



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

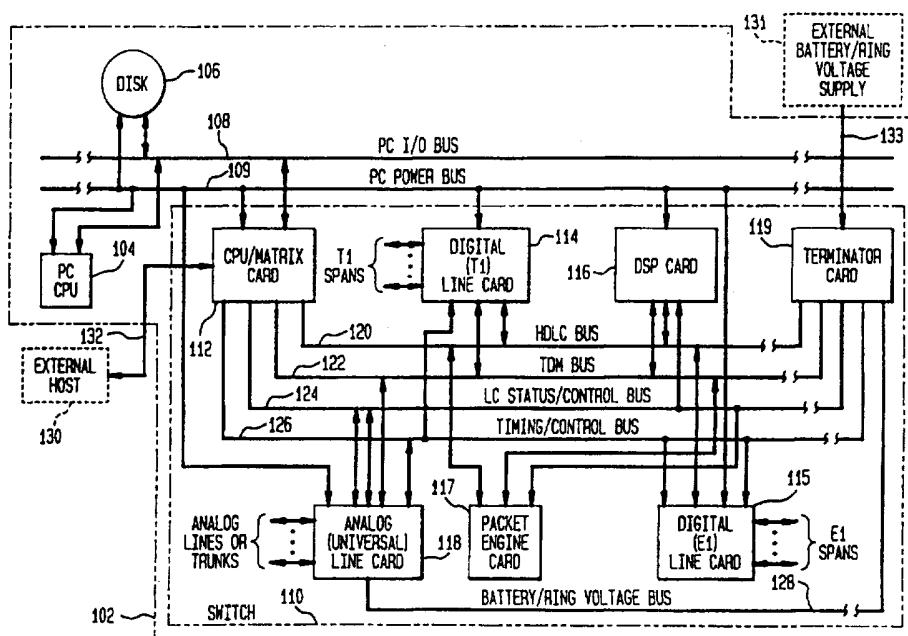
(51) International Patent Classification ⁶ : H04Q 3/62, 11/04, H04L 29/06, H04Q 3/545	A1	(11) International Publication Number: WO 97/20439 (43) International Publication Date: 5 June 1997 (05.06.97)
(21) International Application Number: PCT/US96/18959 (22) International Filing Date: 27 November 1996 (27.11.96) (30) Priority Data: 08/566,414 30 November 1995 (30.11.95) US (71) Applicant: EXCEL, INC. [US/US]; 255 Independence Drive, Hyannis, MA 02601 (US). (72) Inventor: HEBERT, Mark, P.; 34 Harvest Drive, Kingston, MA 02364 (US). (74) Agents: ATTAYA, Michael, E. et al.; Cesari and McKenna L.L.P., 30 Rows Wharf, Boston, MA 02110 (US).		(81) Designated States: AL, AU, BB, BG, BR, CA, CN, CZ, EE, GE, HU, IL, IS, JP, KP, KR, LK, LR, LT, LV, MG, MK, MN, MX, NO, NZ, PL, RO, SG, SI, SK, TR, TT, UA, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published <i>With international search report.</i> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>

(54) Title: TELECOMMUNICATIONS SWITCH HAVING A UNIVERSAL APPLICATIONS PROGRAM INTERFACE FOR STANDARDIZED INTERACTIVE CALL PROCESSING COMMUNICATIONS

(57) Abstract

The present invention is a standardized host-to-switch application program interface (API) for performing call control processing, capable of being customized to meet telecommunications application and network signalling protocol requirements. The universal API comprises one or more generic messages having programmable fields for transmitting commands, status, and data between the host application and the switch. The present invention further comprises a programmable telecommunication switch that provides a user with the ability to define a desired API protocol, either "standard" or custom in nature, for performing any desired switching functions. The present invention includes

a protocol development environment which enables a user to define a separate finite state machine for each port provided by the switch. Each finite state machine may be independently defined by combining a series of elementary processing steps, called atomic functions, into primitives, which are in turn combined with states and events to define the desired state machine. Such state machines may include atomic functions configured to generate predetermined messages under predetermined conditions and containing predetermined information. Such state machines may further include the ability to respond to state events that include the receipt of generic API messages configured to provide the state machine with information from the host application.



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	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

TELECOMMUNICATIONS SWITCH HAVING A UNIVERSAL APPLICATIONS PROGRAM INTERFACE FOR STANDARDIZED INTERACTIVE CALL PROCESSING COMMUNICATIONS

5 BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to the field of telecommunications and, more specifically, to a universal applications program interface (API) for standardized interactive call control processing with a programmable telecommunication switch and a host computer supporting various telecommunications applications.

Description of the Related Art

15 Programmable telecommunication switches are used in a wide variety of applications such as voice messaging, telemarketing services and the like. A computer that runs a telecommunications application program. A customer may programmable switch is usually controlled by a host device, which is typically a either purchase a commercially available application program that is compatible with the host and switch hardware or may elect to write a custom program.

20 In most applications, a programmable switch is connected to a public telephone network by one or more analog trunks or digital spans (e.g., a T1 span) which are terminated at the switch. The switch may also terminate one or more "lines" which are connected to devices such as telephone sets. Communication over any given trunk, span or line is carried out in accordance with an assigned signalling protocol.

25 For various switching system applications, the sequence of switching events must be controlled and the switching functions must be performed in accordance with the requisite protocols. Throughout the world, there are numerous "standard" signalling protocols in use, including E&M wink start, loopstart, groundstart, international compelled R2 using MFR2 address signalling, and E1 Channel Associated
30 Signalling (CAS) protocols using DTMF/MFR1 signalling. Typically, conventional programmable switches are configured such that a particular signalling protocol is associated with a particular trunk, span or line.

To control the telecommunications switch at the various levels necessary to satisfy specialized switching functions, conventional host applications have been configured to generate digital signal commands corresponding to a plurality of switching events. Correspondingly, conventional communications switches have been
5 configured to generate digital signal responses related to the processing of these events at the ports. These messages are constant or "hard-coded" messages, each configured to communicate specific information between the host application and the switch. The interface between the telecommunications application and the switch through which these messages are transferred is referred to as an applications program interface, or
10 API.

Each of the signalling protocols requires predetermined host-to-switch call control processing protocols to be established, each protocol including the exchange of one or more constant messages. Thus, to control the programmable switch to perform the requisite switching events necessary to maintain communications, communications
15 switches must be capable of supporting extremely large number of these specific host-to-switch command messages and associated protocols. Accordingly, each signalling protocol has associated with it one or more different message sets stored and indexed at the host as well as at the switch. The message sets and resulting host-to-switch protocol are also dependent upon specific telecommunications applications
20 requirements, such as the amount and type of information an application requires for it to appropriately control the switch to support a particular signalling protocol.

Furthermore, conventional programmable switches may be connected between the public telephone network and other devices such as a voice messaging system. Because such devices may perform specialized functions and are not intended to
25 connect directly to the public telephone network, they do not typically adhere to standard signalling protocols. Thus, for a user to be able to control the programmable switch in such a fashion that proper communication is maintained both, with the public telephone network and with other devices connected to the switch, complex and varied API signalling protocol requirements must be satisfied. Conventional communications
30 switches implement numerous specific sets of API messages to support these varied requirements.

As a result of the implementation of constant messages and the various telecommunications applications and signalling protocol requirements, there has been no standardization of the interface between the host applications and the telecommunications switch. This has led to increased cost in developing the necessary hardware and software to support specific API protocols to satisfy host applications requirements as well as signalling protocol requirements for each trunk, span, and line.

Furthermore, as a result of having separate and distinct API messages, each dedicated to a specific command or data transfer, the addition of features to the telecommunications switch necessitates the creation and implementation of one or more additional API messages to support the associated API protocol between the host and switch. To implement each new unique message, a costly and time-consuming software change to the switch and host must be made.

What is needed, therefore, is a standardized API message protocol supporting host-to-switch call control processing that may be used regardless of the host application or signalling protocol requirements. Furthermore, such a universal API protocol must be sufficiently flexible and versatile to be customized to support present and future requirements of telecommunications applications and signalling protocols now or later developed.

SUMMARY OF THE INVENTION

The present invention is a standardized host-to-switch application program interface (API) for performing call control processing, capable of being customized to meet telecommunications application and network signalling protocol requirements. The universal API comprises one or more generic messages having programmable fields for transmitting commands, status, and data between the host application and the switch. The present invention further comprises a programmable telecommunication switch that provides a user with the ability to define a desired API protocol, either "standard" or custom in nature, for performing any desired switching functions.

The present invention includes a protocol development environment which enables a user to define a separate finite state machine for each port provided by the

switch. Each finite state machine may be independently defined by combining a series of elementary processing steps, called atomic functions, into primitives, which are in turn combined with states and events to define the desired state machine. Such state machines may include atomic functions configured to generate predetermined messages under predetermined conditions and containing predetermined information. Such state machines may further include the ability to respond to state events that include the receipt of generic API messages configured to provide the state machine with information from the host application.

In addition, the present invention may serve as a development tool for creating customized API protocols having customized standard messages supporting telecommunications applications such as personal communications services (PCS), 800/900 service, voice mail, telemarketing, among others. The present invention may also be used to control or manage a wide variety of communications services within a programmable switch through the transfer of the generic API messages, including conferencing, voice recorded announcements, tone generation, tone reception, call progress analysis, voice recognition, voice compression and fax encoding/decoding.

The universal API of the present invention may be implemented to achieve communications internal to the switch as well. For example, the standardized messages of the universal API may be used to support communications between any software layer within the switch.

Advantageously, the generic message structure of the present invention enables additional call processing features to be added to the telecommunications switch that the host can initiate without implementing additional content-specific API messages dedicated to that feature. This enables the creation of customized API message protocols that can grow beyond the limitations of specific messages for specific features and functions.

Another advantage of the generic message structure of the present invention is that it provides the commonality and flexibility necessary to be a standardized interface for application development. This significantly reduces the complexity of the host/switch communications interface and eliminates the cost of supporting an interface composed of numerous specialized messages.

Another advantage of the present invention is that it provides the user with the ability to transmit and receive information to all software layers of the switch using standardized messages. Significantly, this eliminates the burden of having to store large numbers of distinct messages for managing dissimilar functions performed by the same or different software layers of the switch. Thus, the large message sets stored and indexed in conventional switches is eliminated by the present invention.

Another advantage of the present invention is the increased degree of interaction with the host that can be achieved simply by introducing at various processing points an atomic function that sends and receives data into numerous locations in the switch.

Another advantage of the present invention is that it enables a user to create multiple network signalling protocols by creating separate state machines to address each variation of a signalling protocol. The universal API may be programmed to achieve the necessary communications to support each of these protocol-specific state machines. Thus, the structure of the messages comprising the host-to-switch interface may remain unchanged despite the multiple signalling protocols supported by the switch.

Further features and advantages of the present invention as well as the structure and operation of various embodiments of the present invention are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the left-most one or two digits of a reference number identifies the drawing in which the reference number first appears.

25

BRIEF DESCRIPTION OF THE DRAWINGS

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

30

Figure 1 is a block diagram of a programmable telecommunications switch which may be programmed by a user in accordance with a preferred embodiment of the present invention;

Figure 2 is diagram which depicts the layers of software used to control the
5 switch of Figure 1;

Figures 3A and 3B depict some of the specific features and functions associated with each of the software layers depicted in Figure 2;

Figure 4 is a block diagram of a finite state machine development environment constructed in accordance with a preferred embodiment of the present invention;

10 Figure 5 is a block diagram illustrating the structure and contents of a generic PPL Event Indication and PPL Event Request message of the present invention;

Figure 6 is a block diagram of a PPL Event Indication Acknowledgement and PPL Event Request Acknowledgment message of the present invention.

Figures 7A and 7B are a state diagram of a finite state machine for providing
15 call control processing utilizing the universal API of the present invention to support a highly interactive host telecommunications application requirement;

Figure 7C is an interface diagram of the universal API supporting the call control processing illustrated in Figures 7A and 7B;

Figures 7D and 7E are a diagram of the finite state machine of Figures 7A and
20 7B in which each series of atomic functions is defined as a primitive;

Figures 7F and 7G are tables showing the correspondence between the atomic functions, primitives and states of Figures 7A-7E;

Figures 8A and 8B are a state diagram of a finite state machine for providing call control processing utilizing the universal API of the present invention to support a
25 limited interactive host telecommunications application requirement;

Figure 8C is an interface diagram of the host-to-switch API generated by the call control processing illustrated in Figures 8A and 8B;

Figures 8D and 8E are a diagram of the finite state machine of Figures 8A and 8B in which each series of atomic functions is defined as a primitive;

30 Figures 8F and 8G are tables showing the correspondence between the atomic functions, primitives and states of Figures 8A-8E;

Figure 9 is a functional block diagram illustrating an exemplary process flow to create a PPL Event Indication message;

Figure 10 is a block diagram of the message buffer created by an Layer 4 PPL processor during the creation of the PPL Event Indication message of Figure 9; and

5 Figure 11 is a functional block diagram illustrating an exemplary process flow to create a PPL Event Request message.

DETAILED DESCRIPTION OF THE INVENTION

10 Figure 1 shows a commercially available personal computer (PC) 102 which includes a PC central processing unit (CPU) 104 and a hard disk drive 106 interconnected by a PC input/output (I/O) bus 108 and a PC power bus 109. The PC 102 is preferably a PC-AT[®], sold by International Business Machines (IBM), or a compatible thereof. Other personal computers having more memory or more powerful
15 CPUs than the PC-AT[®] may also be used. The PC 102 preferably operates under an application-oriented operating system, such as DOS[®] or UNIX[®].

The PC 102 consists of a chassis or housing in which a motherboard is mounted, along with the disk drive 106 and other optional assemblies such as floppy disk drives, modems and the like. The PC CPU 104 is mounted on the motherboard,
20 which includes a series of edge connectors into which other boards (cards) may be inserted and thereby connected to the PC I/O and power busses 108 and 109.

A programmable telecommunication switch 110 resides within the PC 102. A CPU/matrix card 112 is inserted into one of the slots on the motherboard and thus connected to the busses 108 and 109. The CPU/matrix card 112 is interconnected with
25 a digital (T1) line card 114, a digital (E1) line card 115, a digital signal processing (DSP) card 116, a packet engine card 117, an analog (universal) line card 118 and a terminator card 119 by four busses: a high level data link control (HDLC) or interprocessor bus 120; a time division multiplex (TDM) bus 122; a line card (LC) status/control bus 124; and a timing/control bus 126. A battery/ring voltage bus 128
30 supplies battery voltage (48VDC) and ringing voltage (109VAC) to the analog line

card 118. The terminator card 119 serves to physically terminate busses 120, 122, 124, 126 and 128.

The line cards 114, 115 and 118 and the DSP card 116 are all connected to and receive their basic operating power from the PC power bus 109. Although only one
5 digital (T1) line card 114, one digital (E1) line card 115 and one analog line card 118 are depicted, it should be understood that additional line cards of any type may be added subject to two physical limitations: (1) the maximum switching capacity of the CPU/matrix card 112, and (2) the physical space within the chassis of the PC 102.

An external host 130, which may comprise a separate personal computer,
10 workstation or other computer, may optionally be connected via a communication channel 132 to the CPU/matrix card 112. The CPU/matrix card 112 preferably includes a conventional RS-232 compatible interface for connecting the channel 132. The external host 130 preferably operates under an application-oriented operating system.

15 If desired, the switch 110 can reside on a passive backplane (no PC CPU 104 or disk 106 present) from which it receives electrical power and be controlled by the external host 130. For example, the present invention may be implemented in other processing platforms such as the expandable telecommunications switch disclosed in copending patent application, serial number 08/207,931, titled Expandable
20 Telecommunications System, assigned to the assignee of the present application and which is hereby incorporated by reference in its entirety.

An external battery/ring voltage supply 131 is connected via a path 133 to the terminator card 119. Supply 131 may comprise, for example, a commercially available power supply.

25 With the exception of the digital (E1) line card 115, the DSP card 116 and the packet engine card 117, details regarding the construction of the various cards shown in Figure 1 are set forth in U.S. Patent No. 5,321,744, titled Programmable Telecommunications Switch for Personal Computer, assigned to the assignee of the present application and which is hereby incorporated by reference in its entirety.

30 Digital (E1) line card 115 is preferably constructed using similar hardware to that

disclosed for T1 line card 114, except for differences in conventional circuitry which allow line card 115 to terminate E1 spans as opposed to T1 spans.

Details regarding the construction of the DSP card 116 and the packet engine card 117 are set forth in U.S. Patent No. 5,349,579, titled Telecommunications Switch
5 With Programmable Communications Services, assigned to the assignee of the present application and which is hereby incorporated by reference in its entirety.

Figure 2 is a layer model of the software used to control the programmable switch 110 of Figure 1. The lefthand column of Figure 2 shows seven layers defined in the Open Systems Interconnection (OSI) reference model. The righthand column of
10 Figure 2 shows five layers used to control switch 2 and their general correspondence to the OSI model.

Referring now to both Figures 1 and 2, the Application Layer 5, which corresponds generally with the Application layer of the OSI model, represents application software which typically runs on either the PC CPU 104 or the external
15 host 130. Application Layer 5 software may be used to implement any of a number of desired telecommunications services such as toll free (800) service, voice mail, automatic call distribution (ACD), to name but a few. Application Layer 5 may communicate with any other layer of the programmable switch through the application program interface (API) of the present invention. When Application Layer 5 resides
20 on external host 130, the API manages communications over communication channel 132. When Application Layer 5 resides on PC CPU 104, the API manages call control processing communications over PC I/O bus 108.

Call Management Layer 104, which corresponds generally with the Presentation, Session and Transport layers of the OSI model, represents software
25 which runs on the CPU/matrix card 12. Call Management Layer 4 is responsible for performing centralized call processing functions and providing a common interface to Application Layer 5 regardless of the type or types of network signalling protocols which may be used within the switch 102. Typically, Call Management Layer 4 performs functions which are required following call setup.

30 Network Signalling Protocol Layer 3 corresponds generally with the Network layer of the OSI model. The software represented by Network Signalling Protocol

Layer 3 runs either on the CPU/matrix card 112 or on line cards which include their own microprocessors, such as line cards 114 or 115 or packet engine card 117, and is responsible for in and out-of-band network signalling supervision as well as network protocol level control of incoming and outgoing calls.

5 Link Layer 2 corresponds generally with the Data Link layer of the OSI model. Link Layer 2 software runs on the CPU/matrix card 112, the line cards which include their own microprocessors, the DSP card 116 or the packet engine card 117 (each of which includes its own microprocessor) and is responsible for the detection as well as physical transfer of network signalling information across a network or line interface.

10 Finally, the Physical Layer 1 corresponds to the Physical layer of the OSI model. Line cards 114, 115 and 118 provide physical T1, E1 and analog electrical interfaces, respectively, to the switch 110.

 Figures 3A and 3B are a tabular listing of representative features and functions provided by each of the software Layers 2-5 of Figure 2. The present invention may
15 be used as a development tool to develop suitable software to implement any of the features and functions shown in Figures 3A and 3B. Illustrative examples of the use of the present invention in the context of each of Layers 2-5 are set forth in U.S. Patent No. 5,426,694, assigned to the assignee of the present invention, herein incorporated by reference in its entirety.

20 Figure 4 is an overall block diagram of a finite state machine development environment, constructed in accordance with a preferred embodiment of the present invention, which enables a customer or user to create and define finite state machines for performing desired telecommunications functions, controlled by one or more applications through the universal API of the present invention. Before considering
25 this Figure in detail, the definitions of certain terms should be addressed.

 As used herein, the term *state* refers to a number which represents the current "context" for a particular channel or port. In a preferred embodiment of the present invention, there are three types of states defined: normal, internal and blocking. *Normal* states can be *wait* states (i.e., a SEIZE ACK state, a condition in which further
30 action is suspended until the occurrence of a particular event) or *stable* states (i.e., a conversation is taking place). *Internal* states are used to test conditions and effectively

operate as decision branches. Normal and internal states may be specified by a customer or user, in accordance with present invention, to define a finite state machine for performing a desired function. *Blocking* states are generated automatically by the present invention and are used, on a channel-by-channel basis, in connection with the management of off-board resources.

An *event* is a number which identifies a condition which is accepted by a particular state. Data may be associated with an event.

An *atomic function* is one which performs an elementary task such as setting a timer. User-specified data may be associated with an atomic function. A *primitive* is a predetermined sequence of atomic functions which is invoked upon the occurrence of a particular event. Users may create or define primitives from a library of available atomic functions. In a preferred embodiment, each primitive may contain up to 20 atomic functions.

