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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 July 2001 (05.07.2001)

PCT

(10) International Publication Number
WO 01/48620 A1

(51) International Patent Classification[7]: G06F 13/28

(21) International Application Number: PCT/US00/35455

(22) International Filing Date:
28 December 2000 (28.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/173,459 29 December 1999 (29.12.1999) US

(71) Applicant *(for all designated States except US)*: **THE JOHNS HOPKINS UNIVERSITY** [US/US]; Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, MD 20723-6099 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **BADE, Paul** [US/US]; 3717 Park Overlook Court, Ellicott City, MD 21042 (US). **KAHN, Steven** [US/US]; 14130 Flint Rock Road, Rockville, MD 20853 (US). **VERVEN, David** [US/US]; 9453 Farewell Road, Columbia, MD 21045 (US).

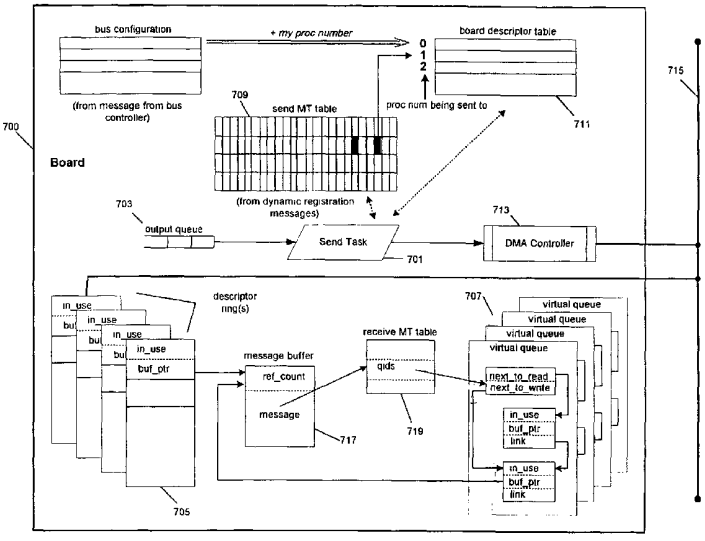
(74) Agents: **COOCH, Francis** et al.; The Johns Hopkins University, Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, MD 20723-6099 (US).

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

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

[Continued on next page]

(54) Title: SYSTEM, METHOD, AND COMPUTER PROGRAM PRODUCT FOR HIGH SPEED BACKPLANE MESSAGING



(57) Abstract: A system and method of enhanced backplane messaging among a plurality of computer boards communicating over a common bus uses a set of pre-allocated buffers on each computer board to receive messages from other computer boards. Each sending computer board is represented on each remote computer board by a descriptor ring with pointers to pre-allocated buffers on that remote computer board. When a sending computer board has a message to deliver to a remote computer board, the sending computer board uses its DMA controller to transfer the message into the pre-allocated buffers on the remote computer board. The sending computer board also sends a mailbox interrupt to the remote computer board. The remote computer board interrupt handler searches its descriptor rings and manipulates a series of pointers to move messages from the descriptor rings to the intended receiving application(s). Pointer manipulation is also used to replenish the descriptor ring(s) with empty buffer(s). In addition, chained DMA transfers are used to eliminate any data transfers by the processor itself across the bus. The chained DMA transfers transfer messages to the pre-allocated buffers, set flags indicating that a message is present in a buffer, send a mailbox interrupt to the remote computer board, and read back the address of the next pre-allocated buffer for that descriptor ring.

WO 01/48620 A1

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

Published:

— *With international search report.*

— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

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

TITLE OF THE INVENTION

System, Method, and Computer Program Product For High Speed Backplane Messaging

FIELD OF THE INVENTION

The present invention relates generally to a system and method for exchanging messages among a plurality of computer boards connected via a bus on a common backplane.

BACKGROUND OF THE INVENTION

Conventional backplane messaging schemes for exchanging messages among processors within a network utilizing a shared memory interconnect (bus) require a processor that wants to send a message (data packet) using the bus to request an empty buffer from the remote computer board processor, then transfer the data packet, and finally to notify the remote computer board processor of the arrival of the data packet. The overhead involved in this form of handshake for buffer allocation/de-allocation can significantly affect network performance.

SUMMARY OF THE INVENTION

A system and method of enhanced backplane messaging among a plurality of computer boards communicating over a common bus uses a set of pre-allocated buffers on each computer board to receive messages from other computer boards. Each sending computer board is represented on each remote computer board by a descriptor ring with pointers to pre-allocated buffers on that remote computer board. When a sending computer board has a message to deliver to a remote computer board, the sending computer board uses its DMA controller to transfer the message into the pre-allocated buffers on the remote computer board. The sending computer board also sends a mailbox interrupt to the remote computer board. The remote computer board interrupt handler searches its descriptor rings and manipulates a series of pointers to move messages from the descriptor rings to the intended receiving application(s). Pointer manipulation is also used to replenish the descriptor ring(s) with empty buffer(s). As a practical matter, pointer manipulation eliminates repeated copying of a message. Moreover, the use of pre-allocated buffers on each remote computer board achieves a significant performance boost over the more conventional technique of buffer request and assignment.

In addition, chained DMA transfers are used to eliminate any data transfers by a computer board processor (CPU) across the bus. The chained DMA transfers transfer messages to the pre-allocated buffers,

set flags indicating that a message is present in a buffer, send a mailbox interrupt to the remote computer board, and read back the address of the next pre-allocated buffer for that descriptor ring.

Other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE FIGURES

FIGURE 1 is a physical representation of the context and environment of the messaging scheme of the present invention.

FIGURE 2 illustrates an individual computer board and the high level processes or functions running on the computer board with respect to the messaging scheme of the present invention.

FIGURE 3 is a high level flow diagram for the initialization and registration process.

FIGURE 4 is a high level flow diagram for the sending process.

FIGURE 5 is a high level flow diagram for the receiving process.

FIGURE 6 is a more detailed message diagram for the initialization and registration process.

FIGURE 7 illustrates a more detailed schematic of the send and receive processes as they reside on a computer board.

FIGURE 8 illustrates examples of data structures used by the messaging scheme of the present invention.

FIGURE 9 is a detailed flow diagram for the sending process.

FIGURE 10 is a detailed flow diagram for the receiving process.

DETAILED DISCLOSURE OF THE INVENTION

FIGURE 1 is a physical representation of the context and environment of the messaging scheme of the present invention. The purpose of the present invention is to provide a system and method for exchanging messages among a plurality of computer boards connected via a backplane bus. The terms backplane and bus may be used interchangeably throughout this description to refer to the means that connect the various computer boards. The term message refers to data or data packets that can be grouped together. Thus, the phrase messaging scheme generally refers to a scheme for sending and receiving data across a common bus or backplane. **FIGURE 1** illustrates a cabinet **107** having three racks. A rack is a combination of a set of computer boards **101** connected to a common backplane **103**. A bus **105** connects a set of computer boards **101** such that messages can be exchanged among the computer boards **101** connected to such a bus **105**.

FIGURE 2 illustrates an individual computer board **201** and the high level processes or functions running on the computer board with respect to the messaging scheme of the present invention. It is to be understood that each computer board likely has other processes and functions running. Only the processes and functions relevant to the present invention are shown, however. There are three processes or functions that comprise the

messaging scheme of the present invention. The first is an initialization and registration process **203**. The initialization and registration process **203** performs preliminary tasks necessary to prepare the computer board **201** for message exchanging with other computer boards. Another process is the send process **205** that is responsible for sending messages to other computer boards that are connected to the common backplane. The messages are obtained from application output queues residing on the computer board. The other process is a receive process **207** that is responsible for receiving messages from other computer boards that are connected to the common backplane and distributing the messages to intended application(s) on the local computer board. A bus **209** is shown to illustrate that each process has access to the bus in order to be able to send or receive data.

FIGURE 3 illustrates a registration process. There are two primary tasks with respect to initializing a computer board to be able to exchange messages with other computer boards. One task is to initialize at least one descriptor ring **301** on the computer board. Descriptor rings are used to receive messages from other computer boards. The other task is to pre-allocate a plurality of buffers **303**. The buffers will be used to store incoming messages. Pre-allocation of the buffers is a significant part of the messaging scheme of the present invention in order to provide a known destination for a message transfer as opposed to requiring the sender to request a new buffer prior to each message transfer. Once the descriptor rings have been initialized and buffers have been pre-allocated, the computer board completes the

registration process by notifying all other computer boards **305** connected to the backplane that it has initialized its descriptor rings and pre-allocated buffers.

Those of ordinary skill in the art can readily devise and implement alternate registration and initialization processes without departing from the spirit or scope of the present invention. Thus, any initialization and registration processes described herein are illustrative and not intended to limit the core sending and receiving techniques presented in this application.

