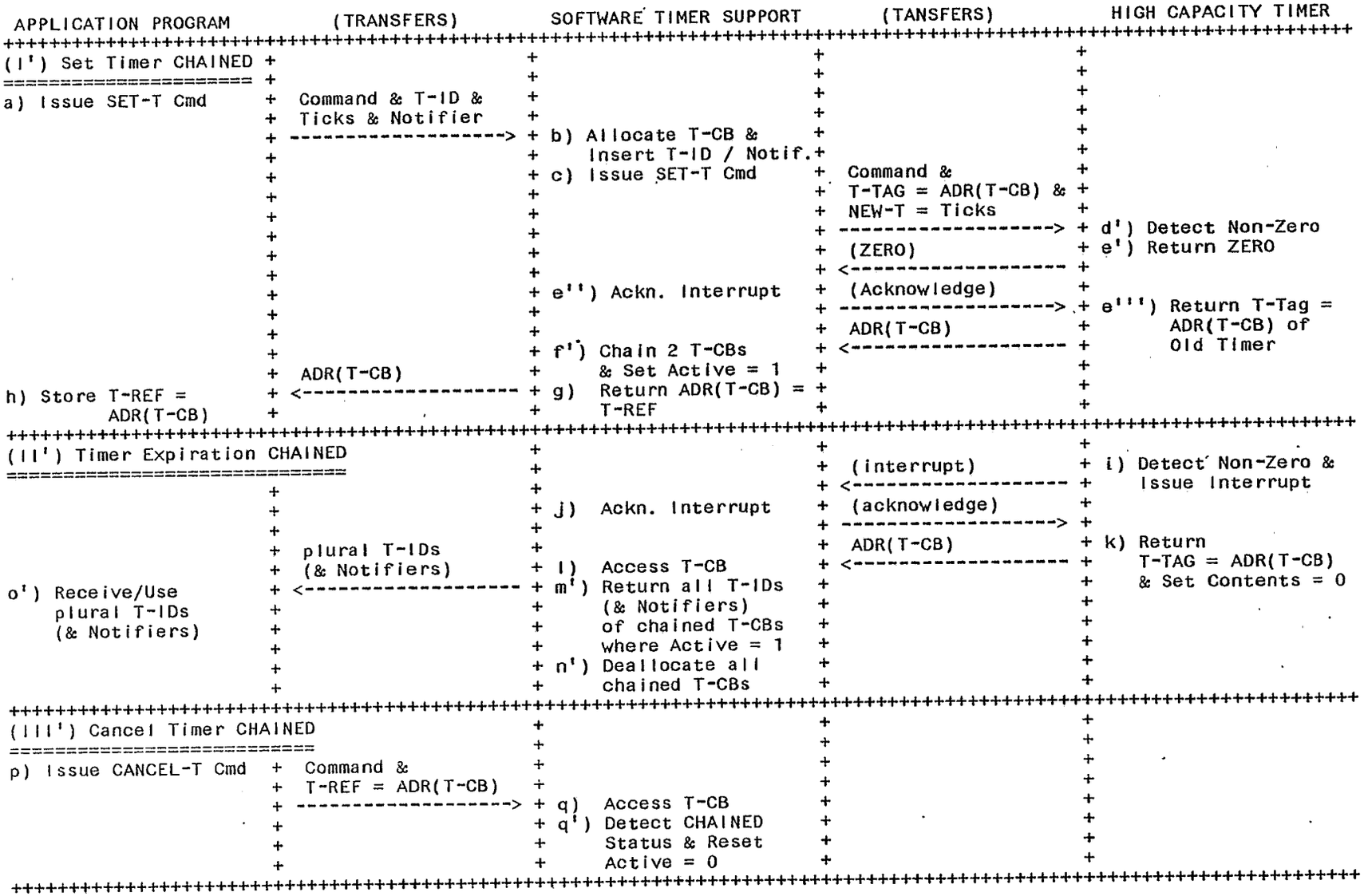
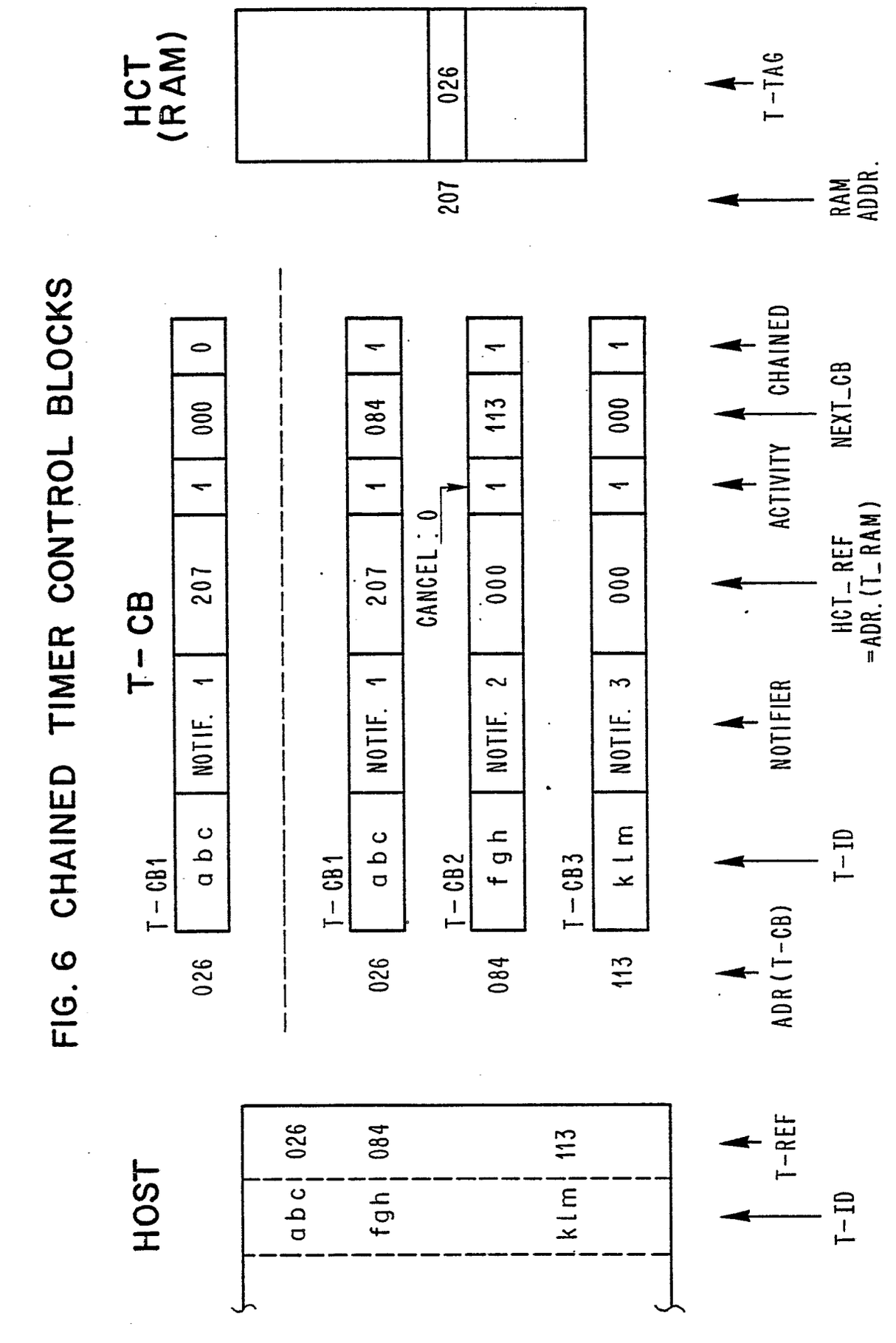

