A data communication arrangement in which an interface processor (100-107) effects the transmission of messages between two processors (300, 500) of a multiprocessor system. The interface processor is connected to the communicating processors via direct memory access circuits (200-207, 400-407). A processor stores messages in a send buffer (252) in memory (3, 10) and controls a pointer (253) in memory indicating the loading of such a buffer. The interface processor reads this pointer and the messages, and writes a pointer (271) and the messages in a received buffer (270) of a receiving processor (500). The interface processor limits the loading of new messages into the send buffer by delaying the updating of an unload pointer (254), creating memory space for new messages, until the receiving processor has processed the transmitted messages. Messages can also be used to initiate the transfer of a block of data from the memory of one processor to that of another. Initialization of the interface processor is a joint effort of the communicating processors.
FOR THE PURPOSES OF INFORMATION ONLY

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
<th>Code</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>Austria</td>
<td>LI</td>
<td>Liechtenstein</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>MC</td>
<td>Monaco</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>DE</td>
<td>Germany, Federal Republic of</td>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>GA</td>
<td>Gabon</td>
<td>SU</td>
<td>Soviet Union</td>
</tr>
<tr>
<td>GB</td>
<td>United Kingdom</td>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>HU</td>
<td>Hungary</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>JP</td>
<td>Japan</td>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INTERFACE PROCESSOR UNIT

Technical Field

This invention relates to a communication arrangement in a multiprocessing system which comprises a plurality of processors each processor having a memory and each processor being connected to a memory access circuit.

Background of the Invention

During the past decade, multiprocessor systems have become increasingly popular as a flexible means of ultimately achieving high-capacity performance. The many complex problems which have restricted the application of multiprocessor systems in the past are gradually being solved, but several still remain.

Among the most serious problems are those associated with transmitting information among the processors which are required to cooperate in the completion of a data processing task. Most of the techniques currently in use adversely affect the capacity and response times of the systems. Two prior art solutions which have been offered to this problem are the use of a very high speed, high capability bus directly interconnecting the processors of a tightly coupled processor system, and the use of a general data communication facility for interconnecting a loosely coupled system. Both of these approaches have significant disadvantages. The first approach requires a substantial investment in hardware in order to have access to a powerful interprocessor bus and demands sophisticated circuitry in order to achieve the high speed performance required of such a bus. The second solution requires a substantial processing overhead in order to prepare messages for transmission among processors and causes considerable delay in the transmission of messages if the data link is loaded near its capacity. Both of these solutions are especially inefficient in cases in which a
very large volume of traffic in the form of interprocessor messages exists between particular pairs of processors. The problem is further aggravated if short messages predominate since the required data processing overhead per message decreases the efficiency of the communicating processors.

Another prior art solution to this kind of problem includes the use of special input/output processors to transmit messages. In this scheme, messages destined for input/output equipment, data links, or other processor systems are placed by a main processor in a memory file accessible by an input/output processor. The input/output processor can then take information out of this file and transmit such information to input/output equipment or to the input/output processor of another main processor system. The disadvantage of this kind of arrangement is that two intermediate processors (in this case input/output processors) as a minimum are required to handle communications between main processors. Delays are encountered in the transmission of the information because two intermediate storage operations are required, and because the cooperation of the two independent intermediate processors is required. These delays adversely affect the response time of the multiprocessor system.

The problems are solved in accordance with this invention in a communication arrangement in which the communication arrangement comprises an interface processor unit connected to the memory access circuit and comprising a central processing unit and a memory, each of the processors of the plurality of processors is adapted to generate and store data messages at first locations defined by an associated first set of addresses and adapted to receive and store messages at second locations defined by an associated second set of addresses, the interface processor unit is adapted to control one of the plurality of memory access circuits to read first data messages from the memory of a first one of the plurality of processors at...
locations defined by the associated first set of addresses of the first processor and to modify an address in the associated first set of addresses of the first processor to limit the locations in which the first processor can store new data messages, and further adapted to control another of the plurality of memory access circuits to write the first messages read from the first processor into the memory means of second one of the plurality of processors at locations defined by the associated second set of addresses of the second and to modify an address of the associated second set of addresses of the second processor to indicate that the first messages have been written into the memory of the second processor.