FIGURE 4 illustrates the sending process generally. A send task is continually in a wait state waiting for the next message it is to send **401**. When a message to send is present **403** the send task extracts the message from an output queue of an on-board application **405**. The send task next determines the availability of pre-allocated buffers on a remote computer board in which the receiving application resides **407**. The buffer availability information is typically gathered by the DMA chain at the end of the last data transfer to the receiving computer board. Next, the send task instructs the DMA controller on the sending computer board to transfer the message to the remote computer board **409**, interrupt the receiving computer board with a mailbox interrupt **411**, and gather information of the availability of message buffers for the next transfer, i.e. read back the descriptor for the next message buffer **413**. The DMA is, in effect, a message passing co-processor that relieves the sending computer board CPU from having to perform any transfers over the bus. If the CPU were to

perform transfers, it could block while waiting for access to a busy bus. Thus, DMA chaining is utilized to enhance the speed and efficiency of the present invention messaging scheme.

FIGURE 5 illustrates the receiving process generally. An interrupt occurs notifying the receive process on the receiving computer board that a message is present **501** in one of the receiving computer board descriptor rings. The receive process searches among the descriptor ring(s) on the receiving computer board for a full message buffer **503**. Once the descriptor ring containing the pointer to the message buffer is found, the receive process extracts the message buffer **505** from the descriptor ring and delivers the message **507** to the intended application using well known pointer manipulation techniques. After delivery, the receive process replenishes the descriptor ring **509** with an empty buffer.

FIGURE 6 is a more detailed message diagram for the initialization and registration process of each computer board that is connected to a common bus on a backplane. The process is described with respect to executing across a VMEbus. One of ordinary skill in the art can readily apply the concepts and principles of the following description and apply same to other buses. Thus, the present invention is not to be construed as limited to a VMEbus implementation.

As a computer board powers up it maps an area of its memory to a window on the VMEbus. At some point following power up, a bus controller probes the bus for computer boards that are connected to the bus. The bus controller then constructs its own bus configuration

table as a result of its probing the bus for computer boards.

Meanwhile, each computer board initializes its own hardware description block and its plurality of receive descriptor rings. The computer board then instructs its send task to create a virtual message queue that is registered to receive bus configuration and dynamic registration messages from the bus controller. Each computer board concludes initialization by setting a flag to notify the bus controller that the computer board is fully initialized.

The bus controller waits for a period of time for all of the computer boards present on the bus to perform the initialization process. Next, the bus controller pre-loads its computer board descriptor table with the board descriptors corresponding to all of the computer boards. The bus controller also pre-loads the send MT table used for sending the bus configuration and dynamic registration messages to all of the computer boards. At this point, the bus controller issues the bus configuration message to all the computer boards.

When each computer board receives the bus configuration message, it builds its own board descriptor table by reading the configuration information available on each computer board. Information in a board descriptor table is used to access a unique receive descriptor ring on a remote computer board when sending message(s) to a remote computer board.

Whenever the bus controller (or any computer board) receives a dynamic registration message, it updates its send MT table with the messages registered for by the computer board that sent the dynamic registration

message. The send task on each computer board is responsible for receiving the dynamic registration message(s) and updating the send MT table. Once the bus controller has collected all of the registration complete messages from each computer board, it sends an initialization complete message to all computer boards to complete the entire system initialization process. Thus, the bus controller has several responsibilities including probing the VMEbus for computer boards, sending bus configuration messages, and sending the initialization complete message.

FIGURE 7 illustrates a more detailed schematic of the send and receive hardware, software, and processes as they reside on a computer board. For ease of illustration, **FIGURE 7** shows a single computer board **700** which includes the sending and receiving components. Each computer board is identical with respect to the message exchanging scheme of the present invention. Thus, reference to **FIGURE 7** with respect to sending a message from one computer board to another computer board is appropriate since the receiving portion of the computer board shown in **FIGURE 7** can be thought of as existing on another computer board connected via the bus.

The sending process is managed by a send task **701**. The send task is communicable with an output queue **703** into which an application places messages to be delivered to an application on a remote computer board. The receiving process includes an array of receive descriptor rings **705** (one ring per sender) to buffer incoming messages and any number of virtual input queues **707** from which applications retrieve their messages. From a receiving application's point of view the virtual input

queues **707** support the publish/subscribe messaging model. Thus, an application can register for any number of message labels without having to know any information about the sender and an application can block on a queue while awaiting arrival of a message.

Tracking a message as it travels from a source to its destination provides a more detailed understanding of board-to-board message transfers. Each computer board includes a send task **701** which extracts messages from the application's output queue (or queues). The send task doesn't actually copy the message. Rather, it peeks at the message type, and then indexes into a send MT (message type) table **709** to determine the destination of the message. Each row of the send MT table **709** table can be viewed as a bit array, where each bit that is set corresponds to the processor number of each remote computer board that has one or more virtual input queues **707** registered for the message.

A board descriptor table **711** is used to maintain all of the pertinent information needed for message delivery to each remote computer board. The board descriptor table **711** includes an entry for each remote computer board and is indexed by processor number. The board descriptor table **711** is composed of information copied from each remote computer board hardware descriptor. Information in the board descriptor table **711** includes how to interrupt the remote computer board, what the computer board address is (in both the processor's address space and on the VMEbus), and the computer board type.

The send task **701** thus consults the send MT table **709** to determine which computer boards are to receive a message. The send task **701** also obtains a board

descriptor for each remote computer board that is to receive a message. With this information (as well as some additional information), the send task **701** can program an onboard DMA controller **713** to transfer the message across a bus **715**. Note that if multiple virtual input queues **707** on a remote computer board have registered for the same message, only one transfer across the bus is necessary.

The DMA transfer is made into an empty memory buffer **717**. The message buffer **717** is pointed to by the descriptor ring **705**, specifically the *buf_ptr* of the next descriptor in the descriptor ring **705** on the receiving computer board. In addition, an *in_use* flag is set for the descriptor ring **705**. The sending process knows the message buffer on the receiving computer board is available because it had previously determined that the *in_use* field for that descriptor ring **705** was not set.

Once the DMA transfer is completed, the receiver is notified via an interrupt. Within an interrupt handler, the receiver indexes into a receive MT table **719**, based on the message type, and obtains a list of local virtual input queues **707** that have registered for the message. The local virtual input queue may also be referred to as an application input queue. For each virtual input queue **707**, a buffer pointer of the queue entry pointed at by a next to write pointer is modified to point to the message buffer **717**, the next to write pointer is modified, and the reference count field in the message buffer **717** is incremented by one. Finally, the in use field of the virtual queue **707** entry is set, the buffer pointer of the descriptor ring **705** is set to point to a new message

buffer **717** (obtained from a global free list), and the in use field of the descriptor ring **705** entry is cleared.

Later, when an application de-queues a message, the in use field of the virtual queue **707** entry is cleared, the virtual queue **707** next to read pointer is modified, and the reference count field of the message buffer **717** is decremented. Because multiple tasks might be de-queuing the same message buffer **717**, updating of the reference count is protected, e.g., by task or interrupt locking. When the reference count goes to zero, the message buffer **717** is placed on the global free list.

FIGURE 8 illustrates examples of data structures used by the messaging scheme of the present invention. A board descriptor **801** can be thought of as the handle to all of the information with respect to a remote computer board that is used by the sending computer board in order to deliver a message to a remote computer board. The board descriptor **801** includes information on the addressing **803** of the remote computer board, a pointer **805** to the sending computer board ring control block, information on the mailbox interrupt **807** to be used upon delivery of a message, and a pointer **809** to a list of all the messages that have been registered for on this computer board, along with their priorities **811**.

The sending computer board ring control block structure **813** contains all the information about the remote descriptor ring that is needed for message transfers. Note that the *next descriptor* field **815** includes a copy of the next descriptor on the remote computer board, as it was copied at the end of the previous DMA chain transfer operation. For efficiency, the next descriptor (containing a pointer to the next

dynamically allocated buffer) is copied during the DMA chain operation, since the *in_use* flag is most likely not set, which indicates that the buffer pointer *(buf_ptr)* is valid.

The receiving computer board descriptor ring control block **817** on the remote computer board, includes information about the receive descriptor ring(s) uniquely associated with the sending computer board. Also, the sending **813** and receiving **817** ring control block structures include fields to monitor ring overflow conditions (dropped messages) and the high water mark for messages in the ring.

FIGURE 9 illustrates the logic flow of the sending process on a given computer board. The computer board's send task extracts messages from an application's output queue or queues **901**. The send task determines the message type and then indexes into the send MT table **903** that it created during the initialization process to determine which computer board is to receive the message. The send task obtains a board descriptor of the computer board that is to receive the message **905**. The send task then programs the DMA controller to transfer the message **907** to the receiving computer board across the bus. The DMA transfer is sent to a pre-allocated empty message buffer on the receiving computer board **909**. Recall, that message buffers were pre-allocated on all computer boards during the initialization and registration process.

