(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 June 2003 (19.06.2003)

PCT

(10) International Publication Number
WO 03/050686 A1

(51) International Patent Classification[7]: G06F 12/14

(21) International Application Number: PCT/IB02/05294

(22) International Filing Date:
11 December 2002 (11.12.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
01/16054 12 December 2001 (12.12.2001) FR

(71) Applicant *(for all designated States except US)*: **SCHLUMBERGER SYSTEMES** [FR/FR]; 50, avenue Jean Jaurès, F-92120 Montrouge (FR).

(71) Applicant *(for MC only)*: **SCHLUMBERGER MALCO, INC** [US/US]; 9800 Reistertown, Owing Mills, MD 21117 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BOUDOU, Alain** [FR/FR]; 12, rue du Moulin, F-78930 Vert (FR). **SIEGELIN, Christoph** [DE/FR]; 32, rue Ginoux, F-75015 Paris (FR). **MARCHETAUX, Jean-Claude** [FR/FR]; 24, rue des Remparts, F-78940 La Queue Lez Yvelines (FR).

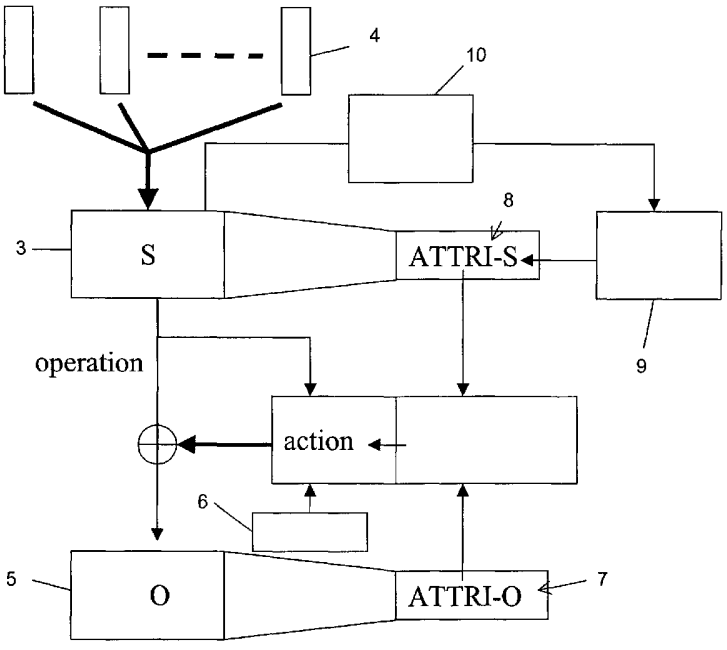
(74) Common Representative: **SCHLUMBERGER SYSTEMES**; c/o Renault, Patricia, 36-38, rue de la Princesse, BP 45, F-78431 Louveciennes (FR).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) Title: ACCESS CONTROL METHOD AND DEVICE IN AN EMBEDDED SYSTEM



(57) Abstract: This invention concerns a method to control the access to objects (5) stored in a memory space of a computer system by a subject (3) stored in a program space of said system. The subject wants to perform an operation on said objects (5). The method according to the invention controls the access to said objects (5) via a dynamic attribute (8) linked to said subject (3) whose value is updated according to the present and previous status(es) of the subject. This invention also concerns the access control device implementing the method described above.

WO 03/050686 A1

Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:

— *of inventorship (Rule 4.17(iv)) for US only*

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

ACCESS CONTROL METHOD AND DEVICE IN AN EMBEDDED SYSTEM

This invention concerns a method and a device to control the access to memory areas attached to applications and software modules of an electronic unit with microprocessor, in particular a microcontroller. The invention is for example applicable to, but not limited to, electronic smartcards.

TECHNICAL FIELD

The smartcard like all embedded computer systems includes a processor, inputs/outputs and memory which is divided into non volatile memory generally used to store programs and data, and volatile memory generally used as processing memory. The embedded systems or portable objects and in particular the smartcards are now performing more and more complex functions due, in particular, to the storage of multiple-application programs: these smartcards are designated by the term "multi-application smartcard".

Regarding the security of multi-application smartcards, isolation of the applications from each other plays an essential role. In a smartcard containing several applications therefore, it is important to be able to deny applications access to data which does not belong to them to prevent malicious use or transmission of information to the outside. For example, it may be necessary to deny a first application access to confidential data belonging to a second application such as cryptographic keys. In particular, since the data is stored permanently in non volatile memory, it may be necessary to restrict reading and updating to authorised applications only.

In addition, due to the increase in memory capacity and processing power of the smartcards, software architecture is becoming more structured,

with functions isolated from each other as modules and distributed on various levels.

For example therefore, in a software architecture for smartcard, we can identify, as shown on figure 1, the core layer "CORE" interfacing directly with the hardware (and comprising in particular the generic non volatile memory access routines), the system layer "OS" concerning the operating system which manages the shared resources and services and the application layer "APPLI" grouping the various software applications. Each layer can be subdivided into sublayers and functional modules: the system layer includes, for example, functional modules such as the cryptographic algorithm processing service, management of memory and input/output resources.

In this type of model, an application no longer accesses its data in memory directly, but via functions located in the core and system layers. The problem of restricting access to stored data to authorised applications only arises, in particular, in this context of indirect accesses.

More generally, a multi-layer modular software architecture can be designed in which firstly a module calls one or more modules to execute some of its tasks and secondly each module has a program and data stored in non volatile memory in clearly defined areas, data for which the handling or use must be restricted to authorised modules only.

PRIOR ART

Generally, memory protection mechanisms can be produced in a software and/or hardware form for interpreted applications and in a hardware form for executed applications as windows opened on the memory (such as a segmentation device) or as semi-static access matrices. Applications are associated with memory areas either during configuration or when selecting

an application. Such mechanisms only allow memory access for predetermined code / data area pairs.

