APPLICATION NOTE

Software Driver of SmartMedia(TM)

(Ver 3.0)

MEMORY PRODUCT & TECHNOLOGY
SAMSUNG ELECTRONICS Co., LTD

Revision History

Revision	Date	Name	Revision Contents
1.0	98/12/10	Y.G. Kim	Development of DOS/FAT FileSystem
1.1	99/03/04	Y.G. Kim	Zone Concept Bug (Up to 32MB)
2.0	99/11/05	Y.G. Kim	Upgrade to 128MB SmartMedia
3.0	00/04/19	Y.G. Kim	Support to SEC'S Parallel Port Interface

Software Driver of SmartMedia(TM)

CONTENTS

1.0 Introduction

Production of External Memory Card was launched to respond to diversified Mobile Application by using the same Chip as TSOP packaged Nand Flash ;this Card necessitated the standardization of Data Structure in order to ensure the compatibility between different equipment. In response to this, SSFDC Forum jointed by chip vendors as well as various system vendors was organized in 1996 to work out the SmartMedia(TM) Logical and Physical specification by modifying DOS/FAT File System compatible to PCMCIA. This Software is a type of FTL (Flash Translation Layer : software between host and Flash to emulate disk) as Flash File Management. The program is based on the ISA Bus Interface system,EPP Interface system developed by SEC is presented for an exemplary platform. This Software has embodied the ECC which detects up to 2 bit errors out of 256 bytes and corrects 1 bit error into the software and the hardware chip as it is used by the Forum. The Software can be used as well for single TSOP NAND Flash without any modifications, but usually they use themselves structures.

- Main feature and configurations

- The Interface which controls nand Flash is based on the configuration of ISA bus interface demo board and Parallel port interface demo board using glue logic designed by SEC.
- If you want to use parallel port interface demo board,you must set into EPP mode in PC BIOS setup menu.
- For Selecting Hardware option,you must modify SM_TARGET by SM_ISA or SM_EPP in smio.h.
- Hardware is only possible with available Hardware ECC Demo Board.
-
- Operation of this Software is based on the DOS/FAT File System.
- This Software supports 1,2,4,8,16,32,64,128MB SmartMedia(TM) Optimized Format and Functions to operate File can use up to 128MBytes in Ver 2.0.
- This Program Source is written by Turbo-C++.
- This information is based on the PC Card Standard.
- The software complies with the Physical Format Specification Ver 1.21 (Published on May.1999) and Logical Format Specification Ver 1.11 (Published on May. 1999) of SSFDC Forum.

2.0 SmartMedia Operating System

Operating System determined by SSFDC Forum is run by DOS/FAT File System identical with HDD to ensure the compatibility with mass storage. DOS/FAT File System (herein after referred to as “the FAT File System”) is a general operating system basically used by various OS. FAT File System operates CHS configuration by introducing the concept of Cluster and Sector. Cluster consists of Sector while the Sector, a basic unit of Write and Read was applied to Optimized Format in the same way as the unit now applied by Mass Storage like HDD, etc and the size of one Sector was determined to accommodate the same 512 Byte as the current HDD for the resulting greater compatibility. Besides, the type of Format is determined depending on how many Sectors will be taken for Cluster which is a basic unit of File Write and Read. Optimized Format aligned one Cluster with the size of one Block to facilitate the operation of SmartMedia(TM). For more details, we will refer to 2.2 PBR, but first of all, let us have a look into FAT File System. FAT File System basically needs the following 4 types of factors and is designed to be operated through their inter-relationship. To begin with, let us have a closer look at those component factors and then discuss FAT File System.

4 Types of Component Factors

2.1 Master Boot Record(MBR)

FAT File System is basically composed of Cylinder,Head,Sector(CHS) and Master Boot Record in the concept of System BIOS as the most important Boot Code among basically necessary Sectors at the time of initial Format is Written at ‘ 0’ Sector which is an absolutely fixed logical sector.

Its size is equal to that of one Sector and for Optimized Format, the size of mainly 1 Sector is set at 512Byte. As a result of this, the first Sector 512Byte is MBR; this Sector is specified in Table 1 as defined in the PC-Card Standard. Contents of MBR show that the location of Partition Boot Record Sector and Number of Last Sector are written by CHS Mode and LBA Mode respectively.

Formula showing relationship between CHS and LBA Mode is as follows:

CHS Mode : Refers to Cylinder,Head,Sector Mode.

LBA Mode : Refers to Logical Block Address indicating sequential (0, 1, 2, 3,….)Counting Value of CHS.

LBA Mode Value = (C x HpC + H) x SpH + S - 1

where C is Cylinder Number(starting with 0)

H is Head Number(starting with 0)

S is Sector Number(starting with 1)

HpC is Head per Cylinder

SpH is Sector per Head

Optimized Format is based on Format of Disk Media(Floppy Disk,Hard Disk) and requires C - H - S parameter. For the compatibility of Data, Parameter must be determined. This parameter has already been set for Optimized Format as shown in Table1.

Table1 Optimized Format Parameter

	1MB	2MB	4MB	8MB	16MB	32MB	64MB	128MB
NumCylinder	125	125	250	250	500	500	500	500
NumHead	4	4	4	4	4	8	8	16
NumSector	4	8	8	16	16	16	32	32
SumSector	2,000	4,000	8,000	16,000	32,000	64,000	128,000	256,000
SectorSize	512	512	512	512	512	512	512	512

Composition of MBR shows the following contents.

Table 2 Master Boot Record Configuration

The Master Boot Record contains the following fields:

Offset	Size(Bytes)	Description
000h	446	Boot Code
1BEh	16	Partition Entry
1CEh	16	Partition Entry
1DEh	16	Partition Entry
1EEh	16	Partition Entry
1FEh	2	Signature Word(0x55AAh)

Only first Partition Entry out of 4 Partition Entries is used and its contents are as follows:.

Table 3 16Bytes Entries Configuration

Offset	Size(Bytes)	Description
00h	1	x 86 Default Boot Partition(00h=Not Default,80h=Default
01h	1	StartHead-Zero-based(0) head number
02h	1	StartSector-One-based(1) sector number.Bits 6,7 are high bits of Zero-based(0) Cylinder number.
03h	1	Start Cylinder
04h	1	Partition Type
05h	1	EndHead-Zero_based(0) head number
06h	1	EndSector-One-based(1) sector number
07h	1	EndCylinder
08h	4	StartSector(relative to beginning of Extended MS-DOS)
0Ch	4	NumSectors

Example of Contents and Application of Master Boot Record at 16MByte SmartMedia(TM)

Master Boot Record (Contents of 0 Sector)

