REPUBLIC OF SOUTH AFRICA
PATENTS ACT, 1978

FORM P.8
(to be lodged in duplicate)

A&A Ref: **142886**

PUBLICATION PARTICULARS AND ABSTRACT
(Section 32(3)(a) - Regulations 22(1)(g) and 31)

21	01	PATENT APPLICATION NO	22	LODGING DATE	43	ACCEPTANCE DATE
		2000/7719		**20 December 2000**		25.7.2001

51	INTERNATIONAL CLASSIFICATION	NOT FOR PUBLICATION
	G06F	CLASSIFIED BY: **ISA**

71	FULL NAME(S) OF APPLICANT(S)

Aristocrat Technologies Australia Pty. Ltd.

72	FULL NAME(S) OF INVENTOR(S)

MACH, Ronald Edward

EARLIEST PRIORITY CLAIMED	COUNTRY		NUMBER		DATE	
	33	**US**	31	**60/086,632**	32	**23 May 1998**

NOTE: The country must be indicated by its International Abbreviation - see schedule 4 of the Regulations

54	TITLE OF INVENTION

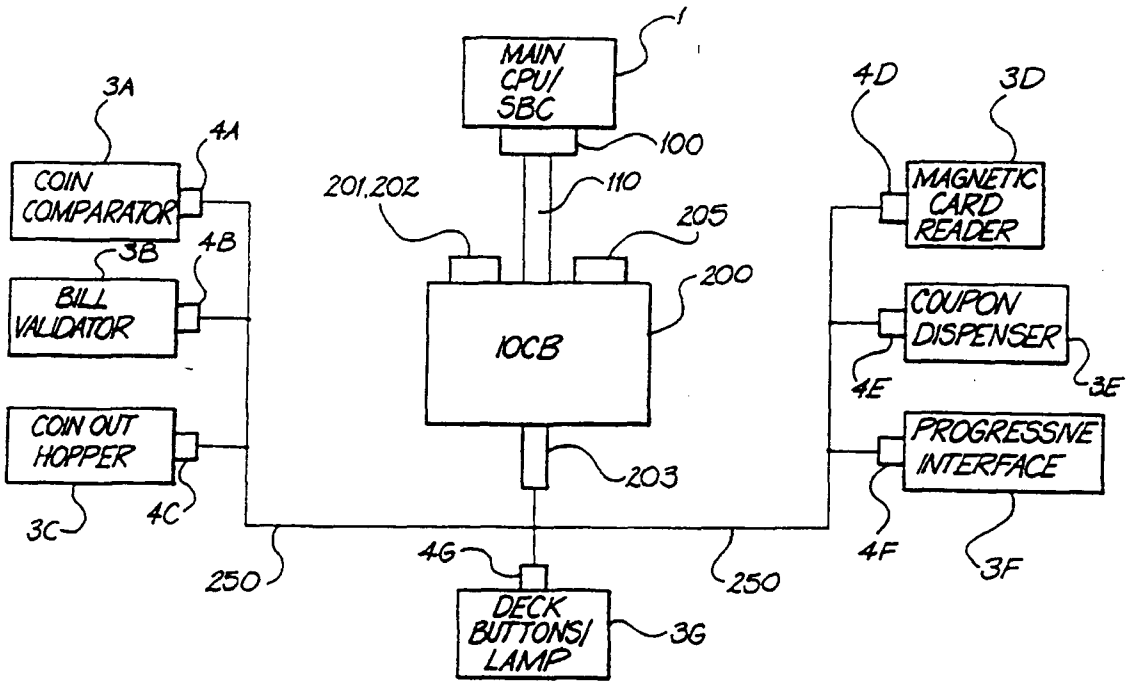
Secured inter-processor and virtual device communications system

57	ABSTRACT (NOT MORE THAN 150 WORDS)

NUMBER OF SHEETS	03

The sheet(s) containing the abstract is/are attached.

If no classification is furnished, Form P.9 should accompany this form.
The figure of the drawing to which the abstract refers is attached.



(57) **Abstract**

The invention comprises an electronically secured inter–processor and virtual device communications system, with an input/output controller board, a multi–drop bus interface to multiple devices, and a parallel interface to an industry standard single board computer. The invention assigns a bus address and virtual identification number to each device and controls communications between the main central processing unit and the devices through a Plug–n–Play protocol.

SECURED INTER-PROCESSOR AND VIRTUAL DEVICE COMMUNICATIONS SYSTEM

FIELD OF THE INVENTION

The present invention is directed generally to an electronically secured inter-processor and virtual device communication system used in gaming machines and is directed more particularly to an electronically secured inter-processor and virtual device communications system with (1) an Input/Output Controller Board (hereinafter referred to as "IOCB"), (2) a multi-drop bus interfacing one or more device modules, (3) a parallel interface to an industry standard main single board computer (hereinafter referred to as the "SBC") of the gaming machine and (4) a IOCB-to-device "Plug-N-Play protocol (the "Plug-N-Play Protocol").

BACKGROUND OF THE INVENTION

Conventional methods of integrating a variety of components in a gaming machine currently require re-designing, re-wiring and re-programming the gaming machine as each new component is added. As technology advances, newer state-of-the-art devices are being offered to manufacturers of gaming machines that would greatly enhance the product, but manufacturers are inhibited, as removing and replacing the old devices with new devices, or simply adding the new device requires reprogramming the SBC and could require the re-design and/or replacement of the SBC.

The industry standard SBC board, independent of the microprocessor used, has been designed using discrete components with EPROM memory chips containing software with an individual bus connected to each device. As most of the gaming machines in the market require some sort of regulatory approval, modifying a previously approved product requires a re-submittal to the regulatory agency with the re-submittal emphasising recertification of the programming of the new device in the CPU. With the software for each device residing in the EPROM, this requires reverification of the entire program.

Especially for manufacturers with a significant installed base of gaming machines, retrofitting these machines with a newer device is an expensive proposition and a logistical nightmare. Utilising a 'Plug-N-Play' concept, the Invention, through the Input/Output Controller Board (IOCB), logically interconnects all the devices in the gaming machine to the SBC. The IOCB's communications to and from each device is based on a network-

capable communications protocol, such as Philips' Inter-Integrated Circuit (I^2C) two-wire Serial Interface (hereinafter "PHILIPS I^2C").

As the IOCB logically interconnects the modules or potential modules to the SBC, the SBC and its EPROM memory chips do not have to be reprogrammed, resigned or replaced. Further, as no device related software resides on the SBC or its EPROMS, no new software needs to be resubmitted to any regulatory agency, as no new software is created. Instead each device in the machine incorporates an intelligent board with microprocessor (the "Device Board") which is programmed specifically to the functions of that device. The Device Board also communicates via the communications protocol, such as PHILIPS I^2C, to the IOCB in a multi-drop configuration. Replacing a device or adding a new device is simply a matter of connecting the communications interface (Clock, Data, Logic Power, System Ground) to the multi-drop bus interconnecting all the devices to the IOCB.

The IOCB programming module continuously monitors the communications protocol interface (the PHILIPS I^2C line) for device activity and relays these actions to and from the SBC. As new devices are added to the link, a specific registration protocol is followed which will allow the device to register with the IOCB. If the registering device has followed all the secured protocols, the specific parameters of the device (device type, Serial Number, etc.) are relayed to the SBC.

The SBC has several modules programmed for all possible devices that may connect to the machine, such as a Coin In, Coin Out, and Bill Acceptor modules. Even though there may be several types of these devices, the appropriate SBC module simply monitors for coins in, coins out and bills accepted. The specific hardware and protocol of each connected device is level converted and formatted by each Device Board to a generic format required by the SBC module.

Accordingly, an object of the invention is an electronically secured inter-processor and virtual device communication system which allows devices to be added to, replaced in or changed in a gaming machine without any need to reprogram or redesign the SBC of the gaming machine.

A further object of the invention is an electronically secured inter-processor and virtual device communication system which allows devices to be added, replaced in, or changed in a gaming machine without a need to replace the SBC of the gaming machine.

An additional object of the invention is an electronically secured inter-processor and virtual device communication system which allows devices to be added to, changed in, or replaced in a gaming machine without the need to modify the software residing in the CPU or any device so that a resubmittal to an appropriate regulatory body would not be necessary.

Still another object of the invention is an electronically secured inter-processor and virtual device communication system which cost-effectively allows devices to be changed in a gaming machine.

Furthermore, an object of the invention is an electronically secured inter-processor and virtual device communication system which allows devices to most efficiently be added to, replaced in or changed in a gaming machine.

An additional object of the invention is an electronically secured inter-processor and virtual device communication system with a protocol which supports dynamic assignment of device addresses.

Still a further object of the invention is an electronically secured inter-processor and virtual device communication system which provides reliable and secure communications among interprocessors.

SUMMARY OF THE INVENTION

These and other objects of the invention, which shall become apparent hereafter, are achieved by providing an electronically secured inter-processor and virtual device communication system, including an IOCB connecting a multitude of peripheral gaming machine devices and the SBC through parallel interface. In an embodiment, the IOCB provides an interface between the SBC of the gaming machine and the machine's devices and is connected to the SBC through a parallel interface. The IOCB uses a multi-drop communication bus for its connection with the devices requiring at least four wires for the clock, data, logic power, and system ground protocol. Each device contains its own CPU board programmed for the specific device. When power is first applied to the gaming machine, the IOCB attempts to establish a link with the SBC by placing a 'link request' transaction in the IOCB's transmit queue and commencing an idle state. The IOCB then remains in an idle state until the SBC acknowledges a physical connection. Once the SBC acknowledges a physical connection, the IOCB sends the 'link request' transaction to the SBC, preferably through the IOCB's Parallel Slave Port (Data) (the "PSP-Data"). When the link is established, the IOCB sends a

framed packet to the SBC containing the Virtual ID and device registration commences. Independently of the SBC and the IOCB, each device's CPU attempts to register the device's hardware with the IOCB through the multi-drop bus. As the IOCB registers each device, the IOCB assigns an communications protocol address, preferably the PHILIPS I^2C address, to the device and creates a device table entry containing data uniquely identifying the device, such as type of device, serial number, and the communications protocol address. As each device is registered, the SBC assigns and stores a virtual identification number (the "Virtual ID"). After all devices are registered, the IOCB stores information about each registered device and transfers any pertinent packets received from a device to the SBC.

In a preferred embodiment, the IOCB is the only external device interfacing the SBC with the gaming machine's devices such as Deck Buttons and Lamps, Coin In Mechanisms, Coin Out Hoppers, Bill Acceptors, Magnetic-Stripe Card Readers, Keypads, Progressive Display Interfaces, Ticket Printers, Coupon Dispensers, Hard Meters and general Security switches in the machine. The IOCB's physical connection to the SBC is preferably via a parallel interface on the SBC, preferably a PC-104 bus which is capable of transfer rates up to 8 million bytes per second. The data transfer between the IOCB and SBC is interrupt-driven and controlled by an 8-bit register.

The IOCB interfaces the listed devices through a Multi-Drop communication scheme using a network-capable communications protocol, such as RS-485, USB, current loop, or preferably Philips Corporation's two-wire Inter-Integrated Circuit (I^2C) serial interface and corresponding communications protocol ("hereinafter "I^2C protocol"), which enables speeds up to 400 kbps.

The communications protocol, preferably the I^2C protocol, ensures reliable transmission and reception of data. When transmitting data, only one apparatus, preferably the IOCB, is the 'master' which initiates transfer on the bus and generates the clock signals to permit that transfer, while the other device(s) acts as the 'slaves'.

The Philips communication protocol ("I^2C Protocol") operates on top of Philip's published industry standard I^2C protocol. Additional information on the PHILIPS I^2C specification can be obtained from the document "*The I^2C*

bus and how to use it", #939839340011, available from the Philips Corporation.

The communications protocol, preferably the I^2C protocol, constitutes the physical layer for the invention's IOCB to Device 'Plug-N-Play' protocol, which is a packet-driven securitized protocol.

The preferably I^2C framed 'Plug-N-Play' protocol supports dynamic assignment of I^2C addresses, facilitates reliable communications between I^2C devices and the IOCB and provides a secured link for inter-processor communications.

A multi-wire Multi-Drop bus, preferably a four-wire multi-drop bus (Clock, Data, Logic Power, System Ground), interconnects all devices throughout the machine to the IOCB. Each device is equipped with a firmware-based CPU board which is programmed to the specific parameters and purpose of each device. Each device's CPU board is capable of communicating with the communications protocol, preferably the I^2C protocol.

Messages are routed to and from the IOCB to the devices using a communications protocol framed packet, preferably an I^2C framed packet comprised of an I^2C address, header, body and footer, as specified below.

To prevent statistical breakdown of the CRC-16, packets will be limited to a maximum of 255 bytes inclusive of the I^2C Address and Footer. Message bodies larger than 248 bytes must be broken up into multiple communications protocol framed packets, preferably I^2C framing packets.

When power is first applied to the gaming machine, the IOCB attempts to establish a link with the SBC by placing a 'link-request' transaction in the IOCB transmit queue, which commences an idle state. The IOCB remains in this idle state until the SBC acknowledges a physical connection. Upon receiving acknowledgment of the physical connection, the IOCB sends this link request transaction to the SBC via preferably the Parallel Slave Port (PSP Data) of the IOCB through a PSP-framed packet containing the Virtual ID of the device. Once the link is established, device registration begins. Neither the IOCB nor the SBC has knowledge of the devices integrated in the machine. Each device's CPU will attempt to 'register' its specific hardware with the IOCB by communication through the communications protocol multi-drop bus, preferably a I^2C Multi-Drop bus. As each device is registered by the IOCB, the IOCB dynamically assigns a communications protocol

address (range 9-76h), preferably an I^2C address, to the device and enters the device information into a device table which stores the device's specific data (type of device, serial number, etc.). As long as power is applied to this device, a device responds to IOCB requests using this communications protocol address, such as an I^2C address. The IOCB also assigns Virtual ID (Circuit Number) to the device referencing the device to the SBC. The SBC uses this Virtual ID to invoke a software driver referencing this device type regardless of its communications protocol, preferably I^2C, address. The Virtual ID remains assigned to the device and, should the device lose power, upon re-registering, the device may receive a different communications protocol, preferably I^2C, address but will maintain the same Virtual ID. This process repeats until all devices have been registered.

As each new device is dynamically registered, a table entry in the IOCB's device table is created for that device. As the IOCB is continuously polling these registered devices, the IOCB will detect removal of a device and will send notification to the SBC.

After device registration, utilising a prioritized polling scheme, the IOCB will query each registered device for status. The IOCB either receives a 'no activity' packet or a packet containing pertinent data for that device. If required, the IOCB transfers a valid packet to the SBC via preferably the PSP-Data port.

If a registered device fails to respond to its poll after a fixed number of retries, the IOCB declares the device 'inactive' by sending an appropriate PSP framed transaction to the SBC.

The IOCB communicates with the SBC via a parallel port connected to a PC-104 bus on a SBC. The interrupt-driven data transfer is controlled by a shared 8-bit register which is used as handshaking flags for flow control.

The communications protocol bus is a multi-wire communications interface, preferably a I^2C bus with a two-wire serial interface developed by the Philips Corporation which enables speeds up to 400 kbps.

The IOCB will be interrupted by the PSP-Data if the SBC initiates any PSP framed transactions. If the SBC initiates a transaction, the IOCB will wrap the PSP framed characters with communications protocol framing characters, such as I^2C framing characters, sending this data to the appropriate device at that communications protocol address.

In any setting, the IOCB's sole responsibility is to direct secured data transactions to and from the SBC and to and from the gaming machine's devices. The IOCB does not initiate or control any functions of the gaming machine.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention would be better understood from the following Detailed Description of the Preferred and Alternate Embodiments, with reference to the accompanying drawings, of which:

Figure 1 is one embodiment of the invention showing the SBC of the gaming machine connected to the IOCB and connected to seven devices through device boards in a multi-drop bus configuration.

Figure 2 is an embodiment of the invention, showing the preferred PC-104 bus of the SBC connected through a preferred parallel interface by an 8-bit bi-directional bus to two parallel slave ports (PSP-Data and Control, and the 8-bit register) of the IOCB.

Figure 3 is a preferred embodiment of the invention, showing the IOCB connected in a multidrop configuration with four wires to five devices.

Figure 4 is a flow chart showing the transmission of data from the SBC to the devices through the IOCB.

Figure 5 is a flowchart showing the transmission of data from the devices to the SBC through the IOCB.

Figure 6 is a flowchart depicting the four-level security of the invention, which ensures the validity of data transfers. and

Figure 7 is a flowchart depicting the registration of devices by the IOCB.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to the drawings, wherein like numerals reflect like elements throughout the several views, Figure 1 depicts the SBC (generally 1) of the gaming machine, as connected to the Input/Output Controller Board ("IOCB") (generally 200). The IOCB 200 is connected in a multi-drop bus configuration 250 to the devices (3A, 3B, 3C, 3D, 3E, 3F, 3G), each device with its own microprocessor inclusive board (respectively, 4A, 4B, 4C, 4D, 4E, 4F, 4G). The devices depicted are a coin comparator 3A, a bill validator coupon 3B, a coin out hopper 3C, a magnetic card reader 3D, a coupon dispenser 3E, a progressive interface 3F, and a deck buttons/lamps 3G. Each device (3A through 3G) has its own board (4A through 4G) (the "Device

Boards") and is virtually connected to the SBC I and to the other devices through the multi-drop bus configuration 250.

