



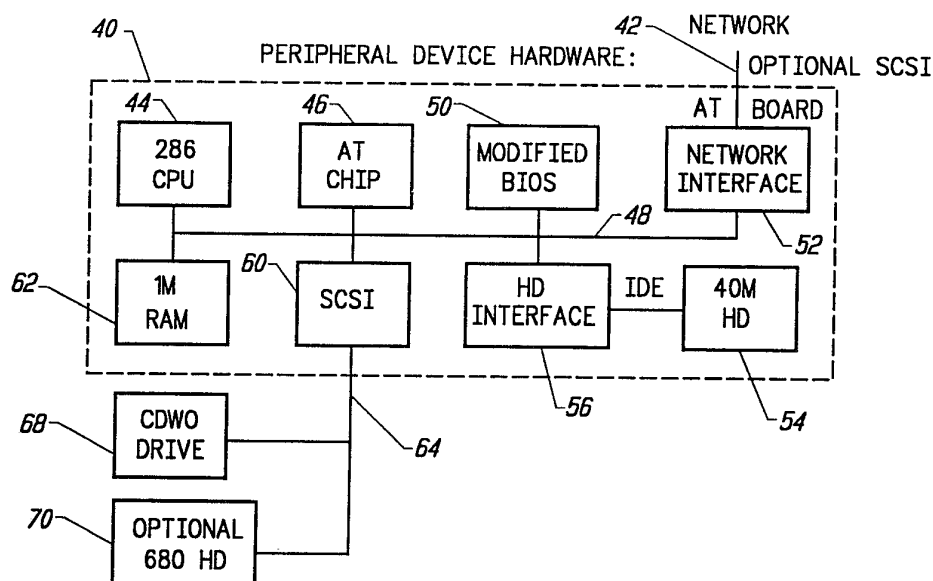
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁵ : G06F 12/00	A1	(11) International Publication Number: WO 93/09496 (43) International Publication Date: 13 May 1993 (13.05.93)
---	----	---

(21) International Application Number: PCT/US92/09274 (22) International Filing Date: 4 November 1992 (04.11.92)	(74) Agents: KENNEY, J., Ernest; Bacon & Thomas, 625 Slaters Lane, Fourth Floor, Alexandria, VA 22314 (US) et al.
(30) Priority data: 786,464 4 November 1991 (04.11.91) US	(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, SE).
(71) Applicant: MERIDIAN DATA, INC. [US/US]; 5615 Scotts Valley Drive, Scotts Valley, CA 95066 (US).	Published With international search report.
(72) Inventors: MEYER, Frederick, P. ; 350 Hidden Valley Road, Soquel, CA 95073 (US). NEAR, Scott, W. ; 116 Adrian Place, Los Gatos, CA 95032 (US). HOSHAW, David, L. ; 2119 Burr Court, Santa Cruz, CA 95062 (US). SAVILLE, Winthrop, L., III ; 4425 Esta Lane, Soquel, CA 95073 (US).	

(54) Title: FILE MANAGEMENT SYSTEM UTILIZING HARDWARE AND SOFTWARE FOR RECORDABLE, OPTICAL COMPACT DISC

HARDWARE SPECIFICATIONS:



(57) Abstract

A file management system utilizing recordable compact disc (CD) hardware and software which provides a low-cost, compact disc recording across a wide variety of computer platforms. The file management system according to the present invention includes an optical memory (CDROM), a host computer (12) including a first operating system and an orange book type of recorder (24) for handling exchange of data with the optical memory. The file management system further includes a peripheral device (20) which interfaces with both the host computer (12) and the orange book recorder (24) and provides file management for the orange book recorder (24) so as to establish direct compatibility between the first operating system and the optical memory.

FOR THE PURPOSES OF INFORMATION ONLY

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

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

File Management System Utilizing Hardware and Software for Recordable, Optical Compact Disc

Background of the Invention

The present invention relates to a file system management peripheral device which provides for the exchange of data between a standard
5 international compact disc format and many different computer operating systems.

Magnetic recording media such as hard disks and diskettes have been the most common type of
10 mass-storage devices for computers. Although magnetic tape is considered a cost-effective way to store information, it has relatively longer access times and is more useful for storing rarely retrieved information. Hard disks with flying
15 heads are faster but expensive. There is a newer technology which uses inexpensive videotape, in either VHS or 8mm format, to store a gigabyte of read/write memory on a small tape cartridge, but

its major drawback is the long access time. One development in mass storage is the "CD ROM," which uses the same optical disk technology as audio CDs (compact disks). More than 600 megabytes can be stored on one side of a 5-inch plastic disk, and the access time is much faster than any tape medium. Unlike audio CDs, there are CD ROM drives for both reading and writing by laser-burning pits in a blank CD. Fully erasable read/write magneto-optic disk memories are becoming available.

The CD ROM market is growing at a fast rate. To date, CD ROM has been primarily used for the distribution of large, stable information bases to corporations for presentation, training, internal data distribution, and document and image management, but there are many other kinds of information bases that may be equally advantageously packaged on CDs. For technical applications, for example, engineering standards and specifications for design, materials handling and construction of facilities that have traditionally been described in large manuals can be stored on CDs. Similarly, CDs can store case histories, statutes, regulations, tax history, legal forms, patents, trademarks and treatise information stored for legal and accounting professionals, demographic data used for analyzing economic patterns, determining site selection of retail businesses or making advertising decisions for market researchers, and, for the general public, publications of all sorts, such as books for search and entry of orders.

Thus, traditional CD ROM information publishers as well as more traditional book and

microfiche publishers and application software publishers are now preparing to package their products on CDs. The value of the information combined with the economics of desktop PC access
5 has been an indication of the growth of the CD ROM industry.

In view of the foregoing, it would be highly desirable to provide a low-cost, easy-to-use
10 peripheral device for making such CDs and their recorders directly compatible to commonly used operating systems of host computers having different architectures.

15 Summary of the Invention

It is an object of the present invention to bring the CD recording technology into the computing main stream.

20

It is another object of the present invention to provide a peripheral device for translating a standard international CD ROM protocol into a transparent generic protocol that can be accessed
25 by many different computer operating systems.

In one preferred embodiment, the present invention provides a file management system including a read/write random access device having
30 removable read/write media and a host computer including a first operating system. The present invention further includes an intelligent peripheral device including means for interfacing with the host computer, means for interfacing with
35 the random access device, and means for providing

dedicated file management for the random access device. The intelligent peripheral device further includes means for providing direct and immediate access to the random access device and means for providing exchange of data between the host computer and the random access device, including means for controlling the transfer of file data to the random access device which is in conformance with international standards.

10

Other objects, features and advantages of the present invention will become apparent from the following detailed description when taken in conjunction with the accompanying drawings.

15

Brief Description of the Drawings

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

20

Fig. 1 is a block drawing showing the manner in which a peripheral device according to the present invention is connected to a host computer and other devices to be served thereby;

25

Fig. 2 is a block diagram of a file system management peripheral device embodying the invention;

30

Fig. 3 shows the functional relationship among software modules contained in a program to be loaded on the host computer of Fig. 1;

35

Fig. 4 shows the functional relationship between the peripheral device software contained in a program to be loaded on the host computer of Fig. 1;

5

Fig. 5 shows the functional relationship among software modules contained in a program loaded on the peripheral device of the present invention of Fig. 2;

10

Fig. 6 is a block diagram of a standard peripheral device embodying the present invention;

Fig. 7 is a block diagram of an integrated peripheral device embodying the present invention;

15

Fig. 8 is a block diagram of an expanded peripheral device embodying the present invention;

Fig. 9 is a block diagram of a juke box peripheral device embodying the present invention.

20

Detailed Description of the Invention

Reference will now be made in detail to the preferred embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it will be understood that they are not intended to limit the invention to those embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims.

35

General Overview:

5 In one preferred embodiment, as shown in FIG.
1, the present invention provides a SCSI/network
peripheral device 20 that provides file system
management for Orange Book recorders 24 which
exchange data in Frankfurt format (CDWO superset
of ISO-9660).

10

The "Orange Book" specification is a
published standard together with published
standards such as ISO-9660 specification; Red Book
specification; Yellow Book specification; and
15 Green Book specification.

These standards are identified as follows:

ISO 10149 Data Interchange on Read-Only
120mm Optical Disks (CD-ROM), also known
as the "Yellow Book".

20

IEC 809 Compact Disc Digital Audio, also
known as the Red Book.

25

CD-I = Compact Disc Digital Interactive,
specified in the CD-I Full Function
Specification ("Green Book"), N.V.
Philips and Sony Corporation.

30

Recordable Compact Disc Systems ("Orange
Book"), N.V. Philips and Sony
Corporation.

The "Frankfurt" Group Proposal is a known
proposal for a Read-Only and Write-Once Compact
35 Disc volume and file structure standard based on

the work and discussions of the Frankfurt Group Technical Committee. The Frankfurt Group is an ad hoc group of companies which share common interests and goals concerning Read-Only CD-ROM and Write-Once (CD-WO) Compact Disc technology.

The description of the present invention will make reference to the Orange Book specification and recorder and also the Frankfurt proposal, the details of which are hereby incorporated by reference.

Referring again to Fig. 1, the file management system 10 allows a peripheral device (PD) 20 to be directly compatible with the operating system of a host computer 12 in the same way that floppy drives and hard drives appear. This removes the requirements for special application software to deal with Orange Book recorders 24 as foreign devices.

The peripheral device 20 attaches directly to a network line 14 or can be controlled from a local SCSI bus 16. The peripheral device 20 has a small internal low-cost hard drive which is used for high performance cacheing. This also removes all apparent real-time requirement of the recorder 24. The peripheral device 20 also has an external SCSI bus 22 that connects to the Orange Book recorder 24 and an optional large hard drive 26 for additional cacheing.

The peripheral device 20 of FIG. 1 can attach to a read/write random access device having removable read/write media 24 such as Orange Book recorders 24 (such as JVC and Philips) that can

read, write and append data to CDROM recordable media. A track-at-once recording technique is used to create CDs that are compatible with all CDROM drives when used with Frankfurt read-only
5 drivers. Block append recording techniques are used to make CDs that are only compatible with Orange Book drives. Block append can be used for smaller incremental recording than track-at-once and eliminates any significant pauses when
10 attempting to read when the cache is recording to disc. The primary use of Block append will be to append small files.

