GROUP EXHIBIT U

US006374276B2

(12) **United States Patent**
Vong et al.

(10) **Patent No.: US 6,374,276 B2**
(45) **Date of Patent: Apr. 16, 2002**

(54) **HANDHELD COMPUTING DEVICE WITH EXTERNAL NOTIFICATION SYSTEM**

(75) Inventors: **William Vong**, Seattle; **Chad Schwitters**, Redmond, both of WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/767,529**

(22) Filed: **Jan. 22, 2001**

Related U.S. Application Data

(62) Division of application No. 08/854,102, filed on May 8, 1997, now Pat. No. 6,209,011.

(51) **Int. Cl.**[7] **G04B 19/00**

(52) **U.S. Cl.** **708/112**; 708/135; 710/260; 713/321

(58) **Field of Search** 708/112, 135; 710/48.51, 260–266; 713/300–324; 324/309–571

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,999,050	A	12/1976	Pitroda	708/112
4,162,610	A	7/1979	Levine	368/41
4,258,354	A	3/1981	Carmon et al.	340/309.4
4,768,176	A	8/1988	Kehr et al.	368/10
4,774,697	A	9/1988	Aihara	368/41
4,780,839	A	10/1988	Hirayama	708/160
4,891,776	A	1/1990	Kuroki et al.	708/111
5,050,138	A	9/1991	Yamada et al.	368/10
5,157,640	A	10/1992	Backner	368/10
5,237,684	A	8/1993	Record et al.	709/302
5,262,763	A	11/1993	Okuyama et al.	345/87
5,416,725	A	5/1995	Pacheco et al.	702/176
5,664,228	A *	9/1997	Mital	710/62
5,754,629	A	5/1998	Kunimori et al.	379/88.13
5,760,690	A	6/1998	French	340/571
5,832,489	A	11/1998	Kucala	707/10
5,900,875	A	5/1999	Haitani et al.	345/349
5,973,612	A *	10/1999	Deo et al.	340/825.44
6,009,338	A	12/1999	Iwata et al.	455/575
6,047,260	A	4/2000	Levinson	705/9
6,128,012	A *	10/2000	Seidensticker, Jr. et al.	345/336

FOREIGN PATENT DOCUMENTS

EP	0698852 A2	2/1996

OTHER PUBLICATIONS

Franklin Day Planner Electronic Organizer, 2.2–2.5, 4.2–4.9, Copyright 1996 (most likely Feb.).

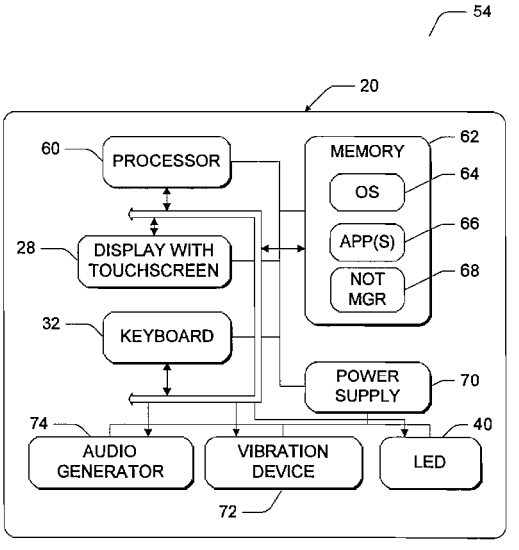
* cited by examiner

Primary Examiner—Christopher B. Shin
(74) *Attorney, Agent, or Firm*—Lee & Hayes, PLLC

(57) **ABSTRACT**

A portable handheld computing device has a notification system that alerts a user of an event regardless of whether the device is on or off. The notification system has a notification mechanism that is activated upon occurrence of the event and remains active until the user acknowledges the activated mechanism. In one implementation, the notification mechanism is a light emitting diode (LED) mounted externally on the handheld computing device. The LED is visible to the user when the lid is closed onto the base (i.e., the device is off) or when the lid is open (i.e., the device is on). The notification mechanism also has a deactivation button mounted externally of the handheld computing device. The user depresses the deactivation button to deactivate the LED. The LED and deactivation button may be integrated as a single component.

10 Claims, 6 Drawing Sheets



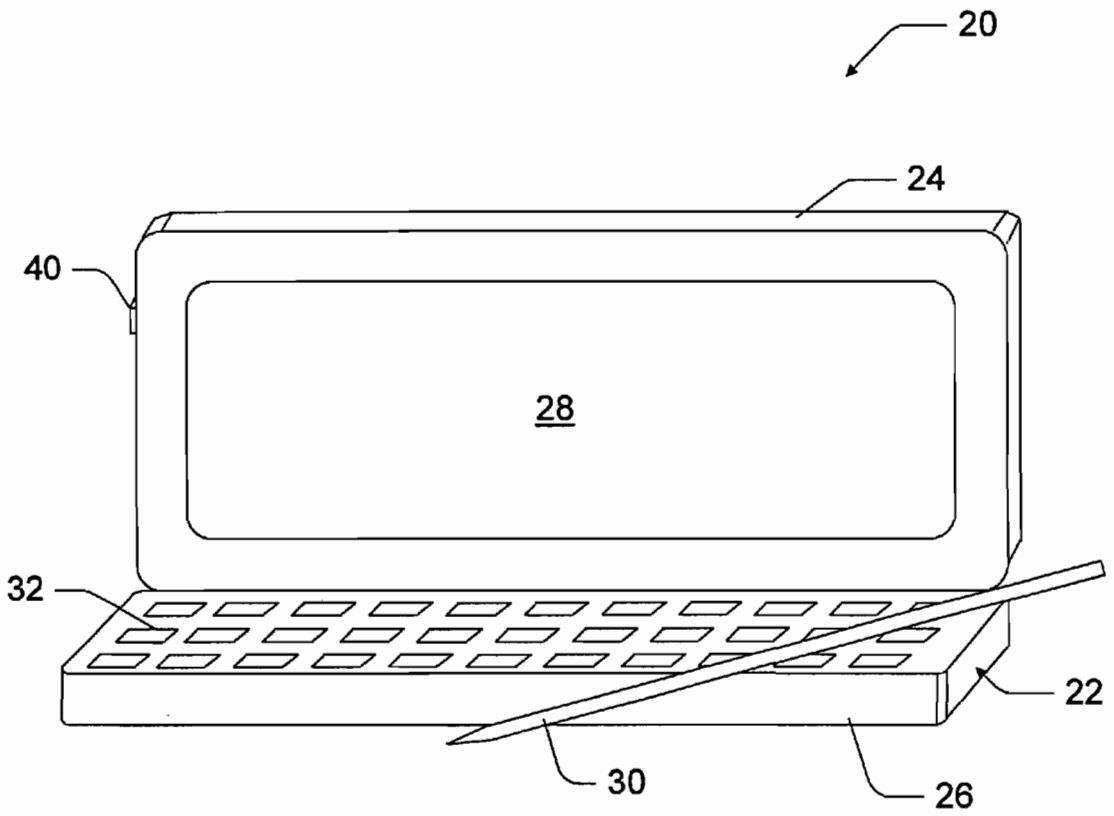


Fig. 1

Fig. 2A

Fig. 2B

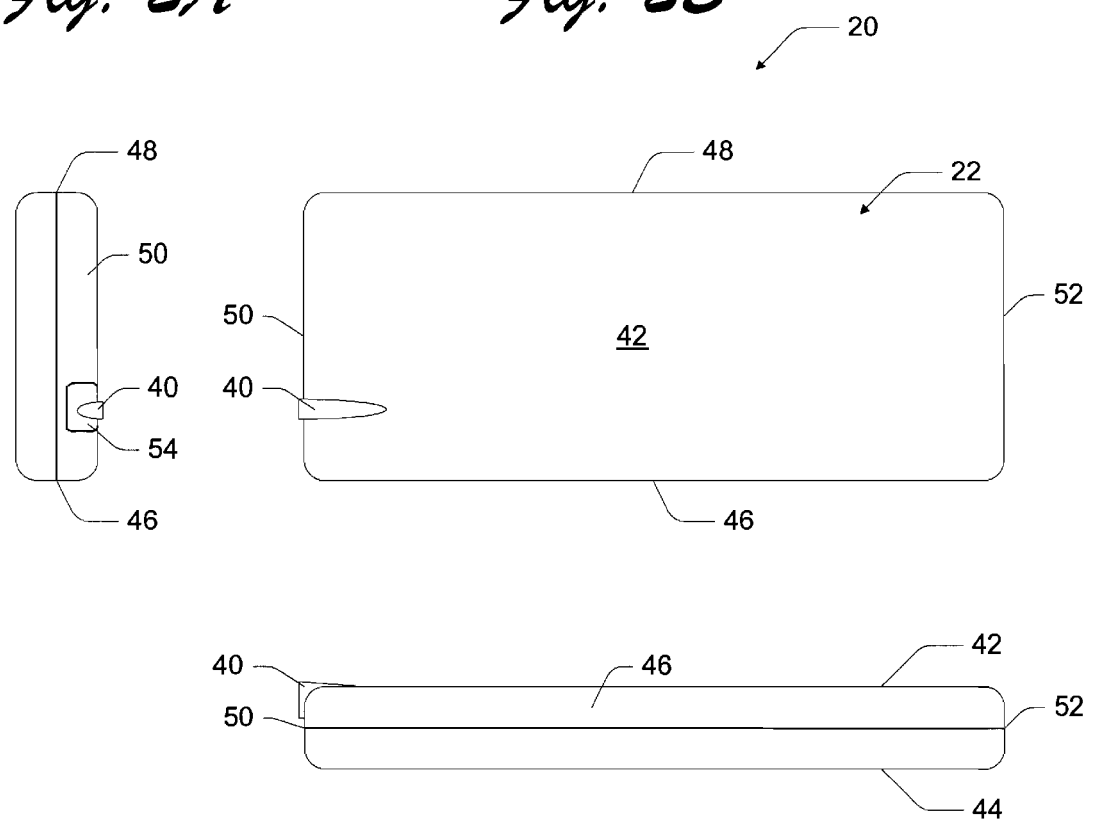


Fig. 2C

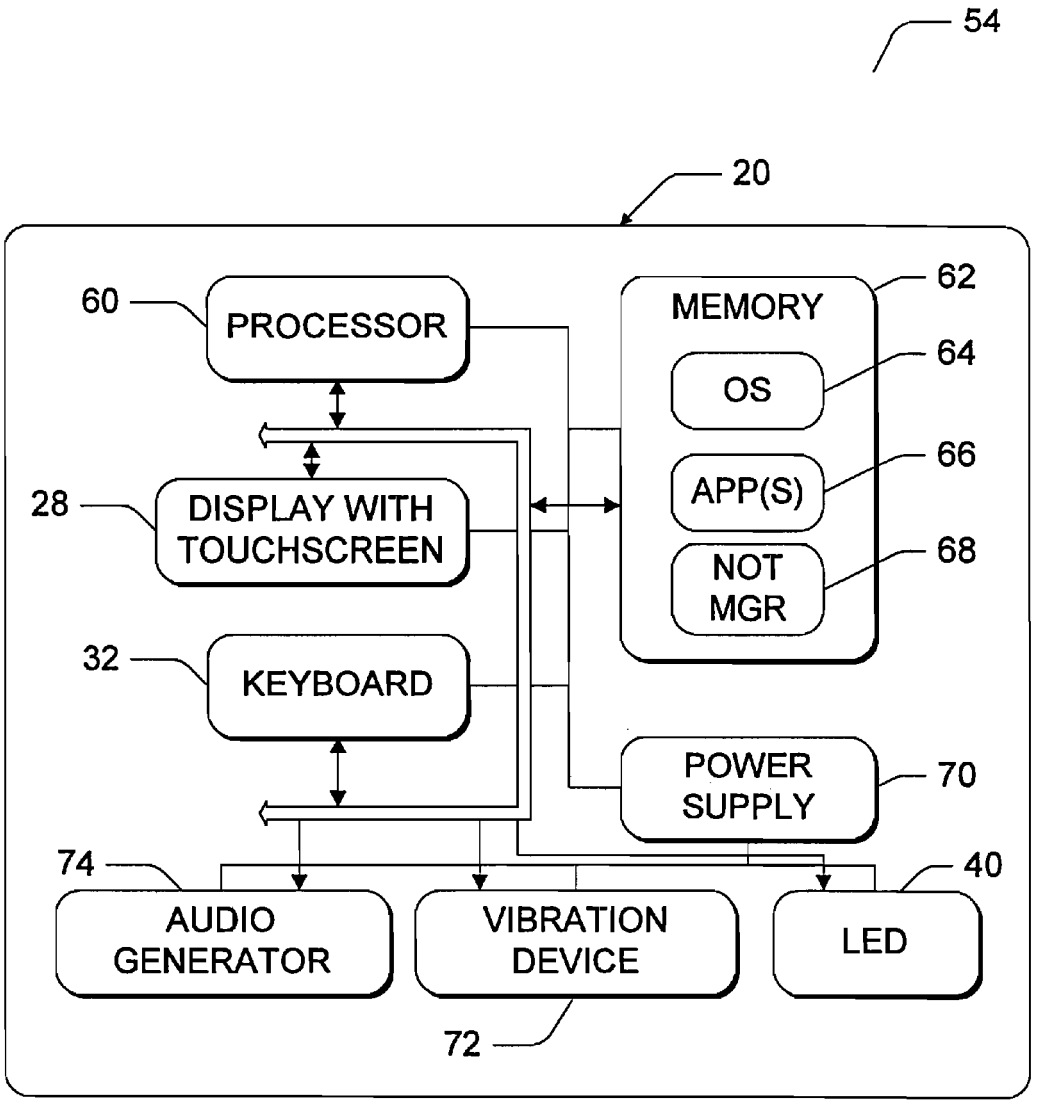

54
20
60
PROCESSOR
MEMORY
62
OS
64
APP(S)
66
NOT MGR
68
28
DISPLAY WITH TOUCHSCREEN
32
KEYBOARD
POWER SUPPLY
70
74
AUDIO GENERATOR
VIBRATION DEVICE
LED
40
72

Fig. 3

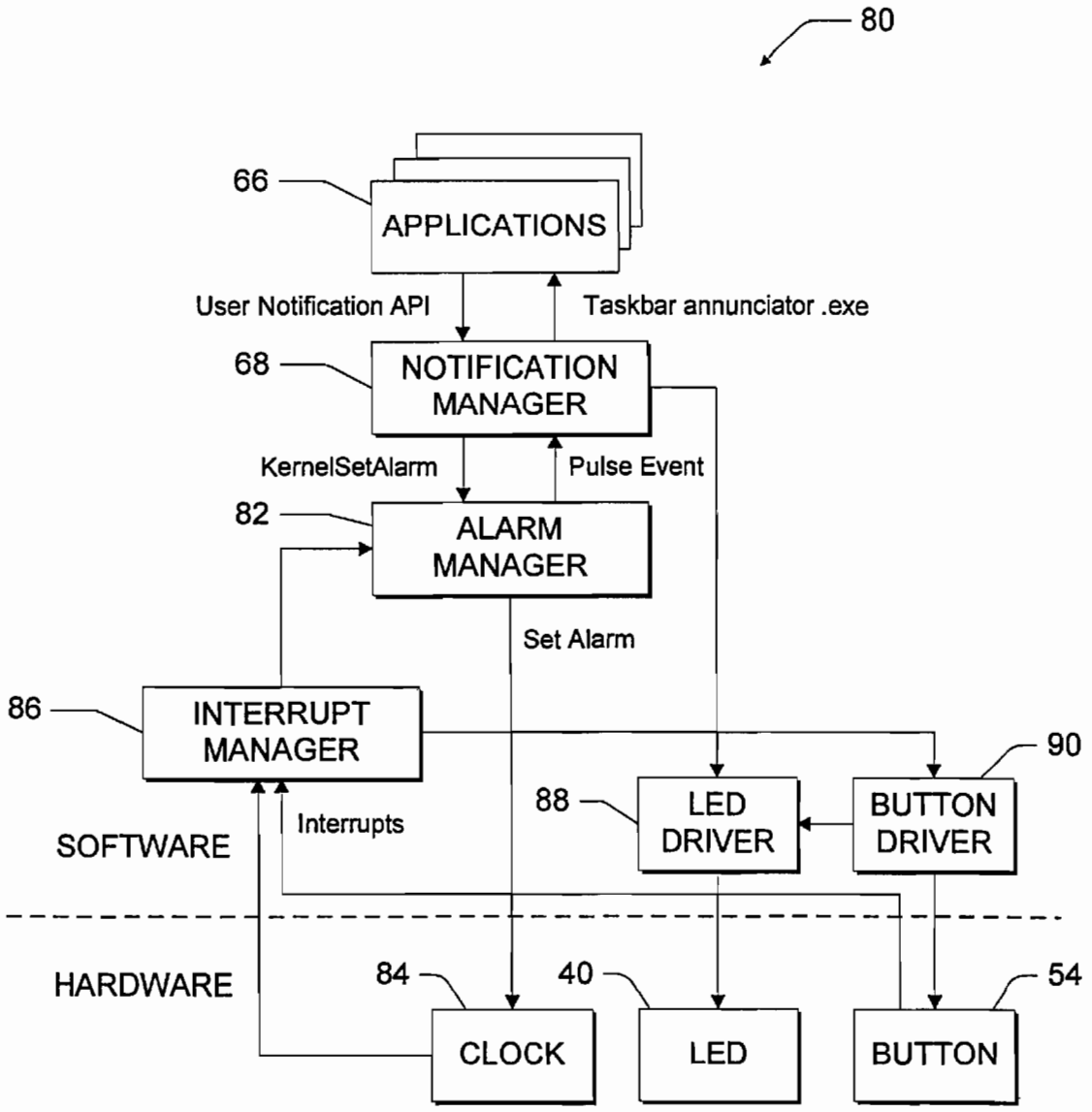

80
66
APPLICATIONS
User Notification API
Taskbar annunciator .exe
68
NOTIFICATION MANAGER
KernelSetAlarm
Pulse Event
82
ALARM MANAGER
Set Alarm
86
INTERRUPT MANAGER
90
88
LED DRIVER
BUTTON DRIVER
Interrupts
SOFTWARE
HARDWARE
84
40
54
CLOCK
LED
BUTTON

Fig. 4

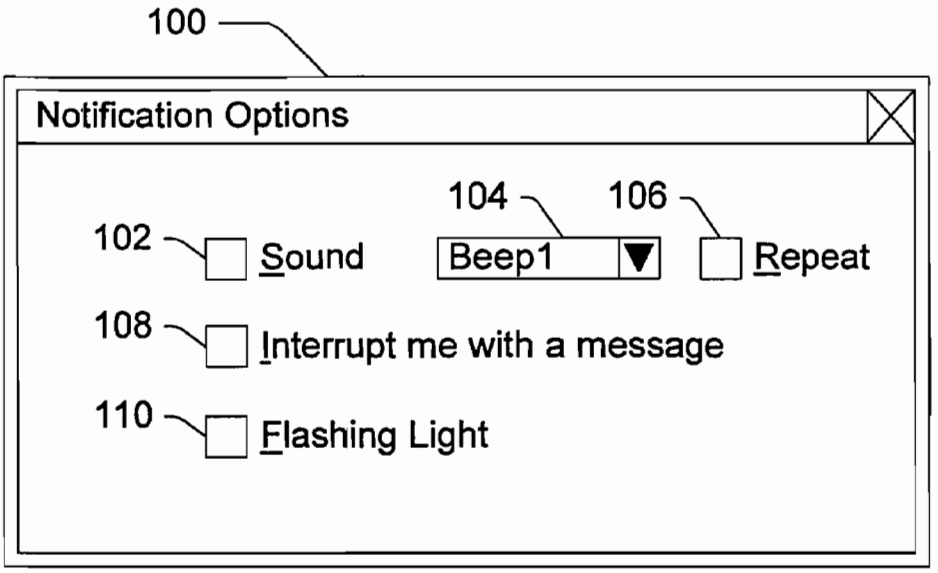

100
Notification Options
104
106
102
Sound
Beep1
Repeat
108
Interrupt me with a message
110
Flashing Light

Fig. 5

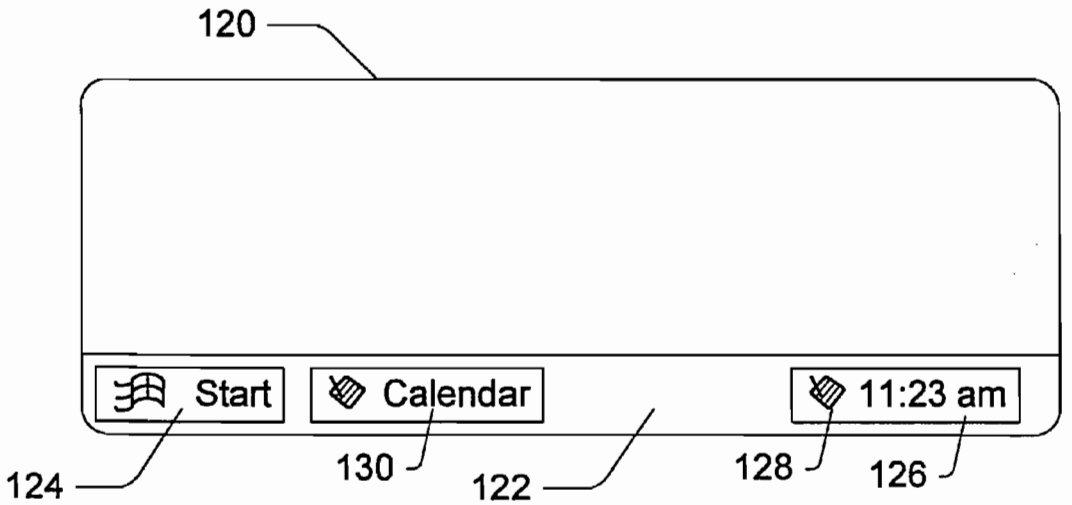

120
Start
Calendar
11:23 am
124
130
122
128
126

Fig. 6

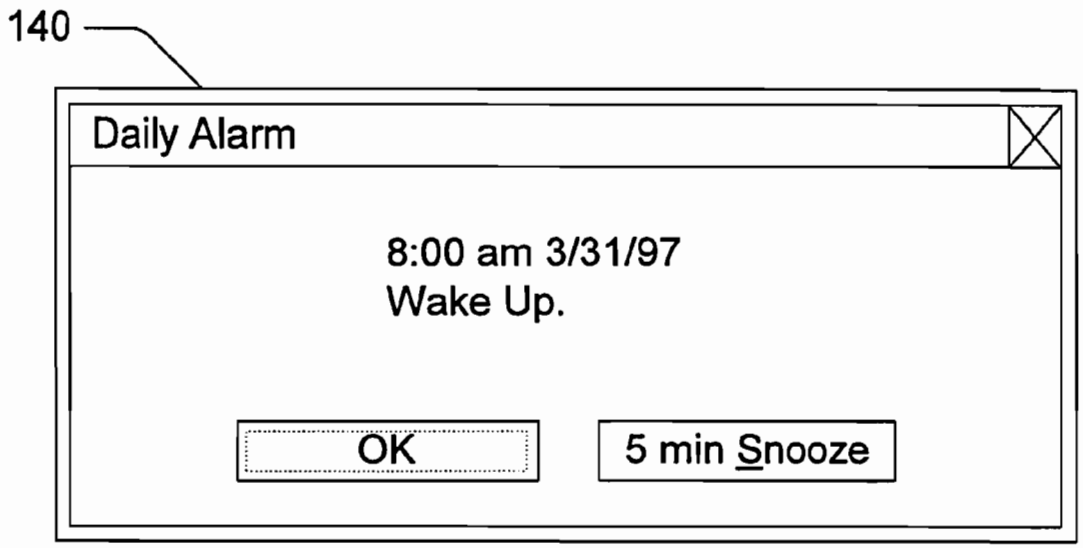


Fig. 7

HANDHELD COMPUTING DEVICE WITH EXTERNAL NOTIFICATION SYSTEM

RELATED APPLICATIONS

This is a divisional of U.S. patent application Ser. No. 08/854,102, filed May 8, 1997, which is now U.S. Pat. No. 6,209,011.

TECHNICAL FIELD

This invention relates to portable handheld computing devices, such as handheld personal computers (H/PCs). More particularly, this invention relates to an external notification system for handheld computing devices.

BACKGROUND OF THE INVENTION

Small, handheld computing devices have been steadily growing in popularity in recent years. The devices go by different names, including palmtops, pocket computers, personal digital assistants, personal organizers, and the like. This disclosure is primarily directed to a class of computing devices referred to as handheld personal computers, or "H/PCs", although aspects of this invention can be implemented other types of handheld computing devices.

H/PCs are small, pocket-sized devices having an LCD (liquid crystal display) with a touch-sensitive screen, a stylus to enter data through the screen, and an input device such as a keypad or miniature QWERTY keyboard. H/PCs have a microprocessor, memory, and are capable of running an operating system and one or more applications on the operating system. Microsoft Corporation recently released the Windows® CE operating system for use on H/PCs, which is a scaled-down version of its popular Windows® operating systems manufactured for personal computers.

One of the most desirable characteristics of H/PCs is their portability. The compact, portable H/PCs provide a user with real computer-like applications—such as email, PIM (personal information management), Internet browser, spreadsheet, word processing. A traveling user can receive email messages, schedule meetings or appointments, and browse the Internet from the H/PC.

Some handheld computing devices can notify a user of a scheduled event, if they are turned on. The device plays an alarm sound, or pops-up a dialog box, to alert the user of the event. However, many handheld computing devices have no means of notifying a user when they are turned off, which is normally the case to conserve power. While some handheld computing devices might be configured to wake up and sound an alarm, such devices typically time out the alarm after a short period. As a result, the user can miss the alarm because it terminates before being noticed. In addition, audio alarms may, on occasions, be too faint for the surrounding environment (e.g., an alarm might be overpowered by noise in an airplane flight) or not sufficiently strong to command a user's attention when the user is not immediately next to the device.

It would be advantageous to develop a notification system for handheld computing devices, such as H/PCs, that notifies a user when an event occurs regardless of whether the device is on or off, open or closed, pocketed, or docked, and which remains active until the user acknowledges it. It would also be advantageous to develop a notification system that provides a lasting external notification to the user, rather than a short-run alarm or a pop-up box that is not externally visible.

SUMMARY OF THE INVENTION

This invention concerns a portable handheld computing device having a notification system that alerts a user of an event regardless of whether the device is on or off, open or closed, pocketed, or docked. The notification system has a notification mechanism that is activated upon occurrence of the event and remains active until the user acknowledges the activated mechanism.

According to an aspect of this invention, the notification mechanism is a light emitting diode (LED) that is (by user option) turned on by the notification system when an event occurs. The LED remains activated until the user takes action to handle the event.

According to another aspect of this invention, the LED is mounted externally on the handheld computing device. More particularly, the handheld device has a casing with a lid and a base. The LED is mounted on the lid's upper surface and wraps around to one of the end surfaces of the lid. In this manner, the LED is visible to the user when the lid is closed onto the base (i.e., the device is off) or when the lid is open (i.e., the device is on).

According to another aspect of this invention, the notification mechanism also has a deactivation button mounted externally of the handheld computing device. The user depresses the deactivation button to deactivate the LED (as well as any other external signals that may be used). In one implementation, the LED and deactivation button are integrated as a single component mounted on the device lid.

According to yet another aspect of this invention, a notification program runs on the handheld computing device and is callable by an application to help schedule events. The notification program sets timers with the system clock, which is always on even when the handheld computer is turned off. When a timer expires, the system clock sends an interrupt to the notification program to wake up the notification program so that it can turn on the LED. The LED is coupled to power so that it can remain on and the notification program can go back asleep. The LED continues emitting light until the user notices and presses the deactivation button.

According to another aspect, the notification program places a taskbar annunciator in the taskbar of an operating graphical user interface window when an event is realized. After depressing the deactivation button in recognition of the LED, the user can actuate the taskbar annunciator with a stylus or other means and jump directly to the source of the event. For instance, actuating the taskbar annunciator might open a window that describes an appointment, which is the root of the event.

According to another aspect, the notification program supports a graphical user interface that enables a user to set notification options specifying how external notification is to operate. For instance, the user might prefer a flashing light in combination with an alarm. The user can set these options through the user interface. The options are saved in a structure that is accessed when a user notification is set.

According to still another aspect, the notification program is called by the applications on the handheld computing device through an application program interface (API). The API defines a time parameter that specifies when the user notification should occur and a type parameter that references the structure containing the user-defined notification options.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference numbers are used throughout the drawings to reference like components and features.

FIG. **1** is a perspective view of a handheld computing device in an open position.

FIG. **2**A provides an end elevation view of the handheld computing device in a closed position.

FIG. **2**B provides a front elevation view of the handheld computing device in a closed position.

FIG. **2**C provides a side elevation view of the handheld computing device in a closed position.

FIG. **3** is a block diagram of the handheld computing device.

FIG. **4** is a block diagram of the hardware/software architecture of a notification system implemented in the handheld computing device.

FIG. **5** is a diagrammatic illustration of a graphical user interface window embodied as an "options" dialog box.

FIG. **6** is a diagrammatic illustration of a screen image presented on a display of the handheld computing device.

FIG. **7** is a diagrammatic illustration of a graphical user interface window embodied as a "notification" dialog box.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIGS. **1** and **2** show a handheld computing device **20**. As used herein, "handheld computing device" means a small general computing device having a processing unit that is capable of running one or more application programs, a display, and an input mechanism that is typically something other than a full-size keyboard. The input mechanism might be a keypad, a touch-sensitive screen, a track ball, a touch-sensitive pad, a miniaturized QWERTY keyboard, or the like.

The handheld computing device **20** of FIGS. **1** and **2**A–**2**C is embodied as a handheld personal computer (H/PC). The terms "handheld computing device" and "handheld personal computer" (or H/PC) are used interchangeably throughout this disclosure. However, in other implementations, the handheld computing device may be implemented as a personal digital assistant (PDA), a personal organizer, a palm-top computer, a computerized notepad, or the like.

Handheld computing device **20** has a casing **22** with a cover or lid **24** and a base **26**. The lid **24** is hingedly connected to the base **26** to pivot between an open position (FIG. **1**) and a closed position (FIGS. **2**A–**2**C). The handheld computing device **20** has an LCD (liquid crystal display) **28** with a touch-sensitive screen mounted in lid **24**. The device is equipped with a stylus **30** to enter data through the touchscreen display **28** and a miniature QWERTY keyboard **32**, which are both mounted in base **26**. The handheld computing device **20** can also be implemented with a wireless transceiver (not shown) such as an IR (infrared) transceiver and/or an RF (radio frequency) transceiver. Although the illustrated implementation shows a two-member H/PC **20** with a lid **24** and a base **26**, other implementations of the H/PC might comprise an integrated body without hinged components, as is the case with computerized notepads (e.g., Newton® from Apple Computers).

In the above respects, the H/PC **20** is of conventional design and will not be described in detail. Many manufacturers make suitable H/PCs. However, unlike conventional H/PCs, the handheld computing device **20** of this invention is further implemented with an external notification system.

In general, the external notification system is designed to alert a user of an event regardless of whether the handheld computing device is presently on or off, or whether the device is presently running a program. The notification system has a notification mechanism that is activated upon occurrence of the event to alert the user. The notification mechanism remains active until the user acknowledges it, even if the handheld computing device is otherwise turned off. The notification mechanism is an external, sensory perceptible mechanism that attracts the user's attention when the device is on or off and regardless of whether the lid is open or closed. The notification mechanism can be implemented in a variety of ways, including a light, an audio generator, a vibration device, or other forms of sensory perceptible mechanisms. In addition, these mechanisms may be used in combination, or with other forms of sensory perceptible notices, such as a visual dialog box on the display.

In the preferred implementation, the external notification mechanism includes an externally mounted LED (light emitting diode) **40**. When activated as a result of an event, the LED is illuminated or made to flash. The LED **40** remains activated until the user acknowledges it.

More particularly, the LED **40** is mounted on the external surface of the H/PC **20** in a location that the user can view the light from different angles and sides of the H/PC. In addition, the LED **40** is positioned to be seen when the lid **24** is open or closed. As shown in FIGS. **2**A–**2**C, the H/PC casing **22** has an upper surface **42** on lid **24**, a lower surface **44** on base **26**, a front side surface **46** (on both lid **24** and base **26**), an opposing back side surface **48** (on both lid **24** and base **26**), and opposing end surfaces **50** and **52** (on both lid **24** and base **26**). The end surfaces **50** and **52** are dimensionally shorter than the elongated side surfaces **46** and **48**.

The LED **40** is positioned on the upper surface **42** and wraps around an upper corner to extend onto the end surface **50**. The LED is raised on the end surface **50** to be visible from the front. The LED may or may not be raised on the upper surface **42**. In this manner, the LED **40** can be viewed when the case **22** is closed, either from above by viewing the LED portion on the upper surface **42** (for instance when the H/PC is sitting on a desk), or from the side by viewing the LED portion on the end surface **50** (for instance when the H/PC is slid upright into a shirt pocket, purse, or briefcase). Additionally, the LED **40** can be viewed when the case **22** is open (FIG. **1**) by viewing the raised LED portion on the end surface **50**. As an alternative to raising the LED on the

end surface , the LED **40** may be configured to wrap around to the inner surface of the lid **24** to be viewable when the case **22** is open. The LED itself might be configured in the illustrated shape, or alternatively a normally shaped LED is configured to emit light into light-transmissive tubing that conforms to the illustrated shape.

As shown in the end view of FIG. **2A**, the H/PC **20** also has a deactivation mechanism to deactivate the LED **40** and any other external notification mechanism. In the illustrated implementation, the deactivation mechanism is a deactivation button **54** that is externally mounted on the end **50** to enable a user to quickly locate and deactivate the external notification mechanisms, regardless of whether the lid is open or closed. In a preferred embodiment, the deactivation button **54** and LED **40** are integrated as a single component. The LED **40** can be constructed to project slightly above the face of the deactivation button **54** to act as a bumper to reduce the likelihood of accidental actuation. In other arrangements, the button may be located separately from the LED.

FIG. **3** shows functional components of the handheld computing device **20**. It has a processor **60**, a memory **62**, a display **28**, and a keyboard **32**. The memory **62** generally includes both volatile memory (e.g., RAM) and non-volatile memory (e.g., ROM, PCMCIA cards, etc.). An operating system **64** is resident in the memory **62** and executes on the processor **60**. The H/PC **20** preferably runs the Windows® CE operating system from Microsoft Corporation. This operating system is a derivative of Windows® brand operating systems, such as Windows® 95, that is especially designed for handheld computing devices. However, the handheld computing device may be implemented with other operating systems.

One or more application programs **66** are loaded into memory **62** and run on the operating system **64**. Examples of applications include email programs, scheduling programs, PIM (personal information management) programs, word processing programs, spreadsheet programs, Internet browser programs, and so forth. The H/PC **20** also has a notification manager **68** loaded in memory **62**, which executes on the processor **60**. The notification manager **68** handles notification requests from the applications **66**, as is described below in more detail with reference to FIG. **4**.

The H/PC **20** has a power supply **70**, which is implemented as one or more batteries. The power supply **70** might further include an external power source that overrides or recharges the built-in batteries, such as an AC adapter or a powered docking cradle.

The H/PC **20** is also shown with three types of external notification mechanisms: an LED **40**, a vibration device **72**, and an audio generator **74**. These devices are directly coupled to the power supply **70** so that when activated, they remain on for a duration dictated by the notification mechanism even though the H/PC processor and other components might shut down to conserve battery power. The LED **40** preferably remains on indefinitely until the user takes action. The current versions of the vibration device **72** and audio generator **74** use too much power for today's H/PC batteries, and so they are configured to turn off when the rest of the system does or at some finite duration after activation.

FIG. **4** shows the software and hardware architecture of a notification system **80** for the H/PC **20**. The notification system **80** has a notification manager **68**, which is callable by the applications **66** through a user notification application program interface (API). The API creates a new user notification or modifies an existing one. It is given as:

```
PegSetUserNotification(hNotification, *AppName,
  *Time, *UserNotification)
```

The API has four parameters, three of which are pointers. The "hNotification" parameter specifies whether the call relates to creating a new user notification or to modifying an existing notification. The parameter is either zero, when a new notification is to be added, or contains an identity of the notification to be modified.

The "AppName" pointer points to a null-terminated string that specifies the name of the application **66** that owns the notification. The system uses the application's primary icon as the taskbar annunciator that is set by the notification system to notify the user and enable immediate-click access to the application responsible for the notification. The use of a taskbar annunciator is described below with reference to FIG. **6**. The "Time" pointer points to the system time structure that specifies a time when the notification should occur. The "UserNotification" pointer points to a Peg_User_Notification structure that describes the events that are to occur when the notification time is reached.

More particularly, the Peg_User_Notification structure is a user configurable structure that holds notifications options preferred by the user. The application passes the user's preferences to the system when scheduling an event by specifying the address of the structure. Each application passes in a structure that applies only to it, so notifications for different applications can be differentiated. Similarly, an application can pass in different structures for different events, so individual notifications can be differentiated.

The Peg_User_Notification structure contains information used to initialize a dialog box user interface that is presented to the user when setting notification options. FIG. **5** shows an example dialog box **100**, which is supported by the notification system **80**. In this example, the dialog box **100** includes an option **102** as to whether to sound an alarm and a drop-down menu **104** that lists various available alarm sounds, such as "Beep". The drop-down menu **104** might contain identities of other ".wav" files containing different alarm sounds the user might prefer. A repeat option **106** is also provided so that the user can elect to have the alarm repeated.

The dialog box **100** also has an option **108** that allows a user to enable or disable a dialog box that can be displayed describing the notification when it goes off. An option **110** allows the user to elect whether to have the LED **40** flash, or not, during notification.

It is noted that the dialog box **100** is provided for example purposes, and other options may be included. For instance, the dialog box **100** might include an option to enable/disable the vibration device **72**, or to combine the external notification mechanisms so the LED **40**, vibration device **72**, and alarm **74** all go off at different times in a continuous cycle.

Once the user fills in the dialog box **100**, the options are stored in the Peg_User_Notification structure. This structure is provided below:

```
UserNotificationType {
    DWORD ActionFlags;
    TCHAR *DialogTitle;
    TCHAR *DialogText;
    TCHAR *Sound
    DWORD MaxSound;
    DWORD Reserved;
}
```

The "ActionFlags" parameter specifies the actions to take when a notification event occurs. This parameter is a combination of bit flags, as set forth in the following table.

Value	Meaning
PUN_LED	Flash LED.
PUN_VIBRATE	Vibrate the Device.
PUN_DIALOG	Display the user notification dialog box. When this structure is passed to the PegSetUserNotification API, the DialogTitle and DialogText pointers point to the title and text.
PUN_SOUND	Play the sound file identified by the Sound pointer.
PUN_REPEAT	Repeat sound file for T seconds.

The "DialogTitle" pointer specifies the title of the user notification dialog box. If this parameter is null, no dialog is displayed. The "DialogText" pointer specifies the text of the user notification dialog box. If this parameter is null, no dialog is displayed. The "Sound" pointer references a buffer that contains the unqualified name of a sound file to play. The "MaxSound" parameter specifies the maximum length of the string that can be copied into the sound buffer. The "Reserved" parameter is reserved for future use and is presently set to zero.

With reference again to FIG. **4**, the notification manager **68** passes a command to an alarm manager **82** to set an alarm for a notification event. The alarm manager **82** generates a set alarm command that is output to the real-time clock **84** to tell the clock to set an alarm at the scheduled time of the notification event. When the clock reaches the event time, it notifies an interrupt manager **86** through an interrupt. The interrupt manager **86** informs the notification manager **68** that the time of the event has arrived. The notification manager **68** then sends out activation commands to an LED driver **88** to turn on LED **40**. A button device driver **90** is also provided to handle interrupts generated when the notification button **54** is depressed to disable the LED **40**.

To explain the architecture in the context of an example situation, suppose the user starts a calendar application **66** and schedules an event notification for 8:00 AM. The user clicks on an "options" button to bring up the dialog box **100** (FIG. **5**) to ensure that the LED and alarm are both enabled. The user then closes the dialog box **100** and saves the clock settings. The calendar application **66** calls the notification manager **68** using the PegSetUserNotification API, which includes a pointer to the structure containing information specifying how the LED and alarm are to behave. The notification manager **68** stores the scheduled notification and examines it in light of any other scheduled user notifications to determine which notification is associated with the next chronological event to occur. Suppose that the calendar notification is next to occur. The notification manager **68** then calls the alarm manager **82**, which in turn sets a hardware alarm for 8:00 AM in real-time clock **84**. The user can then exit the application **66** and turn off the device.

At 8:00 AM, the real-time clock **84** sends an interrupt to interrupt manager **86**. The interrupt manager **86** identifies the interrupt as clock-related, and routes the interrupt to the alarm manager **82**. Upon receiving the interrupt, the alarm manager **82** pulses the event that the notification manager **68** has been waiting on. The notification manager **68** determines that the event is associated with the 8:00 AM calendar alarm and checks the user options to decide how the user wishes to be notified. Assuming that the user wants the light to flash and an alarm to sound, the notification manager **68** calls the LED device driver **88** to start flashing the LED **40** and concurrently plays the selected alarm. The notification manager **68** also creates a taskbar annunciator.

Suppose that the use is not present at 8:00 AM. The handheld computing device **20** times out and turns off. Due to the direct coupling to power, however, the LED **40** remains flashing. The flash rate is selected to minimize power usage, without compromising usability. As one example, the LED flashes once every two seconds for 1/16th of a second. The alarm preferably times out with the computing device **20** to conserve power, but it can be configured to repeat until the user acknowledges the notice.

When the user returns, he/she sees the flashing LED **40** and presses the deactivation button **54**. Depressing button **54** generates an interrupt that is routed by interrupt manager **86** to keyboard device driver **90**. The keyboard driver **90** instructs the LED driver **88** to turn off the LED **40**. It is noted that the clock **84** and deactivation button **54** work the same if the H/PC **20** is already on. They still generate interrupts, only the interrupts do not wake up the device.

The deactivation button **54** is preferably implemented to turn off only the external notification mechanisms (i.e., LED and alarm) because these mechanisms can be annoying and consume a significant quantity of power. The taskbar annunciator and optional dialog box remain intact and are not disabled by the deactivation button **54**. As a result, the user can open the H/PC and glean more information concerning the notice from the taskbar annunciator and dialog box.

When the user opens the H/PC, the notification system **80** optionally causes a taskbar annunciator for the calendar application to be displayed. FIG. **6** shows an example screen image **120** showing the task bar **122** with a "Start" softkey **124** and a time/date area **126**. A taskbar annunciator **128** for the calendar application is displayed in the time/date area **126**. The annunciator **128** is the icon for the calendar application, thereby immediately informing the user that the source of the notification is the calendar application. The user can then click on the taskbar annunciator **128** using the stylus to start the calendar application, as represented by the calendar softkey **130**.

The notification system **80** can optionally display a dialog box that explains the notification to the user; this is faster than starting the originating application, and provides a more noticable notification if the H/PC is being used when the notification event occurs. FIG. **7** shows an example dialog box **140**, which contains a message that informs the user of the 8:00 AM alarm. The user is also presented with the option of accepting the alarm or rescheduling it for an

additional five-minute period. It is noted that tapping the taskbar annunciator **128** or a softkey in the dialog box **140** deactivates the LED **40** and alarm, in the same manner as pressing the deactivation button **54**. Touching any annunciator or acknowledging any dialog box turns off the external signals, since this tells the notification system that the user is aware of the notifications.

The notification system **80** can handle an arbitrary number of notifications from multiple applications. The notification manager **68** handles the scheduled events in temporal order. In some situations, the flashing LED might represent multiple notifications. Since a single flag bit controls the LED, the user presses the deactivation button **54** once to turn off the LED **40**. The user can then open the device and examine the various annunciators to learn which applications are responsible for the notifications.

A full set of user notification APIs is attached to this disclosure as Appendix A.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

APPENDIX A

Notification Reference

The following functions and structures are used with user and application notifications. For further information see the Windows CE Programmer's Reference.

Notification Functions

- PegClearUserNotification
- PegGetUserNotificationPreferences
- PegHandleAppNotifications
- PegRunAppAtEvent
- PegRunAppAtTime
- PegSetUserNotification

Notification Structures

- PEG_USER_NOTIFICATION

API REFERENCE

The PegClearUserNotification function deletes a user notification that was created by a previous call to the function PegSetUserNotification.

```
BOOL PegClearUserNotification(    // notify.h
    HANDLE hNotification    // handle of notification to delete
);
```

Parameters

hNotification

Identifies the user notification to delete.

Return Values

If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE.

See also

Windows CE Notifications, PegSetUserNotification

The PegGetUserNotificationPreferences function queries the user for notification settings by displaying a dialog box showing options that are valid for the current hardware platform.

```
BOOL PegGetUserNotificationPreference(    // notify.h
    HWND hWndParent,      // handle of parent window
    PPEG_USER_NOTIFICATION IpNotification      // structure with notification settings
);
```

Parameters

hWndParent

Identifies the parent window for the notification settings dialog box.

IpNotification

Points to a PEG_USER_NOTIFICATION structure. When calling the function, this structure contains data used to initialize the notification settings dialog box. When the function returns, this structure contains the user's notification settings.

Return Values

If the fucntion succeeds, the return value is TRUE. If the function returns TRUE, the returned settings should be saved, and considered when calling PegSetUserNotification.

If the function fails, the return value is FALSE.

See Also

Windows CE Notifications, PegSetUserNotification, PEG_USER_NOTIFICATION

The PegHandleAppNotifications function marks as "handled" all notifications previously registered by the given application. The function turns off the LED and stops vibration (if they were on) only for the given application's events, and removes the taskbar annunciator.

```
BOOL PegHandleAppNotifications(    // notify.h
    TCHAR *pwszAppName          // name of application whose events are handled
);
```

Parameters

pwszAppName

Points to a null-terminated string that specifies the name of the application whose events are to be marked as "handled".

Return Values

If the function succeeds, the return value is TRUE. If the function fails, the return value is

APPENDIX A-continued

FALSE.
See Also
Windows CE Notifications, PegGetUserNotification, PegSetUserNotification
The PegRunAppAtEvent function starts running an application when the given event occurs.
Windows CE Notes:
Note NOTIFICATION_EVENT_SYSTEM_BOOT is not supported

```
BOOL PegRunAppAtEvent(      // notify.h
    TCHAR *pwszAppName,     // name of application to run
    LONG /WhichEvent        // event at which the application is to run
);
```

Parameters
pwszAppName
Points to a null-terminated string that specifies the name of the application to be started.
/WhichEvent
Specifies the event at which the application is to be started. This parameter can be one of the following values.

Value	Meaning
NOTIFICATION_EVENT_NONE	No events—remove all event registrations for this application.
NOTIFICATION_EVENT_SYNC_END	When data synchronization finishes.
NOTIFICATION_EVENT_ON_AC_POWER	When AC Power is connected.
NOTIFICATION_EVENT_OFF_AC_POWER	When AC power is disconnected.
NOTIFICATION_EVENT_NET_CONNECT	When a network connection is made.
NOTIFICATION_EVENT_NET_DISCONNECT	When the network is disconnected.
NOTIFICATION_EVENT_DEVICE_CHANGE	When a PCMCIA device is changed.
NOTIFICATION_EVENT_IR_DISCOVERED	When an infrared partner is found.
NOTIFICATION_EVENT_RS232_DETECTED	When an RS232 connection is made.
NOTIFICATION_EVENT_RESTORE_END	When a full device data restore completes.

Return Values
If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE.
Remarks
The application is started with a system-defined command line. If there was already an instance of the application running, the new instance must send a private message to the existing instance and then shut down. The command line, which corresponds to the registered event, can be one of the following string constants.

Constant	Value
APP_RUN_AT_BOOT	"AppRunAtBoot"
APP_RUN_AFTER_SYNC	"AppRunAfterSync"
APP_RUN_AT_AC_POWER_ON	"AppRunAtAcPowerOn"
APP_RUN_AT_AC_POWER_OFF	"AppRunAtAcPowerOff"
APP_RUN_AT_NET_CONNECT	"AppRunAtNetConnect"
APP_RUN_AT_NET_DISCONNECT	"AppRunAtNetDisconnect"
APP_RUN_AT_DEVICE_CHANGE	"AppRunDeviceChange"
APP_RUN_AT_IR_DISCOVERY	"AppRunAtIrDiscovery"
APP_RUN_AT_RS232_DETECT	"AppRunAtRs232Detect"
APP_RUN_AFTER_RESTORE	"AppRunAfterRestore"

Remarks
In some cases, the preceding strings are merely the prefix of the command line, and the rest of the command line is used as a parameter.
You should use this function sparingly, because automatically starting an application can confuse the user and cause low-memory conditions on a machine with restricted memory. Ideally, the application should be small and non-intrusive.
See Also
Window CE Notifications, PegRunAppAtTime, PegEventHasOccurred
The PegRunAppAtTime function requests the system to start running the given application at the given time.

```
BOOL PegRunAppAtTime(       // notify.h
    TCHAR *pwszAppName,     // name of application to run
    SYSTEMTIME *IpTime   // time when to run the application
);
```

Parameters
pwszAppName
Points to a null-terminated string that specifies the name of the application to be run.
IpTime
Points to a SYSTEMTIME structure that specifies the time when the given application is to be run. If this parameter is NULL, the existing run request is deleted and no new request is entered.
Return Values
If the function succeeds, the return value is TRUE. If the function fails. the value is FALSE.
Remarks
Calling this function replaces any previous run request.
The system passes the APP_RUN_AT_TIME string to the application as the command line. If an instance of the application is already running, the new instance must send a private message to the existing instance and then shut down.
You should use this function sparingly, because automatically starting an application can confuse the user and cause low-memory conditions on a machine with restricted memory.

APPENDIX A-continued

Ideally, the application should be small and non-intrusive.
See Also
Windows CE Notifications, PegRunAppAtEvent
The PegSetUserNotification function creates a new user notification or modifies an existing one.

```
HANDLE PegSetUserNotification(      // notify.h
    HANDLE hNotification,   // handle of the notification to overwrite, or zero
    TCHAR *pwszAppName,     // name of application that owns this notification
    SYSTEMTIME *lpTime, // time when the notification is to occur
    PPEG_USER_NOTIFICATION lpUserNotification          // contains notification
parameters
);
```

Parameters
hNotification
Identifies the notification to overwrite, or zero to add a new notification.
pwszAppName
Points to a null-terminated string that specifies the name of the application that owns this notification. The system uses the application's primary icon as the taskbar annunciator for the notification. The user can start the application by selecting the annunciator.
lpTime
Points to the SYSTEMTIME structure that specifies the time when the notification should occur.
lpUserNotification
Points to the PEG_USER_NOTIFICATION structure that describes the events that are to occur when the notification time is reached.
Return Values
If the function succeeds, the return value is the handle of the notification. An application can use the handle to overwrite or delete the notification. The return value is zero if the notification could not be set.
Remarks
The notification occurs at the specified time, without starting the application. The application can specify the notification options, including whether to light the LED, generate a sound, or display a dialog box. However, an application typically uses the PegGetUserNotificationPreferences function to allow the user to set the notification options.
The user can start the owning application when the notification occurs. In this case, the system starts a new instance of the application using the APP_RUN_TO_HANDLE_NOTIFICATION string as the prefix of the command line, and the notification handle (converted to a string) as the postfix. If another instance of hte application is already running, the new instance must pass a private message to the old instance and then shut down.
See Also
Windows CE Notifications, PegHandleAppNotifications
The PEG_USER_NOTIFICATION structure contains information used to initialize the user notifications settings dialog box, and receives the user's notification preferences entered by way of the dialog box. Also used when setting a user notification.

```
typedef struct UserNotificationType {
    DWORD ActionFlags;
    TCHAR *pwszDialogTitle;
    TCHAR *pwszDialogText;
    TCHAR *pwszSound;
    DWORD nMaxSound;
    DWORD dwReserved;
}   PEG_USER_NOTIFICATION, *PPEG_USER_NOTIFICATION;
```

Members
ActionFlags
Specifies the action to take when a notification event occurs. This parameter can be a combination of the following flags.
Value Meaning
PUN_LED Flash the LED.
PUN_VIBRATE Vibrate the device.
PUN_DIALOG Display the user notification dialog box. When this structure is passed to the PegSetUserNotification function, the pwszDialogTitle and pwszDialogText members must provide the title and text of the dialog box.
PUN_SOUND Play the sound specified by the pwszSound member. When passed to PSVN, the pwszSound member must provide the name of the sound file.
PUN_REPEAT Repeat the pwszSound for 10–15 seconds. Only valid if PUN_SOUND is set.
Any flag that is not valid on the current hardware platform is ignored.
pwszDialogTitle
Specifies the title of the user notification dialog box. If this parameter is NULL, no dialog is displayed. The PegGetUserNotificationPreferences function ignores this members.
pwszDialogText
Specifies the text of the user notification dialog box. If this parameter is NULL, no dialog is displayed. The PegGetUserNotificationPreferences function ignores this member.
pwszSound
Points to a buffer that contains the unqualified name of a sound file to play. (The file is assumed to reside in the system media directory.) This parameter is ignored if the ActionFlags member does include the PUN_SOUND flag.
nMaxSound

APPENDIX A-continued

Specifies the maximum length of the string that the PegGetUserNotificationPreferences function can copy into the pwszSound buffer. Because the string may be a path name in a future release, the buffer must be at least the length derived by the following expression: PATH_MAX * sizeof(TCHAR). This member is ignored by the PegSetUserNotification function.

dwReserved

Reserved; must be zero.

Remarks

This structure is passed in the PegGetUserNotificationPreferences function. Initial settings are used to populate the dialog. If the function returns TRUE, the returned settings should be saved, and considered when calling PegSetUserNotification. Settings for hardware not on the current device will be ignored.

It is also used when calling PegSetUserNotification, to describe what should happen when the notification time is reached.

See Also

Windows CE Notifications, PegGetUserNotificationPreferences, PegSetUserNotification

What is claimed is:

1. In a portable handheld computing device having an operating system that provides a graphical user interface environment capable of presenting at least one graphical window on a display and an external notification mechanism, a notification program executing on the operating system, comprising:

a graphical user interface which enables a user to set notification options;

a notification manager to manage one or more events, the notification manager generating a command to set a time when an event is scheduled;

an alarm manager to receive the set time command from the notification manager and to generate a set alarm command which informs a clock to set an alarm at the time of the event; and

an interrupt manager to receive an interrupt from the clock when the time of the event arrives and the pass interrupt to the notification manager so that the notification manager can activate the external notification mechanism.

2. A portable handheld computing device comprising:

a processor;

a display;

an operating system executing on the processor to provide a graphical user interface environment capable of presenting at least one graphical window on the display;

at least one application running on the operating system; and

a notification system that is callable by the application to alert a user of an event, the notification system having a sensory perceptible notification mechanism that is activated as a result of the event to notify the user.

3. A portable handheld computing device as recited in claim **2** wherein the notification mechanism comprises a light.

4. A portable handheld computing device as recited in claim **2** wherein the notification mechanism comprises a dialog box presented on the display.

5. A portable handheld computing device as recited in claim **2** wherein the notification mechanism comprises an audio generator.

6. A portable handheld computing device as recited in claim **2** wherein the notification mechanism comprises a vibration device.

7. A portable handheld computing device as recited in claim **2** wherein the notification system also has a deactivation mechanism to deactivate the notification mechanism.

8. A portable handheld computing device as recited in claim **2** wherein the notification mechanism comprises a light emitting diode (LED) and the notification system further comprises a button integrated with the LED to deactivate the LED.

9. A portable handheld computing device as recited in claim **2** wherein the notification system supports a graphical user interface which enables a user to set notification options specifying how the notification mechanism is to operate.

10. A portable handheld computing device as recited in claim **2** wherein:

the graphical user interface provided by the operating system has a taskbar and the event relates to the application: and

the notification system places a taskbar annunciator in the taskbar, which upon actuation, starts the application responsible for the event.

* * * * *

US007454718B2

(12) **United States Patent**
Vale

(10) **Patent No.: US 7,454,718 B2**
(45) **Date of Patent: Nov. 18, 2008**

(54) **BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM**

(75) Inventor: **Peter O. Vale**, Seattle, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 575 days.

(21) Appl. No.: **10/923,438**

(22) Filed: **Aug. 20, 2004**

(65) **Prior Publication Data**

US 2005/0022140 A1 Jan. 27, 2005

(51) **Int. Cl.**
G06F 3/14 (2006.01)
G06F 3/048 (2006.01)

(52) **U.S. Cl.** **715/864**; 715/785; 715/854

(58) **Field of Classification Search** 715/785, 715/864

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,677,708	A *	10/1997	Matthews et al.	345/684
5,739,821	A	4/1998	Ho	
5,874,936	A *	2/1999	Berstis et al.	715/785
5,905,497	A	5/1999	Vaughan	
6,005,573	A *	12/1999	Beyda et al.	715/784
6,061,063	A *	5/2000	Wagner et al.	715/784
6,128,012	A	10/2000	Seidensticker	
6,310,634	B1 *	10/2001	Bodnar et al.	715/854
7,249,325	B1 *	7/2007	Donaldson	715/777
2001/0022839	A1	9/2001	Ishigaki	
2002/0070980	A1	6/2002	Le	
2002/0112237	A1	8/2002	Kelts	
2002/0145631	A1 *	10/2002	Arbab et al.	345/786
2005/0081149	A1	4/2005	Vale	

FOREIGN PATENT DOCUMENTS

EP	056 1684	9/1993
WO	WO 98/29797	7/1998

OTHER PUBLICATIONS

Marran, N.L., Over-The-Air Subscriber Device Management Using CDMA Data and WAP, Virginia Tech, 1999, pp. 165-174.

Ming, Tham and Chaun, Tan Kay, Challenges in Designing User Interfaces for Handheld Communication Devices: A Case Study, Lawrence Erlbaum Assocaites, 1999, pp. 808-812.

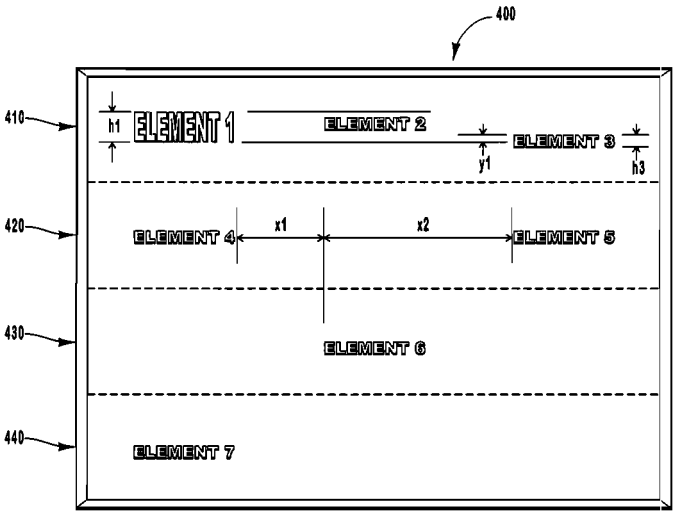
(Continued)

Primary Examiner—William L. Bashore
Assistant Examiner—Stephen Alvesteffer
(74) *Attorney, Agent, or Firm*—Workman Nydegger

(57) **ABSTRACT**

Methods, system, and computer program products for browsing content with a display area and input system that may be limited in comparison to more traditional browsing systems. Movement between and selection of interactive elements generally occurs in a navigation mode, whereas interaction with a single interactive element generally occurs in an edit mode. In navigation mode, a direction input selects the next interactive element in the direction indicated. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls. Switching between navigation mode and edit mode is based on the input received, in view of the input supported, by a particular interactive element. Interactive elements may be limited to the width of available display area.

28 Claims, 12 Drawing Sheets



OTHER PUBLICATIONS

Fox, A.; Goldberg, I.; Gribble, S.D.; Lee, D.C.; Polito, A.; and Brewer, E.A., Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for 3Com PalmPilot, University of California at Berkeley, 1998, pp. 407-424.

Gessler, Stefan and Kotulla, Andreas, PDAs as Mobile WWW Browsers, Germany 1995, pp. 53-59.

Office Action mailed May 31, 2007, cited in related application, U.S. Appl. No. 10/968,717.

Motorola, Digital Wireless Telephone, Model 120c, the User Guide, p. 35, 2001.

Kawachiya, Kiyokuni, et al., "NaviPoint: An Input Device for Mobile Information Browsing", Human Factors in Computing Systems. Conference Proceedings, Los Angeles, California, Apr. 18-23, 1998.

Notice of Allowance dated Jun. 24, 2008 cited in related U.S. Appl. No. 10/968,717 (Copy Attached).

