OPIC
Office de la propriété intellectuelle du Canada

CIPO
Canadian Intellectual Property Office

(12) (19) (CA) **Demande-Application**

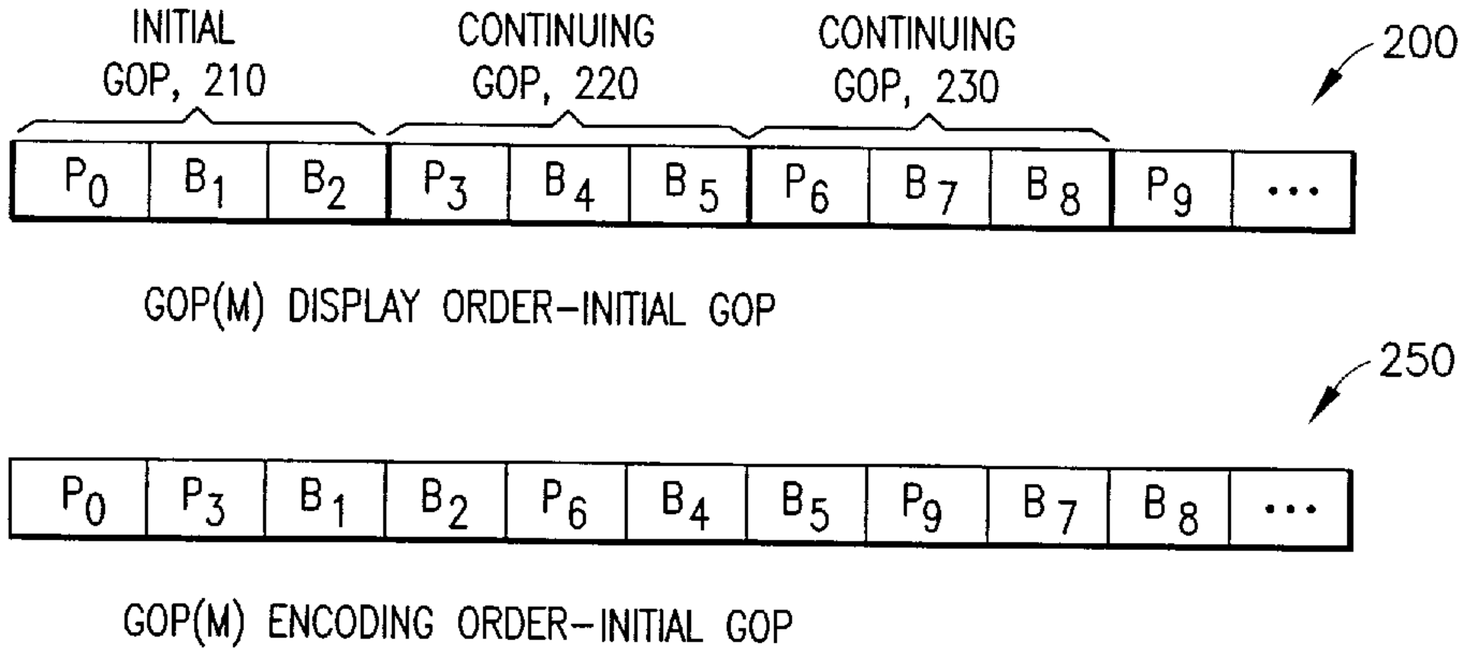
(21) (A1) **2,276,640**
(22) 1999/06/25
(43) 1999/12/30

(72) KORN, PETER A., US
(71) SUN MICROSYSTEMS, INC., US

(51) Int.Cl.[6] G06F 9/46
(30) 1998/06/30 (09/107,013) US

(54) **SYSTEME ET METHODE POUR FACILITER LE TRANSFERT DE DONNEES ENTRE PROGRAMMES TRAITES DANS UN SYSTEME NUMERIQUE D'ORDINATEUR**

(54) **SYSTEM AND METHOD FOR FACILITATING THE TRANSFER OF INFORMATION BETWEEN PROGRAMS PROCESSED IN A DIGITAL COMPUTER SYSTEM**



(57) An information transfer subsystem is disclosed for transferring information between a program being processed in a virtual machine execution environment and a program being processed in a native execution environment in a digital computer system. The information transfer subsystem comprises a virtual machine bridge class object and plurality of native bridge agents, the native bridge agent transferring the information with each other through a native communication mechanism provided by said native execution environment. The virtual machine bridge class object is instantiable in the virtual machine execution environment to handle transfers of the information with the program in said virtual machine execution environment. The virtual machine bridge class object, in turn, can transfer information with a first of the native bridge agents. A second of the native bridge agents is configured to transfer the information with the program being processed in said native execution environment. As noted above, the first and second native bridge agents can transfer the information with each other through a native communication mechanism provided by said native execution environment, thereby to facilitate transfer of the information between programs in the two environments.

ABSTRACT OF THE DISCLOSURE

An information transfer subsystem is disclosed for transferring information between a program being processed in a virtual machine execution environment and a program being processed in a native execution environment in a digital computer system. The information transfer subsystem comprises a virtual machine bridge class object and plurality of native bridge agents, the native bridge agent transferring the information with each other through a native communication mechanism provided by said native execution environment. The virtual machine bridge class object is instantiable in the virtual machine execution environment to handle transfers of the information with the program in said virtual machine execution environment. The virtual machine bridge class object, in turn, can transfer information with a first of the native bridge agents. A second of the native bridge agents is configured to transfer the information with the program being processed in said native execution environment. As noted above, the first and second native bridge agents can transfer the information with each other through a native communication mechanism provided by said native execution environment, thereby to facilitate transfer of the information between programs in the two environments.

P3078

SYSTEM AND METHOD FOR FACILITATING THE TRANSFER OF INFORMATION BETWEEN PROGRAMS PROCESSED IN A DIGITAL COMPUTER SYSTEM

FIELD OF THE INVENTION

The invention relates generally to the field of digital computer systems, and more particularly to arrangements for facilitating communications between programs processed in a digital computer system. The invention particularly provides a arrangement for facilitating transfer of information in a digital computer system between a program or other object being processed in a virtual machine in the digital computer system, and a program or other object being processed in a native execution environment in the digital computer system.

BACKGROUND OF THE INVENTION

Generally, computers process programs, under control of an operating system, in what is may be referred to as a "native execution environment." In such an environment, the program, generally in the native instruction set for the computer, can be executed in a process context. Other programs, on the other hand, which need not be in the native instruction set for the computer, may be executed in a virtual machine execution environment. There are a number of benefits in writing programs for execution in a virtual machine execution environment. For example, unlike programs written for a computer's native execution environment, programs written for execution in the virtual machine execution environment can often be made to be independent of the particular instruction set and operating system for the computer in which the program is being processed.

However, a problem arises if one wishes to transfer information between the programs processed in the native execution environment and the virtual machine execution environment. Several mechanisms have been developed to accomplish such transfer, generally relating to translation of information between the virtual machine execution environment and the native execution environment.

SUMMARY OF THE INVENTION

The invention provides a new and improved arrangement for facilitating the transfer of information in a digital computer system between a program or other object being processed in a virtual machine in the digital computer system, and a program or other object being processed in a native execution environment in the digital computer system.

In brief summary, an information transfer subsystem transfers information between a program being processed in a virtual machine execution environment and a program being processed in a native execution environment in a digital computer system. The information transfer subsystem comprises a virtual machine bridge class object and plurality of native bridge agents, the native bridge agent transferring the information with each other through a native communication mechanism provided by said native execution environment. The virtual machine bridge class object is instantiable in the virtual machine execution environment to handle transfers of the information with the program in said virtual machine execution environment. The virtual machine bridge class object, in turn, can transfer information with a first of the native bridge agents. A second of the native bridge agents is configured to transfer the information with the program being processed in said native execution environment. As noted above, the first and second native bridge agents can transfer the information with each other through a native communication mechanism provided by said native execution environment, thereby to facilitate transfer of the information between programs in the two environments.

BRIEF DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 depicts an illustrative digital cornputer system including an arrangement for facilitating the transfer of information between a programs and instantiated object being processed in a virtual machine execution environment, and programs being processed in a native execution environment, constructed in accordance with the invention;

FIG. 2 is a functional block diagram of an arrangement for facilitating the transfer of information between a programs and instantiated object being processed in a virtual machine execution environment, and programs being processed in a native execution environment, and

FIG. 3 is a flowchart depicting operations performed by the arrangement depicted in FIG. 2.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

FIG. 1 depicts an illustrative computer system 10 including an arrangement for facilitating the transfer of information between a programs and instantiated object being processed in a virtual machine execution environment, and programs being processed in a native execution environment, constructed in accordance with the invention. With reference to FIG. 1, the computer system 10 in one embodiment includes a processor module 11 and operator interface elements comprising operator input components such as a keyboard 12A and/or a mouse 12B (generally identified as operator input element(s) 12) and operator output components such as a video display device 13 with integral speakers 15. The illustrative computer system 10 is of the conventional stored-program computer architecture.

The processor module 11 includes, for example, processor, memory and mass storage devices such as disk and/or tape storage elements (not separately shown) which perform processing and storage operations in connection with digital data provided thereto. The mass storage subsystems may include such devices as disk or tape subsystems, optical disk storage devices and CD-ROM devices in which information may be stored and/or from which information may be retrieved. One

or more of the mass storage subsystems may utilize removable storage media which may be removed and installed by an operator, which may allow the operator to load programs and data into the digital computer system 10 and obtain processed data therefrom. Under control of control information provided thereto by the processor, information stored in the mass storage subsystems may be transferred to the memory for storage. After the information is stored in the memory, the processor may retrieve it from the memory for processing. After the processed data is generated, the processor may also enable the mass storage subsystems to retrieve the processed data from the memory for relatively long-term storage.

The operator input element(s) 12 are provided to permit an operator to input information for processing and/or control of the digital computer system 10. The video display device 13 and speakers 15 are provided to, respectively, display visual output information on a screen 14, and audio output information, which are generated by the processor module 11, which may include data that the operator may input for processing, information that the operator may input to control processing, as well as information generated during processing. The processor module 11 generates information for display by the video display device 13 using a so-called "graphical user interface" ("GUI"), in which information for various applications programs is displayed using various "windows." Although the computer system 10 is shown as comprising particular components, such as the keyboard 12A and mouse 12B for receiving input information from an operator, and a video display device 13 for displaying output information to the operator, it will be appreciated that the computer system 10 may include a variety of components in addition to or instead of those depicted in FIG. 1.

In addition, the processor module 11 may include one or more network or communication ports, generally identified by reference numeral 15, which can be connected to communication links to connect the computer system 10 in a computer network, or to other computer systems (not shown) over, for example, the public telephony system. The ports enable the computer system 10 to transmit information to, and receive information from, other computer systems and other devices in the network.

The invention provides an arrangement, which will be described below in connection with FIG. 2, for facilitating communication in a digital computer system between a program or other object being processed in a virtual machine in the digital computer system, and a program or other object being processed in a native execution environment in the digital computer system. In one embodiment, the virtual machine provides an Java execution environment for processing programs and other objects, which may be in the form of, for example, instantiated classes, which were written in the Java programming language. The native execution environment may be provided by, for example, the operating system which is controlling the computer system, such as, for example, Unix or a Unix-like operating system, Microsoft Windows™ or NT for a personal computer ("PC") or workstation, MacOS for an Apple Macintosh computer, and the like. The program(s) in the native execution environment generally comprise programs which are in the native instruction set of the computer system 10, although they may also be processed in connection with respective process(es) or virtual machines provided by the operating system.

With reference to FIG. 2, the communication arrangement comprises a virtual machine/native accessibility bridge 20 which bridges between a virtual machine execution environment 21 and a native execution environment 22 and facilitates communication between programs and other objects being processed in each execution environment 21 and 22. In the virtual machine execution environment, programs 23, instantiated classes 24 and other objects (not separately shown) are processed under control of a virtual machine control module 25. In one embodiment, the programs 23, instantiated classes 24 and other objects are generated from programs written in the Java programming language, which are processed under control of the virtual machine control module 25. On the other hand, programs 26 processed in the native execution environment generally comprise those in the native instruction set of the computer system 10.

The virtual machine/native accessibility bridge includes three primarily components, namely, a virtual machine bridge class object 30, native bridge dynamic link libraries ("DLLs") 31 and 32 and a native communication mechanism 33. The native communication mechanism 33 generally represents a communication mechanism that is provided by the computer system's operating system,

such as a shared memory mechanism, a messaging mechanism, and so forth. For example, in a shared memory communication mechanism, a predetermined portion of the address space used by the virtual machine execution environment 21 and a predetermined portion of the address space in which the native application programs 26 are processed are shared. In that case, if a program 23 or instantiated class 24 in the virtual machine execution environment 21 needs to transfer information to a native application program 26 in the native execution environment 22, it (that is, the program 23 or instantiated class 24, under control of the virtual machine control module 25) will enable the virtual machine bridge class object 30 to transfer the information to the native bridge DLL 31 associated therewith load the information into the shared portion of the address space. The native bridge DLL 32, in turn, can retrieve the information from the shared portion of the address space and provide it to the native application program 26 that is to receive the information. Conversely, if a native application program 26 in the native execution environment 22 needs to transfer information to a program 23 or instantiated class in the virtual machine execution environment 21, it will enable the native bridge DLL 32 to load the information into the shared portion of the address space. The native bridge DLL 31 will retrieve the information from the shared portion of the address space and provide the information to the virtual machine bridge class object 30, which, under control of the virtual machine control module 25, can provide it (that is, the information) to the virtual machine program 23 or instantiated class 24 that is to receive the information. On the other hand, in a messaging mechanism, the native bridge DLL 31 and native bridge DLL 32, instead of loading information to be transferred into, and retrieving information from, a shared portion of their respective address spaces, will generate a message, including the information to be transferred, that is addressed to the other element, and transfer the information using message transfer mechanisms provided by the operating system. Operations performed in connection with other types of communication mechanisms will be apparent to those skilled in the art.