Single Board Computer (SBC):

The SBC 1 is capable of digital storage, contains a microprocessor, preferably a Pentium II, and preferably contains sound and video capability, wave file capability, networking capability, and modem connections.

Input/Output Controller Board (IOCB):

The IOCB 200 is a Microprocessor based electronic board featuring a microprocessor, preferably a PIC 17C756, preferably on-board Random Access Memory (RAM), preferably Non-Volatile RAM, preferably timer/counters, preferably Capture/Compare/PWM modules, preferably two 8-bit parallel ports (Parallel Slave Port (PSP) Data 201, PSP Control 202), preferably one Serial Communications Interface, preferably two-wire Inter-Integrated Circuit (I^2C) bus 203, preferably internal/external Interrupt sources, preferably a Watchdog Timer, preferably a Brownout detection and preferably Programmable code-protection. The IOCB 200 has the ability to set up and communicate in the communications protocol standard, to communication with at least one device, and to perform distributed processing.

Device Microprocessor Board Hardware:

Attached to each device 3A through 3G in the gaming machine is a Microprocessor based electronic CPU board (the "Device Board.") 4A through 4G. Each board is specifically designed and programmed to interface with the specific function of its associated device. The devices are connected to the IOCB 200 in a multi-drop configuration 250. Each board is capable of communicating in a communications protocol, preferably an Inter-Integrated Circuit (I^2C) communications protocol, and contains a microprocessor, preferably a PIC 16C67. Integrated components of the board (size of RAM, high current output drivers, etc.) are application specific. Each board 4A, 4B, 4C, 4D, 4E, 4F, 4G contains a memory component, preferably a 64-bit serialised memory component, which is used as another security check in the design. The unique serial number of each component of each Device Board is installed into the board at time of production and provides a unique identification number for each board (the "Board"). The Board ID is used in the packet transmissions to and from the IOCB 200 as another signature verification in the calculation of the CRC-16.

Figure 2 depicts the preferred PC-104 port 100 of the SBC 1 connected by preferably an 8 bit bi-directional bus to preferably the IOCB's PSP-Data 201 and PSP-Control 202 port and to the IOCB's 8-bit register. Communication to and from the SBC 1 is accomplished through the IOCB's 8-bit PSP Data and PSP-Control port 201, 202 and the SBC's PC-104 bus 100 and the bi-directional interrupt driven data transfers utilise a shared 8-bit register 205 which regulates data direction and flow control. Depending on its specific task, the IOCB 200 will set and reset the handshaking control bits, with the SBC 1 to monitor these status bits.

Each bit of the 8 bit register 205 preferably is populated as follows:

bit	7	6	5	4	3	2	1	0
	RTR	RA	RTT	TA	CONNECT	0	ZERO	RESET

- **Bit 7 - Ready To Receive (RTR) 202A** preferably indicates, if set, that the IOCB 200 is ready to receive a data byte. If the SBC 1 has a character to send, it reads this statusbit and if set, will send the character. If reset, a time-out interval is initiated and if it expires, the SBC will report an error which locks up the game.
- **Bit 6 - Receive Aborted (RA) 202B** preferably indicates, if set, that the IOCB has detected a communication error while receiving data. The SBC 1 also monitors bit 6 prior to sending a character, and if Bit 6 is set, the SBC 1, will abort the balance of the transmission and retry again.
- **Bit 5 - Ready To Transmit (RTT) 202C** preferably indicates, if set, that the IOCB 200 has data to send. When set, this bit asserts Interrupt Request 11 (IRQl l) on the SBC 1. Once the interrupt has been serviced and the character has been read, the IOCB 200 notifies its PSP hardware 201, 202 and resets this bit. The IOCB 200 then generates a Transmit Data Register Empty interrupt, signalling that another character may be sent.
- **Bit 4 - Transmit Abort (TA) 202D** preferably indicates, if set, that the IOCB 200 has detected an internal transmission error while transmitting a packet to the SBC 1 and that no more data will be sent. If the SBC 1 detects that this bit is set, it will clear any previous characters received and abort the receive process.
- **Bit 3 - Busy 202E** preferably indicates, if set, that the IOCB 200 is busy processing a critical application and prevents the SBC 1 from an erroneous

time-out on a data transfer. The IOCB 200 sets this bit upon entering a critical application, resetting it upon completion. The SBC 1 will initiate a longer time-out interval, but upon expiration, will report an error locking up the game.

- **Bit 2 - 0 202F** This bit is reserved for future use.
- **Bit 1 - Connect 202G** preferably indicates, if set, that the handshaking flags of this register should be ignored. The IOCB 200 sets this bit to the value 1 (indicating high impedance) if the IOCB 200 and SBC 1 are disconnected, as the tri-state inputs of the PSP hardware will be high, The IOCB 200 sets this bit to the value 0 if the IOCB 200 and SBC 1 are connected These tri-state inputs (high, low, high-impedance) prevent interference between multiple devices attempting to access the line and allow the IOCB 200 to act as a traffic controller. The IOCB 200 always resets the bit to prevent erroneous actions based on bit levels being set.
- **Bit 0 - Reset 202H** preferably indicates to the SBC 1, if set, that the IOCB 200 has been up or reset, alerting the SBC 1 to set the 'state' of the gaming devices in the machine. That bit is reset each time initial communications are established between the IOCB 200 and the SBC 1.

Figure 3 depicts the interconnections among the IOCB 200 and five devices 3 in the preferred I^2C multi-drop configuration 250. The IOCB's I^2C port 203 is connected to each Device Board 4 through preferably a four-wire multi-drop bus 250. The preferred I^2C capable multi-drop bus 203 has preferably four wires (clock 251, data 252, Logic Power 253 and System Ground 254) which are distributed through the machine, providing the Device Boards 4 with a means to utilise the clock, data, power and a ground of the IOCB 200. Each device 3 is equipped with a Microprocessor based electronic circuit board 4 (the "Device Board") which is specifically designed to interface with the device's input or output signals depending on the device. The Device Boards 4 are capable of communicating using network-capable communications protocols, preferably Inter-Integrated Circuit (I^2C) protocols, enabling interconnection among the devices 3. In the preferred I^2C multi-drop bus 250, clock 251, data 252, Logic Power 253 and System Ground 254 are distributed in each device providing a clock, data transmission, power and ground to the Device Boards 4. A Device Board 4 can be attached to more than one Device 3. In that event, more than one device will have a

single communications protocol address (preferably an I^2C address), but each device will have a unique Virtual ID.

Figure 4 depicts data transfers of SBC-initiated transmissions between the devices 3 and the SBC 1 through the IOCB 200. Data transfers between the IOCB 200 and the SBC 1 are based on a PSP framed packet 500 with the preferably following protocol:

[Virtual ID][size][sequence #][Command][..body..][ETX][CRC-16]

- **Virtual ID 500A**: this byte preferably contains a circuit number or reference number by the SBC 1 determines which device-specific software driver is used to interpret a message received from the device 3 or to generate the particular data sent to the device 3.
- **Size 500B**: this byte preferably contains the character length of the PSP framed packet from Virtual ID 500A to the CRC-16 500G inclusive.
- **Sequence 500C**: this byte preferably contains the message sender's next sequential number. The message receiver maintains an expected sequential reception number corresponding to the message sender's Virtual ID. This sequence number is initiated to a 0 value and is incremented by 1 for each successful transmission, wrapping at the value of 255 back to a value of 1. The value of 0 is only used on initial setup, and if the value is 0, the message receiver resets its expected sequence number. The sequence number provides additional security to ensure that all transmissions are received (see Figure 6). If the message receiver has accepted the valid transaction (all packet criteria has been satisfied), this constitutes a successful transmission and the message receiver responds to the message sender by transmitting an acknowledge (ACK)packet which will cause the message sender and receiver to increment their sequence numbers.
- **Command 500D**: this byte preferably informs the message receiver what to do with the (if any) in the body of the message. For example, if this bytes contains "ACK", this acknowledges the message sender's last received packet and contain 0 bytes in the body of the message. Similarly, the IOCB 200 sends a Link Request command (with 0 bytes)to the SBC 1 on power-up, which requests a communication link. In an additional example, a Bill Acceptor

transaction with a command of 'B' signifies that the Bill Denomination is in the message body and contains four bytes in the body of the message. "ACK" has the hexadecimal value of 06, indicating a positive acknowledgment. "NAK" has the hexadecimal value of 15, indicating a negative acknowledgment.

- **Body 500E**: this byte preferably contains a variable number of bytes from 0 to 248, contains pertinent data regarding the transaction. For example, this field may contain the denomination of the bill accepted, the coin denomination, or the Player's Account Number processed by the Magnetic Card Reader. The actual specific data contained are determined by the Virtual ID involved.
- **ETX 500F**: this End of Transmission (ETX) byte is preferably used for packet . ETX has a hexadecimal value of 04, signalling End of the Transmission.
- **CRC-16 500G**: this two-byte field preferably is a 16-bit Cyclic Redundancy Check (CRC) value, used for packet validation and security. The CRC-16 value is generated using a 16-bit reverse polynomial-based algorithm performed on each transmitted/received byte. This 16-bit value is initially set to 0 and each byte of each Device Board 4 and the device type byte (Coin Mechanism, Bill Acceptor, etc.) is cyclic redundancy checked (CRC'd). The resultant 16-bit value, called the 'seed', is used as the initial value prior to applying the CRC algorithm to each byte in the packet. A CRC value is generated for each packet and includes the entire packet, from Virtual ID to ETX inclusive. A packet's CRC value is compared to the device's 3 seed and should be equivalent if the packet has been successfully transmitted. The CRC-16 'seeding' is applied to all transactions except the 'R'egister Command, which is used when a device 3 is being registered for inclusion in the device table (see Figure 7). With the Register command, the receiver does not have any knowledge of the Board ID or the type of device registering in the communication packet, the seed is assumed to be 0. In the examples, the CRC-16 value of 'OxCCCC' is used for reference only. The actual 16-bit value would vary depending on the data bytes in the packet.

Communication packets transferred between the SBC 1 and the IOCB 200 (PSP framed packets 500) preferably have a Virtual ID 500A embedded in the packet which is used to steer the transaction to the appropriate software driver of the appropriate device. Communication packets transferred between the IOCB 200 and the device 3 (preferably I^2C framed packets 520) preferably use an I^2C address 520A to steer the transaction. The IOCB 200 directs transactions both between the IOCB 200 and the SBC 1 and between the IOCB 200 and the devices 3. Due to its dual function, the IOCB 200 must wrap 524, 525,526, 527 the SBC-generated PSP packet 500 with preferably I^2C framing 520 prior to sending 528 the packet to a device. Likewise, it must unwrap (Figure 5, 616) the preferably I^2C frame 520 from the device generated PSP packet 500 prior to sending the packet to the SBC 1.

Each Device Board 4 may have more than one physical device attached to it. For example, a Device Board 4 may be attached to two hoppers, one to dispense coins, the other to dispense tokens. As a Device Board 4 has only one communications protocol address, preferably the I^2C address 520A, but two Virtual IDs 500A, the IOCB 200 checks 552 its device table 552 for two table entries with the l same I^2C address 520A. Utilising both the PSP and the I^2C framed protocols, the IOCB 200 directs 528 both SBC 1 generated commands (in our example, 'C' & 'T') to the same I^2C address 520A. The Device 3 at that I^2C address 520A will unwrap 544 the I^2C framing bytes and act on 546, 548 the PSP Virtual ID's Command byte 500D and message body 500E.

The following example further illustrates the process by which the IOCB 200 wraps a PSP framed packet 500. The SBC 1 wants to turn on Deck Lamp #4 with the device's Virtual ID of 126, the sender's next sequential transmission number of 56, and the command byte of 'L'. Using the following PSP packet format 500 and values:

[Virtual ID]	[size]	[sequence #]	[Command]	[body]	[ETX]	[CRC-16]
[126]	[8]	[56]	[L]	[0x04]	[ETX]	[0xCCCC]

The PSP packet 500 will contain the above listed data. Assuming there are no communication errors and the packet criteria is acceptable (see Figure 6), the IOCB 200 references its device table 554, finds this Virtual ID value 126 at I^2C address value 32. By these values, the IOCB 200 determines 551

that the data packet 500 initiated from the SBC 1, determines 553 that the received data packet is not intended for the IOCB 200 and thereby ignores 555 the sequence number 500C, the Command 500D and the body 500E of this packet. The IOCB 200 uses the size byte 500B only to count down the received bytes, i.e. mark the end of the packet 557.

The IOCB 200 then wraps this packet with its own I^2C frame. The IOCB 200 first creates 525 the message body 520 E of an I2C packet 520 from the PSP packet 500. The IOCB 200 assigns the value of "M" 526 to the Command byte 520D in the I^2C packet 520 signifying that the SBC 1 originated this transaction. The IOCB 200 then assigns 527 the next available sequential transmission number from the IOCB 200 to the sequence number 520C for this I^2C address 520A. The I^2C packet 520 is assigned the I^2C address 520A of 32 (524). In the example, the I^2C packet is populated as follows:

[I^2C Address]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[32]	[15]	[143]	[M]	[PSP pkt]	[ETX]	[0xCCCC]

The IOCB 200 then sends 528 the I^2C packet 520 to the device 3. Assuming there are no communication errors and the packet criteria is acceptable 536, 538, the device 3 sends an I^2C framed packet 520 to the IOCB 200 acknowledging (ACK) its transmission 532. Upon receiving the acknowledgment 560, the IOCB 200 creates and sends 560 a PSP-framed packet 500 to the SBC 1 acknowledging the Virtual ID has accepted its transaction.

If the I^2C Command byte 520D of 'M' indicates the SBC 1 originated this transaction 542, as opposed to the IOCB 200, the I^2C body 500F contains the PSP packet 500 sent by the SBC 1. The device 3 at this I^2C address unwraps the I^2C framing bytes 544, reads 546 the PSP framed packet Command ('L' in our example) and acts on 548 the Command by turning on Lamp #4.

Figure 5 depicts data transfers for device-initiated transmissions among the devices 3 and the SBC 1 through the IOCB 200. The communications protocol framed packet, preferably the I^2C framed packet 520, is transferred between the IOCB 200 and the device 3. Each preferred I^2C framed packet is comprised of the following parameters.

- **I^2C Address 520A** - Range 9 to 76h,
  - 0 reserved for Broadcast Address,
  - 1-7 reserved for I^2C implementation,
  - 8 reserved for the IOCB Address,
  - 77h reserved for unregistered devices.
- **Header 520B, 520C, 520D** - Packet Size, Sequence Number, Command (each one byte).

  The "ACK" value for Command has a hexadecimal value of 06, indicating a successful transmission, and the "NAK" value for Command has a hexadecimal value of 15, indicating an unsuccessful transmission.
- **Body 520E** - Up to 248 bytes of binary data.
- **Footer 520F, 520G** - ETX, CRC-16 (2 byte Cyclic Redundancy Check).

