PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :		(11) International Publication Number:
G06F 9/00, 9/26	A1	WO 94/16382
		(43) International Publication Date: 21 July 1994 (21.07.94)

(21) International Application Number: PCT/US93/00117

(22) International Filing Date: 6 January 1993 (06.01.93)

(71) Applicant: THE 3DO COMPANY [US/US]; 1820 Gateway Drive, San Mateo, CA 94404 (US).

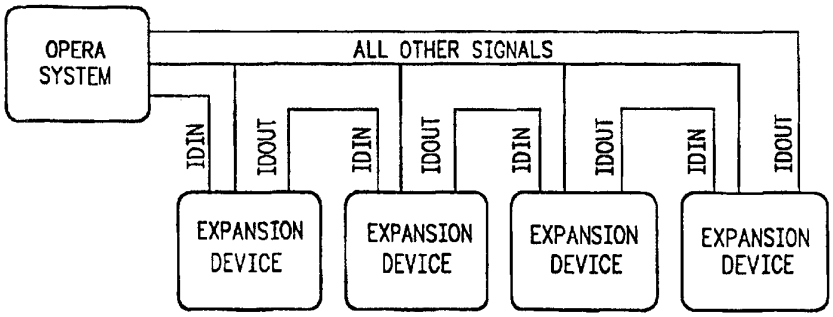
(72) Inventors: TOMPANE, Richard, B.; 3569 Cambridge Lane, Mountain View, CA 94040 (US). DRAKO, Dean, M.; 120 W. Edith, Los Altos, CA 94022 (US). NEEDLE, David, L.; 2981 Northwood Drive, Alameda, CA 94501 (US).

(74) Agent: WOLFELD, Warren, S.; Fliesler, Dubb, Meyer and Lovejoy, Four Embarcadero Center - Suite 400, San Francisco, CA 94111-4156 (US).

(81) Designated States: AT, AU, BB, BG, BR, CA, CH, DE, DK, ES, FI, GB, HU, JP, KP, KR, LK, LU, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, SN, TD, TG).

Published
With international search report.

(54) Title: EXPANSION BUS



(57) Abstract

Method for assigning identification codes to a plurality of peripheral devices, comprises the steps of asserting a first signal to all of the peripheral devices to indicate the start of an identification code assignment procedure, and asserting a series of pulses to all of the peripheral devices beginning after the assertion of the first signal. Each of the peripheral devices except a first and a last one of the peripheral devices asserts a logic transition to a respective subsequent one of the peripheral devices in a daisy chain in response to one of the pulses which occurs after the peripheral device receives the logic transition from a prior one of the peripheral devices in the daisy chain, and the first peripheral device asserts the logic transition to a respective subsequent one of the peripheral devices in the daisy chain in response to one of the pulses which occurs after the assertion of the first signal. Each of the peripheral devices derives a respective unique identification code from the number of such pulses which occur between assertion of the first signal and receipt by the peripheral device of the logic transition.

FOR THE PURPOSES OF INFORMATION ONLY

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

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

EXPANSION BUS

BACKGROUND

1. Field of the Invention

The invention relates to expansion buses for computer-based systems, and more particularly, to techniques for automatically assigning unique identification codes to peripheral devices on such an expansion bus.

2. Description of Related Art

Many computer systems are adapted to be connected to one or more optional peripheral devices, such as disk drives, tape drives and certain types of communications interfaces. Such connections are typically made via a so-called expansion bus which is connected to the host computer and includes one or more connectors into which the optional peripherals can plug.

Expansion buses are often designed such that each device on the bus is associated with a unique address or other identification code. When the host computer desires to access a specific device on the bus, the host computer in some manner drives the identification code onto the bus in conjunction with the access request. The desired device recognizes its own identification code and responds to the request.

The use of unique identification codes requires that each device have such a code assigned to it. While some systems use permanently assigned identification codes, in a system with optional peripherals, it is desirable for such codes to be assigned after all desired devices have been connected to the expansion bus. Furthermore,

it is often desirable that the identification codes be assigned automatically, without any intervention by a user.

SUMMARY OF THE INVENTION

According to the invention, roughly described, a method for assigning identification codes to a plurality of peripheral devices, comprises the steps of:

asserting a first signal to all of the peripheral devices to indicate the start of an identification code assignment procedure; and

asserting a series of pulses to all of the peripheral devices, beginning after the assertion of the first signal,

each of the peripheral devices except a first and a last one of the peripheral devices asserting a logic transition to a respective subsequent one of the peripheral devices in a daisy chain in response to one of the pulses which occurs after the peripheral device receives the logic transition from a prior one of the peripheral devices in the daisy chain,

the first peripheral device asserting the logic transition to a respective subsequent one of the peripheral devices in the daisy chain in response to one of said pulses which occurs after the assertion of the first signal,

each of the peripheral devices deriving a respective unique identification code from the number of such pulses which occur between assertion of the first signal and receipt by the peripheral device of the logic transition.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to particular embodiments thereof, and reference will be made to the drawings, in which:

Fig. 1 is a symbolic diagram of a system according to the invention without external expansion;

Fig. 2 is a symbolic diagram of a system according to the invention with buffered internal expansion;

Fig. 3 is a symbolic diagram of a system according to the invention with external expansion;

Fig. 4 is a symbolic diagram of a system according to the invention with an expander unit;

Fig. 5 is a symbolic diagram of a system according to the invention showing daisy chaining of IDIN and IDOUT signals;

Fig. 6 is a timing diagram illustrating device address assignment after power on;

Fig. 7 is a symbolic schematic diagram of an expander unit ID bypass circuit;

Fig. 8 is a symbolic diagram of an I/O model;

Fig. 9 is a timing diagram illustrating an example of media access bit and RDY- signal operation;

Fig. 10 is a flow chart showing the control flow for a command which returns no data;

Fig. 11 is a flow chart showing the control flow for a command which returns data via the status return FIFO;

Fig. 12 is a flow chart of the control flow for a command which returns data via the data return FIFO;

Fig. 13 is a timing diagram illustrating the sequence of events for a data read operation;

Fig. 14 is a timing diagram illustrating the sequence of events when an error occurs;

Figs. 15 and 16 are timing diagrams for various transactions which may occur on the bus;

Fig. 17 is a timing diagram showing IDIN and IDOUT timing;

Fig. 18 is a timing diagram showing the enable timing for the RDY- signal when a device is selected;

Fig. 19 is a timing diagram showing the disable timing for the RDY- signal when a device is de-selected; and

Fig. 20 is a symbolic block diagram illustrating circuitry in a typical expansion device.

DETAILED DESCRIPTION

An embodiment of the invention will be described with respect to a system in which the host computer is an interactive home entertainment system referred to herein as Opera. The Opera Expansion Bus is designed to provide an inexpensive and simple method of connecting expansion devices to the Opera System. Commercially important features of the Opera Expansion Bus include:

- Logical support of 16 expansion devices
- Physical support of 2 internal devices without buffering
- Automatic configuration of expansion device addresses
- Support of a select and de-select protocol
- Simple interface requirements
- Support of both internal and external expansion devices
- Transfer rates up to 4 MB/second

The Opera Expansion Bus is used for both internal devices and external devices. All devices are logically connected to a single bus. The Opera Expansion bus is a low cost bus designed to support a single master (the Opera System) with multiple slaves (expansion devices). All data is transferred to or from the master. Data is never transferred directly between expansion devices (slaves) on the Opera Expansion Bus.

I. PHYSICAL SPECIFICATION

The Opera Expansion Bus supports internal and external devices. The internal system can support up to two internal devices without additional buffering as shown in Fig. 1. Additional internal expansion devices can be supported if the electrical requirements are satisfied or additional buffering is provided. One method of internal buffering is shown in Fig. 2.

The external portion of the expansion bus is isolated from the internal portion by a set of buffers as shown in Fig. 3. These buffers provide isolation of the Opera internals from the outside world and provide the electrical drive needed for the external cables. Note that only a single Expansion device can be supported on the External Expansion Bus. To support additional external expansion devices an Opera Expander Unit is used as shown in Fig. 4. The Opera Expander Unit buffers the Opera Expansion Bus signals and provides the additional connectors needed to plug in expansion devices. The Opera Expansion Bus Unit could support any number of external expansion devices. Note that each external expansion device only requires a single connector. The Opera Expansion Bus does not use a "daisy chaining" method of connection and therefore does not need two connectors on each expansion device.

Fig. 5 diagrams the wire topology of the Opera Expansion Bus. All Opera Expansion Bus signals except IDIN and IDOUT are connected in parallel as a logical bus. The IDIN and IDOUT signals are "daisy chained" between expansion devices. The daisy chaining, however, is done either in the Opera System or in the Opera Expander Unit. The actual expansion devices do not need two connectors. The IDOUT signal of one expansion device feeds the IDIN signal of the next expansion device. The last device's IDOUT signal is not connected, or in a different embodiment, may be connected to the

Opera System. All other signals on the bus are logically bused in parallel. Such signals may be buffered, but buffering is not required on the IDIN and IDOUT signals.

The connector for the external Opera Expansion Bus has the following pinout. Note that the cable is shielded.

Pin Number	Signal	Pin Number	Signal
1	ADB[0]	13	STB-
2	ADB[1]	14	CMD-
3	ADB[2]	15	RDY-
4	GND	16	GND
5	ADB[3]	17	SEL-
6	ADB[4]	18	WR-
7	ADB[5]	19	IDIN
8	GND	20	IDOUT
9	ADB[6]	21	RESET-
10	ADB[7]	22	GND
11	INT-	23	reserved
12	GND	24	reserved
Shield	Chassis GND	25	POWER

II. LOGICAL OPERATION

A. Signals

The Opera Expansion Bus carries the signals listed in the table below. Note that a "-" following a signal name indicates an active low signal.

Signal Name	I/O (Viewed from Expansion Device)	Description
STB-	I	Strobe signal. Used to indicate all address, data, and command transfers on the Opera Expansion Bus
ADB[7:0]	I/O	Bi-directional Address and Data Bus (tri-state outputs)
WR-	I	Write signal. Indicates command, data, or address information will be transferred from the Opera System to the expansion device. Used in conjunction with the SEL-

			and CMD- signals to determine the transaction type.
	CMD-	I	Command signal. Used in conjunction with the SEL- and WR- signals to determine the transaction type.
	SEL-	I	Selection signal. Used in conjunction with the CMD- and WR- signals to determine the transaction type.
	RDY-	O	Ready signal. Indicates the expansion device has placed data onto the ADB bus. Also used to indicate media access (door open/close events) (Tri-state output).
	INT-	O	Interrupt signal. Indicates drive has data or status information ready for transfer. (Open collector output)
	RESET-	I	Power On Reset signal driven by the Opera System.
	IDIN	I	Input from previous expansion device. Used for device address assignment
	IDOUT	O	Output to next expansion device. Used for device address assignment
	POWER CTL	O	Output to turn on expansion device. Provides a total of 40 mA of 4.5 to 5.5 Volts. This signal is intended to control a power relay in an expansion device.

All signals are connected in parallel (bused) fashion to all the expansion devices except for the IDIN and IDOUT signals.

The WR-, CMD-, STB-, and SEL- signals are always driven by the Opera System. The IDIN signal is always driven by either the previous expansion device or the Opera System. The IDOUT signal is always driven by the expansion device. The RDY- signal is only driven by the currently selected expansion device. ADB[7:0] are driven by either the Opera System or the currently selected expansion device. ADB[7:0] are driven by the Opera System unless a read transaction is being performed in which case they are driven by the selected expansion device. The INT- signal is asynchronous. The RDY- signal must be synchronized to STB-.

