PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{4} **:** **G06K 5/00**	**A1**	**(11) International Publication Number:** **WO 88/ 09019** **(43) International Publication Date:** 17 November 1988 (17.11.88)

(21) International Application Number: PCT/US88/01665

(22) International Filing Date: 16 May 1988 (16.05.88)

(31) Priority Application Number: 051,110

(32) Priority Date: 15 May 1987 (15.05.87)

(33) Priority Country: US

(71) Applicant: SMART CARD INTERNATIONAL, INC. [US/US]; 404 Park Avenue South, Suite 700, 7th Floor, New York, NY 10016 (US).

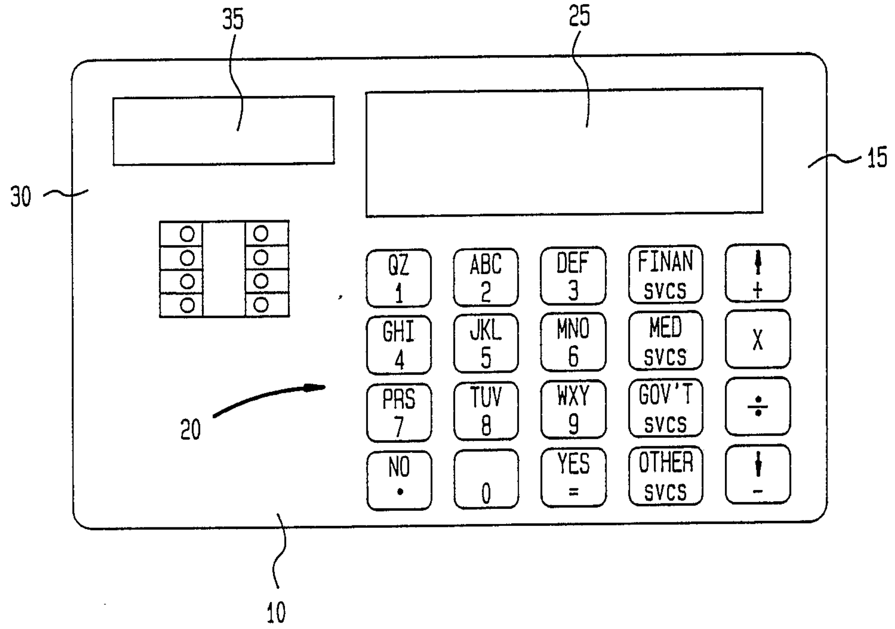
(72) Inventors: GRUPPUSO, Frank, M. ; 59 Valleywood Road, Commack, NY 11725 (US). MAZOWIESKY, Thomas ; 56 Academy Street, Patchogue, NY 11772 (US). RAMANI, Shantilal ; 88 Bethel Road, Albertson, NJ 11507 (US).

(74) Agent: MORRIS, Francis, E.; Pennie & Edmonds, 1155 Avenue of the Americas, New York, NY 10036 (US).

(81) Designated States: AT (European patent), AU, BE (European patent), BR, CH (European patent), DE (European patent), DK, FI, FR (European patent), GB (European patent), IT (European patent), JP, KR, LU (European patent), NL (European patent), NO, SE (European patent), SU.

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: INTELLIGENT PORTABLE INTERACTIVE PERSONAL DATA SYSTEM



(57) Abstract

An intelligent portable interactive personal data system is disclosed. A microprocessor (1100) with memory is contained within a transaction card-shaped housing (10). An alphanumeric keypad (20) and alphanumeric display (25) is located on a surface (15) of the housing (10). At least one port (1190) within the housing (10) is provided for the input and output of information. An operating system is stored in the memory to control the operation of the system through the microprocessor (1100). The operating system provides a device (1115) for generating a plurality of messages on the display (25) that prompts the user during the operation of the system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BG	Bulgaria
BJ	Benin
BR	Brazil
CF	Central African Republic
CG	Congo
CH	Switzerland
CM	Cameroon
DE	Germany, Federal Republic of
DK	Denmark
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
HU	Hungary
IT	Italy
JP	Japan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
MC	Monaco
MG	Madagascar
ML	Mali
MR	Mauritania
MW	Malawi
NL	Netherlands
NO	Norway
RO	Romania
SD	Sudan
SE	Sweden
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
US	United States of America

INTELLIGENT PORTABLE INTERACTIVE PERSONAL DATA SYSTEM

FIELD OF THE INVENTION

This invention relates to a card containing a microprocessor, memories and interfacing capabilities which is described herein as an "intelligent transaction card" or "ITC".

BACKGROUND OF THE INVENTION

The use of transaction cards has increased tremendously over the past couple of years. These cards are used for a variety of purposes, including, credit cards, security-identification cards to control access to secured areas and devices, and bank cards for use in automatic teller machines.

Typically, the transaction card has information encoded on the card to identify the cardholder. This information may be magnetically, electronically or optically encoded. For example, a financial transaction card such as a credit or debit card has the necessary account information stored in a magnetic strip on the back of the card.

As the use of transaction cards has increased, it has become desirable to increase the functionality in the cards by encoding additional information on the card, by making it possible to change such information, and by providing processing and/or input/output capabilities. For example, Moreno, U.S. Patent 3,971,916 describes the use of a semiconductor memory for the storage of information in a card. Ugon, U.S. Patent 4,211,919 describes the addition of a microprocessor to the card to control the input and output of information from the memory. Dreifus, U.S. Patent 4,575,621 also describes a microprocessor based transaction card which can operate in conjunction with a keyboard and terminal or operate in a stand-alone mode without a keyboard and terminal. In the stand-alone mode, the card monitors itself for abnormal conditions which may be caused by component failure or physical intrusion of the card.

These microprocessor based transaction cards are often referred to as "smart cards". However, these cards are still limited in many ways. Of particular interest, they generally are constructed to perform only a limited number of predetermined functions using predetermined areas of memory. Thus, some smart cards prohibit any modification of the programs stored in a card once the card is fully assembled. In Dreifus, system program information is stored in a ROM which cannot be changed. Instead the system information stored in ROM may be modified by information stored in RAM. However, the original programmed function of the card remains in the ROM and is never permanently changed or removed from the card. As a result the card has limited flexibility and typically must be replaced if one desires to change its functionality.

Additional problems arise in attempting to modify prior art smart cards. Because of the limited code memory (and command set) in these devices, prior art smart cards do not perform the memory management task. Rather, the physical address of the data to be stored in or retrieved from the card is determined outside the card and down loaded to the card to access the memory. Specifically, the terminal and/or the application software in a host computer or an intelligent card reader performs the task of determining where in the card memory data is to be stored or retrieved. While such use of physical addresses may be acceptable when only one application is used on the card, it creates numerous problems in supporting cards having multiple applications.

In the first place, since physical addresses are used, the providers of all the application programs used in the cards must coordinate their efforts to avoid using the same memory locations on the card. As will be apparent, this requires at a minimum substantial coordination among the different providers of application programs for the card. In practice, however, it means that new applications cannot be loaded at a later date since the different applications may not be compatible. As a result, all the applications on a card must be known at issuance.

In addition, problems are created when a card with larger memory is issued. For example, the terminals must be updated so that they can address the enhanced memory size of the card.

Further, terminals or card readers that provide access to multiple services on prior art cards must be programmed to operate with the various combinations of applications and file structures that exist on cards or the terminals cannot access data in the cards. Due to the rigid structure and limited command set of prior art cards, the terminal is used to implement a high level system interface that controls the functionality of the card according to application programs used in conjunction with the card. This results in different readers having different sets of commands, limiting the interchangeability of cards across different readers. In addition, the reader device is usually an intelligent device, increasing the cost of the installation of smart card systems. At the same time, because of the multiplicity of applications, not all readers can be programmed with all possible structures that may exist on the cards. It is therefore possible to access the wrong data in the card.

The security systems of the previous cards were limited as well. Typically data was split into free access and secure zones, and a limited number of keys were provided.

These keys could not be encoded or encrypted, and methods for verifying the authenticity of the card or the reader device were limited.

Finally, these cards were not designed to be "user friendly". With prior art transaction cards, the cardholder has to memorize the proper procedure to operate the card.

SUMMARY OF THE INVENTION

In the present invention, a general-purpose reprogrammable intelligent card is disclosed. The card includes an alphanumeric keypad, an alphanumeric display and one or more input/output ports controlled by a microprocessor and programs stored in a memory associated with the microprocessor. The microprocessor is provided with an operating system and may be programmed or reprogrammed for a specific application or for a variety of applications.

The system can be used in conjunction with a terminal device or independently in a stand-alone mode. The card is menu-driven and user friendly. It prompts the cardholder on proper operating procedure and displays clear, concise messages that the cardholder can understand. The flexibility of the card is increased due to the placement of the memory management function in the card instead of in the terminal device thereby enabling the application programs to be resident on the card. An enhanced security system is also provided by the card allowing multiple levels of security to prevent the use of the card by unauthorized persons.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the transaction card of the present invention will be apparent from the following detailed description of the preferred embodiment in which:

FIGS. 1A and 1B illustrate the front and back views of one embodiment of the transaction card of the present invention.

FIGS. 2A and 2B illustrate the front and side views of another embodiment of the transaction card of the present invention.

FIG. 3 illustrates the operating system flow.

FIG. 4 is a flowchart of the keyboard module.

FIG. 5 is a flowchart of the update clock/calendar auto-shutoff module.

FIG. 6 is a flowchart of the communications module.

FIGS. 7A and 7B is a flowchart of the keyboard service routine.

FIG. 8 is a flowchart of the display service routine.

FIG. 9 is a flowchart of the communications service routine.

FIGS. 10A-10AA illustrate the memory structure and the memory management service routines.

FIG. 11 is a flowchart of the application service routine.

FIG. 12 is a flowchart of the system clear/restart service routine.

FIG. 13 illustrates the basic structure and access of the application programs.

FIG. 14 is a flowchart of the cardholder notes application program.

FIG. 15A is a flowchart of the set time application program.

FIG. 15B is a flowchart of the set date application program.

FIG. 15C is a flowchart of an application program to change the cardholder's personal identification number (PIN).

FIG. 16 is a flowchart of the credit/purchase application program.

FIG. 17 is a flowchart illustrating the use of the ITC for transportation.

FIG. 18 is a flowchart for the calculator emulator application program.

FIG. 19 illustrates a preferred embodiment of the display of the transaction card of the present invention.

FIG. 20 is a schematic of one embodiment of the transaction card of the present invention.

FIG. 21 is a schematic of the CPU and custom integrated circuit used in another embodiment of the present invention.

FIG. 22A and 22B illustrates the location of the magnetic card interface in the transaction card of the present invention.

FIG. 23 illustrates the configuration of the ITC to the terminal interface.

FIG. 24 illustrates the circuitry in the ITC terminal interface.

FIG. 25 illustrates the ITC circuitry to read magnetically encoded information from the terminal device.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIGS. 1A and 1B are an illustration of one embodiment of the present invention, a general purpose programmable intelligent transaction card (ITC) 10. Although the card can be of any size, it is preferred that the length and width dimensions of the card are approximately the same as the 3.375" x 2.125" (86mm x 55mm) dimensions of conventional transaction cards. A multifunction alphanumeric keypad 20 and liquid crystal display 25 are located on a front surface 15 of the card. A plurality of electrical and/or optical contacts 30 provide a means for the input and output of information from the ITC. Power for the card may be supplied by a battery (not shown) mounted in the card or by an external source connected to the card through some of contacts 30. Advantageously, an electromagnetic energy receiver such as a solar cell 35 may be used to supply power to the ITC. In addition a magnetic strip 50 may be provided on back surface 55 of the card through which the ITC may interface with existing magnetic strip-based transaction card systems. Further details of the circuitry of the ITC are set forth in conjunction with FIGS. 19-21.

As illustrated in FIGS. 2A and 2B, if the transaction card of the present invention is thicker than the 30 to 65 mil (0.762 to 1.651 mm) thickness of a conventional transaction card, a bottom portion 70 having a width sufficiently narrow to fit into conventional magnetic card readers is provided with a magnetic strip 80 attached to a surface thereof.

Although the ITC card may be tailored for one specific application or type of transaction, the card is designed with an operating system which provides sufficient intelligence and flexibility to be used in conjunction with a variety of application programs. For example, the same ITC card may be programmed to keep track of the cardholder's banking activities, to charge a purchase, or to identify the cardholder in order to gain access into a secured area. It may also be used to store information such as medical histories, travel records, addresses and telephone numbers and appointment calendars. In addition, the ITC card may be programmed to keep time and generate visual or audio alarms and to perform calculations, such as totalling deposits and purchases. The ITC may also be used to independently authorize a credit transaction and generate an approval code thus eliminating the need to use a bank terminal. A feature of the invention which facilitates the use of many different application programs in an ITC is the use of alphanumeric display 25 which provides a menu-driven user interface.

Not only may the same card be used for different applications or types of transactions, but the application programs may be removed, changed, or added to the card at any time. The danger of the software becoming obsolete is minimal since the software may be upgraded quickly and easily. At the same time, since the function of the card depends on the software, a card having the same physical components may be used in numerous applications, thus permitting high volume production and accompanying cost reductions.

The application programs may only be input by authorized manufacturers or issuers of the ITC. This secures the ITC against unauthorized access and programming or reprogramming. The application programs are preferably input by the issuer of the card through the input/output ports. For example, if the ITC is used for payment of public transportation, the transportation authority would load the application into the ITC and issue the card to the cardholder. Once the application is loaded it may be protected by a personal identification number (PIN) or the equivalent such as biometric parameters which must be input into ITC before access to reprogram the application is permitted.

The application information programmed into the card is dependent upon the type of transaction the card is to perform. However, the information generally includes communication protocols, security information, transaction process protocols, as well as cardholder-specific information related to the transaction. For example, if the card is programmed to be used for banking transactions at automatic teller machines (ATM), the application information would include the protocol information for the card to communicate with the ATM, the security code in order to access the ATM, cardholder identification and account information.

A distinct advantage of the ITC is the secured nature of the card. The card may be programmed to any desired level of security. For example, before the cardholder can access any feature or function of the card, he may be required to enter in the proper security code. If data is transmitted from the ITC to another device, the data may be encrypted before transmission. Access to the software programmed into the ITC may have additional security attached to it such that only the supplier of the software and not the cardholder may read or modify the software.

The operating system provides the intelligence and flexibility required for the device. The operating system is an organized collection of programs and data that is specifically designed to manage the resources of the ITC and to simplify the application programs and control their execution on the ITC. These programs are described in detail in conjunction with FIGS. 3-12. Illustrative application programs that may be used in the ITC are described in conjunction with FIGS. 13-17.

The operating system software is written in modular fashion. This allows for dynamically reconfiguring the modules for ease in adding or modifying features. Referring to FIG. 3, the system runs a plurality of system modules in a continuous serial manner. These modules are: keyboard module 100, clock module 105, and communication module 110. The operation of these modules is explained in conjunction with FIGS. 4, 5 and 6, respectively.

The operating system also provides a plurality of service routines to operate or service basic system functions. These service routines include a keyboard service routine, a display service routine, a communication service routine, a memory management service routine, an application service routine, and a system clear/restart service routine. These routines are explained in conjunction with FIGS. 7-12, respectively.

These service routines are utilized by the application programs and the system modules. The modules and service routines are addressed through a vector table allowing for the change or substitution of modules without a major change in the system. The vector table acts like an index to indicate where in memory each module is located.

While in the "idle" state, i.e. while no application programs or service routines are operating, the modules operate in a polling configuration wherein each module is checked sequentially to determine if it has any activity that must be serviced. If the module does not need any servicing, the control of the system will pass on to the next module. For example, keyboard module 100 checks to see if a key has been depressed on the keyboard. If no key has been depressed, the routine is exited and the next module is executed. If a key has been depressed, the keyboard module then reads the keyboard entry. It also provides for the "debouncing" of the key so that no more than one signal is transmitted for each key that has been depressed.

As shown in FIG. 4, the keyboard module first activates at step 122 a keyboard service routine to determine if a key has been depressed or otherwise selected by the cardholder. At step 124 the module tests if a valid key has been entered. If not, the module is exited. If a valid key has been entered, system control is switched to the application service routine at step 128 which determines the particular application, if any, that is requested by the valid key that has been entered. The module is then exited.

After system control is returned from the keyboard module, update clock/calendar auto-shutoff module 105 is activated. The actual clocking of time is implemented in hardware. A crystal provides the timing for an interrupt to increment the elapsed time every 1/2 sec. If the clock is equal to 2400, the date is incremented and the time is reset. The clock module tracks the current date and time on the display as well as controls the auto-shutoff procedure for the system in the idle state. The auto-shutoff procedure automatically turns the ITC off after a predetermined time period has elapsed during which time there had been no activity on the system. An auto-shutoff feature is also provided when the system is not in the idle state.

Referring to FIG. 5, if the date and time are not currently displayed at step 225, the module is exited at step 230. If the date and time are displayed, this indicates the system is on and is in an "idle state", i.e. no applications are currently operating. The date and time are updated at step 235 and the amount of time since the last system activity is checked at step 240. If the time which has elapsed since the last system activity is less than the auto-shutoff time limit, the module is exited, system control is returned and the communication module is initiated. Otherwise the ITC is shut off at step 245 and the module is exited.

Communication module 110 determines if there is any information at the communication ports. If there is information at these ports, the communication service routine shown in FIG. 6 is initiated to read the information into the system and the application service routine is activated to determine the application requested.

Referring to FIG. 6, at step 300 the first communication port is tested or polled. The port is polled by testing the port reset line. If the reset line is low, data is at the port; and at step 315 the communication service routine is activated. The communication service routine executes the proper data handshaking with the proper timing, reads in the data transmitted to the data port and stores it in a memory buffer. Since the only information that is presented to a communication port relates to a specific application, the application service routine is initiated at step 317 after the information is received from the port.

After the first port is serviced, the system tests at step 340 whether all the ports were polled. If all the ports were not polled, steps 300, 305, 315, 317 and 340 are executed for each port until all the ports have been polled. If at step 340 all the ports were polled, the module is exited.

As mentioned earlier, the operating system also comprises a variety of service routines that service or control some of the basic functions of the ITC. These routines are depicted in the flowcharts of FIGS. 7-12.

To understand the keyboard service routine shown in FIGS. 7A and 7B, it is helpful to understand keypad 20. Each key of the keypad is identified by an x-y coordinate pair in an x-y matrix. In each column, the keys are connected to an x-line to which a scanning signal may be applied; and in each row the keys are connected to a y-line on which a signal appears when a key is depressed (or otherwise selected) and its x-line is scanned. To determine whether a key has been depressed, a series of test signals is sent to the x lines of the matrix and the corresponding signal states at the y lines are read and stored.

As shown in FIG. 7A, at step 125 the scan signal for the first x line of the matrix is written to the matrix and at step 130 the signals on the y lines are read and stored in memory. At step 135 it is determined whether all the x lines at the matrix have been written to. If the last x line has not been written to, a scan signal is then written to the next x line at step 140 and the signals on the y lines are read and stored in memory at step 130.

After the last x line has been tested, the system checks at step 150 all the signals received on the y lines to determine if a key was depressed. If no key was depressed, the key register is set to a "no key" value at step 155 and the module is exited.

If a key has been depressed, the key value is determined at step 158. The debounce count timer is then initiated at step 160. If the debounce count is not equal to the time limit at step 165, the debounce count is incremented at step 168 and continues, in steps 165 and 168, until the debounce count is equal to the time limit. The time limit is of a duration such that any bouncing of the keys that may have occurred has ended. At this point, step 170, the matrix is written to and read again as described in steps 125, 130, 135, 140 and 145. At step 175, the newly read signals on the y lines are tested to determine if they are equal to the previously read signals. If not, the purported key depression is invalid, the debounce count is reset at step

185 and the module is exited. If the new signals are equal to the previous signals, the key is valid and the particular key value is stored in the key register at step 195. The debounce count is reset at step 200 and the module is exited.

The display service routine provides the necessary logic to operate the display. When the system determines during the operation of the modules, application programs or other service routines, that the display is to be changed, the display service routine is activated. Referring to FIG. 8, the system determines at step 400 if new data is to be displayed. If no new data is to be displayed, the routine is exited. If new data is to be displayed, at step 410 each character of the new data is converted into segments for a segmented display. This is done through a lookup table which identifies the segments of a display that correspond to each of the possible characters. At step 415 the segment identification is stored in the RAM display buffer. A separate processor, the display controller, cycles through the display buffer and turns the display segments on/off according to the segment identification in the display buffer.

The communication service routine manages the actual input and output of information through the input/output ports in response to a request to read or write data. Referring to FIG. 9, at step 500 the program tests if data is to be read from a port. If data is to be read from a port, at step 505 the appropriate communication handshaking is executed and the data is read in from the input/output port. The data read in is stored at step 510 in a buffer for subsequent use by the module, service routine or application program that requested the communication service routine.

If the communication service routine is required to output data, at step 515 communication is established through the appropriate handshaking at the port with the outside device that is to receive the data. Once the communication is established, at step 520 the data is output through the port. At completion the routine is exited.

The memory in the ITC is separated into three areas: system data area, application data area and transaction data area. The system data area contains basic background system information. The application data area contains the program code for each application. The transaction data area contains data used in specific application programs. The memory management service routine supervises and controls the allocation and use of memory in the three data areas, thereby permitting the addressing of files by logical address not physical address. In addition, the memory management routine provides timing sequences and control signals for proper operation.

More particularly, the card stores transaction data in non-volatile memory, organizing the data into a set of files. There may be any number of files, up to the maximum size of the memory. The memory size may vary, since the operating system accesses the data in a logical, rather than physical manner.

The card may contain multiple files, each one identified by a unique name. These files may have multiple levels of security associated with each file, the file being accessible only after presentation of the proper key(s).

The memory management routine utilizes two tables to control the access to the individual files stored in memory. The first table shown in FIG. 10A is a security table. The security table contains fourteen security flags which may be given a value which indicates that the security is on or off. These values are initially set at the beginning of a session based on the values stored in a common data file (CDF) described below and may be modified through key commands which are executed during the application program. Through the use of the security table the execution of certain commands can be prevented during the execution of all or part of the application program. For example after data has been written into the file, the read command may be executed but the execution of the write command may be prevented to protect the data from being modified.

The other table used in maintaining the secure nature of a system is the Command Access Table illustrated in FIG. 10B. The Command Access Table is used to store the particular security flag to which the routine should refer to determine whether the command should be executed on a particular file. For example, a write command may be identified by a code equal to five. Referring to FIG. 10B, the location corresponding to the command code five has a value of five which indicates to the system to refer to security flag five. Security flag five has a status of on which means that access is permitted. Similarly, a command having a code of nine would not have access to the file since security flag twelve is in an off state. Both the security flag and the Command Access Table values are loaded with default values through the data stored in the CDF when a session begins (a session is initiated when the application program begins to access the data stored in the files and a session is subsequently terminated by the application program). However, during the execution of the application program, the state of the security flags may change which in turn would change the access status of certain commands which refer to those security flags. The status of the security flags is controlled by the application program through the execution of key commands. As will be explained subsequently, the security flag referred to in a key record will change status upon valid execution and validation of the key in the record using the verified key command.

As stated previously, the data is organized into files within the memory. The access method is by logical record numbers, not physical addresses. Multiple files may coexist on the card, completely independent of each other, with separate security levels for each file. Since the data is accessed using logical file names and not physical memory addresses, multiple independent applications may reside on a card with only the file name required for access to data relevant to a particular file. Since the memory management routine controls the physical addressing of files, files may be created or erased at any time, with the card allocating the memory space as required. Cards with different memory sizes may be used interchangeably and quick update of cards in use in the field may be performed, since reprogramming the application software to accommodate specific physical addresses is no longer necessary. In addition, revision or update of the cards in use is quickly and easily performed by simple noninvasive input of software through the keyboard or input ports.

The memory management routine permits various objects in the card to be protected from unauthorized or improper access by an application program. Each file may be limited to a subset of the full set of commands that the card implements. This information is stored in the CDF file and is loaded into security flag table and command access table at the beginning of a session. This subset may be modified during execution of an application program by executing certain key commands and submitting the proper access codes. At the same time, some commands may be permanently denied use. For example, an application program could create a file, write data to the file, and then permanently disable the write function as it applies to that file so that data could subsequently be read from the file, but never written.

Referring to FIG. 10C, the data space for a file 530 is organized in memory 533 in a contiguous manner, with a header 535 identifying the file and containing information about the file, followed by the data 537 itself. The next file's (if any) header 539 follows the last byte of the previous file.

Referring to FIG. 10D, the file header contains information about the file required by the memory management routine of the operating system. Typically the header includes the name of the file 541, type of file 543, the file length 545, read and write security flags 547, 549, extended size pointer 550, a write pointer 552 and file switches 551 which indicate certain conditions of the file.

The name and type of file are used to identify the file when a FIND command (find the file) has been executed. In the preferred embodiment of the invention, no directory is kept for this system. Rather, the file is searched by starting with the first file, referred to as the Common Data File (CDF), and sequentially comparing the requested file name and file type with each file header. If a match is made the particular file is selected. If no match is made, the size of the header and the length of the file as specified by file length 545 are added together by the memory management routine to point to the next file header. In an alternative embodiment, a directory of all files could be kept and the file would be located by searching the directory.

The file can have two different security flags 547, 549 - one flag 547 for read only access and one flag 549 for read/write access. The appropriate flag is selected based on whether the file is opened for read only or read/write access.

The extended size pointer 549 points to an additional area of memory that may be assigned to this file. This allows a one time addition to the file of a fixed length of memory, if the space is available. This would be used to add space when the initial file space is full, and a new record needs to be added.

A write pointer 552 is maintained, indicating where the next available data memory is located. This value is modified by the memory management system when a record is written, a record deleted, or the file extended.

Each file can store data and key records. The records may be of varying length, and there may be multiple records with the same logical ID. There is no limit on the number of records, up to the limit of card memory. Data records are comprised of user data, and may be in the clear or encrypted with one of many algorithms that may be available in the card. Key records are used to secure the system and contain verification keys or pins as well as keys used for encryption.

Each record contains a header that describes the record type, size, security level and switches that control whether the record may be erased, or replaced. Referring to FIG. 10E, the header for a data record comprises command control switches 553 which enable/disable the function to delete or replace data record, a record ID 554, the length of the data contained in the record 555, the read only and read/write pointers 557, 559 which control the read/write access of the record. The data 561 is located immediately following the header.

The header for a key record is similarly illustrated in FIG. 10F. The key header contains command control switches 563 which enable/disable the functions to use the key, to replace the key or to delete the key record. The key header also contains a key record I.D. 565, the length of the key 567, a key record access pointer 569 which enables/disables access to the record, security flag 571 which indicates which security flag in the security flag table to turn on once the key is verified. The limit count for maximum number of invalid key tries allowed 573, algorithm selection value 575 which selects the particular algorithm to use in verifying the key and key verification field 579 which is the value returned to the application program when a key is verified. The key itself 581 is located immediately following the header. Using the key records, data may be input into the card in the clear, encrypted by the card, using a key internally stored in a key record, and then read from the card for transmission to other systems. This provides a much greater level of security, since the key is always stored on the card and is not transferred to the card during the verification process.

Using the key records and key commands, multiple levels of security may be provided. For example, an application program may require three different key commands to be successfully executed before access to a data record is permitted. The key records may be structured such that the first key record used to verify access may, upon verification, turn on the security flag referred to by the second key record (via the access pointer 569). If the security flag referred to by the access pointer is not on, access to the key record itself is not permitted and verification would not be given. Similarly the second key record would turn on the security flag referred to by the access pointer of the third key record upon verification and the third key record would not be accessible unless the second key was verified.

In addition to having the capability to "cascade" key records for verification, each key record may use different keys as well as different encryption algorithms thereby enhancing further the security of the card.

A system default file, the Common Data File (CDF) is located in the memory. This file typically contains security information, discussed below, as well as non-protected informational data such as the user's name and address. When a session is initiated, the CDF is first selected by the memory management routine. Using the data stored in the CDF file, the security flags and the command access table are set. Although the CDF file is automatically selected when a session is initiated, the security flags and command access table are not reset if the file types of the previously accessed file and the current file are the same.

The commands used to access files and control security fields comprise the following fields: the command class, instruction code, option bytes and a length byte. The command class is used by the card to check whether the card supports the function. The instruction code tells the card which command to process. The option bytes may be used differently by various commands to pass record id's, record flags, etc. The length byte tells the card how much (if any) additional data is being sent to the card to complete the command.

As shown in FIG. 10G, once the command bytes are received, the command class is checked to see if the card supports this command. If the class is not supported, an error is returned.

The instruction code is then checked for validity. If the instruction does not exist, an error is returned; otherwise, the instruction code is parsed to recover a command access pointer stored within it. The pointer is used to index into the current table of security flags. If the pointer is set to allow any access, or the appropriate security flag is set, the command is allowed to proceed. Otherwise an error is returned to the card.

Alternatively, each command or group of commands may call individual routines to test the security for the commands, and then proceed with the command processing itself.

The selected command routine is then accessed by selecting the corresponding command routine in the OS.

In a preferred embodiment the following commands may be used to secure and access the files stored in memory

INQUIRY	- Get card type
CREATE	- Create file
FIND	- Find a named file
WRITE	- Write a data record
WRITENC	- Write an encrypted data record
WRITEK	- Write key record
READ DATA	- Read a data record
READ NEXT	- Read next data record
REPLACE	- Replace data portion of data record
DELETE	- Delete data record from file
DELETEK	- Delete key record
VERIFYK	- Verify key record
LOCK	- Lock key record
UNLOCK	- Unlock key record
ENCODE	- Encode data record
ERASE	- Erase file
EXTEND	- Extend file

BROWSE	- List files on card
RAND	- Generate random number
AUTHC	- Authenticate reader
AUTHI	- Authenticate card

In addition, facilities exist to add more commands in the future, and to provide a means to add functions to existing cards, by using the memory to store user definable subroutines, (e.g. encryption algorithms, etc.).

The INQUIRY command (FIG. 10H) returns information about the card. The data returned includes the revision level of the operating system, the card type, and the record ID of the command security table. These values are constants for each particular card.

The CREATE command (FIG. 10I) creates a new file on the card. This command only operates when the current file is the CDF file. A check is first made for the existence of a file with the same name on the card. If the file already exists, an error is generated. The memory management routine then checks to see if the requested memory for the file, along with space for the header, exists in the card. If sufficient memory is not available, the an error, "Insufficient Memory", is returned.

If sufficient memory exists, a File Header is created on the card at the next available memory location. Security information included in the data portion of the command will be copied into the header, along with the size of the file, its name and type, flags, and the initial value of the write pointer for the files.

Upon creation of the file, a code is sent to the application program indicating that the file has been created.

The FIND command (FIG. 10J) is used to locate and open a file for subsequent access. The data portion of the command includes the name and type of file to be located. Starting with the CDF file, the file headers are scanned until the file is located or the end of memory is reached.

For each file, the file name and type in the header is compared with the requested name and type. If the lengths of the names are different, the file is not correct, and an error is generated.

The appropriate security flag is then checked for access in the READ or READ/WRITE mode. If the security flag is not set for access, the ACCESS ERROR code is returned.

If all of the preceding steps are completed successfully, the file is set as the current file and a final check is made to see if the last file accessed was referred to by the same logical name, but may have been a different file type. If the file names are identical, the current security setup is maintained (command access table and initial security flags). Otherwise a new table is loaded for the file.

The WRITE and WRITEK(EY) commands (FIG. 10K) differ only in the information that is written. The WRITE command is used to write data records while the WRITEK command writes key records to a file.

Once the record type is established, a flag is set indicating that no duplicate records are allowed if it is a key record, and the index pointer is set to the start of file. This allows the test for duplicate records to be made, although each file may optionally allow duplicate records to exist by enabling the particular switch in the file header.

The access mode is checked, and if it is READ ONLY, an ACCESS ERROR code is returned. If the file is READ/WRITE, the file is checked to see if duplicate records are allowed. If duplicates are allowed, and the record type is data, execution bypasses to a memory space check. Otherwise a SEARCH is made of the file for a duplicate record ID. If a duplicate exists, a "DUPLICATES NOT ALLOWED" error code is returned.

The memory space test is performed to check if the new record will fit in the file space remaining. If the record is too large, an "INSUFFICIENT MEMORY" error is returned.

If the record is a data record, the data pointer is set to the next available location; otherwise the key pointer is set, since the record would be a key type. The information is written to the memory, and the appropriate record header is created. A code indicating successful completion of the command is then returned.