No reference

* cited by examiner

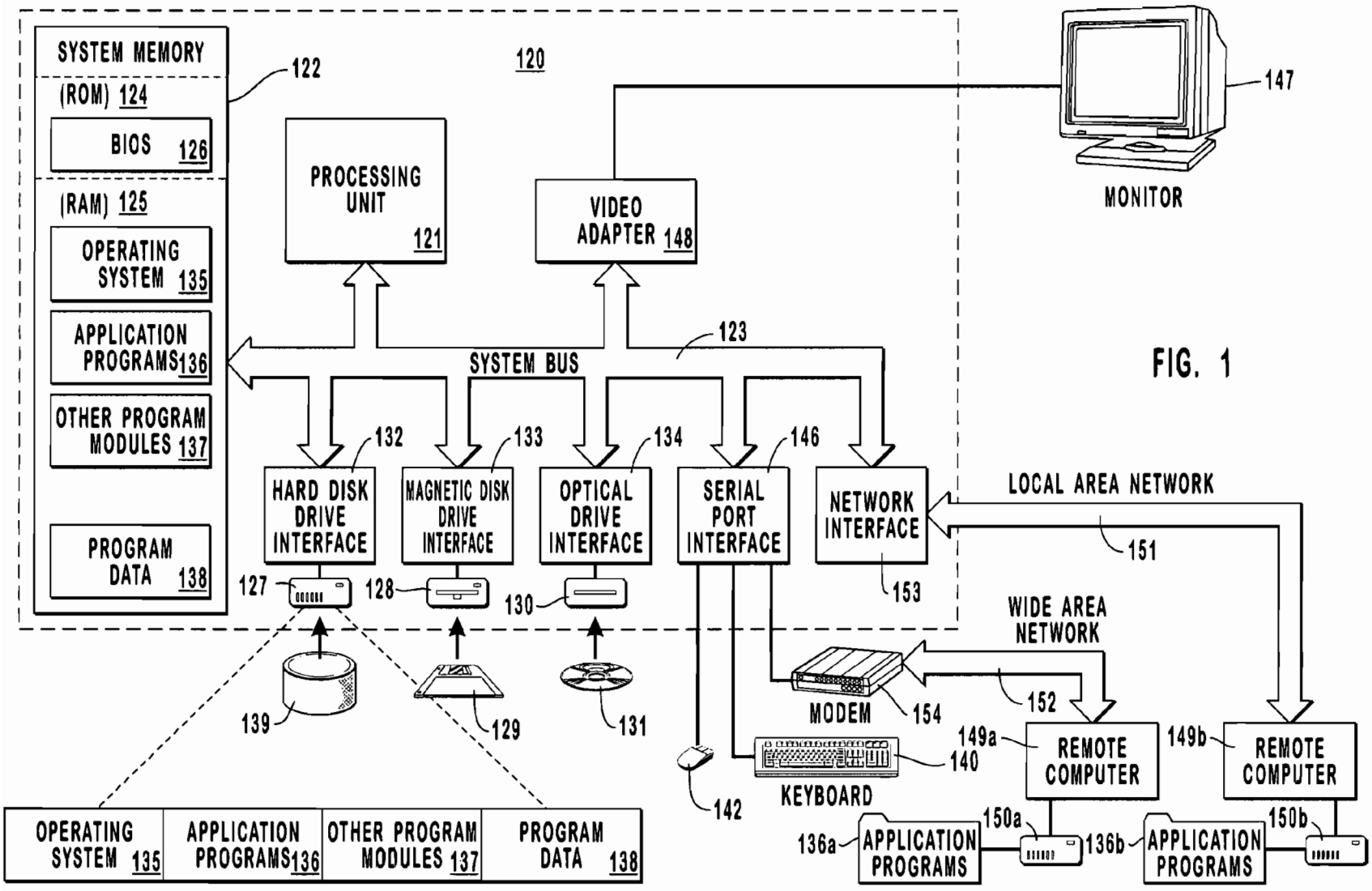

SYSTEM MEMORY
(ROM) 124
BIOS 126
(RAM) 125
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
122
120
PROCESSING UNIT 121
VIDEO ADAPTER 148
147
MONITOR
123
SYSTEM BUS
132
133
134
146
HARD DISK DRIVE INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
LOCAL AREA NETWORK
151
153
127
128
130
139
129
131
WIDE AREA NETWORK
MODEM
154
152
149a
149b
REMOTE COMPUTER
REMOTE COMPUTER
140
KEYBOARD
142
150a
150b
136a
136b
APPLICATION PROGRAMS
APPLICATION PROGRAMS
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138

FIG. 1

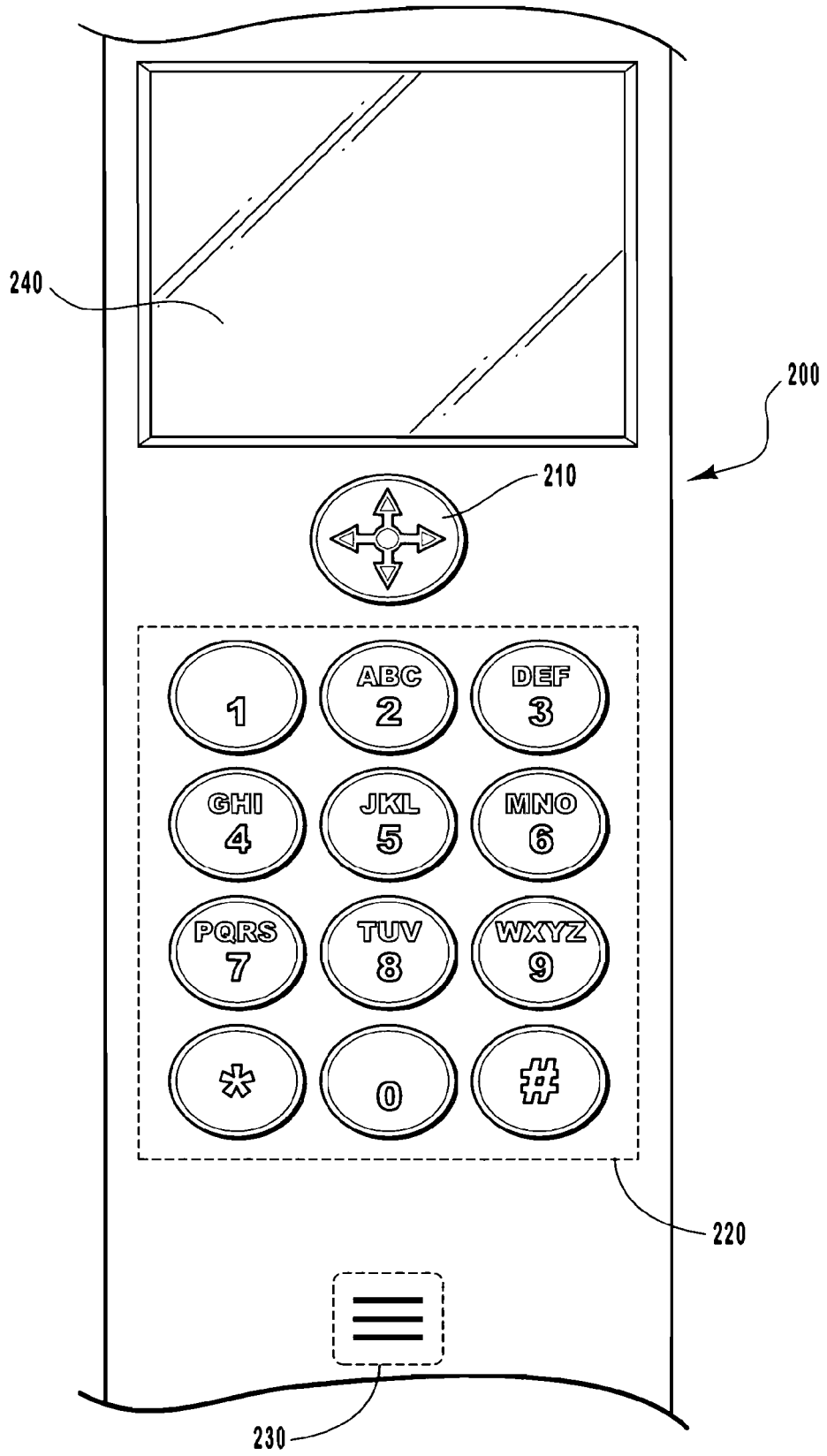

240
200
210
1
ABC 2
DEF 3
GHI 4
JKL 5
MNO 6
PQRS 7
TUV 8
WXYZ 9
*
0
#
220
230

FIG. 2

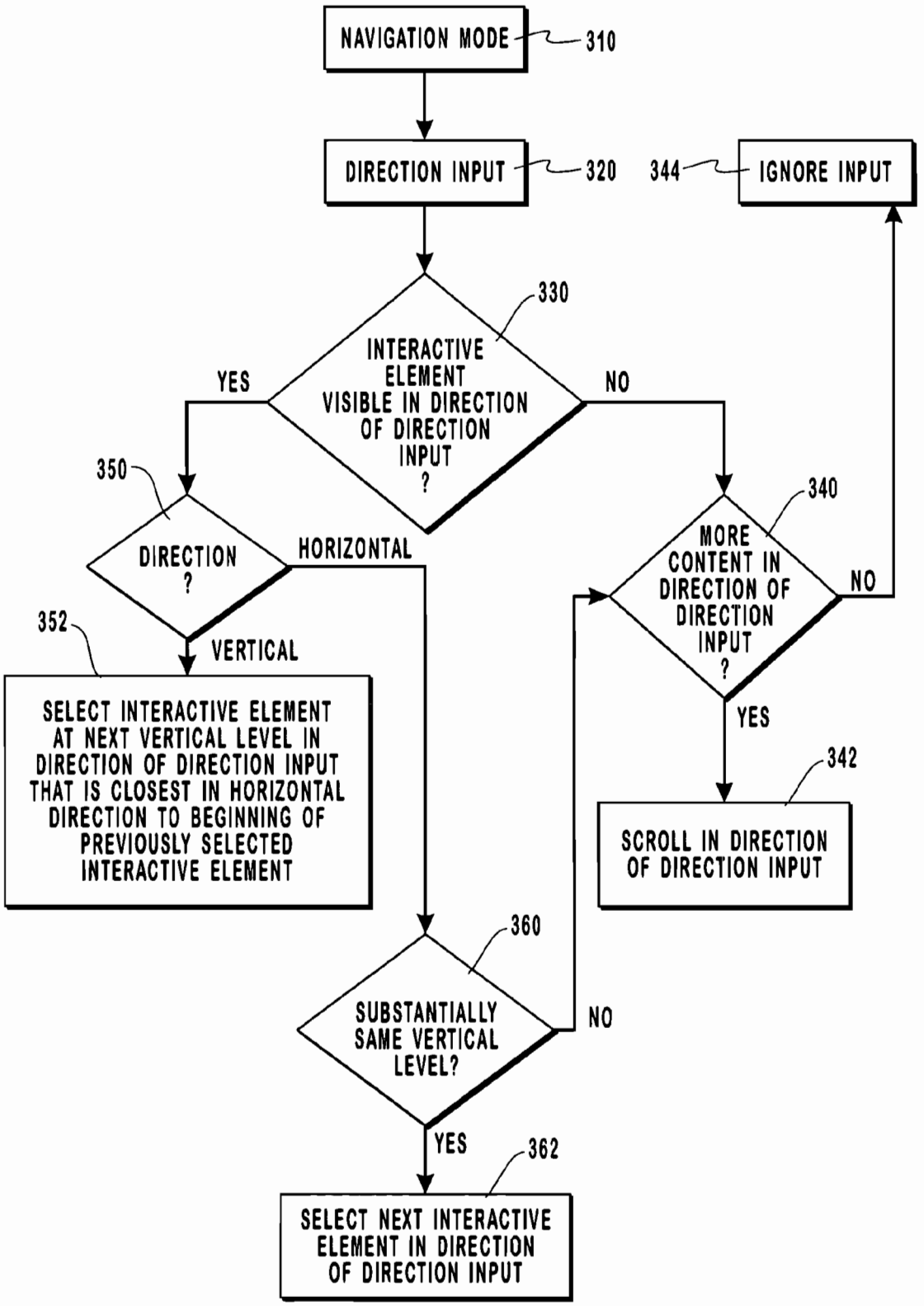

NAVIGATION MODE
310
DIRECTION INPUT
320
344
IGNORE INPUT
330
INTERACTIVE ELEMENT VISIBLE IN DIRECTION OF DIRECTION INPUT ?
YES
NO
350
DIRECTION ?
HORIZONTAL
VERTICAL
352
SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL IN DIRECTION OF DIRECTION INPUT THAT IS CLOSEST IN HORIZONTAL DIRECTION TO BEGINNING OF PREVIOUSLY SELECTED INTERACTIVE ELEMENT
340
MORE CONTENT IN DIRECTION OF DIRECTION INPUT ?
NO
YES
342
SCROLL IN DIRECTION OF DIRECTION INPUT
360
SUBSTANTIALLY SAME VERTICAL LEVEL?
NO
YES
362
SELECT NEXT INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT

FIG. 3

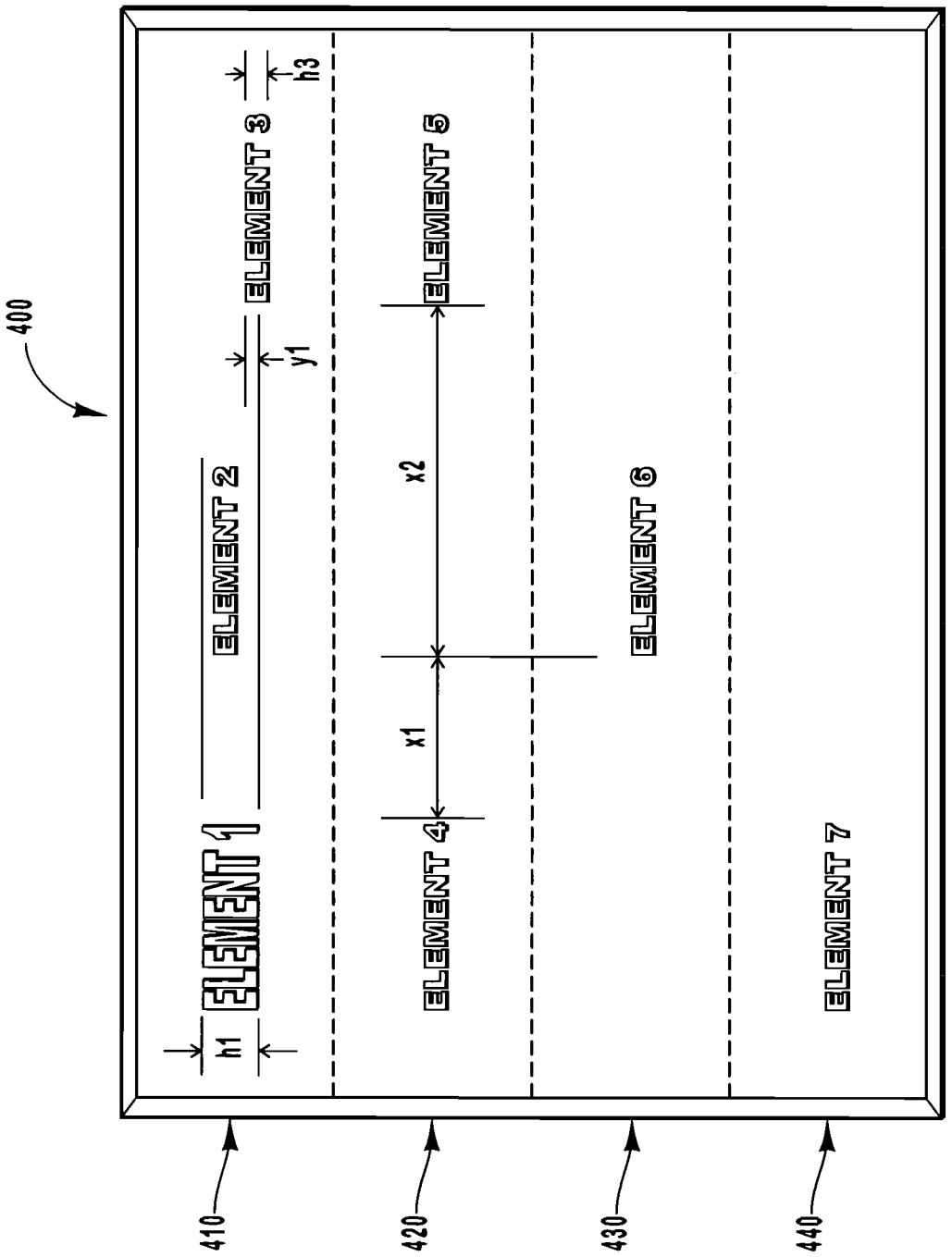

400
ELEMENT 1
ELEMENT 2
ELEMENT 3
ELEMENT 4
ELEMENT 5
ELEMENT 6
ELEMENT 7
h1
h3
y1
x1
x2
410
420
430
440

FIG. 4

FIG. 5A

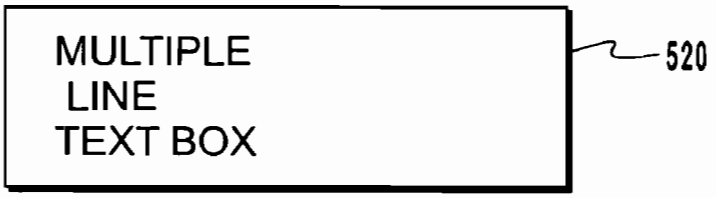


FIG. 5B

FIG. 5C

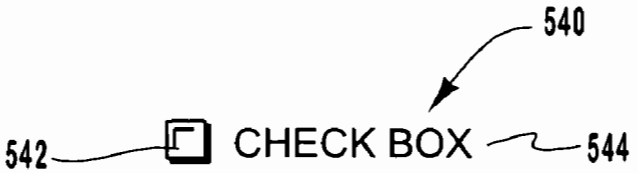


FIG. 5D

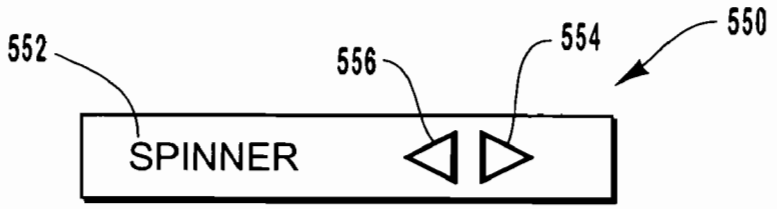


FIG. 5E

FIG. 5F

FIG. 5G

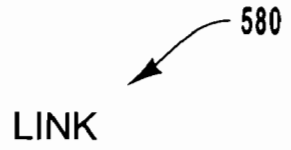


FIG. 5H

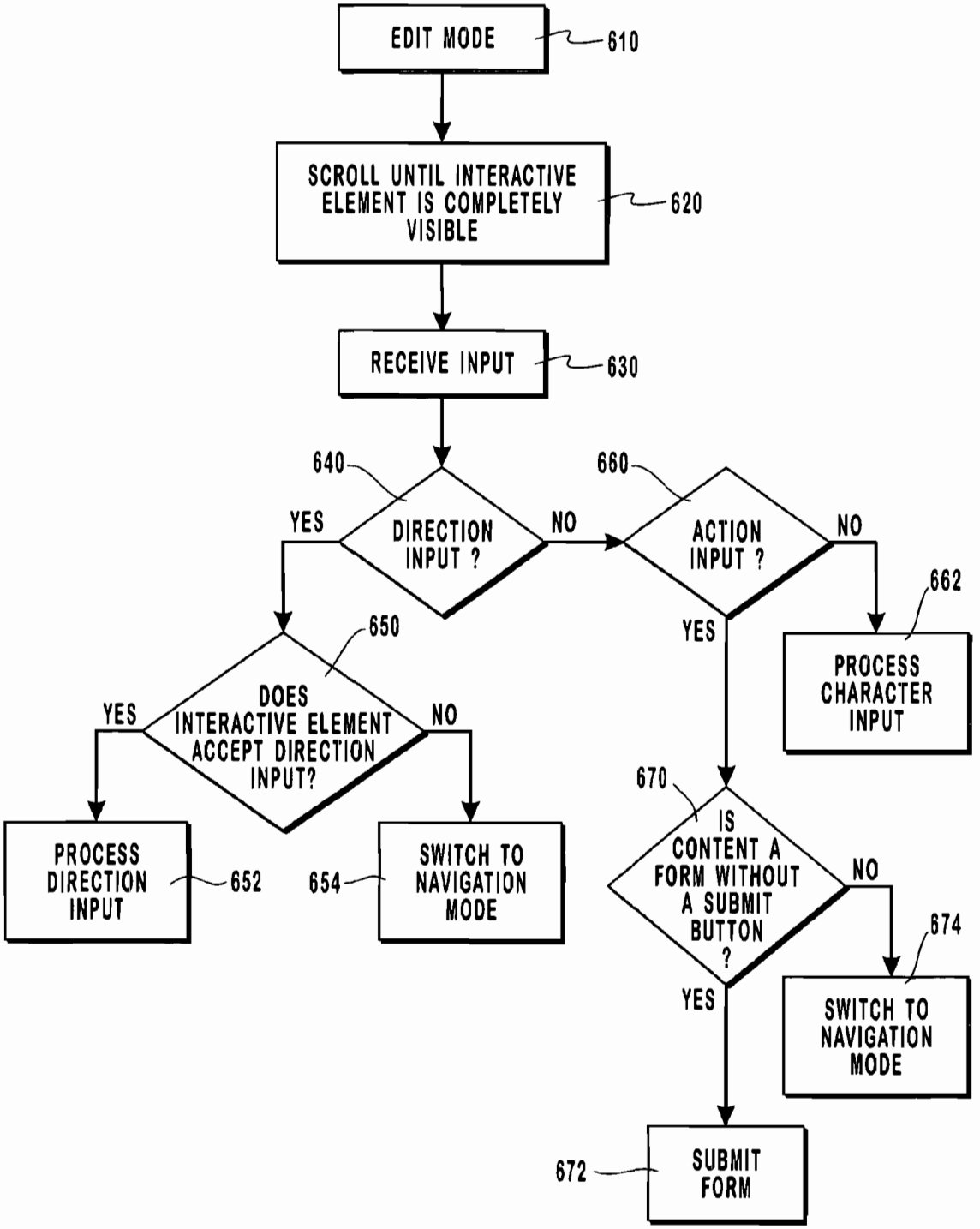

EDIT MODE
610
SCROLL UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
620
RECEIVE INPUT
630
640
DIRECTION INPUT ?
YES
NO
660
ACTION INPUT ?
NO
YES
662
PROCESS CHARACTER INPUT
650
DOES INTERACTIVE ELEMENT ACCEPT DIRECTION INPUT?
YES
NO
PROCESS DIRECTION INPUT
652
654
SWITCH TO NAVIGATION MODE
670
IS CONTENT A FORM WITHOUT A SUBMIT BUTTON ?
NO
YES
674
SWITCH TO NAVIGATION MODE
672
SUBMIT FORM

FIG. 6

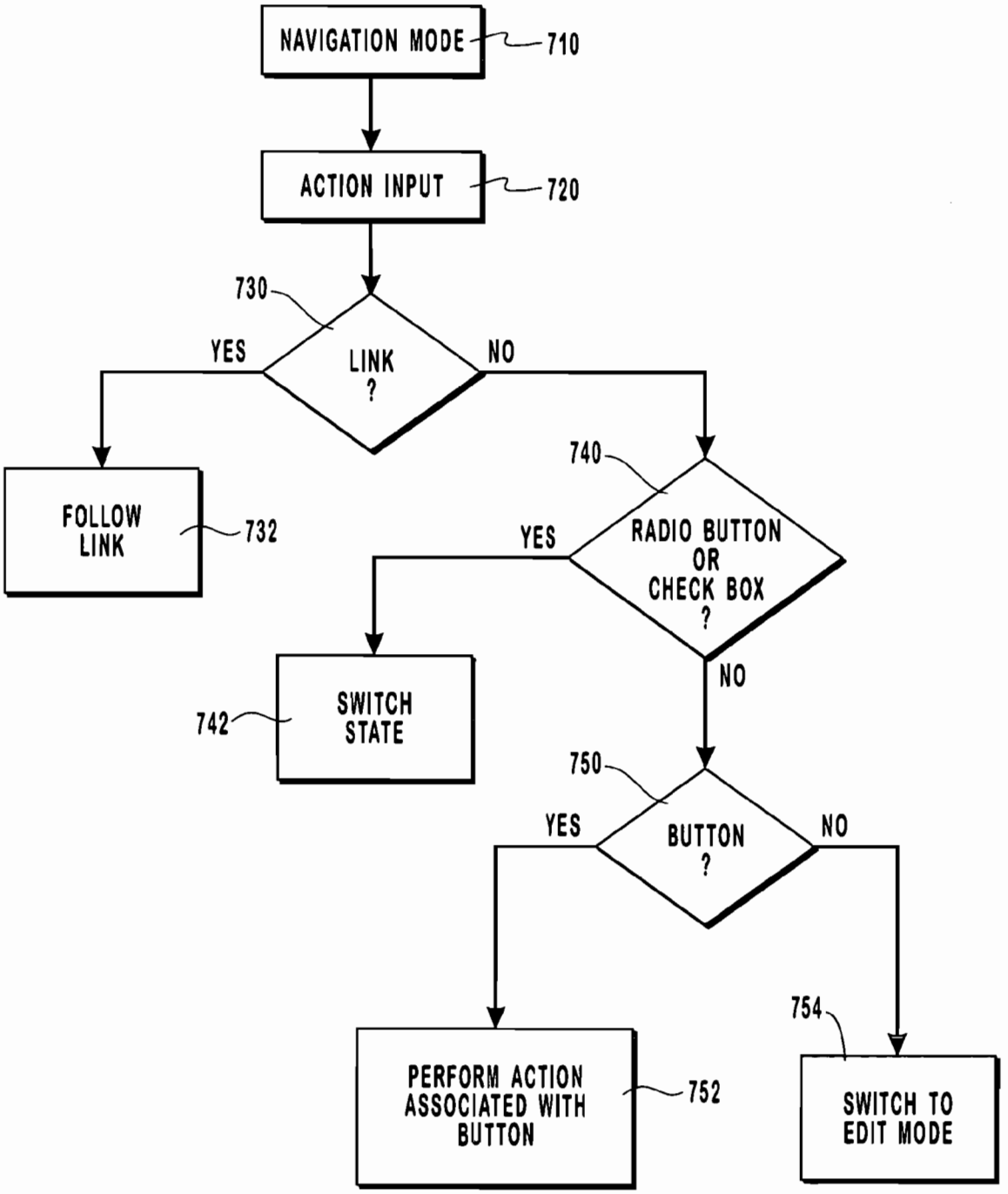

NAVIGATION MODE
710
ACTION INPUT
720
730
LINK ?
YES
NO
FOLLOW LINK
732
740
RADIO BUTTON OR CHECK BOX ?
YES
NO
742
SWITCH STATE
750
BUTTON ?
YES
NO
PERFORM ACTION ASSOCIATED WITH BUTTON
752
754
SWITCH TO EDIT MODE

FIG. 7

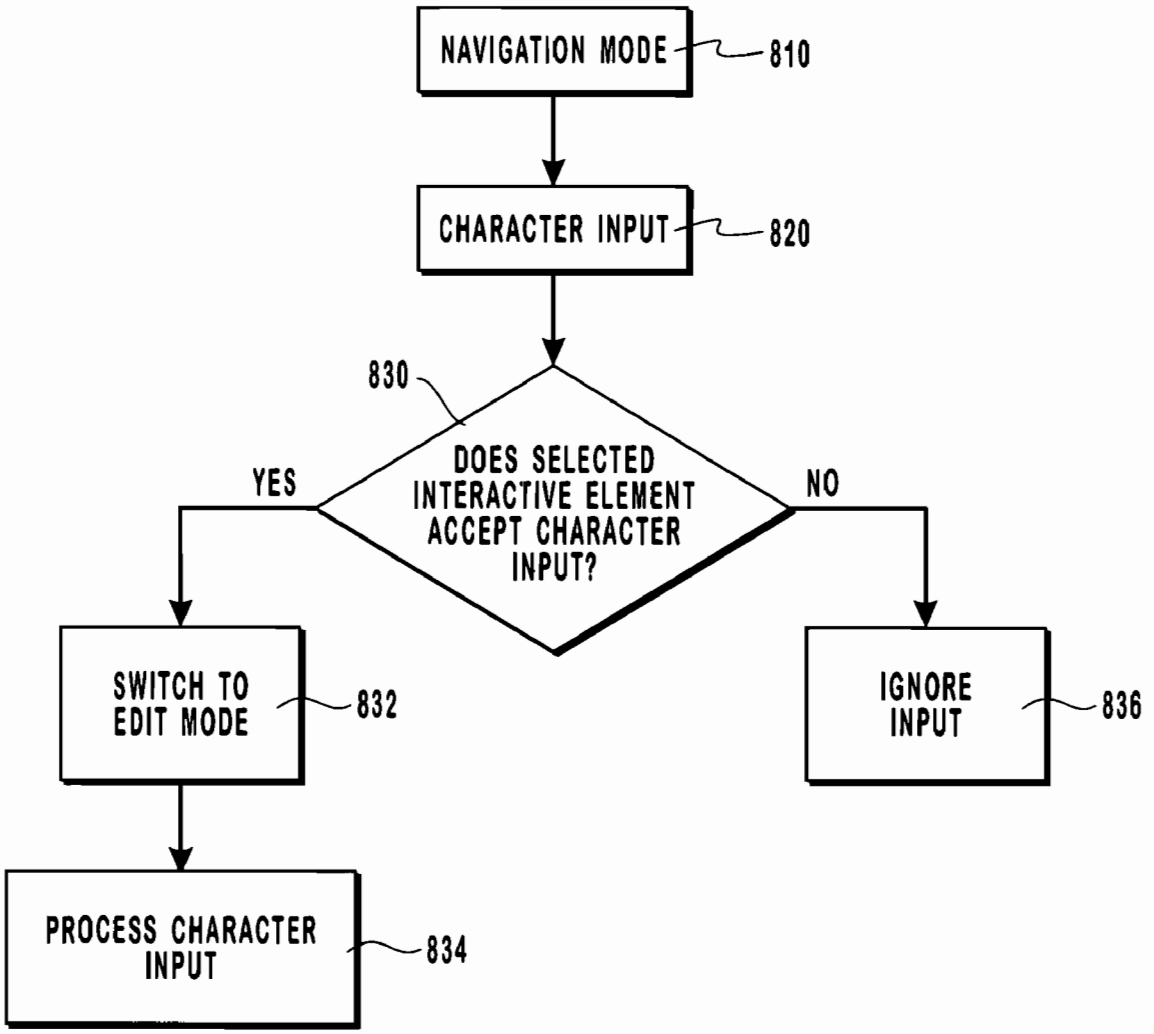

NAVIGATION MODE
810
CHARACTER INPUT
820
830
DOES SELECTED INTERACTIVE ELEMENT ACCEPT CHARACTER INPUT?
YES
NO
SWITCH TO EDIT MODE
832
IGNORE INPUT
836
PROCESS CHARACTER INPUT
834

FIG. 8

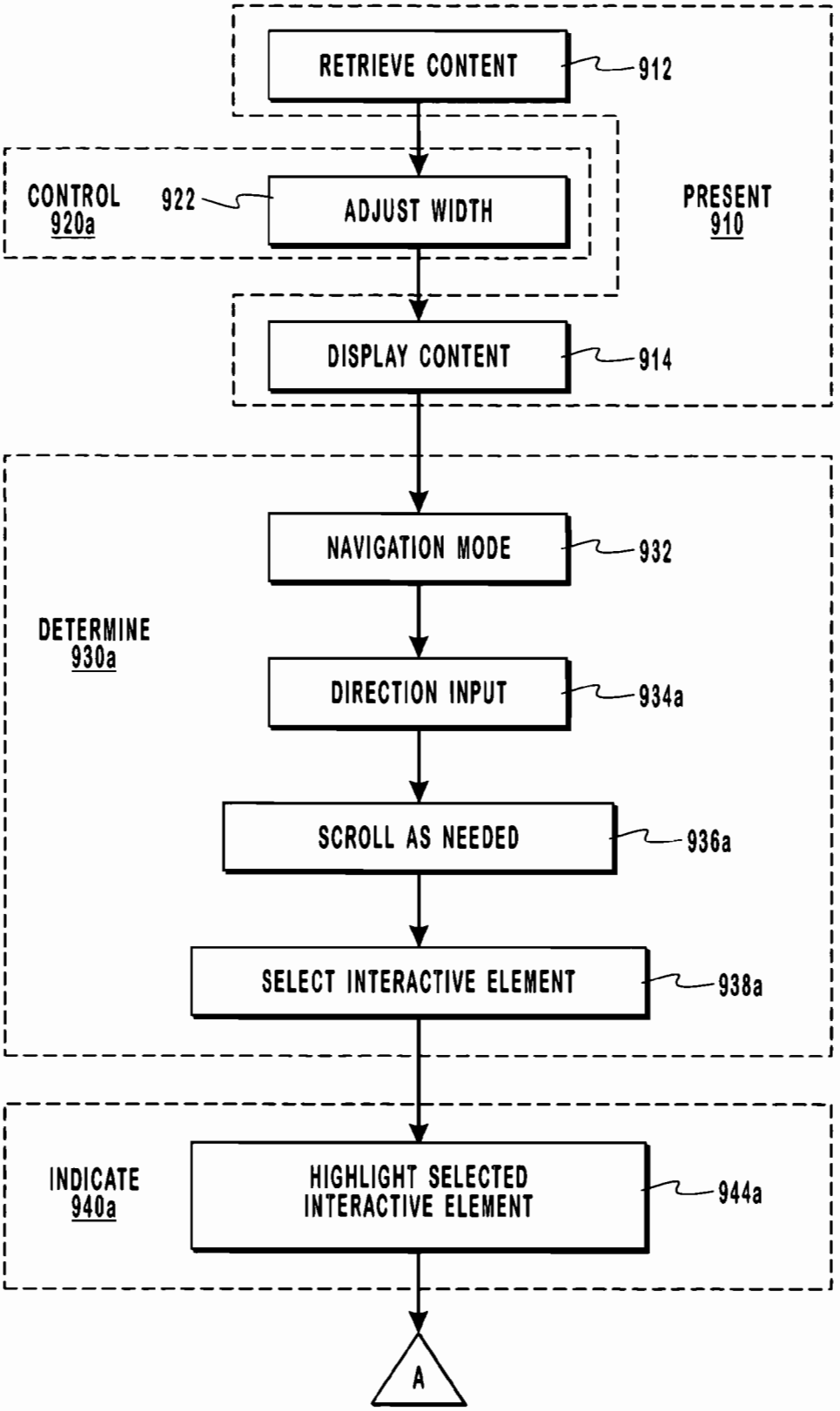

RETRIEVE CONTENT
912
CONTROL
920a
922
ADJUST WIDTH
PRESENT
910
DISPLAY CONTENT
914
NAVIGATION MODE
932
DETERMINE
930a
DIRECTION INPUT
934a
SCROLL AS NEEDED
936a
SELECT INTERACTIVE ELEMENT
938a
INDICATE
940a
HIGHLIGHT SELECTED
INTERACTIVE ELEMENT
944a
A

FIG. 9A

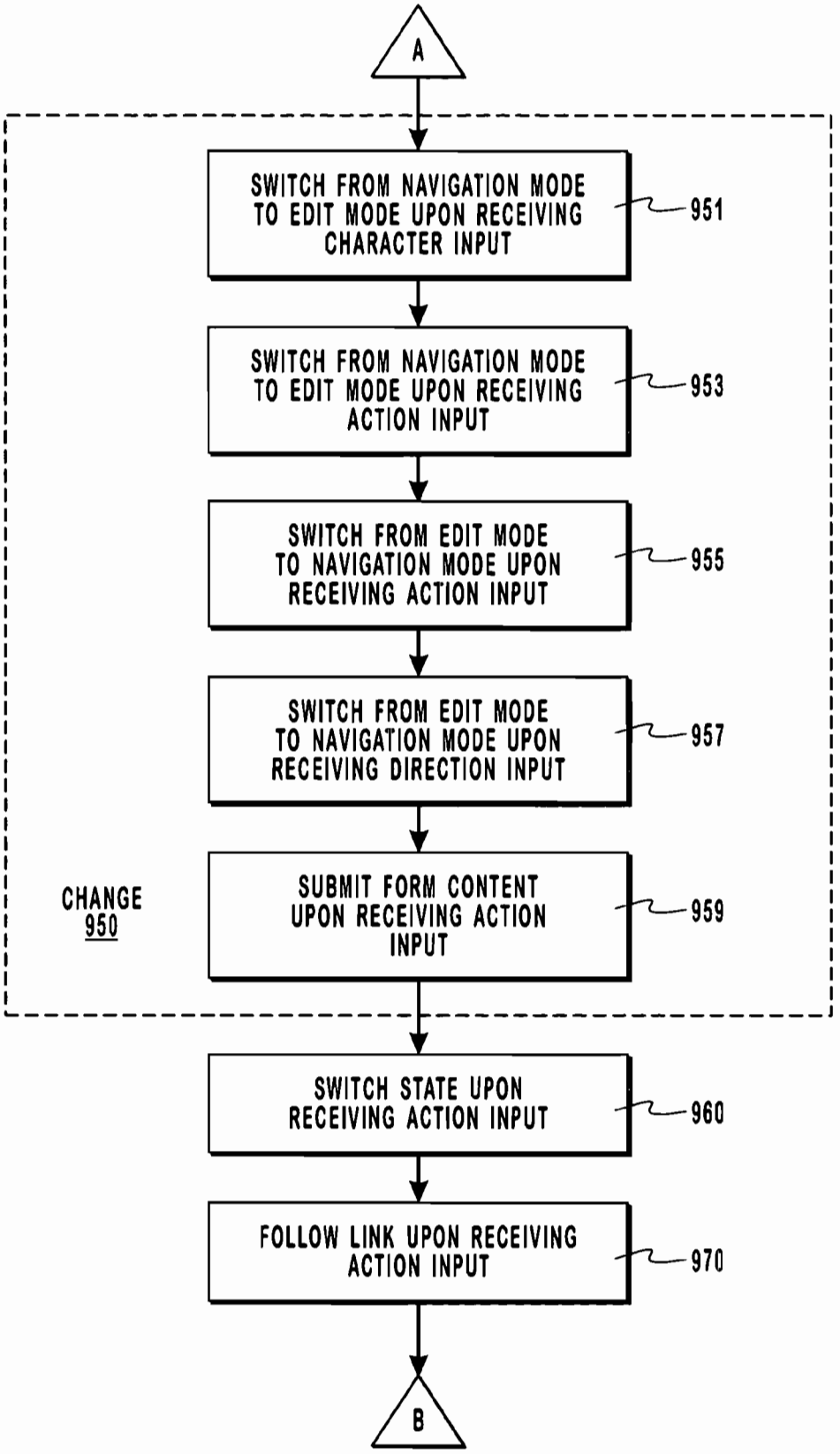

A
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING CHARACTER INPUT
951
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING ACTION INPUT
953
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING ACTION INPUT
955
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING DIRECTION INPUT
957
CHANGE 950
SUBMIT FORM CONTENT UPON RECEIVING ACTION INPUT
959
SWITCH STATE UPON RECEIVING ACTION INPUT
960
FOLLOW LINK UPON RECEIVING ACTION INPUT
970
B

FIG. 9B

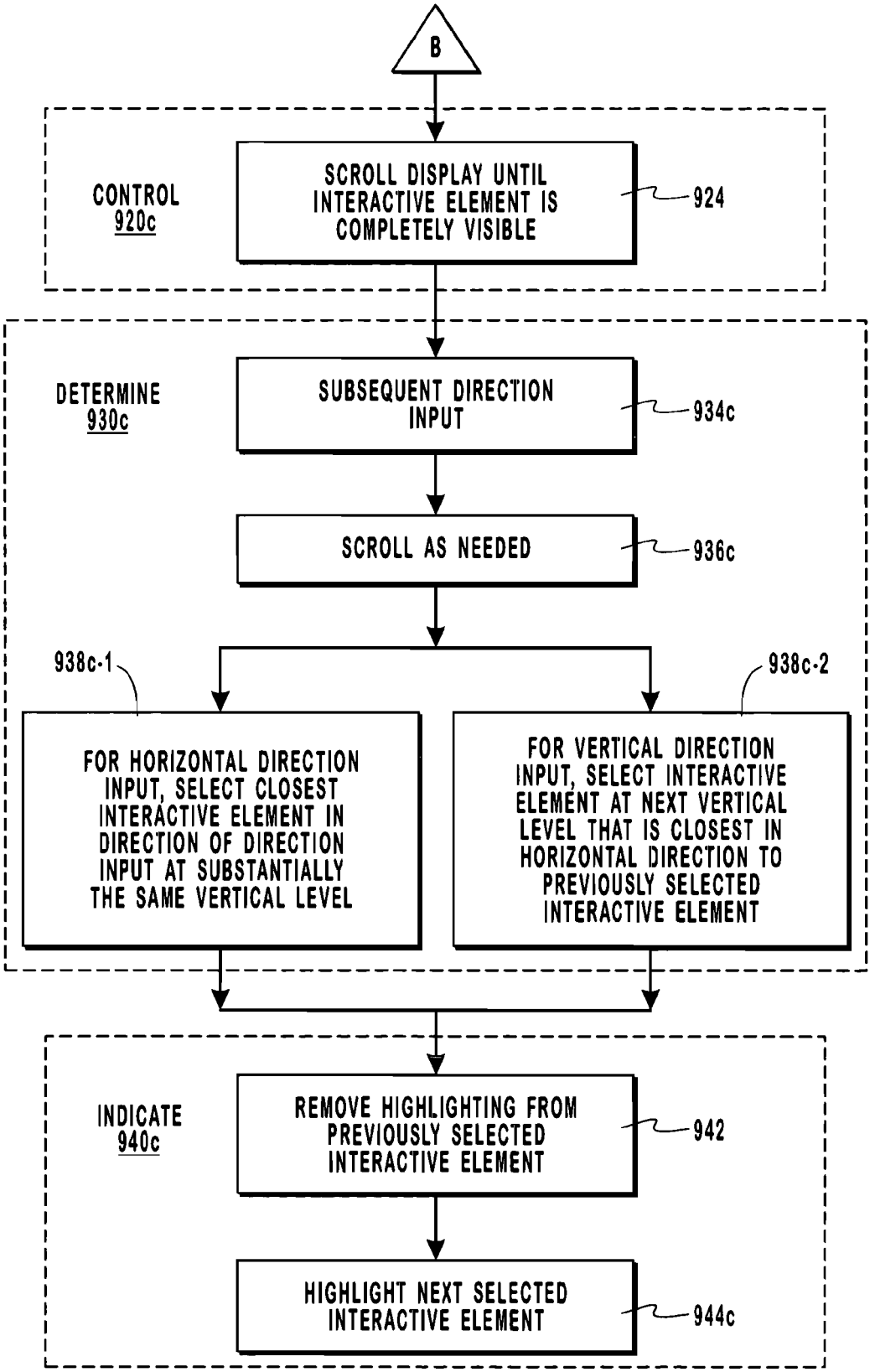

B
CONTROL
920c
SCROLL DISPLAY UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
924
DETERMINE
930c
SUBSEQUENT DIRECTION INPUT
934c
SCROLL AS NEEDED
936c
938c-1
938c-2
FOR HORIZONTAL DIRECTION INPUT, SELECT CLOSEST INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT AT SUBSTANTIALLY THE SAME VERTICAL LEVEL
FOR VERTICAL DIRECTION INPUT, SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL THAT IS CLOSEST IN HORIZONTAL DIRECTION TO PREVIOUSLY SELECTED INTERACTIVE ELEMENT
INDICATE
940c
REMOVE HIGHLIGHTING FROM PREVIOUSLY SELECTED INTERACTIVE ELEMENT
942
HIGHLIGHT NEXT SELECTED INTERACTIVE ELEMENT
944c

FIG. 9C

BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/239,600, entitled, "BROWSER NAVIGATION FOR DEVICES WITH LIMITED INPUT MECHANISMS," filed Oct. 11, 2000, and of U.S. application Ser. No. 09/861,327, entitled "BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM", filed May. 18, 2001, both of which are hereby incorporated by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates to browsing electronic content. More specifically, the present invention relates to methods, systems, and computer program products for browsing content that includes interactive elements using a computerized system with a display area and input system that may be somewhat limited in comparison to the pointing devices and displays typically found in more traditional browsing systems.

2. Background and Relevant Art

Content typically includes interactive elements, such as links and form controls. Activating or following a link causes the content that is associated with the link to be requested and displayed. Selecting a form control allows for interaction with the form control. Traditional browsing systems generally include a keyboard and a pointing device such as a mouse, for activating links and interacting with form controls. Tab order navigation is possible, but may not follow an order expected by the user, especially if scrolling is required to view all of the content.

In traditional browsing systems, a user activates a link or selects a form control by simply placing a mouse pointer over the interactive element and pressing a mouse button. With each mouse press, a user may follow a link, select a text field so that text may be entered from a keyboard, toggle a radio button or checkbox, choose one or more items from a list, or cause the action associated with a button to be executed. The mouse also is used in scrolling the display area, as necessary. Nevertheless, content often is authored to minimize scrolling the display of traditional browsing systems, particularly in the horizontal direction.

Browsing systems with limited input systems and display areas, however, such as a phone having a numeric keypad, a directional control, and an action key, may make it difficult to select and interact with content designed for more traditional browsing systems that make use of pointing devices and have larger display areas. For example, without a pointing device, how are links activated and how are form controls selected? The direction control is a natural choice for scrolling because this operation is similar to many traditional browsing systems. (When no interactive element is selected, arrow keys usually are used for scrolling.) But, without a mouse, selecting individual interactive elements presents a significant challenge.

Tab order navigation does not provide an adequate solution because tab order generally follows the order of interactive elements in the content as authored or written, rather than the order of interactive elements in the content as displayed. Thus, in some situations, tab order moves horizontally, and in other situations, tab order moves vertically. For example, content that includes a table often will have a vertical tab order within individual table cells, but a horizontal tab order from cell to cell. Content outside of a table usually has a horizontal tab order. Because users generally are unaware of whether content includes a table or not, tab order may appear completely arbitrary, moving horizontally one time and vertically the next.

Therefore, when browsing content that includes interactive elements, methods, systems, and computer program products are needed for computerized systems that may have limited display areas and input systems, as compared to the pointing devices and displays typically found in more traditional browsing systems. Furthermore, certain interactive elements may be more intuitive in a browsing context, if those interactive elements operate somewhat differently from how they might function in an operating system shell environment.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a navigation mode and an edit mode for browsing content with a computerized system that may include a somewhat limited display area and/or input system. Navigation mode generally includes movement between and selection of interactive elements, whereas the edit mode generally includes interaction with a single interactive element. In navigation mode, pressing a direction key selects the next interactive element in the direction indicated by the direction key (e.g., up, down, left, right). When moving horizontally, an interactive element is in the direction indicated by the direction control if the interactive element is at substantially the same vertical level. For example, if a later element overlaps a previous element on a given vertical level by any amount, the two elements are considered to be at substantially the same vertical level. Vertical movement is to an interactive element at the next vertical level in the direction indicated by the direction control. If multiple interactive elements lie at the next vertical level, the one closest in the horizontal direction to the beginning of the current interactive element is selected.

To indicate selection, an interactive element is highlighted, such as by placing a selection box around the element. The interactive element remains selected until it is no longer visible (i.e., it has scrolled off the display area) or the next interactive element becomes at least partially visible and is selected. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls in the direction indicated by the direction control.

Switching from navigation mode to edit mode may be accomplished in several ways. For example, once an interactive element allowing character entry is selected, typing a character on the keypad automatically switches from navigation mode to edit mode. Similar to a mouse click, pressing the action button after an interactive element has been selected also switches to edit mode. Where interactive elements only require a mouse click to function in traditional browsing systems, such as links, checkboxes, radio buttons, other buttons, and the like, pressing action uses the selected control (i.e., follows the link, checks or unchecks a checkbox, chooses a radio button, performs the action associated with the button, etc.) rather than switching to edit mode. In edit mode, pressing the action button switches back to navigation mode. If a particular direction key is not allowed in edit mode, pressing the direction key also will exit edit mode. For forms that do not include a submit button on the form, pressing the action key will submit the form, rather than switching to navigation mode.

Certain interactive elements may be limited to the width of the display area that is available for displaying content so that the entire element can be visible at one time. Therefore, the width of an interactive element that exceeds the width of available display area may be adjusted to be less than or equal to the width of available display area. If a selected interactive element is only partially visible, switching into edit mode scrolls the control into full view.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered as limiting its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** illustrates an exemplary system that provides a suitable operating environment for the present invention;

FIG. **2** shows a portion of a wireless telephone;

FIG. **3** is a flow diagram that corresponds to receiving a direction input while navigating between interactive elements;

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation;

FIGS. **5**A-**5**H illustrate various interactive elements;

FIG. **6** is a flow diagram that corresponds to receiving input while editing interactive elements;

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements;

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements; and

FIGS. **9**A-**9**C show an exemplary method for browsing content according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention extends to methods, systems, and computer program products for browsing content that includes interactive elements using a browsing system with a display area and input system that may be limited in comparison to the pointing devices and displays typically found in more traditional browsing systems. As used in this application, the term "browsing system" should be interpreted broadly to encompass any computerized system for locating and presenting content, including text, images, audio, video, computer instructions, and the like. With the popularity of the World Wide Web ("Web"), content frequently is formatted using hypertext markup language ("HTML") and computer instructions often are embedded in content using Javascript. Both HTML and Javascript allow for the creation of interactive elements within content.

Those of skill in the art, however, will recognize that a wide variety of markup and scripting languages exist. In particular, extensible markup language ("XML") is becoming increasingly popular because it allows for user-defined extensions to the language. Note also that content may be converted from one language to another. Furthermore, it is anticipated that additional formats, languages, technology and/or standards for authoring content with interactive elements will become available in the future. The present invention, therefore, does not impose any requirements on how content with interactive elements is authored, whether based on current or future technology. Thus, any reference to HTML and/or Javascript, either explicit or implied, should be interpreted as exemplary of aspects or embodiments of the present invention and not as limiting its scope.

Those skilled in the art also will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, mobile/hand-held devices, such as personal digital assistants ("PDAs") and wireless telephones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. The embodiments of the present invention may comprise a special purpose or general purpose computer including various computer hardware, as discussed in greater detail below.

Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that may be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, flash memory, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such a connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media. Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions.

FIG. **1** and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention may be described in the general context of computer-executable instructions, such as program modules, being executed by computers in network environ-

ments. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represent examples of corresponding acts for implementing the functions described in such steps.

With reference to FIG. **1**, an exemplary system for implementing the invention comprises a general purpose computing device in the form of a generic computer **120**, including a processing unit **121**, a system memory **122**, and a system bus **123** that couples various system components including the system memory **122** to the processing unit **121**. Although some components of computer **120**, such as monitor **147**, keyboard **140**, and mouse **142**, may seem specific to a conventional computer, those of skill in the art will recognize that analogous components may be found in other computing devices. For example, wireless telephones often include an LCD or plasma display area, a numeric keypad, and one or more navigation buttons. Therefore, any component described with reference to generic computer **120** should be interpreted broadly to encompass analogous components that are appropriate for and consistent with a particular implementation or embodiment of the present invention. In some implementations, a component may be connected only intermittently or may be missing entirely.

The system bus **123** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) **124** and random access memory (RAM) **125**. A basic input/output system (BIOS) **126**, containing the basic routines that help transfer information between elements within the computer **120** such as during start-up, may be stored in ROM **124**.

The computer **120** may also include a magnetic hard disk drive **127** for reading from and writing to a magnetic hard disk **139**, a magnetic disk drive **128** for reading from or writing to a removable magnetic disk **129**, and an optical disk drive **130** for reading from or writing to removable optical disk **131** such as a CD-ROM or other optical media. The magnetic hard disk drive **127**, magnetic disk drive **128**, and optical disk drive **130** are connected to the system bus **123** by a hard disk drive interface **132**, a magnetic disk drive-interface **133**, and an optical drive interface **134**, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-executable instructions, data structures, program modules and other data for the computer **120**. Although the exemplary environment described herein employs a magnetic hard disk **139**, a removable magnetic disk **129** and a removable optical disk **131**, other types of computer readable media for storing data can be used, including magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs, and the like. Note that decreasing form factors are making it practical to use at least some of the foregoing components with mobile devices. Furthermore, it is anticipated that future technological advances with respect to size, power consumption, and the like, will lead to an increased selection of storage options.

Program code means comprising one or more program modules may be stored on the hard disk **139**, magnetic disk **129**, optical disk **131**, ROM **124** or RAM **125**, including an operating system **135**, one or more application programs **136**, other program modules **137**, and program data **138**. A user may enter commands and information into the computer **120** through keyboard **140**, pointing device **142**, or other input devices (not shown), such as a numeric keypad, directional buttons, pressure-sensitive software keyboard, microphone, joy stick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit **121** through a serial port interface **146** coupled to system bus **123**. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor **147** or another display device, such as an LCD or gas plasma display, is also connected to system bus **123** via an interface, such as video adapter **148**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer **120** may operate in a networked environment using logical connections to one or more remote computers, such as remote computers **149***a* and **149***b*. Remote computers **149***a* and **149***b* may each be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically include many or all of the elements described above relative to the computer **120**, although only memory storage devices **150***a* and **150***b* and their associated application programs **136***a* and **136***b* have been illustrated in FIG. **1**. The logical connections depicted in FIG. **1** include a local area network (LAN) **151** and a wide area network (WAN) **152** that are presented here by way of example and not limitation. Such networking environments are commonplace in office-wide or enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer **120** is connected to the local network **151** though a network interface or adapter **153**. When used in a WAN networking environment, the computer **120** may include a modem **154**, a wireless link, or other means for establishing communications over the wide area network **152**, such as the Internet. The modem **154**, which may be internal or external, is connected to the system bus **123** via the serial port interface **146**. In a networked environment, program modules depicted relative to the computer **120**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing communications over wide area network **152** may be used.

FIG. **2** shows a portion of a wireless telephone. A wireless telephone is merely one example of a browsing system with limited display and input capabilities. Typically, PDAs and other handheld devices also have limited displays and input systems. The present invention, however, is not necessarily limited to any particular hardware or device, handheld or otherwise. Nevertheless, some benefits provided by the present invention may be more pronounced where displays and/or input systems are less robust than corresponding displays and/or input systems found in traditional browsing systems.

Four-direction and action key **210** is an example of both a navigation key generating direction input and an action key providing action input. Depressing key **210** at any one of the four arrows generates a direction input corresponding to the direction of the arrow. An action input is generated by depressing the center of key **210**. The center of key **210** may be a separate button (not shown) or may be integral with the navigation arrows such that depressing the center generates simultaneous, but conflicting direction input. In other words, simultaneous up and down arrows or simultaneous left and right arrows are interpreted as an action input. Typically, an

action input corresponds to pressing an enter key on a keyboard, but other actions are not precluded, as circumstances may warrant.

A user may enter characters, such as numbers, letters, punctuation, etc., with keypad **220**. Audio input **230** is the mobile telephone's mouthpiece. Display area **240** displays content received while browsing. Note however, that all of display area **240** may not be available for displaying content. For example, portions of display area **240** may be used for titles, menus, switching between tasks, etc. Therefore, available display area generally refers to the portion of display area **240** that is devoted to displaying content, and may represent all or less than all of display area **240**.

As noted previously, the present invention provides for a navigation mode and an edit mode. Edit mode generally is characterized by interaction with a single interactive element and is described more fully in connection with FIG. **6** and the interactive elements shown in FIGS. **5**A-**5**H. In contrast, navigation mode generally is characterized by movement between and selection of interactive elements (i.e., changing focus from one interactive element to another) and is described more fuilly in connection with FIGS. **3** and **4**. Transitions from navigation mode to edit mode and interactive elements that operate in an intuitive manner without using a dedicated edit mode are covered in the description of FIGS. **7** and **8**.

Turning first then to FIG. **3**, during operation in navigation mode **310**, a direction input **320** is received. Decision block **330** determines if an interactive element is at least partially visible in the direction of direction input **320**, either relative to the beginning of the content if no interactive element has been selected or relative to an interactive element that is selected currently. If no interactive element is visible in the direction of the direction input, decision block **340** determines if more content is available in the direction of the direction input. If more content is available in the direction of the direction input, the display scrolls **342** in the direction of the direction input; otherwise, the direction input is ignored **344**.

Returning to decision block **330**, if an interactive element is visible in the direction of direction input **320**, selecting the next interactive element depends on the direction of direction input **320**, unless no interactive element has been selected previously, wherein the interactive element closest to the beginning of the content is selected (not shown). For horizontal input, decision block **360** determines if the visible interactive element lies at substantially the same vertical level as the interactive element that is selected currently. (The meaning of substantially the same vertical level will be described in more detail below, with respect to FIG. **4**.) If substantially at the same level, the interactive element in the direction of the direction input is selected **362**. If the visible interactive element does not lie at the substantially the same vertical level, operation continues with decision block **340**, as described above. For vertical direction input, the interactive element at the next vertical level in the direction of direction input **320** that is closest in the horizontal direction to the beginning of the previously selected interactive element is selected **352**.

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation. The display **400** of content includes vertical levels **410**, **420**, **430**, and **440**. Note that at vertical level **410**, Element **1**, Element **2**, and Element **3** do not display at exactly the same vertical coordinates. Nevertheless, Element **1**, Element **2**, and Element **3** lie at substantially the same vertical level. By allowing for some variation in the vertical display coordinates for interactive elements, navigation is more intuitive.

In particular, the height of Element **1** is h**1** and the height of Element **3** is h**3**. The distance y**1** is the amount that Element **3** overlaps with Element **1**. Alternatively, y**1** may represent the amount of vertical separation between interactive elements rather than the amount of overlap. Note that the present invention does not necessarily limit y**1** to any particular dimension or calculation. Because y**1** reflects the expectations of users, it is possible for y**1** to take on a wide range of values, as may be suitable for a particular embodiment or implementation. However, in at least some circumstances, y**1** is a portion of h**3**, the height of Element **3**, indicating that Element **3** partially overlaps Element **1**. (Although not shown, note also that it may be possible for a single element to span and be reachable from multiple vertical levels. When navigating horizontally from an element spanning multiple vertical levels, the next element is selected from the top most spanned vertical level where an interactive element is visible.) The height of Element **2** is not shown because it overlaps completely with Element **1** and therefore is clearly at the same vertical level as Element **1**.

Initially no interactive element is selected. As indicated with respect to FIG. **3**, a direction input that is not in the direction of an interactive element that is at least partially visible or in a direction that permits scrolling will be ignored (see block **344**). Thus, direction input up or to the left will be ignored. Because initially no interactive element is selected, however, a down direction input or a right direction input will select Element **1** (i.e., the first interactive element relative to the beginning of the content). Once selected, an interactive element is highlighted, such as by drawing a dashed box around the element to indicate that the selected interactive element has focus. The present invention does not necessarily limit the type of highlighting used to indicate selection. It is only relevant for some form of visual cue to occur that is specific to the selected interactive element.

Left and right direction input will select interactive elements in numerical order, either ascending for right direction input or descending for left direction input. Up and down direction input is somewhat more complicated. Beginning with Element **1**, down direction input selects interactive elements in the following order: Element **1**, Element **4**, Element **6**, Element **7**. Beginning with Element **7**, up direction input selects interactive elements in reverse order: Element **7**, Element **6**, Element **4**, Element **1**.

Moving from vertical level **430** to vertical level **420** may include some ambiguity because vertical level **420** includes multiple interactive elements. Selecting between Element **4** and Element **5** depends on the horizontal distances labeled x**1** and x**2**. The distance x**1** represents the horizontal distance from the beginning of Element **6** (the currently selected interactive element when moving from vertical level **430** to vertical level **420**) to the nearest portion of Element **4**. Likewise, the distance x**2** represents the horizontal distance from the beginning of Element **6** to the nearest portion of Element **5**. When selection moves in the vertical direction and more than one interactive element lies at a vertical level, the interactive element closest in the horizontal direction to the beginning of the previously selected interactive element is selected next. In other words, Element **4** is selected if x**1** is less than or equal to x**2**, and Element **5** is selected if x**2** is less than x**1**. Note that the same processing occurs for moving between Element **1** and Element **4**, but the result is obvious. In contrast, there is no ambiguity in moving from Element **4** to Element **6** and then to Element **7**.

FIGS. **5**A-**5**H illustrate various interactive elements, the operation of which will be described in greater detail with respect to FIGS. **6**-**8**. Those of skill in the art will recognize

that the interactive elements shown in FIGS. **5**A-**5**H are merely exemplary of interactive elements that are useful in describing embodiments of the present invention in the context of HTML content. However, as explained previously, the present invention is not necessarily limited to any particular types of interactive elements or any particular type of content. For example, JAVA and Javascript allow for the creation of interactive elements and show that the types of interactive elements within the scope of the present invention are governed only by the creativity of those who author content. In addition to existing interactive elements, therefore, it is fully anticipated that new interactive elements will be developed and should be considered to fall within the meaning of interactive elements as used in this application, regardless of the particular technology that implements and/or deploys a particular interactive element. Furthermore, it should be apparent that the following discussion does not catalog all existing interactive elements, but rather, identifies a sufficient number to adequately describe how the present invention operates.

FIG. **5**A illustrates single line textbox **510** for entry of characters. Although display is limited to a single line, characters within the textbox may allow for scrolling if character entry exceeds the width of textbox **510**. FIG. **5**B illustrates multiple line textbox **520**. Multiple line textbox **520** is also for character entry. Due to display area constraints, multiple line textbox **520** may display as a single line in navigation mode, and then to facilitate editing, expand to a multiple line display in edit mode. A close button may be included with multiple line textbox **520** to assist in returning to navigation mode. Like single line textbox **510**, multiple line textbox **520** may allow for scrolling if character entry exceeds the width of textbox **520**.

Radio button **530**, with button **532** and text **534**, is illustrated in FIG. **5**C. Radio buttons allow for choosing one item and only one item from a group or list of items. FIG. **5**D illustrates checkbox **540**, with button **542** and text **544**. In contrast to radio button **530**, checkbox **540** allows for selecting zero or more items from a group or list of items. FIG. **5**E illustrates spinner **550**, with text **552**, left arrow **556**, and right arrow **554**. Similar to radio buttons, spinner **550** groups related items and allows one and only one to be chosen. Activating left arrow **556** chooses the previous item in the list and activating right arrow **554** chooses the next item in the list. The list may be circular, such that moving through all choices with either arrow returns to the initial choice. Alternatively the list may be linear, having a starting point that may be reached with left arrow **556** and having an ending point that may be reached with right arrow **554**. Picker **560**, with text **562** and right arrow **564**, as illustrated in FIG. **5**F, is similar to checkbox **540**. Activating right arrow **564** displays a list of checkbox options. Like multiple line textbox **520**, a close button may be included with picker **560** to facilitate returning to navigation mode. Activating button **570** of FIG. **5**G causes an action associated with the button to be executed. FIG. **5**H illustrates link **580**, a hypertext markup language link that browses content associated with the link when the link is activated or followed.

FIG. **6** is a flow diagram that corresponds to receiving input while editing a selected interactive element, such as one of those described above in connection with FIGS. **5**A-**5**H. Depending on the selected interactive element, input received while in edit mode may be used by the interactive element (e.g., entering characters into a textbox) or may cause a return to navigation mode (e.g., so that another interactive element may be selected). Following the discussion of edit mode and FIG. **6**, the description of FIGS. **7** and **8** explains how transitions to edit mode are made from the navigation mode identified above, with respect to FIGS. **3** and **4**.

Turning now then to FIG. **6**, during operation in edit mode **610**, an input **630** is received. If the selected interactive element is only partially visible, entering edit mode will scroll **620** the display until the selected element is completely visible. Decision block **640** determines if input **630** is a direction input. If so, decision block **650** determines if the selected interactive element accepts direction input so that the direction input may be processed **652**. For example, direction input may be accepted in single line textbox **510** and multiple line textbox **520** for moving the cursor position, although up and down direction input would not be accepted by single line textbox **510**. Spinner **550** also may accept direction input, with a left direction input choosing a previous item in the spinner list and a right direction input choosing the next item in the spinner list. If decision block **650** determines that the selected interactive element does not accept direction input, operation transitions or switches to navigation mode **654**.

If input **630** is not a direction input, decision block **660** determines if input **630** is an action input. If not, input **630** is processed as character input **662**. Note that FIG. **6** suggests three basic types of input: direction input, action input, and character input. However, the present invention does not necessarily require dividing all input into any particular number of categories. It is expected, therefore, that alternate embodiments may use additional, fewer, or other categories, depending on the needs or preferences of a particular application. Furthermore, alternate embodiments also may include additional, fewer, or other modes of operation, again depending on the needs or preferences of the particular application.

Processing character input depends on the selected interactive element. Ordinarily, character input has greatest application in entering information into single line textbox **510** and multiple line textbox **520**. However, character input also may be used in finding a particular entry in spinner **550** and picker **560**. For example, if spinner **550** or picker **560** are used to chose a state based on state codes, entering a "W" may immediately move to the end of the list, as opposed to moving through a list item by item or page by page as would likely occur using a direction input.

If decision block **660** determines that input **630** is an action input, decision block **670** considers whether the content being browsed is form content without a submit button. Some form content may fail to provide an explicit submit button, in which case an action input is interpreted as a request to submit the form **672**. In the usual case, however, an action input switches from edit mode back to navigation mode **674**. For spinner **550**, switching from edit mode to navigation mode **674** is somewhat different behavior than occurs in an operating system shell context. More specifically, in an operating system shell context, once a spinner has been selected, an action input ordinarily displays a list of radio button options.