B. ID Assignment at Power Up

The Opera Expansion Bus is designed to work with a multi-tasking operating system. To multi-task efficiently the operating system should be able to suspend communication with one expansion device, communicate with a different expansion device, and then resume communication with the original device. More complicated scenarios involving device select, de-select, and re-select are also desirable. The Opera Expansion Bus uses a simple device address method to select and de-select devices.

It is desirable that an Opera System owner not have to set switches for an expansion devices address. To avoid this, the Opera System uses a method of device address assignment after power is turned on. Each device connected to the Opera Expansion Bus is assigned a sequential address (identification code) corresponding to its location on the bus. The expansion device then responds only to its assigned address.

A device determines its address using a counting method. After RESET- is removed from the expansion bus

the Opera System will assert and de-assert the STB- signal 17 times. A device determines its address by counting the number of times STB- is asserted while the IDIN signal is low. An example is shown in Fig. 6.

Once an expansion device observes an STB- assertion while its IDIN pin is high it can determine its address. The Opera System supports expansion devices numbered from 0 to 15. Note that an expansion device should only perform a comparison on the low order 4 bits of the expansion bus address. It should ignore the upper 4 bits of the address. The expansion device must assert (drive high) its IDOUT pin after the assertion of STB- while its IDIN pin is high.

Fig. 7 shows the circuit used by the Opera Expander Unit on the IDIN and IDOUT signals of each connector. This circuit allows the user to connect expansion devices to any of the connectors on the Expander Unit without breaking the "daisy-chain" connections of the IDIN and IDOUT signals.

C. I/O Model

The Opera Expansion Bus operates using a FIFO (first in first out) model for commands, data, and status. These FIFOs are located in each expansion device. The I/O model uses three independent FIFOs for a read only device: 1) Command FIFO, 2) Data Return FIFO, 3) Status Return FIFO. The I/O model uses four independent FIFOs for a Read/Write device: 1) Command FIFO, 2) Data Return FIFO, 3) Data Write FIFO, 4) Status Return FIFO. A diagram is shown in Fig. 8.

The Command FIFO is used by the master (Opera System) to send commands (access requests) to the I/O device. The Data Return FIFO is used by the expansion device to send data to the Opera System. The Data Write FIFO is used by the Opera System to send data to the expansion device. The Status Return FIFO is used by the

expansion device to send status to the Opera System. The I/O model does not specify the size of the FIFO's.

The master (Opera System) writes commands into the Command FIFO of the expansion device. These commands are executed in the order received. The commands may cause the generation of data in the Data Return FIFO or Status Return Fifo or the consumption of data from the Data Write FIFO. For example, a CD-ROM expansion device given a read data command would place data into the Data Return FIFO. The Opera System would later remove this data from the Data Return FIFO.

The I/O model defines a Status Return FIFO which is read by the Opera System to obtain status information. The status information tells the Opera System if any errors occurred during command execution. All commands must generate a minimum of a single byte of information in the Status Return FIFO when the command is completed.

Interrupts are used to indicate that the Status Return FIFO or the Data Return FIFO contain information. The interrupt generated by the Status Return FIFO indicates to the Opera System that the command has been completed.

D. Transactions

The fundamental unit of communications on the Opera Expansion Bus is a transaction. A transaction is initiated by the Opera System on the expansion bus to communicate with an expansion device.

The STB- signal controls the timing of all information transfer between the Opera System and the expansion devices. The Opera System uses the STB- signal in conjunction with three control signals (SEL-, CMD-, and WR-) to control an expansion device. Eight possible expansion bus transactions are possible with the encoding of the SEL-, CMD-, and WR- signals (see table below). The control signals are valid before, during, and after the assertion of the STB- signal. The control

signals, however, are only guaranteed to be valid around the assertion of the STB- signal as defined in the Electrical Specification Section of this document.

SEL-	CMD-	WR-	Transaction Name	Operation
0	0	0	WR_POL	Opera System performing a write to the expansion device's Poll Register
0	0	1	RD_POL	Opera System performing a read of the expansion device's Poll Register
0	1	0	SELECTION	Opera System performing a selection of an expansion device.
0	1	1	Reserved.	
1	0	0	WR_COM	Opera System performing a write of a command byte to the expansion device
1	0	1	RD_STAT	Opera System performing a read of the expansion device's status
1	1	0	WR_DATA	Opera System performing a write of a data byte to the expansion device
1	1	1	RD_DATA	Opera System performing a read of a data byte from the expansion device

The above transactions are used by the Opera System to communicate with expansion devices. The operation of each transaction is summarized below.

Selecting an Expansion Device (SELECTION). An expansion device is selected when a SELECTION transaction is observed in which ADB[7:0] matches the expansion device's address. The expansion device remains selected until a SELECTION transaction is observed which does not match. All transactions except the SELECTION transaction are ignored by all expansion devices which are not selected. Once an expansion device has been

selected the remaining transaction types are used to transfer data and control information to or from the expansion device.

Writing a Command (WR_COM). The Opera System performs a sequence of WR_COM transactions to give the selected expansion device a command. A WR_COM transaction places a single command byte into the expansion device's Command FIFO. The expansion device removes these bytes from the FIFO and executes the command. Commands may be multiple bytes in length. Command length is expansion device dependent.

Reading Status (RD_STAT). The Opera System performs a RD_STAT transaction to obtain a byte of data from the Status Return FIFO. The expansion device drives the ADB[7:0] signals during this transaction with the value of the data in the Status Return FIFO. A command may return multiple pieces of data in the Status Return FIFO. Every command must, however, return a Status Byte which indicates the success or failure of the command. The format of the Status Byte is described in the Status Byte Definition Section below.

Writing Data (WR_DATA). The Opera System performs a WR_DATA transaction to the selected expansion device to place a data byte into the Data Write FIFO.

Reading Data (RD_DATA). The Opera System performs a RD_DATA to the selected expansion device to obtain a data byte from the Data Return FIFO. The expansion device drives the ADB[7:0] signals during this transaction.

Reading Poll Register (RD_POL). The Opera System performs RD_POL transactions to locate the source of an interrupt received on the INT- signal. Since the INT- signal is shared by all expansion devices a method is provided for the Opera System to determine which expansion device(s) generated the interrupt. When a selected device observes an RD_POL transaction it must

place the contents of its Poll Register on the ADB[7:0] lines. Note that a RD_POL transaction does not change the contents of the Poll Register. The Poll Register is defined in the Poll Register Definition section below.

Writing Poll Register (WR_POL). The Opera System performs a WR_POL transaction to change the contents of the Poll Register. The data on the ADB[7:0] lines is placed into the Poll Register. Note that not all bits of the Poll Register are written in a standard fashion. See the Poll Register Definition section for more detail.

E. Status Byte Definition

The Status Byte is returned after the completion of every command by the expansion device. Other bytes may be returned into the Status Return prior to the Status Byte, but the Status Byte must always be returned. The Status Byte contains the following bits:

Bit	Name	Definition
0	device dependent	
1	device dependent	
2	device dependent	
3	device dependent	
4	ERROR	An Error was encountered during execution of the Command. All expected data may not be in the Data Return FIFO.
5	device dependent	
6	device dependent	
7	device dependent	

The ERROR bit is defined in the Status Byte because it is used by the basic I/O Model of the Opera System. The other bits in the Status Byte can be used by an expansion device for any purpose. It is important to

note that a status byte generated without an error implies to the Opera System that all expected data was placed into the Data Return FIFO.

F. Poll Register Definition

The Poll register contains the following bits:

Bit	R/W	Name	Definition
0		Reserved	Returns 0
1	R/W	Interrupt Disable-	The Interrupt Disable- bit disables interrupts when low. This bit is changed only by the Opera System.
2	R/Partial W	Media Access	If the Media Access bit is high the media may have been physically accessed by the user. This bit should be activated when the new media can be accessed by the Opera System (door closing).
3	R/W	Reset-	The Reset- bit can be used to reset the entire expansion device. This bit is set and cleared by the Opera System. Note that this bit must not affect the expansion device's address assignment.
4	R	StatValid-	Indicates the Status Return FIFO contains valid data. Also indicates the expansion device is requesting an interrupt. This bit is high when data is no longer available in the Status Return FIFO.
5	R	ChunkValid-	Indicates the Data Return FIFO contains an entire "chunk" of data. Also indicates the expansion device is requesting an interrupt.

			This bit is high when the Data Return FIFO no longer contains a complete "chunk".
	6	Reserved	Returns 0
	7	Reserved	Returns 0

The Interrupt Disable- and Reset- bits of the Poll Register can be read and written in a standard fashion. Data written to the Chunkvalid- or StatValid- bits of the Poll register is ignored and has no effect. Writes of a zero (low) value to the Media Access bit have no effect. Writes of a one (high) to the Media Access bit clear (set low) the Media Access bit.

The Opera System can determine if the expansion device is driving the INT- line low by examining the Interrupt Disable- bit, the StatValid- bit, and the ChunkValid- bit together. Note that the values in the Poll Register must not change during a RD_POL transaction.

The Poll Register bits must be guaranteed stable during the RD_POL transaction. This can be accomplished by synchronizing all bits to the STB- signal.

The Media Access Bit should be set high when the new media is available (Tray Close on a CD-ROM Drive for example).

G. RDY- Signal Operation

The RDY- signal is used for two functions: 1) It indicates that the ADB[7:0] bus has been driven with valid data during the RD_DATA, RD_STAT, and RD_POL transactions, and 2) It indicates the media may have been physically accessed during a WR_COM or WR_POL transaction. The expansion device should drive the RDY- signal low after it has placed the response data on the ADB[7:0] bus. This indicates to the Opera System that the data is valid on the ADB[7:0] bus. The expansion device should drive the RDY- signal low during a WR_COM

or WR_POL transaction if the Media Access bit is set. It should drive the RDY- signal during all WR_COM or WR_POL transactions which occur when the Media Access Bit is set. This indicates that the media was accessed by the Opera System Owner and that the data obtained from the command might be something unexpected. The Media Access Bit should be set high when the new media is available (Tray Close on a CD-ROM Drive for example).

The RDY- signal is always electrically driven by the selected expansion device. The timing diagrams in the Timing Section describe when the expansion device should drive the RDY- signal. Fig. 9 shows an example of Media Access Bit and RDY- Signal operation.

H. Control Flow

There are two basic methods of Control Flow on the Opera Expansion Bus. The first method is based on polling the expansion device to determine when a command has been completed. This control flow is used for commands that complete very quickly. The second method is based on interrupt generation. This control flow is used for commands which take a significant amount of time to complete.

The two control flow methods are summarized in Fig. 10 for a command which do not return any data to the Opera System. This method might be used to set a parameter in an expansion device. The flow diagrammed on the left uses the polling method of determining when the command has completed. The flow on the right diagrams the interrupt method.