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   .
   .
   .
   .
0190     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01B0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 02
01C0     0a 00 01 03 50 f3 29 00 00 00 d7 7c 00 00 00 00
01D0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01E0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01F0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa
```

Figure 1 Hex dump of the 0 sector of 16MByte SmartMedia

Interpretation of 16Byte of the First Entry among the above contents.

1) 1BFh ~ 1C1h : Indicating the location of Partition Boot Sector(by CHS mode)

For 16MByte SmartMedia, if we calculate values of Partition Boot Record among contents of 16Byte Entries as defined in Table3, the result is :

StartHead = 02h ,StartSector = 0ah , StartCylinder = 0h

Besides, in the case of 16MByte SmartMedia in Table 1,

HpC = 4 , SpH = 10h. Calculation of this into LBA Mode Value will show the location of Partition Boot Record.

Partition Boot Sector = (0h x 4 + 02h) x 10h + 0ah - 1 = 29h (Sector)

2) 1C3h ~ 1C9h : Indicating the location of Last Sector.

* EndSector-Zero-based(1) sector number(Bits 6,7 are high bits of zero-based(0) Cylinder number) ==>EndSector is 10h excluding upper 2bits from 50h and End-Cylinders 1f3h enhanced by 01 binary above f3h according to the Regulations. (For Partition Boot Sector Location, the upper 6,7 bit 00 binary remains unchanged.) As shown in the above item 1), calculation of this into LBA Mode Value shows the following result:

Last Sector = (1f3h x 4 + 03h) x 10h + 10h - 1 = 7CFFh

3) 1CAh ~ 1CDh : Available Partition Size

Available Partition Size is 7CFFh for Total Sector Size from the Zero-based perspective. (For 16MByte SmartMedia, the Table1 shows 32,000(decimal) = >7D00h).

If the value of location of Partition Boot Record is subtracted from this value, the Available Partition Size is obtained at once.

7CFFh - 29h = 7CD6h+1(Zero-based)=7CD7h

==> matches the value of 1CAh,1CDh.

2.2 Partition Boot Record(PBR)

PBR sector means a starting sector of available sector defining the Basic Parameter among the contents of Format in which Label, File System Type (FAT Structure), etc. including BIOS Parameter Block are written. Most values such as BytesPerSector , SectorsPerCluster , NumFATs , RootDirEntries , etc. defined in BIOS Parameter Block included in the above contents represent the definition of Important Parameter indispensable for determining the configuration of Format. Further details are specified in Table 4 and 5.

Table 4 Partition Boot Record Configuration

The Partition Boot Record contains the following fields

Offset	Size(Bytes)	Description	
000h	3	JMP instruction to PBR boot code	
003h	8	OEMName and Version	
00Bh	25	BIOS Parameter Block(BPB)	
024h	1	DriverNumber(00h = Floppy,80h = Fixed)	
025h	1	Reserved, not in use	
026h	1	ExtBootSignature-29h	
027h	4	VolumeID or Serial Number	
02Bh	11	Volume Label-ASCII characters.Padded with blanks if less than eleven(11) characters.	
036h	8	FileSysType-ASCII Characters identifying file system type. Padded with blanks if less than eight(8) characters. One of the following values:	
		Value	Meaning
		FAT12	12-bit File Allocation Table(FAT)
		FAT16	16-bit File Allocation Table(FAT)
03Eh	448	Boot Code	
1FEh	2	Signature Word - 55AAh	

Table 5 BIOS Parameter Block Configuration

The BIOS Parameter Block(BPB) contains the following fields:

Offset	Size(Bytes)	Description
000h	2	BytesPerSector-Number of bytes per sector
002h	1	SectorsPerCluster-Number of sectors in a cluster
003h	2	ReservedSectors
005h	1	NumFATs-Number of Root on the media
006h	2	RootDirEntries-Number of Root Directory entries
008h	2	TotalSectors if Sector is over 65,535 this field is zero and actual number of sectors is in the HugeSectors field.
00Ah	1	MediaIDByte-Used to quickly identify how the media is formatted. F0h: Various types of media F8h : Hard disk,any size F9h : 720KB 3.5” or 1.2MB 5.25” FAh : 320KB 5.25 ” FBh : 640KB 3.5” FCh : 180KB 5.25” FDh : 360KB 5.25” FEh : 160KB 5.25” FFh : 320KB 5.25”
00Bh	2	NumFATSectors-Number of sectors in each FAT
00Dh	2	SectorsPerTrack-Number of sectors on a track
00Fh	2	NumHeads-Number of heads
011h	4	HiddenSectors-Number of hidden sectors
015h	4	HugeSectors-Number of sectors if Total sectors are zero

An examination of the example of 16MByte SmartMedia(TM) operated according to the contents of Partition Boot Record defined as above will be of help to understanding the above contents.

Contents and Example of Application of Partition Boot Record at 16MByte SmartMedia(TM)

Partition Boot Record(Location determined by MBR :29h Sector)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	e9	00	00	20	20	20	20	20	20	20	20	00	02	20	01	00
0010	02	00	01	d7	7c	f8	03	00	10	00	04	00	29	00	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	46	41	54	31	32	20	20	20	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
.																
.																
.																
.																
01B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	aa

Figure 2

An analysis of the above contents shows the following:

1) 0Bh ~ 0Ch : 0200h(512Byte) Byte/Sector
2) 0Dh : 20h (32Sector) Sector/Cluster
3) 0Eh ~ 0Fh : 0001h Reserved Sector
4) 10h : 02h Number of FAT
5) 11h ~ 12h : 0100h(256Entries) Number of Root Directory Entry
6) 13h ~ 14h : 7CD7h(31959) Total Sector Number of Partition
7) 15h : f8h Signature(Hard Disk)
8) 16h ~ 17h : 02h Numsector of FAT
9) 18h ~ 19h : 0010h(16) Number of Sector
10) 1Ah ~ 1Bh : 0004h Number of Head
11) 1Ch ~ 1Dh : 0029h Number of Hidden Sector
12) 1EFh ~ 1FFh : AA55h Signature

2.3 File Allocation Table(FAT)

FAT is important area of finding file's linked-list. The limit of total number of Cluster leads to the use of 12bit FAT and 16bit FAT, etc. and SmartMedia(TM) up to 2M~64MByte follows 12bit FAT Operating System. 128MByte SmartMedia(TM) Format which must use the available Cluster numbering up to 7,996 poses constraints for 12bit FAT and therefore follows 16bit FAT Operating System. The total number of Sector occupied by FAT is the minimum number of Sector which can express the total number of Cluster x 1.5 Bytes. The starting location is from the Sector right next to BPB and the contents of FAT2 for Backup are the same as those of FAT1 and takes up the Sector after the Sector occupied by FAT1. FAT Operating Method is a key to understanding the FAT File System and you may find it difficult if you are mistaken. We believe that explanation and repetition of examples will help you easily understand it. Operation by 12bit FAT means that the Size occupied by one Cluster is 12bits, namely 1.5Bytes. An examination of the meaning of the above figures shows the following composition. (See also 16bit FAT.)

Table 6 In Case of 12-bit FAT

Offset	Content	Description
00h ~02h	F8h,FFh,FFh	FAT ID(3 bytes)
03h and after	00h	

Table 7 In Case of 16-bit FAT

Offset	Content	Description
00h ~03h	F8h,FFh,FF,FFh	FAT ID(4 bytes)
