□Maple Bus 1.0□
Function Type Specifications
FT_1:Storage Function

Revision 0.80

Created by

Sega Enterprises

No. 2 Development Group, CS Development and Production Headquarters

Revision

2/19/1998 0.70	Preview Version
3/31/1998 0.80	First Edition

Contents

1 Overview of Storage Function.....
1.1 Definition of Storage Function.....
1.2 Storage Function Features and Restrictions.....
2 Device IDs.....
2.1 Device ID Configuration.....
2.2 Function Types.....
2.3 Function Definition Block.....
3 Storage Specifications.....
3.1 Block Configuration.....
3.2 Management Area.....
3.2.1 System Area.....
3.2.2 FAT (File Allocation Table) Area.....
3.2.3 File Information.....
3.3 Data Area.....
3.4 Save Area.....
3.5 Format Processing.....
4 Commands.....
4.1 Control Commands.....
4.1.1 Get_Media_Info.....
4.1.2 Block_Read.....
4.1.3 Block_Write.....
4.1.4 Get_Last_Error.....
4.1.5 Data Transfer.....
4.1.6 Device Reply.....
4.1.7 Device Request.....
4.1.8 All Status Request.....
4.1.9 Device Reset.....
4.1.10 Device Kill.....
4.1.11 Device Status.....
4.1.12 Device All Status.....
4.2 Error Commands.....
4.2.1 Function Type Unknown.....
4.2.2 Command Unknown.....
4.2.3 Transmit Again.....
4.2.4 File Error.....
5 Protocol Flow.....
5.1 Get_Media_Info Processing Flow.....
5.2 Block_Read Processing Flow.....
5.3 Block_Write Processing Flow.....
5.4 Get_Last_Error Processing Flow.....
6 Postscript.....

1 Overview of Storage Function

1.1Definition of Storage Function

The Storage Function mounts or permits the mounting of a device (media) in which data (such as parameters or programs) are stored, and is able to freely write and read that data.
The Storage Function must satisfy the conditions listed below.□

1) The Storage Function must be able to read and/or write data to a device (media) in which data can be stored. There are no restrictions concerning its external appearance.
2) The Storage Function must conform with the Maple Bus 1.0 Standard Specifications.

The functions of such a peripheral device include storing data related to the progress of a game, and providing game expansion data by means of optional ROM.

1.2Storage Function Features and Restrictions

The features and restrictions of the Storage Function are listed below.

1) The Storage Function can use any storage device (media) without making any distinctions, including ROM, RAM, EEPROM, flash memory, and magnetic recording media. The Storage Function is also compatible with removable media.
2) The Storage Function can support a maximum of 256 partitions.
3) The size of one partition can range from 256 to 65,536 blocks.
4) The size of one block can range from 512 to 8192 bytes.
5) Based on the above, the maximum size that can be managed in one partition is 512MB (when 1 block = 8192 bytes), and the maximum size that can be managed by the Storage Function is 128GB.
However, after considering management capabilities from a practical standpoint, one block is fixed at 512MB, and the maximum size of one partition is 32MB.
6) File names can be up to 12 characters long. (Normal size only.)
7) Only a root directory is supported. No sub-directories are supported.

2 Device IDs

Device IDs conform with the device ID stipulations in the Maple Bus 1.0 Standard Specifications. The notation used is the memory image in the host.

2.1Device ID Configuration

In Maple Bus 1.0, device IDs are configured as shown below.

bit	7	6	5	4	3	2	1	0
1st Data	FT_{31}	FT_{30}	FT_{29}	FT_{28}	FT_{27}	FT_{26}	FT_{25}	FT_{24}
2nd Data	FT_{23}	FT_{22}	FT_{21}	FT_{20}	FT_{19}	FT_{18}	FT_{17}	FT_{16}
3rd Data	FT_{15}	FT_{14}	FT_{13}	FT_{12}	FT_{11}	FT_{10}	FT_9	FT_8
4th Data	FT_7	FT_6	FT_5	FT_4	FT_3	FT_2	FT_1	FT_0
5th Data	$FD1_{31}$	$FD1_{30}$	$FD1_{29}$	$FD1_{28}$	$FD1_{27}$	$FD1_{26}$	$FD1_{25}$	$FD1_{24}$
6th Data	$FD1_{23}$	$FD1_{22}$	$FD1_{21}$	$FD1_{20}$	$FD1_{19}$	$FD1_{18}$	$FD1_{17}$	$FD1_{16}$
7th Data	$FD1_{15}$	$FD1_{14}$	$FD1_{13}$	$FD1_{12}$	$FD1_{11}$	$FD1_{10}$	$FD1_9$	$FD1_8$
8th Data	$FD1_7$	$FD1_6$	$FD1_5$	$FD1_4$	$FD1_3$	$FD1_2$	$FD1_1$	$FD1_0$
9th Data	$FD2_{31}$	$FD2_{30}$	$FD2_{29}$	$FD2_{28}$	$FD2_{27}$	$FD2_{26}$	$FD2_{25}$	$FD2_{24}$
10th Data	$FD2_{23}$	$FD2_{22}$	$FD2_{21}$	$FD2_{20}$	$FD2_{19}$	$FD2_{18}$	$FD2_{17}$	$FD2_{16}$
11th Data	$FD2_{15}$	$FD2_{14}$	$FD2_{13}$	$FD2_{12}$	$FD2_{11}$	$FD2_{10}$	$FD2_9$	$FD2_8$
12th Data	$FD2_7$	$FD2_6$	$FD2_5$	$FD2_4$	$FD2_3$	$FD2_2$	$FD2_1$	$FD2_0$
13th Data	$FD3_{31}$	$FD3_{30}$	$FD3_{29}$	$FD3_{28}$	$FD3_{27}$	$FD3_{26}$	$FD3_{25}$	$FD3_{24}$
14th Data	$FD3_{23}$	$FD3_{22}$	$FD3_{21}$	$FD3_{20}$	$FD3_{19}$	$FD3_{18}$	$FD3_{17}$	$FD3_{16}$
15th Data	$FD3_{15}$	$FD3_{14}$	$FD3_{13}$	$FD3_{12}$	$FD3_{11}$	$FD3_{10}$	$FD3_9$	$FD3_8$
16th Data	$FD3_7$	$FD3_6$	$FD3_5$	$FD3_4$	$FD3_3$	$FD3_2$	$FD3_1$	$FD3_0$

Fig 2.1□Device IDs

FT: Indicates the peripheral function type.
FD1: First function definition block.
FD2: Second function definition block.
FD3: Third function definition block.

1) FT_{31} to FT_0: Function type
Indicates the function types supported by the peripheral device.
There are a total of 32 function types defined.

2) FD_{31} to FD_0: Function definition block
This block defines the individual elements that comprise a function.
(Up to three different functions can be implemented for a single peripheral device.)

2.2Function Types

This section describes the function type FT within the device ID.
The Storage Function function type is defined by FT_1 = 1.

bit	7	6	5	4	3	2	1	0
1st Data	FT_{31}	FT_{30}	FT_{29}	FT_{28}	FT_{27}	FT_{26}	FT_{25}	FT_{24}
2nd Data	FT_{23}	FT_{22}	FT_{21}	FT_{20}	FT_{19}	FT_{18}	FT_{17}	FT_{16}
3rd Data	FT_{15}	FT_{14}	FT_{13}	FT_{12}	FT_{11}	FT_{10}	FT_9	FT_8
4th Data	FT_7	FT_6	FT_5	FT_4	FT_3	FT_2	1	FT_0

Fig 2.2□Function Type for the Storage Function

For example, in the case of a peripheral device for which only the Storage Function is implemented, the function type is defined by FT = 00-00-00-02h.
If other functions are implemented in a peripheral device, the function type bit that corresponds to that function is set to "1."