Several important values of the peripheral
15 device 20 are:

1. High-performance read/write access to Orange Book recorders not possible in software-only solutions.
- 20 2. Standard peripheral device Protocol and API for universal access to Orange Book recorders from different host computers.
3. Allows Orange Book recorders to appear as native devices.
- 25 4. Direct network access without burden to a File Server.

In one embodiment, the peripheral device
20 of FIG. 1 is a 5.25" form-factor full height unit which may be installed in any cabinet or
30 chassis that supports standard IBM-PC style drive bays.

As seen in FIG. 2, the peripheral device hardware 40 is fully enclosed and includes:

1. 286 based CPU 44 with BIOS 50 (Ampro bus format).
2. AT computer chip 46 for full PC compatibility.
- 5 3. 1M RAM 62.
4. SCSI interface chip 60 (MB89352).
5. Ethernet interface 52 (standard short IBM-PC Board).
6. IDE hard-drive interface 56.
- 10 7. 40M IDE 3.5" hard drive 54 (optional if HD on SCSI bus).
8. Extra ROM socket and extra RAM sockets 62 for up to 4M.
9. Software controlled SCSI terminators.

15

The front panel (not shown) includes:

1. 3-color square LED (lower left panel).
2. Optional 16 character, 2 line LCD
- 20 (center).

The rear Panel (not shown) includes:

1. 8 dip switch block
 - 0-2: SCSI/peripheral device ID.
 - 25 3 : SCSI Termination on/off.
 - 6 : CDWO Eject override.
 - 7 : Self Diagnostics.
2. Standard PC power jack (4 pin).
- 30 3. SCSI 50-pin DIN port (CDWO and HD SCSI bus).
4. Single AT slot gap (Network card or optional SCSI card).

35 Software Specification:

HOST PERIPHERAL DEVICE SOFTWARE (FIGURE 3)
Host peripheral device TSR:

5 The Host peripheral device TSR (terminate and stay resident program) is loaded on the host computer prior to use of the peripheral device and contains the following software modules, as shown in FIG. 3:

- 10 1. Peripheral device Redirector 100 (DOS or other operating system).
2. Peripheral device API 94.
3. Peripheral device Protocol Interface 82.
4. Network Interface 80.
- 15 5. Optional SCSI interface 84.

Peripheral device Protocol Interface 82:

20 Protocol Interface 82 defines a standard protocol for communicating to the peripheral device File System from different operating systems. A specific operating system such as DOS must map between its native file system protocol and the peripheral device Protocol via a
25 redirector. The peripheral device Protocol also accepts file system commands and other commands through the peripheral device API 94 linked with application programs 96. The peripheral device Protocol 82 and peripheral device API 94 are
30 defined for a wide range of operating systems, but must be specially implemented on each unique operating system.

The peripheral device Protocol 82 is called
35 by interrupts with specific registers set (in the

11

case of DOS) and may be called from the following software modules of FIG. 3:

1. Peripheral device Redirector 100 via the peripheral device API 94.
2. Application programs 96 via the peripheral device API 94.
3. MSCDEX 112 through the peripheral device BDD 110 via the peripheral device API 94.

The peripheral device Protocol Interface 82 of FIG. 3 converts these calls to the standard peripheral device Protocol structures to be transferred across the network line 81 or the optional SCSI bus 86 by the Network Interface 80 or the SCSI Interface software 84. This protocol is identical for all operating systems. The peripheral device Protocol includes:

20

Peripheral Device Protocol

The peripheral device protocol defines the command set used by a host computer to talk to the peripheral device device over a transport medium (either network or SCSI). There are 3 general types of commands that will be implemented: 1) user, 2) administrative, 3) file and 4) misc. The user commands include functions such as: login, logout, etc. The administrative commands include format, eject and diagnostics. The file commands include: open file, close file, read file and write file. Each file command will contain a subcommand field which will define which variation is being used (base, UNIX, POSIX, etc.).

35

Command List		
<i>User Commands</i>		
5	ChangePermissions	Reserve
	Flush	ReserveCancel
	Login	WhoIs
	Logout	
<i>Administrative Commands</i>		
10	AddUser	LogoutAll
	ChangePassword	Reboot
	ConfigureSystem	Rebuild
	DeleteUser	Shutdown
	Diagnostics	SystemStats
	Download	UserList
15	Eject	WriteTOC
	Format	
<i>File Commands</i>		
20	ChangeAttributes	ReadBlock
	CloseFile	ReadFile
	CreateDir	ReadPVD
	CreateFile	RenameDir
25	DeleteDir	RenameFile
	DeleteFile	SearchFirst
	FileLock	SearchNext
	FileUnlock	ValidatePath
30	GetVolumeLabel	WriteFile
	OpenFile	WriteImage
<i>Misc Commands</i>		
35	DiskInfo	Status

Implementation*Packet Header*

```

5      struct fsHeader {
          uint      command
          uint      modifier base, unix, posix,
                      etc.
          uint      peripheral device which
10         peripheral device
          uint      drive      which WO drive
          ulong     userId     userId of issuer
          ulong     processId Id for multiuser OS
          uint      status      status returned
15         by peripheral
                                device
          uint      returnLen   length of data
                                being returned
          ulong     handle      file handle
20         uint      dataLen    length of
                                additional data
          char far  *data       address of
                                additional data
                                (filled in by
25         peripheral
                                device net
                                driver)
        }

```

30

User Commands

ChangePermissions - allow user to change
access rights

35

14

```
input:
    fsHeader
    uint      permissions
```

```
5      output:
        uint      status
```

Flush - Writes the contents of the FS buffers
to be written to the WO

```
10     input:
        fsHeader
```

```
        output:
            uint      status
```

15

Login - Allows user to gain access to the
peripheral device

```
        input:
20          fsHeader
            uchar[33] userName
            uchar[33] password
            uint      permissions    access
                                     rights
```

25

```
        output:
            ulong      userId
            uint      status
```

30

Logout - User is terminating access to the
peripheral device

```
        input:
            fsHeader
```

35

15

output:
uint status

Reserve - indicate that user wishes exclusive
5 access to peripheral device

input:
fsHeader
uint permissions

10 output:
uint status

ReserveCancel - cancel a reserve request

15 input:
fsHeader

output:
uint status

20

WhoIs - who is logged in

input:
fsHeader
25 uint control

output:
uint status
uint control
30 uchar[33] userName
ulong userId
uint permissions
uint drive

35

Administrative Commands

AddUser - allow administrator to add a new

user

```
5      input:
        fsHeader
        uchar[33] userName
        uchar[33] password
        uint      securityLevel

10      output:
        uint      status
```

15 ChangeUser - allow administrator to change a
user's data

```
      input:
        fsHeader
        uchar[33] userName
        uchar[33] password
20      uint      securityLevel

      output:
        uint      status
```

25

ConfigureSystem - allow administrator to
tailor system operation

```
      input:
30      fsHeader
        uint      requirePasswords
        uchar[33] serverName
```

35 DeleteUser - delete a user

17

input:

fsHeader

uchar[33] userName

5 output:

uint status

10 Diagnostics - tell peripheral device to
perform diagnostic tests

input:

fsHeader

output:

15 uint status

20 Download - Download a system file to the
peripheral device

input:

fsHeader

uint appendFlag

uchar[1024] fileName

25 uint attributes

uint length

uchar far *data

output:

30 uint status

Eject - Causes an ETD to be written to the
disk (implied Flush)

35 input :

18

fsHeader

output:

uint status

5

Format - Format track structure of blank
media

input:

10

fsHeader

uint mode

uint nTracks number of
tracks to
put on
disk

15

ulong[nTracks] trackSize size of each track

output:

uint status

20

LogoutAll - logout all users except
administrator

input:

25

fsHeader

output:

uint status

30

Reboot - tell peripheral device to reboot
(need to reload DOS)

input:

fsHeader

35

19

```
output:
    uint    status
```

```
5      Rebuild - peripheral device to rebuild file
      system (FS corrupted)
```

```
input:
    fsHeader
```

```
10     output:
        uint    status
```

```
15     Shutdown - tell the peripheral device to shut
      down
```

```
input:
    fsHeader
```

```
20     output:
        uint    status
```

```
      SystemStats - get system statistics: returns
      list of users logged in ....
```

```
25     input:
        fsHeader
      output:
        uint    status
        uint    userCnt    number of
30                                users
```

```
      struct {
        ulong    processId
        uint     permissions
        uint     drive
35     uchar[33] userName
```

20

```
} [userCnt]
```

User List - Provide list of users

input:

```
5          fsHeader
          uint      control
```

output:

```

10          uint      status
          uint      control
          uchar[33] userName
          uchar[33] password
          uint      securityLevel

```

15

WriteTOC - write the Table of Contents to the
WO disk

input:

fsHeader

20

output:

```
uint      status
```

File Commands

25 ChangeAttributes - change the attributes of a
 file

input:

```

    fsHeader
    uchar[1024]    fileName
30          uint    attributes
              uint    newAttributes
              uint    subDataLen    length
              uchar[]    subData    data
                                   specific
35                                   to subCommand

```

21

```

output:
    uint    status

```

```

5      CloseFile - close an open file
      input:
          fsHeader
          ulong    handle
          uint     subDataLen    length
10         uchar[] subData      data
                                     specific
      to                               subCommand

```

```

      output:
15         uint     status

```

```

CreateDir - create a new directory

```

```

input:
    fsHeader
20    uchar[1024]  dirName
    uint          attributes
    uint          subDataLen    length
    uchar[]       subData      data
                                     specific
25                                     to subCommand

```

```

output:
    uint    status

```

```

30      CreateFile - create a file
      input:

```

```

    fsHeader

```

```

35

```

22

```

uchar[1024]    fileName
uint          attributes
uint          mode          file access mode
                               (read/write)
5      uint      subDataLen    length
uchar[]      subData          data
                               specific to
                               subCommand

10      output:
          uint          status
          ulong         handle

15      DeleteDir - delete a directory
input:
      fsHeader
uchar[1024]    fileName
uint          attributes
20      uint      subDataLen    length
uchar[]      subData          data
                               specific
                               to subCommand

25      output:
          uint          status

```

DeleteFile - delete a file
input:

```

30      fsHeader
uint          attributes
uchar[1024]    fileName

