RCMP COPY

TITLE OF THE INVENTION

FILE MANAGING METHOD REQUIRING A CHANGE IN KEY DATA AND IC CARD DEVICE USING THE METHOD

BACKGROUND OF THE INVENTION

The present invention relates to a method of managing a plurality of files in an IC card incorporating an IC chip having a nonvolatile memory of the files and a control device such as a CPU for controlling the nonvolatile memory.

An IC card incorporating an IC chip having a nonvolatile data memory and a control unit such as a CPU for controlling the data memory has recently been noteworthy as a portable data storage medium. In this IC card, the data memory is divided into a plurality of files each storing data necessary for operating an application, and an identification name of the application is input from an external device to render only the file corresponding thereto usable. Since a plurality of items of application data are assigned to their corresponding files and stored in the single IC card, the IC card can thus be used widely. The files include a plurality of data files for storing transaction files and the like and a key file for storing key data.

Furthermore, in the recent IC card, information called an access condition is added to each file, and it is determined whether command access can be gained

or not under condition that a directed key of the card is checked by the information.

During the reception of an IC card from its manufacturer to its user, various operations such as issuance of the card and storage of system basic information into the card, are performed, and the concerned parties vary from operation to operation. For example, one concerned with the issuance of the card is a card issuer (superordinate), while one concerned with the storage of system basic information is an application provider (subordinate). When authorization for access to the card is shifted from the card issuer to the application provider, a provisional key called "transport key" is generally set to the IC card and checked by the application provider, with the result that the application provider can gain access to the IC card afterward.

If the access condition to the file is set such that only the application provider can manage the data of the file, the card issuer cannot permanently set the transport key since there is no application provider key that the issuer must satisfy in order to set the transport key. To avoid this, there is a method for setting one of the application provider key and card issuer key as an access condition to the file. More specifically, the access condition has only to be set such that it is satisfied by collating the card issuer

key when the card issuer sets a transport key of the application provider in the file and that it is satisfied by collating the application provider key when the application provider manages the data of the file. However, the access condition set in the file can be accessed by both the card issuer and application provider and, in this case, the authorization to manage data of the file is not transferred from the card issuer (superordinate) to the application provider (subordinate).

Since the transport key is known to the card issuer, the application provider generally checks and rewrites the transport key in order to prevent any person other than the application provider from getting to know the transport key.

The above-described prior art however complicates the problems caused when data of the IC card is altered. The period of validity for utilizing application data of the IC card is generally changed by checking the key of the application provider. If, therefore, the application data is altered, the application provider is called to account therefor.

However, when the application provider uses the unchanged transport key, the card issuer has to perform the same operation as that of the application provider. In other words, though the alteration of data is due to the application provider's carelessness, the card

issuer is also suspected of the alternation of data.

It is thus impossible to completely execute "transfer of file management authorization to the IC card" which is the original object in utilizing the transport key.

To compulsorily rewrite the transport key and check it in the IC card is not realistic for application pursuing only the convenience of the IC card.

BRIEF SUMMARY OF THE INVENTION

It is accordingly an object of the present invention to provide a file management method which is capable of invalidating a transport key of a medium or the like unless the subordinate changes the transport key after the key is transferred from the superordinate to the subordinate and thus improved in security, and a medium using the file management method.

According to one aspect of the present invention, there is provided a method of managing files, comprises: the steps of setting a key data of a portable storage medium having a plurality of storage areas; determining whether the password is changed; inhibiting data from being processed in the plurality of storage areas when it is determined in the determining step that the key data is not changed; and processing data of the plurality of storage areas in response to a given command when it is determined in the determining

step that the key data is changed.

According to another aspect of the present invention, there is provided a removable storage medium having plural storage fields using the above method, comprises: means for storing a key data and data in a plurality of storage areas thereof; means for determining whether a key data stored in the storage means is changed; means for inhibiting data from being processed in the plurality of storage areas when the determining means determines that the key data is not changed; and means for processing data of the plurality of storage areas in response to a given command when the determining means determines that the key data is changed.

Unless an IC card issuer (superordinate) changes a key set by the issuer, an IC card application provider (subordinate) cannot perform an operation of writing an application into a storage area or the like. since, therefore, the application provider is sure to perform an operation after the key is changed, authorization to manage the files is completely shifted, resulting in improvement in security.

Additional objects and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instrumentalities and

combinations particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate presently preferred embodiments of the invention, and together with the general description given above and the detailed description of the preferred embodiments given below, serve to explain the principles of the invention.

FIG. 1A is a block diagram showing an example of a configuration of a card handling apparatus to which an IC card according to an embodiment of the present invention is applied;

FIG. 1B is a flowchart for explaining an operation of the IC card of the present invention;

FIG. 2 is a block diagram showing an example of a configuration of the IC card;

FIG. 3 is a memory map illustrating an example of a configuration of a data memory;

FIG. 4 is a view showing an example of a format of various types of definition information;

FIG. 5 is a view illustrating an example of a configuration of files set in the data memory;

FIG. 6 is a table showing an example of a configuration of a directory set in the data memory;

FIG. 7 is a flowchart for explaining an operation

of creating a data file;

FIG. 8 is a flowchart for explaining an operation of creating a key elementary file;

FIG. 9 is a flowchart for explaining an operation of setting key data;

FIG. 10 is a flowchart for explaining an operation of changing key data;

FIG. 11 is a flowchart for explaining an operation of collating a key;

FIG. 12 is a flowchart for explaining an operation of gaining access to a data elementary file;,

FIG. 13 is a flowchart for explaining an operation of creating a transport key; and

FIG. 14 to FIG. 16 are views illustrating operations of each data creations.

DETAILED DESCRIPTION OF THE INVENTION

An embodiment of the present invention will now be described with reference to the accompanying drawings.

FIG. 1A shows a configuration of a card handling apparatus to which an IC card is applied as a portable electronic apparatus and which is used as a terminal device of a financial system, a shopping system, etc. According to this configuration, an IC card 1 can be connected to a control unit 3 comprising a CPU, through a card reader/writer 2, and a keyboard 4, a CRT display device 5, a printer 6 and a floppy disk drive 7 are connected to the control unit 3.

FIG. 2 illustrates a configuration of the IC card 1. The IC card 1 includes a control device (e.g., CPU) 11, a nonvolatile data memory 12 capable of erasing data stored therein, a working memory 13, a program memory 14, and a contact portion 15 for getting electrical contact with the card reader/writer 2. Of these, the control device 11, data memory 12, working memory 13 and program memory 14, surrounded with a broken line in FIG. 2, is formed by one IC chip (or a plurality of IC chips) and as known in JIKKAIHEI2-17,381, the IC chip 10 and the contact portion 15 are formed as a IC module and buried into the body of the IC card.

The data memory 12 is constituted of, e.g., an EEPROM and stores various types of data. The working memory 13 is formed of, e.g., a mask ROM and temporarily holds data processed by the control device 11. The program memory 14 is constituted of, e.g., a mask ROM and stores programs of the control device 11.

Further, a program performing each routines of FIG. 7 to FIG. 13 mentioned later is stored in a program memory 14.

The data memory 12 is divided into a control area 120, a directory 121, a free space 122 and an area group 123, as illustrated in FIG. 3. The area group 123 has a plurality of data areas and a key area and can be grouped by the concept called data file (DF).

The master files (MF), which will be described later, are managed together as one form of the data files.

The data files are ones for managing a key area and a data area used for a corresponding application together. The data area is one for storing data which is read or written when necessary, such as transaction data. The key area is one for storing an identification number and the like. Data is written, rewritten and checked in the key area but cannot be read out therefrom. As described above, the data and key areas are included together in the data memory 12 as the area group 123. The respective physical locations of the files and areas are recognized by the control device 11 using the directory 121 in the data memory 12.

A file managing method according to the present invention will now be summarized with reference to the flowchart of FIG. 1B.

A medium such as an IC card to which the method is applied, is manufactured by an IC card issuer (S101), and the card issuer set a transport key in the IC card (S103). The IC card is handed to an application provider who creates an application and stores it in a file of the IC card.

The application provider mounts the IC card on an IC reader, and inputs and checks the transport key (S105), thus gaining access to the IC card. The transport key is changed to one exclusively for

an application supplier (S107). Thus, the IC card cannot be accessed by the former transport key, and the authorization to manage the files is completely shifted from the IC card issuer to the application provider. If it is confirmed that the transport key is changed (S109), the operations of creating data files (FIG. 7), creating key elementary files (FIG. 8), setting key data (FIG. 9), gaining access to elementary files (FIG. 12), etc. can be performed in accordance with a given command.

If the transport key is not changed, none of the above operations are carried out even when the original transport key set by the IC card issuer is input. If a trouble of rewriting the files or the like occurs after the transport key is changed, its responsibility rests with the application supplier who has changed the transport key. In this case, an operation of changing the transport key can be performed. Consequently, the authorization to manage the files of the IC card can be transferred completely between the IC card issuer (superordinate) and application provider (subordinate).

The control area 120 of the data memory 12 shown in FIG. 3 stores the leading address information of the area group 123 and that of the free space 122.

The directory 121 of the data memory 12 stores various types of definition information corresponding to the data files and areas, as illustrated in FIG. 4.

The definition information 201 shown in FIG. 4 defines the name of a data file and is constituted of data PTN for identifying data file name definition information in the directory 121, file serial number DFSN assigned to the present data file, file serial number PFSN of parent file of the present data file, file name DF_{name} assigned to the present data file, data NL representing the length of the file name, and data BCC for checking the validity of these data.

The definition information 202 shown in FIG. 4 defines the management information of a data file and is constituted of data PTN for identifying data file name definition information in the directory 121, file serial number DFSN assigned to the present data file, file serial number PFSN of parent file of the present data file, data file size DFS, AAID for identifying a data area in which additional information of the present data file is stored, TYPE for determining whether the additional information is output or not, UCF for inhibiting a type of the key, DFAC indicative of access condition of the data file, DFST for holding the states of the present data file, the number US of bytes used by data files subordinated to the present data file and areas, and data BCC for checking the validity of these data.

Here, a specific bit (for example 8th bit) of DFST is used as a transport bit meaning wherether the

transport key of the DF is changed.

In particular, when a data file is selected in a data file selection command, which will be described later, the AAID outputs the contents of the data areas shown on the selected file if necessary.

The definition information 203 shown in FIG. 4 defines an area for storing various types of transaction data and is constituted of data PTN for identifying area definition information in the directory 121, a serial number DFSN of a data file to which this area belongs, an identification number AID for gaining access to the area, ATOP representing the leading address of the area, ASIZ indicative of the size of the area, AAC representing access conditions of the area, AST for holding the state of the area, and data BCC for checking the validity of these data.

The definition information 204 shown in FIG. 4 defines an area for storing various types of key data and is constituted of data PTN for identifying key area definition information in the directory 121, a serial number DFSN of a data file to which this area belongs, an identification number kID for gaining access to the area, KTOP representing the leading address of the area, KSIZ indicative of the size of the area, CF indicative of the type of a key, KAC representing access conditions of the key, KST for holding the state of the key, and data BCC for checking the validity of

these data.

The above data PTN is made up of, e.g., one byte, and "00", "01", "02" and "03" are applied to the information 201, 202, 203 and 204, respectively.

FIG. 5 illustrates an example of a configuration of files. In this figure, DFnn indicates a data file, Dnn shows a data area, and Knn represents a key area.

As illustrated in FIG. 5, data files DF1 and DF2, key areas K00 and K01, and data areas D00 and D01 are subordinated to a master file MF. Data files DF1-1 and DF1-2, key areas K11 and K12, and data areas D11 and D12 are subordinated to the data file DF1. Key areas K111 and K112 and data areas D111 and D112 are subordinated to the data file DF1-1, while key areas K121 and K122 and data areas D121 and D122 are subordinated to the data file DF1-2.

Data files DF2-1 and DF2-2, key areas K21 and K22 and data areas D21 and D22 are subordinated to the data file DF2. Key areas K211 and K212 and data areas D211 and D212 are subordinated to the data file DF2-1, while key areas K221 and K222 and data areas D221 and D222 are subordinated to the data file DF2-2.

