SATA-IP exFAT reference design manual

Rev1.1 5-Sep-13

1 Introduction

The exFAT file system is the successor to FAT32 in the FAT family of file systems. It incorporates several improvements over FAT32 such as supporting more than 4 GB file size.

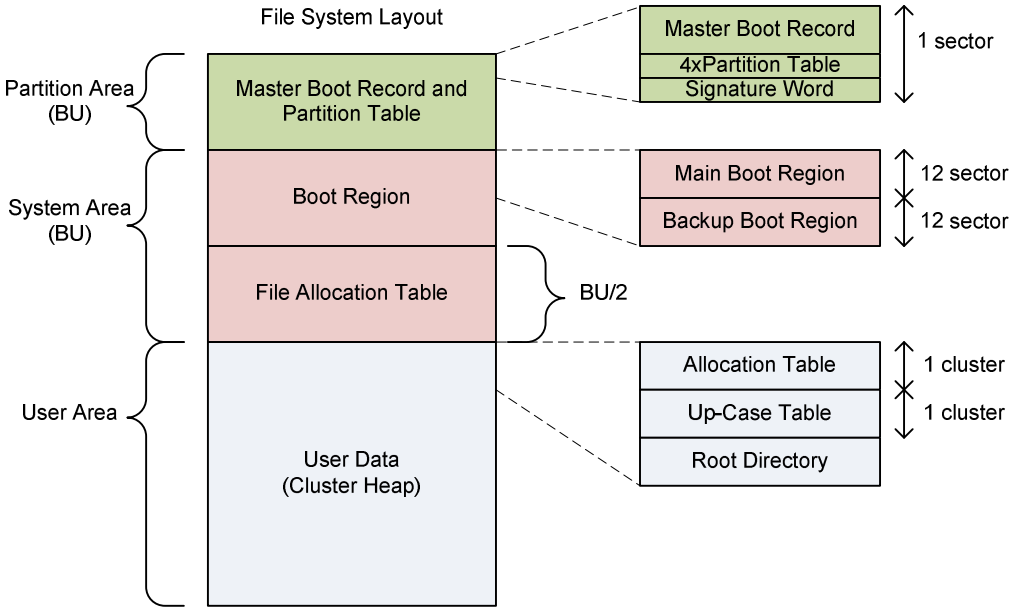

Figure 1-1 Example of File System Layout

Based on one partition table, File system layout can be displayed as shown in Figure 1-1. Disk area can be split into three zones, i.e. Partition Area, System Area, and User Area. Size of Partition and System Area is equal to Boundary Unit. The recommended value of Boundary Unit is shown in Table 1-1. The smallest unit to store data in file or directory is called cluster unit which will also depend on disk capacity. From Table 1-1, the boundary unit is equal to 128 times of cluster size.

Disk Capacity	Sectors per Cluster (sectors)	Boundary Unit (sectors)
32 ~ 128 GB	256	32768
~ 512 GB	512	65536
~ 2 TB	1024	131072

Table 1-1 Recommended Boundary Unit and Cluster Size

Partition Area

The first sector of Partition Area is Master Boot Record which includes executable codes and Partition Table that includes the information to identify the partition. Master Boot Record size is 446 byte size, while one partition table size is 16 byte. 2-byte Signature word of FAT is AA55h. The details of 16-byte data in Partition Table are shown in Figure 1-2. The other sectors in Partition Area are reserved to make area alignment with Boundary Unit size.

	Byte0	Byte1	Byte2	Byte3
0	Boot Indicator	Starting Head	Starting Sector/Cylinder	
1	System ID	Ending Head	Ending Sector/Cylinder	
2	Relative Sector			
3	Total Sector			

Figure 1-2 Partition Table

System Area

System Area contains two Boot Regions and File Allocation Table (FAT). There are 12 sectors in each Boot Region which more details are shown in Figure 1-3. The first sector contains the parameters for the partition such as cluster size, partition size. The twelfth sector contains a repeating pattern of the 4-byte checksum which is calculated from the contents of all other sectors in Boot Region. The Backup Boot Region is a backup of the Main Boot Region for Main Boot Region content recovery. Similar to Partition Area, the free space area in Boot Region is reserved for Boundary Unit alignment.