For example, a Device Board 4 has three devices 3 attached to it: Deck buttons, Deck lamps, and a Coin-in Mechanism. Each of these devices 3 is assigned its own Virtual ID 500A but the Device Board 4 only possesses one I^2C address 520A. In the example, the Coin-in Mechanism's Virtual ID is 41, the Deck button's Virtual ID is 14, and the Deck lamp's Virtual ID is 126. The I^2C address 520A is 39

Assume a coin has been inserted in the gaming machine. The Device Board 4 detects this Coin-in action and, using the Coin-In Mechanism's Virtual ID 500A of 41, sequence number 500C of 214, Command 500D of 'I', and the coin value 500E of 25 (hex 19), the device 3 generates 572, 574. 576, 578 the following PSP packet 500 with the following values:

[Virtual ID]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[41]	[9]	[214]	[,I,]	[0x0019]	[ETX]	[0xCCCC]

The PSP packet 500 is intended for the SBC 1, and, accordingly, the PSP Packet 500 must be encapsulated within an I^2C packet for delivery to the IOCB 200 to be delivered to the SBC 1. The PSP packet 500 becomes the body 520E of the I^2C packet 520 (586), for which the additional following value are assigned 586, 588, 590, 592, 594, 596 to the I^2C packet 520: the IOCB's hard-coded I^2C address 520A of 8, a sequence number 520 C of 79, an

I^2C Command 520D of 'D' signifying "Device" originated, creating the following I^2C packet 520:

[I^2C Address]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[8]	[16]	[79]	[D]	[PSP pkt]	[ETX]	[0xCCCC]

The device 3 waits 610 until the next poll received from the IOCB 200 and then sends 612 the I^2C packet 520 to the IOCB 200. Once the IOCB 200 receives the packet, it checks the "command" byte 520D of the I^2C packet and, if "Command" equals "D" (for device) 614, the IOCB 200 strips 616 the I^2C framing characters. The IOCB 200 then sends 618 the PSP packet 500, extracted from the body 520E of the I^2C packet 520, to the SBC 1 as:

[Virtual ID]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[41]	[9]	[214]	[I]	[0x0019]	[ETX]	[0xCCCC]

This PSP packet 500 is the original packet generated by the Device 3 (see 572, 574, 576, 578, 580,582, 584). The SBC 1, upon receiving a coin-in transaction, checks if there are any communication errors 622 and if the packet criteria is acceptable 620 (see Figure 6). If the packet is okay, the SBC 1 takes appropriate internal action. If the packet has been validly transmitted, the SBC 1 also sends an ACK transaction to the Device 3. First, the SBC 1 generates 624 the following PSP framed packet 500, assigning "Command" 520D the value of "ACK" 626, assigning other values 528, and sends 633 it to the IOCB 200 as:

[Virtual ID]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[41]	[7]	[214]	[ACK]	[0 bytes]	[ETX]	[0xCCCC]

Once again, the IOCB 200 encapsulates 634 this PSP packet 500 with its I^2C framing characters, looks up the I^2C address 520A associated with the Virtual ID 638, assigns the I^2C address 640, assigns the value of "M" to "Command" 520D, creating the following I^2C packet 520. The IOCB 200 then sends this I^2C packet 520 to the I^2C address 520A found in its device table for this Virtual ID 500A:

[I^2C Address]	[size]	[sequence #]	[Command]	[...body...]	[ETX]	[CRC-16]
[39]	[14]	[79]	['M']	[PSP pkt]	[ETX]	[0xCCCC]

Once the Device Board 4 detects the transmission 644 and ascertains that it was sent from the SBC 1 646, the Device Board 4 at the I^2C address 520A strips 648 the I^2C framing characters and decodes 650 the PSP packet 500 embedded in the body 520E the I^2C packet 520. The ACK Command confirms 652 to the Device 3 that the transaction was accepted by the SBC 1.

If appropriate, the SBC 1 generates another transaction 621 to be sent to the Device 3 responsible for updating the hard meter associated with this coin-in transaction.

Figure 6 depicts the preferred security devices of the invention which ensure the validity of the data transmissions. Each time a device 3 or the SBC 1 sends a transmission, upon receipt 670 of the packet, the receiving module (the Receiver) checks the I^2C packet 520 received to ensure validity of the received packet. First, the Receiver counts the number of characters received 672 and compares 674 this value to the "size" field of the received packet. If the packet passes this test, the Receiver then checks 676 if the "sequence number" of the received packet equals the receiver's next expected sequence number. If the packet passes this test, the Receiver checks if the ETX code is detected 678 at the correct index in the packet. Lastly, the Receiver calculates 680 the CRC-16 value of the received packet and checks 682 if the 16-bit CRC value received is equal to the calculated 16-bit CRC. If any of these tests fails, this indicates a communication failure and the Receiver creates a NAK packet 684, assigning "Command" a value of "NAK" 686, assigning the other values 688, and wraps 690 the PSP packet to notify the sender that the transmission failed. Upon receipt of the NAK packet, the sender reconstructs the packet and sends it again. If the receiver detects 698 more than three consecutive attempts which result in failure, the appropriate steps of error notification 700 are taken.

Should a transmitted packet fail any validation check, the receiver will send a Negative Acknowledge (NAK) packet 684, 686, 688 to the sender indicating an error. The sender will retry up to three times 698 before declaring a communication failure 700 causing the sender to re-enter the initialisation state 702 and attempt to re-establish a communication link (see Figure 6).

Figure 7 depicts the preferred device registration by the IOCB 200 of each device. Upon power up 710 or reset 702, the IOCB 200 assigns 712 the same default physical I^2C address 520A (77h) to each device 3. If registration of the device is not necessary, the IOCB 200 allocates 705 bus time for each device 3 to register. The IOCB 200 then dynamically assigns an I^2C address 707 which is clocked out to the responding slave device. The IOCB 200 sets up 709 a table 554 corresponding to this address and populates 709 the table with the data it has received from this slave device. A Virtual ID 711 is assigned to this device 3 which the SBC 1 will use to invoke a software driver pertaining to this device 3. The IOCB 200 creates 713 a PSP packet 500 (Virtual ID, a 'R'egister Command, and the device particulars 715, 717) and sends 719 the PSP Packet 500 to the SBC 1. The SBC 1 checks 729 if this device is on file 727, creates an ACKnowledging PSP Packet 733 with the value of "ACK" for the "Command" 735, checks if the device 3 needs configuration 736, adds configuration parameters to the "Body" of the packet 738, and sends 743 a Register ACKnowledge command to the IOCB 200 confirming this device is valid and commits the tabled entry as valid 737. Upon receipt of the acknowledgment 745, the IOCB 200 adds 747 this I^2C address 520A to its polling list 554. If this particular device 3 needs configuring 736, the SBC 1 embeds 738 the configuration parameters into the message body 500E of the Register ACKnowledge packet sent to the IOCB. The IOCB 200 resends 755 these parameters in the I^2C packet sent to the device 3.

If the SBC 1 has no information on this device 731, or this device 3 is not allowed in this machine 731, a Register NAK command is created 739, 741, and sent 743 to the IOCB 200 signifying registration denial of this device. Upon receiving a "NAK" packet 749, the IOCB 200 re-sends 755 the Register NAK command to the device and removes 751 the device's I^2C address 520A in the table list of devices. The Device Board 4 re-initialises 702, re-configures its address as 77h 712, and attempts to register 705. If after three attempts to register has failed 740, the SBC 1 displays an error message 742.

On initial power-up (Figure 7, 702, 710) or reset, each Device Board 4 is programmed as default I^2C address 77h (see Figure 7) and, upon successful registration, the device's I^2C address is added to the poll list 747. The IOCB 200, periodically polls I^2C address 77h for any response by following I^2C

standard protocols. The protocol mandates the IOCB 200 check if any device other than the IOCB 200 is asserting a clock 771. If not, the IOCB 200 sets the clock 773, raises the clock line 777, and, with the data line 252 high, lowers 775 the clock line 251 (start condition) thereby alerting all I^2C devices 3 on the bus (Slaves), that the IOCB 200 (Master) is sending an address byte. During registration, the IOCB 200 creates 783, 785, 787 and sends 791 an I^2C packet onto the data line. This packet has the address 77h "clocked out" on the data line, i.e., the first seven bits of the I^2C address have a value of 77h and the eighth bit signifies either the IOCB's intent to read a byte from this address or write a byte to this address.

The IOCB 200 expects an Acknowledge on the data line 719 (low condition) from a device 3 at this address. The Master will provide 831 the ninth clock for the expected ACK condition. Any devices 3 which do not have I^2C address 77h ignore 853 this prompt condition 791, 793. If a device or devices 3 are at this address 795 (there may be several devices with the default value 77h on initial power-up), the device 3 asserts 797, 801 the data line low at the appropriate time to respond to the IOCB 200. If the poll is intended for device registration (it is on initial poll), the Master sets the Read flag 785, 787 intending to read the responding device's registering data.

The I^2C protocol mandates a stop condition 833 (a terminating condition for each I^2C transmission) which the Master (IOCB 200 or SBC 1) will provide. Under I^2C mode, the Master of the I^2C line always provides the clock signal 773, even if a Read condition has been established and thus will read 829 the data as it provides the clock 773 by which the Slave sends the data.

If multiple devices respond 803 to the IOCB's 77h address byte, bus arbitration will come into play. As the IOCB provides each clock pulse 773, the devices will assert 811 their particular data bit onto the data line, but each device will first sample 807, 809 the data line to ensure its level is its intended level. If the level is not at the intended level, the particular device resets its I^2C commitment 813 753, and backs out of additional involvement 813 until it receives 815 the next address byte from the IOCB 200. As arbitration may continue for several bytes, the surviving device 3 will have finally completed its registration packet to the IOCB 200.

The IOCB maintains the device table 554 in preferably the following format:

- **Address** - Dynamically assigned I^2C address.
- **Device Type** - Type of device (i.e. Coin mech., Bill acceptor,).
- **Sub-Type** - Specific Make or Model.
- **Serial #** - Unique Board Identification Number (Board ID).
- **Status** - Last reported status.
- **Priority** - Polling priority.
- **Virtual ID** - SBC's end to end reference #.

As the IOCB 200 is continuously polling, the following example will detail a typical poll transaction to the device at I^2C address 39. The poll Command 520D 'Q'uery will have 0 bytes in the message body. The IOCB's next sequential transmission number 520C is 79 (the device 3 at this I^2C 520A address will be expecting this sequence number to be 79). The IOCB's I^2C address 520A is hardcoded at 8.

[I^2C Address]	[size]	[sequenced]	[Command]	[body]	[EXT]	[CRC-16]
[39]	[7]	[79]	['Q']	[O bytes]	[EXT]	[0xCCCC]

The IOCB 200, as Master of the I^2C bus, will clock out these data bytes to the Device Board 4 at this I^2C address 520A. The Write flag is set 789 in the packet's address byte 520A indicating the IOCB 200 is sending data. The device 3 ACKnowledges each byte by asserting 799, 801 the data line at the ACK clock bit time frame. When the IOCB 200 has sent all bytes 819, the Read flag is set 821 in the address byte, and the IOCB 200 re-sends 791 this byte. The IOCB 200, now expecting to receive data, will provide the clock to the Slave, but reads each clocked data line pulse 829 for 1 or 0, capturing 8-bits per byte.

[I^2C Address]	[size]	[sequenced #]	[Command]	[body]	[EXT]	[CRC-16]
[8]	[7]	[79]	[ACK]	[0 bytes]	[EXT]	[0xCCCC]

In this read mode, the Master must assert the ACK pulse 831 after each 8-bits 827 to tell the Slave that the Master has received this byte. When the Master has received a number of bytes equal to "size" 823, it does not assert the ACK pulse and generates the stop condition 833 signifying the transfer has completed. The IOCB 200 receiving an ACK Command 825 to the polling Query, states the device is active but has no information to send.

As the IOCB 200 is the only Master on the I^2C bus 250, the IOCB 200 detects 771 the clockline asserted by any other device attempting to control the bus 250 during the polling process. After a small time-out 835 in the event there is a possible spike or glitch on the bus, the IOCB 200 retests and, if this condition still exists, the IOCB 200 asserts the clock line low 839 disrupting and prevents any further I^2C communications on the bus 250. The IOCB 200 also reports this condition 841 to the SBC 1 for error reporting. After the shut down, the IOCB 200 periodically releases the clock line 843 and retests 845 for the above violation. If a retest shows the clock line is clear 847, the IOCB 200 broadcasts ReRegister Commands 849 to all I^2C devices 3 listed in the device table, requiring each to reset and reregister. The IOCB 200 also reports 851 the cleared condition to the SBC 1.

Due to the inherent security built in the preferred I^2C protocol, and the internal security checks designed into the IOCB 200 and SBC 1, as described above, it is virtually impossible for an alien device to invade the bus, or for a device to attempt communications with another device without the IOCB's knowledge and subsequent intervention.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

CLAIMS:

1. A secured inter-processor and virtual device digital communications system comprising:

a single board digital storage and central processing unit means;

communication port connection means to the single board digital storage and central processing unit means;

a bidirectional connection means between the single board digital storage and central processing unit means and the input/output connection board;

an input/output connection board;

communication port connection means to the input/output connection board;

a multi-drop capable communications protocol;

a Plug-N-Play communications protocol;

a multi-drop bidirectional communications-protocol capable connection means between the Input/Output Controller Board and connectable devices;

at least one connectable device; and

a central processing means for each connectable device.

2. The secured inter-processor and virtual device communications system of claim l, wherein the single board digital storage and central processing unit means is comprised of an industry standard main Central Processing Unit.

3. The secured inter-processor and virtual device communications system as claimed in claim l or 2, wherein the communication port connection means to the single board digital storage and central processing unit means is comprised of a parallel port.

4. The secured inter-processor and virtual device communications system as claimed in any one of claims 1, 2 or 3, wherein the bidirectional connection means between the single board digital storage and central processing unit means and the input/output connection board is comprised of an ISA bus.

5. The secured inter-processor and virtual device communications system of claim 4, wherein the ISA bus is a PC-104 bus.

6. The secured inter-processor and virtual device communications system as claimed in any one of claims 1, 2 or 3, wherein the bidirectional

connection means between the single board digital storage and central processing unit means and the input/output connection board is comprised of an 8-bit bus.

7. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 6, wherein the communication port connection means to the input/output connection board are comprised of two 8-bit parallel ports.

8. The secured inter-processor and virtual device communications system of claim 7, wherein the two 8-bit parallel ports are comprised of a slave means for data and control, and a means to transmit an 8-bit register.

9. The secured inter-processor and virtual device communications system of claim 8, wherein the 8-bit register, comprises:

a bit indicating whether the Input/Output Controller Board is ready to receive data;

a bit indicating whether the Input/Output Controller Board has aborted the reception of data sent by the main central processing unit;

a bit indicating whether the Input/Output Controller Board is ready to send data to the main central processing unit;

a bit indicating whether the Input/Output Controller Board has aborted the sending of data to the main central processing unit;

a bit indicating whether the Input/Output Controller Board is busy processing a critical application;

a bit indicating whether the handshaking bits of the system should be ignored; and

a bit indicating that the Input/Output Controller Board has been powered up or reset.

10. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 9, wherein the multi-drop capable communications protocol is comprised of a network-capable communications protocol.

11. The secured inter-processor and virtual device communications system of claim 10, wherein the multi-drop capable communications protocol is comprised of Philips' I^2C communications protocol.

12. The secured inter-processor and virtual device communications system of claim 10, wherein the multi-drop capable communications protocol is comprised of an RS-485 communications protocol.

13. The secured inter-processor and virtual device communications system of claim 10, wherein the multi-drop capable communications protocol is comprised of an USB communications protocol.

14. The secured inter-processor and virtual device communications system of claim 10, wherein the multi-drop capable communications protocol is comprised of a current loop communications protocol.

15. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 14, wherein the Plug-N-Play communications protocol is comprised of:

a device registering means;

a communications routing means;

a communications arbitrating means; and

a data transfer means.

16. The secured inter-processor and virtual device communications system of claim 15, wherein the Plug-N-Play communications protocol is further comprised of:

a device polling means; and

a communications security means.

17. The secured inter-processor and virtual device communications system of claim 15 or 16, wherein the device registering means is comprised of:

a means of determining the need for registering a device;

a means of assigning a default communication protocol address to a device;

a means of allocating bus time to register the device;

a means of assigning a communication protocol address to a device;

a means of obtaining information about a device;

a means of assigning a virtual address to the device;

a means of updating device parameters stored in a digital device table;

a means of transmitting the registration information to the digital storage and central processing unit;

a means of checking if the device is already registered;

a means of acknowledging the validity of the device registration;

a means of configuring the device;

a means of adding the registered device to the polling list; and

a means of deleting from the registered device to the polling list.

18. The secured inter-processor and virtual device communications system of claim 15, 16 or 17, the communications routing means is comprised of:

a means for acknowledging a physical connection;

a means for communicating with the main digital storage and central processing unit means;

a means for assigning a data packet with a specific protocol address;

a means for assigning a data packet with a unique virtual address;

a means for finding communications protocol addresses;

a means for finding virtual addresses;

a means for finding device information;

a means for wrapping a data packet with a communications protocol address;

a means for wrapping a data packet with information on the apparatus initiating the communication;

a means for wrapping a data packet with other information;

a means for unwrapping communications protocol address and information from a data packet;

a means for transferring data packets initiated by the single board digital storage and central processing unit; and

a means for transferring data packets initiated by the devices.

19. The secured inter-processor and virtual device communications system of claim 17, wherein the means for updating device parameters stored in a digital device table is comprised of:

a means for adding new device addresses and information;

means for deleting device addresses and information; and

a means for modifying device addresses and information.

20. The secured inter-processor and virtual device communications system of claims 17 or 19, wherein the means for updating device information stored in a digital device table is comprised of a means to add, delete, and modify the following information for each connected device:

a communications protocol address;

a device type;

a device make or model;

a device central processing unit serial number;

a last reported device status;

a device polling priority; and

a device virtual address.

21. The secured inter-processor and virtual device communications system of claim 18, the means of finding communications protocol addresses and finding device information is comprised of a means to locate virtual addresses in a digital device table.

22. The secured inter-processor and virtual device communications system of claim 18 or 19, wherein the means of finding virtual addresses and finding device information is comprised of a means to locate communications protocol addresses in a digital device table.

23. The secured inter-processor and virtual device communications system of claim 18, 21 or 22, wherein the means of wrapping a data packet with a communications protocol address and the means of wrapping a data packet with other information are comprised of a means of creating the data packet of claim 31 wherein the body of the wrapped data packet includes the entire unwrapped data packet of claim 32.

24. The secured inter-processor and virtual device communications system of claim 18, 21, 22 or 23, wherein the means of unwrapping communications protocol address and information from a data packet is comprised of a means of extracting the body of the wrapped data packet from the communications protocol wrapped data packet.