A *state/event table* defines the valid events for a particular state and the primitive which is invoked upon the occurrence of each such event. In a preferred embodiment, a state/event table may contain up to 100 states and up to 20 events per state.

A *primitive table* defines the primitives which are used by a state/event table. In a preferred embodiment, a primitive table may contain up to 200 primitives.

A *protocol* is defined as the association of various types of tables, the least of which is a state/event table and primitive table, and is identified by a *protocol ID* (a number).

An *API protocol* is defined as the host-to-switch control protocol between host applications and software layers of the switch.

A *program protocol language* (PPL) is a programmable environment for managing network signalling protocols and communications services.

A *data block*, such as those denoted by reference numbers 40a, 40n, is assigned for each channel (port) 0...n of the switch. Each data block 40a, 40n contains the following information pertaining to its respective channel: the current state of the channel; a pointer to an active state/event table; a pointer to an active primitive table; a pointer to an assigned state/event table; and a pointer to an assigned primitive table.

In the case of channel 0, the active state/event table and active primitive table pointers are pointing, as indicated by the phantom lines, to tables which are associated with a resident protocol 0, denoted by reference number 442a. The assigned state/event table and assigned primitive table pointers for channel 0 are pointing to
5 tables which are associated with a dynamically loaded, customer-defined protocol 1, denoted by reference number 444a.

Other protocols which are present and available for use are resident protocols 1...n (442b, 442c) and downloaded, customer-defined protocols n+1...m (444b, 444c). The resident protocols 442a-442c represent preprogrammed or "standard"
10 protocols, which are typically provided by a manufacturer with a switch. In contrast, the customer-defined protocols n+1...m are created by a customer or user and may be completely "custom" or "proprietary" in nature.

A layer dependent atomic function library 446 is connected to provide information to a state machine engine 448. State machine engine 448 is also connected
15 to receive the active state/event table pointer and active primitive table pointer from each of data blocks 440a-440n. Also, as denoted by reference number 450, utilities are provided for layer dependent environment support.

The function of the state machine engine 448 is to drive each channel in accordance with its assigned protocol, which is defined by the assigned state/event
20 table and assigned primitive table. Upon the occurrence of a valid event for a normal state, a primitive is invoked in accordance with the entries in the assigned state/event table. The state machine engine 448 uses the atomic function library 446 to perform the atomic functions represented by the invoked primitive.

The state machine engine 448 will drive through any necessary internal states,
25 automatically generating appropriate blocking states, until the channel once again reaches a normal state. At that time, processing by the state machine engine 448 is complete until the occurrence of another valid event.

Each channel is initially assigned one of the customer-defined protocols or one of the preprogrammed protocols. This is accomplished by the transmission of an API
30 message from the Application Layer 5 to the Call Management Layer 4, which in turn issues an appropriate message to Layer 3 which may also be configured in accordance

with the API of the present invention. The assigned state/event table pointer and assigned primitive table pointer point to the protocol which was last assigned. Thus, a customer may assign a desired one of the available protocols by simply specifying the appropriate pointers in each data block. In this fashion, the present invention
5 advantageously permits the customer to assign, on a channel-by-channel basis, a desired protocol from among multiple protocols resident within a single switch.

Alternatively, or if the customer elects not to assign protocols to some or all of the channels, default values are preferably provided so that each channel always has a valid protocol (e.g., one of the resident protocols 442a-442c) assigned to it.

10 The active state/event table and active primitive table pointers, which are provided to the state machine engine 448, point to the protocol which is currently controlling the channel.

The protocol assigned to a particular channel is not necessarily permanent and may be dynamically changed in real time in response to the occurrence of a specified
15 event, as described in detail in connection with Figure 7. Further, because the atomic functions provided by the library 446 represent elementary functions, customers or users are advantageously able to implement desired changes in protocols without substantial, or possibly any, changes to the underlying code. In addition, the environment support utilities are provided to simplify protocol development for the
20 customer or user. The utilities provide ready-to-use resource management functions (e.g., timers) which greatly simplify the state machine logic required to implement desired protocols. Different utilities are preferably provided for each software layer since the resources required by each layer may be different.

In accordance with the present invention, call processing control
25 communications between the Application Layer 5, typically residing on external host 130, and the other layers of switch 102 illustrated in Figure 2, is conducted through the transfer of generic messages of the universal API of the present invention. Specifically, in the preferred embodiment of the universal API of the present invention, a single message type, referred to as the PPL Event Request message, is
30 used to transfer all call control processing commands and data from the host application (Layer 5) to the telecommunications switch (all other software layers).

Likewise, a single message, referred to as the PPL Event Indication message, is used to transfer all call control processing status and data from the telecommunications switch to the host applications. These generic API messages have optional fields and are the only messages necessary to maintain call processing regardless of the application requirements, network signalling protocol requirements, or features presently existing or to be added to the telecommunications switch. The programmable switch of the present invention enables a user to define and assign a desired applications program interface protocol, either "standard" or custom in nature, for performing various switching functions to accommodate any of the above requirements.

Referring to Figure 5, a PPL Event Request message is sent from the host to the switch to initiate a host event on a PPL component with optional ICB data. The PPL Event Request message is the only call control processing message passed from the host to the switch and, in the preferred embodiment, having the format of message 500 illustrated in Figure 5. The PPL Event Request message comprises a number of fields and subfields, each of which is described below.

PPL Event Request message includes a frame byte 502 having a constant value identifying it as the first byte of a frame.

Message length field 504 contains the length of the particular PPL Event Request message. This is necessary due to the ability of the generic API messages of the present invention to include optional fields, changing the length of the message. Typically the length field value does not include the frame byte 502.

A message type field 506 contains a constant value identifying the particular message as a PPL Event Request message. The message type field is constant for all PPL Event Request messages.

Sequence number field 508 is a specific numeric identifier assigned to each PPL Event Request message that is generated by the host application. This value is used to distinguish between different PPL Event Request messages transmitted from the host to the switch. For example, when the host acknowledges the receipt of a PPL Event Request message, it includes the sequence number in its acknowledgement to identify

which of the PPL Event Request messages is associated with the status information contained within the acknowledgement.

As noted, every PPL component state machine in a switch is assigned a unique reference number. PPL component ID 510 is a one word field that identifies which
5 PPL component implemented in the switch is referenced by a particular PPL Event Request message 500.

There may be multiple instantiations of a PPL component state machine in a switch at any given time. For example, in the preferred embodiment there is an E1
10 PPL component state machine assigned to each channel. Thus, in the illustrative embodiment wherein a single E1 card supports 256 channels, there may be as many as 256 instantiations of the E1 PPL component state machine, each associated with a distinct channel. In order to selectively provide access to every instantiation of a PPL component, the universal API of the present invention provides the ability to perform multiple levels of addressing. Thus, once the PPL component has been identified in
15 the PPL component ID field 510, an address element field 514 is provided to identify which instantiation(s) of that PPL component state machine is(are) being referenced. As shown in Figure 5, the PPL Event Request message provides the ability to include any number of address element fields 514, and thus may simultaneously communicate with multiple instantiations of a single PPL component state machine. The total
20 number of address element fields 514 included in a PPL Event Request message is provided in address element count field 512.

To accommodate the additional levels of addressing noted above, address element field 514 contains a number of subfields for further identifying which state machine instantiation is to receive the PPL Event Request message. Specifically, an
25 address element type field 516 is provided to reference the hierarchial components of the switch that may contain or be associated with the desired state machine instantiation. In the above example of an E1 PPL state machine instantiation, the address element type field 516 indicates which span and channel the state machine instantiation is associated with. An address information subfield 520 provides specific
30 addresses for each of the hierarchial components indicated in the address element type field 516.

Since the addressing information contained in field 520 varies in accordance with the type of device addressed, the length of the AE field 514 may vary and thus is provided in length subfield 518. It is considered to be apparent to one skilled in the relevant art to use other addressing schemes appropriate for a particular PPL

5 component state machine and switch architecture. It should also be noted that, in the above example, a single state machine instantiation exists for each channel since the PPL component state machine is assigned to each channel individually. However, it is considered to be apparent to one skilled in the relevant art that a particular state machine may be configured to manage any number of channels.

10 The multiple levels of addressing provide the universal API PPL Event Request message with a flexible addressing scheme. This enables a host application, using a single PPL Event Request message, to address a range of state machine instantiations having a common PPL component ID to generate a particular event at all addressed state machines.

15 Each PPL event has a unique ID relative to each PPL component. The PPL event ID field 522 provides the switch with a user-defined PPL event ID that the switch recognizes as being associated with the particular request. The recipient PPL component maps the unique PPL event ID to a PPL event that is unique to that PPL component.

20 Each PPL Event Request message may also contain one or more data fields in the form of information control blocks (ICB)s. ICBs are defined for each PPL component based upon the software layer and the communications protocol supported by that PPL component. Thus, any signalling information may be passed between the host and switch using the generic, programmable messages of the present invention.

25 Also referring to Figure 5, a PPL Event Indication message is sent from the switch to the host by a PPL component to report an event at the ports to the host with optional ICB data. The PPL Event Indication message is the only call control processing message passed from the switch to the host and in the preferred embodiment, has the same format as the PPL Event Request message illustrated in

30 Figure 5. Except as noted below, the fields of the PPL Event Indication message are

identical to and perform the same functions as, the analogous fields of the PPL Event Request message discussed above.

As noted above, there may be multiple instantiations of a PPL component state machine. For the PPL Event Indication message, the address element field(s) indicate
5 which instantiation of a particular state machine is actually invoking the atomic function that generates the PPL event indication message.

In the PPL Event Indication message, the PPL event ID field 522 is a specific value representing the occurrence of a specific event in the switch that results in the PPL Event Indication message being sent from the PPL component state machine. As
10 noted, this is managed with an atomic function that is programmed to send the particular PPL Event Indication message in response to the occurrence of a particular event, the PPL Event ID included in the message being programmed by the user.

It is considered to be obvious to one skilled in the relevant art to configure all transfers of information in the switch using the universal API of the present invention,
15 including all layer-to-layer communications. For example, the exemplary communications described in the above-incorporated U.S. Patent No. 5,426,694 may be replaced with the universal API messages of the present invention.

The PPL components can be layer specific, function specific, interface specific, protocol specific, or channel specific. This enables a host Layer 5 telecommunications
20 application to be as interactive as desired or necessary, accessing each layer of the switch and managing each PPL component regardless of where the component is located. An application can therefore use the universal API interface to manage any PPL component. This provides a consistent and predictable means for managing every PPL component in the switch, regardless of what level of processing is being
25 performed, ranging from, for example, a very detailed signalling analysis, to network signalling, to high-level call routing, to call management connection functions.

In the preferred embodiment of the present invention, the manner in which the data is identified and passed to the host includes the implementation of one or more atomic functions configured to store and retrieve data of a certain type to specific
30 memory locations in conjunction with one or more atomic functions that generate a generic PPL Event Indication providing the host with all previously stored data.

However, as one skilled in the art would find apparent, there are numerous ways in which atomic functions may be configured to pass data in the PPL Event Indication message. For example, a separate atomic function may be implemented to transfer specific types of data in a PPL event indication message.

- 5 The short-hand notation for depicting the PPL Event Request and the PPL Event Indication messages is shown in on the bottom of Figure 5.

Referring to Figure 6, the telecommunications switch responds to the PPL Event Request with a PPL Event Request Response message having the format of message 600. Similarly, the host applications responds to the PPL Event Indication
10 message with a PPL Event Indication Acknowledge message, also having the format illustrated in Figure 6. Generic acknowledgement message 600 includes a frame byte 602, length byte 604, message type 606, and sequence number 608, all of which perform the same function as the corresponding fields in the PPL Event Request message 500. In addition, a status field 610 provides the recipient with message-
15 specific status information. The short-hand notation for depicting the PPL Event Request Acknowledge and the PPL Event Indication Acknowledge messages is shown in Figure 6.

Referring to Figures 7A-8G, two examples of the utilization of the universal API of the present invention to perform interactive voice processing functions are
20 provided below. The first example illustrates a universal API for managing host-to-switch communications when the telecommunications switch is controlled by a highly interactive host application Layer 5 to perform interactive voice announcements. The second example illustrates a universal API for managing host-to-switch communications when the host application Layer 5 has limited interaction with the
25 telecommunications switch to perform the same function. These examples illustrate the ability of the universal API to accommodate various applications requirements.

In the following Figures, a state is depicted as a circle, an atomic function is depicted as a rectangular box, and an event is represented by a word abbreviation located along a path leading out of a state. Information shown in parentheses in an
30 atomic function represents arguments or data that are associated with that function.

Reference numbers are provided below in parentheses when necessary to avoid confusion with other numeric descriptors.

Figures 7A-7B illustrate an example of an application of the present invention in Call Processing Layer 4 with a high level of interaction required by the host application layer 5. In this example, the present invention is used to implement a protocol for providing host application decisionmaking throughout the performance of an interactive voice response to an incoming call.

The protocol begins with the associated channel (channel 1) in normal state NS0, which is the IDLE state 702. Upon the occurrence of the event of layer 3 transmitting to layer 4 a setup message ((50)L4PPLevL3_SETUP_INDICATION)), the atomic function af35 is performed. As noted in its descriptor, the Layer 4 PPL event (L4PPLev) is received from network signalling protocol layer 3 (L3), reporting that it has detected an incoming call (SETUP_INDICATION). The number 50 in the parenthetical preceding the message descriptor is the PPL event ID assigned to that event by the Layer 4 PPL. Thus, when Layer 4 is notified of an incoming call, represented by a PPL event ID of 50, the PPL component state machine in Figure 7A leaves idle state 702 and performs atomic function af35 (704).

Atomic function af35 (704) operates to notify the host application (Layer 5) of the event, assigning to the event a PPL event ID of 1. The host application interprets this PPL event ID (the number 1) as a notification of an incoming call. Referring to Figure 7C, atomic function 35 (704) generates a PPL Event Indication message 701 to notify the host of the incoming call. This PPL Event Indication message has the following format:

PPL Event Ind (L4PPL, ch1, 1)

wherein the PPL component ID indicates the Layer 4 PPL component (L4PPL), the instantiation of the Layer 4 PPL component state machine addressed by this message is the instantiation associated with channel 1 (ch1), and the PPL event ID (1) indicates that an incoming message has been received while the PPL component state machine has been in the idle state NS0.

As shown, the arguments associated with atomic function af35 (704) specify a Layer 4 PPL event ID. Note that more generally, atomic function af35 is a PPL Send Event Indication message atomic function, used whenever a PPL Event Indication message is to be sent to the host Layer 5, each message having arguments for
5 indicating the unique PPL event ID associated with the occurrence of a different event. The host responds with a PPL Event Indication Acknowledge message 703 having the general format:

PPL Event Ind Ack (sequence #, status)

10

wherein the sequence number is the sequence number provided in the PPL Event Indication message 701, and the status indicates the status of the associated PPL event indication message. For purposes of this and the following examples, the PPL Event Indication Acknowledge messages all indicate that the immediately previous PPL Event
15 Indication message was successfully received.

After the telecommunications switch provides the host with notification of an incoming call utilizing the PPL Event Indication message of the present invention, the Layer 4 PPL component state machine enters normal state NS1, which is a WAIT state 706, during which the Layer 4 PPL component waits for the host application to
20 respond to the notification. The host sends a Layer 5 PPL Event Request message 705 (see Figure 7C) with an event ID of 1, indicating that it is requesting that the switch proceed with the call received on channel 1. Message 705 has the following format:

PPL Event Req (L4PPL, ch1, 1)

25

indicating that the switch is to respond to the incoming call received on channel 1. A PPL Event Request message having a PPL Event ID of 1 is interpreted by the Layer 4 PPL component state machine as an event. As illustrated in Figure 7A, the receipt of this PPL Event Request message is assigned by the Layer 4 PPL component state
30 machine a Layer 4 unique PPL event ID of 501, indicating that a Layer 5 PPL Event Request (with a PPL Event ID of 1) has occurred.

In response to PPL event 501, the Layer 4 PPL component state machine performs 5 atomic functions: atomic function af60 (708), atomic function af62 (710), atomic function af140 (712), atomic function af212 (714) and atomic function af50 (716). Atomic function af60 is an atomic function that generates a generic PPL Event Request Acknowledge message in accordance with the present invention. This atomic function af60 may be used whenever a PPL component of the switch is to acknowledge the receipt of a PPL Event Request message. The argument number 16 represents an acknowledgement status that the PPL Event Request message was successfully received. This atomic function af60 (708) generates the PPL Event Request Acknowledge message 707, having the same general format as the PPL Event Indication Acknowledge message described above. The PPL Event Request Acknowledge message 707 indicates that the above incoming message has been successfully received.

Atomic functions af62 (710) and atomic function af140 (712), serve, respectively, to send a connect message (to answer the call) to Layer 3 and to send a message to allocate a DSP resource for interactive digit string collection. Atomic function af212 (714) plays an opening announcement to the caller. As shown, there are two arguments to af212: an announcement ID and an announcement control option. af212 (714) was selected to play an opening announcement. Atomic function af50 (716) is performed to set a timer to wait for a selected period of time for receipt of incoming digits, entered in response to the announcement played by atomic function af212 (714). As shown, the first argument indicates that the multi-purpose timer to be used for performing this function is timer1. The second argument indicates the number of 1000 ms units the timer is to count. Here, the second argument is 10, indicating that timer1 will count for 10 seconds. Thus, atomic function af50 (716) enables timer1 to expire in 10 seconds, during which time the Layer 4 PPL component enters normal state S2, which is WAIT state 718, wherein the PPL state machine waits for a digit to be detected on channel 1.

If the next PPL event is the expiration of timer1 ((191) PPLevTIMER1) indicating that no digits were received with the 10 second period, atomic function af35 (720) is performed to inform the host that no digits were received. The receipt of a

PPL_{ev}TIMER1 message is assigned a PPL event ID of 191 by Layer 4 PPL component. As noted, atomic function 35 generates the PPL Event Indication message of the present invention. Here, however, a PPL Event ID of 4 is included in the PPL Event Indication message, indicating that there has been a failure to detect digits on the selected channel. The format of this PPL Event Indication message is, therefore:

PPL Event Ind (L4PPL, ch1, 4)

indicating that no digits have been detected on the channel associated with the channel 1 instantiation of the Layer 4 PPL component while the Layer 4 PPL component was in the wait state 718.

Since the interface diagram of Figure 7C illustrates the universal API protocol associated with the successful receipt of digits and subsequent processing of the call announcement, this message is not illustrated in Figure 7C. Processing then continues with a series of atomic functions not shown where, under host application control through the implementation of the universal API of the present invention, the telecommunications switch performs various functions in response to the failure to detect digits on channel 1.

If the next event to occur while the Layer 4 PPL component state machine is in wait state 718 is the receipt of a message indicating that digits have been detected on channel 1, ((66)L4PPL_{ev}DSP_RESULT_DIGITS), atomic function af47 (722) is performed. The receipt of this message is assigned a unique PPL Event ID of 66. Thus, when Layer 4 is notified that digits have been received, the Layer 4 PPL component state machine leaves wait state S2 (718) and performs atomic function af47 (722). Atomic function af47 disables the PPL multipurpose timer (timer1) selected by atomic function af50 (716), as indicated in the argument to the atomic function. Atomic function af53 (724) stores the received digits in a selected general purpose register. Here, af53 (724) stores the received digit in general purpose register 1.

In accordance with the present invention, atomic function af36 (726) is an atomic function configured to send a PPL Event Indication message with the contents of a selected general purpose register (argument 2), with a PPL event ID (argument 1).

Here, atomic function af36 (726) sends the contents of the general purpose register 1 to the Layer 5 host application as a PPL Event Indication message 709 having an Event ID of 2. PPL Event Indication message 709 has the format

5 PPL Event Ind (L4PPL, ch1, 2, digit)

wherein L4PPL is the PPL component ID, ch1 is the address element, 2 is the PPL event ID, and "digit" is the received digit. This PPL Event Indication message exemplifies the capability of the universal API of the present invention to transfer data
10 as well as commands utilizing a single generic PPL Event Indication message format. The host responds with a PPL Event Indication Acknowledgement message 711, indicating that the PPL Event Indication message 709 was successfully received. Atomic function af147 (728) is then performed to cancel digit reception by disconnecting the DTMF receiver from channel 1.