Once the transfer is complete, the sending DMA controller sends a mailbox interrupt **911** to the receiving computer board to initiate the receive process on the receiving computer board. Lastly, the DMA controller reads back the address of the next message buffer

according to the receiving computer board descriptor ring **913**.

The DMA controller chains together message transfers across the bus in order to expedite the message sending process when multiple destinations are involved. With respect to the present invention, there are multiple elements in a DMA chain executed by the DMA controller for each destination. One element writes the body and *in_use* bit of message k to a queue on the remote computer board. Another element writes to the remote computer board mailbox interrupt location. Typically, the last element calls for reading the in-use bit of message k+1. At the end of the chain, after the VMEbus has been released, the sending DMA controller interrupts the local CPU. The local CPU then sets up the sending DMA controller for the next transfer while another computer board is free to use the backplane/bus. This scheme calls for one CPU interrupt per out-going message to ensure that the local computer board CPU does not have to poll the DMA controller waiting for a transfer complete status from the DMA controller. The overhead associated with the one CPU interrupt per out-going message may be reduced by having the CPU set up for all pending out-going messages each time it is interrupted by the DMA controller with a transfer complete status. The multiple out-going messages would simply be added in to the DMA controller chain. This is most practical when multiple messages are targeted at different receiver queues.

FIGURE 10 illustrates the logic flow of the receiving process on a given computer board. Following the initialization process, each computer board is ready for communications. As the last part of the DMA chain,

the receiver is notified of the message receipt via a mailbox interrupt **1001**. The receiver then indexes, based on message type, into its receive MT table and obtains a list of local virtual queues that have registered to receive the message **1003**. The buffer pointer field within the virtual queue that is pointed to by the next to write field is modified to point to the message buffer containing the message at issue **1005**. The virtual queue next to write pointer is then modified and the ref count field in the message buffer is incremented **1007** and the *in_use* flag of the virtual queue is set **1009**. At this point, the buffer pointer field of the descriptor ring is set to point to a new message buffer **1011** and the *in_use* flag of the descriptor ring is cleared **1013**.

The receiving application on the receiving computer board de-queues the just queued message on the virtual queue **1015**. After de-queuing, the *in_use* flag of the virtual queue is cleared **1017** and the virtual queue next to read pointer is modified and the ref count field of the message buffer is decremented **1019**. When the ref count value of the message buffer decrements back to zero, a message buffer is placed on the global free list **1021**.

There are several advantages realized by the present invention. Foremost is the decoupling of communications between multiple senders accessing a common receiver. Using pre-allocated buffers eliminates many message buffer management issues associated with transferring data among processors on a bus. DMA (Direct Memory Access) hardware is used for all bus transfers ensuring that the processors never block while waiting for bus access. Thus, the CPU is not responsible for performing

transfers. Multiple point to point connections can be simultaneously active between multiple processors, and mutual exclusion capabilities such as Test-And-Set are not needed. Moreover, all message transfers are done with write commands, which are significantly more efficient than reading across a bus.

It is to be understood that the present invention illustrated herein is readily implementable by those of ordinary skill in the art as a computer program product having a medium with a computer program embodied thereon. The computer program product is capable of being loaded and executed on the appropriate computer processing device(s) in order to carry out the method or process steps described. Appropriate computer program code in combination with hardware implements many of the elements of the present invention. This computer code is often stored on storage media. This media can be a diskette, hard disk, CD-ROM, optical storage media, or tape. The media can also be a memory storage device or collection of memory storage devices such as read-only memory (ROM) or random access memory (RAM). Additionally, the computer program code can be transferred to the appropriate hardware over some type of data network.

The present invention has been described, in part, with reference to flowchart or logic flow diagrams. It will be understood that each block of the flowchart diagrams or logic flow diagrams, and combinations of blocks in the flowchart diagrams or logic flow diagrams, can be implemented by computer program instructions.

These computer program instructions may be loaded onto a general purpose computer, special purpose computer, or other programmable data processing apparatus

to produce a machine, such that the instructions which execute on the computer or other programmable data processing apparatus create means for implementing the functions specified in the flowchart block or blocks or logic flow diagrams.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the flowchart blocks or logic flow diagrams. The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart blocks or logic flow diagrams.

Accordingly, block(s) of flowchart diagrams and/or logic flow diagrams support combinations of means for performing the specified functions, combinations of steps for performing the specified functions and program instruction means for performing the specified functions. It will also be understood that each block of flowchart diagrams and/or logic flow diagrams, and combinations of blocks in flowchart diagrams and/or logic flow diagrams can be implemented by special purpose hardware-based computer systems that perform the specified functions or

steps, or combinations of special purpose hardware and computer instructions.

In the following claims, any means-plus-function clauses are intended to cover the structures described herein as performing the recited function and not only structural equivalents but also equivalent structures. Therefore, it is to be understood that the foregoing is illustrative of the present invention and is not to be construed as limited to the specific embodiments disclosed, and that modifications to the disclosed embodiments, as well as other embodiments, are intended to be included within the scope of the appended claims. The invention is defined by the following claims, with equivalents of the claims to be included therein.

CLAIMS:

1. A method of backplane messaging among a plurality of computer boards connected via a bus, each computer board including a processor and a DMA controller and each computer board capable of sending and receiving messages, said method comprising:

initializing said plurality of computer boards, said initialization for the purpose of preparing each computer board to communicate with the other computer boards;

sending messages, utilizing a the DMA controller included on the computer board and not the processor of the computer board, from any one of the plurality of computer boards to any other of the plurality of computer boards; and

receiving messages on any one of the plurality of computer boards sent from any other of the plurality of computer boards, said messages received into pre-allocated message buffers on said computer boards, said pre-allocated message buffers pointed to by a descriptor ring,

wherein de-coupling of communications between multiple sending computer boards attempting to access a common receiving computer board is achieved.

2. The method of claim 1 wherein said sending step for a computer board comprises:

determining the message type of a message to be sent to a receiving computer board;

indexing into a send table to determine the identity of the receiving computer board;

obtaining board descriptor data for the receiving computer board;

having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

having the DMA controller interrupt the receiver on a receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board.

3. The method of claim 2 wherein said send table is comprised of bit array in which each bit corresponds to a processor number for each computer board.

4. The method of claim 2 wherein said board descriptor data for a computer board is comprised of information including an entry for the computer board, how to interrupt the computer board, the address of the computer board in its own processor space, the address of the computer board on the bus, and the type of the computer board.

5. The method of claim 2 wherein said having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining.

6. The method of claim 5 wherein said having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining utilizing write commands only.

7. The method of claim 1 wherein said receiving step for a computer board comprises:

searching the descriptor ring for a full message buffer containing a received message, said descriptor ring corresponding to a sending computer board;

indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

replenishing the descriptor ring with an empty message buffer.

8. The method of claim 7 wherein a plurality of descriptor rings corresponding to the other computer boards are implemented on a computer board.

9. A method of backplane messaging between a sending computer board having a processor and a receiving computer board having a processor that are connected via a bus, the sending computer board including a DMA controller, said method comprising:

on the sending computer board:

determining the message type of a message to be sent to the receiving computer board;

indexing into a send table to determine the identity of the receiving computer board;

obtaining board descriptor data for the receiving computer board;

having the DMA controller and not the sending board processor transfer the message to a pre-allocated buffer on the receiving computer board;

having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

having the DMA controller interrupt a receiver on the receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board; and

on the receiving computer board:

searching a descriptor ring for a full message buffer containing a received message, said descriptor ring corresponding to the sending computer board;

indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

replenishing the descriptor ring with an empty message buffer.

10. On a computer board having a DMA controller and a processor, a method of sending a message over a bus, said method comprising:

determining the message type of a message to be sent to a receiving computer board;

indexing into a send table to determine the identity of the receiving computer board;

obtaining board descriptor data for the receiving computer board;

having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

having the DMA controller interrupt a receiver on the receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board.

11. On a computer board having a processor, a method of receiving a message from a bus, said method comprising:

searching a descriptor ring for a full message buffer containing a received message;

indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

replenishing the descriptor ring with an empty message buffer.

12. A system for backplane messaging among a plurality of computer boards connected via a bus, each computer board including a processor and a DMA controller and each computer board capable of sending and receiving messages, said system comprising:

means for initializing said plurality of computer boards, said initialization for the purpose of preparing each computer board to communicate with the other computer boards;

means for sending messages, utilizing a the DMA controller included on the computer board and not the processor of the computer board, from any one of the plurality of computer boards to any other of the plurality of computer boards; and

means for receiving messages on any one of the plurality of computer boards sent from any other of the plurality of computer boards, said messages received into pre-allocated message buffers on said computer boards, said pre-allocated message buffers pointed to by a descriptor ring,

wherein de-coupling of communications between multiple sending computer boards attempting to access a common receiving computer board is achieved.