The above items of definition information are stored together in the directory 121, as shown in FIG. 6. A DFSN (data file serial number) is automatically assigned to each item of definition information when a file is created. The control

device 11 recognizes a relationship among the files in accordance with the DFSN and the serial number of the parent file (PFSN) in which the data file definition information is stored.

For example, in the definition information (serial number #13) of data file DF1-1, DFSN is "03" and PFSN is "01". More specifically, when this data file is created, a data file serial number (DFSN) "03" is assigned thereto and at the same time it is recognized that the data file is subordinated to DF1, and the DFSN "01" of the DF1 is assigned to the data file DF1-1 as a parent file serial number PFSN.

FIG. 7 is a flowchart for explaining an operation of creating a data file (DF). This operation will now be described.

When the IC card 1 receives a data file creation command from outside, it recognizes a data file in a usable state or in a current state (hereinafter referred to as current DF) (ST1). In particular, the current DF serves as a master file (MF) immediately after the IC card 1 is electrically activated.

If the current DF is recognized, then an access condition for creating a file is referred to from among access conditions of the current DF definition information. The access condition is compared with only collation state holding area A of a RAM (described later) to determine whether the collation state of

a key is established by request of the access condition (ST2).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST3), thus returning to a command wait state. On the other hand, if it is done, then the name (DF-ID) of a data file is extracted from a command to confirm whether there is data file definition information having parent FSN which is equal in value to FSN of the current DF and also having the same file name as the above extracted name (ST 4).

If the data file definition information is present, a response telegram representing ID duplication abnormality is output (ST5), and the flow returns to the command wait state. If it is not present, data file definition information, as shown in FIG. 4, is created by data file creating data supplied in response to a command (ST6) and then written to a predetermined area (ST7).

If the data file definition information is not normally written (ST8), a response telegram indicative of abnormal write is output (ST9), and the flow returns to the command wait state. If it is normally done (ST8), a response telegram indicative of normal end is output (ST10), the flow returns to the command wait state.

FIG. 8 is a flowchart for explaining an operation

of creating a key elementary file (EF). This operation will now be described.

When the IC card 1 receives a key EF creation command from outside, it recognizes the current DF (ST11).

If the current DF is recognized, then an access condition for creating a file is referred to from among access conditions of the current DF definition information. The access condition is compared with only collation state holding area A of a RAM (described later) to determine whether the collation state of a key is established by request of the access condition (ST12).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST13), thus returning to a command wait state. On the other hand, if it is done, then the name (EF-ID) of an elementary file is referred to from a command telegram to check whether the elementary file name is present within the current DF to be accessed (ST14). If the elementary file name is present, a response telegram representing ID duplication abnormality is output (ST15), and the flow returns to the command wait state.

If it is not present, the size of the key EF designated by the command telegram is compared with that of a free space within the current DF to be

accessed (ST16). In this comparison, it is checked whether the size of the free space is larger than the sum of the designated size of the key EF and the size of directory information used for creating the key EF. When the sum is larger, a response telegram indicating that the designated size is abnormal is output (ST17), and the flow returns to the command wait state.

If the sum is not larger, then the correctness of the size is checked by the type of a key designated by the command telegram (ST18). When the key is a authentication related one and the size is for example 10 bytes or when the key is a collating one and the size is for example 3 to 18 bytes, it is determined that the size is correct. If it is not correct, a response telegram representing that the collation of the designated size is abnormal is output (ST19), and the flow returns to the command wait state.

When the size is correct, key EF definition information to be stored in the directory is created in response to a received command (ST20) and written to a predetermined area (ST21). The eighth bit of status information depends upon the first bit of key type information designated by the command telegram and, in other words, the value of the first bit is set to the eight bit of the status information.

The eight bit of the status information represents whether key data is changed or not. The key data is

not changed when the bit is "1" and it is done when the bit is "0." Consequently, when the first bit of the key type information is "1," the eight bit of the status information is not "0" unless the key data is changed. When the first bit is "0," the eight bit is "0" regardless of whether the key data is changed or not, which means that data is tacitly rewritten.

If the key EF definition information is not normally written to the predetermined area (ST22), a response telegram indicative of abnormal write is output (ST23), thus returning to the command wait state. When the information is normally written (ST22), a response telegram indicative of normal end is output (ST24), and the flow returns to the command wait state.

FIG. 9 is a flowchart for explaining an operation of setting key data. This operation will now be described.

When the IC card 1 receives a key data setting command from outside, it recognizes the current DF (ST31).

If the current DF is recognized, the name (EF-ID) of an elementary file is referred to from a command telegram to check whether the elementary file name is present within the current DF to be accessed (ST32).

If the elementary file name is not present, a response telegram indicative of no corresponding key ID

is output (ST33), and the flow returns to the command wait state.

If it is present, then an access condition for setting key data is referred to from among access conditions of the key EF definition information. This access condition is compared with only collation state holding area A of the RAM (described later) to determine whether the collating state of the key is established by request of the access condition (ST34).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST35), thus returning to the command wait state. On the other hand, if it is done, then it is confirmed whether the key data is present within its corresponding key EF area (ST36).

If the key data is present, a response telegram indicating that the key data has been present is output (step ST37), and the flow returns to the command wait state. If it is not present, then the correctness of the size of input key data is checked by the type of a key designated by the command telegram (ST38). When the key is a authentication related one and the size is for example 8 byte and when the key is a collating one and the size is for example 1 to 16 bytes, it is determined that the size is correct. If it is not correct, a response telegram representing that the size of the input key data is abnormal is output (ST39), and

the flow returns to the command wait state.

When the size is correct, the size defined in the key EF definition information and the size of the input key data are compared with each other (ST40). If the sum of the latter size and "2" is larger than the former size, a response telegram indicative of insufficient area size is output (ST41), and the flow returns to the command wait state. If the sum is not larger, one-byte information and one-byte BCC are added to the key data input in response to the received command, and stored in the key EF area (ST42). The result is output as a response telegram (ST43), and the flow returns to the command wait state.

FIG. 10 is a flowchart for explaining an operation of changing key data. This operation will now be described.

When the IC card 1 receives a key data changing command from outside, it recognizes the current DF (ST51).

If the current DF is recognized, the name (EF-ID) of an elementary file is referred to from a command telegram to check whether the elementary file name is present within the current DF to be accessed (ST52).

If the elementary file name is not present, a response telegram indicative of no corresponding key ID is output (ST53), and the flow returns to the command wait state.

If it is present, then an access condition for changing key data is referred to from among access conditions of the key EF definition information. This access condition is compared with collation state holding areas A and B of the RAM (described later) to determine whether the collating state of the key is established by request of the access condition (ST54).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST55), thus returning to the command wait state. On the other hand, if it is done, then it is confirmed whether the key data is present within its corresponding key EF area (ST56).

If the key data is not present, a response telegram indicating that no key data is present is output (ST57), and the flow returns to the command wait state. If it is present, then the correctness of the size of input key data is checked by the type of a key designated by the command telegram (ST58). When the key is a authentication related one and the size is for example 8 byte and when the key is a collating one and the size is for example 1 to 16 bytes, it is determined that the size is correct. If it is not correct, a response telegram representing that the size of the input key data is abnormal is output (ST59), and the flow returns to the command wait state.

When the size is correct, the size defined in

the key EF definition information and the size of the input key data are compared with each other (ST60). If the sum of the latter size and "2" is larger than the former size, a response telegram indicative of insufficient area size is output (ST61), and the flow returns to the command wait state. If the sum is not larger, one-byte information and one-byte BCC are added to the key data input in response to the received command, and stored in the key EF area (ST62). The result is output as a response telegram (ST63), and the flow returns to the command wait state. The eighth bit of status information of the key EF definition information is set to "0" (ST64).

FIG. 11 is a flowchart for explaining an operation of collating a key. This operation will now be described.

When the IC card 1 receives a key data changing command from outside, it recognizes the current DF (ST71).

If the current DF is recognized, the directory 121 is searched to check whether key EF definition information having a designated file name (ID) is present within the current DF (ST72).

If the key EF definition information is not present, a response telegram indicative of no corresponding key ID is output (ST73), and the flow returns to the command wait state. If it is present, it is

confirmed whether the key is locked or not (ST74). When it is locked, a response telegram representing a locked key is output, then returning to the command wait state (ST75). If it is not present, the key data of the command telegram is collated with that stored in the key EF (ST76). When both coincide with each other (ST77), collation bit designation information is referred to from the key EF definition information to set the bit position, designated by the information, to "1" in area B before the bit is changed and in area A after the bit is changed (ST78). A collation inconsistence counter proper to the key of the key EF definition information is cleared (ST79), and a response telegram indicative of normal end is output (ST80). The flow then returns to the command wait state.

The RAM is divided into collation state holding areas A and B. Whether bit "1" is set to area A or B relies upon the value of the eighth bit of status information of the key EF definition information. This bit represents whether the key defined by the key EF definition information is changed or not. As described later, "0" indicates a changed key, while "1" does an unchanged key. When the bit is "0," it is set to the area A. When the bit is "1," it is set to the area B.

If the keys do not coincide with each other (ST77), the collation bit designating information and

status information are referred to from the key EF definition information to set the bit of one of the areas A and B to "0" by the same procedure as above (ST81).

A collation inconsistence counter proper to the key of the key EF definition information is incremented only one (ST82). When the counted value does not reach the maximum (ST83), a response telegram indicative of collation inconsistence is output (ST84), thus returning to the command wait state. When the counted value reaches the maximum, a response telegram indicating that the key has been locked is output (ST85), thus returning to the command wait state.

The collation state holding area A is referred to in creating files, key EF and key data and changing key data, while the collation sate holding area B is referred to in changing key data.

As described above, if the area A is unchanged when the key is already changed, the bit of the area B is turned on. Therefore, the key has to be changed in creating the files, key EF and key data. Since, furthermore, the areas A and B are referred to when the key is changed, the key can be changed regardless of whether the key has been changed or not.

FIG. 12 is a flowchart for explaining an operation of gaining access to data EF. This operation will now be described.

When the IC card 1 receives a data EF access command from outside, it recognizes the current DF (ST91).

If the current DF is recognized, the name (EF-ID) of an elementary file is referred to from a command telegram to check whether the elementary file name is present within the current DF to be accessed (ST92).

If the elementary file name is not present, a response telegram indicative of no corresponding key ID is output (ST93), and the flow returns to the command wait state.

If it is present, then an access condition is referred to from among access conditions of the key EF definition information in accordance with an access type (read/write/change). This access condition is compared with a collation state holding area A of the RAM to determine whether the collation state of a key is established by request of the access condition (ST94).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST95), thus returning to the command wait state. On the other hand, if it is done, then a corresponding data EF area is accessed (ST96), and its results are output as a response telegram (ST97), thus returning to the command wait state.

As an access condition for setting a key

(transport key) of an application provider, a key of its superordinate, i.e., an IC card issuer is required. This transport key cannot be changed unless its collation is performed, and it is necessary to collate the transport key when the other access commands are executed. In this case, the collation result of the transport key, which has not been changed, is stored in the collation state holding area B. Therefore, even though the other accesses necessitate the transport key, the collation result is not reflected in the collation state holding area A used for these accesses and, in this case, the transport key has no effect.

Since both the areas A and B are referred to in order to change the transport key, the established collation state can be checked and thus a change in the transport key is permitted.

Since, furthermore, the collation state is reflected in the collation state holding area A by collating the changed key, the effect of the key is produced.

In the above embodiment of the present invention, it is possible to determine whether a pseudo change in the key is caused according to the type of the key input when the key EF is created. Thus, the superordinate card issuer selectively sets various types of key by respective requests of the application provider.

In the above embodiment, it is during the key EF

creation command processing when status information representing whether the key is previously changed is set by the key type information. However, the status information can be set during the command processing.

FIG. 13 is a flowchart for explaining an operation of creating a transport key elementary file (EF). This operation will now be described.

When the IC card 1 receives a transport key EF creation command from outside, it recognizes the current DF (ST101).

If the current DF is recognized, then it is checked whether the transport bit of status information DFST set in the DF is turned on or not (ST102).

If the transport bit is turned on, the command is rejected, and a response telegram indicative of impossible transport is output, thus returning to a command wait state (ST103). If the transport bit is turned off, the command is allowed. The transport bit is turned off when the data file is created, and it is turned on when the transport key is changed in response to a key changing command.

