

The invention relates to the use of memories and more particularly to a method and the corresponding device for updating information in a memory and to their use in memory cards. Memories of this kind are described for example in an article entitled "Memories", ELECTRONIQUE RADIO PLANS, No. 511 June 1990, Paris, France, pp. 25-31.

In memory cards enabling the information written into them to be updated, memories of the RAM or EEPROM type in particular, in contradiction to the EPROM memories, in which updates by erasure and re-recording is not possible, a major problem is that relating to the possible interruptions in the feed during the update. The fact is that if an interruption in the feed voltage occurs during the update of an item of information the old value of this latter may only have been partially erased or the new one only partially written into the memory.

The problem is particularly prevalent in updating computer memories. In this sphere solutions have been found. They often involve the use of feed systems without interruptions or of particular methods for the control of access to the information by means of software. An example of such a method is described in Specification EP-A-D 340 981.

Such solutions cannot be adopted in the sphere of

memory cards, as it is impossible to maintain the supply without interruption, and since resources are very limited where the size of the program is concerned the only solution that can be considered is check a posteriori of the erased or written data during the final application in order to detect the errors and to correct them if they exist.

The purpose of the invention is to provide a method and the corresponding device for updating information in a memory, rendering it possible to guarantee that the information to be updated is in a known "pre-update" or "post-update" state but never in an indeterminate state, with the use of a minimum of resources, thus ensuring that if an interruption has occurred, no matter at what moment, the transaction can afterwards be resumed without ambiguity.

The invention also relates to the device designed to operate the said method.

Again, the invention covers the use of such a method and such a device in memory cards.

The invention will be understood more clearly and further characteristics thereof will be indicated by means of the description given hereinafter and by reference to the accompanying drawings.

Figures 1 and 2 respectively illustrate the update stage and the device with which it is carried out according to a first embodiment.

Figures 3 and 4 respectively illustrate the update

stage and the device with which it is carried out according to a second embodiment.

In a re-recordable memory, a memory of the EEPROM or RAM type, the safeguarding of a data writing operation is in most cases only carried out after a prior re-zeroing of the corresponding memory location. Furthermore, the writing operation generally cannot be effected all at once: an item of data is written in bits, octets or words. Consequently, as indicated in the foregoing, the data can only be partially written when owing to an interruption, the sequence of operations necessary for the entry of the data is broken. Such a state for the incorrect data cannot be distinguished from that of a correct item of data, and the use of such incorrect data in an application, e.g. based on a memory card, may have highly prejudicial consequences.

In order to solve the problem of the interruptions which may occur in the course of a memory update, according to a first characteristic of the invention, the update method makes use of a logic indicator which is characteristic of the update and which itself is not updated until after the information to be updated has been written in the memory and then verified. This enables the completeness of the information to be guaranteed even in the case of sudden unexpected interruption while at the same time enabling the aborted operation to be resumed if necessary.

Furthermore, according to a second characteristic of

the invention, in order never to lose track of the previous data in the course of an update, no new data value can be written in the same physical location as the value which it is required to replace. Use is therefore made of at least two memory locations for each one and the same item of data. At the moment when the new value of the data in the memory is written one of the locations contains a value which will henceforward be termed "guaranteed current value", the other containing one termed "non-guaranteed working value".

In a first variant there is associated with each new update a logic indicator associated with it. The updated item of data, which is not transferred, is read in the location where it has been written. If more than two memory locations are used a number of values may be of the "guaranteed current value" type, and it is the order in which the automation system using the data will be able to access these different values that will enable the last guaranteed value to be traced, if necessary, after an interruption.

For this purpose, in this first variant, described hereinafter in greater detail, the device makes use of a cyclic binary counter without indeterminate state, forming a series of logic indicators, and a table of memory locations in which the information is stored.

The cyclic binary counter without indeterminate state is a simple binary logic chain, e.g. a chain of bits in an EPROM or EEPROM memory, designed to ensure that when a bit

passes from the virgin state arbitrarily termed "0" none of the other bits in the chain will be altered. Conversely, when the counter is to be re-zeroed, being virgin for all the bits, the entire chain of bits without exception is affected and all the bits together resume this virgin state, being in an identical state at any given moment during this re-zeroing.