The virtual machine bridge class object 30 is an object that is instantiated in the virtual machine execution environment 21 and allows information to be transferred between the virtual machine execution environment 21 and the native bridge DLL 31 using a conventional mechanism

provided by the virtual machine execution environment 21 to facilitate transfer of information between it and the native execution environment 22. In one embodiment, in which the virtual machine execution environment 21 is used to process programs in the Java programming language, the mechanism can comprise, for example, the JNI ("Java/native interface") mechanism. Each of the native bridge DLLs 31 and 32 comprises a dynamic link library that can be loaded by the operating system upon request from a native application program 26 and linked thereto for use thereby while it (that is, the native application program 26) is being processed. Either the native bridge DLL 31 associated with the virtual machine bridge class object 30, or the native bridge DLL 32 associated with the native application program 26, can be initialized and begin operating first, and when the other begins operating, each will notify the other, after which they can transfer information through the native communication mechanism 33 using a rendezvous mechanism. More specifically, when one of the native bridge DLL 31 or the native bridge DLL 32 is loaded and begins operating, it will query the native communication mechanism 33 to determine whether a notification has been provided by the other that it (that is, the other) has been loaded and is operating. In addition, it (that is, the just-loaded and operating native bridge DLL 31 or the native bridge DLL 32) will generate a notification for transfer to the other through the native communication mechanism 33. When the other (that is, the other of the native bridge DLL 32 or the native bridge DLL 31) is loaded and begins operating it will also query the native communication mechanism 33 and receive the notification previously loaded therein, and in addition, will generate a notification for transfer through the native communication mechanism 33 to the one of the native bridge DLL 31 or the native bridge DLL 32 to be loaded and operating. The first will then receive the notification. When each of the native bridge DLL 31 and the native bridge DLL 32 has received the notification from the other, it is in condition to transfer information to the other through the native communication mechanism 33, and can so notify the virtual machine bridge class object 30 and native application program 26 associated therewith.

Thus, the virtual machine control module 25 can instantiate the virtual machine bridge class as an object 30 in the virtual machine execution environment 21, and the virtual machine bridge

class object 30 can enable the native bridge DLL 31 to be loaded and linked thereto, regardless of whether the native bridge DLL 32 is operating. When, for example, the virtual machine control module 25 instantiates the virtual machine bridge class object 30 and enables it to begin operation, it (that is, the virtual machine bridge class object 30) will enable the native bridge DLL 31 to be loaded and query the native communication mechanism 33 to determine whether the native bridge DLL 32 has provided a notification, and in addition will generate a notification for transfer to the native communication mechanism 33. If the native bridge DLL 31, in response to the query, receives a notification from the native communication mechanism 33, the virtual machine bridge class object 30 it will determine that the native bridge DLL 32 is already operating and that it (that is, the virtual machine bridge class object 30) can transfer information from the programs 23 and instantiated classes 24 to respective native application programs 26 in the native execution environment 22. On the other hand, if the native bridge DLL 31 does not receive a notification from the native communication mechanism 33 in response to the query, the virtual machine bridge class object 30 will wait until it later receives a notification from the native bridge DLL 32 through the native communication mechanism 33.

Corresponding operations occur if the native bridge DLL 32 is loaded and begins operating before the virtual machine bridge class object 30 is instantiated. That is, when the operating system loads the native bridge DLL 32 and enables it to begin operation, it (that is, the native bridge DLL 32) will query the native communication mechanism 33 to determine whether the virtual machine bridge class object 30 has provided a notification, and in addition will generate a notification for transfer to the native communication mechanism 33. If the native bridge DLL 32, in response to the query, receives a notification from the native communication mechanism 33, it will determine that the virtual machine bridge class object 30 is already operating and that it (that is, the native bridge DLL 32) can transfer information from the native application programs 26 to respective and programs 23 and instantiated classes 24 in the virtual machine execution environment 21. On the other hand, if the native bridge DLL 32 does not receive a notification from the native communication mechanism 33 in response to the query, it will wait until it later receives a

notification from the virtual machine bridge class object 30 through the native communication mechanism 33.

In either case, after native bridge DLL 31 and the native bridge DLL 32 have established that the other has been initialized and is operating, they can transfer information between the programs 23 and instantiated classes 24, on the one hand, and the native application programs 26, on the other hand, using the native communication mechanism 33.

A specific example would be helpful in understanding the outputs performed by the arrangement depicted in FIG. 2 in connection with the transfer of information between the programs 23 and instantiated classes 24, on the one hand, and the native application programs 26, on the other hand, using the native communication mechanism 33. In that example, one of the native application programs 26 is a screen reader program, which generates information which enables the speakers 15 of computer system 10 to generate audio manifestations of information which would be displayed on the computer system's video display device 13. In addition, one or more of the programs 23 and/or instantiated classes 24 generates information which would be displayed on the video display device 13. In that case, to enable the screen reader program to generate information for portions of the video display representing the display information that is generated by the programs 23 and/or instantiated classes 24, the display information generated by the programs 23 and/or instantiated classes 24 would need to be provided to the screen reader program.

To accomplish that, the screen reader program would enable the operating system to load the native bridge DLL 32 and link it (that is, the native bridge DLL 32) to the screen reader program. In addition, one of the instantiated classes 24, as an "assistive technology" instantiated class, would be provided in the virtual machine execution environment 21 which can obtain display information from the programs 23 and ones of the other instantiated classes which are generating display information for provision, through the virtual machine/native accessibility bridge 20 and native bridge DLL 31, to the screen reader program. If the virtual machine bridge class object 30 and native bridge DLL 31 are not already load and operating, the assistive technology instantiated class

would enable the virtual machine control module 25 to load the virtual machine bridge class object 30 and enable it to operate. As part of that operation, the virtual machine bridge class object 30 will enable the native bridge DLL 31 to be loaded and linked thereto. Thereafter, the native bridge DLL 31 will generate a notification for transfer to the native bridge DLL through the native communication mechanism 33 as described above. In addition, since the native bridge DLL 32 is already loaded and operating, the native bridge DLL 31 will receive a notification as generated by the native bridge DLL 32 from the native communication mechanism 33 as described above, which, in turn, will provide a notification to the virtual machine bridge class object 30. After the screen reader program and the virtual machine bridge class 30 receive the respective notifications from the native communication mechanism 33, they will be able to transfer information therebetween. In addition, the native bridge DLL 32 will notify the native application program 26 to which it is linked, namely, the screen reader program that information can be transferred therethrough into the virtual machine execution environment 21, and, specifically to the assistive technology instantiated class, and the virtual machine bridge class object 30 may notify ones of the instantiated classes that information can be transferred therethrough into the native execution environment 22.

Thereafter, the screen reader program and the assistive technology instantiated class can communicate to transfer information through the native communication mechanism 33. In that operation, the screen reader program can transfer, as information, to the assistive technology instantiate class a request for transmission to the assistive technology instantiated class requesting video display information generated by a program 23 and/or another instantiated class 24 in the virtual machine execution environment 21. The screen reader program uses the native bridge DLL 32 to transfer the request to the native communication mechanism 33. The native communication mechanism 33 will transfer the request through the native bridge DLL 31 to the virtual machine native bridge class object 30, which, in turn, will transfer the request to the assistive technology instantiated class.

When the assistive technology instantiated class receives the request from the virtual machine bridge class 30, if a program 23 and/or other instantiated class 24 has video display information to

provide to the assistive technology instantiated class, it (that is, the assistive technology instantiated class) will obtain the video display information from a program 23 and/or other instantiated class 24 in the virtual machine execution environment 21 and provide it to the virtual machine bridge class object 30. The virtual machine bridge class object 30 will then transfer the video display information through the native bridge DLL 31 to the native communication mechanism 33, which, in turn, will transfer the video display information to the native bridge DLL 32, which, in turn, will provide it (that is, the video display information) to the screen reader application program. After the screen reader application program receives the video display information, it can use it in a conventional manner.

It will be appreciated that, if the assistive technology instantiated class determines that no program 23 and/or other instantiated class 24 has video display information to be provided thereto, it can generate information so indicating for provision to the screen reader program. The assistive technology instantiated class can provide such information to the screen reader program through the virtual machine bridge class object 30, native communication mechanism 33 and native bridge DLL 32 in the same manner as the video display information as described above.

Thereafter, the screen reader program may again request video display information, and the assistive technology instantiated class can provide video display information, or an indication that there is no video display information to provide, in a similar manner, in a series of iterations. Alternatively, the assistive technology instantiated class may, automatically in response to a single request from the screen reader program, iteratively or continually request video display information from the programs 23 and/or other instantiated classes for provision to the screen reader program in a manner similar to that described above.

With this background, the operations performed by the virtual machine bridge class object 30 and native bridge DLL 32 will be described in connection with the flowchart depicted in FIG. 3. With reference to FIG. 3, after the virtual machine control module 25 instantiates the virtual machine bridge class object 30 and enables it to operate (step 100), the virtual machine bridge class object

30 generates a notification for transfer to the native bridge DLL 32 indicating that it is ready to transfer information therewith through the native communication mechanism 33 (step 101). The virtual machine bridge class object 30 may be instantiated by the virtual machine control module 25 automatically, or in response to a request therefor from a program 23 or instantiated class 24 operating in the virtual machine execution environment 21. After the virtual machine bridge class is instantiated, it can enable the native bridge DLL 31 to be loaded and linked thereto (step 102). In addition, the native bridge DLL 31 will query the native communication mechanism 33 to determine whether it contains a notification from the native bridge DLL 32 indicating that it is ready to transfer information with the virtual machine bridge class object 30 (step 103). If the native bridge DLL 31 makes a negative determination in step 103, it will repeat step 103 until it make a positive determination, after which it will notify the virtual machine bridge class object 30 (step 104). After the virtual machine bridge class object 30 has received a notification from the native bridge DLL 31 in step 104, it will be in a condition to transfer information with the native bridge DLL 32 through the native communication mechanisrn 33, and it (that is, the virtual machine bridge class object 30) can so notify the virtual machine control module 25 or respective programs 23 or instantiated classes 24.

Similarly, after the operating system loads the native bridge DLL and enables it to operate (step 110), the native bridge DLL 32 generates a notification for transfer to the native bridge DLL 31 indicating that it is ready to transfer information therewith through the native communication mechanism 33 (step 111). The native bridge DLL 32 will generally be loaded by the operating system in response to a request therefor from a native application program 26 operating in the native execution environment 22. In addition, the native bridge DLL 32 will query the native communication mechanism 33 to determine whether it contains a notification from the native bridge DLL 31 indicating that it is ready to transfer information with the native bridge DLL 32 (step 112). If the native bridge DLL 32 makes a negative determination in step 112, it will repeat step 112 until it make a positive determination. After it (that is, the native bridge DLL 32) makes a positive determination in step 112, it will be in a condition to transfer information with the virtual machine

bridge class object 30 through the native communication mechanism 33 and native bridge DLL 31, and it (that is, the native bridge DLL) can so notify the native application program 26 for which it was loaded.

After the virtual machine bridge class object 30 has received a notification from the native bridge DLL 32 (step 102), if it receives a request from a program 23 or instantiated class 24 operating in the virtual machine execution environment to transfer information to a native application program 26 (step 120), it will transfer the information to the native bridge DLL 31 (step 121), which, in turn, will transfer to the native communication mechanism 33 (step 122). In that operation, if the native communication mechanism 33 is in the form of a shared portion of the address space, the native bridge DLL 31 can load the information into the shared portion of the address space. In addition, the native bridge DLL 31 may provide an indication which will enable the native bridge DLL 32 to be notified that the shared portion of the address space contains information that it can retrieve. Alternatively, the native bridge DLL 32 may periodically examine the shared portion of the address space to determine whether the native bridge DLL 31 has loaded information for it therein (that is, in the shared portion of the address space). In any case, after the native bridge DLL 32 determines that the shared portion of the address space contains information for it to retrieve, it will retrieve the information (step 123) and provide it (that is, the information) to the native application program 26 for which it was loaded. Corresponding operations can be performed if the native communication mechanism 33 uses other mechanisms, such as, for example, message passing for transferring information between the virtual machine bridge class object 30 and the native bridge DLL 32.

Similarly, after the native bridge DLL 32 has received a notification from the virtual machine bridge class object 30 (step 102), if it receives a request from a native application program 26 to transfer information to a virtual machine bridge class object 30 virtual machine bridge class object 30 (step 130), it will transfer the information to the native communication mechanism 33 (step 131). In that operation, if the native communication mechanism 33 is in the form of a shared portion of the address space, the native bridge DLL 32 can load the information into the shared portion of the

address space. In addition, the native bridge DLL 32 may provide an indication which will enable the native bridge DLL 31 to be notified that the shared portion of the address space contains information that it can retrieve. Alternatively, the native bridge DLL 31 may periodically examine the shared portion of the address space to determine whether the native bridge DLL has loaded information for it therein (that is, in the shared portion of the address space). In any case, after the native bridge DLL 31 determines that the shared portion of the address space contains information for it to retrieve, it will retrieve the information (step 132) and provide it (that is, the information) to the virtual machine bridge class object 30 (step 133), which, in turn, will provide the information a program 23 and/or instantiated class 24 in the virtual machine execution environment 21 (step 134). Corresponding operations can be performed if the native communication mechanism 33 uses other mechanisms, such as, for example, message passing for transferring information between the virtual machine bridge class object 30 and the native bridge DLL 32.