If the transport key EF creation command is allowed, then a access condition to the parent data file (DF) of the current DF is referred to. This condition is compared with the collation state holding area A of the RAM to determine whether the collation state of the key is established or not by request of

the access condition (ST104).

If it is not established, a response telegram indicative of inconsistence of the access conditions is output (ST105), thus returning to a command wait state. On the other hand, if it is done, then the name (EF-ID) of an elementary file is referred to from a command telegram to check whether the elementary file name is present within the current DF to be accessed (ST106). If the elementary file name is present, a response telegram representing ID duplication abnormality is output (ST107), and the flow returns to the command wait state.

If it is not present, the size of the key EF designated by the command telegram is compared with that of a free space within the current DF to be accessed (ST108). In this comparison, it is checked whether the size of the free space is larger than the sum of the designated size of the key EF and the size of directory information used for creating the key EF. When the sum is larger, a response telegram indicating that the designated size is abnormal is output (ST109), and the flow returns to the command wait state.

If the sum is not larger, then the correctness of the size is checked by the type of a key designated by the command telegram (ST110). When the key is a authentication related one and the size is for example 10 bytes or when the key is a collating one and the

size is for example 3 to 18 bytes, it is determined that the size is correct. If it is not correct, a response telegram representing that the collation of the designated size is abnormal is output (ST111), and the flow returns to the command wait state.

When the size is correct, key EF definition information to be stored in the directory is created in response to a received command (ST112) and written to a predetermined area (ST113). The eighth bit of status information depends upon the first bit of key type information designated by the command telegram and, in other words, the value of the first bit is set to the eight bit of the status information.

The eight bit of the status information represents whether key data is changed or not. The key data is not changed when the bit is "1" and it is done when the bit is "0." Consequently, when the first bit of the key type information is "1," the eight bit of the status information is not "0" unless the key data is changed. When the first bit is "0," the eight bit is "0" regardless of whether the key data is changed or not, which means that data is tacitly rewritten.

If the key EF definition information is not normally written to the predetermined area (ST114), a response telegram indicative of abnormal write is output (ST115), thus returning to the command wait state. When the information is normally written

(ST114), a response telegram indicative of normal end is output (ST116), and the flow returns to the command wait state.

A procedure for creating a file for an IC card having the above constitution will now be described.

First, in order to set (create) an elementary file of a transport key, which is to be handed to a card issuer, when the IC card is manufacture, key EF-a, which is subordinated to a master file MF, as shown in FIG. 14, is created and a transport key is set therein according to the flowchart of FIG. 13.

A key is set in the key elementary file EF-a in accordance with the flowchart of FIG. 9. The key is set, referring to a key setting access condition given to the key EF-a.

When the card issuer receives the card in this state, he or she changes the transport key in accordance with the flowchart of FIG. 10. In other words, as shown in FIG. 15, the card issuer changes the transport key set by the manufacturer to a key a' which is known to only the card issuer. To do this, the issuer refers to a key changing access condition set in the key EF. Then, the transport bit given to the master file MF is turned on and, after that, a key subordinated to the master file MF is rejected by the manufacturer.

The card issuer creates data EF-a under the master file MF for himself or herself and then refers to an EF

creating access condition set in the master file MF. The card issuer also creates a data file DF, which is to be opened to the application provider, under the master file MF. The issuer sets the data file DF in accordance with the flowchart of FIG. 7 while referring to an access condition for creating the data file DF set in the master file MF.

The card issuer creates a transport key EF-b, which is to be handed to the application provider, under the data file DF. The transport key EF-b is created in accordance with the flowchart of FIG. 13, referring to not an access condition assigned to the data file DF but that assigned to the master file MF which is a parent file of the data file DF.

The card issuer sets a transport key in the transport key EF-b. This transport key is set in accordance with the flowchart of FIG. 9, referring to an access condition for setting the key, which is assigned to the key EF-b.

When the application provider receives the card in this state, he or she changes the transport key set by the card issuer to a key b' which is known to only the application provider as shown in FIG. 16. To do this, the application provider refers to a key changing access condition set in the key EF. Then, the transport bit given to the data file DF is turned on and, after that, a key subordinated to the data file DF is

rejected by the card issuer.

The application provider creates data EF-b under the data file DF for himself or herself. The data EF-b is created in accordance with the flowchart of FIG. 7, referring to an EF creating access condition set in the data file DF. The application provider also creates a key EF-c under the data file DF for himself or herself. The key EF-c is created in accordance with the flowchart of FIG. 8, referring to an EF creating access condition set in the data file DF.

The following are commands for the IC card of the present invention. Of the commands, a normal key EF creating command and a transport key EF creating command differs from each other in command code. The IC card identifies the contents of the commands on the basis of the command code. More specifically, data is processed in accordance with the flowchart of FIG. 13 when the transport key EF creating command is identified, while data is processed in accordance with the flowchart of FIG. 8 when the normal key EF creating command is identified.

Area (Elementary File) Creating Command

Command Code A/AID(EF-ID)/ASIZ/AAC

Key EF Creating Command

Command Code B/KID/KSIZ/CF/AAC

Transport Key EF Creating Command

Command Code C/KID/KSIZ/CF/AAC

When the transport key EF is created under the current DF as described above, the access condition of the parent DF of the current DF is referred to. The access condition of the parent DF is set by the superordinate, and the transport key EF can be created by the superordinate.

When a transport key EF is set to the transport key EF, the access condition of the transport key EF is referred to. This access condition is set by the superordinate when the transport key EF is created and determined such that the transport key can be set to the transport key EF.

When the application provider sets the key EF under the current DF, he or she refers to the access condition for creating the key EF of the current DF.

However, after the transport key is changed, only the application provider can execute the processing (creation of data EF and key EF) using the access condition of the current DF. Therefore, the access right can be clearly divided.

If the superordinate sets a key as a key EF creating access condition after the transport key is changed, the application provider sets a key EF used by himself or herself after the transport key is changed and prepares an environment in which the access condition of the current DF can be satisfied by collation bit designating information of definition information

of the key EF. Collating the key of the application provider, the access condition of the current DF is satisfied and the data EF can be created. By doing so, the superordinate sets the access condition of the current DF, but only the application provider satisfies the access condition of the current DF without being concerned with the contents of the access condition.

As described above, according to the embodiment of the present invention, the key required for creating the EF (or DF) under the data file DF, is determined by the access condition given to the data file DF, and the access condition can be achieved without designating the superordinate key (in this embodiment the card issuer is a superordinate of the application provider). Thus, the file (memory area) that the application provider wishes to manage can be protected by only the key of the application provider. Consequently, the card issuer is not concerned with the data file (DF) and the authorization to gain access to the file (memory area) is transferred from the card issuer to the application provider.

According to the embodiment of the present invention, the transport bit is turned off when the data file DF is created and it is turned on when the transport key itself is changed by the key changing command. However, for example, the transport bit can be turned on by a different command without being

associated with the key changing command. In this case, the access condition used when the command is executed is set in a target file.

As described above, according to the embodiment of the present invention, there can be provided a method of managing access to a memory wherein the superordinate is not concerned with the data file and the authorization to manage access to the data file or memory area can be transferred from the superordinate to the subordinate.

In the above embodiment, the IC card is used as an electronic apparatus for managing files. The present invention is not limited to this, but can be applied to an electronic apparatus having a memory requiring file management.

As has been described above in detail, according to the present invention, there is provided a file management method which is capable of invalidating a transport key of a medium or the like unless the subordinate changes the transport key after the key is transferred from the superordinate to the subordinate and thus improved in security, and a medium using the file management method.

Additional advantages and modifications will readily occur to those skilled in the art. Therefore, the invention in its broader aspects is not limited to the specific details and representative embodiments,

shown and described herein. Accordingly, various modifications may be made without departing from the spirit or scope of the general inventive concept as defined by the appended claims and their equivalents.

CLAIMS

1. A method of managing files, comprising the steps of:

setting a key data of a portable storage medium having a plurality of storage areas;

determining whether the key data is changed;

inhibiting data from being processed in the plurality of storage areas when it is determined in said determining step that the key data is not changed; and

allowing data processing of the plurality of storage areas in response to a given command when it is determined in said determining step that the key data is changed.

2. A method of managing files according to claim 1, further comprising a step of changing the key data to a new key data in response to a given command when it is determined in said determining step that the key data is not changed.

3. A method of managing files according to claim 1, further comprising a step of selecting a normal operation mode in which data of the plurality of storage areas is processed in response to a given command, regardless of results of said determining step.

4. A method of managing files according to claim 1, further comprising the steps of:

* C L M S *

setting attribute information indicative of the normal operation mode to the key data; and

processing data of the plurality of storage areas in response to a given command, irrespective of results of said determining step, when the key data includes the attribute information.

5. A method of managing files according to claim 1, wherein said allowing step includes a step of creating a data file in each of the plurality of storage areas in response to a given command.

6. A method of managing files according to claim 1, wherein said allowing step includes a step of creating a file of the key data in each of the plurality of storage areas in response to a given command and the file creating is inhibited by the inhibiting means by the inhabiting means before the key data is changed.

7. A method of managing files according to claim 6, wherein said allowing step includes a step of setting data in the file of the key data of each of the plurality of storage areas in response to a given command and the data setting is inhibited by the inhabiting means before the key data is changed.

8. A removable storage medium having plural storage fields, comprising:

means for storing a key data and data in a plurality of storage areas thereof;

means for determining whether an initial key data is changed by comparing the initial key data is compared with the key data stored in said storing means;

means for inhibiting data from being processed in the plurality of storage areas when said determining means determines that the key data is not changed; and

means for processing data of the plurality of storage areas in response to a given command when said determining means determines that the key data is changed.

9. A removable storage medium according to claim 8, further comprising means for changing the key data to a new key data in response to a given command when said determinating means determines that the key data is not changed.

10. A removable storage medium according to claim 8, further comprising means for selecting a normal operation mode in which data of the plurality of storage areas is processed in response to a given command, regardless of results of said determinating means.

11. A removable storage medium according to claim 8, further comprising:

means for storing the key data having attribute information indicative of the normal operation mode; and

means for processing data of the plurality of storage areas in response to a given command, irrespective of results of said determinating means, when the key data includes the attribute information.

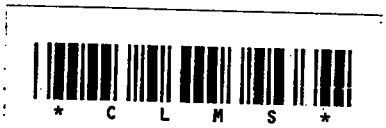
12. A removable storage medium according to claim 8, wherein said processing means includes means for creating a data file in each of the plurality of storage areas in response to a given command and the file creating is inhibited by the inhibiting means before the key data is changed.

13. A removable storage medium according to claim 8, wherein said processing means includes means for creating a file of the key data in each of the plurality of storage areas in response to a given command and the file creating is inhibited by the inhibiting means before the key data is changed.

14. A removable storage medium according to claim 13, wherein said processing means includes means for setting data in the file of the key data of each of the plurality of storage areas in response to a given command and the data setting is inhibited by the inhibiting means before the key data is changed.

15. A removable storage medium according to claim 8, wherein said processing means includes means of processing data in the file of each of the plurality of storage areas in response to a given command.

16. A method of managing access to a plurality of



files of a memory, which allows access to the files by collating a first key, which is set in advance under the files by a superordinate of a system using the memory, with a new key, which is set by a subordinate of the system using the memory,

wherein the superordinate changes the first key to a second key, which is known to only the subordinate, referring to a key changing access condition set in the first key and, when the first key is changed to the second key, the superordinate rejects creation of a key under the files.

17. A method of managing access to a tree-structure file including a superordinate file and a subordinate file, comprising the steps of:

setting an access condition in the superordinate file managed by a superordinate manager;

setting an access condition in a file provided from the superordinate manager to a subordinate manager;

setting a transport key, which is to be transferred from the superordinate manager to the subordinate manager, in the subordinate file, referring to an access condition of the superordinate file;

setting the transport key in the subordinate file when the access condition of the superordinate file is satisfied; and

referring to an access condition of the

* C L M S *

subordinate file and, when the access condition is satisfied, setting a file under the subordinate file.

18. The method according to claim 17, wherein the subordinate file has identification information indicating whether the transport key is allowed to be set, and the transport key is inhibited from being set in the subordinate file based on the identification information.