The WRITENC command (FIG. 10L) will write a data message, which may already be encoded, into the card. Typically such data would be encoded by the encode command. The file mode must be read/write, or an error code is returned. A check is then made for the existence of the key used to encode the data which is a parameter specified in connection with the command. If the key does not exist, an error code is returned. A test is made for the existence of the algorithm used to encode the data and if an invalid algorithm is selected, an error is returned. No processing of the key security or limit checks are made, since this command does not actually access the key.

It is then determined if the file permits duplicate records. If duplicate records are not allowed, a search is made in the file for a duplicate record ID. If a duplicate exists, an error code is returned. If duplicate records are allowed in the file, the search is not executed. A memory space test is then performed to check whether the new record will fit in the file space remaining. If the record is too large, an error is returned.

If sufficient memory space exists, a data pointer maintained by the memory management routine is set to the new record location, the information is written into the memory, and the appropriate record header is created. A code is then returned indicating successful completion of the command.

The READ command (FIG. 10M) has several options. The record may be searched from the beginning of file, or from the current record. Once the starting position is located, the SEARCH routine is called. This routine will search for either a data record or key record. For the present example, the search is for a data record. If a matching record is not found, an error "RECORD NOT FOUND" is returned.

If the record is located, the appropriate security flag is checked. If not set, an access error is returned. If the security check is successful, a pointer to the record is returned, along with a code indicating a successful security check.

The REPLACE command (FIG. 10N) is used to replace the data field of a record. Numerous checks are performed before replacing the record. The record to replace is previously located using the read or read next command, so as to set the data pointer to the correct location. If the record ID of the replace command differs from that of the current record, an invalid record ID error is returned. A check is then performed to determine the record type. The record to be replaced must be a data record, or an error is returned indicating a record mismatch. Next the length of the two records are compared. Both records must be identical in length, or the record mismatch error is returned. The file mode must be set to a read/write mode or an access error will be returned. In addition, each record contains a flag indicating whether or not it may be replaced. If this flag is set, the record is not replaceable, and a replace error code is returned.

If all of the checks pass, the data in the record is replaced with the new data, and a success code is returned.

The DELETE and DELETE KEY commands (FIG. 10O) operate identically, with two exceptions :

A) The DELETE command works only on data records. The DELETE KEY command works only on key records; and

B) The DELETE and DELETE KEY commands do not use the same security flag pointers.

The current data record pointer or current key record pointer (maintained by the memory management routine) must be set to point to the record to delete. In order to delete a record, the file mode must be set to a read/write mode or an access error code is returned. Furthermore, each record contains a switch indicating whether it can be deleted. If the switch is set, the record cannot be deleted, and a command fail error is returned.

The record type is checked, and if it is a data record, the data record pointer is used; for a key record the key record pointer is used. Finally, the record identifier of the record is next compared to the ID submitted with the delete command and, if the IDs do not match, an error indicating an invalid record ID is returned.

If all the checks pass, the record is deleted from the file. In this implementation, the record is deleted by moving the data at memory addresses greater than that of the deleted record into the space vacated by the deleted record. This keeps the data area contiguous and prevents fragmentation of the memory, thus reducing the file management task.

Alternatively, a table could be maintained indicating which records are current, and which deleted, so that a record may be deleted simply by changing the record status in the table.

To set security flags to allow access to various objects, the VERIFY KEY and AUTHENTICATE commands are provided.

The VERIFY KEY command (FIG. 10P) compares a key stored in a file with a key submitted from an external source. This key may be encrypted by one of several optional algorithms in the card to produce a result which is subsequently compared with the submitted key. Because the encryption and the key comparison is performed on the card, a high degree of security is maintained in the card.

The key record is searched for from the beginning of the file using the SEARCH routine (FIG. 10Z). If the key record is not located, an error is returned.

Each key record contains a pointer indicating which security flag (if any) must be previously set to access the current key in order to perform verification. If this key is not set, an access error code is returned.

If access to the key is permitted, the length of the stored and submitted keys are compared. If the lengths of the keys are different, an error is returned. The keys themselves are then compared, using one of several options determined by the parameters associated with the command. A byte to byte compare may be done, or the submitted key may be passed to an encryption process, using another key in the file which is selected according to a value stored in the first key.

If the comparison is successful, a code indicating successful completion is returned, along with the value from the key verified field of the key record verifying that a valid key was used and verification is completed.

Each key in the file may optionally have a limit to the number of verification attempts. This limits the number of unsuccessful attempts to verify the key in the file to prevent the breaking of the key or algorithm. When the number of attempts is exceeded, the key record is locked. Subsequent submission of the correct key with the verify key command will not set the security flag. The lock must be removed with the UNLOCK command described below.

After each failed verification attempt, the limit is checked. If the limit value is equal to zero, an error is returned. If the limit value is greater than zero the limit value is decremented, and if after the limit value was decremented, the limit value is equal to one an error code is returned indicating that one more try remains. If the limit was decremented to zero, the record is locked and an error value is returned indicating that the key record is locked.

The LOCK command (FIG. 10Q) sets a lock bit in a specified key record. This prevents a key from being verified, and the security bit from being set, even if the correct key is presented.

In order to execute the LOCK command, the file must be in a read/write mode, or an access error code is generated. The key record to be locked must be first located using the verify key command. To locate the key record using the verify key command, the key submitted need not be the correct key in order to set the pointer. Once the key record is located, the record ID of the current key is checked against the record ID provided by the LOCK command. If the ID's do not match, an error code is returned.

Once above tests are performed without an error, the key record is locked, and can only be unlocked with the unlock command.

Operation of the UNLOCK command is identical to the LOCK command, except the UNLOCK command clears the flag to unlock the key record instead of setting it.

The ENCODE command (FIG. 10R) is used to encrypt or decrypt data. The operation uses a key that is already stored in a file in the card, providing an extremely secure method of encrypting data.

The key record specified in the command is searched for in the current file. The key record is located by the same routine used in the VERIFY KEY command, and must pass the same security flag tests described with respect to the VERIFY KEY command. If the record is not found or any of the security tests fail, an error code is returned. If the record is found, the algorithm requested to be used in the command is located. One of many algorithms can be used, including algorithms initially loaded into the card as well as those subsequently loaded for a particular application or added security. If the algorithm selected exists and the key is verified, the data is encrypted.

If the key is not verified, the key limit check process as described above is performed

The ERASE FILE command (FIG. 10S) deletes all the data in the specified file and removes the file from the data space. The specified file to be erased must be the current file. If the file named in the command is different from the current file, an error is returned. Before the file is deleted, the file mode is checked, and if it is read only, an access error code is returned. Once the file is deleted, the memory space is reusable to create new files.

The EXTEND command (FIG. 10T) is used to add more memory space to a file when it approaches or reaches its limit. This operation may be performed once for each file. The amount that may be added is preferably a value preset when the file is created. If no memory for extension is specified, the file may not be extended. If the file is specified to be extended, a check is made to determine if there is sufficient free memory in the data space for the file extension. The free memory is compared to the requested amount stored in the file header. If the requested amount exceeds the amount of free space remaining, an error is returned. Next, the file name and file type specified with the command must match the file name and file type of the current file. A check is made for this condition, and if the file names or types differ, an error is returned.

Once all the checks are performed, the requested memory is assigned to the file. Pointers are added to the headers of the original file and the extended memory file to link the new space to the old space. In addition, the file extension value in the original file is set to zero to prevent any further extensions to be made.

The BROWSE command (FIG. 10U) lists the file name and file type of the file in the card currently pointed to by the browse pointer maintained by the memory management function. Each file may be flagged to prevent it from being listed by this command. When the BROWSE command is initiated, the current file must be the CDF, or an error is generated. Next a test is made to see if this is the first time the command has been executed for the current session. If it is the first time, the browse pointer is set to the CDF file. Otherwise the pointer is not reset.

The file header is then read to determine if one of the switches is set which permits execution of the browse command on that file. If the switch is set, the name of the file is not returned, but the browse pointer is advanced to the next file. If the switch is not set, the file name and file type are returned, and the browse pointer is set to the next file. Another browse command may be executed to browse the next file.

When the last file has been displayed, the browse pointer is again set to the CDF file.

The RAND command (FIG. 10V) is used to generate a random number, used during the execution of the AUTHC and AUTHI commands. The random number may optionally be generated using several algorithms. No errors are generated during the execution of this command.

The AUTHC command (FIGS. 10W and 10X) is used to verify that the card is a valid card. A RAND command must have been run previous to this command, or an invalid parameter error is returned. The key record specified with the AUTHC command, is then located. If the key does not exist, an error code is returned. Once the key record is found, the algorithm selected is verified. If the algorithm is verified, the random number and the submitted specified with the command are processed by the algorithm, and the result is compared with the key stored in key record. If the keys match correctly, the security flag specified by the key is set.

If the keys do not match, or the algorithm does not exist, the key limit check process described previously is performed.

The AUTHI command (FIG. 10Y) validates the card with respect to a particular application. The random number generated and a key are used to generate an encrypted result. This is returned to the application, and the application determines if the encrypted result is correct.

The location of the key is specified with the command. If no key exists, an error is returned. Next the algorithm is checked for validity. If the algorithm is valid, the random number and the key are processed by the algorithm, and the result returned to the application.

Referring to FIG. 10AA, the loading of application program code into memory is illustrated. Typically, the code will be input into the system from an external device through one of the electrical or optical input/output ports. At step 601 data specifying the length of the application, a relocation table of absolute address references which must be recalculated after they are placed in memory, the length of the relocation table and the application code itself are input into the system. At step 603 the system checks if enough memory remains unused to store the application program code. If there is not enough memory available to accept the application, a memory error is noted at step 605 and the routine is exited without loading the application. If there is sufficient memory to load the application, the relocation table is temporarily loaded into memory at step 609. At step 611 the starting memory address of the application is determined and the application code is loaded at step 612 into memory starting at that memory address.

Steps 613, 615, 617, 619 and 621 describe the process of relocating address references based on the starting address in memory where the application code is stored and the relative addresses within the application code itself. At step 613 the absolute address for each relative address is determined by adding the starting address to each relative address stored in the relocation table. At step 615 the absolute address is verified to be within the permissible range of memory, that is, the address is checked to insure that it references code that is within the loaded application or globally defined routines (e.g. service routines). This prevents the application from directly addressing other applications or non-existent memory. If the absolute address is not within the permissible range, a memory range error is noted at step 617 and the routine is exited without completing the application loading process. If the absolute address is within the permissible range, the absolute address replaces the relative address in the application code. Steps 613, 615 and 617 are repeated, as necessary, for each element in the relocation table until it is determined, at step 621, that the end of the relocation table has been reached. After all the addresses have been relocated and verified to be within the permissible memory range, the relocation table is erased, at step 623 the starting address of the application is added to a code list containing the addresses of all the application programs, a valid load is noted at step 625, and the routine is exited.

The application service routine determines what application routine, if any, is requested and switches system control to the requested application program. Referring to FIG. 11, the code entered either through the keyboard or through input/output ports is read at step 650 and compared at step 655 to a code list of application programs and their starting addresses in memory. If at step 660 the code entered matches an element on the code list, the corresponding application program is initiated at step 665 and the application service routine is exited. If the code does not match any of the elements in the code list at step 660, an error message is displayed at step 675 if the code originated from the keyboard or is transmitted back to the input/output port if the code originated from an external device connected to the input/output port. The routine is then exited.

Another service routine is the system clear/restart service routine. Referring to FIG. 12, at step 685 the volatile registers are reset. A test is then made at step 690 to determine if a self test is required. If so, the self test is executed at step 695. At step 700, the initialization information required to restart the system is read from non-volatile memory. The cardholder is then prompted to set the date and time at step 705 and the routine is exited.

The operating system permits the ITC to be programmed for a variety of applications. The applications are realized through the application programs stored in the non-volatile memory of the ITC. The application routine programs can be changed, removed or deleted according to the ITC cardholder's needs by the issuer of the card. Examples of the application programs are a cardholder notepad application, set time application, set date application, change pin application, credit/purchase application, transportation application and calculator emulator application program.

The cardholder is led through the proper operation of the ITC by a series of instructions or menus set forth on alphanumeric display 25. This makes the ITC quite easy to use and is of great benefit to the new, unskilled or infrequent user of the ITC who is unfamiliar with the ITC operating procedure.

The cardholder operates the ITC through a plurality of menus. The menus present to the cardholder the options that are available to the cardholder at that specific point in the program. The options presented may be sub-menus of the menu currently being displayed or the options presented may be a selection of variables to be used during the execution of an application program. For example, the cardholder may select a menu item identified by "CREDIT" to activate a credit function. A sub-menu of credit options, such as making a credit transaction or seeing the cardholder's credit balance, is then presented to the cardholder. Once the cardholder selects an application the cardholder is prompted to enter variables used in the application. These variables may be presented in a menu giving the cardholder the ability to select a variable value from a menu. For example, if the cardholder wished to convert currency, the cardholder would indicate what country to convert the currency to. This may be accomplished by selecting a country from a menu list of countries.

As will be illustrated in the following description of application programs in conjunction with FIGS. 13-17, the cardholder uses the "YES", "NO", "NEXT" and "BACK" keys, referred to as the application control keys, to control the execution of the application programs. The NEXT and BACK keys are used to scroll or view different options available at that time to the cardholder and the YES and NO keys are used to enter and exit different menus, applications or portions of applications.

For convenience, the displays generated by the program are depicted in block capital letters enclosed in a box. Operations performed by the user or by the microprocessor are set forth using lower case letters.

When the ITC is operating, the system is normally in an idle state. While in the idle state, the current date and time are displayed. The cardholder may then scroll display 25 (FIG. 1) through the basic functions or applications contained in the ITC, by pressing the NEXT key to go to the next function or the BACK key to return to the previous function. If the cardholder wishes to select one of these functions for data input or output, he scrolls through the functions until the desired function is displayed and then depresses the YES key to activate that function.

This operation of the ITC is illustrated in FIG. 13 which depicts in boxes 752, 754, 756, 758 and 760 five illustrative displays comprising a menu that is generated at display 25 by one embodiment of an ITC of the present invention. In the idle state TIME and DATE are displayed as shown in box 752. By use of the BACK and NEXT keys display 25 may be changed successively to those shown in boxes 752-760. The displays shown in boxes 756-760 are prompts for standard functions which typically are found in any ITC. These functions are described in detail in conjunction with FIGS. 15A-C. Additional functions or applications may be supported by the ITC system according to cardholder requirements. These applications may be accessed using the NEXT/BACK/YES keys or by depressing a function key programmed to activate that particular application. One example of an additional application program that can be implemented in the ITC is the notebook application depicted in FIG. 13. To activate the notebook application, the cardholder scrolls through the application program options until "SEE MY NOTES?" is displayed as illustrated in box 754. The YES key is then depressed to activate the application.

The notebook application is described in more detail in FIG. 14. After selecting this application, the cardholder is prompted by display 25 to enter in his PIN by the prompt depicted in box 762. The entered PIN is then tested at step 764 against a PIN that is stored in the ITC in conjunction with this application program. This prevents unauthorized access to this application program because the cardholder should be the only person who knows his PIN. If an incorrect PIN is entered, system control exits the application program and returns to the point in the program that generates the display shown in box 754.

If the cardholder enters the correct PIN, the system displays on display 25 the first note stored in memory as shown in box 766, indicating to the cardholder that he now can access his notes. Using the NEXT and BACK keys the cardholder can scroll through the different notes as illustrated by boxes 766, 768 and 770 until he finds the particular note he wants to select. The cardholder selects a note to work on (the "current note") by depressing the YES key. The user is then prompted by display 25 to select a particular note function in accordance with the displayed prompts shown in boxes 772, 774, 776 and 778 by scrolling among these functions using the NEXT/BACK keys and selecting the function using the YES key.

To change a note the cardholder scrolls through the options presented until "CHANGE NOTE?" reflected by block 772 is displayed. Upon depression of the YES key, the current note is displayed at block 780 with a cursor highlighting the first letter of the note. Using the NEXT/BACK keys to move the cursor, the displayed note may be changed at step 782 by positioning the cursor on the letter to be changed and altering the letter by depressing a new key on the keypad. The cardholder may save the changes by depressing the YES key whereupon the prompt "NOTE SAVED" shown in block 784 is displayed to tell the cardholder that the note was saved. If the cardholder wishes not to save the changes, the NO key is depressed at which time the message "NO CHANGE" is displayed as shown in block 786. The system then displays again, as indicated by block 788, the current note. To exit the function the NO key is depressed.

To add a note, the cardholder scrolls through the menu presented by display 25 until "ADD NOTE" as shown at block 774 is displayed. He then depresses the YES key to select the function. The current note is then displayed as indicated by block 790. The new note will be added after the current note. If the cardholder depresses the NO key, the display 25 indicates "NO NOTE ADDED" as shown at block 792 and returns at step 794 to display the first note as indicated at block 766. If the cardholder depresses the YES key, a prompt "ENTER NOTE" is displayed, as indicated by block 796 and a blank screen with a cursor is then presented on display 25 to give the cardholder the opportunity to add a note as reflected by blocks 798 and 800. The NEXT and BACK keys enable the cardholder to move the cursor through the text of the note. If no information is added and the YES or NO key is depressed, at step 804 the system returns to block 766 and displays on display 25 the first note stored in memory. If information is entered and the NO key is depressed, the message "NO NOTE ADDED" is displayed as indicated by block 806, and the system returns and displays the first note at block 766. To save the note added the cardholder depresses the YES key. The message "SAVED NOTE" is displayed as shown at block 810 followed by the display of the new note as indicated at block 812.

To erase a note, the cardholder scrolls through display 25 until "ERASE NOTE" as depicted in block 776 is displayed and depresses the YES key. The current note is then displayed. If the cardholder depresses the YES key again, the message "ERASED NOTE" is displayed as shown at block 816 and the next display note in the series of notes represented by blocks 766, 768 and 770 is displayed at block 818. If the cardholder does not wish to delete the note displayed, the NO key is depressed. The message "NO CHANGE" is displayed at block 820, indicating to the cardholder that the note was not erased. The current note is then displayed as indicated in block 822.

If the cardholder wishes to see the current note he scrolls through display 25 until the prompt "SEE NOTE" is displayed as shown in box 778. The note is displayed, as depicted in block 824, upon depression of the YES key.

Once operation of the functions are complete, control returns back to one of the notes represented by boxes 766, 768 and 770 enabling the cardholder to select another notepad function. To exit the notepad application, the cardholder depresses the NO key when the contents of one of boxes 766, 768 or 770 is on display. This returns system control to the display shown in box 754 at which point the user can select another application.

FIGS. 15A, 15B and 15C illustrate the set time, set date and change PIN applications. If the set time function is selected, the hour is first displayed at step 850 and the cardholder is given the opportunity, using the NEXT and BACK keys, to modify the displayed hour. The YES key is then depressed, storing the hour value displayed and displaying the minutes at step 852. The cardholder may then modify the minutes using the NEXT and BACK keys to increment and decrement the numbers. To enter the change, the cardholder depresses the YES key. If the NO key is depressed, the application is exited without making the change.

Similarly, referring to FIG. 15B if the cardholder selects the set date function, the month is first displayed at step 854, followed by the date at step 856 and the year at step 858. The cardholder may modify each using the NEXT/BACK and YES keys.

Referring to FIG. 15C, if the cardholder selects the change PIN function, the cardholder is prompted to enter the current PIN by the display depicted in box 860. The entered PIN is then tested at step 862 against the PIN already stored in the ITC. If the PIN is incorrect, i.e. it does not match the PIN stored in the system, the system gives the cardholder another opportunity to enter the PIN by generating the display shown. The PIN that is entered is tested at step 866. If the correct PIN is entered, the cardholder is prompted for the new PIN he wishes to enter by displays 868 and 870. After the new PIN is entered at step 872, the cardholder is prompted to reenter the new PIN by displays 874, 876. At step 878, the two new PINS are compared. If the PIN entered in response to the display at box 876 does not match the PIN entered in response to the display at box 870, an error message is displayed as shown at box 880 and the cardholder is given another opportunity to enter the correct PIN by returning the program to the point at which the display shown in box 870 is generated. If the cardholder fails to enter in the correct PIN after a predetermined number of tries, the new PIN does not replace the current PIN and the system returns to the point that generates the display of box 860 where the cardholder has to again enter the current PIN. If, at step 878, the cardholder re-enters the new PIN correctly, the current PIN is replaced with the new PIN and the cardholder is informed of this by the display at box 884. The application is then exited.

Numerous other application programs may also be used in the ITC of the present invention. These programs may be stored in addition to or in place of the notebook application depicted in FIG. 14. Each program is accessed by scrolling the program prompts on display 25 by means of the

NEXT and BACK keys and selecting the desired program by means of the YES key. Alternatively, at least some programs may be assigned a dedicated keyboard key for immediate access to that program.

FIG. 16 illustrates a credit/purchase application program. The cardholder activates this application program when performing credit transactions such as the purchase of goods or services. The program checks the cardholder's available credit balance and, if sufficient credit is available, generates an independent approval code authorizing the purchase. The approval code generated by the ITC eliminates the need for an approval code generated by an external credit approval service, for example, a credit card service bureau used by most merchants. The approval code is a unique encrypted code for the particular transaction generated from the amount, account number, account type and time and date of the transaction.

At step 752 (FIG. 13) the system is in an idle state as reflected by the display of the date and time. When the credit function key is depressed, the program is activated, indicated by the display of "CREDIT", as depicted by box 900. The cardholder is then prompted for his PIN by the display shown in box 902 and, following entry of the PIN, its validity is tested at box 903. If the correct PIN is not entered, an error message "PIN INCORRECT" is displayed to the user as shown at block 904 and the system exits the application at block 905 and returns to the idle state. If the correct PIN is entered, display 25 presents to the cardholder a plurality of options in the form of the prompts "MAKE A PURCHASE", "SEE AMOUNT AVAILABLE", "SEE PURCHASES", "ADD TO ACCOUNT" and "SELECT CURRENCY" depicted in blocks 906, 908, 910, 912 and 914. The cardholder scrolls through these prompts using the NEXT and BACK keys.

If the cardholder wishes to make a purchase using the credit available, the cardholder depresses the YES key when the prompt displayed is that of box 906. This activates the function. The cardholder is prompted by a display depicted in block 915 to enter in and verify the amount of the purchase. If the amount entered is incorrect, the cardholder depresses the NO key at step 915; and he is given another opportunity in response to the display "REENTER AMOUNT" depicted in box 914 to enter the correct amount at the prompt depicted at box 915. The cardholder verifies the amount entered by depressing the YES key. After the amount is entered and verified by the cardholder, the amount of the requested purchase is flashed on and off in the display 25, as reflected by box 916 and is compared at step 917 to the cardholder's credit balance stored in the card. If there is insufficient credit, the message "NO CREDIT" is displayed, as depicted by box 918 and the system returns at step 919 to the idle state in which the time and date are displayed as in block 752. If there is sufficient credit to complete the transaction, a unique approval code is generated and displayed indicated by box 920 along with the amount of purchase. The approval code may be noted by the merchant on the credit slip for securing the transaction.

The cardholder has the option at step 918, to exit the credit/purchase application program or perform another function with the application program. If the cardholder depresses NO, the application program is exited and the system returns at step 921 to the idle state in which the time and date are displayed as shown in box 752. If the cardholder depresses YES indicating he wishes to make another purchase, the system returns at step 922 to the point in the program at which the display of box 906 is generated.

The cardholder may view the credit balance available by scrolling the display to that of box 908 and selecting the function by depressing the YES key. The credit balance is then displayed at step 922. Similarly the cardholder may elect to see the purchases or transactions by scrolling the display to that of box 910 and selecting the function. The cardholder may then scroll through the list of purchases, displayed by amount and date, using the NEXT and BACK keys.

The cardholder may add to his credit balance by scrolling the display to that of box 912 and selecting the function. After the function is selected, the cardholder is prompted to enter and verify the correct bank code by the displays depicted in boxes 926 and 928. The bank code is typically provided by the bank to the cardholder and verifies the deposit made by the cardholder. Encoded in the bank code is the cardholder's account number, date of deposit and amount of deposit. If the cardholder makes an error in entering the bank code, he depresses the NO key which prompts him to reenter the bank code through the displays depicted in boxes 926 and 928. If the bank code is correct, the cardholder depresses the YES key. The bank code is then tested and verified at box 930. If the bank code is not verified by the ITC, the function is exited and the system returns to the point evidenced by the display depicted in box 911. If the deposit is verified, at step 932, the balance is updated to reflect the deposit, the new balance is displayed and system control returns to step 911.

The cardholder may also convert his credit into different currency simply by scrolling the display to that of box 914 and selecting the function by depressing the YES key. The cardholder is then presented at step 934 with a list of countries and selects, using the NEXT, BACK and YES keys, the country his wishes to convert his balance to. The cardholder is notified that his currency has been changed by the display in box 936 and his currency credit balance is displayed at step 938 in the selected currency. If the cardholder depresses the NO key at step 934, the function is exited and box 913 is displayed.

FIG. 17 illustrates the use of the ITC for transportation, for example, to pay for train tickets. When the display depicts time and date as depicted in box 852, the function key assigned to this application is depressed. A message indicating that this function has been selected is displayed as shown in box 950 and the cardholder is prompted to enter his PIN by the display depicted in box 952. The PIN is tested at step 953. If the PIN entered is incorrect, an error message "PIN INCORRECT" is displayed as shown at block 954, and the application is exited at block 955 to return to the idle state reflected by the display of the time and date shown in block 752. If the PIN entered is correct, the cardholder is presented a list of operations that can be performed through display of the prompts "MAKE A PURCHASE?", "SEE AMOUNT AVAILABLE?", "SEE PURCHASES?", and "ADD TO ACCOUNT?" as shown in boxes 956, 958, 960 and 962. The cardholder scrolls through the operations using the NEXT and BACK keys until the desired function is displayed and depresses the YES key which selects and activates the function.

To make a ticket purchase at the train station or on the train, the cardholder scrolls through the functions until "MAKE A PURCHASE" is displayed on display 25, as shown in block 956 and depresses the YES key to select the function. The ticket purchase function is activated and the user is prompted to select a one way or round trip ticket by the information shown in blocks 964, 966 that is displayed by display 25. Again, the NEXT, BACK and YES keys are used to scroll the display and make a selection. The cardholder is then prompted by the display at box 968 to indicate whether the senior citizen fare applies and to enter the amount of the fare in response to the prompt shown in box 970. If a mistake is made in entering the amount the cardholder can depress NO; and block 972 is displayed on display 25 indicating that the amount is to be reentered. The cardholder is then prompted to reenter at block 970 and verify the correct amount. When, at block 970, the cardholder verifies the correct amount by depressing the YES key, the amount is flashed on the display, as indicated by box 974 and the system tests at block 976 whether there is sufficient funds to cover the ticket purchase. If there is insufficient funds the message depicted in box 977 "INSUFFICIENT FUNDS" is displayed and in step 978 the system returns to the idle state reflected by the display of the time and date. If the cardholder has sufficient funds in his transportation account to complete the ticket purchase, the amount of the purchase is debited from the account and the approval code is generated and displayed with the amount at step 979. The cardholder is then given the opportunity depicted at blocks 980 and 982 to make another ticket purchase by depressing the YES key or exit the purchase function and return to the idle state by depressing the NO key.

To check the balance remaining in the cardholder's transportation account, the cardholder scrolls through the display 25 until "SEE AMOUNT AVAILABLE?" is displayed, as shown in block 958, and selects the function by depressing the YES key. The account balance is then displayed as indicated by block 984.

To view the date and amount of prior ticket purchases, the cardholder scrolls until "SEE PURCHASES" is displayed, as depicted in block 960, and selects the function. At step 986 the amount and date of the earliest purchase is displayed. The cardholder may then scroll through the display of the list of ticket purchases using the NEXT and BACK keys.

The cardholder can add funds to increase his transportation account balance through the "ADD TO ACCOUNT" function. The cardholder deposits or transfers money to his transportation account at a financial institution or transportation ticket office. The financial institution or ticket office supplies the cardholder with a unique deposit code which includes information such as the cardholder's account number and the date and amount of deposit. The cardholder, upon receipt of the deposit code, initiates the function at the ITC by scrolling until the "ADD TO ACCOUNT" function is displayed on display 25 as shown in block 962. The cardholder is then prompted to enter the code into the system by the prompts shown in boxes 988 and 990.

If the cardholder realizes that an incorrect code was entered in response to the prompt displayed in box 990, he may correct the error by depressing the NO key. The code will be deleted and the prompts to enter the code, shown in blocks 988 and 990, will be again displayed giving the cardholder another opportunity to enter the code. To verify that the cardholder has entered the correct code he depresses the YES key. The system then tests at step 992 whether the code is valid. Once the correct code is entered and verified to be valid, the account balance is updated and the new balance is displayed as indicated by box 994. If the code entered is not valid, the function is exited; and the system exits the function reflected by the display depicted at box 962.

Referring to FIG. 18, at step 1000 the calculator emulator application may be entered from the idle state shown by block 752 by depressing any numeric key. The numeric key depressed becomes the first digit of the number to be used in the calculation and is displayed at step 1002. The cardholder then uses the keypad as a calculator, where the NEXT key represents "+", the BACK key represents "-", the NO key represents a decimal point and the YES key represents "=". The ITC will continue to emulate a calculator at step 1004 until a predetermined key which exits the function is depressed at step 1006. The system then exits the application function at step 1008 and returns to the system idle state reflected by block 752.

The ITC hardware of the present invention comprises a Central Processing Unit (CPU) having Random Access Memory (RAM), Read Only Memory (ROM), input/output ports, keyboard, display, and power supply contained in a housing of similar dimensions as conventional transaction cards.

The CPU, which controls the system and processes the information, comprises a microprocessor preferably a TI 7000 style microcomputer manufactured by Texas Instruments. The RAM contained within the CPU is used for the temporary storage of the program data.

The ROM stores the code for the application software as well as general system software and security information. The ROM is preferably an EEPROM, which permits the information to be erased and reprogrammed electrically and without having to physically access the ROM chip. Thus, for example, out-of-date application software may be removed or the cardholder's PIN may be changed. The information is arranged and structured such that certain information is not accessible without authorization. For example, the security algorithms stored in ROM may be accessible only by the manufacturer of the card who knows the security program access code. The PIN may be changed only by the cardholder since the cardholder is the only one permitted to access the area of ROM where the PIN is stored.

The keyboard provides the cardholder a means to use and communicate with the ITC, for example, to access information stored on the card, to store information in the card, and to use and interact with the card application programs. In addition to a set of alphanumeric keys, at least one programmable function key is provided. The functionality of the programmable key is adaptable to the application. Preferably, program control keys identified by "YES", "NO", "NEXT" and "BACK" are also provided.

The display provides a visual output of information such as message prompts, error messages, transaction information, and the like to lead the cardholder through the proper sequence of steps to operate the card for different applications. The display may have the capability to display one or more lines of information at one time. If a multiple line message is to be displayed, the message may be flashed one or two lines at a time in sequence, each line of the message being displayed sufficiently long for the ITC cardholder to read the message. Alternatively, each line of a multiple line message may be displayed until the ITC cardholder depresses a key to tell the system to display the next line of the message.

The user-friendliness of the card is greatly enhanced by a custom-made liquid crystal display, a preferred embodiment of which is depicted in FIG. 19. FIG. 19 illustrates a two line display, each line having the capability to display 10 characters plus a separately segmented question mark. Each character in the display consists of 14 segments providing for a clear and unambiguous display of the entire alphabet plus numerics. Further, a provision is made for a colon and a decimal point between the characters to allow for the display of time as well decimal units of currency.

The power supply provides the necessary energy to operate the ITC. Preferably the power supply is a battery. The card may also be provided with a solar based power supply and a means for connecting to an external power source to supplement or substitute for the battery. Alternatively the solar based power supply may act solely as a switch to supply the power necessary to turn the ITC on. Once the ITC is turned on, the battery or external power source supplies the power to operate the ITC.

More particularly, with reference to FIG. 20, the circuitry of the present invention comprises a CPU 1100, a latch 1105, an LCD display 1110, a display controller 1115 and an EEPROM 1120. CPU 1100 interfaces with the memory and I/O devices through an eight bit address/data bus 1130 (C0-7) and address bus 1135 (D0-7). Address/data bus 1130 is a bidirectional bus. The low order address byte is multiplexed with data input/output. Address bus 1135 provides the high order address byte.

Any address transmitted on address/data bus 1130 is first buffered (temporarily stored) in latch 1105. The latch 1105 acts as a buffer between the address/data bus 1130 and the memory device 1120.

A variety of control signals are used during the operation of this system. For example, CPU output port pin B_4 1140 generates the latch enable signal (LATCH ENABLE) which is connected to chip enable pin 1145 of latch 1105.