For application to a memory in which it is impossible for one single bit of an octet to be updated without disturbing the others, a chain of octets can be used as logical update indicators.

Memories enabling a logical binary chain of this kind to be constructed are available on the market: the functionality described in the foregoing is standard in most EPROM memories, the simultaneous erasure being effected by exposure to ultra-violet radiation. This functionality likewise exists in certain EEPROM memories mainly used in the monetary sphere and manufactured by SGS THOMSON or by MOTOROLA, i.e. in most memories particularly suitable for memory cards, where each bit of each octet changes from the virgin state "0" to the programmed state "1" without any change in any other bit of the octet.

The association of the cyclic binary counter and a table of memory locations for the data requiring updates makes possible to eliminate the chance of the indeterminate state during interruptions in the electricity supply.

The following detailed description of the update method according to the invention explains the

concatenation of the operations enabling this result to be obtained: let D_0 be the original value written into the memory and D_1 the update value of the data D . As indicated in the foregoing, the data D is written in another memory location than that in which the original value of the data is present: let E_0 be the memory location where the original value D_0 is to be found and E_1 the memory location where the value D_1 will be stored. There is no constraint with regard to the arrangement of the memory locations E_0 and E_1 . It is purely in order to simplify the explanation that these locations will appear contiguous hereinafter.

In a first example we shall assume that the memory is re-recordable: only two memory locations can be allocated to a data update, i.e. E_0 and E_1 .

The cyclic binary counter consists of logic indicators, e.g. binary elements. Let $b_1, \dots, b_n$ be the logic indicators constituting the counter; the only possible values for these indicators are "0" and "1".

For access to information of a length of " N " octets the automation system functions as follows, this description being given by reference to Figure 1. The initial state of the automation system is such that

- $b_0, \dots, b_n = (0, \dots)$
- E_0 contains D_0 and b_0 is placed at "1";
- E_1 has an indeterminate content.

When an update of the data D is requested at a moment t_1 the corresponding automation system is called upon.

The first octet of D_1 is written in the first octet of

E1 at the moment $t_1 + 1$, the second octet of E1 at the moment $t_1 + 2$ and the N^{th} octet of D1 in the N^{th} octet of E1 at the moment $t_1 + N$.

- at the moment $t_1 + N + 1$ the automation system reads the value present in the location E1

- at the moment $t_1 + N + 2$ the value L read in the location E1 is compared with the value D1 of the data D to be updated;

- at the moment $t_1 + N + 3$:

- if the terms of the preceding comparisons are equal the update is terminated and the automation system writes the first virgin bit b_1 of the counter, i.e. b_1 of this first update, which then passes to 1,
- if the terms of the comparison are unequal the writing of the value d_1 in the location E1 was unsatisfactory and the automation system returns an error code.

In the subsequent update of the same data D the value D2 of the data will be written by the same procedure in the location E0 and then verified, the first virgin bit of the counter in the sequence of the bits of the counter being then written (i.e. b_2 , during this second update) when the written value D2 has been verified.

During the I^{th} update of the same data D required at the moment t , the value D_i of this data will still be written by the same procedure, i.e. in E0 if i is even and in E1 if i is odd, the first virgin bit of the counter, b_i ,

being written when the written value D_i has been verified.

The chain of indicator bits ($b_0, b_1, \dots, b_n$) has a length $n + 1$. When the $(n + 1)^{\text{th}}$ update is reached then chain of the indicators is completely at 1, the writing then being forced to the location E_0 and the chain of indicators erased all at once, after which b_0 is placed at "1" following the verification of the data written at E_0 . The device is then returned to a state analogous to the initial state. This operation of erasing the indicators is such that all the bits b_i are re-zeroed simultaneously by one single operation of the ultra-violet radiation exposure type, for example.