```

35

23

	uint	subDataLen	
length	uchar[]	subData	data
			specific
5			to
			subCommand

	output:	
	uint	status
10		

FileLock - protect a range of bytes in a file

input:

	fsHeader	
15	int	handle
	uint	lockType
	ulong	startByte
	ulong	endByte
	uint	subDataLen
20	uchar[]	subData
		length
		data
		specific
		to subCommand

	output:	
	uint	status
25		

FileUnlock - remove a file lock

input:

30	fsHeader	
	ulong	handle
	uint	lockType
	ulong	startByte
	ulong	endByte
35	uint	subDataLen
		length

24

uchar[] subData data
 specific
 to subCommand

5 output:
 uint status

10 disk GetVolumeLabel - get the volume label of a

 input:
 fsHeader

15 output:
 uint status
 ulong searchHandle handle to
 be used
 for search next

20 uchar[256] fileName
 ulong fileLen
 uint attributes
 uchar[7] dateTime
 uint version
 uint subDataLen length
 uchar[] subData data
 specific
 to subCommand

30 OpenFile - open a file
 input:

 fsHeader
 uchar[1024] fileName
 uint attributes

```

uint      subDataLen
length
uchar[]   subData      data
                        specific
                        to subCommand

```

```
uint      status
ulong     handle
```

ReadBlock - read sector(s) from disk (only valid for ISO disks or finished FP disks)

15	fsHeader		
	ulong	sector	starting
			sector number
	uint	numSectors	number of
			sectors to read
20	uint	subDataLen	length
	uchar[]	subData	data
			specific
			to subCommand

```
uint      status
```

input:

```
fsHeader
ulong      handle
ulong      pos          position in
                        file to read from
uint       length       amount to read
```

26

```
uint      subDataLen  length
uchar[]   subData    data specific
                        to subCommand
```

```
5      output:
        uint      status
        uint      length
```

ReadPVD - read Primary Volume Descriptor

```
10     input:
        fsHeader
```

```
        output:
        uint      status
15      ulong      sector
        uchar[2048] PVD data
```

RenameDir - rename a directory

```
20     input:
        fsHeader
        uchar[1024] oldName
        uint      oldAttributes
        uchar[256]  newName
25      uint      newAttributes
        uint      subDataLen
        uchar[]   subData
```

```
        output:
30      uint      status
```

RenameFile - rename a file

```
35     input:
```

27

```

        fsHeader
        uchar[1024]    oldName
        uint           oldAttributes
        uchar[256]     newName
5         uint         newAttributes
        uint           subDataLen
        uchar[]        subData

10        output:
            uint        status

        SearchFirst - find matching file name
        input:
15        fsHeader
        uchar[1024]    specification file name
                        template including full
                        path
        uint           attributes
20        uint         subDataLen    length
        uchar[]        subData       data
                                    specific to
                                    subCommand

25        output:
            uint        status
            ulong       searchHandle  handle to
                                    be used
            for         search
30        next

        uchar[256]     fileName
        ulong          fileLen
        uint           attributes
35        uchar[7]     dateTime

```

28

```

uint      version
uint      subDataLen  length
uchar[]   subData     data
                                specific
5                                to subCommand

```

SearchNext - find next matching file

input:

```

10      fsHeader
        ulong      searchHandle
        uint       subDataLen  length
        uchar[]    subData     data
                                specific
15                                to subCommand

```

output:

```

        uint       status
        ulong      searchHandle  handle to
20                                be used for
                                search next
        uchar[256]  fileName
        ulong      fileLen
        uint       attributes
        uchar[7]    dateTime
25        uint      version
        uint      subDataLen
        length
        uchar[]    subData     data
30                                specific
                                to subCommand

```

ValidatePath - validate that the given path

35 exists

29

input:

fsHeader

uchar[1024] specification full

path name

5 uint attributes

uint subDataLen length

uchar[] subData data

specific

to subCommand

10

output:

uint status

WriteFile - write data to file

15 input:

fsHeader

ulong handle

ulong pos starting

offset

20 uint size amount of
data to
write

uchar[size] data

uint subDataLen length

25 uchar[] subData data

specific

to

subCommand

30

output:

uint status

WriteImage - write single track ISO image to

35 WO disk

30

input:

fsHeader

ulong

pos

starting offset

ulong

size

size of

data

5

uchar[size]

data

output:

uint

status

10

Misc Commands

DiskInfo - return information about the disk

15 in the WO drive

input:

fsHeader

output:

20

uint

status

uint

type

disk type:

ISO/FP,

single/multi

track, etc.

25

TOCStruct TOC

table of

contents

Status - get current status of peripheral

30 device

input:

fsHeader

output:

35

DOS Redirector 100:

The peripheral device Redirector 100 of FIG. 3 intercepts commands from DOS and allows the peripheral device Orange Book recorder to appear as a local hard disks drive. The peripheral device Redirector 100 contains two parts: a general purpose DOS redirector and a DOS to peripheral device converter. The peripheral device redirector 100 supports all standard high level DOS commands such as COPY, DEL and DIR as well as low level access such as open, read, write and close. These commands are converted to the peripheral device Protocol for eventual transmission to the network wire.

Network Interface 80:

The host network interface software 80 of FIG. 3 uses a CD Net peer-to-peer communication protocol along with SPX/IPX as a transfer vehicle. Commands from the peripheral device Protocol 82 will be packetized and sent through ethernet on a Novell network to the peripheral device unit. The peripheral device 40M internal hard drive or optional external drive is not directly viewable by the host system.

SCSI Interface 84:

The SCSI Interface software 84 of FIG. 1 sends data across the SCSI bus 86 in the same format as the Network Interface 80. Data structures that conform to the peripheral device Protocol are sent within a SCSI command block

instead of SPX/IPX as with the Network Interface
80.

5 The SCSI interface 84 is used only to connect
to the peripheral device unit and is not used to
control the Orange Book recorder or optional hard
drive directly.

Peripheral device BDD 110:

10 The peripheral device Block Device Driver
(BDD) 110 of FIG. 3 is used to connect Microsoft
MSCDEX 112 with the peripheral device API 94.
This allows block level access from MSCDEX 112 to
be sent to the peripheral device. This does not
15 give the current implementation of MSCDEX access
to Frankfurt type discs, but does allow ISO-9660
and MSCDEX compatible titles to be read from the
Orange Book drive in the peripheral device.

20 Application Programs 96:

Application programs 96 of FIG. 3 may
communicate with the peripheral device Protocol by
linking to the peripheral device API 94 (unique
for each operating system). Application programs
25 usually use only the System Commands of the
peripheral device API 94 since file access (DOS)
will be intercepted by the redirector 100. File
Commands may be directly issued through the
peripheral device API 94 if desired.

30

Publishing application programs may access
the Orange Book recorder by using Publishing
Commands in the peripheral device API 94. This
allows large single track disks to be published
35 that are compatible with ISO-9660 and MSCDEX 112.

A 680M optional hard drive is utilized on the peripheral device to publish very large tracks. File system access would be limited during publishing of titles.

5

A special LOGIN application program sends special peripheral device functions beyond the supported DOS calls. This program is explained in greater detail below. This LOGIN program

10 includes:

15	LOGIN/LOGOUT	:	Login and logout for permission to read/write to the peripheral device.
20	FORMAT	:	Format the peripheral device media to specific type or finalize track 1 of a disc.
	FLUSH	:	Flush the HD to the CDWO but do not eject.
25	EJECT	:	Flush and eject the media from the CDWO.

The ADMIN program is a separate utility for the network administrator:

30	ADMIN	:	Allow code updates and permission override.
----	-------	---	---

Peripheral device SOFTWARE (FIGURE 4):
System Kernel:

The Kernel software of FIG. 4 controls all system functions such as bootup, multi-tasking, memory management, status, cacheing and utilities. The cacheing software also controls the 40M
5 internal hard drive 54 and optional 680M SCSI hard drive 70 of FIG. 2. This provides the highest possible performance and versatility.

The peripheral device software is
10 automatically downloaded to RAM 62 from the 40M hard disk 54 (reserved area) of FIG. 2 whenever the peripheral device is powered up or reset. Other software and upgrades can be loaded by inserting a special CD into the Orange Book
15 recorder attached to the peripheral device.

Network Interface 132:

The peripheral device contains Network Interface software 132 of FIG. 4 to receive
20 commands from across the network 130. This software is nearly identical to the software described in the Host Software section (reciprocal functionality). Command data structures are extracted and sent to the peripheral device
25 Protocol Translator.

SCSI Interface 150:

The peripheral device contains SCSI interface software 150 to receive commands from across the
30 SCSI bus 153. This software is nearly identical to the software described in the Host Software section (reciprocal functionality). These commands are in the same format as the network Interface. Command data structures are extracted

and sent to the peripheral device Protocol Translator.

5 This SCSI interface 150 is used only to connect to the host computer SCSI bus 153 and is not used to control the Orange Book recorder or optional hard drive directly.

Peripheral device Protocol Translator 136:
10 Translator 136 receives peripheral device Protocol structures from the Network Interface 132 or SCSI Interface 150 and decodes the commands to be sent to the peripheral device File System 142. This software 136 also encodes data to be sent
15 back to the host through either interface 132, 150.

20 Peripheral device File System 142:
The peripheral device File System 142 is a high performance file system that uses a similar file structure to Frankfurt 146. It is optimized for performance and manages data located on the
25 internal cache hard drive 166 and the external appendable CD. The hard drive cache is divided into four areas:
1. 10M for path table information
2. 7M for Orange Book track-at-once
30 buffer
3. 10-650M for general peripheral device File System cache
4. 10M for peripheral device program
space peripheral device File System 142 structure:
35 Not currently defined.

Frankfurt File Structure 146:

The peripheral device File System 142
converts data to the Frankfurt File Structure 146
before writing to disc. It also reads data from
5 any Frankfurt type disc. Path table information
is not written to the disc until the disc is
removed or a Flush is requested.

CDWO Interface 148:

10 CDWO Interface 148 of FIG. 4 controls the
Orange Book recorder and handles all aspects of
the Orange Book specification. This includes all
reading and writing control Track-at-once and
block append is also supported.

15

Hard Drive Interface 162:

Interface 162 is used to control data on the
internal or external hard drive. The drives are
generally accessed through DOS but are also
20 accessed at the block level using interrupt 25H.
The internal drive uses an IDE hardware interface
162 and the external drive uses a SCSI interface
170.

25

SCSI Device Interface 170:

The SCSI Device Interface 170 of FIG. 4
controls the SCSI chip and can handle all SCSI
commands required to manage the Orange Book
recorder and the optional hard drive.

