PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 12/00, 13/00	A1	(11) International Publication Number: WO 96/20445
		(43) International Publication Date: 4 July 1996 (04.07.96)

(21) International Application Number: PCT/US95/16016

(22) International Filing Date: 20 December 1995 (20.12.95)

(30) Priority Data:
08/372,380 23 December 1994 (23.12.94) US

(71) Applicant: NEW MEDIA CORP. [US/US]; One Technology, Building A, Irvine, CA 92718 (US).

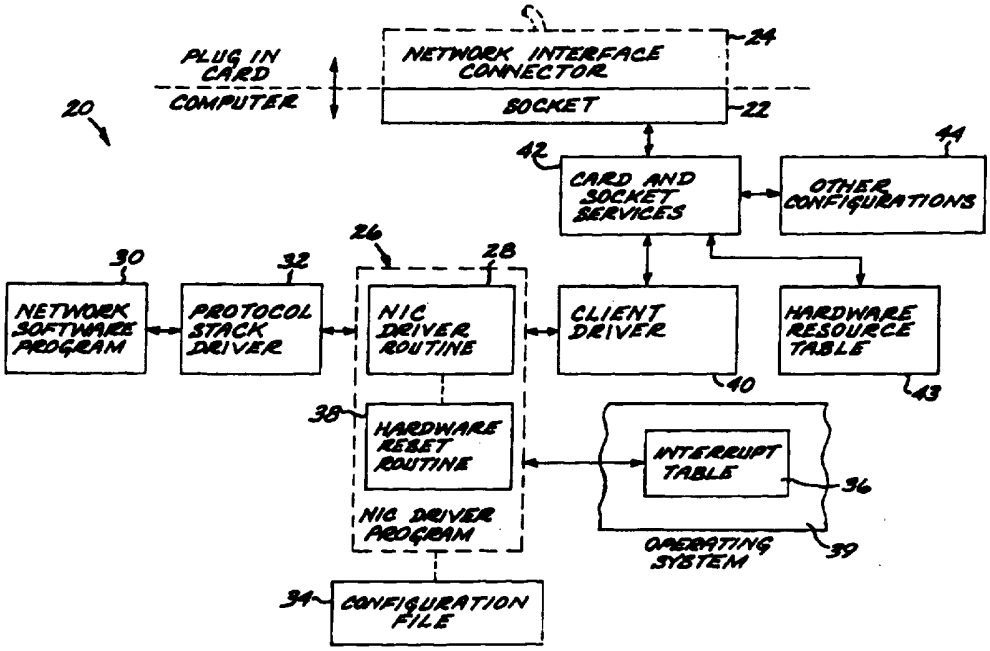
(72) Inventor: POLLARD, Thomas, G.; 13930 Carmel Ridge Road, San Diego, CA 92128 (US).

(74) Agents: GARMONG, Gregory, O.; P.O. Box 12460, Zephyr Cove, NV 89448 (US) et al.

(81) Designated States: AU, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHOD FOR START-UP AND BINDING OF A COMPUTER TO A NETWORK



(57) Abstract

A portable computer (20) is configured to receive a network interface connector (24) (such as the connector to a local area network) but initially has no network interface connector (24) connected thereto. A network interface connector reset and configuration protocol are established, preferably at computer start-up. At a later time, the network interface connector (24) is connected to the portable computer (20). The network interface connector (24) is automatically reset and configured using the protocol previously established, without having to restart the computer (20) or otherwise interrupt its ongoing operations.

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.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

METHOD FOR STARTUP AND BINDING OF A COMPUTER TO A NETWORK

BACKGROUND OF THE INVENTION

This invention relates to computer systems, and, more particularly, to the interfacing of a portable computer to a network.

Portable computers are now available to perform a wide range of computing activities. The smaller the computer in size, the less its internal capability in most cases. Hand-held (also called palmtop) computers therefore rely to a significant degree on external capabilities such as plug-in memories, plug-in computational aids, and the interconnection to a network. The ability to connect to a data network is particularly important for such small computers, because it provides them access not only to other small computers at other locations, but also to data and computational capabilities available at a large central computer.

As an example, a hand-held computer may be used to perform remote-site engineering calculations using a first plug-in computational card. The first card is thereafter removed, and a second plug-in computational card installed in the same slot to aid in cost estimating or other functions. The user next removes the second computational card and inserts a plug-in network interface connector that provides access to a large network. This alternating of plug-in cards and connectors is required because there is usually one interface socket slot (or, at most, two slots) on the computer due to size and weight limitations. Once established, the network interface is used to communicate intermediate or final results to other hand-held computers or to a central facility, to receive additional information, to utilize a central computer facility to perform other calculations which are too complex for the hand-held computer even with plug-in aids, to interface with input/output devices, or other functions.

When the hand-held computer is to be connected to the network, the

network interface connector, typically in the form of a network interface card, must be reset to a known initial state and communications protocols must be established between the computer and the network. At the present time, the network interface card must be inserted into the socket (slot) on the hand-held computer in order to accomplish these initializing functions, which are sometimes termed network startup and binding. If the network interface card is not plugged into the socket, there is an error message when network startup and binding is attempted.

In some cases, the requirement of the physical presence and connection of the network interface connector poses no difficulty. In other instances, the inventor has recognized that it creates a significant problem. If the computer user brings the hand-held computer to the location of the connector in a computational (e.g., a large spread sheet occupies the memory) or interfacing (e.g., the necessary interrupt states for the network are not available due to prior use of the computer) configuration such that the requisite initializing cannot be performed, the user is precluded from connecting to the network until some solution can be found. This obstacle is particularly difficult to overcome where the user is not knowledgeable about the technical features of computer/networking interface, and therefore does not understand the requirements of the interface, as is often the case.

There thus exists a need to provide users more flexibility in their use of personal computers to connect to networks. This need is most acute for small personal computers such as hand-held computers, which have limited memory and interface connector sockets. The present invention fulfills this need, and further provides related advantages.

SUMMARY OF THE INVENTION

The present invention provides a method for accomplishing network startup and binding of a personal computer without connection to the network at the time the startup and binding are performed. The approach is particularly valuable when used in conjunction with hand-held computers which require the swapping of various types of plug-in capabilities in their available (usually one

or at most two) sockets during the normal course of operations. The method of the invention allows the user to freely work back and forth between stand-alone plug-in cards, such as memory or specialized computational cards, and network interface connectors to the network.