25. The secured inter-processor and virtual device communications system of claim 18, 21, 22, 23 or 24, wherein the transferring of data packets initiated by the single board digital storage and central processing unit is comprised of the steps of:

a means for sending the unwrapped data packet to a Input/Output Controller Board;

a means for retrieving the communications protocol address of the receiving device;

a means for directing the unwrapped data packet to the input/output controller board for transmission to the addressed device;

a means for determining the validity of the received wrapped data packet;

a means for reading the unwrapped data packet;

a means for acting upon the instructions in the unwrapped data packet; and

a means for sending a confirmatory data packet to the sending device.

26. The secured inter-processor and virtual device communications system of claim 18, 21, 22, 23 or 24, wherein the means of transferring of data packets initiated by devices is comprised of:

a means for detecting an action of a device;

a means for sending the wrapped data packet to a Input/Output Controller Board;

a means for sending a confirmatory data packet to the sending device;

a means for determining the validity of the received unwrapped data packet;

a means for reading the unwrapped data packet; and

a means for acting upon the instructions in the unwrapped data packet.

27. The secured inter-processor and virtual device communications system of claim 15, 21, 22, 23, 24, 25 or 26, wherein the communications arbitrating means is comprised of:

a means for providing a clock pulse;

a means for enabling devices to sample the data line to ensure the intended level;

a means for enabling devices to assert data onto the data line at the intended level;

a means for enabling devices to reset the protocol commitment; and

a means for enabling devices to wait to assert data onto the data line until receipt of a signal.

28. The secured inter-processor and virtual device communications system as claimed in any one of claims 17 or 19 to 27, wherein the means of polling devices for status is comprised of:

a means for providing the clock;

a means for ensuring that the clock is provided by only one source;

a means for indicating a poll to all devices;

a means for preparing to receive data from devices;

a means for indicating receive status;

a means for indicating a unique device address to be polled;

a means for creating a data packet containing the poll query;

a means for sending a data packet containing the poll query;

a means for devices to prepare to receive a polling data packet;

a means for determining whether the poll is directed to a device;

a means for determining that the device should respond to the poll;

a means for creating a responding data packet with device information to the poll;

a means for transferring a responding data packet to the poll;

a means for determining the type of response to the poll;

a means for determining the validity of the response data packet;

a means for updating device parameters stored in a digital device table; and

a means for acknowledging the success of the poll

29. The secured inter-processor and virtual device communications system as claimed in any one of claims 16 to 24, wherein the communications security means is comprised of:

a means for comparing the size of the data packet received with the size of the data packet sent;

a means for comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent;

a means for comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent;

a means for creating and comparing a unique identification number of the data packet sent with a code identifying the unique identification number of the data packet sent;

a means for sending a confirmatory data packet to the sending device.

30. The secured inter-processor and virtual device communications system of claim 29, wherein the means of comparing the size of the data packet received with the size of the data packet sent is comprised of a means of counting the size of the data packet received with a code identifying the size of the data packet sent.

31. The secured inter-processor and virtual device communications system of claim 29 or 30, wherein the means of comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent is comprised of comparing dual and matching sequential counters for the sending device and the receiving device.

32. The secured inter-processor and virtual device communications system of claim 29, 30 or 31, wherein the means for comparing the end of the data packet sent with a code identifying the location of the end of the data packet

sent is comprised of locating the end of the data packet sent and comparing the location with a code contained in the data packet identifying the location of the end of the data packet received.

33. The secured inter-processor and virtual device communications system of claim 29, 30, 31 or 32, wherein the means for creating and comparing a unique identification number of the data packet sent with a code in the data packet sent identifying the unique identification number of the data packet sent is comprised of a cyclic redundancy check algorithm.

34. The secured inter-processor and virtual device communications system of claim 33, wherein the cyclic redundancy check algorithm is comprised of a 16-bit reverse polynomial-based algorithm performed on each byte sent and on each byte received in a data packet..

35. The secured inter-processor and virtual device communications system of claims 25, 26 or 29, wherein the means for sending a confirmatory data packet to the sending device is comprised of:

a means for creating a confirmatory data packet;

a means for sending the confirmatory unwrapped data packet to the Input/Output Controller Board;

a means for wrapping the confirmatory data packet with framing characters to create a wrapped data packet;

a means for sending the confirmatory data packet to the sender;

a means for receiving the confirmatory data packet;

a means for stripping the wrapped confirmatory data packet; and

a means for reading the unwrapped confirmatory data packet.

36. The secured inter-processor and virtual device communications system in any one of claims 15 to 35, wherein the data transfer means is comprised of:

a communications protocol addressed wrapped data packet; and

a virtual device addressed unwrapped data packet.

37. The secured inter-processor and virtual device communications system of claim 36, wherein the communications protocol addressed wrapped data packet is comprised of a data packet with the following values:

a communications protocol address;

a size of the wrapped data packet;

a next sequence number for the wrapped data packet;

a type of sender of the data packet;

a entire unwrapped data packet;

a last position of the wrapped data packet; and

a unique 16-bit number for the wrapped data packet.

38. The secured inter-processor and virtual device communications system of claim 36 or 37, wherein the virtual device addressed unwrapped data packet is comprised of a data packet with the following values:

an assigned virtual address;

a size of the unwrapped data packet;

a next sequence number for the unwrapped data packet;

a type of sender of the unwrapped data packet;

a device command embodied in the unwrapped data packet;

a means of identifying the last position of the unwrapped data packet; and

a unique 16-bit number for the unwrapped data packet.

39. The secured inter-processor and virtual device communications system as claimed in claim 37 or 38, wherein the communications protocol address is comprised of an I^2C address.

40. The secured inter-processor and virtual device communications system of claims 37, 38 or 39, wherein the unique 16-bit number is a number created by a cyclic redundancy check algorithm.

41. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 40, wherein the multi-drop bidirectional communications-protocol capable connection means is comprised of:

a clocking means;

a data transfer means;

a system grounding means; and

a logic power means.

42. The secured inter-processor and virtual device communications system of claim 41, wherein the clocking means is comprised of a wire contained in a bus connecting the Input/Output Controller Board with each device.

43. The secured inter-processor and virtual device communications system of claim 41 or 42, wherein the data transfer means is comprised of a wire contained in a bus connecting the Input/Output Controller Board with each device.

44. The secured inter-processor and virtual device communications system of claim 41, 42 or 43, wherein the system grounding means is comprised of a

wire contained in a bus connecting the Input/Output Controller Board with each device.

45. The secured inter-processor and virtual device communications system of claim 41, 42, 43 or 44, wherein the logic power means is comprised of a wire contained in a bus connecting the Input/Output Controller Board with each device.

46. The secured inter-processor and virtual device communications system of claim 41, wherein the multi-drop communications-protocol capable bidirectional connection means is comprised of an I^2C bus.

47. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 40, wherein the connectable device is a gaming machine device.

48. The secured inter-processor and virtual device communications system as claimed in any one of claims 1 to 47, wherein the central processing means for each connectable device is connected to at least one device.

49. A method of creating a secured inter-processor and virtual device digital communications system comprising the steps of:

digitally and centrally storing and processing digital information in a single board digital storage and central processing means;

bidirectionally connecting the single board digital storage and central processing unit means and the input/output connection board;

digitally storing and processing digital information in an input/output connection board;

bidirectionally connecting at least one device to the input/output connection board in a multi-drop communications-protocol capable configuration;

having a multi-drop capable communications protocol;

providing a Plug-N-Play communications protocol;

providing at least one connectable device; and

providing a central processing means for each connectable device.

50. The method of creating a secured inter-processor and virtual device communications system of claim 49, wherein the step of bidirectionally connecting the single board digital storage and central processing unit means and the input/output connection board further comprises the step of providing an ISA bus.

51. The method of creating a secured inter-processor and virtual device communications system of claim 50, wherein the step of providing an ISA bus further comprises the step of providing a PC-104 bus.

52. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 49, 50 or 51, wherein the step of bidirectionally connecting the single board digital storage and central processing unit means and the input/output connection board further comprises providing an 8-bit bus.

53. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 49, 50, 51 or 52, wherein the step of bidirectionally connecting at least one device to the input/output connection board in a multi-drop configuration further comprises the step of providing an 8-bit register.

54. The method of creating a secured inter-processor and virtual device communications system of claim 53, wherein the step of providing an 8-bit register further comprises:

indicating whether the Input/Output Controller Board is ready to receive data;

indicating whether the Input/Output Controller Board has aborted the reception of data sent by the main central processing unit;

indicating whether the Input/Output Controller Board is ready to send data to the main central processing unit;

indicating whether the Input/Output Controller Board has aborted the sending of data to the main central processing unit;

indicating whether the Input/Output Controller Board is busy processing a critical application;

whether the handshaking bits of the system should be ignored; and

indicating whether the Input/Output Controller Board has been powered up or reset.

55. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 54, wherein the step of bidirectionally connecting at least one device to the input/output connection board in a multi-drop configuration further comprises the steps of:

providing a clocking means;

transferring data;

the system; and

powering the system.

56. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 54, wherein the step of having a multi-drop capable communications protocol further comprises the step of having a network-capable communications protocol.

57. The method of creating a secured inter-processor and virtual device communications system of claim 56, wherein the step of having a network-capable communications protocol further comprises the step of having a Philips' I^2C communications protocol.

58. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 57, wherein the step of providing a Plug-N-Play communications protocol further comprises the steps of:

registering devices;

routing communications;

arbitrating communications; and

transferring data.

59. The method of creating a secured inter-processor and virtual device communications system of claim 58, wherein the step of providing a Plug-N-Play communications protocol further comprises the steps of:

polling devices for status; and

providing a communications security means.

60. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 58 or 59, wherein the step of registering devices further comprises the steps of:

determining the need for registering a device;

assigning a default communication protocol address to a device;

allocating bus time to register the device;

assigning a communication protocol address to a device;

obtaining information about a device;

assigning a virtual address to the device;

updating device parameters stored in a digital device table;

transmitting the registration information to the digital storage and central processing unit;

checking if the device is already registered;

acknowledging the validity of the device registration;

configuring the device;

adding the registered device to the polling list; and

deleting from the registered device to the polling list.

61. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 58, 59 or 60, wherein the step of routing communications further comprises the steps of:

acknowledging a physical connection;

communicating with the main digital storage and central processing unit means;

assigning a data packet with a specific protocol address;

assigning a data packet with a unique virtual address;

finding communications protocol addresses;

a means for finding virtual addresses;

finding device information;

wrapping a data packet with a communications protocol address;

wrapping a data packet with information on the apparatus initiating the communication;

wrapping a data packet with other information;

unwrapping communications protocol address and information from a data packet;

transferring data packets initiated by the main digital storage and central processing unit; and

transferring data packets initiated by the devices.

62. The secured inter-processor and virtual device communications system as claimed in claim 60 or 61, wherein the step of updating device parameters stored in a digital device table further comprises the steps of:

adding new device addresses and information;

device addresses and information; and

modifying device addresses and information.

63. The method of creating a secured inter-processor and virtual device communications system as claimed in claims 60, 61 or 62, wherein the step of updating device parameters stored in a digital device table further comprises the steps of adding, deleting, and modifying the following information for each connected device:

a communications protocol address;

a device type;

a device make or model;

a device central processing unit serial number;

a last reported device status;

a device polling priority; and

a device virtual address.

64. The method of creating a secured inter-processor and virtual device communications system of claim 61, wherein the step of finding communications protocol addresses and finding device information further comprises the step of locating virtual addresses in a digital device table.

65. The method of creating a secured inter-processor and virtual device communications system of claim 61, wherein the step of finding virtual addresses and finding device information further comprises the step of locating communications protocol addresses in a digital device table.

66. The method of creating a secured inter-processor and virtual device communications system of claim 66, wherein the step of wrapping a data packet with a communications protocol address and the step of wrapping a data packet with other information further comprises the step of creating the data packet of claim 80 wherein the body of the wrapped data packet includes the entire unwrapped data packet of claim 81.

67. The method of creating a secured inter-processor and virtual device communications system of claim 61, wherein the step of unwrapping communications protocol address and information from a data packet further comprises the step of extracting the body of the wrapped data packet from the communications protocol wrapped data packet.

68. The method of creating a secured inter-processor and virtual device communications system of claim 61, wherein the step of transferring of data packets initiated by the main digital storage and central processing unit further comprises the steps of:

sending the unwrapped data packet to a Input/Output Controller Board;

retrieving the communications protocol address of the receiving device;

directing the unwrapped data packet to the input/output controller board for transmission to the addressed device;

determining the validity of the received wrapped data packet;

reading the unwrapped data packet;
acting upon the instructions in the unwrapped data packet; and
sending a confirmatory data packet to the sending device.

69. The method of creating a secured inter-processor and virtual device communications system of claim 61, wherein the step of transferring of data packets initiated by devices further comprises the steps of:
detecting an action of a device;
sending the wrapped data packet to a Input/Output Controller Board;
sending a confirmatory packet;
determining the validity of the received unwrapped data packet;
reading the unwrapped data packet; and
acting upon the instructions in the unwrapped data packet.

70. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 58 to 69, wherein the step of arbitrating communications further comprises the steps of:
providing a clock pulse;
enabling devices to sample the data line to ensure the intended level;
enabling devices to assert data onto the data line at the intended level:
enabling devices to reset the protocol commitment; and
enabling devices to wait to assert data onto the data line until receipt of a signal.

71. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 59 to 70, wherein the step of polling devices for status further comprises the steps of:
providing the clock;
ensuring that the clock is provided by only one source;
indicating a poll to all devices;
preparing to receive data from devices;
indicating receive status;
indicating a unique device address to be polled;
creating a data packet containing the poll query;
sending a data packet containing the poll query;
devices preparing to receive a polling data packet;
determining whether the poll is directed to a device;
determining that the device should respond to the poll;
creating a responding data packet with device information to the poll;

a means of transferring a responding data packet to the poll;

determining the type of response to the poll;

determining the validity of the response data packet;

updating device parameters stored in a digital device table; and

acknowledging the success of the poll

72. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 59 to 71, wherein the step of providing a communications security means further comprises the steps of:

comparing the size of the data packet received with the size of the data packet sent;

comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent;

comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent;

creating and comparing a unique identification number of the data packet sent with a code identifying the unique identification number of the data packet sent; and

sending a confirmatory data packet to the sending device.

73. The method of creating a secured inter-processor and virtual device communications system of claim 72, wherein the step of comparing the size of the data packet received with the size of the data packet sent further comprises the step of counting the size of the data packet received with a code identifying the size of the data packet sent.

74. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 72 or 73, wherein the step of comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent further comprises the step of comparing dual and matching sequential counters for the sending device and the receiving device.

75. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 72, 73 or 74, wherein the step of comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent further comprises the steps of locating the end of the data packet sent and comparing the location with a

code contained in the data packet identifying the location of the end of the data packet received.

76. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 72, 74 or 75, wherein the step of creating and comparing a unique identification number of the data packet sent with a code in the data packet sent identifying the unique identification number of the data packet sent further comprises the step of performing a cyclic redundancy check algorithm.

77. The method of creating a secured inter-processor and virtual device communications system of claim 76, wherein the step of performing a cyclic redundancy check algorithm further comprises the step of performing a 16-bit reverse polynomial-based algorithm on each byte sent and on each byte received in a data packet.

78. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 68 or 69, wherein the step of sending a confirmatory data packet to the sending device further comprises the steps of:

creating a confirmatory data packet;

sending the confirmatory unwrapped data packet to the Input/Output Controller Board;

wrapping the confirmatory data packet with framing characters to create a wrapped data packet;

sending the confirmatory data packet to the sender;

receiving the confirmatory data packet;

stripping the wrapped confirmatory data packet; and

reading the unwrapped confirmatory data packet.

79. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 58 to 78, wherein the step of transferring data further comprises the steps of:

providing a communications protocol addressed wrapped data packet; and

providing a virtual device addressed unwrapped data packet.

80. The method of creating a secured inter-processor and virtual device communications system of claim 79, wherein the step of providing a communications protocol addressed wrapped data packet further comprises the step of creating a data packet with the following values:

a communications protocol address;
a size of the wrapped data packet;
a next sequence number for the wrapped data packet;
a type of sender of the data packet;
an entire unwrapped data packet;
a last position of the wrapped data packet; and
a unique 16-bit number for the wrapped data packet.

81. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 79 or 80, wherein the step of providing a virtual addressed unwrapped data packet further comprises the step of creating a data packet with the following values:

an assigned virtual address;
a size of the unwrapped data packet;
a next sequence number for the unwrapped data packet;
a type of sender of the unwrapped data packet;
a device command embodied in the unwrapped data packet;
a means for identifying the last position of the unwrapped data packet; and
a unique 16-bit number for the unwrapped data packet.

82. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 80 or 81, wherein the step of creating a communications protocol address further comprises the step of providing an I^2C address.

83. The method of creating a secured inter-processor and virtual device communications system as claimed in claim 80, 81 or 82, wherein the step of creating a unique 16-bit number for the data packet further comprises the step of applying a cyclic redundancy check algorithm.