The invention provides a number of advantages. In particular, it provides a mechanism, identified as the virtual machine/native accessibility bridge 20, for transferring information between programs and instantiated classes processed in a virtual machine execution environment, such an execution environment that used to process program written in, for example, the Java programming language, and a native program that is processed in a process or other element in the native execution environment provided by the computer system's operating system. The components of the virtual machine/native accessibility bridge 20, in particular the virtual machine bridge class object 30 and the native bridge DLLs 31 and 32 transfer the information without requiring translation, using a native communication mechanism 33 provided by the computer system's operating system. The native bridge DLLs 31 and 32 can be loaded and initialized separately from each other and utilize a rendezvous mechanism, as described above, to notify the other that they are in condition to transfer information therewith.

It will be appreciated that a number of modifications may be made to the invention as described above. For example, as indicated above, a number of types of mechanisms, provided by the operating system, may be used as the native communication mechanism 33. In addition,

although the invention has been described in connection with a single virtual machine bridge class object 30, for a single virtual machine execution environment 21, and associated DLL 31, and a single instance of a DLL 32, it will be appreciated that a number of such objects 30, for, for example, respective virtual machine execution environments 21, and a number of instances of such DLL's 31 and 32, can transfer information through the native communication mechanism 33.

It will be appreciated that a system in accordance with the invention can be constructed in whole or in part from special purpose hardware or a general purpose computer system, or any combination thereof, any portion of which may be controlled by a suitable program. Any program may in whole or in part comprise part of or be stored on the system in a conventional manner, or it may in whole or in part be provided in to the system over a network or other mechanism for transferring information in a conventional manner. In addition, it will be appreciated that the system may be operated and/or otherwise controlled by means of information provided by an operator using operator input elements (not shown) which may be connected directly to the system or which may transfer the information to the system over a network or other mechanism for transferring information in a conventional manner.

The foregoing description has been limited to a specific embodiment of this invention. It will be apparent, however, that various variations and modifications may be made to the invention, with the attainment of some or all of the advantages of the invention. It is the object of the appended claims to cover these and such other variations and modifications as come within the true spirit and scope of the invention.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

What is claimed is:

1. A method for allocating bits for coding pictures in a bitstream received at a digital video transcoder, comprising the steps of:

(a) providing an assumed distance M' between a first picture of said bitstream and the next closest subsequent P-picture of said bitstream in a display order of said bitstream;

(b) providing an assumed bit budget as a function of said assumed distance M';

(c) coding said first picture in accordance with said assumed bit budget;

(d) determining a picture type of a second picture that immediately follows said first picture in said bitstream in an encoding order of said bit stream;

(e) adjusting said assumed bit budget according to said picture type of said second picture; and

(f) allocating bits for coding said second picture in accordance with said adjusted bit budget.

2. The method of claim 1, wherein:

said assumed bit budget is proportional to said assumed distance and a bit rate of said bitstream, and inversely proportional to a frame rate of said bitstream.

3. The method of claim 1 or 2, wherein:

said pictures in said bitstream form a progressive refresh sequence.

4. The method of one of claims 1 to 3, wherein said first picture is a P-picture and said second picture is a B-picture, comprising the further step of:

determining an actual distance M'' between said first picture and said next closest subsequent P-picture, in said display order, according to said picture type of said second picture.

5. The method of claim 4, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in said display order, and a temporal reference of said first picture, in said display order, plus one picture.

6. The method of claim 4 or 5, comprising the further step of:

allocating bits for coding the remaining M''-1 pictures following said first picture in accordance with said adjusted bit budget.

7. The method of one of claims 4 to 6, wherein:

said adjusting step (e) comprises the step of adjusting said assumed bit budget by +(M''-M')*"bit

rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

8. The method of claim 1, wherein said first and second pictures are P-pictures, comprising the further step of:

determining an actual distance M'' between said first picture and said next closest subsequent P-picture, in said display order, according to said picture type of said second picture.

9. The method of claim 8, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in said display order, and a temporal reference of said first picture, in said display order.

10. The method of claim 8 or 9, wherein said adjusting step (e) comprises the step of:

adjusting said assumed bit budget by -(M'-1)*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

11. The method of one of claims 8 to 10, comprising the further step of:

allocating bits for coding the remaining M''-1 pictures following said first picture in accordance with said adjusted bit budget.

12. The method of one of claims 1 to 5 or 8 to 11, comprising the further step of:

allocating bits for coding a series of M'' pictures following an initial M'' picture that includes said first and second pictures in said bitstream in accordance with said adjusted bit budget.

13. A method for allocating bits for coding a sequence of pictures in a bitstream received at a digital video transcoder, comprising the steps of:

(a) providing an assumed length N' of a particular group of pictures (GOP) of said bitstream;

(b) providing an assumed bit budget as a function of said assumed length N';

(c) allocating bits for coding a first picture of said particular GOP in accordance with said assumed bit budget;

(d) determining a picture type of a second picture that immediately follows said first picture;

(e) adjusting said assumed bit budget according to said picture type of said second picture; and

(f) allocating bits for coding said second picture in accordance with said adjusted bit budget.

14. The method of claim 13, wherein:

said assumed bit budget is proportional to said assumed length N' and a bit rate of said bitstream, and inversely proportional to a frame rate of said bitstream.

15. The method of claim 13 or 14, wherein said first and second pictures are I-pictures, comprising the further steps of:

determining an actual distance M'' between said first picture and the next closest subsequent I-picture of said bitstream according to the picture type of said second picture; and

adjusting said assumed bit budget in accordance with said actual distance M''.

16. The method of claim 15, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in a display order of said bitstream, and a temporal reference of said first picture, in said display order.

17. The method of one of claims 13 to 16, wherein said adjusting step (e) comprises the step of:

adjusting said assumed bit budget by -(N'-1)*"bit rate"/"frame rate", where "bit rate" is a bit rate of

said bitstream, and "frame rate" is a frame rate of said bitstream.

18. The method of one of claims 13 to 17, comprising the further step of:

allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

19. The method of one of claims 13 to 18, comprising the further step of:

providing an assumed distance M' between said first picture and the next closest subsequent P-picture of said bitstream, in a display order of said bitstream;

wherein said bits are allocated for coding said first picture in said step (c) in accordance with said assumed distance M'.

20. The method of claim 19, wherein said first picture is an I-picture and said second picture is a P-picture, comprising the further steps of:

determining an actual distance M'' between said first picture and the next closest subsequent P-picture, in said display order, according to said picture type of said second picture; and

adjusting said assumed length N' of said GOP in accordance with said actual distance M'' to provide an adjusted assumed length N''; wherein:

said adjusting step (e) is responsive to said adjusted assumed length N'.

21. The method of claim 20, wherein:

said assumed length N' of said particular GOP is adjusted in said adjusting step thereof by a factor, M(N'/M'), to provide said adjusted assumed length N'', where M=M''.

22. The method of claim 20 or 21, wherein said adjusting step (e) comprises the step of:

adjusting said assumed bit budget by a factor (N''-N')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

23. The method of claim 20 or 21, wherein said adjusting step (e) comprises the further step of:

adjusting said assumed bit budget by a factor, (M''-1')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

24. The method of one of claims 20 to 23, comprising the further step of:

allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

25. The method of one of claims 20 to 24, comprising the further steps of:

determining if N''>N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP; and, if N''>N:

further adjusting said adjusted bit budget according to N and N''; and

allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget.

26. The method of one of claims 20 to 24, comprising the further steps of:

determining if N''<N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP; and, if N''<N:

further adjusting said adjusted bit budget according to M=M''; and

allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget.

27. The method of claim 13, wherein said first picture is an I-picture and said second picture is a B-picture, comprising the further step of:

providing an assumed distance M' between said first picture and the next closest subsequent P-picture of said bitstream, in a display order of said bitstream;

wherein said bits are allocated for coding said second picture in accordance with said assumed distance M'.

28. The method of claim 27, comprising the further steps of:

determining an actual distance M'' between said first picture and the next closest subsequent P-picture, in said display order, according to said picture type of said second picture; and

adjusting said assumed length N' of said GOP in accordance with an actual distance M=M'' to provide an adjusted assumed length N''; wherein:

said adjusting step (e) is responsive to said adjusted assumed length N'.

29. The method of claim 28, wherein:

said assumed length N' of said particular GOP is adjusted in said adjusting step thereof by a factor, M(N'/M'), to provide said adjusted assumed length N''.

30. The method of claim 28 or 29, wherein said adjusting step (e) comprises the step of:

adjusting said assumed bit budget by a factor (N''-N')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

31. The method of one of claims 28 to 30, comprising the further step of:

allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

32. The method of one of claims 28 to 31, comprising the further step of:

determining if N''>N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP; and, if N''>N:

further adjusting said adjusted bit budget according to N and N''; and

allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget.

33. The method of one of claims 28 to 32, comprising the further steps of:

determining if N''<N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP; and, if N''<N:

further adjusting said adjusted bit budget according to M=M''; and

allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget.

34. The method of one of claims 13 to 33, comprising the further step of:

allocating bits for coding a GOP following said particular GOP in accordance with said adjusted bit budget.

35. The method of one of claims 13 to 34, comprising the further step of:

accounting for changes in an actual distance M'' between I-pictures and/or P-pictures in said particular GOP by determining an actual distance M'' between an I-picture or P-picture that follows said first picture in said particular GOP, in a display order of said bitstream, and a subsequent I-picture or P-picture in said particular GOP; and

adjusting said assumed bit budget in accordance with said actual distance M''.

36. An apparatus for allocating bits for coding pictures in a bitstream received at a digital video transcoder, comprising:

(a) means for providing an assumed distance M' between a first picture of said bitstream and the next closest subsequent P-picture of said bitstream in a display order of said bitstream;

(b) means for providing an assumed bit budget as a function of said assumed distance M';

(c) means for coding said first picture in accordance with said assumed bit budget;

(d) means for determining a picture type of a second picture that immediately follows said first picture in said bitstream in an encoding order of said bit stream;

(e) means for adjusting said assumed bit budget according to said picture type of said second picture; and

(f) means for allocating bits for coding said second picture in accordance with said adjusted bit budget.

37. The apparatus of claim 36, wherein:

said assumed bit budget is proportional to said assumed distance and a bit rate of said bitstream, and inversely proportional to a frame rate of said bitstream.

38. The apparatus of claim 36 or 37, wherein:

said pictures in said bitstream form a progressive refresh sequence.

39. The apparatus of one of claims 36 to 38, wherein said first picture is a P-picture and said second picture is a B-picture, further comprising:

means for determining an actual distance M'' between said first picture and said next closest subsequent P-picture, in said display order, according to said picture type of said second picture.

40. The apparatus of claim 39, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in said display order, and a temporal reference of said first picture, in said display order, plus one picture.

41. The apparatus of claim 39 or 40, further comprising:

means for allocating bits for coding the remaining M''-1 pictures following said first picture in accordance with said adjusted bit budget.

42. The apparatus of one of claims 39 to 41, wherein:

said adjusting means (e) comprises means for adjusting said assumed bit budget by +(M''-M')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

43. The apparatus of one of claims 36 to 42, wherein said first and second pictures are P-pictures, further comprising:

means for determining an actual distance M'' between said first picture and said next closest subsequent P-picture, in said display order, according to said picture type of said second picture.

44. The apparatus of claim 43, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in said display order, and a temporal reference of said first picture, in said display order.

45. The apparatus of claim 43 or 44, wherein said adjusting means (e) comprises:

means for adjusting said assumed bit budget by -(M'-1)*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

46. The apparatus of one of claims 43 to 45, further comprising:

means for allocating bits for coding the remaining M''-1 pictures following said first picture in accordance with said adjusted bit budget.

47. The apparatus of one of claims 36 to 45, further comprising:

means for allocating bits for coding a series of M'' pictures following an initial M'' pictures that includes said first and second pictures in said bitstream in accordance with said adjusted bit budget.

48. An apparatus for allocating bits for coding a sequence of pictures in a bitstream received at a digital video transcoder, comprising:

(a) means for providing an assumed length N' of a particular group of pictures (GOP) of said bitstream;

(b) means for providing an assumed bit budget as a function of said assumed length N';

(c) means for allocating bits for coding a first picture of said particular GOP in accordance with said assumed bit budget;

(d) means for determining a picture type of a second picture that immediately follows said first picture;

(e) means for adjusting said assumed bit budget according to said picture type of said second picture; and

(f) means for allocating bits for coding said second picture in accordance with said adjusted bit budget.

49. The apparatus of claim 48, wherein:

said assumed bit budget is proportional to said assumed length N' and a bit rate of said bitstream, and inversely proportional to a frame rate of said bitstream.

50. The apparatus of claim 48 or 49, wherein said first and second pictures are I-pictures, further comprising:

means for determining an actual distance M'' between said first picture and the next closest subsequent I-picture of said bitstream according to the picture type of said second picture; and

means for adjusting said assumed bit budget in accordance with said actual distance M''.

51. The apparatus of claim 50, wherein:

said actual distance M'' is determined according to a difference between a temporal reference of said second picture, in a display order of said bitstream,

and a temporal reference of said first picture, in said display order.

52. The apparatus of one of claims 48 to 51, wherein said adjusting means (e) comprises:

means for adjusting said assumed bit budget by -(N'-1)*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

53. The apparatus of one of claims 48 to 52, further comprising:

means for allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

54. The apparatus of one of claims 48 to 53, further comprising:

means for providing an assumed distance M' between said first picture and the next closest subsequent P-picture of said bitstream, in a display order of said bitstream;

wherein said bits are allocated for coding said first picture by said coding means (c) in accordance with said assumed distance M'.

55. The apparatus of claim 54, wherein said first picture is an I-picture and said second picture is a P-picture, further comprising:

means for determining an actual distance M'' between said first picture and the next closest subsequent P-picture, in said display order, according to said picture type of said second picture; and

means for adjusting said assumed length N' of said GOP in accordance with said actual distance M'' to provide an adjusted assumed length N''; wherein:

said adjusting means (e) is responsive to said adjusted assumed length N'.

56. The apparatus of claim 55, wherein:

said assumed length N' of said particular GOP is adjusted by said adjusting means thereof by a factor, M(N'/M'), to provide said adjusted assumed length N'', where M=M''.