In exFAT, FAT is only used for keeping chains of clusters of fragmented files. FAT table does not need to be updated for non-fragmented files. The example of FAT after format is shown in Figure 1-4. 4-byte FAT Entry is referred to next cluster number in chain following this cluster. The 1st and 2nd FAT entries are reserved and the values are fixed to 0xFFFFFFF8h and FFFFFFFFh sequentially. Typically, the 3rd – 5th entries are equal to 0xFFFFFFFFh which means that Cluster#2 – 4 are already occupied to be Allocation Bitmap Table, Up-Case Table, and Root Directory consecutively. More details about entry value are shown in Table 1-2.

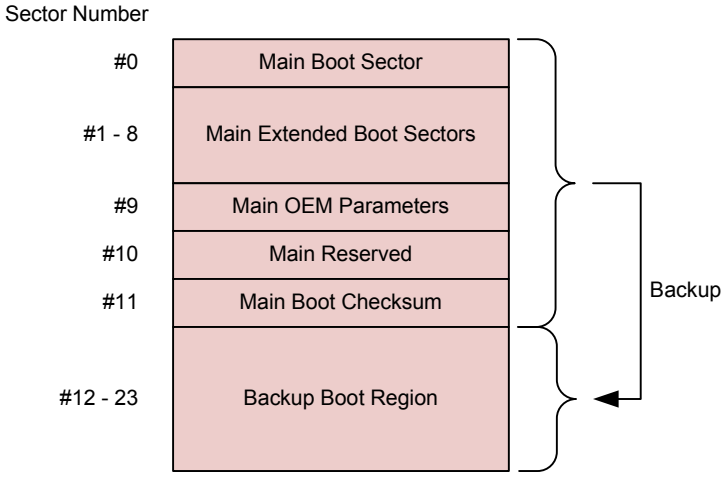

Figure 1-3 Boot Region

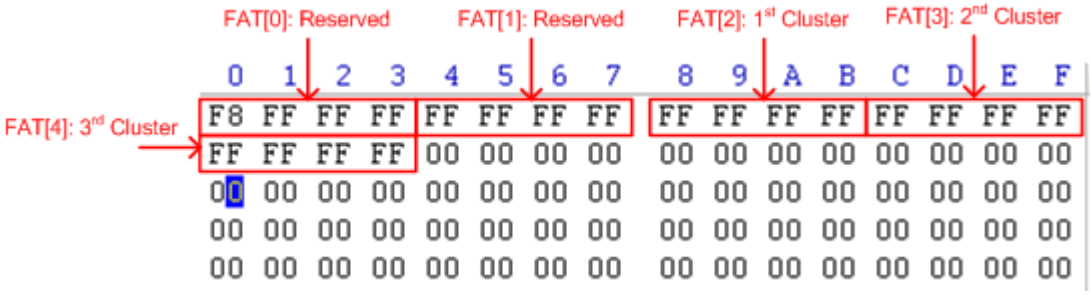

Figure 1-4 Example of File Allocation Table

FAT Entry Value	Contents
00000000h	Cluster is not in use. May be allocated to non-fragmented file or directory.
00000002h to ClusterCount + 1	Cluster is already allocated. The value is the cluster number of the next cluster following this cluster
FFFFFFF7h	Defective cluster
FFFFFFFFh	Already allocated and this is final cluster of the file.
Others	Reserved

Table 1-2 FAT Entry Value

User Area

The first cluster number in User Area is Cluster#2 for storing Allocation Bitmap Table. One bit in Allocation Bitmap Table is referred to 1 cluster data, so 1st byte shows Cluster#2 – 9 available status. If the cluster is free, the Bit in Allocation Bitmap Table is equal to '0'. '1' in Allocation Bitmap Table means this cluster is already allocated.

Cluster#3 stores Up-case table to define the conversion from lower-case to upper-case characters which is important due to the File Name directory entry using Unicode characters and the exFAT file system being case insensitive and case preserving.