In accordance with the invention, there is provided a method for accomplishing network startup and binding of a portable computer having a connector socket which can receive a network interface connector. The method includes the steps of providing network interface connector driver means for communicating data with the connector socket, and providing client driver means for establishing a configuration for the connector socket, which configuration is to be implemented at such time as a network interface connector is introduced into the connector socket. The client driver means determines that no network interface connector is connected to the connector socket and provides that information to the network interface connector driver means. The network interface connector driver means and the client driver means cooperate to establish a hardware resource information set within the portable computer sufficient to operate the network interface connector at such time as the network interface connector is introduced into the connector socket.

The client driver means preferably includes a capability to sense the presence of a network interface connector (e.g., a card) plugged into the connector socket (which is normally in the form of a slot) provided in the computer. Where a connector has been plugged in and communication with an external network is possible through the connector, complete startup, binding, and operation can occur in the normal manner. However, where the client driver means senses that no network interface connector is present and plugged in, the client driver means simulates the presence of the connector to the extent of permitting the necessary reset and configuration protocol to be established which will support the network interface connector when it is later plugged in. Consequently, it is not necessary to repeat these procedures at a later time when the network connection is made, avoiding disruption of activities then underway or the possibility that the required structure will not be available.

The details of the network startup and binding procedure of the invention can vary somewhat according to specific computers, networks, and

network software. Typically, however, the network interface connector driver program provides to the client driver program the location of the reset routine that is used to initialize the network interface connector when it is plugged in, and requested network hardware resources such as the interrupt request level structure and the input/output address of the network interface connector. The network interface connector driver program stores the locations of the interrupt routines in the software interrupt table, so that later interrupts generated through the network interface connector are properly handled by the network interface connector driver routines. The hardware resources are reserved so that they will be available to the network when the connection is made. The node identification of the network interface connector is also placed into a configuration table for later access. Finally, in this setup portion of the methodology, the network program and the client driver are notified that the setup has been completed.

At a later time, after the network interface connector has been plugged into the socket to establish a communication path between the computer and the network, the client driver program configures the connector socket according to the preestablished configuration. The client driver program reads from the network interface connector any hardware-specific information required by the computer for communication and provides this information to the network interface connector driver program. Lastly, the client driver program calls the hardware reset routine using the address provided during setup, and this routine is executed to place the network interface connector into a known state suitable for subsequent communication.

When this final portion of the procedure is executed, all of the required information for performing the hardware reset and establishing the configuration is available to the client driver. It is not necessary to interfere with any other operations of the computer, nor is there any possibility that required information, memory locations, or the like will not be available because all such hardware-support information was previously defined, established, and reserved.

The present invention provides an advance in the art of computers, and particular the small hand-held computers that rely heavily on external support

but have limited numbers of external connector ports. Other features and advantages of the present invention will be apparent from the following more detailed description of the preferred embodiment, taken in conjunction with the accompanying drawings, which illustrate, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic block diagram of the interrelation of hardware and software components of the invention;

Figure 2 is a flow diagram of the approach of the invention;

Figure 3 is a program flow chart for the pre-card-insertion phase of startup and binding; and

Figure 4 is a program flow chart for the post-card-insertion phase of startup and binding.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 schematically depicts the interrelation of the information flows for the pertinent portions of the hardware and software elements of a personal computer 20 that utilizes the present invention. The computer 20 is a portable computer and is preferably a hand-held computer. The computer 20 includes a socket 22, sometimes known as a slot or port, that can receive plug-in connectors to various types of external functions. In the case of interest, a network interface connector 24 can be plugged into the socket 22 when the user of the computer 20 seeks to connect to and utilize an external network. The network interface connector 24 is typically a local area network (LAN) connector in the form of a card.

Within the central processor and random access memory of the computer 20 are a number of software program components to be described next. A network interface connector (NIC) driver program 26 communicates with the socket 22, and more specifically the network interface connector 24 when plugged into the socket 22. The NIC driver 26 includes a network interface

connector (NIC) driver routine 28 that accomplishes the communication of data (information) and signals between the socket 22 and network interface connector 24 (when properly configured), on the one hand, and a network software program 30, on the other hand. Communication with the network software program 30 is accomplished through one or more protocol stack driver programs 32. The NIC driver program 26 accomplishes low-level control of the network interface card 24, and the protocol stack driver programs 32 accomplish higher-level interpretation and formatting of the data.

Information flowing through the network interface connector 24 is in the form of a bidirectional binary information stream containing the data that is to be communicated, and also various types of interrupt and other signalling pulses that notify the computer (if generated by the network) or the network (if generated by the computer) of important events that are to occur. The NIC driver program 26 controls the physical operation of the card, and provides transmit and receive functionality. The protocol stack driver 32 program exchanges data with the NIC driver program 26 for transmission and reception (dialog control), and provides higher-level data encoding, formatting, and interpretation. The NIC driver program 26 and the protocol stack driver programs 32 remain resident within the computer, but are called for use only when required to perform networking interfacing functions.

Within the NIC driver program 26, a hardware reset routine 38 is a set of instructions that, upon execution, resets the hardware of the network interface connector 24. The NIC driver program 26 also accesses files of particular relevance to the present invention. Specifically, a configuration file 34 stores the user's preferred configuration for the NIC.

The NIC driver program 26 communicates with elements of the operating system 39 (such as DOS/Windows). In the operating system 39, an interrupt table 36 contains the addresses of specific software routines stored within the computer 20 to respond to network interrupts received through the network interface connector 24 (when plugged in) from the network. One interrupt vector stored in the interrupt table 36 corresponds to the NIC interrupt service routine of the NIC driver program 26.

The NIC driver program 26 communicates with a client driver program

40. The client driver program 40 utilizes card and socket services programs 42 to configure the hardware socket 22. This configuring establishes hardware resources for the network interface connector 24 (when plugged in) from a hardware resource table 43. Such hardware resources include, for example, the interrupt request level (IRQ) and the input/output (I/O) registers. These resources are identified by the NIC driver program 26 in conjunction with the known requirements of the network software program 30. The NIC driver program 26, network software program 30, protocol stack driver 32, and client driver 40 provide one source of hardware resources to be used in conjunction with the card and socket services program 42. Other configurations, indicated at numeral 44, are supplied by other programs in the computer 20. These other configurations, used in relation to other types of plug-in connectors such as extended memory or computational aids, are not pertinent to the present invention, and are not described in detail.