84. The method of creating a secured inter-processor and virtual device communications system of claim 55, wherein the step of providing a clocking means further comprises the step of including a wire in a bus connecting the Input/Output Controller Board with each device.

85. The method of creating a secured inter-processor and virtual device communications system of claim 55 or 84, wherein the step of transferring data further comprises the step of including a wire in a bus connecting the Input/Output Controller Board with each device.

86. The method of creating a secured inter-processor and virtual device communications system of claim 55, 84 or 85, wherein the step of grounding the system further comprises the step of including a wire in a bus connecting the Input/Output Controller Board with each device.

87. The method of creating a secured inter-processor and virtual device communications system of claim 55, 84, 85 or 86, wherein the step of powering the system further comprises the step of providing a wire in a bus connecting the Input/Output Controller Board with each device.

88. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 87, wherein the step of bidirectionally connecting at least one device to the input/output connection board in a multi-drop connection further comprises the step of containing the clocking means, the data transfer means, the system ground means, and the logic power means in the multi-drop connection.

89. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 88, wherein the step of providing at least one connectable device further comprises the step of providing gaming machine devices.

90. The method of creating a secured inter-processor and virtual device communications system as claimed in any one of claims 49 to 89, wherein the step of providing a central processing means for each connectable device further comprises the step of connecting each central processing means to at least one device.

91. A Plug-N-Play communications protocol for devices comprising:

a device registering means;

a communications routing means;

a communications arbitrating means; and

a data transfer means.

92. The Plug-N-Play communications protocol of claim 91, wherein the Plug-N-Play communications protocol is further comprised of:

a device polling means; and

a communications security means.

93. The Plug-N-Play communications protocol of claim 91 or 92, wherein the device registering means is comprised of:

a means of determining the need for registering a device;

a means of assigning a default communication protocol address to a device;

a means of allocating bus time to register the device;

a means of assigning a communication protocol address to a device;

a means of obtaining information about a device;

a means of assigning a virtual address to the device;

a means of updating device parameters stored in a digital device table;

a means of transmitting the registration information to the digital storage and central processing unit;

a means of checking if the device is already registered;

a means of acknowledging the validity of the device registration;

a means of configuring the device;

a means of adding the registered device to the polling list; and

a means of deleting from the registered device to the polling list.

94. The Plug-N-Play communications protocol of claim 91, 92 or 93, wherein the communications routing means is comprised of:

a means for acknowledging a physical connection;

a means for communicating with the main digital storage and central processing unit means;

a means for assigning a data packet with a specific protocol address;

a means for assigning a data packet with a unique virtual address;

a means for finding communications protocol addresses;

a means for finding virtual addresses;

a means for finding device information;

a means for wrapping a data packet with a communications protocol address;

a means for wrapping a data packet with information on the apparatus initiating the communication;

a means for wrapping a data packet with other information;

a means for unwrapping communications protocol address and information from a data packet;

a means for transferring data packets initiated by the main digital storage and central processing unit; and

a means for transferring data packets initiated by the devices.

95. The Plug-N-Play communications protocol of claim 93 or 94, wherein the means for updating device parameters stored in a digital device table is comprised of:

a means for adding new device addresses and information;

a means for deleting device addresses and information; and

a means for modifying device addresses and information.

96. The Plug-N-Play communications protocol of claims 93, 94 or 95, wherein the means for updating device parameters stored in a digital device table is comprised of a means to add, delete, and modify the following information for each connected device:

a communications protocol address;

a device type;

a device make or model;

a device central processing unit serial number;

a last reported device status;

a device polling priority; and

a device virtual address.

97. The Plug-N-Play communications protocol of claim 94, wherein the means of finding protocol addresses and finding device information is comprised of a means to locate virtual addresses in a digital device table.

98. The Plug-N-Play communications protocol of claim 94, wherein the means of finding virtual addresses and finding device information is comprised of a means to locate communications protocol addresses in a digital device table.

99. The Plug-N-Play communications protocol of claim 94, wherein the means of wrapping a data packet with a communications protocol address and the means of wrapping a data packet with other information are comprised of a means for creating the data packet, wherein the body of the wrapped data packet includes the entire unwrapped data packet.

100. The Plug-N-Play communications protocol of claim 94, wherein the means of unwrapping communications protocol address and information from a data packet is comprised of a means of extracting the body of the wrapped data packet from the communications protocol wrapped data packet.

101. The Plug-N-Play communications protocol of claim 94, wherein the transferring of data packets initiated by the main digital storage and central processing unit is comprised of the steps of:

a means for sending the unwrapped data packet to an Input/Output Controller Board;

a means for retrieving the communications protocol address of the receiving device;

a means for directing the unwrapped data packet to the input/output controller board for transmission to the addressed device;

a means for determining the validity of the received wrapped data packet;

a means for reading the unwrapped data packet;

a means for acting upon the instructions in the unwrapped data packet; and

a means for sending a confirmatory data packet to the sending device.

102. The Plug-N-Play communications protocol of claim 94, wherein the means of transferring of data packets initiated by devices is comprised of:

a means for detecting an action of a device;

a means for sending the wrapped data packet to an Input/Output Controller Board;

a means for sending a confirmatory data packet to the sending device;

a means for determining the validity of the received unwrapped data packet;

a means for reading the unwrapped data packet; and

a means for acting upon the instructions in the unwrapped data packet.

103. The Plug-N-Play communications protocol as claimed in any one of claims 91 to 102, wherein the communications arbitrating means is comprised of:

a means for providing a clock pulse;

a means for enabling devices to sample the data line to ensure the intended level;

a means for enabling devices to assert data onto the data line at the intended level;

a means for enabling devices to reset the protocol commitment; and

a means for enabling devices to wait to assert data onto the data line until receipt of a signal.

104. The Plug-N-Play communications protocol of claim 92, wherein the means of polling devices for status is comprised of:

a means for providing the clock;

a means for ensuring that the clock is provided by only one source;

a means for indicating a poll to all devices;

a means for preparing to receive data from devices;

a means for indicating receive status;

a means for indicating a unique device address to be polled;

a means for creating a data packet containing the poll query;

a means for sending a data packet containing the poll query;

a means for devices to prepare to receive a polling data packet;

a means for determining whether the poll is directed to a device;

a means for determining that the device should respond to the poll;

a means for creating a responding data packet with device information to the poll;

a means for transferring a responding data packet to the poll;

a means for determining the type of response to the poll;

a means for determining the validity of the response data packet;

a means for updating device parameters stored in a digital device table; and

a means for acknowledging the success of the poll.

105. The Plug-N-Play communications protocol of claim 92 or 104, wherein the communications security means is comprised of:

a means for comparing the size of the data packet received with the size of the data packet sent;

a means for comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent;

a means for comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent;

a means for creating and comparing a unique identification number of the data packet sent with a code identifying the unique identification number of the data packet sent; and

a means for sending a confirmatory data packet to the sending device.

106. The Plug-N-Play communications protocol of claim 105, wherein the means of comparing the size of the data packet received with the size of the

data packet sent is comprised of a means of counting the size of the data packet received with a code identifying the size of the data packet sent.

107. The Plug-N-Play communications protocol of claim 105 or 106, wherein the means of comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent is comprised of comparing dual and matching sequential counters for the sending device and the receiving device.

108. The Plug-N-Play communications protocol of claim 105, 016 or 107, wherein the means for comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent is comprised of locating the end of the data packet sent and comparing the location with a code contained in the data packet identifying the location of the end of the data packet received.

109. The Plug-N-Play communications protocol of claim 105, 106, 107 or 108, wherein the means for creating and comparing unique identification number of the data packet sent with a code in the data packet sent identifying the unique identification number of the data packet sent is comprised of a cyclic redundancy check algorithm.

110. The Plug-N-Play communications protocol of claim 109, wherein the cyclic redundancy check algorithm is comprised of a 16-bit reverse polynomial-based algorithm performed on each byte sent and on each byte received in a data packet.

111. The Plug-N-Play communications protocol of claim 101, wherein the means for sending a confirmatory data packet to the sending device is comprised of:

a means for creating a confirmatory data packet;

a means for sending the confirmatory unwrapped data packet to the Input/Output Controller Board;

a means for wrapping the confirmatory data packet with framing characters to create a wrapped data packet;

a means for sending the confirmatory data packet to the sender;

a means for receiving the confirmatory data packet;

a means for stripping the wrapped confirmatory data packet; and

a means for reading the unwrapped confirmatory data packet.

112. The Plug-N-Play communications protocol as claimed in any one of claims 91 to 111, wherein the data transfer means is comprised of:

a communications protocol addressed wrapped data packet; and

a virtual device addressed unwrapped data packet.

113. The Plug-N-Play communications protocol of claim 112, wherein the communications protocol addressed wrapped data packet is comprised of a data packet with the following values:

a communications protocol address;

a size of the wrapped data packet;

a next sequence number for the wrapped data packet;

a type of sender of the data packet;

a entire unwrapped data packet;

a last position of the wrapped data packet; and

a unique 16-bit number for the wrapped data packet.

114. The Plug-N-Play communications protocol of claim 113, wherein the virtual device addressed unwrapped data packet is comprised of a data packet with the following values:

an assigned virtual address;

a size of the unwrapped data packet;

a next sequence number for the unwrapped data packet;

a type of sender of the unwrapped data packet;

a device command embodied in the unwrapped data packet;

a means of identifying the last position of the unwrapped data packet; and

a unique 16-bit number for the unwrapped data packet.

115. The Plug-N-Play communications protocol of claim 113 or 114, wherein the communications protocol address is comprised of an I^2C address.

116. The Plug-N-Play communications protocol of claims 113, 114 or 115, wherein the unique 16-bit number is a number created by a cyclic redundancy check algorithm.

117. A method of creating a Plug-N-Play communications protocol for devices comprising the steps of:

registering devices;

routing communications;

arbitrating communications; and

transferring data.

118. The method of creating a Plug-N-Play communications protocol for devices of claim 117, wherein the method of creating a Plug-N-Play communications protocol further comprises the steps of:

polling devices for status; and

providing a communications security means.

119. The method of creating a Plug-N-Play communications protocol for devices of claim 117 or 118, wherein the step of registering devices further comprises the steps of:

determining the need for registering a device;

assigning a default communication protocol address to a device;

allocating bus time to register the device;

assigning a communication protocol address to a device;

obtaining information about a device;

assigning a virtual address to the device;

updating device parameters stored in a digital device table;

transmitting the registration information to the digital storage and central processing unit;

checking if the device is already registered;

acknowledging the validity of the device registration;

configuring the device;

the registered device to the polling list; and

deleting from the registered device to the polling list.

120. The method of creating a Plug-N-Play communications protocol for devices of claim 117, 118 or 119, wherein the step of routing communications further comprises the steps of:

acknowledging a physical connection;

communicating with the main digital storage and central processing unit means;

assigning a data packet with a specific protocol address;

assigning a data packet with a unique virtual address;

finding communications protocol addresses;

finding virtual addresses;

finding device information;

wrapping a data packet with a communications protocol address;

wrapping a data packet with information on the apparatus initiating the communication;

wrapping a data packet with other information;

unwrapping communications protocol address and information from a data packet;

transferring data packets initiated by the main digital storage and central processing unit; and

transferring data packets initiated by the devices.

121. The method of creating a Plug-N-Play communications protocol for devices of claims 119 or 120, wherein the step of updating device parameters stored in a digital device table further comprises the steps of:

adding new device addresses and information;

deleting device addresses and information; and

modifying device addresses and information.

122. The method of creating a Plug-N-Play communications protocol for devices of claims 119, 120 or 121, wherein the step of updating device parameters stored in a digital device table further comprises the steps of adding, deleting, and modifying the following information for each connected device:

a communications protocol address;

a device type;

a device make or model;

a device central processing unit serial number;

a last reported device status;

a device polling priority; and

a device virtual address.

123. The method of creating a Plug-N-Play communications protocol for devices of claim 120, 121 or 122, wherein the step of finding communications protocol addresses and finding device information further comprises the step of locating virtual addresses in a digital device table.

124. The method of creating a Plug-N-Play communications protocol for devices of claim 120, 121, 122 or 123, wherein the step of finding virtual addresses and finding device information further comprises the step of locating communications protocol addresses in a digital device table.

125. The method of creating a Plug-N-Play communications protocol for devices of claim 120, 121, 122, 123 or 124, wherein the step of wrapping a data packet with a communications protocol address and the step of wrapping a data packet with other information further comprises the step of

creating the data packet of claim 139 wherein the body of the wrapped data packet includes the entire unwrapped data packet of claim 141.

126. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 120 to 125, wherein the step of unwrapping communications protocol address and information from a data packet further comprises the step of extracting the body of the wrapped data packet from the communications protocol wrapped data packet.

127. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 120 to 126, wherein the step of transferring of data packets initiated by the main digital storage and central processing unit further comprises the steps of:

sending the unwrapped data packet to a Input/Output Controller Board;

retrieving the communications protocol address of the receiving device;

directing the unwrapped data packet to the input/output controller board for transmission to the addressed device;

determining the validity of the received wrapped data packet;

reading the unwrapped data packet;

acting upon the instructions in the unwrapped data packet; and

sending a confirmatory data packet to the sending device.

128. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 120 to 127, wherein the step of transferring of data packets initiated by devices further comprises the steps of:

detecting an action of a device;

sending the wrapped data packet to a Input/Output Controller Board;

sending a confirmatory packet;

determining the validity of the received unwrapped data packet;

reading the unwrapped data packet; and

acting upon the instructions in the unwrapped data packet.

129. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 117 to 128, wherein the step of arbitrating communications further comprises the steps of:

providing a clock pulse;

enabling devices to sample the data line to ensure the intended level;

enabling devices to assert data onto the data line at the intended level;

enabling devices to reset the protocol commitment; and

enabling devices to wait to assert data onto the data line until receipt of a signal.

130. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 119 to 128, wherein the step of polling devices for status further comprises the steps of:

providing the clock;

ensuring that the clock is provided by only one source;

indicating a poll to all devices;

preparing to receive data from devices;

indicating receive status;

indicating a unique device address to be polled;

creating a data packet containing the poll query;

sending a data packet containing the poll query;

devices preparing to receive a polling data packet;

determining whether the poll is directed to a device;

determining that the device should respond to the poll;

creating a responding data packet with device information to the poll;

a means of transferring a responding data packet to the poll;

determining the type of response to the poll;

determining the validity of the response data packet;

updating device parameters stored in a digital device table; and

acknowledging the success of the poll

131. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 119 to 128 or claim 130, wherein the step of providing a communications security means further comprises the steps of:

comparing the size of the data packet received with the size of the data packet sent;

comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent;

comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent;

creating and comparing a unique identification number of the data packet sent with a code identifying the unique identification number of the data packet sent; and

sending a confirmatory data packet to the sending device.

132. The method of creating a Plug-N-Play communications protocol for devices of claim 131, wherein the step of comparing the size of the data packet received with the size of the data packet sent further comprises the step of counting the size of the data packet received with a code identifying the size of the data packet sent.

133. The method of creating a Plug-N-Play communications protocol for devices of claim 131 or 132, wherein the step of comparing an expected sequential identification number of the packet received with the sequential identification number of the data packet sent further comprises the step of comparing dual and matching sequential counters for the sending device and the receiving device.

134. The method of creating a Plug-N-Play communications protocol for devices of claim 131, 132 or 133, wherein the step of comparing the end of the data packet sent with a code identifying the location of the end of the data packet sent further comprises the steps of locating the end of the data packet sent and comparing the location with a code contained in the data packet identifying the location of the end of the data packet received.

135. The method of creating a Plug-N-Play communications protocol for devices of claim 131, 132, 133 or 134, wherein the step of creating and comparing a unique identification number of the data packet sent with a code in the data packet sent identifying the unique identification number of the data packet sent further comprises the step of performing a cyclic redundancy check algorithm.

136. The method of creating a Plug-N-Play communications protocol for devices of claim 135, wherein the step of performing a cyclic redundancy check algorithm further comprises the step of performing a 16-bit reverse polynomial-based algorithm on each byte sent and on each byte received in a data packet.

137. The method of creating a Plug-N-Play communications protocol for devices of claim 131 , wherein the step of sending a confirmatory data packet to the sending device further comprises the steps of:

creating a confirmatory data packet;

sending the confirmatory unwrapped data packet to the Input/Output Controller Board;

wrapping the confirmatory data packet with framing characters to create a wrapped data packet;

sending the confirmatory data packet to the sender;

receiving the confirmatory data packet;

stripping the wrapped confirmatory data packet; and

reading the unwrapped confirmatory data packet.

138. The method of creating a Plug-N-Play communications protocol for devices as claimed in any one of claims 117 to 137, wherein the step of transferring data further comprises the steps of:

providing a communications protocol addressed wrapped data packet; and

providing a virtual device addressed unwrapped data packet.

139. The method of creating a secured inter-processor and virtual device communications system of claim 138, wherein the step of providing a communications protocol addressed wrapped data packet further comprises the step of creating a data packet with the following values:

a communications protocol address;

a size of the wrapped data packet;

a next sequence number for the wrapped data packet;

a type of sender of the data packet;

an entire unwrapped data packet;

a last position of the wrapped data packet; and

a unique 16-bit number for the wrapped data packet.

