PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 1/32	A1	(11) International Publication Number: WO 93/06545
		(43) International Publication Date: 1 April 1993 (01.04.93)

(21) International Application Number: PCT/US92/04405

(22) International Filing Date: 26 May 1992 (26.05.92)

(30) Priority data:
759,024 13 September 1991 (13.09.91) US

(71) Applicant: WANG LABORATORIES, INC. [US/US]; One Industrial Avenue, MS 014-B7D, Lowell, MA 01851 (US).

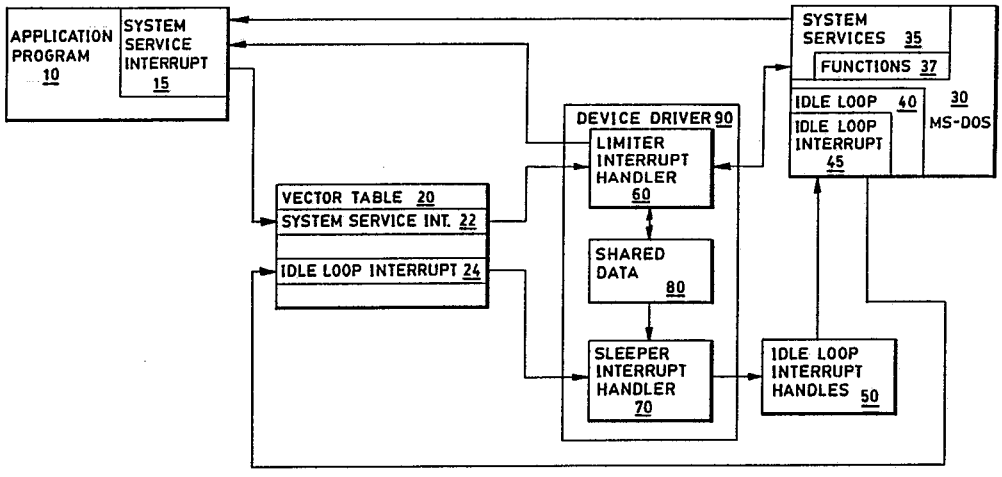
(72) Inventors: BARRETT, David, N. ; 45 Beverlee Road, Tyngsboro, MA 01879 (US). MARTIN, Patricia, A. ; 120 Hayden Road, Groton, MA 01450 (US).

(74) Agents: SHANAHAN, Michael, H. et al.; One Industrial Avenue, MS 014-B7D, Lowell, MA 01851 (US).

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

Published
With international search report.

(54) Title: POWER SAVINGS WITH MS-DOS IDLE LOOP



(57) Abstract

In a portable computer, a device driver program causes the system to conserve power whenever the processor is idle. The program detects the idle processor via a unique interrupt generated by the operating system's idle loop. After detecting the idle state, the program conserves power usage by slowing the system clock speed and halting the processor. The system returns to its normal operating state when the processor detects an external interrupt. Although an operator action generates an external interrupt, the 55 ms periodic timer typically awakens the processor. In addition, the program blocks the power saving measures in cases where the operator would notice performance degradation.

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	**FI**	Finland	**MN**	Mongolia
AU	Australia	**FR**	France	**MR**	Mauritania
BB	Barbados	**GA**	Gabon	**MW**	Malawi
BE	Belgium	**GB**	United Kingdom	**NL**	Netherlands
BF	Burkina Faso	**GN**	Guinea	**NO**	Norway
BG	Bulgaria	**GR**	Greece	**NZ**	New Zealand
BJ	Benin	**HU**	Hungary	**PL**	Poland
BR	Brazil	**IE**	Ireland	**PT**	Portugal
CA	Canada	**IT**	Italy	**RO**	Romania
CF	Central African Republic	**JP**	Japan	**RU**	Russian Federation
CG	Congo	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CH	Switzerland			**SE**	Sweden
CI	Côte d'Ivoire	**KR**	Republic of Korea	**SK**	Slovak Republic
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CS	Czechoslovakia	**LK**	Sri Lanka	**SU**	Soviet Union
CZ	Czech Republic	**LU**	Luxembourg	**TD**	Chad
DE	Germany	**MC**	Monaco	**TG**	Togo
DK	Denmark	**MG**	Madagascar	**UA**	Ukraine
ES	Spain	**ML**	Mali	**US**	United States of America

POWER SAVINGS WITH MS-DOS IDLE LOOP

Background of the Invention

Portable computers allow users to take computing power wherever they go. As portable computers become smaller and lighter, their mobility increases. In addition, a portable computer's usefulness increases as processing power and battery life are increased.