For example, with an access matrix, memory pages are associated with applications when the card is configured by programming page attributes; the role of the hardware at the time of execution is limited to a simple comparison between the identity of the page "owner" and the identity of the module trying to access this page (known by the hardware, for example by the position of the program counter). The disadvantage with these access matrix mechanisms is due to the fact that they require a direct relation between program and data: this relation exists as long as the page owner accesses its data directly (via the microprocessor read / write instructions), but it disappears in configurations where a module other than the owner wants to access the data "in its name". Two examples illustrating this situation are (i) a virtual machine which accesses the data of applications "in their names" and (ii) an EEPROM manager module between the application and the non volatile memory. These "indirect relation" accesses cannot be protected by access matrix type devices.

The invention aims to overcome the disadvantages of the devices and systems of the known state of the art whilst meeting the requirements which arise.

In the context of a multi-layer modular software architecture of a computer system and more particularly in the following description of an embedded system such as a smartcard, the objective of the invention is therefore to propose a method capable of isolating and protecting memory areas attached to said modules.

Still in this context, the objective of the invention is to propose a method capable of controlling the access to the memory and of restricting access to authorised modules only.

SUMMARY OF THE INVENTION

This invention concerns a method to control the access to objects stored in a memory space of a computer system by a subject stored in a program space of said system, said subject wanting to perform an operation on said objects, characterised in that it controls the access to said objects via a dynamic attribute linked to said subject whose value is updated according to the present and previous status(es) of the subject.

This invention also concerns an access control device implementing the method described above as well as an embedded system using such a device. It also concerns the program implementing said method.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described in more detail, referring to the attached drawings, amongst which:

- figure 1 is a diagrammatic representation of a multi-layer modular software architecture of a computer system;
- figure 2 is a flowchart illustrating the main steps of one form of realisation of the access control method according to this invention;
- figure 3 is a diagrammatic representation of an example of chaining modules and memory areas to which said modules want to access;
- figure 4 is a diagram representing an example of chaining modules and memory areas to which said modules have the right to access;
- figure 5 is a flowchart illustrating the steps of the access control method according to figure 2 according to two realisation variants, one shown in solid lines the other in solid and dotted lines;
- figure 6 illustrates a system according to figure 1, showing the access control method.

BEST WAY OF REALISING THE INVENTION

In a computer system 1 with multi-layer modular software architecture as illustrated by figure 1, a module 2, α, β, completes a specified task either upon request from the exterior using a set of input/output commands, upon request by another module, or following a hardware event. A module capable of executing a task upon request from the exterior is hereafter referred to as an application. Execution of said task corresponds to execution of a series of instructions called program (A, B), stored in non volatile memory in a space, called "PROG" space (program space) specific to said module. The series of instructions applies to a set of data stored in non volatile memory in another space, called "DATA" space (data space), respectively the spaces a and b for the programs A and B. The data spaces a and b consist in spaces for which said modules α and β have access rights. As an illustration, on figure 1, activation of module α corresponds to execution of program A stored in the program space PROG; said program A applies to data stored in the data space a. If module α has a right granted by module β, its program accesses the data space b of module B (dotted arrow on figure 1).

While executing its task, one module can call other modules. Consequently, the programs are executed successively, the called module succeeding the calling module.

In the context of system 1 described above, the method according to the invention consists not only of isolating the various software applications but also of isolating all modules in the system (irrespective of its level): the method must ensure that a module in the application layer cannot access data it is not authorised to access via calls to modules in lower layers. The method according to the invention must therefore restrict access to memory spaces to the authorised modules 2 only.

The notion of access control is represented on figure 2: an access control accepts or rejects an operation carried out by an active entity 3 known as the subject "S" stored in the space PROG (for example a processing process) executing in the name of a user 4, on passive entities 5 known as objects "O" of the memory space DATA (for example a container of stored data) or PROG according to rules 6 characterising the policy applied, security attributes 7, 8 associated firstly with the targeted object 5 and secondly with the subject 3 executing for a user 4 and the type of operation. The subject consists of one or more modules 2 executed successively to perform a given operation.

Embedded software such as that, for example, in multi-application smartcards can be described as a subject 3 which firstly runs for external users 4 requiring the various applications and secondly carries out access type operations in non volatile memory.

In the context of this embedded software modular architecture, the invention proposes a method and a device to control the access to objects 5, characterised in that it is based on dynamic attributes 8 linked to the subject 3, whose value depends on the history of the subject, i.e. on its present and past statuses. The method restricts the access to objects to authorised modules 2 only, even when these accesses are indirect, i.e. occurring via one or more other modules.

The method includes the following steps:

- a management step 9 to manage the subject attribute 8, according to the present status and past statuses of the subject 3;
- a step to compare an item of information characterising the targeted object which is the attribute 7 of the object and an item of information characterising the present status and the past statuses of the subject which is the subject attribute 8;

- a filtering step to accept or reject the operation depending on said comparison and optionally on the type of operation and/or on predefined rules 6.

In the following description, the method according to the invention will be described in detail in its application to the data space DATA, i.e. the targeted object is part of said space DATA. So that the method according to the invention is easier to understand, the management step will be described after that of comparison and filtering.

The comparison step consists more precisely in comparing an item of information characterising the targeted memory area, i.e. more precisely the targeted data space (called targeted space) and an item of information characterising the data space associated with the present and past statuses of the current program (called authorised space).

The targeted space is identified by an attribute 7. According to a first realisation variant, the attribute 7 is an item of information associated with the operation. According to a second variant, the attribute 7 is an item of information physically associated with the targeted memory area (form of realisation on figure 2). The authorised space is characterised by an attribute 8 of the same type as that used for the targeted space, in order to simplify the comparison. In the first realisation variant, the attribute is a memory address according to the known model MMU (Memory Management Unit); the current operation specifies a memory address which the subject 3 wants to access. The comparison then consists in checking that the address 7 in question is contained in a window 8 of addresses, the address window 8 corresponding to the authorised space. In the second realisation variant, the current operation selects an attribute 7 which specifies an identifier. The comparison consists in checking the identity between the identifier 7 of the targeted space and the identifier 8 of the authorised space (model MAC-Mandatory Access Control).