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements. During operation in navigation mode **710**, an action input **720** is received. If decision block **730** determines that the selected interactive element is a link, receiving an action input activates or follows the link **732**. If the selected interactive element is a radio button or checkbox, as determined in decision block **740**, the state of the radio button or checkbox is switched. Radio buttons and checkboxes are both examples of interactive elements capable of representing two states.

Because multiple checkboxes may be chosen, switching the state of a checkbox means either (i) an unchecked checkbox is checked, or (ii) a checked checkbox is unchecked. Radio buttons are slightly more complex to explain since only

one item may be chosen at any given time. Therefore, switching the state of a radio button means that (i) if the radio button was not chosen previously, the radio button will be chosen and any other radio button that may have been chosen previously will no longer be chosen, or (ii) a radio button that was previously chosen remains chosen.

If the selected interactive element is a button, as determined in decision block **750**, the action associated with the button is executed or performed **752** upon receiving an action input **720**. Note that links, radio buttons, checkboxes, and buttons are examples of interactive elements that operate in an intuitive manner without using a dedicated edit mode. For other interactive elements, such single line textbox **510**, multiple line textbox **520**, spinner **550**, and picker **560**, an action input switches to edit mode **754**.

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements. During operation in navigation mode **810**, a character input **820** is received. Character input may include numbers, letters, punctuation, white space, etc. If decision block **830** determines that the interactive element does not accept character input, character input **820** is ignored **836**. Otherwise, character input **820** causes a switch from navigation mode to edit mode **832** and character input **820** is processed **834**. For example, if the selected interactive element is a single line textbox **510** or a multiple line textbox **520**, entering a letter while in navigation mode switches to edit mode and places the letter in the textbox. As indicated earlier, spinner **550** and picker **560** also may use character input to find an item in a long list by advancing to and/or choosing an item that matches the character entered.

FIGS. **9A-9C** show an exemplary method for browsing content according to the present invention. A step for presenting (**910**) at least a portion of content on the display area of a browsing system may include the acts of retrieving (**912**) the content and displaying (**914**) the retrieved content. The content may be retrieved from a local or remote source. A step for controlling the width (**920***a*) of an interactive element may include the act of adjusting the width of the interactive element to the width of available display area on the browsing system if the width of the interactive element exceeds the width of available display area.

A step for determining (**930***a*) an interactive element for selection based on a direction input may include the following acts: an act of starting (**932**) in navigation mode by default when content displays; an act of receiving (**934***a*) a direction input from a direction key, such as four-direction and action key **210** of FIG. **2**; an act of scrolling (**936***a*) the display of the content in the direction of the received direction input if less than all of the content is displayed and no interactive element is at least partially visible; and an act of selecting (**938***a*) an interactive element based on the received direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed content. Note that for English content, an up arrow or left arrow relative to the beginning of the display content does not make sense and therefore is ignored.

A step for indicating (**940***a*) that an interactive element is selected may include the act of highlighting (**944***a*) the interactive element. For example, a selection box may be placed around the interactive element or the visual appearance of the interactive element may be otherwise altered such that a selected interactive element is distinguishable from an interactive element that has not been selected. The present invention does not necessarily require any particular type of highlighting.

A step for changing (**950**) the mode of a browsing system may include an act of switching (**951**) from navigation mode to edit mode upon receiving a character input. For example, when an interactive element such as single line textbox **510**, multiple line textbox **520**, spinner **550**, or picker **560** is selected, receiving a character input switches to edit mode. An act of switching (**953**) from navigation mode to edit mode upon receiving an action input also may be included as part of a step from changing (**950**) the mode of a browsing system. An action input is an explicit indication to switch modes, whereas a character input is an implied indication to switch modes.

Once in edit mode, a step for changing (**950**) the mode of a browsing system may include the act of switching (**955**) from edit mode to navigation mode upon receiving an action input. In other words, an action input may be used both for entering and exiting edit mode. Likewise, an act of switching (**957**) from edit mode to navigation mode upon receiving a direction input also may be included within a step form changing (**950**) the mode of a browsing system. For interactive elements that do not accept a particular direction input, such as an up or down arrow in a single line textbox, receiving the particular direction input switches from edit mode to navigation mode.

Additionally, a step for changing (**950**) the mode of a browsing system may include the act of submitting (**959**) form content upon receiving an action input. For content that does not include a submit button, an action input is associated with submitting the form. After submitting a form, the browsing system switches from edit mode to navigation mode because submitting a form usually causes new content to be displayed by a browsing device and, as described above, the browsing system may default to navigation mode as content initially displays.

Some types of interactive elements may operative intuitively without a dedicated edit mode and therefore switching modes may not be necessary for interacting with those elements. The present invention may include the act of switching (**960**) the state of a selected interactive element such as a radio button or checkbox. Similarly, the act of following (**970**) a selected link upon receiving an action input or executing the action (not shown) associated with a button also are within the scope of the present invention.

If selected interactive element is only partially visible, a step for controlling the width (**920***c*) of an interactive element (see **920***a* of FIG. **9A**) also may include the act of scrolling (**924**) the display area until the interactive element is completely visible. Automatically scrolling the display of content ordinarily occurs when switching from navigation mode to edit mode. Upon switching from navigation mode to edit mode, it becomes clear that a particular interactive element is of interest and should be completely visible, whereas in navigation mode, it may be unclear whether interest is in (i) an interactive element selected as a natural consequence of scrolling displayed content, or (ii) other content that becomes visible as content scrolls.

In addition to the acts described in connection with FIG. **9A**, a step for determining (**930***c*) an interactive element for selection based on a direction input may include other acts, such as an act of receiving (**934***c*) a subsequent direction input. If less than all of the content is display and no interactive element is at least partially visible, determining (**930***c*) an interactive element for selection also may include an act of scrolling (**936***c*) the display of the content in the direction of the subsequent direction input. For a horizontal direction input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**1**) the closest interactive element in the direction of the direction input that is at

substantially the same vertical level. For a vertical direction input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**2**) the interactive element at the next vertical level in the direction of the direction input that is closest in the horizontal direction to a previously selected interactive element. A step for indicating (**940***c*) that an interactive element is selected may further include the acts of removing (**942**) the highlighting from a previously selected interactive element and highlighting (**944***c*) the next selected interactive element (see also **940***a* of FIG. **9**A).

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed and desired to be secured by United States Letters Patent is:

1. In either a wireless telephone or personal digital assistant configured for browsing content received from a content source, a method of browsing content that includes one or more interactive elements, the method comprising:

receiving a direction input;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if only a portion of content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to a beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input; and

selecting an interactive element, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein the interactive element is only partially visible when it is selected.

2. A method as recited in claim **1**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

3. A method as recited in claim **1**, wherein browsing includes a navigation mode and an edit mode, the navigation mode being characterized by selection of interactive elements and the edit mode being characterized by interaction with a single interactive element.

4. A method as recited in claim **3**, wherein the interactive element accepts character input, the method further comprising an act of switching from navigation mode to edit mode upon receiving a character input.

5. A method as recited in claim **3**, further comprising:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

6. A method as recited in claim **1**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input.

7. A method as recited in claim **1**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

8. A method as recited in claim **1**, wherein the interactive element exceeds the width of available browsing system display area, the method further comprising an act of adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area.

9. A method as recited in claim **1**, wherein the interactive element is only partially visible in the browsing system display area, the method further comprising

scrolling the browsing system display area until the interactive element is completely visible.

10. A method as recited in claim **1**, further including: highlighting the interactive element to indicate that the interactive element is selected.

11. A method as recited in claim **1**, wherein prior to receiving the direction input, the method includes displaying only a portion of the content.

12. A method as recited in claim **1**, wherein the selected interactive element is a previously selected interactive element, the method further comprising:

receiving a subsequent direction input that corresponds to scrolling the browsing system display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if no other interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

selecting a next interactive element, the selection being based on the subsequent direction input relative to the previously selected interactive element;

removing highlighting from the previously selected interactive element to indicate the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

13. A method as recited in claim **12**, wherein the subsequent direction input is a horizontal direction input, and wherein the next interactive element is selected based on the next interactive element being (i) a closest interactive element in a direction corresponding to the horizontal direction input, that is (ii) at substantially a same vertical level as the previously selected interactive element.

14. A method as recited in claim **12**, wherein the subsequent direction input is a vertical direction input, and wherein the next interactive element is selected based on the next interactive element being at a next vertical level, from the previously selected interactive element, in a direction corresponding to the vertical direction input.

15. A method as recited in claim **14**, wherein multiple interactive elements are displayed at the next vertical level, the next interactive element being selected based on a determination that the next interactive element comprises an interactive element that is closest in horizontal direction to a beginning of the previously selected interactive element.

16. The method of claim **15**, wherein upon determining that the horizontal distance between the beginning portion of the next interactive element and the beginning portion of the previous selected interactive element is a same distance as the beginning portion of another one of the multiple interactive elements from the beginning portion of the previously selected interactive element, the method further includes:

selecting the next interactive element upon determining that the next interactive element is presented prior to said another one of the multiple interactive elements.

17. The method of claim **12**, wherein the previously selected interactive element spans a plurality of vertical levels

and wherein at least one interactive element is located in each of the different vertical levels and wherein selection of the next interactive element is based on a determination that the next interactive element is located on a highest one of the plurality of vertical levels.

18. A computer program product for use in either a wireless telephone or personal digital assistant configured for browsing content received from a content source, the computer program product comprising one or more computer readable media having computer-executable instructions for implementing a method of browsing content that includes one or more interactive elements, the method comprising:

receiving a direction input;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if only a portion of content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to a beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input; and

selecting an interactive element, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein the interactive element is only partially visible when it is selected.

19. A computer program product as recited in claim **18**, wherein the method further includes highlighting the interactive element to indicate that the interactive element is selected.

20. A computer program product as recited in claim **18**, wherein prior to receiving the direction input, the method includes displaying only a portion of the content.

21. In either a wireless telephone or personal digital assistant configured for browsing content received from a content source, a method of browsing content that includes one or more interactive elements, the method comprising:

receiving a direction input;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if only a portion of content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to a beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input; and

selecting an interactive element based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein the interactive element is entirely visible when it is selected.

22. A method as recited in claim **21**, wherein the selection of the interactive element is performed without utilizing tab order navigation.

23. A method as recited in claim **21**, further including: highlighting the interactive element to indicate that the interactive element is selected.

24. A method as recited in claim **21**, wherein prior to receiving the direction input, the method includes displaying only a portion of the content.

25. A computer program product for use in either a wireless telephone or personal digital assistant configured for browsing content received from a content source, the computer program product comprising one or more computer readable media having computer-executable instructions for implementing a method of browsing content that includes one or more interactive elements, the method comprising:

receiving a direction input;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if only a portion of content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to a beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input; and

selecting an interactive element based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein the interactive element is entirely visible when it is selected.

26. A computer program product as recited in claim **25**, wherein the selection of the interactive element is performed without utilizing tab order navigation.

27. A computer program product as recited in claim **25**, wherein the method further includes highlighting the interactive element to indicate that the interactive element is selected.

28. A computer program product as recited in claim **25**, wherein prior to receiving the direction input, the method includes displaying only a portion of the content.

* * * * *

US006822664B2

(12) **United States Patent**
Vale

(10) **Patent No.: US 6,822,664 B2**
(45) **Date of Patent: Nov. 23, 2004**

(54) **BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM**

(75) Inventor: **Peter O. Vale**, Seattle, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 574 days.

(21) Appl. No.: **09/861,327**

(22) Filed: **May 18, 2001**

(65) **Prior Publication Data**

US 2002/0041291 A1 Apr. 11, 2002

Related U.S. Application Data

(60) Provisional application No. 60/239,600, filed on Oct. 11, 2000.

(51) **Int. Cl.**[7] **G06F 3/14**

(52) **U.S. Cl.** **345/864**; 345/856; 345/857; 345/862; 345/863

(58) **Field of Search** 345/864, 856, 345/857, 862, 863

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,128,012	A *	10/2000	Seidensticker et al.	345/685
6,310,634	B1 *	10/2001	Bodnar et al.	345/854
2001/0022839	A1 *	9/2001	Ishigaki	379/433.04
2002/0070980	A1 *	6/2002	Le et al.	345/828
2002/0112237	A1 *	8/2002	Kelts	725/39

OTHER PUBLICATIONS

Motorola, Digital Wireless Telephone, Model 120c, the User Guide, 2001.*

Marran, N.L., *Over–The–Air Subscriber Device Management Using CDMA Data and WAP*, Virginia Tech, 1999, pp. 165–174.

Ming, Tham, and Chuan, Tan Kay, *Challenges in Designing User Interfaces for Handheld Communication Devices: A Case Study*, Lawrence Erlbaum Associates, 1999, pp. 808–812.

Fox, A.; Goldberg, I.; Gribble, S.D.; Lee, D.C.; Polito, A.; and Brewer, E.A., *Experience With Top Gun Wingman: A Proxy–Based Graphical Web Browser for the 3Com PalmPilot*, University of California at Berkeley, 1998, pp. 407–424.

Gessler, Stefan and Kotulla, Andreas, *PDAs as mobile WWW browsers*, Germany 1995, pp. 53–59.

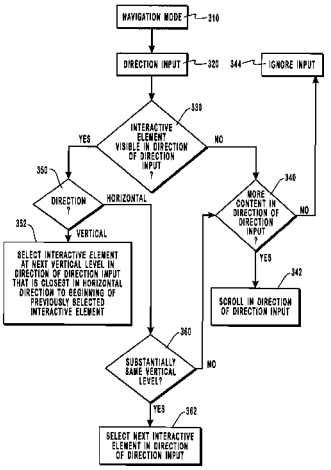
* cited by examiner

Primary Examiner—Kristine Kincaid
Assistant Examiner—Thomas T. Nguyen
(74) *Attorney, Agent, or Firm*—Workman Nydegger

(57) **ABSTRACT**

Methods, system, and computer program products for browsing content with a display area and input system that may be limited in comparison to more traditional browsing systems. Movement between and selection of interactive elements generally occurs in a navigation mode, whereas interaction with a single interactive element generally occurs in an edit mode. In navigation mode, a direction input selects the next interactive element in the direction indicated. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls. Switching between navigation mode and edit mode is based on the input received, in view of the input supported, by a particular interactive element. Interactive elements may be limited to the width of available display area.

65 Claims, 12 Drawing Sheets



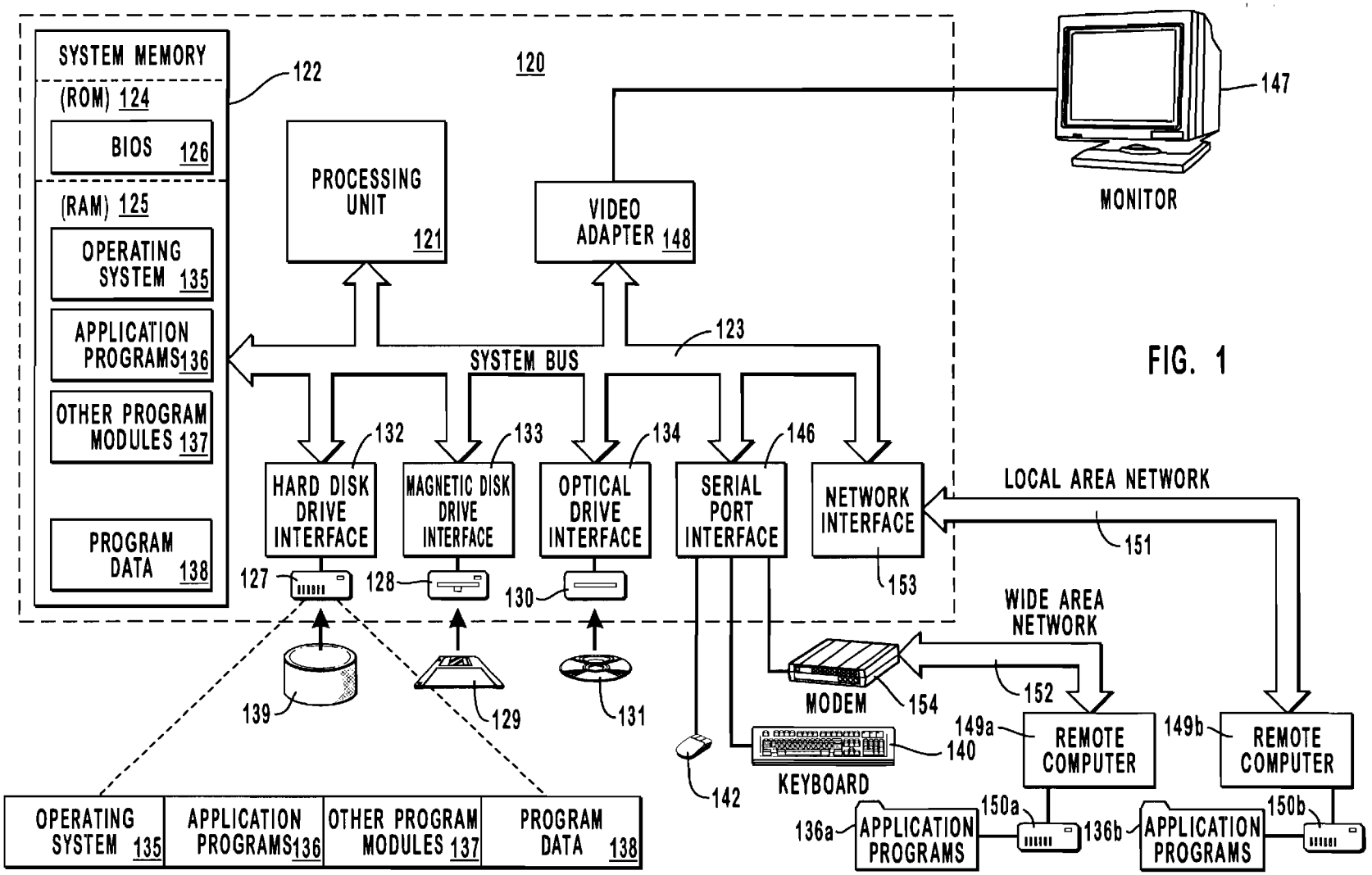

SYSTEM MEMORY
(ROM) 124
BIOS 126
(RAM) 125
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
122
120
PROCESSING UNIT 121
VIDEO ADAPTER 148
147
MONITOR
123
SYSTEM BUS
132
133
134
146
HARD DISK DRIVE INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
LOCAL AREA NETWORK
151
153
127
128
130
WIDE AREA NETWORK
139
129
131
MODEM
154
152
149a
REMOTE COMPUTER
149b
REMOTE COMPUTER
140
KEYBOARD
142
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
136a
APPLICATION PROGRAMS
150a
136b
APPLICATION PROGRAMS
150b

FIG. 1

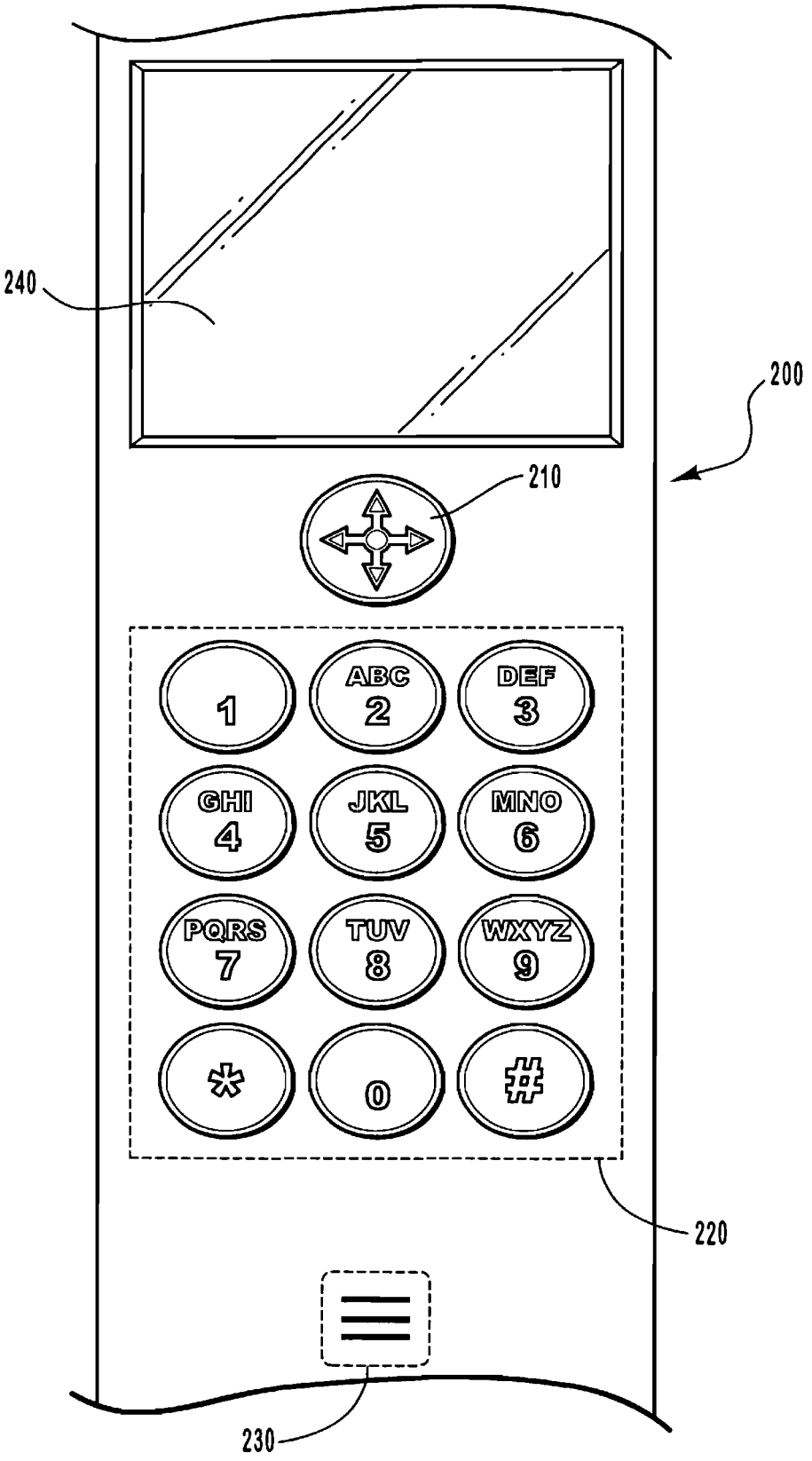

240
200
210
1
ABC 2
DEF 3
GHI 4
JKL 5
MNO 6
PQRS 7
TUV 8
WXYZ 9
*
0
#
220
230

FIG. 2

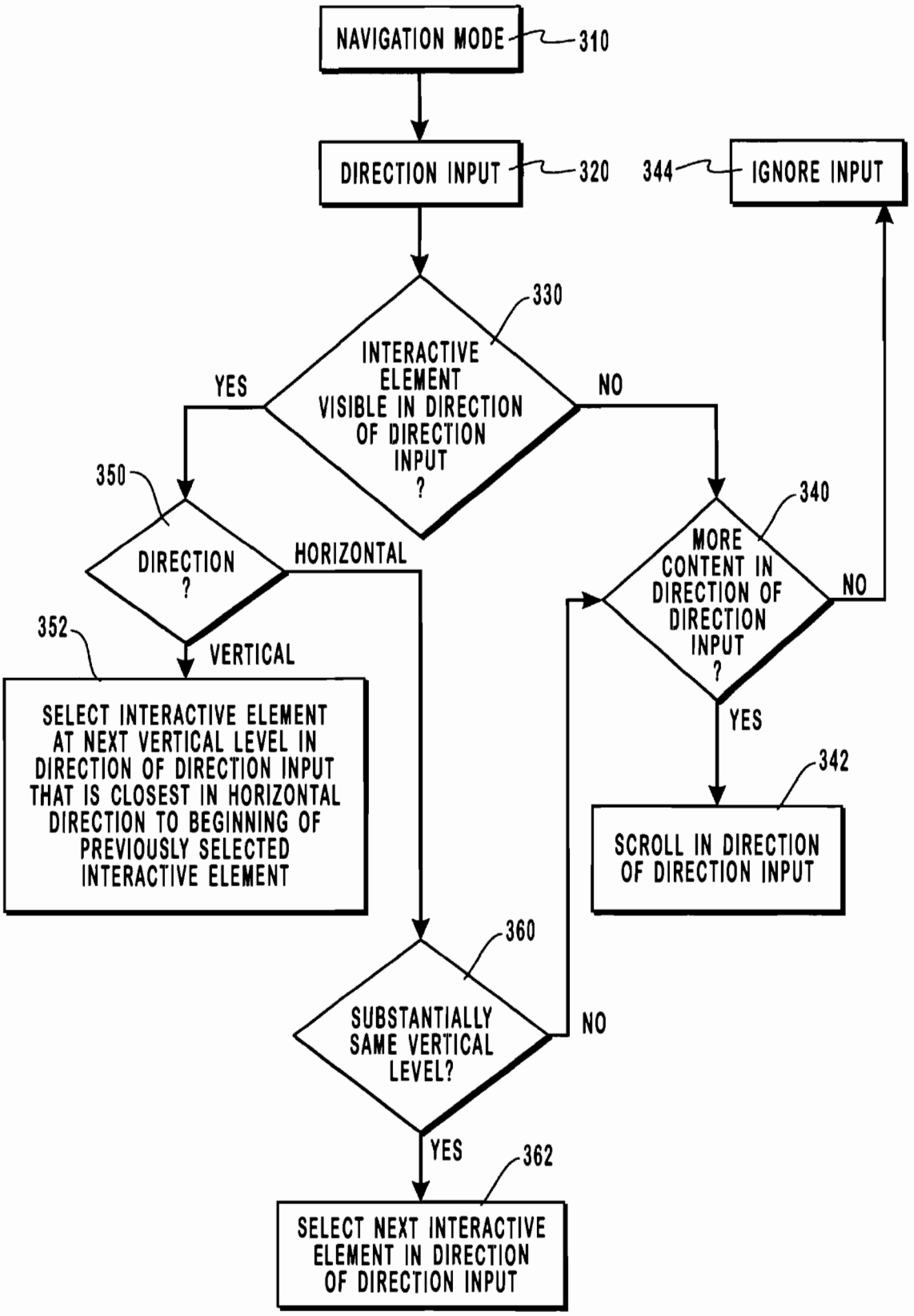

NAVIGATION MODE
310
DIRECTION INPUT
320
344
IGNORE INPUT
330
INTERACTIVE ELEMENT VISIBLE IN DIRECTION OF DIRECTION INPUT ?
YES
NO
350
DIRECTION ?
HORIZONTAL
VERTICAL
352
SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL IN DIRECTION OF DIRECTION INPUT THAT IS CLOSEST IN HORIZONTAL DIRECTION TO BEGINNING OF PREVIOUSLY SELECTED INTERACTIVE ELEMENT
340
MORE CONTENT IN DIRECTION OF DIRECTION INPUT ?
NO
YES
342
SCROLL IN DIRECTION OF DIRECTION INPUT
360
SUBSTANTIALLY SAME VERTICAL LEVEL?
NO
YES
362
SELECT NEXT INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT

FIG. 3

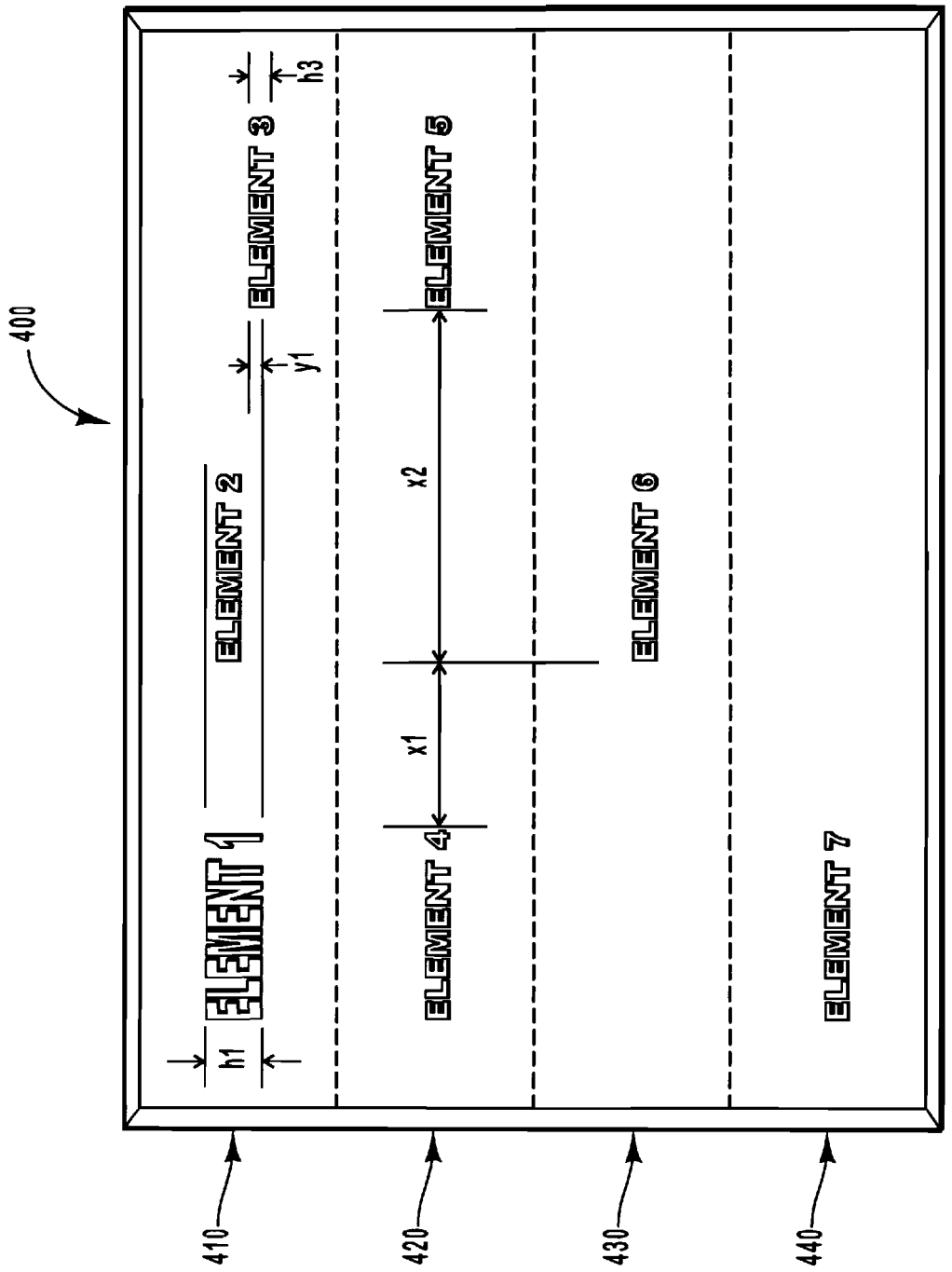

400
ELEMENT 1
ELEMENT 2
ELEMENT 3
ELEMENT 4
ELEMENT 5
ELEMENT 6
ELEMENT 7
h1
h3
y1
x1
x2
410
420
430
440

FIG. 4

FIG. 5A

FIG. 5B

FIG. 5C

FIG. 5D

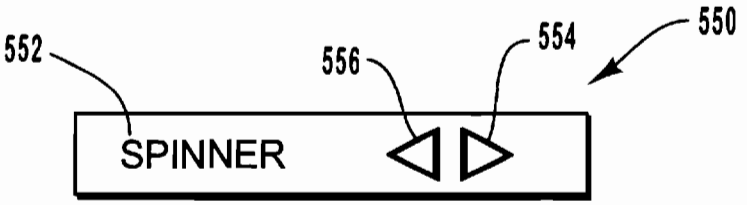


FIG. 5E

FIG. 5F

FIG. 5G

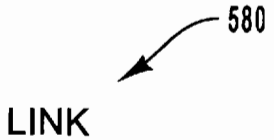


FIG. 5H

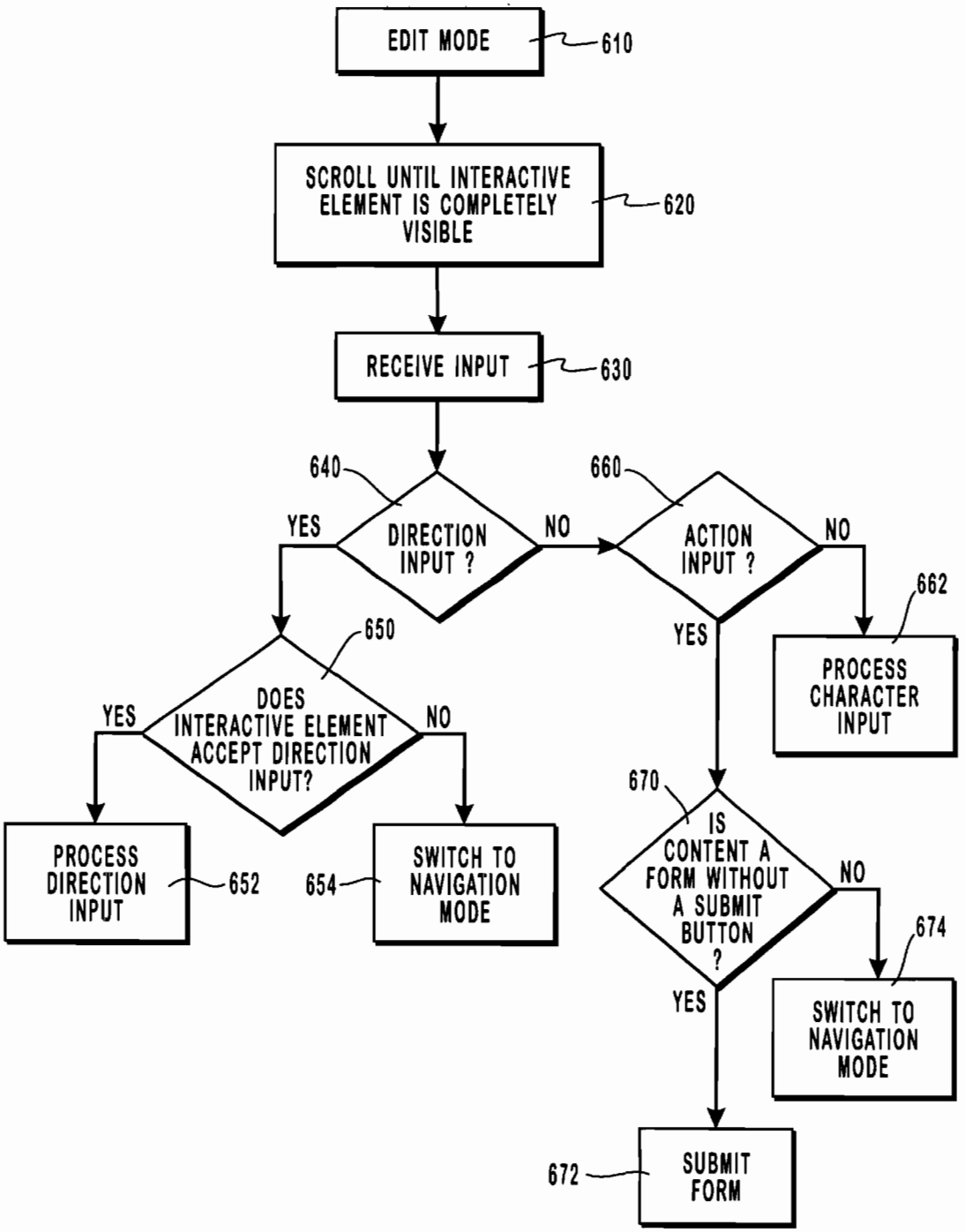

EDIT MODE
610
SCROLL UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
620
RECEIVE INPUT
630
640
DIRECTION INPUT ?
YES
NO
660
ACTION INPUT ?
NO
662
PROCESS CHARACTER INPUT
YES
650
DOES INTERACTIVE ELEMENT ACCEPT DIRECTION INPUT?
YES
NO
PROCESS DIRECTION INPUT
652
654
SWITCH TO NAVIGATION MODE
670
IS CONTENT A FORM WITHOUT A SUBMIT BUTTON ?
NO
674
SWITCH TO NAVIGATION MODE
YES
672
SUBMIT FORM

FIG. 6

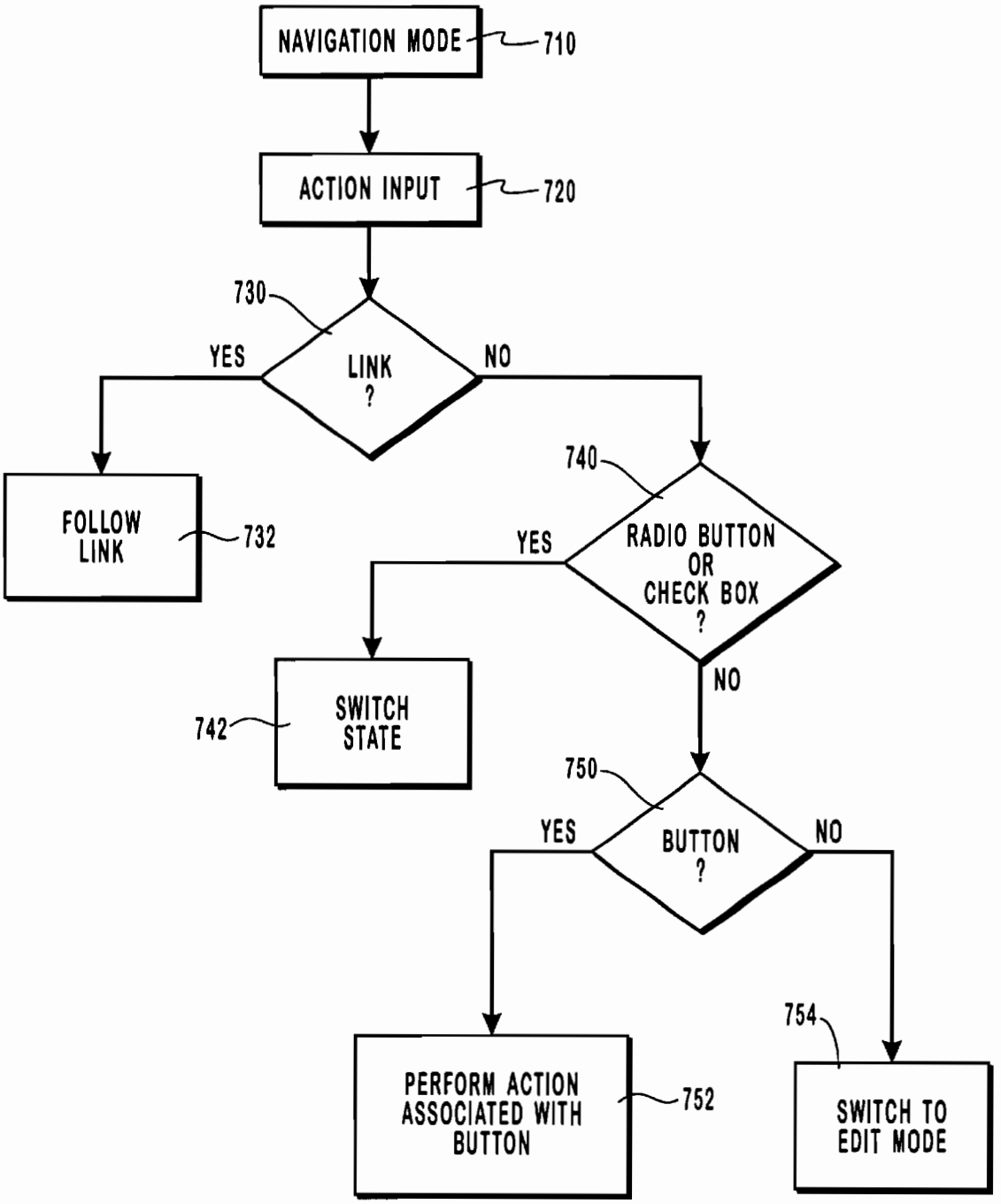

NAVIGATION MODE
710
ACTION INPUT
720
730
LINK ?
YES
NO
FOLLOW LINK
732
740
RADIO BUTTON OR CHECK BOX ?
YES
NO
SWITCH STATE
742
750
BUTTON ?
YES
NO
PERFORM ACTION ASSOCIATED WITH BUTTON
752
754
SWITCH TO EDIT MODE

FIG. 7

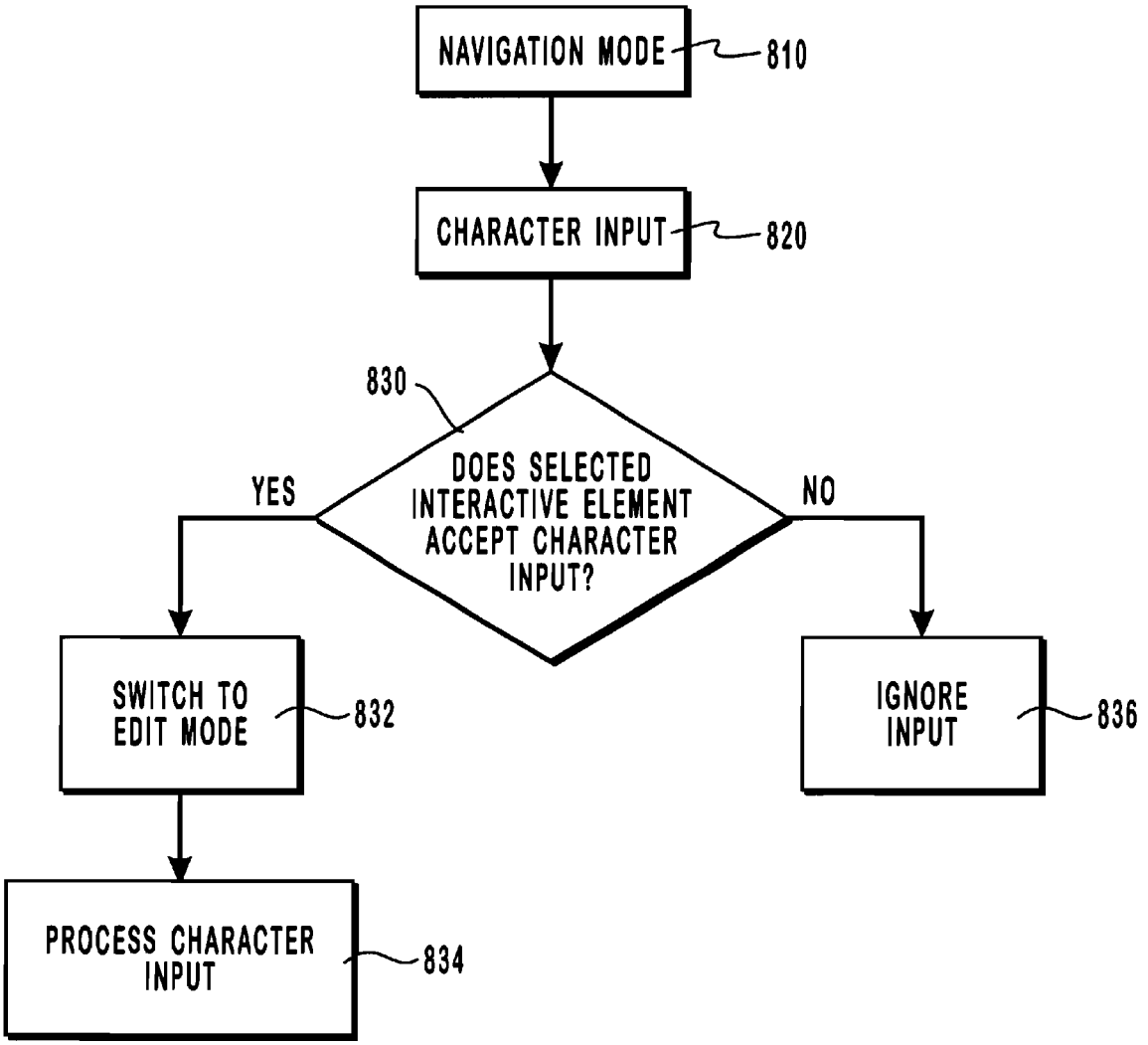

NAVIGATION MODE
810
CHARACTER INPUT
820
830
DOES SELECTED INTERACTIVE ELEMENT ACCEPT CHARACTER INPUT?
YES
NO
SWITCH TO EDIT MODE
832
IGNORE INPUT
836
PROCESS CHARACTER INPUT
834

FIG. 8

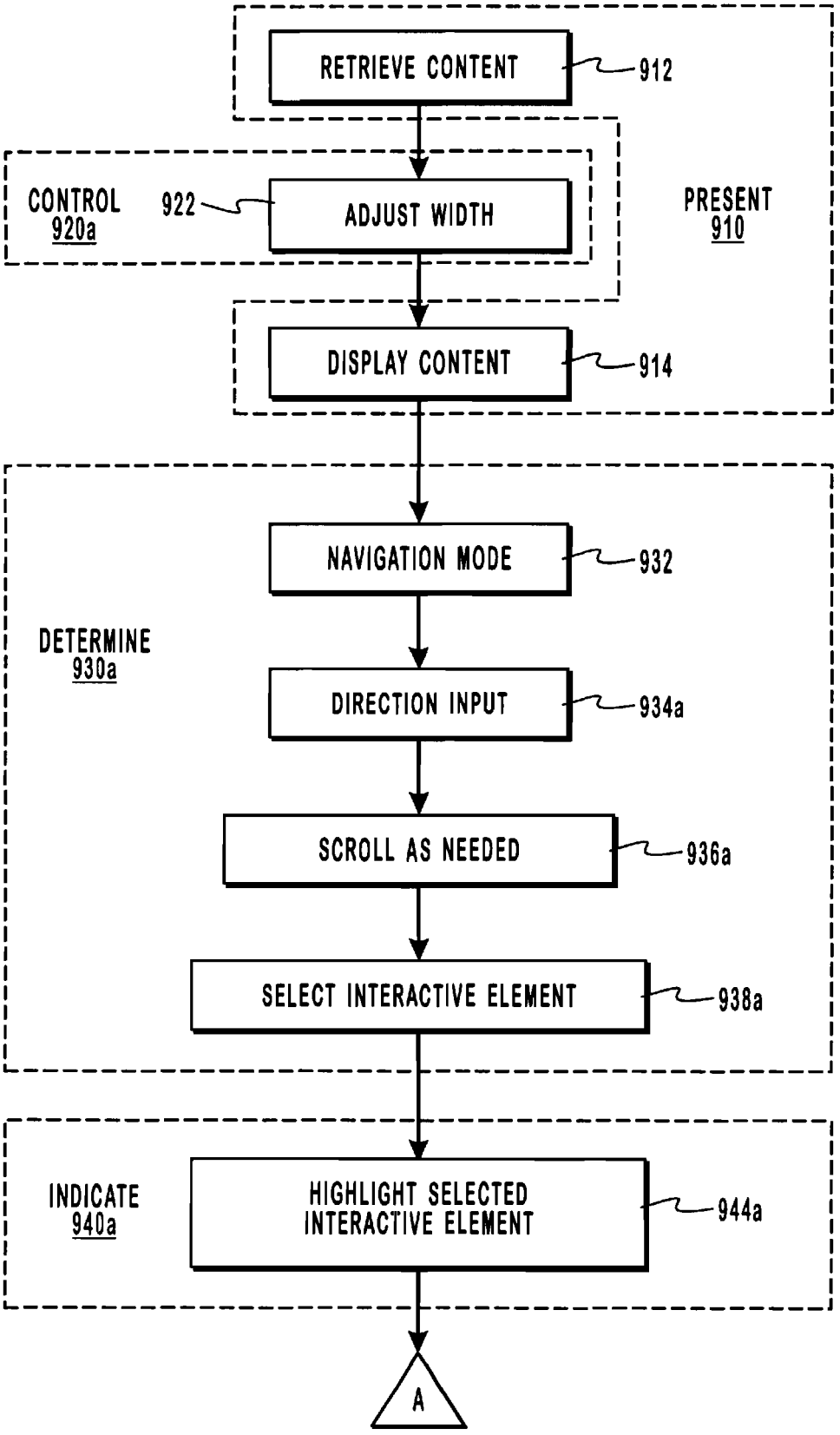

RETRIEVE CONTENT
912
CONTROL
920a
922
ADJUST WIDTH
PRESENT
910
DISPLAY CONTENT
914
NAVIGATION MODE
932
DETERMINE
930a
DIRECTION INPUT
934a
SCROLL AS NEEDED
936a
SELECT INTERACTIVE ELEMENT
938a
INDICATE
940a
HIGHLIGHT SELECTED
INTERACTIVE ELEMENT
944a
A

FIG. 9A

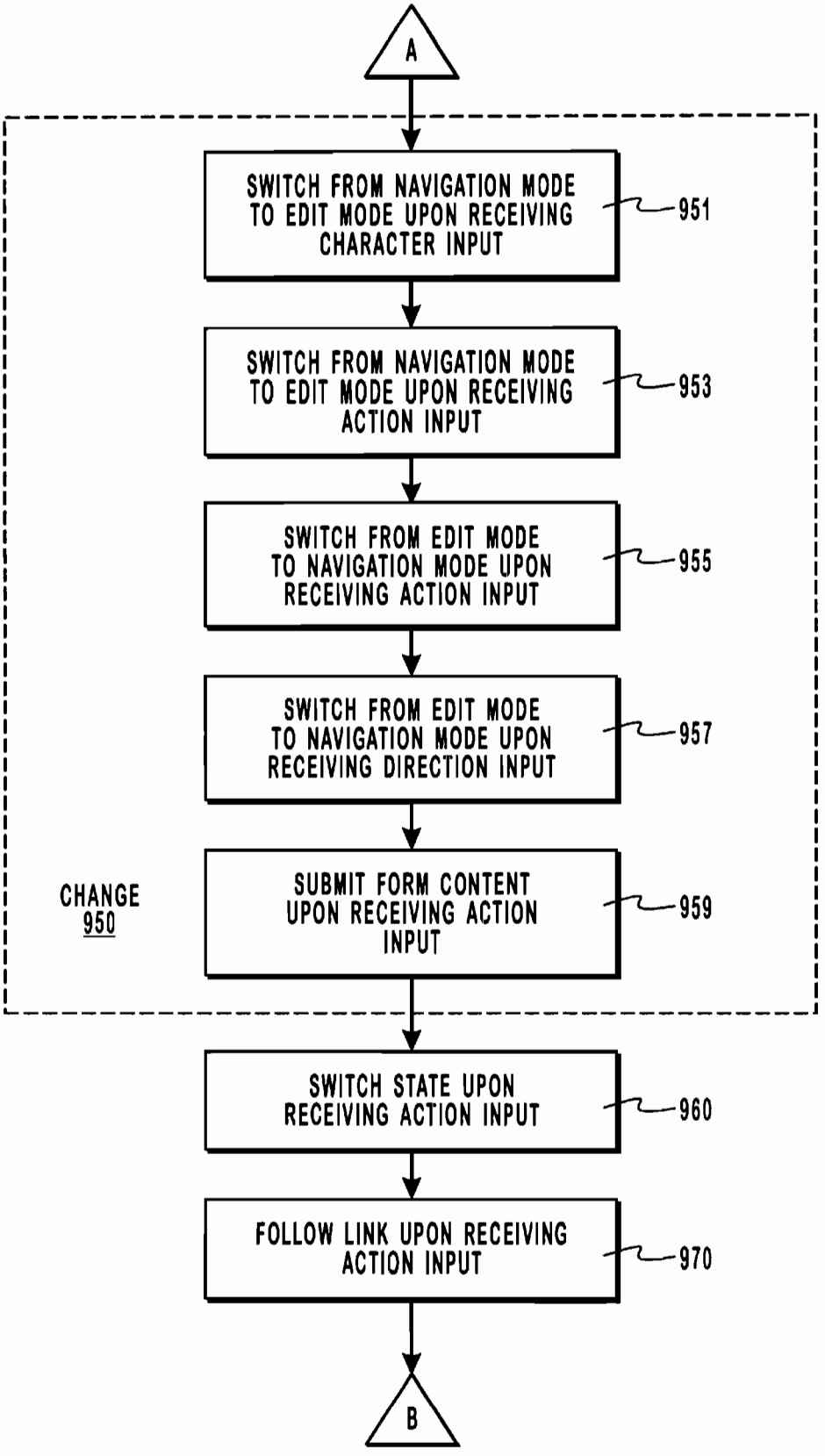

A
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING CHARACTER INPUT
951
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING ACTION INPUT
953
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING ACTION INPUT
955
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING DIRECTION INPUT
957
SUBMIT FORM CONTENT UPON RECEIVING ACTION INPUT
959
CHANGE 950
SWITCH STATE UPON RECEIVING ACTION INPUT
960
FOLLOW LINK UPON RECEIVING ACTION INPUT
970
B

FIG. 9B

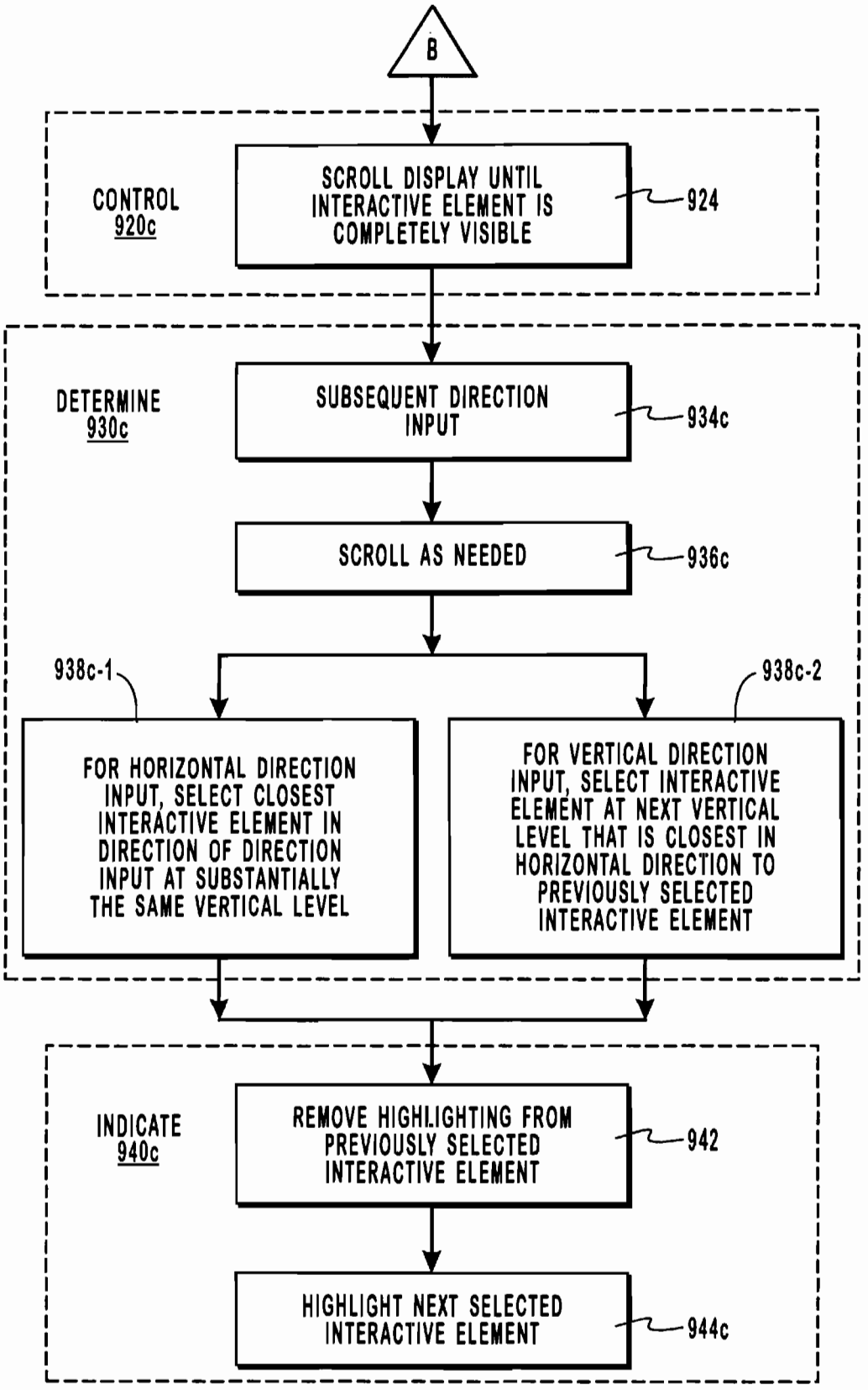

B
CONTROL
920c
SCROLL DISPLAY UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
924
DETERMINE
930c
SUBSEQUENT DIRECTION INPUT
934c
SCROLL AS NEEDED
936c
938c-1
938c-2
FOR HORIZONTAL DIRECTION INPUT, SELECT CLOSEST INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT AT SUBSTANTIALLY THE SAME VERTICAL LEVEL
FOR VERTICAL DIRECTION INPUT, SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL THAT IS CLOSEST IN HORIZONTAL DIRECTION TO PREVIOUSLY SELECTED INTERACTIVE ELEMENT
INDICATE
940c
REMOVE HIGHLIGHTING FROM PREVIOUSLY SELECTED INTERACTIVE ELEMENT
942
HIGHLIGHT NEXT SELECTED INTERACTIVE ELEMENT
944c

FIG. 9C

BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/239,600, entitled, "BROWSER NAVIGATION FOR DEVICES WITH LIMITED INPUT MECHANISMS," filed Oct. 11, 2000, which is hereby incorporated by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates to browsing electronic content. More specifically, the present invention relates to methods, systems, and computer program products for browsing content that includes interactive elements using a computerized system with a display area and input system that may be somewhat limited in comparison to the pointing devices and displays typically found in more traditional browsing systems.

2. Background and Related Art

Content typically includes interactive elements, such as links and form controls. Activating or following a link causes the content that is associated with the link to be requested and displayed. Selecting a form control allows for interaction with the form control. Traditional browsing systems generally include a keyboard and a pointing device such as a mouse, for activating links and interacting with form controls. Tab order navigation is possible, but may not follow an order expected by the user, especially if scrolling is required to view all of the content.

In traditional browsing systems, a user activates a link or selects a form control by simply placing a mouse pointer over the interactive element and pressing a mouse button. With each mouse press, a user may follow a link, select a text field so that text may be entered from a keyboard, toggle a radio button or checkbox, choose one or more items from a list, or cause the action associated with a button to be executed. The mouse also is used in scrolling the display area, as necessary. Nevertheless, content often is authored to minimize scrolling the display of traditional browsing systems, particularly in the horizontal direction.

Browsing systems with limited input systems and display areas, however, such as a phone having a numeric keypad, a directional control, and an action key, may make it difficult to select and interact with content designed for more traditional browsing systems that make use of pointing devices and have larger display areas. For example, without a pointing device, how are links activated and how are form controls selected? The direction control is a natural choice for scrolling because this operation is similar to many traditional browsing systems. (When no interactive element is selected, arrow keys usually are used for scrolling.) But, without a mouse, selecting individual interactive elements presents a significant challenge.

Tab order navigation does not provide an adequate solution because tab order generally follows the order of interactive elements in the content as authored or written, rather than the order of interactive elements in the content as displayed. Thus, in some situations, tab order moves horizontally, and in other situations, tab order moves vertically. For example, content that includes a table often will have a vertical tab order within individual table cells, but a horizontal tab order from cell to cell. Content outside of a table usually has a horizontal tab order. Because users

generally are unaware of whether content includes a table or not, tab order may appear completely arbitrary, moving horizontally one time and vertically the next.

Therefore, when browsing content that includes interactive elements, methods, systems, and computer program products are needed for computerized systems that may have limited display areas and input systems, as compared to the pointing devices and displays typically found in more traditional browsing systems. Furthermore, certain interactive elements may be more intuitive in a browsing context, if those interactive elements operate somewhat differently from how they might function in an operating system shell environment.

SUMMARY OF THE INVENTION

The present invention provides a navigation mode and an edit mode for browsing content with a computerized system that may include a somewhat limited display area and/or input system. Navigation mode generally includes movement between and selection of interactive elements, whereas the edit mode generally includes interaction with a single interactive element. In navigation mode, pressing a direction key selects the next interactive element in the direction indicated by the direction key (e.g., up, down, left, right). When moving horizontally, an interactive element is in the direction indicated by the direction control if the interactive element is at substantially the same vertical level. For example, if a later element overlaps a previous element on a given vertical level by any amount, the two elements are considered to be at substantially the same vertical level. Vertical movement is to an interactive element at the next vertical level in the direction indicated by the direction control. If multiple interactive elements lie at the next vertical level, the one closest in the horizontal direction to the beginning of the current interactive element is selected.

To indicate selection, an interactive element is highlighted, such as by placing a selection box around the element. The interactive element remains selected until it is no longer visible (i.e., it has scrolled off the display area) or the next interactive element becomes at least partially visible and is selected. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls in the direction indicated by the direction control.

Switching from navigation mode to edit mode may be accomplished in several ways. For example, once an interactive element allowing character entry is selected, typing a character on the keypad automatically switches from navigation mode to edit mode. Similar to a mouse click, pressing the action button after an interactive element has been selected also switches to edit mode. Where interactive elements only require a mouse click to function in traditional browsing systems, such as links, checkboxes, radio buttons, other buttons, and the like, pressing action uses the selected control (i.e., follows the link, checks or unchecks a checkbox, chooses a radio button, performs the action associated with the button, etc.) rather than switching to edit mode. In edit mode, pressing the action button switches back to navigation mode. If a particular direction key is not allowed in edit mode, pressing the direction key also will exit edit mode. For forms that do not include a submit button on the form, pressing the action key will submit the form, rather than switching to navigation mode.

Certain interactive elements may be limited to the width of the display area that is available for displaying content so that the entire element can be visible at one time. Therefore,

the width of an interactive element that exceeds the width of available display area may be adjusted to be less than or equal to the width of available display area. If a selected interactive element is only partially visible, switching into edit mode scrolls the control into full view.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered as limiting its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** illustrates an exemplary system that provides a suitable operating environment for the present invention;

FIG. **2** shows a portion of a wireless telephone;

FIG. **3** is a flow diagram that corresponds to receiving a direction input while navigating between interactive elements;

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation;

FIGS. **5**A–**5**H illustrate various interactive elements;

FIG. **6** is a flow diagram that corresponds to receiving input while editing interactive elements;

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements;

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements; and

FIGS. **9**A–**9**C show an exemplary method for browsing content according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention extends to methods, systems, and computer program products for browsing content that includes interactive elements using a browsing system with a display area and input system that may be limited in comparison to the pointing devices and displays typically found in more traditional browsing systems. As used in this application, the term "browsing system" should be interpreted broadly to encompass any computerized system for locating and presenting content, including text, images, audio, video, computer instructions, and the like. With the popularity of the World Wide Web ("Web"), content frequently is formatted using hypertext markup language ("HTML") and computer instructions often are embedded in content using Javascript. Both HTML and Javascript allow for the creation of interactive elements within content.

Those of skill in the art, however, will recognize that a wide variety of markup and scripting languages exist. In particular, extensible markup language ("XML") is becoming increasingly popular because it allows for user-defined extensions to the language. Note also that content may be converted from one language to another. Furthermore, it is anticipated that additional formats, languages, technology and/or standards for authoring content with interactive elements will become available in the future. The present invention, therefore, does not impose any requirements on how content with interactive elements is authored, whether based on current or future technology. Thus, any reference to HTML and/or Javascript, either explicit or implied, should be interpreted as exemplary of aspects or embodiments of the present invention and not as limiting its scope.