30

HOST CDROM SOFTWARE (FIGURE 5)

Peripheral device Extensions:

The peripheral device Extensions allows
Frankfurt discs in CDROM drives to appear as a

local hard disk drive. The peripheral device Extensions consists of:

1. CDRom Redirector 200
2. Frankfurt Read 204
- 5 3. CDRom BDD 208

This software can read any disc created on the peripheral device or any other Frankfurt compatible system. Frankfurt CDRom discs can be read between append operations on any existing
10 CDRom using the peripheral device extensions. Microsoft MSCDEX 210 may still connect to the CDRom Block Device Driver 208 to access ISO-9660 or MSCDEX compatible CDs.

15 CDRom Redirector 200:

The CDRom Redirector 200 of FIG. 5 intercepts commands from DOS and allows CDRom drives to appear as a local hard disk drive. The CDRom Redirector 200 contains two parts: a general
20 purpose DOS redirector and a DOS to peripheral device protocol converter. The CDRom redirector 200 supports many standard high level DOS commands such as COPY, DEL and DIR as well as low level access such as open, read and close. This
25 redirector 200 supports only read commands because CDRom is a read only device. Redirected commands are sent to the Frankfurt Read software 204.

30 Frankfurt Read 204:

This software 204 of FIG. 5 reads Frankfurt format discs and bridges between the CDRom redirector 200 and the CDRom Block Device Driver 208.

This software 204 supports the read requirements of the Frankfurt File Structure and includes:

- 5 - VTOC sector 16 management
- Multiple extents
- Additional legal file name characters
- ETD end of track descriptor

10

CDROM BDD 208:

This is a standard CDROM Block Device Driver 208 used with Microsoft extensions. The Frankfurt Read software 204 accesses the CDROM through
15 Driver 208. MSCDEX 210 may also access this driver to read ISO-9660 or MSCDEX compatible CDs.

20 Peripheral Data File System

A prime design goal of the peripheral device file system, or FS, is to allow high speed access to a CDWO drive. To achieve this goal will be obtained by implementing the FS using database technology.
25 Path table and directory structures will be maintained by a database system to permit the fastest possible access to data contained in the file system.

30 To optimize the usage of the space on the CDWO disk, a track buffering scheme will be used by the FS; data will not be written to the CDWO until the track buffer is filled. This buffering algorithm has two main advantages: 1) it allows for file
35 extent optimizations that would otherwise be

impossible, 2) it minimizes the wastage of space on the media. This imposes an additional burden on the FS: not only does FS have to track multiple file extension, but it must track these extension
5 over different media (i.e. internal hard drive or CDWO).

When a disk is ejected or "fixed", a Frankfort Proposal conversion module will be executed to
10 convert the FS data structures to FP/ISO path table, directories etc. on the disk. When a new disk is inserted or the peripheral device boots, the existing FP/ISO (if any) is read and converted into FS format.

15

Implementation

The FS maintains 9 databases: 1) path 2) file, 3) extent, 4) xaa (eXtended Attribute Area) 5) file lock, 6) users, 7) file handles, 8) search
20 handles and 9) free disk buffers. In addition FS has a data file that contains system data. The path database replaces the path table of the disk; updates to the path structure are more efficient due to the use of indexed access to path records
25 and to the fact that data is not moved during updates as is true for real FP path tables. The file database is used to access the data contained in individual directory entries. The file extent information and xaa data are kept in databases to
30 eliminate the need for variable length file records and to facilitate access. The file lock database contains information about the file locks that are currently in effect. The user database tracks logins and logouts. File handles will be

maintained that associate each file to the user who opened that file.

Database Structures

```

5  Path Database
    ulong          parentId  id of parent
    directory
    uchar[256]     name      name of directory
    uint           level     level in directory
10 structure ( 1 = root )
    ulong          id        id of this
                                directory
    ulong          revision  number of times
                                modified
15 uchar[7]        dateTime  date and time of
                                creation
    uint           flags     file flag
    uint           openCnt   number of users
                                using this directory
20 ulong          fileCnt   number of files in
                                this directory
    uint          xaaLen    length of the xaa
                                data (if any)

25 index 1 = parentId  index 3 = level
    index 2 = name      index 4 = id

30 File Database
    ulong          pathId    id of the path
    entry this file belongs to
    uchar[256]     fileName  name of this file
    ulong          fileId    id of this entry
  
```

41

	uint	recordLen	length of this entry
	ulong	length	file length
	uchar[7]	dateTime	date & time of creation
5	uint	flags	file flags
	uint	unitSize	see 9.1.6
	uint	gapSize	see 9.1.7
	uint	volume	volume which file exists on
10	uint	version	file version number
	ulong	revision	file revision number
	uint	xaaLen	length of xaa data
15	uint	openCnt	number of users that have this file open
	ulong	extCnt	number of extents
	uchar[33]	password	password for file access
20	uint	fileLocks	number of file locks active for this file
25	index 1 = pathId + fileName index 3 = fileId index 2 = fileId		

30 *Extent Database*

	ulong	extId	id of this extent
	ulong	fileId	record number of file that owns this extent
35	uint	mediaType where the extent resides (HD or WO)	

	uint	number	denotes drive if
	mediaType = HD or buffer #		
	ulong	location	location on disk or in buffer
5	ulong	length	length of extent
	ulong	free	amount unused in extent
	index 1 = fileId		index 2 = extId
10			
	<i>XAA Database</i>		
	ulong	xaaId	id of this record
	uint	ownerType	dir/file
15	ulong	ownerId	id of the owner of this record
	uint	length	length of this record
	uchar[64]	data	data
20			
	index 1 = xaaId		index 2 = ownerId
	<i>File Lock Database</i>		
25	ulong	fileId	id of file this lock applies to
	uint	lockType	type of lock
	ulong	userId	user who set this lock
30	ulong	startLock	starting byte position of lock
	ulong	endLock	ending byte position of lock
	index 1 = fileId		index 2 = userId
35			
	<i>User Database</i>		
	ulong	userId	unique userId generated by FS

43

uchar[33]	userName
uchar[33]	password
uint	securityLevel

5 index 1 = userName

Login Database

	ulong	userId	
10	uchar[7]	loginTime when the user logged in	
	ulong	processId	process id of user
	uint	permissions	(exclusive read/write,
15			administrator, etc.)
	uint	drive	CDWO drive
20		index 1 = userId	

File Handle Database

	ulong	handle	unique number generated
25	by FS		
	ulong	userId	id of user who opened
	ulong	fileId	fileId of the open file
	uint	mode	open mode & sharing attributes
30			
		index 1 = handle	index 3 = fileId
		index 2 = userId	

35 *Search Handles*

ulong	userId
ulong	pathId
ulong	fileId

index 1 = userId

5

Free Disk Buffers

uint bufferNo
 10 ulong location
 ulong length

index 1 = location index 2 = length

15

System Data

```

      typedef struct driveInfo {
                ulong       diskId     id of disk
20             ulong       pathSize   projected size of
                                   the path table
                ulong       dirSize    projected size of
                                   the directory
                                   structure
25             uint        last Track  last track that
                                   was written
                TOCStruct TOC       table of contents
                                   for disk
                uint       TOCWritten  TRUE if TOC has
30                                    been written to
                                   disk
                bufferInfo  buffers description of
                                   track buffers
                } driveInfo[nDrives]

```

35

```

typedef struct TOCStruct {

```

45

```

uint      nTracks  number of tracks
                        defined for this
                        disk
ulong[99] trackSize track size of each
5          track
uint[99]  trackUsed flagged if track is
                        already
                        used
    } TOCStruct
10

typedef struct bufferInfo {
uint      nBuffers  number of buffers
                        assigned to this
15          drive
bufferStruct **buffers  pointer to array of
                        bufferStructs
    } bufferInfo

20
typedef struct bufferStruct {
        uint      drive      DOS drive this
                                buffer is on
        ulong      sector    starting sector of
25          the buffer
        ulong      length    length in sectors
    } bufferStruct

```

30 Peripheral Data File System Command Set Implementation

ChangePermissions

- 1) check for valid userId, if not return "invalid user"
- 35 2) check type of disk; if permissions are not compatible, return "wrong type media"
- 3) update user record
- 4) return "ok"

Flush

- 1) check for valid userId, if not return "invalid user"
- 2) return "ok"

5

Login

- 1) do a database lookup on userName, if a match is not found return "invalid user"
- 2) if user is already logged in, return "already logged in"
- 3) if the password field in the database is not null, compare to password in command. If they do not match, return "invalid password".
- 4) check requested permissions against current disk type; if not compatible, return "wrong disk type".
- 5) if exclusive access is requested, and another user is logged in, return "cannot grant exclusive".
- 6) if requested drive is not available, return "invalid drive"
- 7) record current login info in user record.
- 8) return "ok" with userId

25 **Logout**

- 1) check for valid userId, if not return "invalid user"
- 2) search user database for match on userId if no match found, return "not logged in"
- 3) close all file handles owned by the user.
- 4) release all active file locks owned by the user.
- 5) cancel any reserves by this user
- 6) update user record.
- 7) return "ok"
- 8) if no users logged in, check for outstanding reserve requests.
- 9) if a reserve is outstanding, notify user and lock out other logins.

10) if user does not respond to notification,
cancel reserve and unlock logins.

Reserve

- 5 1) check for valid userId, if not return
"invalid user"
- 2) if other users are logged in, return "ok" and
store reserve request.
- 3) compare disk type with requested permissions;
10 if compatible, notify user and lock out other
logins.
- 4) if user fails to respond, cancel reserve and
enable logins.