The filtering step consists in performing an action depending on the result of the comparison step, i.e. accept the operation if the data space targeted by the operation (the targeted space) and the data space for which access is authorised at the time of the operation (the authorised space) correspond, or reject it otherwise. Other characteristics may also be taken into consideration such as the type of operation, which extends the qualification of the targeted space, for example space accessible in read only mode (examples of possible qualification: read, write, programming (writing of 1), erase (writing of 0), or combination of these terms, supply with instructions). Additional rules 6 may also be imposed. For example, access may only be granted to one or several predetermined modules, this restriction applying in addition to the filtering depending on the result of the comparison between the targeted space and the authorised space.

The management step consists in managing the information characterising the authorised data space associated with the current program status: the method calls a mechanism 9 to initialise (considered in this description as an initial update) and update the authorised space. The authorised space is initialised or reinitialised on each change of application (change to application A on figure 4, the modules M1, M2, M3 not being applications but simple modules activated to process a task upon request by another module).

Said management depends on the functional characteristics of the architecture and the security policy to be implemented.

In the context of an embedded software modular architecture in which a module can call another module to subcontract a task, the program of the called module can access its own data and also has the privilege of being able to access the data of the data space belonging to the calling module. During a succession of inter-module calls, the privilege is inherited

from one module to another: a given data space can be accessed by an entire series of modules. In addition to its own data space, a program of a called module can therefore also access the data space of any of the earlier calling modules or of several of them.

Any data handling or use is restricted to the authorised modules only: the access control method according to the invention accepts or rejects operations on data stored in non volatile memory depending on the chaining of inter-module calls and on the choices of each program to access its own data in a transitory way.

A security policy characteristic of the smartcard specifies that a module can only use, handle or access its own data or data corresponding to the task entrusted to it. As shown by the first example (Ex. 1) on figure 3, a called module (B) inherits the privilege of accessing the data space of the task for which it is called, space which may belong to the calling module (A) or which the calling module (B) itself inherited. In the first example, the authorised space does not have to be updated since it does not change. The module may also (second example: Ex. 2) need to access its own data; in this case, the space authorised is updated using a request (explicit or implicit as illustrated on figure 5 and detailed below) by this module. When the module stops accessing its own data space, it returns to the previous context and again accesses the data space for which it received a privilege from the subcontracting call; the space authorised returns to its previous value using a request (explicit or implicit as illustrated on figure 5). In the second example, module B called by module A accesses the data of module A (arrow u) and its own data (arrow v) in a transitory way. Module C, called by module B which was itself called by module A, accesses the data of module B (arrow x) or of module A (arrow y) and its own data (arrow w) in a transitory way.

The update step carried out to authorise the chaining of accesses to targeted spaces as described above is implemented according to various forms of realisation illustrated on figure 4.

Figure 4 represents, in the example illustrated, the calls between modules A, M1, M2, M3, the spaces targeted during the calls and the changes to authorised spaces during a given period (t1, t2).

According to a first form of realisation, the update step can be carried out using a stack of authorised spaces (PILE ESP_AUT on figure 4): each module requests (either implicitly or explicitly) that its own data space is placed on or removed from the stack of authorised spaces, when it needs to access it. In the greyed-out part of figure 4 (letter "o"), before calling module M3 module M2 requests that its own space m2 is placed on the stack of authorised spaces. The stack then consists in the following spaces: a1 m2. Considering the update of the authorised space, module M3 can access space m2 (ACC_MEM). Module M3 then requests access (letter "p") to its own data m3; its own space m3 is placed on the stack of authorised spaces. The stack then consists in the following spaces: a1 m2 m3. The access control device authorises an operation on a data space when the data space targeted by the operation and the data space for which access is authorised at the time of the operation are identical: the data space authorised at the time of the comparison corresponds to the space at the top of the stack (according to the first example: space m2; according to the second example: space m3), those at the bottom of the stack (according to the first example: space a1; according to the second example: spaces a1 m2) being spaces which have been and which will be again authorised spaces.

Note that the stack of authorised spaces may be quite different from the stack of calling modules; for example, therefore, if none of the called modules accesses its own data, the stack of authorised spaces is reduced to

the data space of the first calling module (arrows r, s, t on figure 4 - data space a1).

According to a second form of realisation, a less rigorous security policy is set up. This update policy is implemented via a list type management mechanism: each module can request (either implicitly or explicitly) that its own data space is placed on or removed from the list of authorised spaces.

As illustrated simply on figure 4, each called module can potentially access its own data space and its data space is added to the list (LIST ESP_AUT). List handling is implicit and synchronised on the module calls. In this case, the list of authorised spaces is the same as the list of data spaces associated with the modules identified in the stack of calling modules to which is added the data space of the last called module.

The constraint is less rigorous than in the previous case: in fact, the update mechanism in the second form of realisation allows the operation if there is agreement between the targeted space and one of the authorised spaces on the list. The mechanism therefore allows a calling module to access data of any of the modules called before the access control.

According to a first variant of the update step illustrated on figure 5 in solid lines, the request made by a module 2 to add its data space 8 to, or remove it from, the stack or the list of authorised spaces, is explicit: the request uses one or more internal update commands (CIMJ on figure 5) of type "add (or remove) my space" or "add (or remove) a given space".

According to another realisation variant of the update step illustrated in solid and dotted lines on figure 5, the request made by a module 2 to add its data space 8 to, or remove it from, the stack or the list of authorised spaces, is implicit. The request uses an internal update command (CIMJ)

from a call interception or preliminary processing mechanism 11; said mechanism requests addition (or removal) of the data space corresponding to the targeted space by the operation carried by the call to a called module (or the return to the calling module).

To update the subject attribute according to the subject status, or more particularly, to manage the information characterising the data space associated with the status of the current processing, i.e. the authorised space, the management mechanism must process the update request corresponding to the change of subject status as shown on figure 2.

The protection offered by the access control depends on the level of security provided in the management of said control. In order to increase the security in the access control management therefore, an update management mechanism is planned as an identity check of the subject and of its rights. The request to update the subject attribute 8 is only accepted and taken into account if its issuer, a module (in this case the active module or the interception module) is reliably identified and is authorised to make this request: the subject attribute update management mechanism comprises a mechanism 10 to check the identity of the requester of said update and to check its rights.