In the polling method a SELECTION transaction is performed to select the expansion device. The command bytes are written to the expansion device using WR_COM transactions. The Opera System polls the device using a RD_POL transaction to determine if the command has been completed correctly. In the interrupt method a SELECTION transaction is performed to select the

expansion device. A WR_POL transaction is used to enable interrupts. The command bytes are written to the expansion device using WR_COM transactions. The Opera Expansion Bus is then available for other operations to other expansion devices. After the command is complete the expansion device places a Status Byte into the Status Return FIFO. This generates an interrupt to the Opera System. The Opera System performs SELECTION and RD_POL transactions to locate the source of the interrupt. Once the source is located it uses a WR_POL transaction to disable the interrupt and a RD_STAT transaction to obtain the Status Byte. Note that the interrupt is removed by the expansion device once the RD_STAT command has been performed because the Status Return FIFO is now empty.

Some commands issued to expansion devices may return small amounts of data via the Status Return FIFO. These commands are often read parameter type commands. The flow control used for these commands is the same as above except the Opera System uses RD_STAT transactions to obtain the extra data. A diagram is shown in Fig. 11. Note that the Opera System must know how many bytes will be generated by the command. The Opera system must know how many bytes to read from the Status Return Fifo. A single command type must therefore return a fixed number of bytes even if an error occurs. The Status Byte is the last byte returned by the command.

The control flow for commands which return significant amounts of data is more complex. The basic selection process and command writing portions, however, are the same. The Opera I/O system defines "Chunks" of data. The size of a data Chunk is expansion device dependent and might even change for a single device. The Opera System and the expansion device, however, must operate assuming the same size Chunk. The Chunk's size is normally directly related to the amount of data

buffering in an expansion device. A control flow diagram for an operation which returns a significant amount of data is shown in Fig. 12. Note that the polling method is not used; only the interrupt method is used.

After all the command bytes have been sent to the expansion device other operations may be performed on the Opera Expansion Bus. Once the expansion device has obtained and placed in its Data Return FIFO a Chunk of data or the command has been completed it will generate an interrupt. The Opera System performs SELECTION and RD_POL transactions to locate the source of the interrupt. Once the source is located it uses the ChunkValid- and StatValid- bits to determine its action. The StatValid- bit is set the expansion device has completed the command and the Opera System will take the appropriate action described below. If the ChunkValid- bit is set the expansion device has placed an entire Chunk of data into the Data Return FIFO. The Opera System will disable interrupts with a WR_POL transaction and then remove the data from the Data Return FIFO using RD_DATA transactions. The interrupt will be removed by the expansion device once the data level in the Data Return FIFO falls below the size of Chunk. Note that the Opera System does not clear the interrupt. The Opera System will remove exactly one Chunk from the Data Return FIFO. After the Chunk has been removed the Opera System will enable interrupts using a WR_POL transaction.

If the StatValid- bit is active in the Poll register this implies that a Status Byte is in the Status Return FIFO. The Status Byte's presence implies the command has been completed. There may be any amount of data in the Data Return FIFO: less than a Chunk, an exact Chunk, or more than a Chunk. The Opera System will read the Status Byte using a RD_STAT transaction. If no

error is indicated the data in the Data Return FIFO must match exactly the amount originally requested (minus the amount already transferred) by the Opera System. If the Error bit is set the amount of data in the FIFO is unknown and the FIFO must be Flushed. If the Error bit is not set the Opera System will use RD_DATA transactions to transfer all the remaining data.

I. INT- Signal Operation

The INT- signal is shared by all devices connected to the Opera Expansion Bus. An expansion device does not have to be selected to drive the INT- signal active (low). An expansion device should drive the INT- signal low when a Chunk of data is available for transfer or when any data is available in the Status Return FIFO.

The INT- signal is an asynchronous signal. All bits in the POLL Register, however, must change synchronously to the STB- signal.

The INT- signal is disabled by the Interrupt Disable- Bit in the Poll Register. The Interrupt Disable- bit disables interrupts when it is low. This bit is set and cleared by the Opera System.

J. Sequence of Events

Figs. 13 and 14 describe the sequence of events occurring in an expansion device. These diagrams are only example sequences. Many other sequences are possible. Fig. 13 shows the sequence of events for a Data Read Operation. The listing on the left indicates different components in the system. "Device Activity" is the hardware and software in the expansion device. The "Expansion Bus Activity" row describes the operations which are occurring on the Opera Expansion Bus. The "Command Write Seq" is the sequence of transactions which select the device and write a series of command bytes to the expansion device. The "Int Seq" is the sequence of RD_POL transactions and SELECTION transactions which the Opera System performs to locate

the source of the interrupt. Note that this sequence also includes a WR_POL transaction to disable interrupts (this is what shortens the Expansion Bus Interrupt Signal). The "Read Seq" is the series of RD_DATA transactions which obtain data from the expansion device.

Note that in Fig. 13 the final transfer from the expansion device is not a complete Chunk. Only a partial Chunk is transferred. The Opera System knows the data is available because the Status Byte is generated. If the final transfer from the expansion device is a complete Chunk a Status Byte and a Chunk interrupt would be generated simultaneously. The expansion device must place the Status Byte into the Status Return FIFO and thereby set the StatValid- bit in the Poll Register within 30 microseconds of the Data Interrupt generation.

Fig. 14 shows a sequence of events when a fatal error occurs. This flow is used when the expansion device encounters an error and is unable to return the requested amount of data. During the generation of Chunk 1 the expansion device encounters a fatal error which terminates the command. A status byte is generated by the expansion device which in turn generates an interrupt to the Opera System. The Opera system knows that the interrupt was due status byte generation from the RD_POL transaction. The Status Byte indicates an error has occurred and the Opera System sequences through an error recovery plan. The Opera System may either remove the data which is available from the Data Return FIFO or simply Flush the data in the Data Return FIFO. For the Opera System to remove the data from the Data Return FIFO, however, will require that the Opera System know how much data is in the Data Return FIFO.

K. Command Requirements

An expansion device can define as many commands as it needs. The Opera Expansion Bus does not define the

size, length, or encoding of general purpose commands. The Opera Expansion Bus does, however, require that each device support an identification command. The Read Identification Command is a 7 byte command sent to the expansion device to determine the device type. When an expansion device receives a Read Identification Command it must return 10 bytes of data (via the Status Return Fifo) which identify the expansion device. Only the first byte of the Read Identification Command is of significance. The other 6 bytes are reserved for future use. The first byte of the identification command is 83H. The expansion device must recognize this and respond with 10 bytes of returned data. The 10 bytes of returned data are:

Byte Numbers	Contents
0-1	Two bytes of manufacturer identification
2-3	Two bytes of manufacturer device number
4-5	Two bytes of revision number
6-7	Flag Bytes 0 and 1
8-9	Device Driver Size (in words)

Of the Flag Bytes, only bit 0 of Flag Byte 0 is defined. This bit is a Device Driver bit which indicates to the Opera System that the expansion device has a device driver which it can download. The size (in words) of the device driver is stored in bytes 8-9 of the Expansion Device Identification data. If a device indicates it has a downloadable device driver the Opera System will issue the Download Driver Command to the device. The device must place into its Data Return Fifo the number of words indicated by the Device Driver Size. The Opera System will download this data using RD_DATA transactions and install the device driver. If the device driver is larger than 1K bytes the data will be

transferred using a 1K Chunk size and the control flow defined above for data transfers.

The Download Driver Command is optional. An expansion device only needs to support this command if the Device Driver bit in the Flag Byte 0 is a one. The Download Driver Command is a 7 byte command. Only the first byte of the Download Driver Command is of significance. The other 6 bytes are reserved for future use. The first byte of the Download Driver Command is 87H.

L. Errors Conditions & Recovery

If the Opera System ever performs a RD_STAT or a RD_DATA transaction and the corresponding FIFO does not contain any data the expansion device returns immediately with an undefined piece of data. This, however, causes no side effects at the device.

If the Opera System performs a WR_COM or a WR_DATA transaction and the corresponding FIFO is full the expansion device ignores the data and indicates an error to the Opera System. The expansion device should continue operation using the available data and commands as best it can. The expansion device, however, may indicate an error with commands in progress if a FIFO overflow occurs.

M. Reset and Power Up Conditions

After power up or the assertion of the RESET-signal the following conditions must be true at an expansion device:

· The expansion device is not selected

· The expansion device is in address configuration mode

· The IDOUT bit is low

· The Media Access bit is SET (high)

· The Interrupt Disable- bit is Low (interrupts are disabled)

· The Status Return FIFO is empty

· The Data Return FIFO is empty
· The Data Write FIFO is empty
· The Command FIFO is empty

During assertion of the Reset- bit in the Poll Register only the hardware of the Expansion device is affected. The ChunkValid- or StatValid- bits might be affected because the Status Return FIFO and Data Return FIFO are affected. There is no direct affect on the Poll Register. The following must be true:

· The Status Return FIFO is empty
· The Data Return FIFO is empty
· The Data Write FIFO is empty
· The Command FIFO is empty

III. ELECTRICAL CHARACTERISTICS

A. Timing

Expansion devices must only drive the ADB[7:0] bus when they have been selected, the transaction is a RD_DATA transaction, and the STB- signal is asserted. An expansion device must drive the RDY- signal whenever it has been selected. Figs. 15, 16, 17, 18 and 19 show the timing guaranteed to an expansion device. This does not correspond exactly with the timing which is provided by the Opera System, which is slightly different to account for cable delays and expander unit delays. Note that the exact time periods specified for each of the parameters identified in these Figs. are not important to an understanding of the invention and are omitted for clarity.

B. Termination

All signals except the INT- signal are terminated at both ends with 75 Ohm series resistors. The Opera System includes this termination and expansion devices include this termination on all signals except INT-. The INT- line contains no termination and is connected to +5 Volts via a 1K Ohm resistor in the Opera System. The

RDY- line is connected to +5 Volts via a 4.7K Ohm resistor in the Opera System. This is to hold the RDY- line false when no expansion device is selected.

C. External Expansion Device Maximum Loads

An expansion device must not place more than 60pF of load on any signal of the Opera Expansion Bus. An expansion device must not connect more than 6 inches of PC board trace to any signal of the Opera Expansion Bus. Note that this limitation does not include the cable which connects the Opera System to the expansion device. An expansion device must not draw more than 2 standard TTL loads of current.

D. External Expansion Device Maximum Cable Lengths

The cable which connects the Opera System to an expansion device must not exceed 20 inches in length.

E. Signal Levels

All signals used on the Opera Expansion Bus are TTL level signals.

F. Non-Powered Expansion Devices

It is possible that an Opera Expansion device which is connected to the external Expansion Bus is not powered on. There must be no adverse affects caused by such a situation. The requirements are first that the IDOUT pin of the expansion device must maintain a legal TTL high when the expansion device is not powered. Note that this signal is connected to 4.5 Volts via a 10K Ohm resistor in the Opera System or the Expander Unit. Second, all signals of the expansion device should not draw more than a predefined maximum current when the expansion device is not powered.

IV. EXPANSION DEVICE BUS INTERFACE

Given the description set forth above regarding the manner in which identification codes are assigned to expansion devices on the expansion bus, a wide variety

of circuits for implementing the procedure in an expansion device will be apparent to the person of ordinary skill. One illustrative example of such a circuit is shown in Fig. 20. It comprises a D flip-flop 2002 having a D input connected to receive the IDin signal and a Q output connected to provide an IDout signal to a subsequent device. An inverting clear input is connected to receive the RESET- signal and the clock input is connected to receive the STB- signal from the expansion bus 2004. The IDin signal is also connected to an inverting enable input of a counter 2006, an inverting clear input of which is connected to receive the RESET- signal. The clock input of counter 2006 is connected to receive the STB- signal. The RESET- signal, the STB- signal and other address and control signals on the expansion bus 2004 are coupled to an interface circuit 2008, which interprets its input signals to provide an ADDR output indicating the address of an access request being received over the expansion bus 2004. The count (Q) output of counter 2006 and the ADDR output of interface circuit 2008 are connected to respective A and B inputs of a comparator 2010, which generates a MATCH signal on its A = B output.