Cluster#4 is Root directory. Each directory includes root directory consists of a series of directory entries. One entry size is equal to 32 bytes. The example of Root Directory containing one directory is shown in Figure 1-5. The 1st byte of each entry is entry type, and six entry types are displayed in this example. The description of each entry type is described in Table 1-3

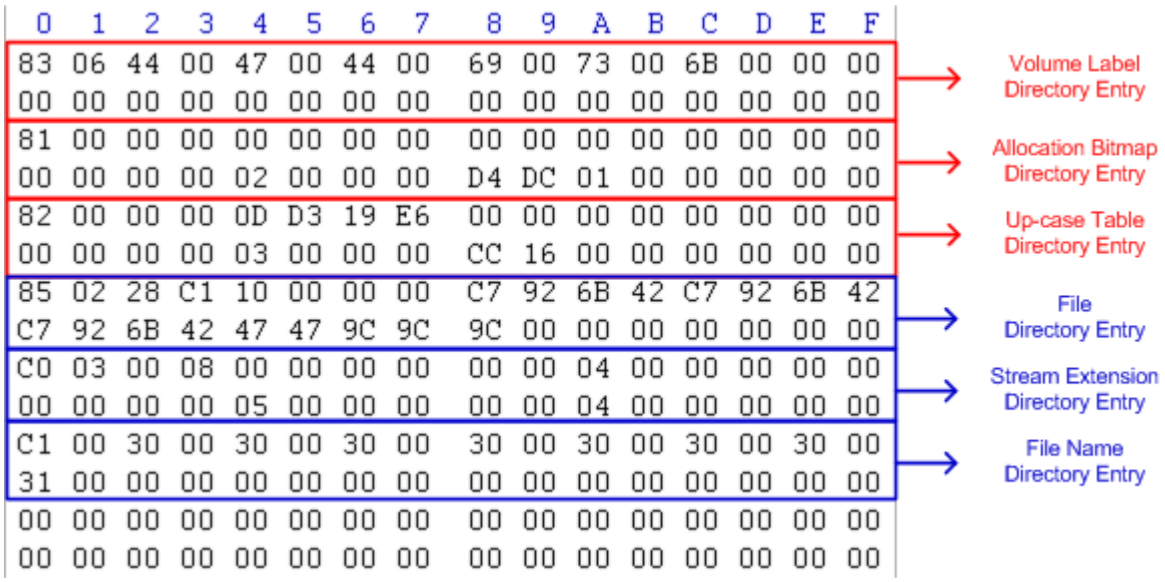

Figure 1-5 Example of Root Directory

Entry Type	Definition	Description
81h	Allocation Bitmap	Define start cluster number and valid length of Allocation Bitmap Table.
82h	Up-case Table	Define start cluster number and valid length of Up-case Table.
83h	Volume Label	Define label as Unicode string for this volume. Entry type is set to 0x03 if Volume Label is not defined.
85h	File	Describe files and directories, store created/modified time, and define the following secondary entry count for this file/directory.
C0h	Stream Extension	Store name length, start cluster number, and valid length of this file/directory.
C1h	File Name	Store the name of this file or directory. One entry can store up to 15 Unicode string, so the maximum number of File Name Entry for each file/directory is 17 to support 255-Character name length.

Table 1-3 FAT Entry Value

The Allocation Bitmap, Up-case Table, and Volume Label Directory are only found in Root directory, while File, Stream Extension, and File Name are found in all directories.