The mechanism 10 must firstly identify the module making the update request and secondly ensure that the request is legal by checking the rights of the module on the targeted data space. The rights are data specific to the module concerned and saved beforehand.

The identity of the calling module is generally obtained by identification of the caller's program space. A patent application filed the same day as this application by the same applicant entitled "method and system for secured identification in a modular software architecture" describes such an identification mechanism.

If an explicit request is made of type "add (or remove) a given space", the right of the program space on the data space in question is saved in memory, for example in a table which connects program space(s) and multiple data space(s) according to the rights of the program spaces on the data spaces. This solution is suitable when data can be shared between several modules. For two or more modules to be able to share a data space, they simply have to share the right to update the authorised space by adding this data space.

More specifically, if an explicit request is made of type "add (or remove) my space", the correspondence between program space and data space must be saved in memory. This solution is suitable when there is bijection between program space and data space.

If the request is implicit, the right of the program space of the calling module on the targeted data space must be saved in memory, for example in a table which connects program space(s) and multiple data space(s) as previously; the table will only be consulted if the targeted space differs from the existing authorised space.

As shown on figures 5 and 6, the mechanism 9 used to initialise and update the authorised space according to the invention is initiated by an external command CE. The external command activates ("ACTIV" on the figures) one of the modules 2 of system 1. Activation of one of said modules initialises ("Init") the authorised space to a particular value allowing the first module called by said external command to access the required memory area. When one module calls another module, the method according to the invention described above is executed.

The method according to the invention has been described in its application to control access to the data memory space. Similarly, the

method according to the invention can be used to control access to the program memory space.

As an example, the case of a three layer architecture like that which can be found in a smartcard is described below (figure 6).

In the case illustrated on figure 6, the comparison and action mechanism is implemented in the hardware at the level of memory access (MEM). By activating (ACTIV) the applications (APPLI), the operating system (OS) initialises the authorised space. The operating system (OS) comprises a call interception mechanism (11) which will implicitly notify to the management (9, 10) of the authorised space, the space targeted by the call made to the system (which may be different from the initialised space if one application has called another). The management (9,10) of the authorised space identifies the calling application and modifies depending on the rights of said calling application the authorised space by giving it the value of the targeted space. The management function of the device is hierarchic, the core being able to modify the authorised space by overload (SURCH). When the update has been made, spaces ATTRI-S and ATTRI-O are compared according to a given comparison rule, the type of operation and specific rules. The access required by the operation is accepted or rejected depending on the result of the comparison.

A special realisation consists in implementing the comparison and action mechanism in the hardware; in this case, one method consists in transferring the top of the stack into a hardware register, another method consists in implementing the whole stack in the hardware. In architectures which are simple or with limited access control, the device management can itself be implemented with specific hardware.

All or part of the system described above can be realised in the hardware.

In addition, the set of modules of said system may take any other form and in particular, modules of said set can be grouped together or split to form new modules.

The system according to the invention is centralised and unique or hierarchic with separate comparison, action and management mechanisms.

This invention therefore concerns a method to control the access to objects 5 stored in a memory space of a computer system by a subject 3 stored in a program space of said system, said subject wanting to perform an operation on said objects 5, characterised in that it controls the access to said objects 5 via a dynamic attribute 8 linked to said subject 3 whose value is updated according to the present and previous status(es) of the subject.

The object attribute is an item of information characterising the memory area targeted by the operation carried out by the subject and the subject attribute is an item of information characterising the authorised memory space associated with the present and past status of the current processing program, i.e. the subject.

The dynamic attribute is an item of information of type memory address(es) or an item of information of type identifier.

A module of the subject explicitly requires the update of the dynamic attribute 8 or implicitly via an underlying call interception mechanism.

CLAIMS

1) Method to control the access to objects (5) stored in a memory space of a computer system by a subject (3) stored in a program space of said system wanting to perform an operation on said objects (5), characterised in that it controls the access to said objects (5) via a dynamic attribute (8) linked to said subject (3) whose value is updated according to the present and previous status(es) of the subject.

2) Access control method according to claim 1, characterised in that it includes the following steps:

- a step to update the dynamic attribute (8) of the subject and manage said update;
- a step to compare an item of information characterising the targeted object (5) which is the attribute of the object and an item of information characterising the present and past status of the subject (3) which is said dynamic attribute of the subject;
- an action step to accept or reject the access depending on the result of said comparison and optionally on the type of operation and/or predefined and saved rules.

3) Method according to claim 1 or 2, characterised in that it controls the access to said object(s) according to the chaining of inter-module calls and the choice of each module to access its own object.

4) Method according to one of claims 1 to 3, characterised in that the dynamic attribute update mechanism is of type stack, the dynamic attribute used for access control being at the top of the stack.

5) Method according to one of claims 1 to 3, characterised in that the dynamic attribute update mechanism is of type list, the dynamic attribute used for access control being all of said list.

6) Method according to one of claims 1 to 5, characterised in that it consists in managing said update by checking the identity and the rights of the module part of the subject and requesting the update of the attribute (8) whether this update is implicit or explicit and in only making said update if the module is authorised to do so.

7) Device to control the access of a subject (3) stored in a program space of a computer system comprising at least memory means and computation means, and wanting to perform an operation on objects (5) stored in a memory space of said system, characterised in that it comprises means to perform an access control to said objects using a dynamic attribute (8) linked to said subject (3) and placed in memory whose value is updated according to the present and previous status(es) of the subject.

8) Device according to claim 7, characterised in that it comprises:

- a mechanism to update a dynamic attribute of the subject and manage the update depending on the present and previous status(es) of the subject;
- a mechanism to compare an item of information characterising the targeted object which is the attribute of the object and an item of information characterising the subject status which is the subject dynamic attribute (8), said items of information being stored in the memory means;
- an action mechanism to accept or reject the operation depending on the previous result, the type of operation and/or predefined rules.

9) Embedded system, characterised in that it comprises the device according to claim 7 or 8.