In operation, when RESET- is asserted, both the D flip-flop 2002 and the counter 2006 are cleared. Thus the expansion device outputs a logic 0 (inactive) on the IDout line and the counter outputs a count of 0. After RESET- goes high, assuming IDin remains low, each rising edge of a pulse on STB- causes the counter 2006 to increment by 1. When IDin goes high (asserted) from a previous device, or from the opera system or expander unit, the counter 2006 will stop counting pulses on the STB- line. Also, on the rising edge of STB- immediately following the assertion of IDin, D flip-flop 2002 will assert IDout to the next expansion device. It can be seen, therefore, that counter 2006 counts the number of

pulses which occur on the STB- line while the RESET- line is de-asserted and before the expansion device receives an asserted signal on the IDin line. The expansion device also asserts its IDout signal in response to the first pulse which the expansion device receives after it receives an asserted signal on its IDin line. In subsequent accesses over the expansion bus 2004, the expansion device compares the address provided on the bus to the count at which the counter 2006 stopped counting, to determine whether the access is intended for the particular expansion device.

Note that though the identification code assignment procedure begins in response to the de-assertion of a reset signal, such de-assertion of a reset signal can equally be considered as the assertion of a start-procedure signal.

V. GENERAL SYSTEM OPERATION

The Opera System performs 17 Select transactions under software control to generate the STB- pulses needed after Reset to determine address assignments. The WR_COM and WR_POL transactions are performed more slowly than the other transactions to allow time for the RDY- signal to return from the expansion device. The Upper Bit of the Address in RD_POL, RD_DAT, RD_STAT transactions are used to control the direction of the Expansion Bus Buffers. This implies that expansion devices must ignore the upper most bit of the address during a SELECT transaction. Expansion Devices should ignore the 4 upper bits of the address during a SELECT Transaction.

An Opera system which desires two Expansion Bus sockets on the back of the unit should have two sets of Expansion Bus Buffers. The second set of Buffers should have built in address comparators to determine drive direction of the buffers. Note also that the Opera

System requires at least one device connected to the internal expansion bus.

The invention has been described with respect to particular embodiments thereof, and it will be understood that numerous variations are possible without departing from its scope.

CLAIMS

1. A method for assigning identification codes to a plurality of peripheral devices, coupled in part in a daisy chain, comprising the steps of:

asserting a first signal to all of said peripheral devices to indicate the start of an identification code assignment procedure; and

asserting a series of pulses to all of said peripheral devices, beginning after said assertion of said first signal,

each of said peripheral devices except a first and a last one of said peripheral devices asserting a logic transition to a respective subsequent one of said peripheral devices in said daisy chain in response to one of said pulses which occurs after the peripheral device receives said logic transition from a prior one of said peripheral devices in said daisy chain,

said first peripheral device asserting said logic transition to a respective subsequent one of said peripheral devices in said daisy chain in response to one of said pulses which occurs after said assertion of said first signal,

each given one of said peripheral devices deriving a respective unique identification code from the number of said pulses which occur between said assertion of said first signal and receipt by said given peripheral device of said logic transition.

2. A method according to claim 1, further comprising the step of asserting said logic transition to said first one of said peripheral devices.

3. A method according to claim 2, wherein said last one of said peripheral devices asserts said logic transition to an unconnected lead in response to one of said pulses which occurs after said last peripheral device receives said logic transition from a prior one of said peripheral devices in said daisy chain.

4. A method for assigning identification codes to a plurality of peripheral devices, coupled in part in a daisy chain, comprising the steps of:

asserting a first signal to all of said peripheral devices to indicate the start of an identification code assignment procedure;

asserting a series of pulses to all of said peripheral devices, beginning after said assertion of said first signal; and

asserting a logic transition to a first one of said peripheral devices in said daisy chain,

each given one of said peripheral devices asserting said logic transition to a respective subsequent one of said peripheral devices in said daisy chain in response to the first one of said pulses which occurs after the given peripheral device receives said logic transition from a prior one of said peripheral devices in said daisy chain, at least one of said peripheral devices in said daisy chain asserting said logic signal to an unconnected lead,

each given one of said peripheral devices counting the number of said pulses which occur between said assertion of said first signal and receipt by the given peripheral device of said logic transition, to derive a unique identification code.

5. Apparatus for communicating with a plurality of expansion devices, comprising:

a host system;

a RESET control line driven by said host system and parallel-coupled to all of said expansion devices;

a STROBE control line driven by said host system and parallel-coupled to all of said expansion devices;

an ID control line driven by said host system for coupling in a daisy-chained manner through all of said expansion devices;

means in said host system for de-asserting said RESET control line and said ID control line and for subsequently pulsing said STROBE line a plurality of times; and

means in each given one of said expansion devices for, while RESET is de-asserted, (1) asserting its ID output to the next expansion device in response to the first one of said STROBE pulses following receipt by said given expansion device of an asserted ID from the previous expansion device, and (2) counting the number of said STROBE pulses which occur while RESET is de-asserted and before receipt by said given expansion device of said asserted ID from the previous expansion device, the resulting count being indicative of an identification number for said given expansion device.

6. An expansion peripheral device for coupling to an expansion bus of a computer-based system, comprising:

a RESET input couplable to said bus;

a STROBE input couplable to said bus;

an ID input couplable to a prior device on said bus;

an ID output couplable to a subsequent device on said bus; and

means for, while said RESET input is de-asserted, (1) asserting said ID output in response to the first pulse received by said expansion device following receipt by said expansion device of an asserted signal on said ID input, and (2) counting the number of pulses which occur on said STROBE input while said RESET input is de-asserted and before receipt by said expansion device of said asserted signal on said ID input, the resulting count being indicative of an identification number for said expansion device.

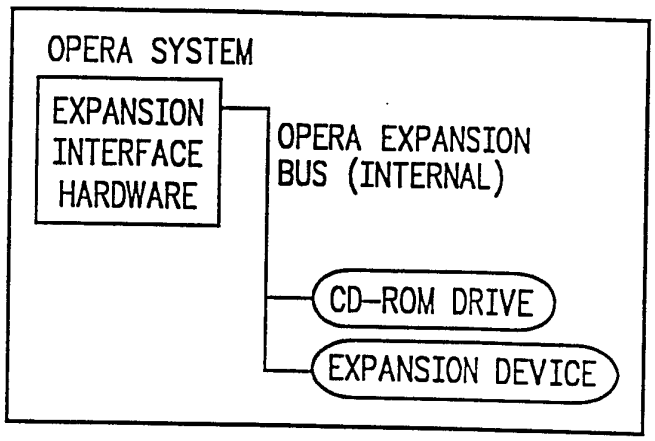


FIG. 1

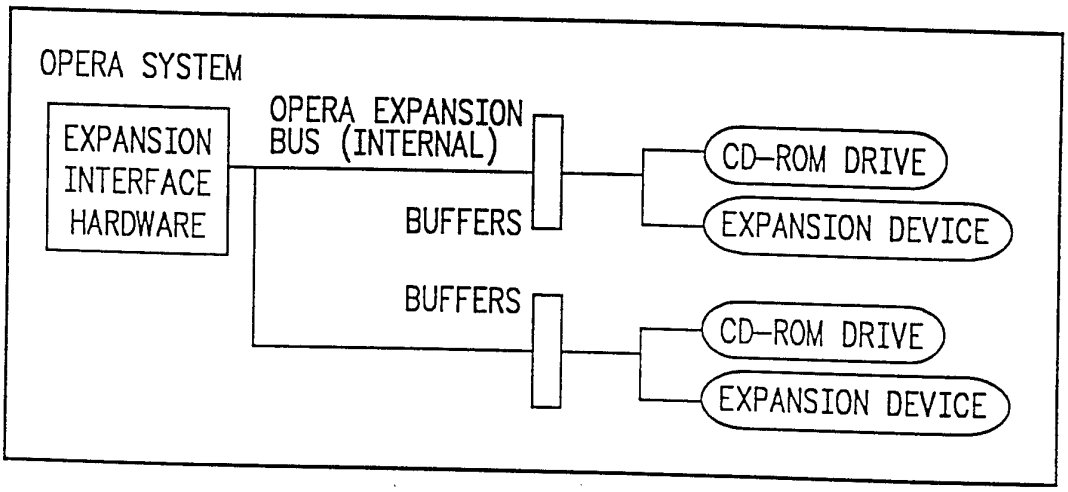


FIG. 2

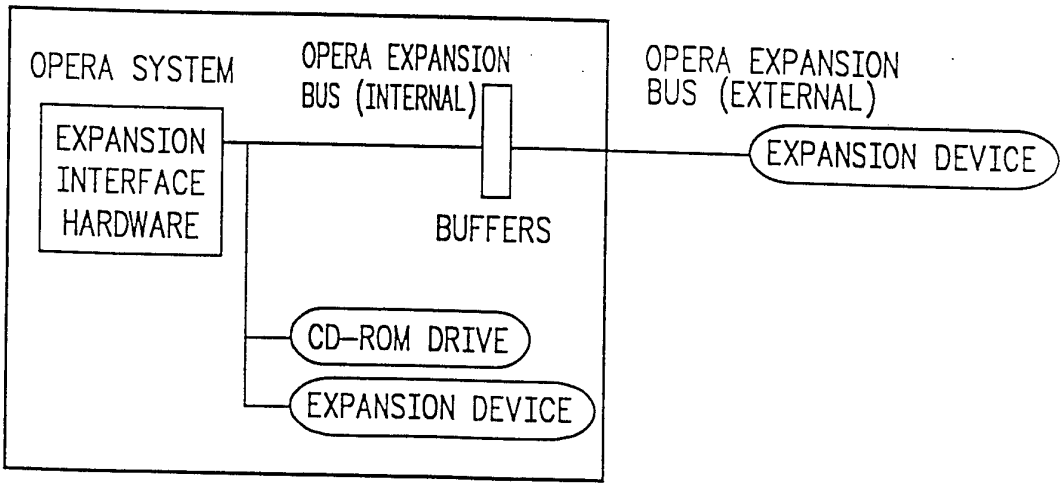


FIG. 3

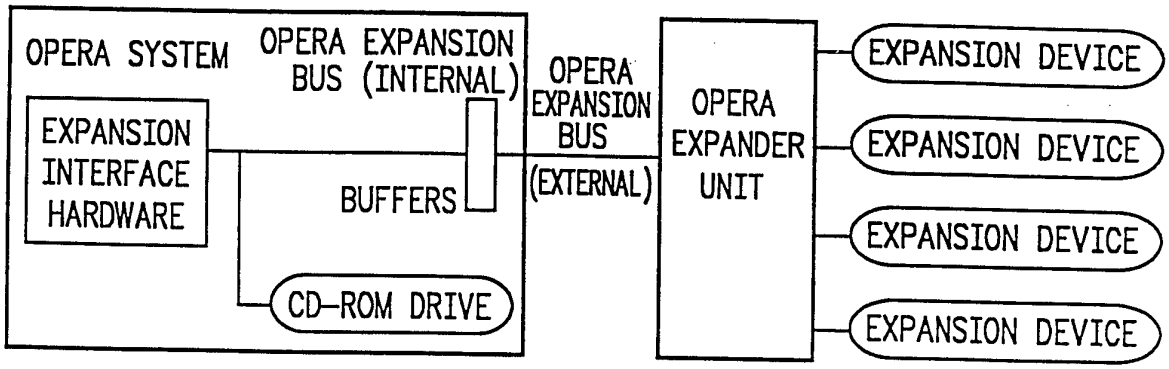

OPERA SYSTEM
OPERA EXPANSION BUS (INTERNAL)
EXPANSION INTERFACE HARDWARE
BUFFERS
CD-ROM DRIVE
OPERA EXPANSION BUS (EXTERNAL)
OPERA EXPANDER UNIT
EXPANSION DEVICE
EXPANSION DEVICE
EXPANSION DEVICE
EXPANSION DEVICE