For a reading of the value of the data D required at a moment t^1 , the automation systems searches, in the sequence of bits of the counter ($b_0 \dots b_n$), the last bit " b_i " at "1": if the value of " i " is even or zero the value of the data D is read in the location E_0 , while if the value of " i " is odd the value of the data D is read in E_1 .

At the reading a process of this kind can only provide a correct value, as the completeness of the writing has been validated. At worst, when it has not been possible to complete an update, the automation system reads the value written in the last correct update and thus possibly a non-updated but never an incorrect value.

Needless to say, a number of different locations can be allowed for one and the same item of data above 2, i.e. $E_0, E_1, \dots, E_m$, with $m < n$, wherein $M + 1$ is the number of locations and $n + 1$ is the number of bits of the counter;

this makes it possible as explained above, to have a number of guaranteed values in progress in the memory locations used in succession.

In the erasable memories of the EEPROM type the possibility of erasing the memory may be profitably utilized in order to operate cyclically on a number of locations $E_0, E_1, E_2, \dots, E_m$: when the automation system detects that the validated value of the updated item of data has been written in the last memory location E_m it writes the subsequent value in the location E_0 : the number b_i of bits written in the counter is such that $i = m + 1 = 0 \text{ modulo } (m + 1)$. For the cyclic use of the memory locations in this way the binary counter enables the location E_k of the last correct value to be found by simply calculating the value $K = i \text{ modulo } (m + 1)$, i being the rank of the last bit at "1" of the chain and assuming the locations to be written in the order $E_0, E_1, \dots, E_m$.

A structure of the kind also enables data to be updated in electrically non-erasable memories such as those of the EPROM type. In this case, when all the locations $E_0, E_1, \dots, E_m$ or all the bits $b_0, b_1, \dots, b_n$ have been written the memory has to be replaced by replacing the card bearing this memory or by allocating new zones of the memory to the updating of this item of data. In this mode it is of advantage for the number of memory locations to be the same as the number of bits of the counter.

The update method described above also applies to the updating memories in which the writing operation is

preceded by an explicit or implicit erasure of the designated location.

To sum up, if " b_i " is the last bit at "1" in the chain of indicators, i being its rank in the chain, the process is such that the automation system gives reading access in the location E_j , which is such that $j = i \text{ modulo } (m + 1)$, and writing access in the location $E_j + 1$. In principle, there is little risk of bits remaining at 1 in a complete erasure of the chain after a cycle of m updates. However, if such an anomaly occurs, e.g. if the erasure operation is prematurely interrupted, there is a risk of error in the automation system in the selection of the location to be read. This risk can be eliminated by simply testing the chain, for the only normal configurations are those of the type "all the b_i bits are at '0'" or "all the bits preceding the last bit b_i at 1 are at 1." All the mixed configurations are abnormal and can only correspond to an attempt to zero the chain of indicators; in this case, as in all "abnormal" cases, the location to be read is E_0 .

The only case of erasure which could not be detected as abnormal when it is in actual fact incomplete is that in which all the bits at "1" which have been erased are on the right of a chain of bits which have otherwise remained at "1", as this configuration is not "abnormal" from the logical point of view and cannot be detected by checking the form of the chain.

To eliminate all risk of error, the same method as previously can be used to check the erasure of the chain of

bits, i.e. a check indicator bit, or Cx associated with the chain of bits bi:

- when the chain of indicators is in normal use, Cx remains in the initial state: virgin = conventionally "0";
- when the chain has to be re-zeroed the bit Cx is placed in the state "1", which means that the chain of indicators is in an intermediate state before re-zeroing. The chain is then erased and its erasure verified, and if the erasure is correct (no bit b: at 1) the check bit associated with the chain Cx is re-zeroed.

A sequence of this kind enables all doubt to be eliminated in cases of sudden unexpected interruption and makes it possible, in the event of an anomaly, to repeat the aborted operation, an error code being returned by the update automation system as soon as it has proved impossible to verify an operation. This improvement is not illustrated in Figure 1.