The R/W output port pin 1160 on the CPU controls whether the memory operation is a read or a write operation. The R/W control line is connected to the output enable pin OE 1170 and write enable pin WE on the ROM. Thus when a read operation is to be executed a logic value of "1" is output on the R/W control line for a read cycle. A write cycle is indicated by outputting a logic value of "0" on the R/W line.

I/O ports A and B on CPU 1100 provide the control· signals for the display controller 1115. They also provide the signals for scanning the x-lines of the keyboard and the ports for reading the signals on the y-lines of the keyboard.

Input/output contacts 30 of the ITC may be optical or electrical. Illustratively one such contact is shown in FIG. 20 as serial input/output port 1190. This port may be connected to any compatible serial device such as a printer or off-line storage device. In the alternative as described below in connection with FIGS. 22-24 an inductive input/output port may be included which emulates the magnetic strip presently found on transaction cards. By emulating a magnetic strip, information may be communicated between the ITC and present day transaction card equipment such as magnetic card readers used in ATMs and point of sale (POS) terminals. The inductive port can also be used in place of an electrical or optical port for connection to a device such as a magnetic card reader.

The segmented liquid crystal display (LCD) is controlled by the LCD controller/driver circuit comprising a 4 bit display mode register (DMO-3), a 320 bit (40x8) display data memory, a timing controller, multiplexers, LCD driver-voltage controller, and row and column drivers. Although a segmented LCD driver and display are described, a dot matrix, bit-mapped display and controller may be used.

The display mode register and display data memory are implemented in RAM on CPU 1100. The display mode register (DMR) is an 8 bit read/write register although only 4 bits are presently in use. The display mode register designates the basic LCD clock frequency that multiplexes the data to the display. The LCD clock frequency is a subdivision of the crystal input frequency and therefore the frame frequency (the frequency at which the display information is presented). The DMR also enables/disables a LCD bias voltage resistor ladder as well as row/column display outputs.

The data display memory is implemented in RAM. The display row/column location and corresponding segment identification are stored in a RAM buffer. This information is accessed by the display controller 1115 for enabling the display 1110. If the display is a dot matrix display, one bit in RAM directly maps to a pixel identified by a row/column location on the display. Therefore, a bit in RAM having a value of "1" turns on the corresponding pixel on the display.

In order to increase the chip function density (i.e. the functionality per unit area), a custom-designed chip incorporating RAM, ROM, clock and LCD controller may replace the individual components as illustrated in FIG. 21. The same control and addressing mechanism as described above would be utilized, however an internal address/data bus would be provided for addressing the RAM and ROM implemented on the chip.

It may be desirable to communicate information stored or calculated in the ITC card to a terminal of a transaction card system. For example, if PIN verification is successfully executed on the ITC, the proper transaction code may be sent by the ITC to the terminal to acknowledge the verification. However, on many of the existing transaction terminals, the communication medium is a magnetic strip containing encoded information. Therefore, in another embodiment of the present invention shown in FIGS. 22A and 22B the ITC is provided with a magnetic head 1200 embedded in the card that can receive and transmit magnetically encoded information.

Transducer 1200 is positioned within the card, as illustrated in FIG. 22A, such that the transducer can be aligned with the head in a card reading device such as a point of sale (POS) terminal 1210 as illustrated in FIG. 22B. Signals representing the data to be communicated are output serially, emulating the data encoded on a magnetic strip. The circuitry acts to simulate a magnetic field pattern that would exist on the magnetic strip of a credit card. Referring to FIG. 23, the data is output serially bit by bit from microprocessor 1100 to analog circuitry 1220 which drives an inductor 1230 that generates a magnetic field pattern which can be read and interpreted by a conventional magnetic read head 1240 in card reading device 1210.

Preferably a simple digital-to-analog converter may be used as analog circuitry 1220, such as the CMOS read circuitry illustrated at FIG. 24. In this CMOS circuit, transistors Q_1, and Q_2 are biased to form a current source. The gate voltage of Q_2 is replicated at Q_3 which in combination with Q_4 form a current inverter. Further, since the gate of Q_4 is also connected to the gate of Q_5, the drain current of Q_3 - Q_4 is mirrored into Q_5's drain current. Similarly, as the gate of Q_2 is also tied to the gate of Q_8, the Q_1 - Q_2 drain current is mirrored into the drain current of Q_8. Thus, Q_5 and Q_8 act as current sources of opposite polarity biased by the Q_1 - Q_2 combination. Q_5 contributes the sourcing current for the load while Q_8 contributes the sinking current for the load. Transistors Q_6 and Q_7 are digital switches controlled by the microprocessor. When a logic '0' is imposed on the gates of Q_6 - Q_7, Q_6 is on and Q_7 is off. Hence, Q_5 drives the inductor with a positive current through Q_6. When a logic '1' is imposed on the gates of Q_6 - Q_7, Q_6 is off and Q_7 is on. At this point, current is supplied to the inductive load. In this fashion, magnetic fields can be generated in the inductor of opposite polarity under software program control. These fields can then be read by a magnetic stripe card reader.

Similarly, it may be desirable to read into the ITC information transmitted by a magnetic card writing device. Referring to the block diagram of FIG. 25, such circuitry comprises a micro inductor 1300 to read the magnetic field pattern generated by the magnetic write head of the transmitting device, a rectifier 1310, an amplifier 1320, an A/D converter 1330 and a buffer 1340. The signal is rectified, amplified, converted to an analog-to-digital signal and stored in a buffer for subsequent access and use.

While the invention has been described in conjunction with the preferred embodiment, it is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description.

As explained above in conjunction with FIG. 16, ITC cards may be readily programmed for use as credit cards. The cards may also be programmed to be used in the transportation industry, for example, to simplify the system of purchasing bus, airplane and train tickets and replace subway tokens and special passes for the students or the elderly and compute fares based on distance. The cards may also be used for government sponsored programs such as the Food Stamp or Medicaid programs to replace current identification and recording procedures and provide other resident services such as licenses and entitlements. The cards may also be programmed and installed by manufacturers of trucks, cars, and buses for anti-theft protection, performance monitoring and warranty administration. ITC cards may be used to secure access to personal and large computers, computer software, homes, apartments, offices, hotel rooms, data networks, military/government/commercial confidential zones, and services available over the phone line, such as mobile phones, videotex services, databases and pay television. The card may also be used as an all purpose access and recording instrument for the delivery of goods and services in a hotel or resort environment, as well as in a travel application storing itineraries and reservations. Instead of signing a voucher, the type and amount of the service is entered into the ITC card for storage and subsequent retrieval.

The ITC card may also be used in the banking field to store account balances, receipts of banking transactions, store electronic travellers checks, and the like. Presently, transaction cards having a magnetic strip encoded thereon are used to access automatic teller machines (ATM). To access an ATM the cardholder inserts his card into a slot where account information is read off the magnetic strip on the card. The cardholder then enters into the ATM his personal identification number (PIN). The cardholder's PIN and account information is checked and verified before permitting the cardholder access to the ATM. PIN verification may instead be performed on the ITC card. The cardholder would enter the PIN into the card. Once the PIN is verified the card transmits a special code to the ATM for access to the system. PIN verification as well as credit balance verification may also be done on the ITC independent of a bank terminal. In such circumstances, the ITC verifies that sufficient credit exists for a transaction and then generates an approval code for the merchant to receive the transaction. The use of the ITC card provides an additional layer of security against fraudulent access not found in other systems. In addition, the PIN may be changed, at any time, by the cardholder without having to change any cardholder information within the ATM system itself.

The ITC may also operate independently, sometimes referred to as in "stand alone" mode. Thus applications, for example, credit verification, electronic cash or travellers checks, and the like, may be done without the need of a terminal device such as an ATM or point of sale (POS) terminal.

An ITC card may be used to store medical information such as the cardholder's complete medical history or medical insurance coverage. The cards may also be used in the education field for the storage and retrieval of school records, activities, class scheduling, and the like.

The ITC card may also be used as an insurance rate quotation device, and a professional time management and billing device for attorneys, CPA;s and consultants.

Numerous other applications will be evident in view of the foregoing description.

What is claimed is:

1. A card comprising:

a housing;

a means contained within the housing to provide power to operate the card;

an alphanumeric keypad located on a surface of the housing for entry of information by a user;

a display located on a surface of the housing for the presentation of information;

a microprocessor contained within the housing;

at least one port in said housing connected to the microprocessor for the input and output of information;

a memory contained within the housing and connected to said microprocessor; and

an operating system stored in the memory and controlling operation of the card through the microprocessor, comprising a means for generating a plurality of messages on the display that prompt the user during the operation of the card.

2. The card of claim 1 wherein the alphanumeric keypad is multifunctional and programmable.

3. The card of claim 1 wherein the display is a segmented display.

4. The card of claim 3 wherein the display is a bit-mapped display.

5. The card of claim 1 further comprising a means for securing the card.

6. The card of claim 1 further comprising circuitry to receive and transmit magnetically encoded information to an external device.

7. The card of claim 1 further comprising at least one application program stored in memory and utilized by the operating system to perform a specific function.

8. The card of claim 7 wherein said operating system comprises:

a plurality of modules operable while in an idle state to monitor the keypad, the ports and to update the date and time; and

a plurality of service routines to control the display, the ports, the keyboard, the memory and the application programs.

9. In a card comprising a multifunction alphanumeric keypad, a display, at least one input/output port and a microprocessor which generates outputs at the port or display in response to inputs at said port or keypad, a method of operating said card comprising the steps of:

prompting a user on the operation of the card through a plurality of messages generated on said display;

entering information into the card through said alphanumeric keypad in response to said messages; and

executing a function based on the information entered through the keypad.

10. The method of claim 9 further comprising the step of providing audio or visual feedback to a user of said card through said display or said input/output port based on the execution of the function.

11. A card comprising:

a housing;

a multifunction alphanumeric keypad located on a surface of the housing for entry of information by a user;

a display for presentation of information located on the surface of the housing;

a means contained within the housing to provide power to operate the card;

a microprocessor contained within the housing;

at least one port in said housing connected to said microprocessor for input and output of information;

a memory contained within the housing and connected to said microprocessor; and

an operating system stored in the memory and controlling operation of the card through the microprocessor, said system providing outputs at said port or display in response to inputs at said port or keypad as well as a means for programming a variety of different application programs into the card whereby a card is provided that can be programmed for any one or more of a variety of functions.

12. The card of claim 12 further comprising at least one undefined programmable function key that may be defined for a specific purpose according to the application.

13. The card of claim 12 wherein the alphanumeric keypad further comprises application control keys which control the selection and execution of the application programs in the intelligent transaction card.

14. The card of claim 12 wherein the display presents a plurality of audio and visual prompts to lead the user through operation of the card.

15. A card comprising:

a housing;

a means contained within the housing to provide power to operate the card;

an alphanumeric keypad located on a surface of the housing for entry of information by a user;

a display located on the surface of the housing for the presentation of information;

a microprocessor contained within the housing to control the operation of the card;

at least one port in said housing connected to the microprocessor for the input and output of information;

a memory contained within the housing and connected to said microprocessor;

an operating system stored in the memory and controlling the operation of the card through the microprocessor;

means for storing in said memory a personal identification number (PIN) known to a user of said card;

means accessible to said user for changing the PIN stored in said memory;

a credit/purchase application program stored in the memory and executed by the microprocessor through the operating system, said credit/purchase application comprising;

means for storing a credit balance;

means for receiving through the keypad the user's PIN and amount of a transaction;

means for verifying that the PIN received through the keypad is equal to a PIN stored in the memory;

means for verifying that a sufficient credit balance exists to execute the transaction; and

means for generating an approval code to be displayed on the display if sufficient credit balance exists and the PIN received through the keypad is equal to the PIN stored in memory;

whereby the card verifies the user's credit balance for a purchase and generates an approval code for the purchase independent of any external terminal device.

16. A card comprising:

a housing;

a means contained within the housing to provide power to operate the card;

an alphanumeric keypad located on a surface of the housing for entry of information by a user;

a display located on a surface of the housing for the presentation of information;

a microprocessor contained within the housing;

at least one port in said housing connected to the microprocessor for the input and output of information;

a memory contained within the housing and connected to said microprocessor;

an operating system stored in the memory and controlling operation of the card, through the microprocessor, said operating system comprising a memory management function which permits access of memory through logical memory addresses;

at least one application program stored in the memory to direct the operating program, said application program comprising commands which store and retrieve information from memory addressed by logical addresses;

whereby multiple application programs may be stored and executed on the same card and the memory used during the execution of the first application program will not overwrite the memory used during the execution of another application program.

17. The intelligent transaction card of claim 16 further comprising a means for securing access to the card comprising:

means for securing access to execute an application program;

means for securing access to a file in memory;

means for securing access to a record in a file; and

means for securing access to the execution of certain commands.

18. The card of claim 17 wherein the means for securing access to the card comprises keys which are stored on the card.

19. The card of claim 16 wherein application programs may be added and removed from the card without disassembling the card.

20. The card of claim 16 wherein the application programs are stored on the card by a card issuer.

21. The card of claim 16 further comprising circuitry to receive and transmit magnetically encoded information to an external device.

22. The card of claim 16 wherein said operating system further comprises:

a plurality of modules operable while in an idle state to monitor the keypad, the ports and to update the date and time; and

a plurality of service routines to control the display, the ports, and the keyboard.

23. A card comprising:

a housing;

a means contained within the housing to provide power to operate the card;

a keypad located on a surface of the housing for entry of information by a user;

an alphanumeric display located on a surface of the housing for the presentation of alphanumeric information;

a microprocessor contained within the housing;

a memory contained within the housing and connected to said microprocessor;

an operating system stored in the memory and controlling operation of the card through the microprocessor, comprising a means for generating a plurality of messages on the display and audio and visual outputs; and

at least one port in said housing connected to the microprocessor for the input and output of magnetically encoded information.

FIG. 1A

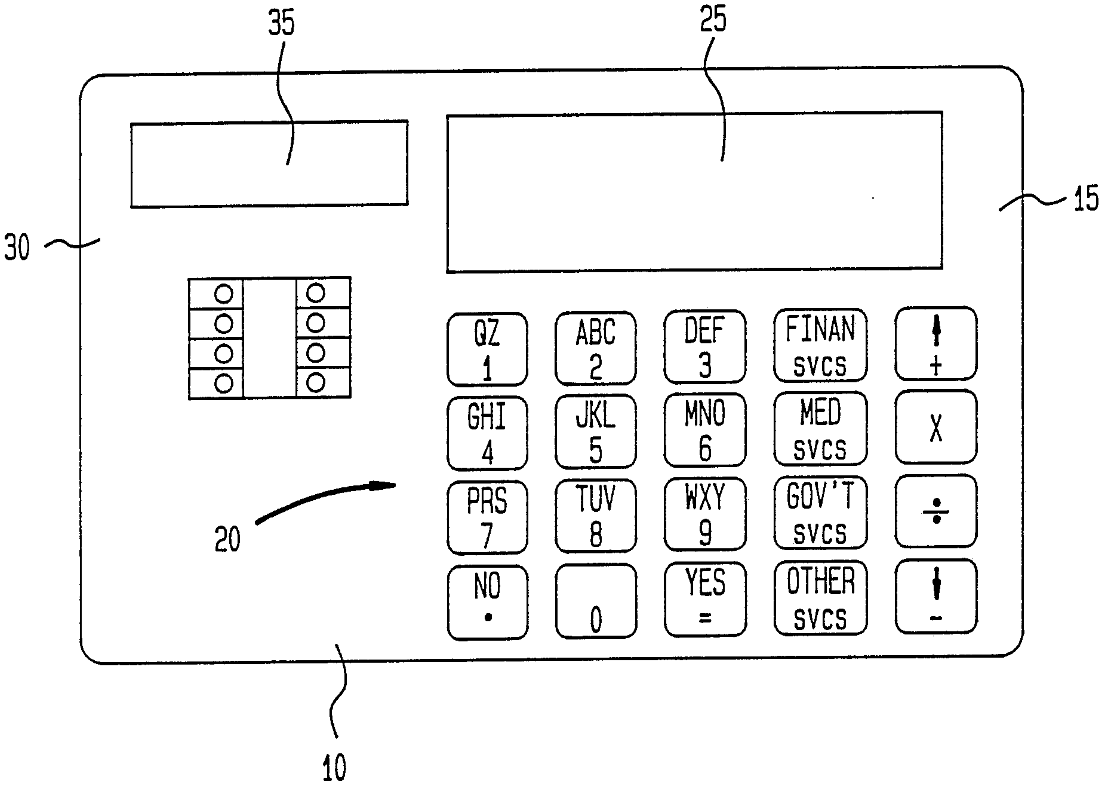

35
25
15
30
QZ 1
ABC 2
DEF 3
FINAN svcs
+
GHI 4
JKL 5
MNO 6
MED svcs
X
PRS 7
TUV 8
WXY 9
GOV'T svcs
÷
20
NO •
0
YES =
OTHER svcs
-
10

FIG. 1B

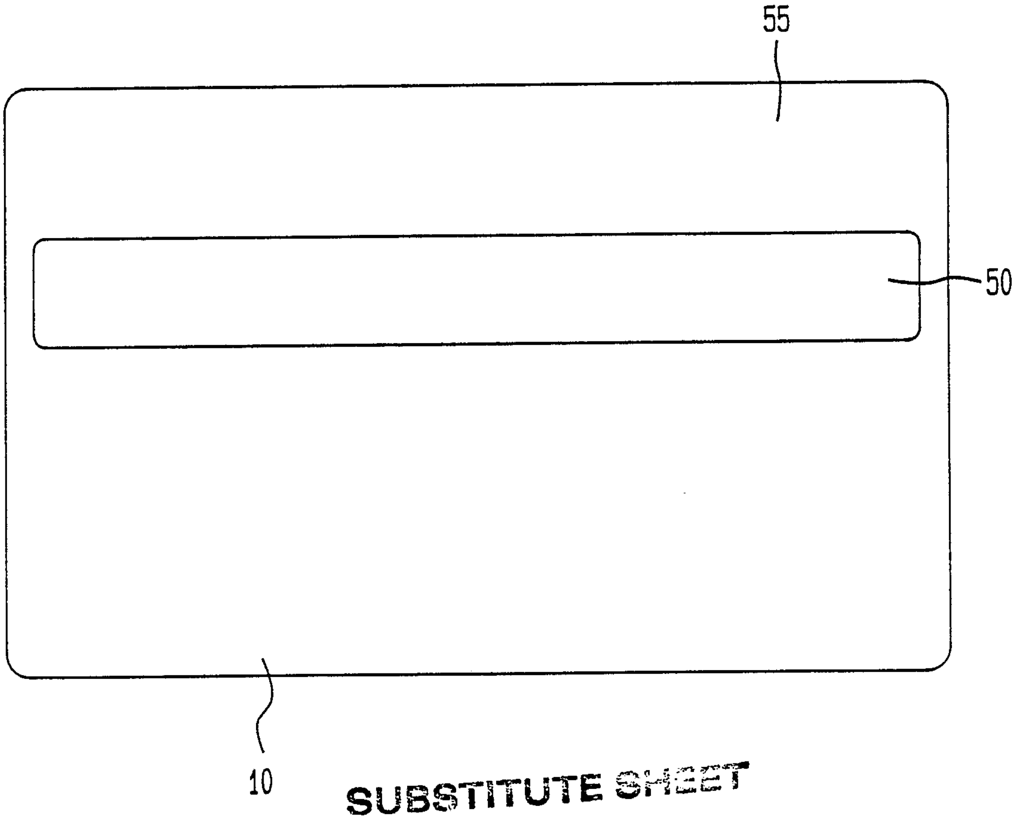

55
50
10

FIG. 2A

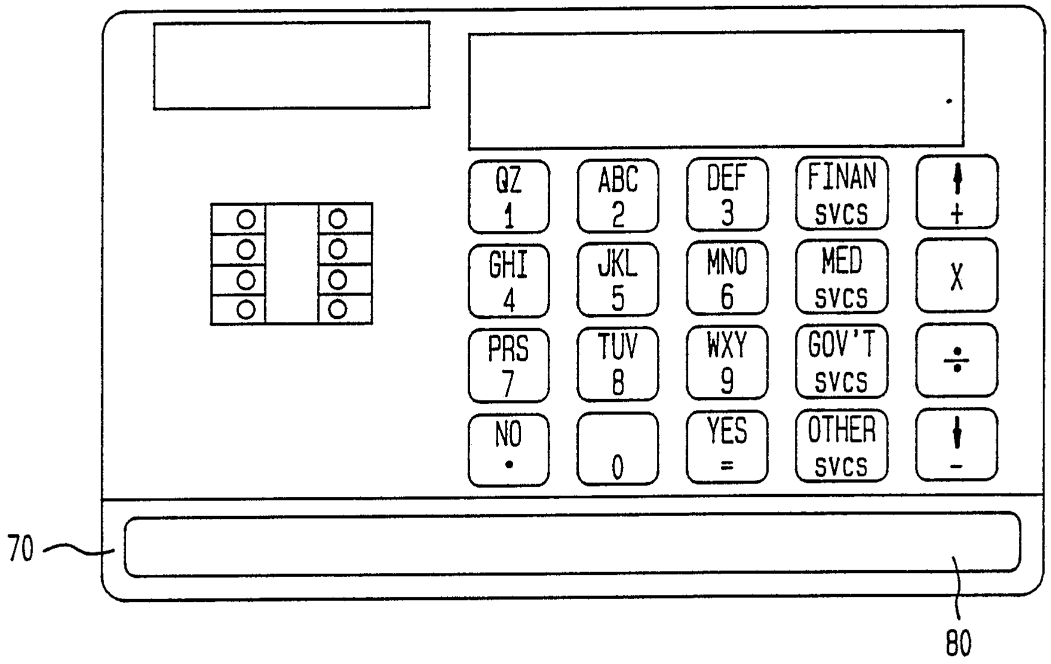

QZ
1
ABC
2
DEF
3
FINAN
svcs
+
GHI
4
JKL
5
MNO
6
MED
svcs
X
PRS
7
TUV
8
WXY
9
GOV'T
svcs
÷
NO
•
0
YES
=
OTHER
svcs
-
70
80

FIG. 2B

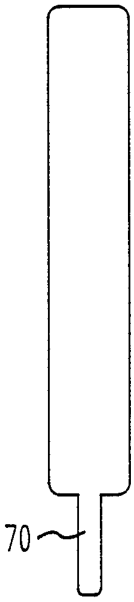

70

FIG. 3

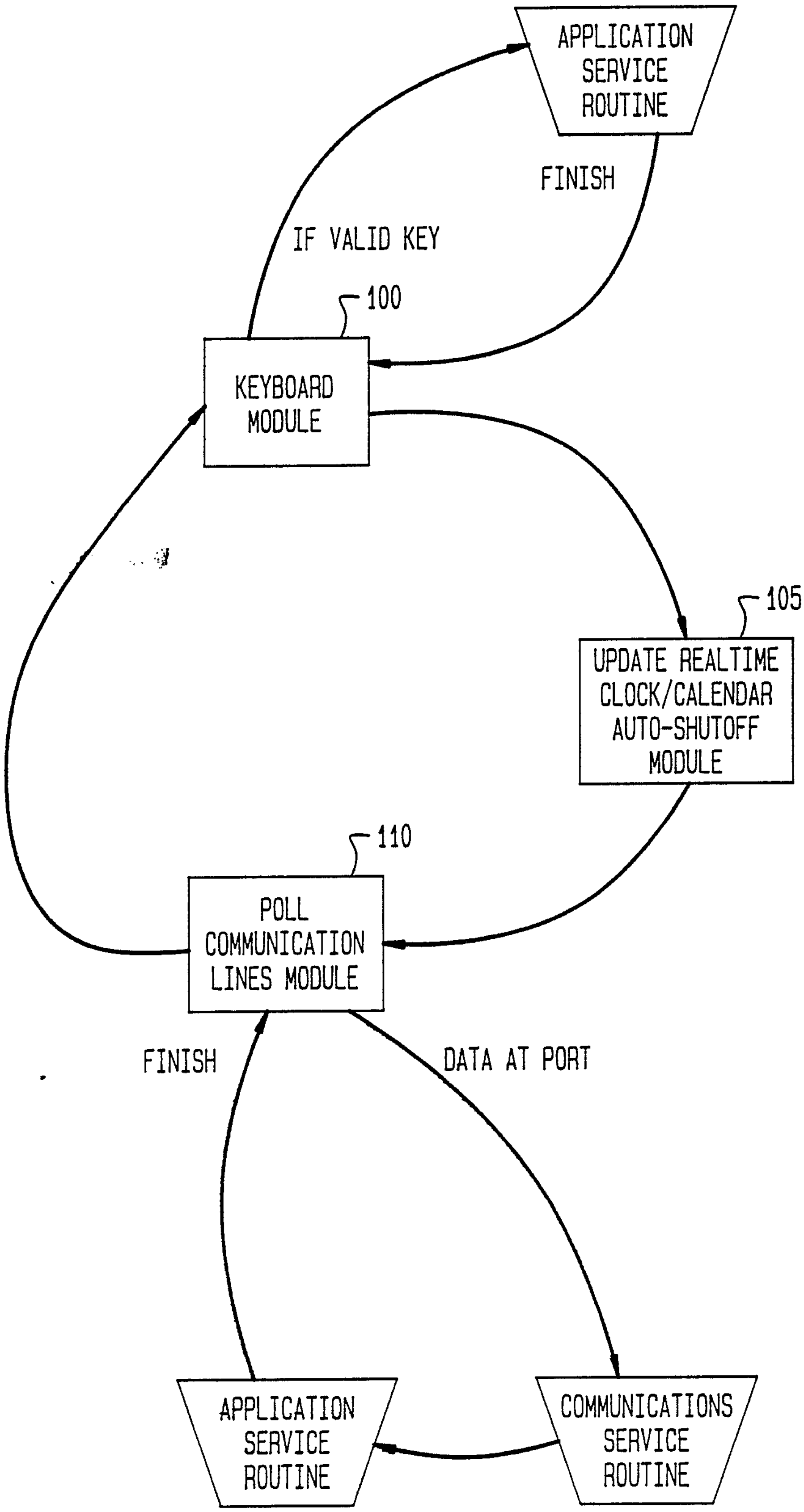

APPLICATION SERVICE ROUTINE
FINISH
IF VALID KEY
100
KEYBOARD MODULE
105
UPDATE REALTIME CLOCK/CALENDAR AUTO-SHUTOFF MODULE
110
POLL COMMUNICATION LINES MODULE
FINISH
DATA AT PORT
APPLICATION SERVICE ROUTINE
COMMUNICATIONS SERVICE ROUTINE

FIG. 4

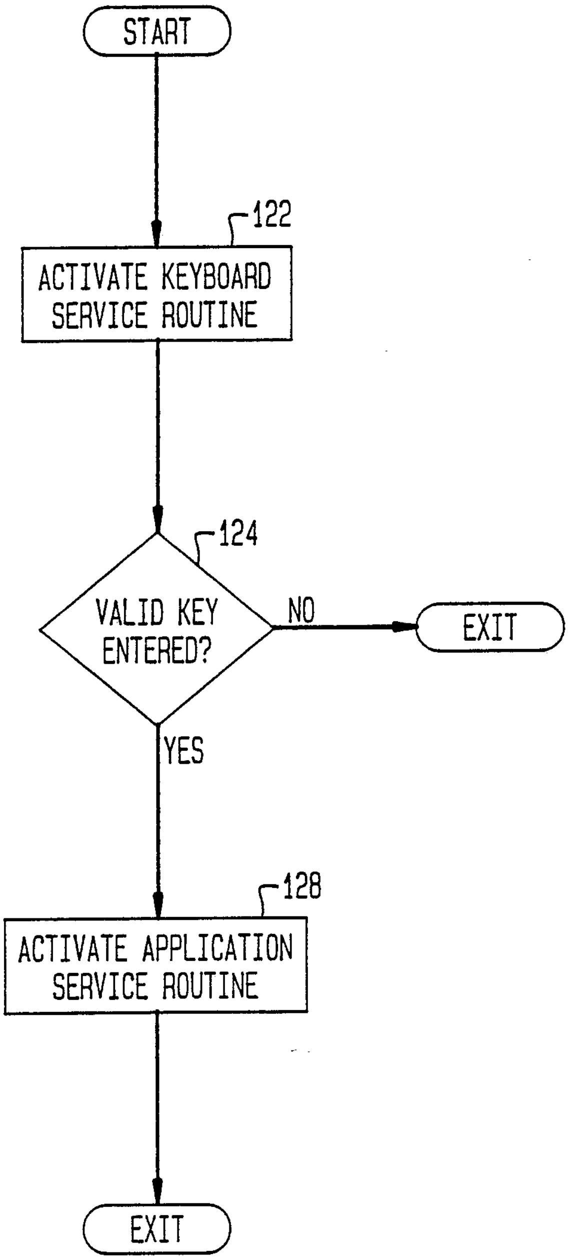

START
122
ACTIVATE KEYBOARD SERVICE ROUTINE
124
VALID KEY ENTERED?
NO
EXIT
YES
128
ACTIVATE APPLICATION SERVICE ROUTINE
EXIT

FIG. 5

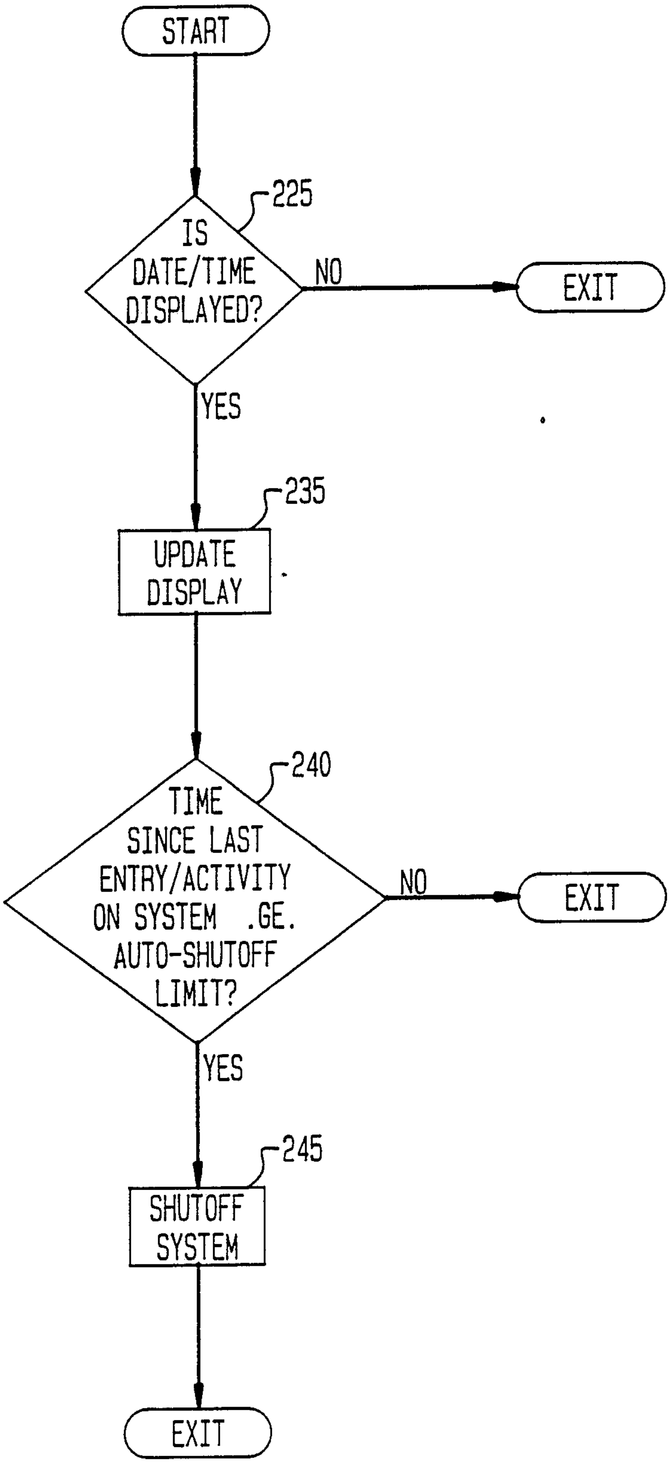

START
225
IS DATE/TIME DISPLAYED?
NO
EXIT
YES
235
UPDATE DISPLAY
240
TIME SINCE LAST ENTRY/ACTIVITY ON SYSTEM .GE. AUTO-SHUTOFF LIMIT?
NO
EXIT
YES
245
SHUTOFF SYSTEM
EXIT