140. The method of creating a Plug-N-Play communications protocol for devices of claim 139, wherein the step of creating a communications protocol address further comprises the step of providing an I^2C address.

141. The method of creating a Plug-N-Play communications protocol for devices of claim 138, 139 or 140, wherein the step of providing a virtual addressed unwrapped data packet further comprises the step of creating a data packet with the following values:

an assigned virtual address;

a size of the unwrapped data packet;

a next sequence number for the unwrapped data packet;

a type of sender of the unwrapped data packet;

a device command embodied in the unwrapped data packet;

means for identifying the last position of the unwrapped data packet; and

a unique 16-bit number for the unwrapped data packet.

142. The method of creating a Plug-N-Play communications protocol for devices of claims 139 and 140 or 141, wherein the step of creating a unique 16-bit number for the data packet further comprises the step of applying a cyclic redundancy check algorithm.

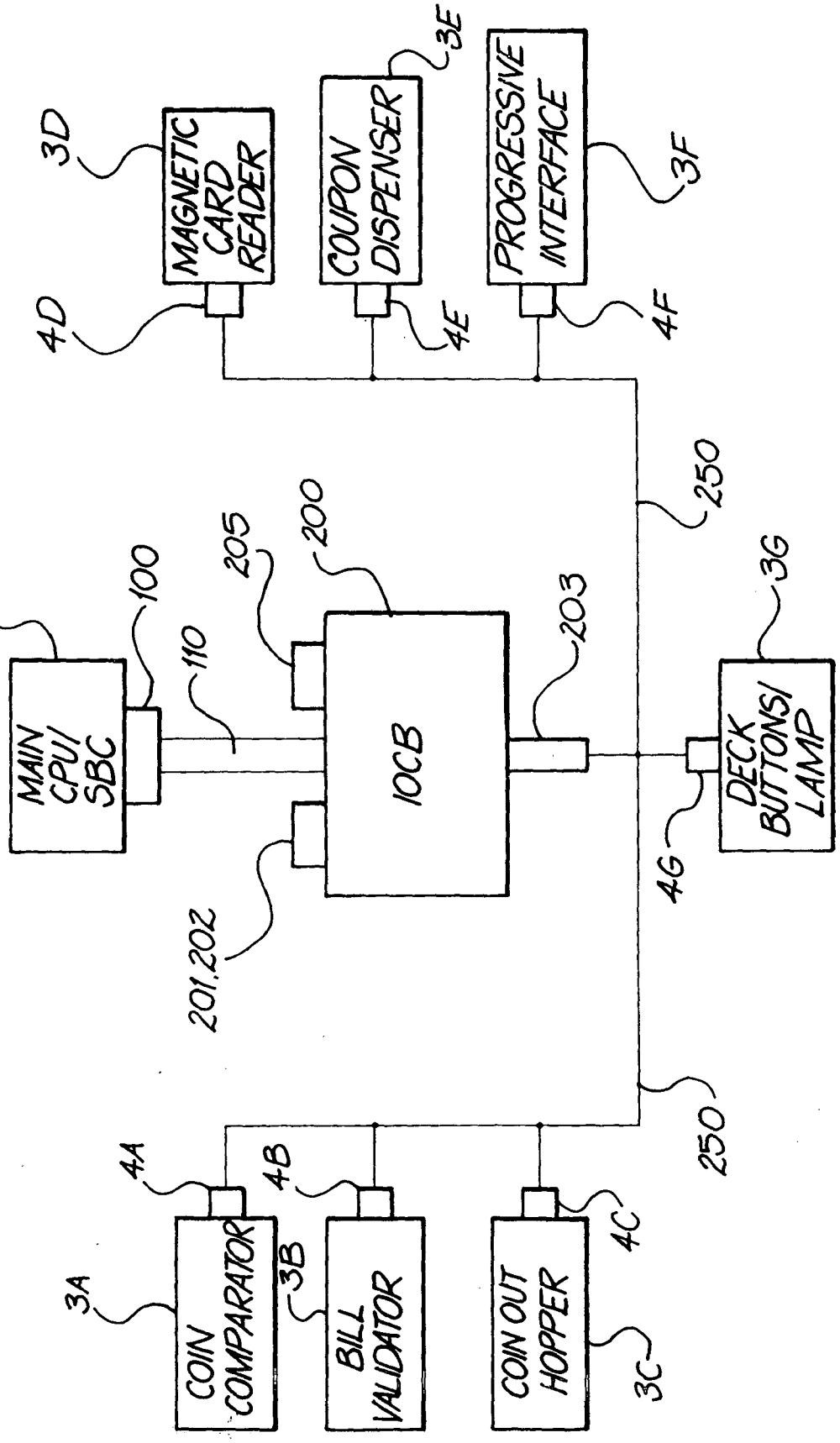

MAGNETIC CARD READER
COUPON DISPENSER
PROGRESSIVE INTERFACE
MAIN CPU/ SBC
IOCB
DECK BUTTONS/ LAMP
COIN COMPARATOR
BILL VALIDATOR
COIN OUT HOPPER
3D
3E
3F
4D
4E
4F
250
1
100
110
205
200
203
3G
4G
201,202
4A
4B
4C
3A
3B
3C
250

FIG. 1

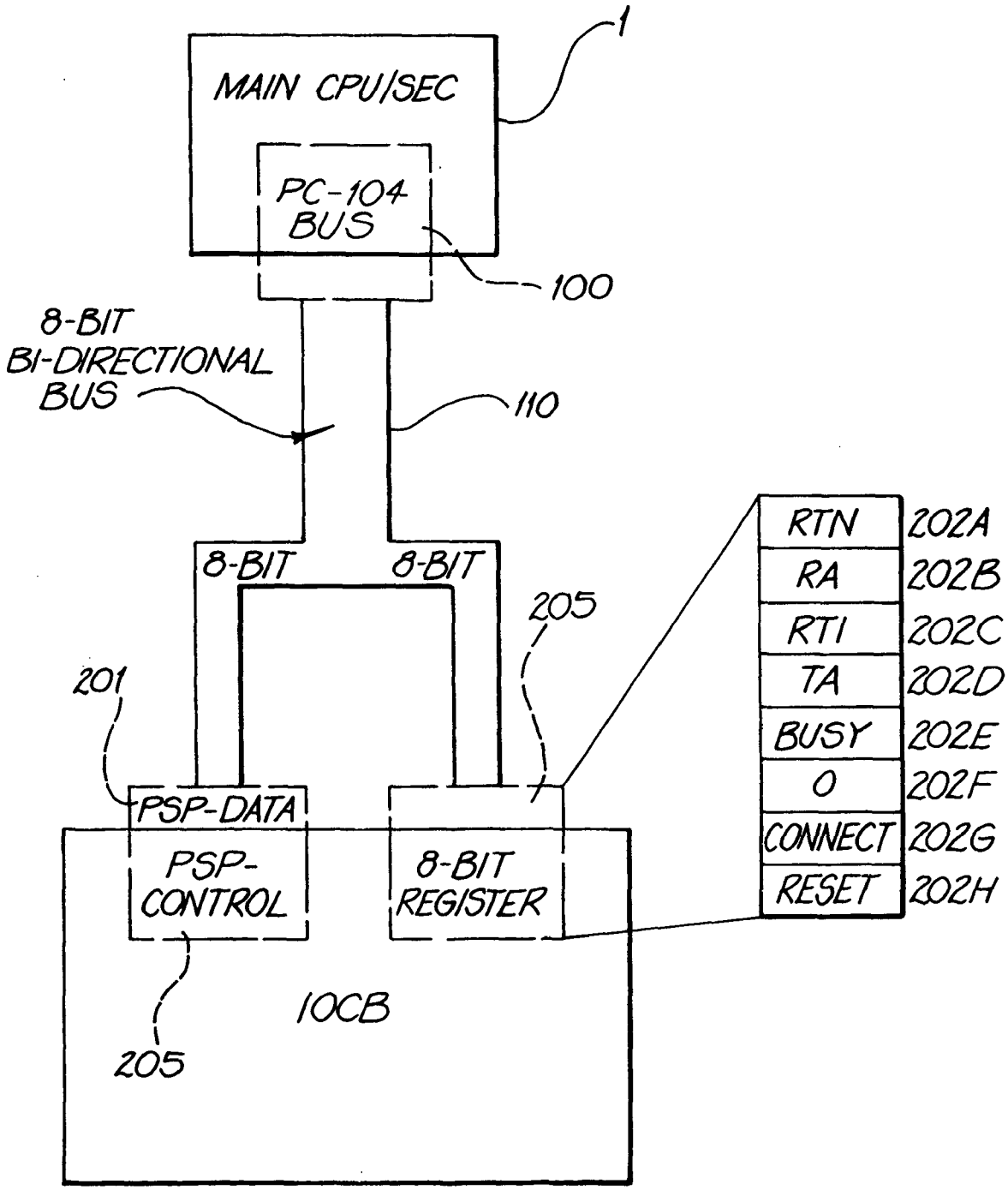

1
MAIN CPU/SEC
PC-104 BUS
100
8-BIT BI-DIRECTIONAL BUS
110
8-BIT
8-BIT
205
201
PSP-DATA
PSP-CONTROL
8-BIT REGISTER
205
10CB
RTN
RA
RTI
TA
BUSY
0
CONNECT
RESET
202A
202B
202C
202D
202E
202F
202G
202H

FIG. 2

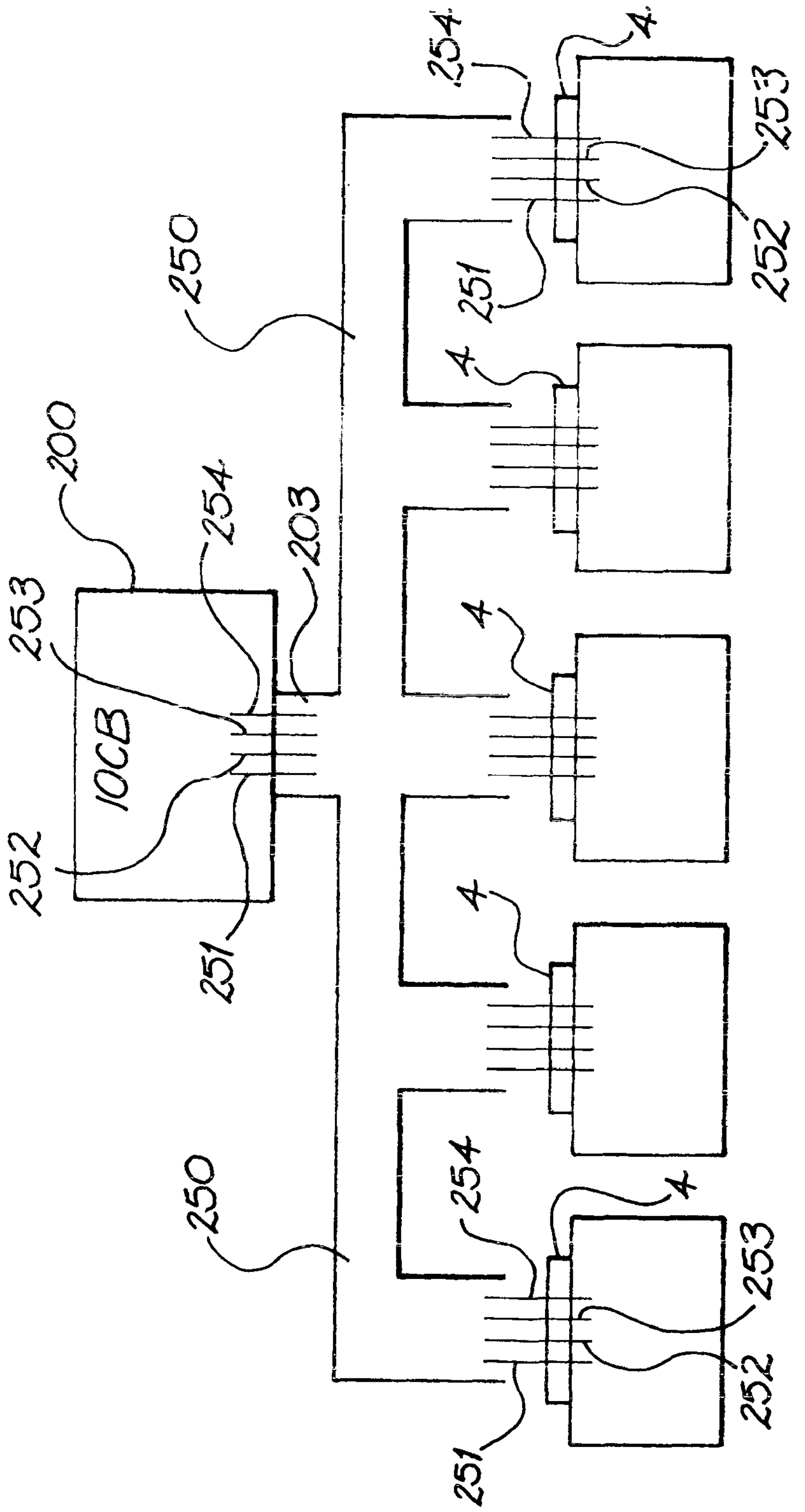


FIG. 3

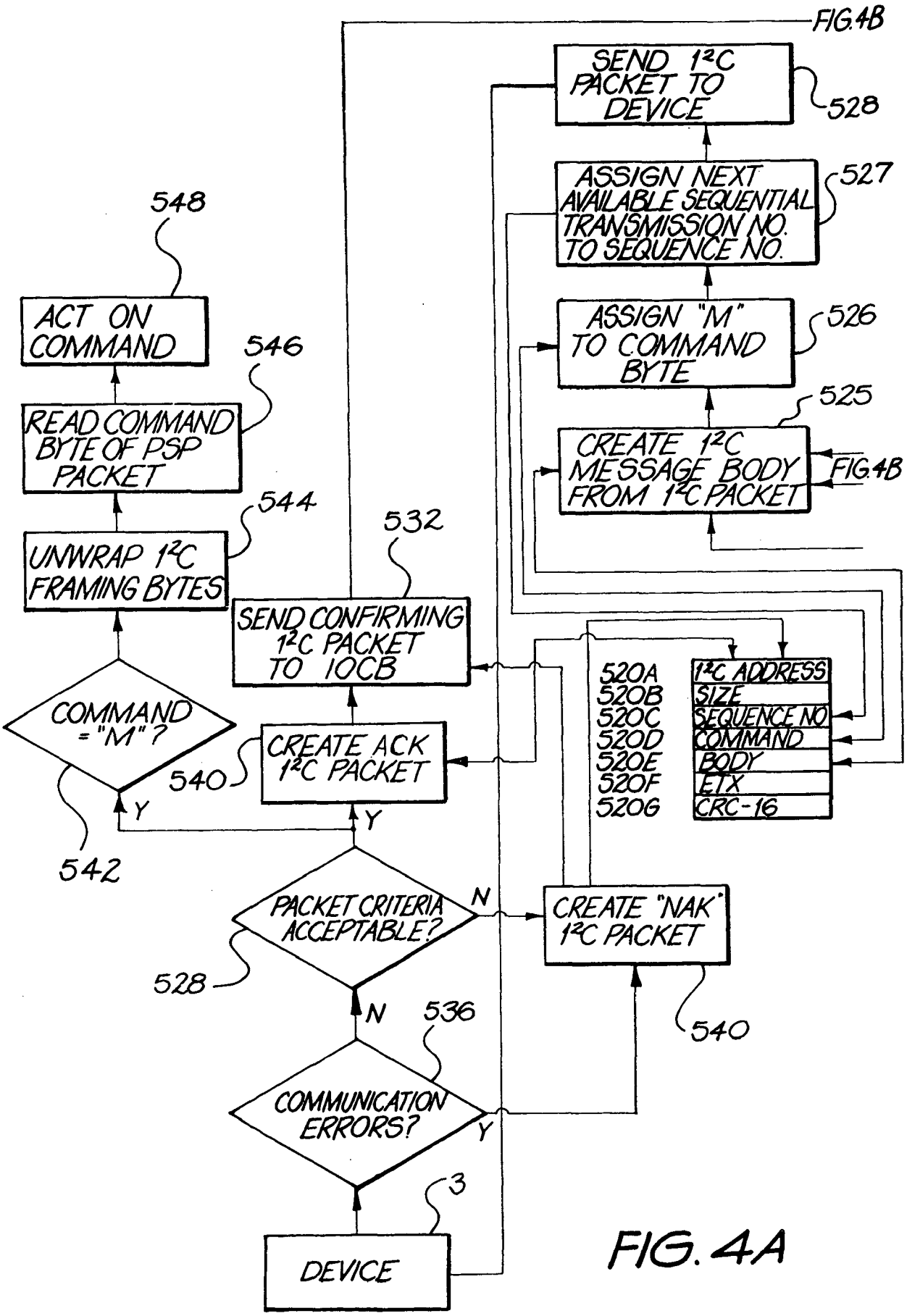


FIG.4B
SEND I²C PACKET TO DEVICE
528
ASSIGN NEXT AVAILABLE SEQUENTIAL TRANSMISSION NO. TO SEQUENCE NO.
527
ASSIGN "M" TO COMMAND BYTE
526
CREATE I²C MESSAGE BODY FROM I²C PACKET
525
FIG.4B
548
ACT ON COMMAND
546
READ COMMAND BYTE OF PSP PACKET
544
UNWRAP I²C FRAMING BYTES
532
SEND CONFIRMING I²C PACKET TO IOCB
520A I²C ADDRESS
520B SIZE
520C SEQUENCE NO
520D COMMAND
520E BODY
520F ETX
520G CRC-16
COMMAND = "M"?
CREATE ACK I²C PACKET
540
Y
542
PACKET CRITERIA ACCEPTABLE?
N
CREATE "NAK" I²C PACKET
528
540
N
536
COMMUNICATION ERRORS?
Y
3
DEVICE