04h and after	00h	

Table 8 FAT Content

12-bit FAT	16-bit FAT	Description
000h	0000h	Unused Cluster
001h	0001h	Reserved
002h ~ FEFh	0002h ~ FFEFh	Next Cluster Number in the chain
FF0h ~ FF6h	FFF0h ~ FFF6h	Reserved
FF7h	FFF7h	Defective Cluster
FF8h ~ FFFh	FFF8h ~ FFFFh	Last Cluster in the chain

* Understanding of FAT Cluster Chain (Reference 1)

FAT Cluster Chain starts after F8h,FFh,FFh (FAT ID) among Data of FAT Sector from which Physical Location is set at 002h Cluster Location. Physical location of FAT data indicates information of the next Cluster or information of Defect Cluster, etc. Let us understand this based on the following example:

In case of FAT Sector Data: F8h FFh FFh 04h 20h 03h FFh XFh XXh, ….. F8h,FFh,FFh represents FAT ID and 3Bytes of 04h,X0h,FFh is divided by 1.5Bytes according to the following rule:

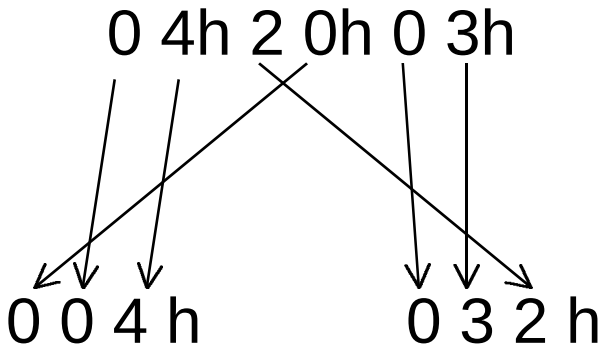


Figure 3

In this way, Physical FAT Data is 004h,032h,FFFh,, etc. emerge and their Physical Location is fixed while Start Cluster Location is set at 002h Cluster Location.

Physical	002h Cluster	003 Cluster	004h Cluster	005 Cluster	…
	004 h	032h	FFFh	XXXh	…

The above calculation shows that the 26th and 27th Byte among Directory 32Bytes Configuration (contents explained in the next Root Dir) becomes Start Cluster (Here it is assumed that 002h is written). So if we proceed to the Physical Location of 002h Cluster location, we find that 004h is written to indicate the value of Next Cluster and if we further proceed to the Physical Location of 004h Cluster location in the same way, we find that FFFh is written to indicate such value. As FFFh is an indication of Last Cluster in the Chain (See Table 8.), a connecting Cluster of this File consists of the size of two Clusters which start from 002h Cluster and end with 004h in the final analysis.

A look at Cluster by Sector shows that connection of 002 Cluster starts from the Sector right next to the Sector occupied by Root Directory Entries. For example, calculation of 16MByte SmartMedia(TM) Optimized Format indicates that since the Sector occupied by Root Directory ranges from Sector 48 to Sector 63, 002h Cluster is as large as one Cluster from Sector 64. (In case of16MByte, SectorPerCluster is Sector 32.) Therefore, 002h Cluster ranges from Sector 64 to Sector 95 and in the same way, a look at the location of Sector of 004h Cluster indicates that it ranges from Sector 128 to Sector 159 (The range from Sector 96 to Sector 127 Sector is an area of 003h Cluster). With respect to the location of Sector, it was explained in '2.4 Root Directory'.

2.4 Root Directory

The size of Root Directory is determined by the usable number of entries.Since Optimized Format defines Root Directory is 256 Entries ,so it's size is 16 Sectors. (1 Entry requires 32 Bytes) As mentioned above, 4 types of area needs at the time of Initial Format.(When the product is released by SEC, it is provided with completion of Optimized Format according to Density) These 4 types of area consist of Data while Sector put together forms Cluster and available Cluster per Density is determined.

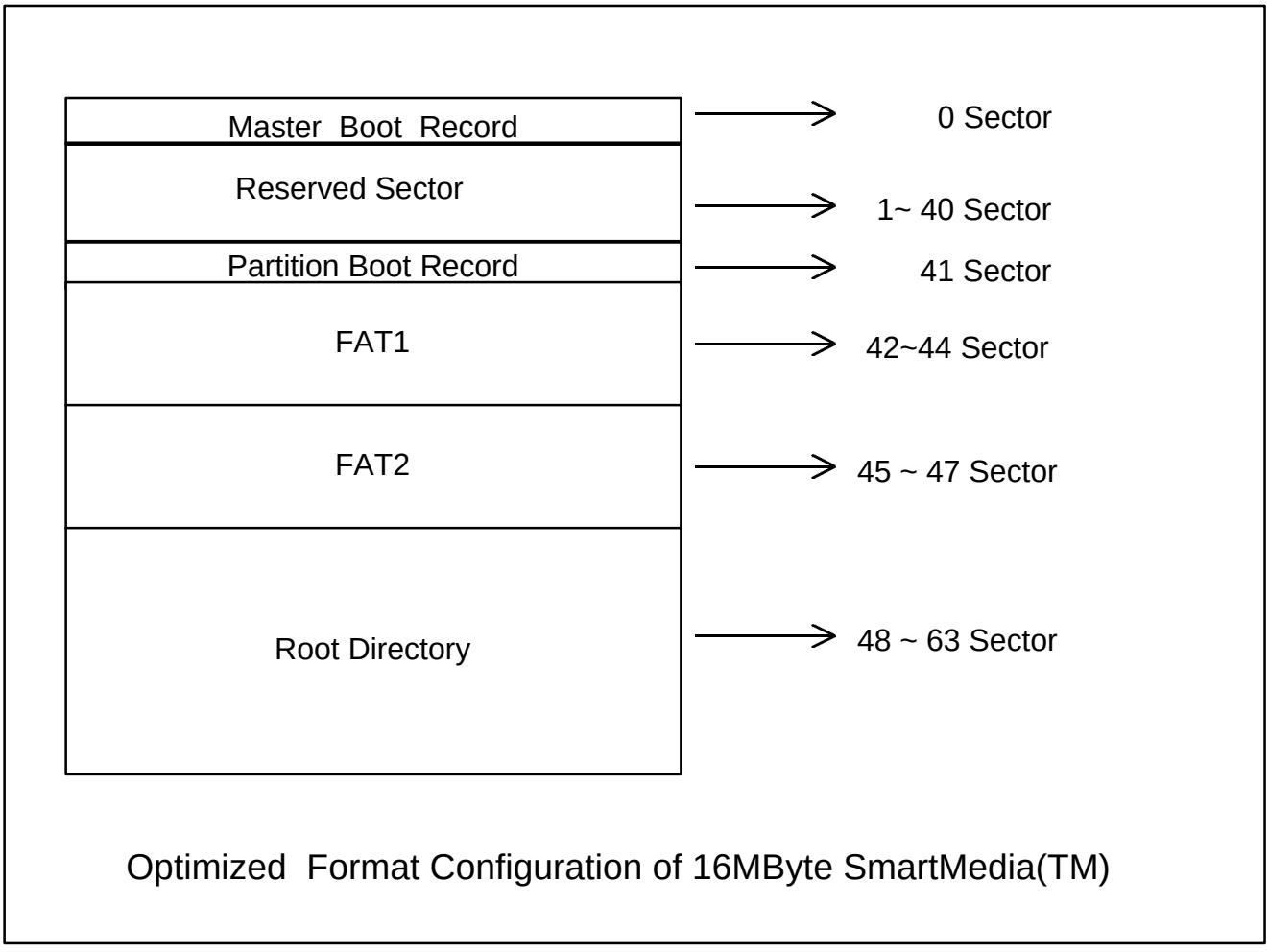


Figure 4 16MByte SmartMedia(TM) Optimized Format

Not only Root Directory, but also Sub Directory has Inform for certain File or Directory expressed in 32Bytes. A look at the configuration thereof shows the following:

Table 9 Directory 32Byte Configuration

In case of SmartMedia(TM),Initialization Values are all zero.

Byte	Content	Initialization Value
0 ~ 7	File Name	*00h,F6h … ….F6h
8 ~ 10	Extension	F6h,F6h,F6h
11	Attribute	F6h
12 ~ 21	Reserved	F6h,F6h … …F6h
22 ~ 23	Time	F6h,F6h
24 ~ 25	Date	F6h,F6h
26 ~ 27	Start Cluster Number	F6h,F6h
28 ~ 31	File Size	F6h,F6h,F6h,F6h

*00h :Unused Directory

E5h : Deleted Directory

2Eh : Sub Directory

A more accurate analysis of Attribute Regulations shows the same meaning as indicated in Table 10.

Table 10 Attribute Field Configuration

The attribute byte of the directory entry is mapped as follows

Bit	Meaning
0	Read-only:tries to open file for write or to delete file with fail
1	Hidden file; is excluded form normal searches.
2	System file; is excluded form normal searches.
3	Volume label; can exist only in root directory
4	Subdirectory; entry is excluded from normal searches.
5	Archive bit; is set to on whenever file is modified.
6	Reserved.
7	Reserved.

A look at the configuration of Time shows the following.

Table 11 Time Field Configuration

The Time field is encoded as:

Bits	Contents
00h ~ 04h	Binary number of 2-second increments(0 - 29,corresponding to 0 - 58 seconds)
05h ~ 0Ah	Binary number of minutes(0 - 59)
0Bh ~ 0Fh	Binary number of hours(0 - 23)

A look at the configuration of Date shows the following:

Table 12 Date Filed Configuration

The Date field is encoded as:

Bits	Contents
00h ~ 04h	Day of month(1 -31)
05h ~ 08h	Month(1 - 12)
09h ~ 0Fh	Year (relative to 1980)

A closer analysis of 32 Byte Content by taking an example of Figure 5 based on reference to the contents of Table 9,10,11,12 shows the following:

43 4F 4E 46 49 47 20 20 53 59 53 20 00 00 00 00 00 00 00 00 00 00 25 43 AF 24 02 00 9C 03 00 00	CONFIG.SYS

Figure 5 Example of filename and extension, 'CONFIG.SYS'

ASCII Code Value of File Name is written 43h,4Fh,4Eh,46h,49h,47h,20h,20h (20h means Space.) in the first 8Bytes, which means 'CONFIG'. Extension is 53h,59h,53h meaning 'SYS' and for Attribute, 20h, namely only the 6th bit is '1' (Archive bit) and thus it means ' File is modified'. The next 10Bytes are Reserved Area and then the next 2Bytes, namely, 4325h is information of Time. So Edited time of CONFIG.SYS are as shown in the following Figure 6.

4 , 3 , 2 , 5			Hex mode
Hours	Minutes	Seconds	
0 1 0 0 , 0	0 1 1 , 0 0 1	0, 0 1 0 1	Binary mode
8h	19h	5h	

Hour : 8h Minute : 19h Seconds : 5h x 2 = Ah

Figure 6 Example of Time field

As indicated in Figure 6, time spent on composing the File was 08 hours, 25 minutes and 10 seconds. 2 Bytes for Date, namely, 24AFh also was determined as shown in Figure7.

2 , 4 , A , F			Hex mode
Year	Month	Day	
0 0 1 0 , 0 1 0	0 , 1 0 1	0, 1 1 1 1	Binary mode
12 h	5h	Fh	

Year : 1980+12h =1998 Month : 5 Date : 15

Figure 7 Example of Date field

As shown in Figure 7, the date on which the File was composed indicates May 15, 1998. The next 2 Bytes are 002h which is the start of cluster as explained in 3.1.3 FAT File System and if we proceed to Physical Location of FAT Sector, we will find the Next Cluster written. And the next 4Bytes are 0000039Ch, which means that the File size is 39Ch. In case of 16MByte Optimized Format,1 Cluster Size is composed of 32 Sectors and 1Sector is 512 Bytes. So if we calculate this, we obtain the following:

Cluster Size = 32(Sector/Cluster)x 512(Byte/Sector) = 16,384 (Byte/Cluster)

Cluster occupied by File is calculated as follows:

Number of Cluster occupied by Total File = File Size / Cluster Size

Namely, Number of Cluster = 924(39Ch) / 16,384 = 1 (Cluster)