Some elements of the structures and software depicted in Figure 1 are known in the art and available commercially. Examples of the basic computer 20, preferably a hand-held computer, include the IBM Thinkpad, the NEC Versa, and the Hewlett Packard Omnibook. Such computers 20 typically have one or two built-in sockets 22, with a standard arrangement to receive a standard network interface connector such as an Ethernet or Token Ring LAN connector. The network software program is supplied by the network that is to be used or can be purchased commercially. Examples of currently available network software programs include NetWare 4.x and Microsoft Windows for Workgroups 3.x. The card in socket services program 42 is available commercially and is a standard component for hand-held computers to service the other configurations 44 available in such computers. Examples of card in socket services programs 42 include System Soft CardSoft™ and Phoenix Card Manager programs.

Figure 2 depicts a preferred embodiment for practicing the startup and binding technique of the invention. The client driver program 40 and NIC driver programs 26 are provided, numerals 60 and 62, respectively. These programs have the capabilities discussed in relation to Figure 1, and perform the functions discussed subsequently. There is an interaction between the client

driver program 40 and the NIC driver program 26 during their execution. Figures 3 and 4 depict this interaction at a program logic flow-chart level of detail for a preferred embodiment of the invention that has been implemented. Figure 3 shows the detailed approach corresponding to the steps 64-72 performed prior to insertion of the network interface connector card 24, and Figure 4 shows the detailed approach for the steps 80-84 wherein the connector card 24 is inserted and binding is completed.

Referring to Figure 2, the client driver program 40, acting through the card and socket services program 42, senses the physical presence of a network interface connector 24 plugged into the socket 22, numeral 64. This sensing can be accomplished in any operable manner. For example, a separate pin could be provided within the connector 24 to mate with a pin receiver in the socket 22. In another approach, electrical continuity between two contacts within the socket 22 could be sensed, with that continuity provided through the corresponding pins of the connector 24 when present. If the network interface connector 24 is already inserted into the socket 22, the standard setup and binding routine is followed and the present invention need not be implemented. If the network interface connector is not already inserted into the socket 22, the approach of the present invention is utilized, and the next-described steps are followed. The client driver program 40 provides the connector insertion status to the NIC driver program 26.

The NIC driver program 26 provides several types of information, numeral 66. The NIC driver program 26 provides the memory address of the hardware reset routine 38. The client driver 40 stores this address so that this routine 38 can be later located and executed. Upon card insertions or extractions, the client driver 40 calls this hardware reset routine 38. The NIC driver program 26 also provides to the client driver program 40 the categories of hardware resources required by the card for proper operation, typically including an interrupt request level (IRQ), input/output (I/O) address range, and/or a memory address range. The NIC driver program 26 also provides to the client driver program 40 any specific hardware resource values that the user of the computer has requested for the card, this information having been previously stored in the configuration file 34.

The client driver program 40 receives this information from the NIC driver program 26 and uses it in the appropriate manner, numeral 68. The memory address of the hardware reset routine 38 is stored for later accessing and execution. The client driver program 40 reserves any specifically requested hardware resources provided in the configuration file 34. The client driver program 40 reserves these hardware resources for later use when the computer is connected to the network and notifies the NIC driver program 26 so that the resources are acceptable.

During typical operation of a NIC, an interrupt service routine (ISR) resides in the NIC driver routine 28. The client driver program 40 reserves a hardware IRQ from the card and socket services programs 42. If there is a specific IRQ value provided in the configuration file 34, this IRQ is reserved by the client driver program 40. If no specific IRQ value is requested in the configuration file 34, the client driver program 40 reserves the first available IRQ provided by the card and socket services program 42.

The reserved IRQ has a corresponding entry in the system interrupt vector table 36. The NIC driver program 26 places the address of the ISR into the system interrupt vector table 36 at the entry corresponding to the IRQ reserved by the client driver program 40, numeral 70. When the network connection is later established, this ISR routine will be called when the hardware IRQ signals that communication is to occur. The NIC driver program 26 places the node identification of the network interface connector 24, which is provided in the configuration file 34, into a configuration table. The network/protocol programs use the node identification during transmission to identify the source.

Lastly, the NIC driver program 26 notifies the client driver 40 that it is loaded and initialized, and is ready for connector insertion, numeral 72. The NIC driver program 26 also notifies the network software program 30 that loading and initializing are complete. At this point, the setup initialization is complete, but no hardware reset or initialization has occurred because the network interface connector 24 has not been inserted into the socket 22. However, at such time as the connector 24 is inserted and requires its reset prior to network communication, the software of the computer requires no data

or program loading that might interfere with ongoing processes, although some functions are performed as described next.

Figure 3 depicts the stepwise implementation of these procedures. When the present approach is activated, a call is made from the NIC driver program 26 to the client driver program 40, numeral 100, for its entry point. The client driver program 40 returns its entry point, numeral 102, which is stored by the NIC driver program 26, numeral 104.

As depicted generally in step 64 of Figure 2, the NIC driver program 26 requests the insertion status of the network interface connector card 24 from the client driver program 40, numeral 106. The insertion status is determined by the client driver program 40, numeral 108, and returned to the NIC driver program 26. The NIC driver program 26 tests the connector card insertion status, numeral 110. If the connector card 24 is already inserted into the socket 22, the standard startup and binding sequence is followed, numeral 112, and the present approach is not required.

The NIC driver program 26 provides addresses and hardware resources (step 66 of Figure 2). The NIC driver program 26 first determines whether the requested hardware resources were specifically requested in the configuration file 34, numeral 114. If so, those values are set, numeral 116. If not, the first available values are selected, numeral 118. These established values are communicated to the client driver program 40, numeral 120.

The client driver program 40 stores the addresses of the reset routines and reserves hardware resources (step 68 of Figure 2). The client driver program 40 determines whether a specific IRQ was requested, numeral 122. If so, that value is reserved, numeral 124. If not, the first available IRQ is reserved, numeral 126.

The client driver program 40 determines whether a specific I/O and/or memory range has been requested for later use in the network application, numeral 128. If so, that I/O and/or memory range is reserved, numeral 130. If not, no specific action is required, and any available memory range will be used at the later time.

