OITDA規格
Standard

光産業技術振興協会規格

Standard of Optoelectronics Industry and Technology Development Association

# 再配置を少なくするファイル配置方策

（File Allocation System with minimized reallocation）

OITDA DC02：2013

第 1 版

制定　2013 年 3 月

審議委員会
光ディスク標準化委員会
Optical Disk Standardization Committee

発行：一般財団法人光産業技術振興協会
Optoelectronics Industry and Technology Development Association（JAPAN）

# 目　次

# まえがき

ファイルシステムは数多く存在するが，領域割付手法がクラスタ（セクタの整数倍のサイズ）のリスト又はエクステント（それぞれがセクタの任意の倍数のサイズ）のリストであるため，ファイルの生成及び削除を繰り返していると，ディスク上で一つのファイルが分断して配置される率が高くなり，結果的には多くのファイルが分断（fragment）状態になり，ファイルシステムの性能劣化を招くことが長年の課題であった。

現状のこの課題の解決方法は，脱分断（defrag）ツールを利用して分断状態の多くのファイルを再配置（reallocate）して連続領域（continuous area）に再配置し，分断状態から一つ一つのファイルを解放することである。

脱分断ツールは時間が掛かり，その間仕事ができなくなるのが課題であり，分断状態をなるべく起こさない領域割当手法が求められていた。

CoPo2（Concatenation of Power of 2）はこの課題の解決方法の一つであり，規格化に値するとの認識で，規格を制定することにした。

この規格の一部が，技術的性質をもつ特許権，出願公開後の特許出願又は実用新案権に抵触する可能性があることに注意を喚起する。一般財団法人光産業技術振興協会は，このような技術的性質をもつ特許権，出願公開後の特許出願及び実用新案権に関わる確認について，責任はもたない。

この規格に関して，ご意見・情報がありましたら，下記連絡先にお寄せください。

連絡先：一般財団法人光産業技術振興協会標準化室
e-mail：opt-st@oitda.or.jp

OITDA 規格

OITDA
DC02：2013

# 再配置を少なくするファイル配置方策

File Allocation System with minimized reallocation

## 序文

この規格は，CoPo2（Concatenation of Power of 2）を用いた UDF のファイルシステムについて規定する。これによって，多数回書き換えを行なったときに生じるファイルの分断状態が起こる確率を減らし，ファイルの再配置を少なくできる。この結果，従来の脱分断ツールの課題であった実行時間が長くなるという欠点を解消，効率の良いファイル管理を実現することができる。

## 1　適用範囲

この規格は，HDD に組み込むことを主眼として，CoPo2 を適用するための UDF のデータ構造の細部を規定する。

この規格は，次の項目を規定する。

－　CoPo2 領域配置方式
－　区画管理との関係
－　当該方式を実現するための管理表
－　当該方式を構成するのに必要な機能
－　ディスク割付の管理のために必要な多層区分割付表の大きさの概要
－　UDF への適用
－　管理データ構造の確保方針
－　UDF への適用データ構造

## 2　引用規格

次に掲げる規格は，この規格に引用されることによって，この規格の規定の一部を構成する。これらの引用規格は，その最新版（追補を含む。）を適用する。

**JIS X 0607**　非逐次記録を用いる追記形及び書換形の情報交換用媒体のボリューム及びファイルの構造

**JIS X 0611**　ユニバーサルディスクフォーマット（UDF）2.01

## 3　記法

この規格は，**JIS X 0611** に規定された記法に従う。

## 4　用語及び定義

この規格で用いる主な用語及び定義は，次による。

**4.1**

**区画**（partition）

ディスク領域を分割してファイルシステムに割り当てられた，連続領域。

**4.2**

**仮想包含区画**（virtual container partition）

実際に割り当てられている区画領域を包含する最小の，区画の割当単位としての基本割当領域の 2 の巾乗で表現できる，区画。

**4.3**

**バディ方式**（buddy）

領域割当手法，基本割当領域の 2 の巾乗で表現できる一つの領域を 2 分割してできるバディ（兄弟）領域を基本割当領域の大きさまで分割してできる領域群を割当領域の候補として検索，要求領域の大きさを包含する最小の空のバディ領域を割り当てる領域割当手法。

**4.4**

**区分けする**（くわけする）（to partition）

区画が基本割当領域の 2 の巾乗で表現できる区画の場合は，これを一つの主区分（区画を主として構成する区分）からなる主区域としてこれにバディ方式で順次 2 分割し，それぞれを区分として分割する。2 の巾乗で表現できない場合は，区画のサイズを構成する 2 進数のビットがたっている大きさの領域を主区分として連結した主区域からなるものとして区画が構成されているとみなす主区分けを実施して，各主区分にバディ手法によって順次 2 分割を実施して，区分として分割するか区画の実領域は主区域としてとらえ，これから仮想包含区画を構成してこれに 2 の巾乗で表現できる区画と同様の分割による区分分割を実施する操作。