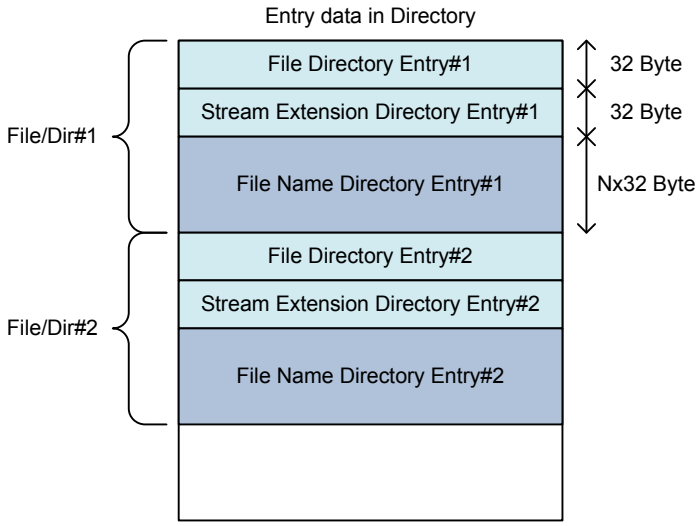

Figure 1-6 Example of Directory

As shown in Figure 1-6, when one file or directory is created, one File Directory Entry, one Stream Extension Directory Entry, and up to 17 File Name Directory Entry will be generated in parent directory. The descriptions of the entries are shown in Figure 1-7 to Figure 1-9.

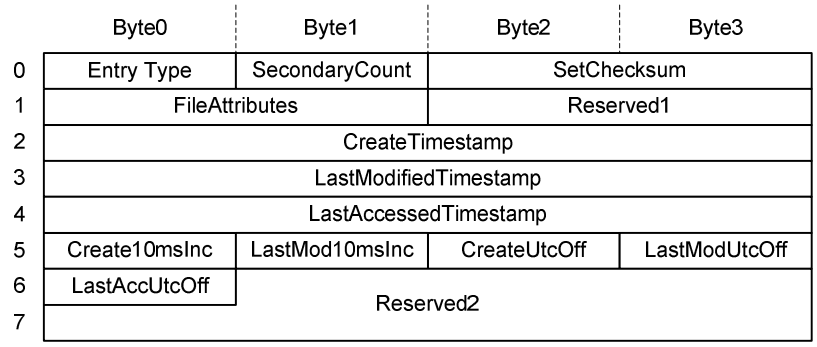

Figure 1-7 File Directory Entry

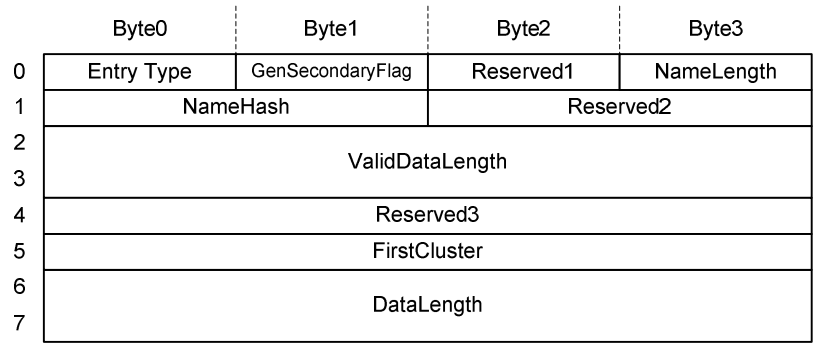

Figure 1-8 Stream Extension Directory Entry

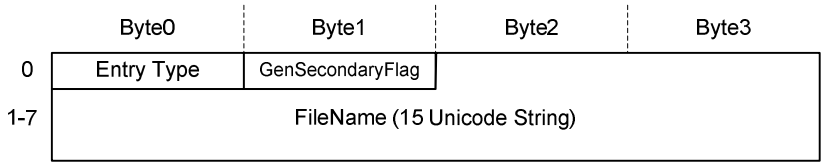

Figure 1-9 File Name Directory Entry

- SecondaryCount in File Directory Entry is the total count of Stream Extension and File Name Directory Entry.
- DataLength in Stream Extension Directory Entry is file/directory size in byte unit.
- FirstCluster in Stream Extension Directory Entry is the first cluster number of file/directory.

2 Hardware description

exFAT demo uses same hardware system with 1-ch SATA host demo reference design. So, please see more details from reference design manual on each hardware platform, “dg_sata_ip_refdesign_host_xxx_en.doc”. The difference between exFAT demo and 1-ch SATA host demo is only CPU firmware.

3 Software description

exFAT software operates with SATA Device by using low-level function from 1-ch SATA host demo reference design to interface with SATA-IP. In reference design, it consists of three codes, i.e.

- “sata_host_ctl.c/.h”: stores low-level function to operate with SATA-IP, such as write, read command.
- “sata_host.h”: stores parameter definition to support multi hardware platform.
- “sata_exfat.c”: exFAT function and main console for the demo to format disk, delete file/directory, create file/directory, read file.