2.3Function Definition Block

This section describes the function definition block (FD) within the device ID.
The function definition block is a 32-bit data table that is inherent to each function.The elements that comprise a function, the data transmission and reception methods, etc., are all determined on the basis of this data.

The following table shows the configuration of the function definition block for the Storage Function.

bit	7	6	5	4	3	2	1	0
1st Data	PT_7	PT_6	PT_5	PT_4	PT_3	PT_2	PT_1	PT_0
2nd Data	BB_7	BB_6	BB_5	BB_4	BB_3	BB_2	BB_1	BB_0
3rd Data	WA_3	WA_2	WA_1	WA_0	RA_3	RA_2	RA_1	RA_0
4th Data	RM	CRC	FD_5	FD_4	FD_3	FD_2	FD_1	FD_0

Fig 2.3□Storage Function Definition Block Configuration

PT: Number of partitions
The number of partitions can be set from 1 to 256.
Number of partitions = PT + 1 [partitions]
Except in special circumstances, the number of partitions is set to "1" (i.e., PT = 00h).

BB: Number of bytes per block
The number of bytes per block can be set over a range from 32 bytes to 8192 bytes.
Number of bytes per block = (BB + 1) x 32 [bytes]
Except in special circumstances, the number of bytes per block is set to "512" (i.e., BB = 0Fh).

RA: Number of accesses needed to read one block

This specifies the number of accesses that are needed in order to read one block of data.

The number of accesses can be set over a range from 1 to 15.

The amount of data transferred in one access is determined by dividing the size of one block by the number of accesses per block. If the result of this division has a remainder, the amount of data is rounded up to the next integer. Any excess is filled with "00h".

When RA = 0, reads are disabled.

Number of accesses = RA [times]

Size of one access = size of one block/RA [bytes]

WA: Number of accesses needed to write one block

This specifies the number of accesses that are needed in order to write one block of data.

The number of accesses can be set over a range from 1 to 15.

The amount of data transferred in one access is determined by dividing the size of one block by the number of accesses per block. If the result of this division has a remainder, the amount of data is rounded up to the next integer. Any excess is filled with "00h".

When WA = 0, writes are disabled.

Number of accesses = WA [times]

Size of one access = size of one block/WA [bytes]

RM: Removable media

This sets whether the media on which the data is stored is removable or not. (Examples of removable data include floppy disks and flash memory cards.)

Media	RM
Fixed	0
Removable	1

Fig 2.4□RM Value

CRC: CRC flag

This sets whether the CRC calculation (16 bits) is required or not for block data reads and writes.

Calculation method	CRC
CRC not needed	0
CRC needed	1

Fig 2.5□CRC Flag Value

FD: Reserved

This value is set to "0."

3 Storage Specifications

3.1Block Configuration

Data is written from/read to the Storage Function in units of blocks.
The configuration of the Storage Function is illustrated below.

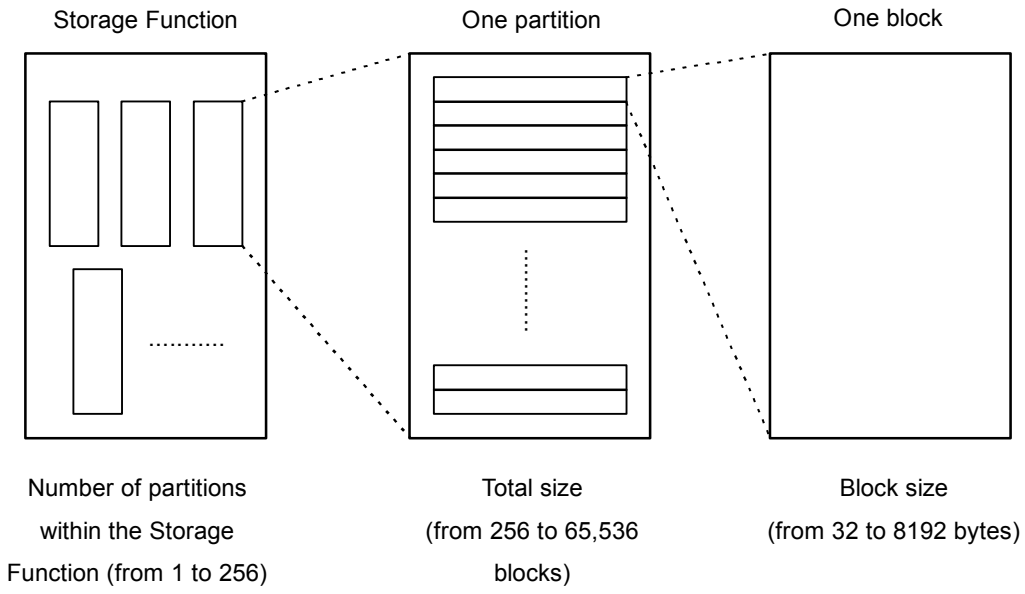

Fig 3.1 Memory Configuration in the Storage Function

The Storage Function can be divided into partitions; one Storage Function can accommodate anywhere from 1 to 256 partitions. (Normally, only one partition is established.)

Each partition is divided into the smallest unit of access, called a "block."
One partition consists of 256 to 65,536 blocks.

The number of partitions in the Storage Function and the size of one block is declared in the function definition block of each Storage Function.
The total size of one partition can be obtained by the host by issuing the [Get_Media_Info] command to each partition.

The block configuration within an individual partition is described in greater detail below.

A partition basically consists of three major areas.

1) Management area: This area contains information that is used to manage the data that is saved.
2) Save area: This area is used to save data and information if blocks in the management area or the data area are damaged.
3) Data area: This area contains data.

Fig. 3.2 shows the block map for one partition.
Each block is assigned a consecutive "block number," starting from "0000h" and continuing until the last block is numbered. Each of the above three areas are positioned in the block map as follows

1) Management area: This area starts from the last block in the block map. Starting from the last block, this area consists of (in order) a system area, a FAT area, and file information. The information on the starting block number of each area and the number of blocks used for the FAT area and the file information is stored in the system area.
(The size of the system area is always one block, regardless of the capacity of the storage device (media).)

2) Save area: This area starts from one block before the management area. The information on the starting block number of this area and the number of blocks used by this area are stored in the system area.
A contiguous area must be allocated for the save area.

3) Data area: This area starts from one block before the save area and extends to block number "0000h". This area is used to store data.

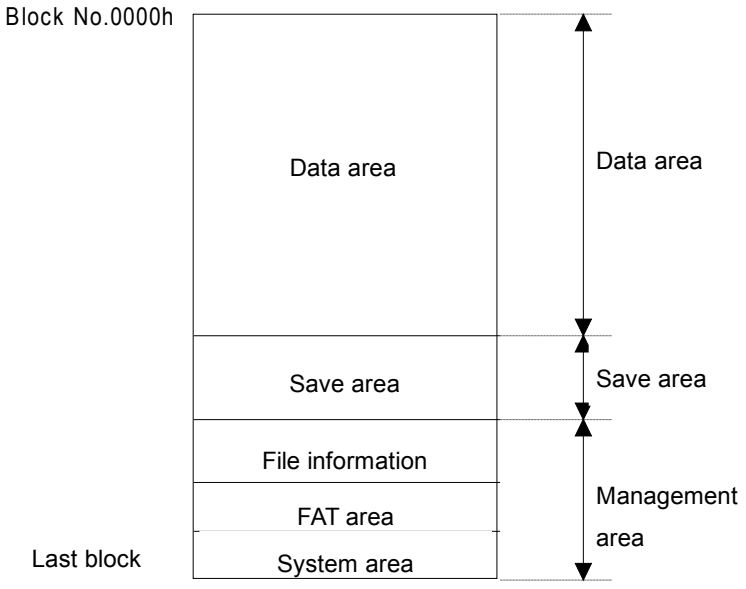

Fig 3.2□Memory Map for One Partition

3.2Management Area

The management area contains information on the data that is stored in the storage device (media), and information concerning the storage device (media) itself.
The management area is divided into three areas.