FIG. 6

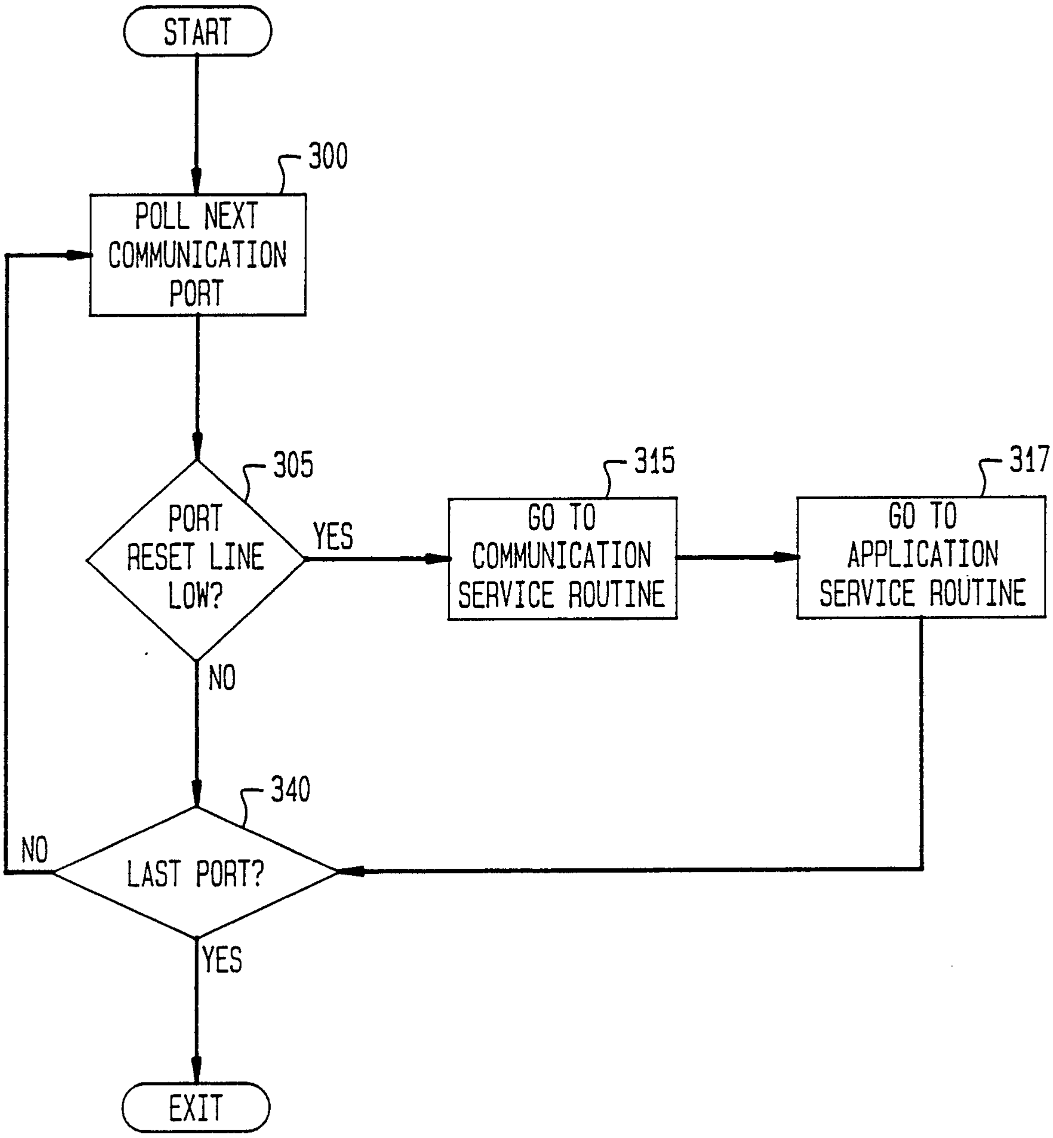

START
300
POLL NEXT COMMUNICATION PORT
305
PORT RESET LINE LOW?
YES
NO
315
GO TO COMMUNICATION SERVICE ROUTINE
317
GO TO APPLICATION SERVICE ROUTINE
340
LAST PORT?
NO
YES
EXIT

FIG. 7A

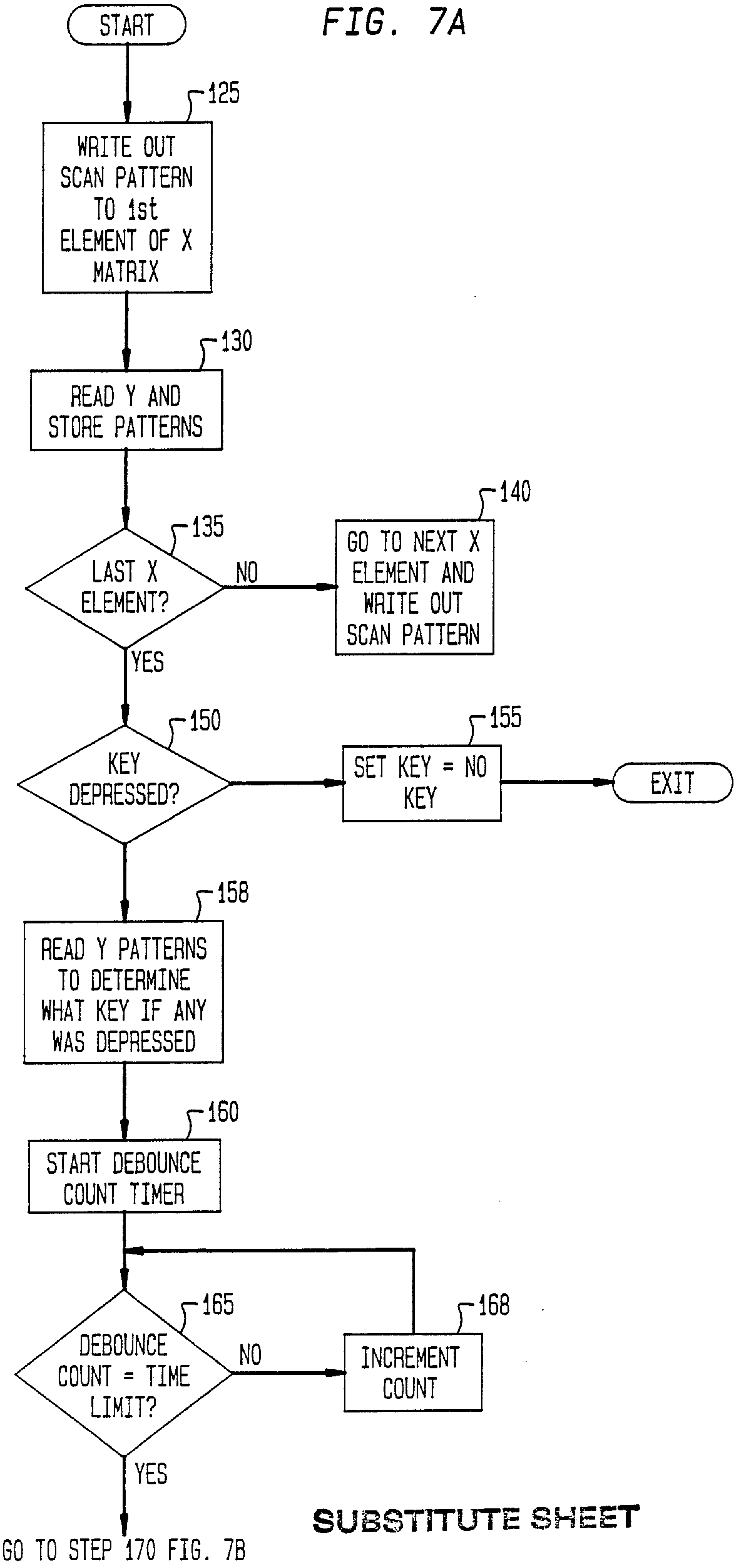

START
125
WRITE OUT SCAN PATTERN TO 1st ELEMENT OF X MATRIX
130
READ Y AND STORE PATTERNS
135
LAST X ELEMENT?
NO
140
GO TO NEXT X ELEMENT AND WRITE OUT SCAN PATTERN
YES
150
KEY DEPRESSED?
155
SET KEY = NO KEY
EXIT
158
READ Y PATTERNS TO DETERMINE WHAT KEY IF ANY WAS DEPRESSED
160
START DEBOUNCE COUNT TIMER
165
DEBOUNCE COUNT = TIME LIMIT?
NO
168
INCREMENT COUNT
YES
GO TO STEP 170 FIG. 7B

FIG. 7B

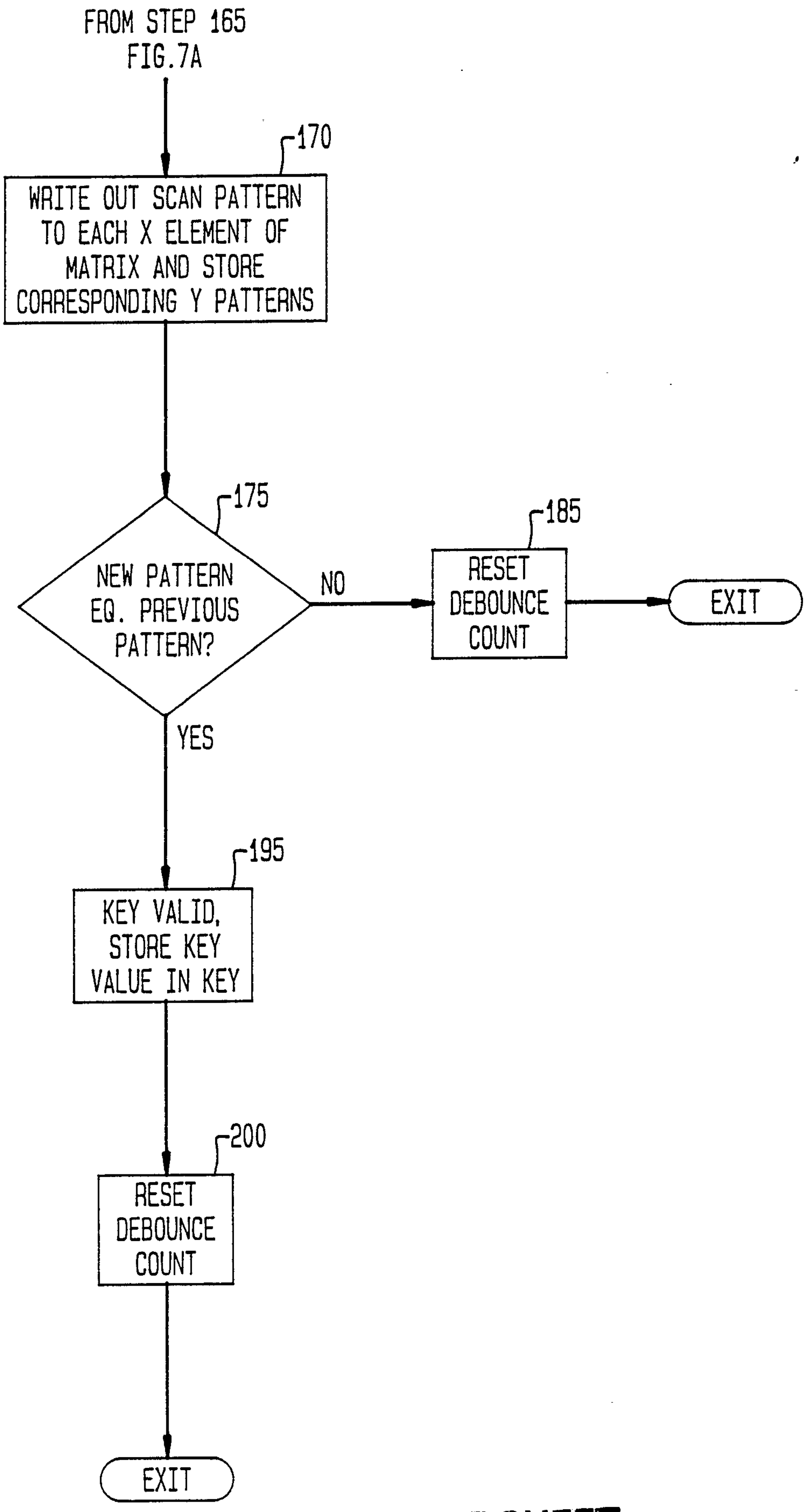

FROM STEP 165
FIG.7A
170
WRITE OUT SCAN PATTERN TO EACH X ELEMENT OF MATRIX AND STORE CORRESPONDING Y PATTERNS
175
NEW PATTERN EQ. PREVIOUS PATTERN?
NO
185
RESET DEBOUNCE COUNT
EXIT
YES
195
KEY VALID, STORE KEY VALUE IN KEY
200
RESET DEBOUNCE COUNT
EXIT

FIG. 8

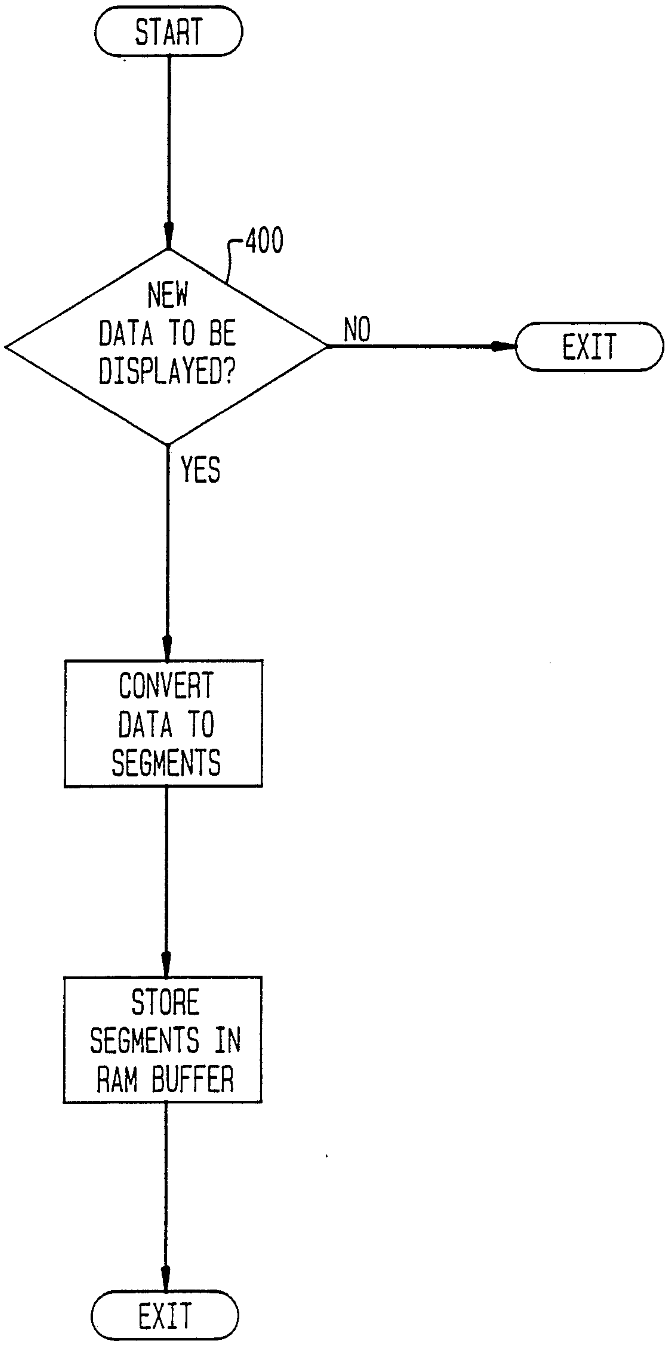

START
400
NEW DATA TO BE DISPLAYED?
NO
EXIT
YES
CONVERT DATA TO SEGMENTS
STORE SEGMENTS IN RAM BUFFER
EXIT

FIG. 9

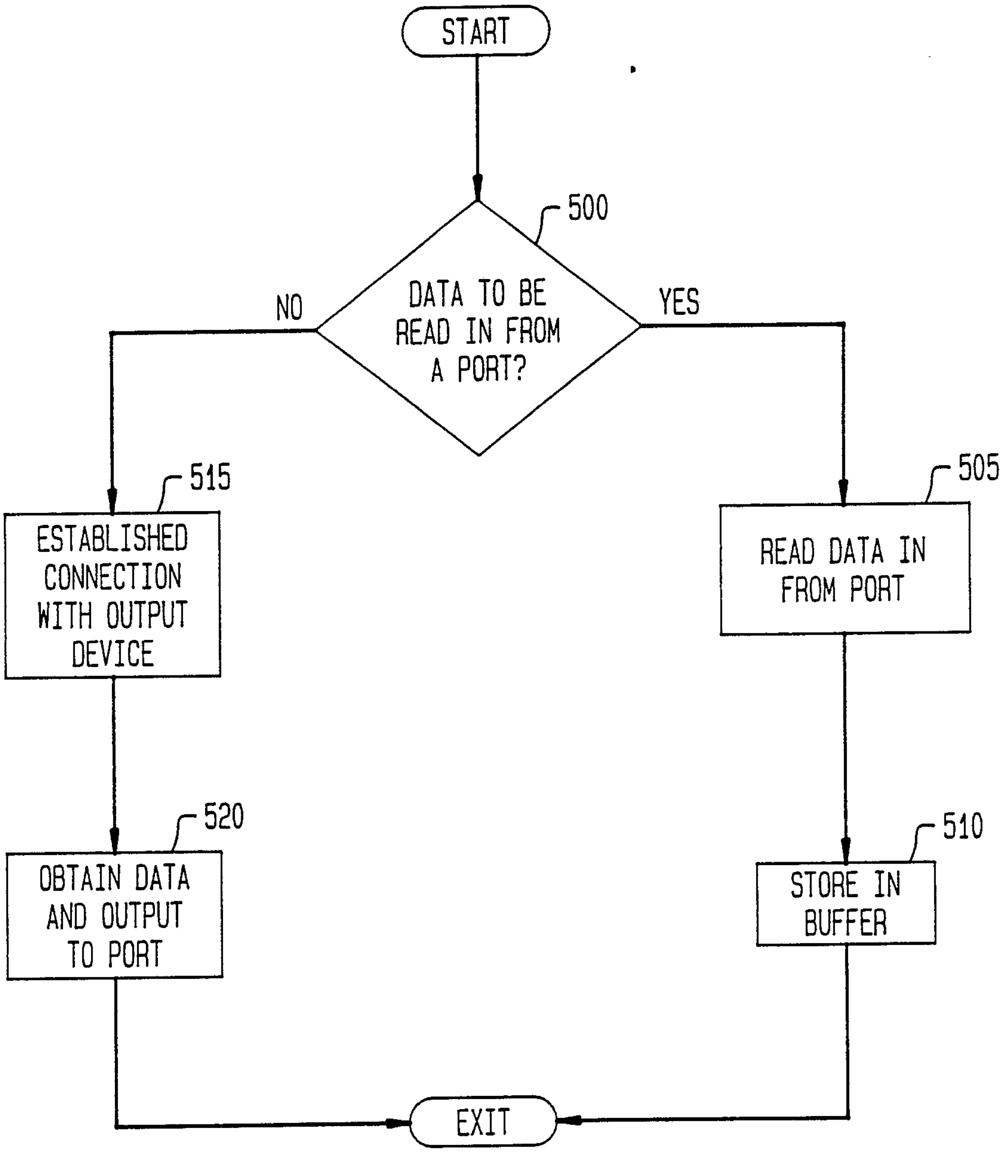

START
500
DATA TO BE READ IN FROM A PORT?
NO
YES
515
ESTABLISHED CONNECTION WITH OUTPUT DEVICE
520
OBTAIN DATA AND OUTPUT TO PORT
505
READ DATA IN FROM PORT
510
STORE IN BUFFER
EXIT

FIG. 10A

SECURITY FLAGS

14	13	12	11	10	9	8	7	6	5	4	3	2	1
ON	OFF	OFF	OFF	ON	ON	OFF	OFF	OFF	ON	OFF	ON	OFF	OFF

FIG. 10B

COMMAND ACCESS TABLE

INST PTR	USE FLAG
1	5
2	7
3	4
4	14
5	5
6	3
7	6
8	8
9	12
10	1
11	10
12	9
13	11
14	3

FIG. 10C

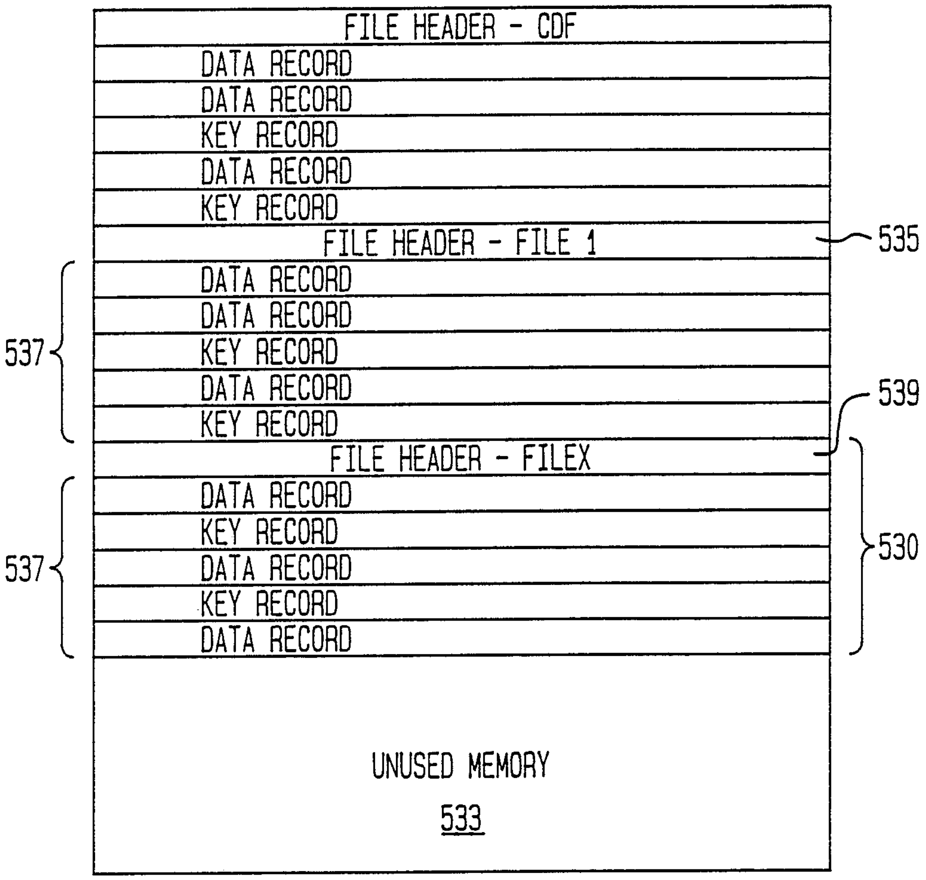

FILE HEADER - CDF
DATA RECORD
DATA RECORD
KEY RECORD
DATA RECORD
KEY RECORD
FILE HEADER - FILE 1
535
537
DATA RECORD
DATA RECORD
KEY RECORD
DATA RECORD
KEY RECORD
539
FILE HEADER - FILEX
537
DATA RECORD
KEY RECORD
DATA RECORD
KEY RECORD
DATA RECORD
530
UNUSED MEMORY
533

FIG. 10D

FILE HEADER

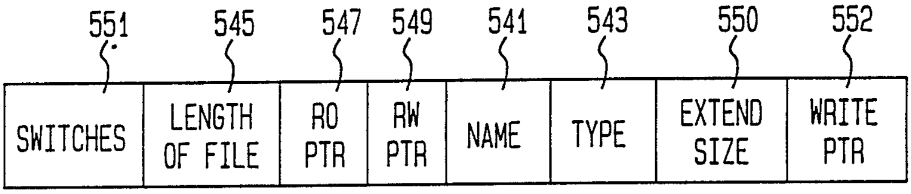


FIG. 10E

DATA RECORD HEADER

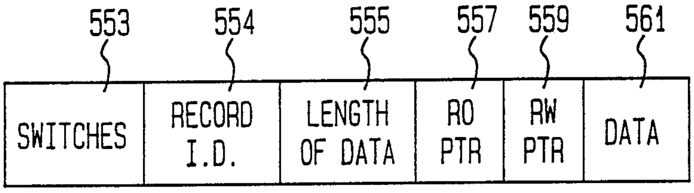


FIG. 10F

KEY RECORD HEADER

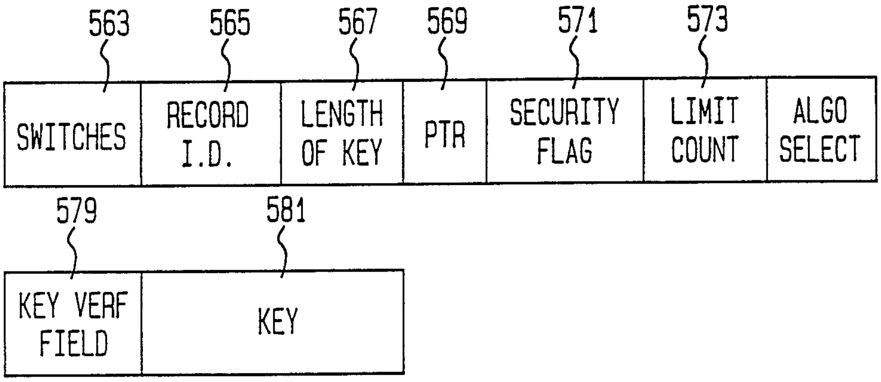


FIG. 10AA

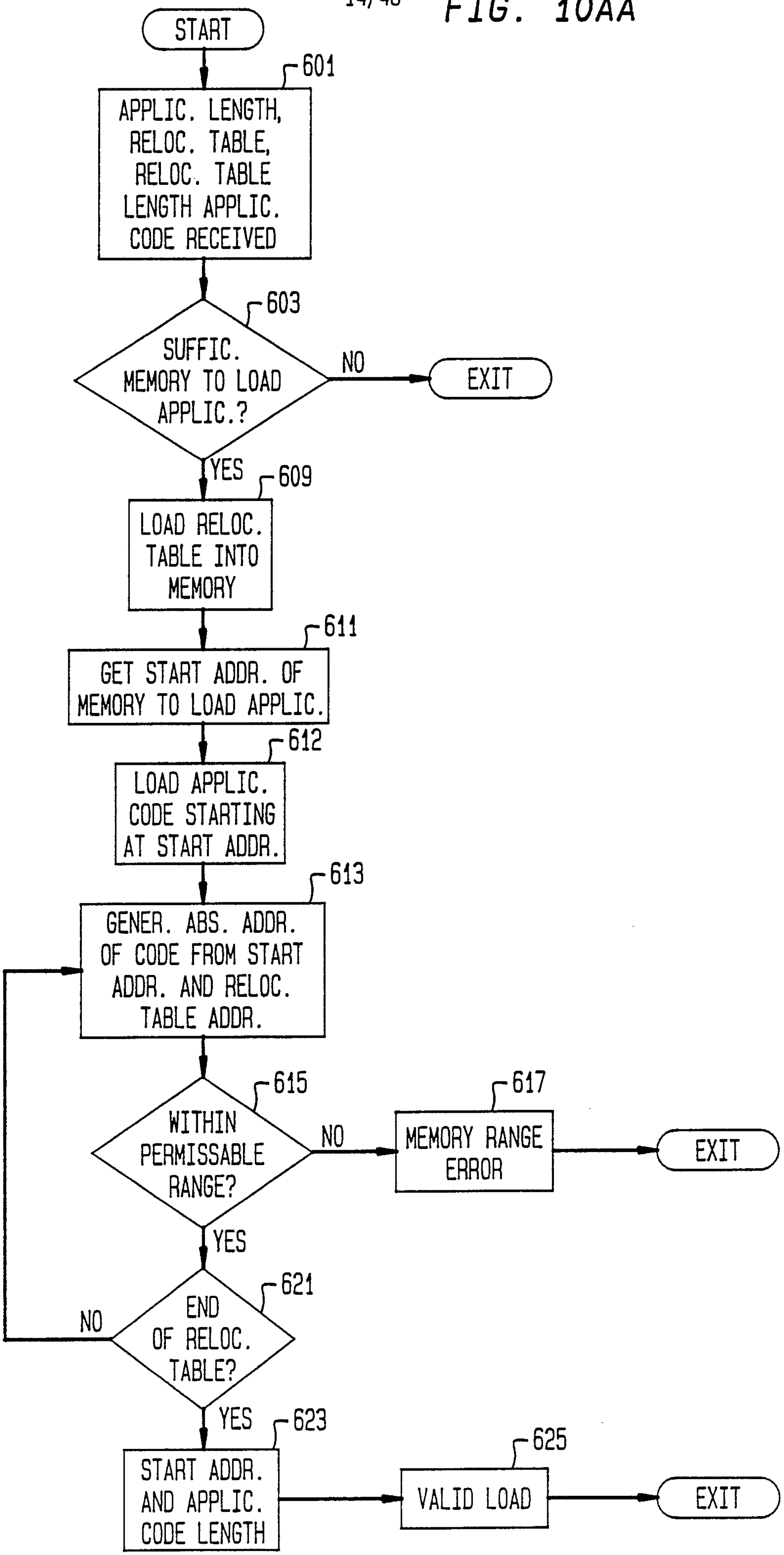

START
601
APPLIC. LENGTH, RELOC. TABLE, RELOC. TABLE LENGTH APPLIC. CODE RECEIVED
603
SUFFIC. MEMORY TO LOAD APPLIC.?
NO
EXIT
YES
609
LOAD RELOC. TABLE INTO MEMORY
611
GET START ADDR. OF MEMORY TO LOAD APPLIC.
612
LOAD APPLIC. CODE STARTING AT START ADDR.
613
GENER. ABS. ADDR. OF CODE FROM START ADDR. AND RELOC. TABLE ADDR.
615
WITHIN PERMISSABLE RANGE?
NO
617
MEMORY RANGE ERROR
EXIT
YES
621
END OF RELOC. TABLE?
NO
YES
623
START ADDR. AND APPLIC. CODE LENGTH
625
VALID LOAD
EXIT

FIG. 10G

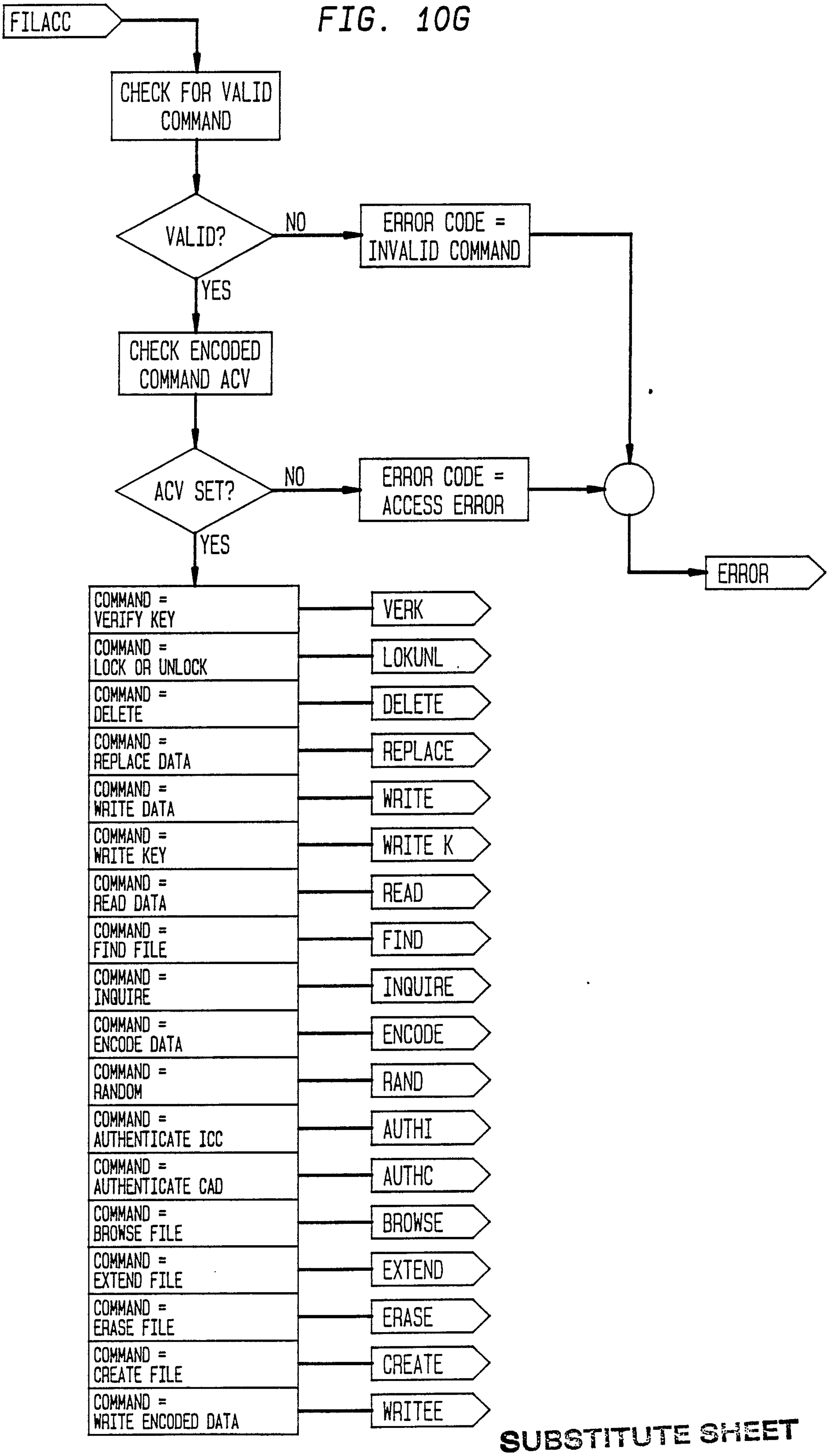

FILACC
CHECK FOR VALID COMMAND
VALID?
NO
ERROR CODE = INVALID COMMAND
YES
CHECK ENCODED COMMAND ACV
ACV SET?
NO
ERROR CODE = ACCESS ERROR
YES
ERROR
COMMAND = VERIFY KEY
VERK
COMMAND = LOCK OR UNLOCK
LOKUNL
COMMAND = DELETE
DELETE
COMMAND = REPLACE DATA
REPLACE
COMMAND = WRITE DATA
WRITE
COMMAND = WRITE KEY
WRITE K
COMMAND = READ DATA
READ
COMMAND = FIND FILE
FIND
COMMAND = INQUIRE
INQUIRE
COMMAND = ENCODE DATA
ENCODE
COMMAND = RANDOM
RAND
COMMAND = AUTHENTICATE ICC
AUTHI
COMMAND = AUTHENTICATE CAD
AUTHC
COMMAND = BROWSE FILE
BROWSE
COMMAND = EXTEND FILE
EXTEND
COMMAND = ERASE FILE
ERASE
COMMAND = CREATE FILE
CREATE
COMMAND = WRITE ENCODED DATA
WRITEE