10) Computer program including program code instructions to execute the steps of the method according to one of claims 1 to 6 when said program is run in a data processing system.

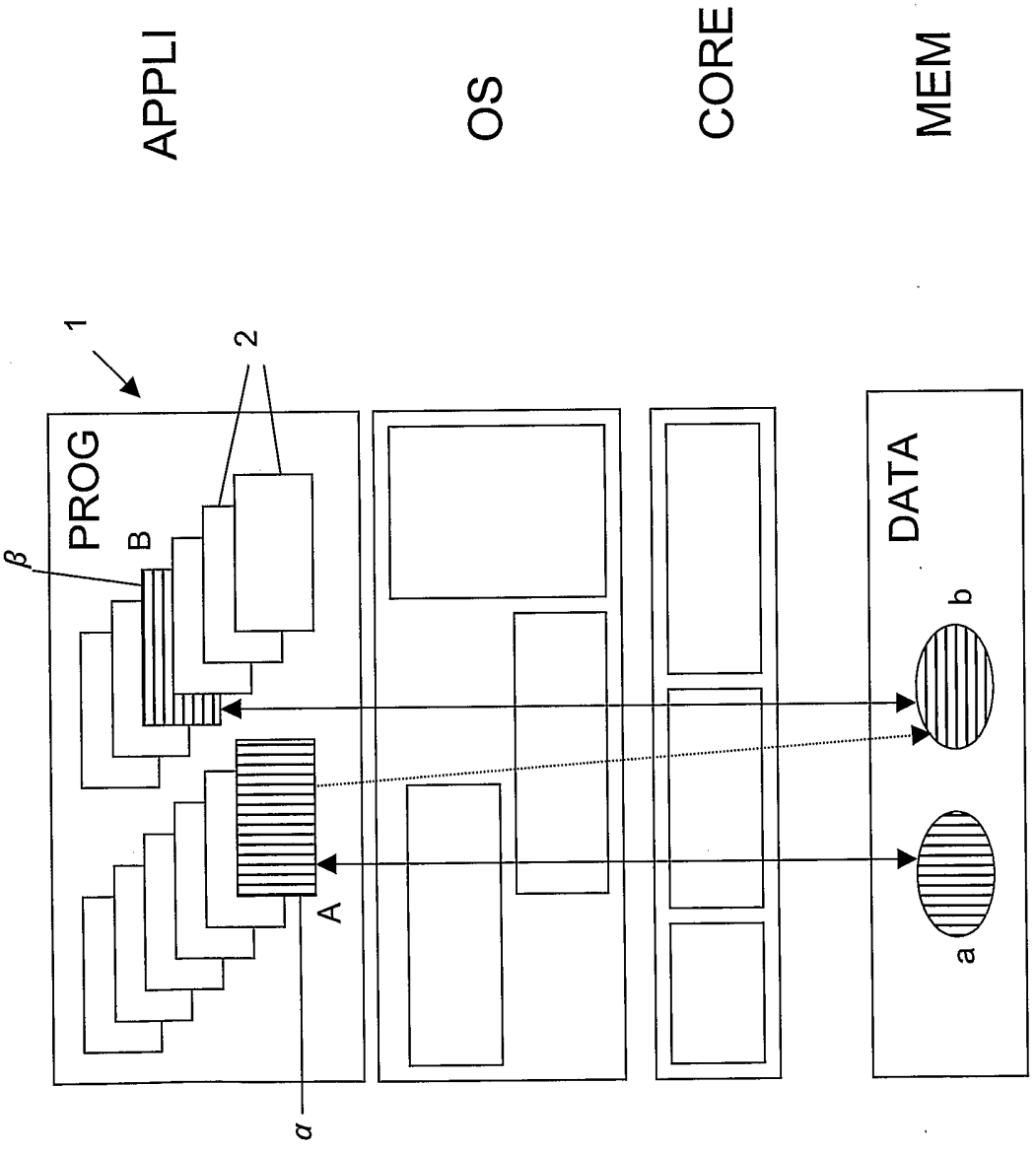


FIG. 1

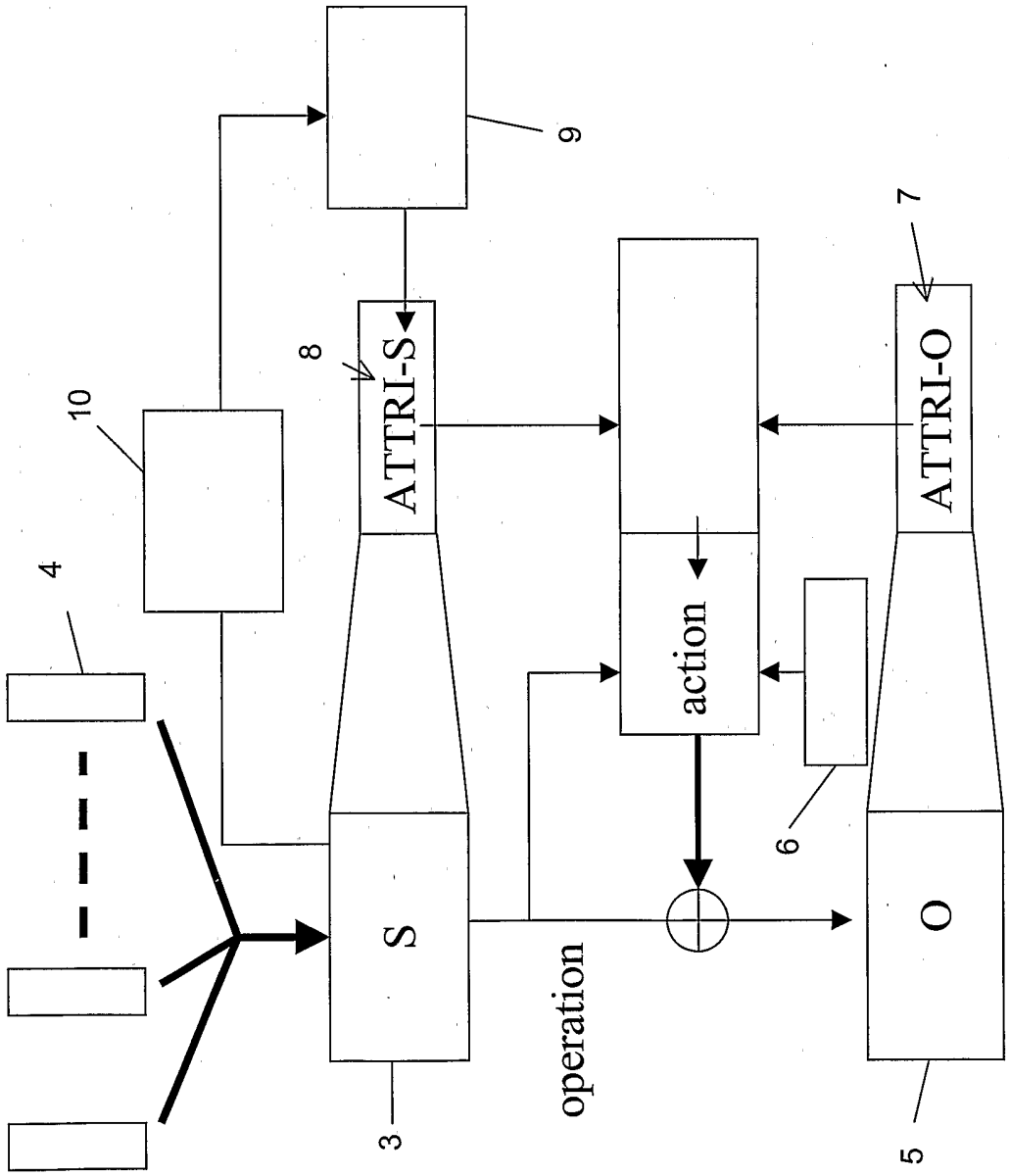


FIG. 2

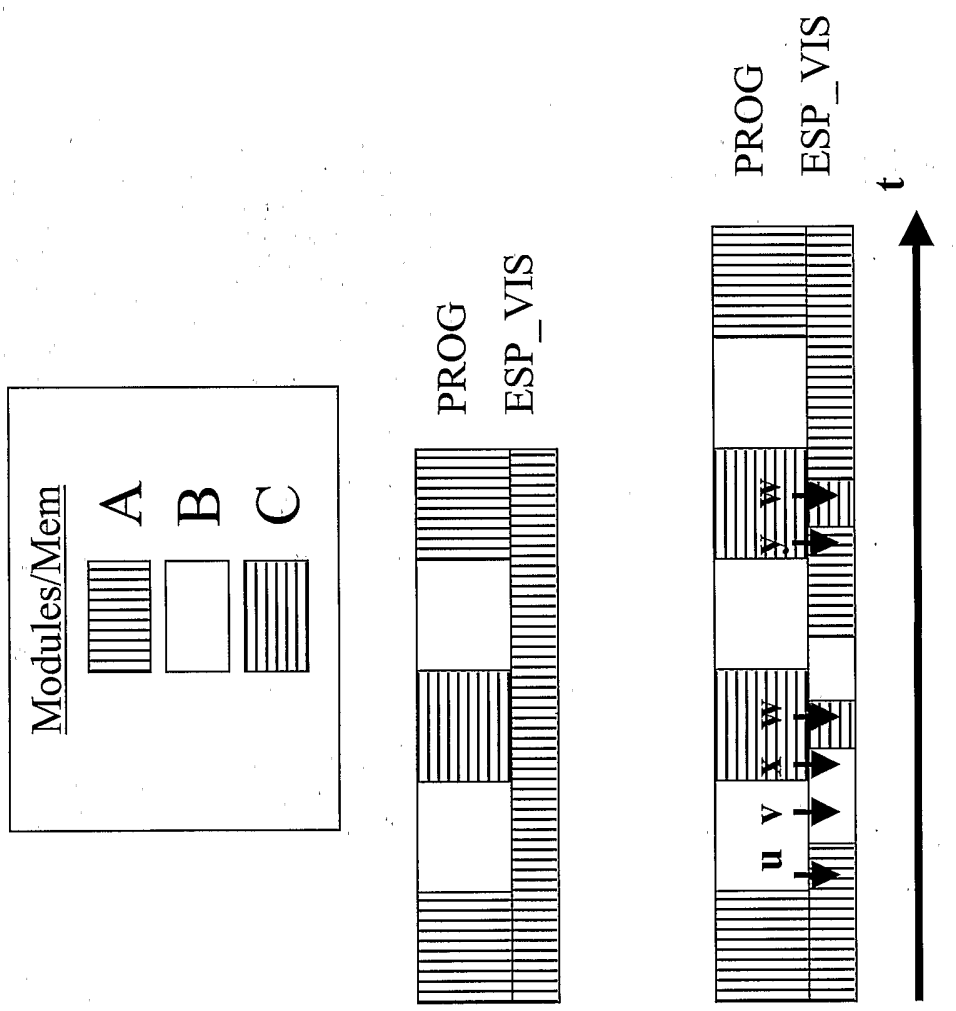


FIG. 3