System area	1 area per partition
FAT area	1 area per partition
File information area	1 area per partition

3.2.1 System Area

The system area is a block where information that is essential to the management of files and data is stored, along with information on the storage device (media).
Because the block number where the system area is located differs for each partition (according to the size of the partition), the block number of the system area is obtained by using the [Get_Media_Info] command.

The size of the system area is always one block, regardless of the capacity of the storage device (media).

If the system area is physically damaged or can no longer be read or written, then the storage device (media) in question becomes unusable. Therefore, writing to the system area should be kept to a minimum. (For example, writing to the system area should be prohibited except when formatting the storage device (media) or when saving the area.)

If the information in the system area differs from information that was obtained by the [Get_Media_Info] command, the Format information is set to "unformatted," and the storage device (media) is reformatted using the media information that was obtained by the [Get_Media_Info] command.

If the Storage Function receives the [Get_Media_Info] command while in the unformatted state, the Storage Function returns predetermined parameters (media information), not the information in the storage area, to the host.

The contents of the system area block are indicated below.

byte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
+000h	Format information															
+010h	Volume label															
+020h	Volume label															
+030h	Date and time created								Reserved							
+040h	Total size		Partition No.		System area block No.		FAT area block No.		Number of FAT area blocks		File information block No.		Number of file information blocks		Volume Icon	Reserved
+050h	Save area block No.		Number of save area blocks		Reserved for execution file				Reserved							
+060h □+1F0h	Reserved															

Fig 3.3□System Area

Item	Data Size	Description
Format information	16byte	Indicates whether the storage device (media) is already formatted or not. If the storage device (media) is already formatted, "55h" is written in all 16 bytes. If the storage device (media) is not formatted, a value other than "55h" is written in all 16 bytes, or in 1 byte.
Volume Label	32byte	This is the label that is assigned to the partition. The contents of this field can be anything. The label is assigned when the storage device (media) is formatted.
Date and time created	8byte	This indicates the date and time when the storage device (media) was formatted. The year, month, day, and time are indicated in BCD code, and the day of the week is indicated by a code from "00h" to "06h" (corresponding to Sunday through Saturday). Example: 19/99/12/31/23/59/day of the week/00h/ The last byte is fixed at "00h."
Total size	2byte	This indicates the total size of the partition. The size is expressed in units of blocks, and can range from "00FFh" (256 blocks) to "FFFFh" (65,536 blocks).
Partition No.	2byte	This indicates the partition number of this partition. Normally, this value is "0000h" (PT = 1).
System area block No.	2byte	This indicates the block number of the block where the system area is located. The number here is the number of the final block in the partition.
FAT area block No.	2byte	This indicates the starting block number of the FAT area. The number here is (system area block number - 1).
Number of FAT area blocks	2byte	This indicates the number of blocks in the FAT area. This value must be at least "0001h."
File information block No.	2byte	This indicates the starting block number of the area where the file information is stored.
Number of file information blocks	2byte	This indicates the number of blocks in the file information area. This value must be at least "0001h."
Volume ICON	1byte	This specifies the number of the icon that is assigned to the partition as a whole. For details, refer to the host Boot ROM specifications.
Save area block No.	2byte	This indicates the starting block number of the save area. This value is "0000h" when there is no save area.
Number of save area blocks	2byte	This indicates the number of blocks in the save area. This value is "0000h" when there is no save area.
Reserved for execution file	2byte	This is fixed at "0000h" when an execution file cannot be executed from this partition. If an execution file can be executed, refer to the specifications for that peripheral for further details.
Reserved	Varies	All reserved areas are fixed at 00h.

Fig 3.4□System Area Parameters

3.2.2 FAT (File Allocation Table) Area

The FAT area manages the arrangement of stored data and the status of the blocks.

Because the block number where the FAT area starts and the number of blocks it includes varies according to the size of the partition, this information is obtained by using the [Get_Media_Info] command.

The FAT uses two bytes to manage the location of one block. Therefore, one block of the FAT area can manage 256 blocks.

The FAT consists of the FAT numbers of subsequent data in the data chain structure.
A "FAT number" indicates the block number where the corresponding data is stored.
"Data End" is written for the FAT number for the last block of data.

The blocks where the FAT area is located are also managed in the FAT.
The FAT has a chain structure that consists of at least one block.
If the FAT consists of only one block, "Data End" is written for that FAT number; if the FAT consists of multiple blocks, the FAT number of the next block is written.
File information also has a chain structure.

The values that are written in the FAT are described below:

Data:	FAT No. of the next data
Data End (the file ends with that block):	FFFAh
Unused:	FFFCh
Block Damaged (the storage media is physically damaged):	FFFFh

If the storage device (media) is formatted, the data in the FAT is cleared with "FFFCh" and "FFFAh" is written in the FAT that indicates the block where the system area is located, while the corresponding values are written for the file information and the FAT area, which both have a chain structure.

The contents of the FAT area are described below. (This example is for data that uses 6 blocks.)

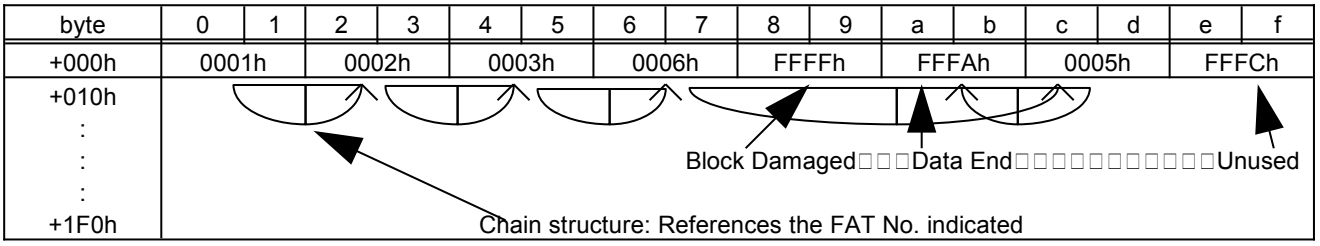

Fig 3.5 Example of Use of the FAT Area

Because each FAT number consists of two bytes, the address that corresponds to a FAT number in the FAT area is twice the FAT number. (Refer to the table below.)

FAT No.	Address in FAT area
0000h	+000h
0001h	+002h
□	□
00C7h	+18Eh
□	□
00F1h	+1E2h
□	□
00FFh	+1FEh
□	□
FFFFh	+1FFFEh

Fig 3.6□Correspondence between FAT Nos. and Addresses

If even one block of the FAT area is physically damaged or cannot be read or written, the entire FAT area is saved to the save area.

3.2.3 File Information

The Storage Function handles all files as data.
"File information" is information concerning files that are stored by the Storage Function.

Because the block number where the file information area starts and the number of blocks it includes varies according to the size of the partition, this information can be obtained by using the [Get_Media_Info] command, or can be obtained from the system area.
Both the information stored in this file information area and the information contained in file headers are needed in order to manage files.

1) File information area data

32 bytes are used per file, so a maximum of 16 files can be managed in one block of the file information area.
The Storage Function only supports a root directory; it does not support sub-directories.

The contents of the file information are shown below.

byte	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
+00h	Status	Copy	Start FAT		File Name											
+10h	Date								Block Size		Header		Reserved			

Fig 3.7□File Information

Item	Data Size	Description
Status	1byte	No data file: 00h Data file: 33h Execution file: CCh Values that indicate any status other than the above are reserved.
Copy	1byte	Copy flag This flag enables/disables copying of the file. If "FFh," the file cannot be copied; if any value other than "FFh," the file can be copied. The number of times a file can be copied can be limited by incrementing this value each time that the file is copied.
Start FAT	2byte	This indicates the FAT number where the file starts.
File Name	12Byte	This is the name that is used to identify the file. The name can consist of up to 12 normal-sized characters.
Date	8byte	This is the date and time when the file was stored (updated). The date and time is written in the same format as the date and time in the system area.
Block Size	2byte	This indicates the number of blocks used for this file.
Header	2byte	This indicates the block where the file header data for the file is located. The block is specified in terms of the number of blocks offset from the starting FAT.
Reserved	4byte	All reserved areas are fixed at 00h.