19. The method according to claim 18, wherein the identification information is updated so as to inhibit the transport key from being set in the subordinate file after the transport key is changed.

20. The method according to claim 18, wherein the identification information is updated in response to a special command.

21. The method according to claim 17, wherein the transport key is set in the subordinate file in response to a transport key setting command, and the file is set under the subordinate file in response to a command other than the transport key setting command.

22. A portable information processing apparatus including a tree-structure file having a superordinate file and a subordinate file, comprising:

a superordinate file in which an access condition is set and which is used by a superordinate manager;

a subordinate file in which an access condition is set and which is provided by the superordinate manager

* C L M S *

and used by a subordinate manager;

first means for referring to the access condition of the superordinate file and setting a transport key, which is to be transferred from the superordinate manager to the subordinate manager, in the subordinate file when the access condition of the superordinate file is satisfied; and

second means for referring to the access condition of the subordinate file and setting a file under the subordinate file when the access condition of the subordinate condition is satisfied.

23. The apparatus according to claim 22, wherein the subordinate file includes a storage area for identification information indicating whether the transport key is allowed to be set and means for inhibiting the transport key from being set in the subordinate file based on the identification information.

24. The apparatus according to claim 22, further comprising means for updating the identification information so as to inhibit the transport key from being set in the subordinate file after the transport key is changed.

25. The apparatus according to claim 22, wherein the apparatus includes means for updating the identification information in response to a special command supplied from the external device by an operation



corresponding to a command supplied from an external device.

26. The apparatus according to claim 22, wherein the first means sets the transport key in the subordinate file in response to a transport key setting command by on operation corresponding to a command supplied from an external device, and the second means sets the file under the subordinate file in response to a command other than the transport key setting command.

27. A storage medium for creating a tree-structure file having a superordinate file and a subordinate file and storing control programs for an information processing apparatus for managing access to each of the files by control means, the control programs include:

a step of creating the superordinate file in which an access condition is set by a superordinate manager;

a step of creating the subordinate file in which an access condition is set such that the subordinate file is provided by the superordinate manager and used by a subordinate manager;

a step of referring to the access condition of the superordinate file and setting a transport key, which is to be transferred from the superordinate manager to the subordinate manager, in the subordinate file when the access condition of the superordinate file is satisfied; and

a step of referring to the access condition of

the subordinate file and setting a file under the subordinate file when the access condition of the subordinate condition is satisfied.

28. The storage medium according to claim 27, wherein a program memory in which the control programs are stored, a data memory to which the files are assigned, a control circuit for managing access to the files, and an interface with an external device are formed in a single module.

29. A file managing method requiring a change in key data and IC card device using the method, substantially as hereinbefore described with reference to the accompanying drawings.



FILE MANAGING METHOD REQUIRING A CHANGE IN KEY DATA AND IC CARD DEVICE USING THE METHOD

ABSTRACT OF THE DISCLOSURE

A file managing method includes a step of setting a key data of a portable storage medium having a plurality of storage areas, a step of determining whether the key data is changed, a step of inhibiting data from being processed in the plurality of storage areas when it is determined that the key data is not changed, and a step of processing data of the plurality of storage areas in response to a given command when it is determined that the key data is changed.

Fig. 2

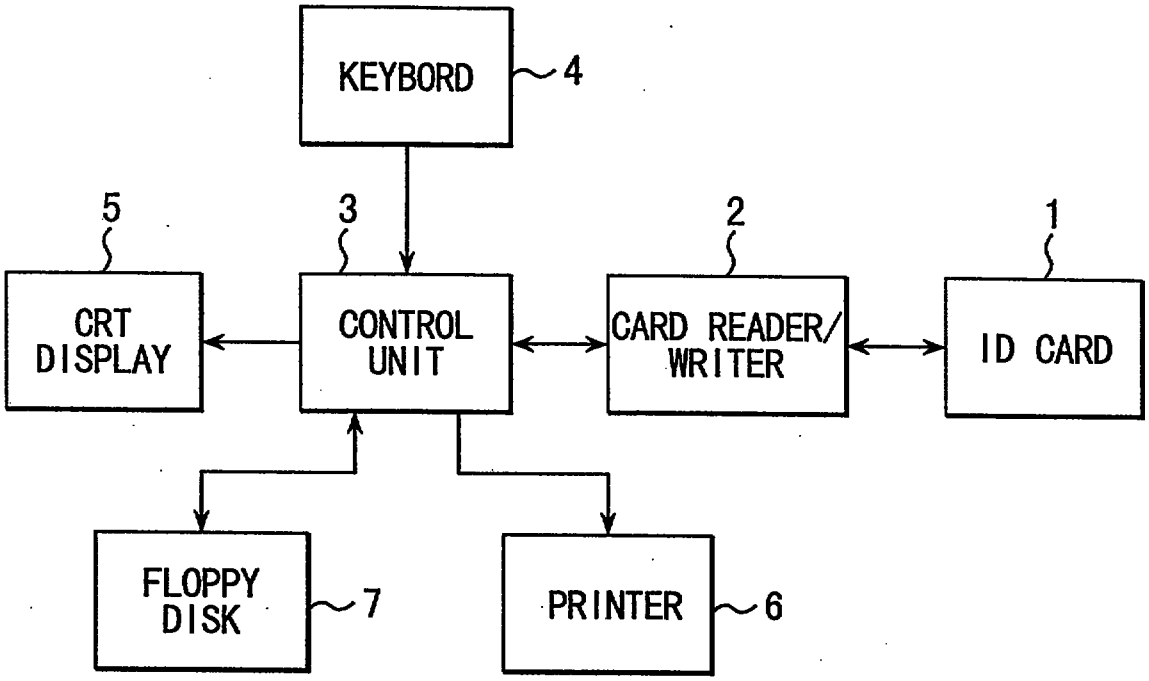

KEYBORD
4
5
3
2
1
CRT DISPLAY
CONTROL UNIT
CARD READER/ WRITER
ID CARD
FLOPPY DISK
7
PRINTER
6

FIG. 1A

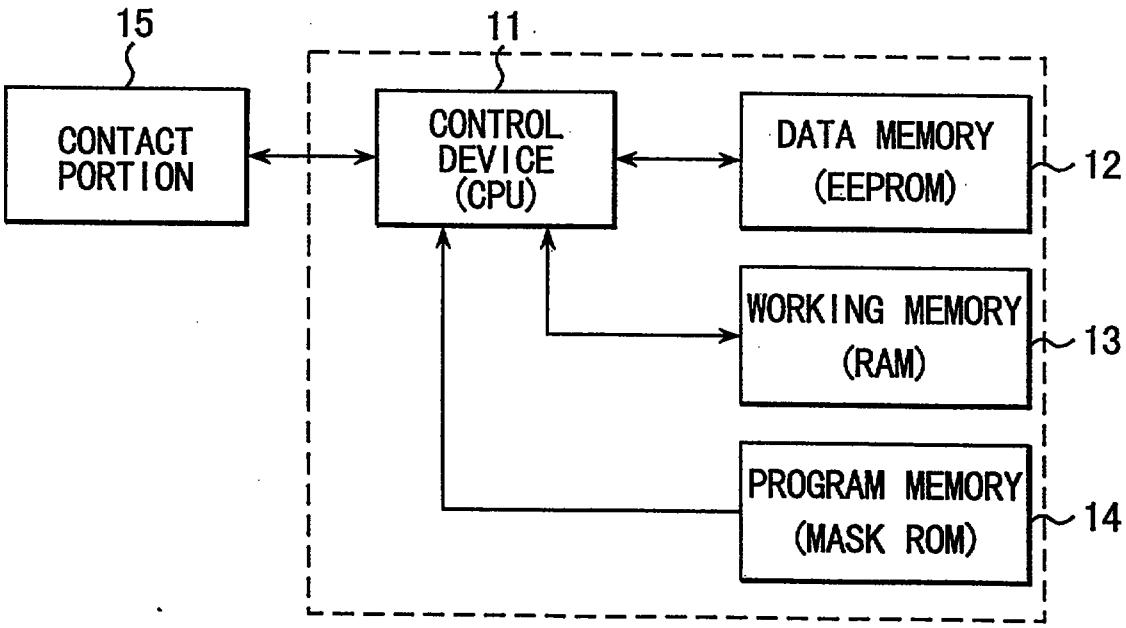

15
11
CONTACT PORTION
CONTROL DEVICE (CPU)
DATA MEMORY (EEPROM)
12
WORKING MEMORY (RAM)
13
PROGRAM MEMORY (MASK ROM)
14

FIG. 2

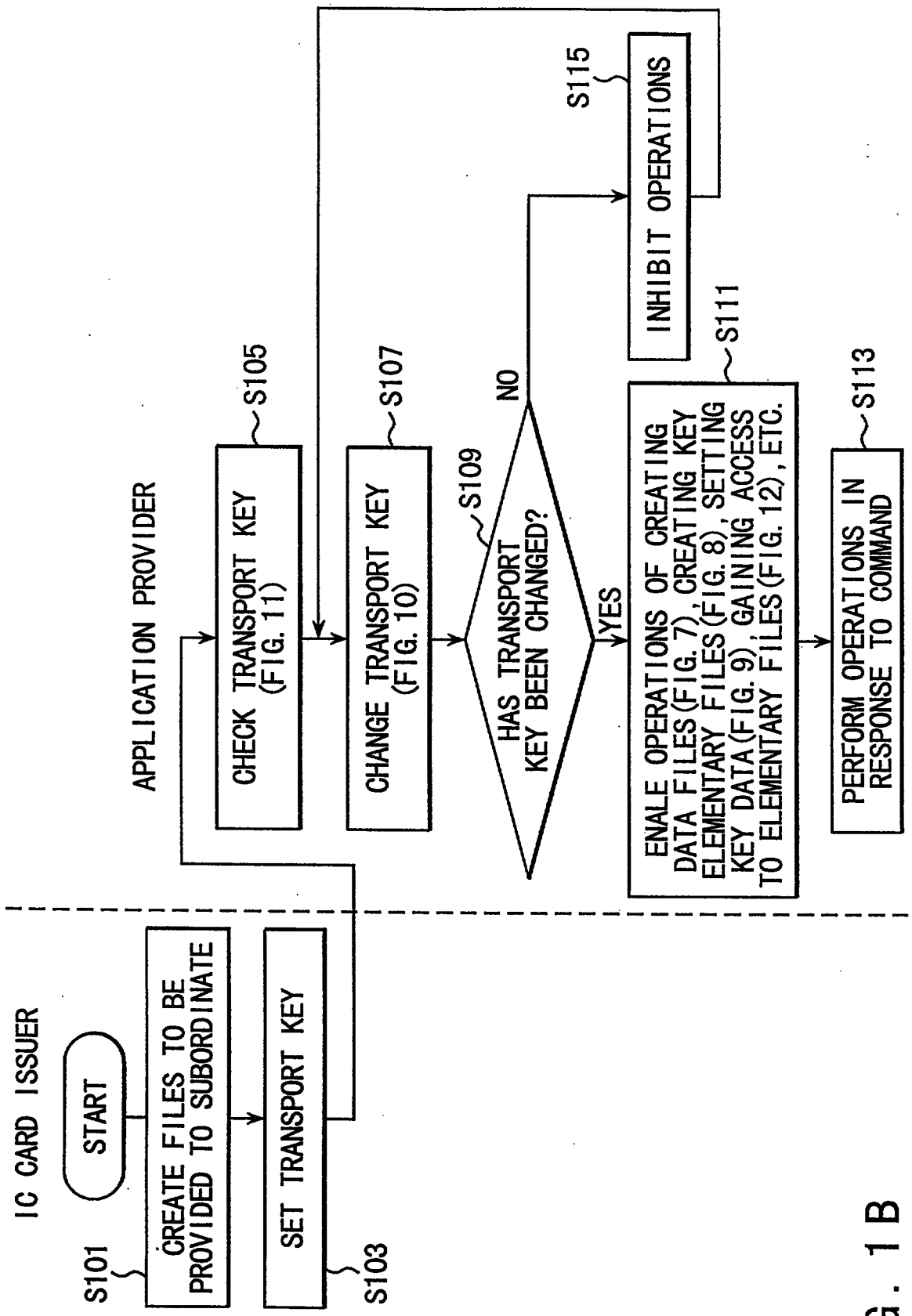

IC CARD ISSUER
START
S101
CREATE FILES TO BE PROVIDED TO SUBORDINATE
SET TRANSPORT KEY
S103
APPLICATION PROVIDER
S105
CHECK TRANSPORT KEY (FIG. 11)
S107
CHANGE TRANSPORT KEY (FIG. 10)
S109
HAS TRANSPORT KEY BEEN CHANGED?
NO
YES
S115
INHIBIT OPERATIONS
S111
ENALE OPERATIONS OF CREATING DATA FILES(FIG. 7), CREATING KEY ELEMENTARY FILES(FIG. 8), SETTING KEY DATA(FIG. 9), GAINING ACCESS TO ELEMENTARY FILES(FIG. 12), ETC.
S113
PERFORM OPERATIONS IN RESPONSE TO COMMAND