Summary of the Invention

It is an object of this invention to provide means for efficiently and reliably transmitting a high volume of relatively short messages plus occasional long messages between the processors of a multiprocessing system.

In accordance with this invention, an interface processor unit has access to the memories of two processors between which messages are to be transmitted. The two communicating processors are adapted to store information in their memories to request the transmission of messages, and to recognize the successful accomplishment of the transmission and processing of such messages. The interface processor unit reads message transmission control information from the sending processor memory indicating that new messages have been loaded into a send storage buffer means, and reads message reception control information from the memory of the receiving processor indicating that received messages have been processed from the receiving processor's receive storage buffer means. The interface processor unit modifies the transmission control information to indicate that messages have been received and processed, and modifies the reception control information to indicate that messages have been stored in
the receiving processor's receive storage buffer means. The transmission control information includes storage pointers indicating how far a send storage buffer means has been loaded by the sending processor, and how much of the data in the send storage buffer means has been received and processed by the receiving processor. Reception control information includes storage pointers indicating how far a receive storage buffer means has been loaded with messages, and how far these messages have been processed by the receiving processor.

Advantageously, the interface processor unit can examine the data in messages transmitted from one processor to another. For example, in order to handle the transmission of a large block of information from an arbitrary address outside a send buffer in a sending processor to an arbitrary address outside the receive buffer in a receiving processor, the interface processor unit examines all messages in order to determine whether a message contains a block transfer request; if so, the interface processor unit determines the addresses of the sending and receiving information blocks in order to carry out the block message transmission.

In order to initialize or otherwise control the interface processor unit, or to set up parameters in this unit, or to audit its memory, it is necessary for a processor to be able to communicate with the interface processor unit. In one embodiment, an interrupt signal is transmitted from a processor through the memory access means to the interface processor unit. Since each of the processors connected to a interface processor unit may operate with independent data bases, it is advantageous to use this mechanism to permit each such processor to control the initialization of that section of the interface processor unit memory which contains the data peculiar to that processor.

The use of an interface processor unit in conjunction with transmission and reception control
information as outlined above leads to a system requiring a
minimum of per message data processing overhead by both
receiving and transmitting processors. Hence, it is
especially useful for multiprocessing systems in which
processors exchange a large volume of short message
traffic; in these systems, large per message overhead would
be intolerable.

Brief Description of the Drawing

The invention may be better understood from the
following detailed description when read with reference to
the drawing in which:

FIG. 1 represents a block diagram of an
illustrative embodiment of the invention showing an
arrangement in which one processor communicates with eight
processors;

FIG. 2 shows the information flow and program
control flow for communications between two processors;
FIG. 3 shows the layout of a typical data message
used in such communications; and

FIG. 4 represents a block diagram of an alternate
multiprocessing arrangement.

Detailed Description

FIG. 1 is a basic block diagram of one embodiment
of the present invention. It shows an arrangement to allow
one main processor, processor 300, to communicate with
eight auxiliary processors, processors 500-507. The
communication with each of these processors is handled by
one of the eight interface processor units (100-107). Each
of the interface processor units 100-107, uses one of the
direct memory access units 200-207 to communicate with
processor 300, and uses one of the direct memory access
units 400-407 to communicate with the associated auxiliary
processor. For the sake of convenience, in this
description a phrase such as: "peripheral interface
unit 100 reads from, writes into, changes, or updates
processor 300 or 500" should be interpreted as a brief way
of saying that peripheral interface unit reads from, writes
into, changes, or updates processor 300 or 500 using direct memory access 200 or 400, respectively.