All portables, however, are powered by batteries. The demand for smaller and lighter portables place restrictions on battery size. Presently, the battery pack in a six pound portable computer usually weighs approximately one pound. Thus, a portable computer's size and weight is significantly influenced by the battery. Manufacturers, therefore, are driven to produce portable computers with even smaller batteries.

At the same time, manufacturers are driven to produce more powerful portables. Manufacturers have increased processing speeds and available memory. As a result, portable computers are approaching the capabilities of stationary personal computers. These performance increases, however, place a greater drain on battery power.

A typical battery pack will power a notebook size computer for approximately two hours of continuous use. A portable computer's usefulness, as a result, is limited. By carrying a spare battery pack, a user can extend the portable computers usefulness another two hours. However, the additional battery pack reduces the computer's portability.

Summary of the Invention

It is desirable to reduce the drain of battery power without adversely affecting the user. In accordance with the present invention, battery power is conserved without any effort from the user or any noticeable affect on the computer's performance.

The present invention extends battery life by decreasing the power drain while the portable computer is idle. Whenever the operating system waits for an event, it enters an idle sequence. The present invention takes advantage of the idle sequence.

The invention comprises an idle state detector, which detects an idle sequence, and a sleeper, which provides an instruction to halt the processor when the idle state detector detects the idle sequence. In the preferred embodiment, the idle sequence generates a unique interrupt. The unique interrupt, in turn, triggers a specific interrupt handler. By thus detecting the unique interrupt, the specific interrupt handler detects the idle state. The sleeper is implemented as the specific interrupt handler program. Whenever the sleeper program executes, the processor is in the idle state.

In particular, the preferred sleeper interrupt handler reduces the system clock speed, turns down hardware devices having a power saving mode, and halts the processor. The processor remains halted until an external interrupt occurs. After the processor services the external interrupt, processing continues until the processor returns to the idle sequence.

In the preferred embodiment, there are identified situations where executing the sleeper program leads to noticeable performance degradation. In response, the Applicant's invention includes a limiter, which detects the identified situations and prevents the sleeper from executing. In particular, the limiter sets a flag in shared data while an identified system service function is executing and maintains a cleared flag otherwise. In the preferred embodiment, both the sleeper and the limiter are interrupt handler programs installed as device drivers. Application program calls to MS-DOS system services trigger execution of the limiter program. Any program call to the MD-DOS idle sequence causes the sleeper program to execute.

Brief Description of Drawings

Fig. 1 shows the logical system overview of a normal computer system operating under Microsoft DOS.

Fig. 2 shows the logical system overview of a computer system operating under Microsoft DOS and enhanced by the present invention.

Fig. 3 is a flowchart of the sleeper.

Fig. 4 is a flowchart of the limiter.

Description of the Preferred Embodiment

In the preferred embodiment, the portable computer is notebook

sized. The central processing unit (CPU) is an Intel model 80386SX microprocessor, normally operating at 16 MHz. The system is operated under Microsoft DOS (MS-DOS).

Fig. 1 depicts the normal operation of MS-DOS. The Intel family of processors support up to 256 interrupts, numbered 0 through 255. During initialization, the boot strap program creates a table 20 indexed by interrupt number for containing each interrupt's vector transfer address. The boot strap program, MS-DOS 30 and application programs 10 load the vectors for defined interrupts 22,24 into the table 20. The processor services interrupts by transferring control to the interrupt's vector address 22,24 which the processor looks up in the table 20. At the vector address is a program called an interrupt handler. Users, however, can cause a new interrupt handler to execute when the processor services a particular interrupt. Although there are several methods for implementing interrupt handlers, MS-DOS provides standardized mechanisms. See e.g., R. Duncan, Advanced MS-DOS, pps. 207-257 (Microsoft Press 1986).

Application programs 10 communicate with MS-DOS 30 using software interrupts. When an application 10 requests a system service, it issues the system service interrupt 15. The processor reads the vector address 22 for the system service interrupt 15 from the vector table 20 and transfers control to the MS-DOS System Service 35. In the preferred embodiment, the system service interrupt 15 is interrupt number 33 (21 hex). A code in the AH register identifies the particular system service function 37 requested by the application program 10.

The system service function 37 may need to wait for an event. If that is the case, the function 37 enters an idle loop 40. The processor continuously executes the instructions in the idle loop 40 until the awaited event occurs. When the event occurs, control returns to the function 37, which then returns control to the application 10. One instruction in the idle loop 40 is an instruction to issue an idle loop interrupt 45. In the preferred embodiment, the idle loop interrupt 45 is interrupt number 40 (28 hex).