FIG. 1B

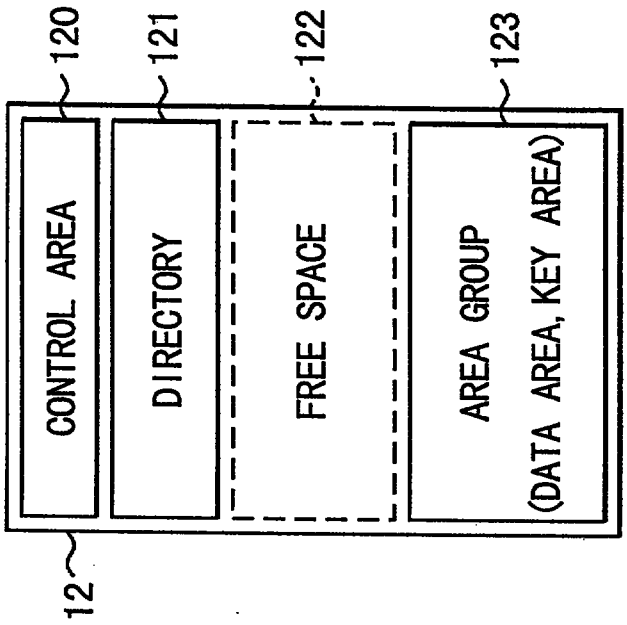

120
121
122
123
CONTROL AREA
DIRECTORY
FREE SPACE
AREA GROUP
(DATA AREA, KEY AREA)
12

FIG. 3

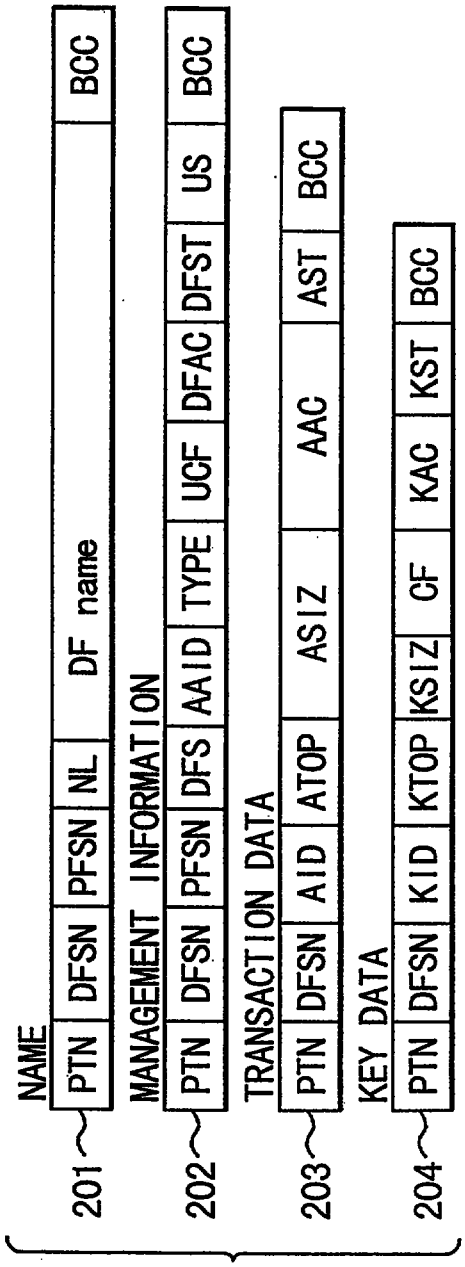

NAME
PTN
DFSN
PFSN
NL
DF name
BCC
201
MANAGEMENT INFORMATION
PTN
DFSN
PFSN
DFS
AAID
TYPE
UCF
DFAC
DFST
US
BCC
202
TRANSACTION DATA
PTN
DFSN
AID
ATOP
ASIZ
AAC
AST
BCC
203
KEY DATA
PTN
DFSN
KID
KTOP
KSIZ
CF
KAC
KST
BCC
204

FIG. 4

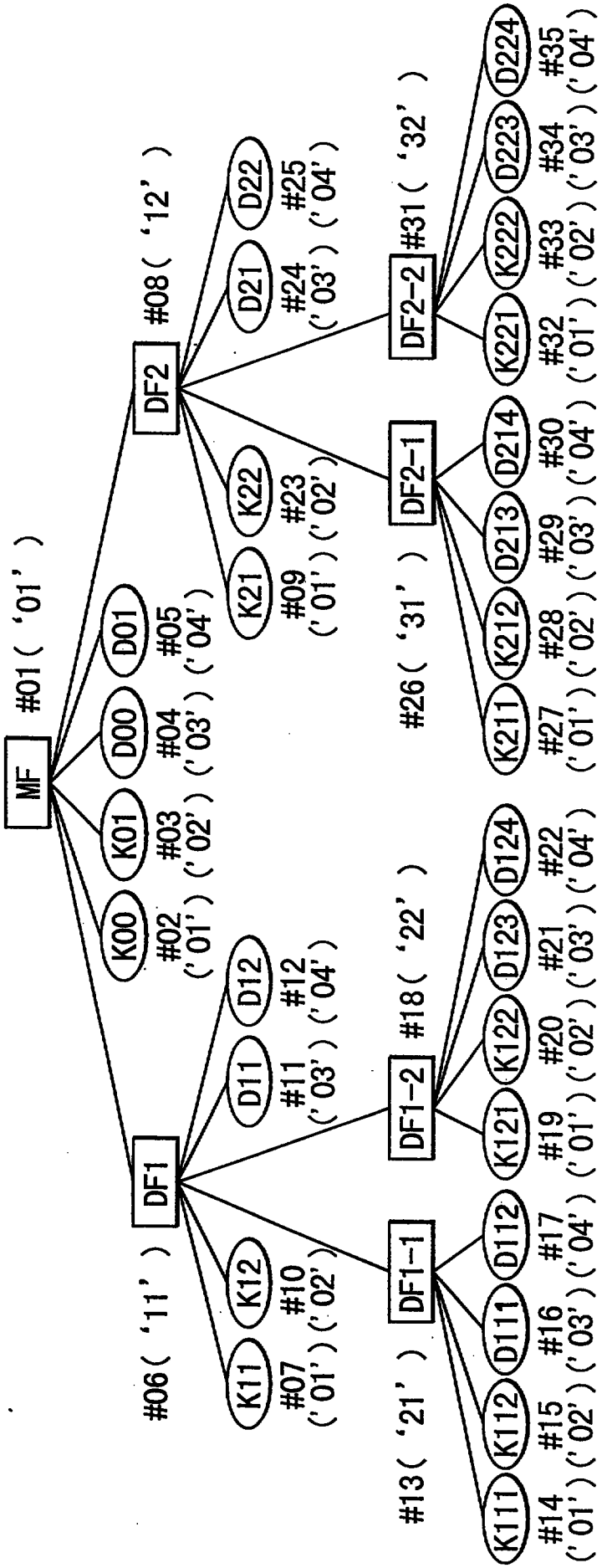

MF
#01 ('01')
K00
#02 ('01') ('02')
K01
#03 ('02') ('03')
D00
#04 ('03') ('04')
D01
#05 ('04')
DF1
#06 ('11')
K11
#07 ('01') ('02')
K12
#10 ('02')
D11
#11 ('03') ('04')
D12
#12 ('04')
DF1-1
#13 ('21')
K111
#14 ('01') ('02')
K112
#15 ('02') ('03')
D111
#16 ('03') ('04')
D112
#17 ('04')
DF1-2
#18 ('22')
K121
#19 ('01') ('02')
K122
#20 ('02') ('03')
D121
#21 ('03') ('04')
D122
#22 ('04')
DF2
#08 ('12')
K21
#09 ('01') ('02')
K22
#23 ('02')
D21
#24 ('03') ('04')
D22
#25 ('04')
DF2-1
#26 ('31')
K211
#27 ('01') ('02')
K212
#28 ('02') ('03')
D211
#29 ('03') ('04')
D212
#30 ('04')
DF2-2
#31 ('32')
K221
#32 ('01') ('02')
K222
#33 ('02') ('03')
D221
#34 ('03') ('04')
D222
#35 ('04')

FIG. 5

SERIAL NUMBER	DFSN	PFSN	FID	
#01	00	00	00	MF DEFINITION INFORMATION
#02	00	–	01	K00 DEFINTION INFORMATION
#03	00	–	02	K01 DEFINTION INFORMATION
#04	00	–	03	D00 DEFINTION INFORMATION
#05	00	–	04	D01 DEFINTION INFORMATION
#06	01	00	11	DF1 DEFINTION INFORMATION
#07	01	–	01	K11 DEFINTION INFORMATION
#08	02	00	12	DF2 DEFINTION INFORMATION
#09	02	–	01	K21 DEFINTION INFORMATION
#10	01	–	02	K12 DEFINTION INFORMATION
#11	01	–	03	D11 DEFINTION INFORMATION
#12	01	–	04	D12 DEFINTION INFORMATION
#13	03	01	21	DF1-1 DEFINTION INFORMATION
#14	03	–	01	K111 DEFINTION INFORMATION
#15	03	–	02	K112 DEFINTION INFORMATION
#16	03	–	03	D111 DEFINTION INFORMATION
#17	03	–	04	D112 DEFINTION INFORMATION
#18	04	01	22	DF1-2 DEFINTION INFORMATION
#19	04	–	01	K121 DEFINTION INFORMATION
#20	04	–	02	K122 DEFINTION INFORMATION
#21	04	–	03	D121 DEFINTION INFORMATION
#22	04	–	04	D122 DEFINTION INFORMATION
#23	02	–	02	K22 DEFINTION INFORMATION
#24	02	–	03	D21 DEFINTION INFORMATION
#25	02	–	04	D22 DEFINTION INFORMATION
#26	05	02	31	DF2-1 DEFINTION INFORMATION
#27	05	–	01	K211 DEFINTION INFORMATION
#28	05	–	02	K212 DEFINTION INFORMATION
#29	05	–	03	D211 DEFINTION INFORMATION
#30	05	–	04	D212 DEFINTION INFORMATION
#31	06	02	32	DF2-2 DEFINTION INFORMATION
#32	06	–	01	K221 DEFINTION INFORMATION
#33	06	–	02	K222 DEFINTION INFORMATION
#34	06	–	03	D221 DEFINTION INFORMATION
#35	06	–	04	D222 DEFINTION INFORMATION