13. The system of claim 12 wherein said means for sending comprises:

means for determining the message type of a message to be sent to a receiving computer board;

means for indexing into a send table to determine the identity of the receiving computer board;

means for obtaining board descriptor data for the receiving computer board;

means for having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

means for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

means for having the DMA controller interrupt the receiver on a receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board.

14. The system of claim 13 wherein said send table is comprised of bit array in which each bit corresponds to a processor number for each computer board.

15. The system of claim 13 wherein said board descriptor data for a computer board is comprised of information including an entry for the computer board, how to interrupt the computer board, the address of the computer board in its own processor space, the address of the computer board on the bus, and the type of the computer board.

16. The system of claim 13 wherein said means for having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining.

17. The system of claim 16 wherein said means for having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining utilizing write commands only.

18. The system of claim 12 wherein said means for receiving comprises:

means for searching the descriptor ring for a full message buffer containing a received message, said descriptor ring corresponding to a sending computer board;

means for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

means for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

means for replenishing the descriptor ring with an empty message buffer.

19. The system of claim 18 wherein a plurality of descriptor rings corresponding to the other computer boards are implemented on a computer board.

20. A system for backplane messaging between a sending computer board having a processor and a receiving computer

board having a processor that are connected via a bus, the sending computer board including a DMA controller, said system comprising:

on the sending computer board:

means for determining the message type of a message to be sent to the receiving computer board;

means for indexing into a send table to determine the identity of the receiving computer board;

means for obtaining board descriptor data for the receiving computer board;

means for having the DMA controller and not the sending board processor transfer the message to a pre-allocated buffer on the receiving computer board;

means for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

means for having the DMA controller interrupt a receiver on the receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board; and

on the receiving computer board:

means for searching a descriptor ring for a full message buffer containing a received

message, said descriptor ring corresponding to the sending computer board;

means for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

means for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

means for replenishing the descriptor ring with an empty message buffer.

21. On a computer board having a DMA controller and a processor, a system for sending a message over a bus, said system comprising:

means for determining the message type of a message to be sent to a receiving computer board;

means for indexing into a send table to determine the identity of the receiving computer board;

means for obtaining board descriptor data for the receiving computer board;

means for having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

means for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

means for having the DMA controller interrupt a receiver on the receiving computer board in order to notify

a receiving computer board that a message has been placed into a message buffer on the receiving computer board.

22. On a computer board having a processor, a system for receiving a message from a bus, said system comprising:

means for searching a descriptor ring for a full message buffer containing a received message;

means for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

means for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

means for replenishing the descriptor ring with an empty message buffer.

23. A computer program product for backplane messaging among a plurality of computer boards connected via a bus, each computer board including a processor and a DMA controller and each computer board capable of sending and receiving messages, the computer program product having a medium with a computer program embodied thereon, the computer program product comprising:

computer program code for initializing said plurality of computer boards, said initialization for the purpose of preparing each computer board to communicate with the other computer boards;

computer program code for sending messages, utilizing the DMA controller included on the computer board and not the processor of the computer board, from any one of the

plurality of computer boards to any other of the plurality of computer boards; and

computer program code for receiving messages on any one of the plurality of computer boards sent from any other of the plurality of computer boards, said messages received into pre-allocated message buffers on said computer boards, said pre-allocated message buffers pointed to by a descriptor ring,

wherein de-coupling of communications between multiple sending computer boards attempting to access a common receiving computer board is achieved.

24. The computer program product of claim 23 wherein said computer program code for sending comprises:

computer program code for determining the message type of a message to be sent to a receiving computer board;

computer program code for indexing into a send table to determine the identity of the receiving computer board;

computer program code for obtaining board descriptor data for the receiving computer board;

computer program code for having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

computer program code for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

computer program code for having the DMA controller interrupt the receiver on a receiving computer board in order to notify a receiving computer board that a message

has been placed into a message buffer on the receiving computer board.

25. The computer program product of claim 24 wherein said send table is comprised of bit array in which each bit corresponds to a processor number for each computer board.

26. The computer program product of claim 24 wherein said board descriptor data for a computer board is comprised of information including an entry for the computer board, how to interrupt the computer board, the address of the computer board in its own processor space, the address of the computer board on the bus, and the type of the computer board.

27. The computer program product of claim 24 wherein said computer program code for having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining.

28. The computer program product of claim 27 wherein said computer program code for having the DMA controller transfer the message to a pre-allocated buffer on the receiving computer board is achieved via DMA chaining utilizing write commands only.

29. The computer program product of claim 23 wherein said computer program code for receiving comprises:

computer program code for searching the descriptor ring for a full message buffer containing a received

message, said descriptor ring corresponding to a sending computer board;

computer program code for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

computer program code for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

computer program code for replenishing the descriptor ring with an empty message buffer.

30. The computer program product of claim 29 wherein a plurality of descriptor rings corresponding to the other computer boards are implemented on a computer board.

31. A computer program product for backplane messaging between a sending computer board having a processor and a receiving computer board having a processor that are connected via a bus, the sending computer board including a DMA controller, the computer program product having a medium with a computer program embodied thereon, said computer program product comprising:

on the sending computer board:

computer program code for determining the message type of a message to be sent to the receiving computer board;

computer program code for indexing into a send table to determine the identity of the receiving computer board;

computer program code for obtaining board descriptor data for the receiving computer board;

computer program code for having the DMA controller and not the sending board processor transfer the message to a pre-allocated buffer on the receiving computer board;

computer program code for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

computer program code for having the DMA controller interrupt a receiver on the receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board; and

on the receiving computer board:

computer program code for searching a descriptor ring for a full message buffer containing a received message, said descriptor ring corresponding to the sending computer board;

computer program code for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

computer program code for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

computer program code for replenishing the descriptor ring with an empty message buffer.

32. On a computer board having a DMA controller and a processor, a computer program product for sending a message over a bus, the computer program product having a medium with a computer program embodied thereon, said computer program product comprising:

computer program code for determining the message type of a message to be sent to a receiving computer board;

computer program code for indexing into a send table to determine the identity of the receiving computer board;

computer program code for obtaining board descriptor data for the receiving computer board;

computer program code for having the DMA controller and not the processor transfer the message to a pre-allocated buffer on the receiving computer board;

computer program code for having the DMA controller read back a descriptor ring address for the next message buffer on the receiving computer board; and

computer program code for having the DMA controller interrupt a receiver on the receiving computer board in order to notify a receiving computer board that a message has been placed into a message buffer on the receiving computer board.

33. On a computer board having a processor, a computer program product for receiving a message from a bus, the computer program product having a medium with a computer program embodied thereon, said computer program product comprising:

computer program code for searching a descriptor ring for a full message buffer containing a received message;

computer program code for indexing into a receive table to obtain a list of virtual queues that have registered for the received message;

computer program code for relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

computer program code for replenishing the descriptor ring with an empty message buffer.

34. A computer board for sending a message over a bus comprising:

a send table including identity information pertaining to at least one receiving computer board;

a processor for:

determining the message type of a message to be sent to said at least one receiving computer board;

indexing into said send table to determine the identity of said at least one receiving computer board;

obtaining board descriptor data for said at least one receiving computer board; and

a DMA controller for:

transferring the message to a pre-allocated buffer on said at least one receiving computer board;

reading back a descriptor ring address for the next message buffer on said at least one receiving computer board; and

interrupting a receiver task on said at least one receiving computer board in order to notify said at least one receiving computer board that a message has been placed into a message buffer on said at least one receiving computer board.

35. The computer board of claim 34 wherein said send table is comprised of bit array in which each bit corresponds to a processor number for each computer board.

36. The computer board of claim 34 wherein said board descriptor data for a computer board is comprised of information including an entry for the computer board, how to interrupt the computer board, the address of the computer board in its own processor space, the address of the computer board on the bus, and the type of the computer board.

37. The method of claim 34 wherein said transferring the message to a pre-allocated buffer on said at least one receiving computer board is achieved via DMA chaining.

38. The method of claim 37 wherein transferring the message to a pre-allocated buffer on said at least one receiving computer board is achieved via DMA chaining utilizing write commands only.

39. A computer board for receiving a message over a bus comprising:

a receive table including a list of virtual queues that have registered for a received message;

a processor for:

searching a descriptor ring for a message buffer containing a received message;

indexing into said receive table to obtain a list of virtual queues that have registered for the received message;

relaying the received message, via pointer manipulation, from the message buffer to the virtual queues that have registered for the received message; and

replenishing the descriptor ring with an empty message buffer.