The client driver program 40 stores the hardware reset routine address, numeral 132. The client driver program 40 then returns the status and reserved

resource values to the NIC driver program 26, numeral 134. The NIC driver program 26 stores the ISR address in the interrupt vector table 36 (step 70 of Figure 2), numeral 136. The NIC driver program 26 notifies the client driver of completion (step 72 of Figure 2), numeral 138, and the client driver program 40 sets a "NIC driver started" flag, numeral 140. The startup routine of steps 64-72 is complete.

Figure 2 also depicts the events that occur after the network interface connector 24, preferably in the form of a card, is inserted, numeral 80, following steps 60-72. The client driver program 40 configures the socket 22 for the required hardware resources, numeral 82. Hardware resources reserved by the client driver program 40 in numeral 68 are configured for use. The IRQ to be used will always have been reserved, and the reserved IRQ will be configured for use. If other hardware resources have been reserved as a result of being specified in the configuration file 34, specifically I/O base address and range and/or memory base address and range, these resources will be configured. If these resources have not been reserved, the client driver configures any available bases and ranges for these resources. The client driver program informs the NIC driver program 26 of the I/O and/or memory ranges selected and configured.

The client driver 40 reads any hardware specific information memory on the network interface connector 24, numeral 82. The client driver 40 configures the socket and informs the NIC driver program 26 of the hardware resources configured. That is, the connector 24 may be far more than simply a conventional pin connector, and may have onboard memory that is used to store hardware specific information required by the computer such as onboard buffer memory. This information is read into the computer and passed to the NIC driver program 26.

Lastly, the client driver 40 calls the hardware reset routine 38 at the address previously provided to it (at numeral 66), numeral 84. The routine 38 is executed to reset the hardware network interface connector 24 to an initial state. At this point, the connector 24 and the NIC driver program 26 are fully prepared and configured for commencing communication between the computer and the network.

Figure 4 depicts the program logical flow sequence for the steps 80-84 of Figure 2. Upon insertion of the network interface connector card 24 into the socket 22, a notification is generated in the client driver program 40, numeral 150. The client driver program 40 checks as to whether the prior startup steps have been performed, numeral 152. If the "NIC driver started" flag was set in step 140, the process proceeds. If not, the computer is not configured for binding according to the invention, and connector card insertion is ignored, numeral 154.

Previously reserved IRQ and memory are next unreserved so that they can be called during the resetting. The IRQ previously reserved in steps 124 and 126 is unreserved, numeral 156. The client driver program 40 also checks to see if specific I/O or memory ranges were reserved in step 130, numeral 158. If so, they are unreserved, numeral 160.

The client driver 40 configures the socket 22, numeral 162, and provides the configuration information to the NIC driver program 26, numeral 164 (step 82 of Figure 2). The client driver 40 further calls upon the NIC driver program 26 to initiate the hardware reset routine (step 84 of Figure 2) in step 164.

The NIC driver program 26 enters the hardware reset routine, numeral 166. It stores the hardware configurations that are to be used, numeral 168, and requests, from the client driver program, any parameters that may be stored on the connector card 24, numeral 170. The client driver program 40 obtains any such necessary information stored on the connector card 24, numeral 172. This information is provided to the NIC driver program 26, which resets and/or initializes the connector card 24 to the desired configuration, numeral 174, to complete the startup and binding.

The approach depicted in Figures 2-4 has the advantage over prior practice that the initial portion (steps 60-72) of the setup and binding is completed prior to insertion of the connector 24. This initial portion is preferably performed when the computer is first turned on, so that the initial portion can be completed in a manner that is consistent with the loading of the operating system and other programs. The computer startup is typically performed well before the user seeks to connect to the network. The user can then use the computer for any desired functions, including the use of plug-in

cards of various types in the socket 22. When the user wishes to communicate with the network, the connector 24 is inserted into the socket 22, hardware reset and initialization are performed, and network communications can begin without interfering with other operations of the computer. In the prior approach it was necessary to have the network interface connector inserted into the socket at the time of startup and binding. This approach could interfere with the other operations of the computer because the computer had to be turned off and then back on when the connector was inserted, potentially interfering with ongoing operations.

Although a particular embodiment of the invention has been described in detail for purposes of illustration, various modifications and enhancements may be made without departing from the spirit and scope of the invention. Accordingly, the invention is not to be limited except as by the appended claims.

CLAIMS

What is claimed is:

1. A method for accomplishing network startup and binding of a portable computer, comprising the steps of:

providing a portable computer configured to receive a network interface connector but having no network interface connector connected thereto;

establishing a network interface connector reset and configuration protocol, without the network interface connector connected to the portable computer;

connecting the network interface connector to the portable computer; and

resetting and configuring the network interface connector using the protocol established in the step of establishing.

2. A method for accomplishing network startup and binding of a portable computer having a connector socket which can receive a network interface connector, the method comprising the steps of:

providing network interface connector driver means for communicating data with the connector socket;

providing client driver means for establishing a configuration for the connector socket, the configuration to be implemented at such time as a network interface connector is introduced into the connector socket;

the client driver means determining that no network interface connector is connected to the connector socket and providing that information to the network interface connector driver means; and

the network interface connector driver means and the client driver means cooperating to establish a hardware resource information set within the portable computer sufficient to operate the network interface connector at such time as the network interface connector is introduced into the connector socket.

3. The method of claim 2, including the additional steps, after the step of the network interface connector driver means and the client driver means cooperating to establish, of

connecting the network interface connector to the computer; and

resetting and configuring the network interface connector using the protocol established in the step of cooperating to establish.

4. A method for accomplishing network startup and binding of a portable computer having a connector socket which can receive a network interface connector, the method comprising the steps of:

providing a network interface connector driver program stored in the portable computer, the network interface connector driver routine being operable to communicate data between a network software program stored in the personal computer and the connector socket;

providing a client driver program stored in the portable computer, the client driver program being operable to configure the connector socket for use by a network interface connector at such time as a network interface connector is introduced into the connector socket;

the client driver program determining that no network interface connector is connected to the connector socket and providing that information to the network interface connector driver program;

the network interface connector driver program providing to the client driver program the locations of interrupt-driven communications routines within the network interface connector driver program, the location of a network interface connector driver hardware reset routine within the network interface connector driver program, and the nature of a set of network hardware resources requested by a user of the portable computer, the set of network hardware resources including at least an interrupt request level;

the client driver program placing the locations of the interrupt-driven communications routines into an interrupt table, storing the location of the network interface connector driver hardware reset routine, and reserving for later use the network hardware resources provided by the network interface connector driver program;

the network interface connector driver program placing the node identification of the network interface connector into a configuration file and providing the node identification to the network software program; and

the network interface connector driver program notifying the client

driver program and the network interface connector driver program of the completion of loading.