Accordingly, CONFIG.SYS File is the File occupying just only 002 Cluster.If we proceed to Physical Location of 002 Cluster at FAT Sector, we will certainly have the Value between FF8h ~ FFFh(indicating Last Cluster).

* Contents of Difference between Root Directory and Sub Directory (Reference 2)

In addition to the contents of Root Directory as already explained in Section 2.4 , it is necessary to know the contents of its relation to Sub Directory. So we will give you a little more detailed explanation of the contents. If we examine the value of Root_Dir_Entries among Parameters of BIOS Parameter Block of Partition Boot Record, we find that the value is 256. This means that the size of Root Directory is fixed logically. A calculation of the size shows that in cases of 1,2,4,8.16,32,64,128MByte Optimized Format, the size occupied by Root Directory is equally256 Entries. Namely, Root Directory Size = 256 (Entries) x 32(Byte) = 8192 Bytes. A conversion of this into 512(Byte/Sector), it occupies 16 Sectors. This means the size of 2 Blocks in case of 1,2MBytes, (2 Cluster size); the size of 1 Block in case of 4,8MBytes (1 Cluster size); and the size of 0.5 Block in case of 16, 32,64,128MBytes. (0.5 Cluster size)

As mentioned above, the size occupied by Root Directory has nothing to do with Cluster; it is the fixed size and lacks expansion at the same time. However, it is not the case with Sub Directory. The basic unit of Sub Directory coincides with 1Cluster in the same way as File and when necessary (when the contents are filled and it is necessary), it is expanded. As long as Usable Cluster exists, any Sub Directory can be expanded. At an initial stage, there is empty in Root Directory.However, for Sub Directory, 32 Bytes information in higher level Directory and its own 32 Bytes information are generated at an initial stage as is the case with the creation of Sub Directory. A close examination of the Inform shows the following:

** If Root Directory is higher level Directory, Start Cluster of higher level Directory is fixed at 000h Cluster.*

2E 20 20 20 20 20 20 20 20 20 10 00 00 00 00 00 00 00 00 00 00 97 89 BA 22 02 00 00 00 00 00	Current subdirectory Information
2E 2E 20 20 20 20 20 20 20 20 10 00 00 00 00 00 00 00 00 00 00 97 89 BA 22 00 00 00 00 00 00	Parent directory Information

Figure 8 Initial Stage subdirectory Contents

As shown in the above Figure 8, when 64Bytes are generated at an initial stage, the Attribute is 10h(indicating directory) and the time and date of generation are created while the File size is expressed as 00h because is Directory.

3.0 Optimized Format Applied to SmartMedia(TM)

3.1 SmartMedia(TM) Initial Format Configuration

SmartMedia(TM) in status of Write is released when Format determined by SSFDC Forum at an initial stage. It includes Block in which Format is written according to Density but additional 2 Blocks are needed before the contents of Format and its contents are as follows:

First Block : Artificial Invalid Block Marking to be done.

Second Block : Block writing Data of CIS(Card Information Structure).

In consideration of the above contents, a summary of necessary Blocks per Density is as shown in the following Table 13.

Table 13 Initial Data structure of SmartMedia

	0 Block	1Block	2 Block	3Block	4Block	5Block
2MB	Bad Block Marking	CIS Block	Format Block	Format Block	Format Block	Format Block
4MB	Bad Block Marking	CIS Block	Format Block	Format Block	Format Block	
8MB	Bad Block Marking	CIS Block	Format Block	Format Block	Format Block	
16MB	Bad Block Marking	CIS Block	Format Block	Format Block		
32MB	Bad Block Marking	CIS Block	Format Block	Format Block		
64MB	Bad Block Marking	CIS Block	Format Block	Format Block	Format Block	
128MB	Bad Block Marking	CIS Block	Format Block	Format Block	Format Block	Format Block

As shown in Table 13, for CIS Block and Format Block, SEC has written initial Format data and therefore, we guarantee they are free from defect.(excluding the First Block)

3.2 Relation between SmartMedia(TM) Optimized Format and Physical Block

In order to understand the question of what relation exists in applying the concept of Sector, Cluster to SmartMedia(TM) Physical Page,Block, it will facilitate the understanding to look at the following 16MByte Optimized Format Configuration .

Table 14 Optimized Format Configuration (Case of 16MB)

Logical Block	Physical Page	Logical Sector	Description
0	0	0	Master Boot Sector
	1	1	Spare Area
	2 ⋮ 29	2 ⋮ 29	
	30	30	
	31	31	
1	0	32	
	1	33	
	2	34	
	3	35	
	4	36	
	5	37	
	6	38	
	7	39	
	8	40	
	9	41	Partition Boot Sector
	10	42	FAT 1
	11	43	
	12	44	
	13	45	FAT 2
	14	46	
	15	47	
	16	48	Root Directory
	17	49	
	18	50	
	19	51	
	20	52	
	21	53	
	22	54	
	23	55	
	24	56	
	25	57	
	26	58	
	27	59	
	28	60	
	29	61	
	30	62	
	31	63	

The point we must not overlook in the above Table 14 is the absence of sequential coincidence between Logical Block and Physical Block of SmartMedia(TM). What is used for Spare Array in the concept of Pointer to use Physical Block of SmartMedia(TM) independently is the Block Address. We can find the Physical Location of Block Address. Let us discuss later the Regulations defining SSFDC Forum with respect to the usage of 512 Bytes in Main Array equal to 1 Sector and 16 Bytes in Spare Array.

3.3 Regulations for SmartMedia(TM) Spare Array and Block Addressing Method

An examination of the following Table 15 shows the Regulations of Spare Array.

Table 15 Spare Area Configuration (16Byte)

<table>
<tr><th>Byte No.</th><th>Contents</th><th>Byte No.</th><th>Contents</th></tr>
<tr><td>512</td><td rowspan="4">User Data Area</td><td>520</td><td rowspan="3">ECC Area-2</td></tr>
<tr><td>513</td><td>521</td></tr>
<tr><td>514</td><td>522</td></tr>
<tr><td>515</td><td>523</td><td rowspan="2">Block Address Area-2</td></tr>
<tr><td>516</td><td>Data Status Flag</td><td>524</td></tr>
<tr><td>517</td><td>Block Status Flag</td><td>525</td><td rowspan="3">ECC Area-1</td></tr>
<tr><td>518</td><td rowspan="2">Block Address Area-1</td><td>526</td></tr>
<tr><td>519</td><td>527</td></tr>
</table>

Based on the size of 1Sector, in case of 1M,2MByte SmartMedia(TM), the length of Page is 256Bytes and the size of Spare Array is 8 Bytes. For this Density, 2Pages must be taken for 1Sector and as a result, higher level 8Bytes of Inform among the above 16 Byte Inform must coincide with Spare of Even-page while lower level 8Bytes coincide with Spare of Odd-page. ECC Area-1 means the value of ECC Generation for 256Bytes in front among 512Byte Sectors while ECC Area-2 is the value of ECC Generation for 256Bytes behind. Block Address Area-1,2 has the same value and this is a key to find the Block of SmartMedia(TM). Therefore, Start has the value identical with the number of Sector per Block from the Block occupied by Master Boot Sector. In other words, this means that it increases regardless of the Sector or Cluster Size. This is the Regulation which is specified in Table 16.

Table 16 Block Address Configuration