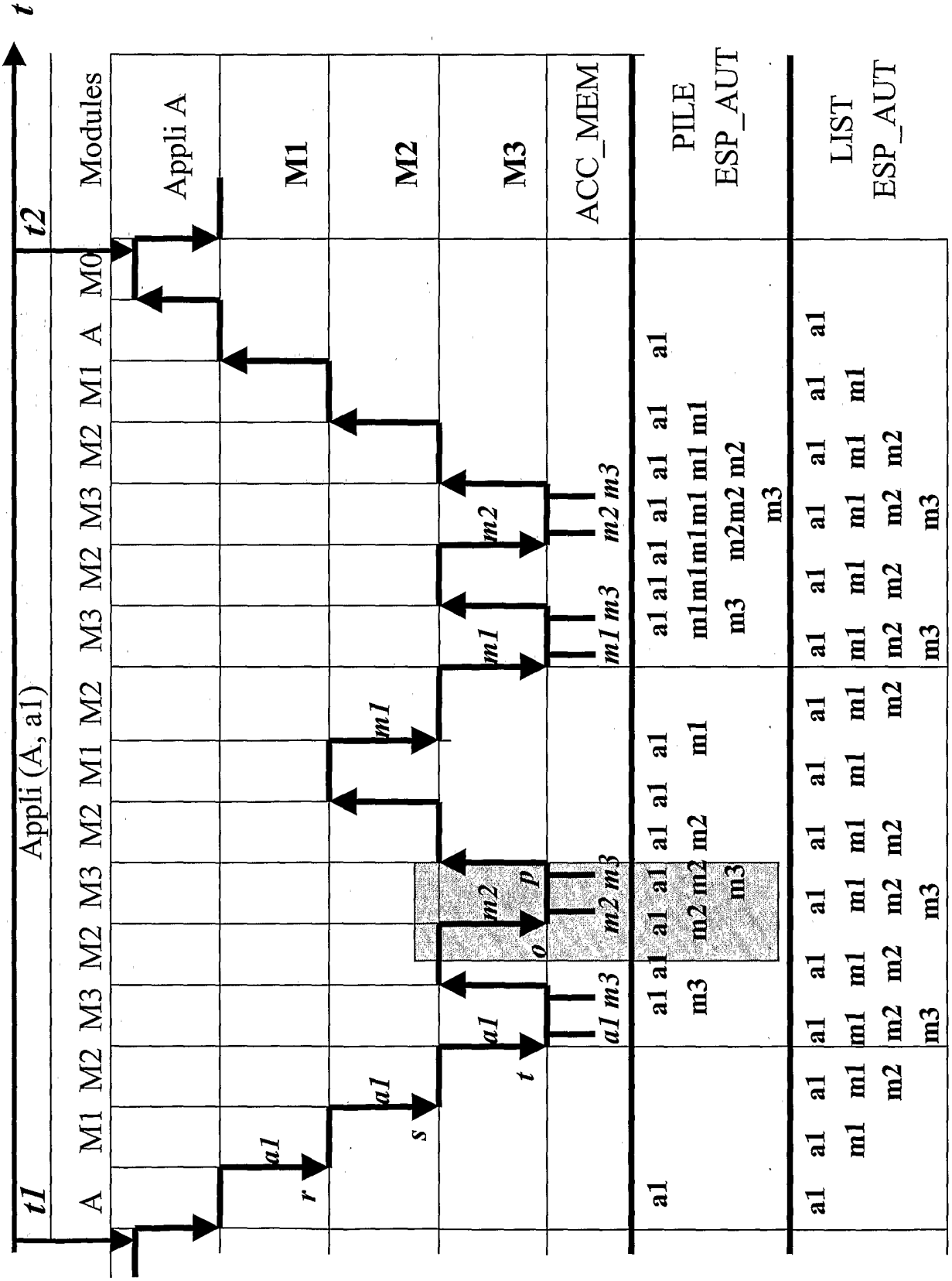

t
t1
t2
Appli (A, a1)
Modules
Appli A
M1
M2
M3
ACC_MEM
PILE
ESP_AUT
LIST
ESP_AUT
A M1 M2 M3 M2 M3 M2 M1 M2 M3 M2 M1 A M0
a1
r
s
t
o
p
m1
m2
m3