Fig 3.8□File Information Parameter

When the Start FAT indicates the management area, or if the block size is "0000h," the file information for this file is assumed to be damaged, and the file information status is set to "00h." There should not be two files with the same file name and status in one partition.

2) File header data

File header data is file information that is stored within the file. The contents of the file header are as shown below.

byte	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
+000h	Storage Comment															
+010h	File Comment															
+020h	File Comment															
+030h	File Comment															
+040h□	File ICON Data															

Fig 3.9□File Header

Item	Data Size	Description
Storage Comment	16byte	The storage comment can accommodate up to 16 normal-sized characters; double-width characters are not supported. The comment is described in ASCII code.
File Comment	48byte	This comment concerns this file. (Either one- or two-byte codes may be used.) This field is used by the host.
File ICON Data	Remaining area	This is the icon data for the file. This field is used by the host.

Fig 3.10□File Header Data Parameters

This file header data can be located in any block in the file. The block where the file header data is located can be found in the file information header.

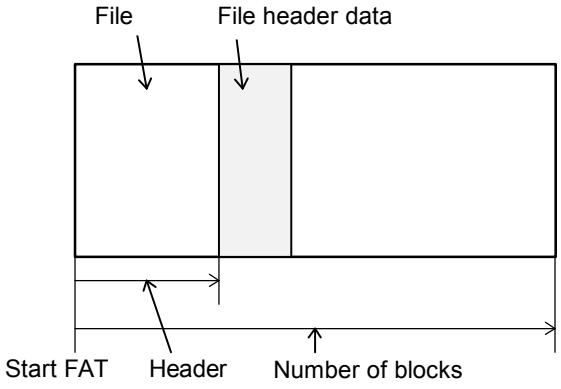

Fig 3.11□File Header Data Position

3.3Data Area

The data area is the area where data is stored.

A single data file can consist of anywhere from one block to all of the blocks in the data area. Because the block number where the data area starts and the size of the data area varies according to the size of the partition, this information is calculated on the basis of information obtained by using the [Get_Media_Info] command.

The starting block number of the data area is determined as follows:
If there is no save area, the starting block number of the data area is (file information block number - number of file information blocks).
If there is a save area, the starting block number of the data area is (save area block number - number of save area blocks).

The size of the data area is the starting block number minus "0000h."

Information on each file is stored in the file information area in the management area and in the FAT area.

3.4Save Area

The save area is used to save the management area or the data area if one of the blocks in those areas becomes damaged.

Because the block number where the save area starts and the number of blocks in the save area varies according to the size of the partition, this information can be obtained by using the [Get_Media_Info] command, or can be obtained from the system area.

If a management area block becomes damaged, move the blocks containing the damaged file information and FAT information into the save area, and, if necessary, overwrite the information in the system area concerning the file information block numbers and the FAT area block numbers.

when saving the FAT, allocate contiguous blocks in the save area for the FAT, and save the entire FAT in the save area.

If the entire save area is in use, blocks at the high end of the data area are used. In this event, the amount of data (the number of blocks) that can be saved is reduced.
(If this situation arises, it is recommended that a message that indicates that the media cannot be used be displayed.)

In the case of a read-only storage device (media) or a storage device (media) that does not require a save area (ROM, SRAM, etc.), it is not necessary to allocate a save area.

3.5Format Processing

Before using the Storage Function, it is necessary to first perform format processing in order to initialize the storage media.
Formatting requires writing "00h" to the entire data area and file information area, "FFFAh" to the system area in the FAT area, and "FFFCh" to the remainder of the FAT area.
In order to indicate that formatting has been completed, write "55h" to the specified location in the system area, and write the media information obtained by [Get_Media_Info] to the system area as well.
If any blocks are damaged, write "FFFFh" to the FAT for that block so that the block will not be used.

4 Commands

This section describes those commands that the Storage Function supports out of the commands included in the Maple Bus 1.0 Standard Specifications.
All of the setting examples assume that the Storage Function is connected to LM-Bus No. 1 of port A.

4.1Control Commands

4.1.1 Get_Media_Info

Issuing authority:	Host	
Command code:	0Ah	
Data size:	02h (8 bytes)	
Data:	Function type:	4 bytes
	PT:	1 byte
	Fixed value:	3 bytes
Expected return value:	[Data Transfer]	
Description:	This command is used to get, from the Storage Function, the media information that is necessary in order to use the other commands. This command must be issued first after connection has been confirmed. An example of this command is shown below.	

Data address	Data	Example setting	Description
+0000h	Command code	0Ah	Specifies [Get_Media_Info]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	02h	The data size is 8 bytes
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h	PT	00h	Specifies the partition number
+0009h	Fixed value	00h	'Fixed to "00h"
+000Ah		00h	
+000Bh		00h	

Fig 4.1□Get_Media_Info Command Example

When the Storage function receives this command, it sends the media information shown in Fig. 4.2 by means of the [Data Transfer] command.
For details on [Data Transfer], refer to section 4.1.5, "Data Transfer."

If the media is removable and is not ready, [File Error] is returned.

Data address	Data	Example setting	Description
+0000h	Command code	08h	Specifies [Data Transfer]
+0001h	Receiver AP	00h	Port A
+0002h	Sender AP	01h	Expansion device (LM-Bus No. 1)
+0003h	Data size	07h	The data size is 2 bytes
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h □ +001Fh	Media information		Refer to Fig. 4.3

Fig 4.2□Response to the Get_Media_Info Command

Data address	+0	+1	+2	+3	+4	+5	+6	+7
+0008h	Total size		Partition No.		System area block No.		FAT area block No.	
+0010h	Number of FAT area blocks		File information block No.		Number of file information blocks		Volume ICON	Reserved
+0018h	Save area block No.		Number of save area blocks		Reserved for execution file			

Fig 4.3□Media Information Configuration

Item	Data Size	Description
Total size	2Byte	Indicates the total size of this partition. (Fixed data)
Partition No.	2Byte	Indicates the partition number of this media. (Fixed data)
System area block No.	2Byte	Indicates the block number where the system area is stored. (Fixed data)
FAT area block No.	2Byte	Indicates the block number where the FAT area starts.
Number of FAT area blocks	2Byte	Indicates the number of blocks in the FAT area.
File information block No.	2Byte	Indicates the block number where the file information area starts.
Number of file information blocks	2Byte	Indicates the number of blocks in the file information area.
Volume ICON	2Byte	Indicates the number of the icon for this media.
Save area block No.	2Byte	Indicates the block number where the save area starts.
Number of save area blocks	2Byte	Indicates the number of blocks in the save area.
Reserved for execution file	4Byte	Fixed to "0000-0000h" if execution files cannot be executed.
Reserved	2Byte	Fixed to "0000h."

Fig 4.4□Media Information Parameters

For details on the media information, refer to section 3.2.1.
The total size, the partition number, and the system area block number values (the data in ROM) are always determined by the partition.

4.1.2 Block_Read

Issuing authority:	Host	
Command code:	0Bh	
Data size:	02h (8 bytes)	
Data:	Function type:	4 bytes
	PT:	1 byte
	Phase:	1 byte
	Block No.:	2 bytes
Expected return value:	[Data Transfer]	
Description:	This command requests the data in the specified block number from the Storage Function.	
	An example of this command is shown below.	