The data in this area indicates address information on the conversion table to be consulted for block-logical-address to physical-address conversion

D7	D6	D5	D4	D3	D2	D1	D0	1,2MByte SM	SM with 4MBytes or more
0	0	0	1	0	BA9	BA8	BA7	262 bytes(even) 259 bytes(odd)	518,523 bytes
BA6	BA5	BA4	BA3	BA2	BA1	BA0	P	263 bytes(even) 260 bytes(odd)	519,524 bytes

BA9 ~ BA0 : Block Address(Values =0 through n,where n = maximum logical block count - 1)

P : Even Parity Bit

As indicated above, if we calculate Block Address based on actual Data, we obtain the value of 1001h,1002h,1004h,1007h in that order.

A look at Specification of Physical Format per SmartMedia Density shows the contents indicated in Table17.

Table 17 Physical Format Specification

Model	Formatted (Byte)	Unformatted (Byte)	Max Logical Blocks	Logical Block Size	Page Size (Byte)
1MByte	1,024,000	1,048,576	250	4,096 Byte	256 + 8
2MByte	2,048,000	2,097,152	500		
4MByte	4,096,000	4,194,304		8,192 Byte	512 (+16)
8MByte	8,192,000	8,388,608	1,000		
16MByte	16,384,000	16,777,216		16,384 Byte	
32MByte	32,768,000	33,554,432	2,000		
64MByte	65,536,000	67,108,864	4,000		
128MByte	131,072,000	134,217,728	8,000		

Note) 'Unformatted' does not include Array Area.

Since MASK ROM does not have any defective Block, there may be more Max Logical Blocks than SmartMedia(TM). Therefore,for the compatibility of Format (if MASK ROM is produced in the future), Max. size is limited to the unit of 1000, 2000 so that the same SmartMedia(TM) may be set. Accordingly, the memory space to accommodate this, as shown in Table 16, 1 ~ 16MByte can accommodate up to BA9, but it is not the case with 32 ~ 128MBytes. So arrangement of Data is defined as indicated in the following Table 18.

Table 18 Data Arrangement (in case of 32 ~ 128MBytes)

Zone No.	Physical Block No.	Application
0	0	CIS/Identify Drive Information Area
	1 ~ 1023	Data Area (Logical Block No. : 0 ~ 999)
1	0 ~ 1023	Data Area (Logical Block No. :1000 ~1999)
:	:	:
Final Zone	0 ~ 1023	Data Area (Logical Block No. : Final Zone No. x 1000 ~ Final Zone No. x 1000 + 999)

Note) 'CIS/Identify Drive Information' Area assigned to Physical Block of Zone 0.
If Physical Block 0 is Invalid Block, it is replaced by Valid Block within Physical Block within Zone 0. The size of Zone is 1000 Blocks.

In addition to 'CIS/Identify Drive Information' Area, Valid Block may be used as Data Area. Information of Invalid Block is designed to be written in Block Status Data of Table 15. Data of Invalid Block are defined as follows:

In case of Invalid Block,

1. 256 + 8 Byte Model includes "0" above 2 bits in the 6th Byte (Address 261) of all Even Pages within Block.
2. 512 +16 Byte Model includes "0" above 2bits in the 6th Byte (Address 517) within All Pages within Block.

3.4 ECC Algorithm(Hamming Code ECC)

ECC is Hamming Code ECC adopted at SSFDC Forum; this Algorithm is characterized by "2 bit Detect and 1bit Correct" Let us examine its theory more specifically.

3.4.1 Hamming Code ECC Theory

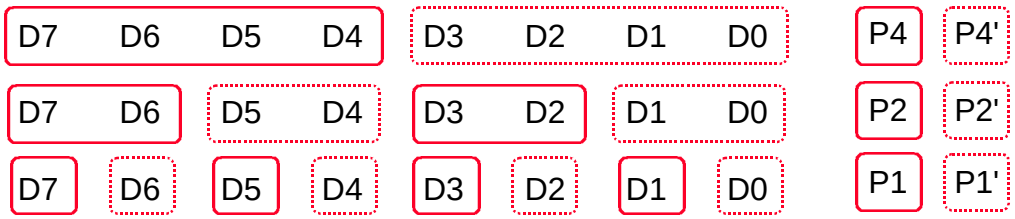
According to Hamming Code ECC Theory, SmartMedia(TM) generates 22bit ECC Code so that it may perform 1 bit Correction per 256Byte. In fact, it is a theory which can apply flexibly Hamming Code ECC per 1Byte or 8Byte, etc. In consideration of characteristics of SmartMedia(TM) Page, the calculation is to be done per 256Byte. A brief examination of its characteristics shows the following:

- ECC code consists of 3Bytes per 256Byte
  - Actually 22bit ECC code per 2048bit
  - 22bit ECC code = 16bit line parity + 6bit column parity
- Data bit assignment table with ECC code

Table 19 Detail Content of 256Byte

I/O 7	I/O 6		I/O 1	I/O 0	
D(00000000,111)	D(00000000,110)		D(00000000,001)	D(00000000,000)	1st Byte
D(00000001,111)	D(00000001,110)		D(00000001,001)	D(00000001,000)	2nd Byte
D(00000010,111)	D(00000010,110)	• • • •	D(00000010,001)	D(00000010,000)	2rd Byte
D(11111110,111)	D(11111110,110)		D(11111110,001)	D(11111110,000)	255th Byte
D(11111111,111)	D(11111111,110)		D(11111111,001)	D(11111111,000)	256th Byte

3.4.2 Example in case of 8bit



P4=D7(+)D6(+)D5(+)D4 P4`=D3(+)D2(+)D1(+)D0 ** (+) means XOR (Even parity)*

P2=D7(+)D6(+)D3(+)D2 P2`=D5(+)D4(+)D1(+)D0

P1=D7(+)D5(+)D3(+)D1 P1`=D6(+)D4(+)D2(+)D0

♦ For example (In case of 1 bit fail)

Original data : 1 0 1 0 1 0 1 0

P4=1(+)0(+)1(+)0=0, P4`=1(+)0(+)1(+)0=0
P2=1(+)0(+)1(+)0=0, P2`=1(+)0(+)1(+)0=0
P1=1(+)1(+)1(+)1=0, P1`=0(+)0(+)0(+)0=0

Changed data : 1 0 1 **1** 1 0 1 0

P4=1(+)0(+)1(+)**1**=**1**, P4`=1(+)0(+)1(+)0=0
P2=1(+)0(+)1(+)0=0, P2`=1(+)**1**(+)1(+)0=**1**
P1=1(+)1(+)1(+)1=0, P1`=0(+)**1**(+)0(+)0=**1**

In here, fail bit locate is column address offset *(P4,P2,P1=100=I/O4)*

Figure 9 Parity Generation (In case of 8bit input)

3.4.3 ECC Generation Method for 256Byte (in case of SmartMedia(TM))

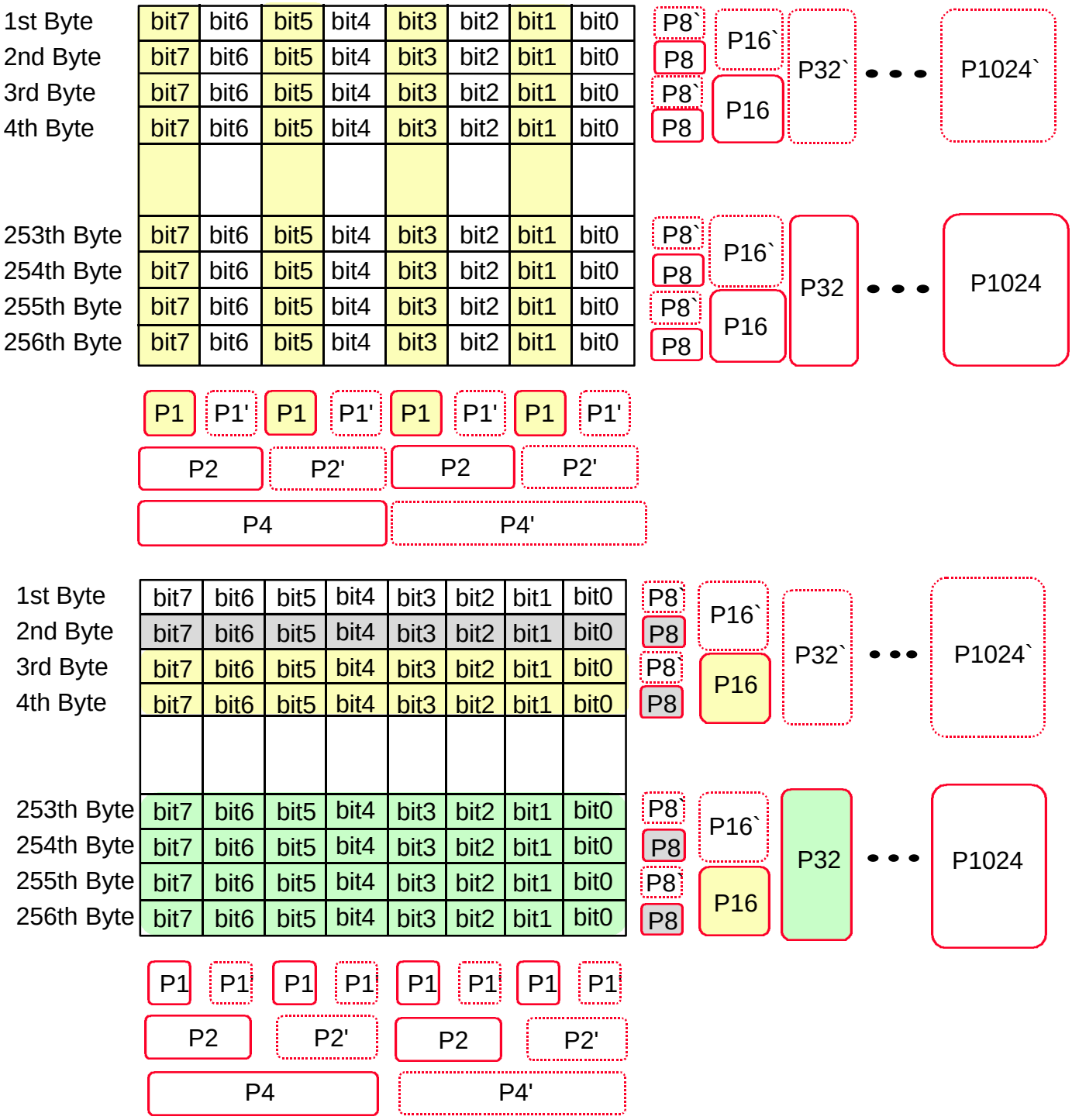


Figure 10 Parity Generation (In case of 256 Byte input)

P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
P8 = bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)bit2(+)bit1(+)bit0(+)P8

3.4.4. ECC Detect Flow Chart

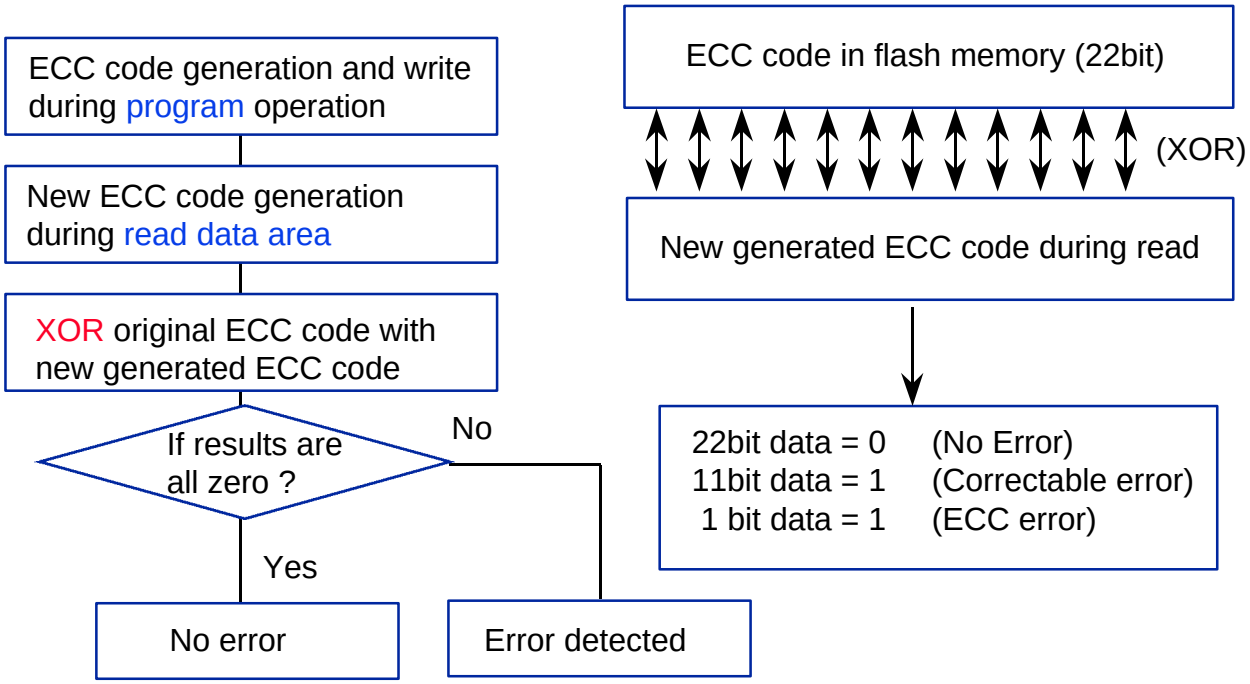


Figure 11 Error Detection Sequence

3.4.5 Hamming Code ECC Applied to SmartMedia

The above Hamming Code ECC is applied to SmartMedia by 3Bytes as follows:

Table 20 ECC code assignment table

I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
P64	P64`	**P32**	P32`	**P16**	P16`	**P8**	P8`
P1024	P1024`	**P512**	P512`	**P256**	P256`	**P128**	P128`
P4	P4`	**P2**	P2`	**P1**	P1`	0	0

P8 ~ P1024 : Line parity
P1 ~ P4 : Column parity

Fail bit address offset
=> **(P1024,P512,P256,P128,P64,P32,P16,P8, P4,P2,P1)**

**** When this ECC Code is applied to SmartMedia(TM), in consideration of characteristics of SmartMedia(TM), Write in Spare Array by taking Inverse. Please keep this fact in mind.**

As mentioned above, we have examined SmartMedia(TM) Optimized Format and its operating method through examples. Of course, the above contents alone may be insufficient to ensure a satisfactory understanding; it seems that the review accompanied by the actual application and practice based on the above contents will ensure full understanding. Next SmartMedia(TM) Software Driver with Hardware showing an overall operation including the above contents is introduced. We hope this will be of some help to source analysis.

4.0 Hardware Interface Configuration

4.1 ISA Demo Board

Figure 12 is the Block Diagram by which showing that SEC Nand Flash Memory can be controlled through ISA interface via PC.. 1M ~128MByte NAND Flash , SmartMedia(TM) can be controlled through this ISA Demo Board.

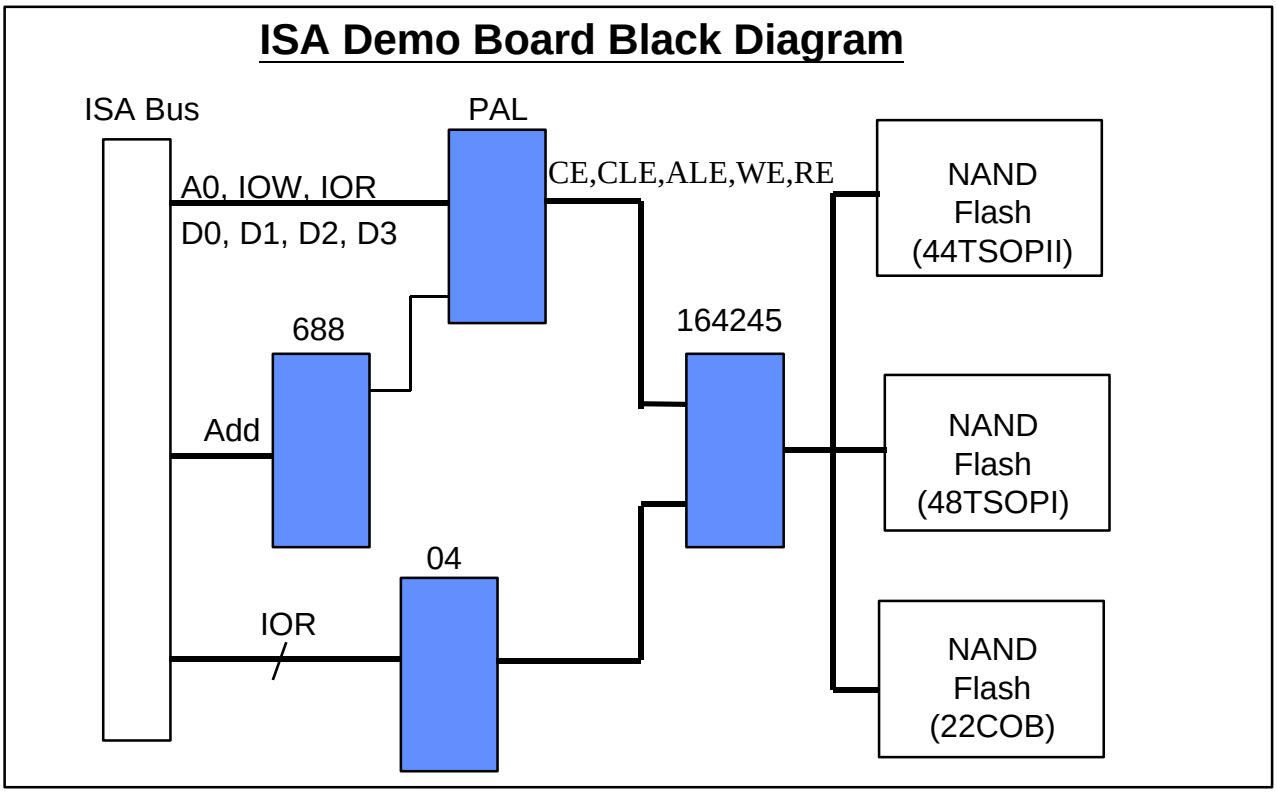


Figure 12 ISA Demo Board Block Diagram

4.2 EPP Parallel Port Demo Board

Figure 13 is the Block Diagram by which showing that SEC Nand Flash Memory can be controlled through ISA interface via PC.. 1M ~128MByte NAND Flash , SmartMedia(TM) can be controlled through this EPP Parallel Port Demo Board.

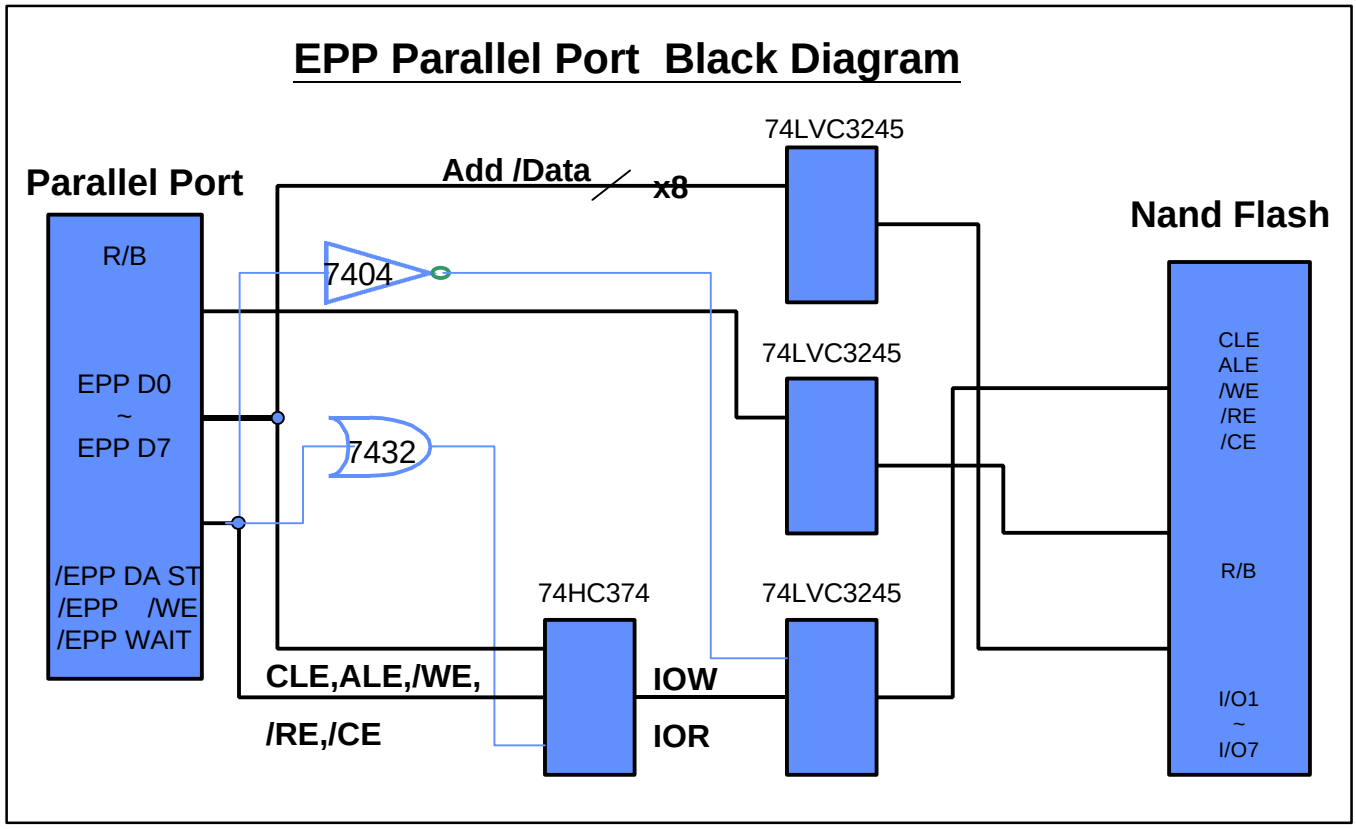


Figure 13 EPP Parallel Port Board Block Diagram