FIG. 1

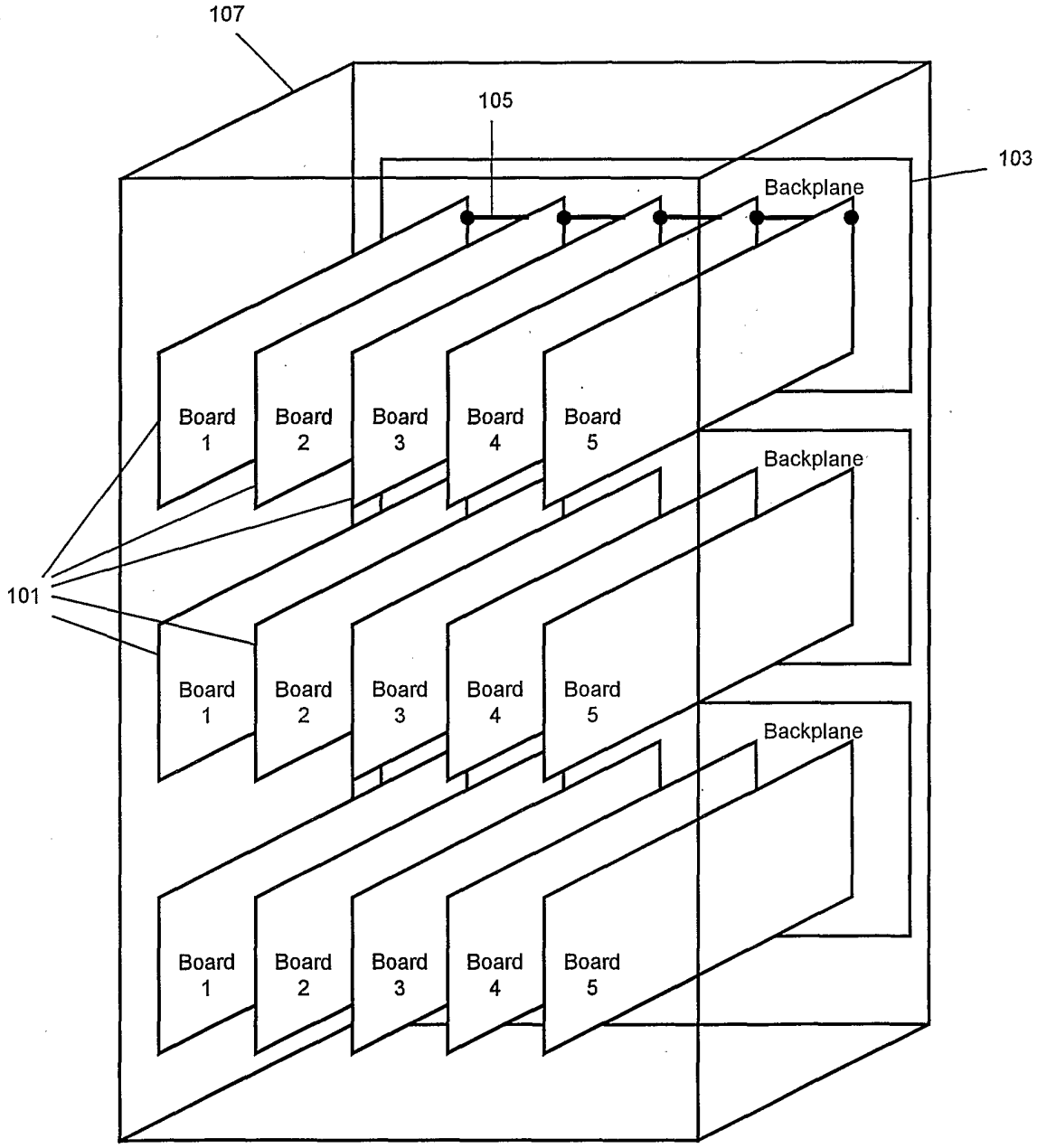

107
105
103
Backplane
Board 1
Board 2
Board 3
Board 4
Board 5
Backplane
Board 1
Board 2
Board 3
Board 4
Board 5
Backplane
Board 1
Board 2
Board 3
Board 4
Board 5
101

FIG. 2

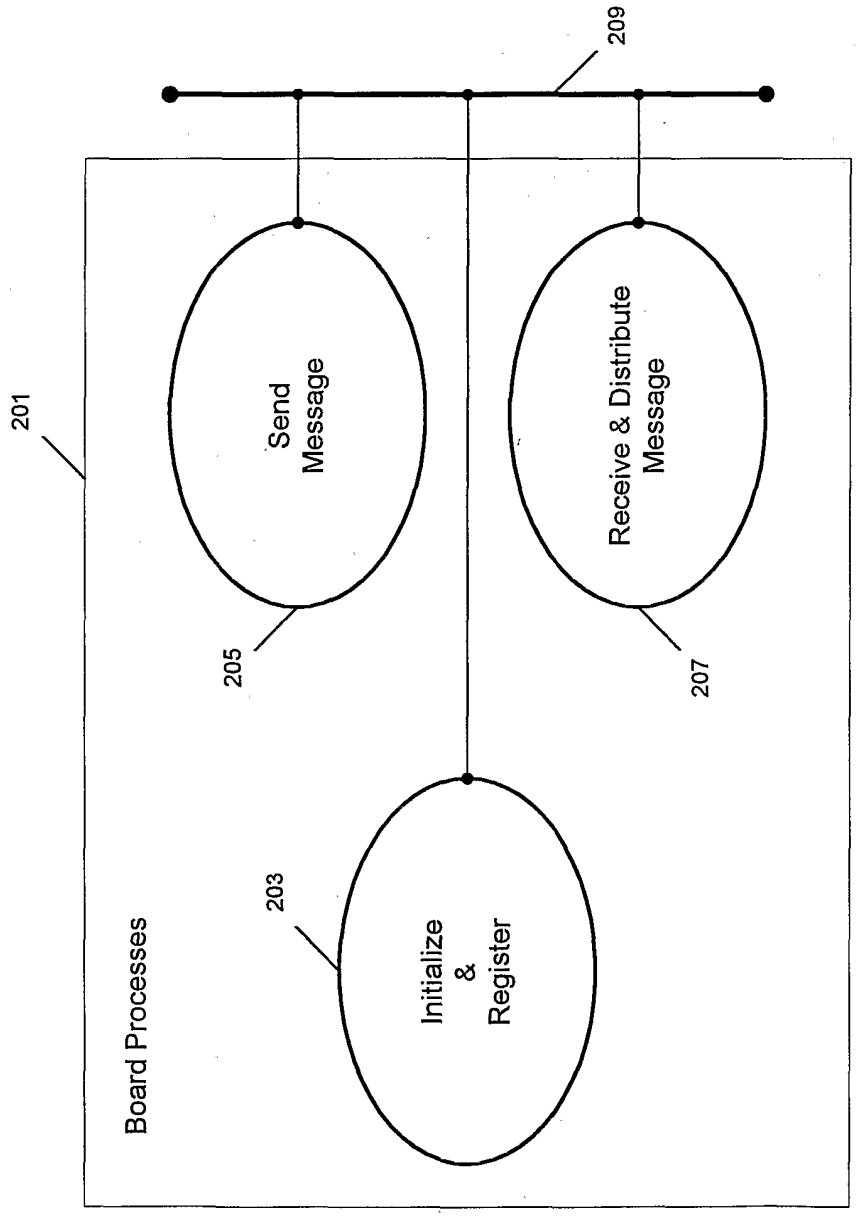

209
201
Send Message
Receive & Distribute Message
205
207
203
Initialize & Register
Board Processes