15 The Layer 4 PPL component state machine then enters state S4 which is a WAIT state 730, wherein the Layer 4 PPL component state machine will wait indefinitely for the host application Layer 5 to direct the Layer 4 PPL component state machine how to respond to the digit that was received. As shown in Figure 7B, in the exemplary embodiment, the host application may send the Layer 4 PPL component any
20 one of 3 different responses, having PPL Event ID 504, 505, and 506.

If a PPL Event Request message 713 having a PPL event ID of 4 is received by the Layer 4 PPL component state machine, ((504)PPLevL5_EVENT_REQ_4) the Layer 4 PPL component state machine assigns a Layer 4 unique PPL event ID 504 to it, indicating that Layer 5 provided a PPL Event Request message having a PPL event
25 ID of 4. PPL Event Request message 711 directs the Layer 4 PPL component state machine to play a specific outgoing announcement. The format of PPL Event Request message 713 is:

PPL Event Req (L4PPL, ch1 , 4)

wherein the PPL component ID indicates that the message is directed towards the Layer 4 PPL component (L4PPL), the address element indicates that channel 1 is the channel by which the communication is occurring (ch1), and the PPL event ID (4) indicates that a specific additional outgoing announcement is to be played on the
5 identified channel.

In response to PPL event 504, Layer 4 performs 4 atomic functions: atomic function af60 (732), atomic function af140 (734), atomic function af212 (736) and atomic function af50 (738). Atomic function af60 (732) is an atomic function that generates a generic PPL Event Request Acknowledge message in accordance with the
10 present invention. As noted, atomic function af60 is used whenever a PPL component is to acknowledge the receipt of a PPL Event Request message. The argument number 16 represents an acknowledgement status that the PPL Event Request message was successfully received. This atomic function generates the PPL Event Request Acknowledge message 764, having the general format described above.

15 Atomic function af140 (734) serves to allocate a DSP resource for interactive digit string collection to channel 1. Atomic function af212 (736) plays an additional outgoing announcement on channel 1. An announcement ID of 3 is indicated by the first argument; no options were selected according to the second argument. Atomic function af50 (738) is performed to set timer1 to wait for a 10 seconds for receipt of
20 incoming digits. The PPL component enters state S5 (740) to wait for the incoming digit for the selected period of time.

The performance of atomic function af212 (736) results in the playing of an announcement to the caller. As shown, there are two arguments to af212: the announcement ID and announcement control options. Atomic function af212 (736)
25 was selected to play an outgoing announcement.

Atomic function af50 (738) is performed to set a timer to wait for a selected period of time for receipt of incoming digits, entered in response to the announcement played by atomic function af212 (736). In accordance with the arguments, atomic function af212 (736) sets timer1 to expire in 10 seconds. During this period, the Layer
30 4 PPL component state machine enters state S5, which is a WAIT state 740 wherein the state machine waits for digits to be received.

In this exemplary embodiment, the host may have alternatively responded with either a PPL Event Request having a PPL event ID of 5 or 6 indicating to the Layer 4 PPL Component to perform other functions not shown. In both cases, the PPL component state machine performs atomic function af60 (731,733, respectively),
5 indicating that the respective PPL Event Request message was successfully received.

If the next event is the expiration of timer1((191) PPLevTIMER1), the protocol again performs an atomic function af35. Atomic function af35 (742) is performed to inform the host that no digits were received within the allotted time. As noted, atomic functions af35 are configured to generate a PPL Event Indication message in
10 accordance with the present invention. Here, a PPL Event ID of 4 is included in the PPL Event Indication message by atomic function af35 (742), indicating that there has been a failure to detect digits on the selected channel. The format of this PPL Event Indication message is:

15 PPL Event Ind (L4PPL, ch1, 4)

indicating that for the channel 1 instantiation of the Layer 4 PPL Component, no digits were detected within the allotted time. Since the interface diagram of Figure 7C illustrates the API protocol associated with the successful processing of the call
20 announcement sequence, this message is not illustrated in that Figure.

If the next PPL event is the expiration of timer1 ((191) PPLevTIMER1) indicating that no digits were received within the 10 second period, atomic function af35 (742) is performed to inform the host that no digits were received by generating the PPL Event Indication message of the present invention with a PPL Event ID of 4,
25 the format of which is:

PPL Event Ind (L4PPL, ch1, 4)

For reasons given above, this is not shown in Figure 7C. Processing then continues
30 with a series of atomic functions not shown where, under host ampliation control through the implementation of the universal API of the present invention, the

telecommunications switch performs various functions in response to the failure to detect digits on channel 1.

If the next event to occur while the Layer 4 PPL component state machine is in wait for digit state 740, is the receipt of a message indicating that digits have been
 5 detected on channel 1, ((66)L4PPLvDSP_RESULT_DIGITS), atomic functions af47 (744), af53 (746) and af36(748) are performed to disable the PPL multipurpose timer (timer1) previously selected, storing the received digits in a selected general purpose register, and sending a PPL Event Indication Message 717, respectively.

In accordance with the present invention atomic function af36 (748) is an
 10 atomic function configured to send a PPL Event Indication message with the contents of general purpose register 1 with a PPL event ID of 3. Here, atomic function af36 (748) sends the contents of the general purpose register 1 to the Layer 5 host application as a PPL Event Indication message 717 having an Event ID of 3. PPL Event Indication message 717 has the format:

15

PPL Event Ind (L4PPL, ch1, 3, digit)

wherein L4PPL is the PPL component ID, ch1 is the address element, 3 is the PPL event ID, and "digit" is the received digit. This event ID indicates that the returned
 20 digit is in response to an af212 atomic function playing an outgoing announcement having an announcement ID of 3. The host responds with a PPL Event Indication Acknowledgement message 719, indicating that the PPL Event Indication message 711 was successfully received.

Referring now to Figure 7D, it may be seen that the each sequence of atomic
 25 functions shown in Figures 7A-7B has been defined as a primitive. In effect, each primitive provides a shorthand way to identify a desired sequence of atomic functions to invoke. The table of Figure 7D lists in tabular format the sequence of atomic functions for each primitive.

Figure 7E is a state/event table that defines the relationships between the states,
 30 events and primitives of Figure 7D. In accordance with a preferred embodiment of the present invention, a customer wishing to create the protocol depicted in Figures 7A-

7B, would need only define the tables shown in Figure 76D and 7E. Those tables would then be downloaded to the switch 102 (Figure 1) through a series of messages from the host device.

Referring to Figures 8A-8G, a second example of the universal API of the present invention configured to manage host-to-switch communications is illustrated. IN this example, the host application Layer 5 has limited interaction with the telecommunications switch while performing these functions. Specifically, the telecommunications switch is configured to automatically respond to the digits that are entered. In contrast to the previous example, the Layer 4 PPL state machine includes internal states responsive to prompts internally generated during digit collection. These aspects of the switch replace the atomic functions generating PPL Event Indication messages to the host providing the received digit, and the subsequent wait states wherein the switch waits for the host to supply it with a PL Event Request message.

Figures 8A-8B illustrate an example of an application of the present invention in Call Processing Layer 4 with a limited level of interaction required by the host application Layer 5 to implement a protocol for providing limited host application decisionmaking in the performance of an interactive voice response to an incoming call.

The protocol begins with the associated channel (channel 1) in normal state S0, which is the IDLE state 702. Upon the occurrence of the event of layer 3 transmitting to layer 4 a setup message ((50)L4PPLvL3_SETUP_INDICATION)), the PPL component state machine in Figure 8A leaves idle state 802 and performs atomic function af35 (804).

Atomic function af35 (804) operates to notify the host application (Layer 5) of the event, assigning to the event a PPL event ID of 1. The host application interprets this PPL event ID value of 1 as a notification of an incoming call. Referring to Figure 8C, atomic function 35 (804) generates a PPL Event Indication message 801 to notify the host of the incoming call. This PPL Event Indication message has the same format as PPL Event Indication message 701, and indicates that the channel 1 instantiation of the Layer 4 PPL component state machine addressed by this message received an

incoming message while the Layer 4 PPL component state machine was in the idle state.

The host responds with a PPL Event Indication Acknowledge message 702 having the general format described above, indicating that the previous PL Event
5 Indication message was successfully received.

After the telecommunications switch provides the host with notification of an incoming call utilizing the PPL Event Indication message of the present invention, the Layer 4 PPL component state machine enters normal state S1, which is the WAIT state 706, during which the Layer 4 PPL component waits for the host application to
10 respond to the notification. The host sends a Layer 5 PPL Event Request message 705 with an event ID of 1, indicating that it is requesting that the switch proceed with the call received on channel 1. This message has the format described above, indicating that an incoming message has been received on channel 1 for that instantiation of the Layer 4 PPL component while the Layer 4 PPL component state machine has been in
15 the idle state.

The receipt of this PPL Event Request message is identified by the Layer 4 PPL component state machine as a PPL event and is assigned a Layer 4 unique PPL event ID of 501, indicating that a Layer 5 PPL Event Request (1) has occurred.

In response to PPL event 501, the Layer 4 PPL component state machine
20 performs 5 atomic functions: atomic function af60 (808), atomic function af62 (810), atomic function af140 (812), atomic function af212 (814) and atomic function af50 (816). As noted, atomic function af60 is used whenever a PPL component is to acknowledge the receipt of a PPL Event Request message. The argument number 16 represents an acknowledgement status that the PPL Event Request message was
25 successfully received. This atomic function af60 (808) generates the PPL Event Request Acknowledge message 807, having the same general format as the PPL Event Indication Acknowledge message described above. The PPL Event Request Acknowledge message 807 indicates that the above incoming message has been
successfully received.

30 Atomic functions af62 (810), af140 (812), af212 (814) and af50 (816) perform the same function as the analogous atomic functions described above with reference to

Figure 7A. Atomic function af50 (816) enables timer1 to expire during which time the Layer 4 PPL component enters normal state S2, which is WAIT state 818, wherein the PPL state machine waits for a digit to be detected on channel 1.

As in the above example, if the next PPL event is the expiration of timer1
5 ((191) PPLvTIMER1) indicating that no digits were received within the selected waiting period, atomic function af35 (820) is performed.

If the next event to occur while the Layer 4 PPL component state machine is in wait state 818 is the receipt of a message indicating that digits have been detected on channel 1, ((66)L4PPLvDSP_RESULT_DIGITS), atomic functions af47 (822), af53
10 (824), and af47 (828) are performed. These atomic functions performs similar functions to the analogous atomic functions described above with reference to Figures 7A and 7B. Note that an atomic function analogous to af36 (726) is not invoked. Thus, the received digit is not provided to the host application.

Atomic function af28 (829) is performed to test the value of the digit stored in
15 the general purpose register used in atomic function af63 (824) to store the received digit. The Layer 4 PPL component state machine enters internal state IS3 which is a TEST state 830, wherein the Layer 4 PPL component state machine tests the value of the digit that was received and stored in general purpose register 1. As shown in Figure 7B, in the exemplary embodiment, the tested digit may have any one of 3
20 different values, each generating an internal event having PPL Event ID 200, 201, and 202.

If a PPL Internal Event message having a PPL event ID of 0
((200)PPLvINT_EVENT_0) is provided to the Layer 4 PPL component state machine, the Layer 4 PPL component state machine assigns a Layer 4 unique PPL
25 event ID 200 to it to indicate that the internal event having a PPL event ID of 0 was received.

In response to PPL event 200, the Layer 4 PPL component state machine performs 3 atomic functions af140 (834), af212 (836) and atomic function af50 (838), each of which perform functions similar to the analogous atomic functions described
30 above with reference to Figures 7A and 7B. Note that an atomic function analogous to af60 (732) is not performed since the switch tests the incoming digit itself, and does

not wait for a host generated PPL Event Request message. Therefore, no acknowledgement is required to be generated.

The Layer 4 PPL component state machine then enters normal state NS4, which is a WAIT state 840 wherein the state machine again waits for digits to be received.

5 If no digits are received, the next event is the expiration of timer1((191) PPLvTIMER1), the protocol performs an atomic function af35 (842) to inform the host that no digits were received within the allotted time. Processing then continues with a series of atomic functions not shown where, under host ampliation control through the implementation of the universal API of the present invention, the
10 telecommunications switch performs various functions in response to the failure to detect digits on channel 1.

 If the next event to occur while the Layer 4 PPL component state machine is in wait for digit state 840, is the receipt of a message indicating that digits have been detected on channel 1, ((66)L4PPLvDSP_RESULT_DIGITS), atomic functions af147
15 (844), af53 (846) are performed. These atomic functions perform similar functions to the analogous atomic functions described above with reference to Figures 7A and 7B. Further, atomic function af28 (847) is performed to test the value of the now second received digit stored in general purpose register 1. A function analogous to atomic function af36 (748) is not performed, thereby not providing the host with a PPL Event
20 Indication message.

 Referring now to Figure 8D-8E, it may be seen that the each sequence of atomic functions shown in Figures 8A-8B has been defined as a primitive. In effect, each primitive provides a shorthand way to identify a desired sequence of atomic functions to invoke. The table of Figure 8F lists in tabular format the sequence of
25 atomic functions for each primitive.

 Figure 8F is a state/event table that defines the relationships between the states, events and primitives of Figure 8D. In accordance with a preferred embodiment of the present invention, a customer wishing to create the protocol depicted in Figures 8A-8B, would need only define the tables shown in Figure 8D and 8E. Those tables
30 would then be downloaded to the switch 102 (Figure 1) through a series of messages from the host device.

Referring to Figures 9-11, a Layer 4 PPL component 901 is executed as part of a Layer 4 PPL processor 902. As noted above, there may be multiple instantiations of a PPL component operating simultaneously, each of which has an associated state machine and tables. All instantiations of a the Layer 4 PPL component 901 are
5 executed on a PPL state machine engine 1104. Layer 4 PPL processor 902 also includes a Layer 4 PPL message processor 1102 for receiving and processing internal PPL Event Request messages 1106. Each PPL processor such as Layer 4 PPL processor 902 may contain any number of PPL components. In the exemplary embodiment shown in Figures 9 and 11, the Layer 4 PPL processor 902 contains a
10 single Layer 4 PPL component 901.

Figure 9 illustrates the process flow related to the invocation and execution of an atomic function to create a PPL Event Indication message. For exemplary purposes, Figure 9 illustrates the functions preformed in relation to the processing of atomic function af35 (704) of primitive #1 (750) to create PPL Event Indication
15 message 701. Atomic function af35 (704) is part of the Layer 4 PPL component 901.

Layer 4 PPL processor 902 resides on CPU/matrix card 112 in the illustrative embodiment discussed above, and invokes atomic functions in accordance with the state/event and primitive tables to perform various functions, including the functions discussed above with reference to the Call Management Layer 4 in Figure 2. As
20 noted, Call Management Layer 4 is responsible for performing centralized call processing functions and providing a common interface to Application Layer 5.

In the preferred embodiment of the present invention, the functions performed by the Layer 4 PPL processor 902 are implemented in a publicly available, proprietary operating system referred to as PSOS, available from Integrated Systems, Inc., Santa
25 Clara, California, USA. However, as will become apparent to one skilled in the relevant art, the present invention may be implemented in any commonly known software program and in any software language now or later developed.

Generally, the Layer 4 PPL processor 902 invokes atomic functions that generate internal representations of the PPL Event Indication message. The internal
30 message is passed to a communications processor 906, also residing on the CPU/matrix

card 112, for translation into a PPL Event Indication message of the universal API of the present invention.

When Layer 4 PPL processor 902 executes a primitive of a PPL component state machine, it invokes each of the atomic functions associated with that primitive, as indicated by the primitive table 780 discussed above. In the example discussed above with reference to Figures 7A and 7B, atomic function af35 (704) is the only atomic function included in primitive #1 (750). As noted, atomic function 35 is a PPL Send Event Indication atomic function used whenever a PPL Event Indication is to be sent to the host, each occurrence of the atomic function having a different PPL event ID to indicate the occurrence of a different event.

A number of functions, each of which is described below, are performed by the Layer 4 PPL atomic function af35 (704) to create the PPL Event Indication message 701 for transmission to a Layer 5 host application. Referring to Figure 10, layer 4 PPL processor 902 allocates a message buffer 1050 to "attach" to the internal representation of the PPL Event Indication message 904 generated by the atomic function af35 (704). The message buffer is used by the PSOS operating system to store the necessary information for creation of the PPL Event Indication message 701.

A pointer to that message buffer is obtained through the PSOS operating system when the buffer is allocated.

Once the message buffer is allocated, destination and source ID fields 1052, 1054 are loaded. The contents of these two fields is determined by the relative location of the transmitting and receiving elements. In addition to enabling a PPL component to communicate with Layer 5 applications residing on the host, the universal API of the present invention may be implemented to manage communications between any two instantiations of any PPL components residing in the same or different PPL processors.

When the universal API is utilized to achieve communications between PPL components that are located in, or "owned", by the same PPL processor, then the source and destination ID fields are loaded with the PPL component ID. Otherwise, the source and destination ID fields are loaded with the processor virtual ID, and the message type field 1058 is used by the destination PPL component or application to

direct the message to the appropriate instantiation of the desired PPL component residing in the destination PPL processor. The message type field 1058 contains a unique message type identifier that is associated with a specific PPL component.

The Layer 4 PPL processor 902 also loads a PPL event ID into the associated
5 field 1056 of the message buffer, the event ID identified in the PPL primitive table 780 provided to atomic function af35 (704). In the example illustrated in Figure 7, atomic function af35 (704) indicates the detection of an incoming call during idle state S0, and is assigned a PPL event ID of 1.

An ICB count field 1062 is loaded with the number of trailing ICB data fields
10 1064, if any.

The Layer 4 PPL processor 902 transfers the data buffer 1050 to the PPL Event Indication message 904. This is accomplished by invoking a function that attaches the allocated buffer 1050 to the PPL Event Indication message 904 by providing the communications processor 906 with a pointer to the data buffer 1050. The
15 communications processor 906 reformats the API messages of the present invention from the internal representation usable by the PPL processor 902 to the format shown in Figure 5 for transmission to the host application. Communications processor 906 performs well known translating operations typical of message handling communications processors, and is considered to be well known in the art.
20 Communications processor 906 then transmits the PPL Event Indication message 701 to an applications program located in host computer 130 via the API interface 908.

The scheme discussed above with respect to the Layer 4 PPL processor 902 is shown below by the following pseudo-code. It is envisioned that this pseudo-code can be used to generate source code for the present invention in any suitable language, such
25 as C, C++ or PASCAL:

1. allocate psos msg data buffer (ppl_data_buff);
2. psos_msg.ppl_component=L4PPL;
3. psos_msg.event_id=event_id;
4. psos_msg.destination = HOST;
- 30 5. psos_msg.source = L4PPL;
6. psos_msg.timeslot_addr = CHANNEL 1;

7. l4ppl_send_msg (psos_msg, comm_queue);

Figure 11 illustrates the process flow related to the receipt and processing of a PPL Event Request message. For exemplary purposes, Figure 11 illustrates the functions related to the processing of PPL Event Request message 705 received at wait
5 state S1 (706), invoking primitive 2 atomic functions as shown in primitive table 780.

The communications processor 1006 performs the inverse operation of that performed above with respect to the PPL Event Indication message. That is, the communications processor 1006 receives the API version of the PPL Event Request message 705 over API 1008 and translates it into an internal PSOS PPL Event Request
10 message 1106. Communications processor 1006 transmits the PPL Event Request message 1106 to the Layer 4 PPL processor 1002. A Layer 4 PPL message processor 1102 receives and processes the internal PPL Event Request message 1106 and generates a distinct Layer 4 PPL event for the Layer 4 PPL state machine engine 1104.

The Layer 4 PPL message processor 1102 converts the PPL event ID 522 of
15 message 705 into a Layer 4 unique PPL event ID for the Layer 4 PPL state machine 1104 by adding a layer 5 event request base value of 500 to the PPL event ID. Thus, in the exemplary embodiment, the Layer 4 PPL message processor 1102 adds a base value of 500 to the PPL Event Request message PPL event ID of 1 to result in a Layer 4 unique PPL event ID of 501.

20 Once the PPL Event Request message is mapped to a Layer 4 unique PPL event ID, a pointer is derived to the selected channel's PPL data based upon the address element value in the message that contained a logical span and channel ID. In other words, the Layer 4 PPL message processor 1102 converts the logical address into a physical address, i.e., a physical time slot in switch 102. Then the PPL state machine
25 engine is invoked, typically as a function call, with the channel pointer for the PPL component instantiation data block and a pointer to the data block associated with the PPL Event Request message 1106 that was received from the communications processor 1006.