Those skilled in the art also will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, mobile/hand-held devices, such as personal digital assistants ("PDAs") and wireless telephones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. The embodiments of the present invention may comprise a special purpose or general purpose computer including various computer hardware, as discussed in greater detail below.

Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that may be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, flash memory, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such a connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media. Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions.

FIG. **1** and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention may be described in the general context of computer-executable instructions, such as program modules, being executed by computers in network environments. Generally, program modules include routines, programs, objects, components, data structures,

etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represent examples of corresponding acts for implementing the functions described in such steps.

With reference to FIG. **1**, an exemplary system for implementing the invention comprises a general purpose computing device in the form of a generic computer **20**, including a processing unit **21**, a system memory **22**, and a system bus **23** that couples various system components including the system memory **22** to the processing unit **21**. Although some components of computer **20**, such as monitor **47**, keyboard **40**, and mouse **42**, may seem specific to a conventional computer, those of skill in the art will recognize that analogous components may be found in other computing devices. For example, wireless telephones often include an LCD or plasma display area, a numeric keypad, and one or more navigation buttons. Therefore, any component described with reference to generic computer **20** should be interpreted broadly to encompass analogous components that are appropriate for and consistent with a particular implementation or embodiment of the present invention. In some implementations, a component may be connected only intermittently or may be missing entirely.

The system bus **23** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) **24** and random access memory (RAM) **25**. A basic input/output system (BIOS) **26**, containing the basic routines that help transfer information between elements within the computer **20**, such as during start-up, may be stored in ROM **24**.

The computer **20** may also include a magnetic hard disk drive **27** for reading from and writing to a magnetic hard disk **39**, a magnetic disk drive **28** for reading from or writing to a removable magnetic disk **29**, and an optical disk drive **30** for reading from or writing to removable optical disk **31** such as a CD-ROM or other optical media. The magnetic hard disk drive **27**, magnetic disk drive **28**, and optical disk drive **30** are connected to the system bus **23** by a hard disk drive interface **32**, a magnetic disk drive-interface **33**, and an optical drive interface **34**, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-executable instructions, data structures, program modules and other data for the computer **20**. Although the exemplary environment described herein employs a magnetic hard disk **39**, a removable magnetic disk **29** and a removable optical disk **31**, other types of computer readable media for storing data can be used, including magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs, and the like. Note that decreasing form factors are making it practical to use at least some of the foregoing components with mobile devices. Furthermore, it is anticipated that future technological advances with respect to size, power consumption, and the like, will lead to an increased selection of storage options.

Program code means comprising one or more program modules may be stored on the hard disk **39**, magnetic disk **29**, optical disk **31**, ROM **24** or RAM **25**, including an operating system **35**, one or more application programs **36**, other program modules **37**, and program data **38**. A user may enter commands and information into the computer **20** through keyboard **40**, pointing device **42**, or other input devices (not shown), such as a numeric keypad, directional buttons, pressure-sensitive software keyboard, microphone, joy stick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit **21** through a serial port interface **46** coupled to system bus **23**. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor **47** or another display device, such as an LCD or gas plasma display, is also connected to system bus **23** via an interface, such as video adapter **48**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer **20** may operate in a networked environment using logical connections to one or more remote computers, such as remote computers **49***a* and **49***b*. Remote computers **49***a* and **49***b* may each be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically include many or all of the elements described above relative to the computer **20**, although only memory storage devices **50***a* and **50***b* and their associated application programs **36***a* and **36***b* have been illustrated in FIG. **1**. The logical connections depicted in FIG. **1** include a local area network (LAN) **51** and a wide area network (WAN) **52** that are presented here by way of example and not limitation. Such networking environments are commonplace in office wide or enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer **20** is connected to the local network **51** through a network interface or adapter **53**. When used in a WAN networking environment, the computer **20** may include a modem **54**, a wireless link, or other means for establishing communications over the wide area network **52**, such as the Internet. The modem **54**, which may be internal or external, is connected to the system bus **23** via the serial port interface **46**. In a networked environment, program modules depicted relative to the computer **20**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing communications over wide area network **52** may be used.

FIG. **2** shows a portion of a wireless telephone. A wireless telephone is merely one example of a browsing system with limited display and input capabilities. Typically, PDAs and other handheld devices also have limited displays and input systems. The present invention, however, is not necessarily limited to any particular hardware or device, handheld or otherwise. Nevertheless, some benefits provided by the present invention may be more pronounced where displays and/or input systems are less robust than corresponding displays and/or input systems found in traditional browsing systems.

Four-direction and action key **210** is an example of both a navigation key generating direction input and an action key providing action input. Depressing key **210** at any one of the four arrows generates a direction input corresponding to the direction of the arrow. An action input is generated by depressing the center of key **210**. The center of key **210** may be a separate button (not shown) or may be integral with the navigation arrows such that depressing the center generates simultaneous, but conflicting direction input. In other words, simultaneous up and down arrows or simultaneous left and right arrows are interpreted as an action input. Typically, an action input corresponds to pressing an enter key on a keyboard, but other actions are not precluded, as circumstances may warrant.

A user may enter characters, such as numbers, letters, punctuation, etc., with keypad **220**. Audio input **230** is the mobile telephone's mouthpiece. Display area **240** displays content received while browsing. Note however, that all of display area **240** may not be available for displaying content. For example, portions of display area **240** may be used for titles, menus, switching between tasks, etc. Therefore, available display area generally refers to the portion of display area **240** that is devoted to displaying content, and may represent all or less than all of display area **240**.

As noted previously, the present invention provides for a navigation mode and an edit mode. Edit mode generally is characterized by interaction with a single interactive element and is described more fully in connection with FIG. **6** and the interactive elements shown in FIGS. **5**A–**5**H. In contrast, navigation mode generally is characterized by movement between and selection of interactive elements (i.e., changing focus from one interactive element to another) and is described more fully in connection with FIGS. **3** and **4**. Transitions from navigation mode to edit mode and interactive elements that operate in an intuitive manner without using a dedicated edit mode are covered in the description of FIGS. **7** and **8**.

Turning first then to FIG. **3**, during operation in navigation mode **310**, a direction input **320** is received. Decision block **330** determines if an interactive element is at least partially visible in the direction of direction input **320**, either relative to the beginning of the content if no interactive element has been selected or relative to an interactive element that is selected currently. If no interactive element is visible in the direction of the direction input, decision block **340** determines if more content is available in the direction of the direction input. If more content is available in the direction of the direction input, the display scrolls **342** in the direction of the direction input; otherwise, the direction input is ignored **344**.

Returning to decision block **330**, if an interactive element is visible in the direction of direction input **320**, selecting the next interactive element depends on the direction of direction input **320**, unless no interactive element has been selected previously, wherein the interactive element closest to the beginning of the content is selected (not shown). For horizontal input, decision block **360** determines if the visible interactive element lies at substantially the same vertical level as the interactive element that is selected currently. (The meaning of substantially the same vertical level will be described in more detail below, with respect to FIG. **4**.) If substantially at the same level, the interactive element in the direction of the direction input is selected **362**. If the visible interactive element does not lie at the substantially the same vertical level, operation continues with decision block **340**, as described above. For vertical direction input, the interactive element at the next vertical level in the direction of direction input **320** that is closest in the horizontal direction to the beginning of the previously selected interactive element is selected **352**.

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation. The display **400** of content includes vertical levels **410**, **420**, **430**, and **440**. Note that at vertical level **410**, Element 1, Element 2, and Element 3 do not display at exactly the same vertical coordinates. Nevertheless, Element 1, Element 2, and Element 3 lie at substantially the same vertical level. By allowing for some variation in the vertical display coordinates for interactive elements, navigation is more intuitive.

In particular, the height of Element 1 is h1 and the height of Element 3 is h3. The distance y1 is the amount that Element 3 overlaps with Element 1. Alternatively, y1 may represent the amount of vertical separation between interactive elements rather than the amount of overlap. Note that the present invention does not necessarily limit y1 to any particular dimension or calculation. Because y1 reflects the expectations of users, it is possible for y1 to take on a wide range of values, as may be suitable for a particular embodiment or implementation. However, in at least some circumstances, y1 is a portion of h3, the height of Element 3, indicating that Element 3 partially overlaps Element 1. (Although not shown, note also that it may be possible for a single element to span and be reachable from multiple vertical levels. When navigating horizontally from an element spanning multiple vertical levels, the next element is selected from the top most spanned vertical level where an interactive element is visible.) The height of Element 2 is not shown because it overlaps completely with Element 1 and therefore is clearly at the same vertical level as Element 1.

Initially no interactive element is selected. As indicated with respect to FIG. **3**, a direction input that is not in the direction of an interactive element that is at least partially visible or in a direction that permits scrolling will be ignored (see block **344**). Thus, direction input up or to the left will be ignored. Because initially no interactive element is selected, however, a down direction input or a right direction input will select Element 1 (i.e., the first interactive element relative to the beginning of the content). Once selected, an interactive element is highlighted, such as by drawing a dashed box around the element to indicate that the selected interactive element has focus. The present invention does not necessarily limit the type of highlighting used to indicate selection. It is only relevant for some form of visual cue to occur that is specific to the selected interactive element.

Left and right direction input will select interactive elements in numerical order, either ascending for right direction input or descending for left direction input. Up and down direction input is somewhat more complicated. Beginning with Element 1, down direction input selects interactive elements in the following order: Element 1, Element 4, Element 6, Element 7. Beginning with Element 7, up direction input selects interactive elements in reverse order: Element 7, Element 6, Element 4, Element 1.

Moving from vertical level **430** to vertical level **420** may include some ambiguity because vertical level **420** includes multiple interactive elements. Selecting between Element 4 and Element 5 depends on the horizontal distances labeled x1 and x2. The distance x1 represents the horizontal distance from the beginning of Element 6 (the currently selected interactive element when moving from vertical level **430** to vertical level **420**) to the nearest portion of Element 4. Likewise, the distance x2 represents the horizontal distance from the beginning of Element 6 to the nearest portion of Element 5. When selection moves in the vertical direction and more than one interactive element lies at a vertical level, the interactive element closest in the horizontal direction to the beginning of the previously selected interactive element is selected next. In other words, Element 4 is selected if x1 is less than or equal to x2, and Element 5 is selected if x2 is less than x1. Note that the same processing occurs for moving between Element 1 and Element 4, but the result is obvious. In contrast, there is no ambiguity in moving from Element **4** to Element 6 and then to Element 7.

FIGS. **5**A–**5**H illustrate various interactive elements, the operation of which will be described in greater detail with respect to FIGS. **6**–**8**. Those of skill in the art will recognize that the interactive elements shown in FIGS. **5**A–**5**H are merely exemplary of interactive elements that are useful in

describing embodiments of the present invention in the context of HTML content. However, as explained previously, the present invention is not necessarily limited to any particular types of interactive elements or any particular type of content. For example, JAVA and Javascript allow for the creation of interactive elements and show that the types of interactive elements within the scope of the present invention are governed only by the creativity of those who author content. In addition to existing interactive elements, therefore, it is fully anticipated that new interactive elements will be developed and should be considered to fall within the meaning of interactive elements as used in this application, regardless of the particular technology that implements and/or deploys a particular interactive element. Furthermore, it should be apparent that the following discussion does not catalog all existing interactive elements, but rather, identifies a sufficient number to adequately describe how the present invention operates.

FIG. **5**A illustrates single line textbox **510** for entry of characters. Although display is limited to a single line, characters within the textbox may allow for scrolling if character entry exceeds the width of textbox **510**. FIG. **5**B illustrates multiple line textbox **520**. Multiple line textbox **520** is also for character entry. Due to display area constraints, multiple line textbox **520** may display as a single line in navigation mode, and then to facilitate editing, expand to a multiple line display in edit mode. A close button may be included with multiple line textbox **520** to assist in returning to navigation mode. Like single line textbox **510**, multiple line textbox **520** may allow for scrolling if character entry exceeds the width of textbox **520**.

Radio button **530**, with button **532** and text **534**, is illustrated in FIG. **5**C. Radio buttons allow for choosing one item and only one item from a group or list of items. FIG. **5**D illustrates checkbox **540**, with button **542** and text **544**. In contrast to radio button **530**, checkbox **540** allows for selecting zero or more items from a group or list of items. FIG. **5**E illustrates spinner **550**, with text **552**, left arrow **556**, and right arrow **554**. Similar to radio buttons, spinner **550** groups related items and allows one and only one to be chosen. Activating left arrow **556** chooses the previous item in the list and activating right arrow **554** chooses the next item in the list. The list may be circular, such that moving through all choices with either arrow returns to the initial choice. Alternatively the list may be linear, having a starting point that may be reached with left arrow **556** and having an ending point that may be reached with right arrow **554**. Picker **560**, with text **562** and right arrow **564**, as illustrated in FIG. **5**F, is similar to checkbox **540**. Activating right arrow **564** displays a list of checkbox options. Like multiple line textbox **520**, a close button may be included with picker **560** to facilitate returning to navigation mode. Activating button **570** of FIG. **5**G causes an action associated with the button to be executed. FIG. **5**H illustrates link **580**, a hypertext markup language link that browses content associated with the link when the link is activated or followed. FIG. **6** is a flow diagram that corresponds to receiving input while editing a selected interactive element, such as one of those described above in connection with FIGS. **5**A–**5**H. Depending on the selected interactive element, input received while in edit mode may be used by the interactive element (e.g., entering characters into a textbox) or may cause a return to navigation mode (e.g., so that another interactive element may be selected). Following the discussion of edit mode and FIG. **6**, the description of FIGS. **7** and **8** explains how transitions to edit mode are made from the navigation mode identified above, with respect to FIGS. **3** and **4**.

Turning now then to FIG. **6**, during operation in edit mode **610**, an input **630** is received. If the selected interactive element is only partially visible, entering edit mode will scroll **620** the display until the selected element is completely visible. Decision block **640** determines if input **630** is a direction input. If so, decision block **650** determines if the selected interactive element accepts direction input so that the direction input may be processed **652**. For example, direction input may be accepted in single line textbox **510** and multiple line textbox **520** for moving the cursor position, although up and down direction input would not be accepted by single line textbox **510**. Spinner **550** also may accept direction input, with a left direction input choosing a previous item in the spinner list and a right direction input choosing the next item in the spinner list. If decision block **650** determines that the selected interactive element does not accept direction input, operation transitions or switches to navigation mode **654**.

If input **630** is not a direction input, decision block **660** determines if input **630** is an action input. If not, input **630** is processed as character input **662**. Note that FIG. **6** suggests three basic types of input: direction input, action input, and character input. However, the present invention does not necessarily require dividing all input into any particular number of categories. It is expected, therefore, that alternate embodiments may use additional, fewer, or other categories, depending on the needs or preferences of a particular application. Furthermore, alternate embodiments also may include additional, fewer, or other modes of operation, again depending on the needs or preferences of the particular application.

Processing character input depends on the selected interactive element. Ordinarily, character input has greatest application in entering information into single line textbox **510** and multiple line textbox **520**. However, character input also may be used in finding a particular entry in spinner **550** and picker **560**. For example, if spinner **550** or picker **560** are used to chose a state based on state codes, entering a "W" may immediately move to the end of the list, as opposed to moving through a list item by item or page by page as would likely occur using a direction input.

If decision block **660** determines that input **630** is an action input, decision block **670** considers whether the content being browsed is form content without a submit button. Some form content may fail to provide an explicit submit button, in which case an action input is interpreted as a request to submit the form **672**. In the usual case, however, an action input switches from edit mode back to navigation mode **674**. For spinner **550**, switching from edit mode to navigation mode **674** is somewhat different behavior than occurs in an operating system shell context. More specifically, in an operating system shell context, once a spinner has been selected, an action input ordinarily displays a list of radio button options.

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements. During operation in navigation mode **710**, an action input **720** is received. If decision block **730** determines that the selected interactive element is a link, receiving an action input activates or follows the link **732**. If the selected interactive element is a radio button or checkbox, as determined in decision block **740**, the state of the radio button or checkbox is switched. Radio buttons and checkboxes are both examples of interactive elements capable of representing two states.

Because multiple checkboxes may be chosen, switching the state of a checkbox means either (i) an unchecked

checkbox is checked, or (ii) a checked checkbox is unchecked. Radio buttons are slightly more complex to explain since only one item may be chosen at any given time. Therefore, switching the state of a radio button means that (i) if the radio button was not chosen previously, the radio button will be chosen and any other radio button that may have been chosen previously will no longer be chosen, or (ii) a radio button that was previously chosen remains chosen.

If the selected interactive element is a button, as determined in decision block **750**, the action associated with the button is executed or performed **752** upon receiving an action input **720**. Note that links, radio buttons, checkboxes, and buttons are examples of interactive elements that operate in an intuitive manner without using a dedicated edit mode. For other interactive elements, such single line textbox **510**, multiple line textbox **520**, spinner **550**, and picker **560**, an action input switches to edit mode **754**.

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements. During operation in navigation mode **810**, a character input **820** is received. Character input may include numbers, letters, punctuation, white space, etc. If decision block **830** determines that the interactive element does not accept character input, character input **820** is ignored **836**. Otherwise, character input **820** causes a switch from navigation mode to edit mode **832** and character input **820** is processed **834**. For example, if the selected interactive element is a single line textbox **510** or a multiple line textbox **520**, entering a letter while in navigation mode switches to edit mode and places the letter in the textbox. As indicated earlier, spinner **550** and picker **560** also may use character input to find an item in a long list by advancing to and/or choosing an item that matches the character entered.

FIGS. **9**A–**9**C show an exemplary method for browsing content according to the present invention. A step for presenting (**910**) at least a portion of content on the display area of a browsing system may include the acts of retrieving (**912**) the content and displaying (**914**) the retrieved content. The content may be retrieved from a local or remote source. A step for controlling the width (**920***a*) of an interactive element may include the act of adjusting the width of the interactive element to the width of available display area on the browsing system if the width of the interactive element exceeds the width of available display area.

A step for determining (**930***a*) an interactive element for selection based on a direction input may include the following acts: an act of starting (**932**) in navigation mode by default when content displays; an act of receiving (**934***a*) a direction input from a direction key, such as four-direction and action key **210** of FIG. **2**; an act of scrolling (**936***a*) the display of the content in the direction of the received direction input if less than all of the content is displayed and no interactive element is at least partially visible; and an act of selecting (**938***a*) an interactive element based on the received direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed content. Note that for English content, an up arrow or left arrow relative to the beginning of the display content does not make sense and therefore is ignored.

A step for indicating (**940***a*) that an interactive element is selected may include the act of highlighting (**944***a*) the interactive element. For example, a selection box may be placed around the interactive element or the visual appearance of the interactive element may be otherwise altered such that a selected interactive element is distinguishable from an interactive element that has not been selected. The present invention does not necessarily require any particular type of highlighting.

A step for changing (**950**) the mode of a browsing system may include an act of switching (**951**) from navigation mode to edit mode upon receiving a character input. For example, when an interactive element such as single line textbox **510**, multiple line textbox **520**, spinner **550**, or picker **560** is selected, receiving a character input switches to edit mode. An act of switching (**953**) from navigation mode to edit mode upon receiving an action input also may be included as part of a step from changing (**950**) the mode of a browsing system. An action input is an explicit indication to switch modes, whereas a character input is an implied indication to switch modes.

Once in edit mode, a step for changing (**950**) the mode of a browsing system may include the act of switching (**955**) from edit mode to navigation mode upon receiving an action input. In other words, an action input may be used both for entering and exiting edit mode. Likewise, an act of switching (**957**) from edit mode to navigation mode upon receiving a direction input also may be included within a step form changing (**950**) the mode of a browsing system. For interactive elements that do not accept a particular direction input, such as an up or down arrow in a single line textbox, receiving the particular direction input switches from edit mode to navigation mode.

Additionally, a step for changing (**950**) the mode of a browsing system may include the act of submitting (**959**) form content upon receiving an action input. For content that does not include a submit button, an action input is associated with submitting the form. After submitting a form, the browsing system switches from edit mode to navigation mode because submitting a form usually causes new content to be displayed by a browsing device and, as described above, the browsing system may default to navigation mode as content initially displays.

Some types of interactive elements may operative intuitively without a dedicated edit mode and therefore switching modes may not be necessary for interacting with those elements. The present invention may include the act of switching (**960**) the state of a selected interactive element such as a radio button or checkbox. Similarly, the act of following (**970**) a selected link upon receiving an action input or executing the action (not shown) associated with a button also are within the scope of the present invention.

If selected interactive element is only partially visible, a step for controlling the width (**920***c*) of an interactive element (see **920***a* of FIG. **9**A) also may include the act of scrolling (**924**) the display area until the interactive element is completely visible. Automatically scrolling the display of content ordinarily occurs when switching from navigation mode to edit mode. Upon switching from navigation mode to edit mode, it becomes clear that a particular interactive element is of interest and should be completely visible, whereas in navigation mode, it may be unclear whether interest is in (i) an interactive element selected as a natural consequence of scrolling displayed content, or (ii) other content that becomes visible as content scrolls.

In addition to the acts described in connection with FIG. **9**A, a step for determining (**930***c*) an interactive element for selection based on a direction input may include other acts, such as an act of receiving (**934***c*) a subsequent direction input. If less than all of the content is display and no

interactive element is at least partially visible, determining (**930***c*) an interactive element for selection also may include an act of scrolling (**936***c*) the display of the content in the direction of the subsequent direction input. For a horizontal direction input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**1**) the closest interactive element in the direction of the direction input that is at substantially the same vertical level. For a vertical direction input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**2**) the interactive element at the next vertical level in the direction of the direction input that is closest in the horizontal direction to a previously selected interactive element. A step for indicating (**940***c*) that an interactive element is selected may further include the acts of removing (**942**) the highlighting from a previously selected interactive element and highlighting (**944***c*) the next selected interactive element (see also **940***a* of FIG. **9**A).

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed and desired to be secured by United States Letters Patent is:

1. In either a wireless telephone or personal digital assistant configured for browsing content received from a content source, and having a display area and input system that is limited as compared to pointing devices and displays often found in more traditional browsing systems, and wherein one or more interactive elements within content received from the content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, a method of browsing content that includes one or more interactive elements, the method comprising acts of:

displaying at least a portion of the content on a display area of a browsing system;

receiving a direction input generated by activating a navigation key;

while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input;

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein said interactive element can be only partially visible; and

highlighting the interactive element to indicate that the interactive element is selected.

2. A method as recited in claim **1**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

3. A method as recited in claim **1**, wherein browsing includes a navigation mode and an edit mode, the navigation mode being characterized by selection of interactive elements and the edit mode being characterized by interaction with a single interactive element.

4. A method as recited in claim **3**, wherein the interactive element accepts character input, the method further comprising an act of switching from navigation mode to edit mode upon receiving a character input.

5. A method as recited in claim **3**, wherein the interactive element accepts character input or allows for one or more items to be selected from a group of one or more items, the method further comprising an act of switching from navigation mode to edit mode upon receiving an action input.

6. A method as recited in claim **3**, further comprising the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving an action input.

7. A method as recited in claim **3**, wherein the interactive element does not accept a direction input, the method further comprising the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

8. A method as recited in claim **3**, wherein the interactive element is part of form content that does not include a submit element, the method further comprising the acts of:

browsing in edit mode; and

submitting the form content upon receiving an action input.

9. A method as recited in claim **1**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input.

10. A method as recited in claim **1**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

11. A method as recited in claim **1**, wherein the interactive element exceeds the width of available browsing system display area, the method further comprising an act of adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area.

12. A method as recited in claim **1**, wherein the interactive element is only partially visible in the browsing system display area, the method further comprising acts of:

adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area if the width of the interactive element exceeds the width of available browsing system display area; and

scrolling the browsing system display area until the interactive element is completely visible.

13. A method as recited in claim **3**, wherein the selected interactive element is a previously selected interactive element, the method further comprising the acts of:

receiving a subsequent direction input that corresponds to scrolling the browsing system display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if less than all of the content is displayed and no other interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

selecting a next interactive element that is at least partially visible, the selection being based on the subsequent

direction input relative to the previously selected interactive element;

removing highlighting from the previously selected interactive element to indicate that the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

14. A method as recited in claim **13** wherein the subsequent direction input is a horizontal direction input, and wherein the next interactive element is selected based on the next interactive element being (i) a closest interactive element in the direction of the horizontal direction input, that is (ii) at substantially the same vertical level as the previously selected interactive element.

15. A method as recited in claim **13** wherein the subsequent direction input is a vertical direction input, and wherein the next interactive element is selected based on the next interactive element being at the next vertical level, from the previously selected interactive element, in the direction of the vertical direction input.

16. A method as recited in claim **15**, wherein multiple interactive elements are displayed at the next vertical level, the next interactive element being selected based on the next interactive element being an interactive element that is closest in horizontal direction to the beginning of the previously selected interactive element.

17. In either a wireless telephone or personal digital assistant having a display area and input system, the input system including a direction key and an action key, wherein the input system and display area are limited as compared to a pointing device and larger display area often found in more traditional browsing systems, and wherein one or more interactive elements within content received from a content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, a method of browsing content that contains one or more interactive elements, wherein the browsing includes an edit mode and a navigation mode, the met hod comprising acts of:

starting in the navigation mode;

displaying at least a portion of the content on the wireless telephone or personal digital assistant display area;

receiving a direction input generated by the direction key of the wireless telephone or personal digital assistant;

while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, scrolling the display of the content in the direction of the direction input;

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content; and

placing a selection box around the interactive element to indicate that the interactive element is selected.

18. A method as recited in claim **17**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

19. A method as recited in claim **17**, wherein the interactive element accepts character input, the method further comprising an act of switching from navigation mode to edit mode upon receiving a character input.

20. A method as recited in claim **17**, wherein the interactive element accepts character input or allows for one or more items to be selected from a group of one or more items, the method further comprising an act of switching from navigation mode to edit mode upon receiving an action input.

21. A method as recited in claim **17**, the method further comprising the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

switching from edit mode to navigation mode upon receiving an action input.

22. A method as recited in claim **17**, wherein the interactive element does not accept a direction input, the method further comprising the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

23. A method as recited in claim **17**, wherein the interactive element is part of form content that does not include a submit element, the method further comprising the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

submitting the form content upon receiving an action input.

24. A method as recited in claim **17**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input while in navigation mode.

25. A method as recited in claim **17**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

26. A method as recited in claim **17**, wherein the interactive element exceeds the width of available wireless telephone or personal digital assistant display area, the method further comprising an act of adjusting the width of the interactive element to be less than or equal to the width of available wireless telephone or personal digital assistant display area.

27. A method as recited in claim **17**, wherein the interactive element is only partially visible in the wireless telephone or personal digital assistant display area, the method further comprising acts of:

adjusting the width of the interactive element to be less than or equal to the width of available wireless telephone or personal digital assistant display area if the width of the interactive element exceeds the width of available wireless telephone or personal digital assistant display area; and

scrolling the wireless telephone or personal digital assistant display area until the interactive element is completely visible.

28. A method as recited in claim **17**, wherein the selected interactive element is a previously selected interactive element, the method further comprising the acts of:

receiving a subsequent direction input that corresponds to scrolling the wireless telephone or personal digital assistant display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if less than all of the content is displayed and no other

interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

selecting a next interactive element that is at least partially visible, the selection being based on the subsequent direction input relative to the previously selected interactive element;

removing highlighting from the previously selected interactive element to indicate that the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

29. A method as recited in claim **28** wherein the subsequent direction input is a horizontal direction input, and wherein the next interactive element is selected based on the next interactive element being (i) a closest interactive element in the direction of the horizontal direction input, that is (ii) at substantially the same vertical level as the previously selected interactive element.

30. A method as recited in claim **28** wherein the subsequent direction input is a vertical direction input, and wherein the next interactive element is selected based on the next interactive element being at the next vertical level, from the previously selected interactive element, in the direction of the vertical direction input.

31. A method as recited in claim **30** wherein multiple interactive elements are displayed at the next vertical level, the next interactive element being selected based on the next interactive element being an interactive element that is closest in horizontal direction to the beginning of the previously selected interactive element.

32. In either a wireless telephone or personal digital assistant configured for browsing content received from a content source, and having a display area and input system that is limited as compared to pointing devices and displays often found in more traditional browsing systems, and wherein one or more interactive elements within content received from the content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, a method of browsing content that includes one or more interactive elements, wherein the browsing includes an edit mode and a navigation mode, the method comprising steps for:

presenting at least a portion of the content on a display area of a browsing system;

receiving a direction input generated by activating a navigation key;

upon receiving a direction input, determining an interactive element has been selected based on the direction input, wherein the interactive element is only partially visible;

visually indicating that the interactive element is selected; and

automatically scrolling the browsing system upon determining that the interactive element has been selected and that the interactive element is only partially visible in the browsing system display area, wherein the browsing system display area is automatically scrolled until the interactive element is completely visible.

33. A method as recited in claim **32**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

34. A method as recited in claim **32**, further comprising a step for changing the mode of the browsing system.

35. A method as recited in claim **34**, wherein the interactive element accepts character input, and wherein the step for changing the mode of the browsing system comprises an act of switching from navigation mode to edit mode upon receiving a character input.

36. A method as recited in claim **34**, wherein the interactive element accepts character input or allows for one or more items to be selected from a group of one or more items, and wherein the step for changing the mode of the browsing system comprises an act of switching from navigation mode to edit mode upon receiving an action input.

37. A method as recited in claim **34**, wherein the step for changing the mode of browsing comprises the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving an action input.

38. A method as recited in claim **34**, wherein the interactive element does not accept a direction input, and wherein the step for changing the mode of browsing comprises the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

39. A method as recited in claim **34**, wherein the interactive element is part of form content that does not include a submit element, and wherein the step for changing the mode of browsing comprises an acts of:

browsing in edit mode; and

submitting the form content upon receiving an action input.

40. A method as recited in claim **32**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input while in navigation mode.

41. A method as recited in claim **32**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

42. A method as recited in claim **32**, wherein the interactive element exceeds the width of available browsing system display area, the method further comprising a step for controlling the width of the interactive element.

43. A method as recited in claim **42**, wherein the step for controlling the width of the interactive element comprises the acts of:

adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area if the width of the interactive element exceeds the width of available browsing system display area.

44. A method as recited in claim **32**, wherein the step for determining an interactive element for selection based on a direction input comprises acts of:

receiving a direction input that corresponds to scrolling the browsing system display area, the direction input being generated by activating a navigation key;

while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible, scrolling the display of the content in the direction of the direction input; and

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, of no interactive element has been previously selected,

based on the direction input relative to the beginning of the displayed portion of the content.

45. A method as recited in claim **44**, wherein the selected interactive element is a previously selected interactive element, and wherein the step for determining an interactive element for selection based on a direction input further comprises acts of:

receiving a subsequent direction input that corresponds to scrolling the browsing system display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if less than all of the content is displayed and no other interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

selecting a next interactive element that is at least partially visible, the selection being based on the subsequent direction input relative to the previously selected interactive element.

46. A meted as recited in claim **45** wherein the step for indicating that the interactive element is selected comprises acts of:

removing highlighting from the previously selected interactive element to indicate that the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

47. A method as recited in claim **45** wherein the subsequent direction input is a horizontal direction input, and wherein the next interactive element is selected based on the next interactive element being (i) a closest interactive element in the direction of the horizontal direction input, that is (ii) at substantially the same vertical level as the previously selected interactive element.

48. A method as recited in claim **45** wherein the subsequent direction input is a vertical direction input, and wherein the next interactive element is selected based on the next interactive element being at the next vertical level, from the previously selected interactive element, in the direction of the vertical direction input.

49. A method as recited in claim **48** wherein multiple interactive elements are displayed at the next vertical level, the next interactive element being selected based on the next interactive element being an interactive element that is closest in horizontal direction to the beginning of the previously selected interactive element.

50. In either a wireless telephone or personal digital assistant configured for browsing content received from a content source, and having a display area and input system that is limited as compared to pointing devices and displays often found in more traditional browsing systems, and wherein one or more interactive elements within content received from the content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, a computer program product for implementing a method of browsing content that includes one or more interactive elements, the computer program product comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method; and wherein said method is comprised of machine-executable instructions performing acts of:

displaying at least a portion of the content on a display area of a browsing system;

receiving a direction input generated by activating a navigation key;

while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input;

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein said interactive element can be only partially visible; and

highlighting the interactive element to indicate that the interactive element is selected.

51. A computer program product as recited in claim **50**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinners a radio button, a checkbox, a button, and a picker.

52. A computer program product as recited in claim **50**, wherein browsing includes a navigation mode and an edit mode, the navigation mode being characterized by selection of interactive elements and the edit mode being characterized by interaction with a single interactive element.

53. A computer program product as recited in claim **52**, wherein the interactive element accepts character input, the method further comprising an act of switching from navigation mode to edit mode upon receiving a character input.

54. A computer program product as recited in claim **52**, wherein the interactive element accepts character input or allows for one or more items to be selected from a group of one or more items, the method further comprising an act of switching from navigation mode to edit mode upon receiving an action input.

55. A computer program product as recited in claim **52**, the method further comprising the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving an action input.

56. A computer program product as recited in claim **52**, wherein the interactive element does not accept a direction input, the method further comprising the acts of:

browsing in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

57. A computer program product as recited in claim **52**, wherein the interactive element is part of form content that does not include a submit element, the method further comprising the acts of:

browsing in edit mode; and

submitting the form content upon receiving an action input.

58. A computer program product as recited in claim **50**, wherein the interactive element is capable of representing two stares, the method further comprising an act of switching from one state to the other upon receiving an action input.

59. A computer program product as recited in claim **50**, wherein the interactive element comprises a link, the method

further comprising an act of following the link upon receiving an action input.

60. A computer program product as recited in claim **50**, wherein the interactive element exceeds the width of available browsing system display area, the method further comprising tin act of adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area.

61. A computer program product as recited in claim **50**, wherein the interactive element is only partially visible in the browsing system display area, the method further comprising acts of:

adjusting the width of the interactive element to be less than or equal to the width of available browsing system display area if the width of the interactive element exceeds the width of available browsing system display area; and

scrolling the browsing system display area until the interactive element is completely visible.

62. A computer program product as recited in claim **50**, wherein the selected interactive element is a previously selected interactive element, the method further comprising the acts of:

receiving a subsequent direction input that corresponds to scrolling the browsing system display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if less than all of the content is displayed and no other interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

being selecting a next interactive element that is at least partially visible, the selection being based on the subsequent direction input relative to the previously selected interactive element;

removing highlighting from the previously selected interactive element to indicate that the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

63. A computer program product as recited in claim **62** wherein the subsequent direction input is a horizontal direction input and wherein the next interactive element is selected based on the next interactive element being (i) a closest interactive element in the direction of the horizontal direction input, that is (ii) at substantially the same vertical level as the previously selected interactive element.

64. A computer program product as recited in claim **62** wherein the subsequent direction input is a vertical direction input, and wherein the next interactive element is selected based on the next interactive element being at the next vertical level, from the previously selected interactive element in the direction of the vertical direction input.

65. A computer program product as recited in claim **62** wherein multiple interactive elements are displayed at the next vertical level, the next interactive element being selected based on the next interactive element being an interactive element that is closest in horizontal direction to the beginning of the previously selected interactive element.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

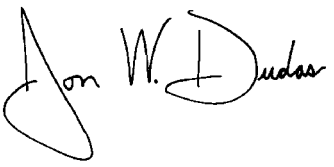
PATENT NO. : 6,822,664 B2
APPLICATION NO. : 09/861327
DATED : November 23, 2004
INVENTOR(S) : Peter O. Vale

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 15
Line 39, change "met hod" to --method--
Line 52, before "scrolling", insert --automatically--
Line 59, after "content", insert --wherein said interactive element can be only partially visible--

Signed and Sealed this

Nineteenth Day of February, 2008



JON W. DUDAS
Director of the United States Patent and Trademark Office

US007421666B2

(12) **United States Patent**
Vale

(10) **Patent No.: US 7,421,666 B2**
(45) **Date of Patent: *Sep. 2, 2008**

(54) **BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM**

(75) Inventor: **Peter O. Vale**, Seattle, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 525 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/968,717**

(22) Filed: **Oct. 19, 2004**

(65) **Prior Publication Data**

US 2005/0081149 A1 Apr. 14, 2005

Related U.S. Application Data

(63) Continuation of application No. 09/861,324, filed on May 18, 2001, now abandoned.

(51) **Int. Cl.**
G06F 3/14 (2006.01)
G06F 3/048 (2006.01)

(52) **U.S. Cl.** **715/864**; 715/785; 715/854

(58) **Field of Classification Search** 715/785, 715/864, 273

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,677,708	A *	10/1997	Matthews et al.	345/684
5,739,821	A	4/1998	Ho	
5,905,497	A	5/1999	Vaughan	
6,005,573	A *	12/1999	Beyda et al.	715/784
6,061,063	A *	5/2000	Wagner et al.	715/784
6,128,012	A	10/2000	Seidensticker	
6,310,634	B1 *	10/2001	Bodnar et al.	715/854
6,778,192	B2	8/2004	Arbab	
7,249,325	B1 *	7/2007	Donaldson	715/777

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0561684	9/1993

(Continued)

OTHER PUBLICATIONS

Marran, N.L., Over-The -Air Subscriber Device Management Using CDMA Data and WAP, Virginia Tech, 1999, pp. 165-174.

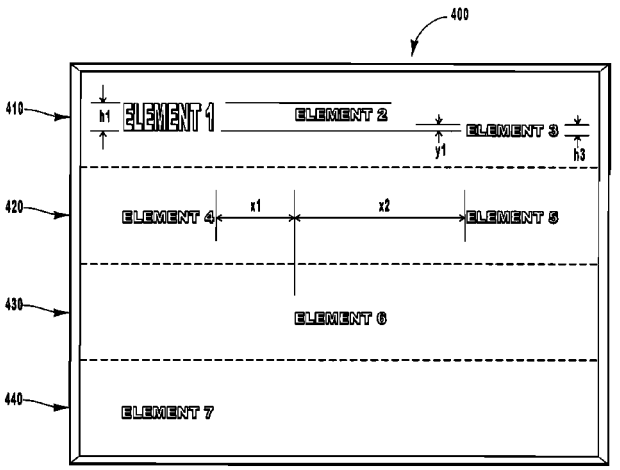
(Continued)

Primary Examiner—William L. Bashore
Assistant Examiner—Stephen Alvesteffer
(74) *Attorney, Agent, or Firm*—Workman Nydegger

(57) **ABSTRACT**

Methods, system, and computer program products for browsing content with a display area and input system that may be limited in comparison to more traditional browsing systems. Movement between and selection of interactive elements generally occurs in a navigation mode, whereas interaction with a single interactive element generally occurs in an edit mode. In navigation mode, a direction input selects the next interactive element in the direction indicated. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls. Switching between navigation mode and edit mode is based on the input received, in view of the input supported, by a particular interactive element. Interactive elements may be limited to the width of available display area.

21 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

2001/0022839	A1	9/2001	Ishigaki	
2002/0070980	A1	6/2002	Le	
2002/0112237	A1	8/2002	Kelts	
2002/0145631	A1*	10/2002	Arbab et al.	345/786
2005/0022140	A1	1/2005	Vale	

FOREIGN PATENT DOCUMENTS

WO	WO 98/29797	7/1998

OTHER PUBLICATIONS

Ming, Tham and Chaun, Tan Kay, Challenges in Designing User Interfaces for Handheld Communication Devices: A Case Study, Lawrence Erlbaum Assocaites; 1999, pp. 808-812.

Fox, A.; Goldberg, I.; Gribble, S.D.; Lee, D.C.; Polito, A.; and Brewer, E.A., Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for 3Com PalmPilot, University of California at Berkeley, 1998, pp. 407-424.

Gessler, Stefan and Kotulla, Andreas, PDAs as Mobile WWW Browsers, Germany 1995, pp. 53-59.

Office Action mailed May 31, 2007, cited in related U.S. Appl. No. 10/923,438.

Kawachiya, Kiyokuni, et al., "NaviPoint: An Input Device for Mobile Information Browsing", Human Factors in Computing Systems. Conference Proceedings, Los Angles, California, Apr. 18-23, 1998.

Notice of Allowance dated May 14, 2008 cited in related U.S. Appl. No. 10/923,438 (Copy Attached).

Motorola, Digital Wireless Telephone, Model 120c, the User Guide, p. 35, 2001.

* cited by examiner

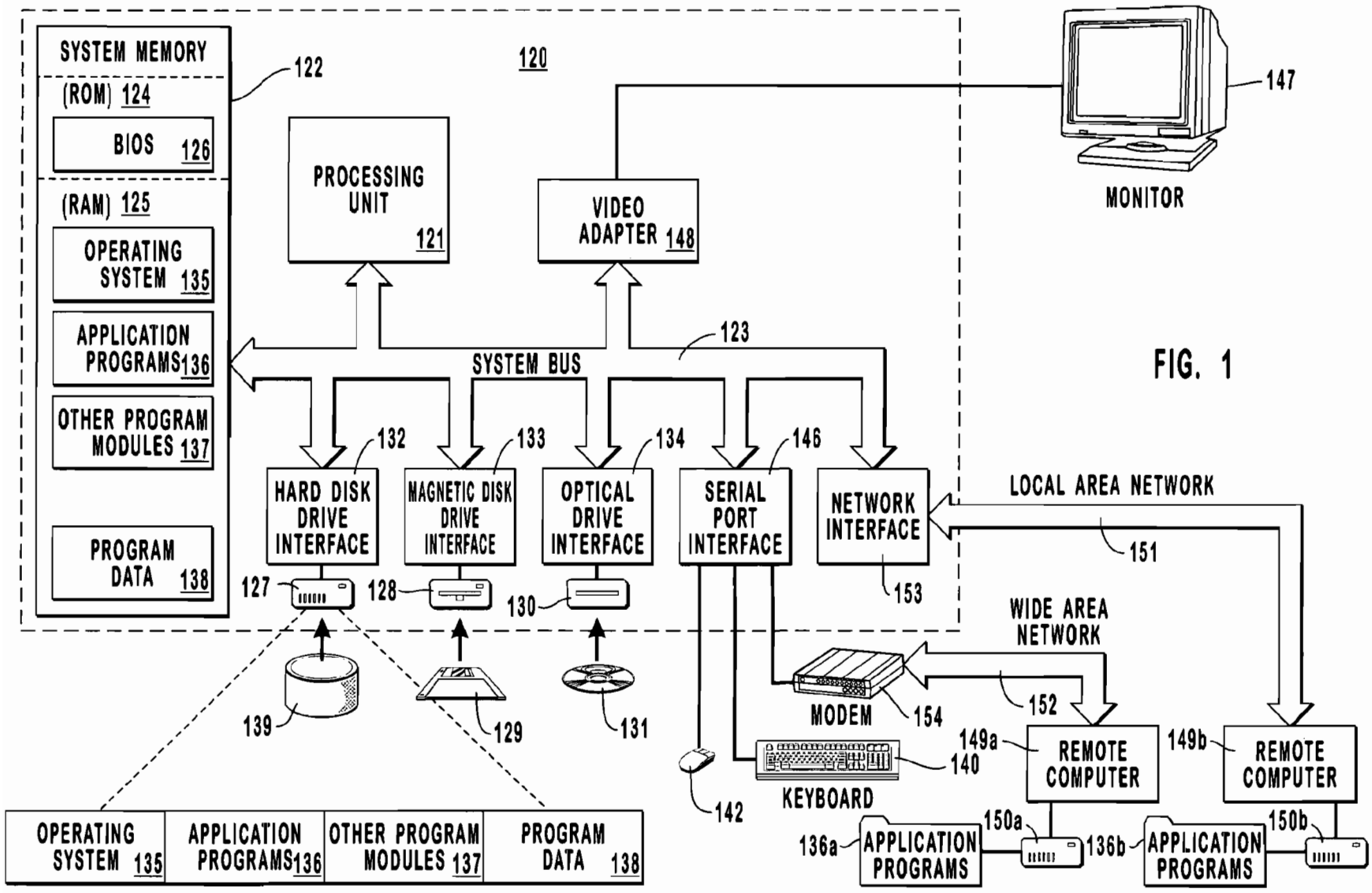

SYSTEM MEMORY
(ROM) 124
BIOS 126
(RAM) 125
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
122
120
PROCESSING UNIT 121
VIDEO ADAPTER 148
147
MONITOR
123
SYSTEM BUS
132
133
134
146
HARD DISK DRIVE INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
LOCAL AREA NETWORK
151
153
127
128
130
WIDE AREA NETWORK
139
129
131
MODEM
154
152
149a
REMOTE COMPUTER
149b
REMOTE COMPUTER
140
KEYBOARD
142
OPERATING SYSTEM 135
APPLICATION PROGRAMS 136
OTHER PROGRAM MODULES 137
PROGRAM DATA 138
136a
APPLICATION PROGRAMS
150a
136b
APPLICATION PROGRAMS
150b

FIG. 1

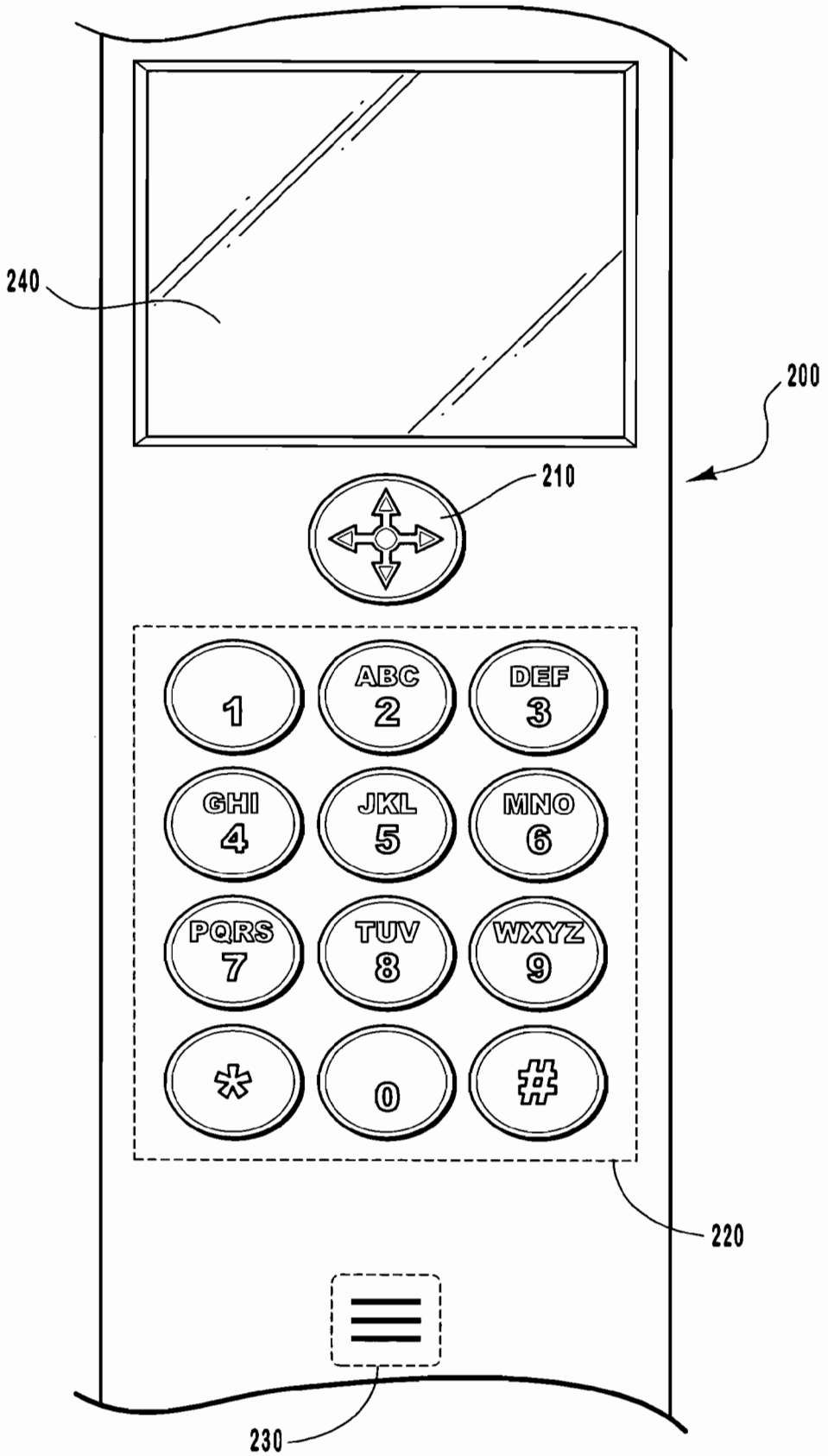

240
200
210
1
ABC 2
DEF 3
GHI 4
JKL 5
MNO 6
PQRS 7
TUV 8
WXYZ 9
*
0
#
220
230

FIG. 2

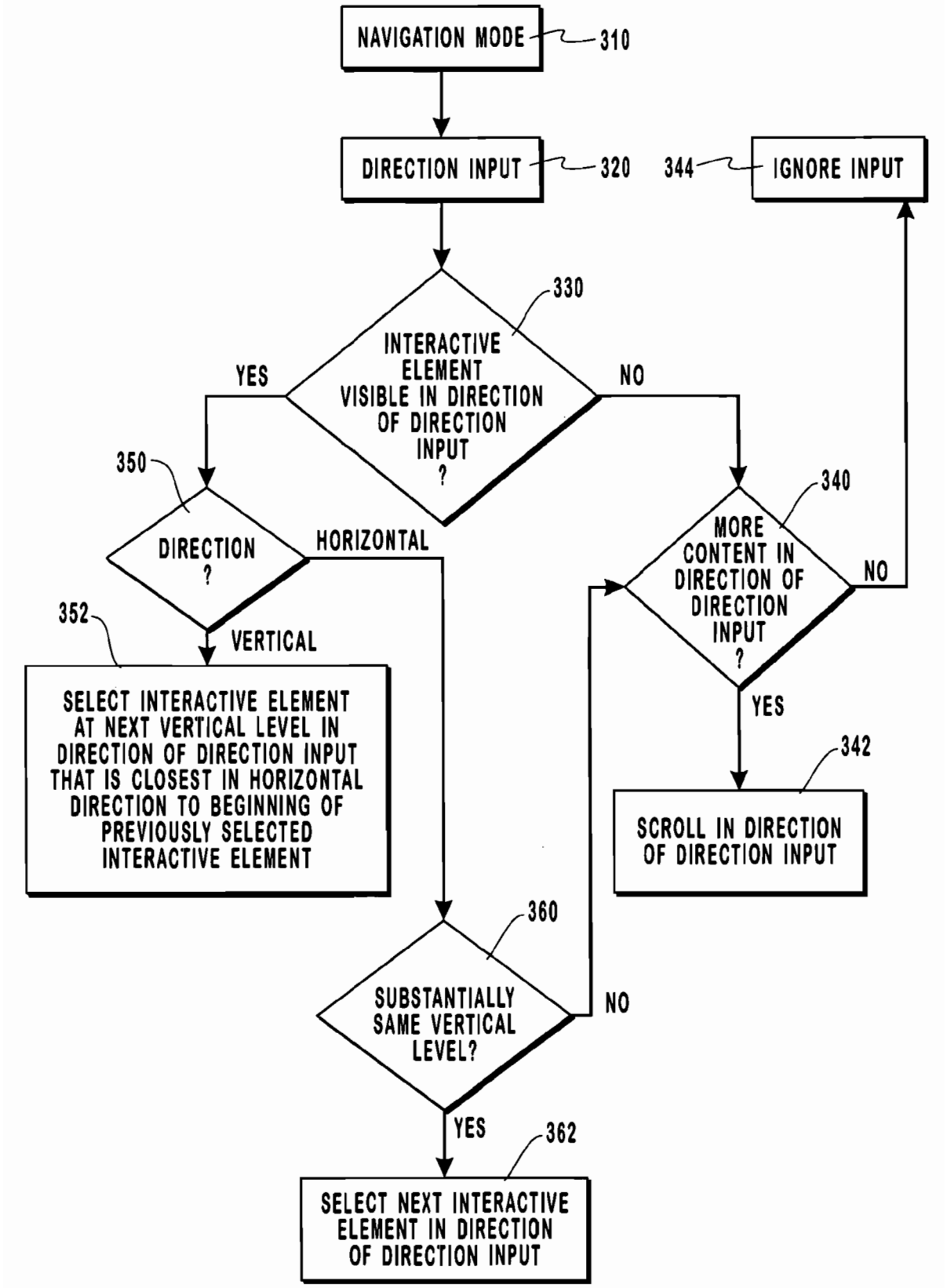

NAVIGATION MODE
310
DIRECTION INPUT
320
344
IGNORE INPUT
330
INTERACTIVE ELEMENT VISIBLE IN DIRECTION OF DIRECTION INPUT ?
YES
NO
350
DIRECTION ?
HORIZONTAL
VERTICAL
352
SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL IN DIRECTION OF DIRECTION INPUT THAT IS CLOSEST IN HORIZONTAL DIRECTION TO BEGINNING OF PREVIOUSLY SELECTED INTERACTIVE ELEMENT
340
MORE CONTENT IN DIRECTION OF DIRECTION INPUT ?
NO
YES
342
SCROLL IN DIRECTION OF DIRECTION INPUT
360
SUBSTANTIALLY SAME VERTICAL LEVEL?
NO
YES
362
SELECT NEXT INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT

FIG. 3

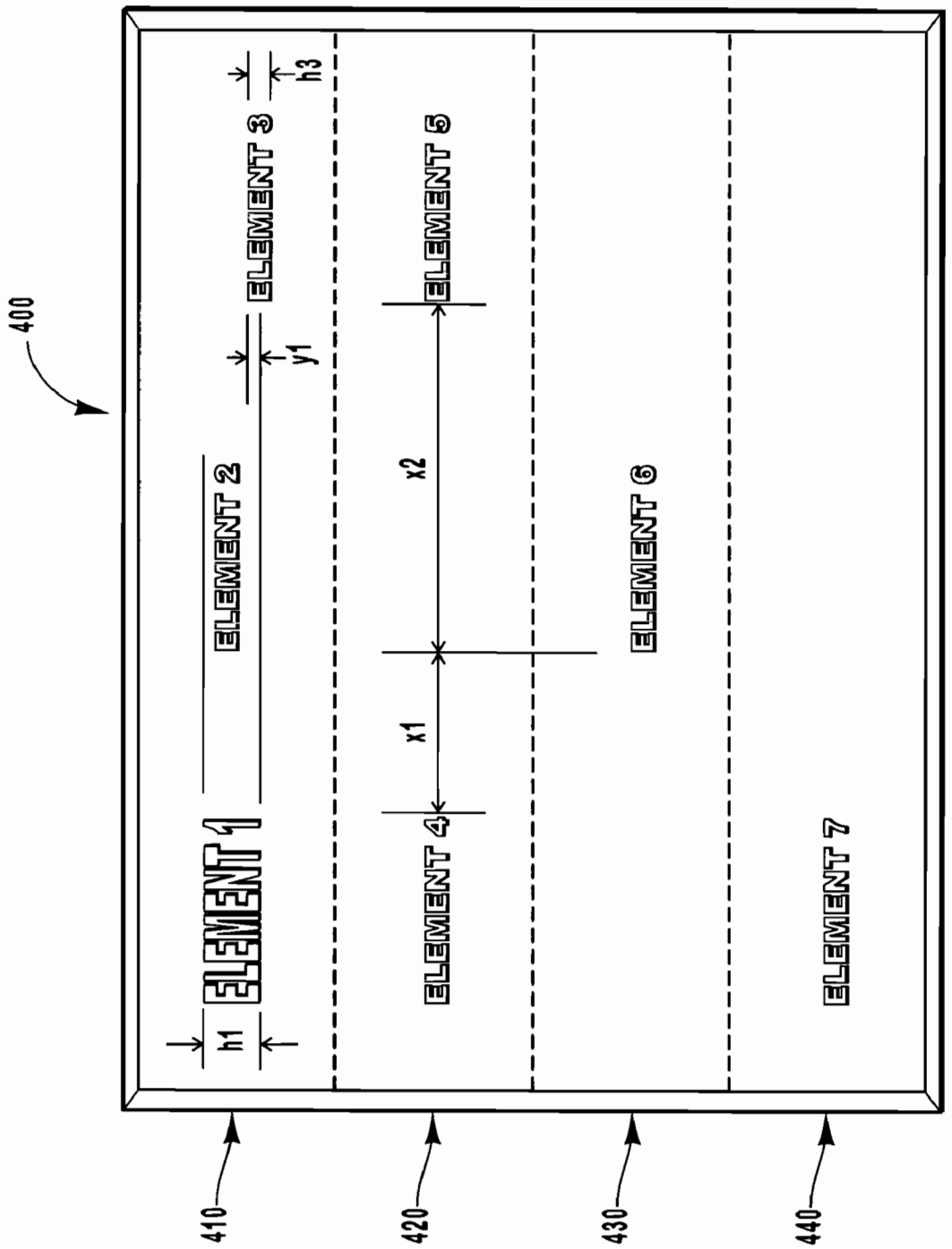

400
ELEMENT 1
ELEMENT 2
ELEMENT 3
ELEMENT 4
ELEMENT 5
ELEMENT 6
ELEMENT 7
h1
h3
y1
x1
x2
410
420
430
440

FIG. 4

FIG. 5A

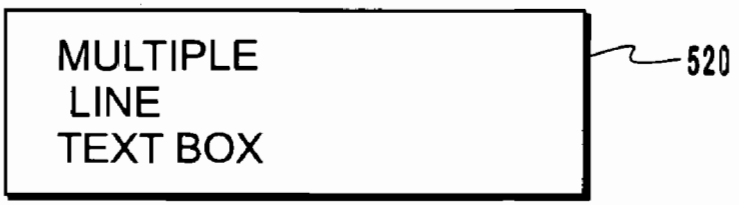


FIG. 5B

FIG. 5C

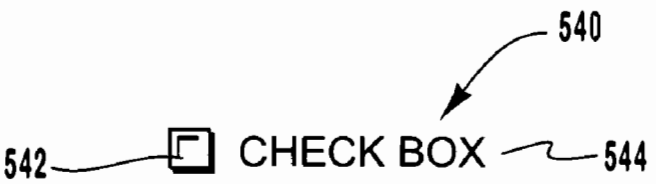


FIG. 5D

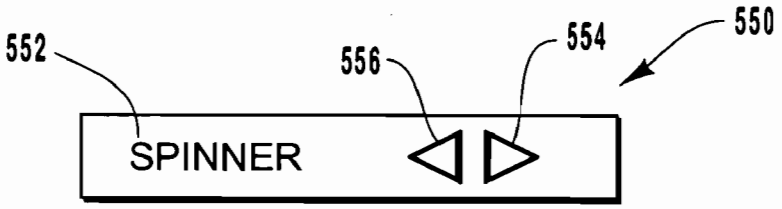


FIG. 5E

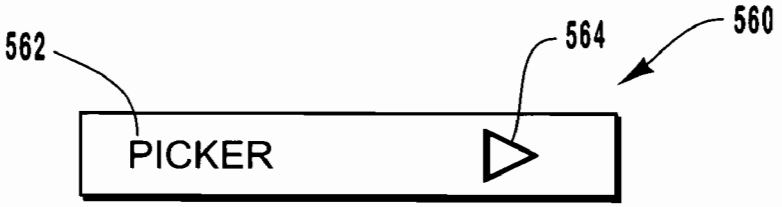


FIG. 5F

FIG. 5G

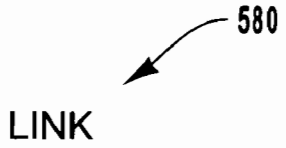


FIG. 5H

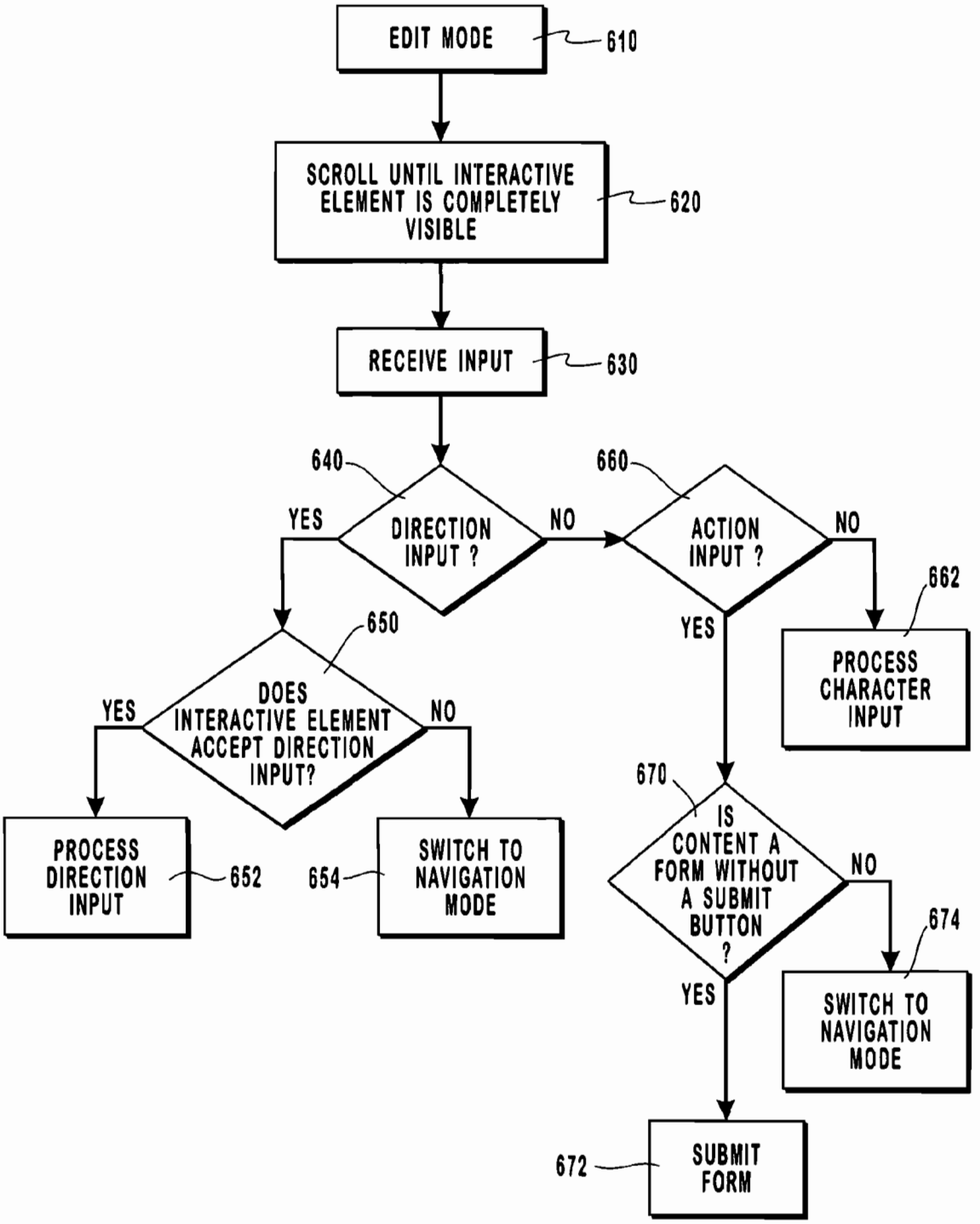

EDIT MODE
610
SCROLL UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
620
RECEIVE INPUT
630
640
DIRECTION INPUT ?
YES
NO
660
ACTION INPUT ?
NO
662
PROCESS CHARACTER INPUT
YES
650
DOES INTERACTIVE ELEMENT ACCEPT DIRECTION INPUT?
YES
NO
PROCESS DIRECTION INPUT
652
654
SWITCH TO NAVIGATION MODE
670
IS CONTENT A FORM WITHOUT A SUBMIT BUTTON ?
NO
674
SWITCH TO NAVIGATION MODE
YES
672
SUBMIT FORM