5.0 Software Driver for SmartMedia(TM)

5.1 Software Introduction

Software driver has been made available by using FAT File System of Optimized Format based on SSFDC Forum and consists of functions such as File Copy in optional Directory, optional Directory Create and Delete, etc. by focusing on File Input and Output and of additionally necessary functions. A full understanding of the contents described in the foregoing chapters up to the chapter4 is required before Software coding is reviewed. At the same time, when Software driver is used, it is desirable to use it by Modify to smoothly obtain the functions necessary for the System enabling Interface Protocol without fail after the review.

- Caution

1. For Selecting Hardware option,you must modify SM_TARGET by SM_ISA or SM_EPP in smio.h.
2. Hardware is only possible with available Hardware ECC Demo Board.

5.2 Software Main Functions

This Software enables the following functions:

1) File Copy Function
   -. Copy data from HDD to SmartMedia(TM)
   -. Copy data from SmartMedia(TM) to HDD
   -. Delete Single file
2) Directory Function
   -. Display Directory information
   -. Make Directory
   -. Delete Directory
3) Label Create and Update Function
4) SmartMedia(TM) Optimized Format Function
5) SmartMedia(TM) Main Parameter Display Function
6) Read SmartMedia(TM) Function

5.2.1 File Copy Function

1) Copy data from HDD to SmartMedia(TM)

Function to copy Optional File in HDD to SmartMedia(TM); it performs the following functions:

(1) Initial_Setup Function : It calculates Logical Address after ID Check, identifies PBR from the contents of MBR, searches Spare Data of Physical Block by referring to the contents of BPB at PBR and thus sets up Lookup Table with Entire Physical Block Address and Inform of Status. It is an initial stage to set up Parameter for reference in the following Flow; it is a function as a first step to set up the configuration of Format and Sector Location FAT Inform, etc. before certain function is performed. (Function which is entered at all times before the Start of most main Functions later) This flow is shown in the following Flow Chart:

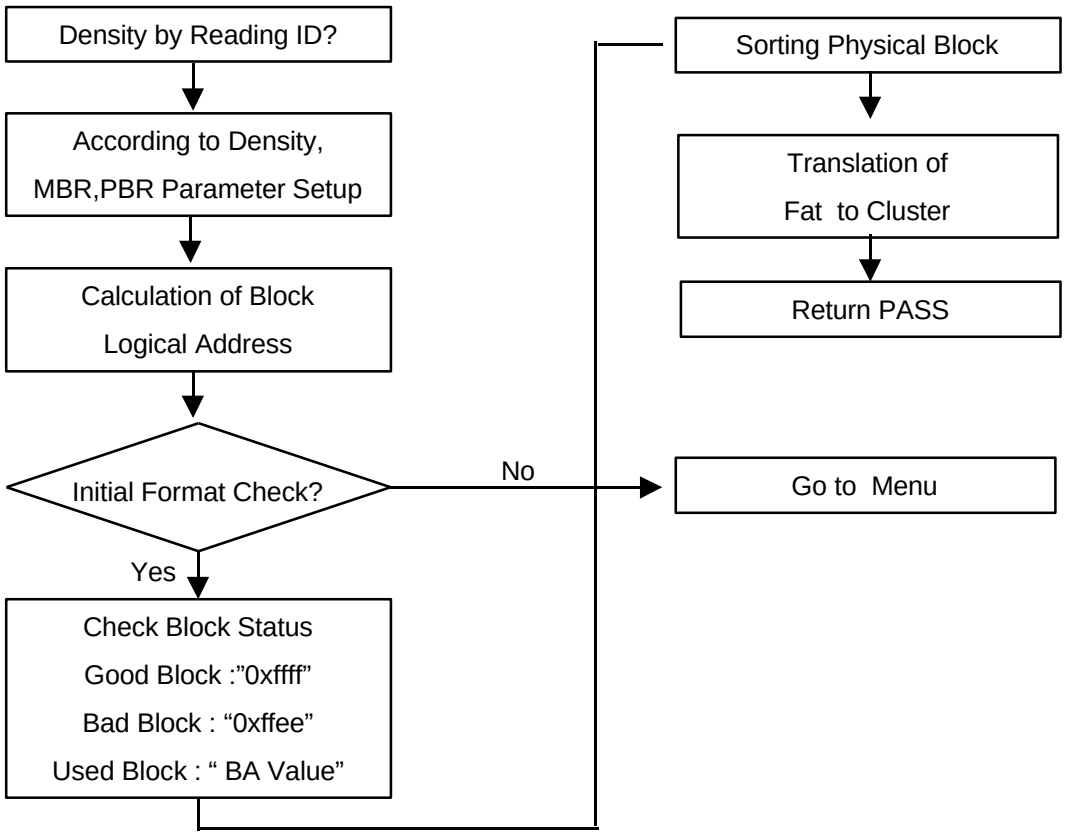


Figure 14 Initial Setup Flow Chart

(2) From_HDD to_SM Function : It receives Input of File to Copy and Input of Directory and File Name to Write from User and first judges if it is File size permitting Write or if there is an Area where it is possible to Write, verifies Directory selected by User and searches Start Cluster of such Directory to determine FAT Cluster which can be used in FAT Information before writing File Data. And it updates File Inform 32 Bytes in Directory Cluster already verified and lastly updates FAT Data. When Directory and FAT are Updated, it first searches Usable Physical Valid Block from Lookup Table, newly modifies Block Address of this Block and then refers to Data to Update and moves Data to be updated to the Block newly taken while reading Data from the Entire Block. After all data are moved, it erases the entire Block and marks Status as Usable Block.

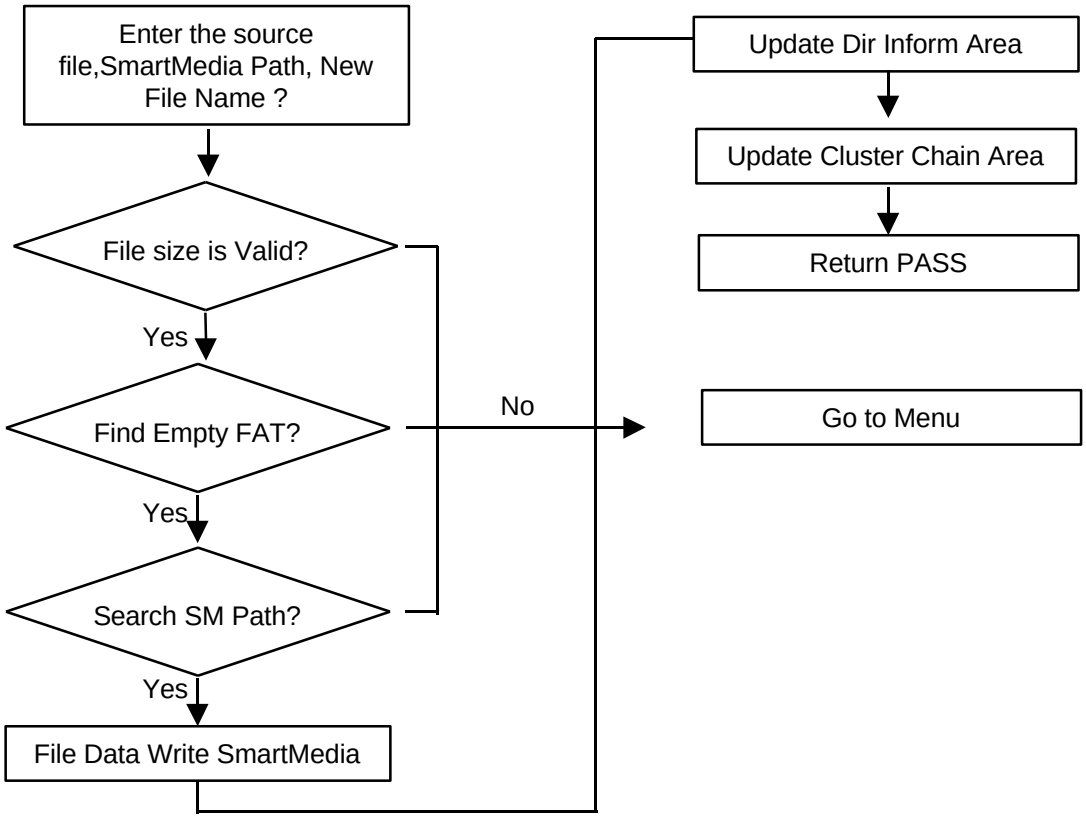


Figure 15 HDD to SM Copy Flow Chart

*Example of Lookup Table (Reference 3)

It requires much information to create the basically necessary Cluster information and information to find in the searched Directory or to write in File. First of all. Initial Setup Function is to perform the function necessary to discover the current status of the SmartMedia(TM) and here it reads necessary information from Partition Boot Record starting from a stage of Format. (Search_Master_Partition Function Executed) After the unit of Sector and units of Cluster are determined here, Block_Status_Check Function to monitor status of each Block of current SmartMedica is performed to check status of the zone's entire Block. A parameter to monitor such status is Block a[Physical block number] arrangement and this Value is defined by this Software as follows:

- Block_a Parameter Definition

Used Valid Block is block_a[Physical block number] = bl_addr(Block Address value)

Invalid Block is block_a[Physical block number] = 0xffee(Invalid Mark is defined as '0xffee')

CIS Block is block_a[Physical block number] = 0 (Actual Block Addess Value is '0x0000'.)

Unused Valid Block[Physical block number] = 0xffff

And also block_a parameter indicating invalid Block as follows(0xffee).

Block_Status_Check function searches the entire Block and thereafter determines the value of sort block Parameter for the entire Block of SmartMedia at Sort_Physical_Block Function. The value of sort_block parameter detects Block of SmartMedia equal to the value of Block Address. Let us explain the contents of Figure16.

Block_a[0]= 0Xffee	Sort_block[0]= 2	Block Address
Block_a[1]= 0x00	Sort_block[1]= 4	Value Ordering
Block_a[2]= 0x1001	Sort_block[2]= 9	0x1001
Block_a[3]= 0xffff	Sort_block[3]= 8	0x1002
Block_a[4]= 0x1002	Sort_block[4]= 0	0x1004
Block_a[5]= 0xffff	Sort_block[5]= 0	0x1007
Block_a[6]= 0Xffff	Sort_block[6]= 0	0x1008
Block_a[7]= 0xffee	Sort_block[7]= 0	.
Block_a[8]= 0X1007	Sort_block[8]= 0	.
Block_a[9]= 0x1004	Sort_block[9]= 0	.
.	.	.
.	.	.

Figure 16 Example of Look Up Table

To help the understanding of Look Up Table serving as reference whenever certain function is performed, elaboration on this is made as follows: It is important to find first of all where in SmartMedia(TM) there is Block equivalent to '0x1001' of the first Block Address (because MBR is fixed at 0 Sector.) in order to discover the contents of 512 Bytes of Master Boot Record. As is seen in Block_a[2]=0x1001, Block to be searched in SmartMedia(TM) is the 3rdPhysical Block (if 0 is deemed as the first Block), This means that the first 512 Bytes are equivalent to Master Boot Record. In this way, we can see that the Block equivalent to the second Block Address Value'0x1002' is Block 4 (5th Block). As mentioned above, we can easily find the desired Data by checking Spare Inform of SmartMedia(TM) per Function from Look Up Table for initialization or using the method to update Look Up Table each time. It is essential to well determine the concept of how to compose the Look Up Table and it is possible to perform the job of writing new Data by searching location of Block, status of Block and Usable, Valid Block based on this value.

2) Copy data from SmartMedia(TM) to HDD Function

Function to copy optional File in SmartMedia to HDD; it performs the following main Func.

(1)Initial_Setup Function (Explanation is dispensed with as this Function is commonly performed by all functions in the future.)

(2) From_SM_to_HDD Function: Similar function to Write File from SmartMeida(TM) to HDD in opposition to Function of '1) Copy data from SmartMedia(TM) to HDD' As Flow is similar, see above ' 1)Copy data from SmartMedia(TM) to HDD'.

3) Delete Single File Function

Function to Delete optional File in SmartMedia(TM); it performs the following Functions:

(1) Initial_Setup Function

(2) Single_File_Delete Function