The details of exFAT design are follows.

In the demo, the maximum transfer size of each write/read transaction is equal to Boundary Unit which will be found during operating format command while File and Directory size will be access by Cluster unit. Following Table 1-1, write/read function in “sata_host_ctl” need to support the transfer size up to 64 MB (BU size). The demo can support 8 GB – 2 TB disk capacity.

Similar to 1-ch SATA host demo, DDR3 is used for data buffer transferring between CPU and SATA-IP. TX/RX_SATA_FIS area stores non-Data FIS packet while DATA_FIS area stores Data FIS packet, as shown in Figure 3-1. Two additional areas are designed, i.e. TMP_DATA to store temporary data and TEST_PATTERN to store test pattern data for creating and verifying file.

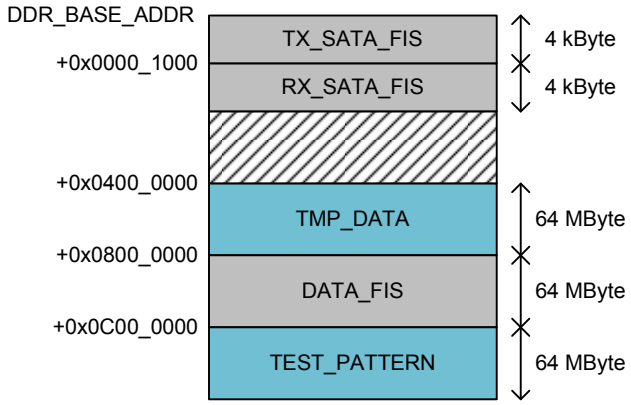

Figure 3-1 DDR Memory Map

The demo is designed to support five commands, i.e. format, create file/directory, read file/directory, delete, and change current directory. The details of each command are follows.

- Format

The sequence to format SATA device by exFAT file system is follows.

(1) Get disk size and set Cluster and Boundary Unit size, following Table 1-1.
(2) Prepare 1 sector of Master Boot Record data to TMP_DATA area and call function to write the data to SATA Device at sector#0.
(3) Prepare 12 sector of Boot Region data to TMP_DATA area and call function 2 times to write the data to SATA Device at address = BU and BU + 12 for backup.
(4) Prepare BU/2 size of FAT table data to TMP_DATA area and call function to write the data to SATA Device at address = 3*BU/2.
(5) Prepare Allocation Bitmap table, Up-Case Table, and Root Directory to TMP_DATA area and call function to write 3 consecutive cluster to SATA Device, starting cluster address = 2.
(6) Now File System Layout will be similar to Figure 1-1.