**4.5**

**主区分け**（しゅくわけ）（master partitioning）

区画を区分けする最初の段階で区画を主として構成する区分を同定する処理。

**4.6**

**主区分**（しゅくぶん）（master divided-partition）

主区分けによって同定された区分。

**4.7**

**区分**（くぶん）（divided-partition）

主区分を順次２分割して得られた分割領域。

**4.8**

**区分層**（divided-partition level）

当該区分の大きさの層を基本割当領域と比較した時の２の巾乗値を層値として表現した値。

**4.9**

**区分対**（divided-partition pair）

区分を 2 分割した時に得られるお互いにバディである区分の対。

**4.10**

**区分番号又は区番**（divided-partition number）

区分け時に，上位層の区分から順次個別に付番される，区分の番号。

**4.11**

**主区分番号**（master divided-partition number）

主区分の番号。

**4.12**

**区画構成主区分表**（master divided-partitions table）

区画を構成する主区分を管理する表。

**4.13**

**主区分番号管理表**（master divided-partition number management table）

区画を構成する主区分の主区分番号を各区分層に対応させて管理する表。

**4.14**

**末尾番号管理表**（end position management table）

各区分層の末尾となっている区分位置の区分番号を管理する表。

**4.15**

**最上位区分層管理表**（highest divided-partition level management table）

区画を構成する区分の最上位の区分層を管理する表。

**4.16**

**多層区分管理関連表**（multilevel-divided-partition management tables）

多層的に層分けされた区分構成を管理するための複数の表で，主区分番号管理表，末尾番後管理表，最上位区分層管理表からなる。UDF では CoPo2 管理表として参照。

**4.17**

**多層区分割付表**（multilevel-divided-partition allocation table）

多層的に層分けされた，各区分層の各区分の割付状態を 2 ビットで管理した表。

**4.18**

**区域**（segment）

大きさが基本割当領域の 2 の巾乗(1 区分区域)の場合は，該当する層の区分で構成される領域，大きさが基本割当領域 2 の巾乗の多項式の和（大きさが 2 進数表現で複数のビットが 1 である，多層区分区域）で構成される場合には，これを包含する最小の基本割当領域の 2 の巾乗の区分から，多層区分け（多項式の各項の大きさの区分を大きい順から切り出し）で切り出した複数の区分を連結して構成される連続領域。

**4.19**

**多層区分け**（multilevel-partitioning）

多層区分区域を獲得する手続きで，これを包含する最小の，基本割当領域の 2 の巾乗の区分から，多項式の各項の大きさの該当区分を多層的な区分層において大きい順から順次切り出してこれを対象として複数の区分を連結した区域として多層区分区域を獲得する操作。

**4.20**

**包含区分**（minimum-divided-partition containing a multilevel segment）

多層区分区域を包含する最小の区分。

**4.21**

**隣接多層区域**（contiguous multilevel segment）

包含区分から多層区分区域を切り出した際に包含区分内に残った隣接する区域で，多層的に小さい順から大きい順の区分が連結して並ぶ区域。

**4.22**

**1 次割付**（first-pass allocation）

区域を獲得する時に，要求区域を包含する区分を 1 次的に割付ける操作。1 区分区域の場合はこれで割付が終わる。多層区分区域の場合は 2 次割付が必要である。

**4.23**

**2 次割付**（second-pass allocation）

多層区分区域を獲得する時に，1 次割付区分から，多層区分けをして，多層区分区域を割付ける操作。

**4.24**

**仮割付**（provisional allocation）

区域を獲得する時にこれを包含する区分が見つからない場合に，より上位の層の空区分を探索し，探索結果区分を仮割付区域として獲得する操作。

**注記** 後続する処理で，仮割付区域から 1 次割付区域を獲得し，多層区分区域の獲得の場合は，更に 2 次割付を実施して要求区域を獲得する。

**4.25**

**割付状態**（allocation status）

区分を新規に区域として割付けることが可能かどうかを示す状態で，空，使用中及び保留がある。

**4.26**

**空**（available）

新規区域を割付可能な状態。空 1 及び空 2 がある。空探索では空 1 を最初に探索して，見つからない場合に空 2 を探索する。

**4.27**

**空 1**（first-pass available）

初期状態では空として空 1 を採用する。使用中が解放された場合には空 1 を状態割付する。

**4.28**

**空 2**（second-pass available）

多層区分区域を割付けた際に発生する多層隣接区域に空 2 を状態割付する。

**4.29**

**使用中**（occupied）

当該区分が使用中である。

**注記** 使用中が連続する多層区分区域が，ファイルシステム上で同一のファイルに割り当てられているがどうかは保証することができない。

**4.30**

**保留**（reserved）

区分の状態を 4 状態に集約した場合に，空（空 1，空 2），使用中に属さない，状態。

**注記** 空，使用中の区分の下位の層の区分は保留にする。ある区分を切り出して，下位の区分に使用中の状態を割付ける場合，当該区分及び中間に存在する下位に使用中を含む区分は保留にする。

## 5 CoPo2（Concatenation of Power of 2）適用の前提事項と基本的な考え

### 5.1 前提条件

HDD に UDF を実装する上での前提条件は，次の通りとする。

**a)** 1 つのディスク・ボリュームは，ディスクに適用される区画管理システムによって，それぞれ独立の

連続領域として分割された，ディスクの区画（partition）から構成されるものとする。

**b)** 1 つの区画は，セクタサイズの整数倍であるクラスタサイズの整数倍のサイズのディスク上の連続領域である。

### 5.2 CoPo2 の基本的な考え

CoPo2 は，1 つの区画に一つのファイルシステムを適応するときに，ファイルシステムの要求する記憶領域に対し，なるべく大きな連続領域を配置できる領域配置方式を導入する。これによって，分断を起こりにくくした領域配置を常に適用し，ファイルの分断状態が起こる確率を減らし，ファイルの再配置を少なくすることができる。

連続領域を配置する伝統的方式はバディ（buddy）方式である。バディ方式は，2 進数として与えられた要求クラスタ数を確保する場合，クラスタ数を包含する最小の 2 の巾乗の空きクラスタ数を確保し，余った領域を遊ばせることで連続領域を確保する。この余った領域のことをスラック（slack）と呼び，これは無駄な領域となる。この無駄は，要求領域サイズが大きいほど大きくなる。このため，従来のバディ方式は小メモリシステムのメモリ管理では利用されることがあったが，ファイルシステムで適用されることはあまりない。