The Layer 4 PPL component state machine engine 1104 processes the Layer 4
30 unique PPL event ID, searching the state/event table 790 for that PPL Event ID for the present state. If a matching event is found, the state machine engine 1104 invokes the

identified primitive in the state/event table, retrieving from primitive table 780 the atomic functions associated with the primitive ID.

Thus, in the illustrative embodiment, the Layer 4 PPL component state machine engine 1104 processes PPL event 501, locating the PPL Event ID of 501 for the present state S1 (706) in the state/event table 790 and invoking the atomic functions associated with primitive #2. The PPL state machine engine 1104 enters the state indicated in the Layer 4 PPL state/event table 790 after processing all the atomic functions associated with primitive 2.

Pseudo-code for Layer 4 PPL message processor 1102 as contemplated by embodiments of the present invention is disclosed below:

1. `ppl_event=psos_msg.ppl_data_buff®ppl_event + ppl_L5_event_req_base`
2. `ppl_chan_ptr = ppl_data [psos_msg.hdr.timeslot_addr]`
3. `ppl_stmch(ppl_chan_ptr, ppl_event, psos.msg)`

15

It should be understood that embodiments of the present invention can be implemented in hardware, software or a combination thereof. In such embodiments, the various components and steps would be implemented in hardware and/or software to perform the functions of the present invention. Any presently available or future developed computer software language and/or hardware components can be employed in such embodiments of the present invention. In particular, the pseudo-code discussed above can be especially useful for creating the software embodiments.

While the invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention. Furthermore, the terms and expressions which have been employed are used as terms of description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding any equivalents of the features shown and described or portions thereof, but it is recognized that various modifications are possible within the scope of the invention claimed.

CLAIMS

- 1 1. A telecommunications system, comprising:
2 a host device;
3 a programmable telecommunication switch, connected in communicating
4 relationship with and responsive to said host device, for performing call processing
5 functions related to communication paths established between various ones of a
6 plurality of channels; and
7 a universal applications program interface (API) having standardized messages
8 for communication between said telecommunications switch and said host device.
- 1 2. The system of claim 1, wherein said universal API comprises:
2 a single message type, referred to as a programmable protocol language (PPL)
3 event request message, for transferring all call control processing commands and data
4 from said host device to said telecommunications switch; and
5 a single message type, referred to as the PPL event indication message, for
6 transferring all call control processing status and data from said telecommunications
7 switch to said host device.
- 1 3. The system as in claim 2, wherein said telecommunications switch further
2 comprises one or more instantiations of one or more finite state machines, each of said
3 one or more finite state machines representing one of said one or more protocols and
4 comprising,
5 one or more libraries each containing one or more predetermined functions;
6 one or more predetermined logical states;
7 at least one predetermined event associated with each said one or more
8 predetermined logical states, each said at least one predetermined event uniquely
9 identified relative to each said one or more PPL component state machines; and
10 wherein upon an occurrence of one of said one or more predetermined events, a
11 predetermined primitive associated with the occurring event is invoked, said

12 predetermined primitive comprising a predetermined series of one or more said
13 predetermined functions.

1 4. The system as in claim 3, wherein
2 said switch further comprises one or more state machine engines,
3 wherein each of said one or more state machines is configured to be interpreted
4 by one or more state machine engines.

1 5. The system of claim 3, wherein said PPL event request message comprises:
2 a PPL component ID for identifying which of said one or more state machines
3 is referenced by a particular PPL event request message;
4 one or more address elements, each identifying one of said one or more
5 instantiations of said state machine identified in said PPL component ID field; and
6 a user-defined PPL event ID representing an associated one of said at least one
7 predetermined events associated with the particular state machine identified in said PPL
8 component ID field.

1 6. The system of claim 5, wherein said PPL event request message further
2 comprises:
3 an address element type associated with said address element, for referencing
4 components of said telecommunications switch associated with said state machine
5 instantiation identified by said associated address element; and
6 a component address of each of said telecommunications switch components
7 identified by said address element type.

1 7. The system of claim 5, wherein said PPL event request message further
2 comprises:
3 one or more data fields associated with each of said one or more state
4 machines, for transferring call control processing information from said host device to
5 said telecommunications switch.

1 8. The system of claim 3, wherein said PPL event indication message comprises:
2 a PPL component ID for identifying which of said one or more state machines
3 is referenced by a particular PPL event indication message;
4 one or more address elements, each identifying one of said one or more
5 instantiations of said state machine has invoked the function that generates said PPL
6 event indication message; and
7 a user-defined PPL event ID representing an occurrence of a specific one of
8 said at least one event in said telecommunications switch that results in a particular
9 PPL event indication message being generated by said state machine.

1 9. The system of claim 8, wherein each said PPL event indication message is
2 generated by one of said one or more functions configured to send a PPL event
3 indication message in response to the occurrence of a particular event, and wherein
4 each said one or more address element fields comprises:
5 an address element type referencing PPL components of said
6 telecommunications switch associated with said state machine instantiation identified by
7 said one or more address elements; and
8 a PPL component address of each of said PPL components identified by said
9 address element type.

1 10. The system of claim 8, wherein said PPL event indication message further
2 comprises:
3 one or more data fields associated with each of said one or more state
4 machines, for transferring call control processing information from said host device to
5 said telecommunications switch.

1 11. A telecommunications system, comprising:
2 a host device;
3 a programmable telecommunication switch, connected in communicating
4 relationship with and responsive to said host device, for performing call processing

5 functions related to communication paths established between various ones of a
6 plurality of channels; and
7 means for effecting communications between said switch and said host using a
8 programmable universal applications program interface (API) including standardized
9 messages for transmitting information between said host and said switch.

1 12. The system of claim 11, wherein said API is configured to be customized to
2 meet telecommunications application and network signaling protocol requirements.

1 13. The system of claim 11, wherein said API comprises:
2 a first predetermined message format for all messages transferring call control
3 processing information from said host device to said telecommunications switch; and
4
5 a second predetermined message format for all messages transferring call
6 control processing information from said telecommunications switch to said host
7 device.

1 14. The system as in claim 13, wherein said telecommunications switch further
2 comprises:
3 one or more programmable protocol language (PPL) component state machines,
4 each of which represents one of said one or more protocols for performing said call
5 processing functions, said one or more state machines responsive to one or more
6 predetermined events associated with said state machine.

1 15. The system as in claim 14, wherein
2 said switch further comprises one or more state machine engines, and wherein
3 each said one or more finite state machines comprises one or more libraries of
4 predetermined functions,
5 wherein each of said one or more finite state machines is configured to be
6 interpreted by one of said one or more state machine engines.

1 16. The system as in claim 15, wherein each of said one or more state machines is
2 defined by a functional combination of a state/event table and a primitive table,
3 wherein
4 said state/event table defines one or more predetermined logical states and at
5 least one of said one or more predetermined events associated with each said one or
6 more predetermined logical states, and wherein
7 said primitive table defines one or more primitives each of which comprises a
8 predetermined series of one or more said predetermined functions, whereby upon an
9 occurrence of one of said one or more predetermined events, a predetermined primitive
10 associated with the occurring event is invoked.

1 17. The system of claim 14, wherein said first predetermined message format is a
2 variable length PPL event request message comprising:
3 a PPL component ID for identifying which of said one or more PPL component
4 state machines is referenced by a particular PPL event request message.

1 18. The system of claim 14, wherein said telecommunications switch further
2 comprises:
3 one or more instantiations of each of said one or more PPL component state
4 machines.

1 19. The system of claim 18, wherein said first message format is a variable length
2 PPL event request message comprises:
3 a PPL component ID identifying which of said one or more PPL component
4 state machines is referenced by a particular PPL event request message; and
5 one or more address elements, each identifying one of said one or more
6 instantiations of said PPL component state machine identified by said PPL component
7 ID.

1 20. The system of claim 19, wherein each said one or more address elements
2 comprises:

3 an address element type for referencing components of said telecommunications
4 switch associated with said state machine instantiation identified by said one or more
5 address elements; and
6 PPL component address information providing specific addresses for each of
7 said PPL components identified by said one or more address elements.

1 21. The switch as in claim 18, wherein said PPL event request message comprises a
2 programmable variable addressing scheme to address a desired one or more of said at
3 least one instantiation of each of said plurality of PPL component state machines.

1 22. The system of claim 19, wherein,
2 each said one or more events is uniquely identified relative to each said one or
3 more PPL component state machines, and further wherein,
4 said PPL event request message further comprises a PPL event ID field
5 containing a user-defined event ID representing an associated event unique to a
6 particular PPL component state machine identified in said PPL component ID field.

1 23. The system of claim 22, wherein said PPL event request message further
2 comprises:
3 one or more data fields associated with each of said one or more PPL
4 component state machines, for transferring call control processing information from
5 said host to said switch.

1 24. The system of claim 14, wherein said second predetermined message format is
2 a variable length PPL event indication message comprises:
3 a PPL component ID for identifying which of said one or more PPL component
4 state machines is referenced by a particular PPL event indication message.

1 25. The system as in claim 18, wherein said PPL event indication message
2 comprises a programmable variable addressing scheme to address a desired one or

3 more of said at least one instantiation of each of said plurality of PPL component state
4 machines.

1 26. The system of claim 18, wherein said first message format is a variable length
2 PPL event indication message comprises:

3 a PPL component ID identifying which of said one or more PPL component
4 state machines is referenced by a particular PPL event indication message; and
5 one or more address elements, each identifying one of said one or more
6 instantiations of said PPL component state machine has invoking the function that
7 generates said PPL event indication message.

1 27. The system of claim 26, wherein said PPL event indication message is
2 generated by one of said one or more functions configured to send the particular PPL
3 event indication message in response to the occurrence of a particular event, and
4 wherein each said one or more address element fields comprises:

5 an address element type subfield for referencing components of said switch
6 associated with said state machine instantiation identified in said one or more address
7 element fields; and

8 an address information subfield provides specific addresses for each of the
9 hierarchical components indicated in said address element type field.

1 28. The system of claim 26, wherein,

2 each said one or more events is uniquely identified relative to each said one or
3 more PPL component state machines, and further wherein,

4 said PPL event indication message further comprises a PPL event ID field
5 containing a user-defined event ID representing the occurrence of a specific event in
6 said switch that results in a particular PPL event indication message being generated by
7 said PPL component state machine.

1 29. The system of claim 23, wherein said PPL event indication message further
2 comprises:

3 one or more data fields associated with each of said one or more PPL
4 component state machines, for transferring call control processing information from
5 said host to said switch.

1 30. The system of claim 11, wherein said call processing functions include
2 dynamically connecting or disconnecting communication paths between various ones of
3 a plurality of channels.

1 31. A functionally-layered programmable telecommunication switch comprising:
2 controllable-switching means for dynamically connecting or disconnecting
3 communication paths between various ones of a plurality of channels in response to
4 messages generated by a telecommunications services application;
5 one or more instantiations of a plurality of programmable protocol language
6 (PPL) component state machines, each of which is associated with a PPL component
7 of said telecommunications switch and each of which represents one of a plurality of
8 protocols configured to perform call processing functions with respect said plurality of
9 channels, wherein said plurality of PPL component state machines are functionally
10 associated with the functional layers of the telecommunications switch including said
11 PPL components; and
12 a programmable universal applications program interface (API) for transferring
13 standardized messages between said functional layers and between said functional
14 layers and said telecommunications services application.

1 32. The telecommunications switch of claim 31, wherein said functional layers of
2 said telecommunication switch comprise:
3 an application layer comprising said telecommunications service applications
4 configured to operate in conjunction with one or more of said functional layers to
5 perform telecommunications service functions.

1 33. The telecommunications switch of claim 32, wherein said application layer
2 resides in said telecommunications switch and further wherein said universal API

3 defines communications that occur over a bus internal to said telecommunications
4 switch.

1 34. The telecommunications switch of claim 32, wherein said functional layers
2 further include an application layer residing in a host computer system coupled to said
3 telecommunications switch, and wherein said API defines communications that occur
4 over a communication channel coupling said telecommunications switch and said host
5 device.

1 35. The telecommunications switch of claim 32, wherein said telecommunications
2 services provided by said telecommunication service application includes toll-free
3 service functions.

1 36. The telecommunications switch of claim 32, wherein said telecommunications
2 service provided by said telecommunication service application includes voice mail
3 service functions.

1 37. The telecommunications switch of claim 32, wherein said telecommunications
2 service provided by said telecommunication service application includes automatic call
3 distribution service functions.

1 38. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes interactive
3 voice-response functions.

1 39. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes personal
3 communications services functions.

1 40. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes tone
3 generation functions.

1 41. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes call
3 conferencing functions.

1 42. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes call
3 management functions.

1 43. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes call
3 progress tone control functions.

1 44. The telecommunications switch of claim 32, wherein said call processing
2 function performed by said telecommunication service application includes inbound
3 call routing and queuing functions.

1 45. The telecommunications switch of claim 31, wherein said functional layers
2 comprise:
3 a call management layer for performing centralized call processing functions.

1 46. The telecommunications switch of claim 45, wherein said centralized call
2 processing functions performed by said call management layer includes recorded
3 announcement control functions for interactive voice response application support.

1 47. The telecommunications switch of claim 45, wherein said centralized call
2 processing functions performed by said call management layer includes reconnection
3 and transfer functions.

1 48. The telecommunications switch of claim 45, wherein said centralized call
2 processing functions performed by said call management layer includes the provision
3 of multiple call management features.

1 49. The telecommunications switch of claim 45, wherein said centralized call
2 processing functions performed by said call management layer includes conferencing
3 connection management functions.

1 50. The telecommunications switch of claim 31, wherein said functional layers
2 comprise:
3 a network signaling protocol layer for performing network signaling functions.

1 51. The telecommunications switch of claim 50, wherein said network signally
2 functions performed by said network signalling protocol layer includes in- and out-of-
3 band network signalling supervision.

1 52. The telecommunications switch of claim 50, wherein said network signalling
2 functions performed by said network signalling protocol layer includes network
3 protocol level control of incoming and outgoing calls.

1 53. The telecommunications switch of claim 31, wherein said functional layers
2 comprise:
3 a link layer for detecting and transferring network signaling information across
4 a network or line interface.

1 54. The telecommunications switch of claim 53, wherein said link layer runs on a
2 CPU/matrix card.

1 55. The telecommunications switch of claim 53, wherein said link layer runs on
2 line cards.

1 56. The telecommunications switch of claim 53, wherein said functions performed
2 by said link layer include T1 robbed bit signal scanning.

1 57. The telecommunications switch of claim 53, wherein said functions performed
2 by said link layer include E1 channel associated signaling scanning.

1 58. The telecommunications switch of claim 53, wherein said functions performed
2 by said link layer include T1/E1 line interface frame alarm control.

1 59. The telecommunications switch of claim 53, wherein said functions performed
2 by said link layer include DSP tone generation control.

1 60. The telecommunications switch of claim 53, wherein said functions performed
2 by said link layer include DSP recorded voice announcement control.

1 61. The telecommunications switch of claim 31, said functional layers comprise:
2 a physical layer implemented in line cards providing physical and electrical
3 network and line interfaces to the switch.

1 62. The system of claim 31, wherein said API comprises:
2 predetermined message formats for all messages transferring call control
3 processing commands, status and data between said functional layers.

1 63. The switch as in claim 31, wherein each said one or more PPL component state
2 machines comprises,
3 one or more libraries each containing one or more predetermined functions;
4 one or more predetermined logical states;
5 at least one predetermined event associated with each said one or more
6 predetermined logical states, each said at least one predetermined event uniquely
7 identified relative to each said one or more PPL component state machines; and

8 wherein upon an occurrence of one of said one or more predetermined events, a
9 predetermined primitive associated with the occurring event is invoked, said primitive
10 comprising a predetermined series of one or more said predetermined functions.

1 64. The switch as in claim 63, wherein
2 said switch further comprises one or more state machine engines, and wherein
3 each said one or more component state machines comprises one or more
4 libraries of predetermined functions,
5 wherein said each said one or more component state machines is configured to
6 be interpreted by one or more state machine engines.

1 65. The switch as in claim 63, each said one or more finite state machines
2 comprising,
3 one or more libraries each containing one or more predetermined functions;
4 one or more predetermined logical states;
5 at least one predetermined event associated with each said one or more
6 predetermined logical states, each said at least one predetermined event uniquely
7 identified relative to each said one or more PPL component state machines; and
8 a predetermined series of one or more said predetermined functions,
9 wherein upon an occurrence of one of said one or more predetermined events, a
10 predetermined primitive associated with the occurring event is invoked.

1 66. The switch as in claim 65, wherein each of said one or more state machines is
2 defined by a functional combination of a state/event table and a primitive table,
3 wherein
4 said state/event table defines one or more predetermined logical states and at
5 least one of said one or more predetermined events associated with each said one or
6 more predetermined logical states, and wherein
7 said primitive table defines one or more primitives each of which comprises a
8 predetermined series of one or more said predetermined functions.

1 67. The switch as in claim 65, wherein
2 said switch further comprises one or more state machine engines, and wherein
3 wherein said each said one or more finite state machines is configured to be
4 interpreted by one or more state machine engines.

1 68. The system of claim 65, wherein said PPL event request message comprises:
2 a PPL component ID for identifying which of said one or more PPL component
3 state machines is referenced by a particular PPL event request message;
4 one or more address element fields, each identifying one of said one or more
5 instantiations of said PPL component state machine identified in said PPL component
6 ID field; and
7 a PPL event ID field containing a user-defined event ID representing an
8 associated event unique to a particular PPL component state machine identified in said
9 PPL component ID field.

1 69. The system of claim 68, wherein each said one or more address element fields
2 comprises:
3 an address element type subfield for referencing components of said switch
4 associated with said state machine instantiation identified in said one or more address
5 element fields; and
6 an address information subfield provides specific addresses for each of the
7 hierarchical components indicated in said address element type field.

1 70. The system of claim 68, wherein said PPL event request message further
2 comprises:
3 one or more data fields associated with each of said one or more PPL
4 component state machines, for transferring call control processing information from
5 said host to said switch.

1 71. The system of claim 65, wherein said PPL event indication message comprises:

2 a PPL component ID for identifying which of said one or more PPL component
3 state machines is referenced by a particular PPL event indication message;

4 a PPL component ID field for identifying which of said one or more PPL
5 component state machines is referenced by a particular PPL event indication message;

6 one or more address element fields, each identifying one of said one or more
7 instantiations of said PPL component state machine has invoking the function that
8 generates said PPL event indication message; and

9 a PPL event ID field containing a user-defined event ID representing the
10 occurrence of a specific event in said switch that results in a particular PPL event
11 indication message being generated by said PPL component state machine.

1 72. The system of claim 71, wherein said PPL event indication message is
2 generated by one of said one or more functions configured to send the particular PPL
3 event indication message in response to the occurrence of a particular event, and
4 wherein each said one or more address element fields comprises:

5 an address element type subfield for referencing components of said switch
6 associated with said state machine instantiation identified in said one or more address
7 element fields; and

8 an address information subfield provides specific addresses for each of the
9 hierarchical components indicated in said address element type field.

1 73. The system of claim 71, wherein said PPL event indication message further
2 comprises:

3 one or more data fields associated with each of said plurality PPL component
4 state machines, for transferring call control processing information between said
5 functional layers, said data blocks defined for each said plurality of PPL components
6 based upon which of said functional layers said PPL component state machine is
7 associated with and the communications protocol supported by that PPL component
8 state machine.

1 74. A universal applications program interface (API) for standardized interactive
2 call processing communications between functional layers of a telecommunications
3 system including a telecommunications switch and a host device coupled to the switch,
4 comprising:
5 a first programmable message for transferring all call control processing
6 commands and data from said host to said functional layers of said telecommunications
7 switch; and
8 a second programmable message for transferring all call control processing
9 status and data from said functional layers of said telecommunications switch to said
10 host.

1 75. The API of claim 74, wherein the telecommunications switch comprises:
2 one or more instantiations of one or more PPL component state machines each
3 of which represents a call processing protocol, each said one or more finite state
4 machines comprising,
5 one or more libraries each containing one or more predetermined functions;
6 one or more predetermined logical states;
7 at least one predetermined event associated with each said one or more
8 predetermined logical states, each said at least one predetermined event uniquely
9 identified relative to each said one or more PPL component state machines; and
10 wherein upon an occurrence of one of said one or more predetermined events, a
11 predetermined primitive associated with the occurring event is invoked, said primitive
12 comprising a predetermined series of one or more said predetermined functions,
13 wherein said one or more predetermined events includes receipt of said first
14 programmable message and wherein at least one of said one or more predetermined
15 functions generates said second programmable message.