15 ReserveCancel

- 1) check for valid userId, if not return
"invalid user"
- 2) if reserve is still queued, remove if from
the queue and return "ok"
- 20 3) if reserve is in the process of activation,
cancel reserve and enable logins.
- 4) return "ok"

ChangeAttributes

- 25 1) check for valid userId, if not return
"invalid user"
- 2) check for valid drive, if not return "invalid
drive"
- 3) find match for drive + fileName + attributes;
30 if not found return "file not found"
- 4) modify attributes
- 5) return "ok"

CloseFile

- 35 1) check for valid userId, if not return
"invalid user"
- 2) check for valid file handle, if not return
"invalid handle"

- 3) release any outstanding file locks for this file
- 4) decrement open count in path entry
- 5) decrement open count in file entry
- 5 6) delete handle
- 7) return "ok"

CreateDir

- 1) check for valid userId, if not return "invalid user"
- 10 2) check for valid drive, if not return "invalid drive"
- 3) validate path; if invalid return "path not found"
- 15 4) create new entry in path database
- 5) create new entry in file database
- 6) return "ok"

CreateFile

- 20 1) check for valid userId, if not return "invalid user"
- 2) check for valid drive, if not return "invalid drive"
- 3) validate path; if invalid return "path not found"
- 25 4) create new entry in file database
- 5) increment open count in path entry
- 6) increment open count in file entry
- 7) create new handle
- 30 8) return "ok"

DeleteDir

- 1) check for valid userId, if not return "invalid user"
- 35 2) check for valid drive, if not return "invalid drive"
- 3) validate path; if invalid return "path not found"

4) if fileCnt or openCnt in path entry not 0,
return "path not empty"

4) delete entry in path database

5) delete entry in file database

5 6) return "ok"

FileLoc

1) check for valid userId, if not return
"invalid user"

10 2) check for valid file handle, if not return
"invalid handle"

3) check file lock database for overlapping
locks

4) write new locks

15 5) return "ok"

FileUnlock

1) check for valid userId, if not return
"invalid user"

20 2) check for valid file handle, if not return
"invalid handle"

3) check file lock database for overlapping
locks

4) write modified locks

25 5) return "ok"

OpenFile

1) check for valid userId, if not return
"invalid user"

30 2) check for valid drive, if not return "invalid
drive"

3) validate path; if invalid return "path not
found"

4) find entry in file database; if not found
35 return "file not found"

5) increment open count in path entry

6) increment open count in file entry

7) return "ok" and handle

ReadBlock

- 1) check for valid userId, if not return "invalid user"
- 2) check for valid drive, if not return "invalid drive"
- 3) is disk compatible with block I/O; if not return "block I/O not supported"
- 4) verify sector range is valid; if not return "sector not found"
- 5) return "ok" and data

ReadFile

- 1) check for valid userId, if not return "invalid user"
- 2) check for valid file handle, if not return "invalid handle"
- 3) verify byte range is valid; if not return "read past eof"
- 4) check for file locks; if bytes are locked return "sharing violation"
- 5) return "ok" and data

SearchFirst

- 1) check for valid userId, if not return "invalid user"
- 2) check for valid drive, if not return "invalid drive"
- 3) validate path; if invalid return "path not found"
- 4) find first file matching specification; if none return "file not found"
- 5) return "ok", search handle and matching file info

SearchNext

- 1) check for valid userId, if not return "invalid user"
- 2) check for valid search handle, if not return "file not found"

3) find next file matching specification; if
none return "file not found"

4) return "ok" and matching file info

5 WriteFile

1) check for valid userId, if not return
"invalid user"

2) check for valid file handle, if not return
"invalid handle"

10 3) check for file locks; if bytes are locked
return "sharing violation"

4) if no room for data return "disk full"

5) find which file extent byte range is located
in

15 a) if extent is on WO:

i) if the start byte is greater than
the start of the extent, change the end byte of
the extent to start - 1.

20 ii) if the end byte is less than the
end of the extent, create a new extent for end + 1
to end of extent

iii) create an extent for start to end
and write data to HD

b) if extent is on HD:

25 i) if no data follows this extent,
just enlarge the extent and write the data to HD.

ii) create a new extent and write the
data to HD

6) return "ok"

30

WriteImage

1) check for valid userId, if not return
"invalid user"

35 2) check for valid drive, if not return "invalid
drive"

3) check that data starts at next consecutive
byte position from previous command; if not return
"missing data"

4) write data to WO

5) return "ok"

AddUser

- 1) check for valid userId (administrator only),
5 if not return "invalid user"
- 2) create new user record
- 3) return "ok"

10 ChangePassword

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) find user record for userName; if no match
return "record not found"
- 15 3) update record
- 4) return "ok"

DeleteUser

- 1) check for valid userId (administrator only),
20 if not return "invalid user"
- 2) find user record for userName; if no match
return "record not found"
- 3) delete record
- 4) return "ok"
- 25

Diagnostics

30 Eject

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) see if any users are logged in; if so return
"in use"
- 35 3) flush any data in HD buffers to WO
- 4) write ETD to WO
- 5) eject disk
- 6) return "ok"

Format

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) check for valid drive, if not return "invalid
5 drive"
- 3) check for consistency between mode and number
of tracks; if a problem return "invalid format"
- 4) check individual track sizes; if invalid
return "bad track size"
- 10 5) check that total of tracks will fit on disk;
if not return "disk full"
- 6) save format info in system data area
- 7) return "ok"

15 LogoutAll

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) prevent any new users from logging in
- 3) logout all users not currently executing a
20 command
- 4) as each outstanding command terminates,
logout user
- 5) return "ok"

25

LogoutUser

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 30 2) wait for user's current command to complete
- 4) logout user
- 5) return "ok"

Reboot

- 35 1) check for valid userId (administrator only),
if not return "invalid user"
- 3) lock out all users
- 2) reboot peripheral device

Rebuild

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) if any users are logged in, return "busy"
- 5 3) lock out all users
- 3) rebuild file system
- 3) re-enable logins
- 4) return "ok"

10

Shutdown

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 2) if any users are logged in, return "busy"
- 15 3) lock out all users
- 4) close all databases and system files
- 5) return "ok"
- 6) halt

20

SystemStats

- 1) check for valid userId (administrator only),
if not return "invalid user"
 - 2) return list of users currently logged on
- 25

UserList

- 1) check for valid userId (administrator only),
if not return "invalid user"
- 30 2) return list of users records

WriteTOC

- 1) check for valid userId (administrator only),
35 if not return "invalid user"
- 2) if TOC has already been written, return "TOC
already exists"
- 3) write TOC to WO
- 4) return "ok"

User Operation:

The peripheral device is connected directly to a Novell Network line and is powered on. All the software required to boot up on the network is contained on the peripheral device internal hard drive. Updates may be installed by inserting a special CD into the peripheral device Orange Book drive before powering on.

10 The following software must be installed in each workstation intending to access the peripheral device:

1. Host peripheral device TSR.
2. Peripheral device LOGIN program that
15 contains: LOGIN, LOGOUT, FORMAT, FLUSH, and EJECT functionality.
3. Peripheral device ADMIN program (Network administrator only).

20 The Host peripheral device TSR is installed from AUTOEXEC.BAT or executed separately by the user (it can be initiated from LOGIN as well).

25 The user must LOGIN to the peripheral device unit to establish read and write permission. Read permission is always assumed at LOGIN and exclusive read may be specified to block other users from reading. Write permission is also assumed at LOGIN and exclusive write may be specified to block other users from writing. When the user is logged in with read/write permission,
30 a disc in the peripheral device unit is read/written using any DOS read/write access until LOGOUT is run. LOGIN and LOGOUT do not support any passwords.

35 If the disc is blank, the FORMAT utility (in LOGIN) must be run to determine how the disc will be used. A blank disc may be formatted in the following ways.

1. TOC (table of contents) is written with 99 7M tracks

2. TOC is written with special user defined track sizes

3. No TOC is written and indicates dynamic tracks

5 Disc type 1 is 100% compatible with all CDROM drives if the Meridian CDROM Redirector or other Frankfurt driver is used. This formats the disc into 99 equal length tracks of approximately 7 MBytes. The TOC is written to reserve these
10 tracks, but not actual track data is written at FORMAT time. This provides a floppy like media that appears as a single track disc that can be written many times. The 100% CDROM compatibility allows the disc to be read on any CDROM drive
15 between write sessions. Cached data on the internal hard drive is automatically written to disc when 7M of data is available. Path table information is only written to the disc at the end of a session (disc is removed from the Orange Book
20 drive).

Disc type 2 is 100% compatible with all CDROM drives and is used to publish standard ISO-9660 CDROMs. These discs are not accessed through the peripheral device File System but support the
25 special Publishing Commands in the peripheral device Protocol specification. MSCDEX is required to read standard ISO-9660 discs created with peripheral device.

Disc type 3: Not currently defined.

30 It is also possible to put the CDROM Redirector onto track 1 of any type disc to allow the standard MSCDEX to read and execute our redirector. This ensures compatibility for users that do not currently have a redirector (assuming
35 that block append is not used).

The FLUSH utility is used to write all current data in the cache hard drive including path tables to the disc in the Orange Book recorder. This does not eject the disc, but

guarantees that the Orange Book disc is current and may be removed with no further writing.

The EJECT utility is used to flush the current cache and eject the CDROM from the CDWO.

5 The EJECT function may also be an option in LOGOUT. The peripheral device traps the eject button the CDWO and prohibits this functionality. The EJECT utility is the only way to eject the disc unless the appropriate rear-panel switch is
10 set in the peripheral device.

The ADMIN utility is on a separate floppy and allows the network administrator to:

1. Update new versions of downloadable peripheral device code.
- 15 2. Read/write permission override.

The present invention can be utilized in various configurations, such as shown in FIGS. 6 - 9:

1. Standard Peripheral (FIG. 6)
- 20 2. Integrated Peripheral (FIG. 7)
3. Expandable Peripheral (FIG. 8)
4. Jukebox Peripheral (FIG. 9)

Standard Peripheral (FIG. 6):

25 Standard peripheral 240 combines a peripheral device 5.25" unit 242 with a JVC, Philips or other Orange Book recorder 24 in a small box with a power supply. Communication to the host can be Network or SCSI control 250 (not both). This can
30 connect directly to a Novell network and receive peer to peer communications from the host computer. This can optionally connect directly to a host computer through the SCSI bus. This product provides appendable backup, archival and
35 data interchange. Additional host publishing software will also allow the creation of more complex discs such as CDROMXA, CDI and audio discs.

Integrated Peripheral (FIG. 7):

The integrated peripheral 260 of FIG. 7 places the peripheral device hardware/software 266 (not in 5.25" form factor) inside of an existing recorder product 264 such as the Philips Orange Book drive. Communication to the host can be Network or SCSI control 268 (not both).

Expandable Peripheral (FIG 8):

The expandable peripheral 270 of FIG. 8 uses a PC tower with many slots for CDROM such as 280, 281, 282 and 5.25" Orange Book recorders, such as 274, 275, 276. This allows many recorders and CDROM drives to be controlled from the peripheral device 5.25" unit or 386/486 CPU system.