Each of the auxiliary processors, 500-507, consists of memory, a central processing unit, and some form of input/output equipment. Eight auxiliary processors and interface processor units are depicted to illustrate the concepts of this invention. The auxiliary processors 500 through 507 may be any of a number of well-known processors, using, for example, a commercially available microprocessor as a central processing unit. The memory of the processor and the central processing unit can be accessed via a standard direct memory access unit. The direct memory access units 200-207 and 400-407 are well-known and commercially available.

In this embodiment, eight direct memory access units are used to communicate with eight different processor interface units. In alternate embodiments, only one interface processor unit might be required using only one direct memory access unit to communicate with processor 300, or one multiport direct memory access unit could be used to communicate between processor 300 and eight processor interface units.

In one specific embodiment, a main processor 300 is used to control a telephone central office, not shown in the drawing. The processor comprises a memory 310, a central processing unit 320, and an input/output unit 330. The input/output unit 330 represents the switching network customarily used to interconnect telephone customers and the equipment required to receive information from customers and from other central offices. The latter information is used to control the switching network which allows customers to be interconnected.

The auxiliary processors are used for various auxiliary data processing or input/output control functions. Messages between processor 300 and one of the auxiliary processors 500-507 are placed in separate sets of send and receive buffers in memory 310 of processor 300.
Each interface processor unit 100-107 thus communicates with a different pair of buffers in memory 310. In this description, only communications between processors 300 and 500 are described in detail. Communications between processor 300 and one of the processors 501-507 are handled in the same manner, except that they use different buffer pairs in memory 310, different interface processor units, and different direct memory access units.

Communications between processors 300 and 500 are in the form of messages and data transferred via interface processor unit 100. Interface processor unit 100 includes a central processing unit 112, such as a microprocessor, and a memory 111. In this example, it is assumed that processor 500 is used for communicating with a centralized telephone traffic analysis system, (not shown). Processor 500 is connected to the traffic processing center via a data link connection which is part of the input/output system 530. From this data link, processor 500 receives a message that detailed traffic data are to be accumulated in the system. Such an incoming message acts as a request to processor 500 to accumulate traffic data, process this data, and send appropriately summarized information to the traffic center. In order to accumulate the traffic data, processor 500 must send a message to processor 300 requesting two kinds of actions: processor 300 must accumulate some traffic data internally (for example, data on usage of the switching network) and must send messages to processor 500 to allow that processor to accumulate and process other traffic data. In this example, it is necessary for main processor 300 to notify processor 500 of the duration, destination, and selected trunk used on each interoffice call.

The central processing unit 112 in interface processor unit 100 controls all of the transmission of messages from processor 300 to processor 500 and vice versa. Interface processor unit 100 has access to the memory of processor 300 via direct memory access 200 and
uses this access to read messages from and to write messages into processor 300. The messages from processor 300 memory 310 are read into interface processor unit 100 memory 111, and the messages destined for processor 300 are written from interface processor unit 100 memory 111 into processor 300 memory 310. Similarly, messages destined for processor 500 are written from the interface processor unit 100 memory 111 via the direct memory access 400 into processor 500 memory 510, and messages from processor 500 to processor 300 are read from processor 500 memory 510 by the direct memory access 400 into interface processor unit 100 memory 111.

Because of the need for continuous telephone service, standby equipment (not shown) is commonly provided for critical portions of the system such as processors, direct memory access units, and peripheral interface units.

FIG. 2 is a graphical representation of programs and data blocks used in transmitting, receiving, and processing messages. For the sake of clarity, arrows show the predominant direction of information flow.

Processor 300 and processor 500 are each under the control of a control program referred to as an operating system. Each operating system carries out its work by initiating other programs called processes. Each process is executed by calling on appropriate program functions to carry out specific tasks. The flexibility of a multiprocessor system is substantially enhanced by allowing interaction among processes which are executed in different processors. These processes then communicate with each other via data messages.