FIG. 4

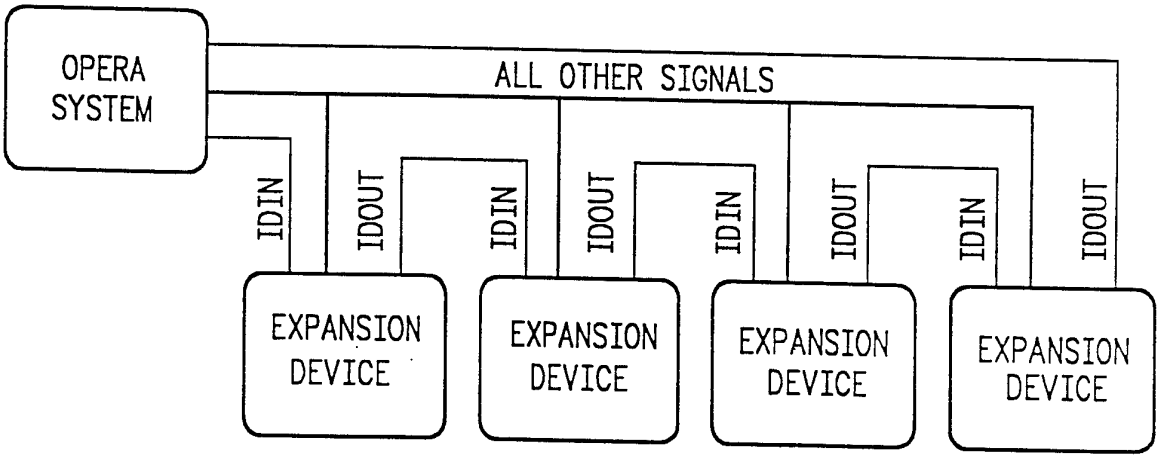

OPERA SYSTEM
ALL OTHER SIGNALS
IDIN
IDOUT
IDIN
IDOUT
IDIN
IDOUT
IDIN
IDOUT
EXPANSION DEVICE
EXPANSION DEVICE
EXPANSION DEVICE
EXPANSION DEVICE

FIG. 5

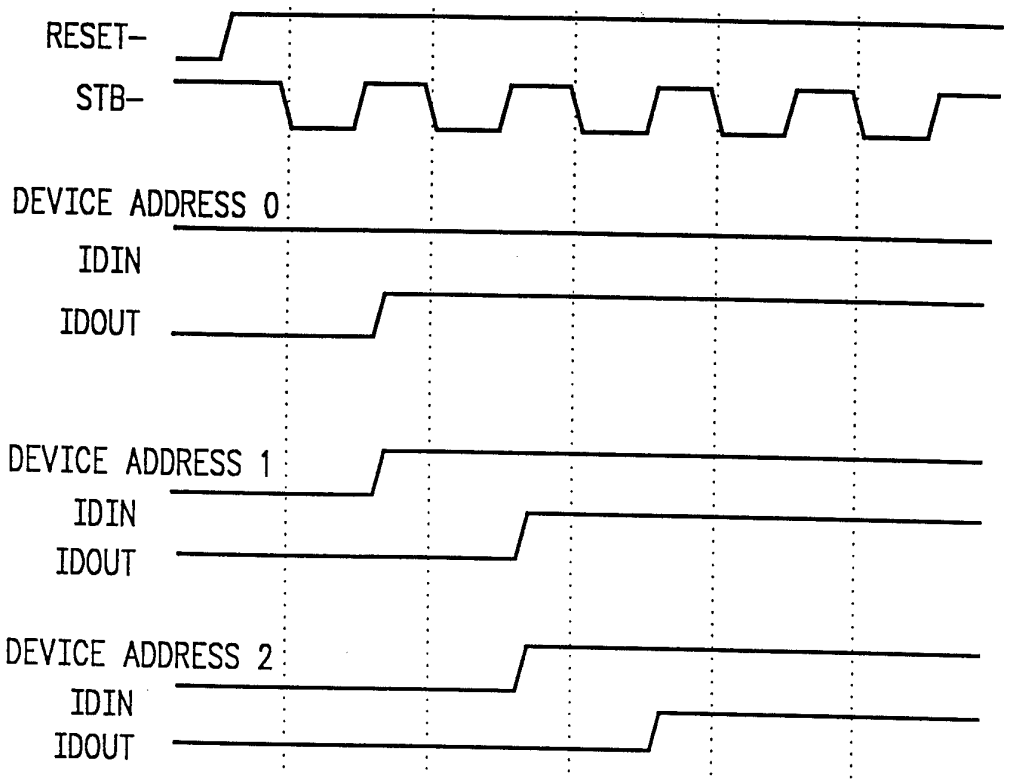

RESET-
STB-
DEVICE ADDRESS 0
IDIN
IDOUT
DEVICE ADDRESS 1
IDIN
IDOUT
DEVICE ADDRESS 2
IDIN
IDOUT

FIG. 6

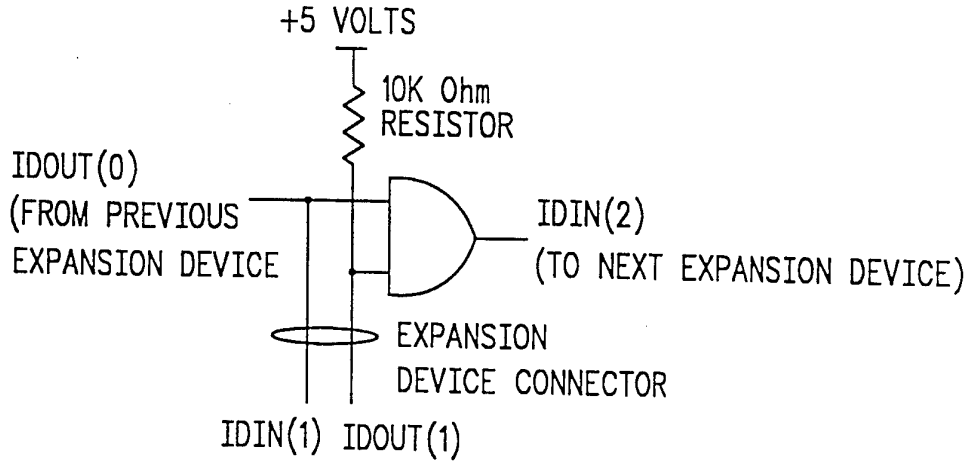

+5 VOLTS
10K Ohm
RESISTOR
IDOUT(0)
(FROM PREVIOUS
EXPANSION DEVICE
IDIN(2)
(TO NEXT EXPANSION DEVICE)
EXPANSION
DEVICE CONNECTOR
IDIN(1) IDOUT(1)

FIG. 7

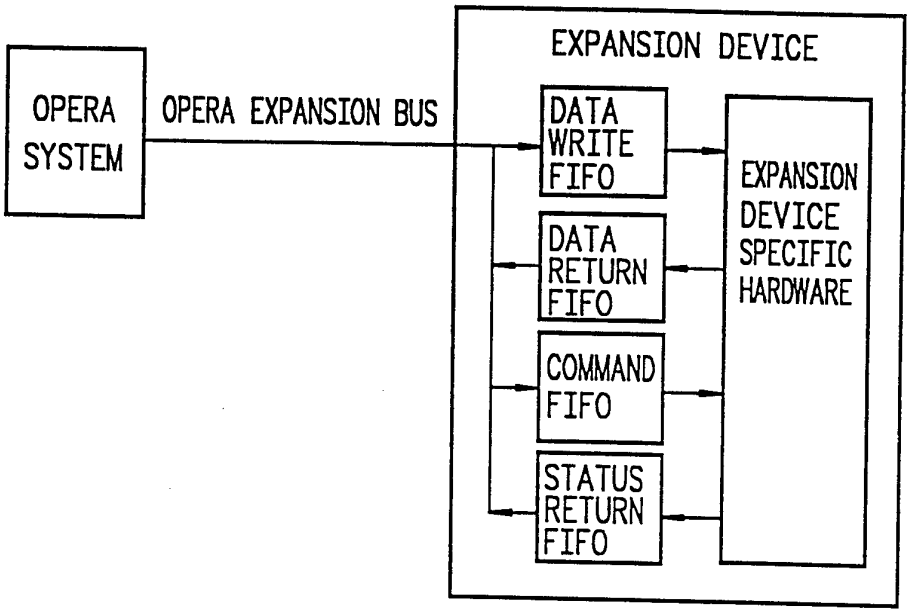

OPERA SYSTEM
OPERA EXPANSION BUS
EXPANSION DEVICE
DATA WRITE FIFO
DATA RETURN FIFO
COMMAND FIFO
STATUS RETURN FIFO
EXPANSION DEVICE SPECIFIC HARDWARE

FIG. 8

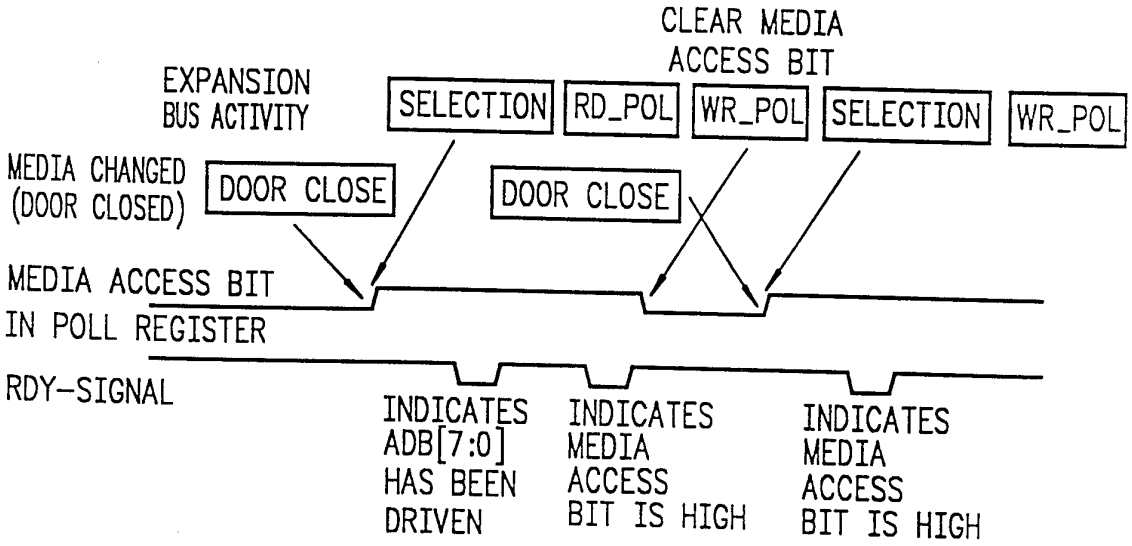

CLEAR MEDIA ACCESS BIT
EXPANSION BUS ACTIVITY
SELECTION
RD_POL
WR_POL
SELECTION
WR_POL
MEDIA CHANGED (DOOR CLOSED)
DOOR CLOSE
DOOR CLOSE
MEDIA ACCESS BIT IN POLL REGISTER
RDY-SIGNAL
INDICATES ADB[7:0] HAS BEEN DRIVEN
INDICATES MEDIA ACCESS BIT IS HIGH
INDICATES MEDIA ACCESS BIT IS HIGH