Jukebox Peripheral (FIG. 9):

The jukebox peripheral 290 of FIG. 9 includes a 5.25" peripheral device 294 with an Orange Book recorder 296 inside of a Jukebox 292. This provides the most powerful backup and archival system with access to over 100 blank and recorded discs.

According to the present invention, a file management system has been described which includes a read/write random access device having removable read/write media and a host computer including a first operating system. The present invention further provides an intelligent peripheral device comprising means for interfacing with the host computer, means for interfacing with the random access device, and means for providing dedicated file management for the random access device.

The present invention also includes means for providing direct and immediate access to the random access device, and means for providing exchange of data between the host computer and the random access device, including means for

controlling the transfer of file data (conformed file data) to the random access device which is in conformance with international standards.

According to a further aspect, the present
5 invention also includes means for controlling the transfer of the conformed file data to the random access device independently of the host computer.

According to still another aspect of the present invention, the peripheral device further includes
10 means for compressing the file data before transfer to the random access device while maintaining the conformance with the international standard.

The foregoing descriptions of specific
15 embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and it should be
20 understood that many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable
25 others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended
30 hereto and their equivalents.

WHAT IS CLAIMED IS:

1. In a file management system including a read/write random access device having removable
5 read/write media and a host computer including a first operating system, an intelligent peripheral device comprising:
means for interfacing with said host computer,
10 means for interfacing with said random access device,
means for providing dedicated file management for said random access device,
means for providing direct and immediate
15 access to said random access device, and
means for providing exchange of data between said host computer and said random access device, including means for controlling the transfer of file data to said random access device which is in
20 conformance with international standards.
2. A peripheral device as in Claim 1 including means for controlling the transfer of said file data to said random access device independently of said host computer.
3. A peripheral device as in Claim 2 including means for compressing and decompressing said file data before transfer to said random access device while maintaining said conformance
5 with said international standards.
4. In a file management system comprising a read/write random access device having removable read/write media a host computer including a first operating system,
5 an intelligent peripheral device including means for interfacing with said host computer,

- means for interfacing with said random access device,
- 10 means for providing dedicated file management for said random access device,
- means for providing direct and immediate access to said random access device, and
- means for providing exchange of data between
- 15 said host computer and said random access device, including means for controlling the transfer of conformed file data to said random access device.

5. In a file management system including optical memory means, a host computer including a first operating system, and an orange book recorder for handling exchange of data with said
- 5 optical memory means, a peripheral device comprising:

- means for interfacing with said host computer,
- means for interfacing with said orange book
- 10 recorder, and
- means for providing file management for said orange book recorder so as to establish direct compatibility between said first operating system and said optical memory means.

6. A file management system comprising optical memory means,
- a host computer including a first operating system, and
- 5 an orange book recorder for handling exchange of data with said optical memory means, and
- a peripheral device including
- means for interfacing with said host
- 10 computer,
- means for interfacing with said orange book recorder, and

means for providing file management for said
orange book recorder so as to establish direct
15 compatibility between said first operating system
and said optical memory means.

7. A file management system as in Claim 6
wherein said host computer includes a terminate
and stay resident (TSR) program.

8. A file management system as in Claim 7
wherein said TSR program includes a peripheral
device redirector.

9. A file management system as in Claim 7
wherein said TSR program includes a peripheral
device applications program interface (API).

10. A file management system as in Claim 7
wherein said TSR program includes a peripheral
device protocol interface.

11. A file management system as in Claim 7
wherein said TSR program includes a network
interface.

12. A file management system as in Claim 7
wherein said TSR program includes a SCSI
interface.

13. A file management system as in Claim 9
wherein said TSR program includes a block device
driver (BDD) for connecting MSCDEX with said API.

14. A file management system as in Claim 9
wherein said file management system includes
different operating systems and wherein said
peripheral device protocol interface means
5 includes means communicating to said peripheral
device from said different operating systems.

15. A file management system as in Claim 14 wherein said peripheral device includes peripheral device software means, including kernel software means for controlling all system functions.

16. A file management system as in Claim 15 including network interface software means for interfacing with said host computer.

17. A file management system as in Claim 15 including SCSI interface software means for interfacing with said host computer.

18. A file management system as in Claim 15 including peripheral device file system means for managing data with said optical memory means.

19. A file management system as in Claim 18 including peripheral device protocol translator means for receiving peripheral device protocol structures from said interface means and for
5 decoding the commands to be sent to said peripheral device file system means.

20. A file management system as in Claim 19 including file structure means interfacing to said peripheral device file system means for converting data to said file structure before writing to said
5 optical memory means.

21. A file management system as in Claim 20 including compact Disc Write Once (CDWO) interface means connected between said file structure means and said orange book recorder for controlling the
5 operation of said orange book recorder.

22. A file management system as in Claim 21 wherein said optical memory means include host CDROM software means.

23. A file management system as in Claim 22 wherein said CDROM software means include a CDROM redirector for allowing CDROM drives to appear as a local hard disk drive.

24. A file management system as in Claim 23 wherein said CDROM redirector software means includes a CDROM block device driver.

25. A file management system as in Claim 24 wherein said CDROM redirector software means includes read means for bridging said CDROM redirector and said CDROM block device driver.