Data address	Data	Example setting	Description
+0000h	Command code	0Bh	Specifies [Block_Read]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	02h	The data size is 8 bytes
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h	PT	00h	Specifies the partition number
+0009h	Phase		Specifies the phase. (See below)
+000Ah	Block No.		Specifies the block number to be read
+000Bh			

Fig 4.5□Block_Read Command Example

If it is not possible to read one block of data in one access (i.e., RA >= 2), divide the data in the block and make (value of RA) requests. "Phase" is used in this case to indicate which group of divided data from the block is to be read.

Example: Reading data from a 512-byte block in three accesses (RA = 3)
Because 512/3 = 170.6666..., the amount of data to be read in one access is 171 bytes. Although this means a total of 513 bytes of data will be transferred in the three accesses, the data in excess of one block (512 bytes) is filled with "00h." (This means that the 171st byte of the third data transfer will be "00h.")

"Phase" always starts with "00h," and continues with "01h," "02h," and "03h." If another command is issued in the middle of the Block_Read phases or nonconsecutive phase numbers are issued, [File Error] is returned.

The block data that was read is returned by means of the [Data Transfer] command.

4.1.3 Block_Write

Issuing authority:	Host	
Command code:	0Ch	
Data size:	02h + n (8 bytes + n x 4 bytes)	
Data:	Function type:	4 bytes
	PT:	1 byte
	Phase:	1 byte
	Block No.:	2 bytes
	Write data:	n x 4 bytes
Description:	This command writes data in the specified block number in the Storage Function.	
	An example of this command is shown below.	

Data address	Data	Example setting	Description
+0000h	Command code	0Ch	Specifies [Block_Write]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	02h + n	
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h	PT	00h	Specifies the partition number.
+0009h	Phase		Specifies the phase. (See below)
+000Ah	Block No.		Specifies the block number to be read
+000Bh			
+000Ch □	Write Data		Block data that is to be written (n x 4 bytes)

Fig 4.6□Block_Write Command Example

If it is not possible to write one block of data in one access (i.e., WA >= 2), divide the data in the block and make (value of WA) requests. "Phase" is used in this case to indicate which group of divided data from the block is to be written.

Example: Writing data for a 512-byte block in three accesses (WA = 3)
Because 512/3 = 170.6666..., the amount of data to be written in one access is 171 bytes.
Although this means a total of 513 bytes of data will be transferred in the three accesses,

the data in excess of one block (512 bytes) is filled with "00h." (This means that the 171st byte of the third data transfer will be "00h.")

"Phase" always starts with "00h," and continues with "01h," "02h," and "03h." If another command is issued in the middle of the Block_Write phases or if nonconsecutive phase numbers are issued, [File Error] is returned, and the data is not written in that block.

4.1.4 Get_Last_Error

Issuing authority:	Host	
Command code:	0Dh	
Data size:	02h (8 bytes)	
Data:	Function type:	4 bytes
	PT:	1 byte
	Phase:	1 byte
	Block No.:	2 bytes
Description:	This command is used to determine whether an error occurred in the command (group) that was last issued to the Storage Function at the location specified by PT and Block. If no error occurred, [Device Reply] is returned; if an error did occur, [File Error] is returned. An example of this command is shown below.	

Data address	Data	Example setting	Description
+0000h	Command code	0Dh	Specifies [Get_Last_Error]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	02h	The data size is 8 bytes
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h	PT	00h	Specifies the partition number.
+0009h	Phase		Specifies the phase
+000Ah	Block No.		Specifies the block number that was processed
+000Bh			

Fig 4.7□Get_Last_Error Command Example

This command is issued at the next access following the command (group) for which a check for errors is being made.
Use the same PT and Block No. values that were used in the command that was executed previously ([Block_Read], [Block_Write], etc.).
For Phase, add one to the Phase value from the command that was issued previously.

Example: Checking to see if the write was performed correctly after writing to a block through four accesses

1st access: Block_Write (Phase = 0)	Device Reply is returned.
2nd access: Block_Write (Phase = 1)	Device Reply is returned.
3rd access: Block_Write (Phase = 2)	Device Reply is returned.
4th access: Block_Write (Phase = 3)	Device Reply is returned.
5th access: Get_Last_Error (Phase = 4)	Device Reply is returned. (Normal)
	File error is returned. (Error occurred)

4.1.5 Data Transfer

Issuing authority: Storage Function

Command code: 08h

Data size: For [Get_Media_Info]: 07h

For [Block_Read]: 02h + n (n x 4 bytes)

Data: For [Get_Media_Info]:

- Function type: 4 bytes
- Media information: 24 bytes

For [Block_Read]:

- Function type: 4 bytes
- PT: 1 byte
- Phase: 1 byte
- Block No.: 2 bytes
- Block data: n x 4 bytes

Description: This command returns data in response to a request from the host.

This command returns media information for [Get_Media_Info].

For details, refer to section 4.1.1, "Get_Media_Info."

This command returns the block data for [Block_Read].

An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	08h	Specifies [Data Transfer]
+0001h	Receiver AP	00h	Port A
+0002h	Sender AP	01h	Expansion device (LM-Bus No. 1)
+0003h	Data size	02h + n	
+0004h	Function type	00h	Specifies the function type as "Storage"
+0005h		00h	
+0006h		00h	
+0007h		02h	
+0008h	PT	00h	Specifies the partition number.
+0009h	Phase		Specifies the phase
+000Ah	Block No.		Specifies the block number
+000Bh			
+000Ch □	Block Data		Data from the block that was read (n x 4 bytes)

Fig 4.8□Data_Transfer Command Example

4.1.6 Device Reply

Issuing authority: Storage Function
Command code: 07h
Data size: 00h
Data: None
Description: This command is returned to the host as the reply command if the command that the host sent was processed normally by the Storage Function.
An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	07h	Specifies [Device Reply]
+0001h	Receiver AP	00h	Port A
+0002h	Sender AP	01h	Expansion device (LM-Bus No. 1)
+0003h	Data size	00h	The data size is 0 bytes

Fig 4.9□Device Reply Command Example

4.1.7 Device Request

Issuing authority: Host
Command code: 01h
Data size: 00h
Data: None
Expected return value: [Device Status]
Description: This command requests [Device Status] from the receiver AP peripheral device.
After initialization, the Storage Function does not respond to any other commands until it receives this command.
An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	01h	Specifies [Device Request]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	00h	The data size is 0 bytes

Fig 4.10□Device Request Command Example

4.1.8 All Status Request

Issuing authority: Host
Command code: 02h
Data size: 00h
Data area: None
Expected return value: [Device All Status]
Description: This command requests all device statuses (both Fixed Device Status and Free Device Status) from the receiver AP peripheral device.
An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	02h	Specifies [All Status Request]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	00h	The data size is 0 bytes

Fig 4.11□All Status Request Command Example

4.1.9 Device Reset

Issuing authority: Host
Command code: 03h
Data size: 00h
Data area: None
Expected return value: [Device Reply]
Operation sequence: (1) [Device Reply] is returned.
(2) The peripheral is initialized.
Description: This command can be used to initialize the device specified as the Receiver AP.
The storage media is not initialized.
An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	03h	Specifies [Device Reset]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	00h	The data size is 0 bytes

Fig 4.12□Device Reset Command Example

4.1.10 Device Kill

Command code: 04h
Data size: 00h
Data area: None
Expected return value: [Device Reply]
Operation sequence: (1) [Device Reply] is returned.
(2) The peripheral stops operating.
Description: This command does not request any operation on the part of the peripheral device that is specified as the Receiver AP.
The Storage Function goes into standby (consuming only the standby current), and does not accept any commands.
In order to make the Storage Function resume operations, it is necessary either to execute a hardware reset or to turn off the power and then restart.
An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	04h	Specifies [Device Kill]
+0001h	Receiver AP	01h	Expansion device (LM-Bus No. 1)
+0002h	Sender AP	00h	Port A
+0003h	Data size	00h	The data size is 0 bytes

Fig 4.13□Device Kill Command Example

4.1.11 Device Status

Issuing authority: Peripheral device
Command code: 05h
Data size: 1Ch (28)
Data area:

Device ID:	16 bytes
Region code:	1 byte
Manufacturer's name:	31 bytes
License:	60 bytes
Standby current consumption:	2 bytes
Maximum current consumption:	2 bytes

Description: This command returns the Fixed Device Status data in response to a [Device Request] command from the host.