5. The method of claim 4, including the additional steps, after the step of notifying, of

connecting the network interface connector to the socket;

the client driver program configuring the connector socket;

the client driver program reading from the network interface connector any hardware specific information required for communication between the network interface connector driver program and the network interface connector and providing this information to the network interface connector driver program; and

the client driver program calling the hardware reset routine using the address provided during the step of the network interface connector driver program providing.

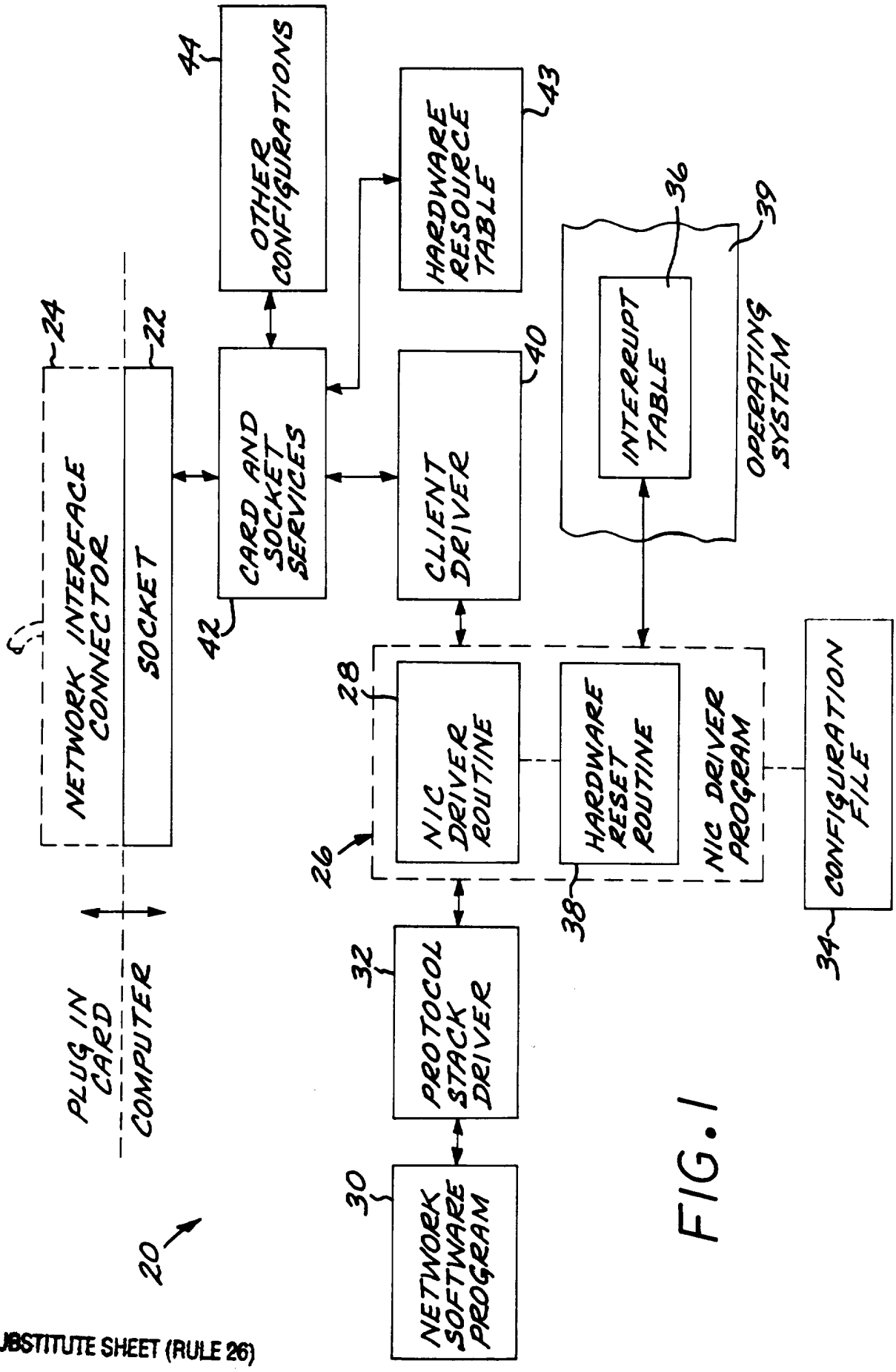

20
PLUG IN CARD
COMPUTER
NETWORK INTERFACE CONNECTOR
SOCKET
24
22
CARD AND SOCKET SERVICES
42
CLIENT DRIVER
40
OTHER CONFIGURATIONS
44
HARDWARE RESOURCE TABLE
43
INTERRUPT TABLE
36
OPERATING SYSTEM
39
NIC DRIVER ROUTINE
28
HARDWARE RESET ROUTINE
NIC DRIVER PROGRAM
26
38
CONFIGURATION FILE
34
PROTOCOL STACK DRIVER
32
NETWORK SOFTWARE PROGRAM
30