In this example, a process 250, which is executed in main processor 300, must send a message to process 274, executed in processor 500. The message is prepared by the sending process and placed in a portion of memory called a send buffer, by an interface program, the interface processor unit message handler 251 (PIUMH). One of the functions of the PIUMH 251 of processor 300 is to translate
from the number of the destination process, included as part of the message, to the identity of the destination processor. Since process 274 executes on processor 500, messages to process 274 must be stored in the send buffer 252, which contains messages destined for processor 500.

To signal that a new message has been stored in the send buffer 252, a load pointer 253 is updated. The processor interface unit detects this condition, reads the message, and writes it into the receive buffer 270 of processor 500. It signals processor 500 that a message has been placed in the receive buffer 270 by updating the load pointer 271. Processor 500 also has a PIUMH, 273, which unloads the message from the receive buffer 270 and passes it on to the receive process 274. The unload pointer 272 is changed by PIUMH 273 when it recognizes that a message has been satisfactorily received, and has passed this message on to receive process 274.

Unload pointer 254 is used to limit the range of addresses in which PIUMH 251 can write new messages. Old messages cannot be discarded until assurance has been received that no retransmission of the messages will be necessary. Therefore, the unload pointer 254 of send buffer 252 is updated after receiving processor 500 has processed the message and has updated unload pointer 272.

In this embodiment, send and receive buffers of identical length in corresponding locations of memory of processors 300 and 500 are used. This allows all pointers to be the same when there are no messages to be transferred; allows differences among the load pointers to be used as an indication to the interface processor unit that a message is ready for transmission; and allows the receive buffer unload pointer to be copied into the send buffer unload pointer after a message has been processed.

Interface processor unit 100 recognizes the need for transmitting a message by reading the load pointer 253 of send buffer 252 and comparing it with the load
pointer 271 of receive buffer 270. If the two pointers do not match, a new message has been loaded into send buffer 252, and should be transmitted to receive buffer 270. Also, if the load pointer 271 and unload pointer 272 of the receive buffer 270 do not match, PIUMH 273 recognizes that a message has been received and should be sent to a receiving process for further processing. After the message has been processed, PIUMH updates the unload pointer 272, which the interface processor unit copies into unload pointer 254 of the send buffer 252 of processor 300, in order to create space in the send buffer for new messages. This mechanism automatically limits the rate of which processor 300 can generate new messages to the rate of which processor 500 can accept and process these messages.

Both the send buffer 252 and receive buffer 270 are first-in/first-out buffers arranged in a circular fashion; the first location in each of these buffers immediately succeeds the last location. This allows an indefinite number of messages to be written into these buffers as long as the buffers are cleared of old messages before new messages must be stored. New messages may be added to a send buffer as long as the load pointer continues to lag the unload pointer. The PIUMH may not load a message into a send buffer if the result would be to cause the load pointer to overtake the unload pointer.

A conventional first-in/first-out buffer is usually loaded by a first mechanism and unloaded by a second mechanism. These mechanisms are the execution of different programs on the same or different processors. The purpose of a load pointer is to indicate from the first to the second mechanism how far the buffer has been loaded. The purpose of the unload pointer is to indicate from the second to the first mechanism how far the first mechanism may write in the buffer. Sometimes, these pointers serve additional functions such as indicating where the second mechanism should start its unloading process; since this
information is controlled only by the second mechanism, it can be retained separately and need not be part of the shared buffer control information. In this embodiment, in which the send and receive buffers are synchronized, it is convenient to use the load pointer of a receive buffer as the source of the information on where to start the unloading process in a send buffer, but this information, which is not controlled or needed by the sending processor, could be retained alternatively in the processor interface unit memory. An advantage of the present embodiment is that the system can continue operation even if trouble occurs in the interface processor unit and a standby unit, which would not have up-to-date pointer information, is switched into service.