□An example of this command is shown below.□

Data address	Data	Example setting	Description
+0000h	Command code	05h	Specifies [Device Status]
+0001h	Receiver AP	00h	Port A
+0002h	Sender AP	01h	Expansion device (LM-Bus No. 1)
+0003h	Data size	1Ch	The data size is 112 bytes
+0004h □ +0013h	Device ID		Specifies the device ID
+0014h	Region code		Specifies the region code
+0015h □ +0033h	Manufacturer's name		Specifies the manufacturer's name
+0034h □ +006Fh	License		Specifies the license information
+0070h	Standby current consumption		Specifies the standby current consumption
+0071h			
+0072h	Maximum current consumption		Specifies the maximum current consumption
+0073h			

Fig 4.14□Device Status Command Example

4.1.12 Device All Status

Issuing authority: Peripheral device

Command code: 06h

Data size: 1Ch + n

Data area: Fixed Device Status: 112 bytes

- Device ID: 16 bytes
- Region code: 1 byte
- Manufacturer's name: 31 bytes
- License: 60 bytes
- Standby current consumption: 2 bytes
- Maximum current consumption: 2 bytes

Free Device Status: n x 4 bytes

Description: This command returns both the Fixed Device Status and Free Device Status in response to the [All Status Request] command from the host.

4.2Error Commands

4.2.1 Function Type Unknown

Issuing authority:	Peripheral device
Command code:	FEh
Data size:	00h
Data area:	None
Description:	This error command is returned when the function type that was received does not exist for the peripheral device.
Possible causes:	(1) The function type specification is incorrect. (2) The data description is incorrect. (3) The device ID was garbled. (4) The data became garbled during transmission.
Action:	(1) Correct the function type specification. (2) Correct the data description. (3) Send [Device Request] again to get the device ID. (4) Try the transmission again. (Retry three times, and then handle in the same manner as a time out.)

4.2.2 Command Unknown

Issuing authority:	Storage Function
Command code:	FDh
Data size:	00h
Data area:	None
Description:	This error command is returned when the command that was received is not supported by the Storage Function.
Possible causes:	(1) The command specification is incorrect. (2) The data description is incorrect. (3) The device ID was garbled. (4) The data became garbled during transmission.
Action:	(1) Correct the command specification. (2) Correct the data description. (3) Send [Device Request] again to get the device ID. (4) Try the transmission again. (Retry three times, and then handle in the same manner as a time out.)

4.2.3 Transmit Again

Issuing authority:	Host, Storage Function
Command code:	FCh
Data size:	00h
Data area:	None
Description:	This error command requests that the same data be transmitted again when some type of error was found in data that was received.
Possible causes:	(1) A parity error occurred. (2) A data overflow occurred. (3) The data became garbled during transmission. (4) Other cause.
Action:	Send the data again. (Retry three times, and then handle in the same manner as a time out.)

4.2.4 File Error

Issuing authority:	Storage Function
Command code:	FBh
Data size:	01h
Data:	Function error code: 4 bytes
Description:	This command is send to the host as a reply command when some type of error occurred in the Storage Function while processing a command from the host (if a series of commands being processed were divided into several groups, those instructions are regarded as one process). An example of this command is shown below.

Data address	Data	Example setting	Description
+0000h	Command code	FBh	Specifies [File Error]
+0001h	Receiver AP	00h	Port A
+0002h	Sender AP	01h	Expansion device (LM-Bus No. 1)
+0003h	Data size	01h	The data size is 4 bytes
+0004h	Function error code		Refer to Fig. 4.10
+0005h			
+0006h			
+0007h			

Fig 4.15□File Error Command Example

The function error code details are displayed below

bit	7	6	5	4	3	2	1	0
1st Data	FE_{31}	FE_{30}	FE_{29}	FE_{28}	FE_{27}	FE_{26}	FE_{25}	FE_{24}
2nd Data	FE_{23}	FE_{22}	FE_{21}	FE_{20}	FE_{19}	FE_{18}	FE_{17}	FE_{16}
3rd Data	FE_{15}	FE_{14}	FE_{13}	FE_{12}	FE_{11}	FE_{10}	FE_9	FE_8
4th Data	FE_7	FE_6	FE_5	FE_4	FE_3	FE_2	FE_1	FE_0

Fig 4.16□Function Error Codes

If a particular error occurred, that corresponding item is set to "1;" if that error did not occur, that item is set to "0."

The errors are described below.
If the action taken by the host does not clear the error, then disconnect processing is executed and a user message is output.

□FE_0□PT Error

Description:	The specified partition does not exist.
Storage Function operation:	All data involved when the error occurred is damaged.
Host action:	Send the data again (up to three times).

□FE_1□Phase Error

Description:	The Phase + 1 value is greater than the value of RA (when executing Block_Read). The Phase + 1 value is greater than the value of WA (when executing Block_Write). The Phase value changed by an amount other than "+ 1". (For example: Phase 0 -> 2 -> 1)
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Send the data again (up to three times). In the case of a block, start over from Phase = 0.

□FE_2□Block Error

Description:	The specified partition does not exist.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Send the data again (up to three times).

□FE_3□Physical Write Error

Description:	The data was not written in the storage device (media) correctly.
Storage Function operation:	The incorrect data is written in the storage device (media), and is not erased.
Host action:	Send the data again (up to three times). If this error is detected even after three retries, the data is saved in the save area.

□FE_4□Length Error

Description:	The specified data size does not match the size of the data that was sent.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Send the data again (up to three times).

□FE_5□Domain Error

Description:	An access was made to an access-prohibited area.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Change the area being accessed and then send the data again.

□FE_6□Drive not Ready

Description:	Removable media was not mounted.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Send the data again (up to three times, then handle in the same manner as a time out.)

□FE_7□CRC Error

Description:	While the CRC flag was set to "1," a CRC error occurred while receiving block data.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Recalculate the CRC and then send the data again (up to three times).

□FE_{31}□Undefined Error

Description:	An undefined error occurred.
Storage Function operation:	All data involved when the error occurred (the entire block) is damaged.
Host action:	Send the data again (up to three times, then handle in the same manner as a time out.)

□All other error codes are reserved.

5 Protocol Flow

The following diagram provides a general overview of the basic communications protocol between the host and the Storage Function.

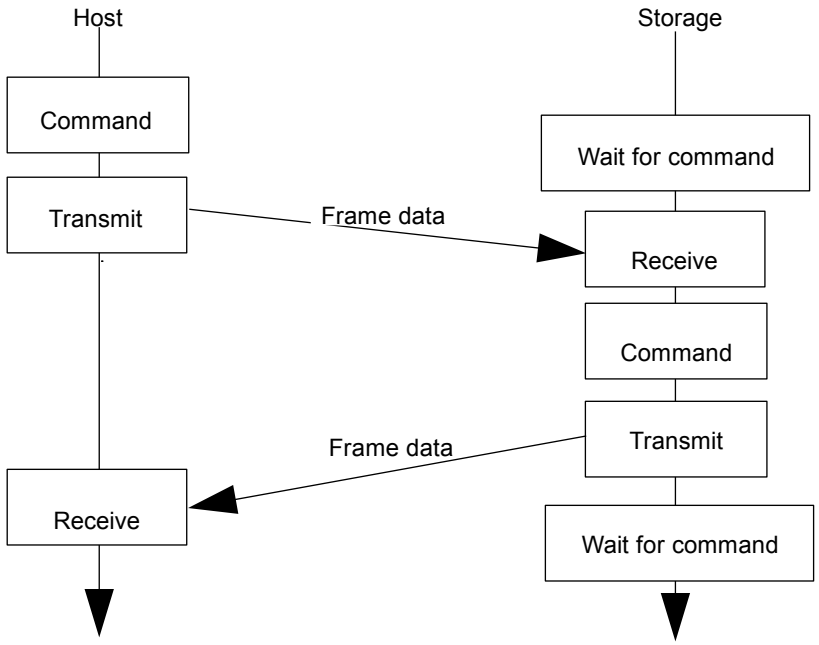

Fig 5.1 Overview of Communications Protocol

5.1Get_Media_Info Processing Flow

The flow of processing for the Get_Media_Info operation is shown below.