FIG. 6

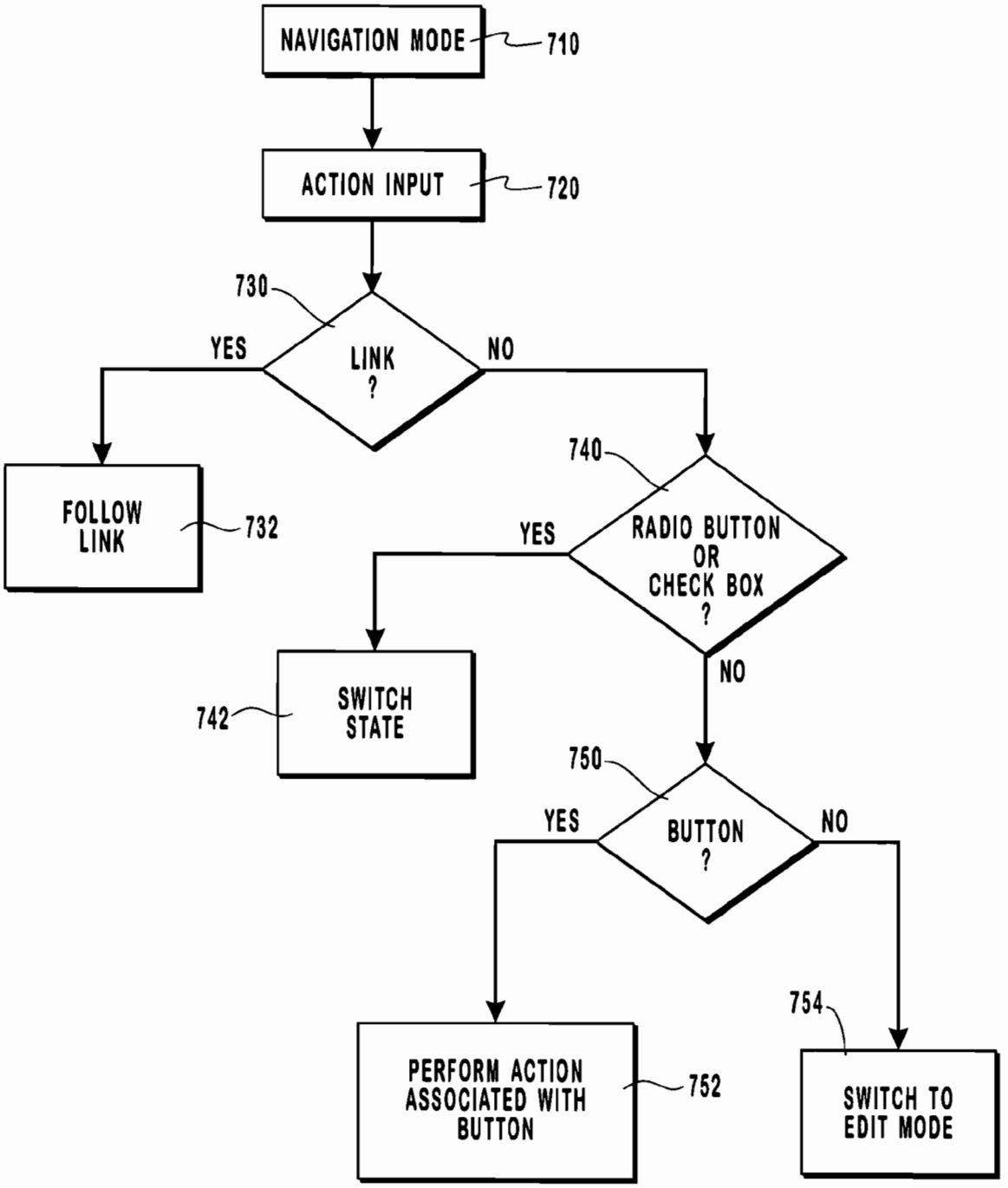

NAVIGATION MODE
710
ACTION INPUT
720
730
LINK ?
YES
NO
FOLLOW LINK
732
740
RADIO BUTTON OR CHECK BOX ?
YES
NO
SWITCH STATE
742
750
BUTTON ?
YES
NO
PERFORM ACTION ASSOCIATED WITH BUTTON
752
754
SWITCH TO EDIT MODE

FIG. 7

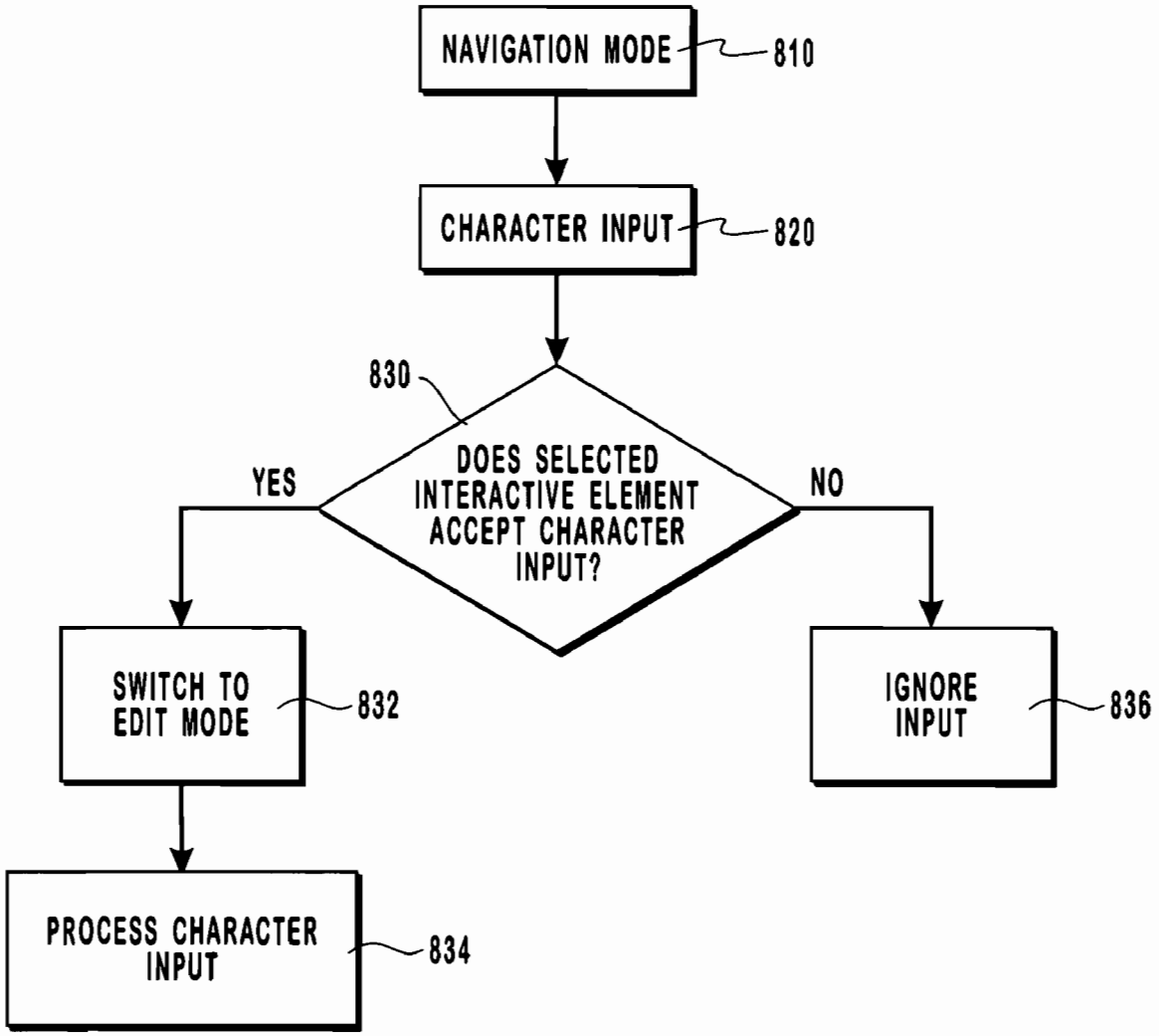

NAVIGATION MODE
810
CHARACTER INPUT
820
830
DOES SELECTED INTERACTIVE ELEMENT ACCEPT CHARACTER INPUT?
YES
NO
SWITCH TO EDIT MODE
832
IGNORE INPUT
836
PROCESS CHARACTER INPUT
834

FIG. 8

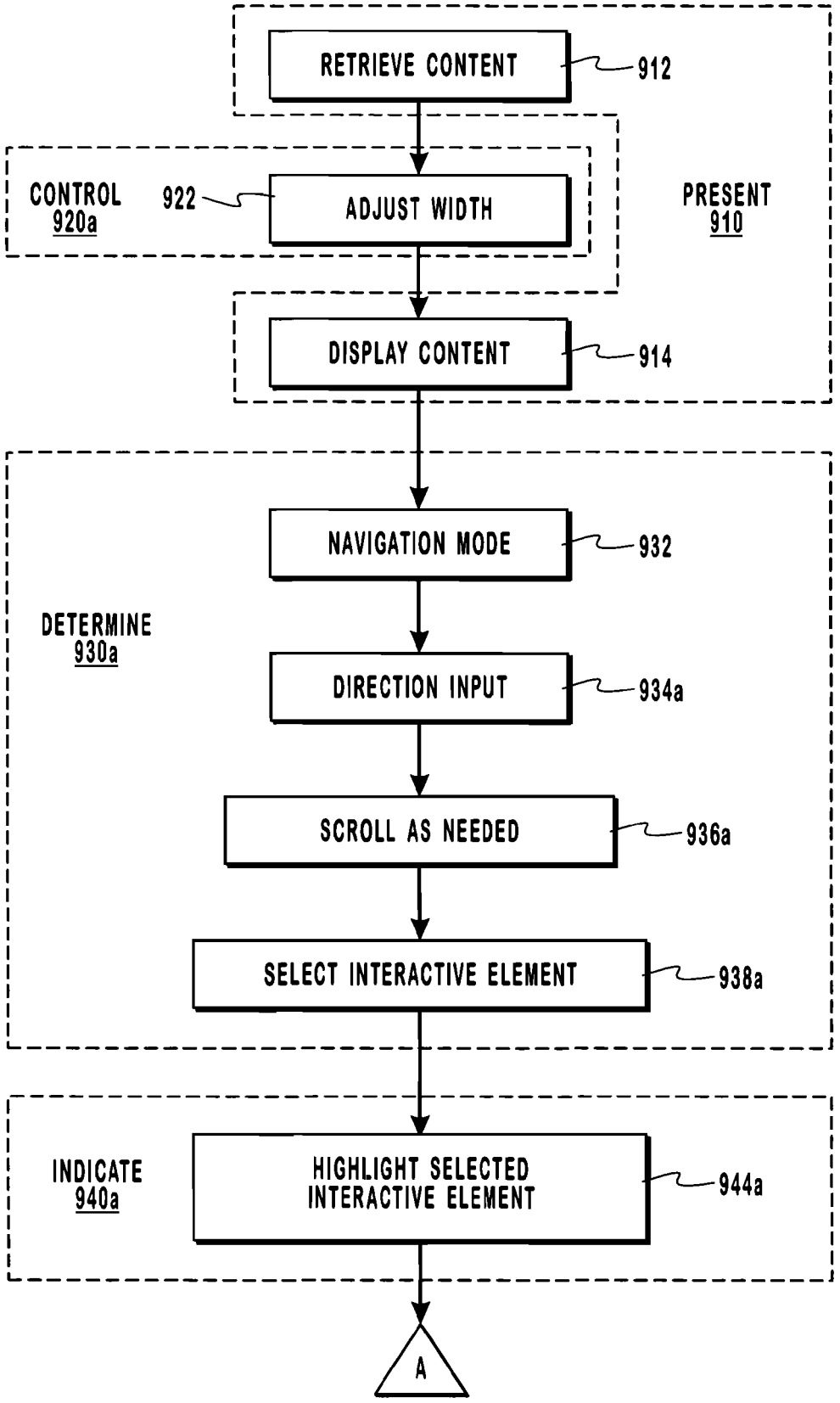

RETRIEVE CONTENT
912
CONTROL
920a
922
ADJUST WIDTH
PRESENT
910
DISPLAY CONTENT
914
NAVIGATION MODE
932
DETERMINE
930a
DIRECTION INPUT
934a
SCROLL AS NEEDED
936a
SELECT INTERACTIVE ELEMENT
938a
INDICATE
940a
HIGHLIGHT SELECTED
INTERACTIVE ELEMENT
944a
A

FIG. 9A

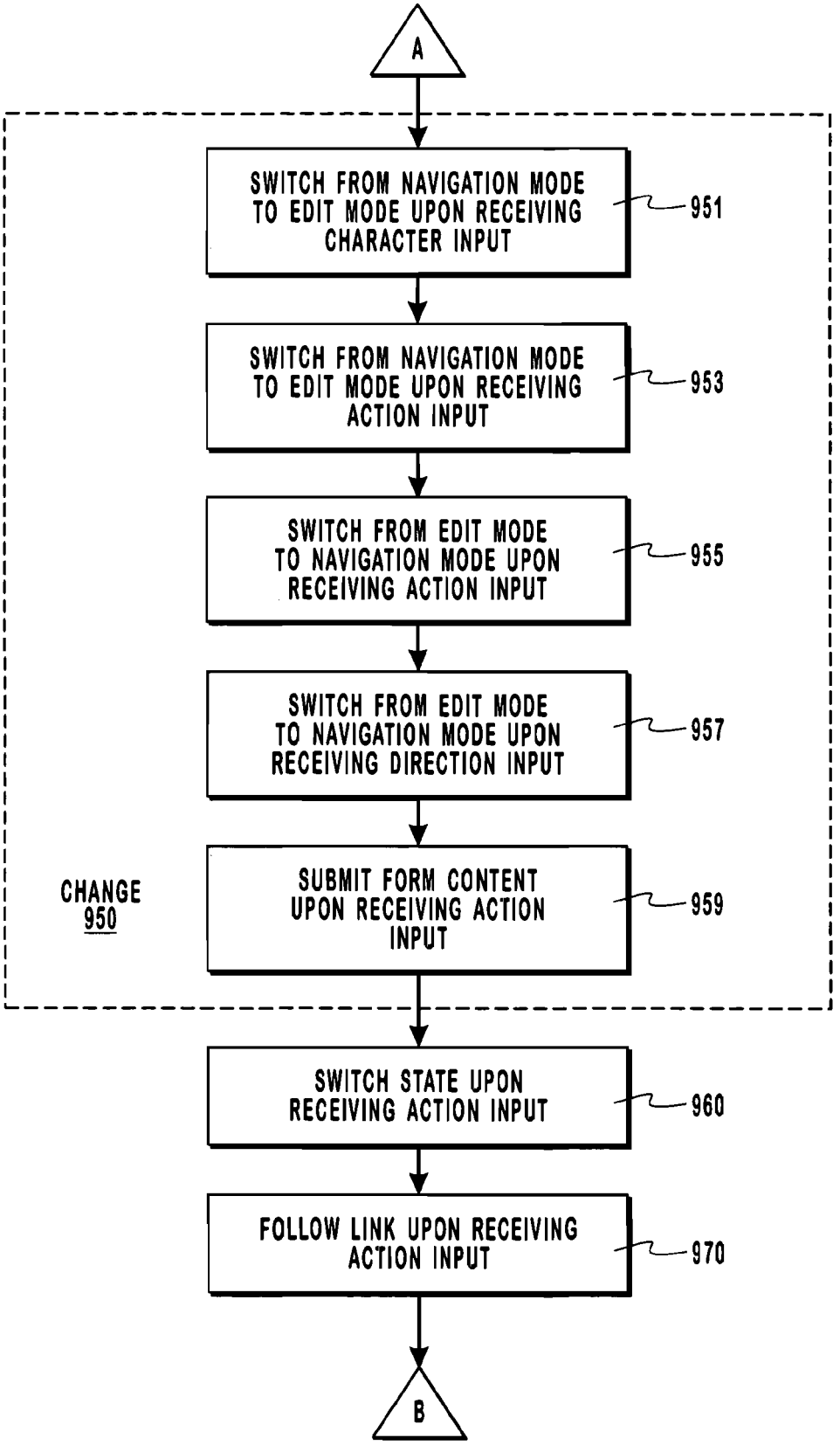

A
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING CHARACTER INPUT
951
SWITCH FROM NAVIGATION MODE TO EDIT MODE UPON RECEIVING ACTION INPUT
953
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING ACTION INPUT
955
SWITCH FROM EDIT MODE TO NAVIGATION MODE UPON RECEIVING DIRECTION INPUT
957
SUBMIT FORM CONTENT UPON RECEIVING ACTION INPUT
959
CHANGE 950
SWITCH STATE UPON RECEIVING ACTION INPUT
960
FOLLOW LINK UPON RECEIVING ACTION INPUT
970
B

FIG. 9B

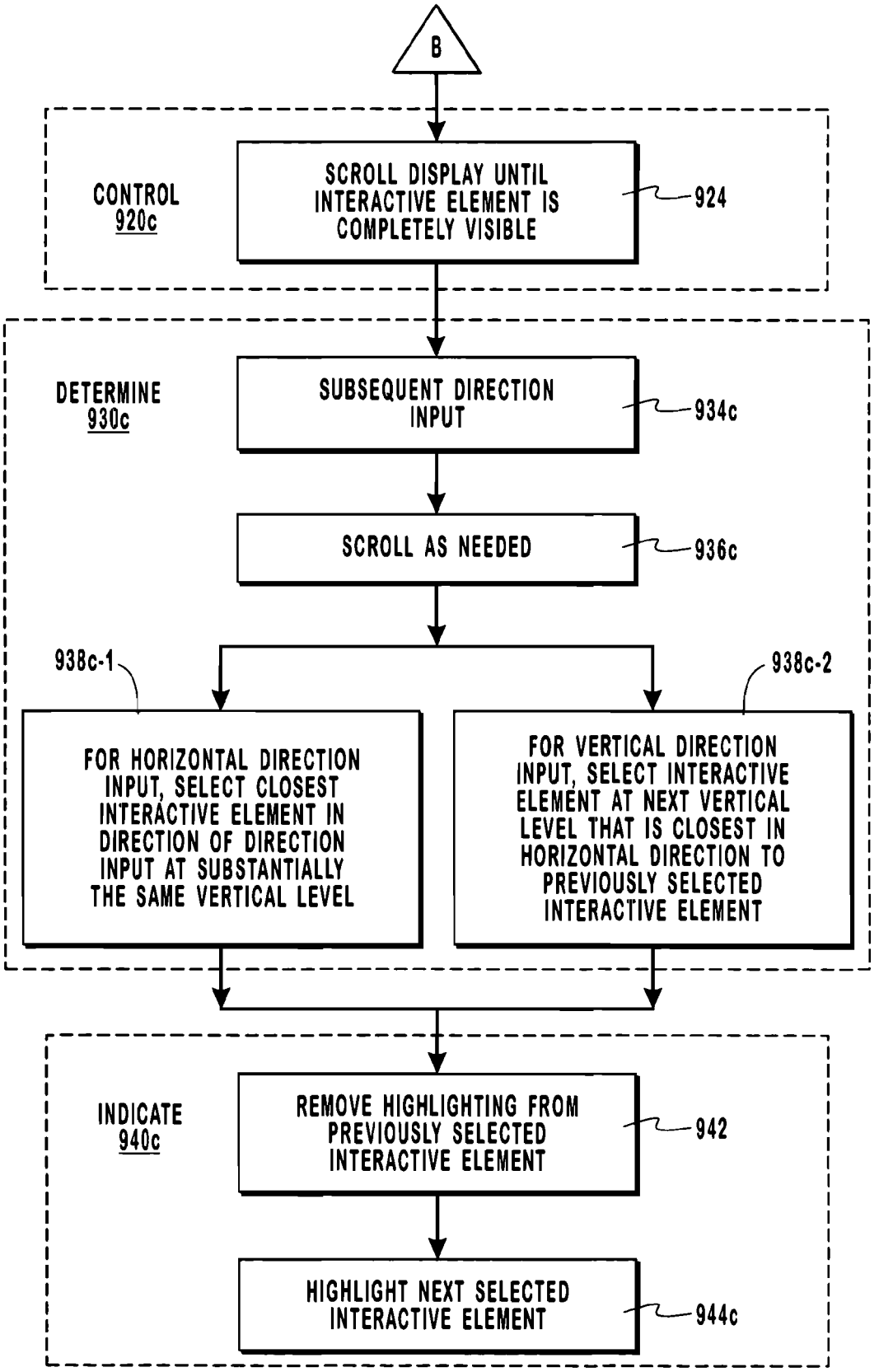

B
CONTROL
920c
SCROLL DISPLAY UNTIL INTERACTIVE ELEMENT IS COMPLETELY VISIBLE
924
DETERMINE
930c
SUBSEQUENT DIRECTION INPUT
934c
SCROLL AS NEEDED
936c
938c-1
938c-2
FOR HORIZONTAL DIRECTION INPUT, SELECT CLOSEST INTERACTIVE ELEMENT IN DIRECTION OF DIRECTION INPUT AT SUBSTANTIALLY THE SAME VERTICAL LEVEL
FOR VERTICAL DIRECTION INPUT, SELECT INTERACTIVE ELEMENT AT NEXT VERTICAL LEVEL THAT IS CLOSEST IN HORIZONTAL DIRECTION TO PREVIOUSLY SELECTED INTERACTIVE ELEMENT
INDICATE
940c
REMOVE HIGHLIGHTING FROM PREVIOUSLY SELECTED INTERACTIVE ELEMENT
942
HIGHLIGHT NEXT SELECTED INTERACTIVE ELEMENT
944c

FIG. 9C

BROWSER NAVIGATION FOR DEVICES WITH A LIMITED INPUT SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation application of commonly-assigned U.S. patent application Ser. No. 09/861,327 filed May 18, 2001, entitled "Browser Navigation for Devices with a Limited Input System". That patent application claims priority to U.S. Provisional Application No. 60/239,600, entitled, "BROWSER NAVIGATION FOR DEVICES WITH LIMITED INPUT MECHANISMS," filed Oct. 11, 2000, both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates to browsing electronic content. More specifically, the present invention relates to methods, systems, and computer program products for browsing content that includes interactive elements using a computerized system with a display area and input system that may be somewhat limited in comparison to the pointing devices and displays typically found in more traditional browsing systems.

2. Background and Relevant Art

Content typically includes interactive elements, such as links and form controls. Activating or following a link causes the content that is associated with the link to be requested and displayed. Selecting a form control allows for interaction with the form control. Traditional browsing systems generally include a keyboard and a pointing device such as a mouse, for activating links and interacting with form controls. Tab order navigation is possible, but may not follow an order expected by the user, especially if scrolling is required to view all of the content.

In traditional browsing systems, a user activates a link or selects a form control by simply placing a mouse pointer over the interactive element and pressing a mouse button. With each mouse press, a user may follow a link, select a text field so that text may be entered from a keyboard, toggle a radio button or checkbox, choose one or more items from a list, or cause the action associated with a button to be executed. The mouse also is used in scrolling the display area, as necessary. Nevertheless, content often is authored to minimize scrolling the display of traditional browsing systems, particularly in the horizontal direction.

Browsing systems with limited input systems and display areas, however, such as a phone having a numeric keypad, a directional control, and an action key, may make it difficult to select and interact with content designed for more traditional browsing systems that make use of pointing devices and have larger display areas. For example, without a pointing device, how are links activated and how are form controls selected? The direction control is a natural choice for scrolling because this operation is similar to many traditional browsing systems. (When no interactive element is selected, arrow keys usually are used for scrolling.) But, without a mouse, selecting individual interactive elements presents a significant challenge.

Tab order navigation does not provide an adequate solution because tab order generally follows the order of interactive elements in the content as authored or written, rather than the order of interactive elements in the content as displayed. Thus, in some situations, tab order moves horizontally, and in other situations, tab order moves vertically. For example, content that includes a table often will have a vertical tab order within individual table cells, but a horizontal tab order from cell to cell. Content outside of a table usually has a horizontal tab order. Because users generally are unaware of whether content includes a table or not, tab order may appear completely arbitrary, moving horizontally one time and vertically the next.

Therefore, when browsing content that includes interactive elements, methods, systems, and computer program products are needed for computerized systems that may have limited display areas and input systems, as compared to the pointing devices and displays typically found in more traditional browsing systems. Furthermore, certain interactive elements may be more intuitive in a browsing context, if those interactive elements operate somewhat differently from how they might function in an operating system shell environment.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a navigation mode and an edit mode for browsing content with a computerized system that may include a somewhat limited display area and/or input system. Navigation mode generally includes movement between and selection of interactive elements, whereas the edit mode generally includes interaction with a single interactive element. In navigation mode, pressing a direction key selects the next interactive element in the direction indicated by the direction key (e.g., up, down, left, right). When moving horizontally, an interactive element is in the direction indicated by the direction control if the interactive element is at substantially the same vertical level. For example, if a later element overlaps a previous element on a given vertical level by any amount, the two elements are considered to be at substantially the same vertical level. Vertical movement is to an interactive element at the next vertical level in the direction indicated by the direction control. If multiple interactive elements lie at the next vertical level, the one closest in the horizontal direction to the beginning of the current interactive element is selected.

To indicate selection, an interactive element is highlighted, such as by placing a selection box around the element. The interactive element remains selected until it is no longer visible (i.e., it has scrolled off the display area) or the next interactive element becomes at least partially visible and is selected. If no interactive element is at least partially visible in the direction indicated or if a selected interactive element is only partially visible, the display scrolls in the direction indicated by the direction control.

Switching from navigation mode to edit mode may be accomplished in several ways. For example, once an interactive element allowing character entry is selected, typing a character on the keypad automatically switches from navigation mode to edit mode. Similar to a mouse click, pressing the action button after an interactive element has been selected also switches to edit mode. Where interactive elements only require a mouse click to function in traditional browsing systems, such as links, checkboxes, radio buttons, other buttons, and the like, pressing action uses the selected control (i.e., follows the link, checks or unchecks a checkbox, chooses a radio button, performs the action associated with the button, etc.) rather than switching to edit mode. In edit mode, pressing the action button switches back to navigation mode. If a particular direction key is not allowed in edit mode, pressing the direction key also will exit edit mode. For forms

that do not include a submit button on the form, pressing the action key will submit the form, rather than switching to navigation mode.

Certain interactive elements may be limited to the width of the display area that is available for displaying content so that the entire element can be visible at one time. Therefore, the width of an interactive element that exceeds the width of available display area may be adjusted to be less than or equal to the width of available display area. If a selected interactive element is only partially visible, switching into edit mode scrolls the control into full view.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered as limiting its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** illustrates an exemplary system that provides a suitable operating environment for the present invention;

FIG. **2** shows a portion of a wireless telephone;

FIG. **3** is a flow diagram that corresponds to receiving a direction input while navigating between interactive elements;

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation;

FIGS. **5**A-**5**H illustrate various interactive elements;

FIG. **6** is a flow diagram that corresponds to receiving input while editing interactive elements;

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements;

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements; and

FIGS. **9**A-**9**C show an exemplary method for browsing content according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention extends to methods, systems, and computer program products for browsing content that includes interactive elements using a browsing system with a display area and input system that may be limited in comparison to the pointing devices and displays typically found in more traditional browsing systems. As used in this application, the term "browsing system" should be interpreted broadly to encompass any computerized system for locating and presenting content, including text, images, audio, video, computer instructions, and the like. With the popularity of the World Wide Web ("Web"), content frequently is formatted using hypertext markup language ("HTML") and computer instructions often are embedded in content using Javascript. Both HTML and Javascript allow for the creation of interactive elements within content.

Those of skill in the art, however, will recognize that a wide variety of markup and scripting languages exist. In particular, extensible markup language ("XML") is becoming increasingly popular because it allows for user-defined extensions to the language. Note also that content may be converted from one language to another. Furthermore, it is anticipated that additional formats, languages, technology and/or standards for authoring content with interactive elements will become available in the future. The present invention, therefore, does not impose any requirements on how content with interactive elements is authored, whether based on current or future technology. Thus, any reference to HTML and/or Javascript, either explicit or implied, should be interpreted as exemplary of aspects or embodiments of the present invention and not as limiting its scope.

Those skilled in the art also will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, mobile/hand-held devices, such as personal digital assistants ("PDAs") and wireless telephones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. The embodiments of the present invention may comprise a special purpose or general purpose computer including various computer hardware, as discussed in greater detail below.

Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that may be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, flash memory, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to oak carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such a connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media. Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions.

FIG. **1** and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although

not required, the invention may be described in the general context of computer-executable instructions, such as program modules, being executed by computers in network environments. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represent examples of corresponding acts for implementing the functions described in such steps.

With reference to FIG. **1**, an exemplary system for implementing the invention comprises a general purpose computing device in the form of a generic computer **120**, including a processing unit **121**, a system memory **122**, and a system bus **123** that couples various system components including the system memory **122** to the processing unit **121**. Although some components of computer **120**, such as monitor **147**, keyboard **140**, and mouse **142**, may seem specific to a conventional computer, those of skill in the art will recognize that analogous components may be found in other computing devices. For example, wireless telephones often include an LCD or plasma display area, a numeric keypad, and one or more navigation buttons. Therefore, any component described with reference to generic computer **120** should be interpreted broadly to encompass analogous components that are appropriate for and consistent with a particular implementation or embodiment of the present invention. In some implementations, a component may be connected only intermittently or may be missing entirely.

The system bus **123** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) **124** and random access memory (RAM) **125**. A basic input/output system (BIOS) **126**, containing the basic routines that help transfer information between elements within the computer **120**, such as during start-up, may be stored in ROM **124**.

The computer **120** may also include a magnetic hard disk drive **127** for reading from and writing to a magnetic hard disk **139**, a magnetic disk drive **128** for reading from or writing to a removable magnetic disk **129**, and an optical disk drive **130** for reading from or writing to removable optical disk **131** such as a CD-ROM or other optical media. The magnetic hard disk drive **127**, magnetic disk drive **128**, and optical disk drive **130** are connected to the system bus **123** by a hard disk drive interface **132**, a magnetic disk drive-interface **133**, and an optical drive interface **134**, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-executable instructions, data structures, program modules and other data for the computer **120**. Although the exemplary environment described herein employs a magnetic hard disk **139**, a removable magnetic disk **129** and a removable optical disk **131**, other types of computer readable media for storing data can be used, including magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs, and the like. Note that decreasing form factors are making it practical to use at least some of the foregoing components with mobile devices. Furthermore, it is anticipated that future technological advances with respect to size, power consumption, and the like, will lead to an increased selection of storage options.

Program code means comprising one or more program modules may be stored on the hard disk **139**, magnetic disk **129**, optical disk **131**, ROM **124** or RAM **125**, including an operating system **135**, one or more application programs **136**, other program modules **137**, and program data **138**. A user may enter commands and information into the computer **120** through keyboard **140**, pointing device **142**, or other input devices (not shown), such as a numeric keypad, directional buttons, pressure-sensitive software keyboard, microphone, joy stick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit **121** through a serial port interface **146** coupled to system bus **123**. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor **147** or another display device, such as an LCD or gas plasma display, is also connected to system bus **123** via an interface, such as video adapter **148**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer **120** may operate in a networked environment using logical connections to one or more remote computers, such as remote computers **149***a* and **149***b*. Remote computers **149***a* and **149***b* may each be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically include many or all of the elements described above relative to the computer **120**, although only memory storage devices **150***a* and **150***b* and their associated application programs **136***a* and **136***b* have been illustrated in FIG. **1**. The logical connections depicted in FIG. **1** include a local area network (LAN) **151** and a wide area network (WAN) **152** that are presented here by way of example and not limitation. Such networking environments are commonplace in office-wide or enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer **120** is connected to the local network **151** through a network interface or adapter **153**. When used in a WAN networking environment, the computer **120** may include a modem **154**, a wireless link, or other means for establishing communications over the wide area network **152**, such as the Internet. The modem **154**, which may be internal or external, is connected to the system bus **123** via the serial port interface **146**. In a networked environment, program modules depicted relative to the computer **120**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing communications over wide area network **152** may be used.

FIG. **2** shows a portion of a wireless telephone. A wireless telephone is merely one example of a browsing system with limited display and input capabilities. Typically, PDAs and other handheld devices also have limited displays and input systems. The present invention, however, is not necessarily limited to any particular hardware or device, handheld or otherwise. Nevertheless, some benefits provided by the present invention may be more pronounced where displays and/or input systems are less robust than corresponding displays and/or input systems found in traditional browsing systems.

Four-direction and action key **210** is an example of both a navigation key generating direction input and an action key providing action input. Depressing key **210** at any one of the four arrows generates a direction input corresponding to the direction of the arrow. An action input is generated by depressing the center of key **210**. The center of key **210** may be a separate button (not shown) or may be integral with the navigation arrows such that depressing the center generates simultaneous, but conflicting direction input. In other words,

simultaneous up and down arrows or simultaneous left and right arrows are interpreted as an action input. Typically, an action input corresponds to pressing an enter key on a keyboard, but other actions are not precluded, as circumstances may warrant.

A user may enter characters, such as numbers, letters, punctuation, etc., with keypad **220**. Audio input **230** is the mobile telephone's mouthpiece. Display area **240** displays content received while browsing. Note however, that all of display area **240** may not be available for displaying content. For example, portions of display area **240** may be used for titles, menus, switching between tasks, etc. Therefore, available display area generally refers to the portion of display area **240** that is devoted to displaying content, and may represent all or less than all of display area **240**.

As noted previously, the present invention provides for a navigation mode and an edit mode. Edit mode generally is characterized by interaction with a single interactive element and is described more fully in connection with FIG. **6** and the interactive elements shown in FIGS. **5**A-**5**H. In contrast, navigation mode generally is characterized by movement between and selection of interactive elements (i.e., changing focus from one interactive element to another) and is described more filly in connection with FIGS. **3** and **4**. Transitions from navigation mode to edit mode and interactive elements that operate in an intuitive manner without using a dedicated edit mode are covered in the description of FIGS. **7** and **8**.

Turning first then to FIG. **3**, during operation in navigation mode **310**, a direction input **320** is received. Decision block **330** determines if an interactive element is at least partially visible in the direction of direction input **320**, either relative to the beginning of the content if no interactive element has been selected or relative to an interactive element that is selected currently. If no interactive element is visible in the direction of the direction input, decision block **340** determines if more content is available in the direction of the direction input. If more content is available in the direction of the direction input, the display scrolls **342** in the direction of the direction input; otherwise, the direction input is ignored **344**.

Returning to decision block **330**, if an interactive element is visible in the direction of direction input **320**, selecting the next interactive element depends on the direction of direction input **320**, unless no interactive element has been selected previously, wherein the interactive element closest to the beginning of the content is selected (not shown). For horizontal input, decision block **360** determines if the visible interactive element lies at substantially the same vertical level as the interactive element that is selected currently. (The meaning of substantially the same vertical level will be described in more detail below, with respect to FIG. **4**.) If substantially at the same level, the interactive element in the direction of the direction input is selected **362**. If the visible interactive element does not lie at the substantially the same vertical level, operation continues with decision block **340**, as described above. For vertical direction input, the interactive element at the next vertical level in the direction of direction input **320** that is closest in the horizontal direction to the beginning of the previously selected interactive element is selected **352**.

FIG. **4** shows several interactive elements and their positions relative to each other for use in describing the selection of interactive elements during navigation. The display **400** of content includes vertical levels **410**, **420**, **430**, and **440**. Note that at vertical level **410**, Element **1**, Element **2**, and Element **3** do not display at exactly the same vertical coordinates. Nevertheless, Element **1**, Element **2**, and Element **3** lie at substantially the same vertical level. By allowing for some variation in the vertical display coordinates for interactive elements, navigation is more intuitive.

In particular, the height of Element **1** is h**1** and the height of Element **3** is h**3**. The distance y**1** is the amount that Element **3** overlaps with Element **1**. Alternatively, y**1** may represent the amount of vertical separation between interactive elements rather than the amount of overlap. Note that the present invention does not necessarily limit y**1** to any particular dimension or calculation. Because y**1** reflects the expectations of users, it is possible for y**1** to take on a wide range of values, as may be suitable for a particular embodiment or implementation. However, in at least some circumstances, y**1** is a portion of h**3**, the height of Element **3**, indicating that Element **3** partially overlaps Element **1**. (Although not shown, note also that it may be possible for a single element to span and be reachable from multiple vertical levels. When navigating horizontally from an element spanning multiple vertical levels, the next element is selected from the top most spanned vertical level where an interactive element is visible.) The height of Element **2** is not shown because it overlaps completely with Element **1** and therefore is clearly at the same vertical level as Element **1**.

Initially no interactive element is selected. As indicated with respect to FIG. **3**, a direction input that is not in the direction of an interactive element that is at least partially visible or in a direction that permits scrolling will be ignored (see block **344**). Thus, direction input up or to the left will be ignored. Because initially no interactive element is selected, however, a down direction input or a right direction input will select Element **1** (i.e., the first interactive element relative to the beginning of the content). Once selected, an interactive element is highlighted, such as by drawing a dashed box around the element to indicate that the selected interactive element has focus. The present invention does not necessarily limit the type of highlighting used to indicate selection. It is only relevant for some form of visual cue to occur that is specific to the selected interactive element.

Left and right direction input will select interactive elements in numerical order, either ascending for right direction input or descending for left direction input. Up and down direction input is somewhat more complicated. Beginning with Element **1**, down direction input selects interactive elements in the following order: Element **1**, Element **4**, Element **6**, Element **7**. Beginning with Element **7**, up direction input selects interactive elements in reverse order: Element **7**, Element **6**, Element **4**, Element **1**.

Moving from vertical level **430** to vertical level **420** may include some ambiguity because vertical level **420** includes multiple interactive elements. Selecting between Element **4** and Element **5** depends on the horizontal distances labeled x**1** and x**2**. The distance x**1** represents the horizontal distance from the beginning of Element **6** (the currently selected interactive element when moving from vertical level **430** to vertical level **420**) to the nearest portion of Element **4**. Likewise, the distance x**2** represents the horizontal distance from the beginning of Element **6** to the nearest portion of Element **5**. When selection moves in the vertical direction and more than one interactive element lies at a vertical level, the interactive element closest in the horizontal direction to the beginning of the previously selected interactive element is selected next. In other words, Element **4** is selected if x**1** is less than or equal to x**2**, and Element **5** is selected if x**2** is less than x**1**. Note that the same processing occurs for moving between Element **1** and Element **4**, but the result is obvious. In contrast, there is no ambiguity in moving from Element **4** to Element **6** and then to Element **7**.

FIGS. **5**A-**5**H illustrate various interactive elements, the operation of which will be described in greater detail with respect to FIGS. **6**-**8**. Those of skill in the art will recognize that the interactive elements shown in FIGS. **5**A-**5**H are merely exemplary of interactive elements that are useful in describing embodiments of the present invention in the context of HTML content. However, as explained previously, the present invention is not necessarily limited to any particular types of interactive elements or any particular type of content. For example, JAVA and Javascript allow for the creation of interactive elements and show that the types of interactive elements within the scope of the present invention are governed only by the creativity of those who author content. In addition to existing interactive elements, therefore, it is fully anticipated that new interactive elements will be developed and should be considered to fall within the meaning of interactive elements as used in this application, regardless of the particular technology that implements and/or deploys a particular interactive element. Furthermore, it should be apparent that the following discussion does not catalog all existing interactive elements, but rather, identifies a sufficient number to adequately describe how the present invention operates.

FIG. **5**A illustrates single line textbox **510** for entry of characters. Although display is limited to a single line, characters within the textbox may allow for scrolling if character entry exceeds the width of textbox **510**. FIG. **5**B illustrates multiple line textbox **520**. Multiple line textbox **520** is also for character entry. Due to display area constraints, multiple line textbox **520** may display as a single line in navigation mode, and then to facilitate editing, expand to a multiple line display in edit mode. A close button may be included with multiple line textbox **520** to assist in returning to navigation mode. Like single line textbox **510**, multiple line textbox **520** may allow for scrolling if character entry exceeds the width of textbox **520**.

Radio button **530**, with button **532** and text **534**, is illustrated in FIG. **5**C. Radio buttons allow for choosing one item and only one item from a group or list of items. FIG. **5**D illustrates checkbox **540**, with button **542** and text **544**. In contrast to radio button **530**, checkbox **540** allows for selecting zero or more items from a group or list of items. FIG. **5**E illustrates spinner **550**, with text **552**, left arrow **556**, and right arrow **554**. Similar to radio buttons, spinner **550** groups related items and allows one and only one to be chosen. Activating left arrow **556** chooses the previous item in the list and activating right arrow **554** chooses the next item in the list. The list may be circular, such that moving through all choices with either arrow returns to the initial choice. Alternatively the list may be linear, having a starting point that may be reached with left arrow **556** and having an ending point that may be reached with right arrow **554**. Picker **560**, with text **562** and right arrow **564**, as illustrated in FIG. **5**F, is similar to checkbox **540**. Activating right arrow **564** displays a list of checkbox options. Like multiple line textbox **520**, a close button may be included with picker **560** to facilitate returning to navigation mode. Activating button **570** of FIG. **5**G causes an action associated with the button to be executed. FIG. **5**H illustrates link **580**, a hypertext markup language link that browses content associated with the link when the link is activated or followed.

FIG. **6** is a flow diagram that corresponds to receiving input while editing a selected interactive element, such as one of those described above in connection with FIGS. **5**A-**5**H. Depending on the selected interactive element, input received while in edit mode may be used by the interactive element (e.g., entering characters into a textbox) or may cause a return to navigation mode (e.g., so that another interactive element may be selected). Following the discussion of edit mode and FIG. **6**, the description of FIGS. **7** and **8** explains how transitions to edit mode are made from the navigation mode identified above, with respect to FIGS. **3** and **4**.

Turning now then to FIG. **6**, during operation in edit mode **610**, an input **630** is received. If the selected interactive element is only partially visible, entering edit mode will scroll **620** the display until the selected element is completely visible. Decision block **640** determines if input **630** is a direction input. If so, decision block **650** determines if the selected interactive element accepts direction input so that the direction input may be processed **652**. For example, direction input may be accepted in single line textbox **510** and multiple line textbox **520** for moving the cursor position, although up and down direction input would not be accepted by single line textbox **510**. Spinner **550** also may accept direction input, with a left direction input choosing a previous item in the spinner list and a right direction input choosing the next item in the spinner list. If decision block **650** determines that the selected interactive element does not accept direction input, operation transitions or switches to navigation mode **654**.

If input **630** is not a direction input, decision block **660** determines if input **630** is an action input. If not, input **630** is processed as character input **662**. Note that FIG. **6** suggests three basic types of input: direction input, action input, and character input. However, the present invention does not necessarily require dividing all input into any particular number of categories. It is expected, therefore, that alternate embodiments may use additional, fewer, or other categories, depending on the needs or preferences of a particular application. Furthermore, alternate embodiments also may include additional, fewer, or other modes of operation, again depending on the needs or preferences of the particular application.

Processing character input depends on the selected interactive element. Ordinarily, character input has greatest application in entering information into single line textbox **510** and multiple line textbox **520**. However, character input also may be used in finding a particular entry in spinner **550** and picker **560**. For example, if spinner **550** or picker **560** are used to chose a state based on state codes, entering a "W" may immediately move to the end of the list, as opposed to moving through a list item by item or page by page as would likely occur using a direction input.

If decision block **660** determines that input **630** is an action input, decision block **670** considers whether the content being browsed is form content without a submit button. Some form content may fail to provide an explicit submit button, in which case an action input is interpreted as a request to submit the form **672**. In the usual case, however, an action input switches from edit mode back to navigation mode **674**. For spinner **550**, switching from edit mode to navigation mode **674** is somewhat different behavior than occurs in an operating system shell context. More specifically, in an operating system shell context, once a spinner has been selected, an action input ordinarily displays a list of radio button options.

FIG. **7** is a flow diagram that corresponds to receiving an action input while navigating between interactive elements. During operation in navigation mode **710**, an action input **720** is received. If decision block **730** determines that the selected interactive element is a link, receiving an action input activates or follows the link **732**. If the selected interactive element is a radio button or checkbox, as determined in decision block **740**, the state of the radio button or checkbox is switched. Radio buttons and checkboxes are both examples of interactive elements capable of representing two states.

Because multiple checkboxes may be chosen, switching the state of a checkbox means either (i) an unchecked check-

box is checked, or (ii) a checked checkbox is unchecked. Radio buttons are slightly more complex to explain since only one item may be chosen at any given time. Therefore, switching the state of a radio button means that (i) if the radio button was not chosen previously, the radio button will be chosen and any other radio button that may have been chosen previously will no longer be chosen, or (ii) a radio button that was previously chosen remains chosen.

If the selected interactive element is a button, as determined in decision block **750**, the action associated with the button is executed or performed **752** upon receiving an action input **720**. Note that links, radio buttons, checkboxes, and buttons are examples of interactive elements that operate in an intuitive manner without using a dedicated edit mode. For other interactive elements, such single line textbox **510**, multiple line textbox **520**, spinner **550**, and picker **560**, an action input switches to edit mode **754**.

FIG. **8** is a flow diagram that corresponds to receiving a character input while navigating between interactive elements. During operation in navigation mode **810**, a character input **820** is received. Character input may include numbers, letters, punctuation, white space, etc. If decision block **830** determines that the interactive element does not accept character input, character input **820** is ignored **836**. Otherwise, character input **820** causes a switch from navigation mode to edit mode **832** and character input **820** is processed **834**. For example, if the selected interactive element is a single line textbox **510** or a multiple line textbox **520**, entering a letter while in navigation mode switches to edit mode and places the letter in the textbox. As indicated earlier, spinner **550** and picker **560** also may use character input to find an item in a long list by advancing to and/or choosing an item that matches the character entered.

FIGS. **9**A-**9**C show an exemplary method for browsing content according to the present invention. A step for presenting (**910**) at least a portion of content on the display area of a browsing system may include the acts of retrieving (**912**) the content and displaying (**914**) the retrieved content. The content may be retrieved from a local or remote source. A step for controlling the width (**920***a*) of an interactive element may include the act of adjusting the width of the interactive element to the width of available display area on the browsing system if the width of the interactive element exceeds the width of available display area.

A step for determining (**930***a*) an interactive element for selection based on a 2 direction input may include the following acts: an act of starting (**932**) in navigation mode by default when content displays; an act of receiving (**934***a*) a direction input from a direction key, such as four-direction and action key **210** of FIG. **2**; an act of scrolling (**936***a*) the display of the content in the direction of the received direction input if less than all of the content is displayed and no interactive element is at least partially visible; and an act of selecting (**938***a*) an interactive element based on the received direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed content. Note that for English content, an up arrow or left arrow relative to the beginning of the display content does not make sense and therefore is ignored.

A step for indicating (**940***a*) that an interactive element is selected may include the act of highlighting (**944***a*) the interactive element. For example, a selection box may be placed around the interactive element or the visual appearance of the interactive element may be otherwise altered such that a selected interactive element is distinguishable from an interactive element that has not been selected. The present invention does not necessarily require any particular type of highlighting.

A step for changing (**950**) the mode of a browsing system may include an act of switching (**951**) from navigation mode to edit mode upon receiving a character input. For example, when an interactive element such as single line textbox **510**, multiple line textbox **520**, spinner **550**, or picker **560** is selected, receiving a character input switches to edit mode. An act of switching (**953**) from navigation mode to edit mode upon receiving an action input also may be included as part of a step from changing (**950**) the mode of a browsing system. An action input is an explicit indication to switch modes, whereas a character input is an implied indication to switch modes.

Once in edit mode, a step for changing (**950**) the mode of a browsing system may include the act of switching (**955**) from edit mode to navigation mode upon receiving an action input. In other words, an action input may be used both for entering and exiting edit mode. Likewise, an act of switching (**957**) from edit mode to navigation mode upon receiving a direction input also may be included within a step form changing (**950**) the mode of a browsing system. For interactive elements that do not accept a particular direction input, such as an up or down arrow in a single line textbox, receiving the particular direction input switches from edit mode to navigation mode.

Additionally, a step for changing (**950**) the mode of a browsing system may include the act of submitting (**959**) form content upon receiving an action input. For content that does not include a submit button, an action input is associated with submitting the form. After submitting a form, the browsing system switches from edit mode to navigation mode because submitting a form usually causes new content to be displayed by a browsing device and, as described above, the browsing system may default to navigation mode as content initially displays.

Some types of interactive elements may operative intuitively without a dedicated edit mode and therefore switching modes may not be necessary for interacting with those elements. The present invention may include the act of switching (**960**) the state of a selected interactive element such as a radio button or checkbox. Similarly, the act of following (**970**) a selected link upon receiving an action input or executing the action (not shown) associated with a button also are within the scope of the present invention.

If selected interactive element is only partially visible, a step for controlling the width (**920***c*) of an interactive element (see **920***a* of FIG. **9**A) also may include the act of scrolling (**924**) the display area until the interactive element is completely visible. Automatically scrolling the display of content ordinarily occurs when switching from navigation mode to edit mode. Upon switching from navigation mode to edit mode, it becomes clear that a particular interactive element is of interest and should be completely visible, whereas in navigation mode, it may be unclear whether interest is in (i) an interactive element selected as a natural consequence of scrolling displayed content, or (ii) other content that becomes visible as content scrolls.

In addition to the acts described in connection with FIG. **9**A, a step for determining (**930***c*) an interactive element for selection based on a direction input may include other acts, such as an act of receiving (**934***c*) a subsequent direction input. If less than all of the content is display and no interactive element is at least partially visible, determining (**930***c*) an interactive element for selection also may include an act of scrolling (**936***c*) the display of the content in the direction of the subsequent direction input. For a horizontal direction

input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**1**) the closest interactive element in the direction of the direction input that is at substantially the same vertical level. For a vertical direction input, determining (**930***c*) an interactive element for selection may include an act of selecting (**938***c*-**2**) the interactive element at the next vertical level in the direction of the direction input that is closest in the horizontal direction to a previously selected interactive element. A step for indicating (**940***c*) that an interactive element is selected may further include the acts of removing (**942**) the highlighting from a previously selected interactive element and highlighting (**944***c*) the next selected interactive element (see also **940***a* of FIG. **9**A).

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed and desired to be secured by United States Letters Patent is:

1. A computer program product for use in either a wireless telephone or personal digital assistant having a display area and input system, the input system including a direction key and an action key, wherein the input system and display area are limited as compared to a pointing device and larger display area often found in more traditional browsing systems, and wherein one or more interactive elements within content received from a content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, the computer program product comprising one or more computer-readable storage media having stored computer-executable instructions for implementing a method of browsing content that contains one or more interactive elements, wherein the browsing includes an edit mode and a navigation mode, the method comprising acts of:

starting in the navigation mode;

displaying at least a portion of the content on the wireless telephone or personal digital assistant display area;

receiving a direction input generated by the direction key of the wireless telephone or personal digital assistant;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input;

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein said interactive element can be only partially visible; and

placing a selection box around the interactive element to indicate that the interactive element is selected.

2. A computer program product as recited in claim **1**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

3. A computer program product as recited in claim **1**, wherein the interactive element accepts character input, the method further comprising an act of switching from navigation mode to edit mode upon receiving a character input.

4. A computer program product as recited in claim **1**, wherein the interactive element accepts character input or allows for one or more items to be selected from a group of one or more items, the method further comprising an act of switching from navigation mode to edit mode upon receiving an action input.

5. A computer program product as recited in claim **1**, wherein the method further comprises the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

switching from edit mode to navigation mode upon receiving an action input.

6. A computer program product as recited in claim **1**, wherein the interactive element does not accept a direction input, the method further comprising the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

switching from edit mode to navigation mode upon receiving the direction input.

7. A computer program product as recited in claim **1**, wherein the interactive element is part of form content that does not include a submit element, the method further comprising the acts of:

browsing with the wireless telephone or personal digital assistant in edit mode; and

submitting the form content upon receiving an action input.

8. A computer program product as recited in claim **1**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input while in navigation mode.

9. A computer program product as recited in claim **1**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

10. A computer program product as recited in claim **1**, wherein the interactive element exceeds the width of available wireless telephone or personal digital assistant display area, the method further comprising an act of adjusting the width of the interactive element to be less than or equal to the width of available wireless telephone or personal digital assistant display area.

11. A computer program product as recited in claim **1**, wherein the interactive element is only partially visible in the wireless telephone or personal digital assistant display area, the method further comprising acts of:

adjusting the width of the interactive element to be less than or equal to the width of available wireless telephone or personal digital assistant display area if the width of the interactive element exceeds the width of available wireless telephone or personal digital assistant display area; and

scrolling the wireless telephone or personal digital assistant display area until the interactive element is completely visible.

12. A computer program product as recited in claim **1**, wherein the selected interactive element is a previously selected interactive element, the method further comprising the acts of:

receiving a subsequent direction input that corresponds to scrolling the wireless telephone or personal digital assis-

tant display area, the subsequent direction input being generated by activating a navigation key;

while the subsequent direction input is being received, if less than all of the content is displayed and no other interactive element is at least partially visible in the direction of the subsequent direction input, scrolling the display of the content in the direction of the subsequent direction input;

selecting a next interactive element that is at least partially visible, the selection being based on the subsequent direction input relative to the previously selected interactive element;

removing highlighting from the previously selected interactive element to indicate that the previously selected interactive element is no longer selected; and

highlighting the next interactive element to indicate that the next interactive element is selected.

13. A computer program product for use in either a wireless telephone or personal digital assistant configured for browsing content received from a content source, and having a display area and input system that is limited as compared to pointing devices and displays often found in more traditional browsing systems, and wherein one or more interactive elements within content received from the content source may behave differently in a browsing context than the one or more interactive elements behave in an operating system shell context, the computer program product comprising one or more computer-readable storage media having stored computer-executable instructions for implementing a method of browsing content that includes one or more interactive elements, wherein the browsing includes an edit mode and a navigation mode, the method comprising steps for:

presenting at least a portion of the content on a display area of a browsing system, the content including at least one interactive element which is displayed in such a manner that it is only partially visible;

receiving a direction input generated by activating a navigation key;

upon receiving a direction input, determining that the interactive element, which is only partially visible, has been selected based on the direction input;

visually indicating that the interactive element, which is only partially visible, is selected; and

automatically scrolling the browsing system upon determining that the interactive element has been selected and that the interactive element is only partially visible in the browsing system display area, wherein the browsing system display area is automatically scrolled until the entire interactive element becomes completely visible.

14. A computer program product as recited in claim **13**, wherein the browsing system comprises one of a wireless telephone and a personal digital assistant.

15. A computer program product as recited in claim **13**, wherein the interactive element comprises one of a link, single line textbox, a multiple line textbox, a spinner, a radio button, a checkbox, a button, and a picker.

16. A computer program product as recited in claim **13**, further comprising a step for changing the mode of the browsing system.

17. A computer program product as recited in claim **13**, wherein the interactive element is capable of representing two states, the method further comprising an act of switching from one state to the other upon receiving an action input while in navigation mode.

18. A computer program product as recited in claim **13**, wherein the interactive element comprises a link, the method further comprising an act of following the link upon receiving an action input.

19. A computer program product as recited in claim **13**, wherein the interactive element exceeds the width of available browsing system display area, the method further comprising a step for controlling the width of the interactive element.

20. A computer program product as recited in claim **13**, wherein the step for determining an interactive element for selection based on a direction input comprises acts of:

receiving a direction input that corresponds to scrolling the browsing system display area, the direction input being generated by activating a navigation key;

while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible, scrolling the display of the content in the direction of the direction input; and

selecting an interactive element that is at least partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content.

21. A computer program product for use in either a wireless telephone or personal digital assistant having a display area and input system that are limited as compared to a pointing device and larger display area often found in more traditional browsing systems, the computer program product comprising one or more computer-readable storage media having stored computer-executable instructions for implementing a method of browsing content that contains one or more interactive elements, the method comprising:

displaying at least a portion of content on the display area of either a wireless telephone or personal digital assistant;

receiving a direction input generated by the wireless telephone or personal digital assistant;

in a situation in which no interactive element is displayed as being visually selected on the display area, and while the direction input is being received, if less than all of the content is displayed and no interactive element is at least partially visible in the direction of the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, automatically scrolling the display of the content in the direction of the direction input;

selecting an interactive element that is only partially visible, the selection being based on the direction input relative to a previously selected interactive element or, if no interactive element has been previously selected, based on the direction input relative to the beginning of the displayed portion of the content, wherein said interactive element is only partially visible; and

graphically indicating that the interactive element is selected.

* * * * *

US006256642B1

(12) **United States Patent**
Krueger et al.

(10) **Patent No.: US 6,256,642 B1**
(45) **Date of Patent: Jul. 3, 2001**

(54) **METHOD AND SYSTEM FOR FILE SYSTEM MANAGEMENT USING A FLASH-ERASABLE, PROGRAMMABLE, READ-ONLY MEMORY**

(75) Inventors: **William J. Krueger**, Redmond; **Sriram Rajagopalan**, Bellevue, both of WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **07/828,763**

(22) Filed: **Jan. 29, 1992**

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/205**
(58) **Field of Search** 395/600; 370/82; 707/205

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,408,273		10/1983	Plow	364/200
4,435,752	*	3/1984	Winkelman	395/600
4,507,752		3/1985	McKenna et al.	364/900
4,584,644		4/1986	Larner	364/200
4,606,002		8/1986	Waisman et al.	364/200
4,630,234		12/1986	Holly	364/900
4,704,678		11/1987	May	364/200
4,939,598	*	7/1990	Kulakowski et al.	360/48
4,942,541	*	7/1990	Hoel et al.	364/519
4,953,080		8/1990	Dysart et al.	364/200
4,953,122		8/1990	Williams	364/900
4,989,132		1/1991	Mellender et al.	364/200
5,025,367		6/1991	Gurd et al.	364/200
5,029,125	*	7/1991	Sciupac	395/600
5,060,147		10/1991	Mattheyses	364/200
5,115,504		5/1992	Belove et al.	395/600
5,132,853	*	7/1992	Kulakowski	369/44.27
5,161,256	*	11/1992	Iijima	902/26
5,229,992	*	7/1993	Jurkevich et al.	370/82
5,247,516	*	9/1993	Bernstein et al.	370/82
5,247,658	*	9/1993	Barrett et al.	707/1
5,257,141	*	10/1993	Matsumi et al.	360/32
5,268,870	*	12/1993	Harari	365/218
5,392,427	*	2/1995	Barrett et al.	395/600

FOREIGN PATENT DOCUMENTS

0 251 461	1/1988	(EP) .
0 251 461 A3	1/1988	(EP) G06F/12/02
0 389 396	9/1990	(EP) .
0 389 396 A2	9/1990	(EP) G06F/3/06
0 439 920	8/1991	(EP) .
0 439 920 A3	8/1991	(EP) G06F/12/02

OTHER PUBLICATIONS

Gait et al., "The Optical File Cabinet: A Random–Access File System for Write–Once Optical Disks," *IEEE Computer,* 21(6), 11–22 (1988).
Garfinkel, "Designing a Write–Once File System," *Dr. Dobb's Journal,* 16(1), 78–86 (1991).

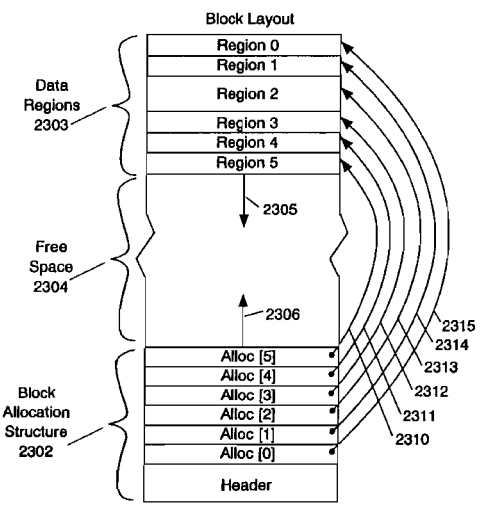
(List continued on next page.)

Primary Examiner—Wayne Amsbury
(74) *Attorney, Agent, or Firm*—Leydig, Voit & Mayer Ltd.

(57) **ABSTRACT**

A method and system for memory management of a block-erasable flash-EProm. The system comprises a FEProm manager and a file system. The FEProm manager manages memory allocation and deallocation of the FEProm. The file system is a hierarchical directory system and uses the FEProm manager to allocate and deallocate memory. In a preferred embodiment, the FEProm manager of the present invention provides for allocation of free space, deallocation of allocated space, and reclamation of deallocated space in a block-erasable FEProm. Each block of the FEProm contains a block allocation structure, data regions, and free space. The block allocation structure contains an allocation array which describes the allocation of the data region.

13 Claims, 33 Drawing Sheets



OTHER PUBLICATIONS

Lahti et al., "Store Date in a Flash," *Byte,* Nov. 1990, 311–318.

Zales et al., "Intel Flash EPROM for In–System Reprogrammable Nonvolatile Storage," *Microprocessors and Microsystems,* 14(8), 543–549 (1990).

Robert L. Kruse; "Data Structures & Program Design"; *Prentice–Hall, Inc.;* 1984; Pp. 40–83.

Doug Cooper and Michael Clancy; "Oh! Pascal!"; 1982; Pp. 475–523.

Simson Garfinkel; "Designing a Write–Once File System"; *Dr. Dobb's Journal;* vol. 16, No. 1; Jan. 1991; Pp. 78, 80, and 82–86.

Phillip L. Barrett et al., U.S. Patent application No. 07/430,746, filed Oct. 31, 1989, for A Method and System for File System Management Using Flash–Erasable, Programmable, Read–Only Memory, pp. 1–27 and Figs. 1A–22.