57. The apparatus of claim 55 or 56, wherein said adjusting means (e) comprises:

means for adjusting said assumed bit budget by a factor (N''-N')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

58. The apparatus of claim 57, wherein said adjusting means (e) comprises:

means for adjusting said assumed bit budget by a factor, (M''-1')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

59. The apparatus of one of claims 55 to 58, further comprising:

means for allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

60. The apparatus of one of claims 55 to 59, further comprising:

means for determining if N''>N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP;

means for further adjusting said adjusted bit budget according to N and N'' if N''>N; and

means for allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget if N''>N.

61. The apparatus of one of claims 55 to 59, further comprising:

means for determining if N''<N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP;

means for further adjusting said adjusted bit budget according to M=M'' if N''<N; and

means for allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget if N''<N.

62. The apparatus of claim 48, wherein said first picture is an I-picture and said second picture is a B-picture, further comprising:

means for providing an assumed distance M' between said first picture and the next closest subsequent P-picture of said bitstream, in a display order of said bitstream;

wherein said bits are allocated for coding said second picture in accordance with said assumed distance M'.

63. The apparatus of claim 62, further comprising:

means for determining an actual distance M'' between said first picture and the next closest subsequent P-picture, in said display order, according to said picture type of said second picture; and

means for adjusting said assumed length N' of said GOP in accordance with an actual distance M=M'' to provide an adjusted assumed length N''; wherein:

said adjusting means (e) is responsive to said adjusted assumed length N'.

64. The apparatus of claim 63, wherein:

said assumed length N' of said particular GOP is adjusted by said adjusting means thereof by a factor, M(N'/M'), to provide said adjusted assumed length N''.

65. The apparatus of claim 63 or 64, wherein said adjusting means (e) comprises:

means for adjusting said assumed bit budget by a factor (N''-N')*"bit rate"/"frame rate", where "bit rate" is a bit rate of said bitstream, and "frame rate" is a frame rate of said bitstream.

66. The apparatus of one of claims 63 to 65, further comprising:

means for allocating bits for coding the remaining pictures in said particular GOP following said second picture in accordance with said adjusted bit budget.

67. The apparatus of one of claims 63 to 66, further comprising:

means for determining if N''>N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP;

means for further adjusting said adjusted bit budget according to N and N'' if N''>N; and

means for allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget if N''>N.

68. The apparatus of one of claims 63 to 66, further comprising:

means for determining if N''<N, an actual length of said particular GOP, after coding N'' pictures in said particular GOP;

means for further adjusting said adjusted bit budget according to M=M'' if N''<N; and

means for allocating bits for coding pictures in said bitstream following the N'' coded pictures according to said further adjusted bit budget if N''<N.

69. The apparatus of one of claims 48 to 68, further comprising:

means for allocating bits for coding a GOP following said particular GOP in accordance with said adjusted bit budget.

70. The apparatus of one of claims 48 to 69, further comprising:

means for accounting for changes in an actual distance M'' between I-pictures and/or P-pictures in said particular GOP by determining an actual distance M'' between an I-picture or P-picture that follows said first picture in said particular GOP, in a display

order of said bitstream, and a subsequent I-picture or P-picture in said particular GOP; and

means for adjusting said assumed bit budget in accordance with said actual distance M''.

71. A method for allocating bits for coding a sequence of pictures in a bitstream received at a digital video transcoder, comprising the steps of:

(a) providing a length of groups of pictures (GOPs) of said bitstream;

(b) providing a distance between each I-picture and a next successive P-picture in said GOPs;

(c) maintaining a count of a number of remaining I-, P- and B-pictures, respectively, in each GOP as each picture in said GOPs is coded;

(d) providing an assumed bit budget in response to said steps (a), (b) and (c) for coding pictures in said GOPs;

(e) verifying, and adjusting, if necessary, said length after each I-picture in said GOPs is coded;

(f) verifying, and adjusting, if necessary, said distance after each I-picture and/or P-picture in said GOPs is coded;

(g) verifying, and adjusting, if necessary, the count of the number of remaining I-, P- and B-pictures, in response to said steps (e) and (f); and

(g) adjusting, if necessary, said assumed bit budget in response to said steps (e), (f) and (g).

72. The method of claim 71, wherein:

said distance is assumed when initially coding said GOPs and is corrected to an actual distance within a time for coding two pictures in said bitstream.

73. The method of claim 71 or 72, wherein:

said length is assumed when initially coding said GOPs and is corrected to an actual length within a time for coding one GOP in said bitstream.

74. An apparatus for allocating bits for coding a sequence of pictures in a bitstream received at a digital video transcoder, comprising:

(a) means for providing a length of groups of pictures (GOPs) of said bitstream;

(b) means for providing a distance between each I-picture and a next successive P-picture in said GOPs;

(c) means for maintaining a count of a number of remaining I-, P- and B-pictures, respectively, in each GOP as each picture in said GOPs is coded;

(d) means for providing an assumed bit budget in response to said means (a), (b) and (c) for coding pictures in said GOPs;

(e) means for verifying, and adjusting, if necessary, said length after each I-picture in said GOPs is coded;

(f) means for verifying, and adjusting, if necessary, said distance after each I-picture and/or P-picture in said GOPs is coded;

(g) means for verifying, and adjusting, if necessary, the count of the number of remaining I-, P- and B-pictures, in response to said means (e) and (f); and

(g) means for adjusting, if necessary, said assumed bit budget in response to said means (e), (f) and (g).

75. The apparatus of claim 74, wherein:

said distance is assumed when initially coding said GOPs and is corrected to an actual distance within a time for coding two pictures in said bitstream.

76. The apparatus of claim 74 or 75, wherein:

said length is assumed when initially coding said GOPs and is corrected to an actual length within a time for coding one GOP in said bitstream.

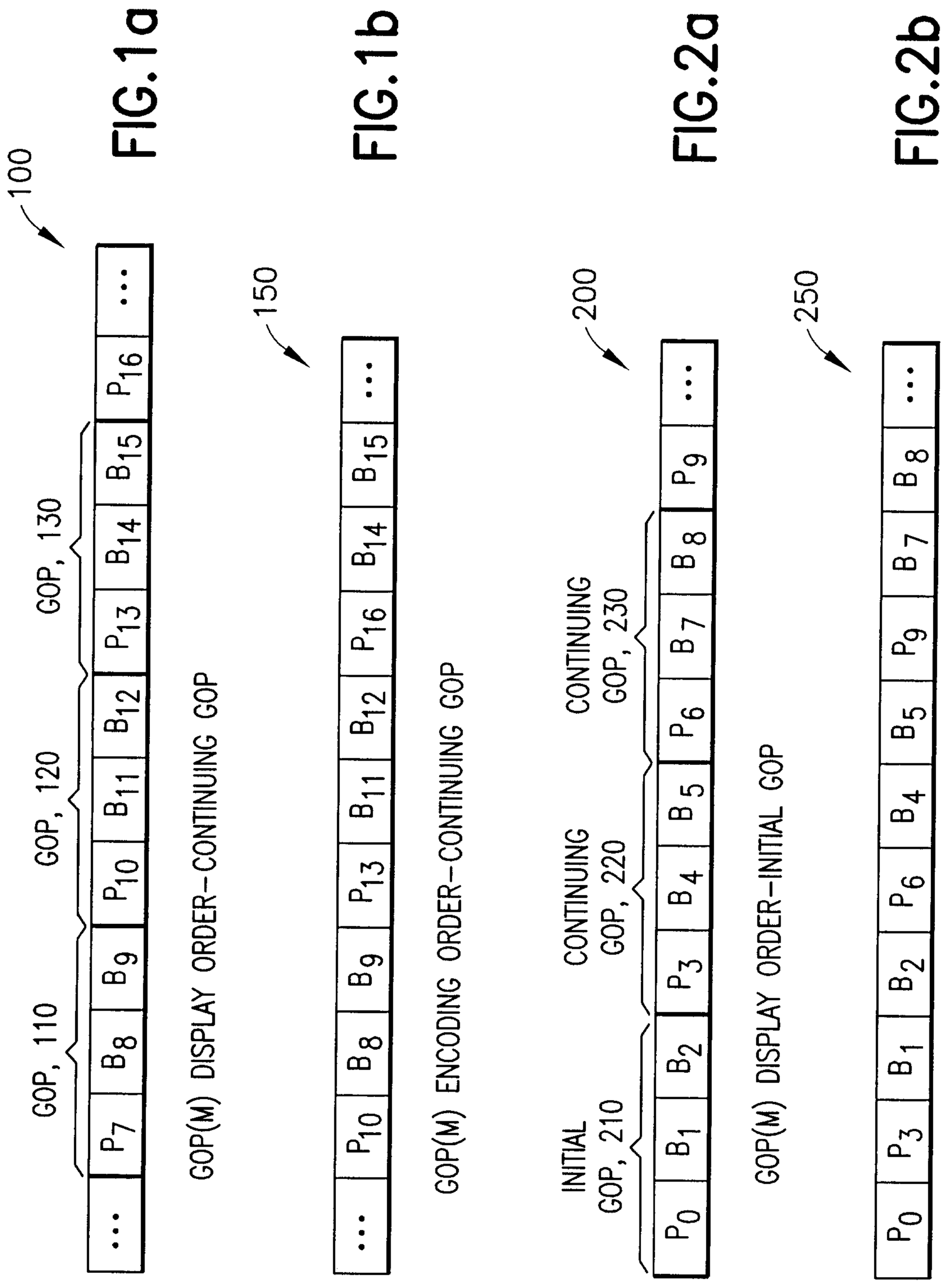


FIG.1a

FIG.1b

FIG.2a

FIG.2b

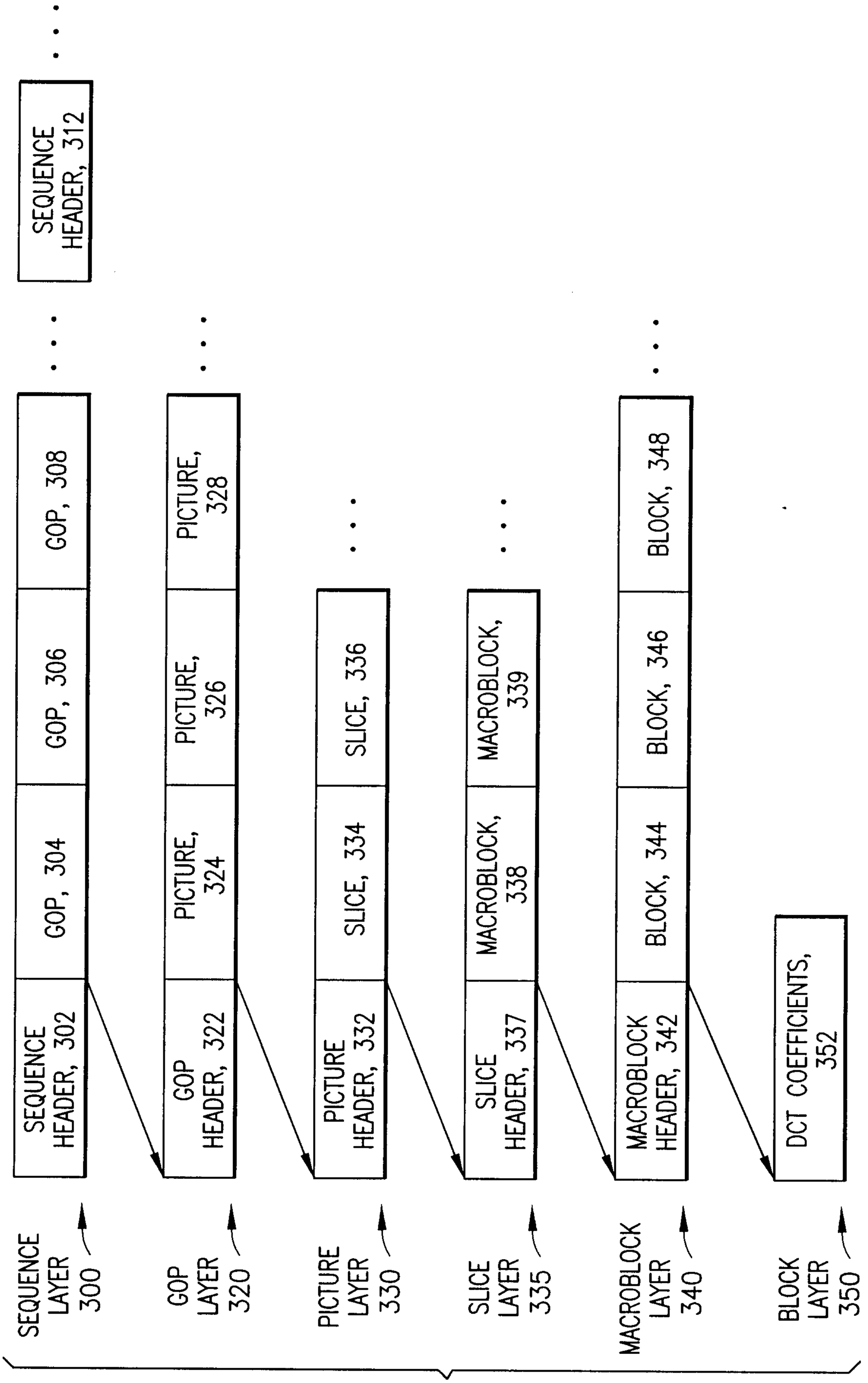
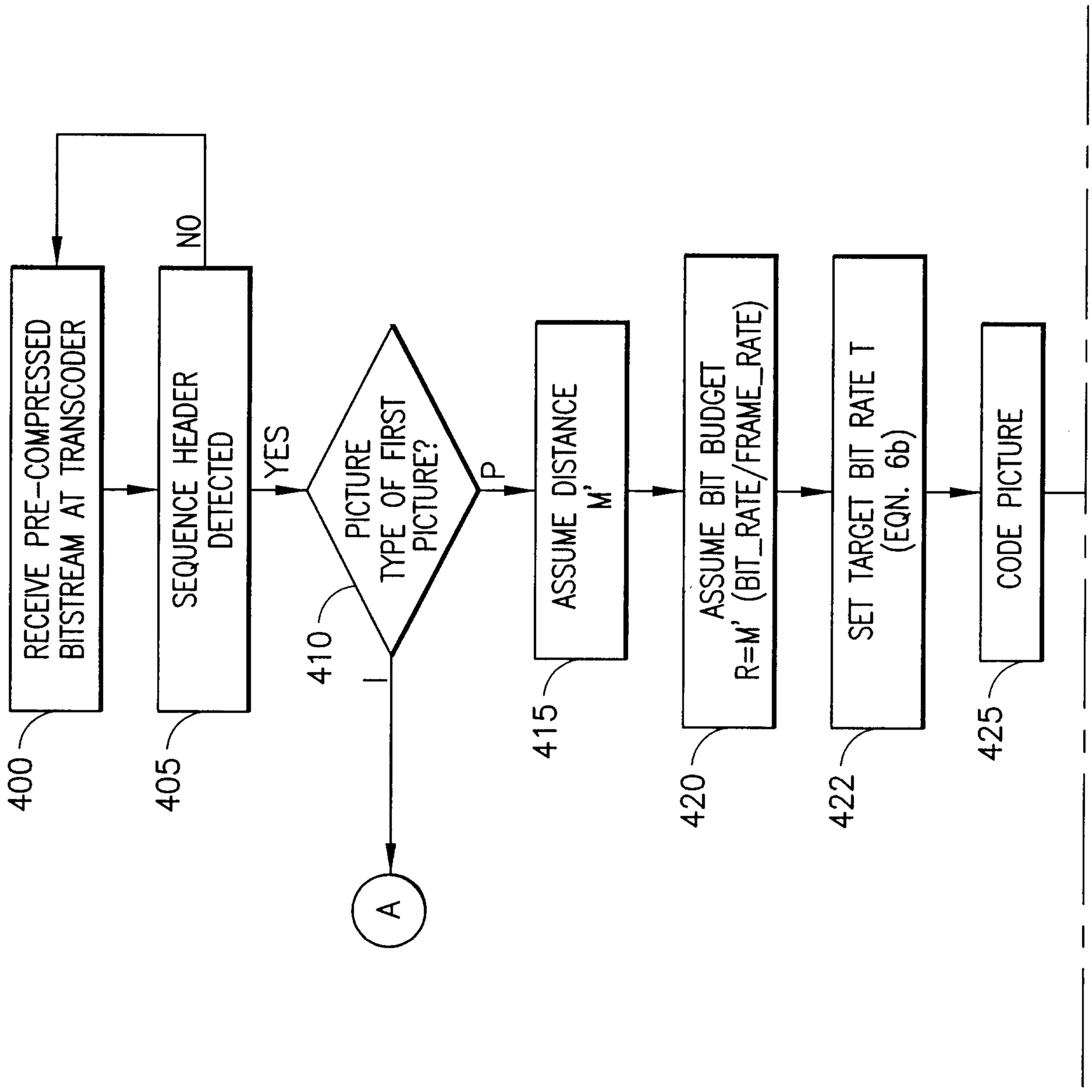