Upon receiving the idle loop interrupt instruction 45, the processor

transfers control to the idle loop interrupt handler 50 based on the vector address 24. The idle loop interrupt handler 50 executes its software instructions and returns control to the idle loop 40.

As depicted in Fig. 2, the present invention installs an MS-DOS Device Driver 90 to modify the interrupt interface for the MS-DOS System Service calls 15 and the idle loop interrupt 45. R. Duncan, supra., discusses the details of implementing device drivers. In particular, the device driver's initialization routine checks the BIOS version, stores the original vector addresses for the system service interrupt 22 and idle loop interrupt 24 in shared data 80, and modifies the vector addresses 22,24 to point to the limiter 60 and sleeper 70 programs. By using the device driver approach, the present invention realizes a reduced memory requirement because MS-DOS reclaims the memory used for the first-time initialization instructions. The present invention is installed as a character device driver, with the initialization routine coded at the end of the driver. The initialization routine relays its start address to MS-DOS as the first usable memory address. Therefore, MS-DOS is free to reuse the memory occupied by the initialization routine.

An application program 10 calls MS-DOS System Services 35 using the system service interrupt 15. The present invention modifies the interrupt interface 20 to include a limiter interrupt handler program 60. Likewise, the instant invention supplies a sleeper interrupt handler program 70 for the idle loop interrupt 45. The limiter program 60 communicates with the sleeper program 70 via shared data 80.

Fig. 3 shows the flowchart of the sleeper program 70, which contains the actual power saving instructions. Upon entry, the sleeper program 70 checks 71 the flag in shared memory 80. If the flag is set then the limiter program 60 is restricting the functioning of the sleeper program 70 and the sleeper program 70 does not implement the power saving features. Instead, the sleeper program exits 78 by transferring control to the original idle loop interrupt handler 50 based on the original vector address. On the other hand, if the flag in shared memory 80 is not set then the limiter program 60

is not restricting the functioning of the sleeper program 70 and the sleeper program 70 implements the power saving features.

In the preferred embodiment, the system clock frequency is decreased to its minimum value. Barrett et al., in U.S. Patent Serial Number 611,990, herein incorporated by reference, discloses the method for reducing the clock frequency. In addition, the preferred embodiment requires there be no floppy disk accesses in progress. Reducing the system clock speed while a floppy disk access is in progress results in a Direct Memory Access (DMA) failure. To prevent a DMA failure, the sleeper program 70 checks 72 a flag maintained by the ROM Basic Input/Output System (BIOS) to see if a floppy disk access is in progress and, if not, the sleeper program 70 reduces 73 the clock speed. Otherwise, the sleeper program 70 does not reduce the clock speed. In either case, however, the next step is to enable interrupts 74 so the suspended processor detects external interrupts. The sleeper program 70 next issues a halt instruction 75, which suspends processor activity. The processor stays suspended until the processor detects an external interrupt.

Halting the processor results in two power saving benefits. First, the halt reduces the amount of transistor switching within the processor itself. Therefore, the processor requires less power to operate. Second, system bus activity stops. In the preferred embodiment, the portable computer uses pseudo-static memory. Pseudo-static memory chips enter a low-power, self-refresh mode while not accessed, and, once accessed, return to full-power mode. Hereth et al., in U.S. Patent Number 4,710,903, herein incorporated by reference, discuss the operation of pseudo-static memories in detail. Because there is no bus activity, there are no memory accesses occurring. Because there are no memory accesses occurring, the pseudo-static memory automatically goes into the low-power, self-refresh mode. Therefore, memory decreases its power demands as a direct result of the halt instruction 75. These power saving features continue until an external interrupt occurs.

The external interrupt results from a user keystroke, cursor movement

provided by a mouse, electromagnetic stylus or light pen device, or another hardware event. The most common hardware event to occur and, thus, wake the processor is the periodic system timer, which triggers an external interrupt every 55 milliseconds. Thus, the computer does not remain in the suspended state indefinitely.

A hardware mechanism services 76 the external interrupt. The resulting memory accesses cause the pseudo-static memory to power up. In the preferred embodiment, a hardware mechanism automatically increases the system clock frequency during processing of the external interrupt. After servicing the interrupt, control returns to the sleeper program 70 at the instruction after the halt instruction. The sleeper program 70 corrects registers 77 reflecting the system speed. The sleeper program 70 then exits 78 by transferring control to the original idle loop interrupt handler 50 based on the original vector address.