FIG. 3

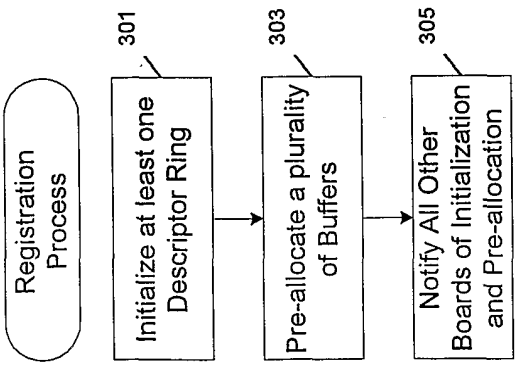


FIG. 4

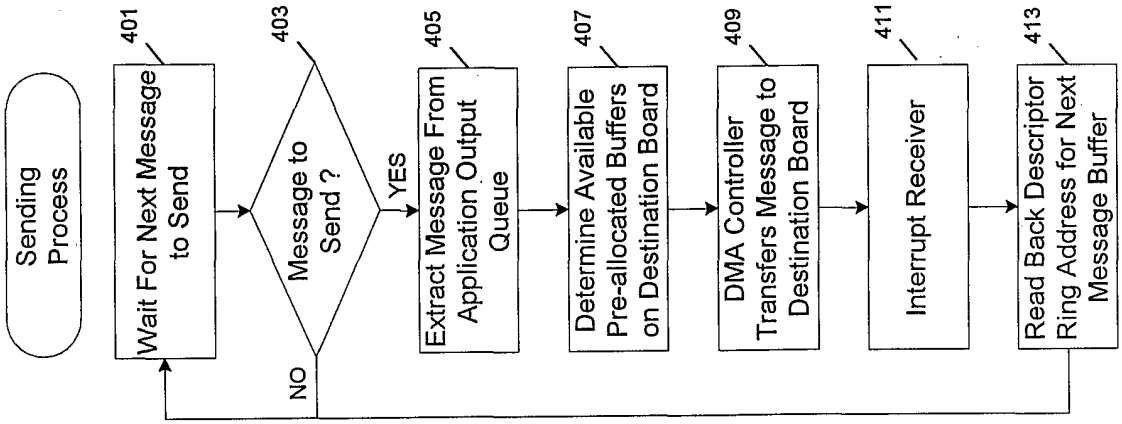


FIG. 5

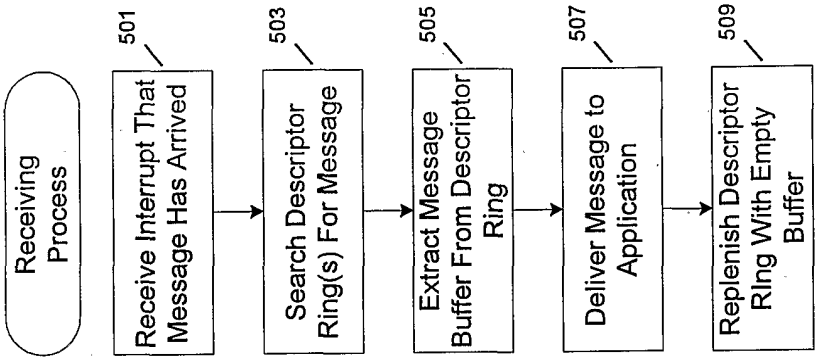


FIG. 6

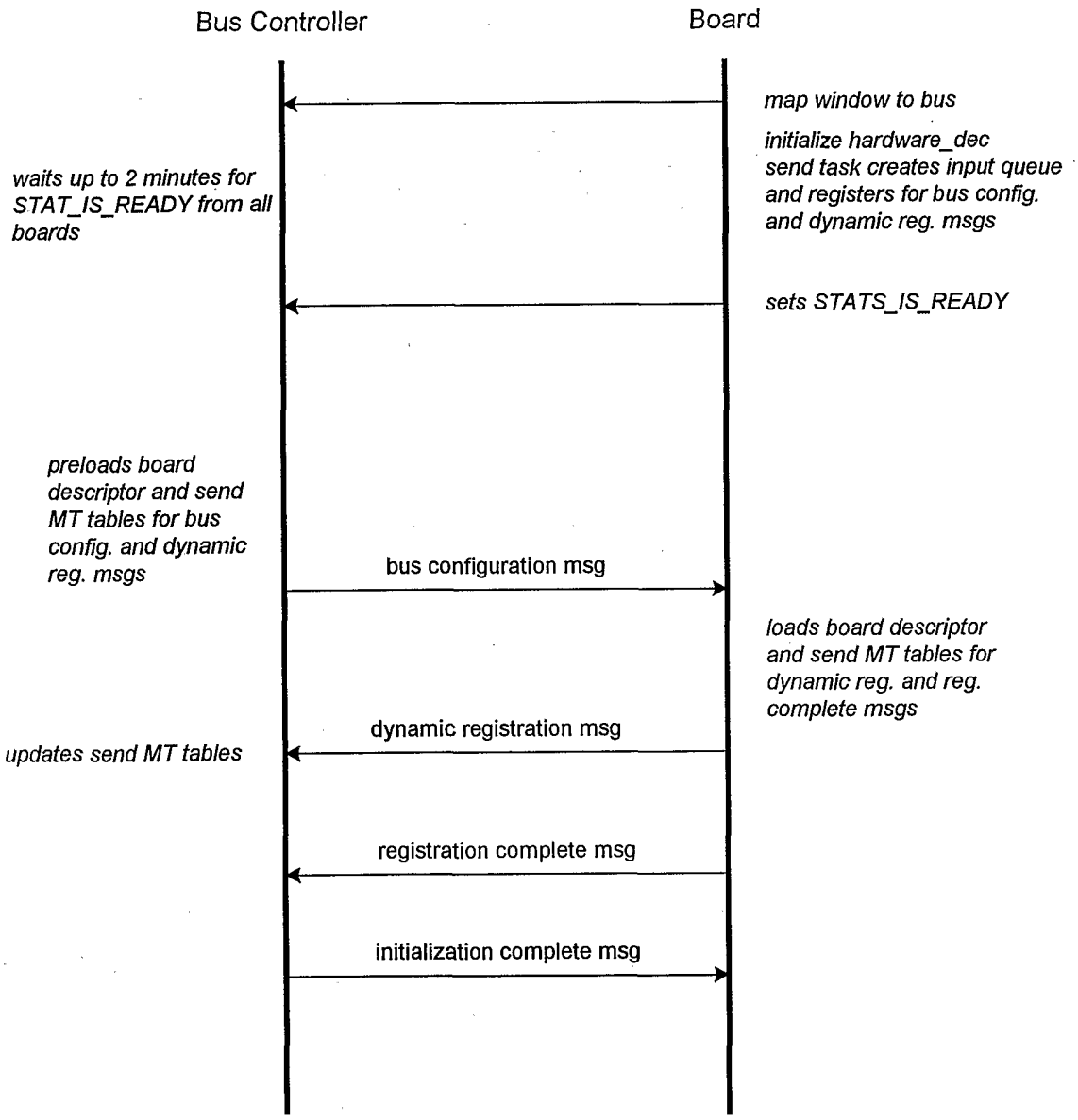

Bus Controller
Board
map window to bus
initialize hardware_dec
send task creates input queue
and registers for bus config.
and dynamic reg. msgs
waits up to 2 minutes for
STAT_IS_READY from all
boards
sets STATS_IS_READY
preloads board
descriptor and send
MT tables for bus
config. and dynamic
reg. msgs
bus configuration msg
loads board descriptor
and send MT tables for
dynamic reg. and reg.
complete msgs
dynamic registration msg
updates send MT tables
registration complete msg
initialization complete msg