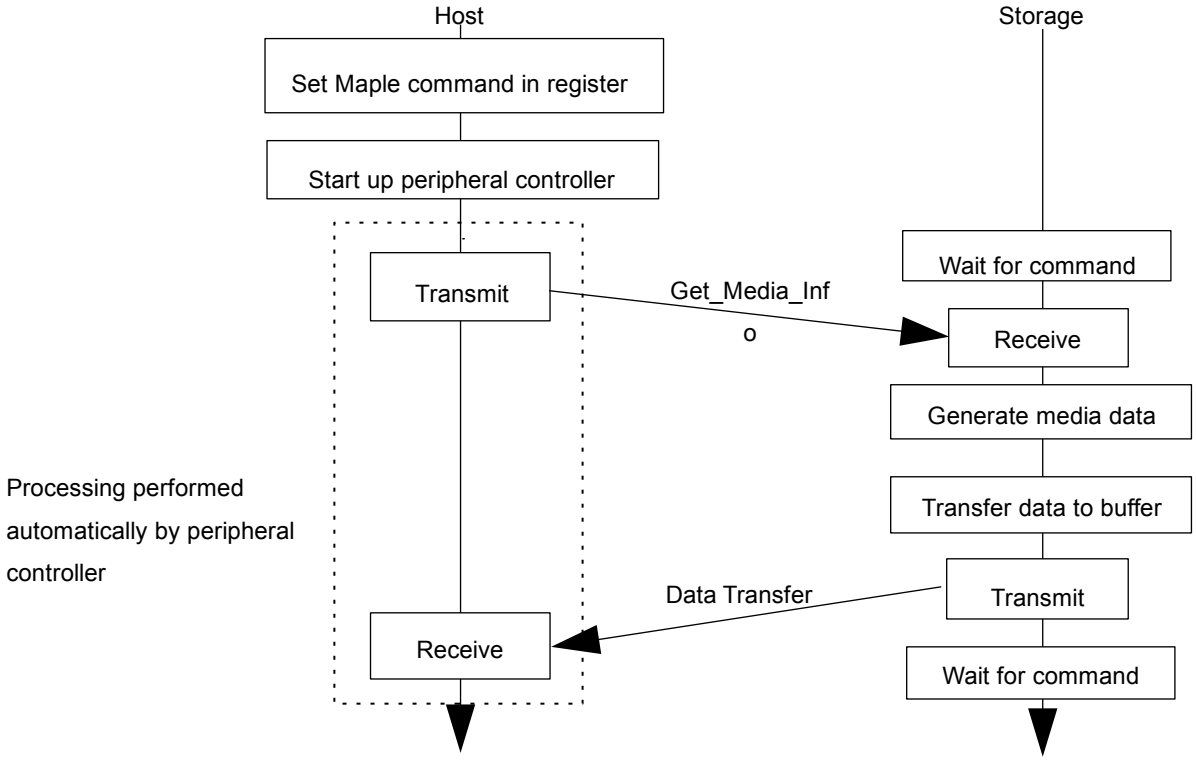

Fig 5.2 Overview of Get_Media_Info Processing Flow

5.2Block_Read Processing Flow

The flow of processing for the Block_Read operation is shown below.

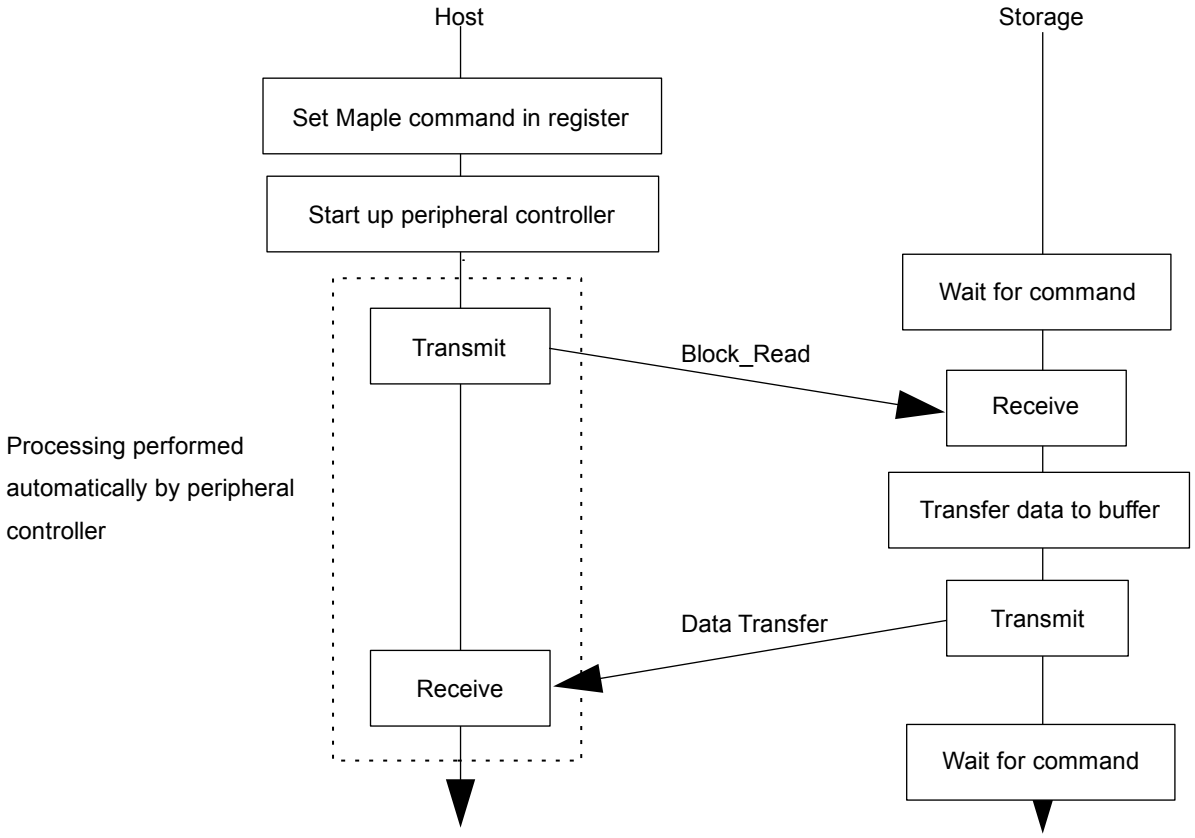

Fig 5.3□Overview of Block_Read Processing Flow

If phases are used, the above process is repeated the necessary number of times.

5.3Block_Write Processing Flow

The flow of processing for the Block_write operation is shown below.