CoPo2 方式では，バディ方式の弱点であるスラックを新規の領域割り当てとして再活用することで，バディ方式の欠点を解消し，更にファイルの分断状態が起こる確率を減らしたファイルシステムを提供することができる。

## 6 CoPo2 領域配置方式

### 6.1 CoPo2 の基本構造

バディ（buddy）方式では，二進数で 1000 のクラスタサイズの割当可能な領域は以下のような 2 の巾乗サイズの符番が可能な区分（divided-partition）の区分ツリーで表現可能である:

```
                          (#0,#1)                                                   区分層 3(サイズ=1000)
          (#0,#2)                                   (#1,#3)                         区分層 2(サイズ=0100)
    (#0,#4)       (#1,#5)           (#2,#6)           (#3,#7)                     区分層 1(サイズ=0010)
(#0,#8) (#1,#9) (#2,#10) (#3,#11) (#4,#12) (#5,#13) (#6,#14) (#7,#15)             区分層 0(サイズ=0001)
```

ここで，( )の中は（#同一区分層内区番，#区番）である。ここで区番は区分番号の省略形である。#を付けて番号であることを表現している。

このような状態で，0101 のサイズの区域（segment）が必要な場合，区番 2（区分番号 2）の区分すなわち区分 2 と区分 12 とを確保すれば良い。なおバディ方式をディスクの領域管理に適用した場合はこのようなことが出来ないので，丸め上げた 1000 の区分である区分 1 を確保すれば良い。実際には区分 2 と区分 12 が 0101 の領域として活用され，スラックとなる余りの区分 13 と区分 7 は有効活用が出来ない。

バディ方式の領域管理では，セクタの整数倍であるクラスタを基本割当単位とし，任意のクラスタ数の管理リストをクラスタの空と使用中を 1 ビットで管理したビットマップテーブルで支援してきた。しかし，スラック領域を活用可能にするためには，各クラスタの空・使用中を管理する 1 ビットのビットマップテーブルだけで管理するのはできないので，各区分の状態を管理できるようにする必要がある。そこで，2 ビットで管理できる手法を示す。

2 ビット表現で区分の状態を表現するための指標は，区分はツリー構成で，上下関係の中で互いを制約

しているので，このことを考慮する必要がある。

当然，空及び使用中の指標が必要となるが，初期状態では，区番 1 の区分が唯一空で，領域割当対象となり，自立的に割当対象にすることが出来る。他の区分はこれに従属しており，領域割当の対象ではない。一方，0101 の区域を割り当てると，区分 1 は，下位の区分に主体性をゆずり，区分 2 と区分 12 を使用中にして両区分を連結してサイズ 0101 の区域を確保し，この結果，関連上下区分も状態が変化する。この結果スラック領域となった区分 13 と区分 7 はスラック域としての多層隣接区域にある空である。

このことから，実用性を考慮し，2 ビットに区分状態を

**a)** 割当対象としての空 (0) については，空 1: (00) とスラック部分を空として利用可能とする空 2: (01) の状態に振り分け

**b)** 割当対象にできない塞(1)状態については，自立的に割当済みとなっている使用中: (11) と，上下の割付状態に従属して塞がっている意味で割付処理に対しては保留: (10) という状態に振り分るとすれば

空: (0)　通常の空: (0)　=空 1: (00)

隣接多層区域における空:(1)　=空 2: (01)

塞: (1)　従: (0)　=保留: (10)

自: (1)　=使用中: (11)

の 4 状態に縮退する。

以下に，区画割当の初期状態を示す。

(#0,#1;00)　区分層 3(サイズ=1000)

(#0,#2;10)　(#1,#3;10)　区分層 2(サイズ=0100)

(#0,#4;10)　(#1,#5;10)　(#2,#6;10)　(#3,#7;10)　区分層 1(サイズ=0010)

(#0,#8;10) (#1,#9;10) (#2,#10;10) (#3,#11;10) (#4,#12;10) (#5,#13;10) (#6,#14;10) (#7,#15;10)　区分層 0(サイズ=0001)

上記区画の初期状態では，区分 1 が空 1 : (00) で，他の区分は保留: (10) 状態になる。この状態で 0101 のサイズの区域を割り付けると，以下に示すように区分 1 は保留: (10) で，区分 2 は使用中: (11) に，その下位区分は変化無し，区分 3 は変化無しで，区分 12 は使用中: (11)，区分 7 と区分 13 は空 2: (01)，区分 14 と 15 は変化無しとなる。

(#0,#1;10)　区分層 3(サイズ=1000)

(#0,#2;11)　(#1,#3;10)　区分層 2(サイズ=0100)

(#0,#4;10)　(#1,#5;10)　(#2,#6;10)　(#3,#7;01)　区分層 1(サイズ=0010)

(#0,#8;10) (#1,#9;10) (#2,#10;10) (#3,#11;10) (#4,#12;11) (#5,#13;01) (#6,#14;10) (#7,#15;10)　区分層 0(サイズ=0001)

### 6.2　区画管理との関係

領域割当のベースは区画であるが，区画は一般的にクラスタ数の 2 の巾乗ではない。これにバディ方式を適用するにあたっては，実区画活用方式と仮想包含区画方式の 2 つがある。

#### 6.2.1　実区画活用方式

区画の大きさが 1011 のクラスタ数で表現されるとする。これはバディ方式では表現できない。バディ方式との整合性を考慮し, この区画は 1000 (8) と 0010 (2) と 0001 (1) のクラスタ数のバディ区分に区分けして連結してできた区画とする。この区画構成は，この 2 進数表現で 1 の立った区分が区画を主に構成する区分ということで主区分（master divided-partition）と呼ぶ。