Fig. 4 shows the flowchart of the limiter program 60. Upon entry, the AH register contains the system service function code identifying the particular system service function 37 called by the application 10. Halting the processor during the idle loop 40 degrades several identified system service functions. To reduce visible degradation, the limiter program 60 compares 61 the code in the AH register with the codes corresponding to the identified functions. The identified functions are the Character Output (code 2), Auxiliary Output (code 4), Printer Output (code 5), Direct Console Input/Output (code 6), Output Character String (code 9), and Get Input Status (code 11) functions. If the code in the AH register does not correspond to any identified function, then the limiter program exits by transferring control 62 to the system service interrupt handler 35 based on the original vector address. As a result of the exit, the system service 35 returns directly to the application 10 upon completing the requested function.

On the other hand, if the AH register contains an identified function code then the limiter program 60 signals the sleeper program 70 by setting 63 a flag located in shared memory 80. Next, the limiter program 60 calls 64 the original system service 35 using the original interrupt vector. As a

result of the call 64, the system service 35 returns control to the limiter program 60 upon completion of the requested function. After control returns to the limiter program 60, it pushes 65 the results returned from the system service call onto the system stack. The limiter program 60 clears 66 the flag in shared memory 80 and exits by returning 67 to the application program 10.

Although this invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

For example, the applicants implemented the limiter program and sleeper program as device driver interrupt handlers. The applicants recognize the interrupt handlers could also be implemented as MS-DOS Terminate-and-Stay-Resident programs. In addition, the operating system code could be modified to include the limiter and sleeper program instructions in-line. Some functions of the limiter and sleeper programs also could be implemented through hardware devices, instead of entirely through software instructions.

Although the preferred embodiment relies on a particular hardware arrangement, the invention could be used on any MS-DOS computer with only minor modifications. For example, another computer's DMA implementation might not be affected by a reduction in clock speed. In that case, there would be no need for the sleeper program to check for floppy disk accesses. Another computer may have additional devices with a software selectable power-saving mode. Software instructions to place these devices into low-power mode could be added to the sleeper program. Furthermore, the invention is not limited to computers using pseudo-static memory. Any computer memory with a low-power mode will benefit from the lack of memory accesses. In any case, any portable computers will benefit from halting the processor while it is idle.

What is claimed is:

1. A data processing system including a processor and having software instructions stored in memory for processing by the processor, the software instructions including an idle sequence, the processor and software instructions further comprising:

an idle state detector for detecting an idle sequence; and

a sleeper for providing a software instruction to the processor to halt processing of software instructions when the idle state detector detects an idle sequence.

2. A data processing system including a processor and having software instructions stored in memory for processing by the processor, the software instructions including an idle sequence, the processor and software instructions further comprising:

an interrupt generated by the idle sequence;

a limiter for identifying a code sequence which generates said interrupt and which would be substantially degraded by a halt instruction to the processor; and

an interrupt handler responding to said interrupt generated by the idle sequence, the interrupt handler checking whether a sequence that would be substantially degraded has been identified and, if not, providing an instruction to the processor to halt processing of software instructions.

3. The data processing system of Claim 2 wherein the interrupt handler further comprises software instructions to decrease the clock frequency.

4. The data processing system of Claim 2 wherein the memory has a power-saving mode entered when no memory accesses are

occurring.

5. The data processing system of Claim 4 wherein the memory is pseudo-static memory.

6. The data processing system of Claim 2 wherein the limiter is an interrupt handler.

7. The data processing system of Claim 2 wherein the interrupt handler is implemented as a device driver.

8. The data processing system of Claim 2 wherein the processor's instruction pointer continues normal operation upon receipt of any subsequent interrupt.

9. The data processing system of Claim 8 wherein the interrupt handler returns control to the idle sequence.

10. The data processing system of Claim 2 wherein the idle sequence is an idle loop and the interrupt is unique to the idle loop.

11. A data processing system including a processor and having software instructions stored in memory for processing by the processor, the software instructions including an idle loop, the processor and software instructions further comprising:

an interrupt generated by and substantially unique to an idle loop;

an interrupt handler for identifying a code sequence which generates said idle loop interrupt and which would be substantially degraded by a halt instruction to the processor;

an interrupt handler responsive to said idle loop interrupt, the interrupt handler checking whether a sequence that would be

substantially degraded has been identified and, if not, provides software instructions to the processor to decrease system clock frequency and halt processing of software instructions;

an instruction pointer which continues operation when any subsequent interrupt is received, causing the interrupt handler to return to the idle loop.