1 76. The switch as in claim 75, wherein
2 said telecommunications switch further comprises one or more state machine
3 engines, and wherein

4 each said one or more PPL component state machines comprises one or more
5 libraries of said one or more predetermined functions, and wherein
6 each said one or more PPL component state machines is configured to be
7 interpreted by said one or more state machine engines.

1 77. The switch as in claim 75, wherein said API further comprises:
2 an acknowledge message including a status field providing the recipient with
3 message-specific status information,
4 wherein said host device transmits said acknowledge message to said
5 telecommunications switch upon receipt of said first programmable message and
6 wherein at one of said one or more predetermined functions is configured to transmit
7 said acknowledge message to said host device upon receipt of said second
8 programmable message.

1 78. A method for developing call-associated protocols for performing call
2 processing functions related to communication paths established between various ones
3 of a plurality of channels in a programmable telecommunications switch, said call
4 processing function associated with the functions performed by a particular functional
5 layer of said switch, the method comprising the steps of:
6 (a) creating one or more state/event tables each of which defines,
7 a plurality of predetermined logical states,
8 one or more predetermined events associated with each of said plurality of
9 predetermined logical states, said one or more predetermined events including receipt
10 of one or more application program interface (API) messages generated at the same or
11 different functional layer as said created call-associated protocol, and
12 a primitive associated with each said one or more predetermined events,
13 wherein said primitive is invoked upon an occurrence of said one or more associated
14 events;
15 (b) creating one or more primitive tables each of which defines a
16 predetermined series of predetermined layer-dependent functions for each said

17 primitive, one or more of said predetermined functions generating an API message to
18 said functional layer; and
19 (c) creating one or more protocols each of which is represented by a
20 predetermined association of one or more of said state/event tables and one or more of
21 said one or more primitive tables.

1 79. The method of claim 78, further comprising the steps of:

2 (d) storing said one or more call-associated protocols stored within said
3 programmable telecommunications switch; and

4 (e) executing said one or more call-associated protocols within said
5 telecommunications switch.

1 80. The method as in claim 78, wherein each of said one or more protocols is
2 represented by a finite state machine having access to a comprising a library containing
3 definitions of said predetermined functions, and configured to be interpreted by a state
4 machine engine, said state machine engine operating in response to said pointers.

1 81. A functionally-layered programmable telecommunication switch comprising:
2 a layer-specific processor having a state machine engine configured to execute
3 an instantiation of a PPL component state machine representing a call processing
4 protocol associated with a communications channel in the switch, said state machine
5 invoking one or more predetermined functions in accordance with a current state and
6 the occurrence of a predetermined event,
7 wherein said one or more predetermined functions includes generating a first
8 application program interface (API) message having a first predetermined message
9 format for all messages transferring call control processing information from said state
10 machine; and
11 wherein said predetermined event is one of a plurality of events including the
12 receipt of a second API message having a second predetermined message format for all
13 messages transferring call control processing to said state machine.

1 82. The switch of claim 81, wherein said layer-specific processor is configured to
2 process said first and second API messages in an internally-represented form, and
3 wherein said switch further comprises:
4 a communications processor, coupled to said layer-specific processor,
5 configured to convert between internally-represented API message form and said
6 universal standardized API message form, and further configured to transmit said first
7 universal API message.

1 83. The switch of claim 82, wherein said system includes a host configured to
2 support applications residing in an application layer and further wherein said API
3 messages are transmitted from the switch to the host and vice versa.

1 84. The switch of claim 81, wherein said processor comprises:
2 an atomic function message buffer including a destination identification field
3 and a source identification field containing respective addresses of a source and
4 receiving PPL component instantiation residing in the same or different PPL
5 processors; and
6 means for attaching said message buffer to said internal representation of said
7 PPL event indication message generated by said atomic function.

1 85. The switch of claim 84, wherein said message buffer further comprises:
2 a PPL event identification field identifying the event that generated said atomic
3 function.

1 86. The switch of claim 84, wherein said message buffer further comprises:
2 one or more data fields containing information associated with said event and
3 said atomic function for a receiving instantiation.

1 87. The switch of claim 83, wherein said plurality of PPL component state
2 machines are layer specific.

1 88. The switch of claim 83, wherein said plurality of PPL component state
2 machines are function specific.

1 89. The switch of claim 83, wherein said plurality of PPL component state
2 machines are interface specific.

1 90. The switch of claim 83, wherein said plurality of PPL component state
2 machines are protocol specific.

1 91. The switch of claim 83, wherein said plurality of PPL component state
2 machines are protocol specific.

1 92. A method for communicating between two layers of a functionally-layered
2 programmable telecommunication switch system utilizing a standardized universal
3 application program interface (API), the method comprising the steps of :
4 (1) invoking one or more instantiations of a layer-specific program protocol
5 language (PPL) component state machine at a layer-specific PPL processor having a
6 state machine engine, each of said one or more instantiations representing a call
7 processing protocol;
8 (2) invoking atomic functions in accordance with state/event and primitive
9 tables defining said state machine and stored in the processor to perform various
10 functions, said atomic functions generating internal representations of a API event
11 indication message; and
12 (3) transferring said internally-represented PPL event indication message to
13 a communications processor coupled to said processor for translation into a universal
14 standardized PPL event indication message.

1 93. The method of claim 92, further comprising the step of:
2 (4) transmitting said universal API message to another PPL component state
3 machine instantiation residing in the same or different PPL processor.

1 94. The method of claim 92, wherein said system includes a host supporting
2 applications residing in an application layer and wherein the method further comprises
3 the step of:

4 (4) transmitting said universal API message to an application located on the
5 host.

1 95. The method of claim 93, further comprising the steps of:

2 (5) creating an atomic function message buffer, by said processor, including
3 a destination identification field and a source identification field containing respective
4 addresses of said source and receiving PPL component instantiation; and

5 (6) attaching said message buffer to said internal representation of said PPL
6 event indication message generated by said atomic function.

1 96. The method of claim 95, wherein said message buffer further comprises:

2 a PPL event identification field identifying the event that generated said atomic
3 function.

1 97. The method of claim 95, wherein said message buffer further comprises:

2 one or more data fields containing information associated with said event and
3 said atomic function for a receiving instantiation.

1 98. The method of claim 92, further comprising the steps of:

2 (4) receiving at said communications processor, a universal API PPL event
3 request message;

4 (5) translating at said communications processor, said universal API PPL
5 event request message into an internally-represented PPL event request message;

6 (6) transmitting said internally-represented PPL event request message to
7 said layer-specific PPL processor;

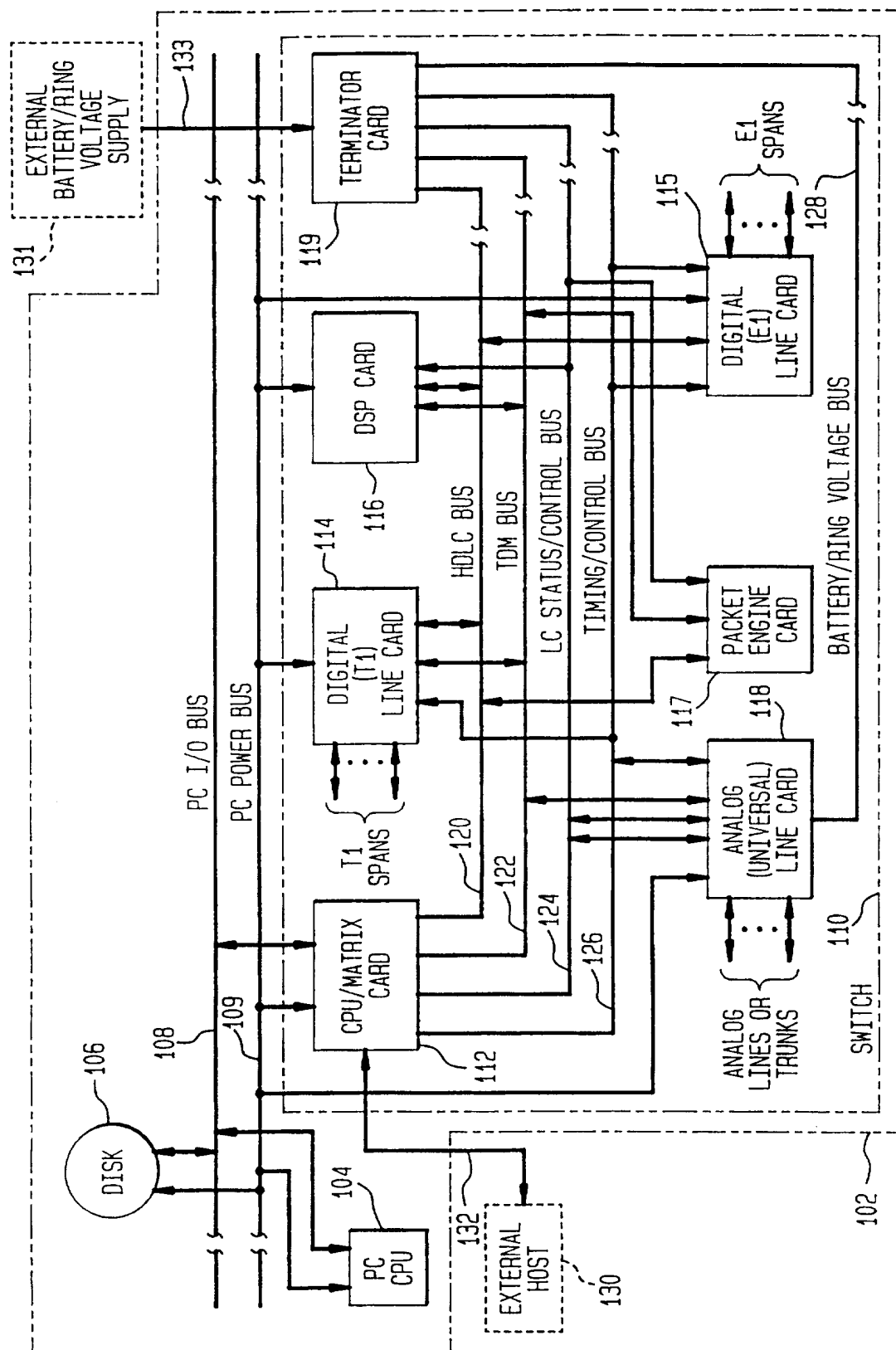
8 (7) receiving and processing, at said layer-specific PPL processor, said
9 internally-represented PPL event request message; and

- 10 (8) converting a PPL event ID value included in said internally-represented
11 PPL event request message into a layer-specific unique PPL event identification value
12 for said layer-specific state machine; and
13 (9) processing the layer-unique PPL event identification value by said layer-
14 specific PPL component state machine engine.

- 1 99. The method of claim 98, wherein said step (9) comprises the steps of:
2 (a) searching said state/event table for that PPL event identification value
3 for the present state of said state machine;
4 (b) when a matching event is found, invoking the identified primitive in the
5 state/event table, retrieving from primitive table the atomic functions associated with
6 the primitive ID;
7 (c) invoking an identified primitive in the state/event table; and
8 (d) retrieving from primitive table the atomic functions associated with the
9 primitive ID.

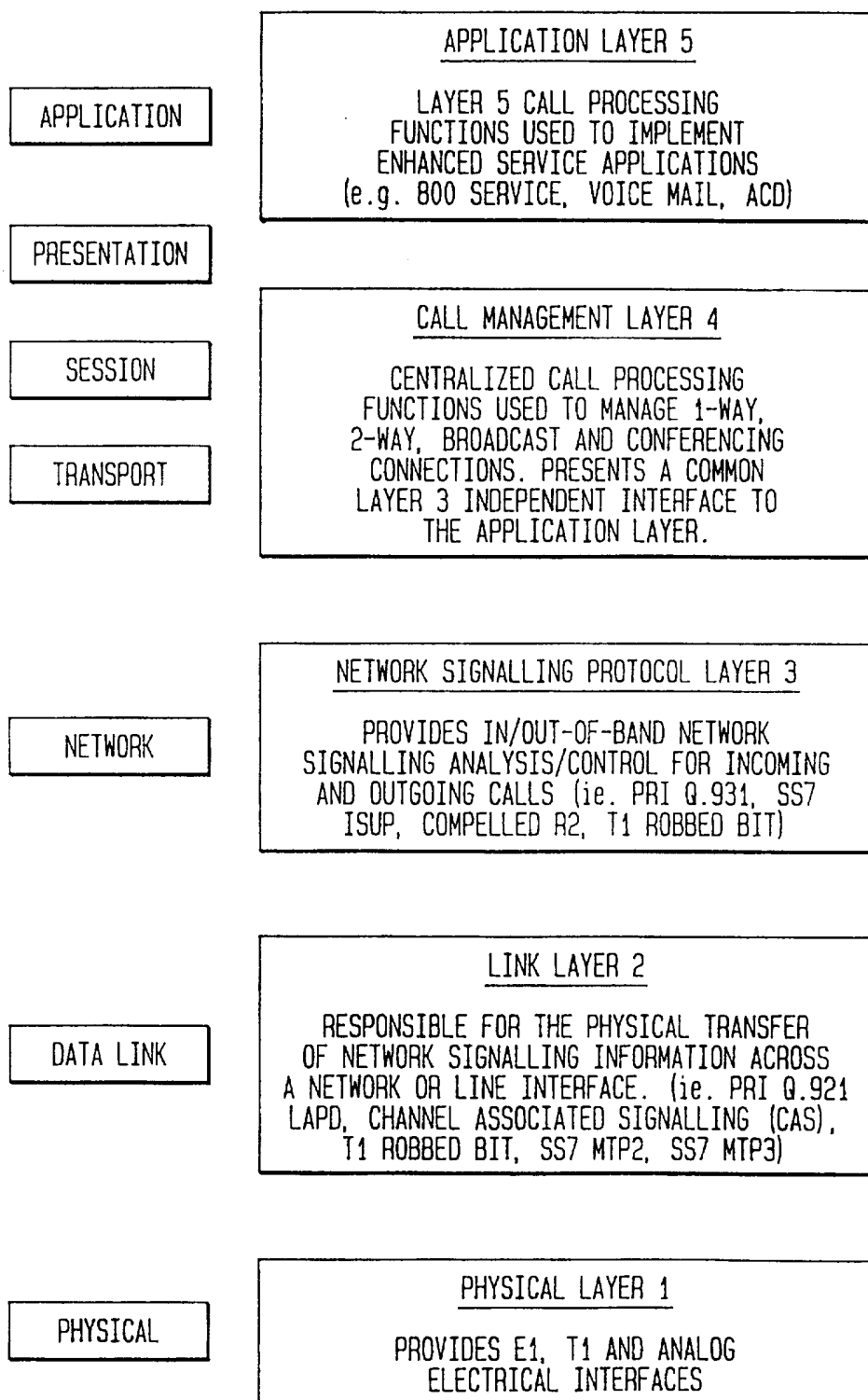
1/24

FIG. 1



2/24

FIG. 2



3/24

FIG. 3A**APPLICATION LAYER (LAYER 5)**

- **MATRIX/LINE CARD MANAGEMENT**
 - DOWNLOAD CONTROL
 - ALARM PROCESSING
 - REDUNDANT MATRIX CONTROL
- **CONFIGURATION MANAGEMENT**
 - MATRIX CONFIGURATION
 - LINE CARD CONFIGURATION
- **HIGH LEVEL (LAYER 5) CALL PROCESSING**
 - INTERACTIVE DIGIT COLLECTION
 - RECORDED ANNOUNCEMENT CONTROL FOR INTERACTIVE VOICE RESPONSE APPLICATION SUPPORT
 - BROADCASTING/CONFERENCING CONTROL
 - INBOUND CALL ROUTING/QUEUEING
 - OUTBOUND CALL INITIATION WITH DIGIT OUTPUTTING
 - ADDRESS DIGIT ROUTING TO CHANNELS/CHANNEL GROUPS
 - CALL HUNTING FOR OUTBOUND CHANNEL SELECTION
 - CALL PROGRESS TONE CONTROL FOR INBOUND/OUTBOUND CALLS
 - MULTIPLE CALL MGMT FEATURES (TRANSFER, HOLD, CONFERENCING, CALLBACK, FORWARDING, ETC.)
 - CALL DETAIL RECORDING

CALL MANAGEMENT LAYER (LAYER 4)

INTERACTIVE RECORDED ANNOUNCEMENT CONTROL (USER DIGIT DRIVEN)
LAYER 4/APPLICATION LAYER (LAYER 5) INITIATED CALL PARK
1-WAY/2-WAY/CONFERENCE CONNECTION MANAGEMENT
RECONNECTION (TRANSFER)
LAYER 4 OUTGOING OUTSEIZE INITIATION FOR 2-WAY CONNECTIONS
MULTIPLE CALL MGMT FEATURES (TRANSFER, HOLD, CONFERENCING, CALLBACK, FORWARDING, ETC.)
CUSTOMIZATION OF INSEIZE COMPLETION REPORT SENT TO THE APPLICATION LAYER (LAYER 5)

NETWORK PROTOCOL LAYER (LAYER 3)

- **IN BAND LINE/ADDRESS SIGNALLING CONTROL**
 - E&M INTERFACE
 - LOOPSTART, GROUNDSTART TRUNK INTERFACES
 - LOOPSTART, GROUNDSTART LINE INTERFACES
 - MULTI-WINK MFR1 FEATURE GROUP D
 - DTMF DIALED NUMBER IDENTIFICATION SERVICES (DNIS)
 - COMPELLED R2 FOR INTERNATIONAL E1 INTERFACES
 - IN BAND/EXTENDED IN BAND/MULTI-WINK COIN SIGNALLING
 - CUSTOM T1/E1 SERVICE CARD INTERFACES
- **OUT OF BAND SIGNALLING CONTROL**
 - ISDN PRIMARY RATE LAYER 3 Q.931
 - SS7 ISDN USER PART (ISUP)

4/24

FIG. 3B**LINK LAYER (LAYER 2)**

- T1 ROBBED BIT SIGNALLING SCANNING
- E1 CHANNEL ASSOCIATED SIGNALLING SCANNING
- T1/E1 LINE INTERFACE FRAME ALARM CONTROL
- DSP TONE RECEPTION CONTROL
 - IN BAND ADDRESS SIGNALLING (MFR1, MFR2, DTMF)
 - CALL PROGRESS ANALYSIS
- DSP TONE GENERATION CONTROL
 - IN BAND ADDRESS SIGNALLING (MFR1, MFR2, DTMF)
 - CALL PROGRESS TONE GENERATION
 - CUSTOM CALL PROGRESS TONE GENERATION
- SS7 MTP2/MTP3
- DSP RECORDED VOICE ANNOUNCEMENT CONTROL
- DSP CONFERENCE GENERATION CONTROL
- GENERIC DSP FUNCTION CONTROL
 - DSP PROCESSOR TO MFDSP MAIN PROCESSOR DSP CONTROL/
INFORMATIONAL MESSAGES
 - MFDSP MAIN PROCESSOR TO DSP PROCESSOR CONTROL/
INFORMATIONAL MESSAGES
 - MFDSP MAIN PROCESSOR DSP FUNCTION ANALYSIS/CONTROL
- LAPD Q.921