FIG. 3 shows the format of the message transmitted between processor 300 and processor 500. Every message has a standard format initial header 351, and an arbitrary length body 360. The header has a number of fields which are fixed in location in every message, and are interpreted by the receiving PIUMH and the interface processor unit. The fields are used to accomplish a number of purposes. The "from" field 352 is the identification of the sending process number. The "to" field 353 is the identification of the receiving process number. The "message size" field 354 is an indication of the total length of the message including both header and body. The "type of message" field 355 is an indication of whether this is, for example, a simple message (MSG), or one of the messages which are used to prepare for the transmission of a block of data. The "status" field 356 is used to indicate the nature of a failure. The "identity" field 359 identifies one of a series of messages from different sources, in order to allow any responses to be associated with the proper source. The "block size" field 362 indicates the length of a block to be transferred when a block transfer is requested.
Transmission of a simple data message requires only a simple exchange of the message and acknowledgment. When such a message is sent from the source processor 300 to the destination processor 500, the message type field 355 contains a number representing "Message" (MSG). After the receiving process 274 has received and processed the message passed on by the PIUMH 273, it may, optionally, generate an acknowledge message back to the send process 250. The acknowledgment message, with the message type field 355 containing a number representing "Message Acknowledgment" (MSG-ACK), would be placed in send buffer 290 by PIUMH 273; it is transmitted in essentially the same way as a simple data message.

The interface processor unit can also be used to transfer a block of data from an arbitrary location in memory of one processor to an arbitrary location in memory of another processor. If the data is transferred to the processor making the request, this is called a read block transfer; if it is transferred from the processor making the request, this is called a write block transfer. Such a transfer of information can be processed more efficiently as a block transfer than as a series of data messages, each containing a portion of the block of data to be transferred.

As mentioned previously, in response to the request from the traffic center, processor 300 sends a message to processor 500 for every completed interoffice call. Processor 300 also directly accumulates different traffic data, such as data on usage of the switching network, and stores this data in its own memory 310. When the time comes to process this data further in order to send the results to the traffic center, the data must be transmitted from processor 300 to processor 500. The best mechanism for accomplishing this data transfer is the write block transfer mechanism.

The interface processor unit plays a much broader role in the case of block transfer than it does for the
transfer of simple messages. For the block transfer, also illustrated in FIG. 2, the interface processor unit must recognize that it is to read information, not only from the send buffers of the sending processor, but from an address indirectly specified by one of the messages. The interface processor unit examines the message type field 355 of the header 351 of every message to see if the message involves the preparation for a block transfer. When this type of message is recognized, the interface processor unit must find the address of the sending and receiving processor message blocks in order to carry out the block transfer.

Each processor involved in the block transfer will assign a special job buffer to help control this transfer. Job buffer 630 in processor 300 and job buffer 730 in processor 500 contain the address entries 631 and 731, and size entries 632 and 732, of the transmitted or received block of information, the status entries 633 and 733 of this block transfer, and the number of the job buffer in the other processor 634 and 734. The interface processor unit 100 reads the sending (357) and receiving (358) job buffer number fields which define the addresses of these job buffers, and then reads the job buffers in order to find the initial address, the size of the block to be transmitted, and the destination address of the block.

The process begins when the sending processor 300 sends an initial message to the receiving processor 500. The message is transmitted in the conventional way, as described above, and is placed in receive buffer 270. The message contains the sending job buffer number 630 in the send job buffer field 357, and indicates the nature of the block transfer request in the type field 355. In this case, a "write" (WT) message, i.e., a message to initiate a write block transfer, is sent in the type field 355. At some later time, the processor 500 PIUMH 273 queries the load 271 and unload 272 pointers to see if any messages have been received. When PIUMH 273 recognizes that a message has been received, it unloads the message. It
recognizes a request for a block transfer in the type field 355 and assigns a job buffer 730 to the block transfer. PIUMH 273 then sends the message to the receiving process 275.