FIG. 9

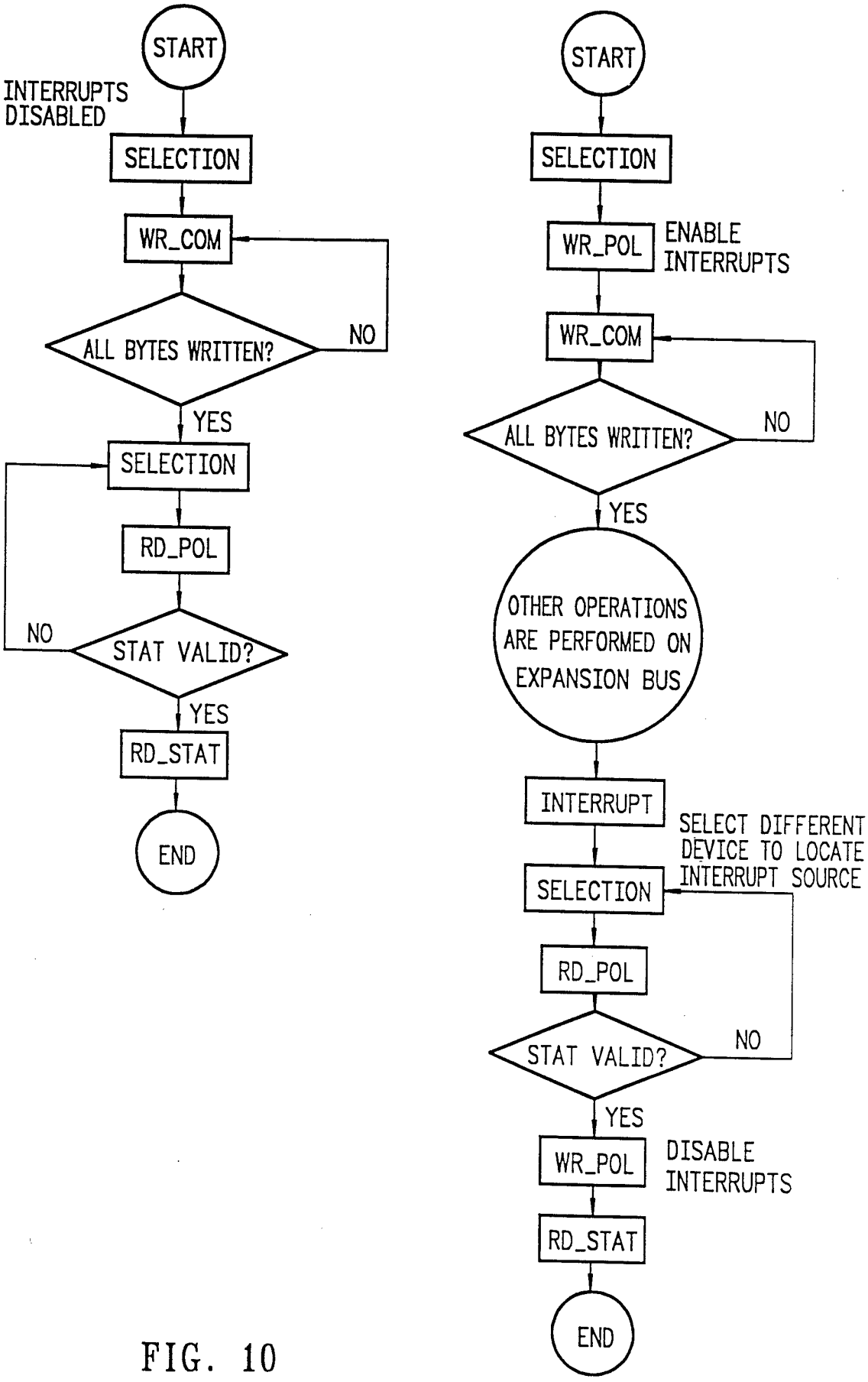

START
INTERRUPTS DISABLED
SELECTION
WR_COM
ALL BYTES WRITTEN?
NO
YES
SELECTION
RD_POL
NO
STAT VALID?
YES
RD_STAT
END
START
SELECTION
WR_POL
ENABLE INTERRUPTS
WR_COM
ALL BYTES WRITTEN?
NO
YES
OTHER OPERATIONS ARE PERFORMED ON EXPANSION BUS
INTERRUPT
SELECT DIFFERENT DEVICE TO LOCATE INTERRUPT SOURCE
SELECTION
RD_POL
STAT VALID?
NO
YES
WR_POL
DISABLE INTERRUPTS
RD_STAT
END

FIG. 10

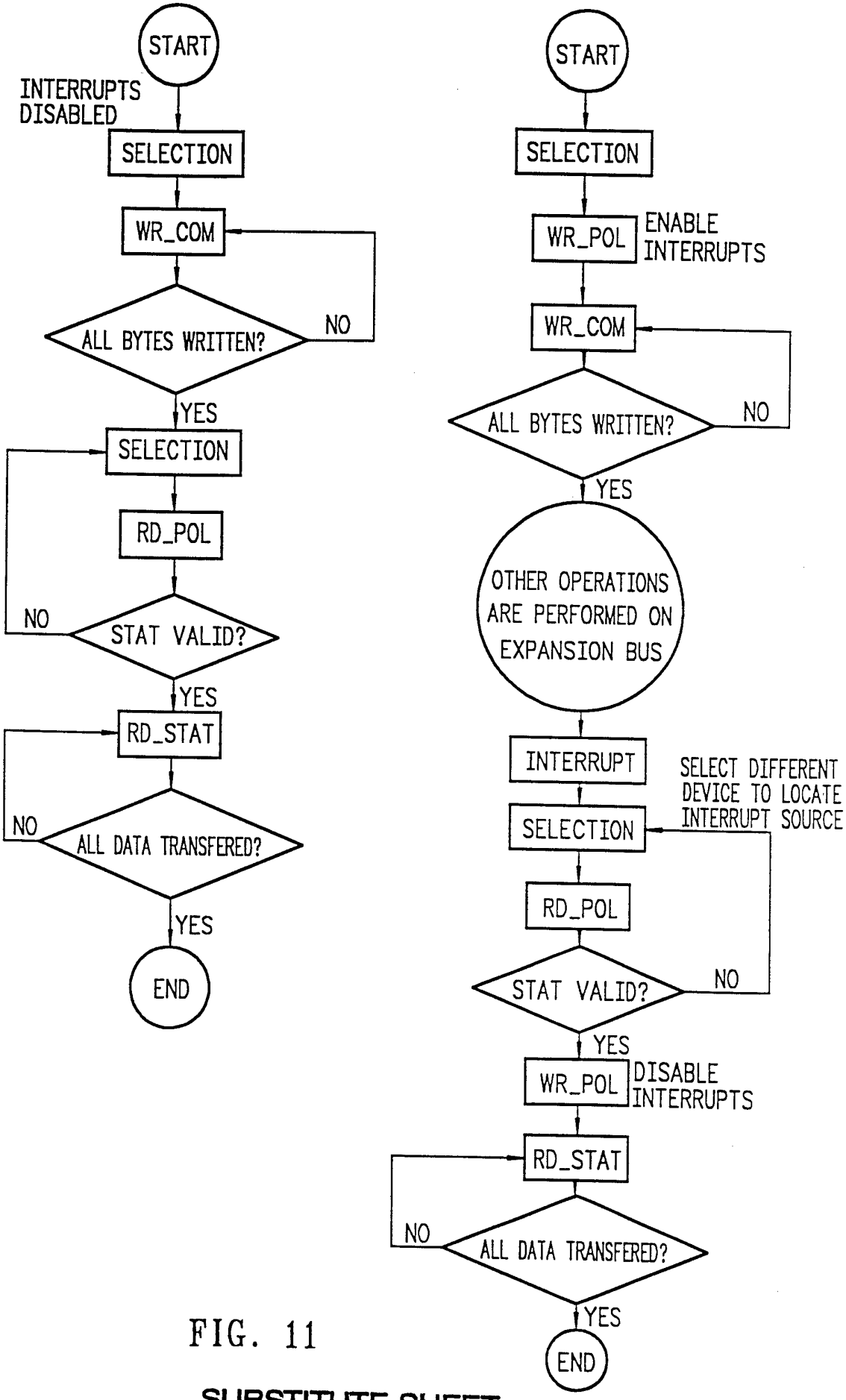

START
INTERRUPTS DISABLED
SELECTION
WR_COM
ALL BYTES WRITTEN?
NO
YES
SELECTION
RD_POL
NO
STAT VALID?
YES
RD_STAT
NO
ALL DATA TRANSFERED?
YES
END
START
SELECTION
WR_POL
ENABLE INTERRUPTS
WR_COM
ALL BYTES WRITTEN?
NO
YES
OTHER OPERATIONS ARE PERFORMED ON EXPANSION BUS
INTERRUPT
SELECT DIFFERENT DEVICE TO LOCATE INTERRUPT SOURCE
SELECTION
RD_POL
STAT VALID?
NO
YES
WR_POL
DISABLE INTERRUPTS
RD_STAT
NO
ALL DATA TRANSFERED?
YES
END