FIG. 4

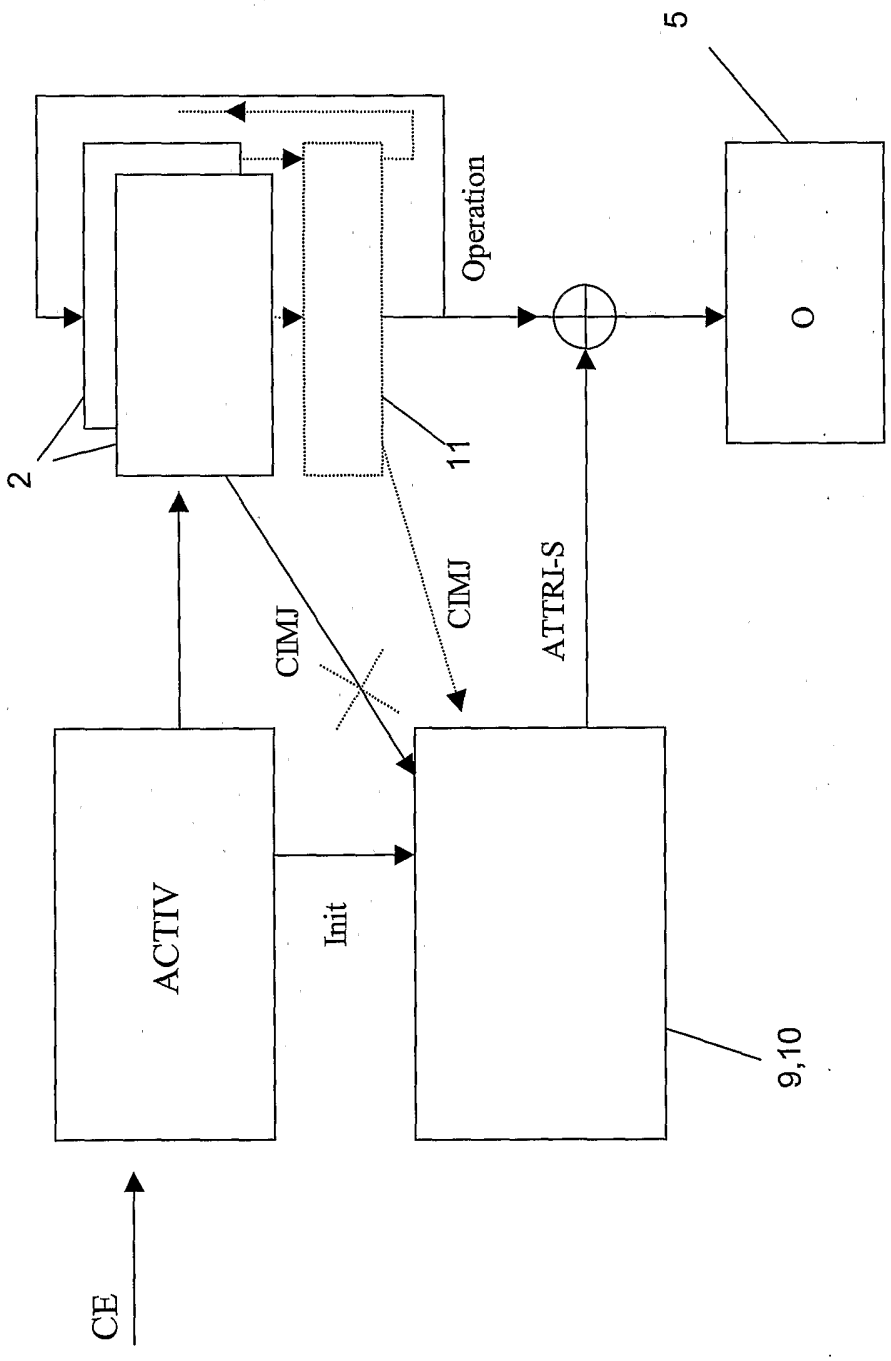


FIG. 5

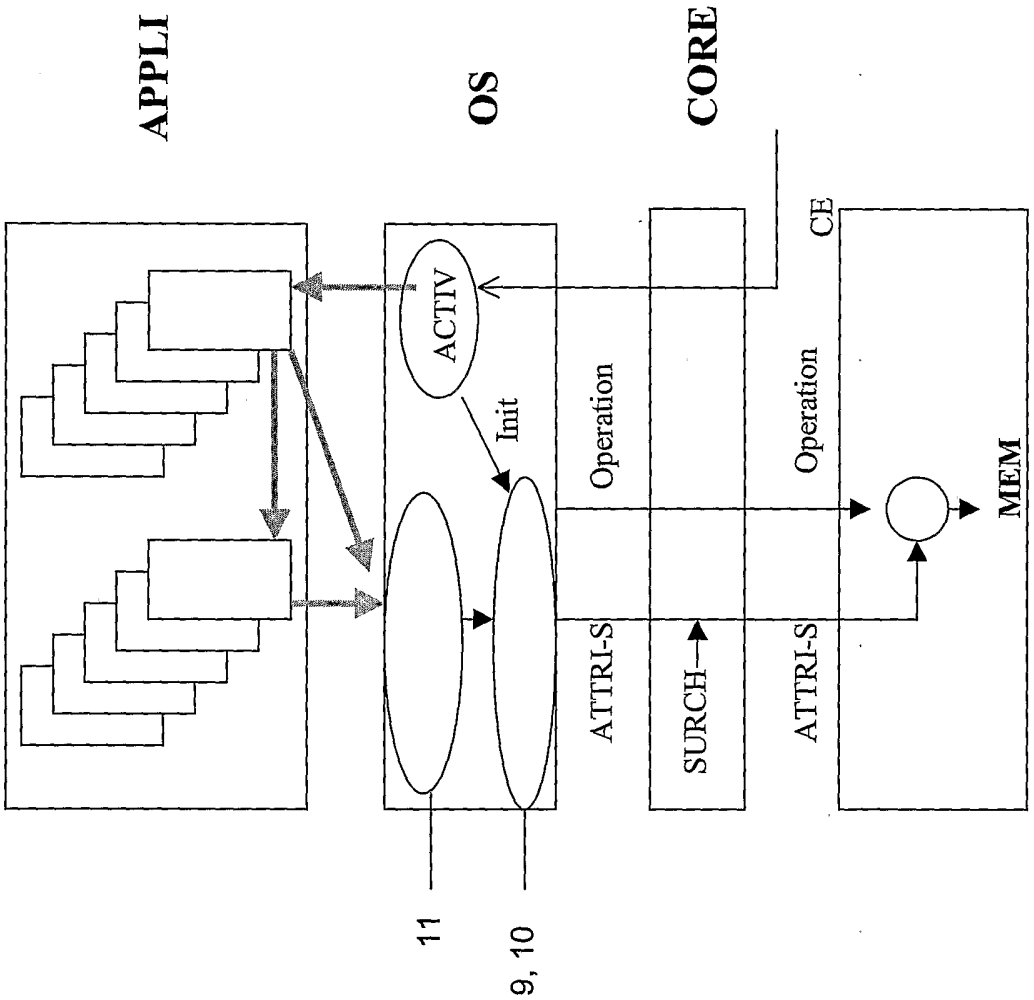

APPLI
OS
CORE
ACTIV
Init
Operation
Operation
CE
MEM
ATTRI-S
SURCH
ATTRI-S
11
9, 10

FIG. 6

INTERNATIONAL SEARCH REPORT

Intern application No
PCT/IB 02/05294

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	EP 1 168 184 A (ST MICROELECTRONICS SA) 2 January 2002 (2002-01-02) column 3, paragraph 11 -column 4, paragraph 22	1-10
A	US 5 452 431 A (BOURNAS JEAN-PIERRE) 19 September 1995 (1995-09-19) column 2, line 1 -column 5, line 3	1-10
A	US 5 912 453 A (DAO TRONG SON ET AL) 15 June 1999 (1999-06-15) column 3, line 17 -column 6, line 11	1-10

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 March 2003	10/03/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	NIELSEN, O

1

INTERNATIONAL SEARCH REPORT

Inter Application No
PCT/IB 02/05294

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1168184 A	02-01-2002	FR 2811096 A1	04-01-2002
		EP 1168184 A1	02-01-2002
		US 2002016890 A1	07-02-2002
US 5452431 A	19-09-1995	FR 2683357 A1	07-05-1993
		DE 69223920 D1	12-02-1998
		DE 69223920 T2	18-06-1998
		EP 0540095 A1	05-05-1993
		JP 5217035 A	27-08-1993
US 5912453 A	15-06-1999	DE 19536169 A1	03-04-1997
		EP 0766211 A2	02-04-1997
		JP 9223200 A	26-08-1997