* cited by examiner

Figure 1A
Prior Art

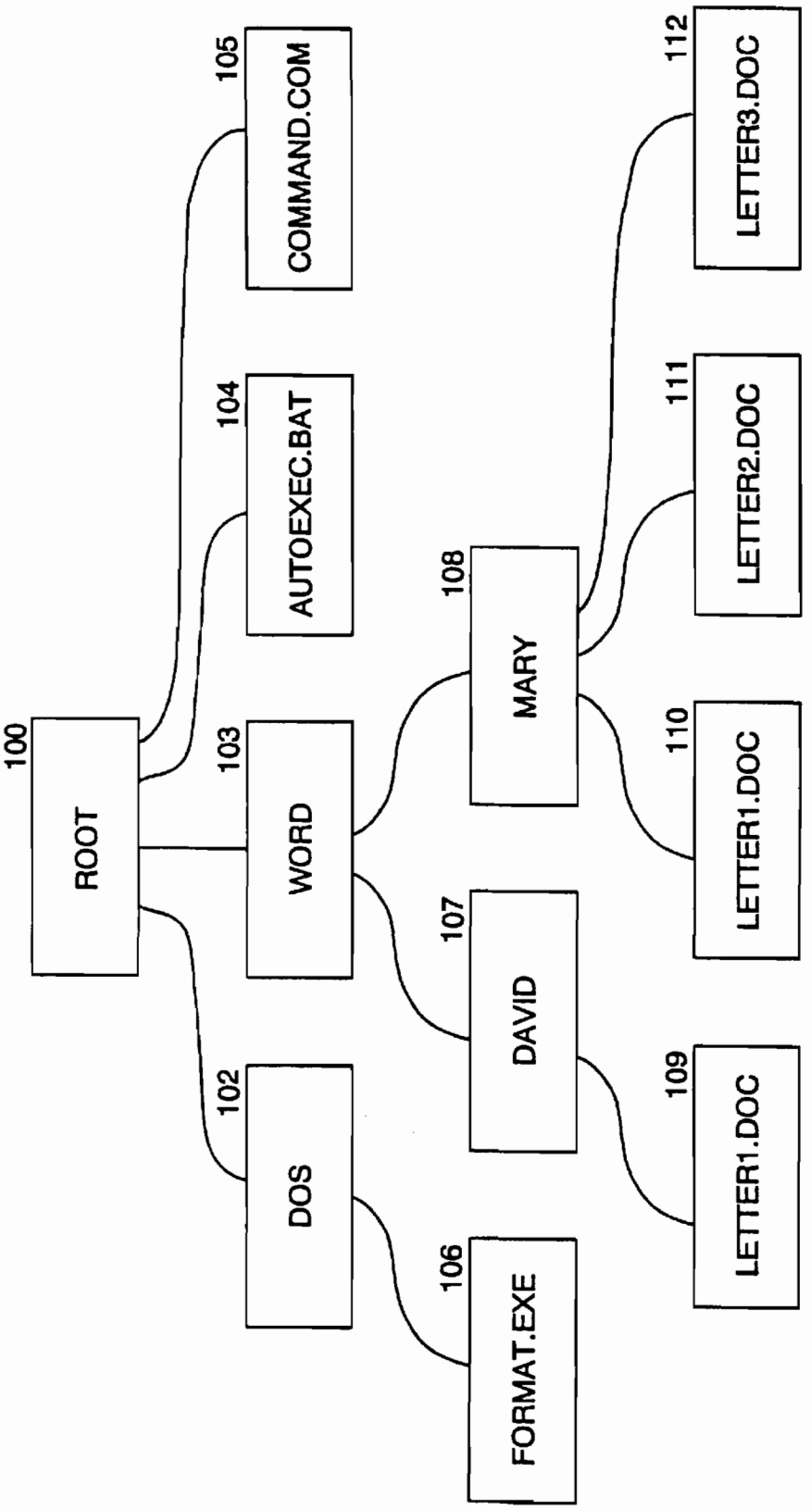


Figure 1B

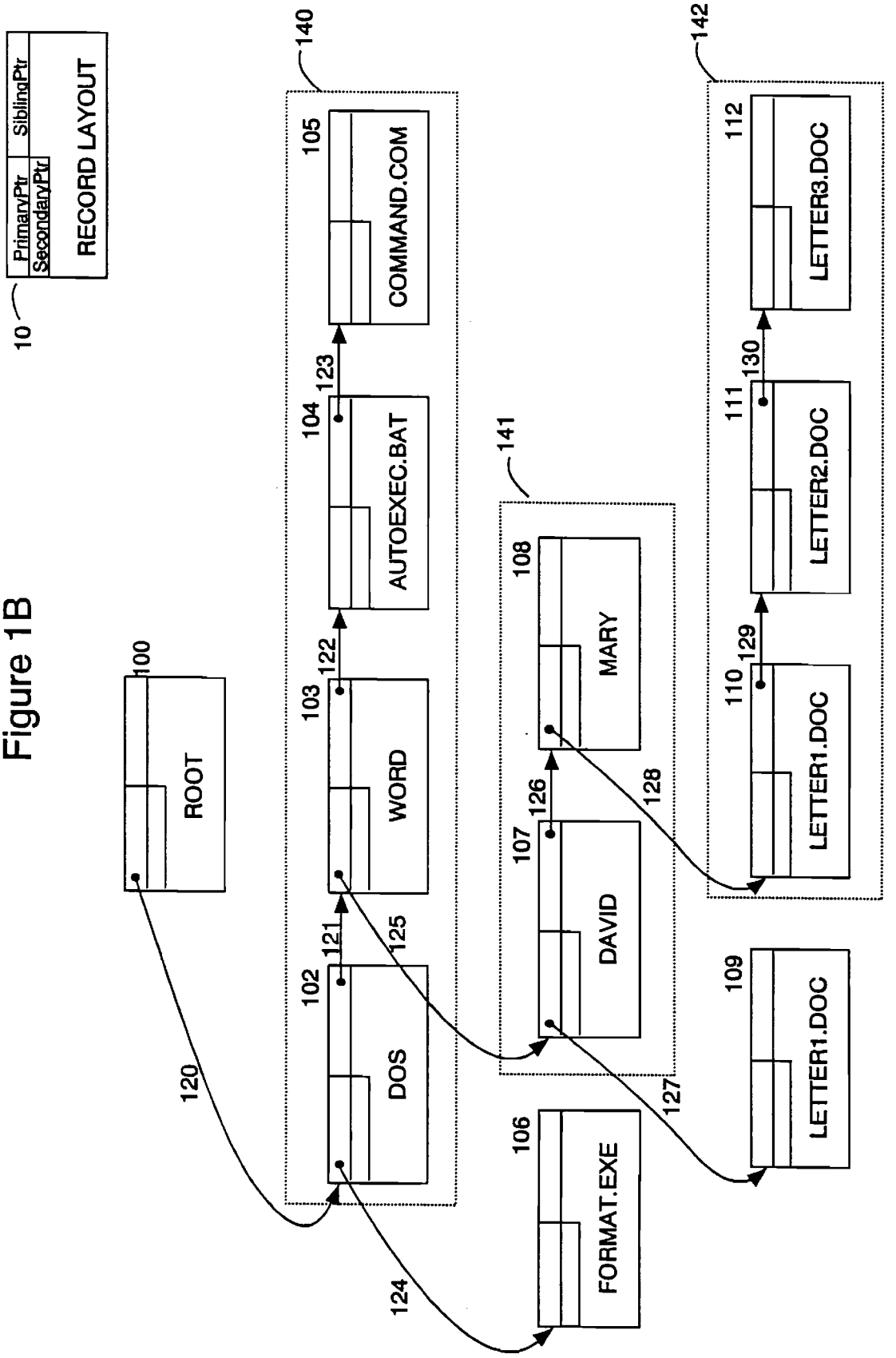

10
PrimaryPtr
SiblingPtr
SecondaryPtr
RECORD LAYOUT
100
ROOT
140
102
DOS
103
WORD
104
AUTOEXEC.BAT
105
COMMAND.COM
120
121
122
123
124
125
126
127
128
141
106
FORMAT.EXE
107
DAVID
108
MARY
109
LETTER1.DOC
110
LETTER1.DOC
111
LETTER2.DOC
112
LETTER3.DOC
129
130
142

Figure 1C

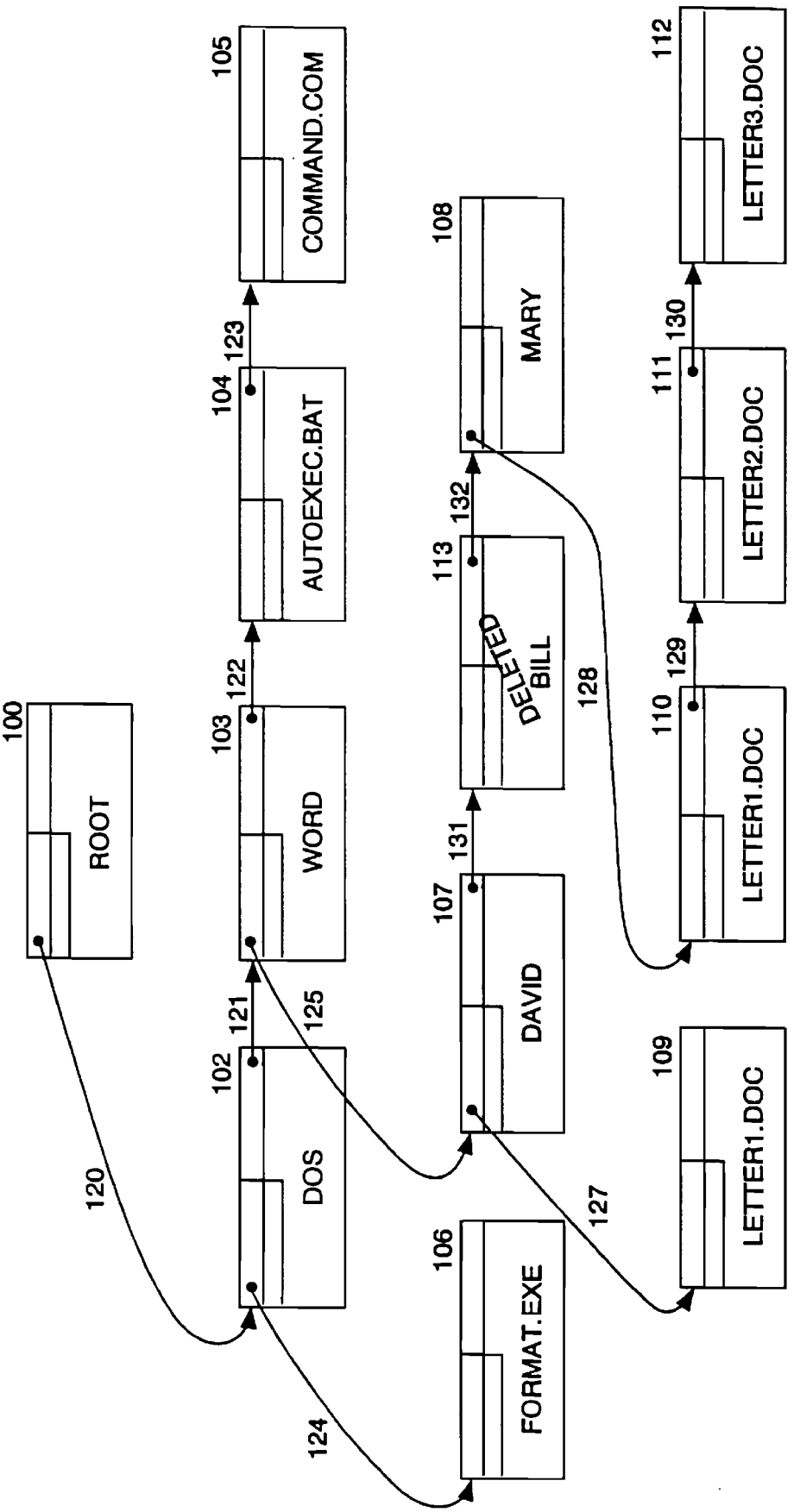

100
ROOT
103
WORD
122
104
AUTOEXEC.BAT
123
105
COMMAND.COM
102
DOS
121
125
107
DAVID
131
113
DELETED BILL
132
108
MARY
106
FORMAT.EXE
120
124
127
128
110
LETTER1.DOC
129
111
LETTER2.DOC
130
112
LETTER3.DOC
109
LETTER1.DOC

Figure 2A

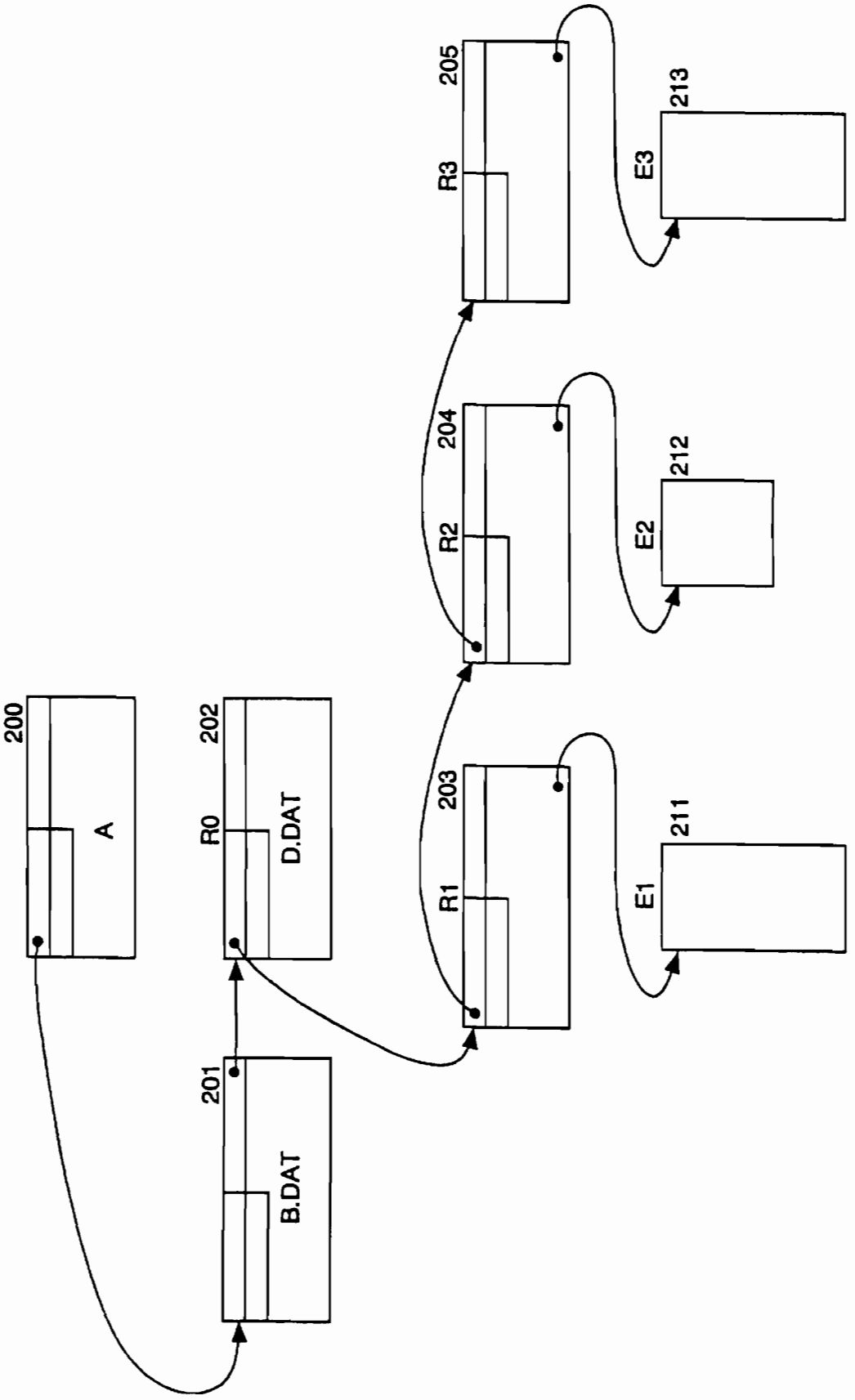

200
A
202
R0
D.DAT
201
B.DAT
203
R1
204
R2
205
R3
211
E1
212
E2
213
E3

Figure 2B

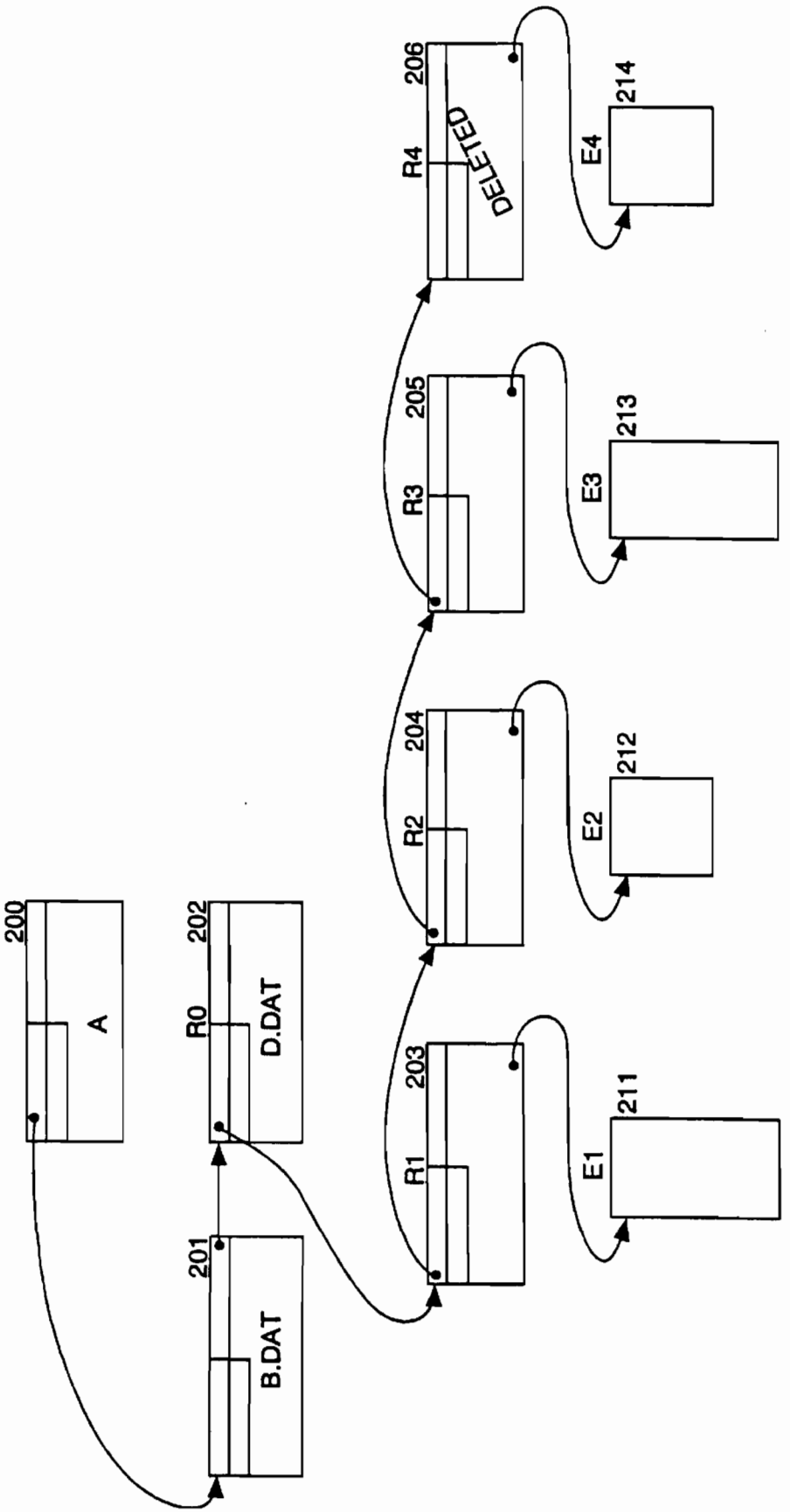

200
A
202
R0
D.DAT
201
B.DAT
203
R1
E1
211
204
R2
E2
212
205
R3
E3
213
206
R4
DELETED
E4
214

Figure 26

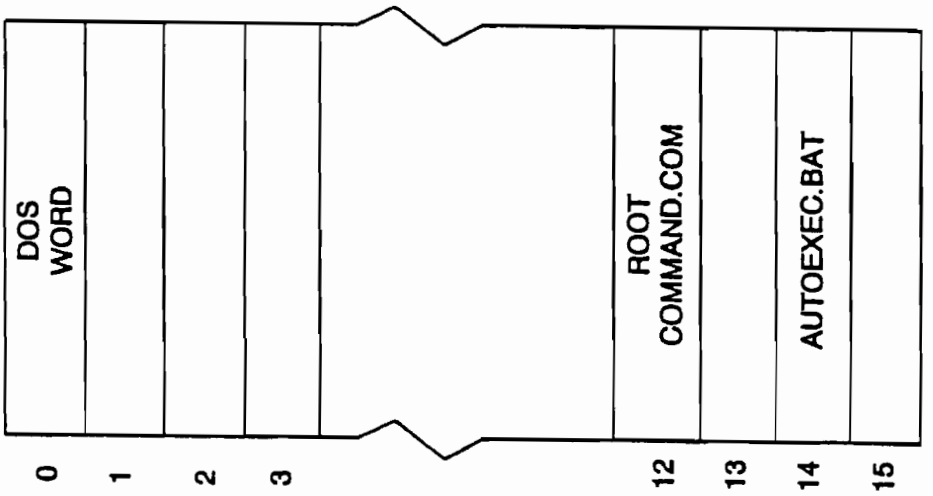
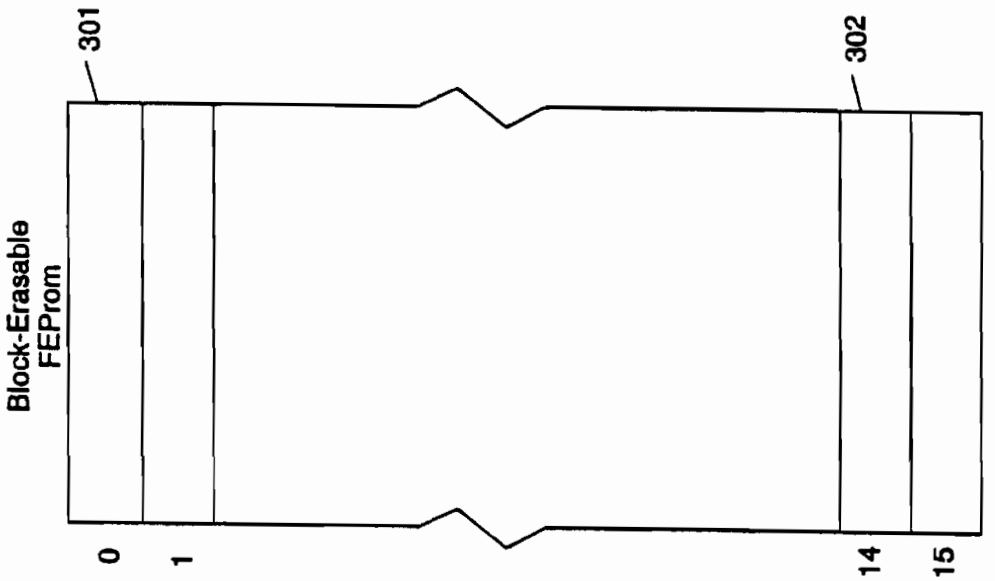


Figure 3
Prior Art



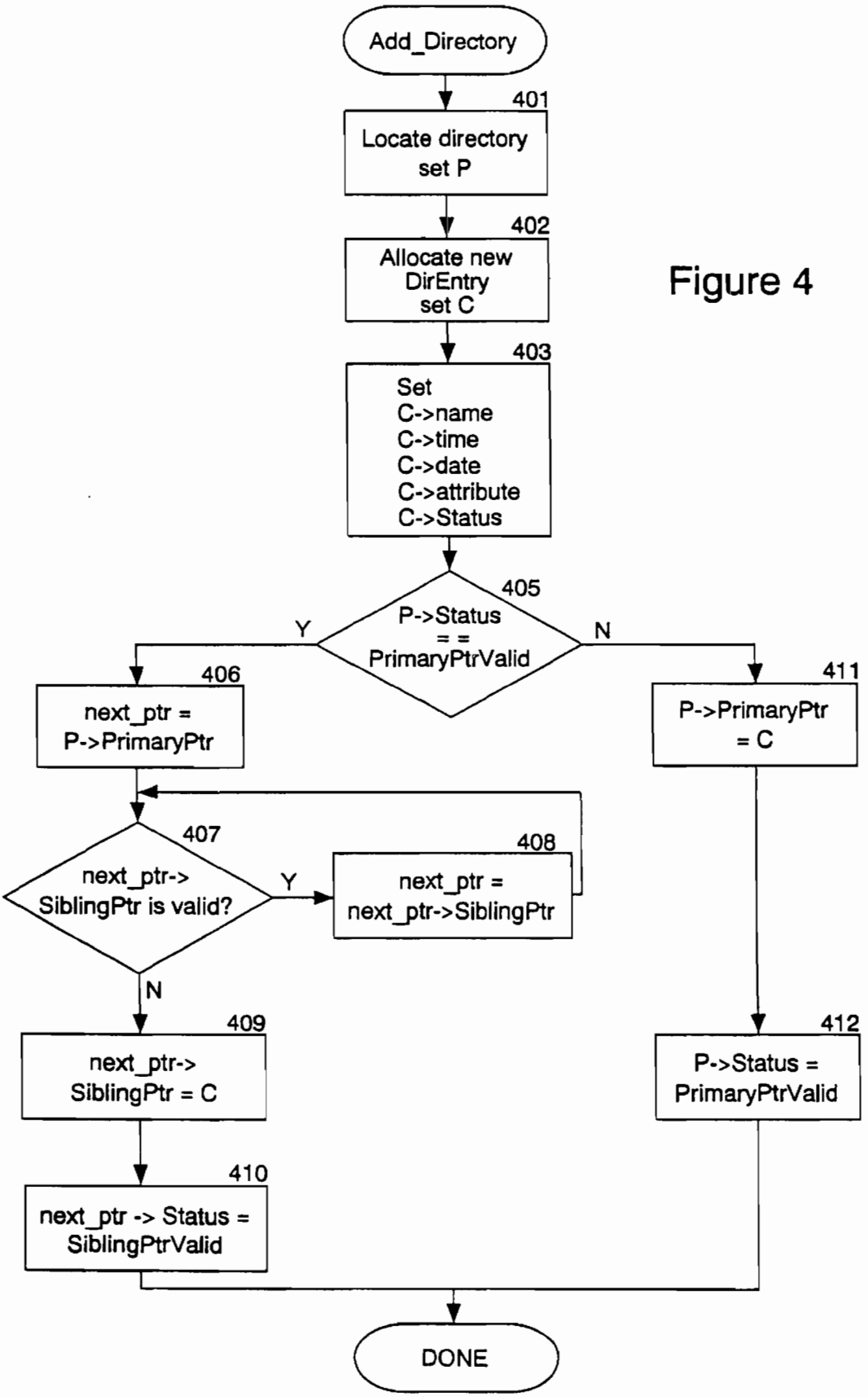

Add_Directory
401
Locate directory
set P
402
Allocate new
DirEntry
set C
403
Set
C->name
C->time
C->date
C->attribute
C->Status
405
P->Status
= =
PrimaryPtrValid
Y
N
406
next_ptr =
P->PrimaryPtr
411
P->PrimaryPtr
= C
407
next_ptr->
SiblingPtr is valid?
Y
408
next_ptr =
next_ptr->SiblingPtr
N
409
next_ptr->
SiblingPtr = C
412
P->Status =
PrimaryPtrValid
410
next_ptr -> Status =
SiblingPtrValid
DONE

Figure 4

Figure 5

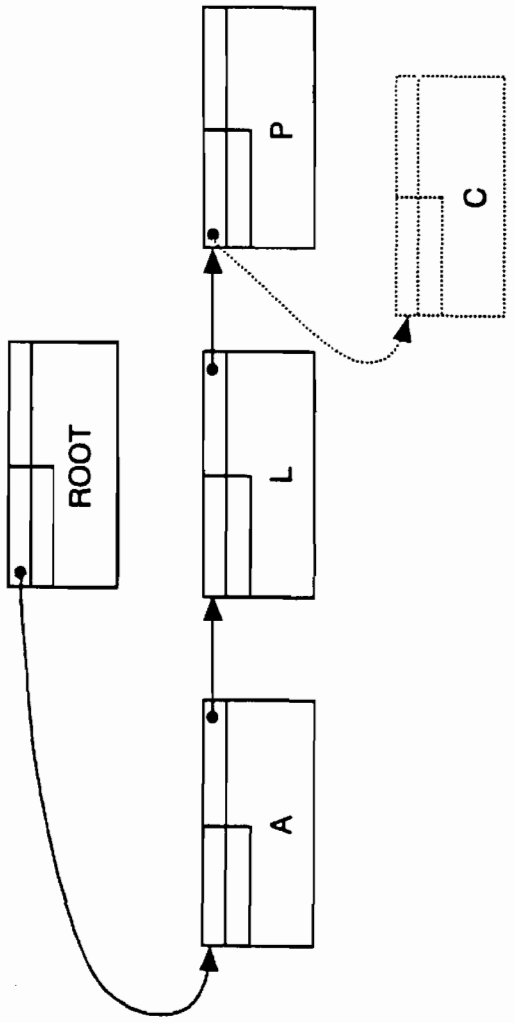
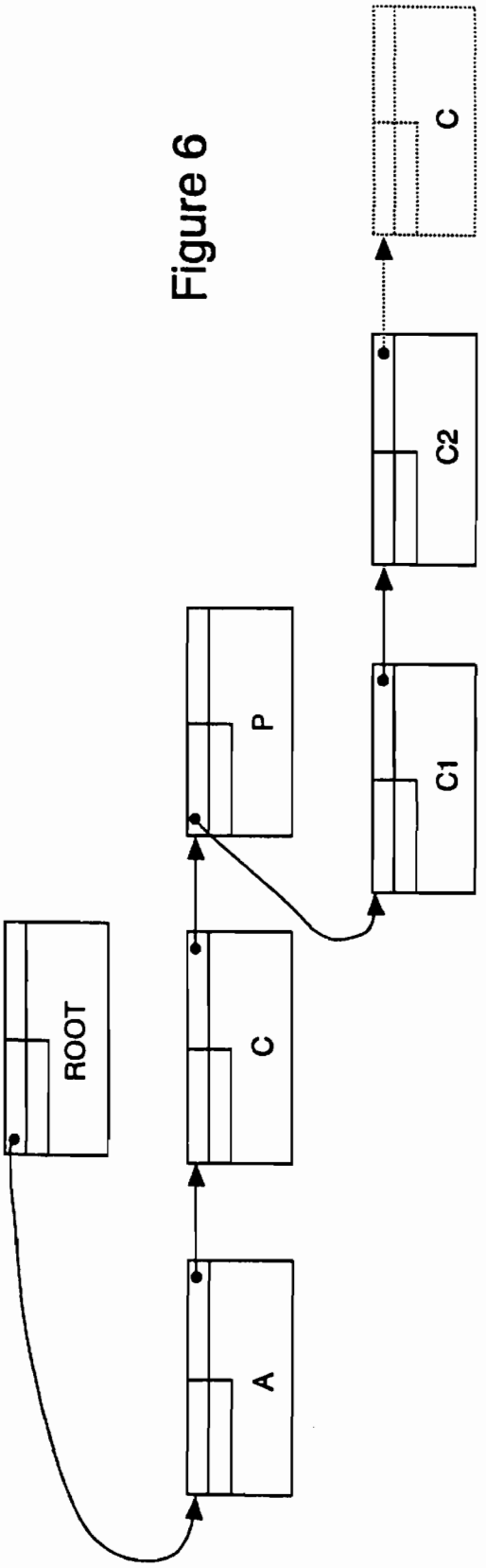


Figure 6



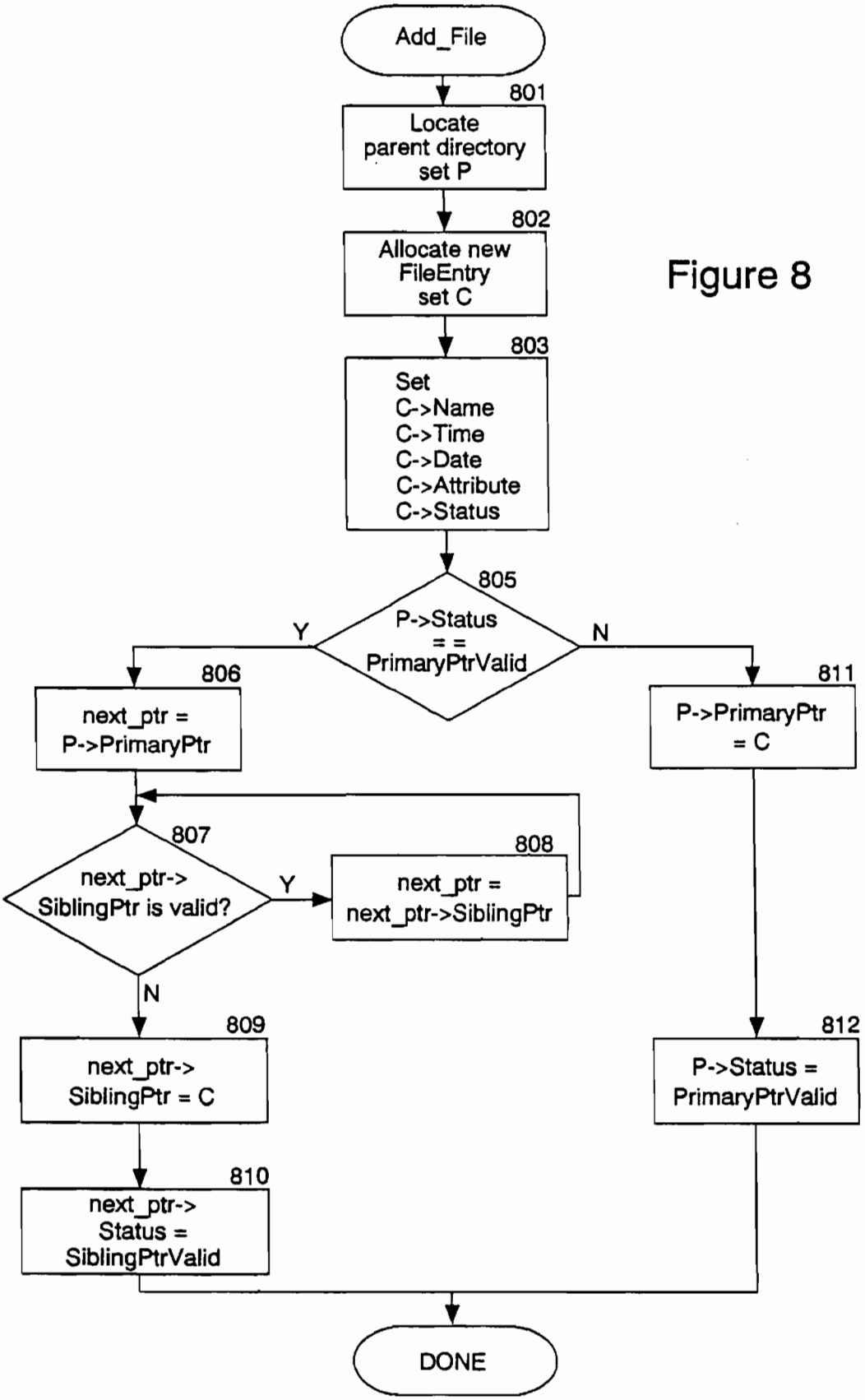

Add_File
801
Locate parent directory set P
802
Allocate new FileEntry set C
803
Set C->Name C->Time C->Date C->Attribute C->Status
805
P->Status == PrimaryPtrValid
Y
N
806
next_ptr = P->PrimaryPtr
811
P->PrimaryPtr = C
807
next_ptr-> SiblingPtr is valid?
Y
808
next_ptr = next_ptr->SiblingPtr
N
809
next_ptr-> SiblingPtr = C
812
P->Status = PrimaryPtrValid
810
next_ptr-> Status = SiblingPtrValid
DONE

Figure 8

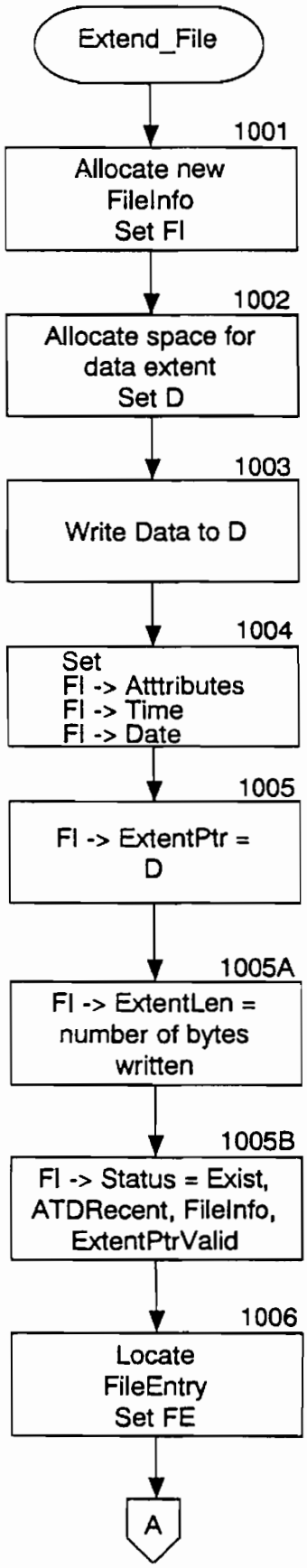

Extend_File
1001
Allocate new FileInfo Set FI
1002
Allocate space for data extent Set D
1003
Write Data to D
1004
Set FI -> Atttributes FI -> Time FI -> Date
1005
FI -> ExtentPtr = D
1005A
FI -> ExtentLen = number of bytes written
1005B
FI -> Status = Exist, ATDRecent, FileInfo, ExtentPtrValid
1006
Locate FileEntry Set FE
A

Figure 10

Figure 10 (cont'd)

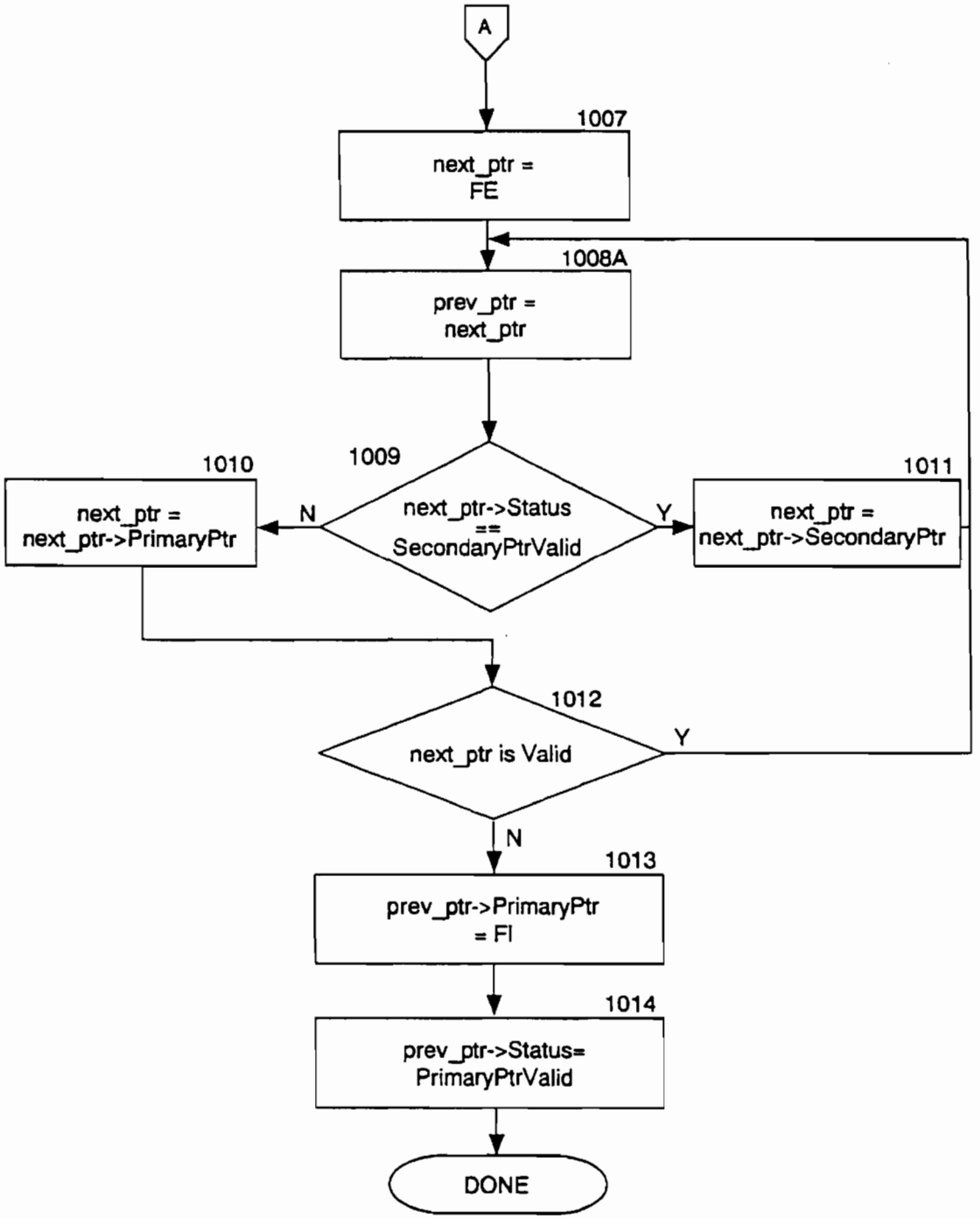
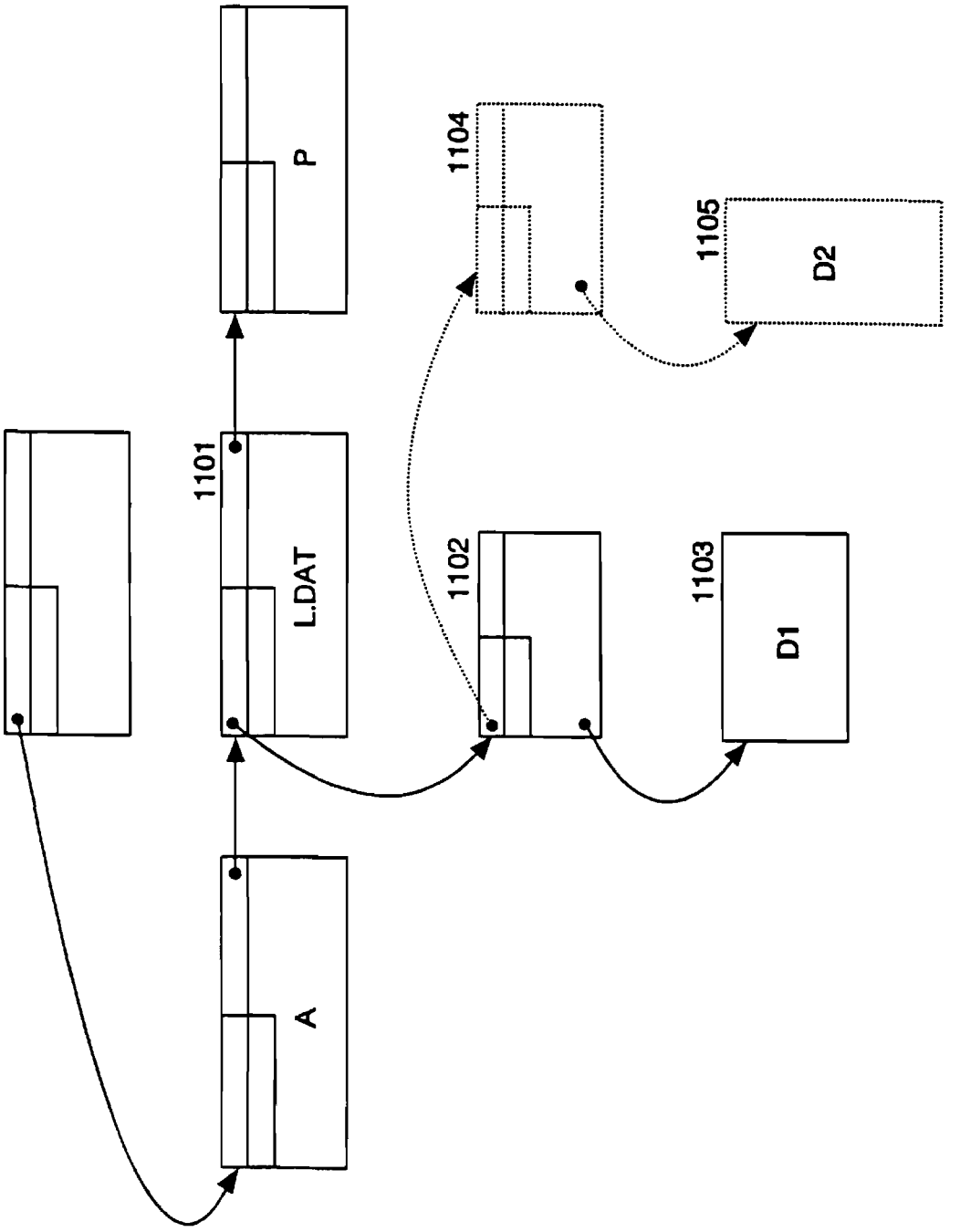


Figure 11



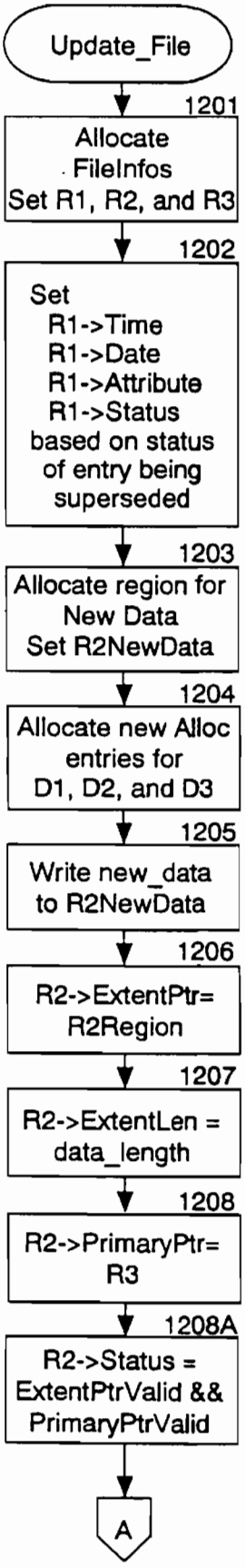

Update_File
1201
Allocate FileInfos Set R1, R2, and R3
1202
Set R1->Time R1->Date R1->Attribute R1->Status based on status of entry being superseded
1203
Allocate region for New Data Set R2NewData
1204
Allocate new Alloc entries for D1, D2, and D3
1205
Write new_data to R2NewData
1206
R2->ExtentPtr= R2Region
1207
R2->ExtentLen = data_length
1208
R2->PrimaryPtr= R3
1208A
R2->Status = ExtentPtrValid && PrimaryPtrValid
A

Figure 12

Figure 12 (cont'd)

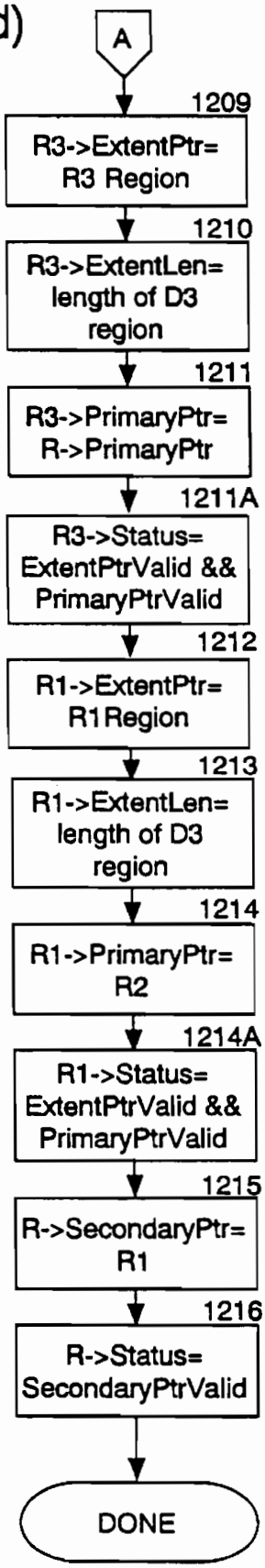


Figure 13

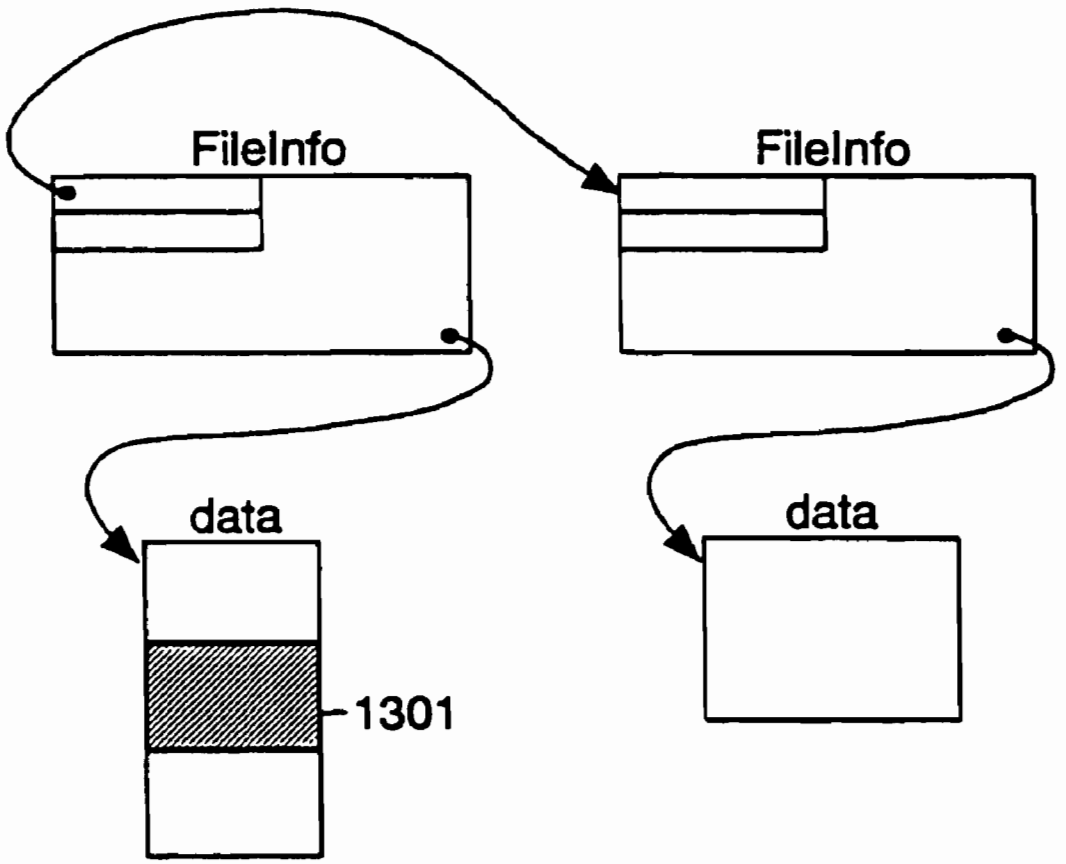


Figure 14

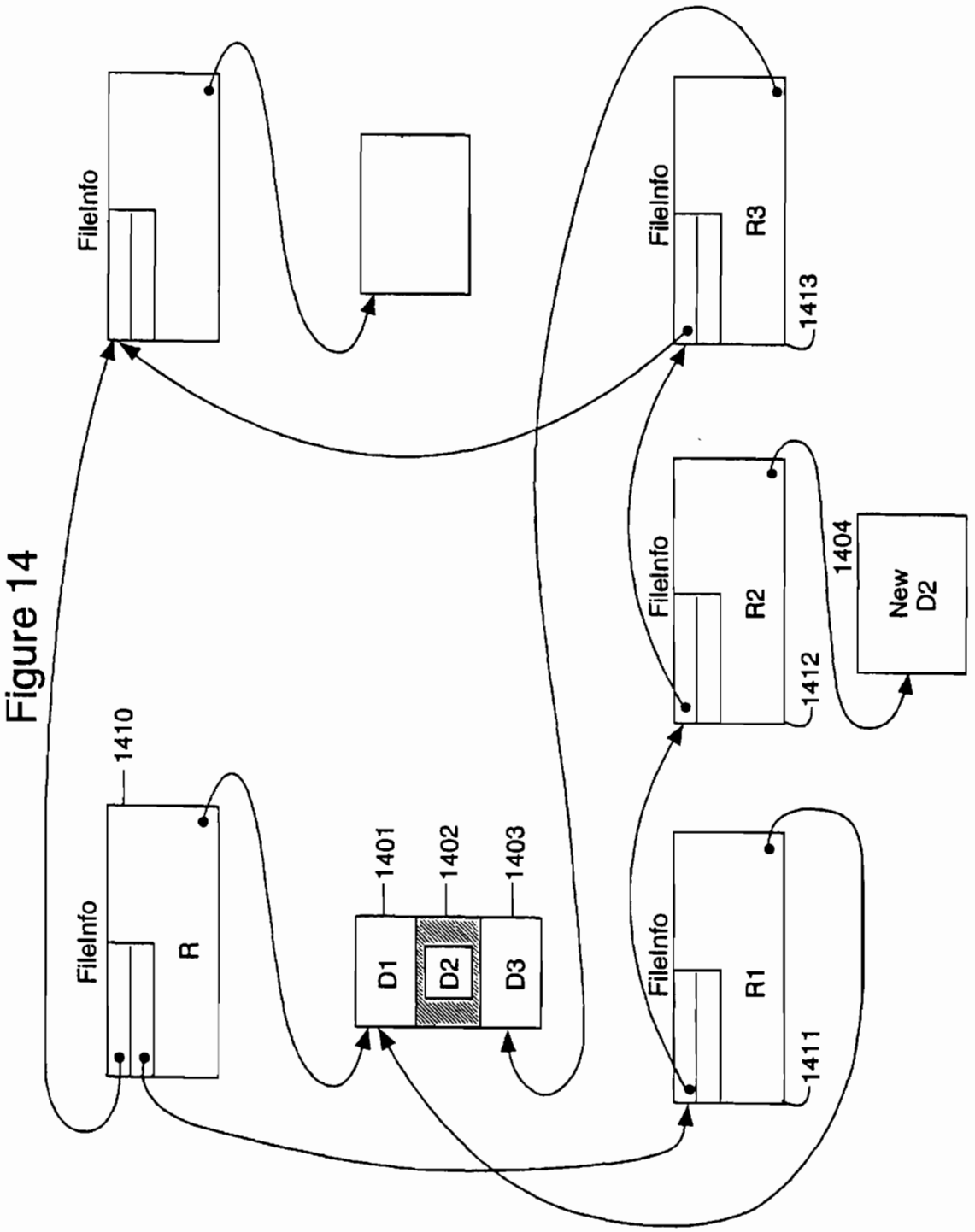
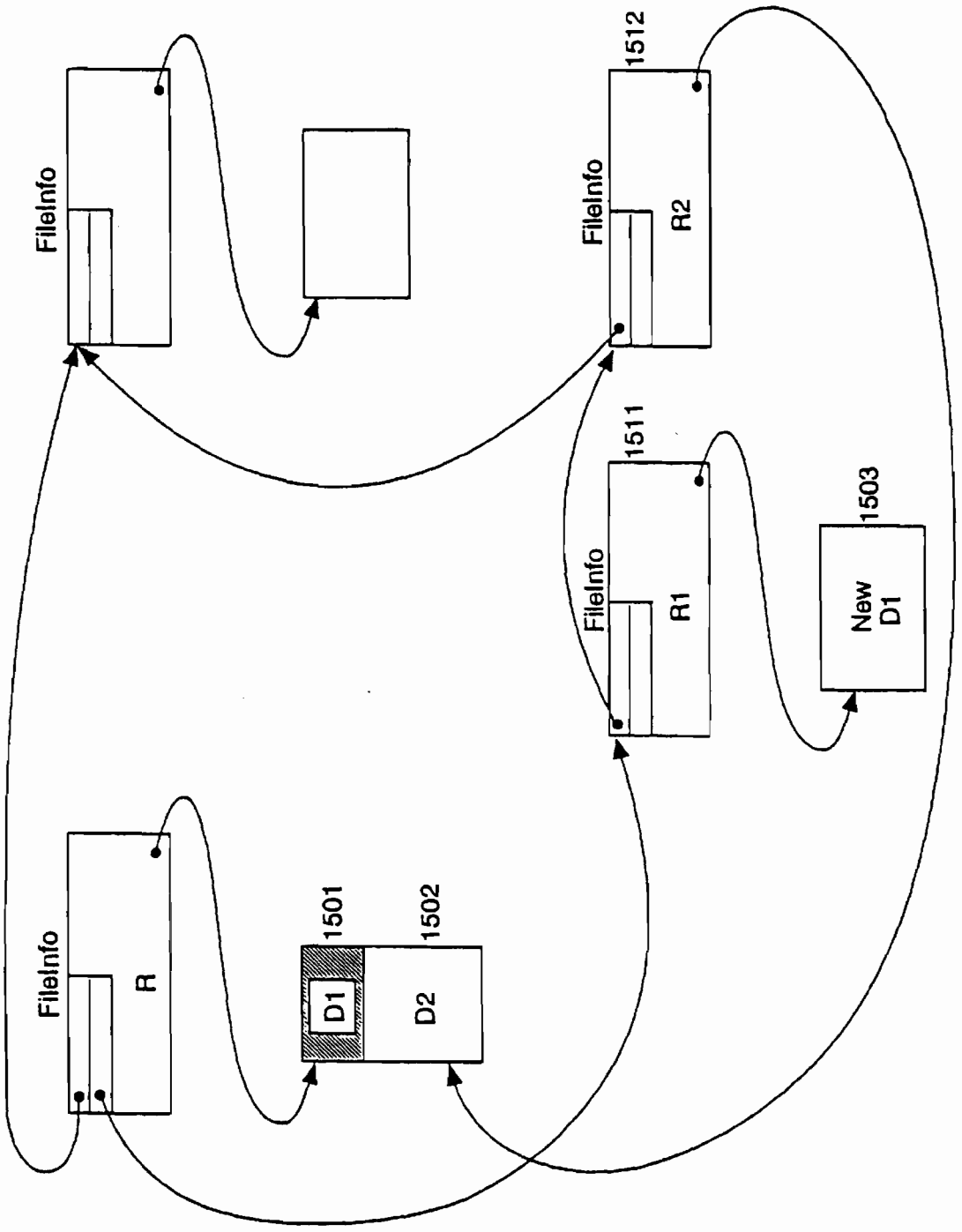

FileInfo
R
1410
D1
D2
D3
1401
1402
1403
FileInfo
R1
1411
FileInfo
R2
1412
1404
New D2
FileInfo
R3
1413

Figure 15


FileInfo
1512
FileInfo
R2
1511
FileInfo
R1
1503
New
D1
FileInfo
R
1501
D1
1502
D2

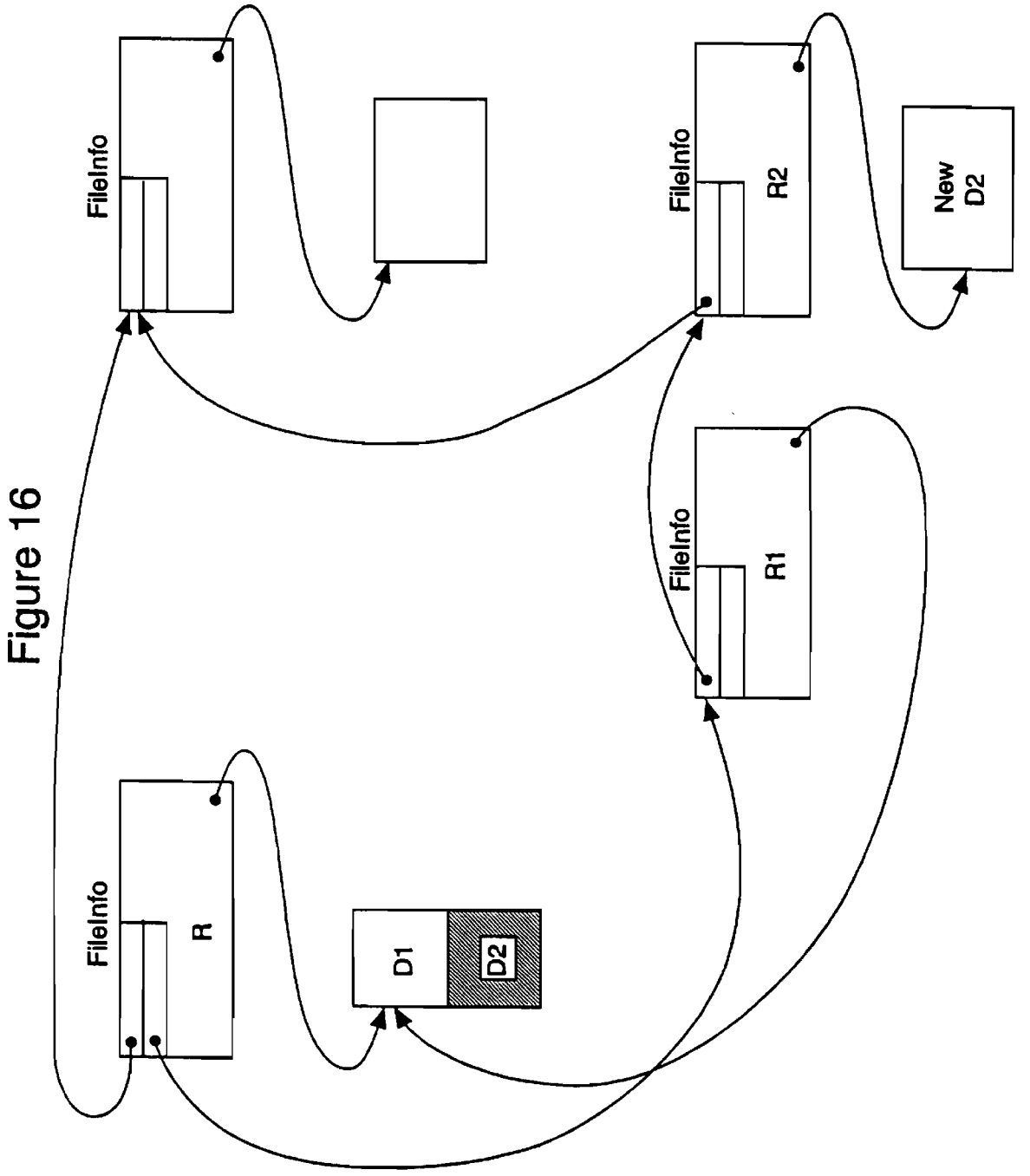


Figure 16

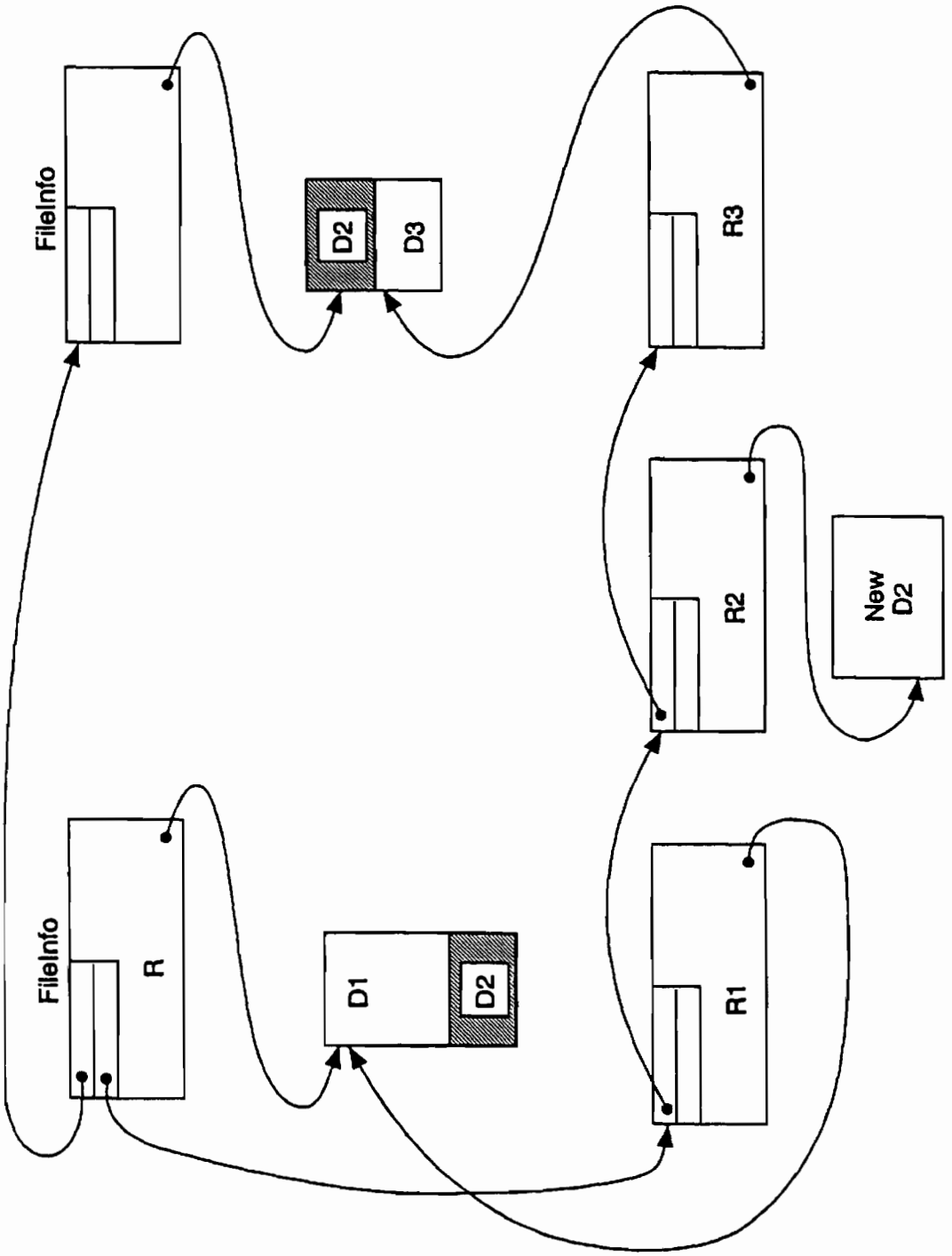


Figure 17

Figure 18

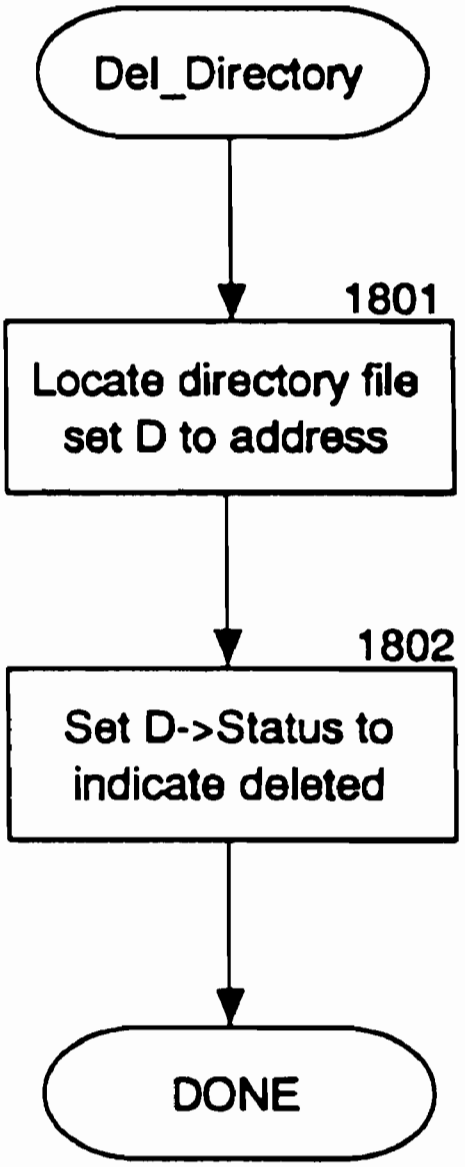

Del_Directory
1801
Locate directory file
set D to address
1802
Set D->Status to
indicate deleted
DONE