FIG. 11

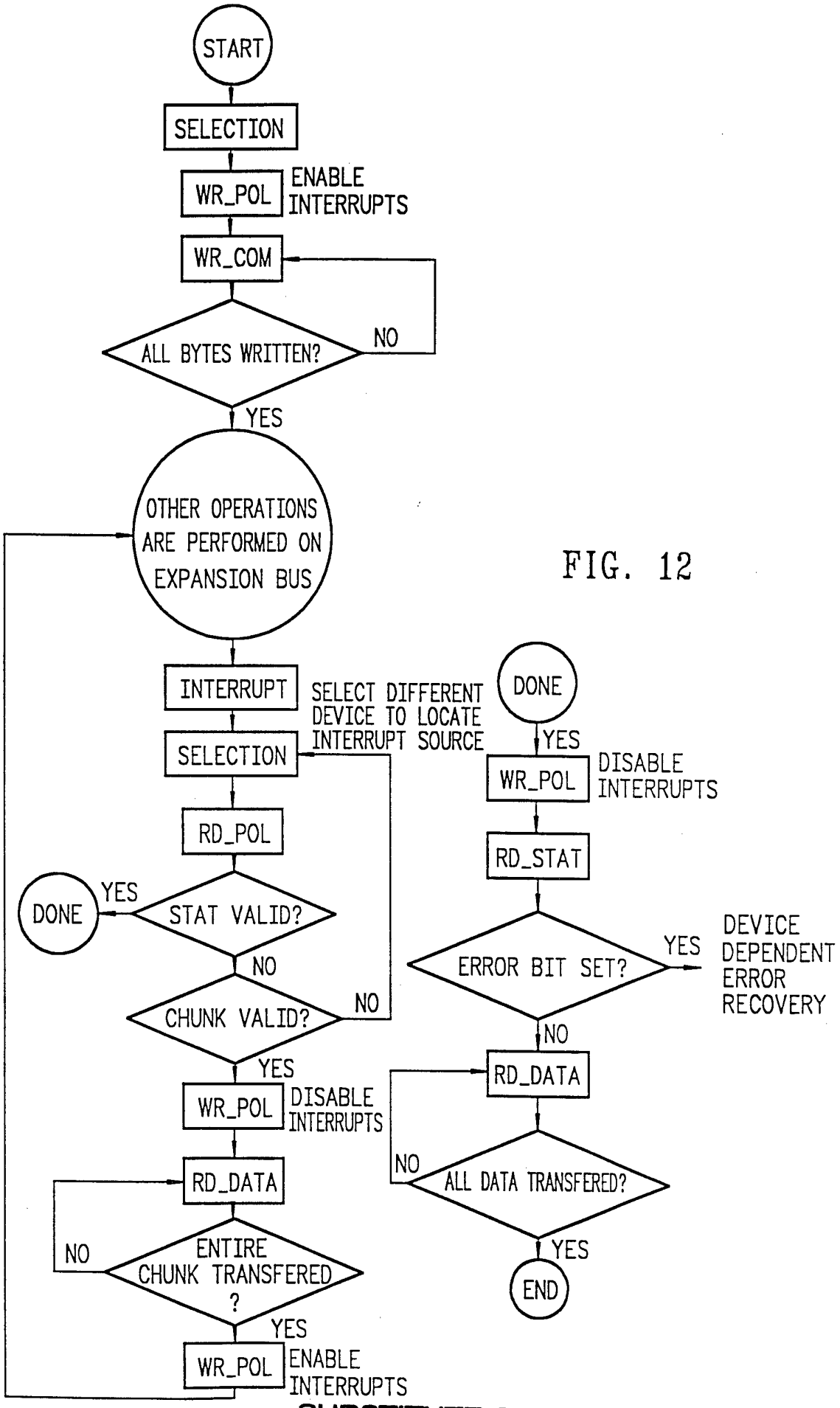

START
SELECTION
WR_POL
ENABLE INTERRUPTS
WR_COM
ALL BYTES WRITTEN?
NO
YES
OTHER OPERATIONS ARE PERFORMED ON EXPANSION BUS
INTERRUPT
SELECT DIFFERENT DEVICE TO LOCATE INTERRUPT SOURCE
SELECTION
RD_POL
DONE
YES
STAT VALID?
NO
CHUNK VALID?
NO
YES
WR_POL
DISABLE INTERRUPTS
RD_DATA
NO
ENTIRE CHUNK TRANSFERED ?
YES
WR_POL
ENABLE INTERRUPTS
DONE
YES
WR_POL
DISABLE INTERRUPTS
RD_STAT
ERROR BIT SET?
YES
DEVICE DEPENDENT ERROR RECOVERY
NO
RD_DATA
NO
ALL DATA TRANSFERED?
YES
END

FIG. 12

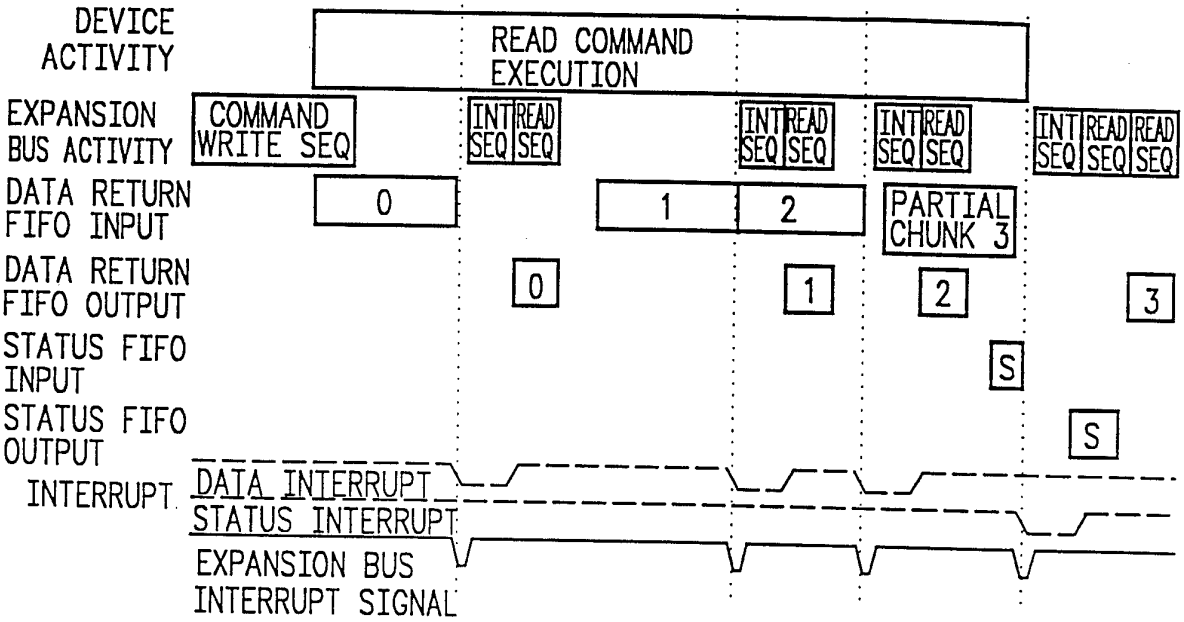

DEVICE ACTIVITY
READ COMMAND EXECUTION
EXPANSION BUS ACTIVITY
COMMAND WRITE SEQ
INT SEQ
READ SEQ
DATA RETURN FIFO INPUT
0
1
2
PARTIAL CHUNK 3
DATA RETURN FIFO OUTPUT
STATUS FIFO INPUT
S
STATUS FIFO OUTPUT
INTERRUPT
DATA INTERRUPT
STATUS INTERRUPT
EXPANSION BUS INTERRUPT SIGNAL

FIG. 13

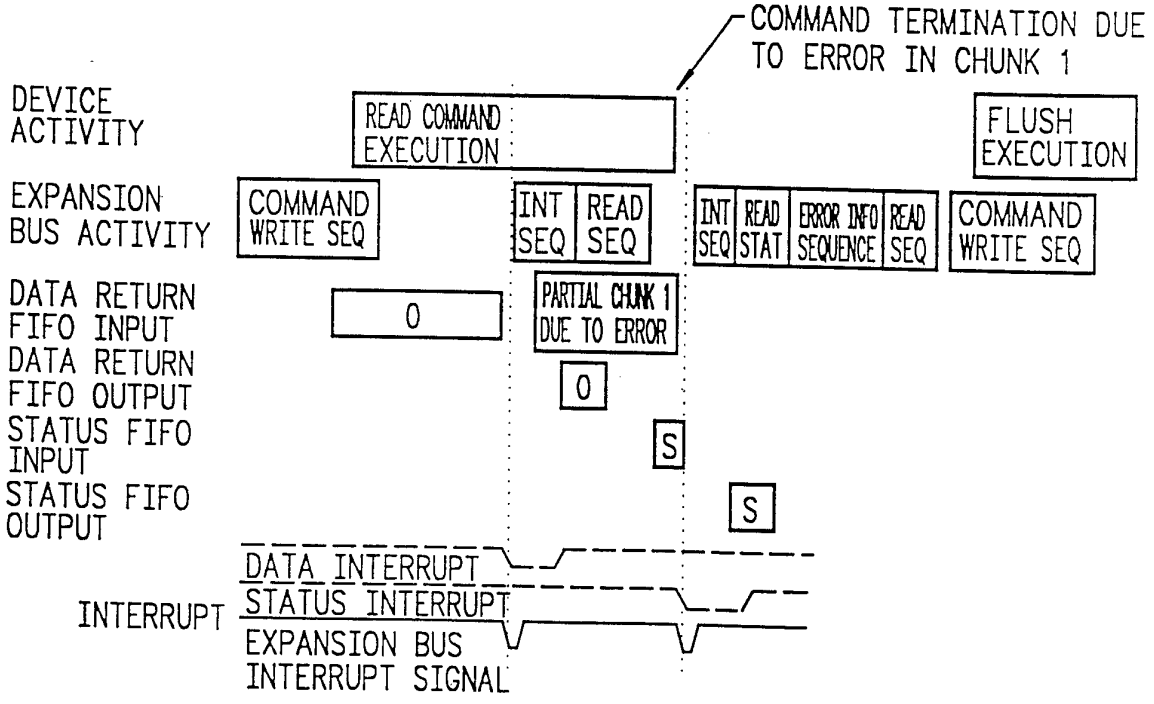

COMMAND TERMINATION DUE TO ERROR IN CHUNK 1
DEVICE ACTIVITY
READ COMMAND EXECUTION
FLUSH EXECUTION
EXPANSION BUS ACTIVITY
COMMAND WRITE SEQ
INT SEQ
READ SEQ
READ STAT
ERROR INFO SEQUENCE
DATA RETURN FIFO INPUT
0
PARTIAL CHUNK 1 DUE TO ERROR
DATA RETURN FIFO OUTPUT
STATUS FIFO INPUT
S
STATUS FIFO OUTPUT
INTERRUPT
DATA INTERRUPT
STATUS INTERRUPT
EXPANSION BUS INTERRUPT SIGNAL

FIG. 14

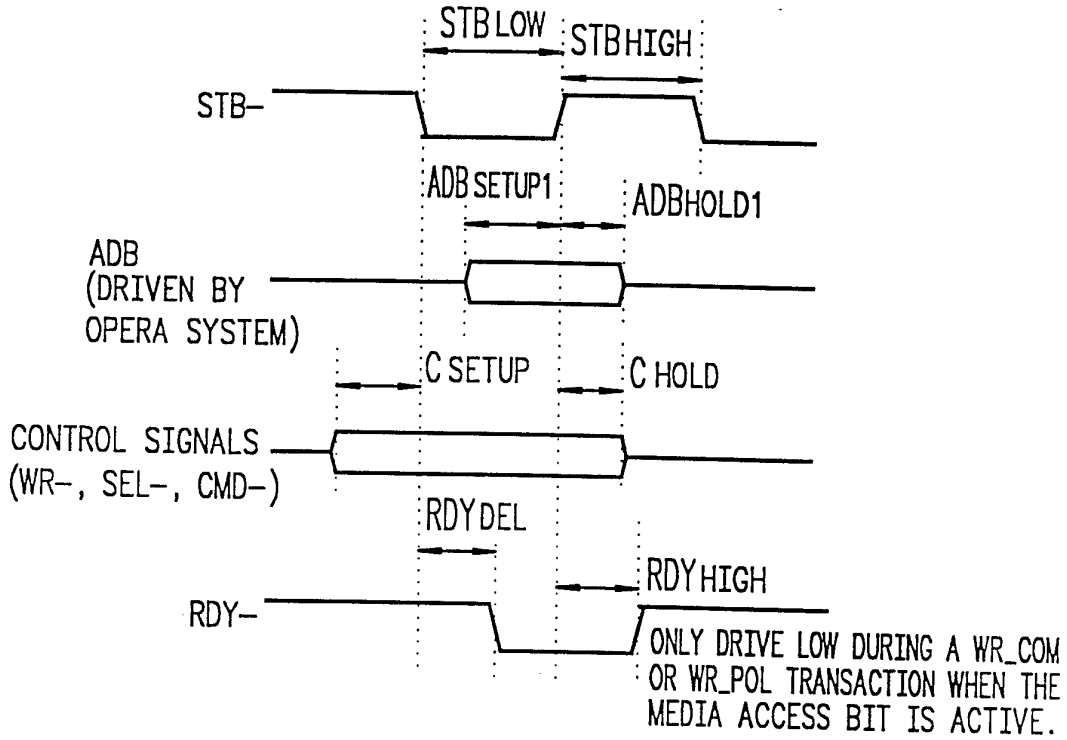

STB LOW
STB HIGH
STB–
ADB SETUP1
ADB HOLD1
ADB
(DRIVEN BY
OPERA SYSTEM)
C SETUP
C HOLD
CONTROL SIGNALS
(WR–, SEL–, CMD–)
RDY DEL
RDY HIGH
RDY–
ONLY DRIVE LOW DURING A WR_COM
OR WR_POL TRANSACTION WHEN THE
MEDIA ACCESS BIT IS ACTIVE.