SEQUENCE LAYER 300
GOP LAYER 320
PICTURE LAYER 330
SLICE LAYER 335
MACROBLOCK LAYER 340
BLOCK LAYER 350
SEQUENCE HEADER, 302
GOP HEADER, 322
PICTURE HEADER, 332
SLICE HEADER, 337
MACROBLOCK HEADER, 342
DCT COEFFICIENTS, 352
SEQUENCE HEADER, 312
GOP, 308
GOP, 304
GOP, 306
PICTURE, 328
PICTURE, 324
PICTURE, 326
SLICE, 334
SLICE, 336
MACROBLOCK, 338
MACROBLOCK, 339
BLOCK, 344
BLOCK, 346
BLOCK, 348

FIG.3

FIG.4a

FIG.4a-1
FIG.4a-2

FIG.4a-1



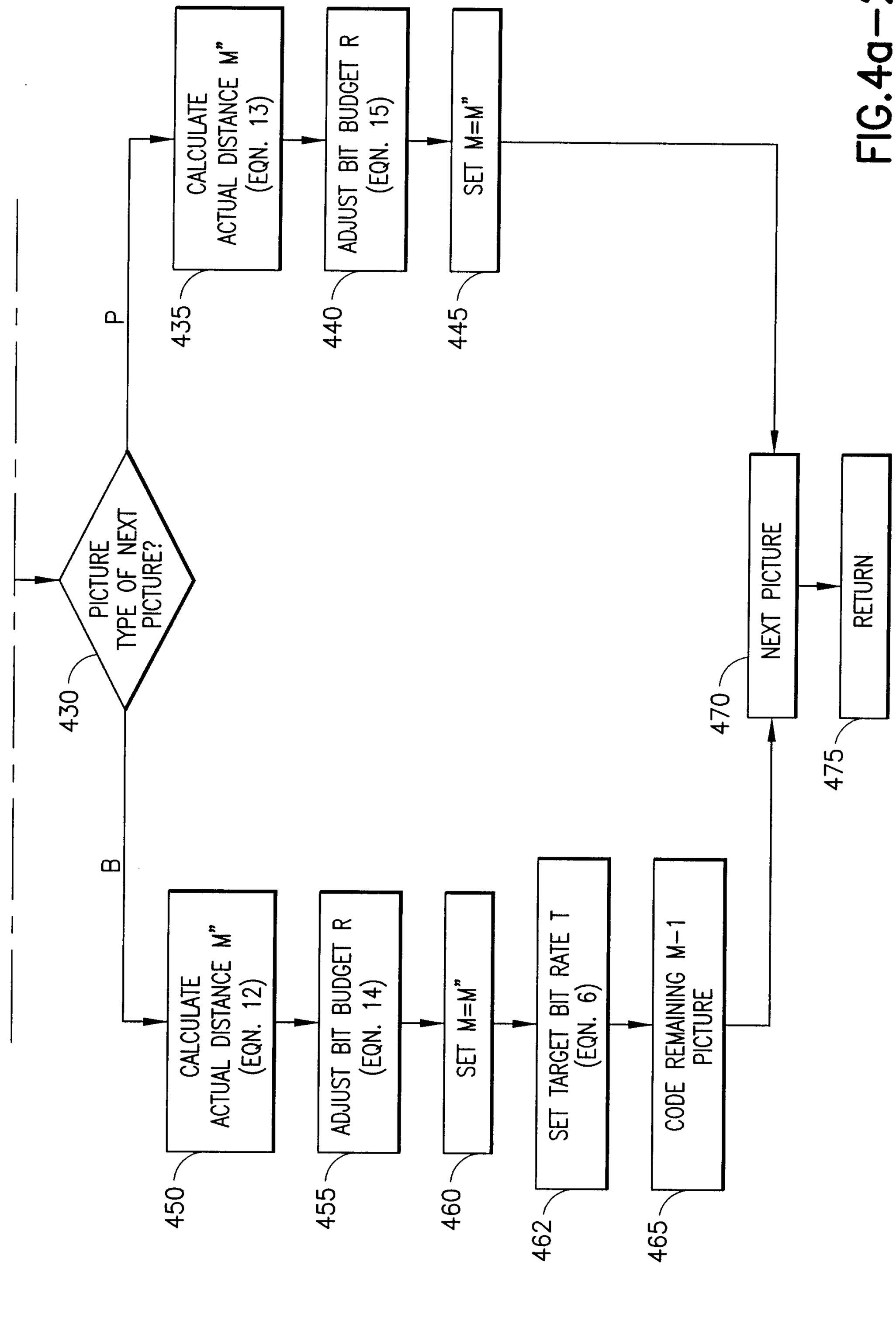

P
B
PICTURE TYPE OF NEXT PICTURE?
430
CALCULATE ACTUAL DISTANCE M" (EQN. 13)
435
ADJUST BIT BUDGET R (EQN. 15)
440
SET M=M"
445
CALCULATE ACTUAL DISTANCE M" (EQN. 12)
450
ADJUST BIT BUDGET R (EQN. 14)
455
SET M=M"
460
SET TARGET BIT RATE T (EQN. 6)
462
CODE REMAINING M-1 PICTURE
465
NEXT PICTURE
470
RETURN
475