- Create File/Directory

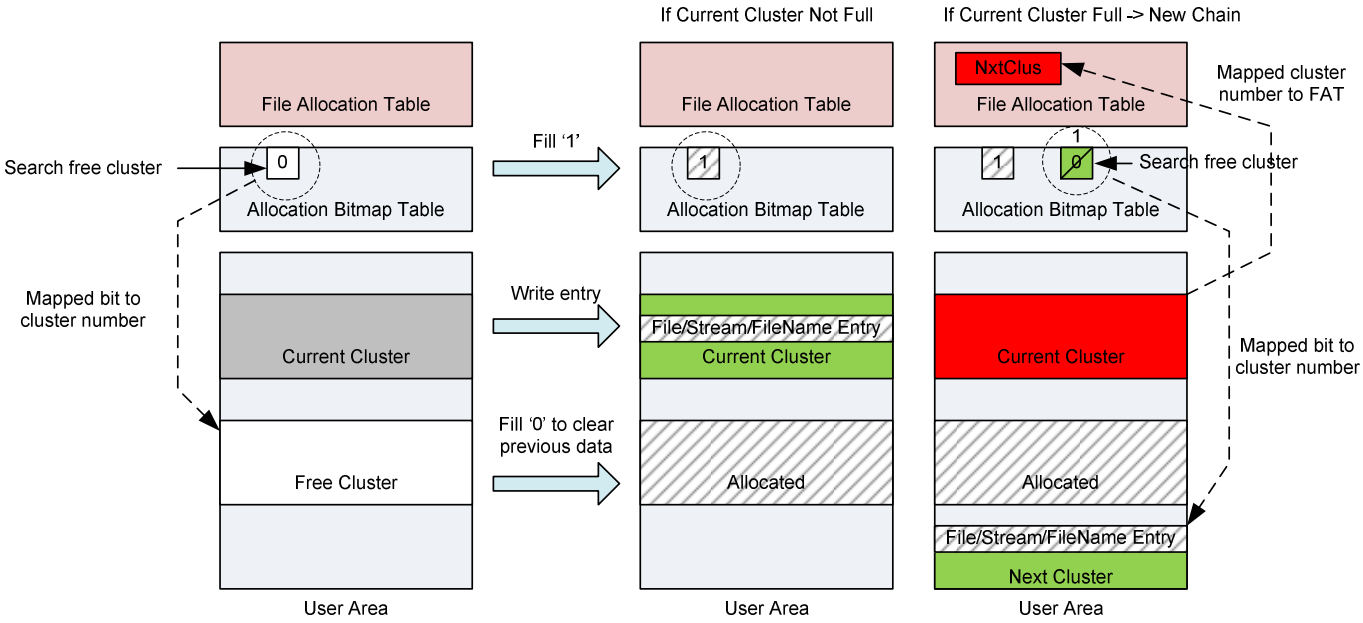

Figure 3-2 Create Directory Sequence

The sequence to create directory is follows.
(1) Search one free cluster available from Allocation Bitmap Table (Cluster address = 2).
(2) Set that bit = '1' to allocate cluster in Allocation Bitmap Table.
(3) Read the entry in current directory to search available space to add File Directory Entry in current directory.
(4) If available area in current cluster is not enough, search another free cluster and then create FAT chain in current directory by writing next free cluster to FAT table.
(5) Add File, Stream Extension, and File Name Directory Entry to the available space and call function to update entry in current directory.

To create file, the sequence is follows.
(1) Prepare test pattern data to TEST_PATTERN area.
(2) Search free cluster available from Allocation Bitmap Table. Free cluster size must be equal to file size from user input.
(3) a. If available space is not fragmented, data from TEST_PATTERN area will be burst transfer to SATA device until complete. Allocation Bitmap will be filled to '1' to allocate the filled area.
b. If available space is fragmented, one cluster data will be transferred for each transaction until complete. FAT chain is created by filling FAT table by next cluster number in chain. One bit of Allocation Bitmap will be filled to '1' for each transaction to allocate one cluster.
(4) File, Stream Extension, and File Name Directory Entry will be added to current directory.
(5) Similar to create directory flow, if current cluster has not enough space to add entry, new FAT chain will be created.