26. A file management system as in Claim 6 wherein said peripheral device is a standard peripheral.

27. A file management system as in Claim 6 wherein said peripheral device is an integrated peripheral.

28. A file management system as in Claim 6 wherein said peripheral device is an expanded peripheral.

29. A file management system as in Claim 6 wherein said peripheral device is a jukebox peripheral.

10

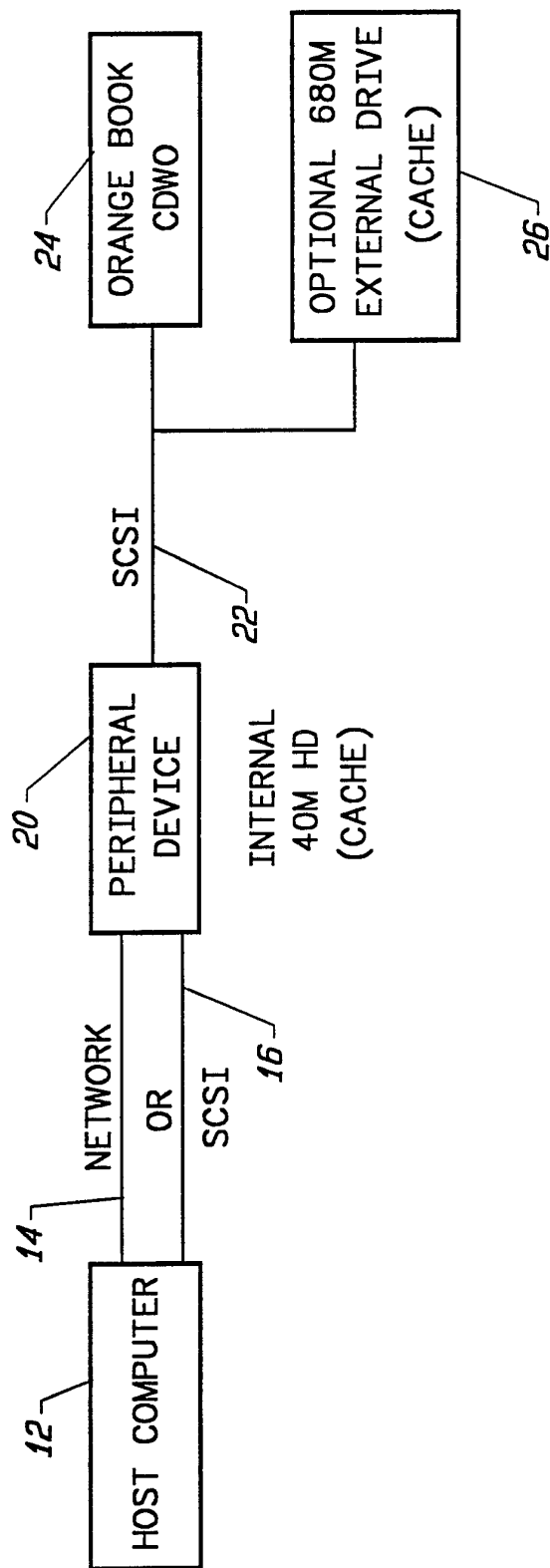
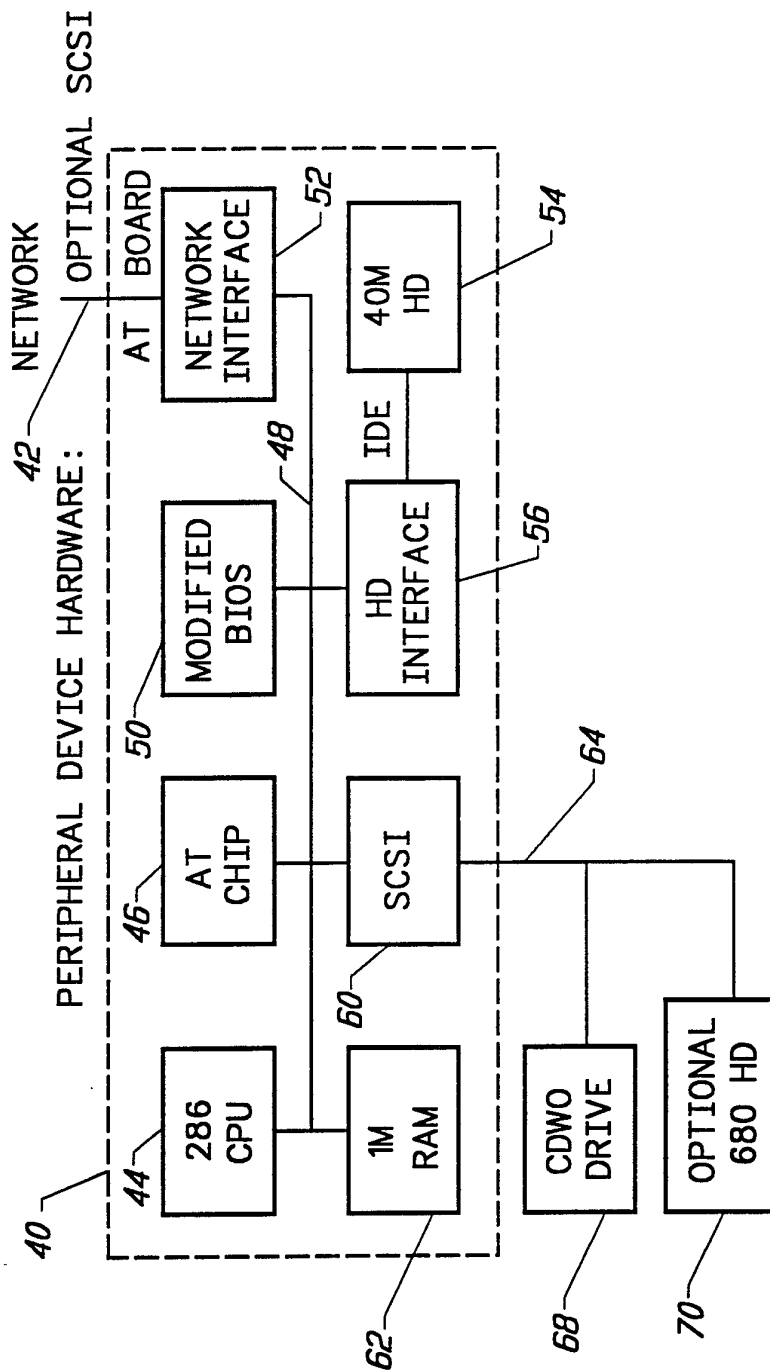
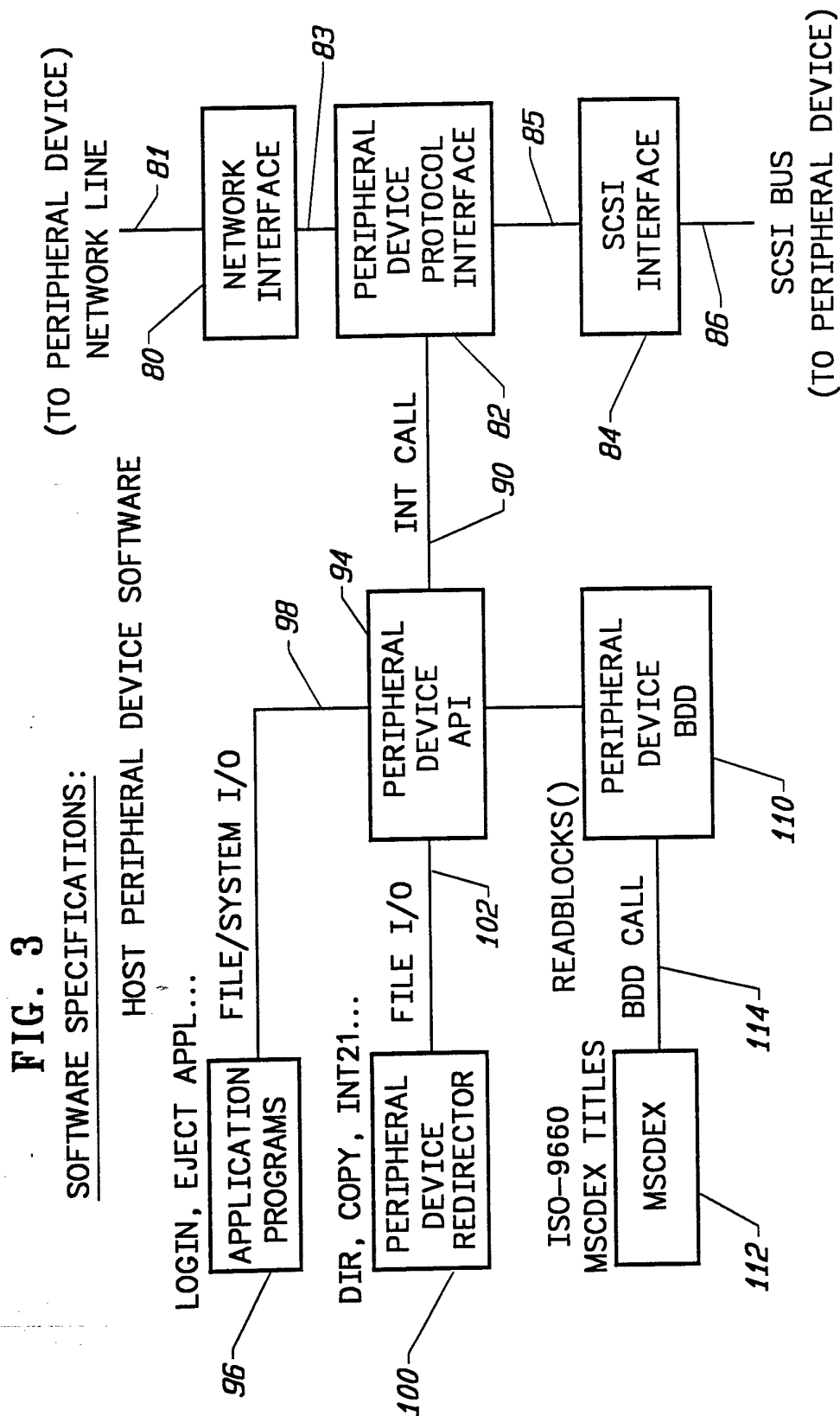


FIG. 1

FIG. 2
HARDWARE SPECIFICATIONS:





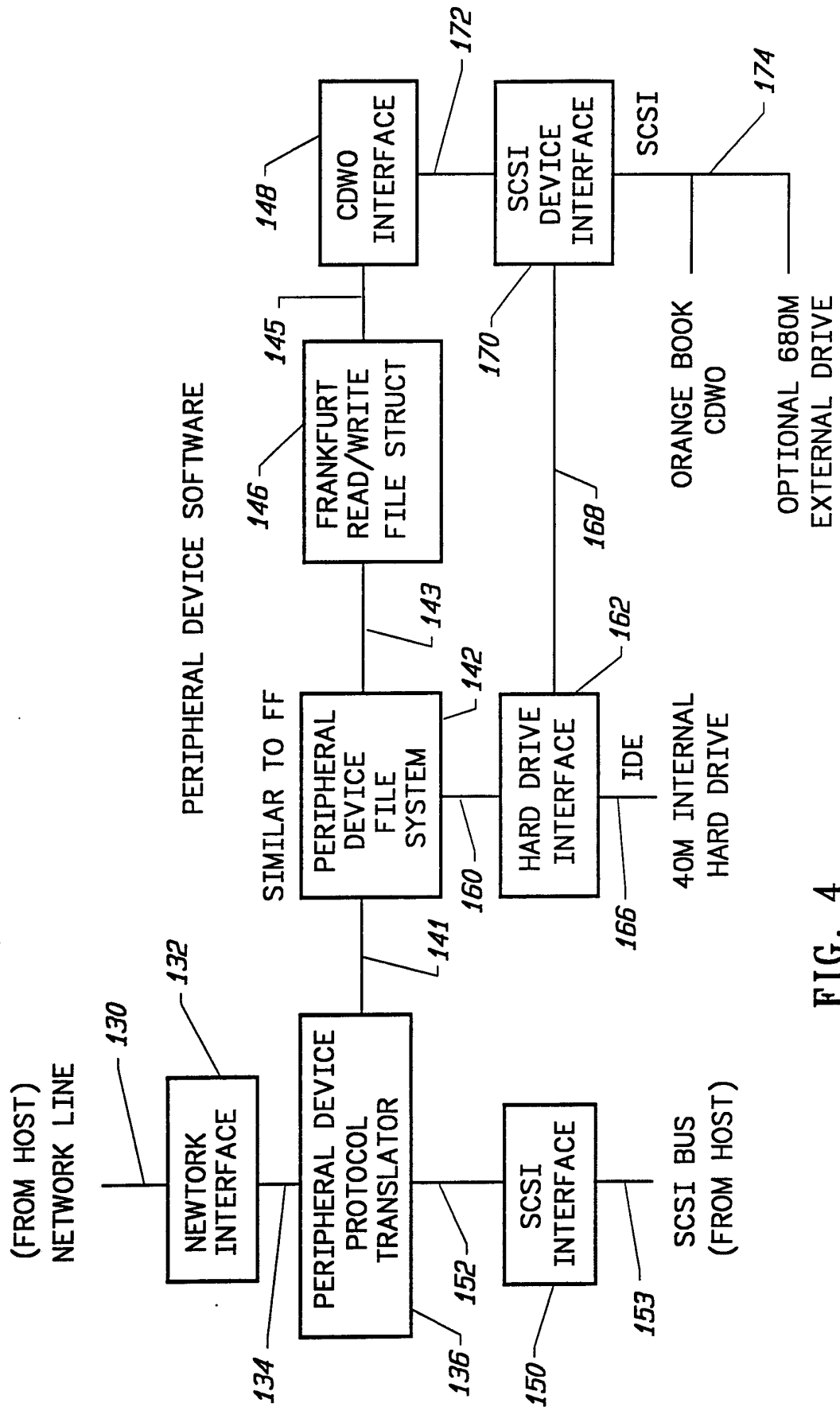
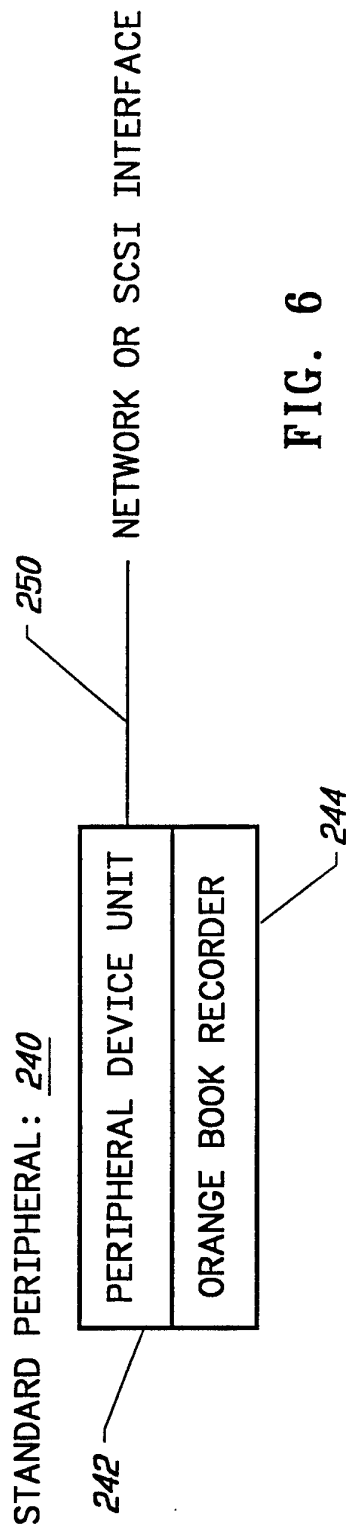
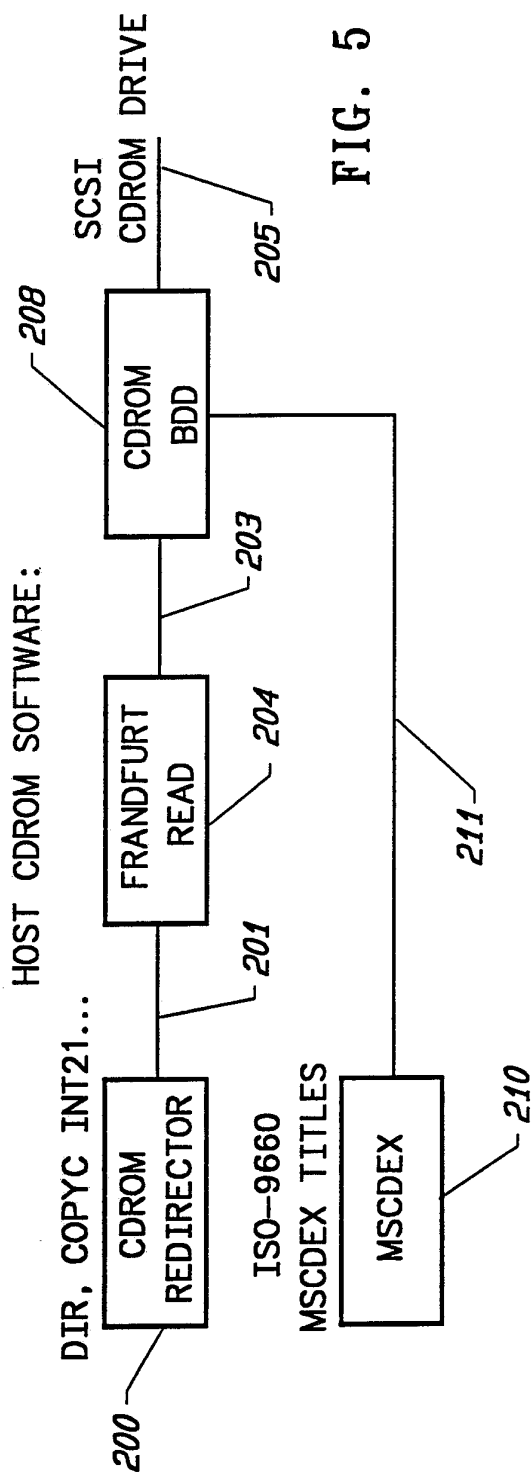
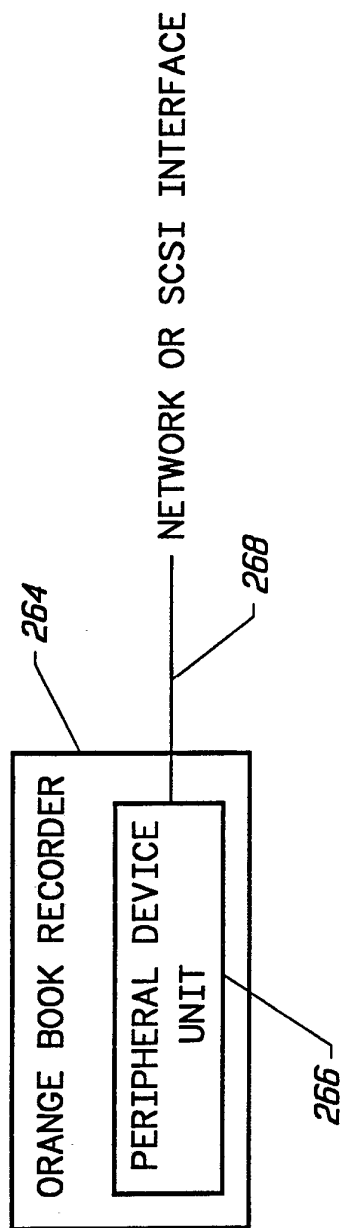