Process 275 also recognizes that a block transfer is to be made and that memory for receiving this information must be assigned. Process 275 assigns destination block 760. PIUMH places the address of block 760 in job buffer 730 in location 731, the size of the block to be transferred, obtained from the block size field 362, in location 732, and the present status of the transfer in location 733. At this point, the status is simply that everything is normal and that the message has not been transmitted. The job buffers also contain a field (634 and 734) indicating the number of the job buffer controlling the other end of the block transfer; this field is set up in the job buffer of the responding processor by the PIUMH, in this case PIUMH 273 in processor 500. The field is used as an integrity check to help insure that errors are restricted to either processor and not propagated.

Process 275 also generates a write activate message (WTA) which is placed in send buffer 290 by PIUMH 273. Interface processor unit 100 reads this message and recognizes the WTA message type in type field 355. Message type WTA is one which the interface processor unit must recognize as indicating that the interface processor unit must prepare to execute a block transfer. At this point, the interface processor unit has all the information it needs. The header of the WTA message contains an indication of the send job buffer, (transmitted as part of the previous message, and returned in the present message) and receive job buffer whose number was inserted by PIUMH 273, from the send 357 and receive buffer 358 number fields. The interface processor unit will thus be able to find the addresses of the origin block 660 and the destination block 760. It will also be able to identify
the size of the block to be transferred from either 632 or 732 in job buffers 630 and 730 of the sending and receiving processors.

The interface processor unit then begins the process of transferring the block of data from origin block 660 to destination block 760. The two size fields may be different if it is desirable to reserve a larger block than necessary on one side. This will occur, for example, for a transfer of a block of data of unknown but limited length. A block transfer will transmit only necessary data from one processor to the other; the smaller length field is controlling. The interface processor unit accomplishes the block transfer by reading data from origin block 660 and writing the data into destination block 760.

When the block transfer is completed, the interface processor unit sends a WTA message to processor 300. When PIUMH 251 sees the WTA message, it informs the send process 258 by sending a write acknowledgment (WT-ACK) message that processor 500 has received the block of information. PIUMH 251 then sends a Write Activate Acknowledgement (WTA-ACK) message back to processor 500 to inform receive process 275 that the block transfer has been completed.

When a interface processor unit is initially placed in service, or when a interface processor unit has been out of service for a time, it is necessary to initialize the unit. Initialization of a interface processor unit is carried out in the following manner. Processor 300, which includes circuitry for generating and accepting interrupt signals, sends a command via direct memory access 200 to interface processor unit 100 to suspend direct memory access operations. This command, which must be channeled to central processor unit 112, is treated as an interrupt. Central processing units including microprocessors include circuitry for accepting and generating interrupt signals. Next, another command, also treated by central processing unit 112 as an
interrupt, is sent requesting that an interrupt command be sent by the interface processor unit 100 via direct memory access 400 to processor 500 to request initialization. Processor 500, using direct memory access 400, initializes a portion of the memory 111 dedicated to the special parameters which are associated with processor 500 of interface processor unit 100. When it has finished, processor 500 sends an interrupt command via direct memory access 400 to interface processor unit 100 to request that it send an interrupt command via direct memory access 200 to processor 300 to request additional initialization. Processor 300 via direct memory access 200 writes into interface processor unit 100 memory 111 the additional initialization data. Finally, processor 300 sends another interrupt to interface processor unit 100 indicating the completion of initialization; the interface processor unit 100 is now ready to begin normal operation. This method of initialization permits each of the two processors 300 and 500 to initialize a portion of the memory 111 of the interface processor unit 100. This can be useful if processors 300 and 500 have relatively independent data bases.

In a system which uses duplicated equipment, such equipment may be considered to be in one of three states: on-line, actively performing system tasks; off-line, out of service; or standby, ready to go into the on-line state upon demand. An initialization scheme such as that described above would be used whenever a interface processor unit is placed on-line in an unduplicated system, or, in a duplicated system, when a interface processor unit is switched from the off-line state directly into the on-line state because of a failure occurring while one unit is off-line. When a interface processor unit is switched from standby to on-line, a much simpler procedure than that described can be used since initialization of parameters is not required. Moreover, when a routine maintenance switch of a interface processor unit from standby to on-line is
made, this switch can be initiated by the normal message mechanism. The normal message mechanism can also be used to control a switch from the off-line to the standby state.