5/24

FIG. 4

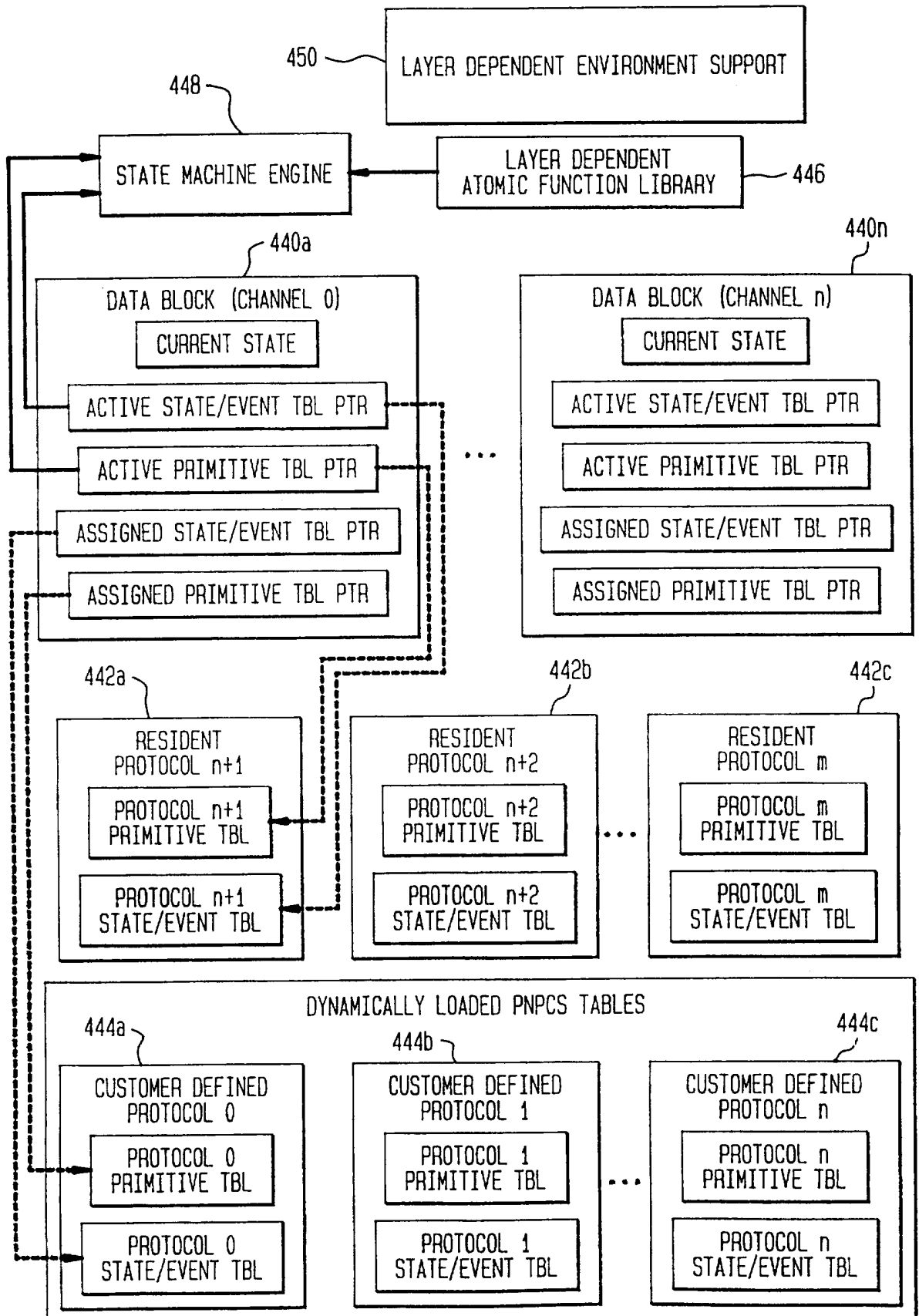
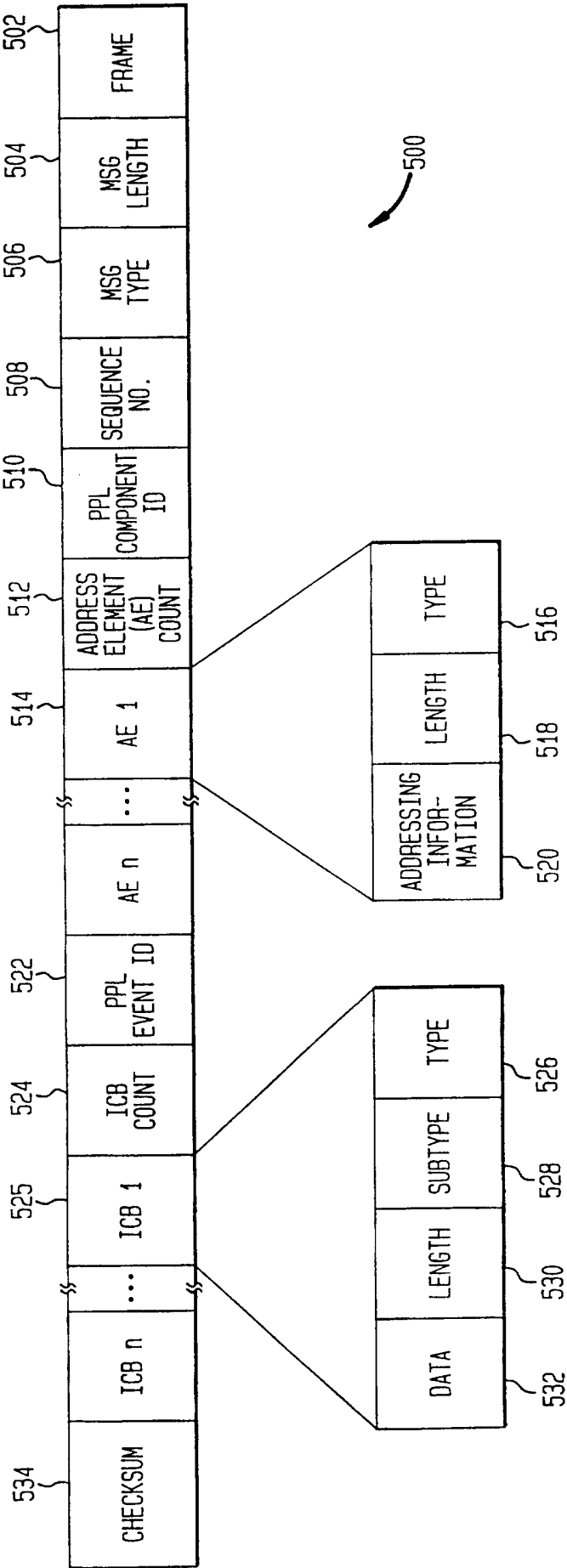


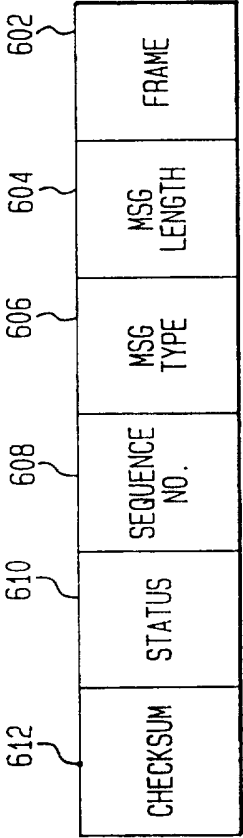
FIG. 5



PPL EVENT REQ (PPL COMP. ID, CHANNEL NO., EVENT ID, DATA)
PPL EVENT IND (PPL COMP. ID, CHANNEL NO., EVENT ID, DATA)

7/24

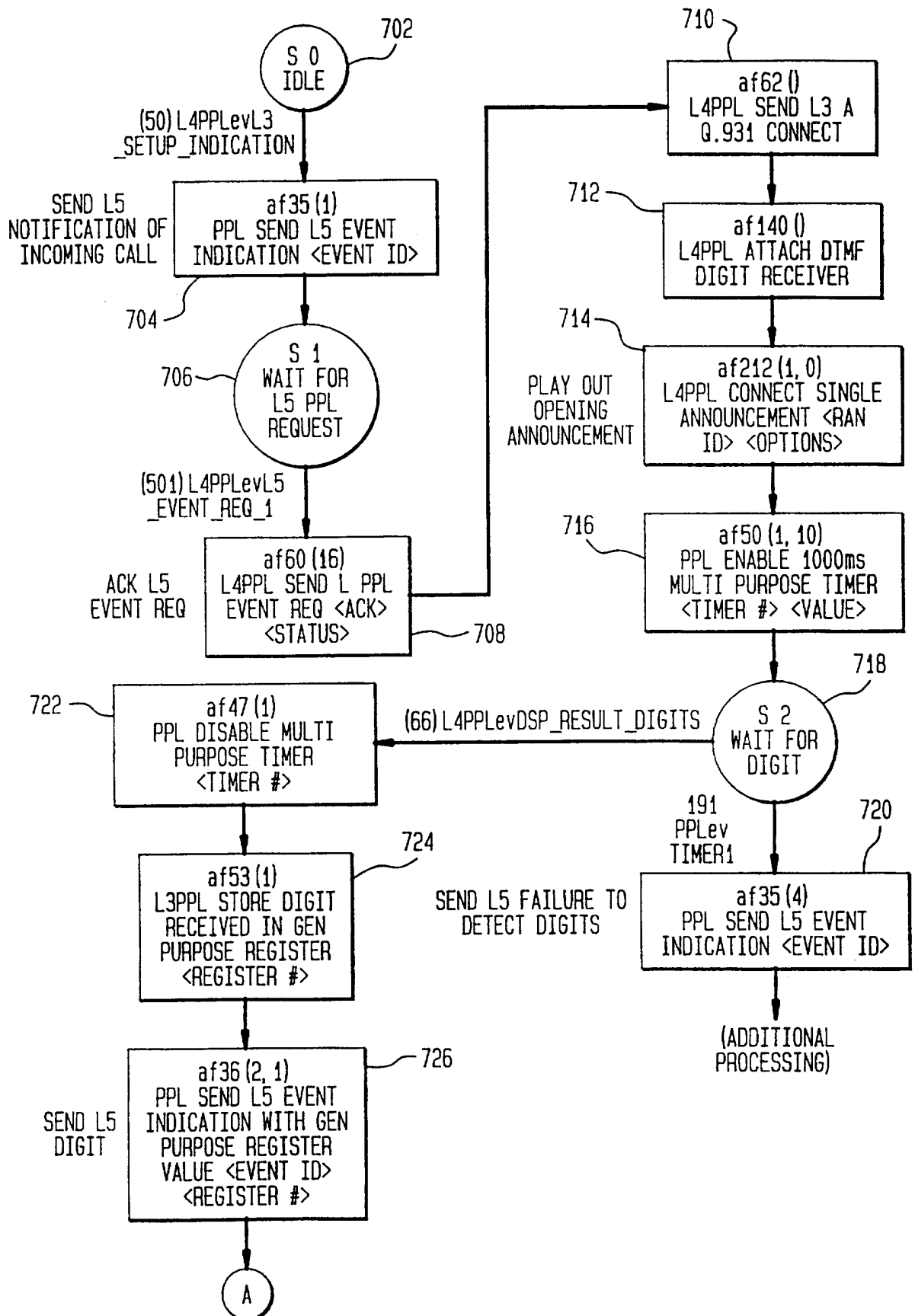
FIG. 6



PPL EVENT REQ ACK (SEQUENCE NO, STATUS)
PPL EVENT IND ACK (SEQUENCE NO, STATUS)

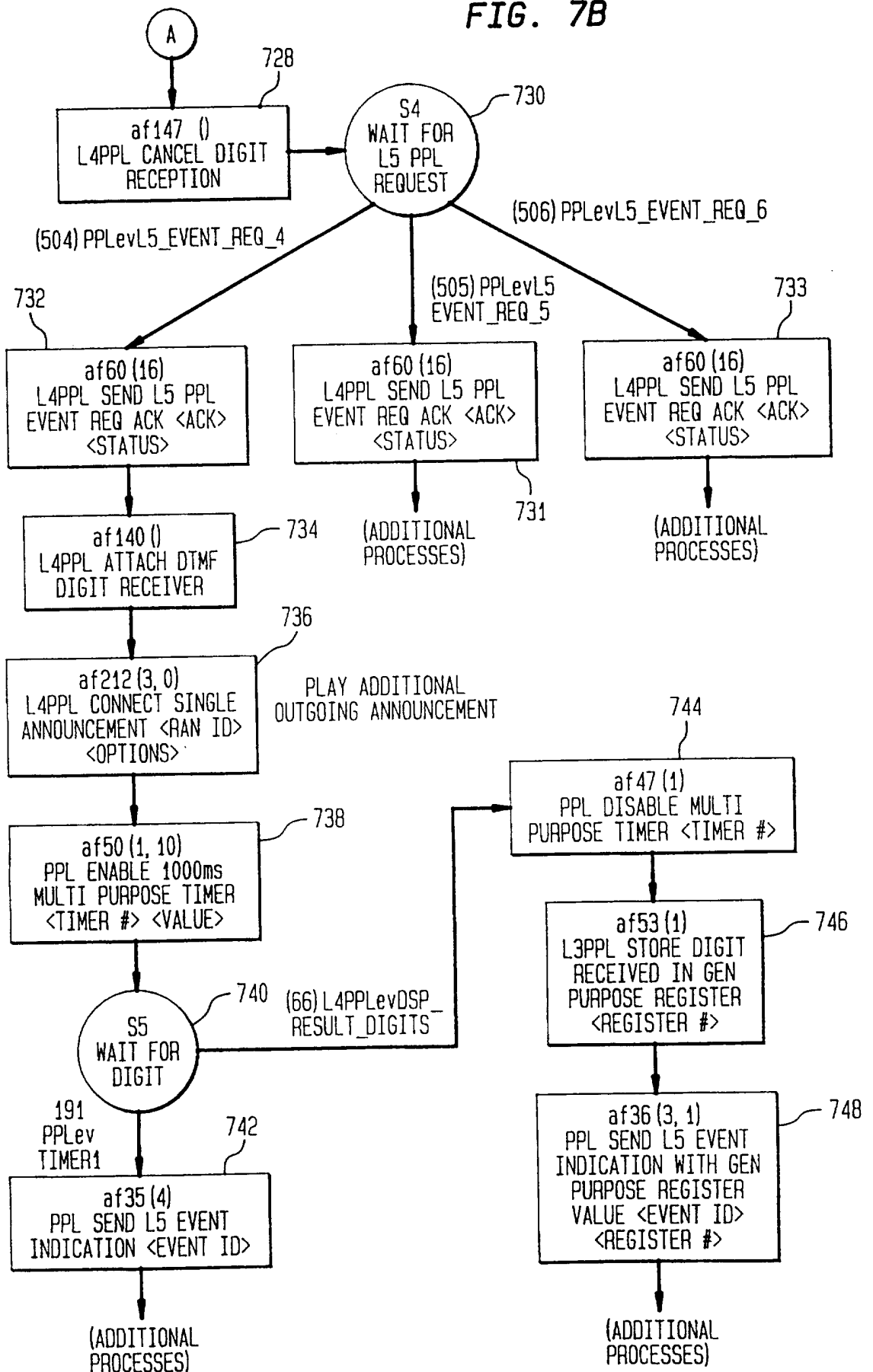
8/24

FIG. 7A



9/24

FIG. 7B



10/24

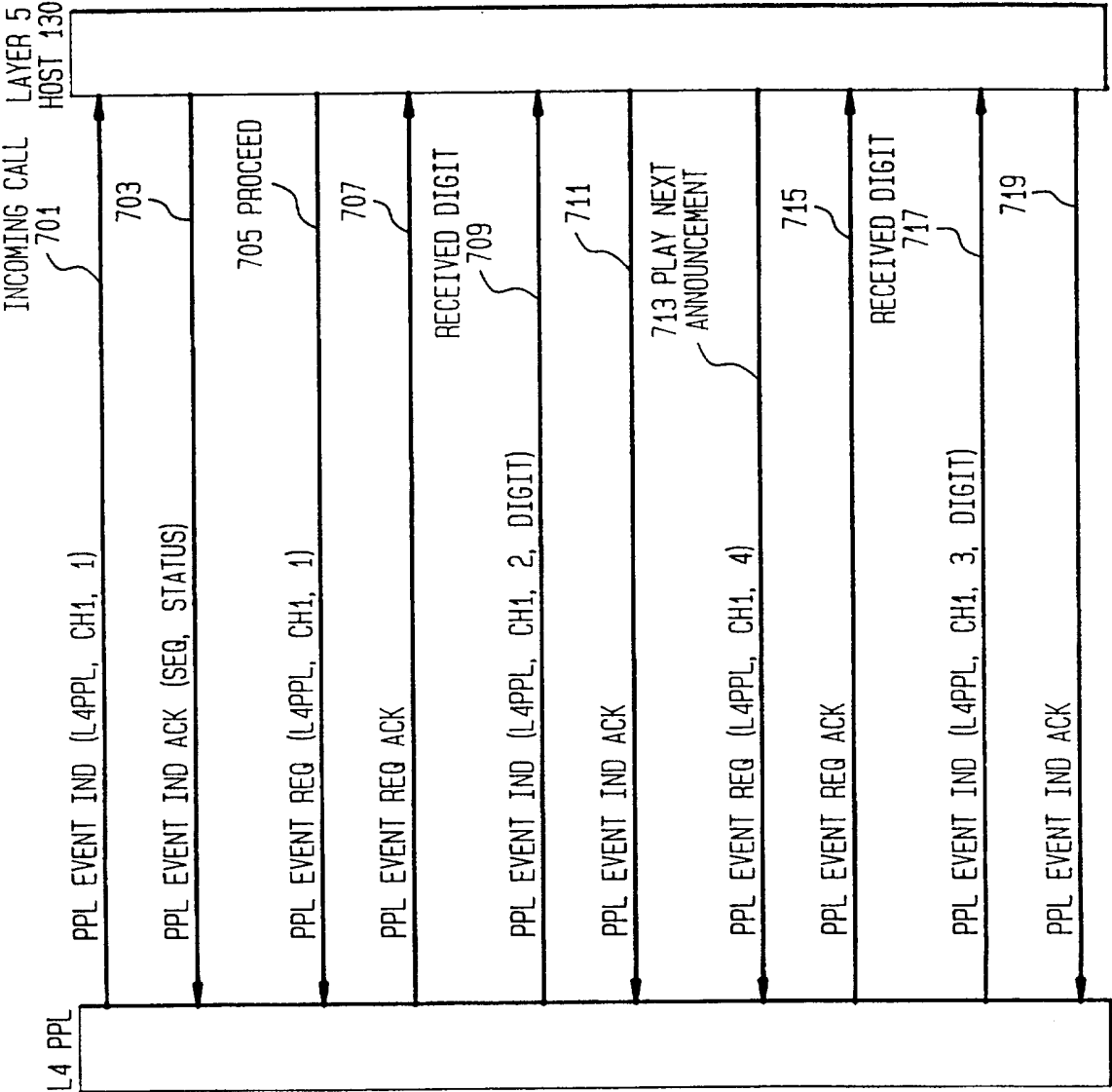
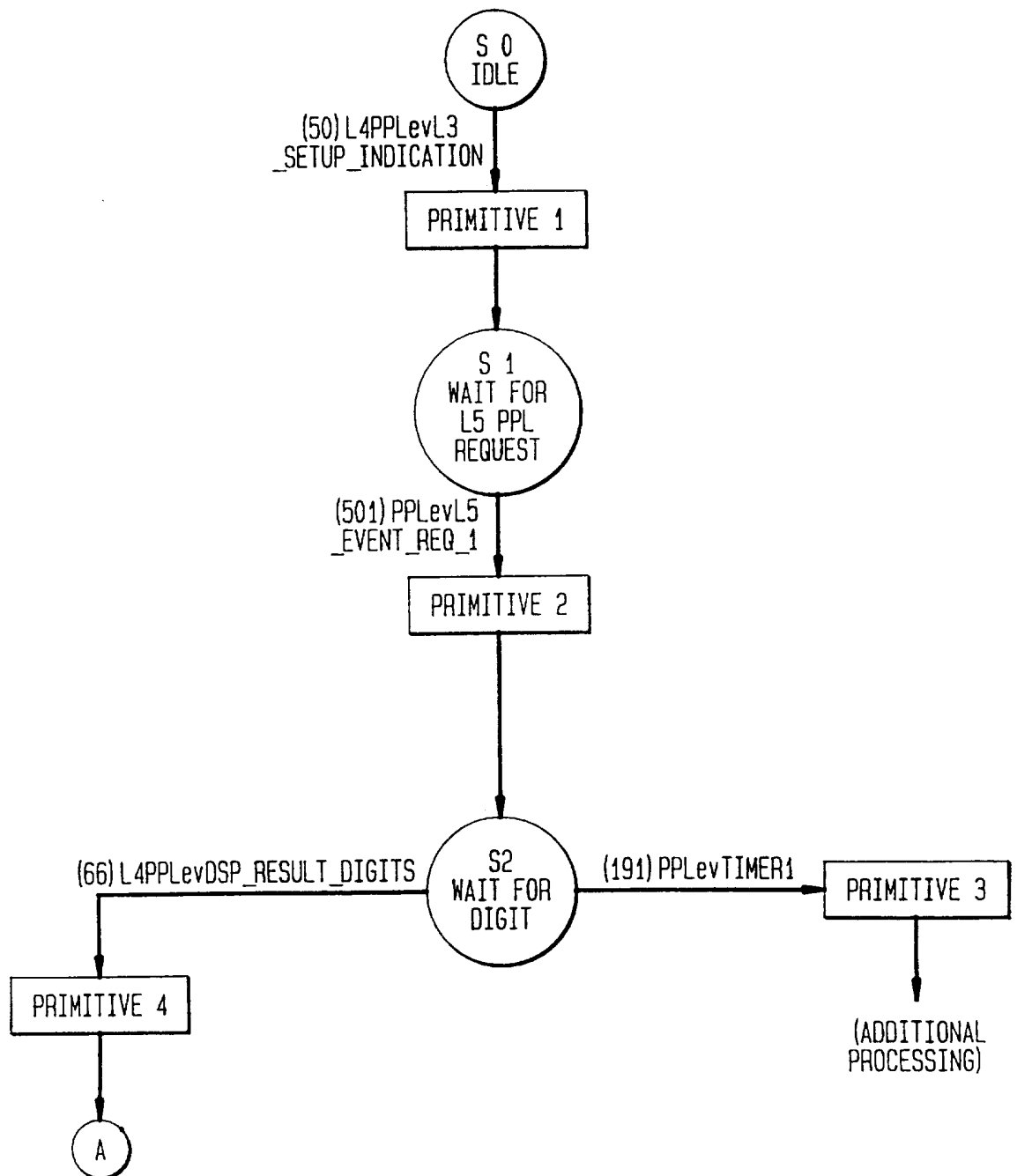