FIG. 4A

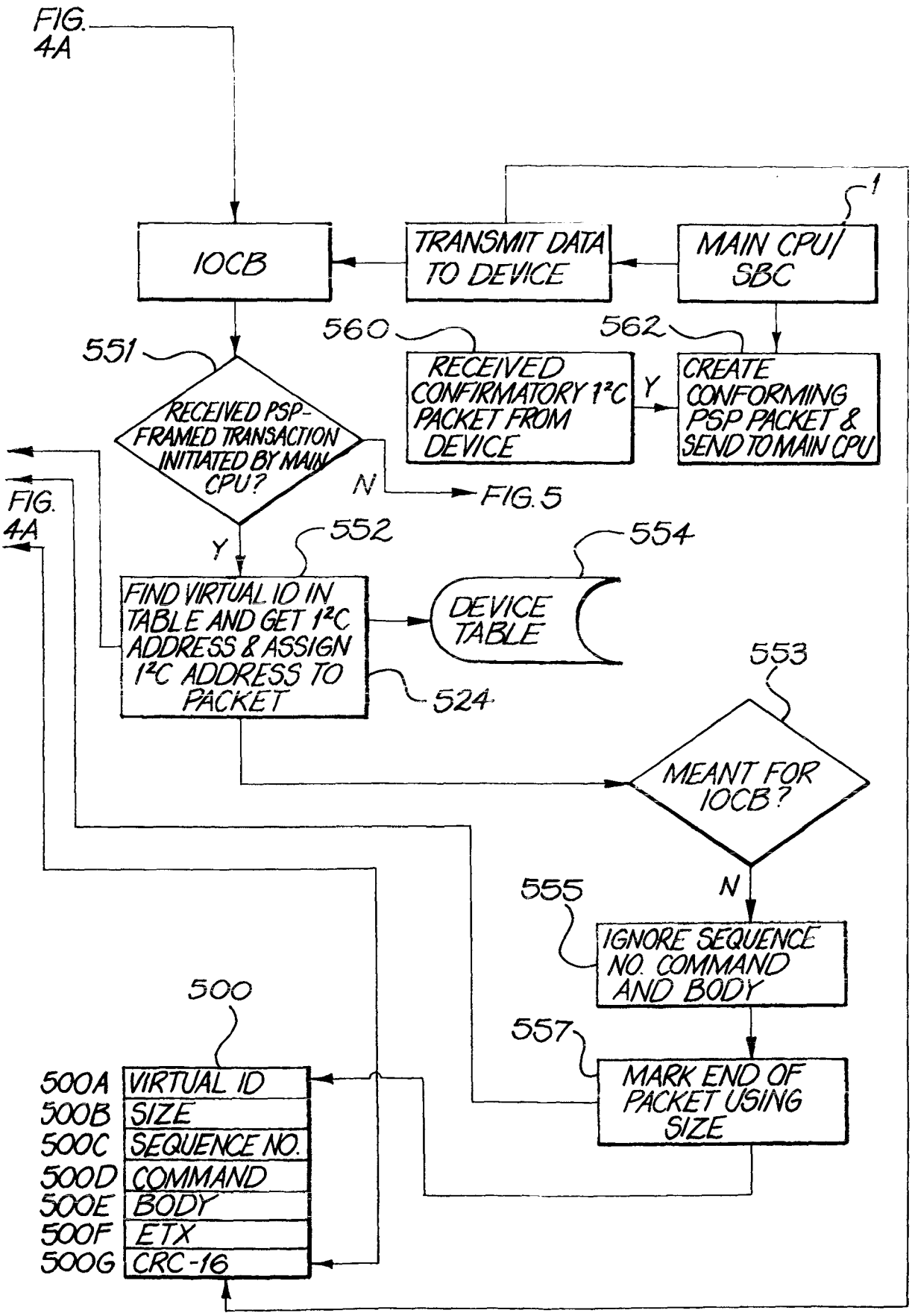


FIG. 4A
1
MAIN CPU/ SBC
TRANSMIT DATA TO DEVICE
IOCB
560
562
551
RECEIVED CONFIRMATORY I²C PACKET FROM DEVICE
Y
CREATE CONFORMING PSP PACKET & SEND TO MAIN CPU
RECEIVED PSP-FRAMED TRANSACTION INITIATED BY MAIN CPU?
N
FIG. 5
FIG. 4A
Y
552
554
FIND VIRTUAL ID IN TABLE AND GET I²C ADDRESS & ASSIGN I²C ADDRESS TO PACKET
DEVICE TABLE
524
553
MEANT FOR IOCB?
N
555
IGNORE SEQUENCE NO. COMMAND AND BODY
557
MARK END OF PACKET USING SIZE
500
500A VIRTUAL ID
500B SIZE
500C SEQUENCE NO.
500D COMMAND
500E BODY
500F ETX
500G CRC-16

FIG. 4B

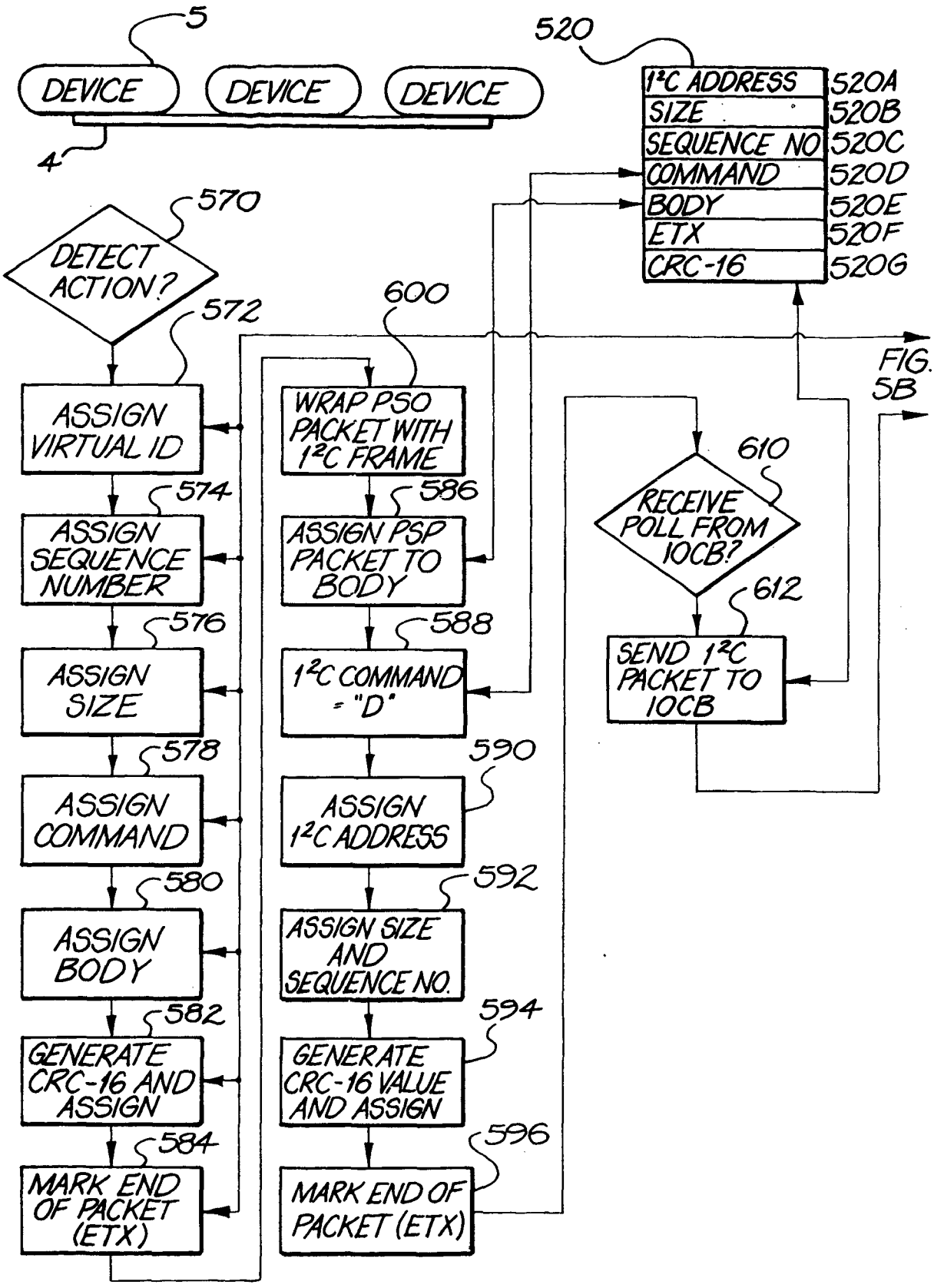

5
DEVICE
DEVICE
DEVICE
4
520
I²C ADDRESS 520A
SIZE 520B
SEQUENCE NO 520C
COMMAND 520D
BODY 520E
ETX 520F
CRC-16 520G
570
DETECT ACTION?
572
600
ASSIGN VIRTUAL ID
574
ASSIGN SEQUENCE NUMBER
576
ASSIGN SIZE
578
ASSIGN COMMAND
580
ASSIGN BODY
582
GENERATE CRC-16 AND ASSIGN
584
MARK END OF PACKET (ETX)
WRAP PSO PACKET WITH I²C FRAME
586
ASSIGN PSP PACKET TO BODY
588
I²C COMMAND = "D"
590
ASSIGN I²C ADDRESS
592
ASSIGN SIZE AND SEQUENCE NO.
594
GENERATE CRC-16 VALUE AND ASSIGN
596
MARK END OF PACKET (ETX)
610
RECEIVE POLL FROM IOCB?
612
SEND I²C PACKET TO IOCB
FIG. 5B

FIG. 5A

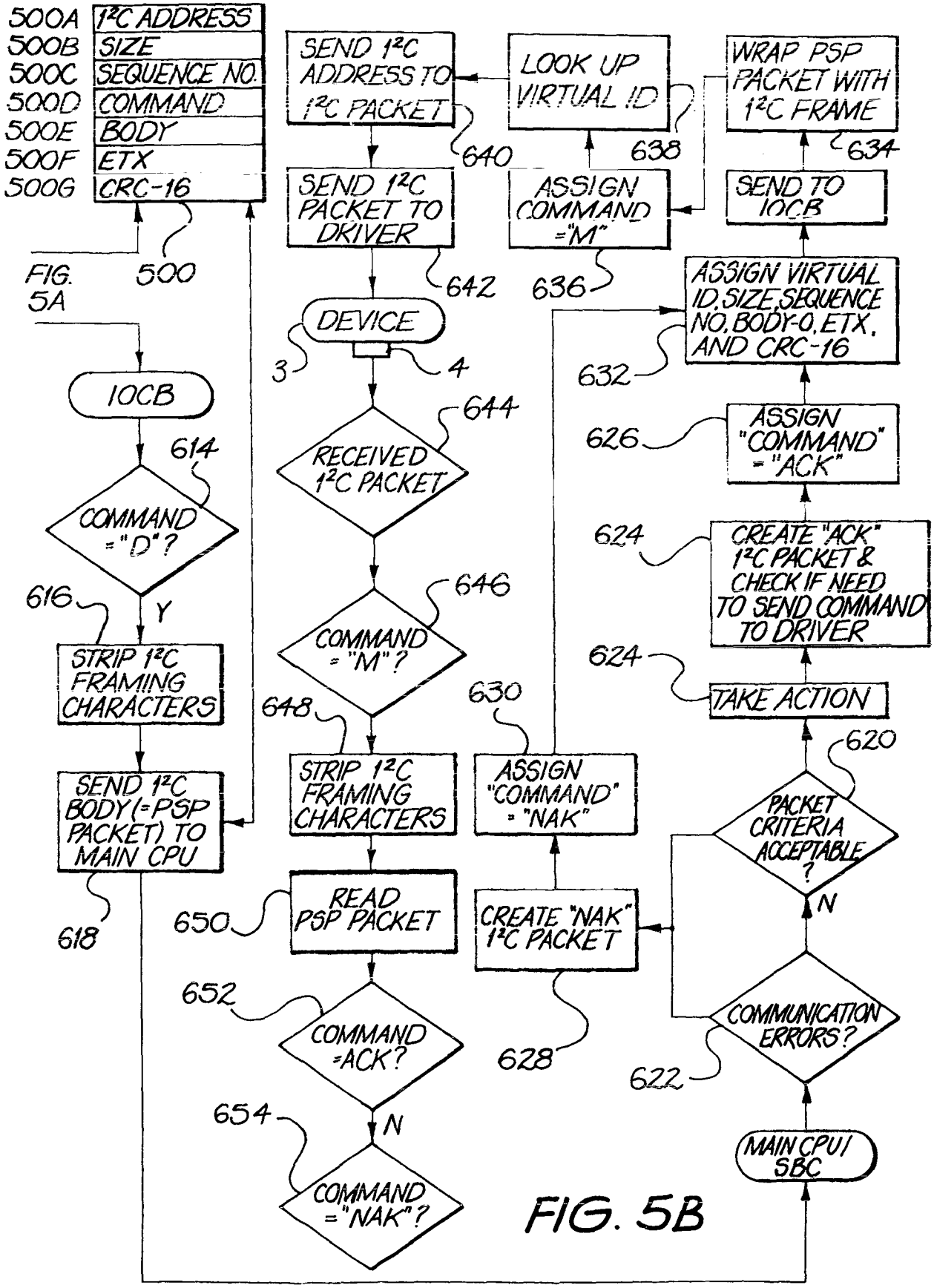

500A I²C ADDRESS
500B SIZE
500C SEQUENCE NO.
500D COMMAND
500E BODY
500F ETX
500G CRC-16
500
FIG. 5A
IOCB
614
COMMAND = "D"?
Y
616
STRIP I²C FRAMING CHARACTERS
SEND I²C BODY (=PSP PACKET) TO MAIN CPU
618
SEND I²C ADDRESS TO I²C PACKET
640
SEND I²C PACKET TO DRIVER
642
DEVICE
3
4
644
RECEIVED I²C PACKET
646
COMMAND = "M"?
648
STRIP I²C FRAMING CHARACTERS
650
READ PSP PACKET
652
COMMAND = ACK?
654
N
COMMAND = "NAK"?
LOOK UP VIRTUAL ID
638
ASSIGN COMMAND = "M"
636
630
ASSIGN "COMMAND" = "NAK"
CREATE "NAK" I²C PACKET
628
WRAP PSP PACKET WITH I²C FRAME
634
SEND TO IOCB
ASSIGN VIRTUAL ID, SIZE, SEQUENCE NO, BODY-0, ETX, AND CRC-16
632
ASSIGN "COMMAND" = "ACK"
626
624
CREATE "ACK" I²C PACKET & CHECK IF NEED TO SEND COMMAND TO DRIVER
624
TAKE ACTION
620
PACKET CRITERIA ACCEPTABLE ?
N
COMMUNICATION ERRORS?
622
MAIN CPU/ SBC