FIG. 1

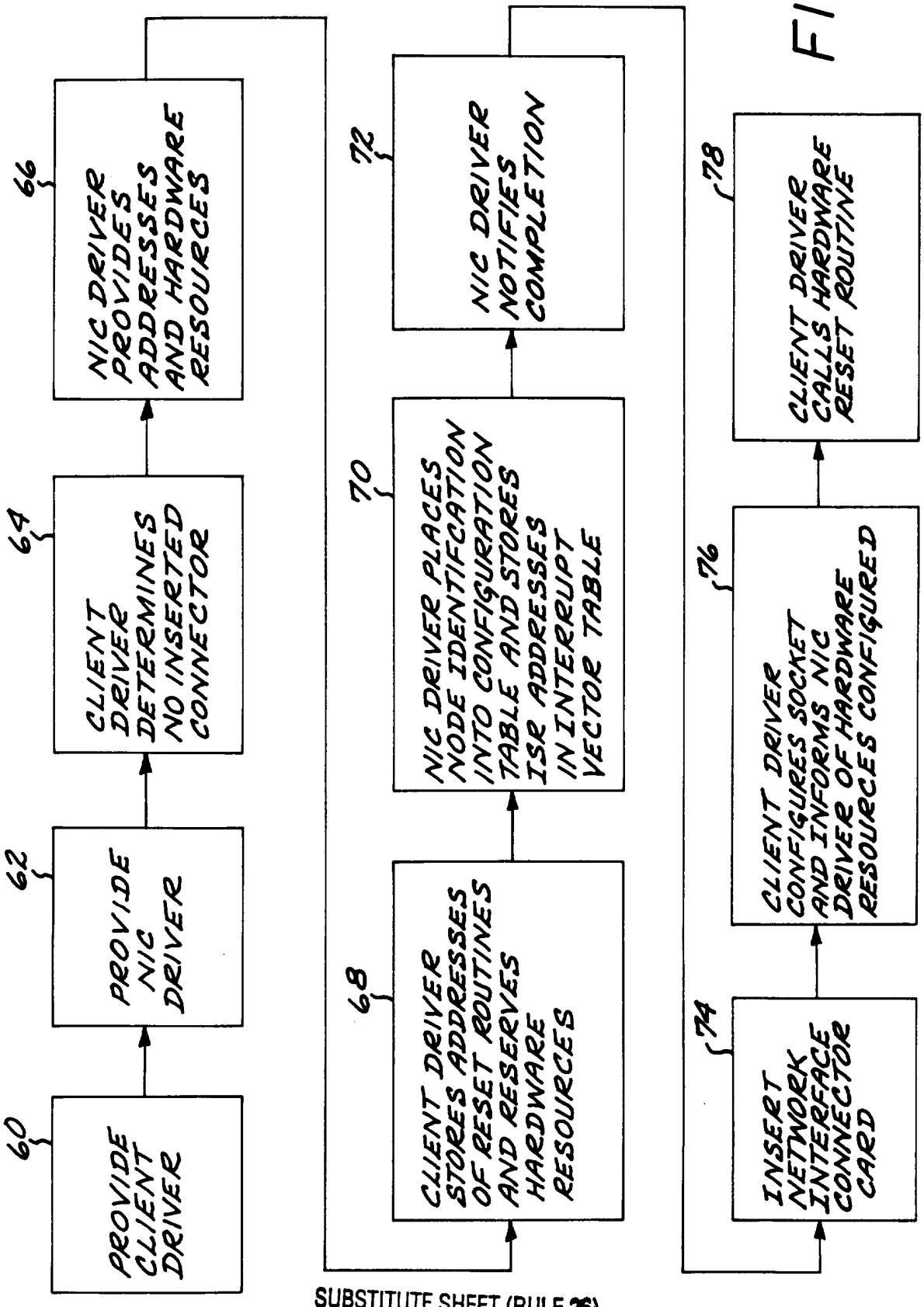

60
PROVIDE CLIENT DRIVER
62
PROVIDE NIC DRIVER
64
CLIENT DRIVER DETERMINES NO INSERTED CONNECTOR
66
NIC DRIVER PROVIDES ADDRESSES AND HARDWARE RESOURCES
68
CLIENT DRIVER STORES ADDRESSES OF RESET ROUTINES AND RESERVES HARDWARE RESOURCES
70
NIC DRIVER PLACES NODE IDENTIFICATION INTO CONFIGURATION TABLE AND STORES ISR ADDRESSES IN INTERRUPT VECTOR TABLE
72
NIC DRIVER NOTIFIES COMPLETION
74
INSERT NETWORK INTERFACE CONNECTOR CARD
76
CLIENT DRIVER CONFIGURES SOCKET AND INFORMS NIC DRIVER OF HARDWARE RESOURCES CONFIGURED
78
CLIENT DRIVER CALLS HARDWARE RESET ROUTINE