FIG. 10H

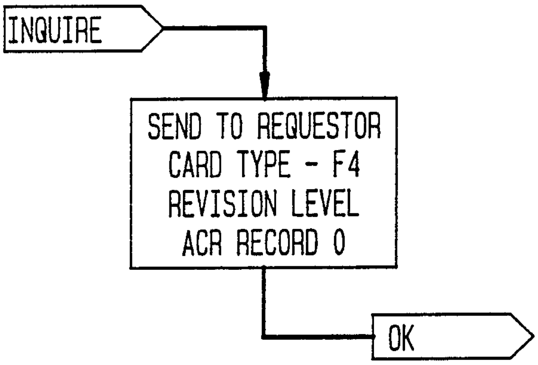

INQUIRE
SEND TO REQUESTOR CARD TYPE - F4 REVISION LEVEL ACR RECORD 0
OK

FIG. 10I

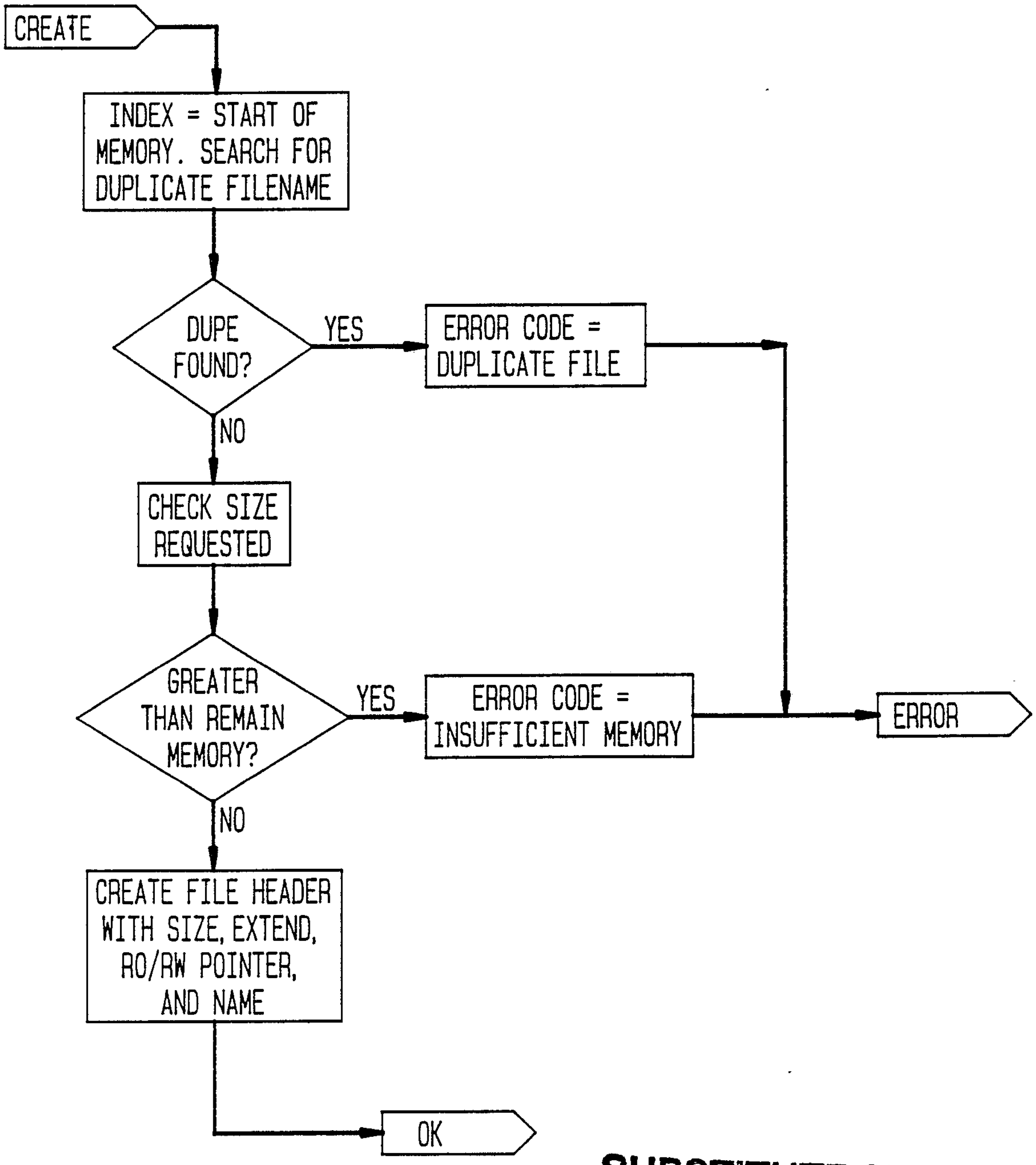

CREATE
INDEX = START OF MEMORY. SEARCH FOR DUPLICATE FILENAME
DUPE FOUND?
YES
ERROR CODE = DUPLICATE FILE
NO
CHECK SIZE REQUESTED
GREATER THAN REMAIN MEMORY?
YES
ERROR CODE = INSUFFICIENT MEMORY
ERROR
NO
CREATE FILE HEADER WITH SIZE, EXTEND, RO/RW POINTER, AND NAME
OK

FIG. 10J

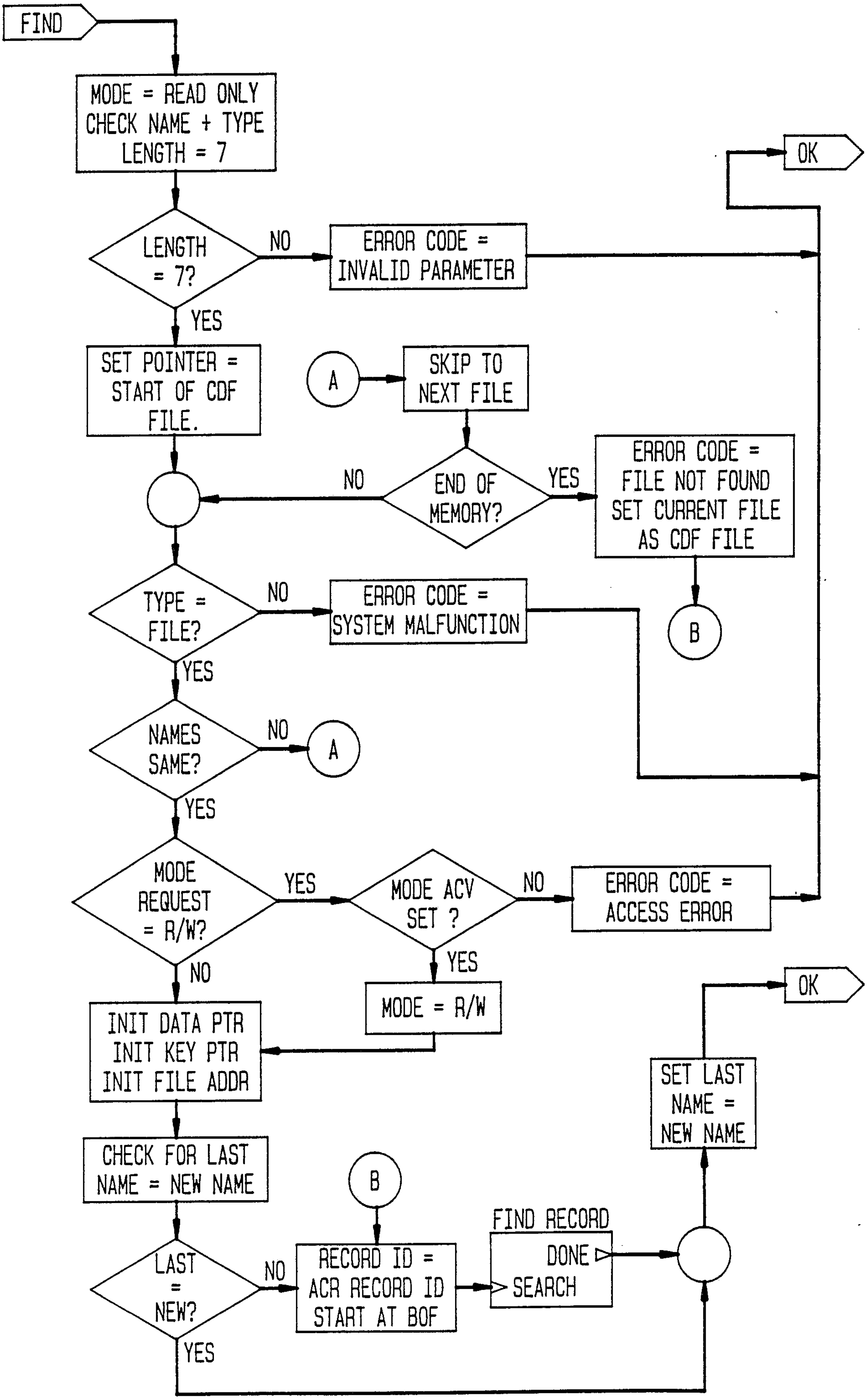

FIND
MODE = READ ONLY
CHECK NAME + TYPE
LENGTH = 7
LENGTH = 7?
NO
ERROR CODE = INVALID PARAMETER
OK
YES
SET POINTER = START OF CDF FILE.
A
SKIP TO NEXT FILE
NO
END OF MEMORY?
YES
ERROR CODE = FILE NOT FOUND SET CURRENT FILE AS CDF FILE
TYPE = FILE?
NO
ERROR CODE = SYSTEM MALFUNCTION
B
YES
NAMES SAME?
NO
A
YES
MODE REQUEST = R/W?
YES
MODE ACV SET ?
NO
ERROR CODE = ACCESS ERROR
YES
NO
MODE = R/W
OK
INIT DATA PTR
INIT KEY PTR
INIT FILE ADDR
SET LAST NAME = NEW NAME
CHECK FOR LAST NAME = NEW NAME
B
FIND RECORD
DONE
SEARCH
LAST = NEW?
NO
RECORD ID = ACR RECORD ID START AT BOF
YES

FIG. 10K

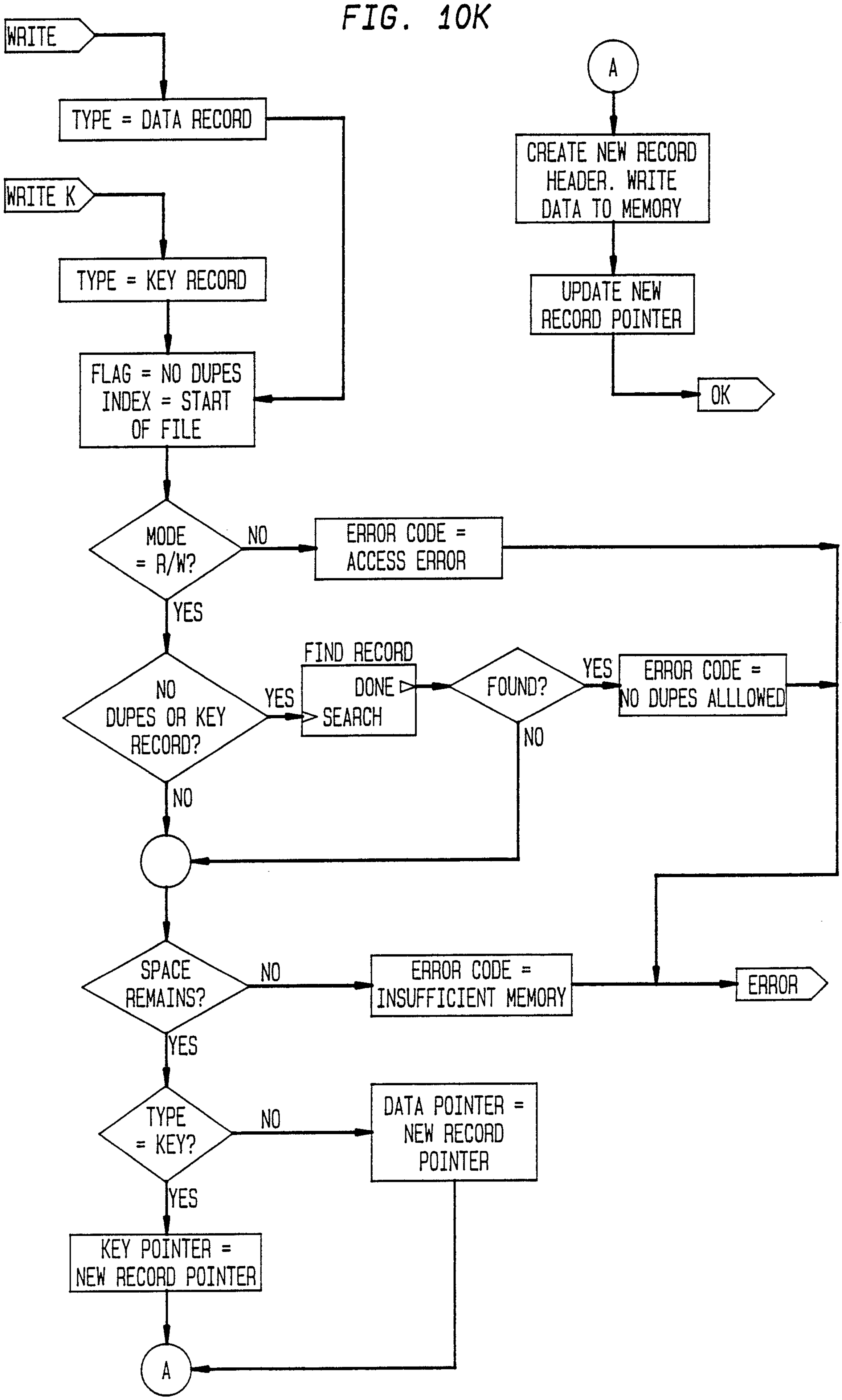

WRITE
TYPE = DATA RECORD
WRITE K
TYPE = KEY RECORD
FLAG = NO DUPES INDEX = START OF FILE
MODE = R/W?
NO
ERROR CODE = ACCESS ERROR
YES
NO DUPES OR KEY RECORD?
YES
FIND RECORD
DONE
SEARCH
FOUND?
YES
ERROR CODE = NO DUPES ALLLOWED
NO
NO
SPACE REMAINS?
NO
ERROR CODE = INSUFFICIENT MEMORY
ERROR
YES
TYPE = KEY?
NO
DATA POINTER = NEW RECORD POINTER
YES
KEY POINTER = NEW RECORD POINTER
A
A
CREATE NEW RECORD HEADER. WRITE DATA TO MEMORY
UPDATE NEW RECORD POINTER
OK

FIG. 10L

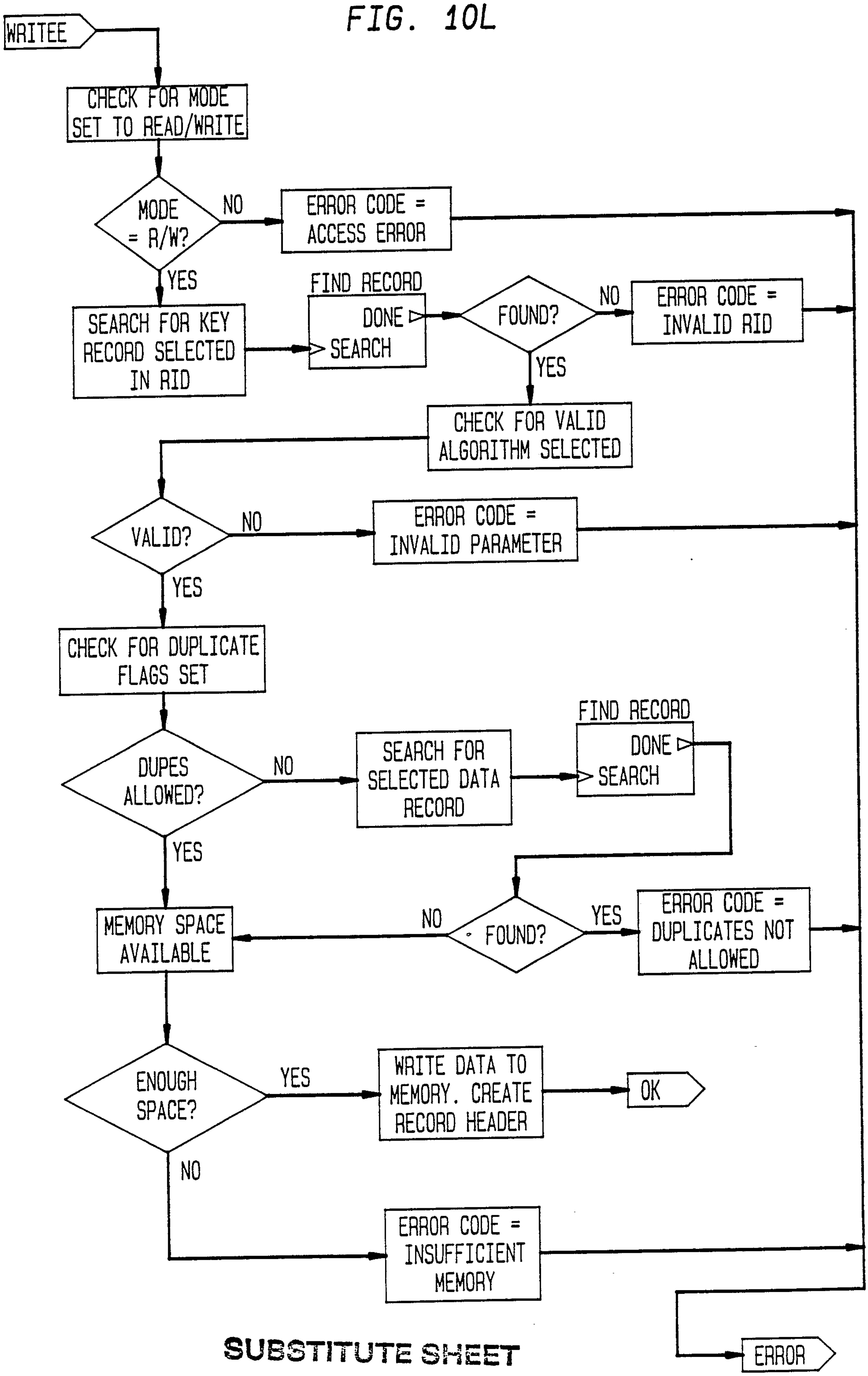

WRITEE
CHECK FOR MODE SET TO READ/WRITE
MODE = R/W?
NO
ERROR CODE = ACCESS ERROR
YES
SEARCH FOR KEY RECORD SELECTED IN RID
FIND RECORD
DONE
SEARCH
FOUND?
NO
ERROR CODE = INVALID RID
YES
CHECK FOR VALID ALGORITHM SELECTED
VALID?
NO
ERROR CODE = INVALID PARAMETER
YES
CHECK FOR DUPLICATE FLAGS SET
DUPES ALLOWED?
NO
SEARCH FOR SELECTED DATA RECORD
FIND RECORD
DONE
SEARCH
YES
MEMORY SPACE AVAILABLE
NO
FOUND?
YES
ERROR CODE = DUPLICATES NOT ALLOWED
ENOUGH SPACE?
YES
WRITE DATA TO MEMORY. CREATE RECORD HEADER
OK
NO
ERROR CODE = INSUFFICIENT MEMORY
ERROR

FIG. 10M

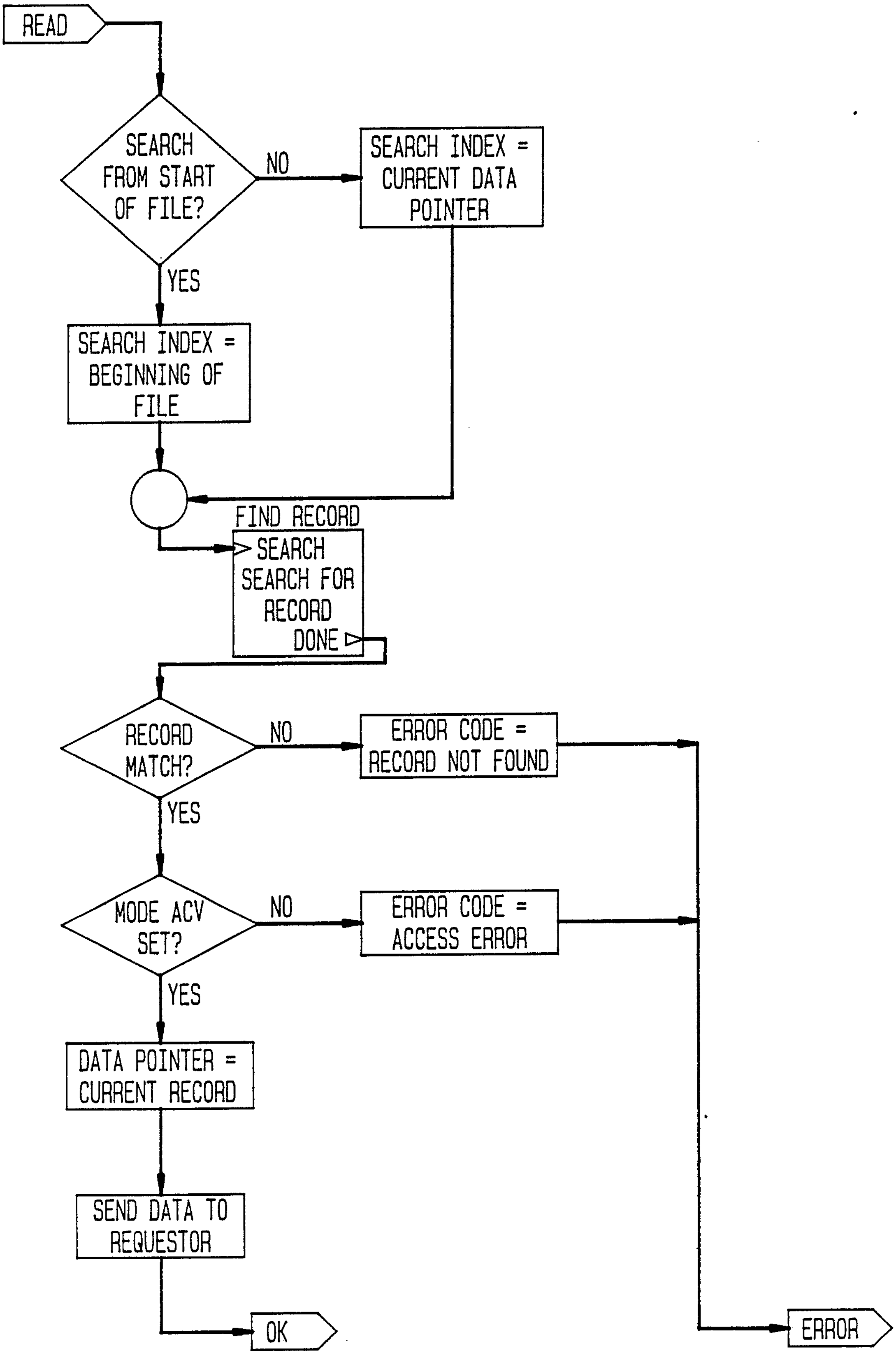

READ
SEARCH FROM START OF FILE?
NO
SEARCH INDEX = CURRENT DATA POINTER
YES
SEARCH INDEX = BEGINNING OF FILE
FIND RECORD
SEARCH
SEARCH FOR RECORD
DONE
RECORD MATCH?
NO
ERROR CODE = RECORD NOT FOUND
YES
MODE ACV SET?
NO
ERROR CODE = ACCESS ERROR
YES
DATA POINTER = CURRENT RECORD
SEND DATA TO REQUESTOR
OK
ERROR

FIG. 10N

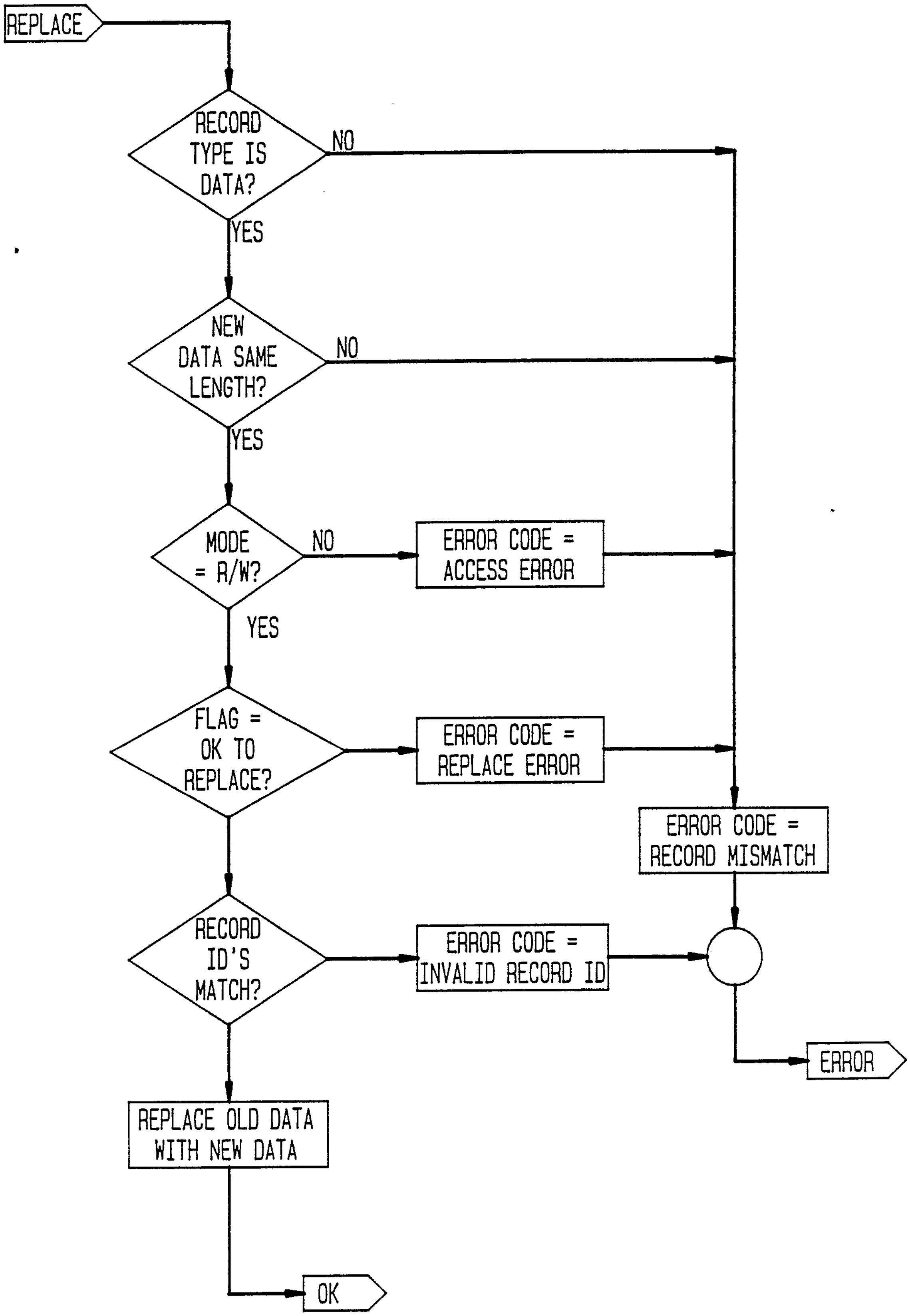

REPLACE
RECORD TYPE IS DATA?
NO
YES
NEW DATA SAME LENGTH?
NO
YES
MODE = R/W?
NO
ERROR CODE = ACCESS ERROR
YES
FLAG = OK TO REPLACE?
ERROR CODE = REPLACE ERROR
ERROR CODE = RECORD MISMATCH
RECORD ID'S MATCH?
ERROR CODE = INVALID RECORD ID
ERROR
REPLACE OLD DATA WITH NEW DATA
OK

FIG. 100

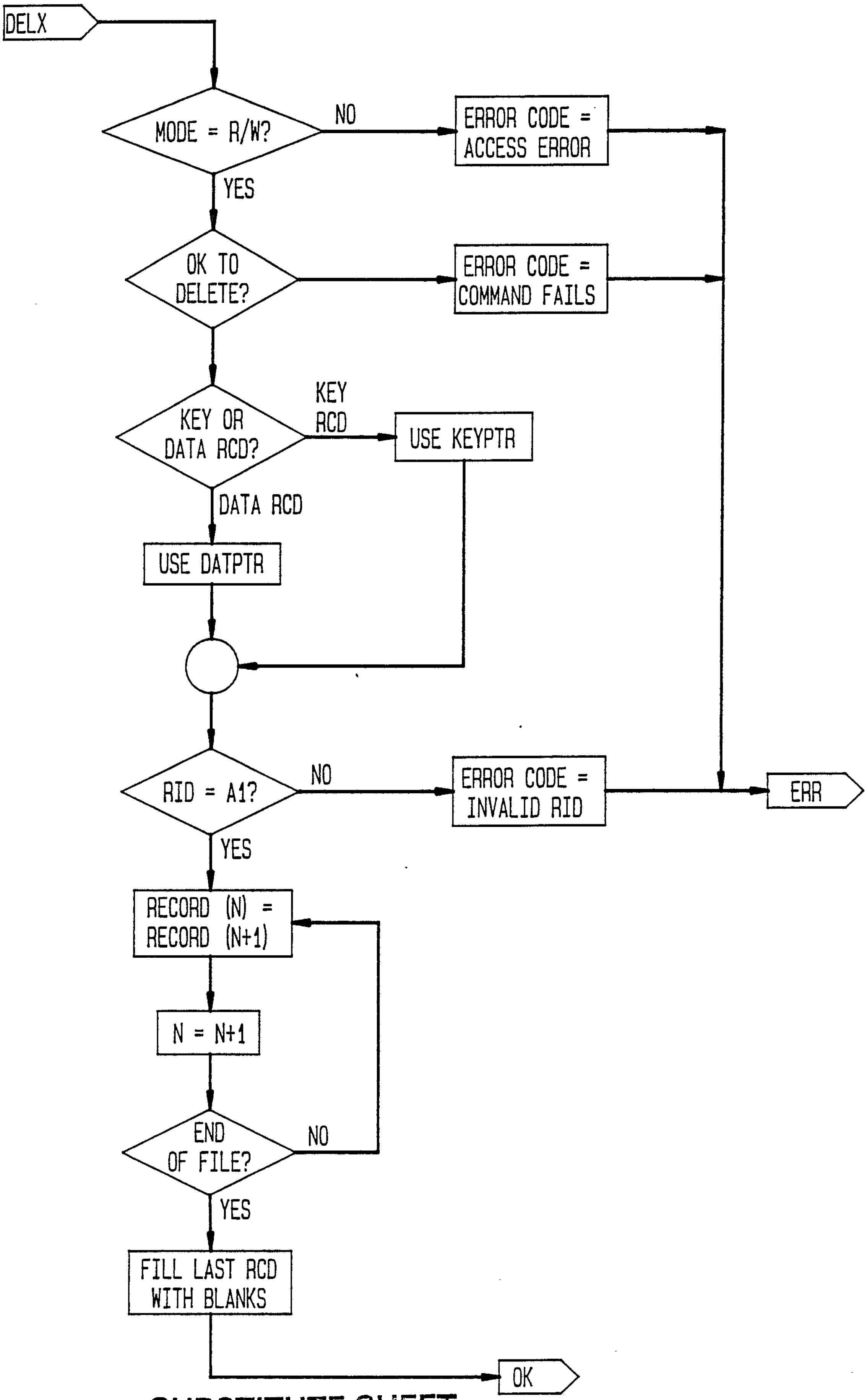

DELX
MODE = R/W?
NO
ERROR CODE = ACCESS ERROR
YES
OK TO DELETE?
ERROR CODE = COMMAND FAILS
KEY OR DATA RCD?
KEY RCD
USE KEYPTR
DATA RCD
USE DATPTR
RID = A1?
NO
ERROR CODE = INVALID RID
ERR
YES
RECORD (N) = RECORD (N+1)
N = N+1
END OF FILE?
NO
YES
FILL LAST RCD WITH BLANKS
OK