This specific embodiment has been designed to match the characteristics of the specific processors used in one system, and the requirements for the kind of message traffic which is characteristic of that system. For other systems in which there is a substantial volume of information flow among many processors, a processor unit may be connected to all of these processors. The interface processor unit then acts as a message switch. A structure for such a system is illustrated in FIG. 4. The interface processor unit 900 is connected to a group of direct memory access circuits (911-916) each of which is connected to one of the interconnected processors (901-906). In this type of configuration, the method of operation is essentially as described with respect to FIG. 1 through 3. Each processor 901 through 906 would have a send buffer and a receive buffer for communicating with each other processor. The sending processors would use the same basic techniques for loading messages to one of the send buffers, and modifying a load pointer in that buffer; the receiving processors would use the same technique of examining load and unload pointers to see if any messages have been received, and modifying the unload pointer when a received message has been processed. The interface processor unit would control the load pointers of receive buffers and the unload pointer of send buffers. Block transfers can also be handled in the same way as described above.

It is to be understood that the above-described embodiment is merely illustrative of the principles of this invention; other arrangements may be devised by those skilled in the arts without departing from the spirit and scope of the invention.
1. A communication arrangement in a multiprocessing system which comprises:

   a plurality of processors (901-906, 300, 500) each
   processor having a memory (310, 510) and each processor
   being connected to a memory access circuit (911-916, 200-
   207, 400-407);

   CHARACTERIZED IN THAT
   the communication arrangement comprises:
   an interface processor unit (900, 100-107)
   connected to the memory access circuit and comprising a
   central processing unit (112) and a memory (111);
   each of the processors of the plurality of
   processors is adapted to generate and store data messages
   at first locations (252; 290) defined by an associated
   first set of addresses (253, 254; 291, 292) and adapted to
   receive and store messages at second locations (280; 270)
   defined by an associated second set of addresses (281,
   282; 271, 272);

   the interface processor unit (900, 100-107) is
   adapted to control one of the plurality of memory access
   circuits to read first data messages from the memory of a
   first one (300) of the plurality of processors at locations
   (290) defined by the associated first set of addresses
   of the first processor (253, 254) and to modify an address
   in the associated first set of addresses (254) of the first
   processor to limit the locations in which the first
   processor can store new data messages, and further adapted
   to control another of the plurality of memory access
   circuits to write the first messages read from the first
   processor into the memory means of second one (500) of
   the plurality of processors at locations (270) defined by
   the associated second set of addresses of the second (271,
   272) and to modify an address (271) of the associated
   second set of addresses of the second processor to indicate
   that the first messages have been written into the memory
   of the second processor.
2. A communication arrangement in accordance with claim 1

CHARACTERIZED IN THAT
the first processor (300) is adapted to receive
and store second data messages at locations (280) in the
memory of the first processor defined by the second set of
addresses (281, 282) in the first processor, and the second
processor (500) is adapted to generate and store the second
data messages at locations (290) in the memory of the
second processor defined by the first set (291, 292) of
addresses, in the second processor; and
the interface processor unit (100) is further
adapted to control the second memory access circuit to read
the second data messages from the memory of the second
processor at locations defined by addresses of the first
set of the second processor and to modify an address in the
first set of the second processor to limit the locations in
which the second processor (500) can store new second data
messages, and further adapted to control the first memory
access to write the second data messages read from the
memory of the second processor into the memory of the first
processor at locations defined by addresses of the second
set of the first processor and to modify an address (282)
of the second set.

3. A communication arrangement in accordance with
claim 2

CHARACTERIZED IN THAT
the first and second messages comprise block
transfer messages requesting the transfer of a block of
data from one of the memories of the first and second
processors to the other of the memories of the first and
the second processors, and indicating block location
addresses for the block transfer; and
the interface processor unit (100) is adapted to
control the first and the second memory access to read the
block of data from the one of the memories of the first and
second processors and to write the block into the other of
the memories of the first and second processor.