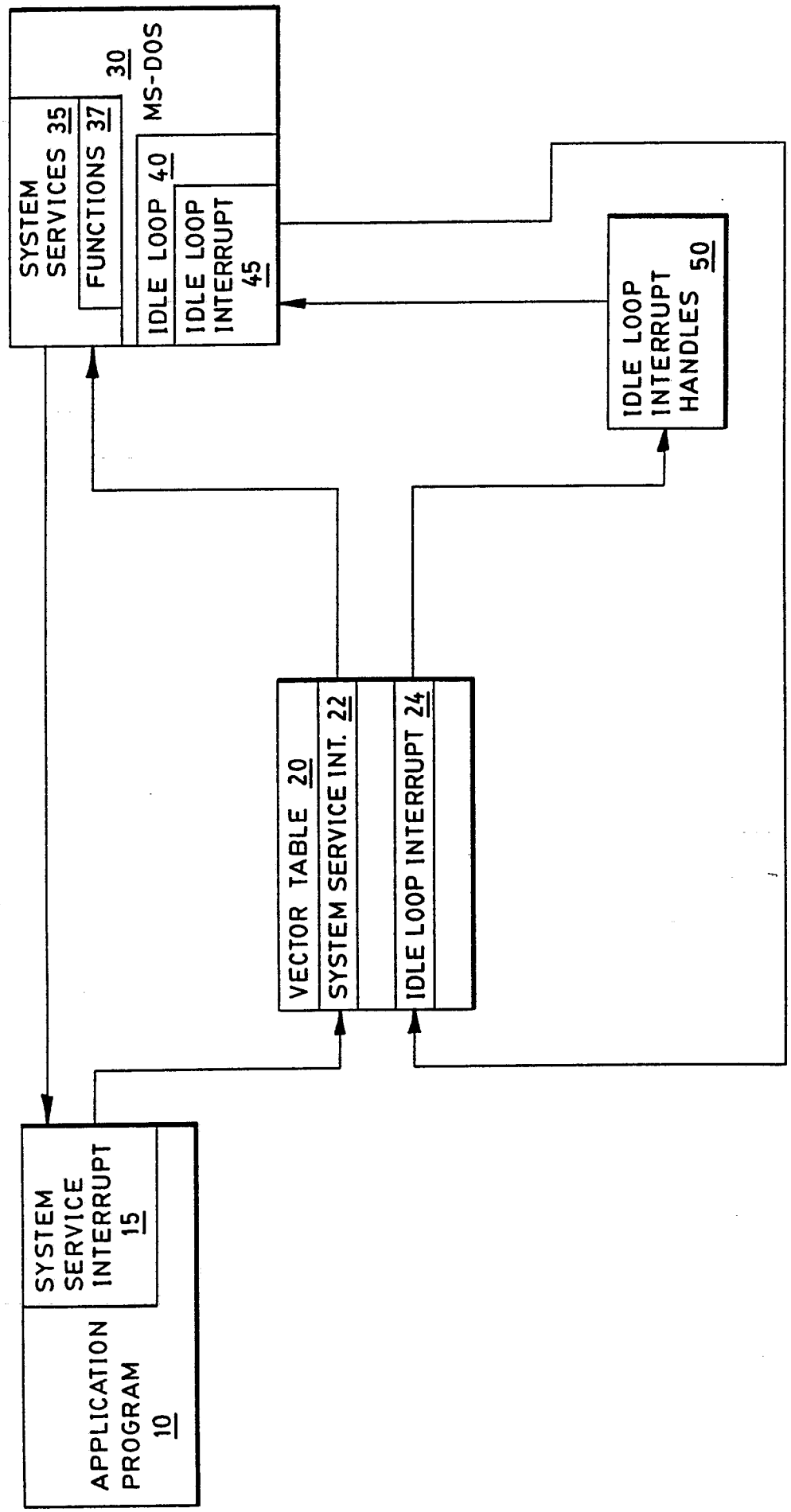

APPLICATION PROGRAM 10
SYSTEM SERVICE INTERRUPT 15
VECTOR TABLE 20
SYSTEM SERVICE INT. 22
IDLE LOOP INTERRUPT 24
IDLE LOOP INTERRUPT HANDLES 50
MS-DOS 30
SYSTEM SERVICES 35
FUNCTIONS 37
IDLE LOOP 40
IDLE LOOP INTERRUPT 45