Figure 19

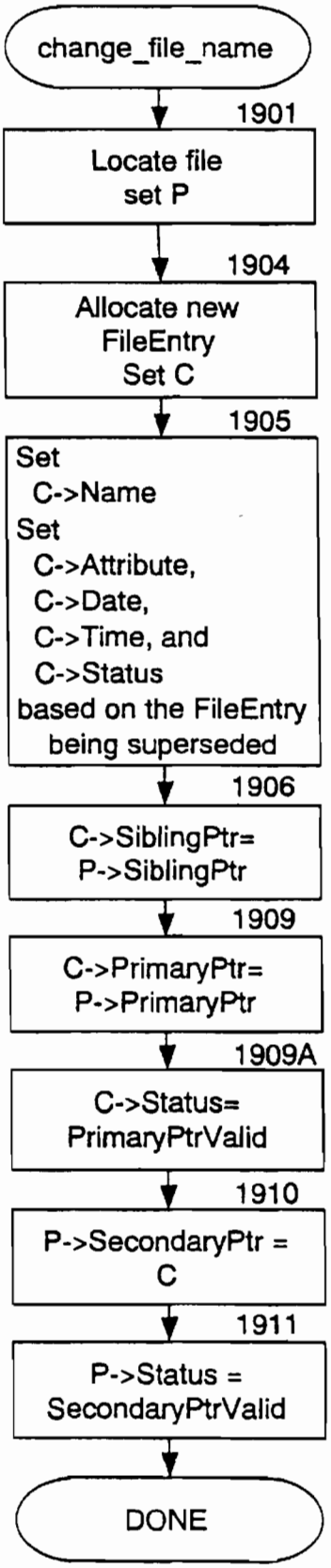

change_file_name
1901
Locate file
set P
1904
Allocate new
FileEntry
Set C
1905
Set
C->Name
Set
C->Attribute,
C->Date,
C->Time, and
C->Status
based on the FileEntry
being superseded
1906
C->SiblingPtr=
P->SiblingPtr
1909
C->PrimaryPtr=
P->PrimaryPtr
1909A
C->Status=
PrimaryPtrValid
1910
P->SecondaryPtr =
C
1911
P->Status =
SecondaryPtrValid
DONE

Figure 20

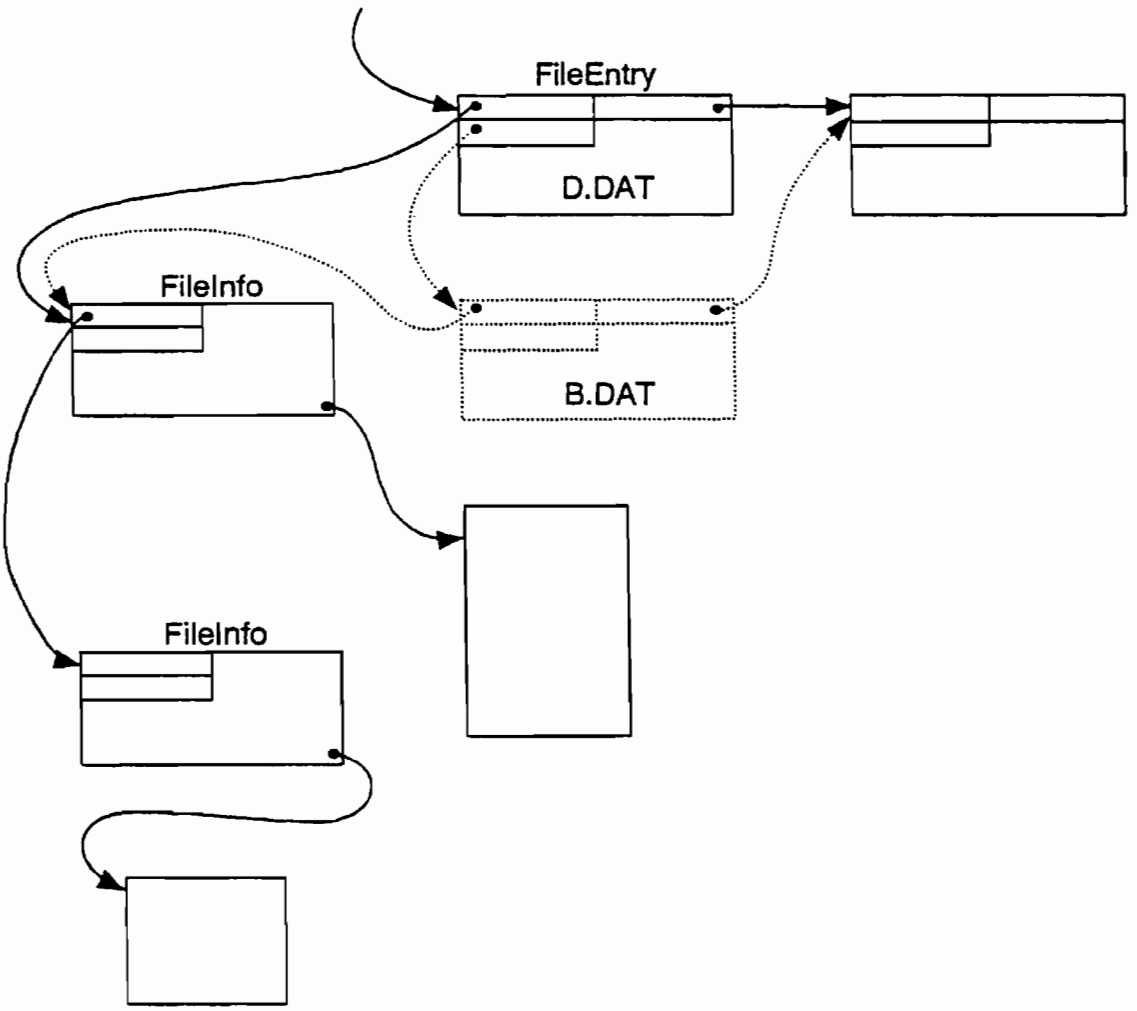
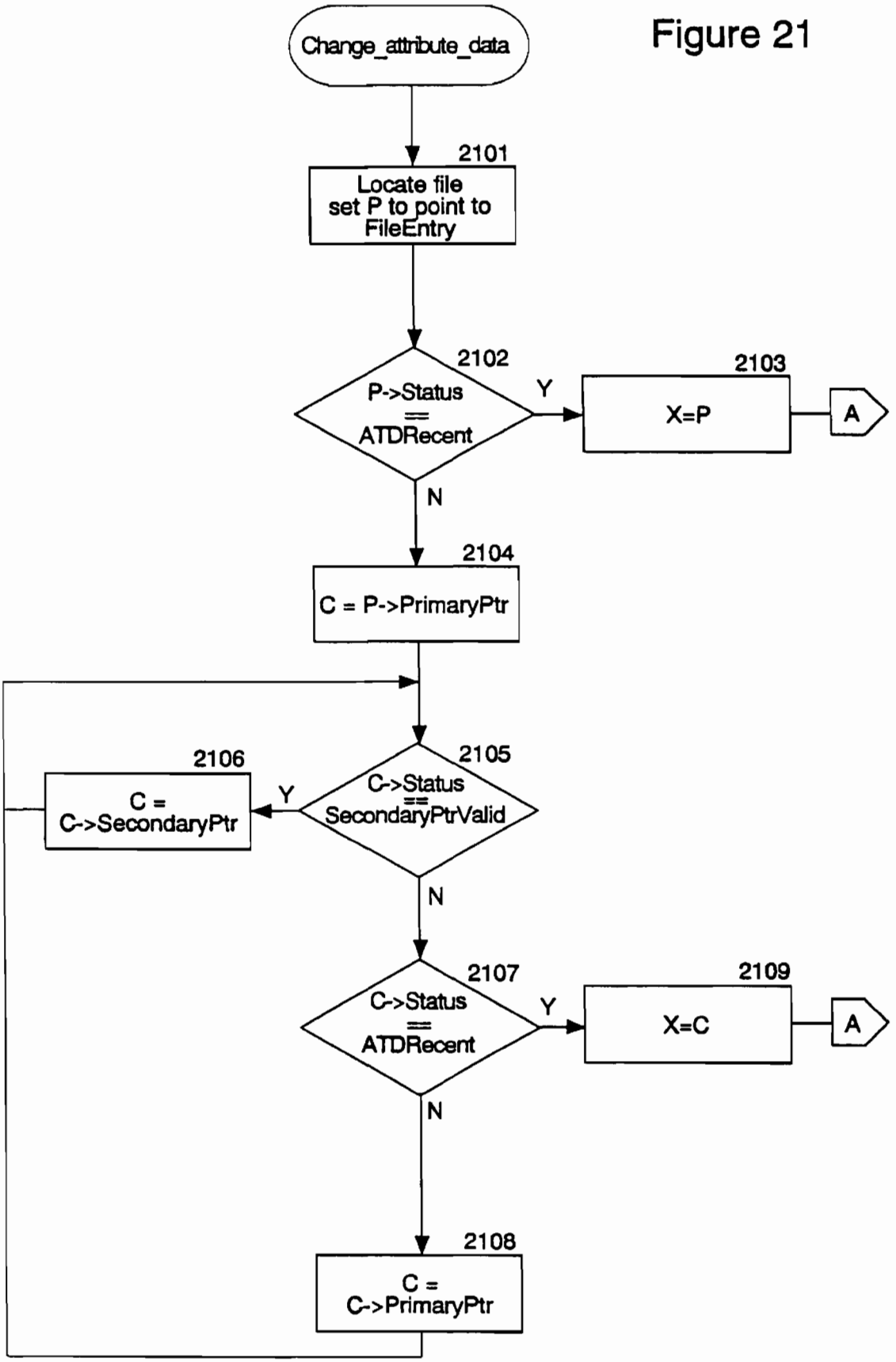


Figure 21


Change_attribute_data
2101
Locate file
set P to point to
FileEntry
2102
P->Status
==
ATDRecent
Y
2103
X=P
A
N
2104
C = P->PrimaryPtr
2105
C->Status
==
SecondaryPtrValid
Y
2106
C =
C->SecondaryPtr
N
2107
C->Status
==
ATDRecent
Y
2109
X=C
A
N
2108
C =
C->PrimaryPtr

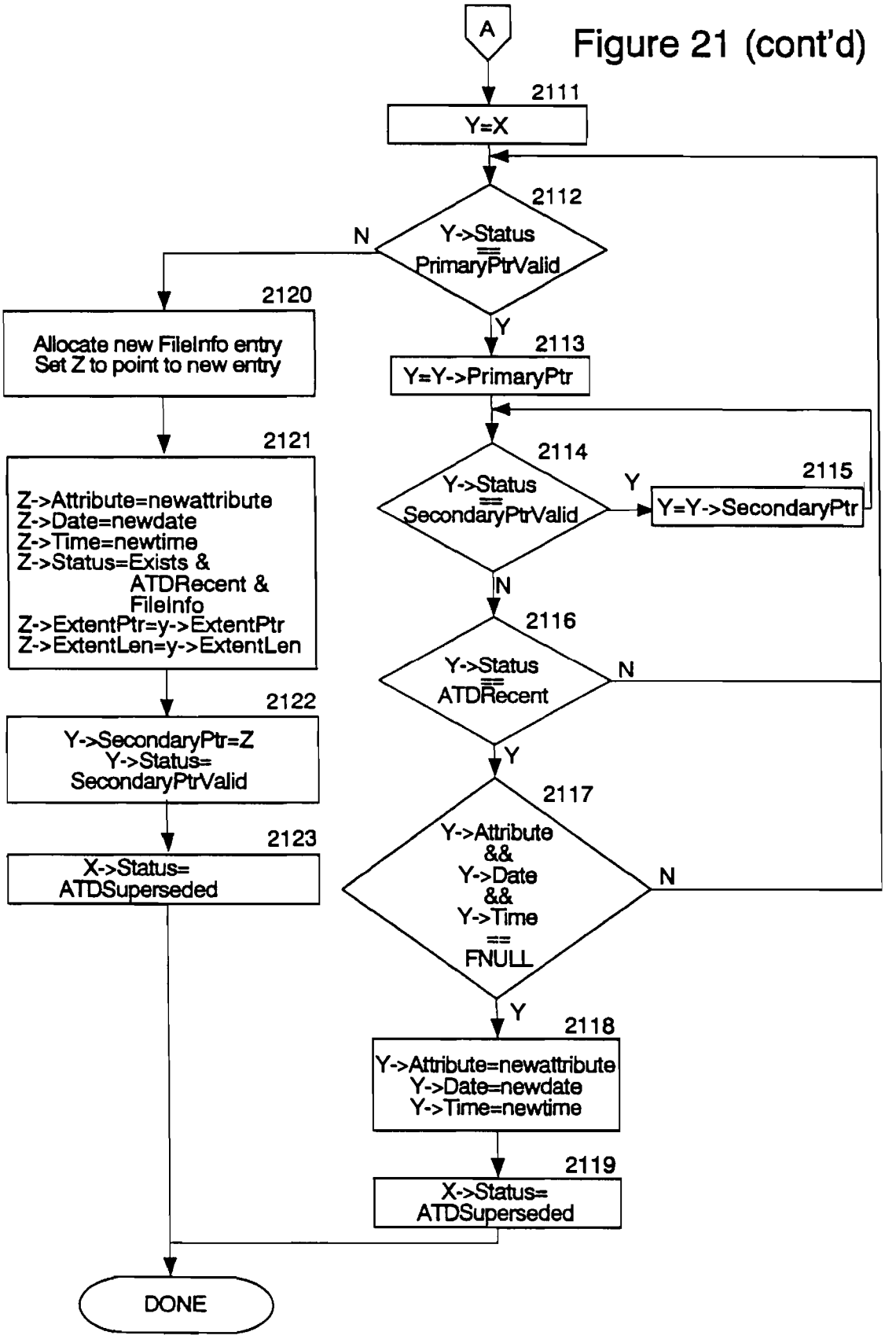

A
Figure 21 (cont'd)
2111
Y=X
2112
Y->Status == PrimaryPtrValid
N
Y
2120
Allocate new FileInfo entry
Set Z to point to new entry
2113
Y=Y->PrimaryPtr
2121
Z->Attribute=newattribute
Z->Date=newdate
Z->Time=newtime
Z->Status=Exists & ATDRecent & FileInfo
Z->ExtentPtr=y->ExtentPtr
Z->ExtentLen=y->ExtentLen
2114
Y->Status == SecondaryPtrValid
Y
2115
Y=Y->SecondaryPtr
N
2116
Y->Status == ATDRecent
N
Y
2122
Y->SecondaryPtr=Z
Y->Status= SecondaryPtrValid
2117
Y->Attribute && Y->Date && Y->Time == FNULL
N
2123
X->Status= ATDSuperseded
Y
2118
Y->Attribute=newattribute
Y->Date=newdate
Y->Time=newtime
2119
X->Status= ATDSuperseded
DONE

Figure 23A

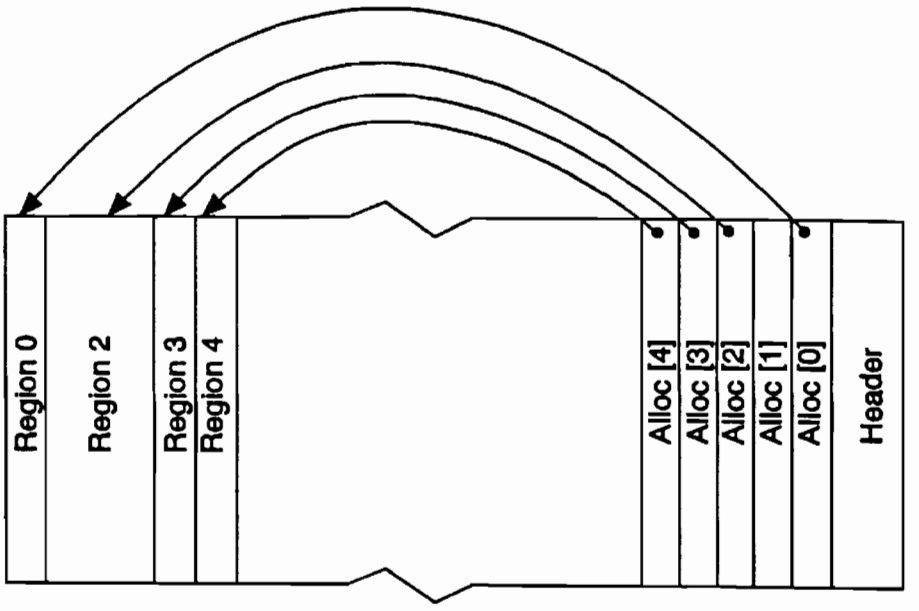


Figure 23

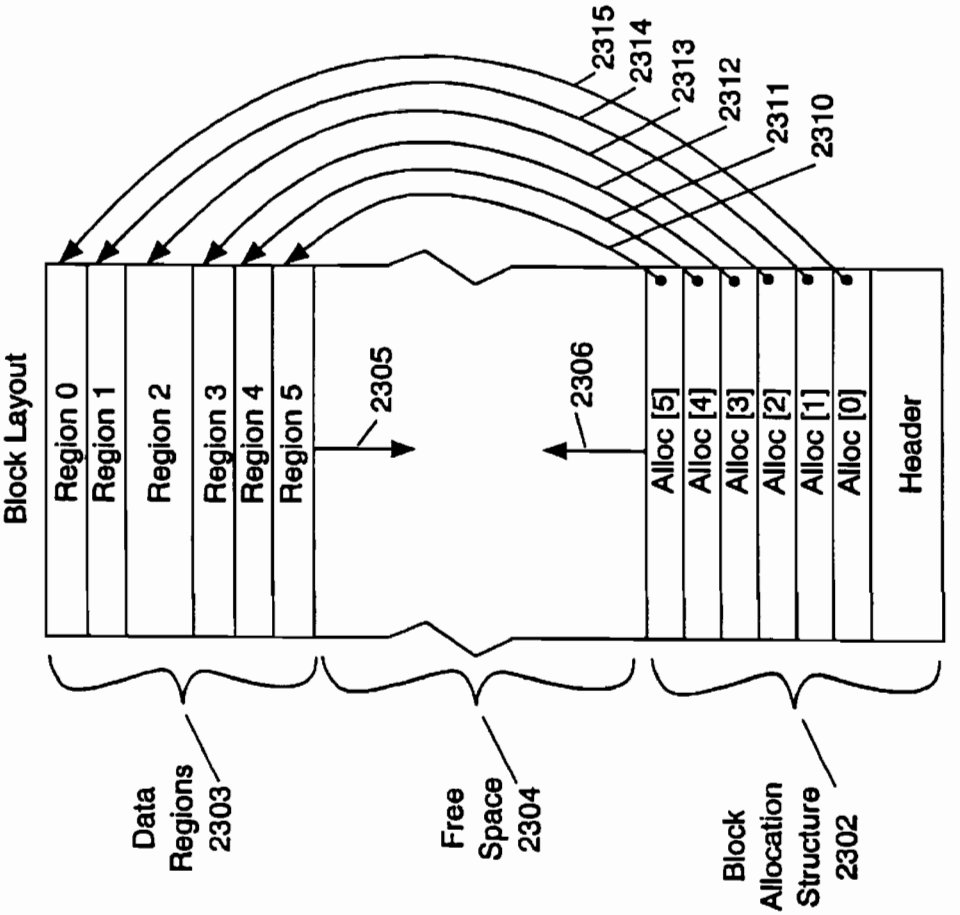
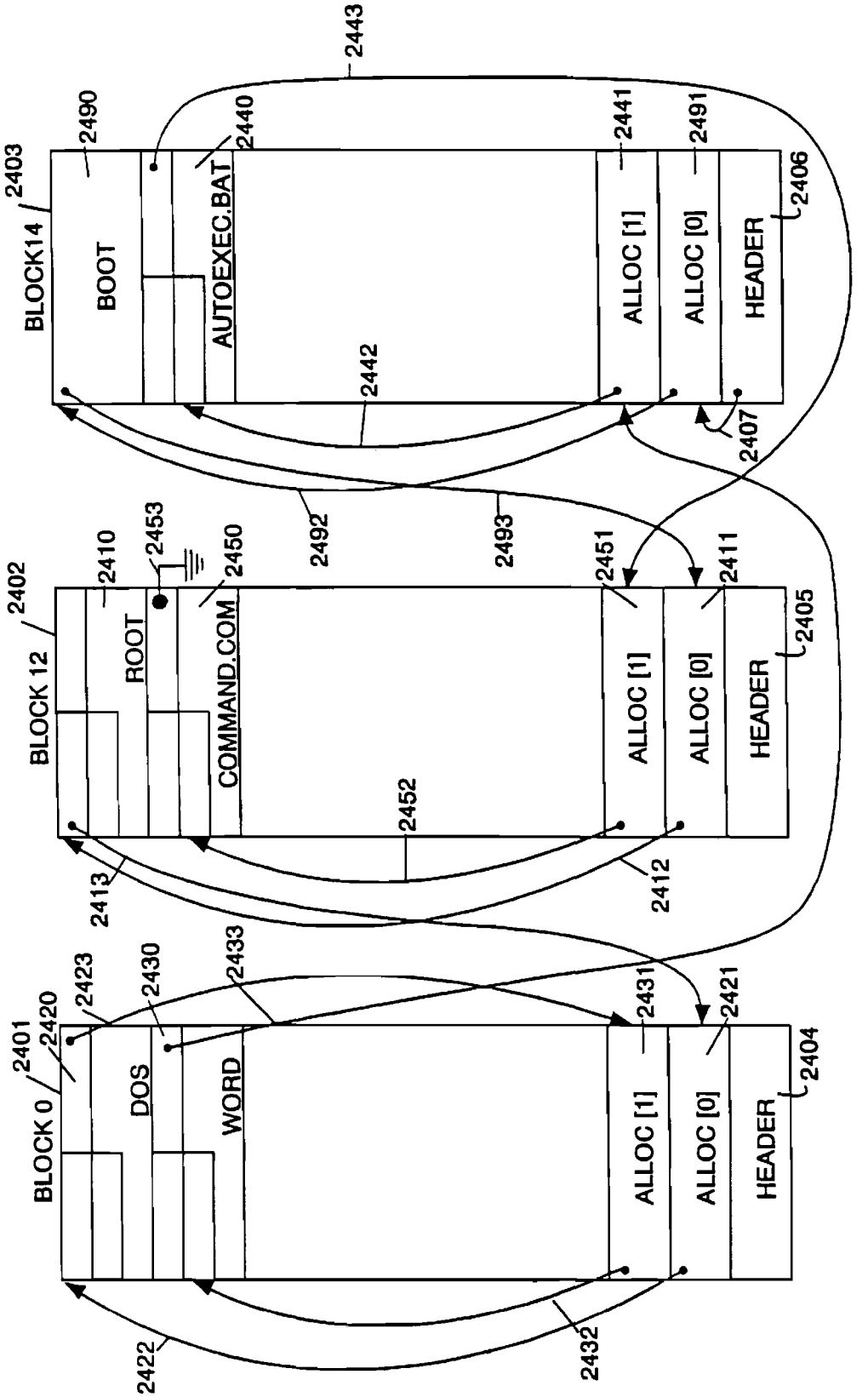


Figure 24


2443
2490
2440
2441
2491
2403
2406
BLOCK14
BOOT
AUTOEXEC.BAT
ALLOC [1]
ALLOC [0]
HEADER
2442
2407
2492
2493
2410
2453
2450
2451
2411
2402
2405
BLOCK 12
ROOT
COMMAND.COM
ALLOC [1]
ALLOC [0]
HEADER
2452
2413
2412
2423
2430
2433
2431
2421
2420
2401
2404
BLOCK 0
DOS
WORD
ALLOC [1]
ALLOC [0]
HEADER
2422
2432

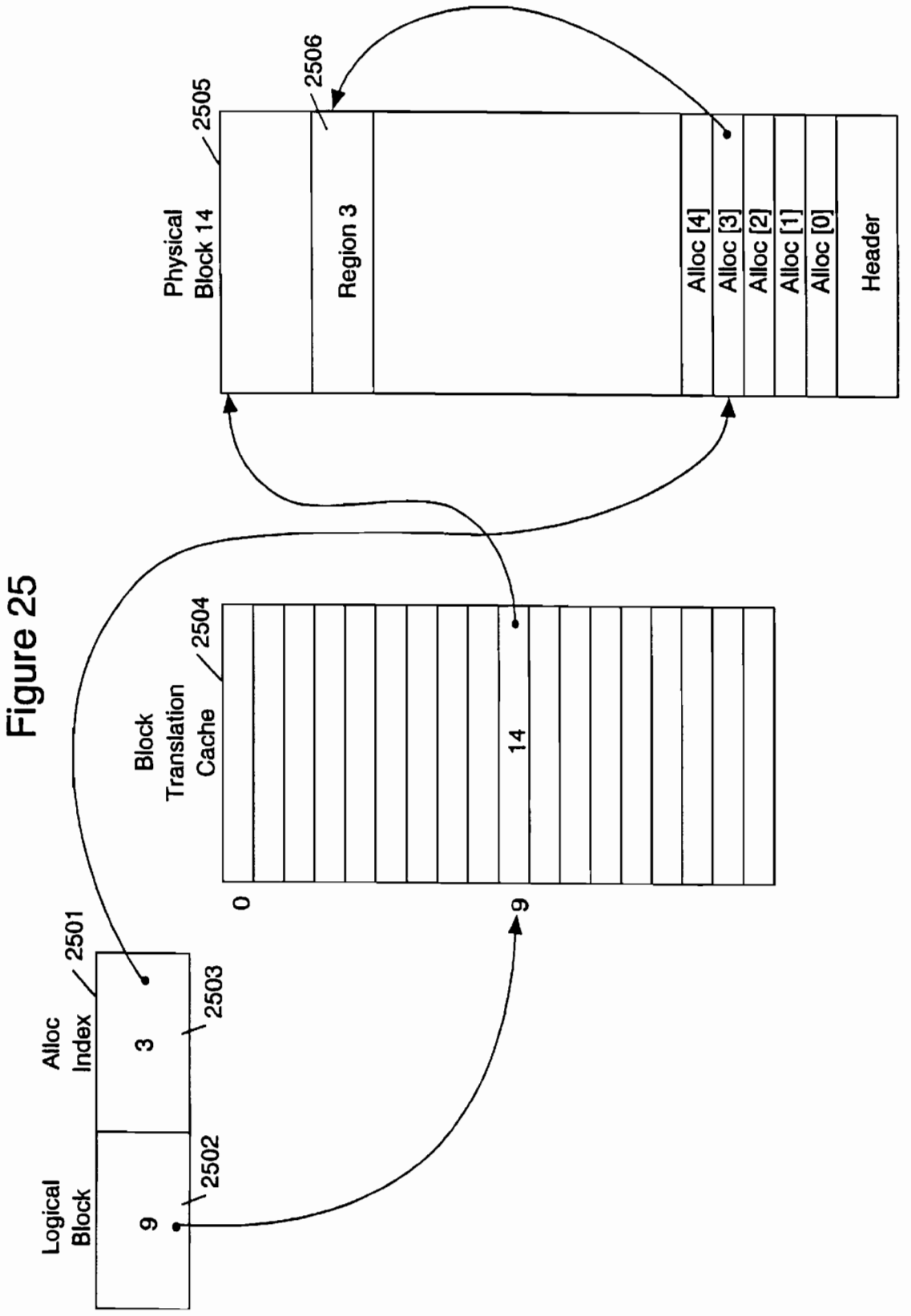
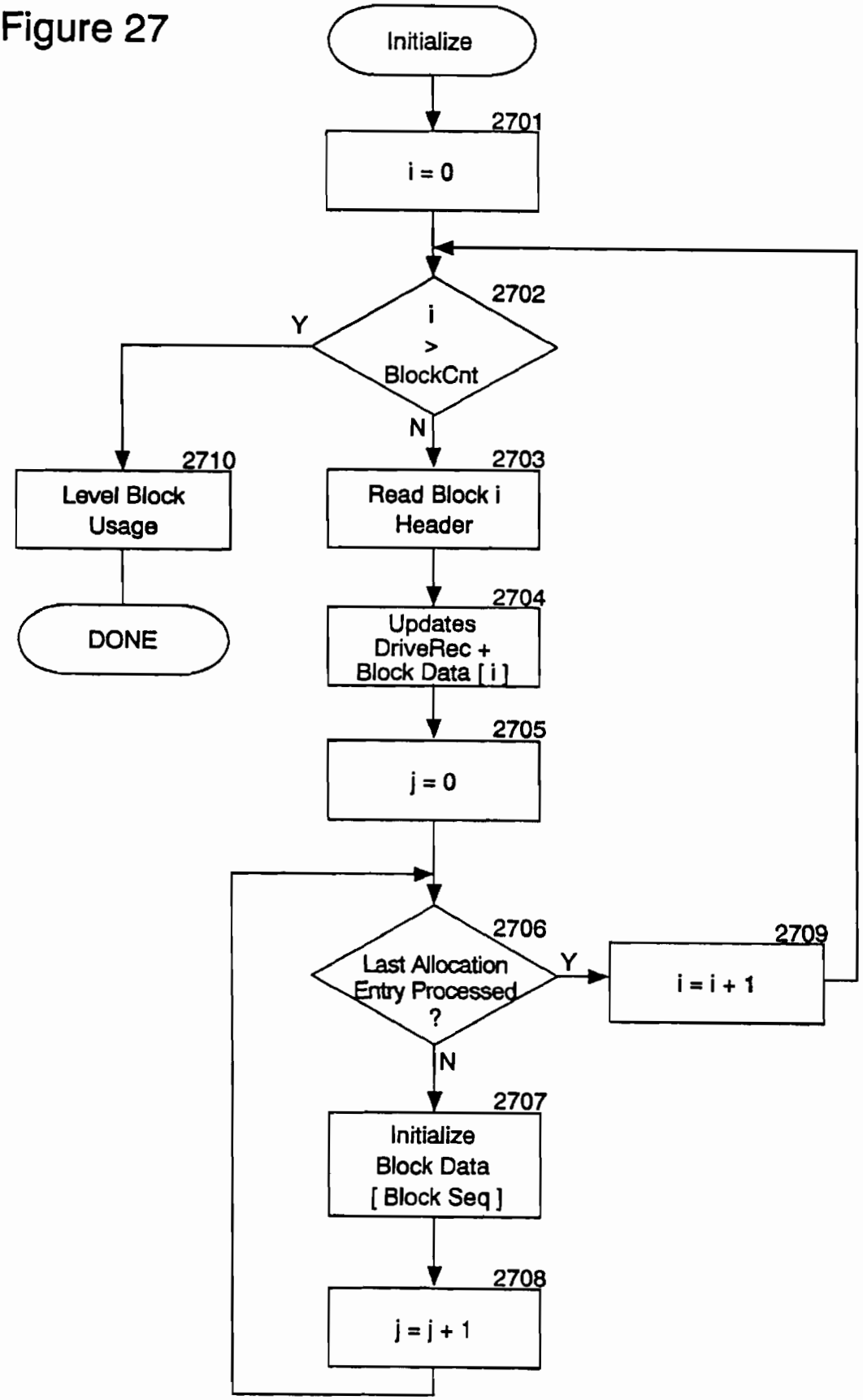

Logical Block
9
2502
Alloc Index
3
2503
2501
Block Translation Cache
2504
0
14
9
Physical Block 14
2505
Region 3
2506
Alloc [4]
Alloc [3]
Alloc [2]
Alloc [1]
Alloc [0]
Header

Figure 25

Figure 27


Initialize
2701
i = 0
2702
i > BlockCnt
Y
N
2710
Level Block Usage
DONE
2703
Read Block i Header
2704
Updates DriveRec + Block Data [i]
2705
j = 0
2706
Last Allocation Entry Processed ?
Y
2709
i = i + 1
N
2707
Initialize Block Data [Block Seq]
2708
j = j + 1

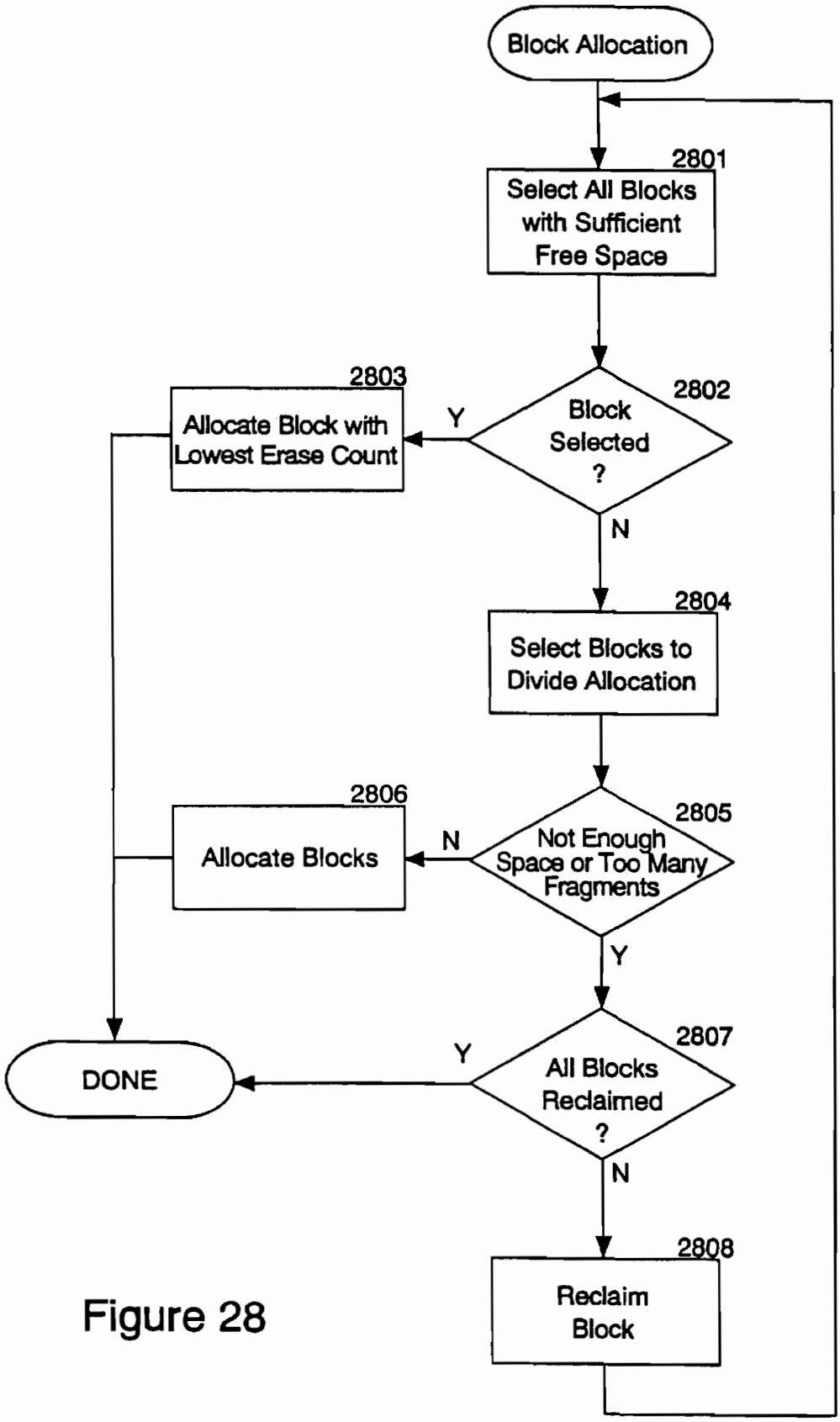

Block Allocation
2801
Select All Blocks with Sufficient Free Space
2802
Block Selected ?
Y
2803
Allocate Block with Lowest Erase Count
N
2804
Select Blocks to Divide Allocation
2805
Not Enough Space or Too Many Fragments
N
2806
Allocate Blocks
Y
2807
All Blocks Reclaimed ?
Y
DONE
N
2808
Reclaim Block

Figure 28

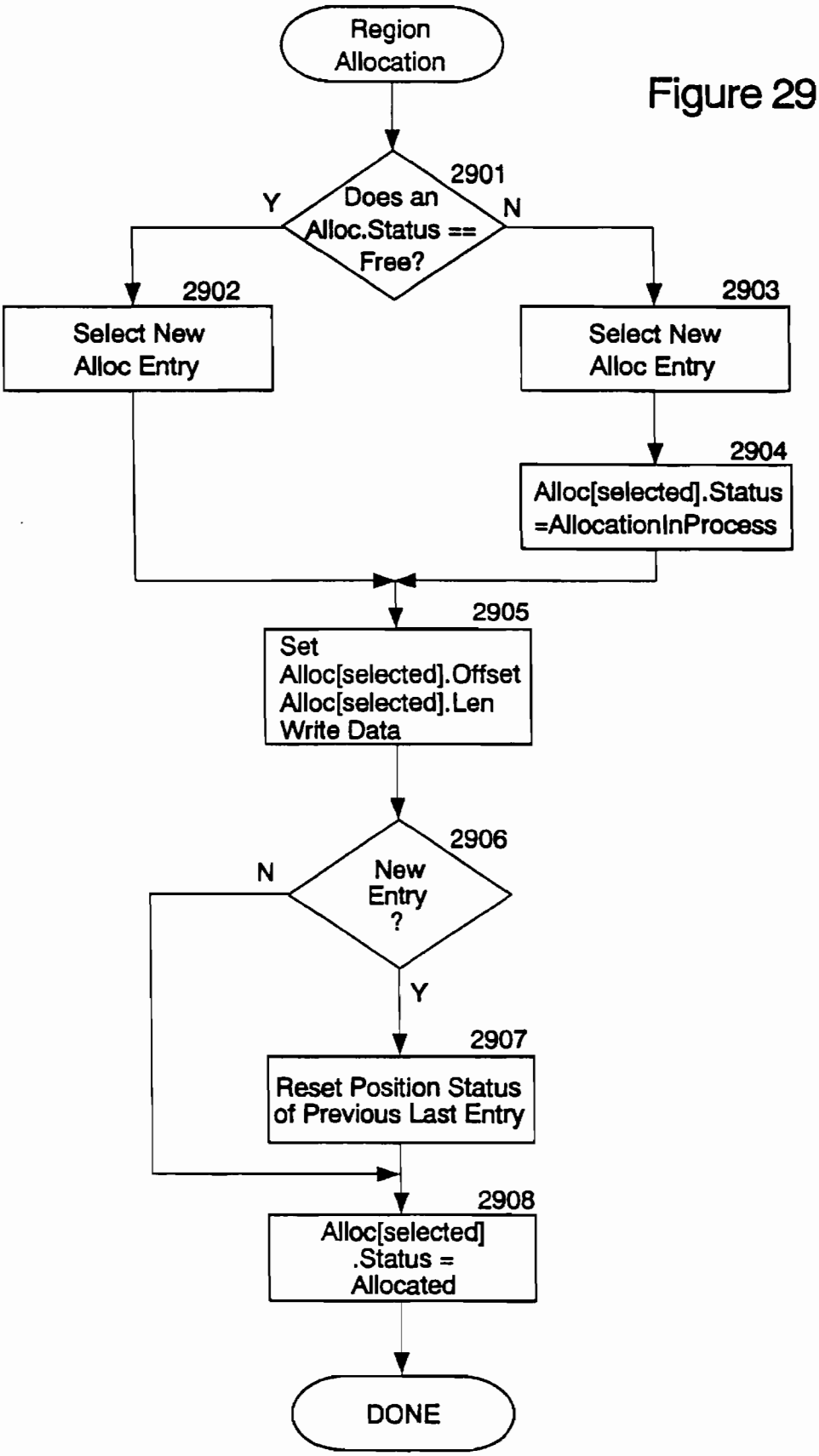

Region Allocation
2901
Does an Alloc.Status == Free?
Y
N
2902
Select New Alloc Entry
2903
Select New Alloc Entry
2904
Alloc[selected].Status =AllocationInProcess
2905
Set Alloc[selected].Offset Alloc[selected].Len Write Data
2906
New Entry ?
N
Y
2907
Reset Position Status of Previous Last Entry
2908
Alloc[selected] .Status = Allocated
DONE

Figure 29

Figure 30

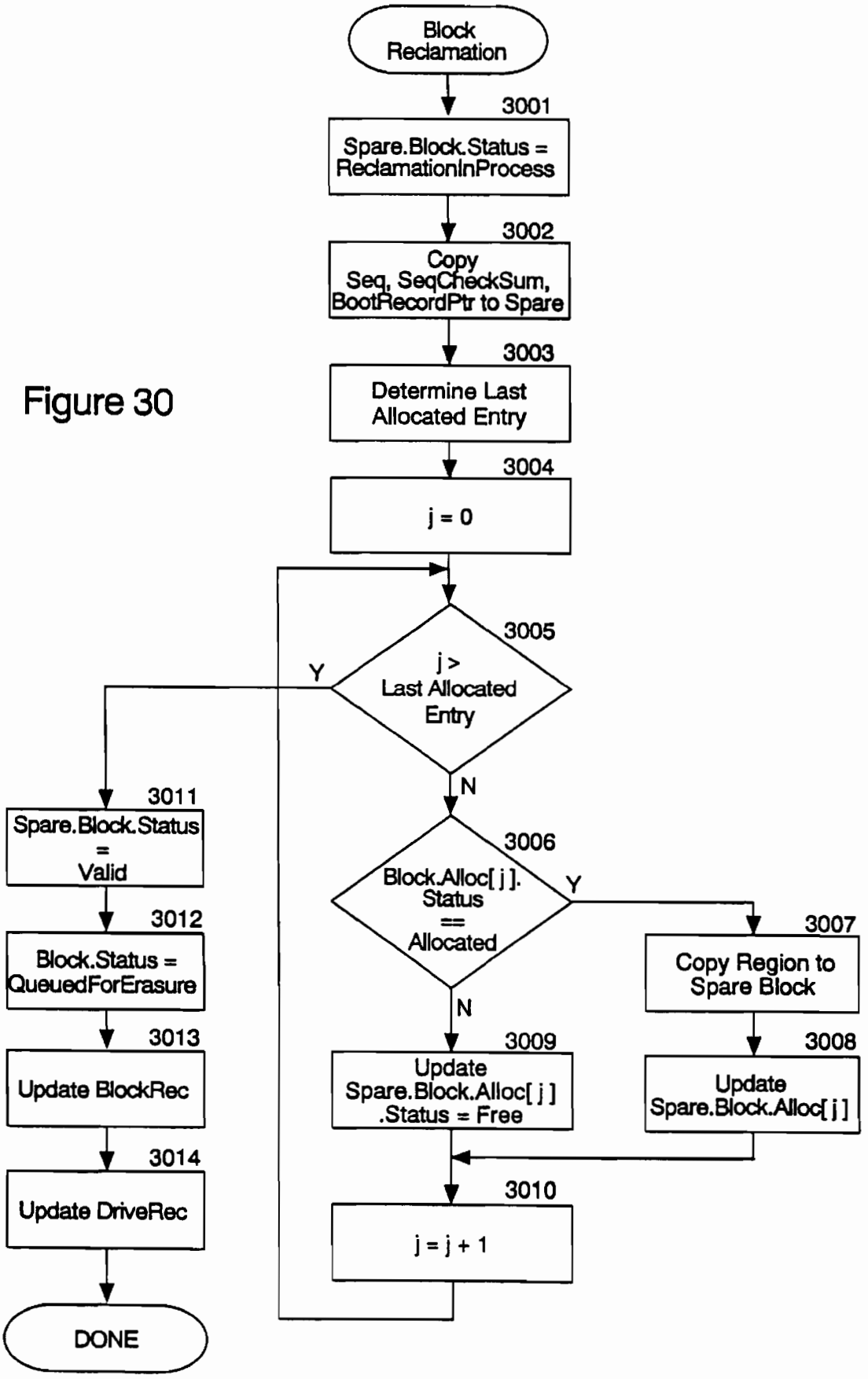

Block Reclamation
3001
Spare.Block.Status = ReclamationInProcess
3002
Copy Seq, SeqCheckSum, BootRecordPtr to Spare
3003
Determine Last Allocated Entry
3004
j = 0
3005
j > Last Allocated Entry
Y
N
3011
Spare.Block.Status = Valid
3012
Block.Status = QueuedForErasure
3013
Update BlockRec
3014
Update DriveRec
DONE
3006
Block.Alloc[j]. Status == Allocated
Y
N
3007
Copy Region to Spare Block
3008
Update Spare.Block.Alloc[j]
3009
Update Spare.Block.Alloc[j] .Status = Free
3010
j = j + 1

Figure 31

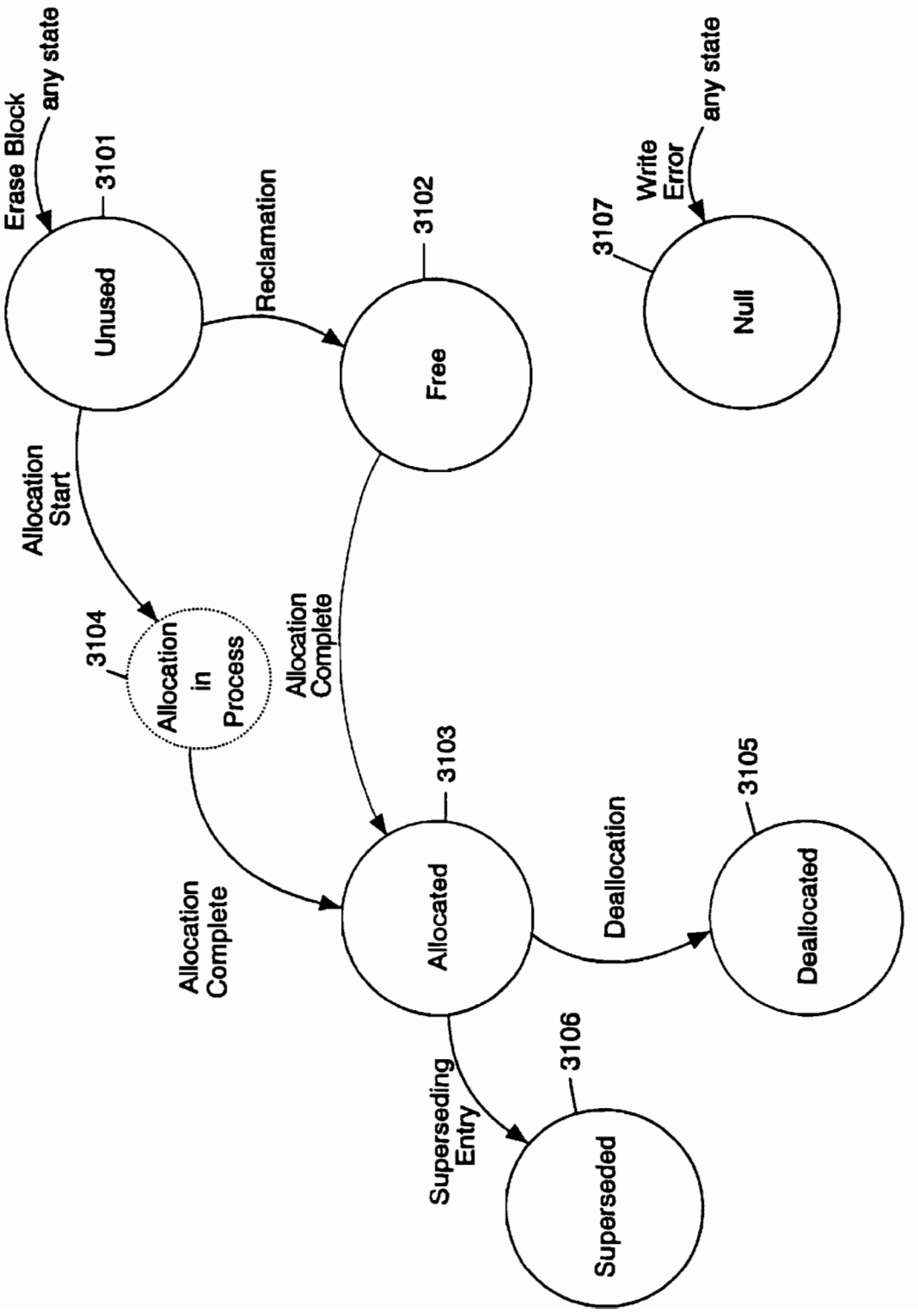
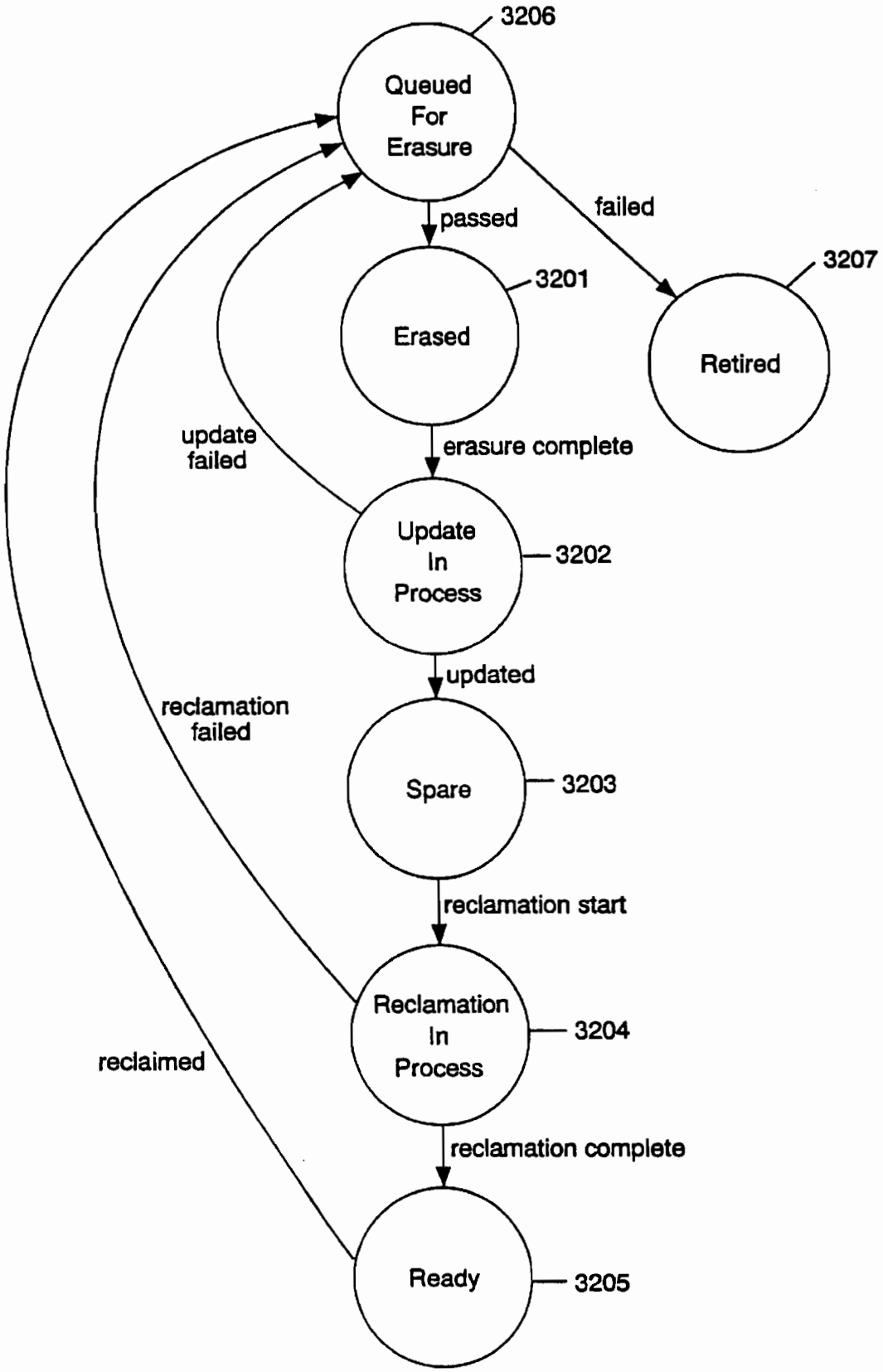

Erase Block
any state
3101
Unused
Reclamation
3102
Free
3107
Write Error
any state
Null
Allocation Start
3104
Allocation in Process
Allocation Complete
Allocation Complete
3103
Allocated
Deallocation
3105
Deallocated
Superseding Entry
3106
Superseded

Figure 32


3206
Queued For Erasure
failed
passed
3207
3201
Erased
Retired
update failed
erasure complete
Update In Process
3202
updated
reclamation failed
Spare
3203
reclamation start
Reclamation In Process
3204
reclaimed
reclamation complete
Ready
3205

METHOD AND SYSTEM FOR FILE SYSTEM MANAGEMENT USING A FLASH-ERASABLE, PROGRAMMABLE, READ-ONLY MEMORY

TECHNICAL FIELD

This invention relates generally to a computer system for managing files and, more specifically, to a method and system for managing files stored on a flash-erasable, programmable, read-only memory (FEProm).

BACKGROUND ART

A computer system generally supports the storage of information on both volatile and nonvolatile storage devices. The difference between a volatile and nonvolatile storage device is that when power is disconnected from a volatile storage device the information is lost. Conversely, when power is disconnected from a nonvolatile storage device the information is not lost. Thus, the storing of information on a nonvolatile storage device allows a user to enter information at one time and retrieve the information at a later time, even though the computer system may have been powered down. A user could also disconnect a nonvolatile storage device from a computer and connect the storage device to a different computer to allow the different computer to access the information.

The information stored on nonvolatile storage devices is generally organized into files. A file is a collection of related information. Over the course of time, a computer system can store hundreds and thousands of files on a storage device, depending upon the capacity of the device. In addition to storing the information, the computer system will typically read, modify, and delete the information in the files. It is important that the computer system organize the files on the storage device so that the storing, reading, modifying, and deleting can be accomplished efficiently.

File systems, which are generally part of a computer operating system, were developed to aid in the management of the files on storage devices. One such file system was developed by Microsoft Corporation for its Disk Operating System (MS-DOS). This file system uses a hierarchical approach to storing files. FIG. 1A shows a pictorial representation of the directory structure for a storage device. Directories contain a logical group of files. Directories organize files in a manner that is analogous to the way that folders in a drawer organize the papers in the drawer. The blocks labeled DOS, WORD, DAVID, and MARY represent directories, and the blocks labeled AUTOEXEC.BAT, COMMAND.COM, FORMAT.EXE, LETTER2.DOC, LETTER.DOC, and two files named LETTER1.DOC represent files. The directory structure allows a user to organize files by placing related files in their own directories. In this example, the directory WORD may contain all the files generated by the word-processing program WORD. Within directory WORD are two subdirectories DAVID and MARY, which aid in further organizing the WORD files into those developed by David and those developed by Mary.

Conventional file systems take advantage of the multiple-write capability of the nonvolatile store devices. The multiple-write capability allows any bit of information on the storage device to be changed from a one to zero and from a zero to one a virtually unlimited number of times. This capability allows a file to be written to the storage device and then selectively modified by changing some bits of the file.

The disadvantage of the conventional storage devices with multiple-write capability, such as a disk, is their slow

speed relative to the speed of the internal computer memory. Conversely, the advantage of these storage devices over computer memory include their non-volatility, low cost, and high capacity.

A storage device known as a Flash-EProm (FEProm) has the speed of internal computer memory combined with the nonvolatility of a computer disk. This device is an EProm-type (Erasable, Programmable, Read-Only Memory) device. The contents of the FEProm can be erased by applying a certain voltage to an input rather than by shining ultraviolet light on the device like the typical EProm. The erasing sets each bit in the device to the same value. Like other EProms, the FEProm is a nonvolatile memory. The FEProms are comparable in speed to the internal memory of a computer. Initially, and after erasure, each bit of the FEProm is set to a 1. A characteristic of the FEProm as with other EProms is that a bit value of 1 can be changed to a 0, but a bit value of 0 cannot be changed to a 1. Thus, data can be written to the EProm to effect the changing of a bit from a 1 to a 0. However, once a bit is changed to a 0, it cannot be changed back to a 1, that is, unless the entire FEProm is erased to all ones. Effectively, each bit of the FEProm can only be written once but read many times between subsequent erasures. Moreover, each bit of an FEProm can only be erased and set to 0 a limited number of times. The limited number of times defines the effective life of an FEProm.

The typical time to access an FEProm varies according to the type of access and several other factors. The read access time is in the range of hundreds of nanoseconds, and there is no limit as to the number of times a byte may be read. The write access time is typically in the range of tens of microseconds. The write access time is affected by the number of times the byte has been erased, the device temperature, and the byte-density of the FEProm. Although there is no theoretical limit to the number of times a byte may be written, the erase limit provides a practical write limit. The erase time for an FEProm is in the range of a few seconds. The erase time is affected by the number of times the FEProm has been erased, the device temperature, and the byte-density of the FEProm.

Because conventional file systems assume that the storage device has the multiple-write capability, these file systems are not appropriate for the FEProm, which effectively has only a single-write capability. It would be desirable to have a file system that supports a storage device based on the FEProm. Such a file system would have the speed of computer memory and the nonvolatility of computer disks.

Conventional storage devices, such as computer disks, are block addressable, rather than byte addressable. A byte is the unit of addressability of the internal memory of the computer, that is, the computer can write or read one byte (typically, eight bits) at a time, but not less. When the computer writes to or reads from a disk it must do so in groups of bytes called a block. Block sizes can vary, but typically are a power of two (128, 256, 512, etc.). For example, if only one byte on a disk is to be changed, then the entire number of bytes in the block size must be written. This may involve the reading of the entire block from disk into the computer memory, changing the one byte (the internal memory is byte addressable), and writing the entire block to the disk.

Conventional file systems store data in a way that leaves unused portions of blocks. The file systems store data from only one file in any given block at a time. The file systems do not, for example, store data from one file in the first 50 bytes of a block and data from another file the last 78 bytes

of a 128-byte block. If, however, the length of a file is not an even multiple of the block size, space at the end of a block is unused. In the example above, the last 78 bytes of the block would be unused. When a disk uses a large block size such as 4096, up to 4095 bytes of data can be unused. Although this unused space may be a negligible amount on a disk drive that has multi-write capability and that can store millions of bytes, it may be a significant amount on a storage device without multi-write capability and without the capacity to store millions of bytes of data.

The FEProm, in contrast to typical storage devices, is byte addressable, rather than block addressable. It would be desirable to have a file system that would support the byte addressability of an FEProm.

An FEProm can also be organized in a block-erasable format. A block-erasable FEProm contains a number of blocks, typically 16, that can be independently erased. For example, FIG. **3** shows a schematic diagram of a block-erasable FEProm **301** with 16 blocks, numbered 0 to 15. Each one of the blocks can be independently erased without affecting the contents of the other blocks. Block numeral 14 **302** can be erased without affecting the data in the other blocks. It would be desirable to have a file system that would support the block-erasable FEProm.