FIG. 7

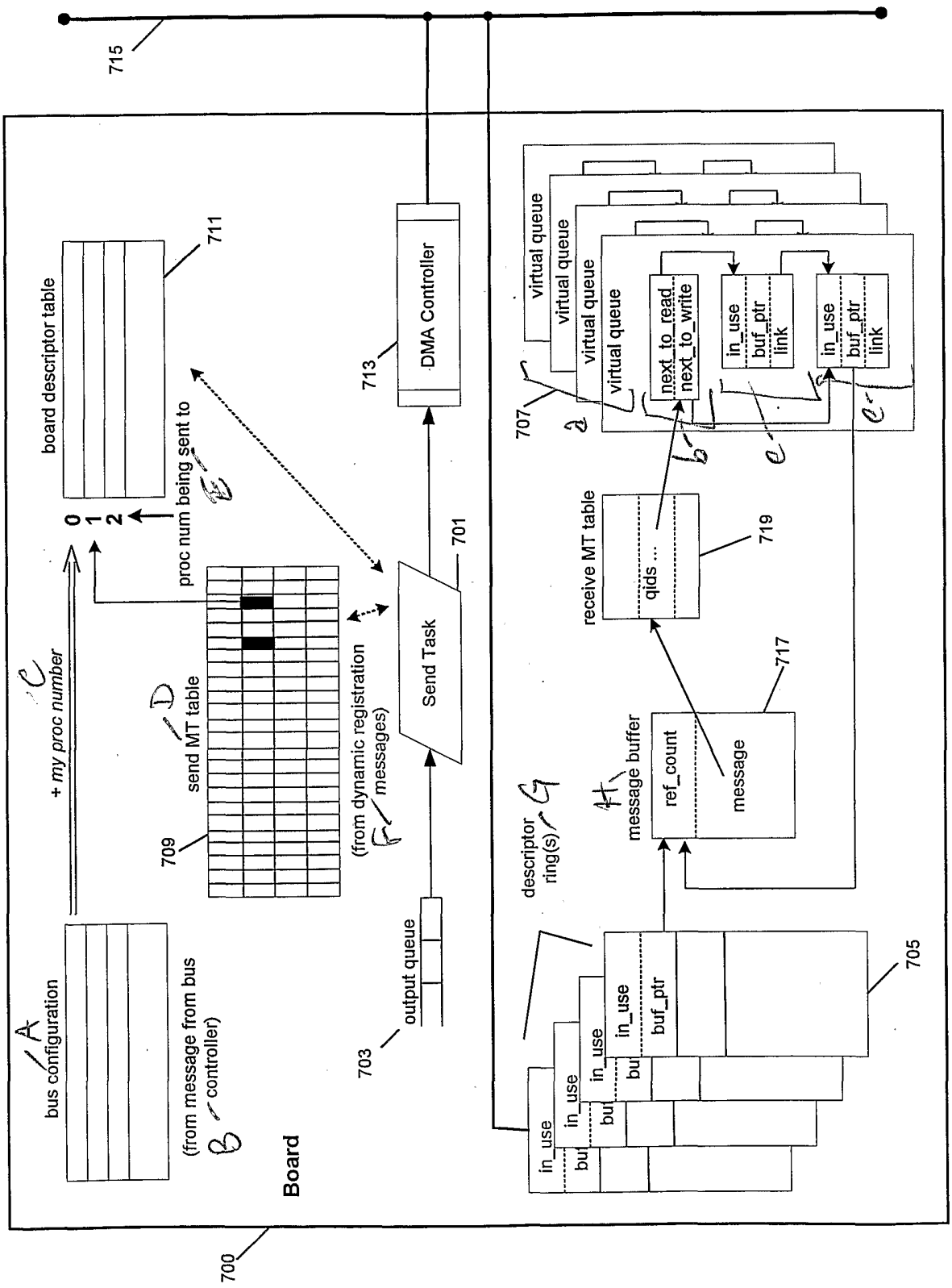

715
711
board descriptor table
0
1
2
proc num being sent to
+ my proc number
bus configuration
(from message from bus controller)
Board
700
709
send MT table
(from dynamic registration messages)
Send Task
701
713
DMA Controller
703
output queue
707
virtual queue
next_to_read
next_to_write
in_use
buf_ptr
link
receive MT table
qids ...
719
717
message buffer
ref_count
message
descriptor ring(s)
705
buf_ptr

FIG. 8

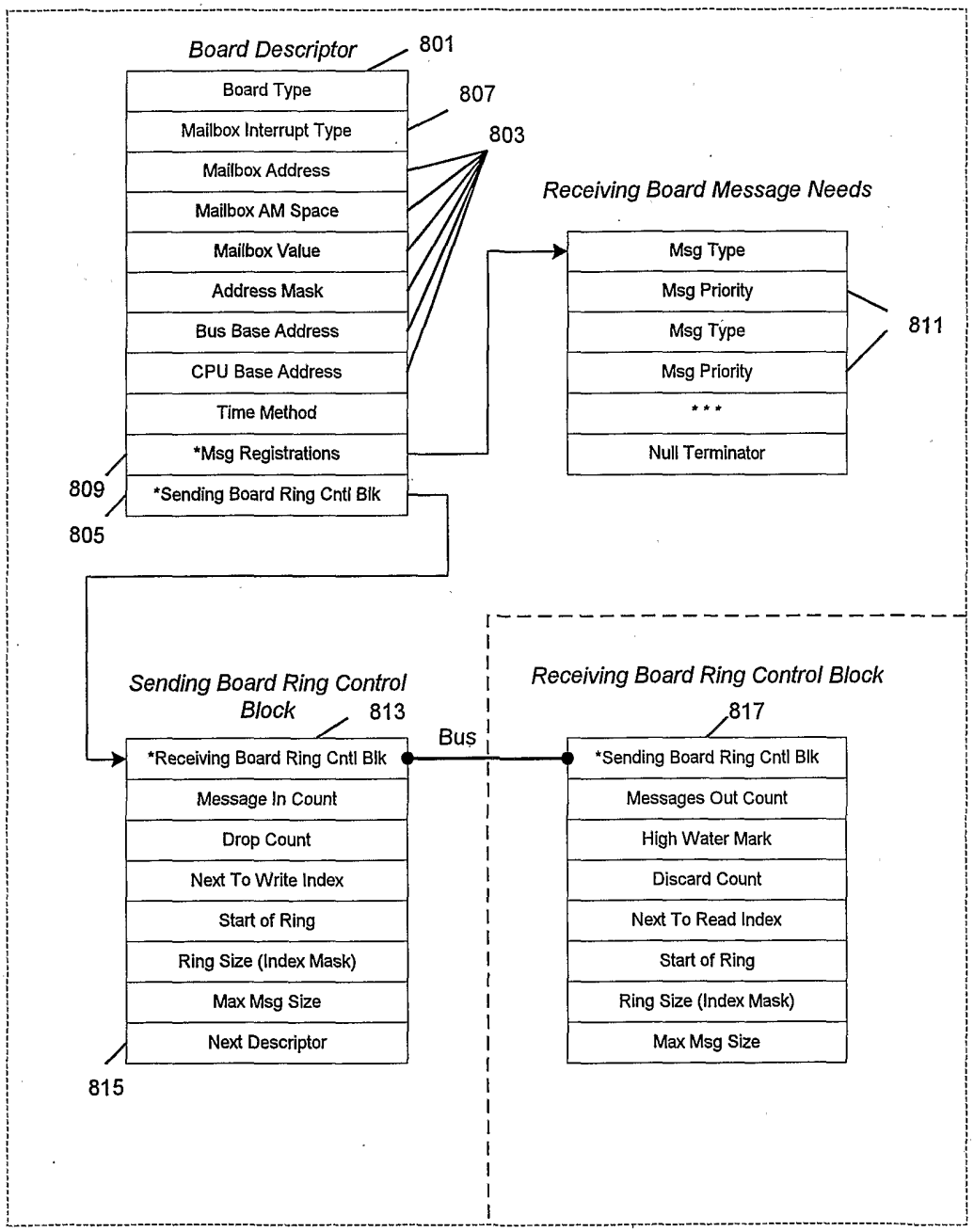

Board Descriptor
801
Board Type
807
Mailbox Interrupt Type
803
Mailbox Address
Mailbox AM Space
Mailbox Value
Address Mask
Bus Base Address
CPU Base Address
Time Method
*Msg Registrations
809
*Sending Board Ring Cntl Blk
805
Receiving Board Message Needs
Msg Type
Msg Priority
Msg Type
Msg Priority
* * *
Null Terminator
811
Sending Board Ring Control Block
813
*Receiving Board Ring Cntl Blk
Message In Count
Drop Count
Next To Write Index
Start of Ring
Ring Size (Index Mask)
Max Msg Size
Next Descriptor
815
Bus
Receiving Board Ring Control Block
817
*Sending Board Ring Cntl Blk
Messages Out Count
High Water Mark
Discard Count
Next To Read Index
Start of Ring
Ring Size (Index Mask)
Max Msg Size