この区画構成は，区画構成主区分構成表ということで，区画サイズのクラスタ数の二進数表現で構成する。本例では 1011 である。

下記に示す例では，区番 1 ((#1))，区番 8 ((#8))と区番 19 ((#19))が主区分である。

区画サイズ(11=8+2+1)　((8　　)) ((2　　)) ((1 ))
区分層 3(サイズ 8)　　((#1　　))
区分層 2(サイズ 4)　　(#2　　) (#3　　)
区分層 1(サイズ 2)　　(#4　　) (#5　　) (#6　　) (#7　　) ((#8　　))
区分層 0(サイズ 1)　　( #9) (#10) (#11) (#12) (#13) (#14) (#15) (#16) (#17) (#18) ((#19))

区番については，ここでは区分層の大きい方から 1 を開始番号として追番で，符番をしている。

各区分層と主区分との関係を管理するために主区分番号管理表を作成し, 各区分層にその値を設定する，主区分が存在する層には当該主区分番号を設定し，該当区分が存在しない区分層には該当しないことを表現するために－1 を設定する。

上記例では区分層数が 6 なので (-1,-1,1,-1,8,19) と設定する。

各区分層の先頭区分番号も管理する必要があり，先頭番号管理表を作成して，各層に当該先頭番号を設定する。上記例では， (-1,-1,1,2,4,9) と設定する。

総区分数については，総区分数管理表で管理する。上記で示した例では，19 を設定する。

全多層区分の割付状態の管理は，多層区分割付表で管理する。下記で“/”は層の区切りである。上記例では， (00/10,10/10,10,10,10,10,00/10,10,10,10,10,10,10,10,10,10,00) を最初に設定する。
主区分は空 1 を設定し，他は保留を設定する。

区分層の大きい区分の下位層の右隣の各サイズの区番は, 符番方式を決めた各種管理表を用いることで, 計算で求めることができる。

### 6.2.2　仮想包含区画活用方式

仮想包含区画活用方式は，与えられた実区画を包含する最小の 2 の巾乗の仮想包含区画を考え，これにバディ方式の符番を適用する方式である。符番方式が実区画活用方式と異なるだけで，領域管理をバディ方式に適応したのと同じである。

区画サイズ (11=8+2+1)　((8　　))((2　　))((1 ))からなる実区画から，下記に示す仮想包含区画を構築，これを構成する仮想区分に区番 1 を符番して，以下実区分にも仮想区分にも，同様のバディ方式の符番を適用する。

区分層 4(サイズ 16)　(#1　　仮想包含区画を構成する仮想区分　　)
区分層 3(サイズ 8)　((#2　　))　　#3 が仮想区分
区分層 2(サイズ 4)　(#4　　) (#5　　)　　#6,#7 が仮想区分
区分層 1(サイズ 2)　(#8　) (#9　) (#10　) (#11　) ((#12　)) #13,#14,#15 仮想区分

区分層 0(サイズ 1)　　(#16)(#17)(#18)(#19)(#20)(#21)(#22)(#23)(#24)(#25)((#26))