FIG. 6

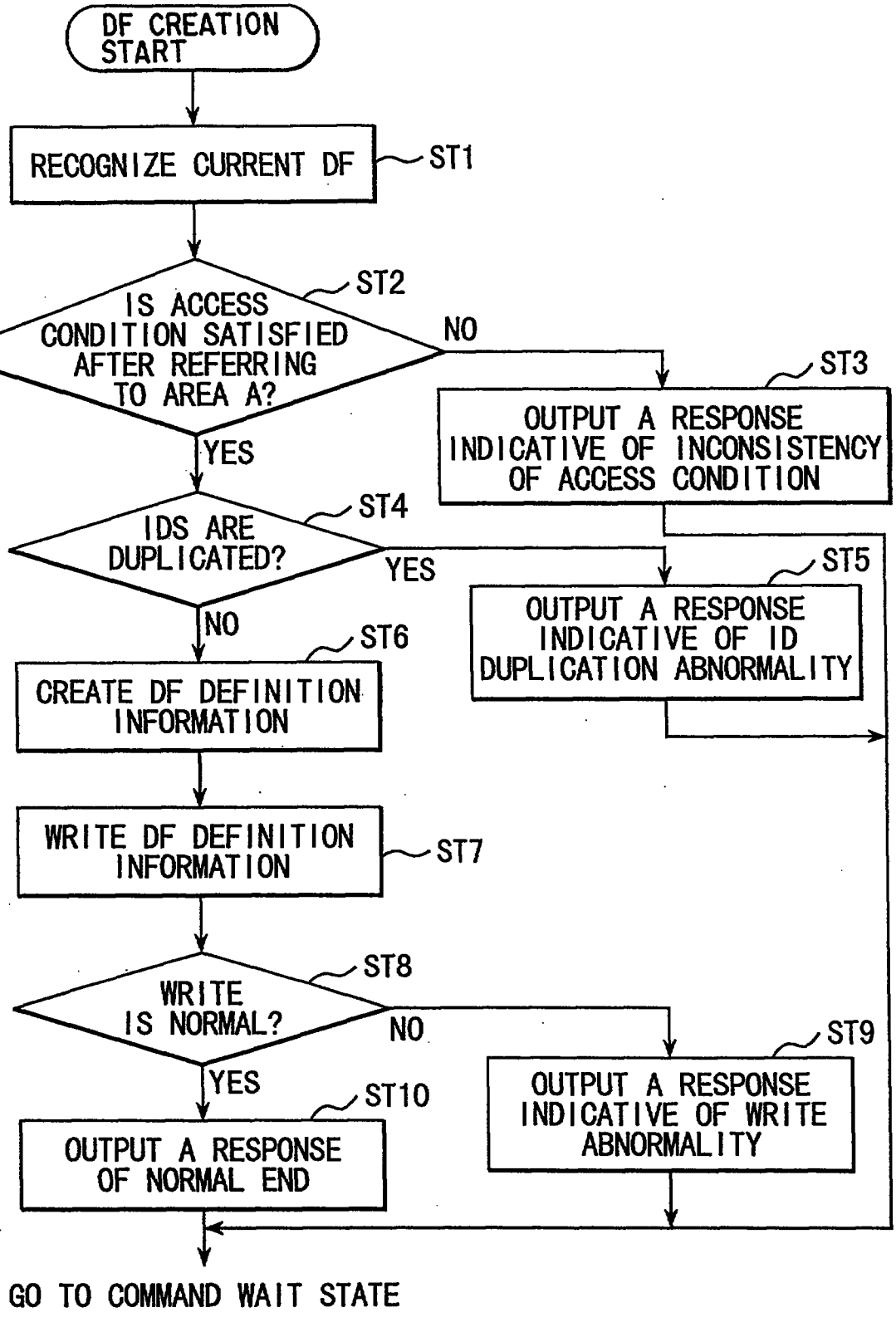

DF CREATION START
RECOGNIZE CURRENT DF
ST1
IS ACCESS CONDITION SATISFIED AFTER REFERRING TO AREA A?
ST2
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INCONSISTENCY OF ACCESS CONDITION
ST3
IDS ARE DUPLICATED?
ST4
YES
NO
OUTPUT A RESPONSE INDICATIVE OF ID DUPLICATION ABNORMALITY
ST5
CREATE DF DEFINITION INFORMATION
ST6
WRITE DF DEFINITION INFORMATION
ST7
WRITE IS NORMAL?
ST8
NO
YES
OUTPUT A RESPONSE INDICATIVE OF WRITE ABNORMALITY
ST9
OUTPUT A RESPONSE OF NORMAL END
ST10
GO TO COMMAND WAIT STATE

FIG. 7

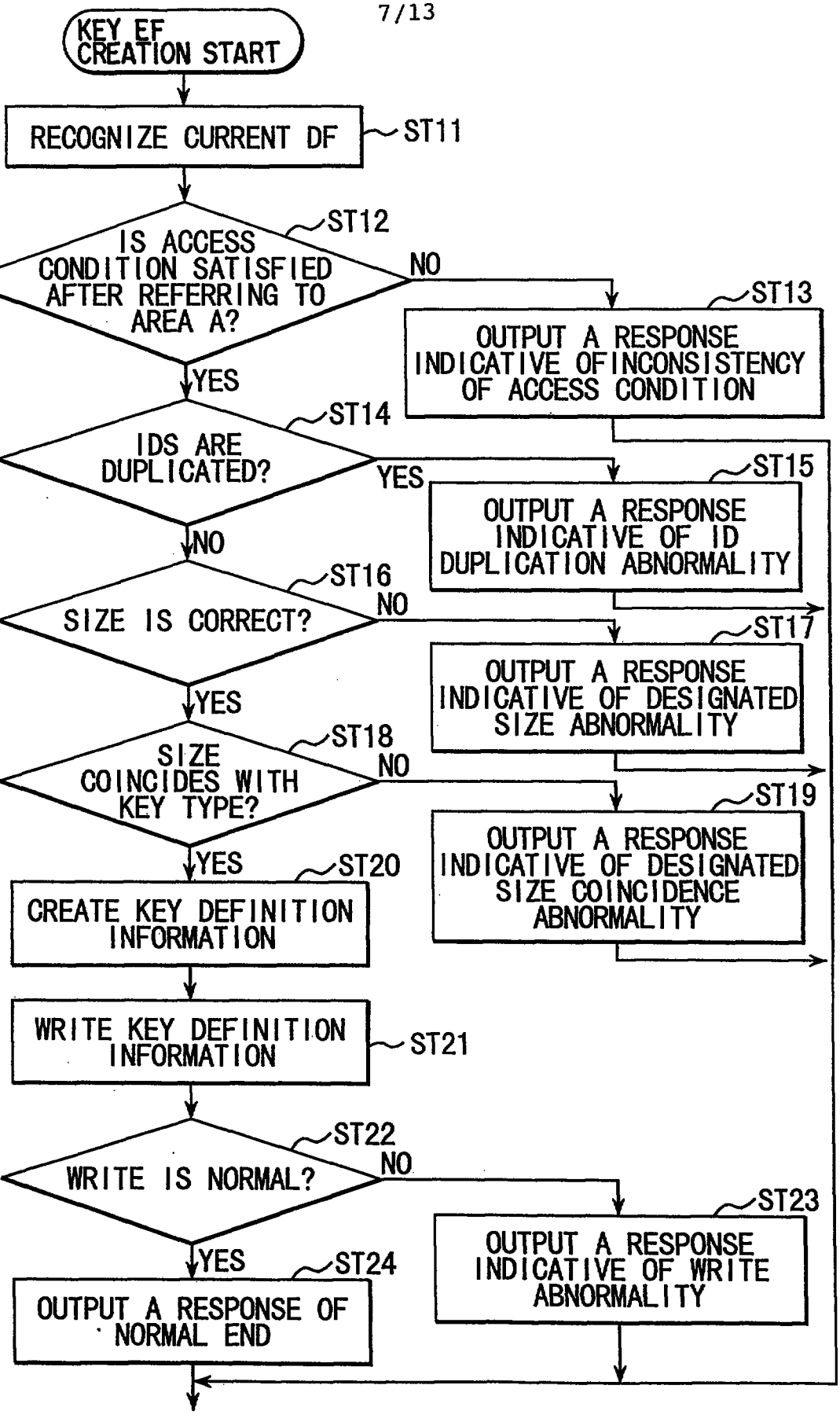

KEY EF CREATION START
RECOGNIZE CURRENT DF
ST11
IS ACCESS CONDITION SATISFIED AFTER REFERRING TO AREA A?
ST12
NO
YES
OUTPUT A RESPONSE INDICATIVE OFINCONSISTENCY OF ACCESS CONDITION
ST13
IDS ARE DUPLICATED?
ST14
YES
NO
OUTPUT A RESPONSE INDICATIVE OF ID DUPLICATION ABNORMALITY
ST15
SIZE IS CORRECT?
ST16
NO
YES
OUTPUT A RESPONSE INDICATIVE OF DESIGNATED SIZE ABNORMALITY
ST17
SIZE COINCIDES WITH KEY TYPE?
ST18
NO
YES
OUTPUT A RESPONSE INDICATIVE OF DESIGNATED SIZE COINCIDENCE ABNORMALITY
ST19
CREATE KEY DEFINITION INFORMATION
ST20
WRITE KEY DEFINITION INFORMATION
ST21
WRITE IS NORMAL?
ST22
NO
YES
OUTPUT A RESPONSE INDICATIVE OF WRITE ABNORMALITY
ST23
OUTPUT A RESPONSE OF NORMAL END
ST24
GO TO COMMAND WAIT STATE

FIG. 8

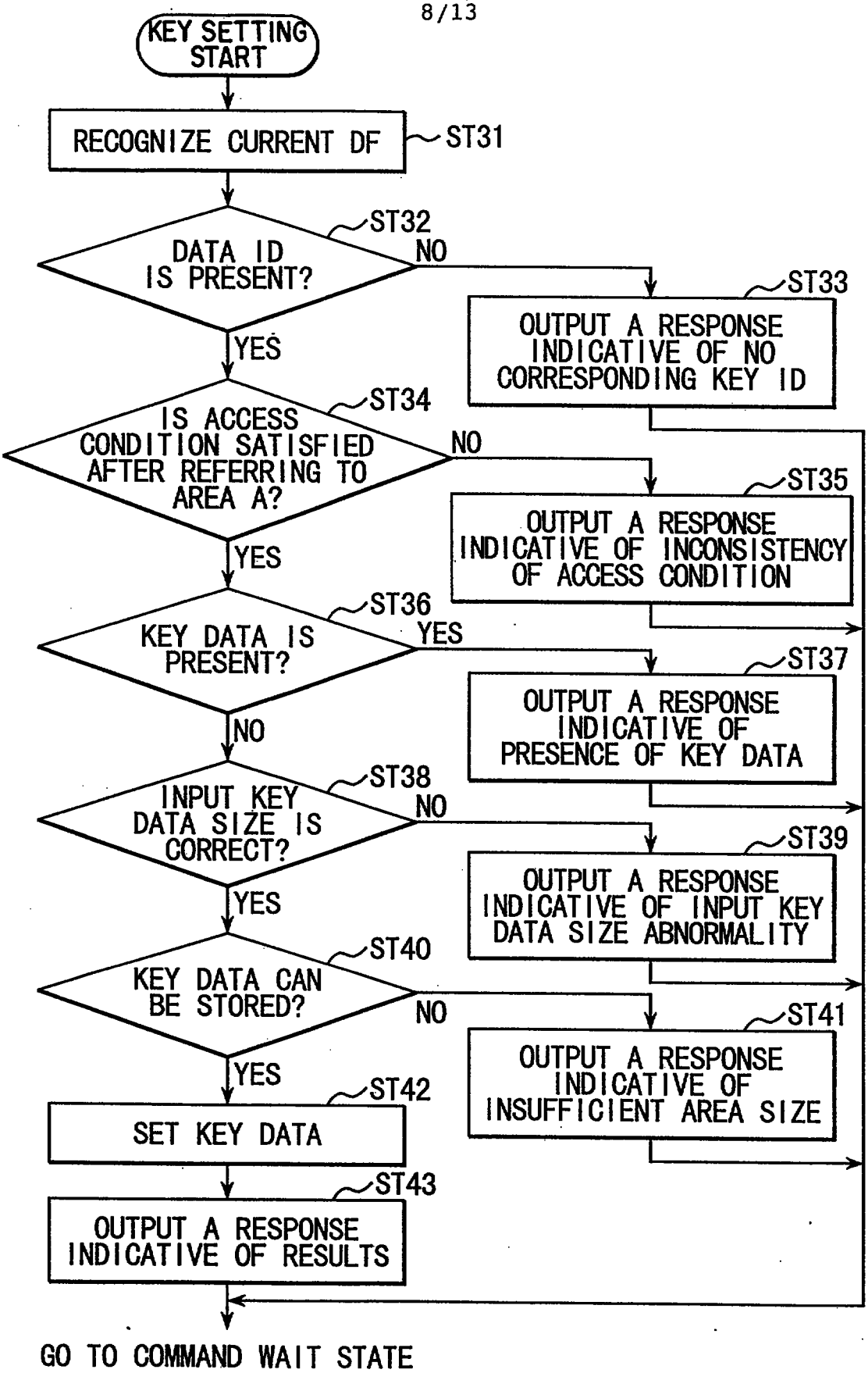

KEY SETTING START
RECOGNIZE CURRENT DF
ST31
DATA ID IS PRESENT?
ST32
NO
YES
OUTPUT A RESPONSE INDICATIVE OF NO CORRESPONDING KEY ID
ST33
IS ACCESS CONDITION SATISFIED AFTER REFERRING TO AREA A?
ST34
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INCONSISTENCY OF ACCESS CONDITION
ST35
KEY DATA IS PRESENT?
ST36
YES
NO
OUTPUT A RESPONSE INDICATIVE OF PRESENCE OF KEY DATA
ST37
INPUT KEY DATA SIZE IS CORRECT?
ST38
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INPUT KEY DATA SIZE ABNORMALITY
ST39
KEY DATA CAN BE STORED?
ST40
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INSUFFICIENT AREA SIZE
ST41
SET KEY DATA
ST42
OUTPUT A RESPONSE INDICATIVE OF RESULTS
ST43
GO TO COMMAND WAIT STATE