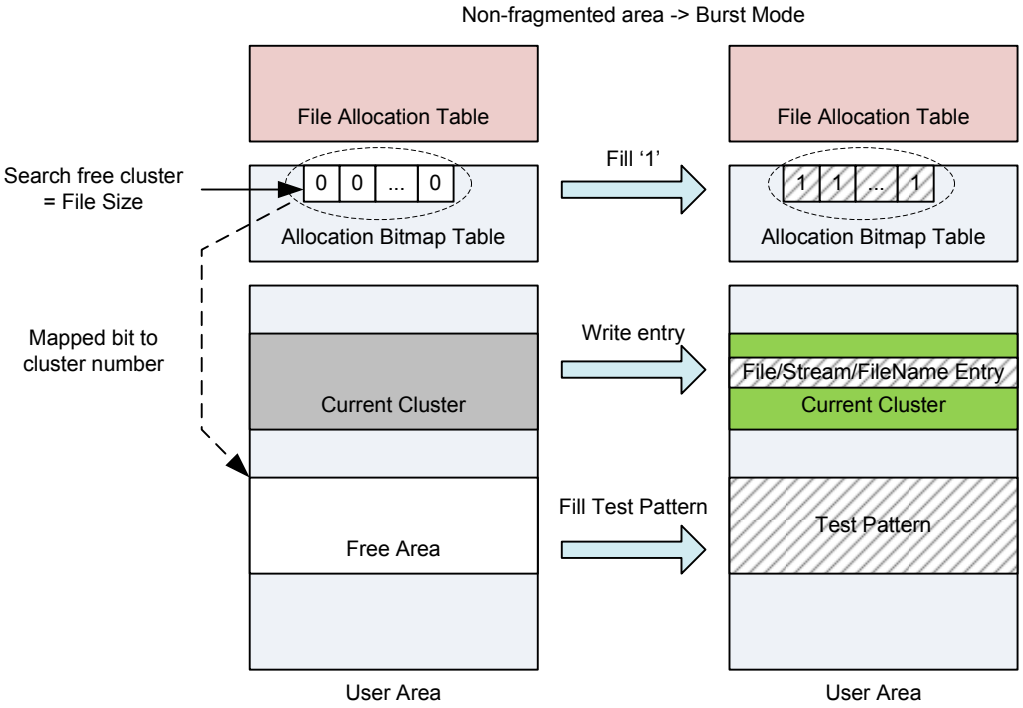

Figure 3-3 Create new File for Non-fragmented area

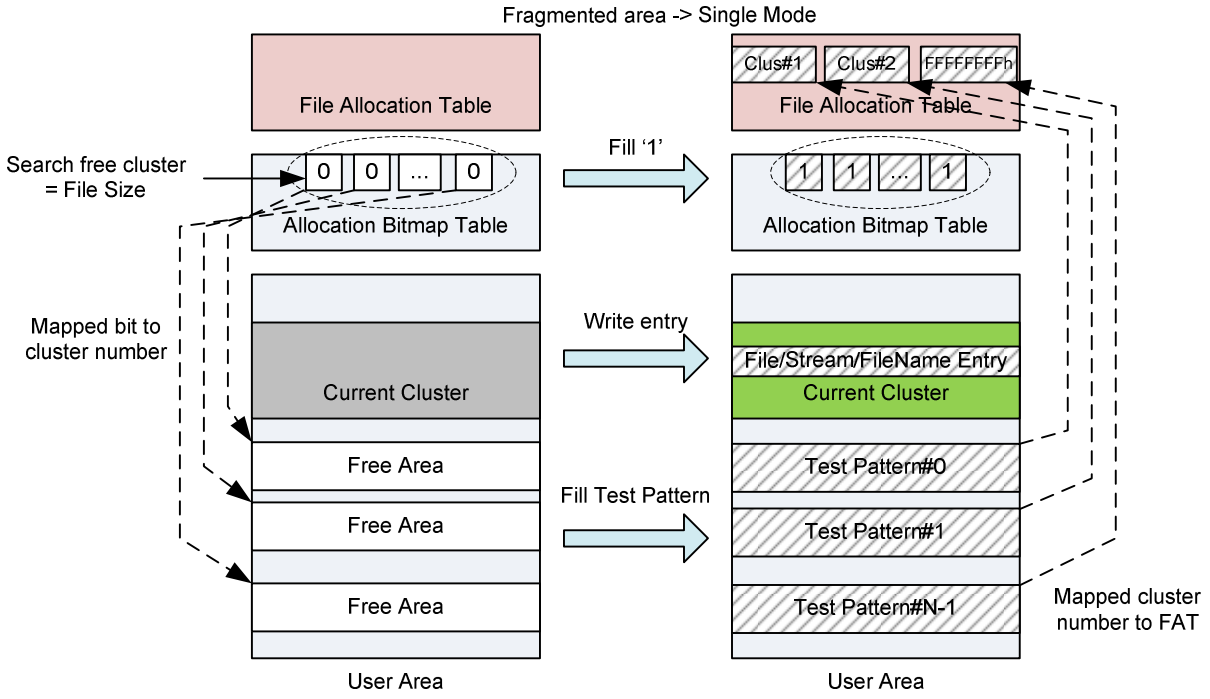

Figure 3-4 Create new File for Fragmented area

- Read File/Directory

The sequence to dump directory list within current directory is follows.
(1) Dump all data in current cluster to DATA_FIS.
(2) Scan all entries in current directory that which entry is valid and then print out only valid file/directory until end of cluster.
(3) End operation if current cluster is end and no more FAT chain. If FAT chain is found, read next cluster address from FAT table and then go back to step (1).