サイズ 11 の区画にて当該方式を適用した場合について，**図 1** にその割り当て様子を示す。

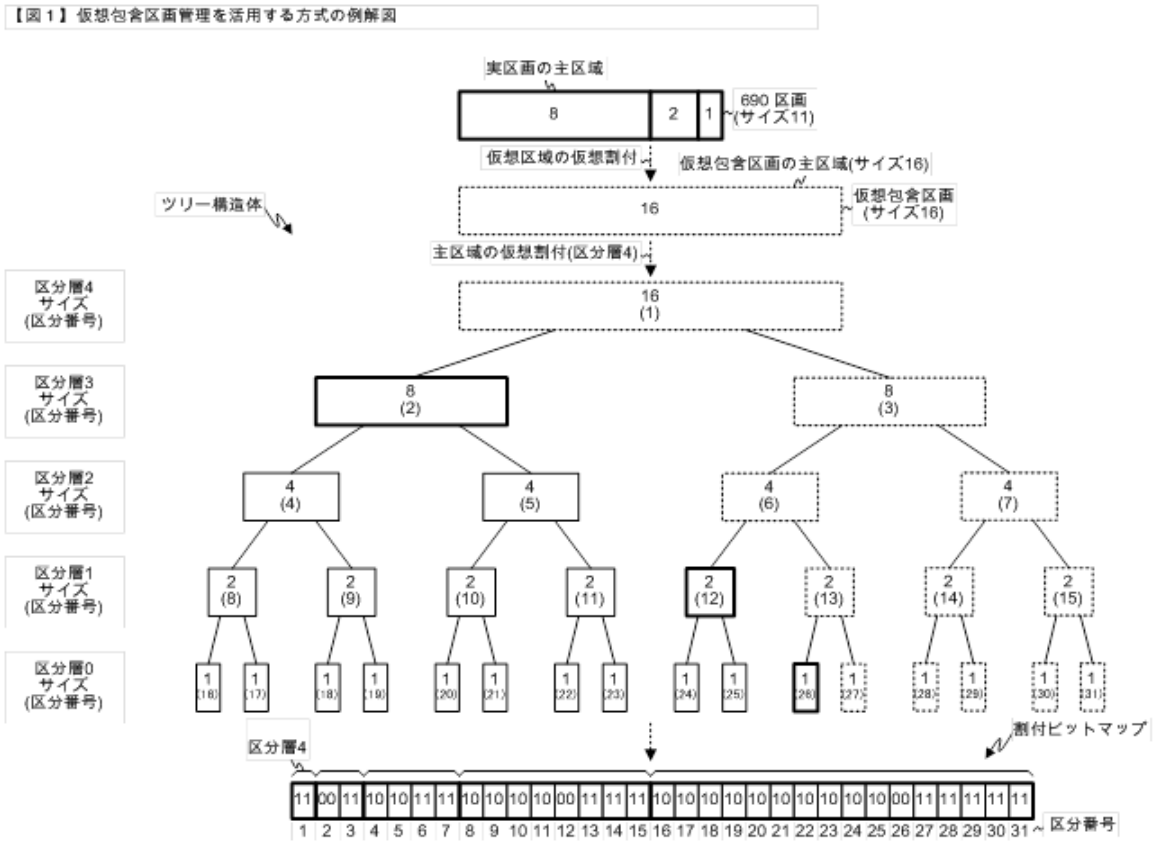


**図 1ー仮想区画管理を活用する方式の例**

### 6.2.3　仮想包含区画活用方式

この規格は，バディ方式の自然な拡張である，仮想包含区画管理方式を採用する。

## 6.3　当該方式を実現するための管理表

この規格で必要な関連管理表とその役割の例を**図 2** に示す。

必要な関連管理表は下記に示す。

**a)**　区画サイズから構成する，区画構成主区分表（T1）

**b)**　多層区分け構造を管理するための多層区分管理表（T2）

**1)**　主区分番号を管理する主区分番号管理表（T2a）
　管理表中に示す“-1”は，割当てがないことを意味する。

**2)**　各区分層での末尾区分番号を管理する末尾区分番号管理表（T2b）
　管理表中に示す“-1“は，割当てがないことを意味する。

**3)**　最上位区分層を管理する最上位区分層管理表（T2c）

**4)**　区分層毎の割付状態を管理する多層区分割付表(T3)。
　割付状態は空 1: (00)，空 2: (01)，保留: (10)，使用中: (11)

【図2】本規格に必要な関連管理表とその役割を例解する図

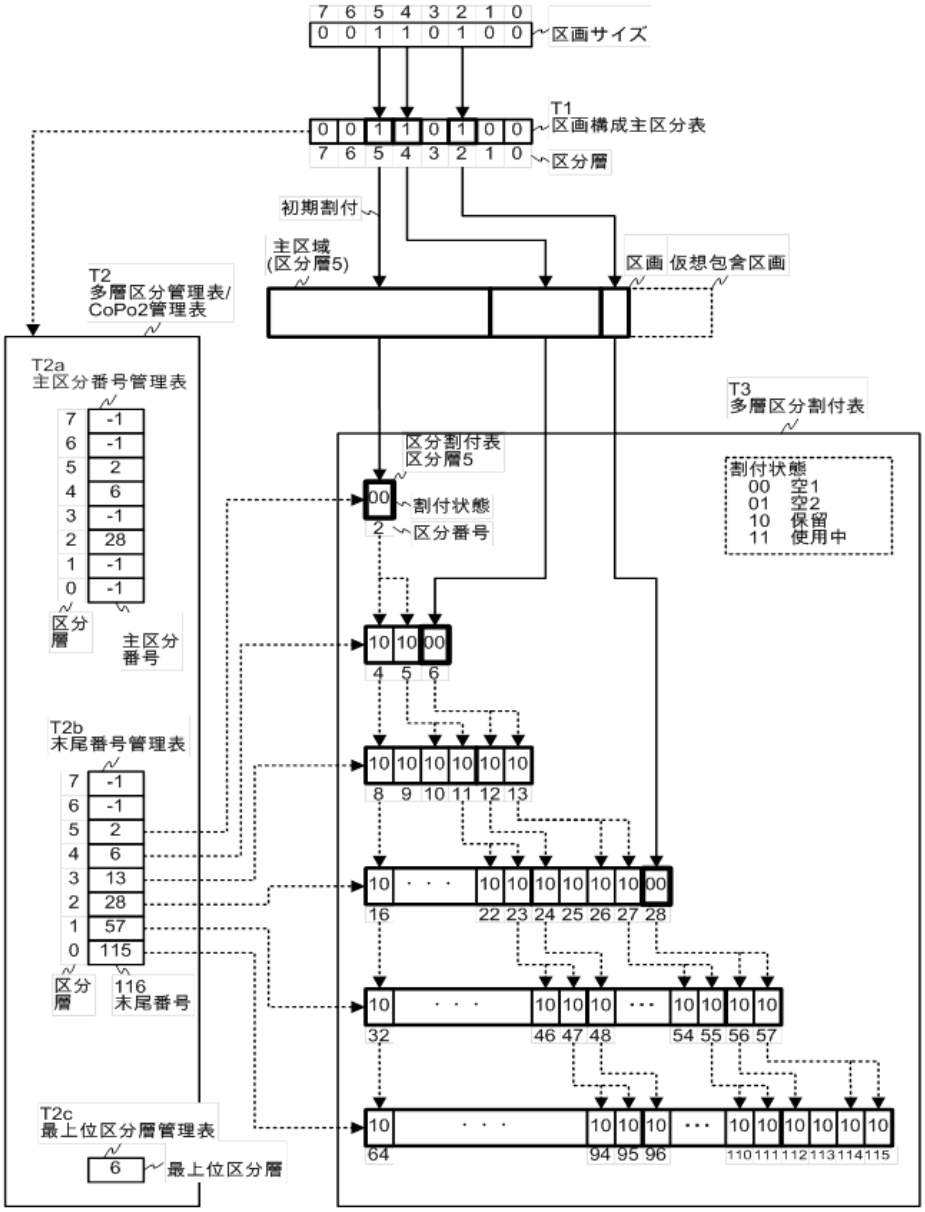


**図 2ーこの規格に必要な関連管理表とその役割**

### 6.4 当該方式を構成するのに必要な機能

本方式を実装するには，下記のようなモジュール部品が必要である。

#### 6.4.1 初期化部

##### 6.4.1.1 区画サイズ取得部

OS から区画サイズを取得して区画構成主区分表を作成する。

##### 6.4.1.2 多層区分割付表生成部

空区分は対象層の区分の空 1 を探して，無い場合には対象層の空 2 を探す。

#### 6.4.2 多層区分管理部

#### 6.4.2.1　空区分探索部

空区分は対象層の区分の空 1 を探して，無い場合には対象層の空 2 を探す。

#### 6.4.2.2　区域割付部

区域割付要求を受けて，区域を包含する空区分を探索する。

**a)** 見つかった場合には，1 次割付処理をする。

**1)** 1 区分区域の割付の場合には，見つかった区分を返却する。

**2)** 多層区分区域の割付の場合には，2 次割付をして，これから多層区域を切り出し，残りを隣接多層区域として設定する。

**b)** 見つからなかった場合には，より上位の層の空区分を探索する。

**1)** 見つかった場合には，見つかった区分を仮割付して，1 次割付区分を切り出し，残りの隣接多層区域の各区分には空 1 を設定，切り出した区分に 1 次割付処理を実施する。

**2)** 見つからなかった場合には，区域割付不可とする。

#### 6.4.2.3　区域解放部

区域の解放は，解放区域と隣接区域を調べ，次の通り実施する。

**a)** 両区域を構成する区分の両方が空の場合には，これを連結して両区分を包含する上位区分の解放操作を実施する。連結解放ができる間は上位区分に向けてこれを繰り返す。

**b)** 上記以外の場合は解放対象区分のみを解放して終了する。

## 7　ディスク割付の管理のために必要な多層区分割付表の大きさの概要

管理領域で大きなものは，多層区分割付表であり，その大きさは，セクタサイズと区画容量とに依存する。

代表的な媒体のセクタサイズ，総容量で計算してみると**表 1** のようになる。

**表 1ー代表的な媒体の多層区分割付表の大きさ**

| | セクタサイズ | 総容量 | 総セクタ数 | 区分数 | 必要管理領域 |
|---|---|---|---|---|---|