FIG. 7C

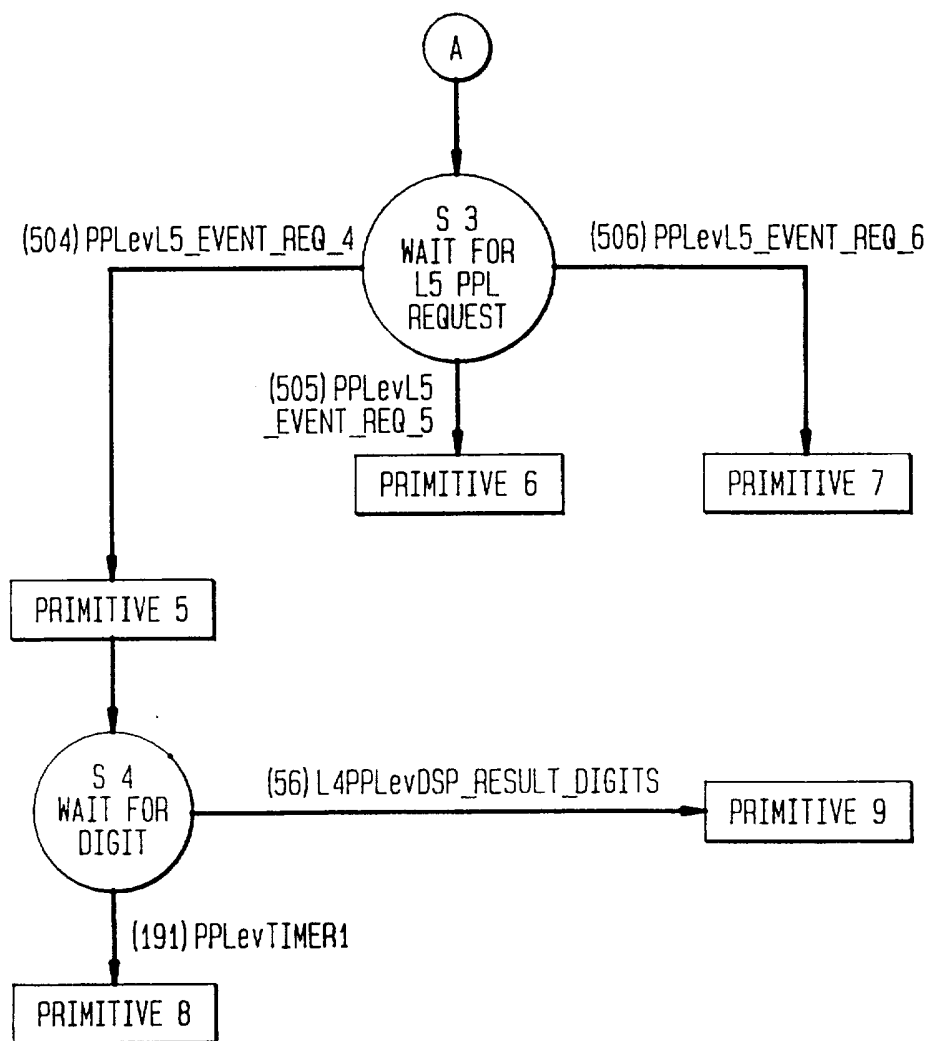
11/24

FIG. 7D



12/24

FIG. 7E



13/24

FIG. 7F

780

PRIMITIVE ID	1ST ATOMIC FUNCTION	2ND ATOMIC FUNCTION	3RD ATOMIC FUNCTION	4TH ATOMIC FUNCTION	5TH ATOMIC FUNCTION
PRIMITIVE #1	af035 (0x01, 0x00)				
PRIMITIVE #2	af060 (0x16, 0x00)	af062 (0x00, 0x00)	af140 (0x00, 0x00)	af212 (0x01, 0x00)	af050 (0x01, 0x10)
PRIMITIVE #3	af035 (0x04, 0x00)				
PRIMITIVE #4	af047 (0x01, 0x00)	af053 (0x01, 0x00)	af036 (0x02, 0x01)	af147 (0x00, 0x00)	
PRIMITIVE #5	af060 (0x16, 0x00)	af140 (0x00, 0x00)	af212 (0x03, 0x00)	af050 (0x01, 0x10)	
PRIMITIVE #6	af060 (0x16, 0x00)				
PRIMITIVE #7	af060 (0x16, 0x00)				
PRIMITIVE #8	af035 (0x04, 0x00)				
PRIMITIVE #9	af047 (0x01, 0x00)	af053 (0x01, 0x00)	af036 (0x03, 0x0)		

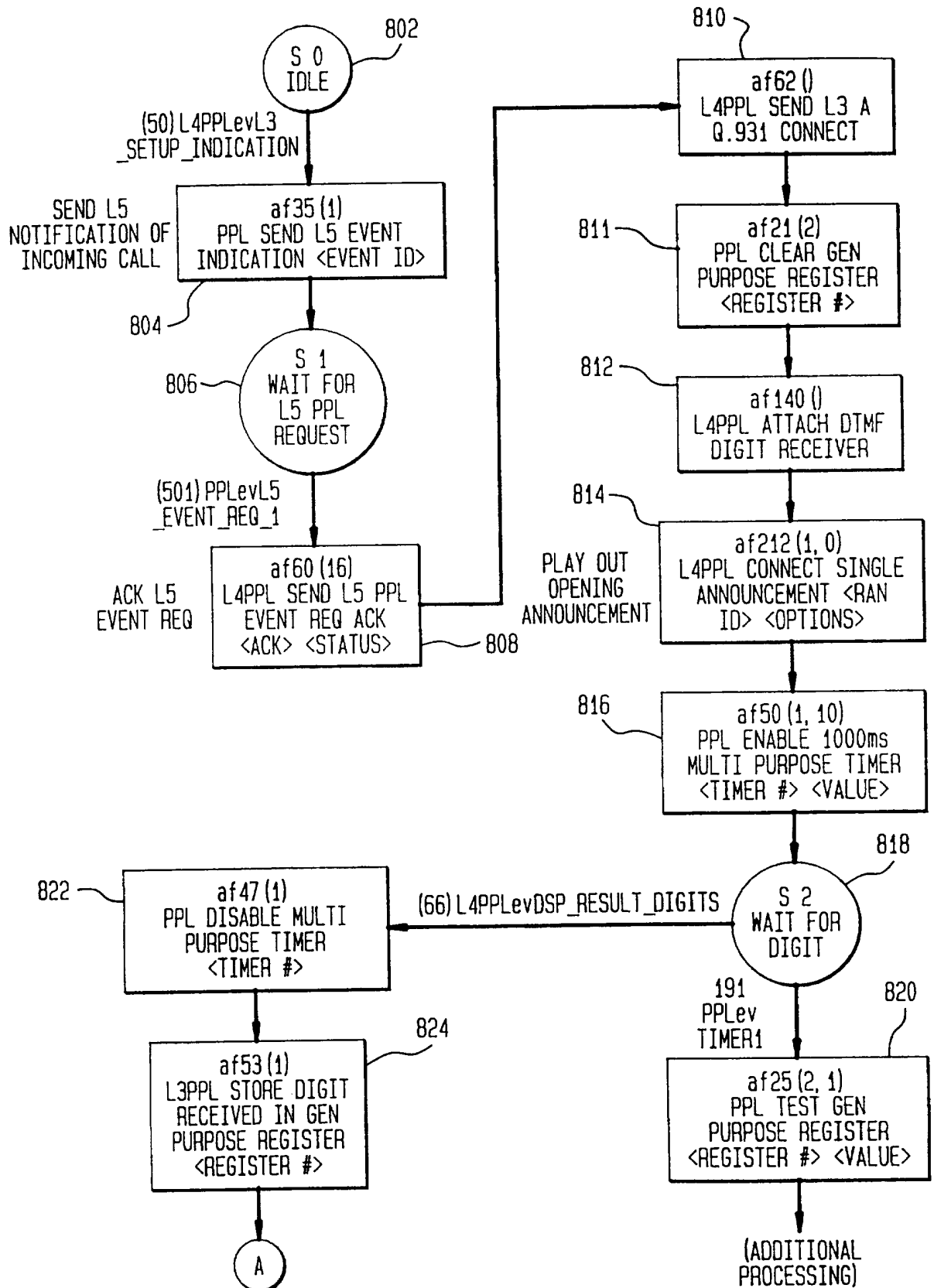
FIG. 7G

790

STATE NO.	EVENT	EVENT ID	PRIMITIVE ID	NEXT STATE	STATE TYPE
STATE 0	L4PPLeVL3_SETUP_INDICATION	ev050	PRIMITIVE 1	STATE 1	NORMAL
STATE 1	L4PPLeVL5_EVENT_REQ_1	ev501	PRIMITIVE 2	STATE 2	NORMAL
STATE 2	PPLevTIMER1 L4PPLeVDSP_RESULT_DIGITS	ev191	PRIMITIVE 3	STATE ?	UNKNOWN
		ev066	PRIMITIVE 4	STATE 3	NORMAL
STATE 3	L4PPLeVL5_EVENT_REQ_4 L4PPLeVL5_EVENT_REQ_5 L4PPLeVL5_EVENT_REQ_6	ev504	PRIMITIVE 5	STATE 4	NORMAL
		ev505	PRIMITIVE 6	STATE ?	UNKNOWN
		ev506	PRIMITIVE 7	STATE ?	UNKNOWN
STATE 4	PPLevTIMER1 L4PPLeVDSP_RESULT_DIGITS	ev191	PRIMITIVE 8	STATE ?	UNKNOWN
		ev066	PRIMITIVE 9	STATE ?	UNKNOWN

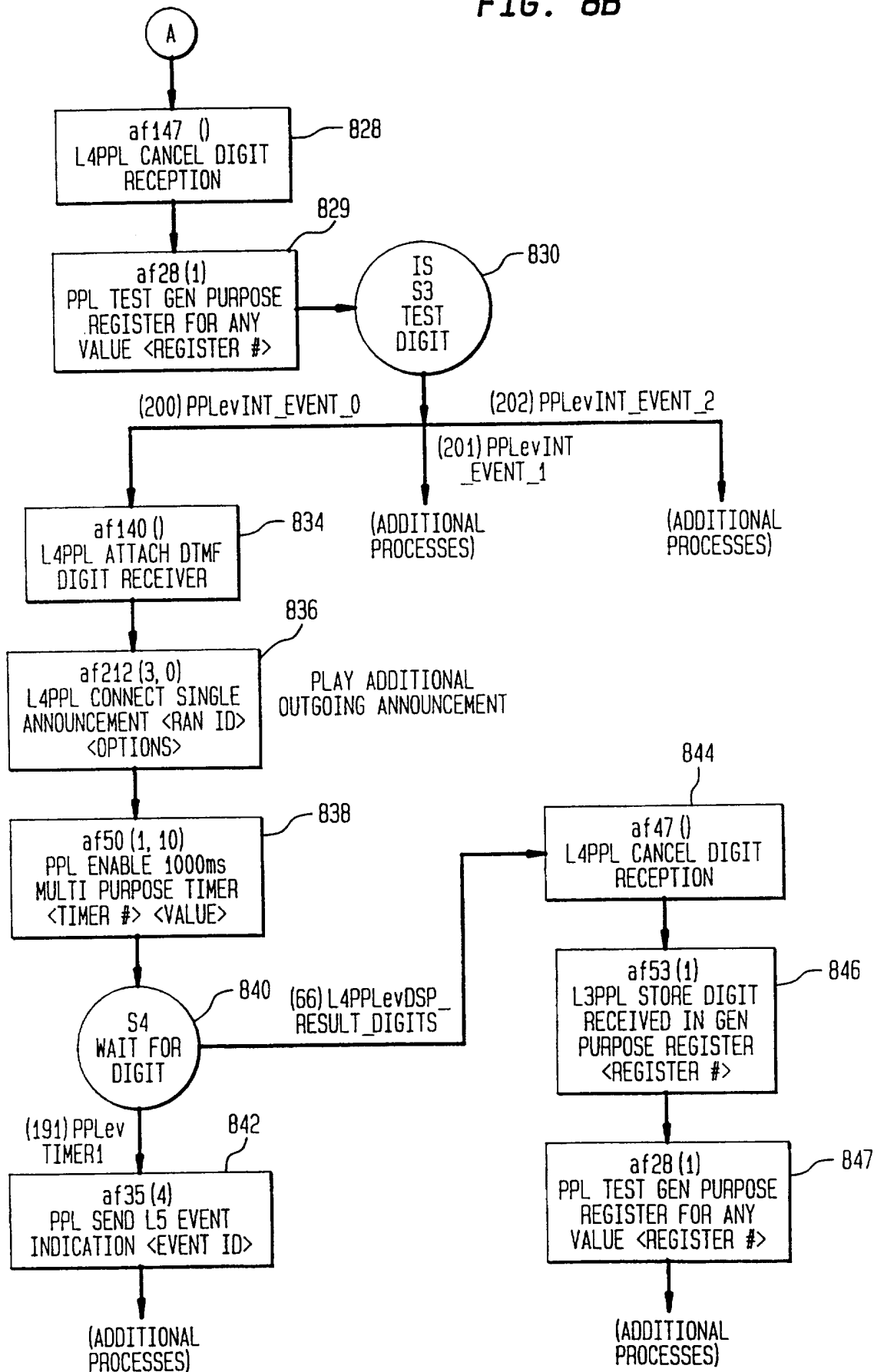
15/24

FIG. 8A



16/24

FIG. 8B



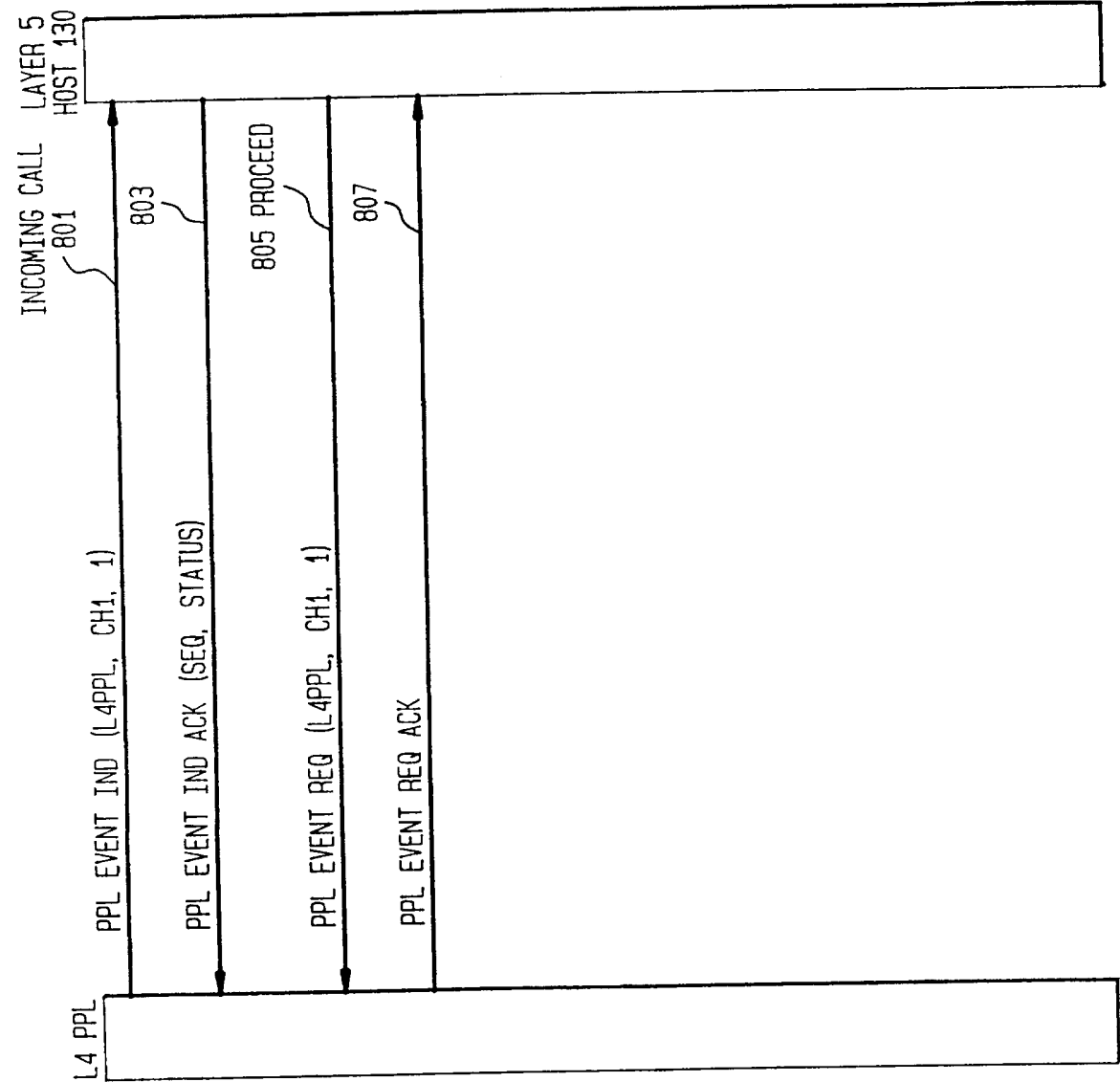
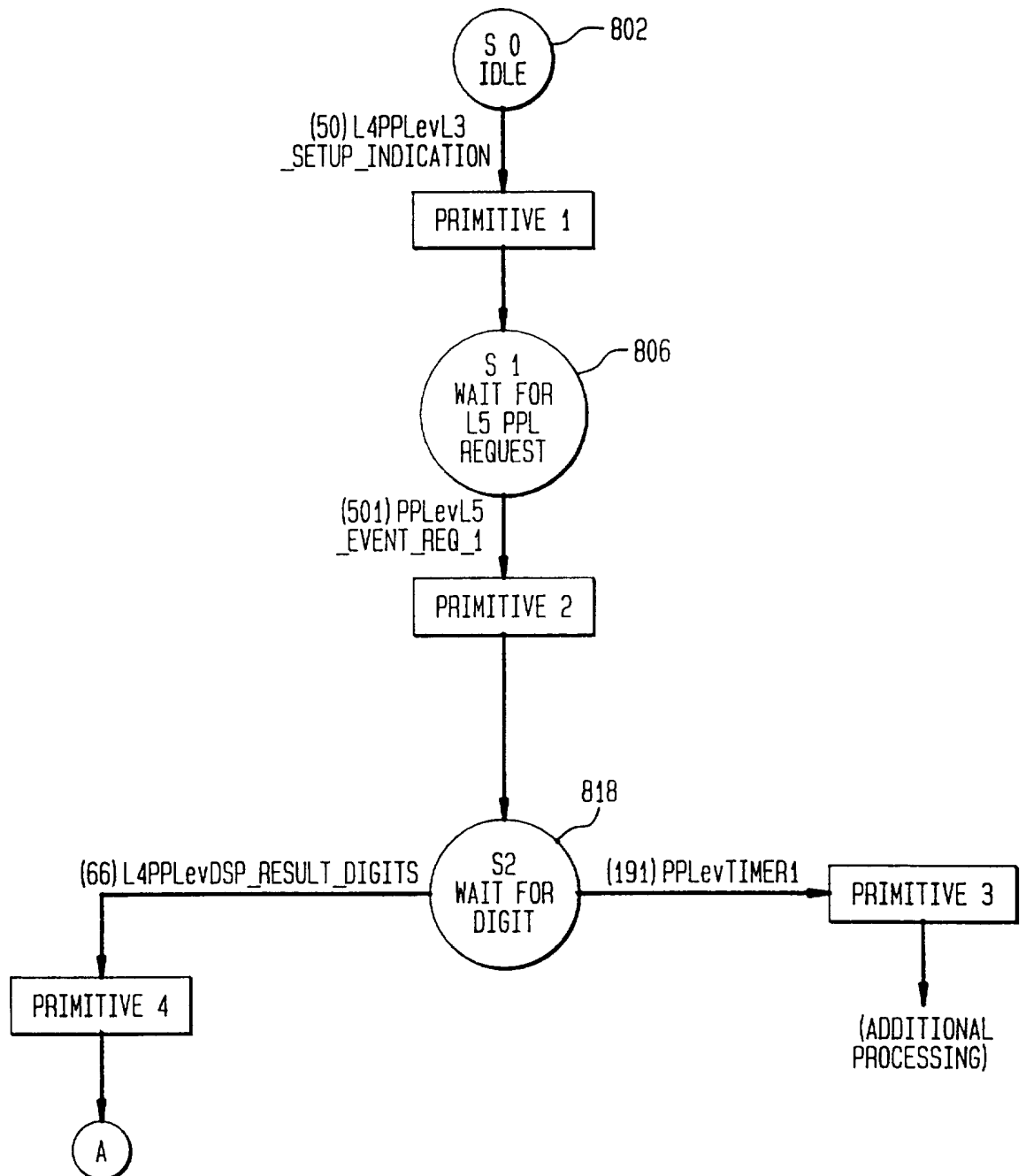