FIG. 10P

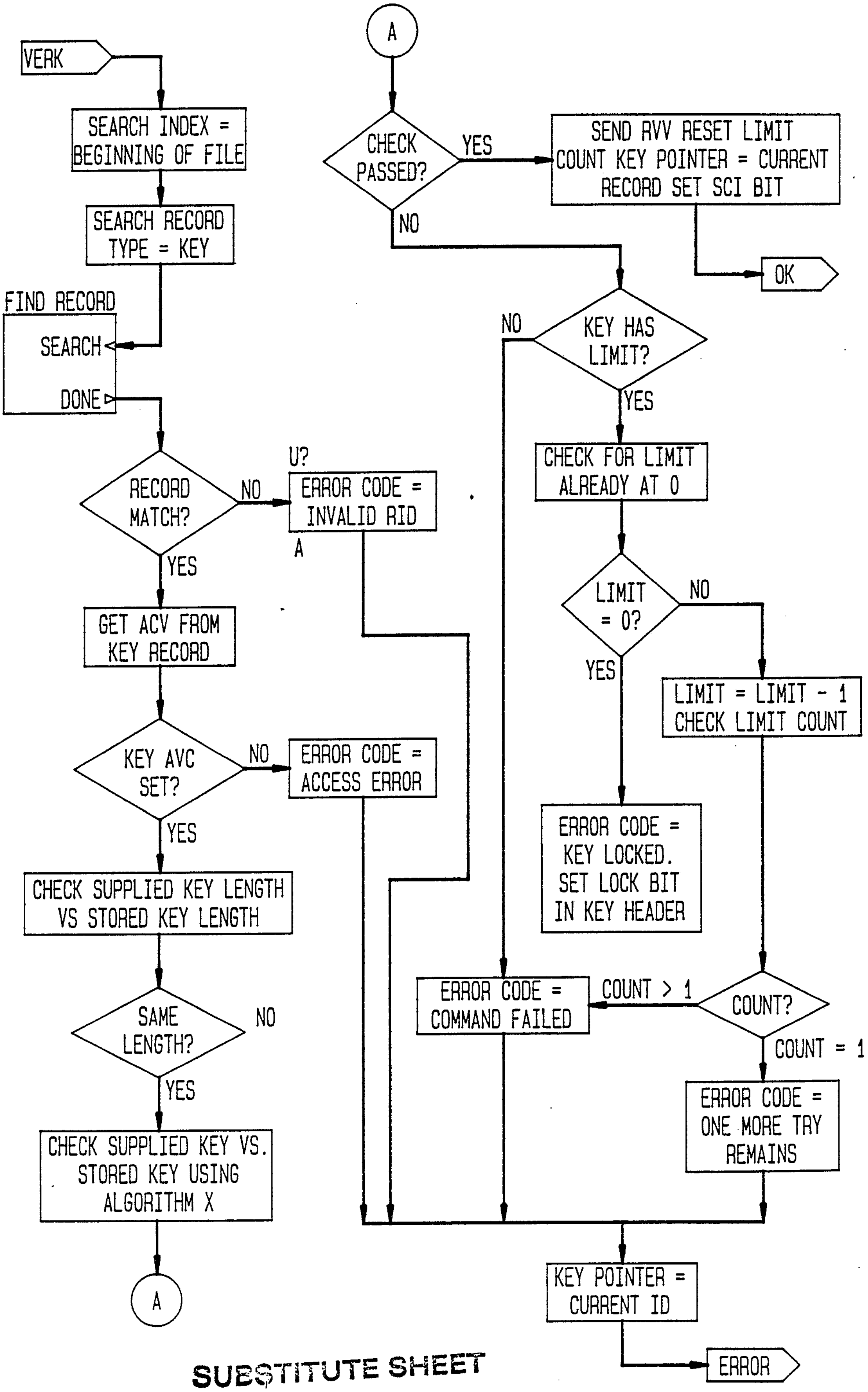

A
VERK
SEARCH INDEX = BEGINNING OF FILE
SEARCH RECORD TYPE = KEY
FIND RECORD
SEARCH
DONE
RECORD MATCH?
NO
YES
U?
ERROR CODE = INVALID RID
A
GET ACV FROM KEY RECORD
KEY AVC SET?
NO
YES
ERROR CODE = ACCESS ERROR
CHECK SUPPLIED KEY LENGTH VS STORED KEY LENGTH
SAME LENGTH?
NO
YES
CHECK SUPPLIED KEY VS. STORED KEY USING ALGORITHM X
A
CHECK PASSED?
YES
NO
SEND RVV RESET LIMIT COUNT KEY POINTER = CURRENT RECORD SET SCI BIT
OK
KEY HAS LIMIT?
NO
YES
CHECK FOR LIMIT ALREADY AT 0
LIMIT = 0?
NO
YES
LIMIT = LIMIT - 1 CHECK LIMIT COUNT
ERROR CODE = KEY LOCKED. SET LOCK BIT IN KEY HEADER
ERROR CODE = COMMAND FAILED
COUNT > 1
COUNT?
COUNT = 1
ERROR CODE = ONE MORE TRY REMAINS
KEY POINTER = CURRENT ID
ERROR

FIG. 10Q

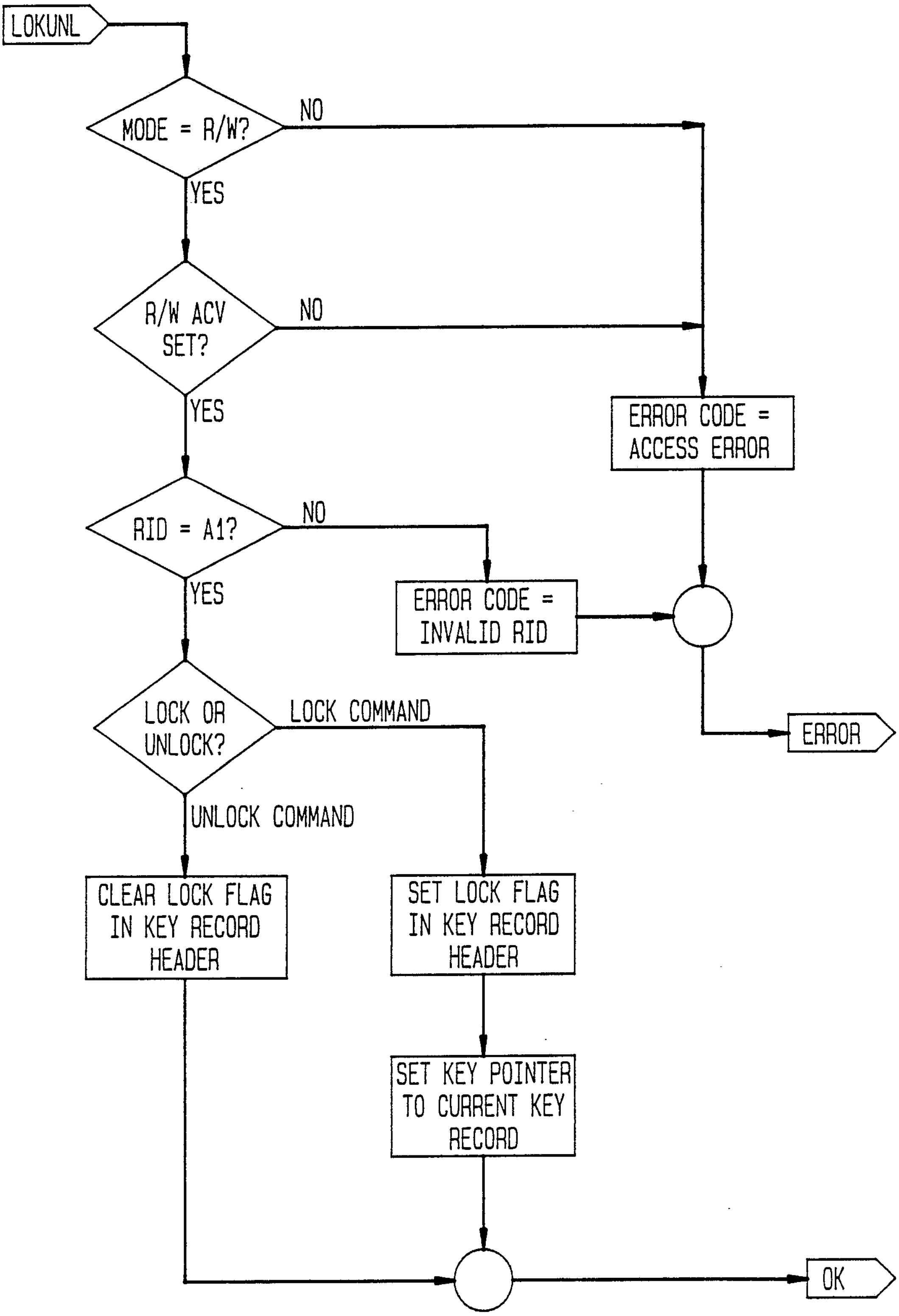

LOKUNL
MODE = R/W?
NO
YES
R/W ACV SET?
NO
YES
ERROR CODE = ACCESS ERROR
RID = A1?
NO
YES
ERROR CODE = INVALID RID
LOCK OR UNLOCK?
LOCK COMMAND
UNLOCK COMMAND
ERROR
CLEAR LOCK FLAG IN KEY RECORD HEADER
SET LOCK FLAG IN KEY RECORD HEADER
SET KEY POINTER TO CURRENT KEY RECORD
OK

FIG. 10R

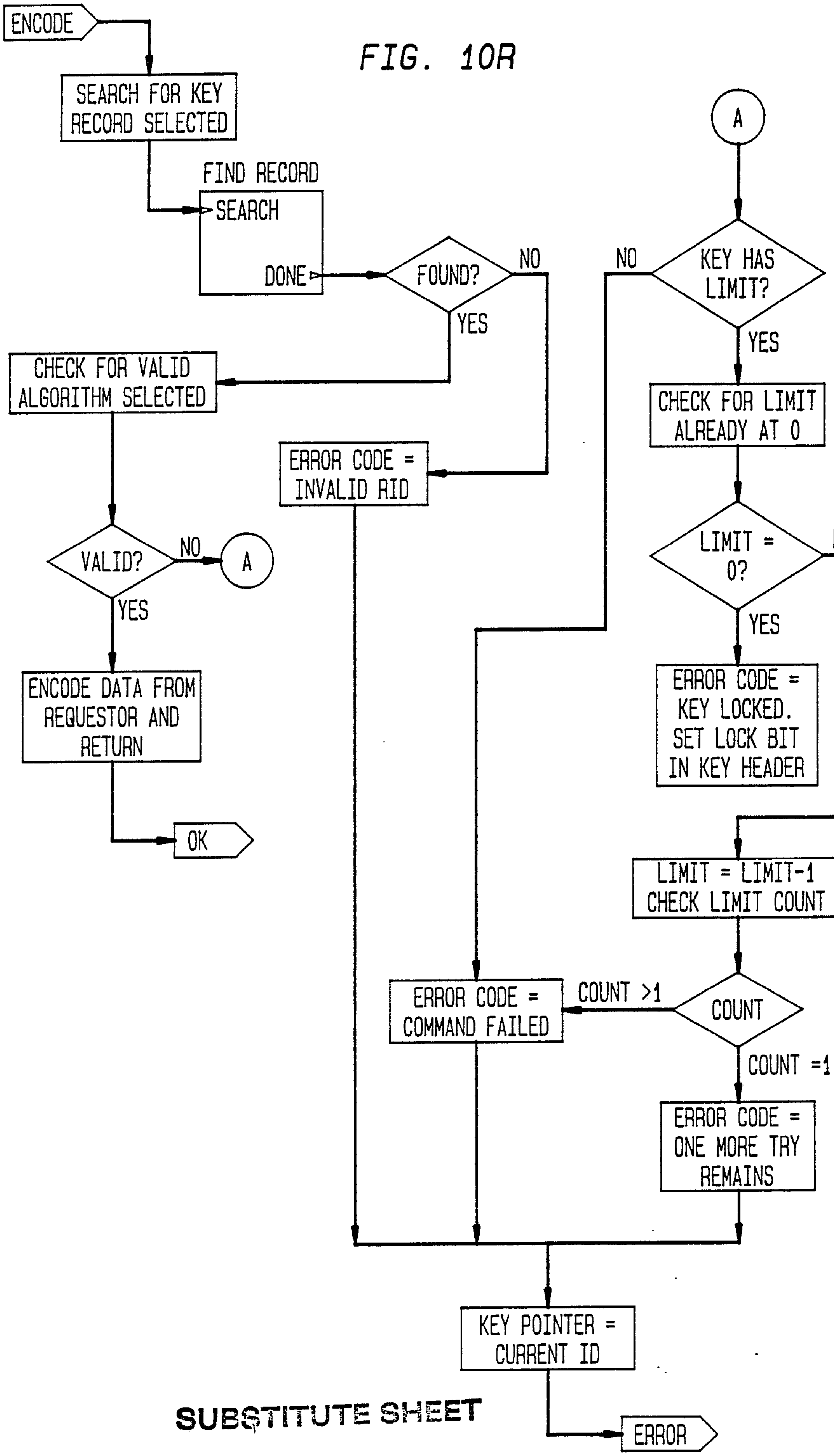

ENCODE
SEARCH FOR KEY RECORD SELECTED
FIND RECORD
SEARCH
DONE
FOUND?
NO
YES
CHECK FOR VALID ALGORITHM SELECTED
ERROR CODE = INVALID RID
VALID?
NO
A
YES
ENCODE DATA FROM REQUESTOR AND RETURN
OK
A
KEY HAS LIMIT?
NO
YES
CHECK FOR LIMIT ALREADY AT 0
LIMIT = 0?
NO
YES
ERROR CODE = KEY LOCKED. SET LOCK BIT IN KEY HEADER
LIMIT = LIMIT-1 CHECK LIMIT COUNT
COUNT
COUNT >1
COUNT =1
ERROR CODE = COMMAND FAILED
ERROR CODE = ONE MORE TRY REMAINS
KEY POINTER = CURRENT ID
ERROR

FIG. 10S

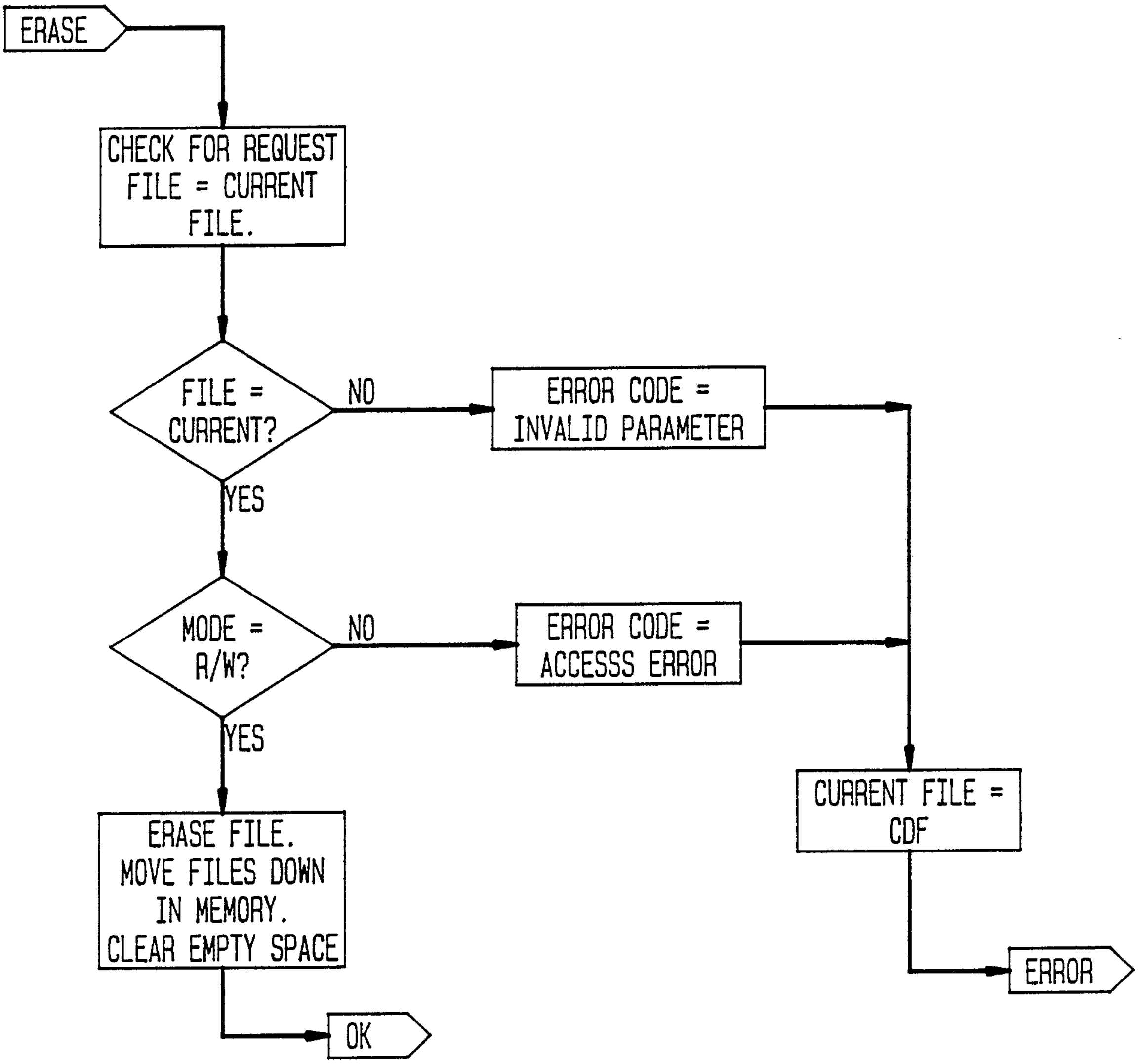

ERASE
CHECK FOR REQUEST FILE = CURRENT FILE.
FILE = CURRENT?
NO
ERROR CODE = INVALID PARAMETER
YES
MODE = R/W?
NO
ERROR CODE = ACCESSS ERROR
YES
ERASE FILE. MOVE FILES DOWN IN MEMORY. CLEAR EMPTY SPACE
CURRENT FILE = CDF
OK
ERROR

FIG. 10T

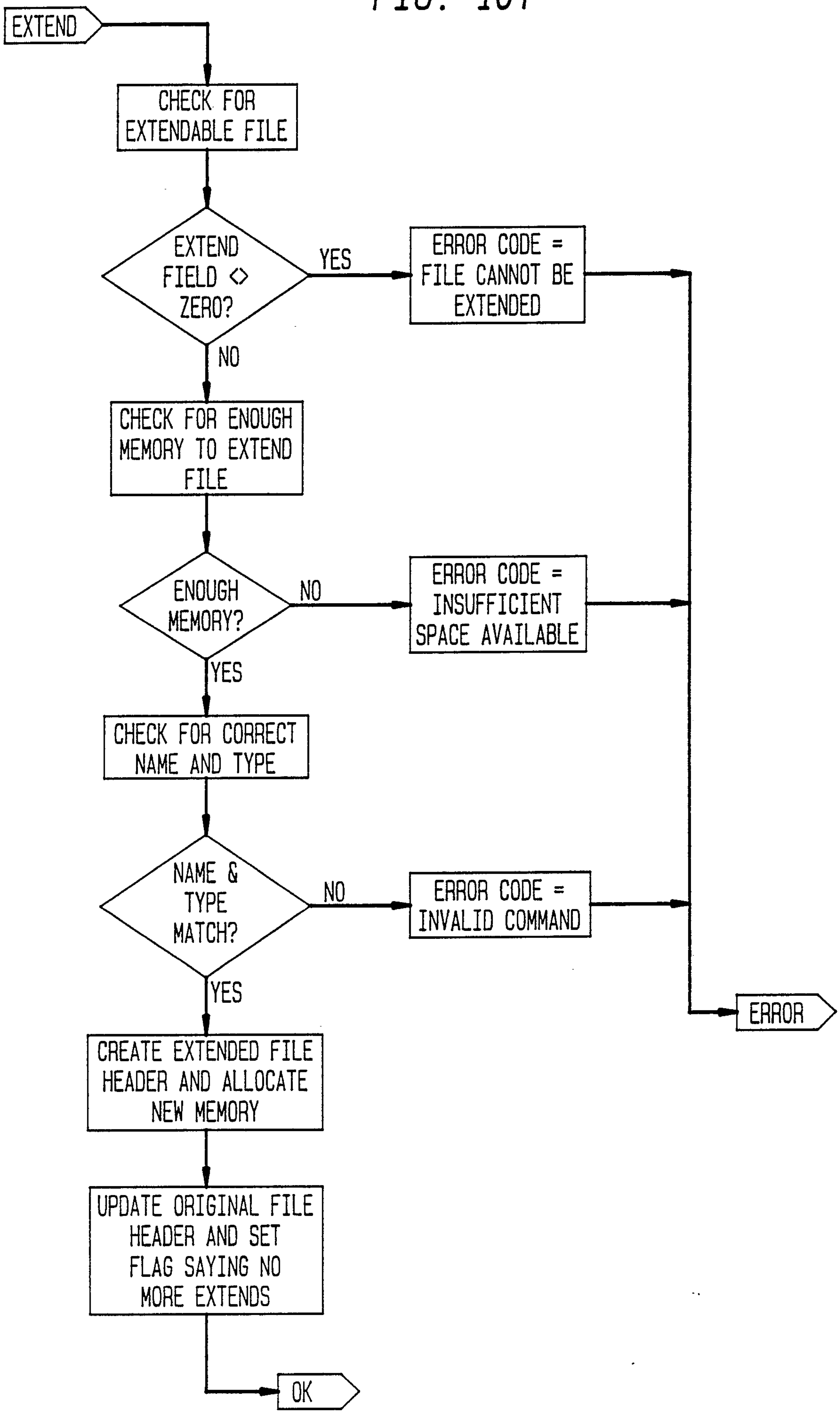

EXTEND
CHECK FOR EXTENDABLE FILE
EXTEND FIELD <> ZERO?
YES
ERROR CODE = FILE CANNOT BE EXTENDED
NO
CHECK FOR ENOUGH MEMORY TO EXTEND FILE
ENOUGH MEMORY?
NO
ERROR CODE = INSUFFICIENT SPACE AVAILABLE
YES
CHECK FOR CORRECT NAME AND TYPE
NAME & TYPE MATCH?
NO
ERROR CODE = INVALID COMMAND
YES
ERROR
CREATE EXTENDED FILE HEADER AND ALLOCATE NEW MEMORY
UPDATE ORIGINAL FILE HEADER AND SET FLAG SAYING NO MORE EXTENDS
OK

FIG. 10U

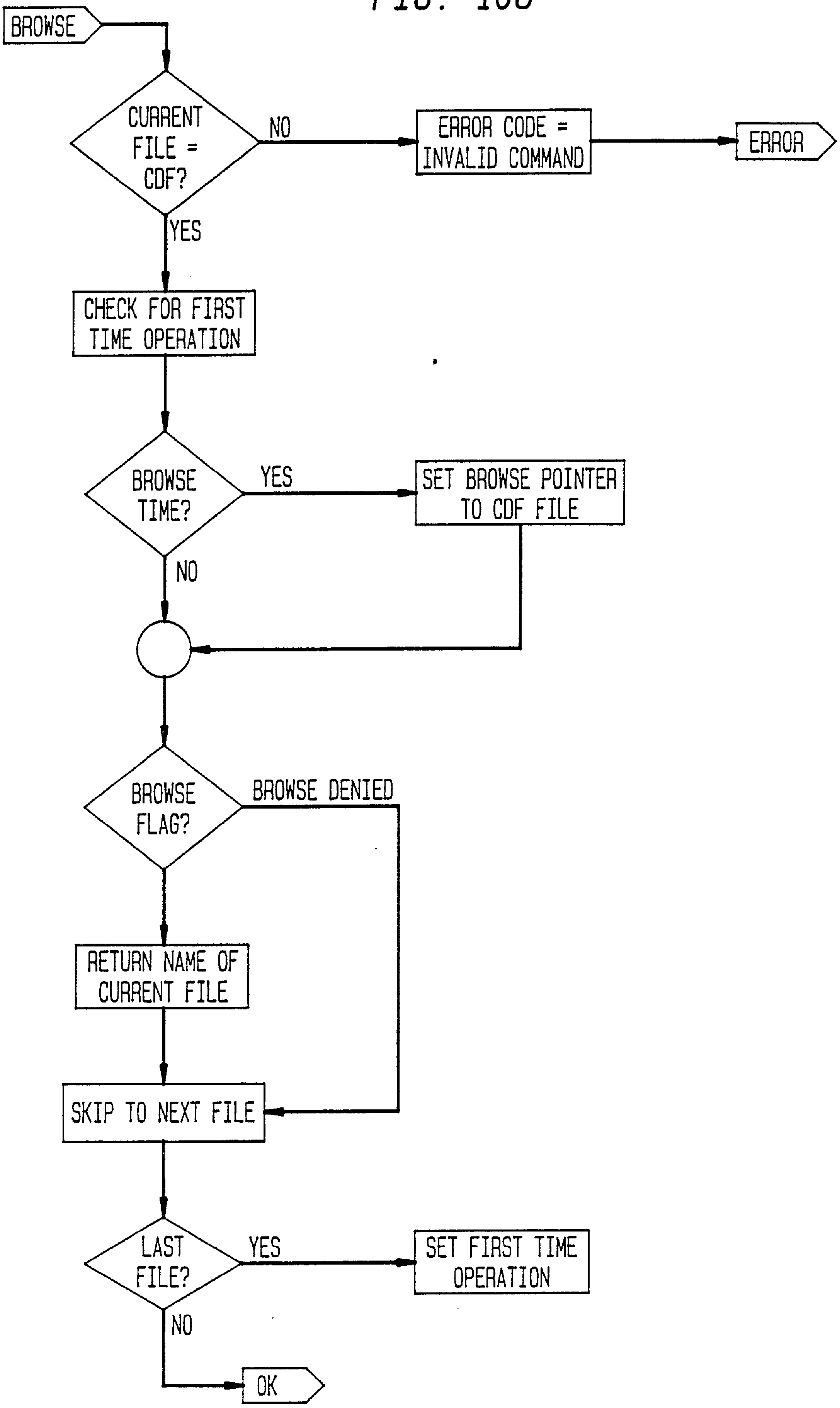

BROWSE
CURRENT FILE = CDF?
NO
ERROR CODE = INVALID COMMAND
ERROR
YES
CHECK FOR FIRST TIME OPERATION
BROWSE TIME?
YES
SET BROWSE POINTER TO CDF FILE
NO
BROWSE FLAG?
BROWSE DENIED
RETURN NAME OF CURRENT FILE
SKIP TO NEXT FILE
LAST FILE?
YES
SET FIRST TIME OPERATION
NO
OK

FIG. 10V

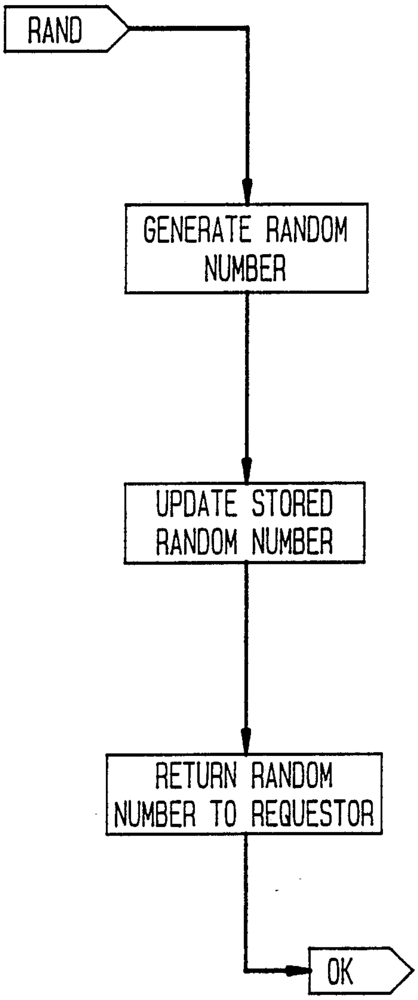

RAND
GENERATE RANDOM NUMBER
UPDATE STORED RANDOM NUMBER
RETURN RANDOM NUMBER TO REQUESTOR
OK

FIG. 10W

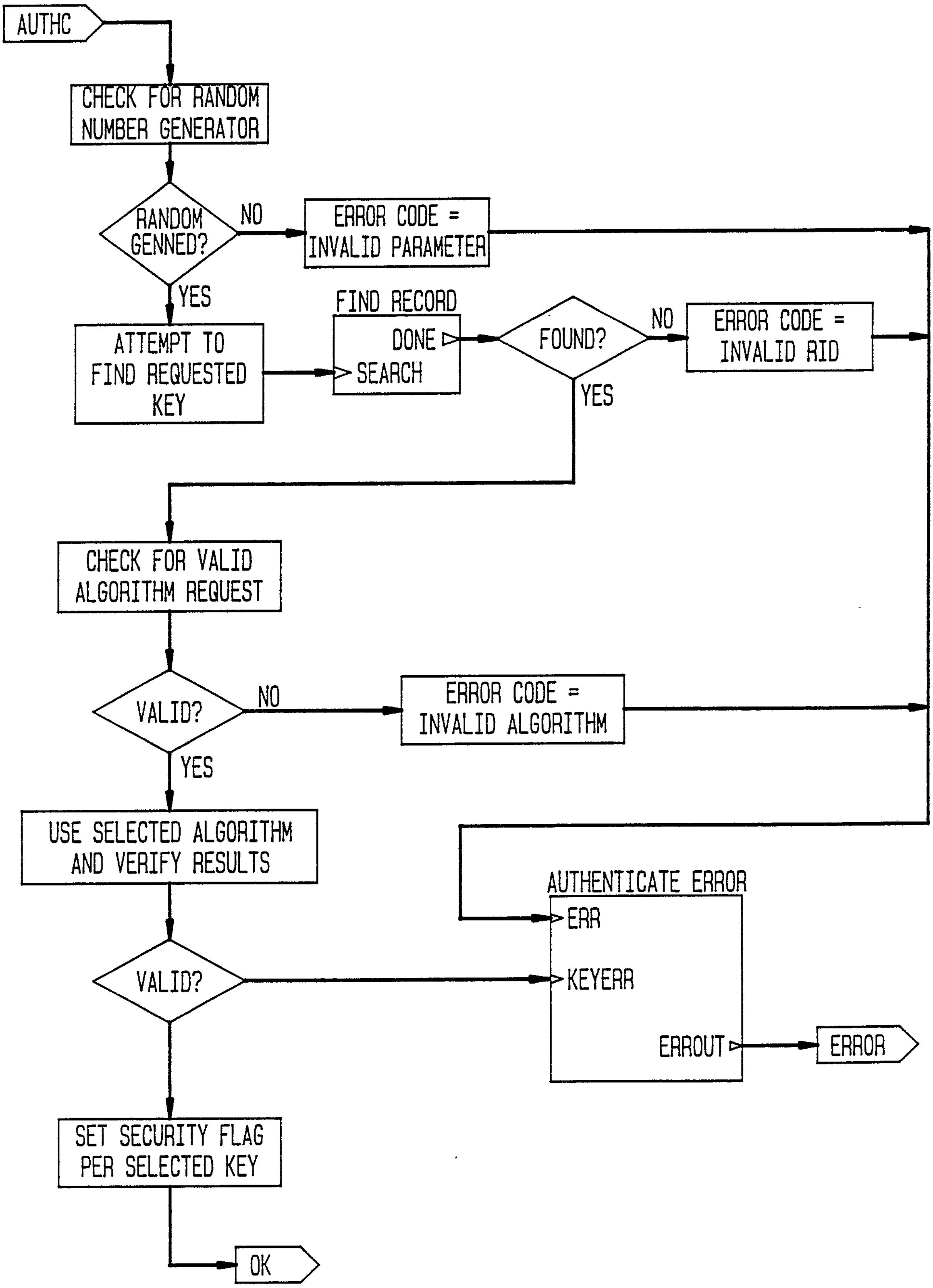

AUTHC
CHECK FOR RANDOM NUMBER GENERATOR
RANDOM GENNED?
NO
ERROR CODE = INVALID PARAMETER
YES
ATTEMPT TO FIND REQUESTED KEY
FIND RECORD
DONE
SEARCH
FOUND?
NO
ERROR CODE = INVALID RID
YES
CHECK FOR VALID ALGORITHM REQUEST
VALID?
NO
ERROR CODE = INVALID ALGORITHM
YES
USE SELECTED ALGORITHM AND VERIFY RESULTS
AUTHENTICATE ERROR
ERR
KEYERR
ERROUT
ERROR
VALID?
SET SECURITY FLAG PER SELECTED KEY
OK