| ブルーレイ相当 | 2 KB | 128 GB | 64 M | 128 M | 256 Mbit=32 MB=16 K セクタ=総容量の 1/4000 |
| HDD | 512 B | 2 TB | 4 G | 8 G | 16 Gbit=2 GB=4 MB セクタ=総容量の 1/1000 |

## 8　UDF への適用

### 8.1　適用媒体

ここでは，現状の市場でファイル分断による課題を抱えている，HDD レコーダへの適応を可能とすべく HDD への適用について示す

### 8.2　UDF ボリューム構造の HDD への適用の基本的考え方

HDD ボリュームの区画管理システムから与えられたディスクの区画を一つの UDF 適用ボリュームとしてとらえて UDF のボリューム管理を適用する。したがって HDD の当該区画の先頭から LSN（Logical Sector Number）の 0 が開始する。UDF のファイルシステムが適用されるのは，主ボリューム記述子列の区画記述子の区画開始位置の指す論理ブロックから区画長の値でカバーされる連続領域である。LBN（Logical Block Number）の 0 は区画開始位置から符番される。UDF では，この領域に CoPo2 を適用する。

### 8.3　管理データ構造の確保方針

#### 8.3.1　構成主区分管理表

区画記述子（Partition Descriptor）の区画長（Partition Length）を利用する。

#### 8.3.2　処理システム用情報の利用

区画記述子の中の処理システム用（Implementation Use）を利用して下記を記述する。

##### 8.3.2.1　総区分数（Total Numbr of Deveided Partitions）

8 バイトの Uint64 を Total Number of Divided Partitions として確保する。

##### 8.3.2.2　最上位区分層管理表

区分層の最大は 31 であるから，1 バイトの byte を最上位区分（MaxmumLevel）として確保する。

##### 8.3.2.3　主区分番号管理表（MasterDeveidedPartitions）

最大区画は 2 TB，4 G（512 byte）セクタである。最大のセクタ番号は，4 バイトで表現可能である。

区分番号はその倍の値になるので，一層に 8 バイトの Unit64 を確保し，層は最大 31 段となる

##### 8.3.2.4　末尾番号管理表（LevelEndPositions）

一層に 8 バイトの Unit64 を確保し，層は最大 31 段となる。

#### 8.3.3　多層区分割付表

最近の HDD では，セクタサイズは 512 B，UDF でカバーするサイズ 2 TB を想定すると，総セクタ数は 4 G セクタ，区分数は 8 G，全体で 16 Gbit（2 GB，4 M セクタ）の領域が必要である。

UDF ファイル領域は，通常，未割付け空間記述子（Unallocated Space Descriptor）からポイントされる空間ビットマップ記述子（Space Bitmap Descriptor）で管理される。この規格では，区画記述子の処理用システム用（Implementation Use）に CoPo2 区画ヘッダ記述子を設定し，CoPo2 区画ヘッダ記述子の short_ad として，多層区分管理ビットマップ（Multi level Divided Partitions Allocation Bit Map）フィールドからポイントされる空間ビットマップ記述子（Space Bitmap Descriptor）の中のビットマップ（Bitmap）領域で，2 ビットマップを用いてファイルの領域管理を可能とする。