The sequence to read file is follows.
(1) Dump all data in current cluster to DATA_FIS.
(2) Receive file name input from user and then search entry in current cluster which file name is matched.
(3) End operation if file name not match while current cluster is end cluster and no more FAT chain. Continue to search in next cluster if FAT chain is found.
(4) Read first cluster information from matching file directory entry and dump data from first cluster to DATA_FIS.
(5) Start dumping data from DATA_FIS to user console until exit command detect or start reading/verifying data until end of file.

- Change Current Directory

The sequence to change to sub-directory is follows.
(1) Receive directory name from user and search until directory name matching, similar to Step (1) – Step (3) of read file sequence.
(2) Change current cluster of system to the first cluster value of matching file directory entry. Now user can access all entries within new directory.

Note: The demo can support up to 15 sub-directory levels from global parameter limitation.

To change directory to parent directory, it will read cluster address of parent directory from global parameter in firmware and then change to new cluster address value.

- Delete

The sequence to delete file is follows.
(1) Receive file name from user and search until file name matching, similar to Step (1) – Step (3) of read file sequence.
(2) Delete file entry from parent directory by changing entry type from 85h, C0h, and C1h to 05h, 40h, and 41h (Bit[7] is changed from '1' to '0').
(3) Write modified entry from DDR buffer to SATA Device.

The sequence to delete directory is follows.
(1) Receive directory name from user and search until directory name matching, similar to Step (1) – Step (3) of read file sequence.
(2) Change current cluster of system to first cluster value of matching file directory entry, and delete each entry within that cluster until end of cluster and end of FAT chain. If entry of sub-directory is found, current directory will be changed to the sub-directory for deleting all internal entries. After that, go back to continue delete the entry of directory.

- Necessary consideration

- exFAT software source code of this design is designed to be reference code and not include error check or recovery from illegal/unexpected behavior.
- The demo can support up to 15 sub-directory levels. User needs to modify firmware for additional supported level.

Figure 3-5 shows reference design operation result on serial terminal screen.

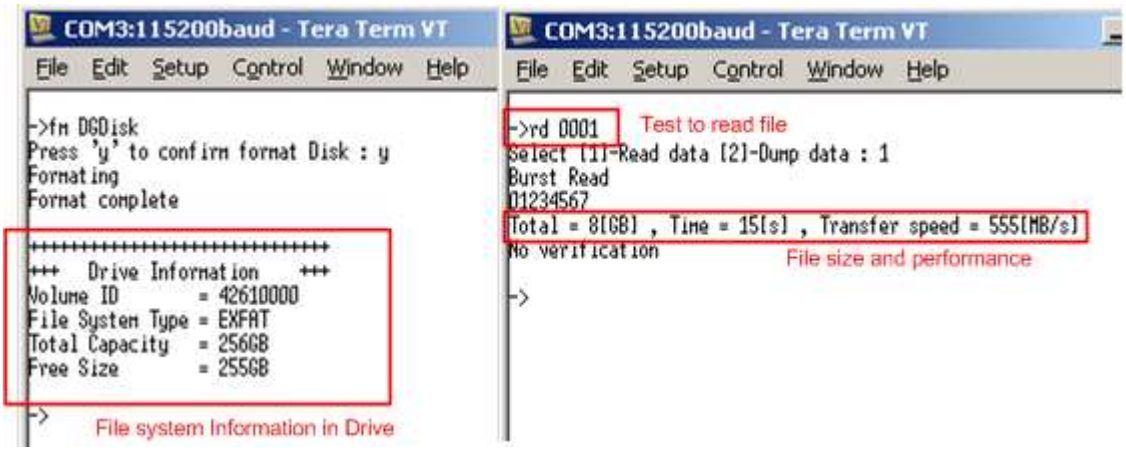

Figure 3-5 Operation result sample screen

4 Revision History

Revision	Date	Description
1.0	14-Mar-13	Initial release
1.1	5-Sep-13	Update to support multi h/w platform