FIG. 4a-2

FIG.4b

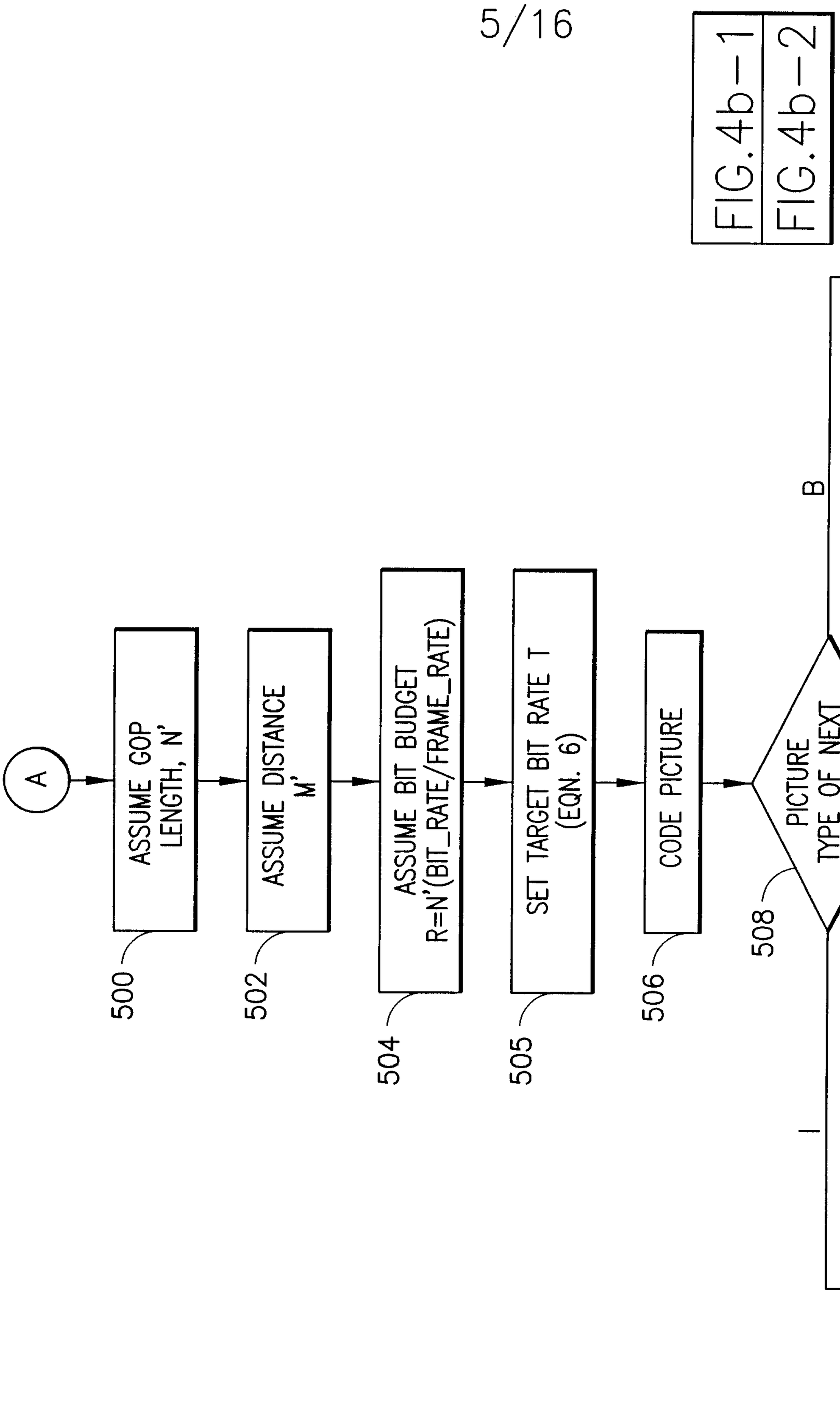


FIG.4b-1

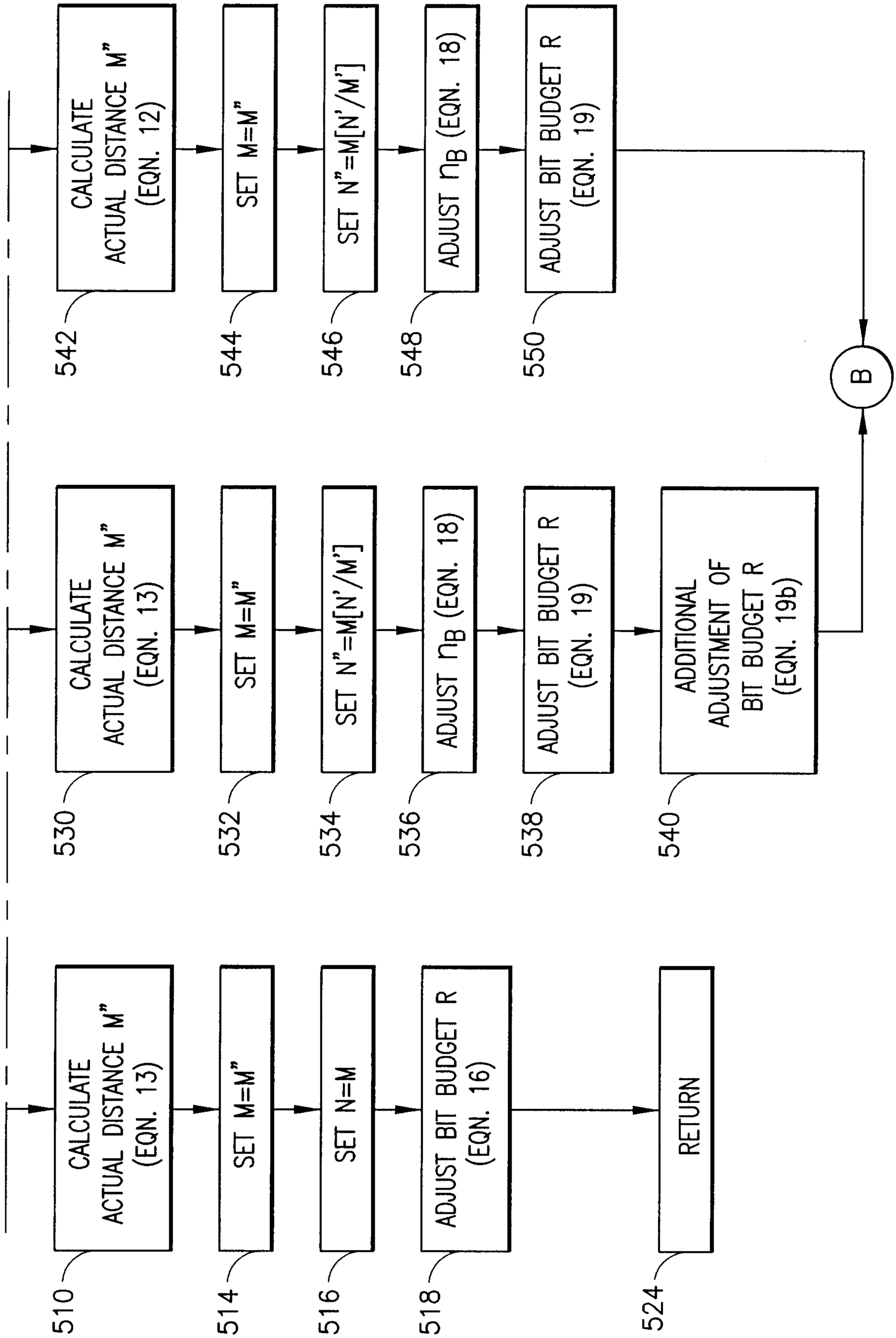

CALCULATE ACTUAL DISTANCE M" (EQN. 12)
542
SET M=M"
544
SET N"=M[N'/M']
546
ADJUST η_B (EQN. 18)
548
ADJUST BIT BUDGET R (EQN. 19)
550
B
CALCULATE ACTUAL DISTANCE M" (EQN. 13)
530
SET M=M"
532
SET N"=M[N'/M']
534
ADJUST η_B (EQN. 18)
536
ADJUST BIT BUDGET R (EQN. 19)
538
ADDITIONAL ADJUSTMENT OF BIT BUDGET R (EQN. 19b)
540
CALCULATE ACTUAL DISTANCE M" (EQN. 13)
510
SET M=M"
514
SET N=M
516
ADJUST BIT BUDGET R (EQN. 16)
518
RETURN
524

FIG.4b-2

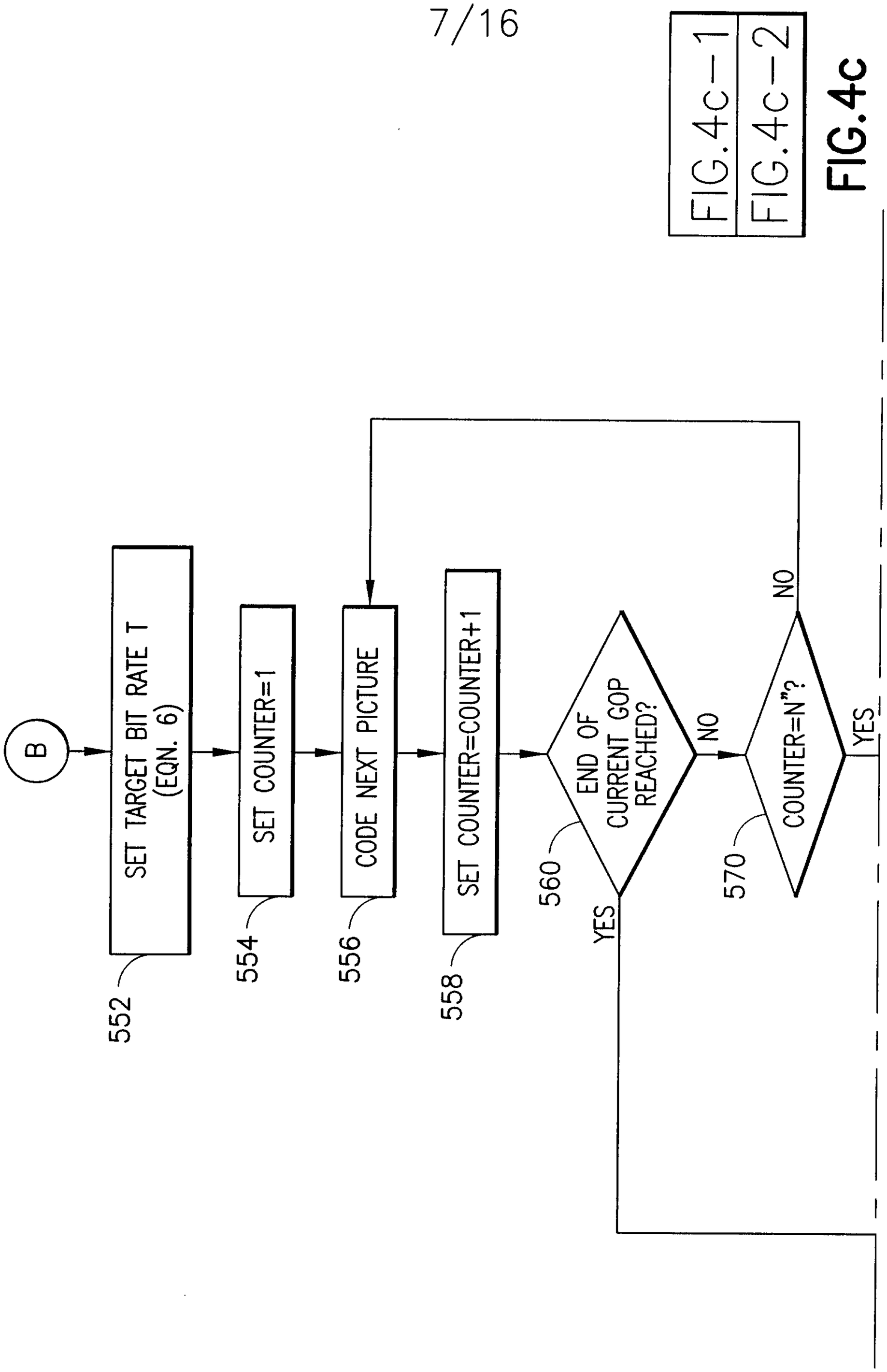


FIG.4c-1
FIG.4c-2
B
SET TARGET BIT RATE T (EQN. 6)
552
SET COUNTER=1
554
CODE NEXT PICTURE
556
SET COUNTER=COUNTER+1
558
END OF CURRENT GOP REACHED?
560
YES
NO
COUNTER=N"?
570
YES

FIG.4c

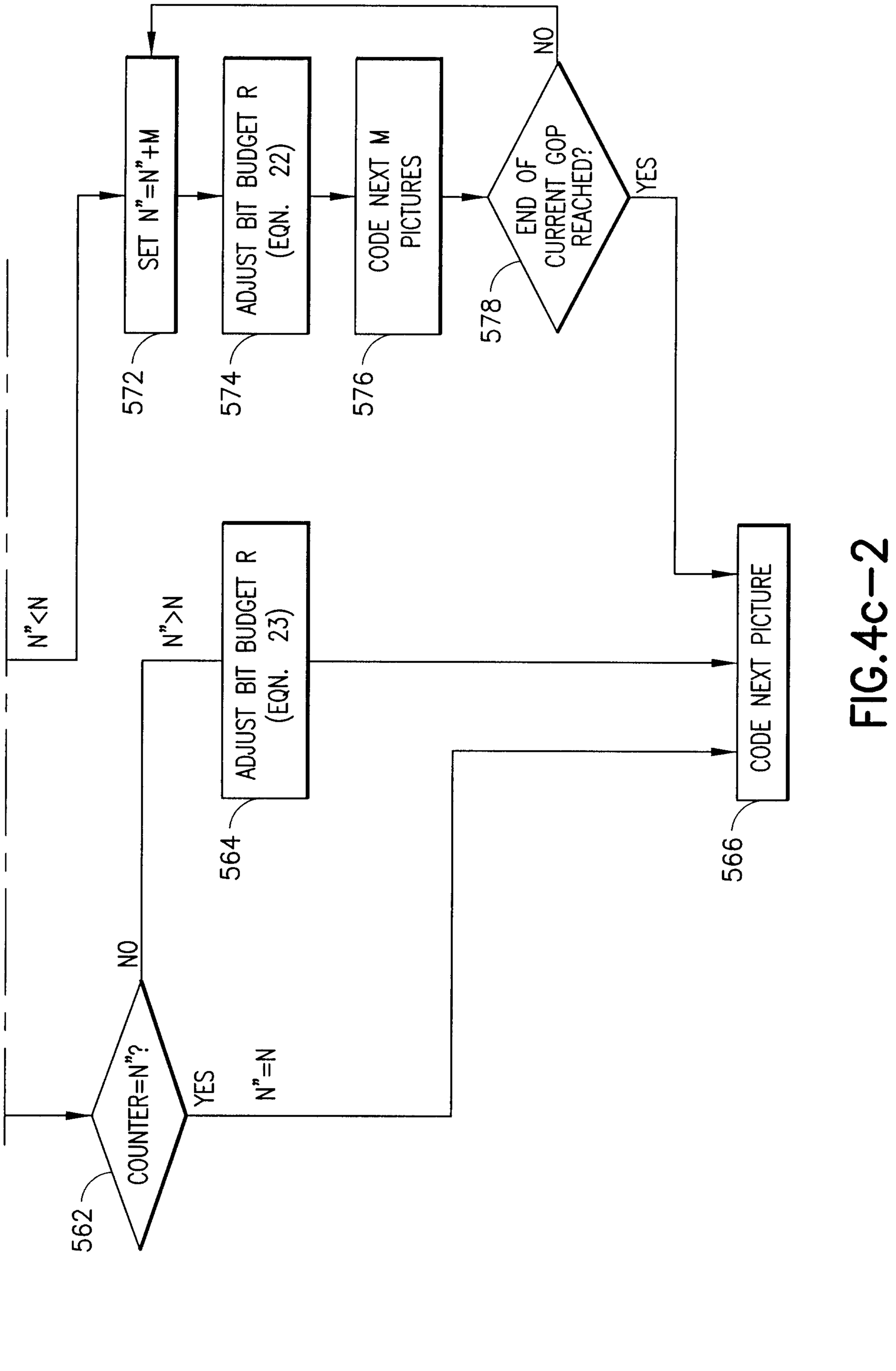

SET N"=N"+M
572
ADJUST BIT BUDGET R (EQN. 22)
574
CODE NEXT M PICTURES
576
END OF CURRENT GOP REACHED?
578
NO
YES
N"<N
N"=N
ADJUST BIT BUDGET R (EQN. 23)
564
CODE NEXT PICTURE
566
NO
COUNTER=N"?
562
YES
N"=N