FIG. 8C

18/24

FIG. 8D



19/24

FIG. 8E

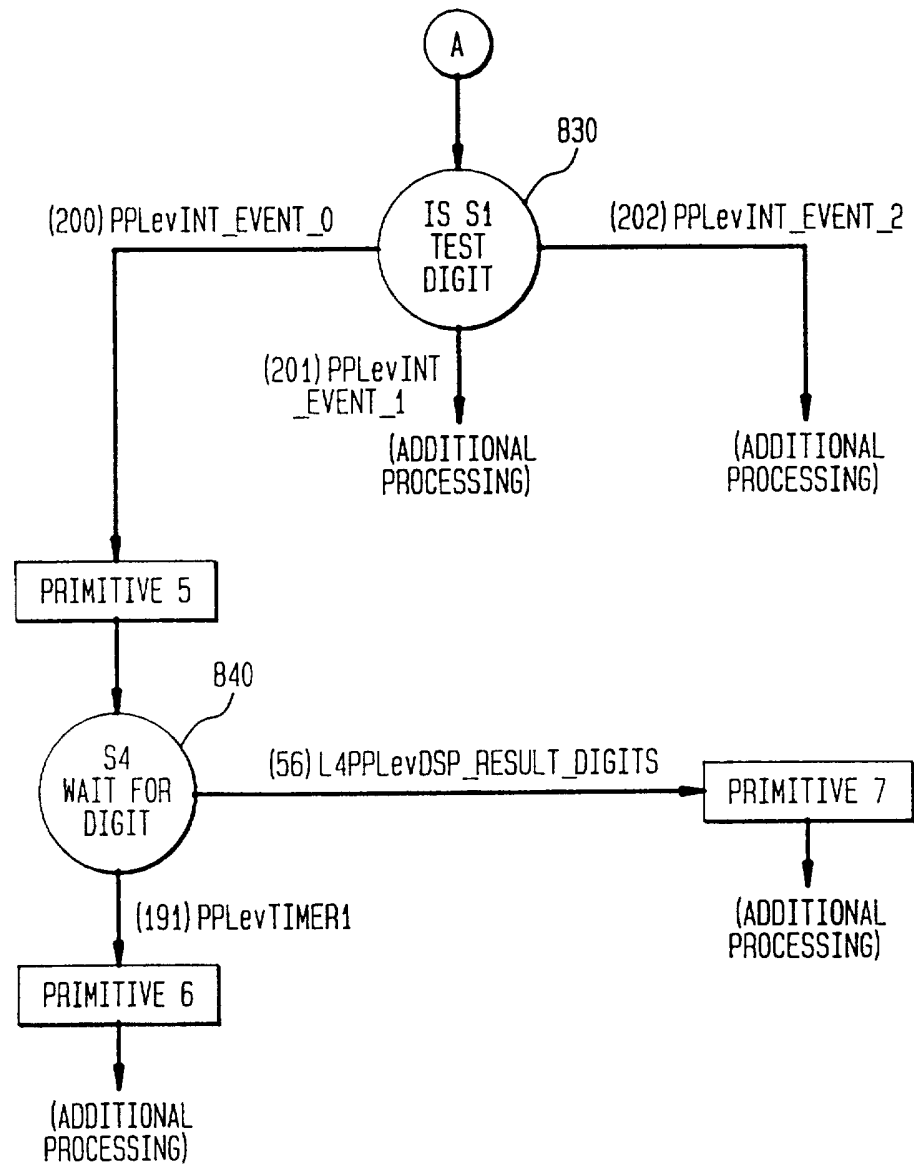


FIG. 8F

880

PRIMITIVE ID	1ST ATOMIC FUNCTION	2ND ATOMIC FUNCTION	3RD ATOMIC FUNCTION	4TH ATOMIC FUNCTION	5TH ATOMIC FUNCTION
PRIMITIVE #1	af035 (0x01, 0x00)				
PRIMITIVE #2	af060 (0x16, 0x00)	af062 (0x00, 0x00)	af140 (0x00, 0x00)	af212 (0x01, 0x00)	af050 (0x01, 0x10)
PRIMITIVE #3	af025 (0x02, 0x01)				
PRIMITIVE #4	af047 (0x01, 0x00)	af053 (0x01, 0x00)	af147 (0x60, 0x00)	af026 (0x01, 0x00)	
PRIMITIVE #5	af140 (0x00, 0x00)	af212 (0x03, 0x00)	af050 (0x01, 0x10)		
PRIMITIVE #6	af035 (0x04, 0x00)				
PRIMITIVE #7	af147 (0x01, 0x00)	af053 (0x01, 0x00)	af028 (0x01, 0x00)		

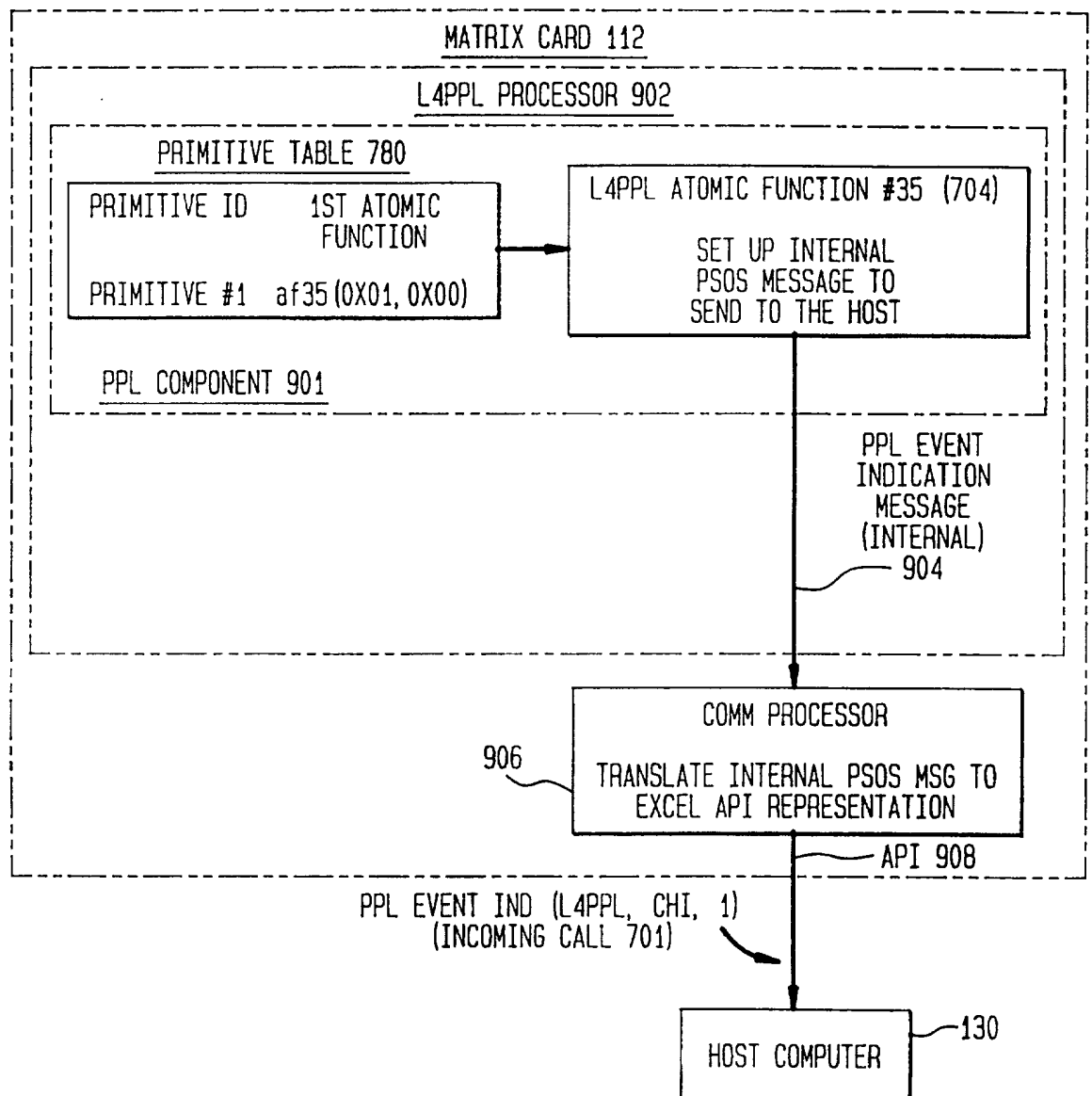
FIG. 8G

890

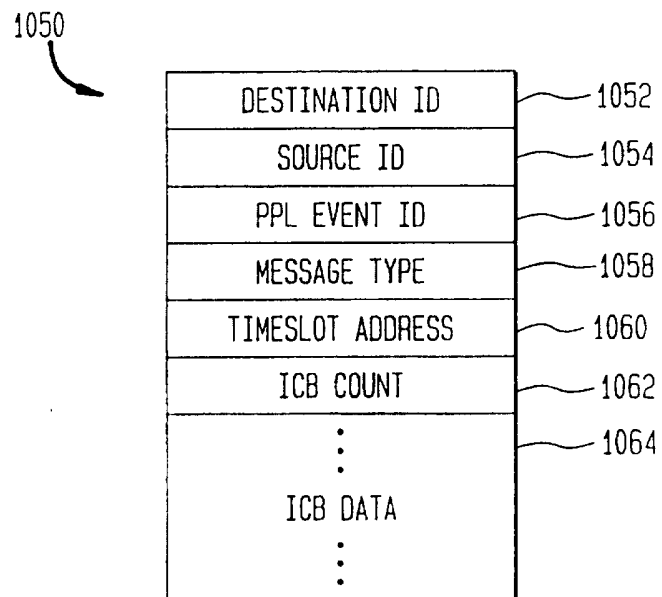
STATE NO.	EVENT	EVENT ID	PRIMITIVE ID	NEXT STATE	STATE TYPE
STATE 0	L4PPLevL3_SETUP_INDICATION	ev050	PRIMITIVE 1	STATE 1	NORMAL
STATE 1	L4PPLevL5_EVENT_REQ_1	ev501	PRIMITIVE 2	STATE 2	NORMAL
STATE 2	PPLevTIMER1 L4PPLevDSP_RESULT_DIGITS	ev191 ev066	PRIMITIVE 3 PRIMITIVE 4	STATE ? STATE 3	UNKNOWN INTERNAL
STATE 3	PPLevINT_EVENT_0 PPLevINT_EVENT_1 PPLevINT_EVENT_2	ev200 ev201 ev202	PRIMITIVE 5 PRIMITIVE ? PRIMITIVE ?	STATE 4 STATE ? STATE ?	NORMAL UNKNOWN UNKNOWN
STATE 4	PPLevTIMER1 L4PPLevDSP_RESULT_DIGITS	ev191 ev066	PRIMITIVE 6 PRIMITIVE 7	STATE ? STATE ?	UNKNOWN UNKNOWN

22/24

FIG. 9

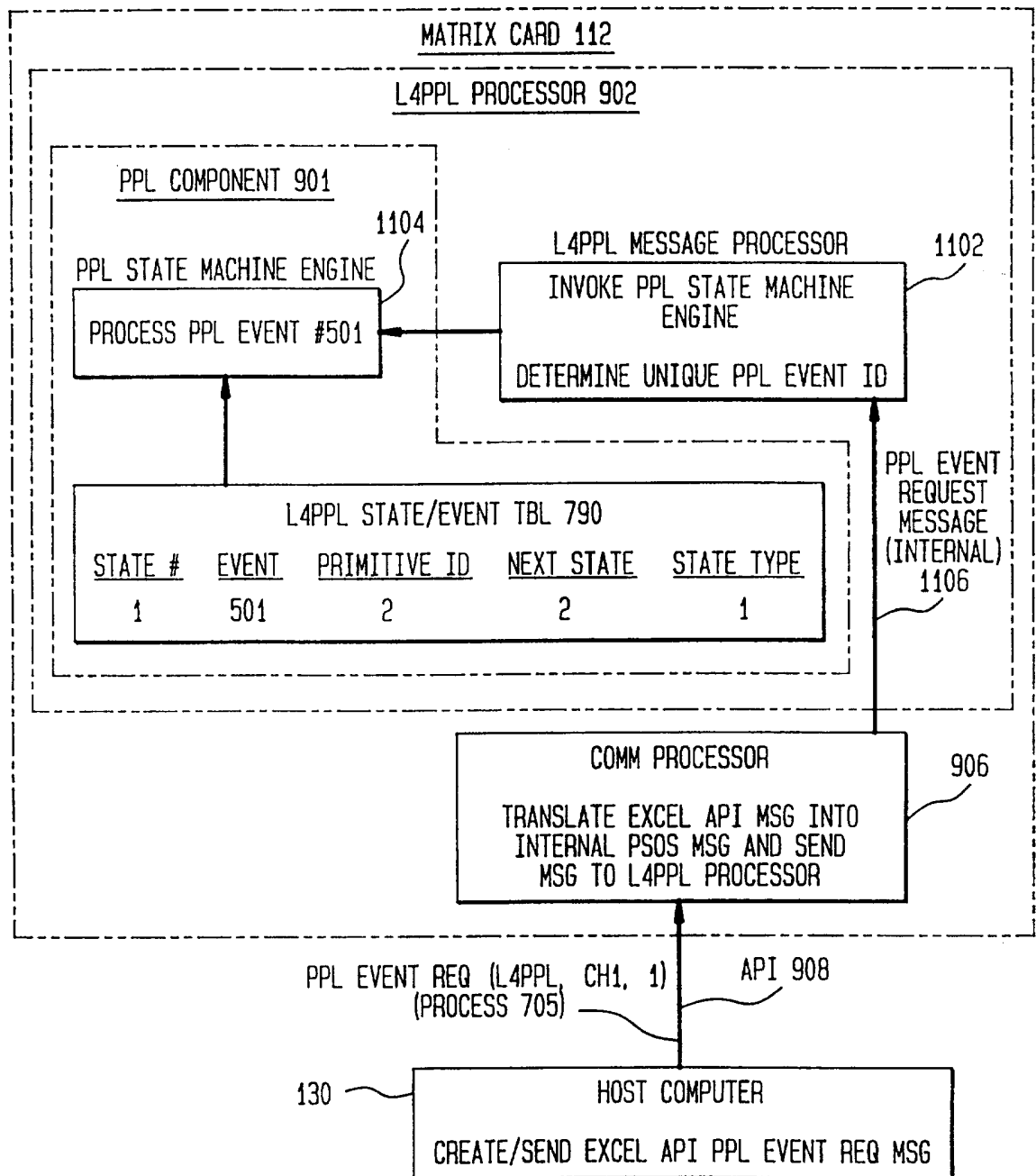


23/24

FIG. 10

24/24

FIG. 11



INTERNATIONAL SEARCH REPORT

International Application No.
PC1/US 96/18959

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04Q3/62 H04Q11/04 H04L29/06 H04Q3/545

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 6 H04Q H04L

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)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 5 546 453 A (HEBERT) 13 August 1996 see the whole document ---	1,11,31, 74,78, 81,92
A	US 5 426 694 A (HEBERT) 20 June 1995 cited in the application see the whole document ---	1,11,31, 74,78, 81,92
A	COMPUTER COMMUNICATIONS REVIEW, vol. 21, no. 4, 1 September 1991, NEW YORK (US), pages 197-205, XP000234937 TSCHUDIN C: "FLEXIBLE PROTOCOL STACKS" see the whole document ---	1,11,31, 74,78, 81,92
-/--		

☒ Further documents are listed in the continuation of box C.

☒ 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

20 March 1997

Date of mailing of the international search report

01. 04. 97

Name and mailing address of the ISA
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

Authorized officer

De Muyt, H

INTERNATIONAL SEARCH REPORT

International Application No

PC1/US 96/18959

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	MICROPROCESSING AND MICROPROGRAMMING, vol. 38, no. 1 / 05, 1 September 1993, AMSTERDAM (NL), pages 741-748, XP000383837 RUMEN STAINOV: "DYNAMIC PROTOCOL CONFIGURATION FOR MULTIMEDIA NETWORKS" see the whole document ---	1,11,31, 74,78, 81,92
A	IEEE COMMUNICATIONS MAGAZINE, vol. 31, no. 8, 1 August 1993, NEW YORK (US), pages 38-45, XP000393761 BOUMEZBEUR R ET AL: "SPECIFYING TELEPHONE SYSTEMS IN LOTOS" see the whole document ---	1,11,31, 74,78, 81,92
A	INTERNATIONAL CONFERENCE ON COMMUNICATIONS - PAPER 317.2, vol. 1, 14 - 18 June 1992, CHICAGO (US), pages 442-447, XP000326907 ATSUSHI ITO ET AL: "TRANSFORMATION TECHNIQUE BETWEEN SPECIFICATION IN SDL AND SPECIFICATION IN MESSAGE SEQUENCE CHARTS FOR DESIGNING PROTOCOL SPECIFICATIONS" see the whole document ---	1,11,31, 74,78, 81,92
A	EP 0 555 997 A (AMERICAN TELEPHONE AND TELEGRAPH COMPANY) 18 August 1993 see the whole document ---	1,11,31, 74,78, 81,92
A	EP 0 680 186 A (KONINKLIJKE PTT NEDERLAND N.V.) 2 November 1995 see the whole document ---	1,11,31, 74,78, 81,92
A	US 5 455 827 A (KRAUSE ET AL) 3 October 1995 see the whole document -----	1,11,31, 74,78, 81,92

INTERNATIONAL SEARCH REPORT

information on patent family members

International Application No

PCT/US 96/18959

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5546453 A	13-08-96	US 5426694 A AU 7929694 A BG 100473 A CA 2173247 A EP 0724804 A FI 961533 A NO 961351 A PL 316494 A WO 9510912 A	20-06-95 04-05-95 30-09-96 20-04-95 07-08-96 04-04-96 02-04-96 20-01-97 20-04-95
US 5426694 A	20-06-95	AU 7929694 A BG 100473 A CA 2173247 A EP 0724804 A FI 961533 A NO 961351 A PL 316494 A WO 9510912 A US 5546453 A	04-05-95 30-09-96 20-04-95 07-08-96 04-04-96 02-04-96 20-01-97 20-04-95 13-08-96
EP 555997 A	18-08-93	CA 2088395 A JP 6006406 A	11-08-93 14-01-94
EP 680186 A	02-11-95	NL 9400682 A JP 7302248 A	01-12-95 14-11-95
US 5455827 A	03-10-95	WO 9523492 A	31-08-95