FIG. 1

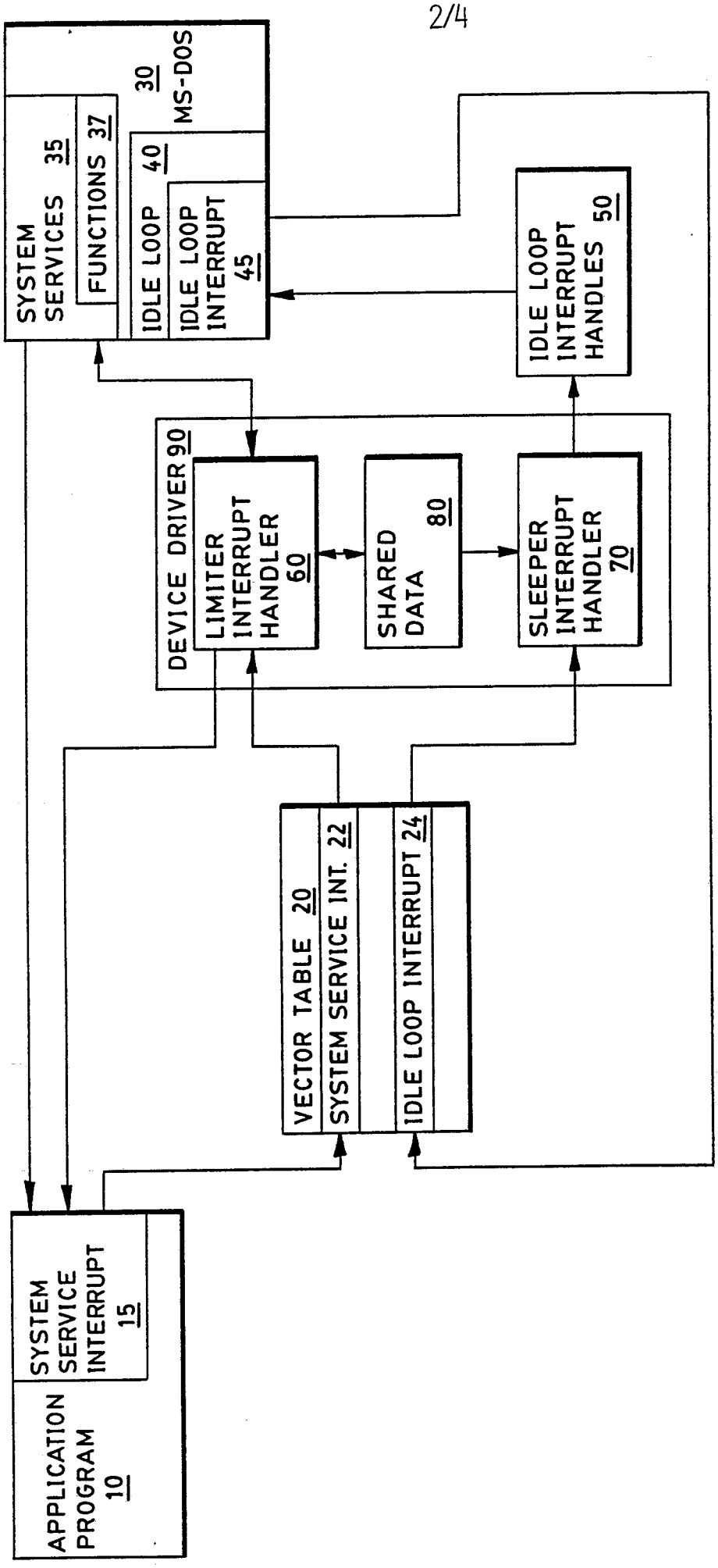

APPLICATION PROGRAM 10
SYSTEM SERVICE INTERRUPT 15
VECTOR TABLE 20
SYSTEM SERVICE INT. 22
IDLE LOOP INTERRUPT 24
DEVICE DRIVER 90
LIMITER INTERRUPT HANDLER 60
SHARED DATA 80
SLEEPER INTERRUPT HANDLER 70
IDLE LOOP INTERRUPT HANDLES 50
MS-DOS 30
SYSTEM SERVICES 35
FUNCTIONS 37
IDLE LOOP 40
IDLE LOOP INTERRUPT 45