FIG. 2

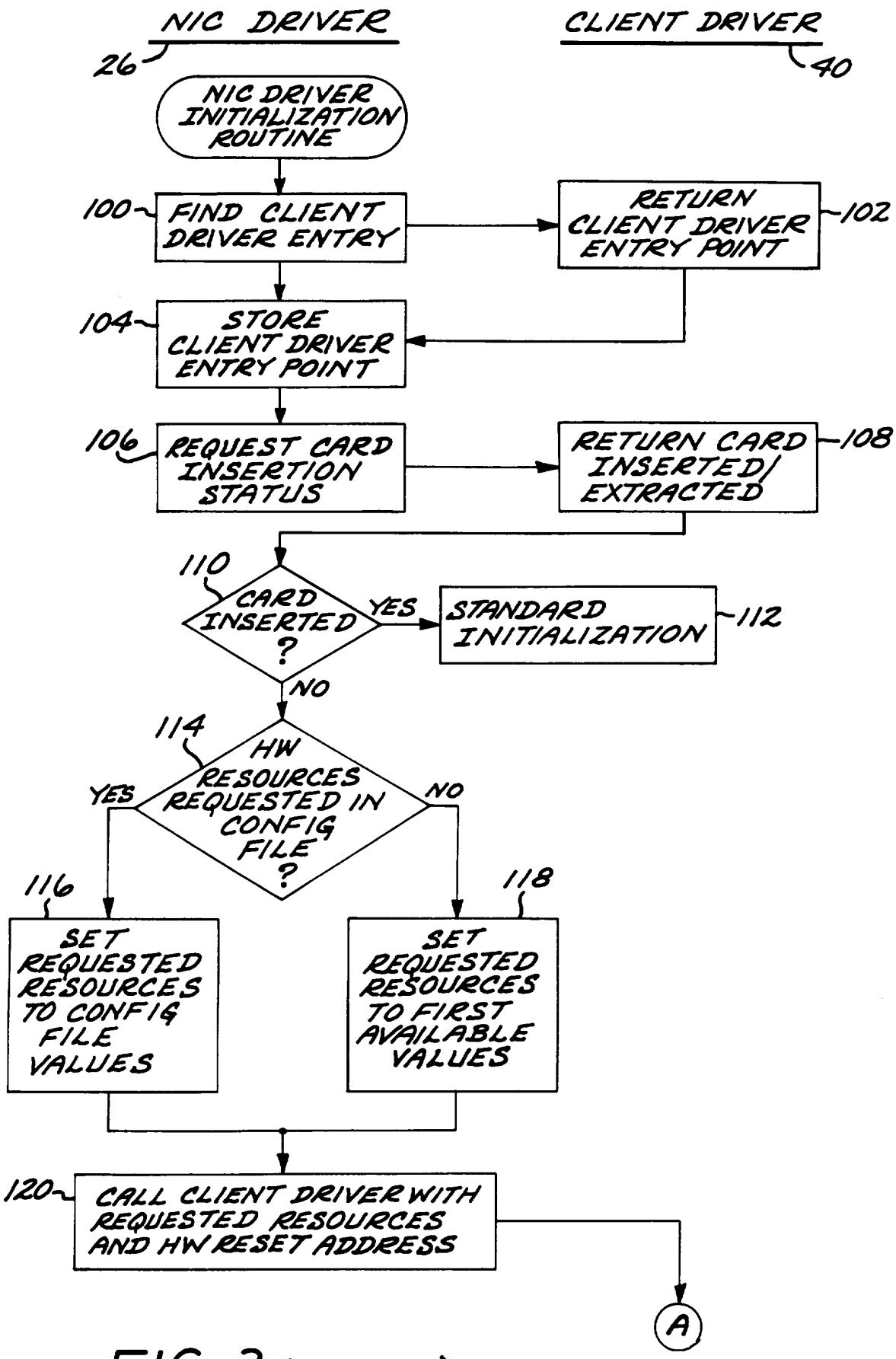

NIC DRIVER
26
CLIENT DRIVER
40
NIC DRIVER INITIALIZATION ROUTINE
100
FIND CLIENT DRIVER ENTRY
RETURN CLIENT DRIVER ENTRY POINT
102
104
STORE CLIENT DRIVER ENTRY POINT
106
REQUEST CARD INSERTION STATUS
RETURN CARD INSERTED/ EXTRACTED
108
110
CARD INSERTED ?
YES
STANDARD INITIALIZATION
112
NO
114
HW RESOURCES REQUESTED IN CONFIG FILE ?
YES
NO
116
SET REQUESTED RESOURCES TO CONFIG FILE VALUES
118
SET REQUESTED RESOURCES TO FIRST AVAILABLE VALUES
120
CALL CLIENT DRIVER WITH REQUESTED RESOURCES AND HW RESET ADDRESS
A

FIG. 3 (SHEET 1)

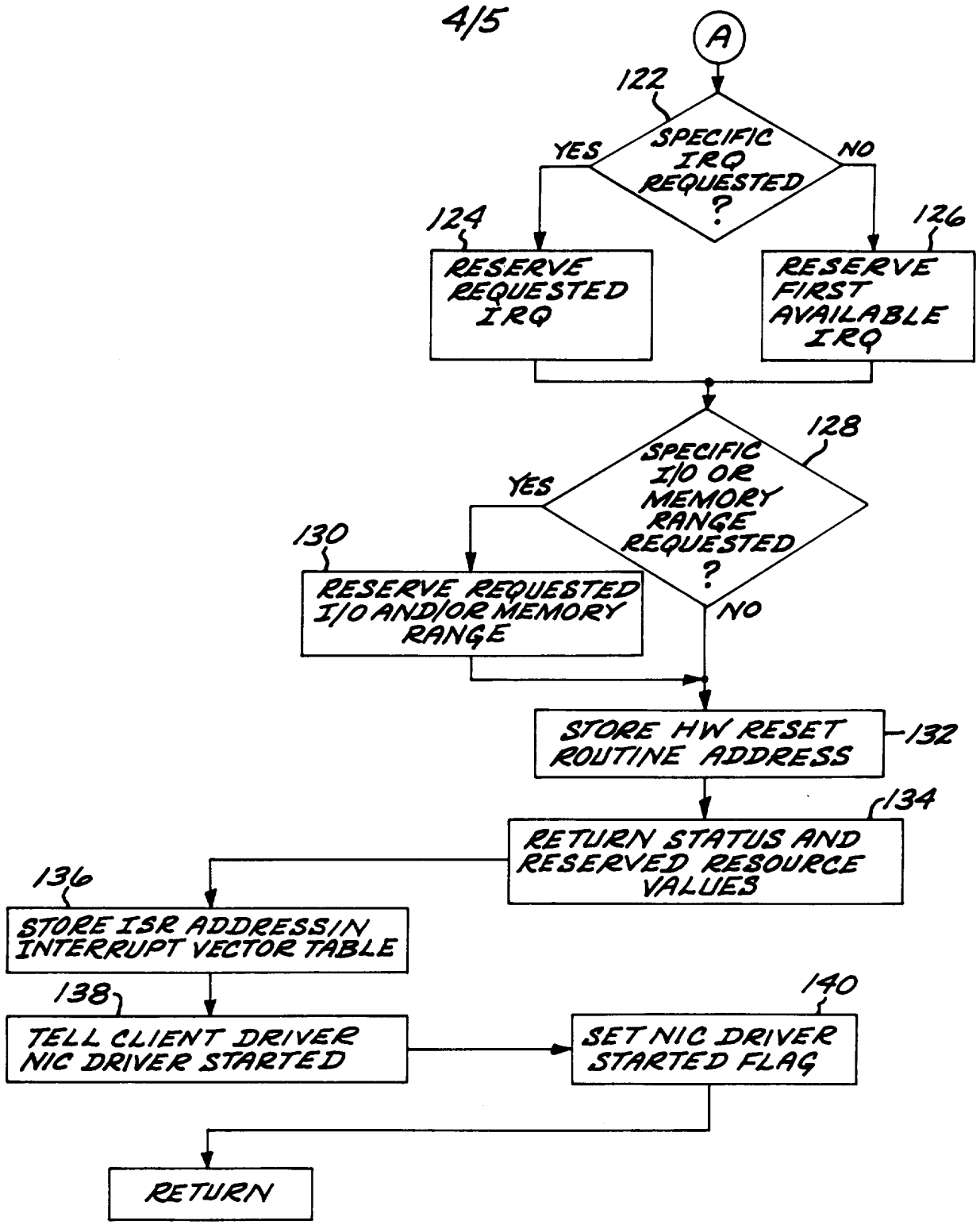

A
122
SPECIFIC IRQ REQUESTED ?
YES
NO
124
RESERVE REQUESTED IRQ
126
RESERVE FIRST AVAILABLE IRQ
128
SPECIFIC I/O OR MEMORY RANGE REQUESTED ?
YES
NO
130
RESERVE REQUESTED I/O AND/OR MEMORY RANGE
132
STORE HW RESET ROUTINE ADDRESS
134
RETURN STATUS AND RESERVED RESOURCE VALUES
136
STORE ISR ADDRESS IN INTERRUPT VECTOR TABLE
138
TELL CLIENT DRIVER NIC DRIVER STARTED
140
SET NIC DRIVER STARTED FLAG
RETURN