FIG. 5B

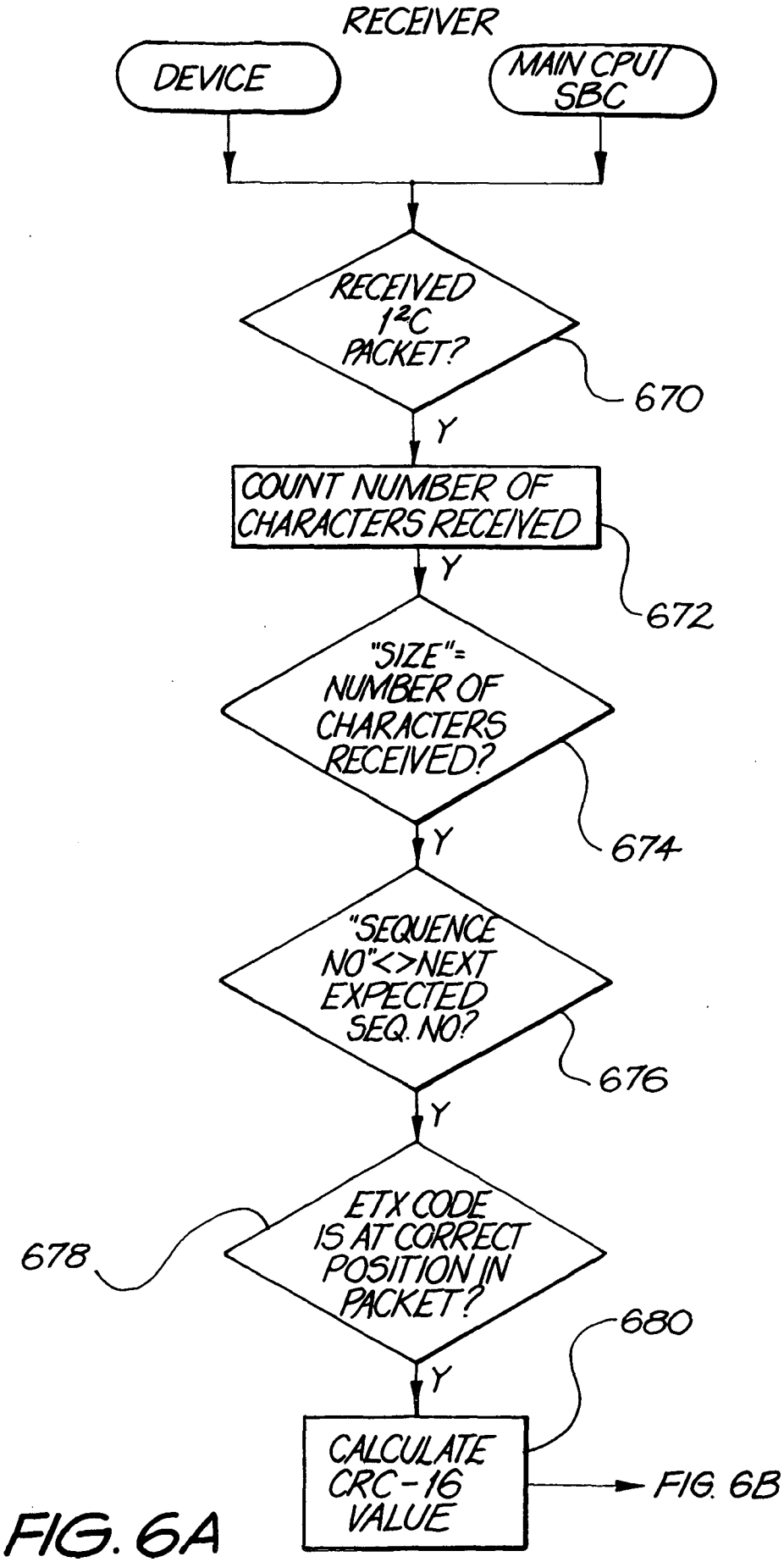

RECEIVER
DEVICE
MAIN CPU/ SBC
RECEIVED I²C PACKET?
670
Y
COUNT NUMBER OF CHARACTERS RECEIVED
Y
672
"SIZE"= NUMBER OF CHARACTERS RECEIVED?
674
Y
"SEQUENCE NO"<>NEXT EXPECTED SEQ. NO?
676
Y
678
ETX CODE IS AT CORRECT POSITION IN PACKET?
680
Y
CALCULATE CRC-16 VALUE
FIG. 6B

FIG. 6A

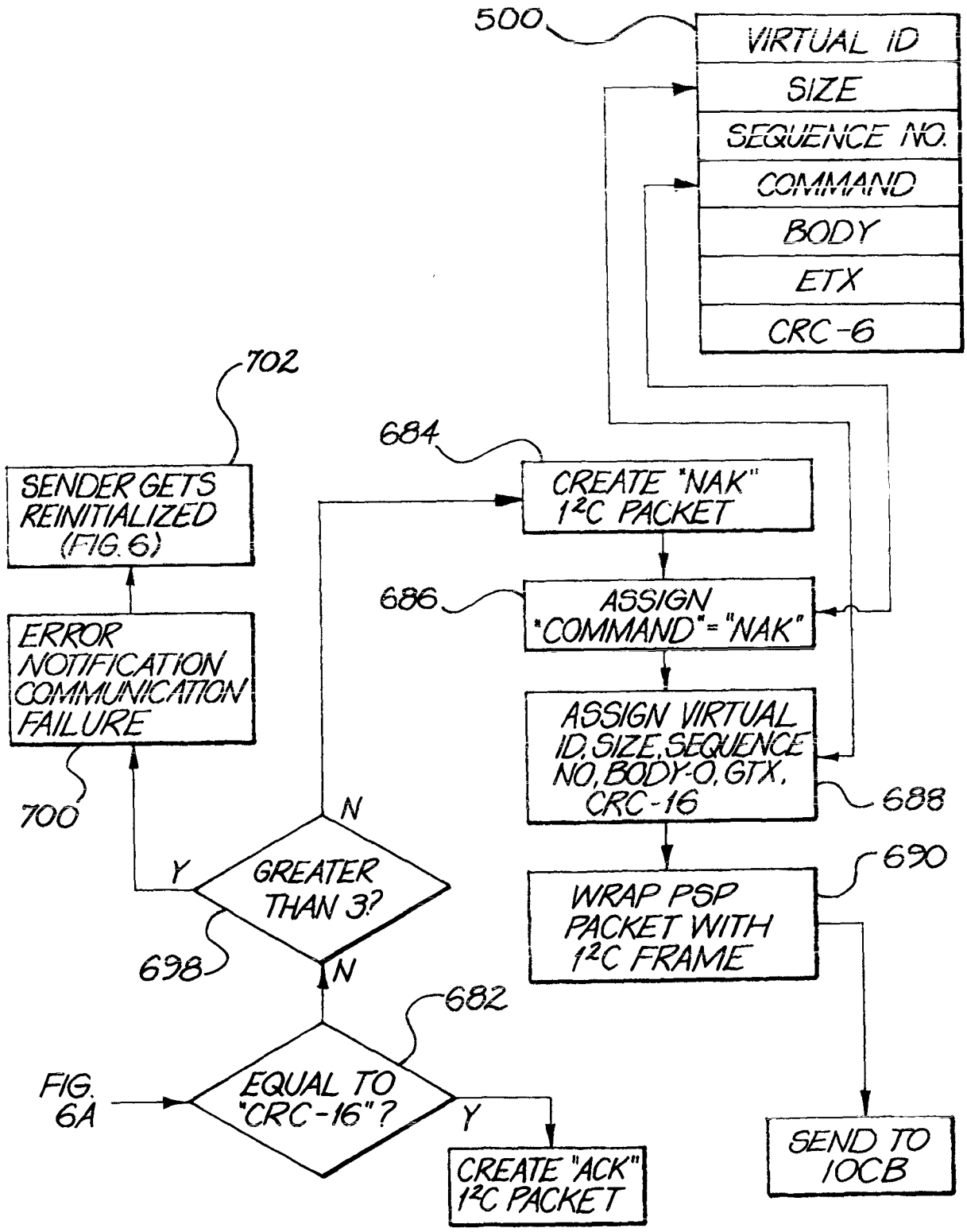

500
VIRTUAL ID
SIZE
SEQUENCE NO.
COMMAND
BODY
ETX
CRC-6
702
SENDER GETS REINITIALIZED (FIG. 6)
684
CREATE "NAK" I²C PACKET
686
ASSIGN "COMMAND" = "NAK"
ASSIGN VIRTUAL ID, SIZE, SEQUENCE NO, BODY-0, GTX, CRC-16
688
690
WRAP PSP PACKET WITH I²C FRAME
ERROR NOTIFICATION COMMUNICATION FAILURE
700
N
Y
GREATER THAN 3?
698
N
682
FIG. 6A
EQUAL TO "CRC-16"?
Y
CREATE "ACK" I²C PACKET
SEND TO IOCB

FIG. 6B

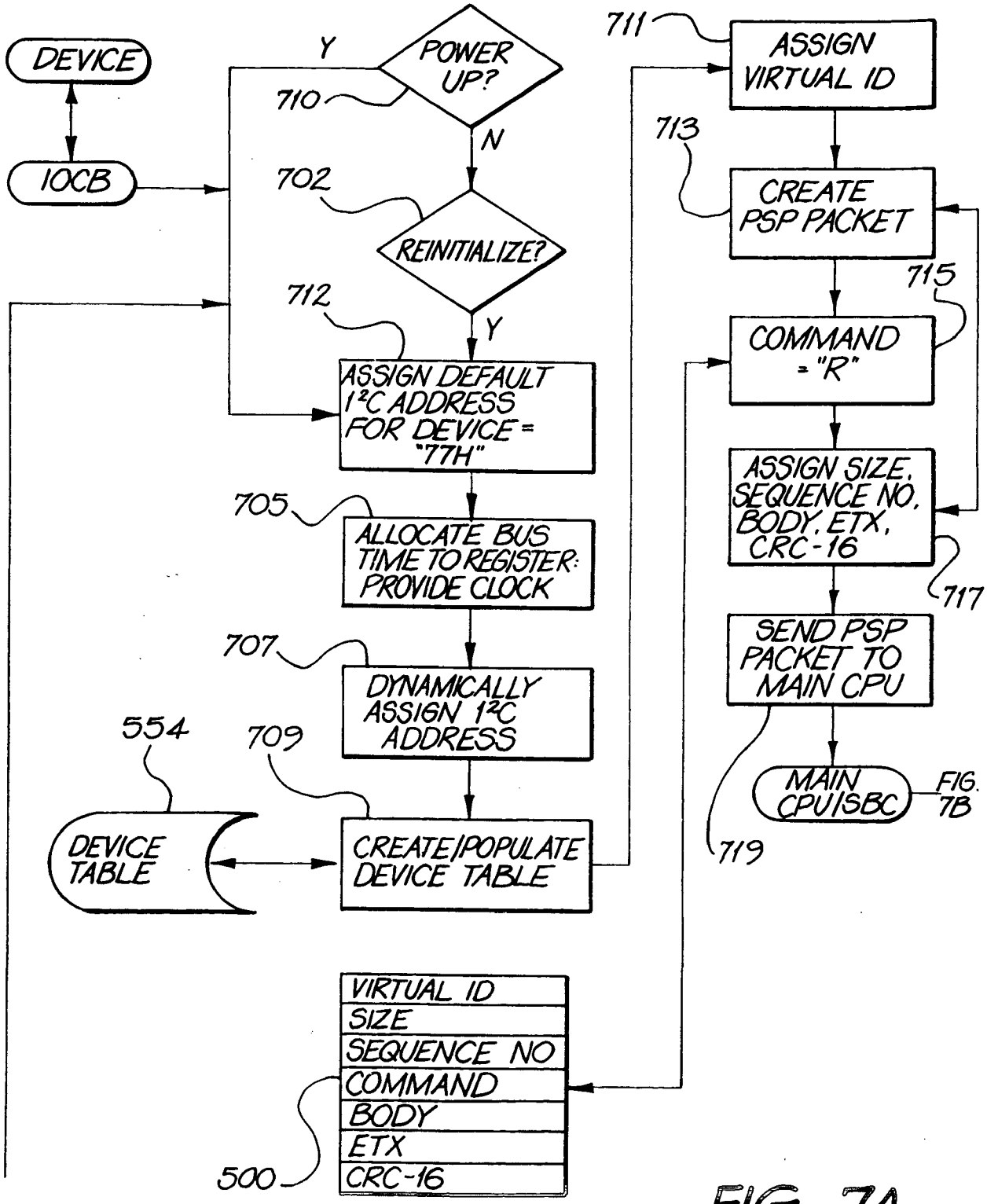

DEVICE
IOCB
POWER UP?
Y
N
710
702
REINITIALIZE?
Y
712
ASSIGN DEFAULT I²C ADDRESS FOR DEVICE = "77H"
705
ALLOCATE BUS TIME TO REGISTER: PROVIDE CLOCK
707
DYNAMICALLY ASSIGN I²C ADDRESS
554
709
DEVICE TABLE
CREATE/POPULATE DEVICE TABLE
711
ASSIGN VIRTUAL ID
713
CREATE PSP PACKET
715
COMMAND = "R"
ASSIGN SIZE, SEQUENCE NO, BODY, ETX, CRC-16
717
SEND PSP PACKET TO MAIN CPU
719
MAIN CPU/SBC
FIG. 7B
VIRTUAL ID
SIZE
SEQUENCE NO
COMMAND
BODY
ETX
CRC-16
500

FIG. 7A

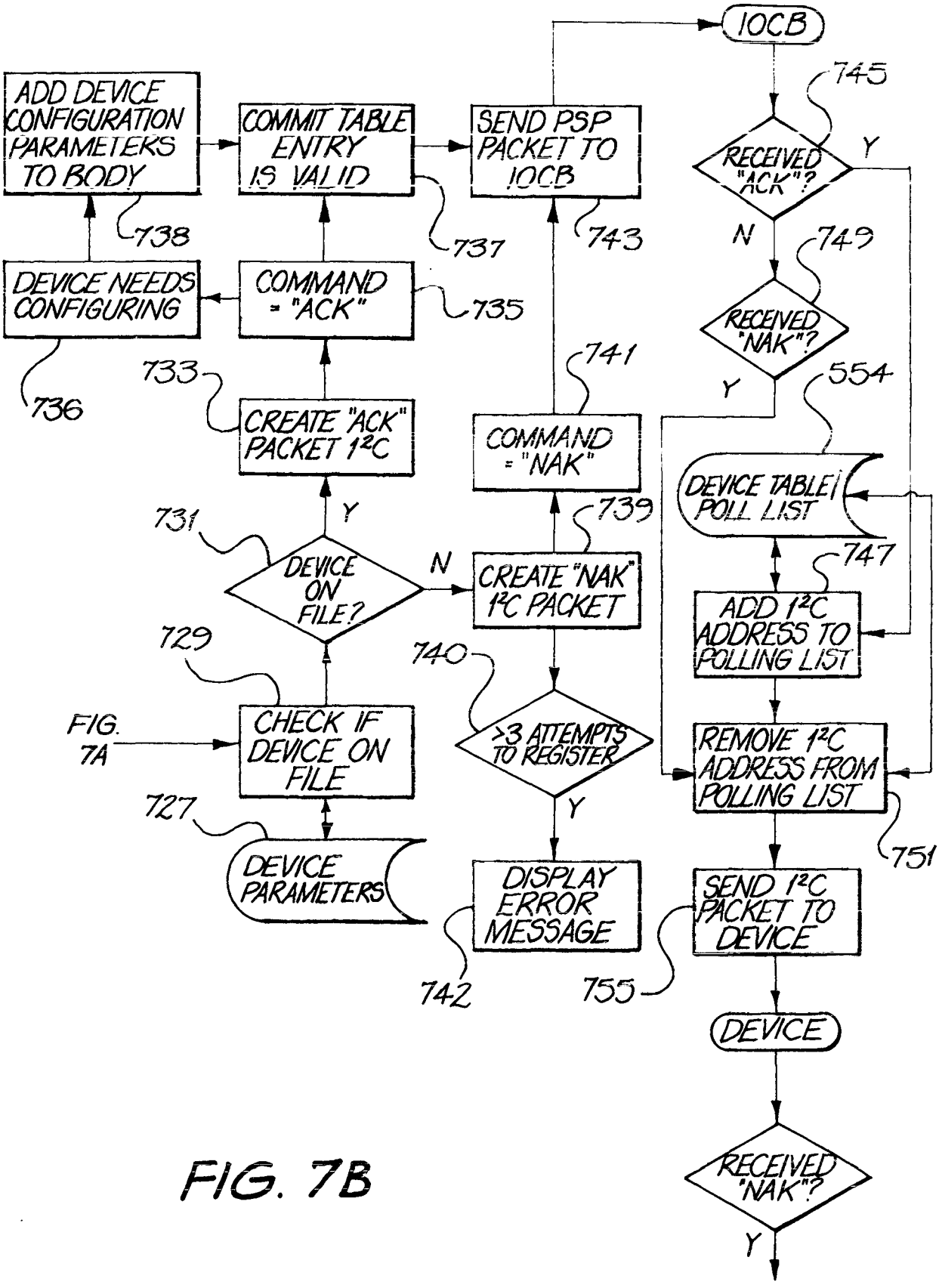

IOCB
745
ADD DEVICE CONFIGURATION PARAMETERS TO BODY
COMMIT TABLE ENTRY IS VALID
SEND PSP PACKET TO IOCB
RECEIVED "ACK"?
Y
N
749
738
737
743
DEVICE NEEDS CONFIGURING
COMMAND = "ACK"
735
RECEIVED "NAK"?
741
554
733
736
CREATE "ACK" PACKET I²C
COMMAND = "NAK"
Y
DEVICE TABLE/ POLL LIST
731
739
DEVICE ON FILE?
N
CREATE "NAK" I²C PACKET
747
ADD I²C ADDRESS TO POLLING LIST
729
740
FIG. 7A
CHECK IF DEVICE ON FILE
>3 ATTEMPTS TO REGISTER
REMOVE I²C ADDRESS FROM POLLING LIST
727
751
DEVICE PARAMETERS
DISPLAY ERROR MESSAGE
SEND I²C PACKET TO DEVICE
742
755
DEVICE
RECEIVED "NAK"?
Y

FIG. 7B