4. A method for transferring a data message from a
first memory to a second memory in a multiprocessing system
which comprises a first processor having the first memory
and a second processor having the second memory, a first
and second memory access circuit connected to the first and
second memory respectively, and an interface processor unit
adapted to control the first and second memory access
circuit comprising the steps of:

storing a first data message at locations in the
first memory defined by a first set of addresses;

reading the first data message from the first
memory in accordance with addresses from the first set of
addresses by the interface processor unit;

writing from the interface processor unit the
first data message in the second memory in accordance with
addresses defined by the second set of addresses;

modifying the second set of addresses by the
interface processor unit after the first message has been
stored in the second memory;

verifying the integrity of the first message by
the second processor;

modifying the second set of addresses by the
second processor after the first message has been verified;
and

reading the second set of addresses by the
interface processor unit and modifying the first set of
addresses by the interface processor unit after the second
set of addresses has been so modified.

5. The method of claim 4 in which the interface
processor unit is further adapted to recognize block
transfer messages requesting the transfer of a block of
data between the processors, further comprising the steps of:

- generating block transfer messages requesting a block transfer by one of the processors indicating the size of the block to be transferred, the block location address in the one processor, and the direction of transfer;
- reading the block transfer messages by the interface processor unit and writing the message to the other of the processors;
- processing the block transfer message by the other processor and generating a return message indicating a block location address in the other processor;
- reading the return message by the interface processor;
- transferring the block of data by means of the interface processor unit in accordance with the direction of transfer.
# INTERNATIONAL SEARCH REPORT

**International Application No.**
PCT/US82/01675

## I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) 3

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

<table>
<thead>
<tr>
<th>Int. CL.</th>
<th>Classification Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>G06F 304</td>
<td></td>
</tr>
<tr>
<td>G06F 15/6</td>
<td></td>
</tr>
</tbody>
</table>

**U.S. CL.**
364/200, 900

## II. FIELDS SEARCHED

<table>
<thead>
<tr>
<th>Classification System</th>
<th>Classification Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.</td>
<td>364/200, 900</td>
</tr>
</tbody>
</table>

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

## III. DOCUMENTS CONSIDERED TO BE RELEVANT 14

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of Document, 14 with indication, where appropriate, of the relevant passages 17</th>
<th>Relevant to Claim No. 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US,A, 4,298,928 (Etch et al) 03 November 1981</td>
<td>1-5</td>
</tr>
<tr>
<td>A</td>
<td>US,A, 4,244,018 (Mui) 06 January 1981</td>
<td>1-5</td>
</tr>
<tr>
<td>A</td>
<td>US,A, 4,228,496 (Katzman et al) 14 October 1980</td>
<td>1-5</td>
</tr>
<tr>
<td>A</td>
<td>US,A, 4,181,936 (Kober) 01 January 1980</td>
<td>1-5</td>
</tr>
<tr>
<td>A</td>
<td>US,A, 3,848,233 (Lotan et al)</td>
<td>1-5</td>
</tr>
</tbody>
</table>

* Special categories of cited documents: 15
  - "A" document defining the general state of the art which is not considered to be of particular relevance
  - "E" earlier document but published on or after the international filing date
  - "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
  - "O" document referring to an oral disclosure, use, exhibition or other means
  - "P" document published prior to the international filing date but later than the priority date claimed

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

**X** document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

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

**A** document member of the same patent family

## IV. CERTIFICATION

<table>
<thead>
<tr>
<th>Date of the Actual Completion of the International Search</th>
<th>Date of Mailing of this International Search Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 January 1983</td>
<td>09 FEB 1983</td>
</tr>
</tbody>
</table>

International Searching Authority
ISA/US

Signature of Authorized Officer
[Signature]

---

13 January 1983
ISA/US

[Signature]