FIG. 3 (SHEET 2)

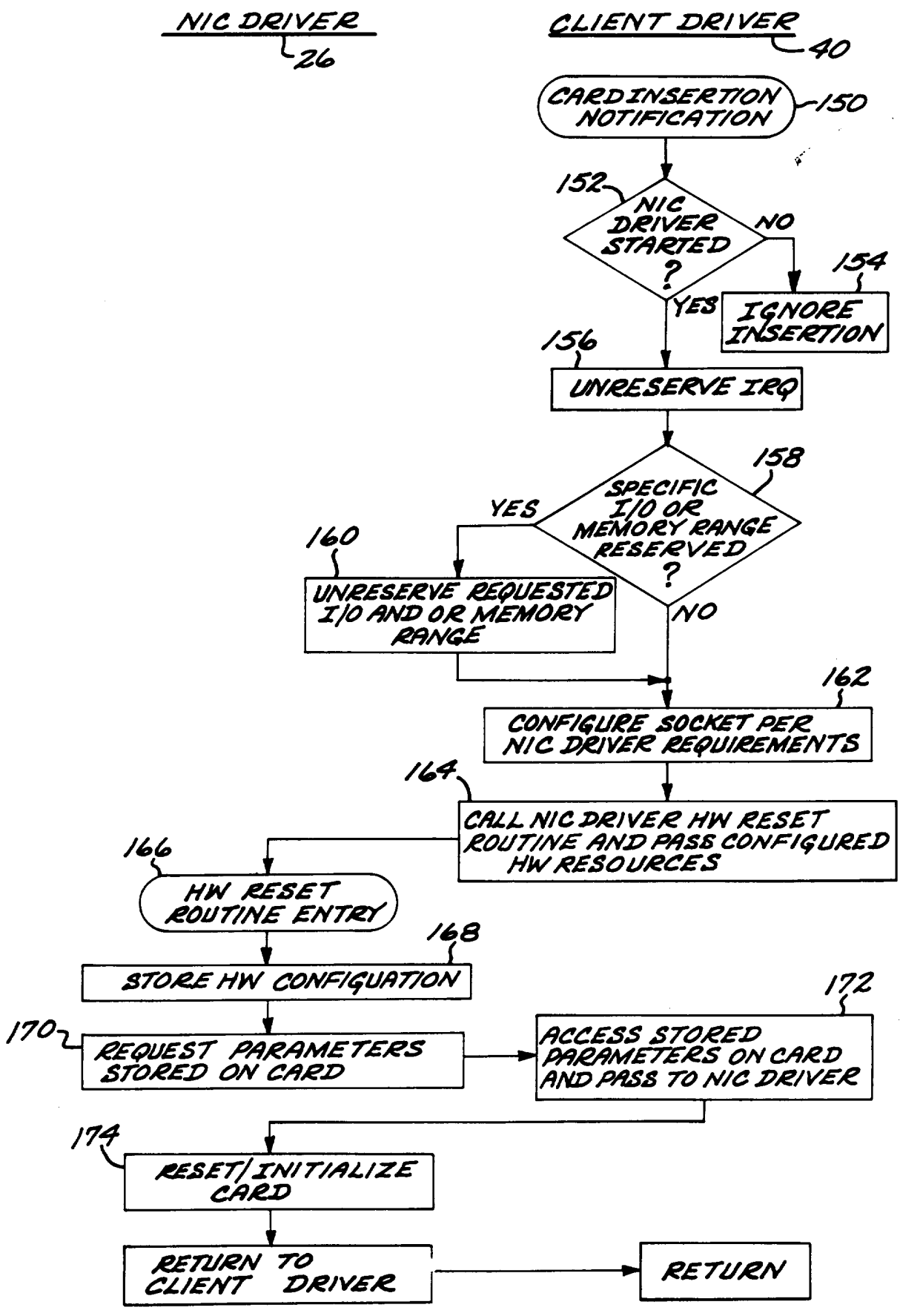

NIC DRIVER
26
CLIENT DRIVER
40
CARD INSERTION NOTIFICATION
150
152
NIC DRIVER STARTED ?
NO
154
IGNORE INSERTION
YES
156
UNRESERVE IRQ
158
SPECIFIC I/O OR MEMORY RANGE RESERVED ?
YES
160
UNRESERVE REQUESTED I/O AND OR MEMORY RANGE
NO
162
CONFIGURE SOCKET PER NIC DRIVER REQUIREMENTS
164
CALL NIC DRIVER HW RESET ROUTINE AND PASS CONFIGURED HW RESOURCES
166
HW RESET ROUTINE ENTRY
168
STORE HW CONFIGUATION
170
REQUEST PARAMETERS STORED ON CARD
172
ACCESS STORED PARAMETERS ON CARD AND PASS TO NIC DRIVER
174
RESET/INITIALIZE CARD
RETURN TO CLIENT DRIVER
RETURN

FIG.4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/16016

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 12/00, 13/00
US CL : 395/200.10, 800, 700

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)

U.S. : 395/200.10, 800, 700

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 practicable, search terms used)

Dialog database, internet linux sites at tsx-11.mit.edu

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	"Linux Ethernet HOWTO", Part 2/2, chapter 5, posted 16 March 1994 at tsx-11.mit.edu internet site. Page 2 (lines 1-25) and page 2 (line 35) through page 4 (line 16).	1-5
X	"A 3c589 Etherlink 3 Ethernet driver for Linux" published electronically by tsx-11.mit.edu internet site by Donald Becker, dated 3 May 1994. Page 2 (lines 30-32) and page 3 (lines 5-17)	1-3

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"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
"E"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 28 FEBRUARY 1996	Date of mailing of the international search report 18 MAR 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer MEHMET GECKIL Telephone No. (703) 305-9676

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