It searches File to delete and if such File exists, it updates Dir Inform of this File and FAT Data. It does not erase actual File data but the method to write after erasing Block during next action is used.(It doesn't cause any problem to operation if function to delete this area is used.)

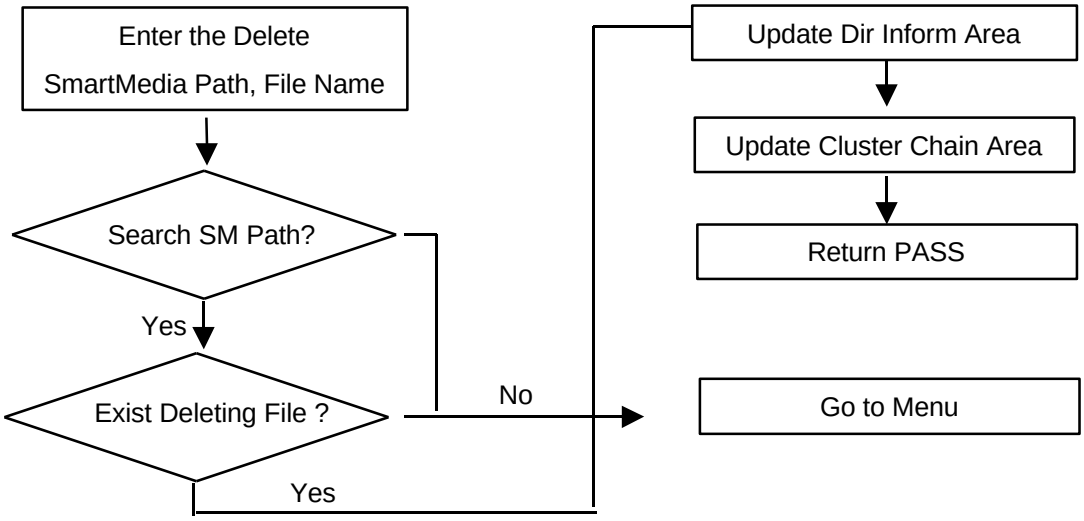


Figure 17 Single File Delete Flow Chart

5.2.2 Directory Menu Function

This Menu is a function related to Directory and includes the function to display a List of File contained in certain Directory, the function to create Directory, and the function to delete Directory, etc.

1) Display Directory Function

(1) Initial_Setup Function

(2) Directory_Display Function

This Function displays Information on File. Sub-Dir contained in current Directory on Monitor similar to 'Dir' Function provided by DOS.

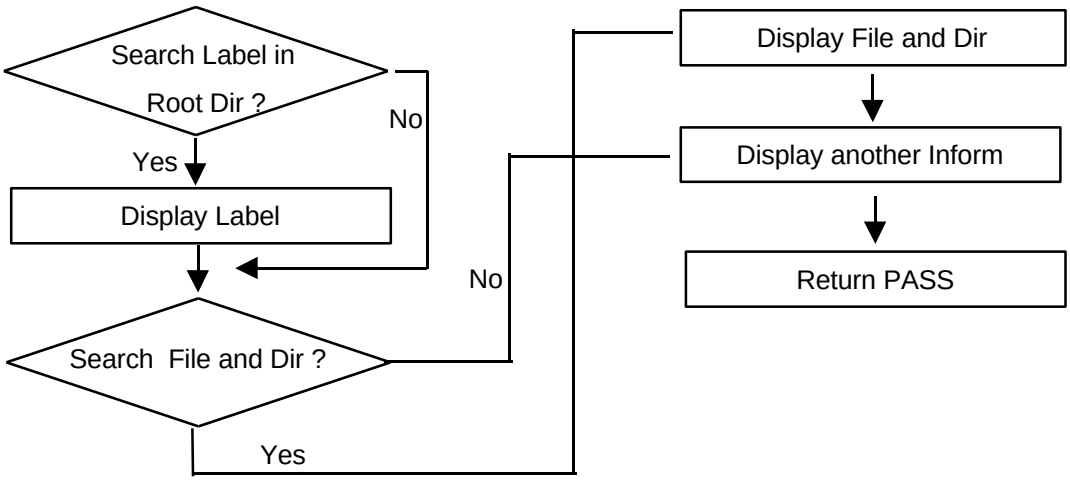


Figure 18 Directory Display Flow Chart

2) Make Directory Function

(1) Initial_Setup Function

(2) Make_Dir Function

Function to generate New Sub-Directory in certain Path

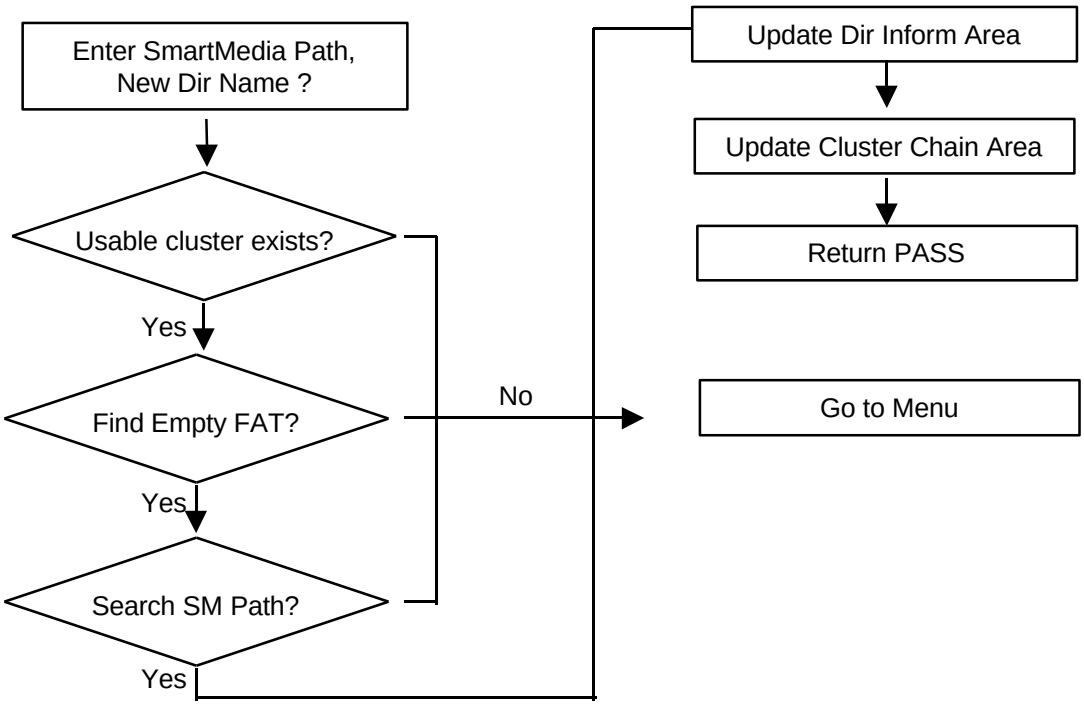


Figure 19 Make Directory Flow Chart

3) Delete Directory Function

(1) Initial_Setup Function

(2) Single_Dir_Delete Function

Function to Delete Sub-Directory in certain Path.

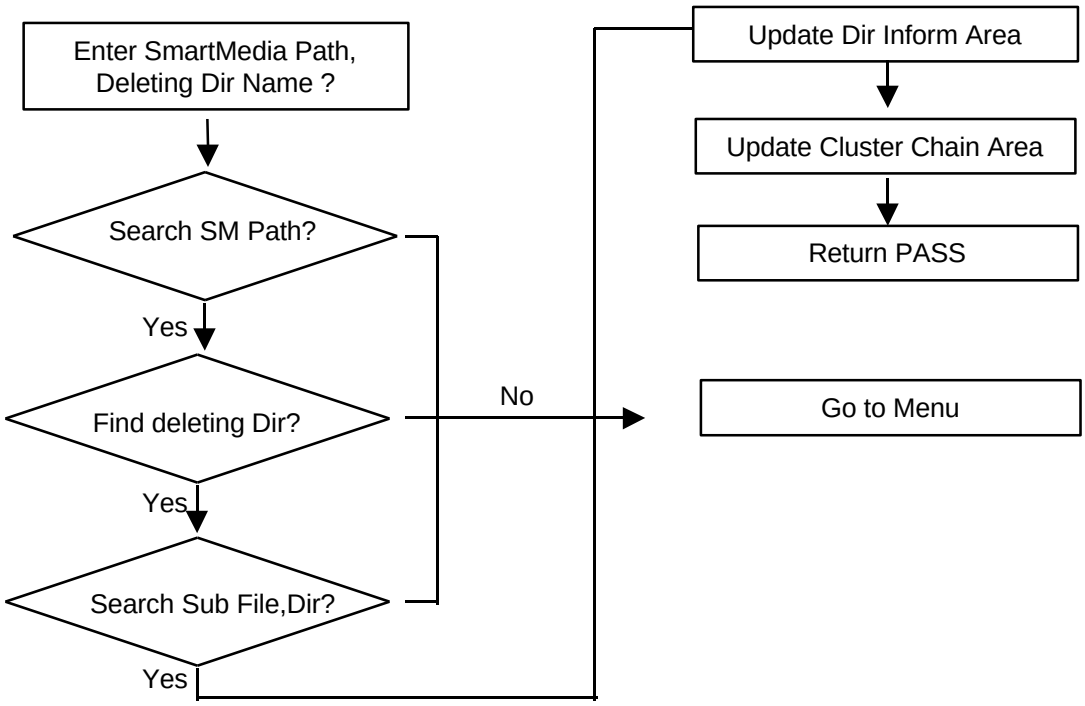


Figure 20 Delete Directory Flow Chart

If there is File or Sub-Directory to Delete, this File and Dir must be deleted first. Therefore, it is supposed to go to Menu with the Message of such contents.

* Data Updating Flow Algorithm (Reference 4)

In case of Update Directory Data and Update FAT Data requiring Flow for Update, the following Algorithm is used in consideration of characteristics of SmartMedia.

Data Updating Flow

Step1 : Data to Update is occurred.

Step2 : Valid New Block is taken to be newly connected to block_a,Sort_block.
(Value of Parameter explained in Figure 28 Look Up Table is newly connected.)

Step3 : Page Data not required to be Updated in Block having existing Data is read and written after being moved to the newly connected Block. And only in case of Writing Data to be Updated, Data is modified to Write.

Step4 : Block having existing Data is erased. (Unused Valid Block is made.)

5.2.3 Label Menu Function

Menu to create and update Label; Label is recognized only if it is written in Root Directory without fail. It is the Function supported by DOS and it is presented here to be of some help to User as reference.

(1) Initial_Setup Function

(2) Update_Label Function

It occupies only 32 Byte Inform in Root Directory and can be Updated. The important characteristics of Data is that up to 11 Characters can be recognized by using File Name and Extension Area among 32 Byte Inform and Attribution is defined as 08h. (See Table 10.) A look at the Flow of Update_Label Function shows the following:

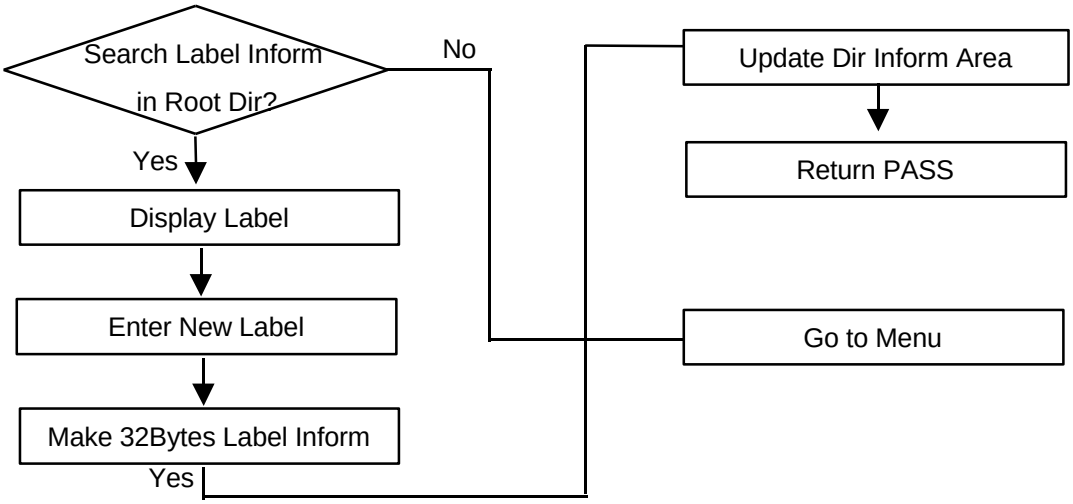


Figure 21 Update Label Flow Chart

* Basic Function needed when File(Dir) is Created,Deleted (Reference 5)

When File is Created,

(1) File Content Write,(2) Update Dir Inform , (3) Update FAT Inform, etc....

When File is Deleted,

(1) Update Dir Inform , (2) Update FAT Inform, etc.…

When Dir is Created,

(1) Update Dir Inform , (2) Update FAT Inform, etc....

When Dir is Deleted,

(1) Update Dir Inform , (2) Update FAT Inform, etc.…

As is clear from the above, Flow varies slightly depending on what Function is performed, but it consists of 3 basic contents and it is essential to perform these.

(1) File Content Write is the content of actual File and must be performed during Cycle.

(2) Update Dir Inform is an essential Routine which must use 32 Byte Inform when certain Action takes place.

(3) Connection of Cluster Chain is newly modified for application through Update FAT Inform and performance is completed for the New Action. An examination bearing this point in mind will be of help to the understanding.

5.2.4 SmartMedia(TM) Format Menu

This Function is to Write Optimized Format defined per Density through ID Check of SmartMedia(TM) Optimized Format. It is an operation to initialize SmartMedia(TM) and performs the following Functions.

1)Read_ID Function

2)Erase_With_Bad Function : If the 6th Bye is not '0xff', it recognizes it as Invalid Block and erases Data of Valid Block.

3)Optimized_Parameter Function: It defines the value of MBR,PBR according to Density.

4)Bad_Block_Mark Function : It does the Marking by treating the First Block as the Invalid Block based on SmartMedia Forum.

5)Master_Partition_Cal Function: After it finds the location of PBR from MBR Data, it discovers main Parameter Values from PBR and then sets it up. Based on this, 4 types of factors will be used for SmartMedia in the future.

6) Cal_BLA Function : It calculates BLA Value.

7) CIS_Inform : It Writes CIS Data in the 2nd Block.(PCMCIA Card Information)

As shown in Figure 16, the above Process is the function write in Block the Data necessary for writing the First Block : Invalid Block, Second Block : CIS Block, Third Block or more : Format Block.

It is to be specially noted that this is a stage where it is sequentially Write Block as much as physically needed since the concept of Sector,Cluster is not yet familiar to you. Therefore, it is a stage of calculating how far Sector having PBR is from MBR and then implementing sequentially FAT Data, Root Directory at the next location while Algorithm creating Look Up Table is also made available through Initial-Setup when other action is performed in the future. If there is an Invalid Block at the location of Physical Block, it is the routine to Write in the next Block.

* Relation between Page and Sector (Reference 6)

The length of 1Page is 512Byte +16Byte(Spare Array) starting from 4MByte SmartMedia (TM). This coincides with the size of 1Sector of 512Bytes. (in case of 2MB, 2Pages are considered as 1Sector.) Optimized Format used as 'Standard ' by SSFDC Forum ensures the convenience of operation by aligning the size of 1Cluster with the size of 1 Block of SmartMedia(TM). Similarly 16 Sector (Page) from 4MB to 8MB is 1 Block and thus 1 Cluster is set as 16 Sector in this Density. For 16MB or more, 32Sector is 1Cluster. This alignment of the size of 1 Block with 1Cluster is the Optimized Format.

Accordingly, the size of 1Sector for 1MB,2MB is equal to 2 Page and therefore the different length of Page depending on Density within Sector_Read,Sector_Write Function designed to accommodate all of these makes it necessary to distinguish the methods of Read from SmartMedia. As basics of Real Time Mapping, one has made the Routine for handling Error bit according to the degree of its occurrence by checking whether there is any Error bit from the time of Page_Read based on ECC Data of Spare At the time of Sector_Write, one determines whether there is Error by the Routine of reading again and comparing not only Status Check but Data one wanted to Write. Real Time Mapping is smoothly ensured by solving the problem of Error in an early stage as seen above.