FIG.4c-2

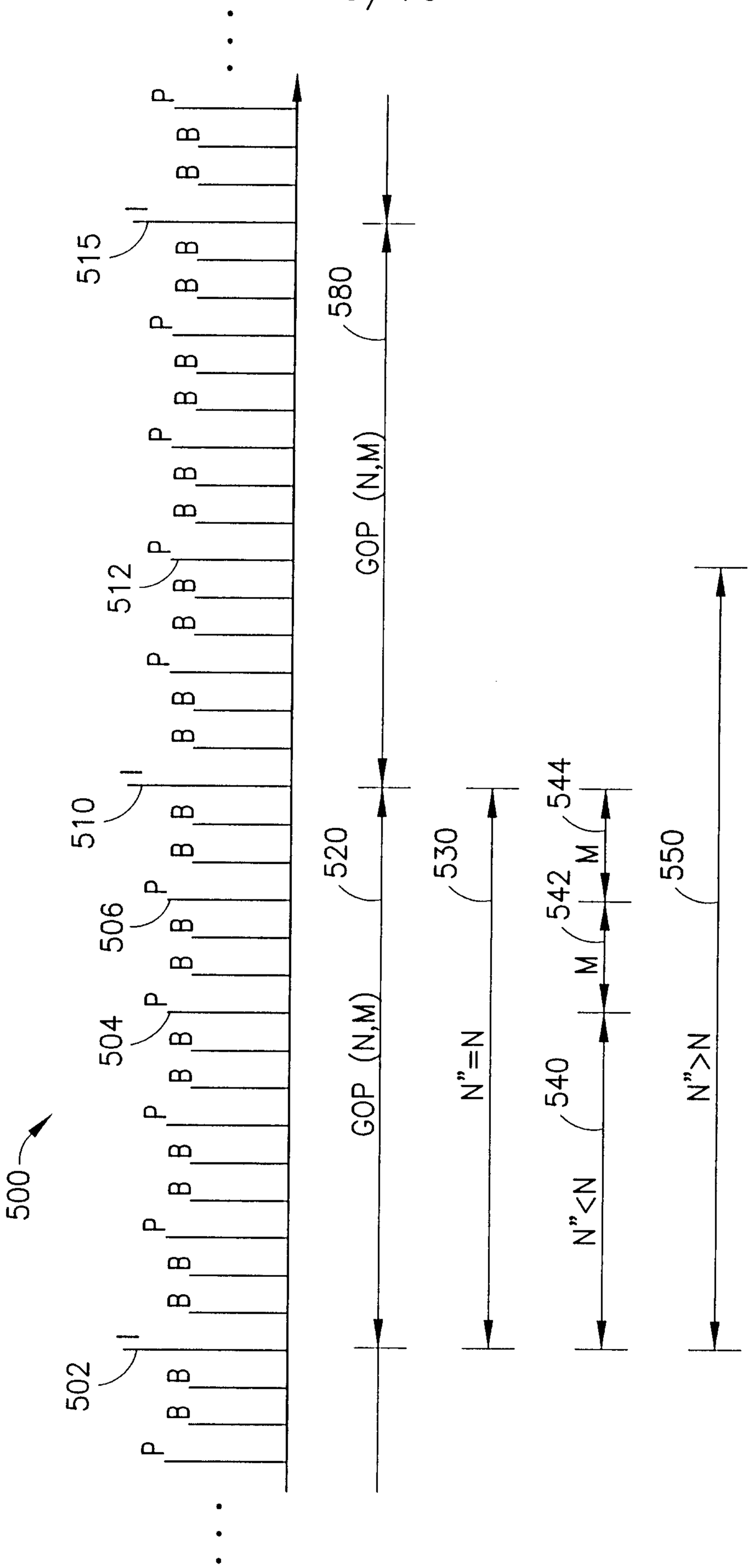

500
502
504
506
510
512
515
I
P
B
580
GOP (N,M)
520
GOP (N,M)
530
N"=N
540
N"<N
542
M
544
M
550
N">N

FIG.5

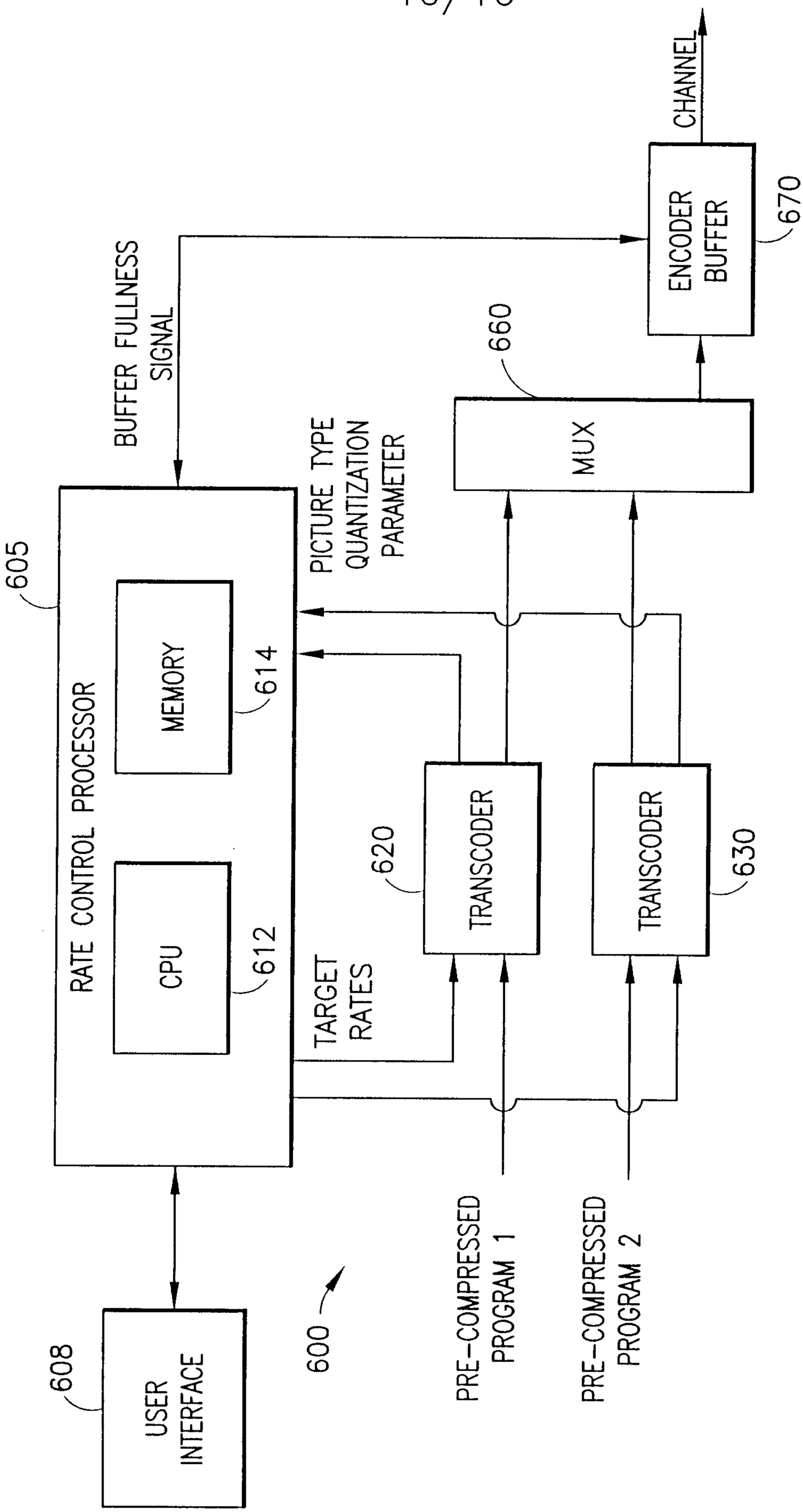

PRE-COMPRESSED PROGRAM 1
PRE-COMPRESSED PROGRAM 2
600
608
USER INTERFACE
605
RATE CONTROL PROCESSOR
CPU
612
MEMORY
614
TARGET RATES
620
TRANSCODER
630
TRANSCODER
PICTURE TYPE QUANTIZATION PARAMETER
BUFFER FULLNESS SIGNAL
660
MUX
670
ENCODER BUFFER
CHANNEL

FIG.6

FIG.7a

FIG.7a–1	FIG.7a–2

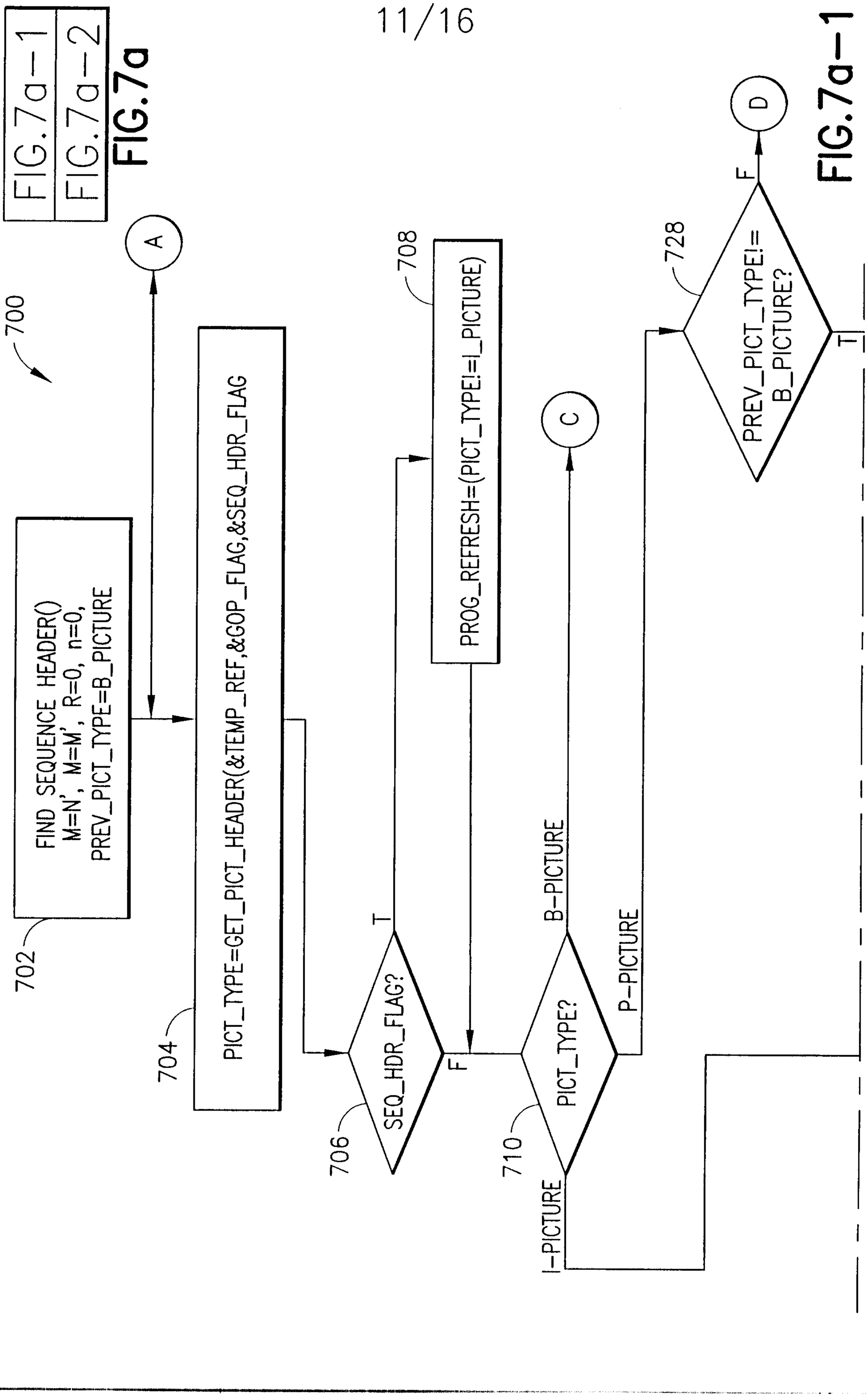


FIG.7a–1

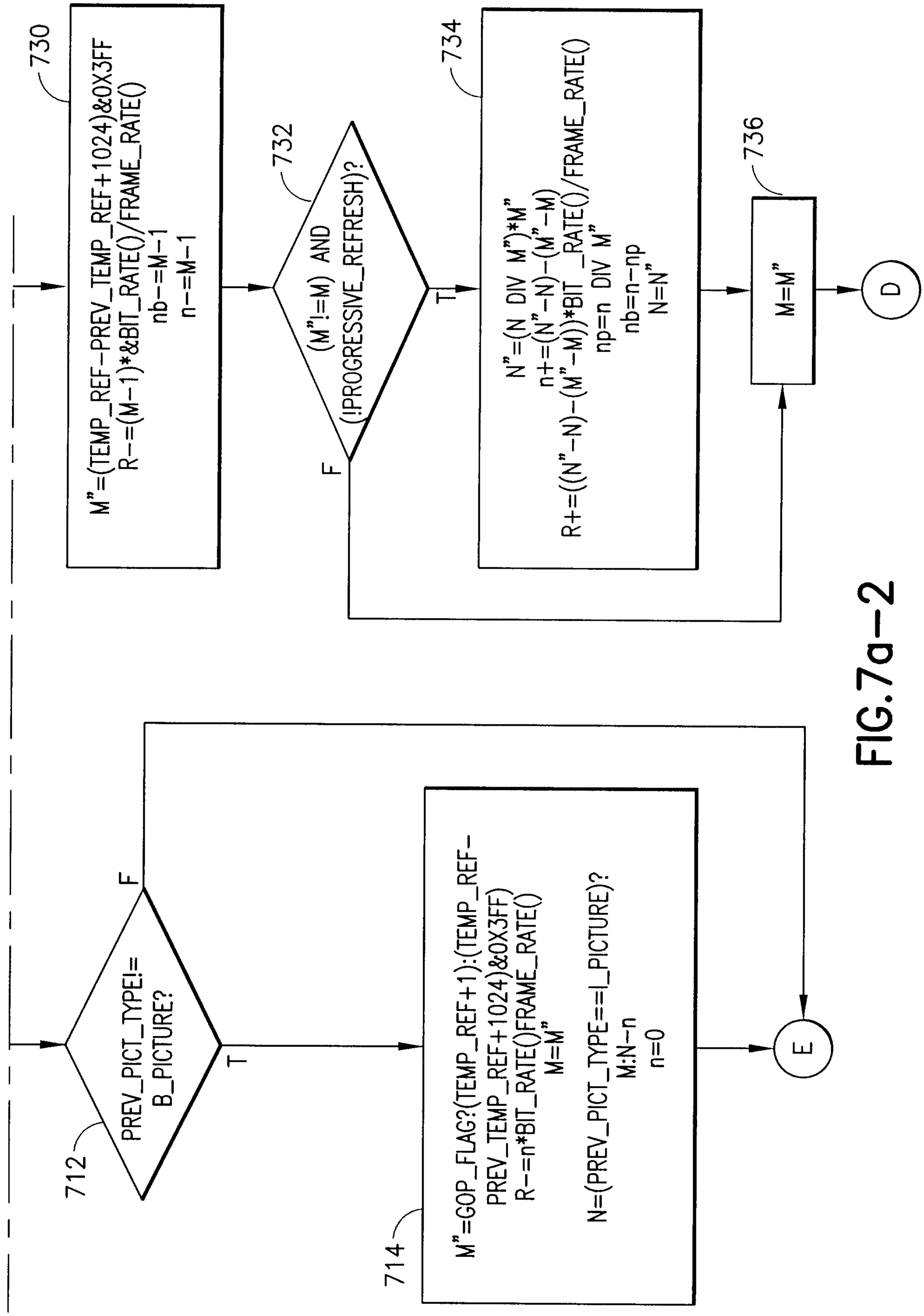

712
PREV_PICT_TYPE!=
B_PICTURE?
F
T
714
M"=GOP_FLAG?(TEMP_REF+1):(TEMP_REF-
PREV_TEMP_REF+1024)&0X3FF)
R-=n*BIT_RATE()FRAME_RATE()
M=M"
N=(PREV_PICT_TYPE==I_PICTURE)?
M:N-n
n=0
E
712
F
(M"!=M) AND
(!PROGRESSIVE_REFRESH)?
T
734
N"=(N DIV M")*M"
n+=(N"-N)-(M"-M)
R+=((N"-N)-(M"-M))*BIT_RATE()/FRAME_RATE()
np=n DIV M"
nb=n-np
N=N"
736
M=M"
D
730
M"=(TEMP_REF-PREV_TEMP_REF+1024)&0X3FF
R-=(M-1)*&BIT_RATE()/FRAME_RATE()
nb-=M-1
n-=M-1