FIG. 10X

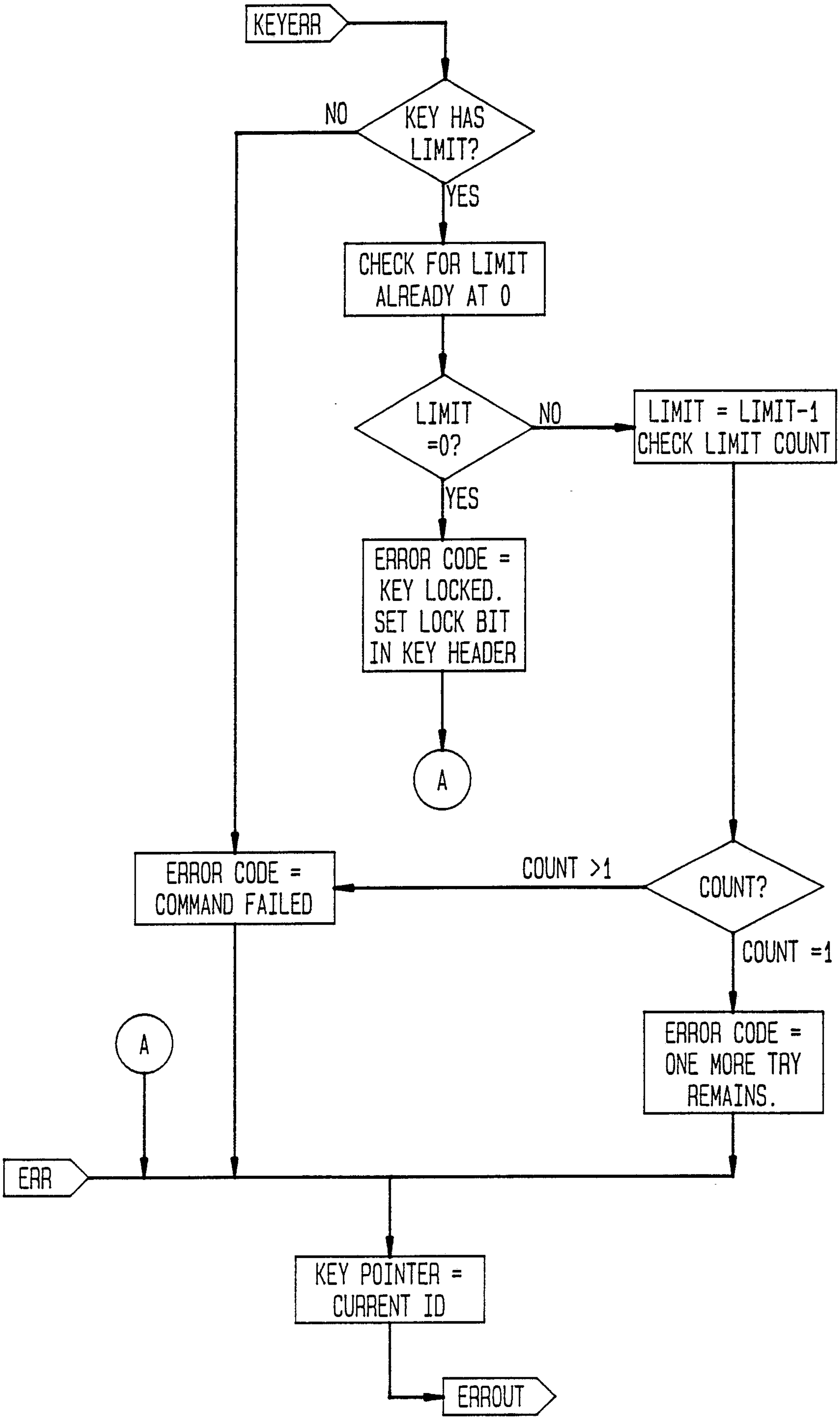

KEYERR
KEY HAS LIMIT?
NO
YES
CHECK FOR LIMIT ALREADY AT 0
LIMIT =0?
NO
YES
LIMIT = LIMIT-1 CHECK LIMIT COUNT
ERROR CODE = KEY LOCKED. SET LOCK BIT IN KEY HEADER
A
ERROR CODE = COMMAND FAILED
COUNT >1
COUNT?
COUNT =1
ERROR CODE = ONE MORE TRY REMAINS.
A
ERR
KEY POINTER = CURRENT ID
ERROUT

FIG. 10Y

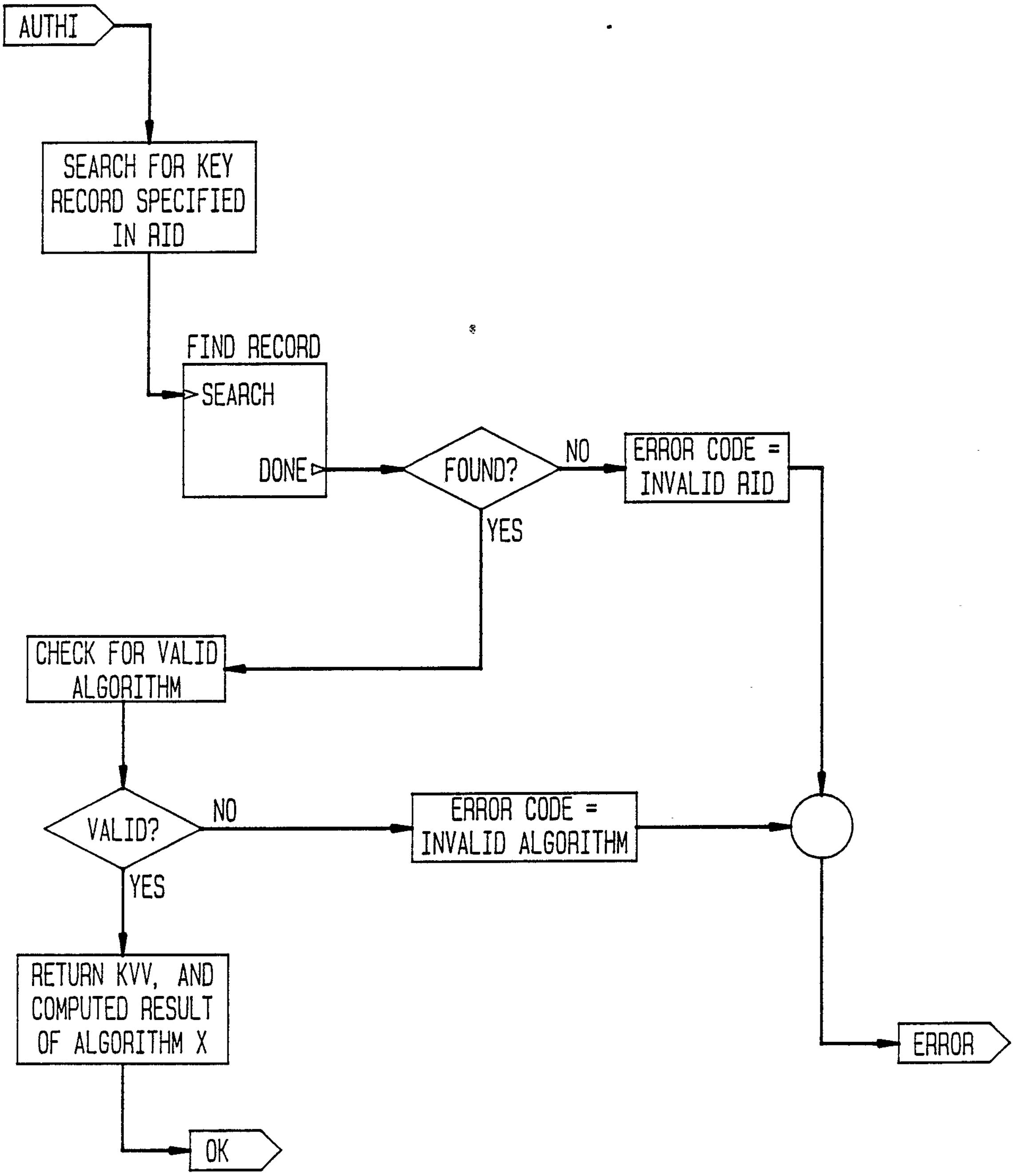

AUTHI
SEARCH FOR KEY RECORD SPECIFIED IN RID
FIND RECORD
SEARCH
DONE
FOUND?
NO
ERROR CODE = INVALID RID
YES
CHECK FOR VALID ALGORITHM
VALID?
NO
ERROR CODE = INVALID ALGORITHM
YES
RETURN KVV, AND COMPUTED RESULT OF ALGORITHM X
OK
ERROR

FIG. 10Z

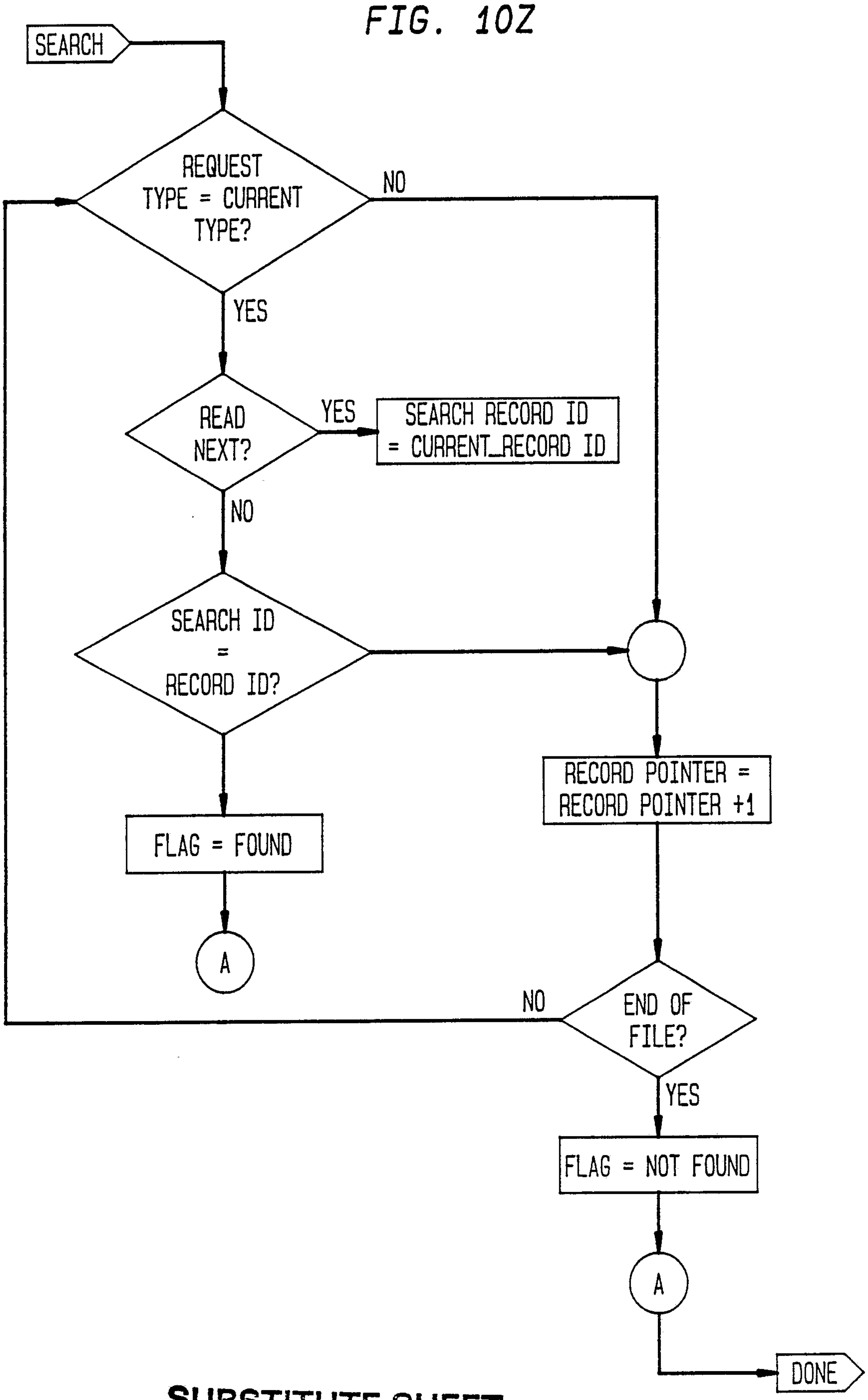

SEARCH
REQUEST TYPE = CURRENT TYPE?
NO
YES
READ NEXT?
YES
SEARCH RECORD ID = CURRENT_RECORD ID
NO
SEARCH ID = RECORD ID?
FLAG = FOUND
A
RECORD POINTER = RECORD POINTER +1
NO
END OF FILE?
YES
FLAG = NOT FOUND
A
DONE

FIG. 11

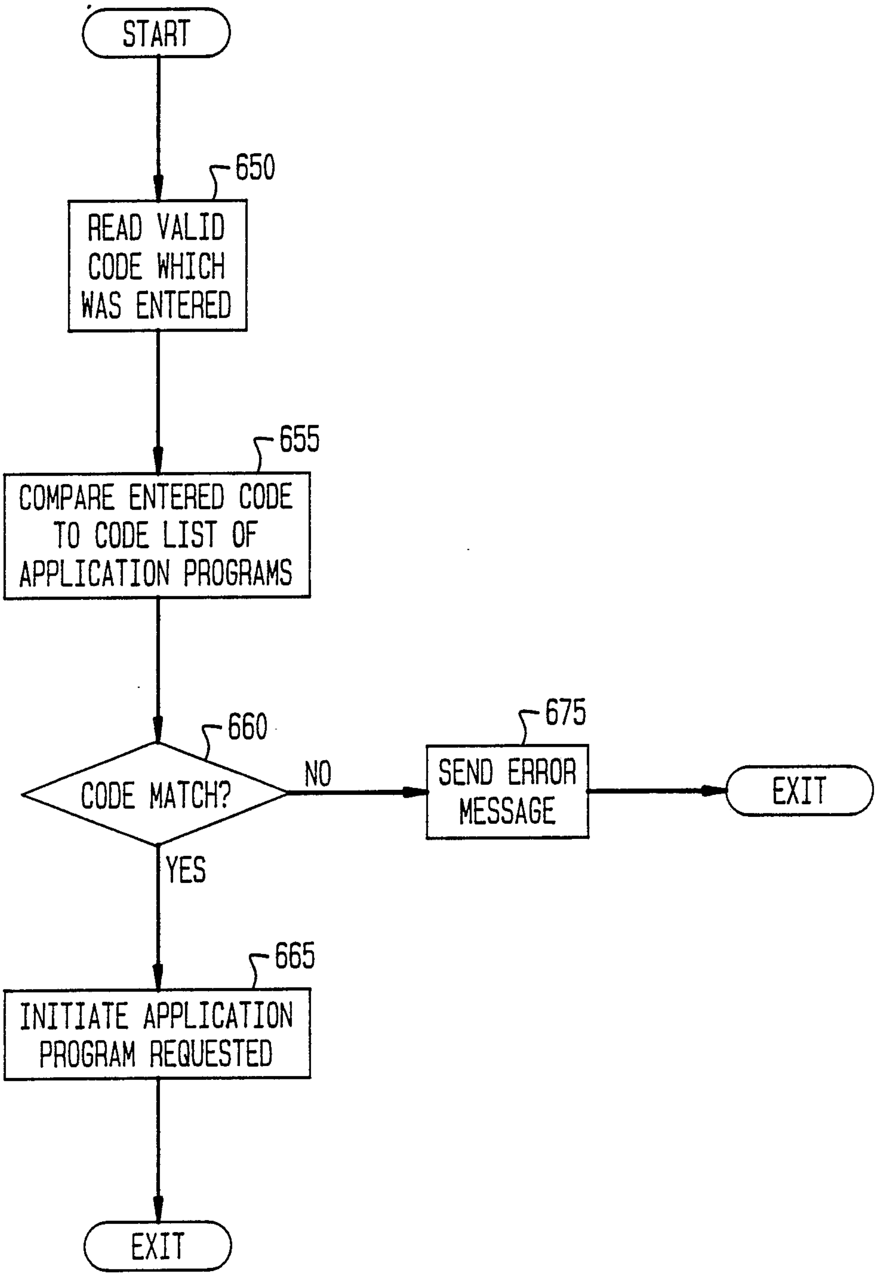

START
650
READ VALID CODE WHICH WAS ENTERED
655
COMPARE ENTERED CODE TO CODE LIST OF APPLICATION PROGRAMS
660
CODE MATCH?
NO
675
SEND ERROR MESSAGE
EXIT
YES
665
INITIATE APPLICATION PROGRAM REQUESTED
EXIT

FIG. 12

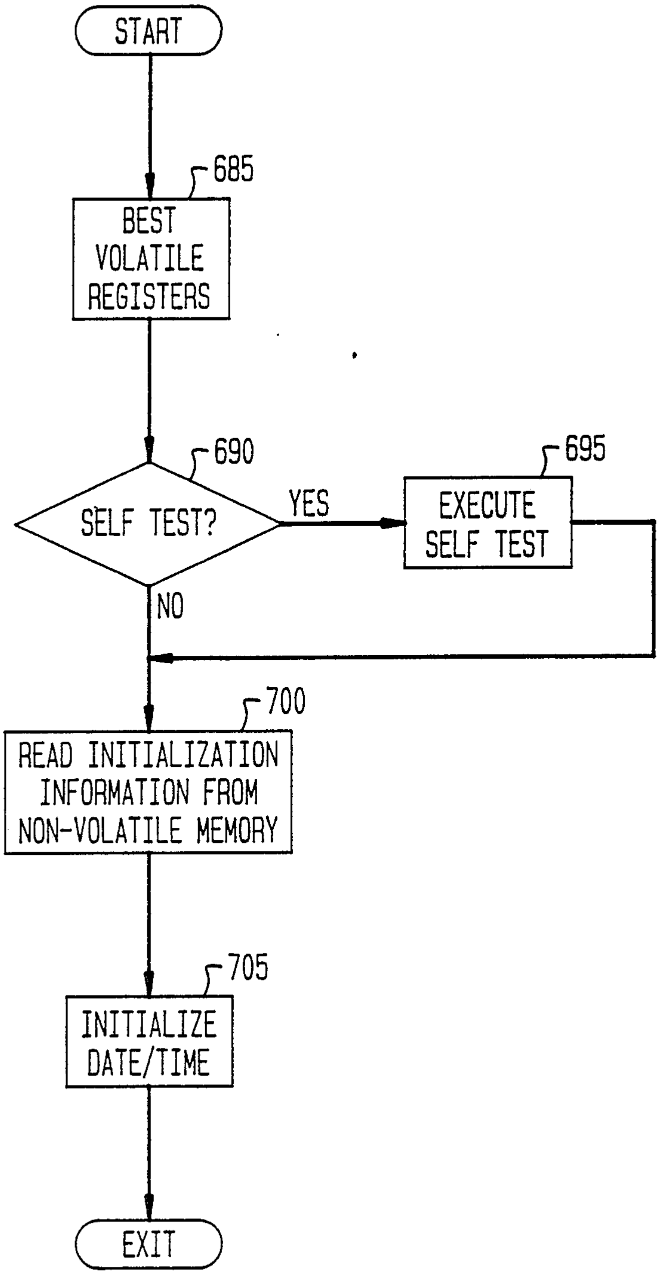

START
685
BEST VOLATILE REGISTERS
690
SELF TEST?
YES
NO
695
EXECUTE SELF TEST
700
READ INITIALIZATION INFORMATION FROM NON-VOLATILE MEMORY
705
INITIALIZE DATE/TIME
EXIT

FIG. 13

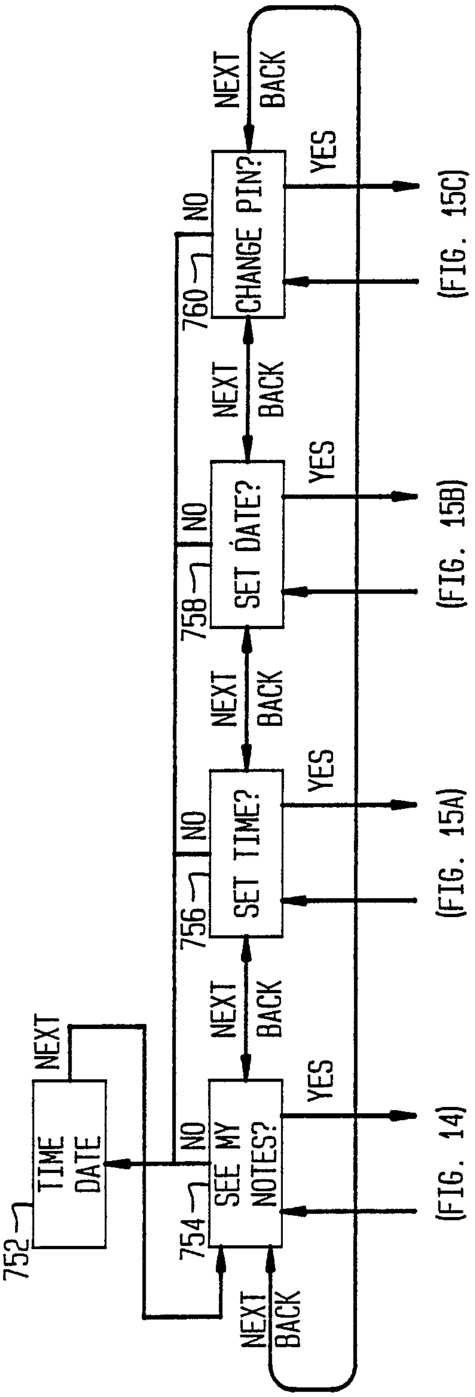
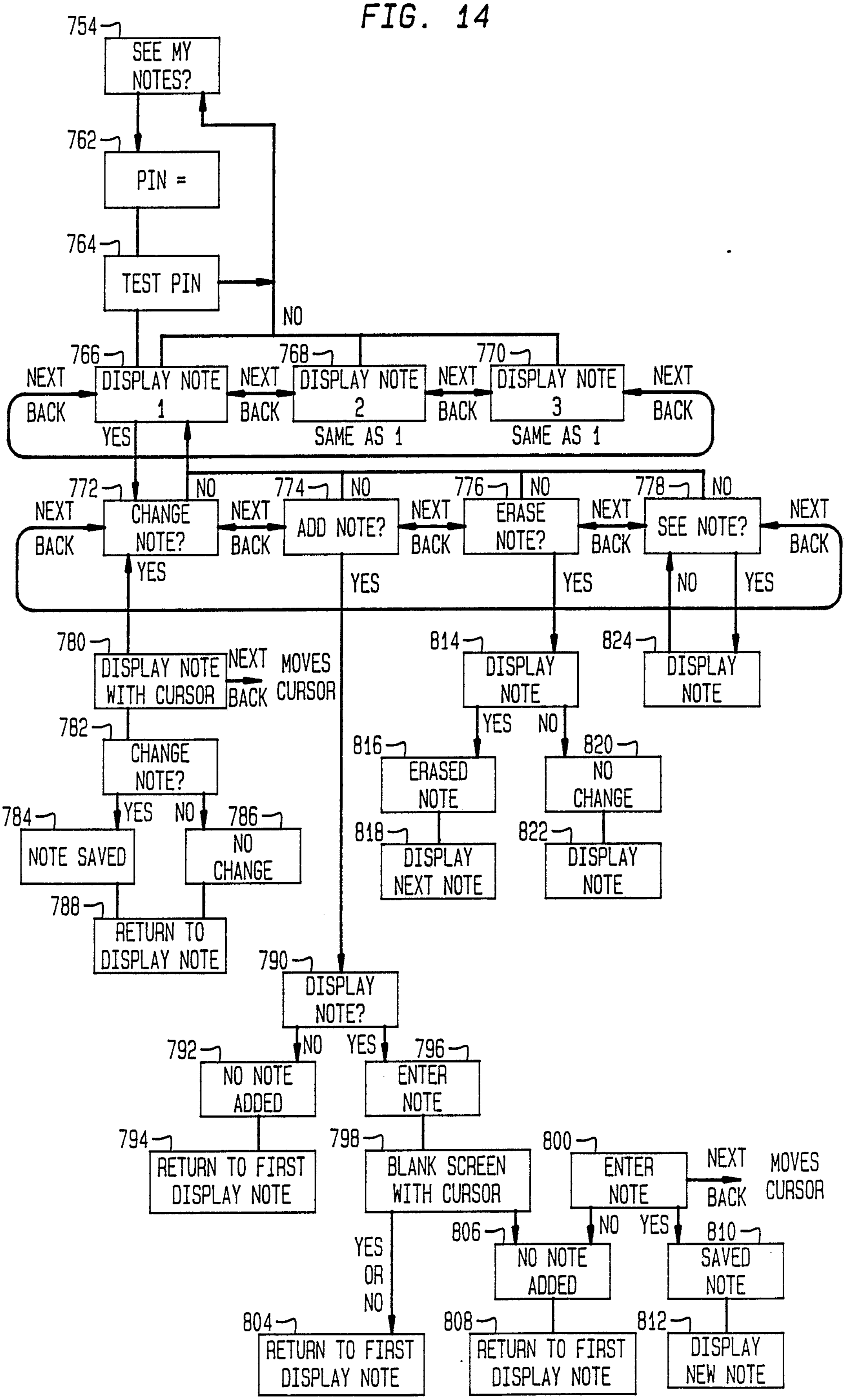

752
TIME
DATE
NEXT
754
SEE MY
NOTES?
NO
YES
(FIG. 14)
NEXT
BACK
756
SET TIME?
NO
YES
(FIG. 15A)
NEXT
BACK
758
SET DATE?
NO
YES
(FIG. 15B)
NEXT
BACK
760
CHANGE PIN?
NO
YES
(FIG. 15C)
NEXT
BACK