FIG. 9

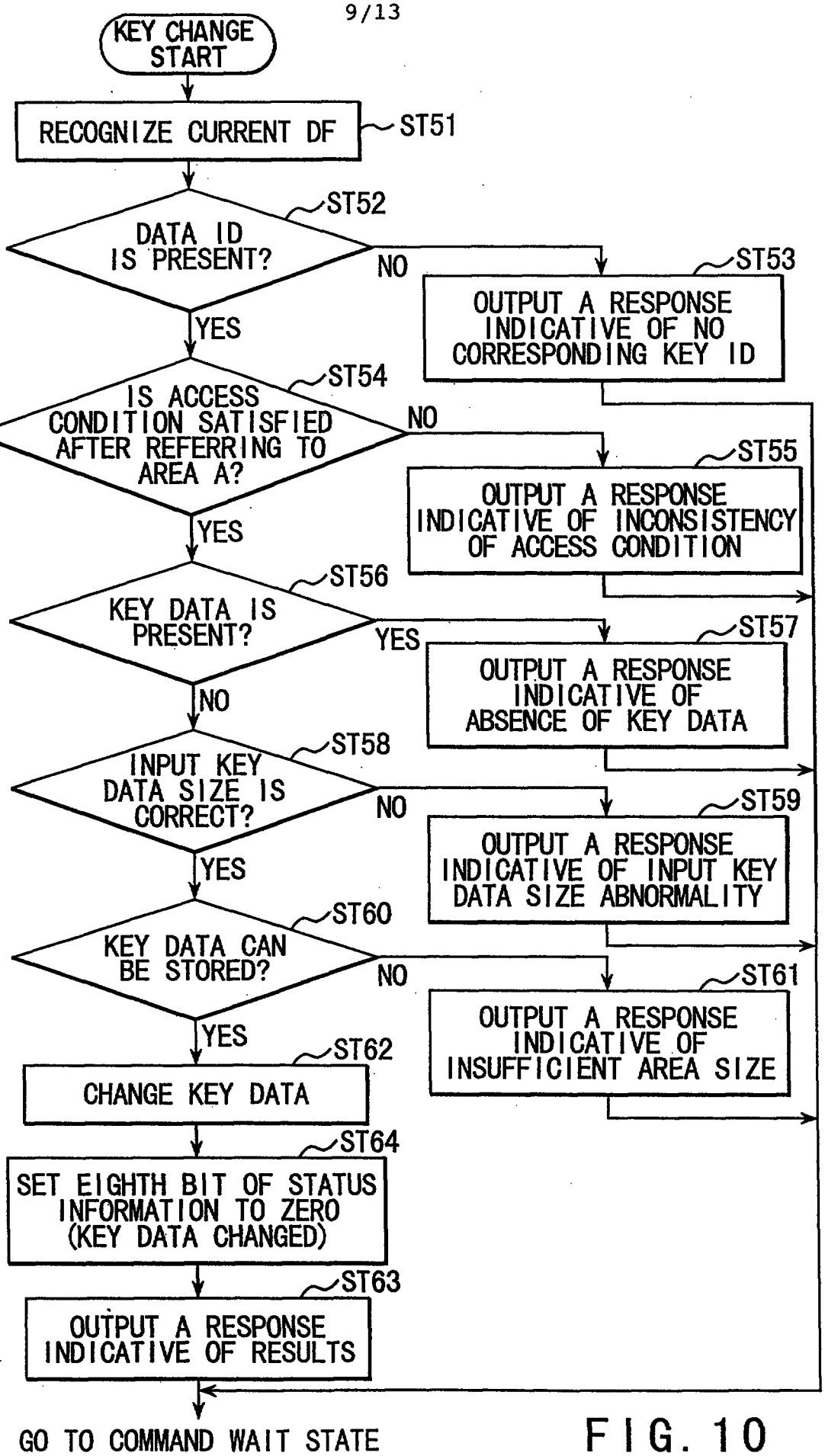

KEY CHANGE START
RECOGNIZE CURRENT DF
ST51
DATA ID IS PRESENT?
ST52
NO
YES
OUTPUT A RESPONSE INDICATIVE OF NO CORRESPONDING KEY ID
ST53
IS ACCESS CONDITION SATISFIED AFTER REFERRING TO AREA A?
ST54
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INCONSISTENCY OF ACCESS CONDITION
ST55
KEY DATA IS PRESENT?
ST56
YES
NO
OUTPUT A RESPONSE INDICATIVE OF ABSENCE OF KEY DATA
ST57
INPUT KEY DATA SIZE IS CORRECT?
ST58
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INPUT KEY DATA SIZE ABNORMALITY
ST59
KEY DATA CAN BE STORED?
ST60
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INSUFFICIENT AREA SIZE
ST61
CHANGE KEY DATA
ST62
SET EIGHTH BIT OF STATUS INFORMATION TO ZERO (KEY DATA CHANGED)
ST64
OUTPUT A RESPONSE INDICATIVE OF RESULTS
ST63
GO TO COMMAND WAIT STATE

FIG. 10

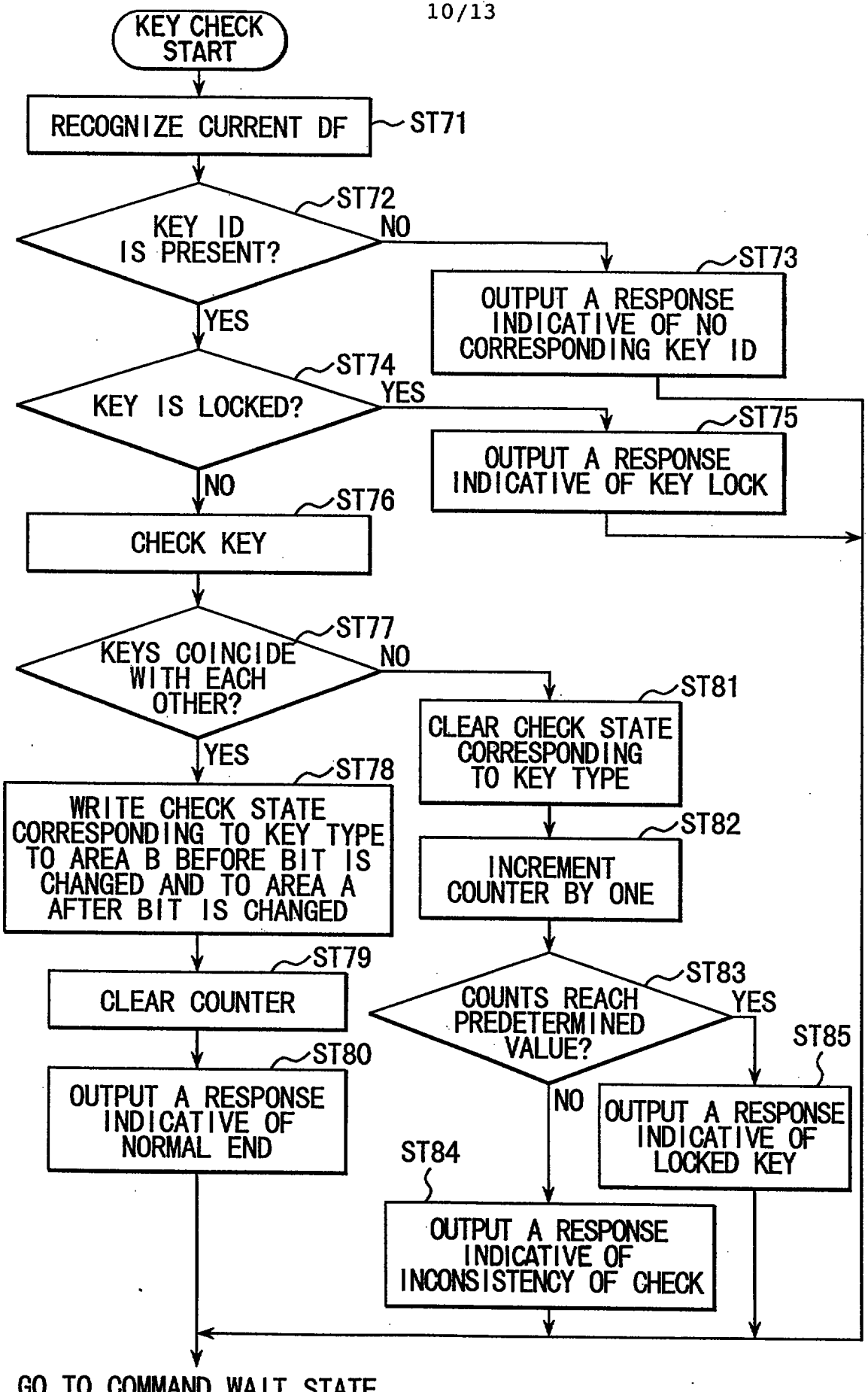

KEY CHECK START
RECOGNIZE CURRENT DF
ST71
KEY ID IS PRESENT?
ST72
NO
YES
OUTPUT A RESPONSE INDICATIVE OF NO CORRESPONDING KEY ID
ST73
KEY IS LOCKED?
ST74
YES
NO
OUTPUT A RESPONSE INDICATIVE OF KEY LOCK
ST75
CHECK KEY
ST76
KEYS COINCIDE WITH EACH OTHER?
ST77
NO
YES
CLEAR CHECK STATE CORRESPONDING TO KEY TYPE
ST81
WRITE CHECK STATE CORRESPONDING TO KEY TYPE TO AREA B BEFORE BIT IS CHANGED AND TO AREA A AFTER BIT IS CHANGED
ST78
INCREMENT COUNTER BY ONE
ST82
CLEAR COUNTER
ST79
COUNTS REACH PREDETERMINED VALUE?
ST83
YES
NO
OUTPUT A RESPONSE INDICATIVE OF NORMAL END
ST80
OUTPUT A RESPONSE INDICATIVE OF LOCKED KEY
ST85
OUTPUT A RESPONSE INDICATIVE OF INCONSISTENCY OF CHECK
ST84
GO TO COMMAND WAIT STATE