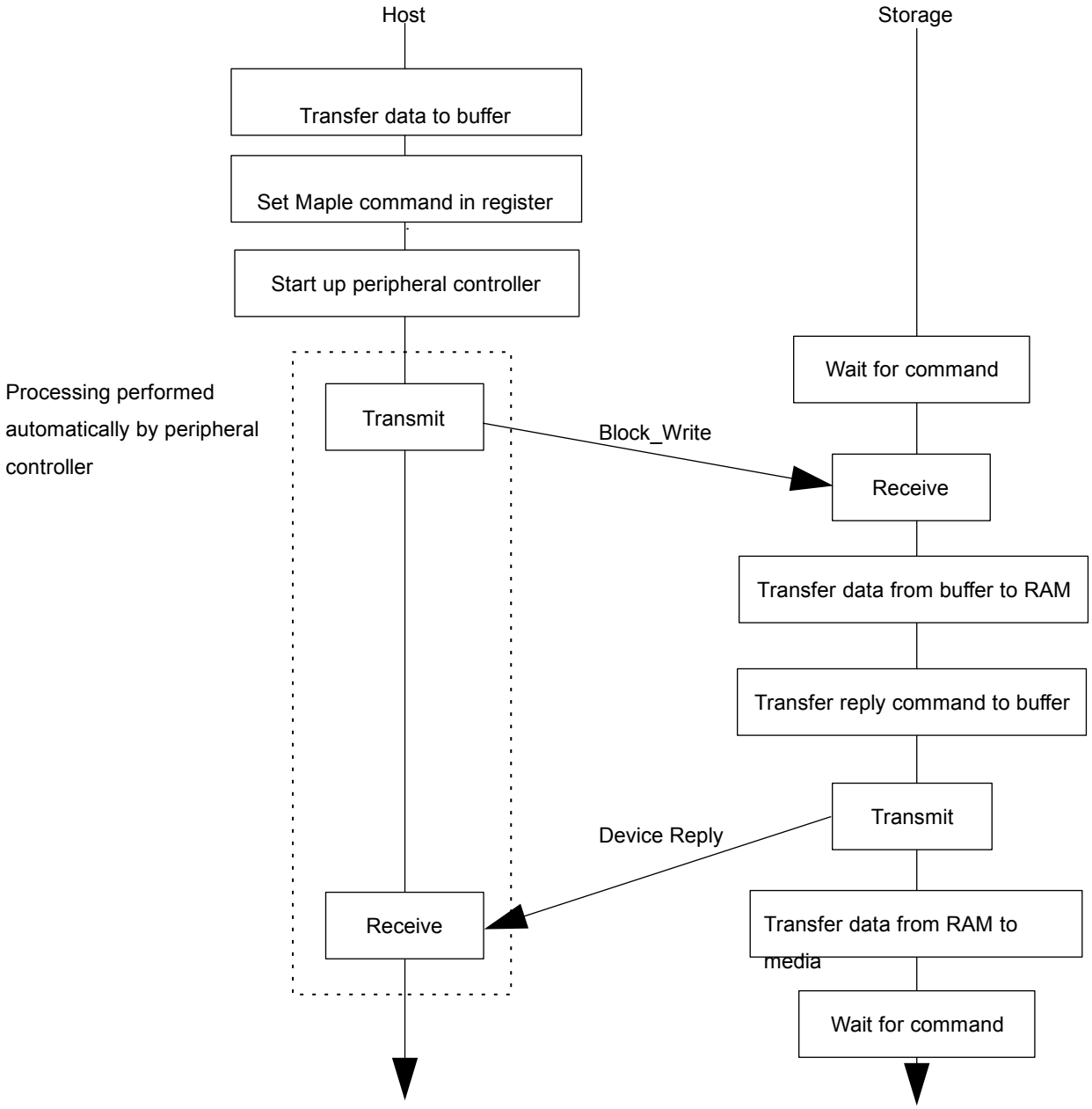

Fig 5.4□Overview of Block_Write Processing Flow

If phases are used, the above process is repeated the necessary number of times.
After the [Block_Write] processing is completed, a check for errors in the write data is made by sending [Get_Last_Error] to the host in the next access.

5.4Get_Last_Error Processing Flow

The flow of processing for the Get_Last_Error operation is shown below.

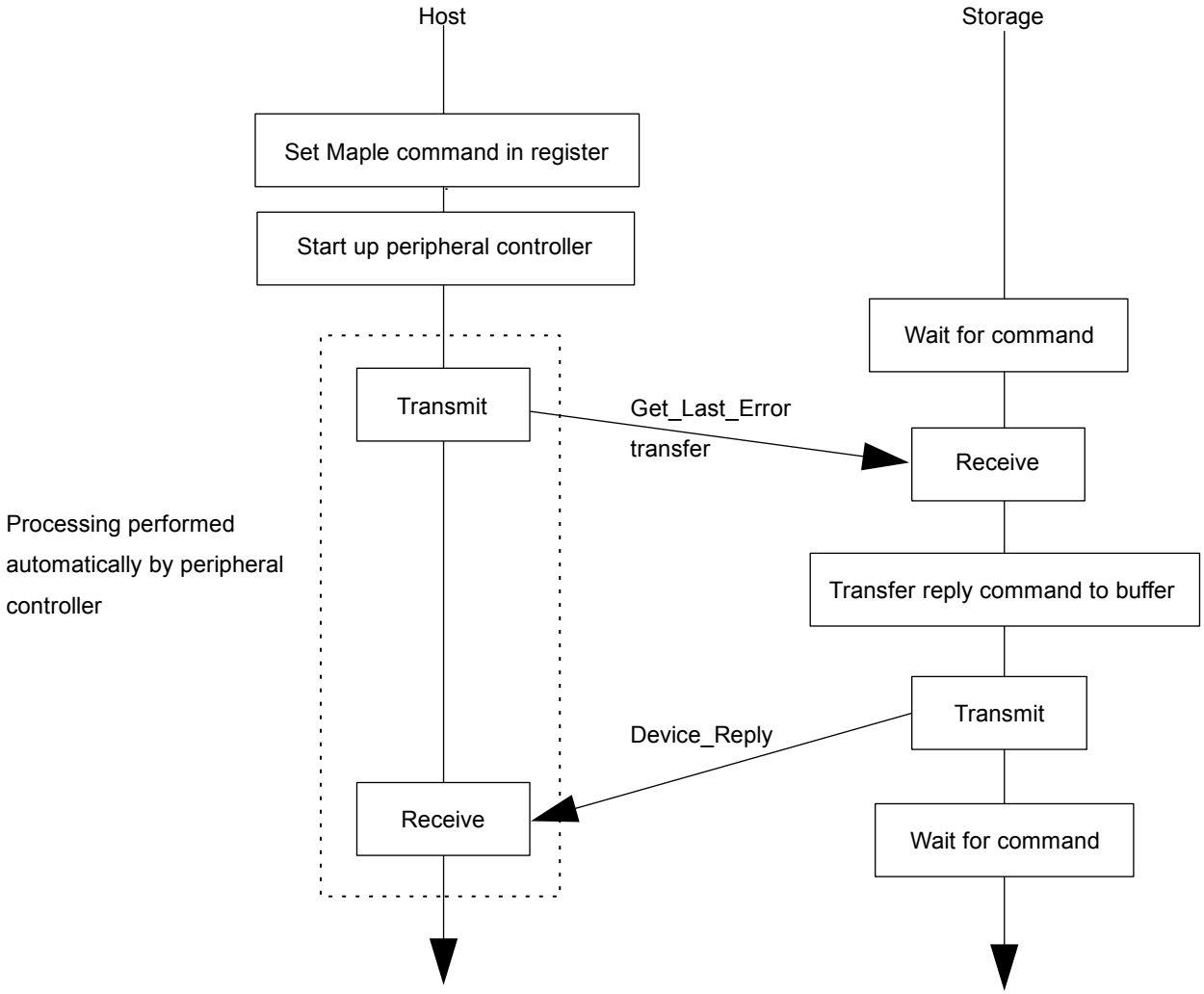

Fig 5.5 Overview of Get_Last_Error Processing Flow

6 Postscript

The contents of these specifications are subject to change in whole or in part until the official version (Rev. 1.0) is distributed.