The update device designed for the operation of the method described in the foregoing is illustrated in Figure 2: programmable processor 10 associated with a data memory 20. In this Figure 2 it was assumed that these elements, microprocessor and memory, were associated on a memory and microprocessor card 30. But this microprocessor, in the automation system intended for the application called up by this card, may be outside the card. The chain of indicators 40 has been shown as part of a memory 50

independent of the memory 20. It may also be part of the memory 20, provided this memory has the functionality described in the foregoing. The procedure is then controlled by a software system effecting the stages of the process which are described above and may for this purpose be provided in the microprocessor. But the update automation system may likewise be constructed by the aid of specialized circuits effecting the elementary operations described above. These two modes of operation are within the scope of the man of the art.

In Figure 2 the memory 20 has been illustrated with a table of memory locations TD (E0, E1,.....E5), i.e. $m=5$, and the chain of indicators 40, with 12 indicators ($b_0, b_1, \dots, b_{11}$). In the example illustrated in which ($b_0, b_1, \dots, b_9$) are at "1" and (b_{10}, b_{11}) at "0", if a reading is required the item of data will be read at the location E3, while if an update is required the item of data will be written in the location E4. As indicated above, no data transfer takes place in this first variant, while on the other hand the item of data must be read in the ad hoc location following a test on the indicator bits.

A variant of the present invention likewise uses two memory locations, but in order to ensure that the item of data will always be read in the same memory location E, the value of the updated item of data written in another memory emplacement and then verified is then transferred thereto. The process described hereinafter in greater detail by reference to Figure 3 consists of a request for an update

for an item of data D of which an initial value D0 has been written in the location E by writing, in a memory space M provided for this purpose,

- the value of D1 of the updated item of data,
 - the address (E) at which it is to be written, and
 - an indicator in a state 0, for example, for "incorrect" indicating that the value of the data has to be updated. The indicator will take the value "0" a value different from that of the initial state of the memory, the value "1" itself being equal to the initial state of the memory.
- If the application only necessitates the updating of an item of data using this memory space it is possible to dispense with the storing of the address, which is always the same and which is an item of data for the update.

All these items of information having been written in the memory space provided for the purpose, a reading of this space is effected in order to verify the written data:

- in the event of a writing error or an interruption, the verification does not enable the coincidence to be established, and the value of the data at the address E is not updated. An error code is transmitted and causes the update operation to be resumed by a new recording;
- in the event of coincidence, the data present in the memory space is recopied at the associated address, i.e. at the location E.

For this purpose the automation system for the operation of the card effects the following sequence of operations in a manner transparent to the user, illustrated in Figure 3:

- In a first stage the state of the indicator b is detected:
 - if the indicator is at "1" ("true") no update is required and no incomplete update has been carried out, and the value of the useful data can be read at its address E;
 - if the indicator is at "0" ("false") an update is in progress or an update is required, and the rest of this update operation is carried out as follows:
 - erasure of the value of the data D present in the location E;
 - the data D1, present in the memory space M where it has been verified, is written in the location E;
 - the indicator is positioned at 1 ("true"), indicating that the update in progress is terminated and that the value of the data D at its address E is a guaranteed value.

This variant likewise makes it possible to have, in the useful memory for the application controlled by the card, data which are invariably coherent. The fact is that after the card has received the update order the process described above enables the completeness of the data to be

preserved. If the last part of the sequence relating to the transfer of the data to its definitive memory location is interrupted, e.g. by an interruption in the supply, the sequence will be resumed when the supply is restored, because the indicator is always in the position "0" ("false") and shows that an update is required or has been left incomplete, the new value being the memory space not accessible for reading. The device always has a coherent value available, since the old value is never erased without the system knowing the new value, and the updated value always ends by being transferred prior to a reading.

As in the first variant described in the foregoing, for this second variant the device illustrated in Figure 4 is such that when the memory is associated with a microprocessor, e.g. in the application to cards with a memory (20) and with a microprocessor (10), the automation system can be produced by software and can be present in the mask for the microprocessor. It may likewise be produced by specific circuits, the automation system then produced exercising the control over the program of the processor in order to effect this updating sequence.