FIG. 11

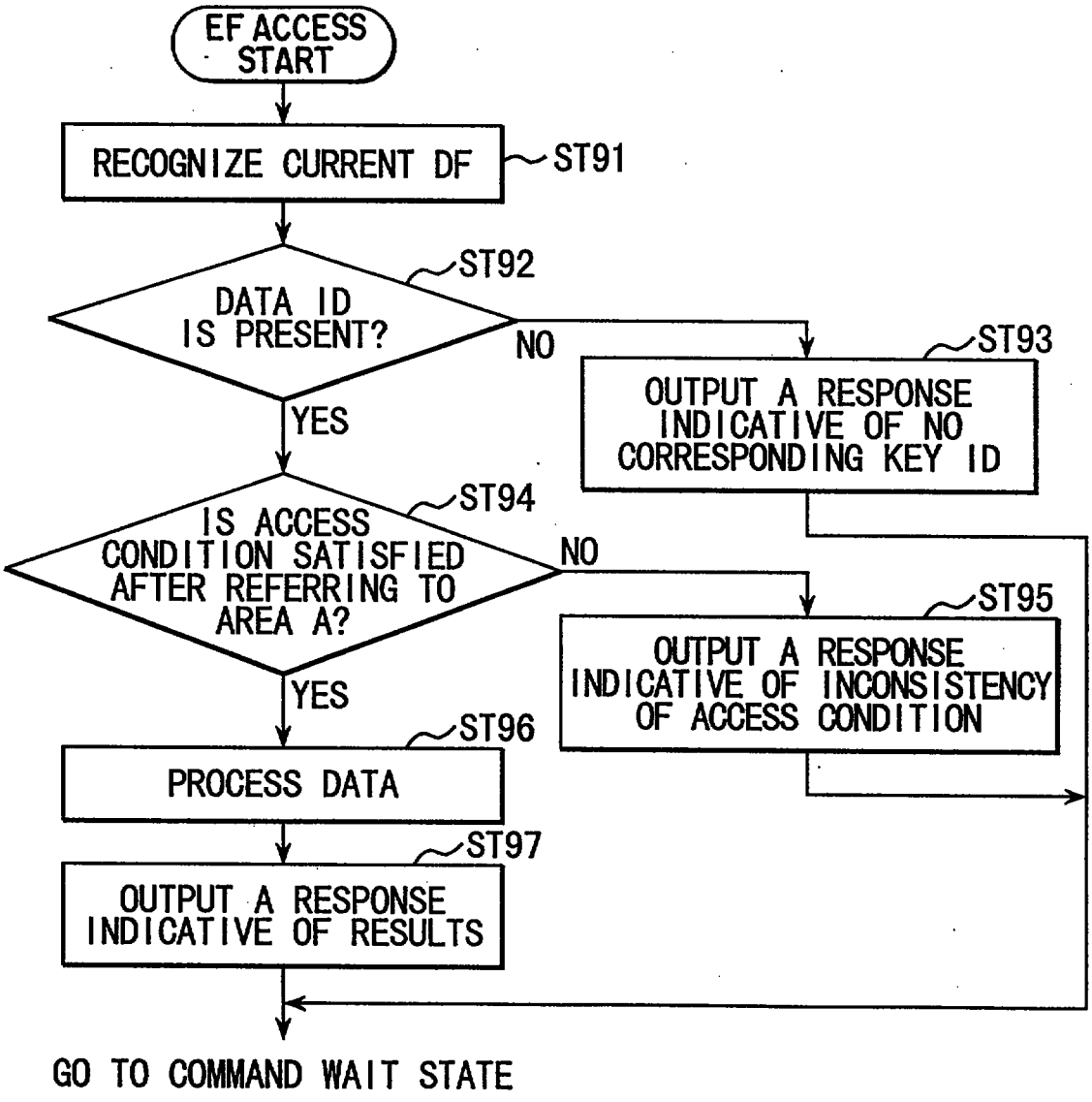

EF ACCESS START
RECOGNIZE CURRENT DF
ST91
DATA ID IS PRESENT?
ST92
NO
YES
OUTPUT A RESPONSE INDICATIVE OF NO CORRESPONDING KEY ID
ST93
IS ACCESS CONDITION SATISFIED AFTER REFERRING TO AREA A?
ST94
NO
YES
OUTPUT A RESPONSE INDICATIVE OF INCONSISTENCY OF ACCESS CONDITION
ST95
PROCESS DATA
ST96
OUTPUT A RESPONSE INDICATIVE OF RESULTS
ST97
GO TO COMMAND WAIT STATE

FIG. 12

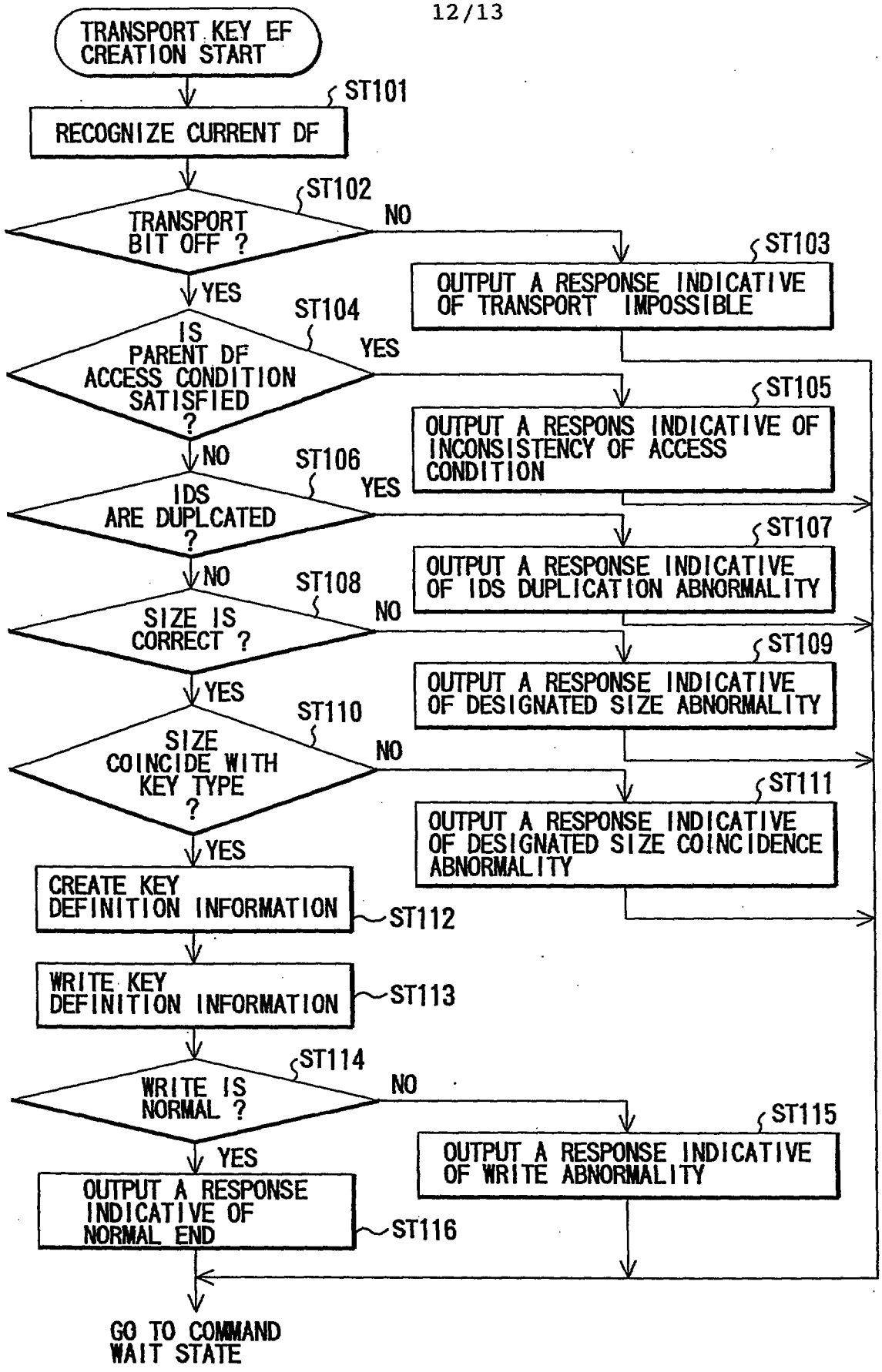

TRANSPORT KEY EF CREATION START
ST101
RECOGNIZE CURRENT DF
ST102
TRANSPORT BIT OFF ?
NO
ST103
OUTPUT A RESPONSE INDICATIVE OF TRANSPORT IMPOSSIBLE
YES
ST104
IS PARENT DF ACCESS CONDITION SATISFIED ?
YES
ST105
OUTPUT A RESPONS INDICATIVE OF INCONSISTENCY OF ACCESS CONDITION
NO
ST106
IDS ARE DUPLCATED ?
YES
ST107
OUTPUT A RESPONSE INDICATIVE OF IDS DUPLICATION ABNORMALITY
NO
ST108
SIZE IS CORRECT ?
NO
ST109
OUTPUT A RESPONSE INDICATIVE OF DESIGNATED SIZE ABNORMALITY
YES
ST110
SIZE COINCIDE WITH KEY TYPE ?
NO
ST111
OUTPUT A RESPONSE INDICATIVE OF DESIGNATED SIZE COINCIDENCE ABNORMALITY
YES
CREATE KEY DEFINITION INFORMATION
ST112
WRITE KEY DEFINITION INFORMATION
ST113
ST114
WRITE IS NORMAL ?
NO
ST115
OUTPUT A RESPONSE INDICATIVE OF WRITE ABNORMALITY
YES
OUTPUT A RESPONSE INDICATIVE OF NORMAL END
ST116
GO TO COMMAND WAIT STATE

FIG. 13

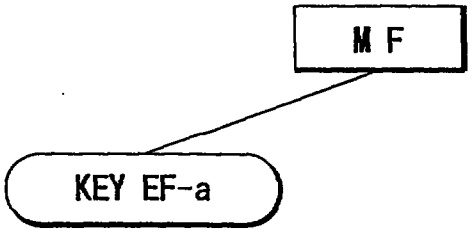


FIG. 14

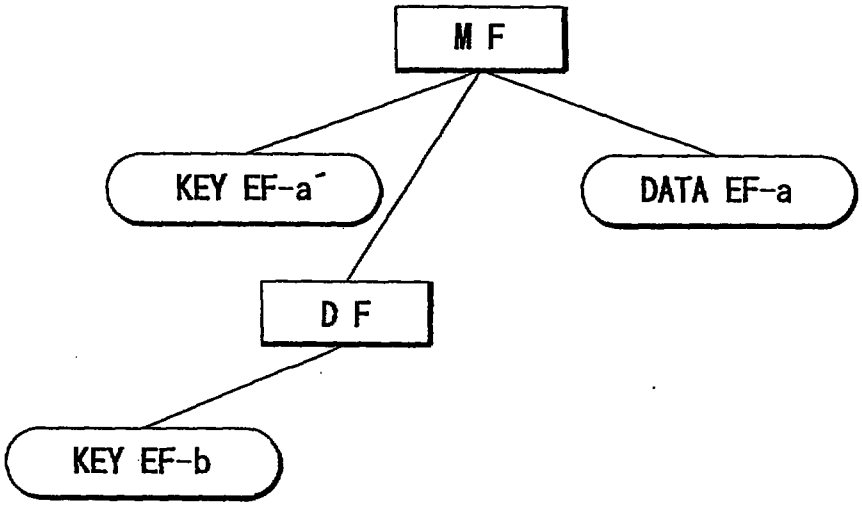


FIG. 15

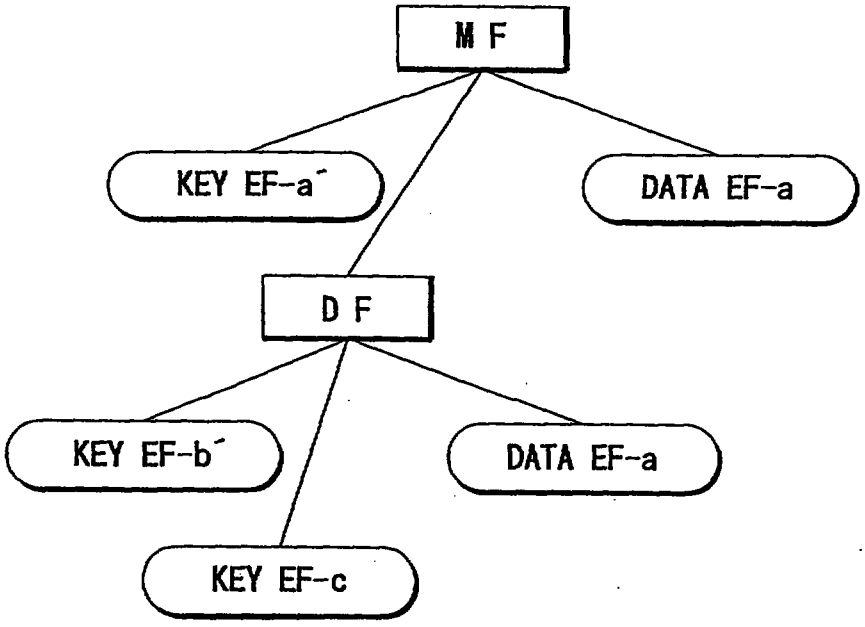


FIG. 16