5.2.5 Information on SmartMedia(TM) Menu

Function of Displaying Value of various Parameters of SmartMedia(TM).

1)Format Parameter Information Display Function

(1)Initial_Setup Function

(2)Information_Display Function : Function of displaying the important Parameters as follows:

-.Sector Per Block -.Total Cylinder

-.Total Hidden Sector -.Bytes Per Sector

-.Sectors Per Cluster -.Reserved Sector

-.Number of FATs -.Root Dir Entries

-.Sector Per Root -.Media ID Byte

-.Sector Per FAT -.Sector Per Track

-.Numbers of Head -.Location of First FAT Sector

-.Location of Second FAT Sector -.Location of Root Sector

-.Logical Total Sector -.Cluster Size

-.Total Sector -.Total Cluster

It displays after judging whether Optimized Format per Density is correct by comparing the above values with the Checksum Value already calculated as Checksum Value per Density. Based on this, one can determine whether the current Format aligns with Optimized Format.

2)FAT Data Status Display Function

Function to help analyze and understand the current status of FAT by displaying FAT Data as Cluster Value.

5.2.6 Read SmartMedia(TM) Menu Physically

Function by which one can view physically the status of SmartMedia in Page, in Block and in whole and Function to analyze the status of SmartMedia Data actually written as was executed by User.

1) Page Read Function : Function to display Page Address by having it input.
2) Block Read Function : Function to display the desired Block Address after having it input.
3) All Block Read Function : Function to display part of Main Array and Spare Array by adjusting them to the size of the screen to view the outline of the whole.

Each Sub-Menu is structured to enable various functions to be continuously performed within Program without leaving it by application of function to return to Previous Menu.

* Limitations of this software (Reference 7)

An examination of the problems involved in the System requiring additional functions due to the constraints for support of Soft driver posed by the diverse types of System shows that the following points must be taken into consideration.

1) Expandability of Sub Directory must be taken into consideration.
This software driver has created only 1Cluster for Directory and Search has been done under the assumption that when necessary functions are performed, Directory Size is only 1Cluster.The size of Root Directory is usually fixed at 256 Entries, namely, 16 Sector (if the size of 1 Sector is 512 Bytes) due to Parameter referred to as Root_Dir _Entries and this is constant. However, the size of Sub Directory is expandable by being allocated Cluster depending on the need of Data to be additionally input, which is not backed up by this Software.

2) Usage for FAT Back-Up was not taken into consideration.
FAT Data is the most important Data which need to be protected in the best way. Therefore,FAT2 Area is allocated to serve as reference for Reading ,Writing Error. However this Software driver failed to show a lot of utilization in this area.

3) All File Delete,Copy Function was not taken into consideration.

This Software driver is composed to perform the function of Single File or Directory. We think that the Function of handling a variety of File or Directory can be expanded on an extension line of these functions.

4) When System is applied, an efficient control of Buffer is required.

Soft driver is characterized by the Source created to help System Design-In with respect to utilization of SmartMedia(TM) by User making use of Optimized Format which is not Soft driver created to develop Specific System. For this reason, sufficient notes are used and reduction of Buffer used was not taken into consideration. A look at the size of used Buffer used shows the following:

Global Variable Size : About 22.8KB

ROM Size is about 121 KB.(Including sample application)

In consideration of performance of necessary functions per System, it is desirable to pay enough attention to the ROM Size.

5) Expandability of File Name provided by Win95 was not taken into consideration.

While compatibility like expandability of File Name provided by Win 95 is not in place, User must consider the necessary matters.That concept has not yet been fully taken into consideration and, if necessary, due consideration shall be given to this aspect in the next Version.

6. Conclusion

the understanding of Optimized Format and Hardware,Software driver embodying this will hopefully provide assistance in utilizing SmartMedia Operating System. Though it leaves something to be desired, we sincerely hope that this will be conducive to Design-In and we will make up for shortfalls through continuous Version-Up.

Appendix A

Difference according to density

Difference between 16MB and 32~128MB (1)

Data Array Configuration (In case 32 ~ 128MByte)

Zone	Physical Block	Description
0	0	CIS/Identify Drive Information Area
	1 ~ 1023	Data Area (Logical Block : 0 ~ 999)
1	0 ~ 1023	Data Area (Logical Block :1000 ~1999)
:	:	:
Ã Ø ¾Zone	0 ~ 1023	Data Area (Logical Block : Zone x 1000 + 999)

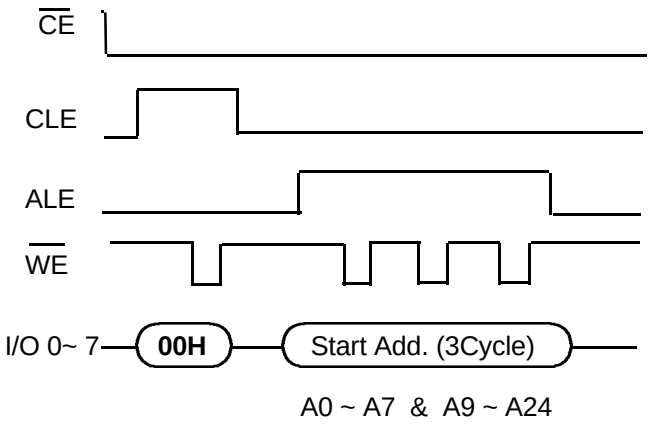
* CIS/Identify Drive Information Area ==>Zone 0

Size of zone is 1000 Block.

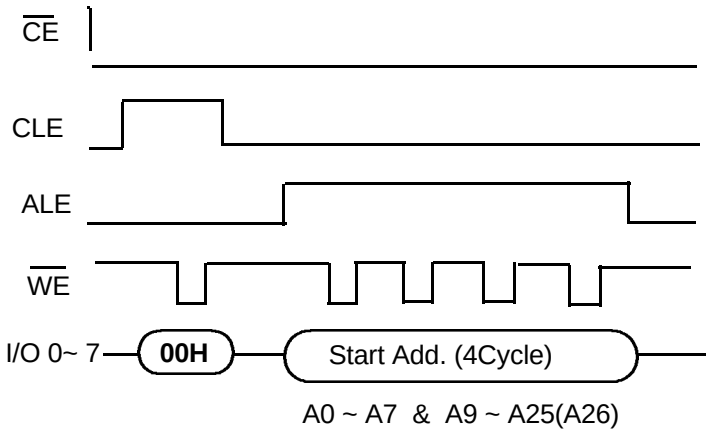
Difference between 32MB and 64,128MB(2)

- 64,128MByte SmartMedia(TM) needs fourth Address Cycle

1.Timing Diagram(In case 32MByte)



2.Timing Diagram(In case 64,128MByte)



Address Configuration

Add Cycle	D7	D6	D5	D4	D3	D2	D1	D0
1st Cycle	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
2 nd Cycle	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
3 rd Cycle	PA15	PA14	PA13	PA12	PA11	PA10	PA9	PA8
4 th Cycle	PA23	PA22	PA21	PA20	PA19	PA18	PA17	PA16

CA0 ~ CA7 : Column Address PA0 ~ PA23 : Page Address

Model	Valid Page Address	'L' Level(ADD3)	'L' Level(ADD4)
1MByte	PA0 ~ PA11	PA12 ~ PA 15	
2MByte	PA0 ~ PA12	PA13 ~ PA 15	
4MByte	PA0 ~ PA12	PA13 ~ PA 15	
8MByte	PA0 ~ PA13	PA14, PA 15	
16MByte	PA0 ~ PA14	PA15	
32MByte	PA0 ~ PA15	-	
64MByte	PA0 ~ PA16		PA17 ~ PA23
128MByte	PA0 ~ PA17		PA18 ~ PA23

Difference between 64MB and128MB(3)

12bit FAT
Useful Cluster Number = 2^{12} = 4096
(Approximately 4000 Cluster Count Available)

16bit FAT Operation
Useful Cluster Number = 2^{16} = 65536
(Approximately 64000 Cluster Count Available)

12,16bit FAT Operation Configuration

12-bit FAT	16-bit FAT	Description
000h	0000h	Unused Cluster
001h	0001h	Reserved
002h ~ FEFh	0002h ~ FFEFh	Next Cluster Number in the chain
FF0h ~ FF6h	FFF0h ~ FFF6h	Reserved
FF7h	FFF7h	Defective Cluster
FF8h ~ FFFh	FFF8h ~ FFFFh	Last Cluster in the chain

1~ 64MB(250 ~ 4,000 Cluster Chain Needs) : 12bit FAT Operation
128MB(8,000 Cluster Chain Needs) : 16bit FAT Operation

Reference Literatures

1. SmartMedia(TM) Specification

- SSFDC Format Technical Group

2. ADVANCED MS-DOS

- Microsoft Press, Ray Duncan

3. PC Card Standard(March 1997)

- PCMCIA/JEIDA

4. SEC SmartMedia(TM) Data Book(Flash Memory,1999)

- SAMSUNG Electronics