FIG.7a-2

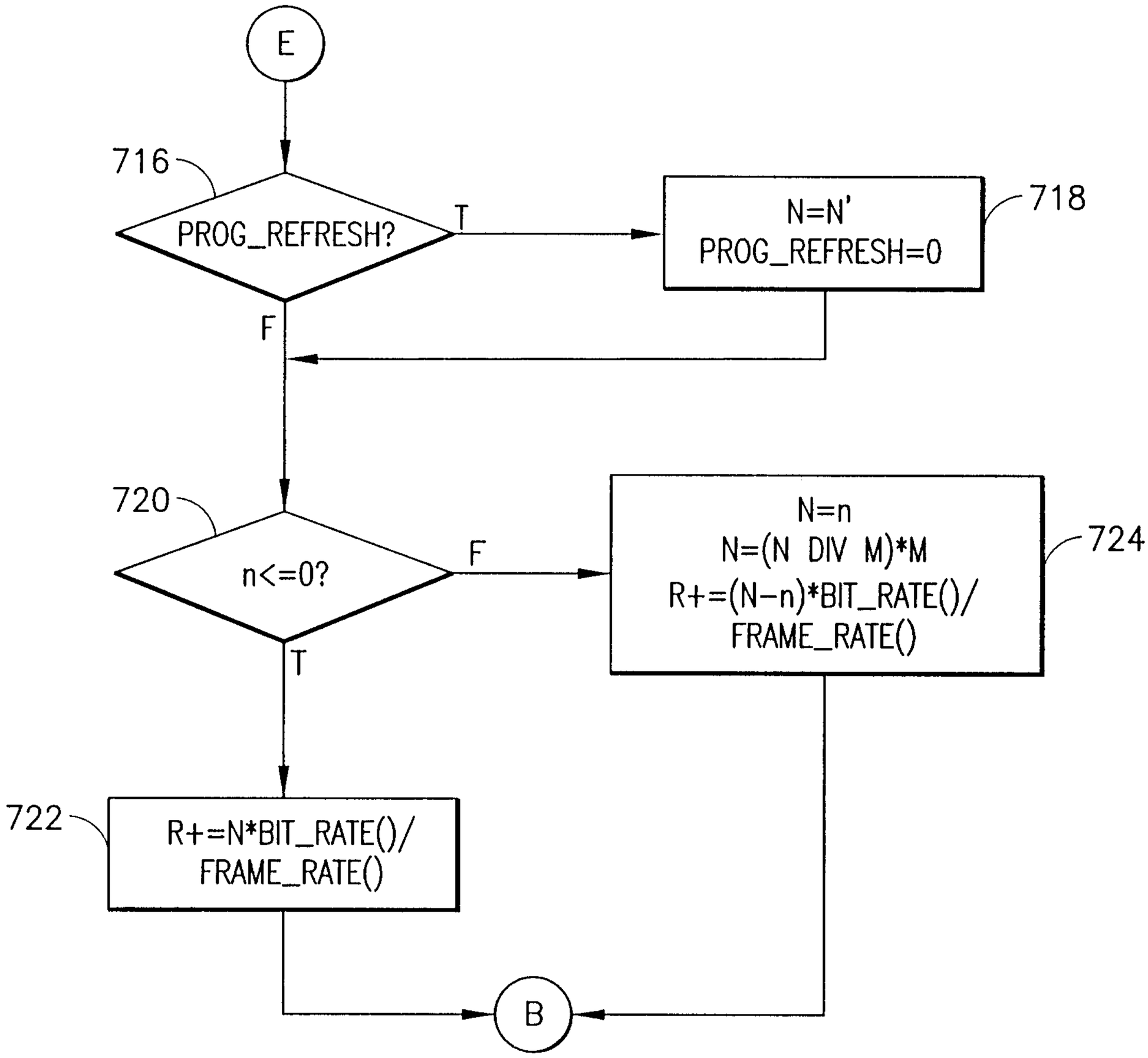

E
716
PROG_REFRESH?
T
N=N'
PROG_REFRESH=0
718
F
720
n<=0?
F
N=n
N=(N DIV M)*M
R+=(N-n)*BIT_RATE()/
FRAME_RATE()
724
T
722
R+=N*BIT_RATE()/
FRAME_RATE()
B

FIG.7b

FIG.7c

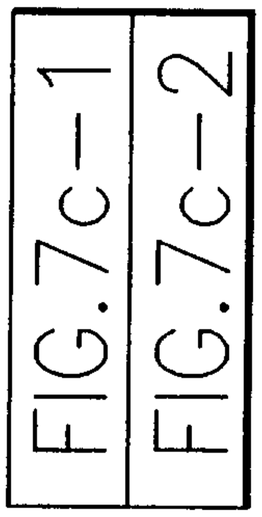


FIG.7c-1

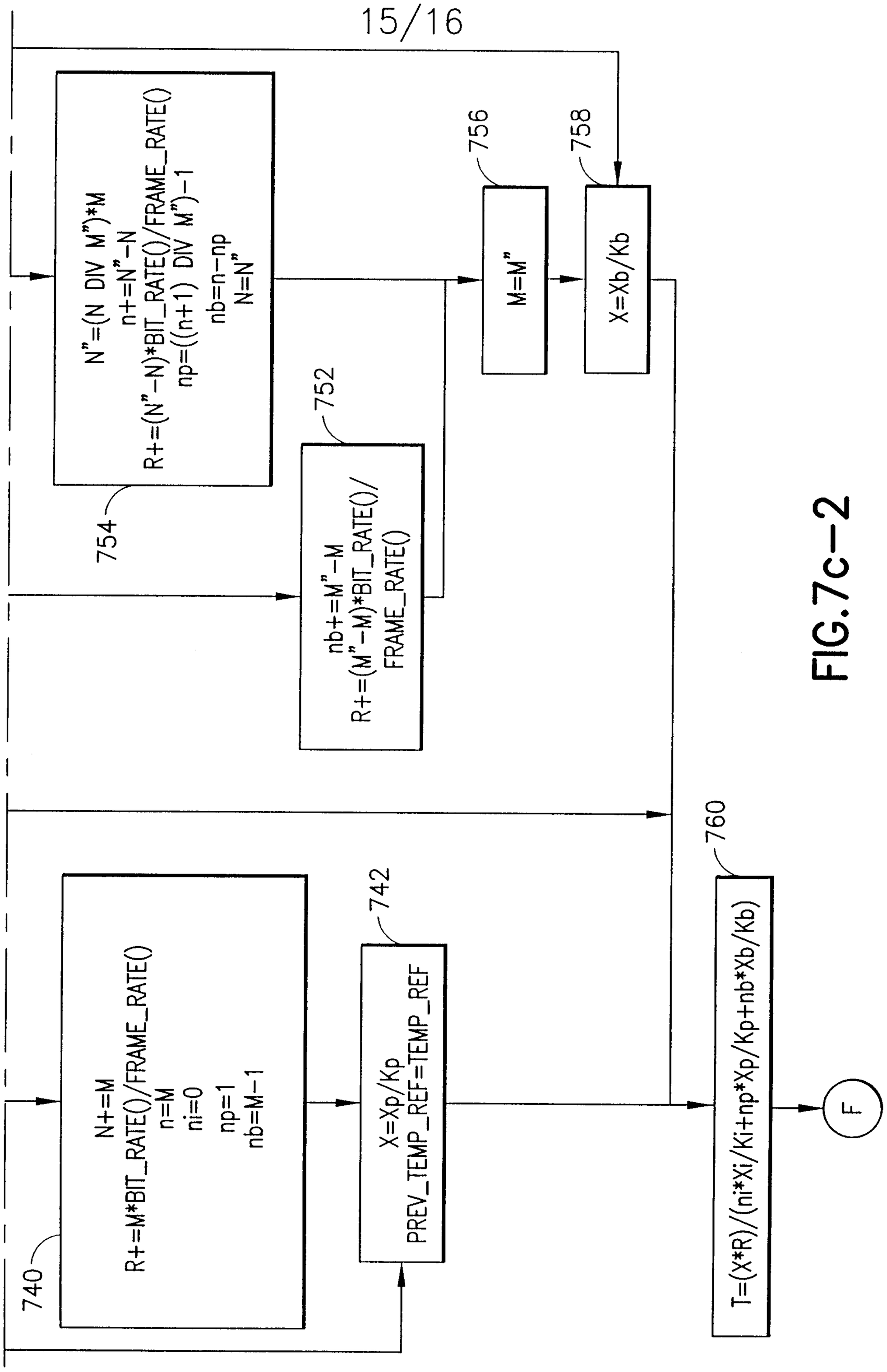

740
N+=M
R+=M*BIT_RATE()/FRAME_RATE()
n=M
ni=0
np=1
nb=M−1
742
X=Xp/Kp
PREV_TEMP_REF=TEMP_REF
754
N"=(N DIV M")*M
n+=N"−N
R+=(N"−N)*BIT_RATE()/FRAME_RATE()
np=((n+1) DIV M")−1
nb=n−np
N=N"
752
nb+=M"−M
R+=(M"−M)*BIT_RATE()/
FRAME_RATE()
756
M=M"
758
X=Xb/Kb
760
T=(X*R)/(ni*Xi/Ki+np*Xp/Kp+nb*Xb/Kb)
F

FIG.7c−2

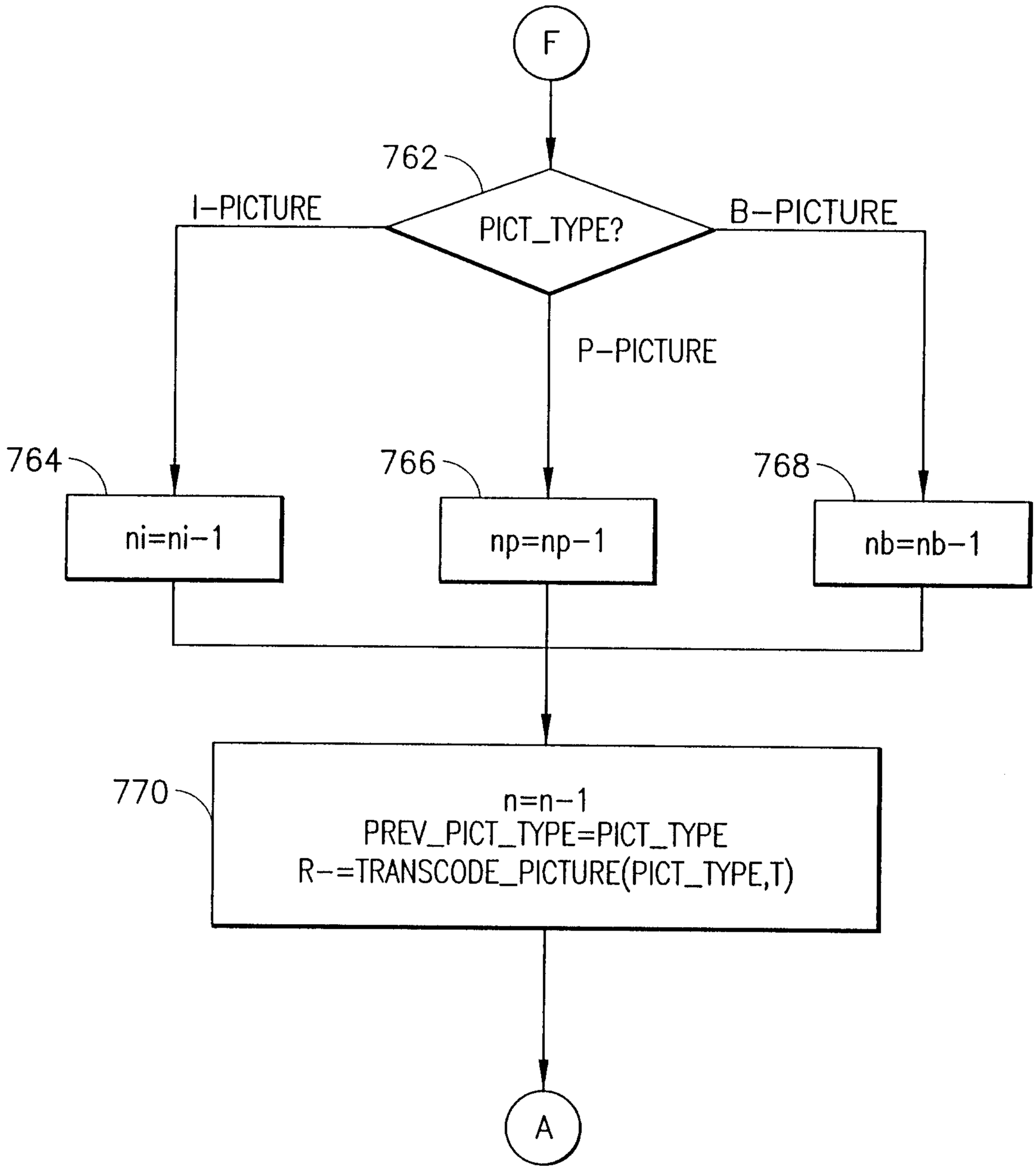

F
762
PICT_TYPE?
I–PICTURE
P–PICTURE
B–PICTURE
764
ni=ni–1
766
np=np–1
768
nb=nb–1
770
n=n–1
PREV_PICT_TYPE=PICT_TYPE
R–=TRANSCODE_PICTURE(PICT_TYPE,T)
A

FIG.7d