FIG. 9

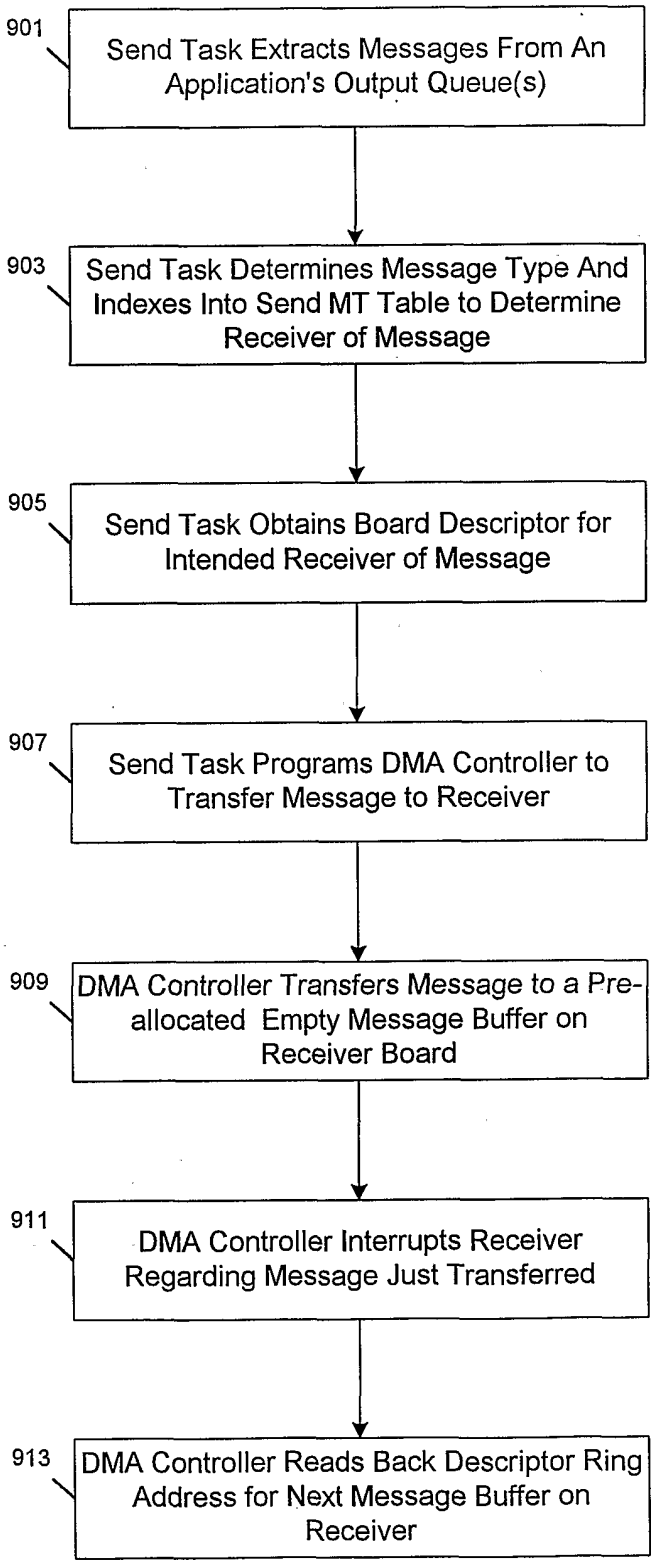
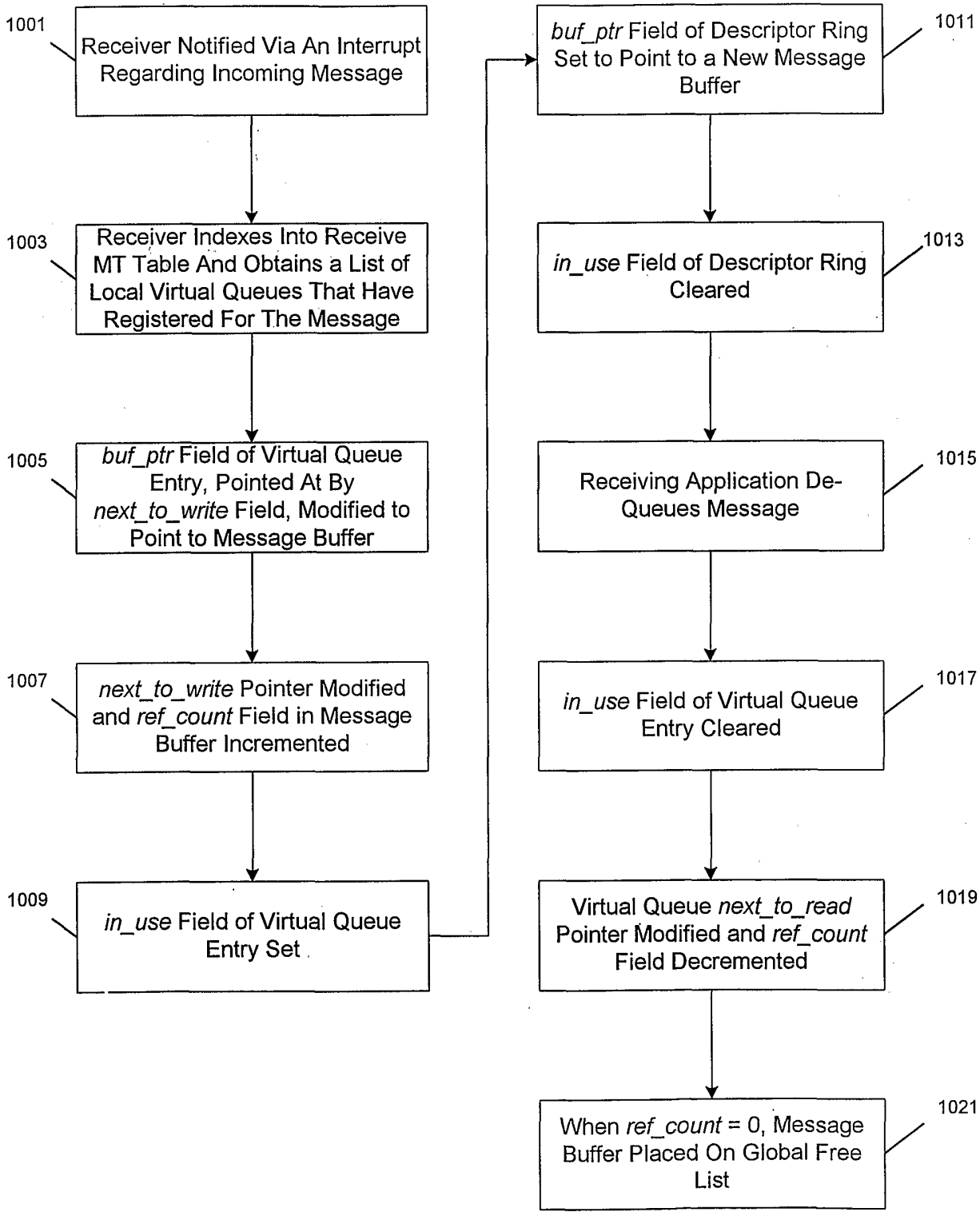

901
Send Task Extracts Messages From An Application's Output Queue(s)
903
Send Task Determines Message Type And Indexes Into Send MT Table to Determine Receiver of Message
905
Send Task Obtains Board Descriptor for Intended Receiver of Message
907
Send Task Programs DMA Controller to Transfer Message to Receiver
909
DMA Controller Transfers Message to a Pre-allocated Empty Message Buffer on Receiver Board
911
DMA Controller Interrupts Receiver Regarding Message Just Transferred
913
DMA Controller Reads Back Descriptor Ring Address for Next Message Buffer on Receiver

FIG. 10


1001
Receiver Notified Via An Interrupt Regarding Incoming Message
1003
Receiver Indexes Into Receive MT Table And Obtains a List of Local Virtual Queues That Have Registered For The Message
1005
buf_ptr Field of Virtual Queue Entry, Pointed At By next_to_write Field, Modified to Point to Message Buffer
1007
next_to_write Pointer Modified and ref_count Field in Message Buffer Incremented
1009
in_use Field of Virtual Queue Entry Set
1011
buf_ptr Field of Descriptor Ring Set to Point to a New Message Buffer
1013
in_use Field of Descriptor Ring Cleared
1015
Receiving Application De-Queues Message
1017
in_use Field of Virtual Queue Entry Cleared
1019
Virtual Queue next_to_read Pointer Modified and ref_count Field Decremented
1021
When ref_count = 0, Message Buffer Placed On Global Free List

INTERNATIONAL SEARCH REPORT

Inte nal Application No
PC T/US 00/35455

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F13/28

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

B. FIELDS SEARCHED

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

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

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

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 4 876 664 A (BITTORF BRADLEY J ET AL) 24 October 1989 (1989-10-24) column 1, line 22 - line 30; figures 14,15 column 2, line 15 - line 34 column 19, line 62 -column 20, line 23 column 21, line 65 -column 22, line 20 ---	1,12,23
Y	EP 0 646 876 A (NIPPON TELEGRAPH & TELEPHONE) 5 April 1995 (1995-04-05) page 3, line 15 - line 56; figures 1,2 page 5, line 20 - line 35 ---	1,12,23
A	EP 0 899 657 A (IBM) 3 March 1999 (1999-03-03) column 4, line 30 - line 57 column 5, line 14 - line 24; figure 4 column 5, line 49 -column 6, line 2 ---	1-39
	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 April 2001	07/05/2001

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

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	WO 91 11768 A (AUSPEX SYSTEMS INC) 8 August 1991 (1991-08-08) page 9, line 26 -page 10, line 15 page 11, line 10 - line 17 page 12, line 26 - line 33 page 13, line 8 - line 32	1-39
A	US 4 449 182 A (KECK DALE R ET AL) 15 May 1984 (1984-05-15) claim 1	1,12,23

INTERNATIONAL SEARCH REPORT

formation on patent family members

Inte nal Application No
PCT/US 00/35455

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4876664 A	24-10-1989	NONE	
EP 0646876 A	05-04-1995	JP 7105099 A	21-04-1995
		JP 8063442 A	08-03-1996
		DE 69424114 D	31-05-2000
		DE 69424114 T	09-11-2000
		US 5617537 A	01-04-1997
EP 0899657 A	03-03-1999	US 6026448 A	15-02-2000
WO 9111768 A	08-08-1991	AU 6431990 A	21-08-1991
		AU 7435294 A	15-12-1994
		CA 2074530 A	03-08-1991
		EP 0512993 A	19-11-1992
US 4449182 A	15-05-1984	CA 1183273 A	26-02-1985
		DE 3280286 D	07-02-1991
		DE 3280451 D	26-05-1994
		DE 3280451 T	17-11-1994
		EP 0077008 A	20-04-1983
		EP 0304540 A	01-03-1989
		JP 58501740 T	13-10-1983
		WO 8301326 A	14-04-1983