FIG. 15

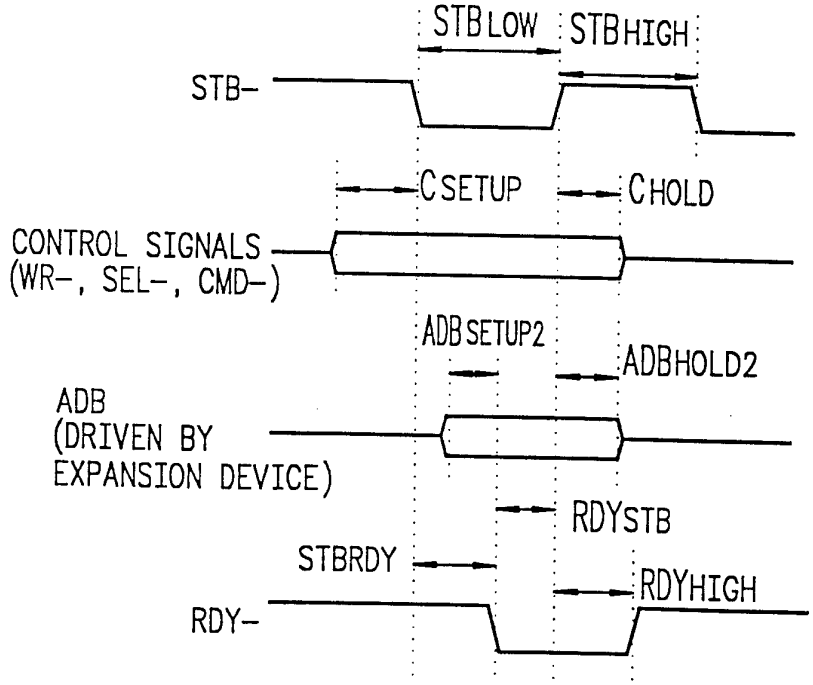
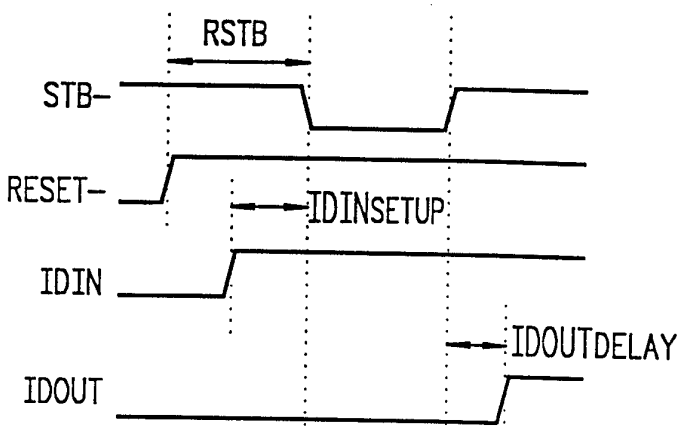

STB LOW
STB HIGH
STB–
C SETUP
C HOLD
CONTROL SIGNALS
(WR–, SEL–, CMD–)
ADB SETUP2
ADB HOLD2
ADB
(DRIVEN BY
EXPANSION DEVICE)
RDY STB
STB RDY
RDY HIGH
RDY–

FIG. 16



IDIN AND IDOUT TIMING

FIG. 17

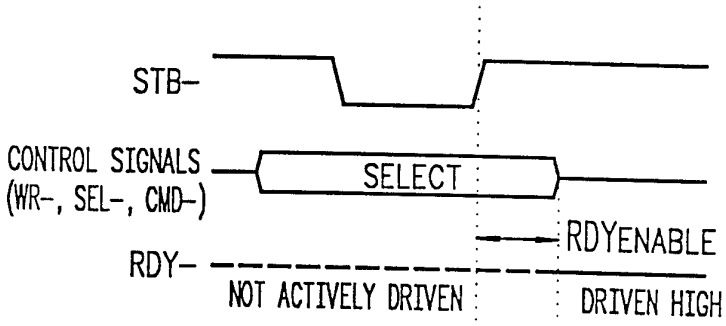


FIG. 18

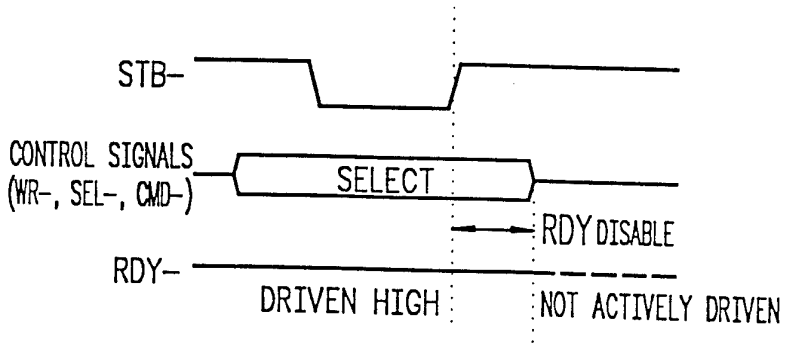


FIG. 19

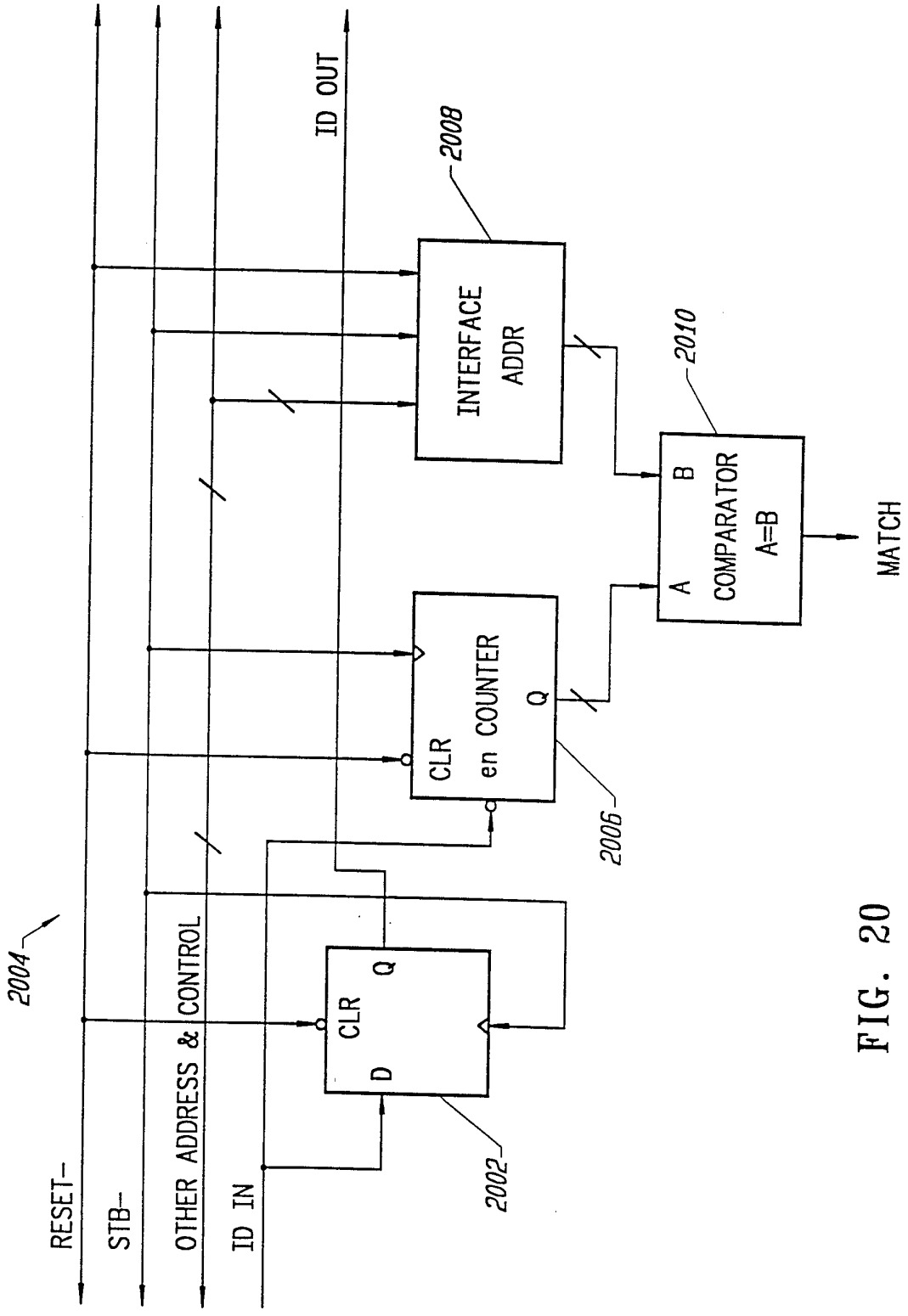

RESET-
STB-
OTHER ADDRESS & CONTROL
ID IN
ID OUT
2004
2002
CLR
D
Q
2006
CLR
en
COUNTER
Q
2008
INTERFACE
ADDR
2010
A
B
COMPARATOR
A=B
MATCH

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/00117

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 9/00,9/26
US CL :395/275,800; 340/825.65,825.67,825.05
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/275, 800; 340/825.65, 825.67, 825.05

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

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

APS TEXT SEARCH, REPORT ATTACHED.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US,A, 4,360,870 (MCVEY) 23 NOVEMBER 1982 See figures 2 and 3, col. 2, line 60- col. 3, line 2.	1,4-6
Y	US,A, 4,617,566 (DIAMOND) 14 OCTOBER 1986 See figure 1	1,4-6
Y	US,A, 4,155,075 (WECKENMANN ET AL) 15 MAY 1979 See figures 2-3, col. 2, lines 3-26	1-2,4-6
A	US,A, 4,114,138 (DEMERS) 12 SEPTEMBER 1978 See figures 1-2	1-6

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 25 FEBRUARY 1993	Date of mailing of the international search report 19 MAR 1993
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer MENG-AI T. AN Telephone No. (703) 305-9678

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

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/00117

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,E	US,A,5,179,670 (FARMWALD ET AL) 12 JANUARY 1993 See figures 1A and 3, Abstract, col. 1, line 59-col. 2, line 43	1-6