### 8.4　UDF への適用データ構造

CoPo2 で利用するデータ構造については，UDF のセキュリティ拡張（**OSTA Secure UDF 1.00**）の拡張方法を踏襲する。

#### 8.4.1　一般

##### 8.4.1.1　実体識別子（Entity Identifier）

```
struct EntityID {                /* JIS X 0607    1/7.4 */
        Uint8           Flags;
        char            Identifier[23];
        char            IdentifierSuffix[8];
}
```

**a)　識別子添字**（Identifier Suffix）

識別子添字（*IdentifierSuffix*）欄のフォーマットは，識別子の種別に依存する。論理ボリューム記述子の実体識別子の当該欄については，拡張を実施する。

#### 8.4.2　ボリュームデータ構造

##### 8.4.2.1　論理ボリューム記述子（Logical Volume Descriptor）

```
struct LogicalVolumeDescriptor{                  /* JIS X 0607    3/10.6 */
        struct tag          DescriptorTag;
        Uint32              VolumeDescriptorSequenceNumber;
        struct charspec     DescriptorCharacterSet;
```

| | |
|---|---|
| dstring | LogicalVolumeIdentifier[128]; |
| Uint32 | LogicalBlockSize; |
| struct EntityID | **DomainIdentifier;** |
| byte | LogicalVolumeContentsUse[16]; |
| Uint32 | MapTableLength; |
| Uint32 | NumberofPartitionMaps; |
| struct EntityID | ImplementationIdentifier; |
| byte | ImplementationUse[128]; |
| extent_ad | IntegritySequenceExtent; |
| byte | PartitionMaps[]; |

}

**a) 範囲識別子**（DomainIdentifier）

この欄は，この規格で定義する範囲に適合する論理ボリュームの内容であることを示す。したがって，この範囲識別子（*DomainIdentifier*）は，

“***OSTA UDF Compliant**”

に指定しなければならない。

**b) 識別子添字**（Identifier Suffix）

**表 2－範囲識別子添字欄フォーマット（*Domain IdentifierSuffix field format*）**

| RBP | 長さ | 名前 | 内容 |
|---|---|---|---|
| 0 | 2 | **UDF 版数（UDF Revision）** | Uint16（= #0A00） |
| 2 | 1 | **範囲フラグ（Domain Flags）** | Uint8 |
| 3 | 5 | **CoPo2 版数(CoPo2 Version)** | Unit8 |
| 8 | 4 | 予備（Reserved） | バイト（= #00） |

**1) UDF 版数**（UDF Revision）

#0A00 つまり UDF A.00 を設定し CoPo2 を適用した版数の開始版数番号とする。

**2) CoPo2 版数**（CoPo2 Version）

CoPo2 の版数を設定。

**3) 範囲フラグ**（Domain Flags）

CoPo2 ビットマップ管理を適用することを指定するビットを設定。

**表 3－範囲フラグ（*Domain Flags*）**

| RBP | 意味 |
|---|---|
| 0 | ハード書き込み保護 |
| 1 | ソフト書き込み保護 |
| 2 | セキュア UDF |
| 3 | **CoPo2 ビットマップ管理を適用** |
| 4 -7 | 予備（Reserved） |

### 8.4.2.2 論理ボリューム保全記述子（Logical Volume Integrity Descriptor）

```
struct LogicalVolumeIntegrityDesc{                    /* JIS X 0607  3/10.10 */
        struct tag          DescriptorTag;
        Timestamp           RecordingDateAndTime;
        Uint32              IntegrityType;
        struct extend_ad    NextIntegrityExtent;
        byte                LogicalVolumeContentsUse[32];
        Uint32              NumberOfPartitions;
        Uint32              LengthOfImplementationUse;
        Uint32              FreeSpaceTable[];
        Uint32              SizeTable[];
        byte                ImplementationUse[];
}
```

**a)　処理システム用**（ImplementationUse）

論理ボリューム保全記述子（Logical Volume Integrity Descriptor）の処理システム用（ImplementationUse）領域は，**表 4** に示す構成でなければならない。

**表 4－処理システム用フォーマット** ***（ImplementationUse format）***

| RBP | 長さ | 名前 | 内容 |
|---|---|---|---|
| 0 | 32 | 処理システム識別子（ImplementationID） | EntityID |
| 32 | 4 | ファイル数（Number of Files） | Uint32 |
| 36 | 4 | ディレクトリ数（Number of Directories） | Uint32 |
| 40 | 2 | **UDF 読出し最小版数（Minimum UDF Read Revision）** | Uint16（= #0201） |
| 42 | 2 | **UDF 書込み最小版数（Minimum UDF Write Revision）** | Uint16（= #0A00） |
| 44 | 2 | **UDF 書込み最大版数（Maximum UDF Write Revision）** | Uint16（= #0A00） |
| 46 | ?? | 処理システム用（Implementation Use） | バイト |
| **注記**　??は可変長欄を示す。 | | | |

**1)　UDF 読出し最小版数（Minimum UDF Read Revision）**：この媒体中の全ての構造を読み出すために，処理システムが利用可能とする必要がある UDF 版数の最小値でなければならない。この番号は，10 進数の 2 進表示で記録しなければならない。#0201 つまり UDF 2.01 を設定。

**2)　UDF 書込み最小版数（Minimum UDF Write Revision）**：この媒体中の全ての構造を更新するために，処理システムが利用可能とする必要がある UDF 版数の最小値でなければならない。#0A00 つまり UDF A.00 を設定し CoPo2 を適用した版数の開始版数番号とする。

**3)　UDF 書込み最大版数（Maximum UDF Write Revision）**：媒体中を更新した処理システムが利用可能とする UDF 版数の最大値でなければならない。処理システムは，媒体の更新によって，媒体がこの欄の現状の値より大きい UDF 版数を必要とするようになった場合だけ，この欄を更新しなければならない。#0A00 つまり UDF A.00 を設定。