The device may nevertheless equally well be implanted on simple memory components, only using a wired logic system (PLA). This is then a circuit coupled to the reading and writing control inputs of the memory.

The invention is not confined to the two variants specifically described above: it is applied to the update of a memory word but can also be applied to the update of

a memory zone of smaller dimension or on the other hand to a wide assembly of several memorization "units", words or octets. In all cases the new value of the data is written in a memory location different from that in which the preceding value is available, this value being verified before the old value can be replaced in the application, a logic indicator changing state in order to mark the end of the update and the reading accessibility to the new value.

**CLAIMS**

1. Method for updating information in a memory, comprising the following stages:

- an initial or proceeding information value (D0) has been written into the memory from a first memory location (E0);

- in a first stage the writing of the new value (D1) is effected in a different memory location (E1) in the memory and is associated with a chain of logic indicators initially in a first logic state and maintained in this state in the course of the updatings;

- in a second stage a verification of the new written value is carried out, this value being guaranteed in the event of coincidence with the value of which the writing has been affected and an error code being emitted in the event of non-coincidence; and

- in a final stage the end of the updating causes the logic indicator to pass to a second logic state, the update value then being accessible for reading, characterised by the fact that, the memory being erasable and re-recordable and comprising two memory locations (E0, E1) for each of the items of information capable of being updated,

- the locations are written alternately during successive updates of the information;

- each operation of writing a new value of the information causes after it has been verified the change of state of the first bit still in the first state in the chain;

- the reading of the last value of the information, updated and verified, is effected in one or other of the two memory locations, according to the parity of the range of the last bit in the second state of the chain of indicators.

2. Method according to claim 1, characterised in



that, the memory comprising, for each of the items of information capable of being updated, a memory location (E) in which this data is always read, the initial or preceding value being written in this location and another memory location only used for writing the update values,

- while after the second stage of the verification of the data written in the other memory location the data verified is transferred to the location (E), where it will be read and where it replaces the preceding value.

3. Method according to claim 2, characterised in that, the other memory location intended for the update being used for different items of information to be updated, the writing of the new value of the data in the first stage is completed by the writing of the address (A(E)) of the memory location (E) to which it will be transferred.

4. Method according to either of claims 2 or 3, characterised in that the logic indicator associated with the writing of the new value is positioned in the first logic state at the beginning of the first stage of the update in progress.

5. Method according to any one of claims 1 to 4, characterised in that

- when the chain of the logic indicators is entirely in the second state, this chain of indicators is completely returned to the first stage in one single erasure operation;

- the storage of the update value of the information following such an erasure is effected in the first location of a table of locations, the first logic indicator of the chain changing its state after the first verification of the written value.

6. Method according to claim 5, characterised by the fact that

- before the erasure of the chain of logic



indicators a check bit initially in a first state is brought to a second state;

- the check bit is returned to the first state after the complete erasure of the chain has been verified.

7. Updating device intended for the implementation of the method according to any one of the preceding claims, characterised by the fact that it comprises an automation system (10) designed for controlling the different stages of the update of the memory (20) with which it is associated.

8. Device according to claim 7, characterised in that the automation system constructed in the form of a software system set in operation by a microprocessor associated with the memory (20) to be updated.

9. Device according to claim 7, characterised in that the automation system is a circuit coupled to the control input of the memory (20).

10. Device according to claim 7, characterised by the fact that a chain of logic indicators without an indeterminate state (40) is operated for the updating of an item of information (D), this chain of indicators being a succession of binary elements without indeterminate state, such that the change of state from an element of the first to the second state is carried out without affecting other elements, while conversely all the binary elements pass from the first state to the second state together by simultaneous erasure of the chain.

11. Use of the updating method according to any one of claims 1 to 9 in a memory card.

12. Use according to claim 11, characterised by the fact that the memory card comprises a microprocessor (10), the method for updating the information in the memory (20) being implemented by a logic system of the microprocessor (10) of the card (30).