FIG. 2

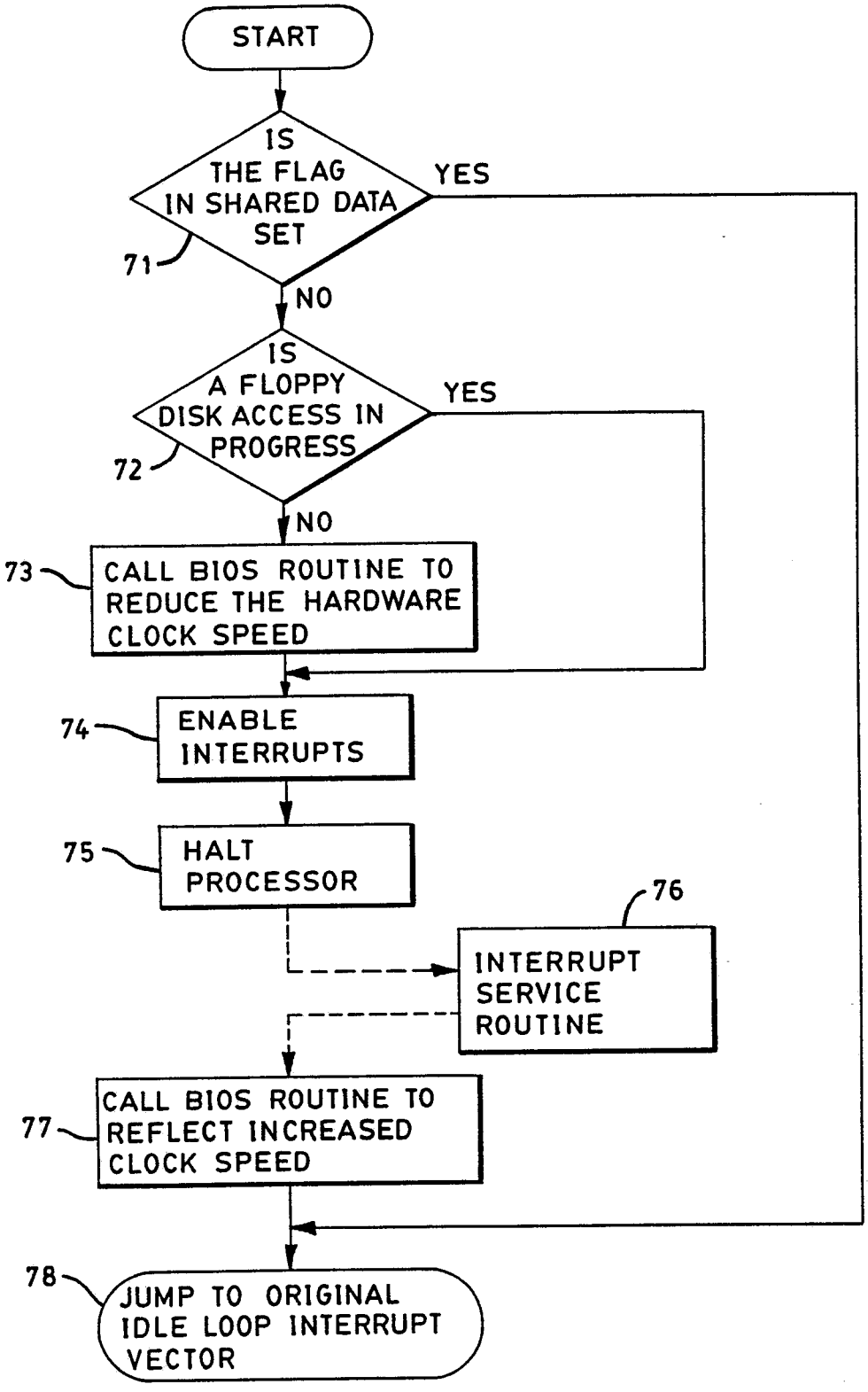

START
IS THE FLAG IN SHARED DATA SET
YES
71
NO
IS A FLOPPY DISK ACCESS IN PROGRESS
YES
72
NO
73
CALL BIOS ROUTINE TO REDUCE THE HARDWARE CLOCK SPEED
74
ENABLE INTERRUPTS
75
HALT PROCESSOR
76
INTERRUPT SERVICE ROUTINE
77
CALL BIOS ROUTINE TO REFLECT INCREASED CLOCK SPEED
78
JUMP TO ORIGINAL IDLE LOOP INTERRUPT VECTOR