APPLICATION PROGRAM
(TRANSFERS)
SOFTWARE TIMER SUPPORT
(TANSFERS)
HIGH CAPACITY TIMER
(I') Set Timer CHAINED
a) Issue SET-T Cmd
Command & T-ID &
Ticks & Notifier
b) Allocate T-CB &
Insert T-ID / Notif.
c) Issue SET-T Cmd
Command &
T-TAG = ADR(T-CB) &
NEW-T = Ticks
d') Detect Non-Zero
e') Return ZERO
(ZERO)
e'') Ackn. Interrupt
(Acknowledge)
e''') Return T-Tag =
ADR(T-CB) of
Old Timer
ADR(T-CB)
f') Chain 2 T-CBs
& Set Active = 1
h) Store T-REF =
ADR(T-CB)
g) Return ADR(T-CB) =
T-REF
(II') Timer Expiration CHAINED
(interrupt)
i) Detect Non-Zero &
Issue Interrupt
j) Ackn. Interrupt
(acknowledge)
ADR(T-CB)
k) Return
T-TAG = ADR(T-CB)
& Set Contents = 0
plural T-IDs
(& Notifiers)
l) Access T-CB
m') Return all T-IDs
(& Notifiers)
of chained T-CBs
where Active = 1
o') Receive/Use
plural T-IDs
(& Notifiers)
n') Deallocate all
chained T-CBs
(III') Cancel Timer CHAINED
p) Issue CANCEL-T Cmd
Command &
T-REF = ADR(T-CB)
q) Access T-CB
q') Detect CHAINED
Status & Reset
Active = 0

FIG. 5

FIG. 6 CHAINED TIMER CONTROL BLOCKS



European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 88 81 0591

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.4)
A	US-A-4 193 120 (YELLO) * Figure 1A; column 2, lines 12-15; column 4, lines 61-60; column 6, lines 47-52; column 14, lines 51-61; column 15, lines 14-27,64-66 *	1,3,7,8,13	G 04 G 15/00
A	US-A-4 079 366 (WONG) * Figures 1,2,4; column 4, lines 24-39; column 5, lines 3-9; column 6, lines 64-68; column 7, line 18 - column 8, line 19, lines 39-64 *	1,3,7,8	
A	US-A-4 677 541 (SINGHI) * Whole document *	1,3,7,8,13	
A	PATENT ABSTRACTS OF JAPAN, vol. 10, no. 324 (P-512)[2380], 5th November 1986; & JP-A-61 130 893 (NEC CORP.) 18-06-1986	1-3	

TECHNICAL FIELDS SEARCHED (Int. Cl.4)

G 04 G 15

The present search report has been drawn up for all claims.

Place of search	Date of completion of the search	Examiner
THE HAGUE	25-04-1989	WEISS P.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)