DISCLOSURE OF THE INVENTION

It is an object of the present invention to provide a method of storing data on a file storage device and, in particular, a block-erasable, flash-erasable, programmable, read-only memory.

It is another object of the present invention to provide a computer memory manager for allocating and deallocating memory in a block-erasable FEProm.

It is another object of the present invention to provide a method for tracking the number of times a block has been erased in a block-erasable FEProm.

It is another object of the present invention to provide a block-erasable FEProm with a data structure that facilitates memory allocation.

It is another object of the present invention to provide a method of allocating a block for the storage of data.

It is another object of the present invention to provide a method of reclaiming deallocated space in a block-erasable FEProm.

It is another object of the present invention to provide a file system for a block-erasable FEProm.

These and other objects, which will become apparent as the invention is more fully described below, are obtained by a method and system for managing memory in a block-erasable, flash-erasable, programmable, read-only memory. In a preferred embodiment, the system comprises a block-erasable FEProm with a block header, a block allocation table, a data storage area, a block allocation routine for selecting a block in which to store data, a data area allocation routine for selecting an entry in the block allocation table and a portion of the data storage area, and a storage routine for storing the data. In preferred embodiments, the system includes a file manager to implement a file system for the block-erasable FEProm.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1A** shows a sample hierarchical or tree-structured organization of directories and files.

FIG. **1B** shows a linked-list structure that represents the same directory structure of FIG. **1A**.

FIG. **1C** shows an alternate linked-list structure that represents the same directory structure of FIG. **1A**.

FIG. **2A** shows a linked-list structure for the file named "\A\D.DAT".

FIG. **2B** shows an alternate linked-list structure for the file named "\A\D.DAT".

FIG. **3** shows the layout of a block-erasable FEProm in a preferred embodiment of the present invention.

FIG. **4** shows a flow diagram of the Add_Directory routine in a in a preferred embodiment of the present invention.

FIGS. **5** and **6** show before and after pictorials of the directory structure with a newly added directory in a preferred embodiment of the present invention.

FIG. **8** shows a flow diagram of the Add_File routine in a preferred embodiment of the present invention.

FIG. **10** shows a flow diagram of the Extend_File routine in a preferred embodiment of the present invention.

FIG. **11** shows a sample directory and file layout using a preferred embodiment of the present invention.

FIG. **12** shows a flow diagram of the Update_File routine in a preferred embodiment of the present invention.

FIGS. **13** and **14** show a sample portion of a file before and after it is updated in a preferred embodiment of the present invention.

FIGS. **15**, **16**, and **17** show sample portions of a file after the file is updated in a preferred embodiment of the present invention.

FIG. **18** shows a flow diagram of the Del_Directory routine in a preferred embodiment of the present invention.

FIG. **19** shows a flow diagram of the Change_File_Name routine in a preferred embodiment of the present invention.

FIG. **20** shows the before and after representation of the directory structure for a file that has had its name changed.

FIG. **21** shows a flow diagram of the Change_Attribute_Data routine in a preferred embodiment of the present invention.

FIG. **23** shows the layout of a block in a preferred embodiment.

FIG. **23A** shows the layout of the block in FIG. **23** after reclamation in a preferred embodiment.

FIG. **24** shows sample block allocation structures and regions for the sample of FIG. **26**.

FIG. **25** shows the dereferencing of a handle.

FIG. **26** shows a sample block allocation for a portion of the directory hierarchy of FIG. **1B**.

FIG. **27** is a flow diagram of the initialization process in a preferred embodiment.

FIG. **28** is a flow diagram of the Block Allocation routine in a preferred embodiment.

FIG. **29** is a flow diagram of the Region Allocation routine in a preferred embodiment.

FIG. **30** is a flow diagram of the Block Reclamation routine in a preferred embodiment.

FIG. **31** is a state diagram of the states of the Alloc entries.

FIG. **32** is a state diagram of the states of the blocks.

DETAILED DESCRIPTION OF THE INVENTION

In a preferred embodiment, the present invention provides a method and system for the memory management of a

block-erasable FEProm. The system is described as an FEProm manager and a file system. The FEProm manager manages memory allocation and deallocation of the FEProm. The file system is a hierarchical directory system and uses the FEProm manager to allocate and deallocate memory. Alternatively, the FEProm manager and file system can be integrated to achieve certain optimizations. However, the use of a distinct FEProm manager allows an FEProm to store data from different file systems and even non-file system data.

FEProm Manager

The FEProm manager of the present invention provides for allocation of free space, deallocation of allocated space, and reclamation of deallocated space in a block-eraseable FEProm. In preferred embodiments, as shown in FIG. **23**, each block of the FEProm contains a block allocation structure **2302**, data regions **2303**, and free space **2304**. The block allocation structure **2302** contains a header and an allocation array. The header is a fixed-length structure that contains status information for the block. The allocation array is variable length; the entries of the allocation array describe the data regions. Table A shows the data structure of the Block Allocation Structure. The structure is shown in the C programming language format along with a description of the structure variables. The array Alloc is the allocation array and the other variables compose the header. The data regions are variable-length regions that hold the data stored in the FEProm. The free space **2304** is space that is not allocated to the block allocation structure or the data regions. The block allocation structure **2302** and the data regions **2303** are located at opposite ends of the block. As regions are added, the block allocation structure **2302** and the data region grow towards each other as indicated by arrows **2305** and **2306**. The Alloc entries in the block allocation structure contain offsets **2310–2315** to the corresponding region in the block. In a preferred embodiment, the block allocation structure contains data that is specific to the data stored in the regions.

TABLE A

Data Structure

```
struct BlockAllocation
{
    struct
    {
        byte        Status;
        byte        offset[3];
        word        Len;
    }               Alloc[ ];
    dword           BootRecordPtr;
    dword           EraseCount;
    word            BlockSeq;
    word            BlockSeqChecksum;
    word            Status;
}
```

Definitions

Alloc		variable length array structure that defines the regions within the block
Status		status of region
bit #		
5-2	1111	Unused
	1011	Intermediate state
	0111	Free
	0011	Allocated
	0001	Deallocated
	0010	Superseded
	0000	Null

TABLE A-continued

7-6	11	unused entry
	10	last entry
	00	not last entry
Offset		offset, relative to the beginning of the block, of this region
Len		length, in bytes, of region
BootRecordPtr		handle to boot record when the FEProm is used as a file storage device
EraseCount		number of times the block has been erased
BlockSeq		logical sequence of block within the FEProm
BlockSeqChecksum		checksum of block sequence number
Status		
bit #		
1-0	11	boot record not in block (when the FEProm contains file system data)
	10	boot record in block
	00	boot record superseded
15-10	110000	Ready
	0XXXXX	QueuedForErasure
	111111	Erased
	111110	UpdateInProcess
	111100	Spare Block
	111000	ReclamationInProcess
	000000	Retired

The FEProm manager allocates a data region in a block by adding an Alloc entry, setting the variable Offset to point to the first location in free space, setting the variable Len to the length of the region, and setting the variable Status to indicate allocated. The FEProm manager deallocates a region by setting the variable Status in the corresponding Alloc entry to deallocated. Deallocated space is generally not available for reallocation because it may have been set to a non-FNULL value. Deallocated space is set to FNULLs before it is reallocated. The process of setting the deallocated space to FNULLs and making it available for allocation is block reclamation. Deallocated space is reclaimed by copying the allocated regions to second block and setting the Alloc entries in the second block to point to the new offsets. The block reclamation process ensures that the Alloc entries in the second block are in the same position relative to the block header as they were in the copied-from block. The FEProm manager uses a logical block sequence number, variable BlockSeq in the header, to identify a specific logical block of data. As a block is copied during reclamation, the physical block number will change but the logical block sequence number does not change.

The FEProm manager provides a handle to the allocated region rather than the address of the start of the region. The handle contains two parts: (1) a logical block sequence number, which indirectly references a physical block, and (2) an index to an Alloc entry, which indirectly references a region within the physical block. A handle is dereferenced by determining the physical block that corresponds to the logical block sequence number and by accessing the variable Offset in the Alloc entry that is indexed by the index part of the handle, and adding the variable Offset to the starting address of the physical block. This dereferencing produces the address of the region. The use of handles allows the FEProm manager to reclaim a block without adjusting the links to the data regions that may exist. All that needs to be updated is a translation cache (explained below), which resides in memory and maps the logical block sequence numbers to physical blocks, and the offsets in the Alloc array. When a handle is dereferenced, the new offset is used to produce the correct address. FIG. **25** shows the derefer-

encing of a handle **2501**, which has a logical block sequence number **2502** of 9 and an Alloc index **2503** of 3. The block translation cache **2504** maps logical block numbers to physical block numbers. The FEProm manager maintains the cache. When one physical block is moved to another physical block during block reclamation, the FEProm manager adjusts the translation cache to map the logical block sequence number to the new physical block. In the example of FIG. **25**, the logical block sequence number 9 maps to physical block 14 **2505**. The Alloc array for physical block number **14** is indexed by the Alloc index **2503** of the handle **2501**. The variable Offset of the Alloc[3] entry contains the offset of region 3 **2506** in physical block number 14 **2505**. The address of region 3 **2506** is determined by adding the offset to the address of physical block **2505**.

FIG. **29** is a flow diagram of the Region Allocation routine that allocates a new region within a block. The input parameters to this routine are the data to store in the region and length of the data. The routine returns the handle to the allocated region. Alternatively, this routine would not write the data, but just allocate space. The routine assumes that the block has sufficient free space for the region and an additional Alloc entry. FIG. **31** is a state diagram for the states of an Alloc entry. An Alloc entry can be in one of the following states: unused, allocated, deallocated, superseded, free, or null. The entry is also in a transitional state while allocation is in process. If an entry is unused, then it is past the last entry in the Alloc array, that is, it is not part of the array. If the entry is allocated, then the corresponding region is currently allocated. If the entry is deallocated, then the corresponding region is not needed, but the region has not yet been reclaimed. If an entry is superseded, then another Alloc entry(ies) corresponds to the same data region. During reclamation, the data in a superseded entry is ignored because another entry(ies) points to the data region. If the entry is free, then the corresponding region has been reclaimed and the Alloc entry can be reused when a new region is added to the block. If an entry is null, then a problem occurred during writing to the entry and it is not used until erased. If the entry is in the allocation in process transitional state, then some of the data in the entry may be valid but not necessarily all.

Referring to FIG. **31**, all entries are initially in the unused state **3101** and transition to the unused state **3101** whenever the block is erased. An entry transitions from the unused state **3101** to the allocated state **3103** through the allocation in process state **3104**. An entry in the free state **3102** transitions to the allocated state **3103**. An entry in the unused state **3101** transitions to the free state **3102** as a result of a block reclamation when that entry was deallocated or superseded and not located after the last allocated entry in the block to be reclaimed. An entry in the allocated state **3103** transitions to the deallocated state **3105** when the corresponding region is deallocated. An entry in the allocated state **3105** transitions to the superseded state **3106** when another entry or entries are allocated that correspond to the same region. Finally, an entry in any state transitions to the null state **3107** on a write error to the entry.

Referring to FIG. **29**, in block **2901**, the system determines whether an Alloc entry has a status of free. If such an entry is found, then the system will reuse that entry and continues a block **2902**, else the system continues at block **2903**. In block **2902**, the system selects the free Alloc entry and continues at block **2905**. In block **2903**, the system allocates and selects a new Alloc entry and continues in block **2904**. The new Alloc entry is the entry just after the entry marked last. In block **2904**, the system sets the variable Status for the selected Alloc entry to indicate that allocation is in process. The allocation in process state is a transitional state that indicates that the data may be in an inconsistent state. In block **2905**, the system sets the variables Offset and Len and writes the data to a data region starting at the first location in free space. In block **2906**, if the Alloc entry was new, then the system continues at block **2907**, else the system continues at block **2908**. In block **2907**, the system resets the Status of the previous last Alloc entry to indicate that it is no longer the last entry in the Alloc structure. In block **2908**, the system sets the variable Status for the selected Alloc entry to allocated, which indicates that the data is now in a consistent state. The system is then done with the allocation.

As discussed above, performance of a block of an FEProm degrades as the erase count increases. It is preferred to keep the erase count for each of the blocks approximately equal, referred to as "leveled." In normal operation, the erase counts will not be leveled. For example, if executable files are written to a block, there may never be a need to erase that block. Thus, the erase count for that block will remain constant as the erase count for other blocks increases. A preferred embodiment uses several approaches to level the block erase counts. First, during boot-up or whenever an FEProm is loaded (initialization), the FEProm manager scans the blocks to obtain data to manage the FEProm. In a preferred embodiment, this data includes the erase count for each block. To help level the erase counts, the FEProm manager swaps the data in a block with a high erase count with the data in a block with a low erase count. Because this swapping may be time consuming, it is desirable to minimize the number of swaps that occur during initialization. Moreover, the swapping need only be performed when the difference between erase counts exceeds a threshold value or ratio. Second, whenever the FEProm manager performs a reclamation on a block, it preferably selects an available block with a low erase count. Third, whenever the FEProm manager allocates a region, it allocates the region in a block with a low erase count. Fourth, the FEProm manager tracks the number of block erasures. If the number of erasures exceeds a threshold value, the FEProm manager determines whether the difference in erase counts for any two blocks exceeds the threshold value or ratio. If the threshold is exceeded, the manager swaps the data in the blocks.

The FEProm manager preferably maintains the erase count in the header of each physical block. Whenever a block is erased, the FEProm manager writes the incremented erase count back into the block header. Whenever a block is copied, the erase count is not transferred. Each block retains its own erase count. Alternatively, the erase counts could be stored in a single block. However, this alternate method has a couple disadvantages over the preferred method. First, all erase counts could be lost by a single block failure. Second, when a block is erased, the erase count block must be updated. Eventually, the erase count block must be erased and rewritten.

FIG. **28** is a flow diagram of the Block Allocation routine that selects which of the blocks in the block-erasable FEProm is to be used to allocate a region. The FEProm manager determines which block to allocate based on several factors. First, the FEProm manager allocates the block with enough free space that has the lowest erase count. Allocating blocks with low erase counts will help ensure that the blocks are leveled. Second, the FEProm manager allocates multiple blocks if there is no block with sufficient free space. The data is divided into several regions each of which is stored in a different block. Third, the FEProm manager

will not permit allocation when too many fragments will result. The input parameters to this routine are the length of the region to be allocated and whether the region can be stored in multiple blocks. In block **2801**, the system selects all blocks with sufficient free space to store the data. In a preferred embodiment, the system determines the length of the free space based on the location of the start of free space and the number of Alloc entries. This data is preferably stored in an FEProm manager buffer during initialization and updated as needed. The system also ensures that there is sufficient space to add a new Alloc entry, if necessary. In one embodiment, the system performs a block reclamation for those blocks that meet the reclamation criteria before determining if the block has sufficient free space. In an alternate embodiment, no reclamation occurs until it is determined that there is not sufficient free space. In block **2802**, if at least one block was selected, then there is enough free space in a single block to hold the region and the system continues at block **2803**, else the system continues at block **2804**. In block **2803**, the system determines which of the selected blocks have the lowest erase count and allocates that block and the system is done with the block allocation. In block **2804**, the system selects a set of blocks that has enough free space to hold the region data. In block **2805**, if the total free space and deallocated space is not sufficient to hold the data or there would be too many fragments, the system continues at block **2807**, else the system continues at block **2806**. If the region data must be stored in a single block, then selecting two blocks is too many fragments. Also, if the data is to be stored in several blocks, then a reclamation may be appropriate. In block **2806**, the system allocates the selected blocks and the block allocation is done. In block **2807**, if all the blocks have been reclaimed, then there is insufficient room on the FEProm to store the data and the block allocation is done, else the system continues at block **2808**. In block **2808**, the system reclaims a block and loops to block **2801** to determine if there is now enough free space.

FIG. **30** is a flow diagram of the Block Reclamation routine, which reclaims the deallocated regions in a block. The input parameters are the number of the block to be reclaimed and the physical number of a spare block. A block may be reclaimed at several different times. First, a block may be reclaimed (as discussed above) when there is insufficient free space to satisfy an allocation request. Second, the FEProm manager can track the number of write accesses to the FEProm. If the number of writes exceeds a threshold number, then the manager determines whether any blocks should be reclaimed. A block should be reclaimed when the ratio of deallocated space to block size exceeds a threshold value. One skilled in the art would appreciate that block reclamation can occur at other times, such as when an FEProm is first loaded. The FEProm manager reclaims a block by copying the allocated regions to a spare block, a block that has been erased. By copying only the allocated regions, the deallocated regions are reclaimed. Alternatively, the FEProm manager could copy the allocated regions to non-FEProm memory, then erase the block, and copy the regions back to the block. However, this method requires enough non-FEProm memory to store the allocated regions and has the potential to lose data should a power failure occur after erasure but before the block is rewritten. In the preferred method, the FEProm manager copies the allocated regions in the block to be reclaimed to the spare block and copies the Block Allocation Structure adjusting the variable Offset to reflect the new region locations in the spare block. FIG. **23**A show an example of the block layout of FIG. **23** after regions 1 and 5 are reclaimed. The allocated regions have been copied to be contiguous. The corresponding Alloc entries are update to point to the new region locations. The Alloc[1] entry is still needed even though region 1 has been reclaimed. Because of the use of handles, all the Alloc entries must maintain their same position in the Block Allocation Structure. However, the variable Status for the Alloc[1] entry is set to free, and it can be used to point to the next region that is added to the reclaimed block. Since there were no Alloc entries after the Alloc[5] entry, it was not needed as a placeholder for any handles, and it was removed. The position status of Alloc[4] entry indicates that it is the last entry in the Alloc array.

FIG. **32** is a state diagram for the state of a block. The state of a block is stored in the header in variable Status. The states of a block are erased **3201**, update in process **3202**, spare **3203**, reclamation in process **3204**, ready **3205**, queued for erasure **3206**, and retired **3207**. A newly erased block is in the erased state **3201**. An erased block normally transitions to the update in process state **3202** and then to the spare state **3203**. The update in process state **3202** indicates that certain data in the header such as erase count is being updated. Once this update is complete, the block transitions to the spare state **3203**. If the update fails, the block transitions to the queued for erasure state **3206**. A block in the spare state **3203** transitions to the reclamation in process state **3204** when the block is to receive the data from a block being reclaimed. The reclamation in process state **3204** is a transitional state that indicates that the block allocation structure may not be in a consistent state. Once the data is consistent, the block transitions to the ready state **3205**. If, however, an error occurs while in the reclamation in process state **3205**, the block transitions to the queued for erasure state **3206** after a reclamation has occurred for that block. A block in the queued for erasure state **3206**, transitions to the erased state **3201** when erased. If the erase fails, the block transitions to the retired state **3207**. The block stays in the retired state forever. When an FEProm is first initialized, the blocks are set to either the ready state or the spare state. Blocks in the ready state can contain data; blocks in the spare state do not contain data.

Referring to FIG. **30**, in block **3001**, the system sets the variable Status for the spare block to be reclaimed to indicate that reclamation is in process. In block **3002**, the system copies the variables Seq, SeqCheckSum, and BootRecordPtr from the header of the block to be reclaimed to the spare block. In block **3003**, the system determines the position of the last Alloc entry in the allocated state for the block to be reclaimed. During the reclamation process, the Alloc entries after the last allocated entry are ignored. Thus, the reclaimed block will have the states of those entries set to unused. In blocks **3004** through **3010**, the system copies each Alloc entry up to the last allocated entry to the spare block. In block **3004**, the system initializes index j, which indexes the Alloc entries. In block **3005**, if index j is greater than the index of the last allocated entry as determined in block **3003**, then all the Alloc entries and corresponding regions have been copied and the system continues at block **3011**, else the system continues at block **3006**. In block **3006**, if the state of the entry is allocated, then the system continues at block **3007**, else the system continues at block **3009**. In block **3007**, the system copies the region data corresponding to the Alloc[j] entry to the spare block. In block **3008**, the system updates the variable Offset in the Alloc[j] entry of the spare block to indicate the location of the copied region and copies the variables Status (setting the position status as appropriate) and Len and continues at block **3010**. In block **3009**, the system updates the state of the Alloc[j] entry of the

spare block to indicate that the entry is free. In block **3010**, the system increments index j to index the next Alloc entry and loops to block **3005**. In block **3011**, the system sets the state of the spare block to ready. In block **3012**, the system sets the state of the block to be reclaimed to queued for erasure. After the processing for block **3011** is complete but before the processing for block **3012** is complete both the spare block and the block to be reclaimed have valid data. If processing is interrupted before the processing for block **3012** completes, then the FEProm will contain two blocks with the same logical sequence number. The system preferably checks for this condition during initialization of the FEProm. At that time, the system can complete the processing of block **3012**. In block **3013**, the system updates the variables PhysicalBlockNum, FirstFreeByteOffset, LenDeallocatedSpace, and AllocStructCnt in BlockData [Seq] to reflect the state of the spare block (described below). In block **3014**, the system updates the DriveRec to adjust the list of spare blocks and the reclamation is done (described below).

TABLE B

Data Structures

```
struct          DriveRec
{   word        BlockCnt
    word        SpareBlockCnt
    dword       BlockSize
    word        RootDirPtr
    word        SpareBlockPtr[ ]
}
struct          ConfigRec
{
    word        WriteAccessCntThreshold
    word        EraseCntThreshold
    word        BlockReclamationThreshold
    word        BlockEraseLevelingThreshold
}
struct          BlockRec
{
    byte        Flags
    word        PhysicalBlockNum
    dword       FirstFreeByteOffset
    dword       LenDeallocatedSpace
    word        AllocStructCnt
    dword       BlockEraseCnt
    word        FirstUseableAllocEntry
    word        FreeAllocEntryCnt
}   BlockData[ ]
```

Definitions

BlockCnt	number of physical blocks in the FEProm
BlockSize	number of bytes in a block
RootDirPtr	handle to the data region that contains the root directory when the FEProm is used as a file storage device
SpareBlockPtr[ ]	variable-length array that contains the physical block numbers of the spare blocks
SpareBlockCnt	number of spare blocks
WriteAccessCntThreshold	number of writes to the FEProm that will cause the system to determine if any blocks should be reclaimed
EraseCntThreshold	number of erases to the FEProm that will cause the system to determine if block leveling should occur
BlockReclamationThreshold	ratio of deallocated space to block size that triggers block reclamation
BlockEraseLevelingThreshold	difference between minimum and maximum erase counts that will trigger the leveling process between the blocks
PhysicalBlockNum	physical block number that the contains the logical block
FirstFreeByteOffset	offset in the physical block of the first byte of free space
LenDeallocatedSpace	total length of the deallocated regions in the physical block
FirstUseableAllocEntry	index of first useable Alloc entry in block
FreeAllocEntryCnt	number of free Alloc entries in block
AllocStructCnt	number of Alloc entries
BlockEraseCnt	number of times the physical block was erased

When an FEProm is first loaded, the FEProm manager scans the FEProm to initialize internal data structures shown in Table B. The structure DriveRec contains data relating to the device, the structure ConfigRec contains data relating to configuration parameters, and the array BlockData contains an entry with data for each physical block in the FEProm. The array BlockData is the block translation cache. During initialization, the FEProm manager initializes each of the variables in array BlockData and the variables relating to spare blocks in the structure DriveRec. The other variables in the struct DriveRec are system defined variables. In a preferred embodiment, the FEProm manager stores information that is specific for the type of region data in these data structures. For example, if the FEProm is used as a file storage device, the data structures may contain the handle to the root directory. FIG. **27** is a flow diagram of the initialization process in a preferred embodiment. This procedure initializes the DriveRec and the BlockData structures by scanning the block allocation structure for each block on the FEProm. In blocks **2701** through **2709**, the system loops reading data from each block. In block **2701**, the system initializes index i, which indexes the physical block currently being accessed. In block **2702**, if index i is greater than the number of blocks in the FEProm, then all blocks have been scanned and the system continues at block **2710**, else the system continues at block **2703**. In block **2703**, the system reads the header for the block indexed by index i. In block **2704**, the system updates the DriveRec and BlockData [i] data. If the block is a spare block, then the system increments SpareBlockCnt and adds the block to the SpareBlockPtr array. In a preferred embodiment, the system also scans for information that is specific for the data stored in the regions. For example, if the FEProm is used as a file system then if the block contains the boot record, the system sets the BootRecPtr, reads the boot record, and sets the RootDirPtr. In blocks **2705** through **2708**, the system loops processing the data in each Alloc entry. In block **2705**, the system initializes index j, which indexes the Alloc entries. In block **2706**, if the system has processed the last Alloc entry, then the system continues at block **2709**, else the system continues at block **2707**. In block **2707**, the system updates the BlockData[BlockSeq] data based on the Alloc entry indexed by j. The system updates the variables FirstFreeByteoffset, LenDeallocatedSpace, and AllocStructCnt. The system sets the variable Physical BlockNum to index i, which initializes the translation cache. In block **2708**, the system increments

index j to index the next Alloc entry and loops to block **2706**. In block **2709**, the system increment index i to index the next block in the FEProm and loops to block **2702**. In block **2710**, the system then levels the block usage. The system scans the BlockData array to determine the block with the maximum BlockEraseCnt and the block with the minimum BlockEraseCnt. The system then swaps the data between the blocks. The system first copies the maximum block to a spare block. The system then erases the maximum block. The system copies the data from the minimum block to the erased block and preferably performs a block reclamation while copying. The system erases the minimum block and copies the data from the spare block to the minimum block and preferably performs a block reclamation while copying.

The FEProm manager of the present invention can support media that is not blocked or that is not erasable. The block reclamation and block erase count leveling processes rely on block-erasability. Thus, these processes should be disabled if the media does not support block erasability. In a preferred embodiment, these processes can be effectively disabled by setting the spare block count to 0. The FEProm manager relies on at least one spare block to activate these processes. If the media is not blocked, an arbitrary block size can be selected as a logical block. In a preferred embodiment, the size of the block should not be so large that the offset in the allocation array entries cannot address the entire block and should not be so small that the block header and allocation array comprise an appreciable percent of the block size or that the translation cache is too large.

The FEProm manager allows for dynamic recovery from FEProm write and erase errors. A write error is generated whenever a memory location cannot be set to the value specified. These errors can be caused either by failure of the hardware or by trying to write a value to a memory location that requires a 1 in a certain bit which already has a been changed to a 0.

Write errors can occur while writing to a data region, a block header, and an allocation array entry. In a preferred embodiment, when a write error occurs while writing to a data region, the FEProm manager sets the block to the deallocated state. The FEProm manager then tries to write the data to a different data region by restarting the region allocation process as shown in FIG. **29**.

When a write error occurs while writing to an allocation array entry, the FEProm manager sets the allocation array entry to the null state. If the error occurred while setting an entry to the superseded, deallocated, free, or allocation in process state, then the setting of the entry to the null state will leave the FEProm in a consistent state. However, if the error occurred while setting an entry to the allocated state, then the FEProm will be in an inconsistent state because the data region has no corresponding allocation array entry that is in the allocated state. The FEProm manager allocates another entry for the data region. An error may also occur while setting an entry to the null state. Since the null state is defined as a status value of Os, then an error when setting an entry to the null state is necessarily a hardware error. If an error occurs, then a reclamation may be needed that specifies that the corresponding region must be reclaimed. For example, if an error occurs while setting an entry to the deallocated state and again while setting the entry to the null state, then the entry will be in the allocated state. If left in this state, this entry and corresponding data region would never be reclaimed under normal reclamation.

When an error occurs while writing to a block header, then the FEProm manager sets the block to the queued-for-erasure state. If an error occurs while setting a block to the queue-for-erase state, the FEProm manager sets the block to the retired state. If an error occurs while setting a block to the retired state, then the error may be unrecoverable.

When a write error occurs while erasing a block, the FEProm manager sets the block to the retired state. If the retired block was a spare block, then the FEProm manager operates with fewer spare blocks. Alternatively, the FEProm manager attempts to locate a block with no allocation array entries that are allocated. The FEProm manager then erases the located block and sets it to the spare state. Whenever no spare blocks are available, then the FEProm manager treats the FEProm as if it were not erasable as discussed above.

The present invention also provides for dynamic error recovery should the FEProm be in an inconsistent state. For example, referring to FIG. **29**, if the FEProm is removed after the offset is written in block **2905**, but before the status is updated from the free state to the allocated state in block **2908**, then the FEProm will be in an inconsistent state. When the FEProm is next loaded, the FEProm manager would see that the allocation entry is free and attempt to reuse it. However, an attempt to write to the offset would fail (unless the same data was being written to the offset). As discussed above, the FEProm manager would recover by setting the entry to the null state and would restart the region allocation process to select a different entry. If a portion of the data is written to the data region before the FEProm is removed, then an error would be detected when the FEProm manager tries to write data over that region. This error would be handled as described above.

File System

The present invention provides a directory-based hierarchical file system for an FEProm device. A hierarchical file system provides for storing files in logical groupings. A preferred embodiment uses a linked-list data structure to implement both the directory hierarchy and the internal file storage. FIG. **1A** shows a typical hierarchical directory structure. The MS-DOS operating system, which is available from Microsoft Corporation of Redmond, Washington, implements a file system with a hierarchical directory structure. As shown in FIG. **1A**, the directory ROOT **100** contains two subdirectories (DOS **102** and WORD **103**) and two files (AUTOEXEC.BAT **104** and COMMAND.COM **105**). The directory DOS **102** contains one file (FORMAT.EXE **106**). The directory WORD **103** contains two subdirectories (DAVID **107** and MARY **108**) at the next lower level. The directory DAVID **107** contains one file LETTER1.DOC **109**. The directory MARY **108** contains three files LETTER1.DOC **110**, LETTER2.DOC **111**, and LETTER3.DOC **112**.

FIG. **1B** shows a possible linked list that implements the directory structure of FIG. **1A** in a preferred embodiment. The directory ROOT record **100** (the terms record and entry are used interchangeably in this specification) has a pointer **120**, which points to a linked list **140** of subdirectory and file records at the next lower level. The linked list **140** comprises directory records DOS **102** and WORD **103** and file records AUTOEXEC.BAT **104** and COMMAND.COM **104** linked by pointers **121**, **122**, **123**. The subdirectory record DOS **102** has a pointer **124** to the file record **106** at the next lower level, and the subdirectory record WORD **103** has a pointer **125** to a linked list **141** of subdirectory records at the next lower level. The linked list **141** comprises directory records DAVID **107** and MARY **108** linked by pointer **126**. The subdirectory record DAVID **107** has a pointer **127** to the file at the next lower level, and the subdirectory record MARY **108** has a pointer **128** to a linked list **142** of file records at the next lower level. The linked list **142** comprises file

records LETTER1.DOC **110**, LETTER2.DOC **111**, and LETTER3.DOC **112** linked by pointers **129** and **130**. The template **10** shows the record layout used throughout the drawings. In a preferred embodiment, the records shown in FIG. **1**B are DirEntry and FileEntry structures as described below.

FIG. **1**B represents just one possible linked list arrangement that represents FIG. **1**A. The arrangement would be different if files had been added but then deleted or if the name of a directory was changed. FIG. **1**C shows another possible arrangement. FIG. **1**C represents the same directory hierarchy as FIG. **1**A, but the directory BILL **113** existed at one time but has been deleted. Because an FEProm device can be written only once (unless erased), in a preferred embodiment of the present invention, directory record BILL **113**, as shown in FIG. **1**C, is not physically removed from the linked list. A directory or file record is deleted from the linked list by setting the status of the directory or file entry. If the directory or file was stored on a computer disk, then directory record BILL **113** could be physically removed by rewriting the pointer **131** in directory record DAVID **107** to point to directory record MARY **108**.

A preferred embodiment also uses a linked-list data structure to link the extents that compose a file.

Each file has a file record associated with it that contains, among other data, the name of the file and that is linked into the directory hierarchy as described above. An extent is a contiguous area of memory that contains data for the file. Each file comprises one or more extents, which contain the file data. Each extent has an extent record associated with it. The extent record contains, among other data, a pointer to the extent and the length of the extent. FIG. **2**A shows the extents of the file "\A\D.DAT" **202**. The extent records R1 **203**, R2 **204**, and R3 **205** are linked and contain a pointer to the corresponding extents E1 **211**, E2 **212**, and E3 **213**. The file is the logical concatenation of extents E1 **211**, E2 **212**, and E3 **213**. In a preferred embodiment, the extent records are FileInfo structures as described below.

FIG. **2**A represents just one possible linked list arrangement for file "\A\D.DAT" **202**. FIG. **2**B shows another arrangement that represents the same file. The extent E4 **214** was added to the file but then deleted. In a preferred embodiment, the extent record R4 **206** is not physically removed from the linked list of extents that compose the file. Rather, the extent record R4 **206** is logically removed by setting status of the record to indicate the record is deleted.

Tables C and D contain several data structures used in a preferred embodiment of the present invention. The data structure shown in Table C is the BootRecord. The BootRecord contains some general information relating to the identification of the file system, the version number of file system that can access the FEProm, pointer to the root directory, and additional data as explained below. The first and second structures shown in Table D are the DirEntry and FileEntry structures. One of these structures is allocated for each directory and file. The structures are identical. The variable SiblingPtr points to the next sibling in the linked list of DirEntry and FileEntry structures at the same level in the directory hierarchy. The variables PrimaryPtr and SecondaryPtr are fully described below. The third structure is the FileInfo structure. Each file extent has an associated FileInfo structure. The variable PrimaryPtr points to the FileInfo structure for the file.

TABLE C

Data Structure

```
struct BootRecord
{   word    Signature;
    dword   SerialNumber;
    word    FFSWriteVersion;
    word    FFSReadVersion;
    word    TotalBlockCount;
    word    SpareBlockCount;
    dword   BlockLen;
    dword   RootDirectoryPtr;
    word    Status;
    byte    VolumeLabelLen;
    word    BootCodeLen;
    byte    VolumeLabel[ ];
    byte    BootCode[ ];
}
```

Definition

Signature	a value which indicates that the media supports this file system
SerialNumber	combined with VolumeLabel is a unique identifier for the particular FEProm
FFSWriteVersion	version number in high byte and revision number in low byte of file system that is required to write to this volume
FFSReadVersion	version number in high byte and revision number in low byte of the earliest version of file system that is required to read this volume
TotalBlockCount	total number of blocks, including spare blocks, in the FEProm
SpareBlockCount	number of blocks available for block reclamation and error recovery
BlockLen	length of a block in bytes
RootDirectoryPtr	pointer to the root directory
Status	data specifying file name formats
VolumeLabelLen	number of characters in the volume label
BootCodeLen	number of bytes in boot code array; if 0 then media is not bootable
VolumeLabel[ ]	volume label
BootCode[ ]	boot code for the operating system

TABLE D

Data Structure

```
struct DirEntry
{
    word    Status;
    dword   SiblingPtr;
    dword   PrimaryPtr;
    dword   SecondaryPtr;
    byte    Attributes;
    word    Time;
    word    Date;
    byte    NameLen;
    byte    Name[8];
    byte    Ext[3];
}
struct FileEntry
{
    word    Status;
    dword   SiblingPtr;
    dword   PrimaryPtr;
    dword   SecondaryPtr;
```

TABLE D-continued

```
    byte        Attributes;
    word        Time;
    word        Date;
    byte        NameLen;
    byte        Name[8];
    byte        Ext[3];
}
struct FileInfo
{
    word        Status;
    dword       ExtentPtr;
    dword       PrimaryPtr;
    dword       SecondaryPtr;
    byte        Attributes;
    word        Time;
    word        Date;
    word        ExtentLen;
    word        UncompressedExtentLen;
}
```

Definitions

Name	directory/file name
Ext	file extension
Status	
bit #	
0	1:record exists in the directory structure (Exists) 0:record has been deleted from the directory structure (Deleted)
1	1:record contains current attributes, date, and time data (ATDRecent) 0:record contains data that has been superseded or no data (ATDSuperseded)
3-2	11:Undefined 10:FileInfo 01:FileEntry 00:DirEntry
4	1:PrimaryPtr is not valid 0:PrimaryPtr is valid (PrimaryPtrValid)
5	1:SecondaryPtr is not valid 0:SecondaryPtr is valid (secondaryPtrValid)
6	1:Sibling Ptr/ExtentPtr is not valid 0:SiblingPtr/ExtentPtr is valid (SiblingPtrValid/ExtentPtrValid)
DirEntry	
15-7	reserved
FileEntry	
7	1:file is not compressed 0:file is compressed
15-8	identification of compression algorithm
FileInfo	
7	1:file is not compressed 0:file is compressed
8	1:extent does not contain first segment of a compressed block 0:extent contains first segment of a compressed block
9	1:extent does not contain last segment of a compressed block 0:extent does contain last segment of a compressed block
15-10	reserved
SiblingPtr	pointer to next DirEntry or FileEntry in sibling chain
ExtentPtr	pointer to the extent associated with the FileInfoEntry
PrimaryPtr	DirEntry: points to the first DirEntry or FileEntry on the next lower level in the directory hierarchy FileEntry: points to the linked list of FileInfo entries associated with the file FileInfo: points to the next FileInfo entry for the file
SecondaryPtr	DirEntry: points to the next DirEntry entry for the directory; all contents of this entry except the SecondaryPtr are invalid and are to be ignored FileEntry: points to the next FileEntry entry for the file; all contents of this entry except the SecondaryPtr are invalid and are to be ignored FileInfo: points to the next FileInfo entry for the file
Attributes	file attributes such as read-only, read/write, etc.
Time	time of creation or modification
Date	date of creation or modification
NameLen	length in bytes of name and extension
Name[8]	name
Ext[3]	Extension
ExtentLen	length of extent in bytes

The file system of the present invention distributes the directory and file structures across block boundaries in a block-erasable FEProm. The file system uses the FEProm manager to allocate and deallocate storage in the FEProm. The file system uses handles as described above as pointers in the linked lists. In the following, term handles and pointers are used interchangeably. FIG. **26** shows a sample block allocation for a portion of the directory hierarchy of FIG. **1**B. The portion shown in FIG. **26** comprises the DirEntry and FileEntry records for the directory ROOT, directory DOS, directory WORD, file AUTOEXEC.BAT, and file COMMAND.COM. Block 0 contains directory DOS, and directory WORD; block 12 contains directory ROOT and file COMMAND.COM; and block 14 contains file AUTOEXEC.BAT.

FIG. **24** shows sample block allocation structures and regions for the sample of FIG. **26**. FIG. **24** shows block 0 **2401**, block 12 **2402**, and block 14 **2403**. Block 0 **2401** contains regions **2420** and **2430**. Region **2420** contains the DirEntry for directory DOS, and region **2430** contains the DirEntry for directory WORD. Block 0 **2401** also contains header **2404**. Alloc[0] entry **2421** corresponding to region **2420**, and Alloc[1] entry **2431** corresponding to region **2430**. Block 12 **2402** contains regions **2410** and **2450**. Region **2410** contains the DirEntry for directory ROOT, and region **2450** contains the FileEntry for file COMMAND.COM. Block 12 **2402** also contains the block header **2405**, Alloc[0] entry **2411** corresponding to region **2410**, and Alloc[1] entry **2451** corresponding to region **2450**. Block 14 **2403** contains regions **2490** and **2440**. Region **2490** contains the boot record, and region **2440** contains the FileEntry for file AUTOEXEC.BAT. Block 14 **2403** also contains the block header **2406**, Alloc[0] entry **2491** corresponding to region **2490**, and Alloc[1] entry **2441** corresponding to region **2440**.

In FIG. **24**, the pointers **2407**, **2413**, **2423**, **2433**, **2443**, **2453**, and **2493** define the directory hierarchy starting from pointer **2407** to the boot record in block header **2406**. The boot record **2490** is in block 14 **2403**. The variable Status for block 14 **2403** indicates that the block contains the boot directory. The BootRecordPtr **2407** points to the Alloc[0] entry **2491** for the boot record. Alloc[0] entry **2491** contains the variable Offset **2492**, which contains the offset of region **2490**. Region **2490** contains the boot record. The boot record contains pointer RootDirectoryPtr **2493**, which points to the

Alloc[0] entry **2411** corresponding to directory ROOT. The Alloc[0] entry **2411** contains variable Offset **2412**, which contains the offset of region **2410**. Region **2410** contains the DirEntry for directory ROOT. The PrimaryPtr **2413** of directory ROOT points to Alloc[0] entry **2421** corresponding to directory DOS. Alloc[0] entry **2421** contains the variable Offset **2422**, which contains the offset of region **2420**. Region **2420** contains the DirEntry for directory DOS. The pointer SiblingPtr **2423** of directory DOS points to Alloc[1] entry **2431** for directory WORD. The Alloc[1] entry **2431** contains the variable Offset **2432**, which contains the offset of region **2430**. Region **2430** contains the DirEntry for directory WORD. The pointer SiblingPtr **2433** of directory WORD points to Alloc[1] entry **2441** for file AUTOEXEC.BAT. The Alloc[1] entry **2441** contains variable Offset **2442**, which contains the offset of region **2440**. Region **2440** contains the FileEntry for file AUTOEXEC.BAT. The pointer SiblingPtr **2443** for file AUTOEXEC.BAT points to the Alloc[1] entry **2451** for file COMMAND.COM. The Alloc[1] entry **2451** contains variable Offset **2452**, which contains the offset of region **2450**. Region **2450** contains the FileEntry for file COMMAND.COM. The SiblingPtr **2453** is set to FNULL indicating the end of the linked list.

The file system allows for directories to be added and deleted, and files to be created, extended, and modified. FIG. **4** shows a flow diagram for the routine that adds a directory to the FEProm. The input parameters to this routine are the complete pathname of the new directory and attribute data for the new directory. This routine will set the variable P to contain the address of the DirEntry for the parent directory and the variable C to contain the address of the DirEntry for the child directory. For example, the path name "\P\C" means that a directory C is to be created that is a subdirectory of P, which is a subdirectory of the root directory. FIG. **5** shows when directory C would be the first subdirectory of P, and FIG. **6** shows when directory C would not be the first subdirectory of P. Referring to FIGS. **5** and **6**, the solid lines show the directory structure before directory C is added and the broken lines show the directory structure after directory C is added. In block **401** of FIG. **4**, the system locates directory P by following the path from the root directory and setting variable P to point to DirEntry for directory P. When locating directory P, the system follows the variable PrimaryPtr unless superseded by the variable SecondaryPtr. In block **402**, the system allocates a region for the DirEntry for directory C. The system allocates the region by invoking the procedures of the FEProm manager. The system sets the variable C to point to the allocated region. In the following, the handle that is returned from the FEProm manager is referred to as the pointer to the regions. In block **403**, the system sets the variables Name, Time, Date, and Attributes in the newly allocated record and sets the variable Status to indicate that the newly allocated entry is a directory entry.

In blocks **405** through **412**, the system links the new directory entry into the old directory structure. In blocks **406** through **410**, the system handles the situation where the new directory is not the first subdirectory of P. In blocks **411** and **412**, the system handles the situation where the new directory is the first subdirectory of P. In block **405**, if P->Status indicates the PrimaryPtr is valid, then directory P has or has had a subdirectory, and the system continues at block **406**, else directory P has had no subdirectory and the system continues at block **411**. In block **411**, the system sets P->PrimaryPtr to point to directory C, the newly allocated directory entry to effect the linking to the new directory. In block **412**, the system sets P->Status to indicate that the variable PrimaryPtr is valid and then the routine is done.

In block **406**, the system sets the variable next_ptr equal to P->PrimaryPtr. The variable next_ptr contains the pointer to the next directory in the linked list of sibling subdirectories. In block **407**, if Status of the record pointed to by next_ptr indicates SiblingPtr is valid, then the system continues at block **408**, else the end of the linked list of siblings has been reached and the system continues at block **409**. In block **408**, the system sets next_ptr equal to the SiblingPtr of the record pointed to by next_ptr, which advances next_ptr to point to the next directory in the linked list, and continues at block **407** to determine if the end of the linked list has been reached. When searching for the end of the linked list of siblings, the system follows the variable SiblingPtr. In block **409**, the system sets SiblingPtr of the record pointed to by next_ptr equal to the pointer to DirEntry for directory C. In block **410**, the system sets Status of the record pointed to by next_ptr to indicate that the SiblingPtr in the entry that points to the newly allocated directory entry is valid and then the routine is done.

FIG. **8** shows a flow diagram of the routine that adds a FileEntry record into the file system for a new file. Since FileEntry records are simply leaf nodes of the hierarchical tree-structured file system, the routine that adds the FileEntry records is very similar to the routine for DirEntry records, which is Add_Directory, shown in FIG. **4**. The significant difference is that the variable Status is set to indicate the record is a file in block **803**.

FIG. **10** shows a flow diagram of the routine to add data onto the end of a file. This routine is passed the complete pathname, the data to write, and the number of bytes to write. FIG. **11** shows a sample layout of the directory structure that contains the file \L.DAT that is to be extended. The solid lines show the structure before the file is extended and the broken lines show the structure after the file is extended. Initially, the file L.DAT has FileEntry record **1101**, FileInfo record **1102**, and extent **1103** associated with it. The broken lines represent a FileInfo record **1104** with the data to be added to the file in extent D2 **1105**.

Referring to FIG. **10** in block **1001**, the system allocates a region for new FileInfo record in the FEProm and sets the variable FI to point to that record. In block **1002**, the system allocates a region for the data extent and sets the variable D to point to the extent. In block **1003**, the system writes the data to the allocated block. In block **1004**, the system sets the variables Attributes, Time, and Date in the allocated FileInfo entry. In block **1005**, the system sets FI->ExtentPtr to the handle of the allocated extent. In block **1005**A, the system sets FI->ExtentLen to contain the length of the extent. In block **1005**B, the system sets FI->Status to Exists, ATDRecent, FileInfo, and ExtentPtrValid. In block **1006**, the system locates the FileEntry record for the file to be extended and sets FE to the address of that record. In a preferred embodiment, the system would locate the FileEntry record before allocating the new extent and FileInfo record to ensure that the file exists before any allocation is done.

In blocks **1007** through **1012**, the system locates the last FileInfo record (if one exists) for the file to be extended. The system follows the PrimaryPtr or the SecondaryPtr of the FileEntry record and the FileInfo records. A valid SecondaryPtr indicates that the record pointed to by the PrimaryPtr has been superseded by the data in the record pointed to by the SecondaryPtr. In block **1007**, the system sets pointer next_ptr equal to the pointer to the FileEntry record. In block **1008**A, the system sets the pointer prev_ptr equal to next_ptr. When the last FileInfo record in the file is located, the pointer prev_ptr will contain the pointer to that record.

In block **1009**, if Status of the record pointed to by next_ptr indicates that the SecondaryPtr is valid, then the data in the record pointed to by the PrimaryPtr has been superseded and the system continues at block **1011**, else the system continues at block **1010**. In block **1010**, the system sets next_ptr equal to PrimaryPtr of the record pointed to by next_ptr to get the pointer to the next record in the linked list and continues at block **1012**. In block **1011**, the system sets next_ptr equal to SecondaryPtr of the record pointed to by next_ptr to get the pointer to the next record in the linked list and continues at block **1008**A. In block **1012**, if next_ptr is valid, then the end of the linked list has been reached and the system continues at block **1013**, else the system continues at **1008**A to process the next record in the linked list. In block **1013**, the system sets PrimaryPtr of the record pointed to by prev_ptr equal to the pointer to FI to effect the extending of the file. In block **1014**, the system sets Status of the record pointed to by prev_ptr equal to PrimaryPtr-Valid and the routine is done.

FIG. **12** shows a flow diagram for the routine, Update_File, that updates the data in a file. The parameters for this routine are R, the address of the FileInfo block that is to have its associated extent modified; extent_offset, the offset into the extent for the new data; new_data, the new data; and data_length, the length of the new data. Since the FEProm is effectively a write once device, at least until a block is erased, a region where data is stored cannot be rewritten when an update to a file occurs. In a preferred embodiment, the updated data is written to a different region of the FEProm, as described below.

FIG. **13** shows a typical portion of the linked list of the FileInfo records for a file. The Update_File routine will replace the data represented by the shaded area **1301**. FIG. **14** shows the structure of the linked list after the modified data has been written to the FEProm. Three FileInfo records R1 **1411**, R2 **1412**, and R3 **1413**, have been inserted into the linked list. The entire extent is not rewritten, rather only the portion that actually changed is rewritten. The routine divides the extent into three sections, D1 **1401**, D2 **1402**, and D3 **1403**. Sections D1 **1401** and D3 **1403** contain data that is not changed by the update, and section D2 **1402** contains the data that will change. Each section will have a corresponding FileInfo record. The FileInfo records R1 **1411**, R2 **1412**, and R3 **1413** are linked through their PrimaryPtr fields. Also, the ExtentPtr field in R1 **1411** and R3 **1413**, are set to point to their corresponding extent sections, and the ExtentLen fields are set. A new extent is allocated for the new data corresponding to the section new D2 **1404**, which is pointed to by record R2 **1412**. The SecondaryPtr of record R **1410** points to FileInfo R1 **1411** to indicate that the PrimaryPtr of R **1410** is superseded. The PrimaryPtr of FileInfo record R3 **1413** is set to the value contained in the PrimaryPtr of FileInfo record R **1410** to complete the link.

The Update_File routine depends upon there being sufficient space in the block that contains the extent to add three new Alloc entries. These three Alloc entries will redefine the extent into three regions rather than one region. If there is not sufficient space, then a reclamation of the block may produce enough free space, and the Update_File routine can be invoked. If, however, there is not sufficient free space, the data in the extent is moved to a new extent. If the data is moved, then the new data can be integrated with the old data and written to one region in the new block with only one FileInfo record. The region in the old block is deallocated. In a preferred embodiment, the FEProm manager supports adding Alloc entries to point to portions of an existing region. The example of FIG. **14** would need three new Alloc entries added to the block, which would correspond to newly defined regions associated with D1, D2, and D3. The Alloc entry for D2 would be deallocated and the Alloc entries for D1 and D3 would be allocated. The status of the old Alloc entry that corresponded to the region comprising sections D1, D2, and D3 would be set to indicate that it has been superseded. A status of superseded indicates that the Alloc entry is essentially deallocated with no corresponding region.

In block **1201** of FIG. **12**, the system allocates three regions for the FileInfo records and sets the variables R1, R2, and R3 to contain the addresses of the regions. In block **1202**, if R->Status indicates ATDRecent, then the system sets R1->Time, R1->Date, and R1->Attributes to the values in R and sets R1->Status to ATDRecent, else the system leaves these fields FNULL. In a preferred embodiment, the FEProm manager supports the setting of the Alloc entries to superseded and allocating Alloc entries for an existing region. In block **1203**, the system allocates a region for the new data and sets R2NewData to the address of the region. In block **1204**, the system allocates three Alloc entries. The entries are initialized to point to D1, D2, and D3. The status of the Alloc entry that pointed to the region comprising D1, D2, and D3 is set to superseded. In block **1205**, the system writes new_data to the new region addressed by R2NewData. In blocks **1206** through **1208**A, the system sets the data in FileInfo record R2. In block **1206**, the system sets R2->ExtentPtr equal to the pointer to the region for the new data. In block **1207**, the system sets R2->ExtentLen equal to the length of the new region. In block **1208**, the system sets R2->PrimaryPtr to the pointer to R3. In block **1208**A, the system sets R2->Status to indicate the ExtentPtr and PrimaryPtr are valid.

In blocks **1209** through **1211**A, the system sets the data in FileInfo record R3. In block **1209**, the system sets R3->ExtentPtr equal to the pointer to the D3 region. In block **1210**, the system sets R3->ExtentLen equal to the length of the D3 region. In block **1211**, the system sets R3->PrimaryPtr equal to R->PrimaryPtr. In block **1211**A, the system sets R3->Status to indicate that ExtentPtr and PrimaryPtr are valid.

In blocks **1212** through **1214**A, the system sets the data in FileInfo record R1. In block **1212**, the system sets R1->ExtentPtr equal to the pointer to region D1. In block **1213**, the system sets R1->ExtentLen equal to the length of region D3. In block **1214**, the system sets R1->PrimaryPtr to the pointer to R2. In block **1214**A, the system sets R1->Status to indicate that ExtentPtr and PrimaryPtr are valid.

In block **1215**, the system sets R->SecondaryPtr equal to the pointer to R1. In block **1216**, the system sets R->Status to indicate that the SecondaryPtr is valid. Then the routine is done.

FIGS. **15** and **16** show the FileInfo list for a couple special cases of file updates. The routine for processing for these special cases is a subset of the routine needed for processing the general case, Update_File, shown in FIG. **12**. In FIG. **15**, data starting at the beginning of an extent is updated. Section D1 **1501** contains the data at the beginning of the extent to be updated and section D2 **1502** contains the data at the end of the extent that is not updated. Only two new FileInfo records are needed. The first FileInfo record R1 **1511** points to the new data **1503** and the second FileInfo record R2 **1512** points to the old data in section D2 **1502**. A similar situation occurs when data that ends on an extent boundary is updated as shown in FIG. **16**. As in the general case for a file update, the old region that contains D1 and D2

is subdivided into two regions by allocating two new allocation table entries in the block that contains the old region. Also, if there is not sufficient space for the entries, the unmodified data is moved to a new block.

FIG. **17** shows a linked list for FileInfo records when the updated data spans extent boundaries.

FIG. **18** shows a flow diagram of a routine that deletes a directory from the FEProm. The routine to delete a file is similar except that the associated FileInfo records are deallocated. This routine sets the status of the DirEntry to indicate it is deleted. In block **1801**, the system locates the directory to be deleted and sets the variable pointer D to contain the address of the directory. In block **1802**, the system sets D->Status to indicate that the directory is deleted.

The name of a directory or file is changed by allocating a new DirEntry or FileEntry, respectively, and then setting the SecondaryPtr of the old entry to point the new entry. FIG. **20** shows the file entry for "D.DAT" in solid lines and the changes in broken lines when the name is changed to "B.DAT". The new entry points to the linked list of FileInfo entries, the directory structure, and the extents associated with the old entry.

FIG. **19** is a flow diagram of a preferred subroutine that implements the changing of a file name. The subroutine for changing a directory name is similar, except that there are no associated extents. The input parameters to this routine are the pathname of the file and the new file name. In block **1901**, the system searches through the directory structure and locates the file whose name is to be changed and sets the variable P to point to the FileEntry. The system searches for the last FileEntry in the linked list of entries for the file. A file will have an entry for each name change.

In block **1904**, the system allocates a region for the new FileEntry and sets the variable C to point to the region. In block **1905**, the system sets C->Name equal to the new file name and sets C->Attributes, C->Date, C->Time, and sets C->Status based on the file entry being superseded to ATDRecent. In block **1906**, the system sets C->SiblingPtr equal to P->SiblingPtr to link the entry into the directory hierarchy. In block **1909**, the system sets C->PrimaryPtr equal to P->PrimaryPtr to link the new entry to the list of extents. In block **1909A**, the system sets C->Status to indicate ExtentPtr and PrimaryPtr are valid. In block **1910**, the system sets P->SecondaryPtr equal to the pointer to C. In block **1910A**, the system sets P->Status to indicate SecondaryPtr is valid to complete the replacement of the old entry and the routine completes.

FIG. **21** shows a flow diagram of a routine that changes the attribute data of a file. The attribute data associated with a file is changed by selecting the first FileInfo entry with a status of ATDRecent and with the attribute, date, and time fields set to FNULL. If no such field exists, then a new FileInfo entry is created and selected. The system then sets the attributes, date, and time fields in the selected FileInfo entry. The FileInfo entry that previously stored the most recent attribute, date, and time data has its status set to ATDSuperseded. The input parameters are the pathname and the attribute data. In block **2101**, the system searches through the directory structure to locate the file and sets the variable P to point to the FileEntry. In block **2102**, if P->Status indicates ATDRecent, then the FileEntry contains the most recent attribute data and the system continues at block **2103**, else the system continues at block **2104**. In block **2103**, the system sets variable X to variable P and continues at block **2111**. In block **2104**, the system sets variable C equal to P->PrimaryPtr. In blocks **2105** through **2108**, the system loops searching for the FileInfo entry with status indicating the most recent attribute data. In block **2105**, if C->Status indicates that the SecondaryPtr is valid, then the system continues at block **2106**, else the system continues at block **2107**. In block **2106**, the system sets variable C equal to C->SecondaryPtr to point to the overriding entry and loops to block **2105**. In block **2107**, if C->Status indicates ATDRecent, then the FileInfo entry contains the most recent attribute data and the system continues at block **2109**, else the system continues at block **2108**. In block **2108**, the system sets variable C equal to C->PrimaryPtr and loops to block **2105**. In block **2109**, the system sets variable X to variable C and continues at block **2111**.

In block **2111**, the system initializes variable Y to variable X. Variable X points to the entry with the most recent attribute data. In block **2112** through **2119**, the system locates the next entry with a status of most recent and attribute data set to FNULLS. In block **2112**, if Y->Status indicates that PrimaryPtr is valid, then the system continues to block **2113**, else a new entry is to be added and the system continues at block **2120**. In block **2113**, the system sets variable Y equal to Y->PrimaryPtr. In block **2114**, if Y->Status indicates that the SecondaryPtr is valid, then the system continues at block **2115**, else the system continues at block **2116**. In block **2115**, the system sets variable Y equal to Y->SecondaryPtr and loops to block **2114**. In block **2116**, if Y->Status is set to ATDRecent, then the system continues at block **2117**, else the system loops to block **2112**. In block **2117**, if Y->Attribute, Y->Date, and Y->Time equal FNULL, then the system continues at block **2118**, else the system loops to block **2112**. In block **2118**, the system sets Y->Attribute, Y->Date, and Y->Time to the new attribute data. In block **2119**, the system sets X->Status equal to ATDSuperseded and the routine completes.

In blocks **2120** through **2123**, the system allocates and initializes a new FileInfo entry. In block **2120**, the system allocates a new FileInfo entry and sets variable Z to point to the new entry. In block **2121**, the system sets Z->Attribute, Z->Date, and Z->Time to the new attribute data, sets Z->Status to Exists, ATDRecent, and FileInfo, sets Z->ExtentPtr to Y->ExtentPtr, and sets Z->ExtentLen to Y->ExtentLen. In block **2122**, the system sets Y->SecondaryPtr equal to variable Z and Y->Status to indicate that SecondaryPtr is valid. In block **2123**, the system sets X->Status equal to ATDSuperseded to indicate that entry no longer contains the current attribute data and the routine completes.

Although the present invention has been described in terms of preferred embodiment, it is not intended that the invention be limited to this embodiment. Modifications within the spirit of the invention will be apparent to those skilled in the art. The scope of the present invention is defined by the claims that follow.

What is claimed is:

1. A manager for a computer memory comprising:

a block allocation routine, the memory divided into blocks of memory locations, each block having an allocation table and a data region divided into data areas, each allocation table having entries corresponding to region data areas, the block allocation routine for selecting a block in which to store data;

a data area allocation routine for selecting a data area within the data region for the selected block in which to store data, for selecting an allocation table entry to correspond to the selected data area, and for setting the selected allocation table entry to correspond to the selected data area and to an allocated state; and

a storage routine for storing data in the selected data area.

2. The manager for a computer memory of claim 1 further comprising:

a data area deallocation routine for setting an allocation table entry that is in the allocated state to a deallocated state; and

a block reclamation routine for reclaiming data areas corresponding to allocation table entries that are in that deallocated state.

3. The manager for a computer memory of claim 1 further comprising:

an initialization routine, each block having header information, the initialization routine for gathering information from the headers and from the allocation tables and storing the gathered information in a memory cache.

4. A method of reclaiming deallocated space in a block-eraseable, programmable, read-only memory, the memory having blocks, the method comprising the steps of:

identifying data regions as deallocated or allocated in a block to be reclaimed;

erasing a spare block; and

copying allocated data regions from the block to be reclaimed to the spare block whereby a memory area corresponding to the deallocated data region is reclaimed for allocation.

5. The method of claim 4 wherein the allocated data regions are copied into contiguous memory locations in the spare block.

6. A method of addressing a data region in a computer memory device, the memory divided into blocks, each block having a physical block number, the method comprising the steps of:

storing an allocation table in each block, the allocation table having entries that indicate an offset of a data region within the block and that have an entry index;

storing a logical block number in each block;

identifying a data region by logical block number and allocation table entry index; and

generating an address to the identified data region based on the logical block number and the allocation table entry index.

7. The method of claim 6 wherein the step of generating an address includes the steps of:

determining the physical block number from the logical block number, each block having a corresponding start address;

retrieving the offset from the allocation table entry in the determined physical block number that is indexed by the allocation table entry index; and

adding the retrieved offset to the start address of the block with the determined physical block number to generate the address of the identified data region.

8. The method of claim 6 or 7 wherein the computer memory device is a block-erasable, programmable, read-only memory.

9. A method of leveling block erasures in a block-erasable, programmable, read-only memory, the method comprising the steps of:

identifying a first block that has been erased;

identifying a second block that has been erased a fewer number of times than the first block; and

swapping the data in the first block with the data in the second block.

10. A method of managing memory in a block-erasable, programmable, read-only memory, the memory being divided into blocks of memory locations, each block having an allocation table and a data region divided into data areas, each allocation table having entries corresponding to region data areas, the method comprising the steps of:

selecting a block in which to store data;

selecting a data area within the data region for the selected block in which to store data;

selecting an allocation table entry to correspond to the selected data area;

setting the selected allocation table entry to correspond to the selected data area and to an allocated state; and

storing data in the selected data area.

11. The method of claim 10, further comprising the steps of:

setting an allocation table entry that is in the allocated state to a deallocated state; and

reclaiming data areas corresponding to allocation table entries that are in that deallocated state.

12. The method of claim 10, each block having header information, further comprising the steps of:

gathering information from the headers and from the allocation tables; and

storing the gathered information in a memory cache.

13. A method of managing memory in a block-erasable, programmable, read-only memory, the memory being divided into blocks of memory locations, each block have a table and a data region divided into data areas, each table having entries corresponding to the data areas, the method comprising the steps of:

selecting a block in which to store data;

selecting a data area within the data region for the selected block in which to store data;

selecting a table entry to correspond to the selected data area;

setting the selected table entry to correspond to the selected data area and to indicate that the data area contains data; and

storing data in the selected data area.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,256,642 B1
DATED : July 3, 2001
INVENTOR(S) : Krueger et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 4,
Line 11, "in a in a preferred" should read -- in a preferred --.

Column 6,
Line 39, "to second" should read -- to a second --.

Column 7,
Line 62, "a block" should read -- at block --.

Column 10,
Line 2, "update" should read -- updates --.

Column 12,
Line 8, "the contains the" should read -- contains the --.

Column 13,
Line 2, "increment" should read -- increments --.
Line 35, "has a been" should read -- has been --.

Column 14,
Line 1, "queue-for-erase" should read -- queued-for-erasure --.
Line 46, "LETTERI.DOC" should read -- LETTER1.DOC --.

Column 15,
Line 26, "Each file…" should not begin a new paragraph.

Column 19,
Line 40, "In block…" should begin a new paragraph.

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

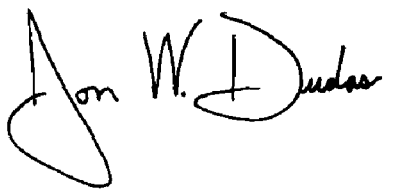
PATENT NO. : 6,256,642 B1
DATED : July 3, 2001
INVENTOR(S) : Krueger et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 25,
Line 18, "eraseable" should read -- erasable --.

Signed and Sealed this

Ninth Day of March, 2004



JON W. DUDAS
Acting Director of the United States Patent and Trademark Office