FIG. 3

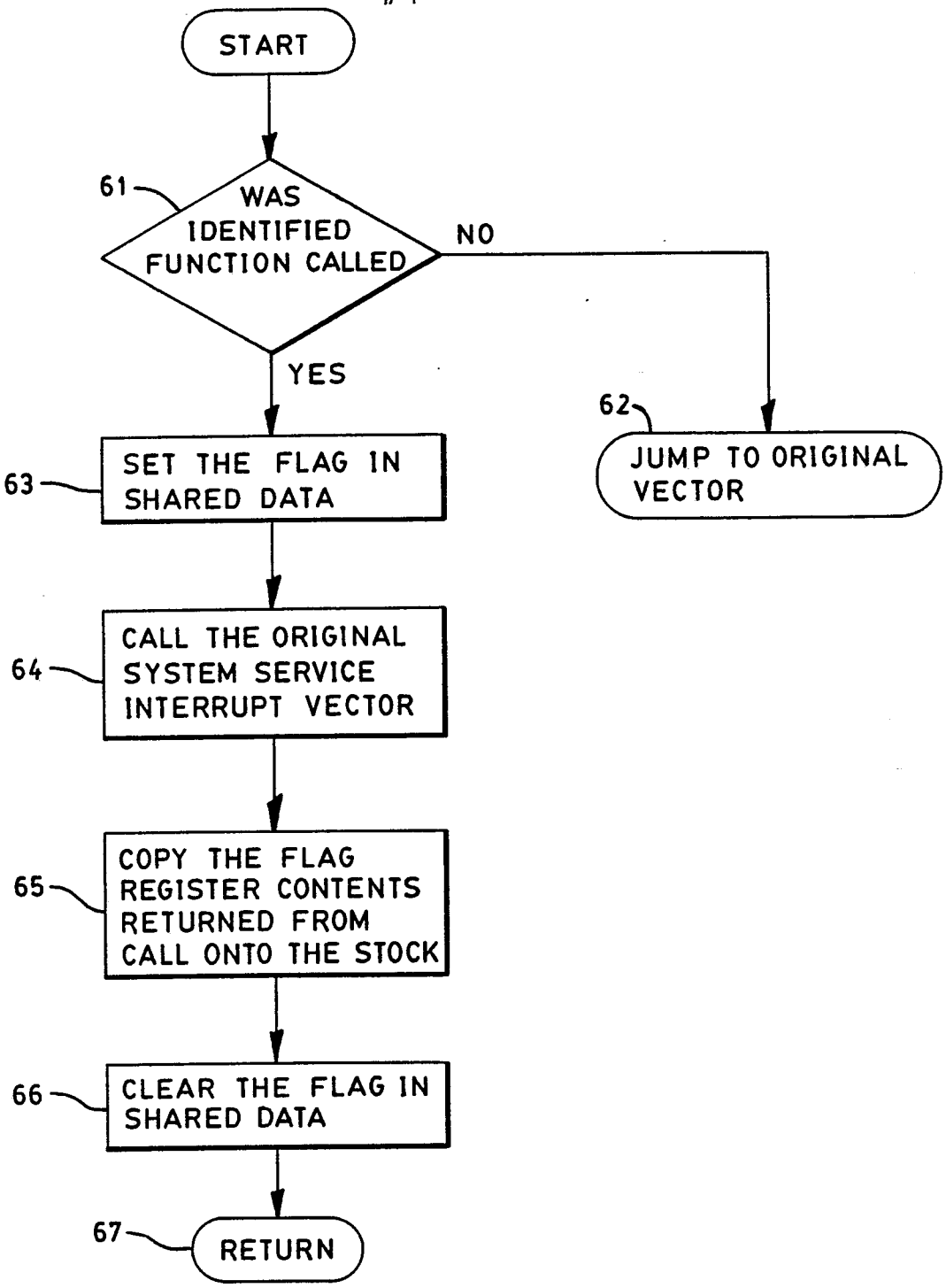

START
61
WAS IDENTIFIED FUNCTION CALLED
NO
YES
62
JUMP TO ORIGINAL VECTOR
63
SET THE FLAG IN SHARED DATA
64
CALL THE ORIGINAL SYSTEM SERVICE INTERRUPT VECTOR
65
COPY THE FLAG REGISTER CONTENTS RETURNED FROM CALL ONTO THE STOCK
66
CLEAR THE FLAG IN SHARED DATA
67
RETURN

FIG. 4

INTERNATIONAL SEARCH REPORT

PCT/US 92/04405

International Application No

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

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

Int.Cl. 5 G06F1/32

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category °	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
X	EP,A,0 426 410 (TEXAS INSTRUMENTS INC) 8 May 1991 see column 8, line 10 - line 15 see column 9, line 14 - line 28; figure 2B see column 9, line 47 - column 10, line 6 see column 11, line 15 - line 33	1
Y		2,3,8, 10,11
Y	WO,A,9 100 566 (POQET COMPUTER CORP) 10 January 1991 see page 37, line 7 - page 38, line 1 see page 38, line 24 - line 33	2,3,8, 10,11

° Special categories of cited documents :[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

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

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

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

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
23 SEPTEMBER 1992	01. 10. 92
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	LA GIOIA C.

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. US 9204405 SA 61784

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on 23/09/92
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0426410	08-05-91	CN-A-	1054496	11-09-91
		JP-A-	3210617	13-09-91
WO-A-9100566	10-01-91	AU-A-	6031390	17-01-91
		EP-A-	0479887	15-04-92