FIG. 14



SUBSTITUTE SHEET

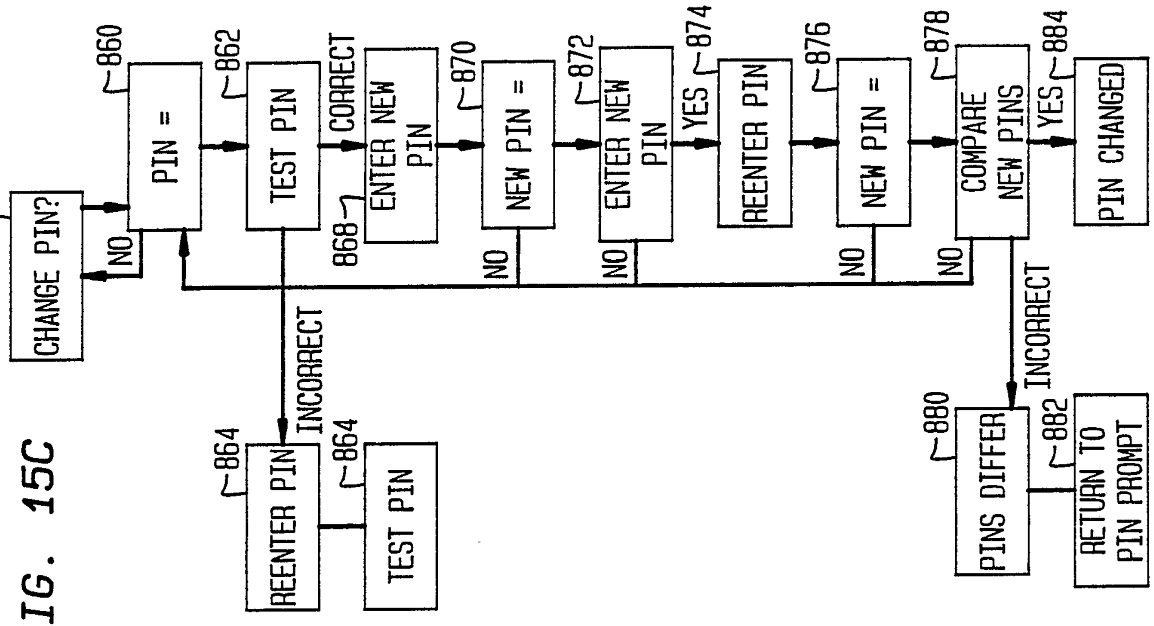


FIG. 15C

FIG. 15B

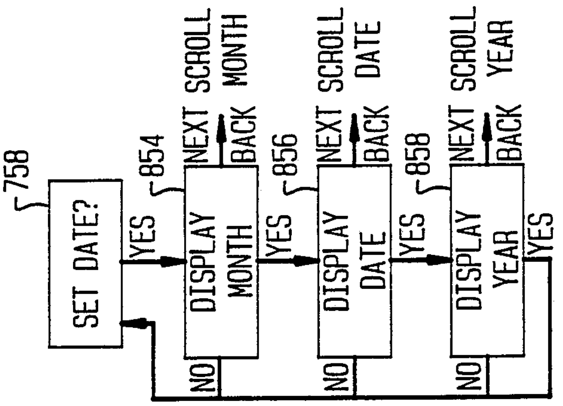


FIG. 15A

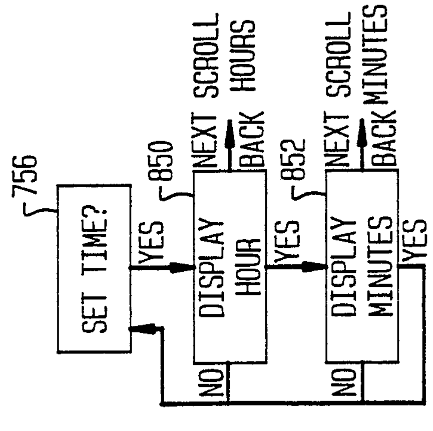


FIG. 16

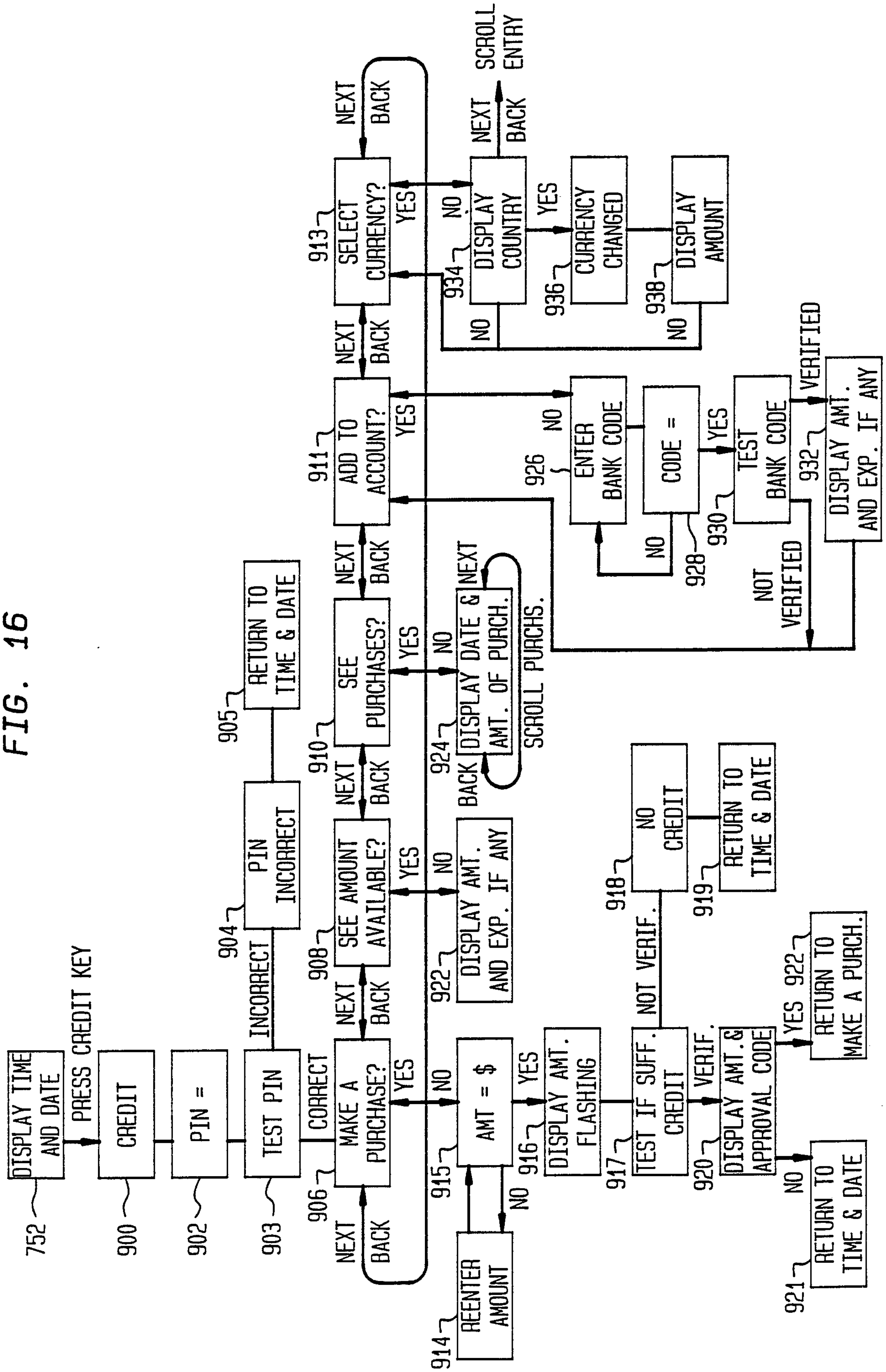
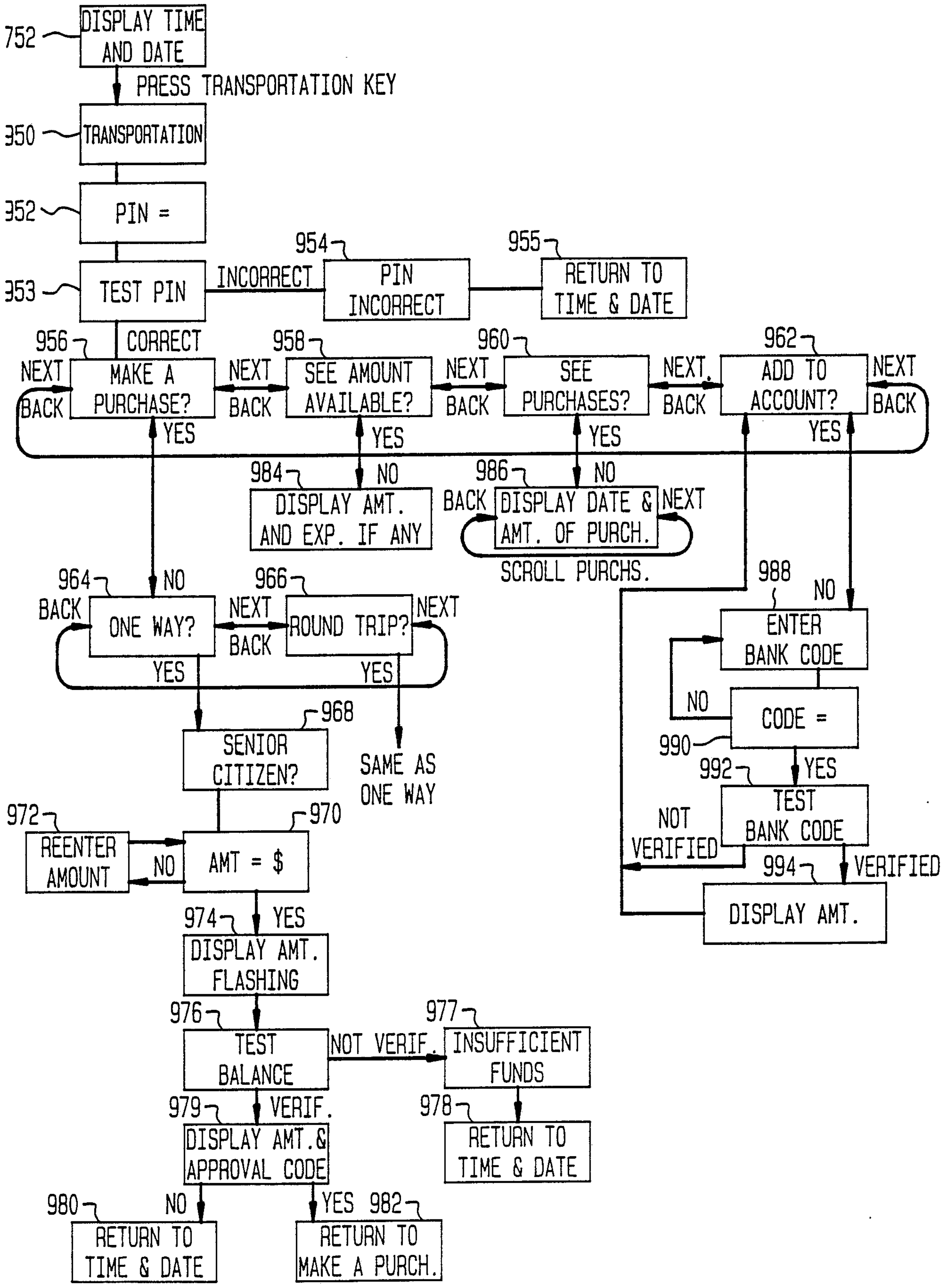
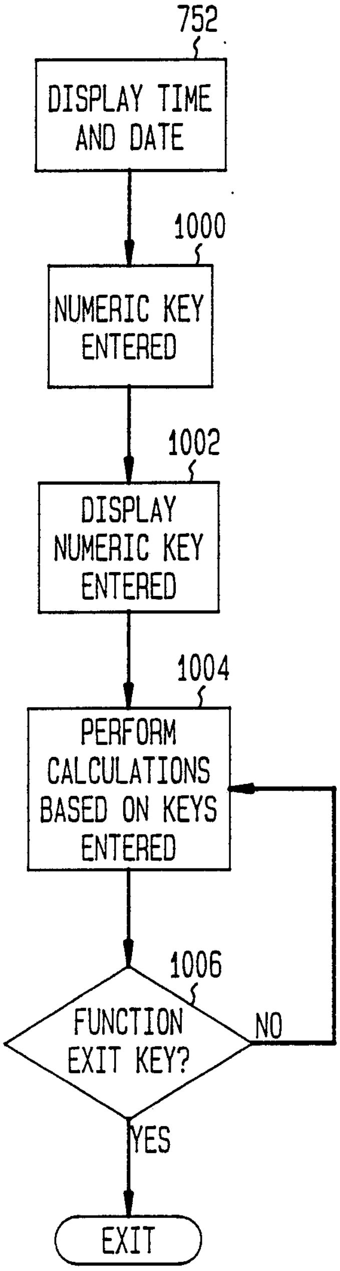

752
DISPLAY TIME AND DATE
PRESS CREDIT KEY
900
CREDIT
902
PIN =
903
TEST PIN
CORRECT
INCORRECT
904
PIN INCORRECT
905
RETURN TO TIME & DATE
906
MAKE A PURCHASE?
NEXT
BACK
YES
NO
908
SEE AMOUNT AVAILABLE?
910
SEE PURCHASES?
911
ADD TO ACCOUNT?
913
SELECT CURRENCY?
914
REENTER AMOUNT
915
AMT = $
916
DISPLAY AMT. FLASHING
917
TEST IF SUFF. CREDIT
VERIF.
NOT VERIF.
918
NO CREDIT
919
RETURN TO TIME & DATE
920
DISPLAY AMT. & APPROVAL CODE
921
RETURN TO TIME & DATE
922
RETURN TO MAKE A PURCH.
922
DISPLAY AMT. AND EXP. IF ANY
924
DISPLAY DATE & AMT. OF PURCH.
SCROLL PURCHS.
926
ENTER BANK CODE
928
CODE =
930
TEST BANK CODE
VERIFIED
NOT VERIFIED
932
DISPLAY AMT. AND EXP. IF ANY
934
DISPLAY COUNTRY
936
CURRENCY CHANGED
938
DISPLAY AMOUNT
SCROLL ENTRY

FIG. 17



752
DISPLAY TIME AND DATE
PRESS TRANSPORTATION KEY
350
TRANSPORTATION
352
PIN =
353
TEST PIN
INCORRECT
954
PIN INCORRECT
955
RETURN TO TIME & DATE
CORRECT
956
NEXT
BACK
MAKE A PURCHASE?
958
NEXT
BACK
SEE AMOUNT AVAILABLE?
960
NEXT
BACK
SEE PURCHASES?
962
NEXT
BACK
ADD TO ACCOUNT?
NEXT
BACK
YES
YES
YES
YES
984
NO
DISPLAY AMT. AND EXP. IF ANY
986
NO
BACK
DISPLAY DATE & AMT. OF PURCH.
NEXT
SCROLL PURCHS.
964
BACK
NO
ONE WAY?
966
NEXT
BACK
ROUND TRIP?
NEXT
YES
YES
968
SENIOR CITIZEN?
SAME AS ONE WAY
972
REENTER AMOUNT
NO
970
AMT = $
974
YES
DISPLAY AMT. FLASHING
976
TEST BALANCE
NOT VERIF.
977
INSUFFICIENT FUNDS
979
VERIF.
DISPLAY AMT.& APPROVAL CODE
978
RETURN TO TIME & DATE
980
NO
RETURN TO TIME & DATE
YES
982
RETURN TO MAKE A PURCH.
988
NO
ENTER BANK CODE
NO
CODE =
990
992
YES
TEST BANK CODE
NOT VERIFIED
994
VERIFIED
DISPLAY AMT.

FIG. 18


752
DISPLAY TIME AND DATE
1000
NUMERIC KEY ENTERED
1002
DISPLAY NUMERIC KEY ENTERED
1004
PERFORM CALCULATIONS BASED ON KEYS ENTERED
1006
FUNCTION EXIT KEY?
NO
YES
EXIT

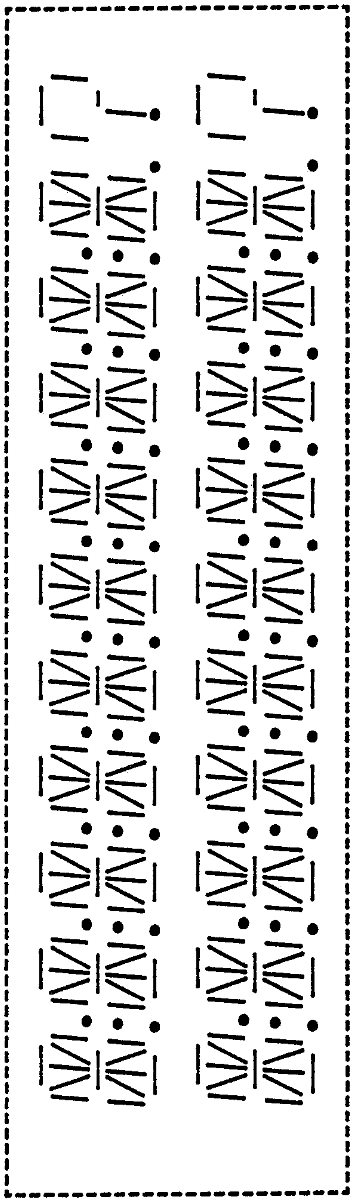


FIG. 19

FIG. 20

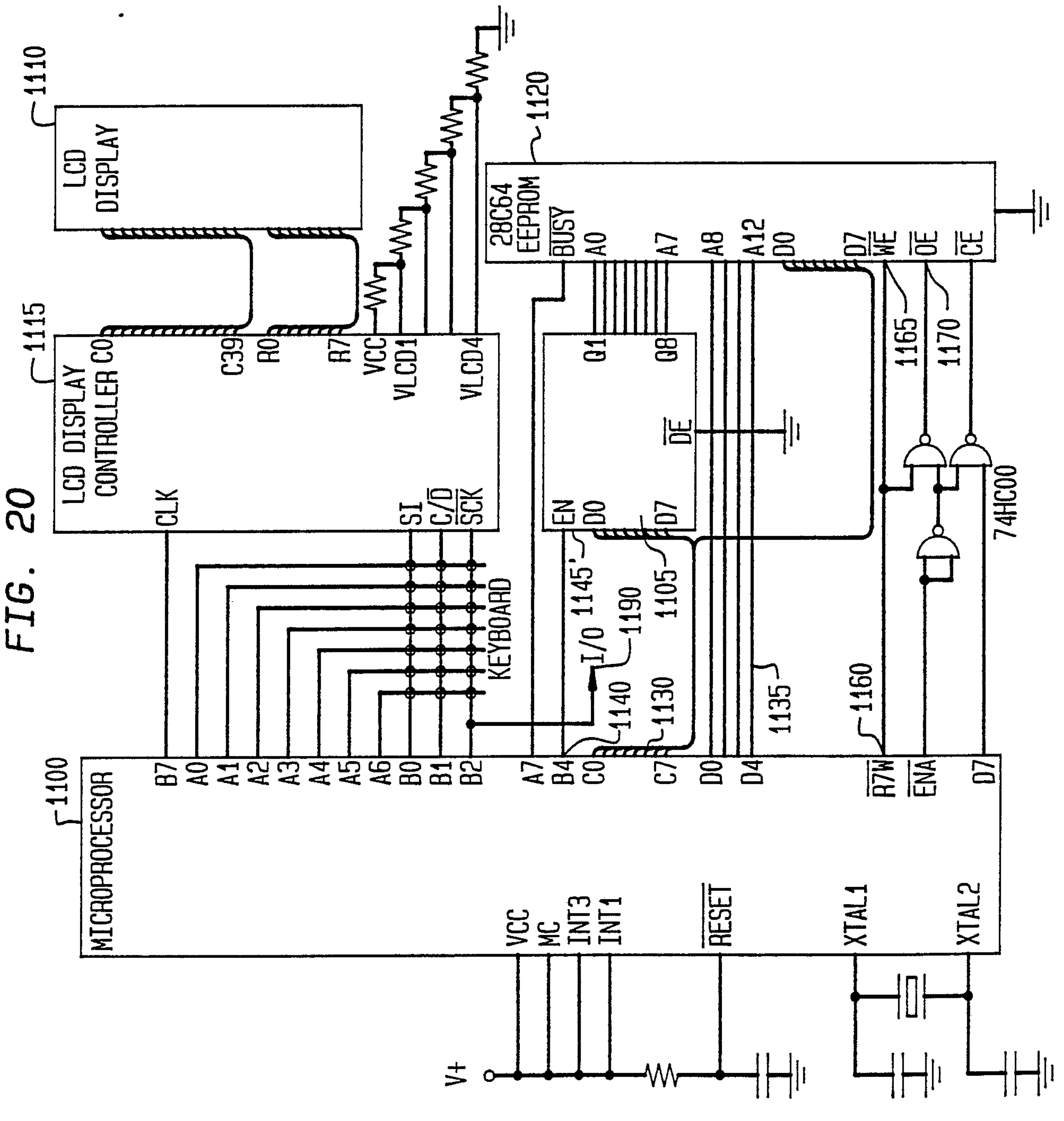

1110
LCD DISPLAY
1115
LCD DISPLAY CONTROLLER
CLK
C0
C39
R0
R7
VCC
VLCD1
VLCD4
SI
C/D
SCK
1120
28C64 EEPROM
BUSY
A0
A7
A8
A12
D0
D7
WE
OE
CE
Q1
Q8
OE
EN
D0
D7
1145'
1190
1105
I/O
1140
1130
1135
1160
1165
1170
74HC00
KEYBOARD
1100
MICROPROCESSOR
B7
A0
A1
A2
A3
A4
A5
A6
B0
B1
B2
A7
B4
C0
C7
D0
D4
R/W
ENA
D7
VCC
MC
INT3
INT1
RESET
XTAL1
XTAL2
V+

FIG. 21

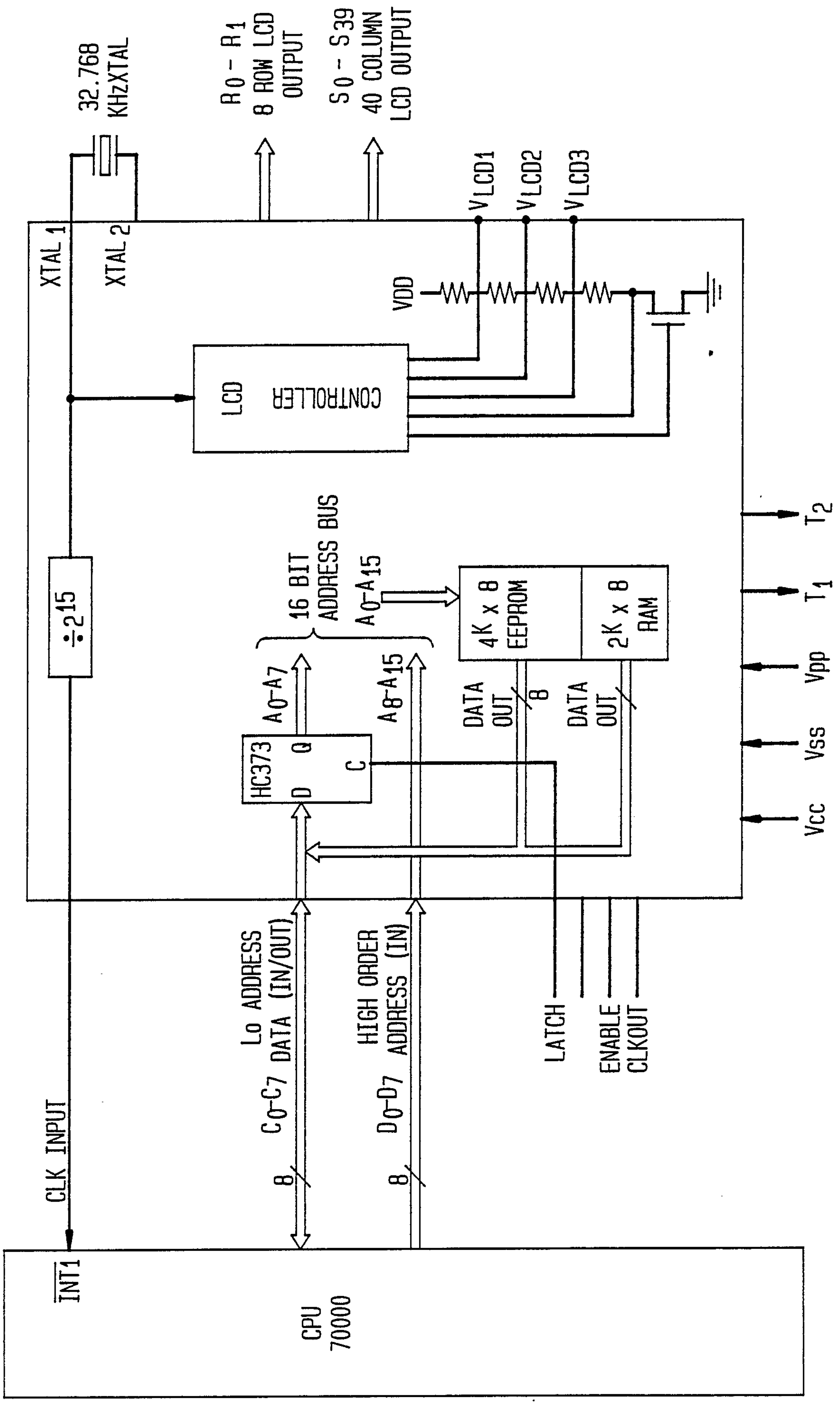

32.768 KHzXTAL
R0 - R1 8 ROW LCD OUTPUT
S0 - S39 40 COLUMN LCD OUTPUT
XTAL 1
XTAL 2
VLCD1
VLCD2
VLCD3
VDD
LCD CONTROLLER
÷2^15
16 BIT ADDRESS BUS A0-A15
4K x 8 EEPROM
2K x 8 RAM
A0-A7
A8-A15
HC373
D
Q
C
DATA OUT
8
DATA OUT
T2
T1
Vpp
Vss
Vcc
Lo ADDRESS C0-C7 DATA (IN/OUT)
HIGH ORDER D0-D7 ADDRESS (IN)
8
8
LATCH
ENABLE
CLKOUT
CLK INPUT
INT1
CPU 70000

FIG. 22A

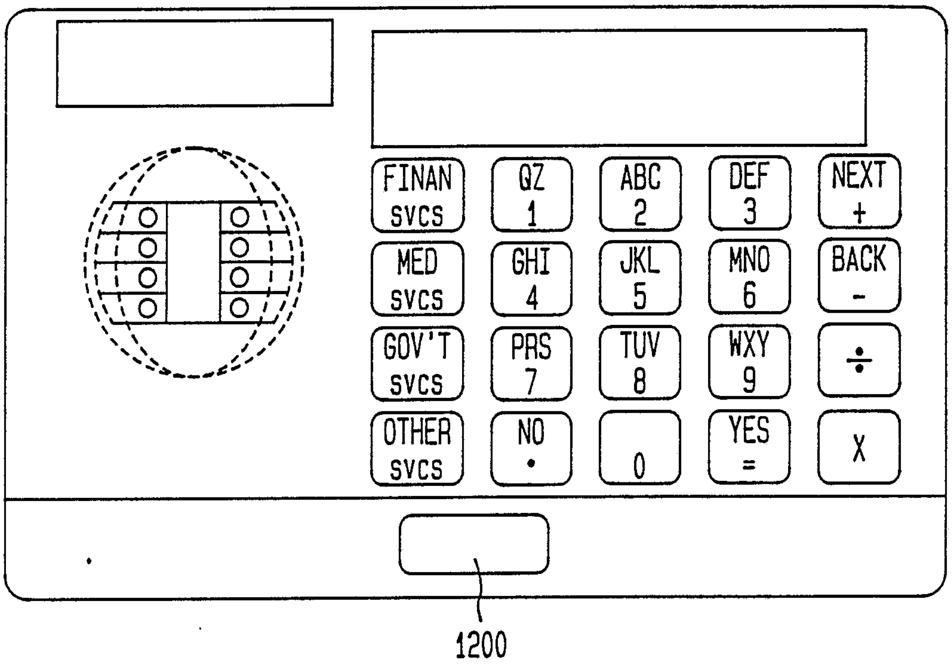


FIG. 22B

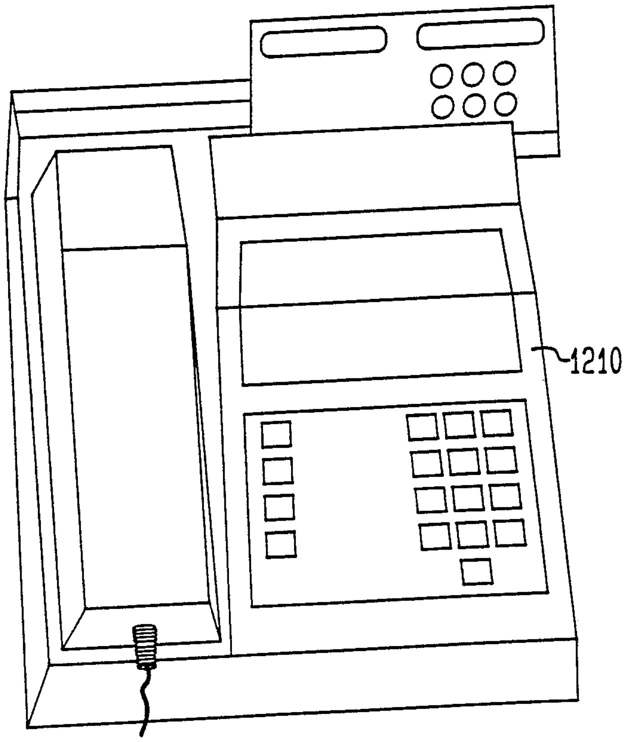


FIG. 23

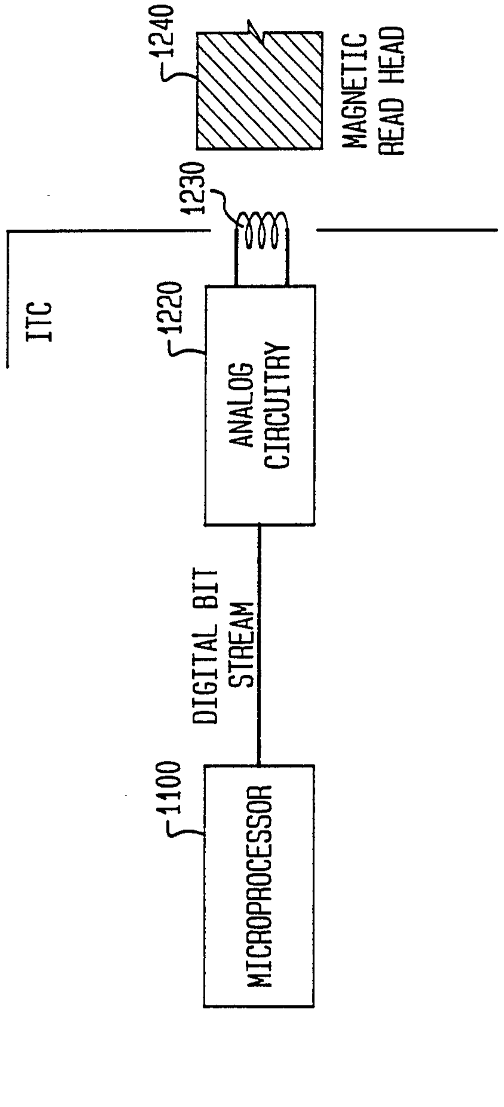

1240
MAGNETIC READ HEAD
1230
ITC
1220
ANALOG CIRCUITRY
DIGITAL BIT STREAM
1100
MICROPROCESSOR

FIG. 24

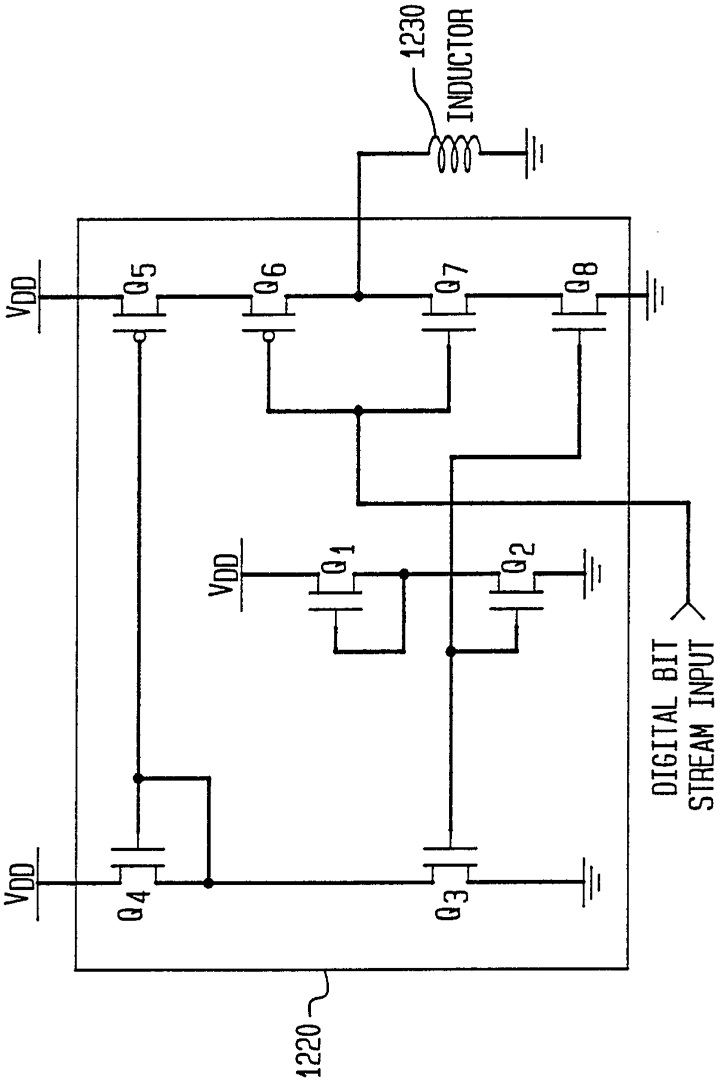
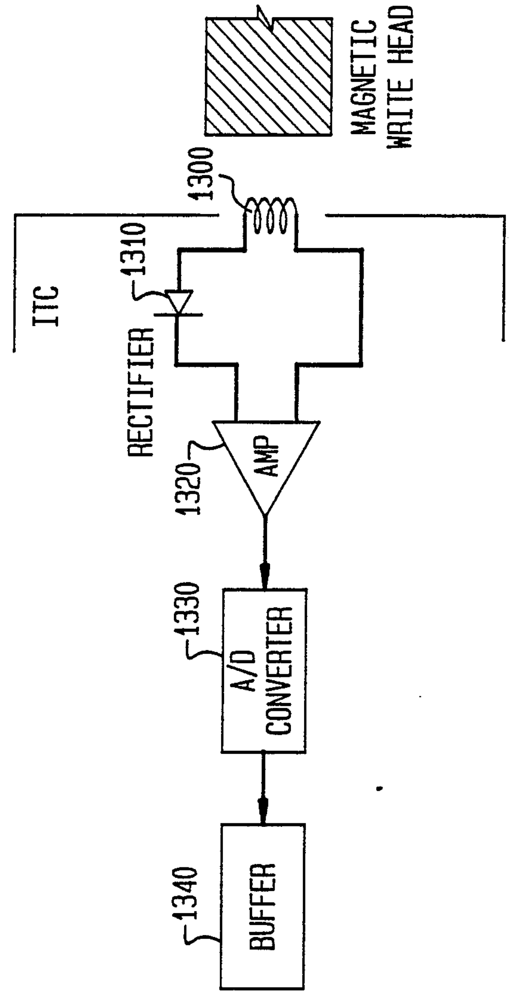

1230
INDUCTOR
VDD
Q5
Q6
Q7
Q8
VDD
Q1
Q2
DIGITAL BIT STREAM INPUT
VDD
Q4
Q3
1220

FIG. 25


MAGNETIC WRITE HEAD
1300
1310
ITC
RECTIFIER
AMP
1320
1330
A/D CONVERTER
1340
BUFFER

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US88/01665

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC (4): G06K 5/00
U.S. CL. 235/380, 492; 340/825.33

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
U.S.	235/492, 487, 493, 379,380, 382, 382.5; 340/825.31, 825.32, 825,33, 825.34

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
Y	US, A, 4,614,861 (PAVLOV ET AL), 30 September 1986 See the entire document.	1-23
Y,P	US, A, 4,701,601 (FRANCINI ET AL), 20 October 1987 See the entire document.	1-23
Y	US, A, 4,298,793 (MELIS ET AL), 3 November 1981 See the abstract.	1-23
Y,P	US, A, 4,677,657 (NAGATA ET AL), 30 June 1987 See the entire document.	14, 23
A,P	US, A, 4,697,072 (KAWANA), 29 September 1987 See Figure 1	—
A,T	US, A, 4,752,678 (RIKUNA), 21 June 1988 See Figures 1 and 2	—
A	US, A, 4,587,409 (NISHIMURA ET AL) 6 May 1986 See the entire document.	—

* Special categories of cited documents: [10]

"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

"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

IV. CERTIFICATION

Date of the Actual Completion of the International Search 26 July 1988	Date of Mailing of this International Search Report 12 SEP 1988
International Searching Authority ISA/US	Signature of Authorized Officer Philip H. Leung Philip H. Leung