#### 8.4.2.3　区画記述子（Partition Descriptor）

```
struct PartitionDescriptor {                          /* JIS X 0607  3/10.5 */
        struct tag          DescriptorTag;
        Uint32              VolumeDescriptorSequenceNumber;
```

```
        Uint16          PartitionFlags;
        Uint16          PartitionNumber;
        struct EntityID PartitionContents;
        byte            PartitionContentsUse[128];
        Uint32          AccessType;
        Uint32          PartitionStartingLocation;
        Uint32          PartitionLength;
        struct EntityID ImplementationIdentifier;
        byte            ImplementationUse[128]
        byte            Reserved[156];
}
```

**a) 区画内容用**（PartitionContentsUse）

この欄には区画ヘッダ記述子（Partition Header Descriptor）の内容が記述される。

**b) 区画開始位置**（PartitionStartingLocation）

この欄の値は，HDD のセクタ長の整数倍でなければならない。

**c) 区画長**（PartitionLength）

この欄の値は，HDD のセクタ長の整数倍でなければならない。

**d) 処理システム識別子**（ImplementationIdentifier）

識別子値として“*CoPo2 Compliant”を設定。

**e) 処理システム用**（ImplementationUse）

この欄には，CoPo2 区画ヘッダ記述子を設定。

### 8.4.3 ファイルデータ構造

#### 8.4.3.1 区画ヘッダ記述子（Partition Header Descriptor）

```
struct PartitionHeaderDescriptor{                    /* JIS X 0607  4/14.3 */
        struct short_ad UnallocatedSpaceTable;
        struct short_ad UnallocatedSpaceBitmap;
        struct short_ad PartitionIntegrityTable;
        struct short_ad FreedSpaceTable;
        struct short_ad FreedSpaceBitmap;
        byte            Reserved[88];
}
```

**a) 未割付空間ビットマップ**（UnallocatedSpaceBitmap）

この欄は用意だけしておき，後方書込み互換を実現する時に最新の値を生成する。

**b) 区画保全表**（PartitionIntegrityTable）

区画保全エントリは使用しないため，全て 0 を指定しなければならない。

#### 8.4.3.2 CoPo2 区画ヘッダ記述子（CoPo2 Partition Header Descriptor）

```
struct PartitionHeaderDescriptor{
        struct short_ad UnallocatedSpaceTable;
        struct short_ad Multi level Divided PartitionsStatusBitmap;
        struct short_ad PartitionIntegrityTable;
```

```
        struct short_ad     FreedSpaceTable;
        struct short_ad     FreedSpaceBitmap;
        struct short_ad     CoPo2ManageTableAd
        byte                Reserved[80];
}
```

**a)　多層区分割付ビットマップ**（Multi level Divided PartitionsStatusBitmap）

各区分の状態を 2 ビットで管理する。

**b)　区画保全表**（PartitionIntegrityTable）

区画保全エントリは使用しないため，全て 0 を指定しなければならない。

**c)　CoPo2 管理表領域**（CoPo2ManageTableAd）

CoPo2 管理表の領域を設定。

#### 8.4.3.3　空間ビットマップ記述子（Space Bitmap Descriptor）

```
struct SpaceBitmap{         /* JIS X 0607    4/14.12 */
        struct Tag          DescriptorTag;
        Uint32              NumberOfBits;
        Uint32              NumberOfBytes;
        byte                Bitmap[];
}
```

未割付空間ビットマップ，多層区分割付ビットマップから参照される。

**a)　記述子タグ**（struct Tag DescriptorTag）

空間ビットマップ（*SpaceBitmap*）記述子に関する記述子タグの記述子 CRC（*DescriptorCRC*）欄の計算及び保守は，任意機能である。CRC を保守できない場合，記述子 CRC 欄及び記述子 CRC 長（*DescriptorCRCLength*）欄の両方を 0 に設定する。

#### 8.4.3.4　CoPo2 管理表（CoPo2ManageTable）

**表 5－CoPo2 管理表**（CoPo2ManageTable）

| RBP | 長さ | 名前 | 内容 |
|---|---|---|---|
| 0 | 16 | 記述子タグ用 | DescriptorTag |
| 16 | 32 | **総区分数**（TotalNumberOfDividedPartitions） | EntityID |
| 48 | 4 | **最上位区分層**（MuximumPartitonLevel） | Uint32 |
| 52 | 256 | **主区分番号管理表**（MuximumPartitonLevel） | Uint64[32] |
| 308 | 256 | **末尾番号番号管理表**（EndPositionNumberTable） | Uint16[32] |
| 564 | 460 | 予備(Reserve) | |

主区分番号管理表と末尾番号管理表については，-1 相当のところは，全てのビットを on にする。

---

**参考文献**

**OSTA Secure UDF 1.00**，Secure Universal Disk Format Specification Revision 1.00, Optical Storage Technology Association (OSTA), http://www.osta.org/

この OITDA 規格は，一般財団法人光産業技術振興協会光ディスク標準化委員会の審議により制定したものである。
この資料についてのご意見又はご質問は，下記にご連絡ください。

OITDA 規格：

再配置を少なくするファイル配置方策

（File Allocation System with minimized reallocation）

規格番号：OITDA DC01：2013　第 1 版

第 1 版 発行日：2012 年 3 月 7 日

発行者：一般財団法人光産業技術振興協会
住所：〒112-0014 東京都文京区関口 1-20-10
住友江戸川橋駅前ビル 7F
電話：03-5225-6431　FAX：03-5225-6435
e-mail：opt-st@oitda.or.jp　（標準化室）