FIG. 4



INTEGRATED PERIPHERAL: 260

FIG. 7



EXPANDABLE PERIPHERAL: 270

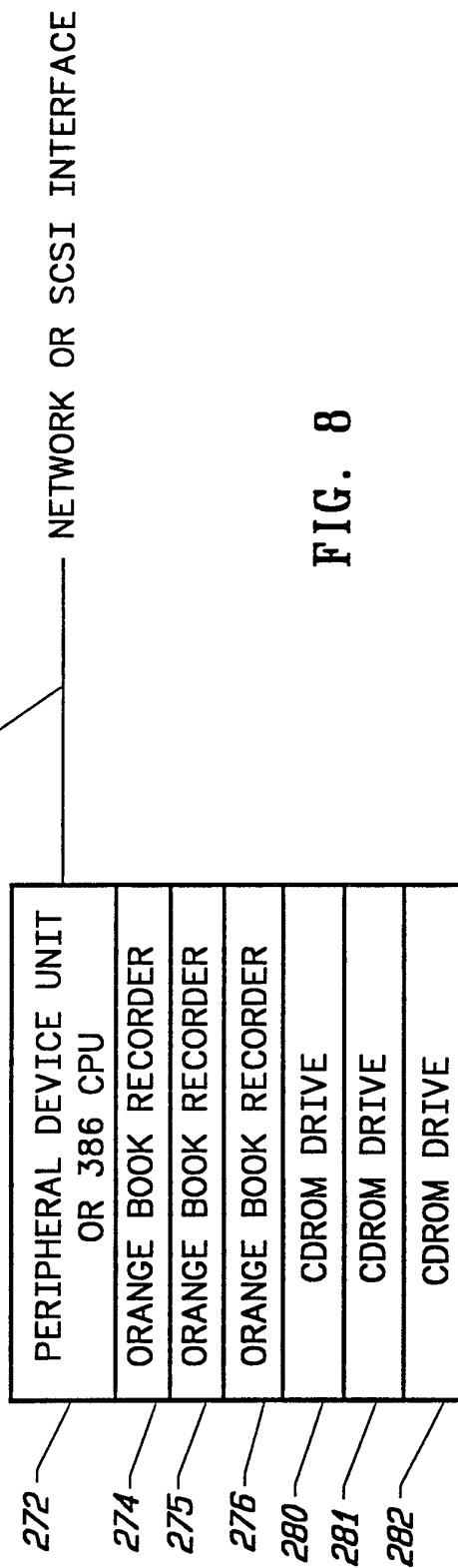


FIG. 8

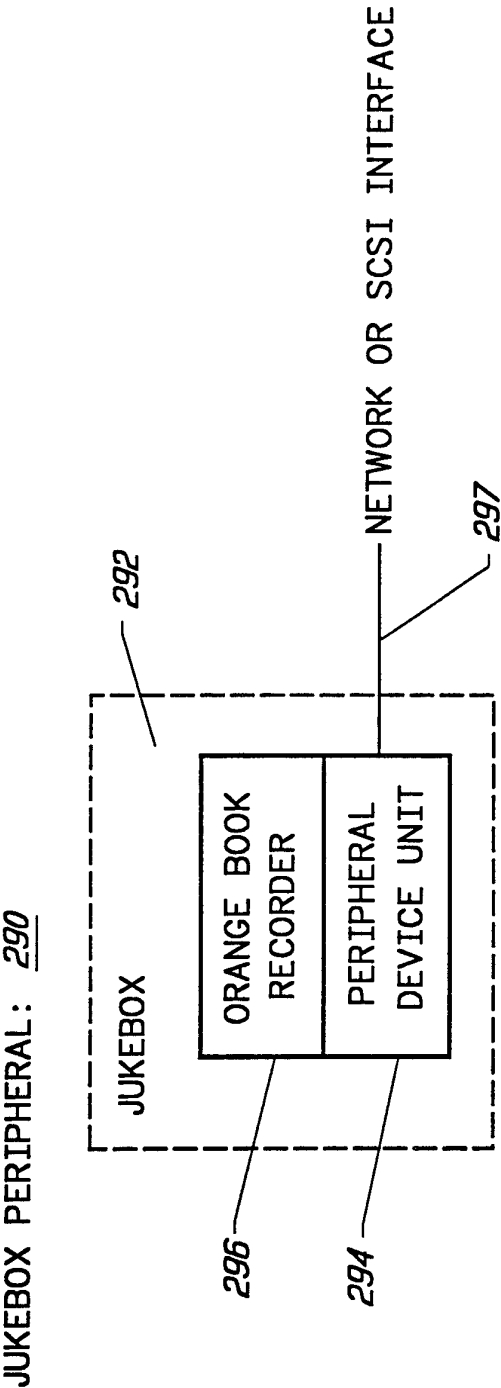


FIG. 9

INTERNATIONAL SEARCH REPORT

PCT/US92/09274

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 12/00

US CL :395/500,200

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/500,200

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

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

DIALOG: (JVC OR PHILIPS) AND SCSI AND ISO(W) 9660

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,E	US,A 5,168,444 (CUKOR ET AL.) 01 DECEMBER 1992, SEE ENTIRE DOCUMENT	1-29
Y	US,A 5,040,110 (MIKI ET AL.) 13 AUGUST 1991, SEE ENTIRE DOCUMENT	1-29
Y	US,A 4,949,248 (CARO) 14 AUGUST 1990, SEE ENTIRE DOCUMENT	1-29
Y	PC SOURCES, OCTOBER 1991, HILL ET AL., OPTICAL EXPLOSION, P. 389(8).	1-29
Y	COMPUTER GRAPHICS WORLD, AUGUST 1990, MICHAEL L. SENA, DATA DELUGE, P. 123(4).	26-29



Further documents are listed in the continuation of Box C.



See patent family annex.

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

Date of the actual completion of the international search

16 DECEMBER 1992

Date of mailing of the international search report

11 JAN 1993

Name and mailing address of the ISA/W
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. NOT APPLICABLE

Authorized officer

WILLIAM M. TREAT

Telephone No. (703) 308-1643

Nguyen Ngoc Ho
NGUYEN NGOC-HO
INTERNATIONAL DIVISION