組込システム向けＦＡＴファイルシステム

SAVER60 for FAT

ユーザーズ・マニュアル

第1．05版（平成 17 年　2 月 10 日）

ウェスコム株式会社　Wescomlst

はじめに

この度は、「SAVER60 for FAT」をご購入いただきありがとうございます。

本マニュアルは、FAT ファイルシステム「SAVER60 for FAT」の C 言語における使用方法について記述したものです。

なお、本マニュアルは、RTOS(Real Time OS)及び C 言語に関する一般的知識をもっておられる事を前提としています。

また、「SAVER60 for FAT」で使用する I/O ドライバに関しては「I/O ドライバ作成マニュアル」をご参照ください。

ご注意

(1) 本マニュアルの一部、またはすべてを無断で転載複製する事は、固くお断りします。
(2) 本マニュアルに記載されている製品、及び製品の仕様につきましては、製品改良その他により予告無しに変更することがあります。
(3) 本製品の運用結果につきましては、責任を負いかねますのでご了承下さい。
(4) 本マニュアルの内容に関し、お気づきの点などございましたら、弊社担当者にご連絡いただければ、幸いです。

改訂履歴

版番号	改定日	改訂内容
01.00	2004/08/02	初版
01.01	2004/08/30	誤記訂正
01.02	2004/09/24	3－1. ユーザー値定義追加
01.03	2004/10/14	Scandisk 追加
01.04	2004/11/4	誤記訂正
01.05	2005/02/10	FMR_Write その他 *3 記述削除 3－1. ユーザー値定義に DFMR_RWBUFFSIZE 追加

目次

1. 概要

1-1. 特徴

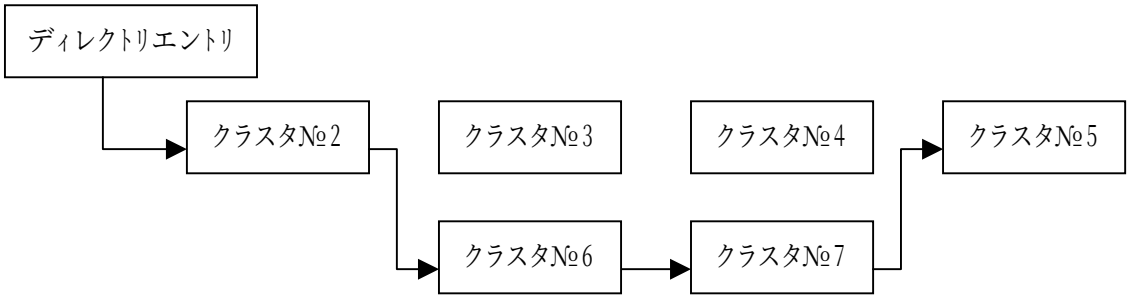
本ファイルシステム:SAVER60 for FAT は、耐電源障害性を充分に考慮した組込システム向け FAT ファイルシステムです。
具体的には下記のような電源障害に対する保護機能を実現しています。

- ファイルへの書込みごとに、ディレクトリ情報の更新を完了させる事により、電源障害直前までに実行されたファイル書込み結果を保護し、ファイルシステム全体の一貫性を維持します。

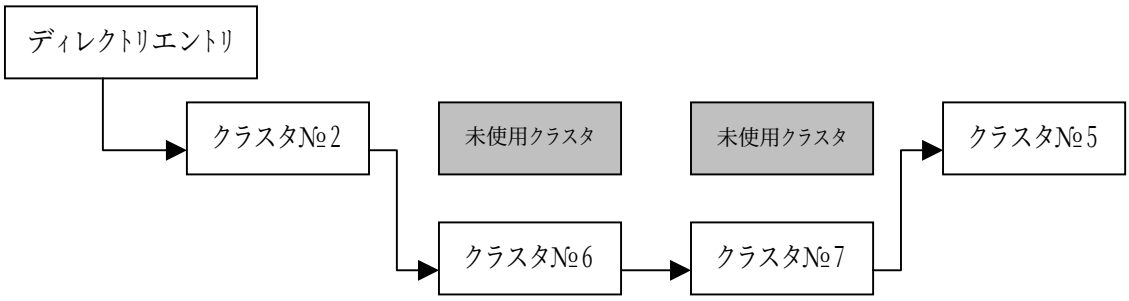
- データの更新は、空クラスタに更新データを書込み、クラスタチェーンを付け替え後、旧データが使用しているクラスタを開放するので、データ更新中に電源障害が発生しても、旧データは保護されます。

(1)更新前

(2)クラスタ№3，4更新



(3)更新前クラスタ開放



1-2. 主な機能

(1)ファイル記憶装置定義

メディア：本ファイルシステムにて、ファイルを実際に格納する記憶装置の種類をメディアと称します。
現在想定しているメディアには、IDE、RAM ディスク、PCMCIA カード等があります。

I/O ドライバ：メディアに対して物理的な I/O 処理を行うプログラムを I/O ドライバと称します。
I/O ドライバは本システムに接続されるメディア毎に必要となります。
但し、同一メディアで複数の記憶装置が接続されている場合は、同一 I/O ドライバで I/O 処理を行うものとします。
I/O ドライバでは、本ファイルシステム側からの論理的 I/O 要求を各メディアの物理的な I/O 要求に変換します。
また、本ファイルシステムではチャネル単位（後述）での排他制御を行うので、I/O ドライバは、チャネル単位でリエントラント可能な構造とする必要があります。

チャネル：各メディア内で同時動作可能な単位をチャネルと称します。
I/O 要求の排他制御は、このチャネル単位で行います。
通常は、記憶装置をコントロールするコントローラがチャネルの単位となります。
例えば、PCMCIA コントローラが 2 つ搭載されている場合、PCMCIA ドライバはチャネル数を2チャネルとして管理します。
但し、IDE コントローラのような場合、1 コントローラでも Primary と Secondary で同時動作が可能な為、IDE ドライバでは、コントローラが1つでもチャネル数を2チャネルとして管理することになります。

チャネル番号：チャネルを識別する為の番号です。
I/O ドライバ単位の 0 から始まる連続した正の整数。
本ファイルシステムでの有効範囲は 0x00～0x0F の最大 16 チャネルです。

デバイス：各チャネルに接続される記憶装置をデバイスと称します。
チャネルに記憶装置が 3 台接続されている場合、デバイス数は3となります。

デバイス番号：デバイスを識別する為の番号です。
チャネル単位の 0 から始まる連続した正の整数。
本ファイルシステムでの有効範囲は 0x00～0x0F の最大 16 デバイスです。

ユニット：デバイス内の記憶領域をユニットと称します。
HDD 装置のように、デバイス内がパーティションに分割されるような記憶装置では、パーティション単位をユニットとします。
例えば、HDD 装置が4パーティションに分割されている場合、ユニット数は4となります。
デバイス内が複数の記憶領域に分割されない場合は、ユニット数は1となります。

ユニット番号：ユニットを識別する為の番号です。
チャネル単位の 0 から始まる連続した正の整数。
本ファイルシステムでの有効範囲は 0x00～0x0F の最大 16 ユニットです。

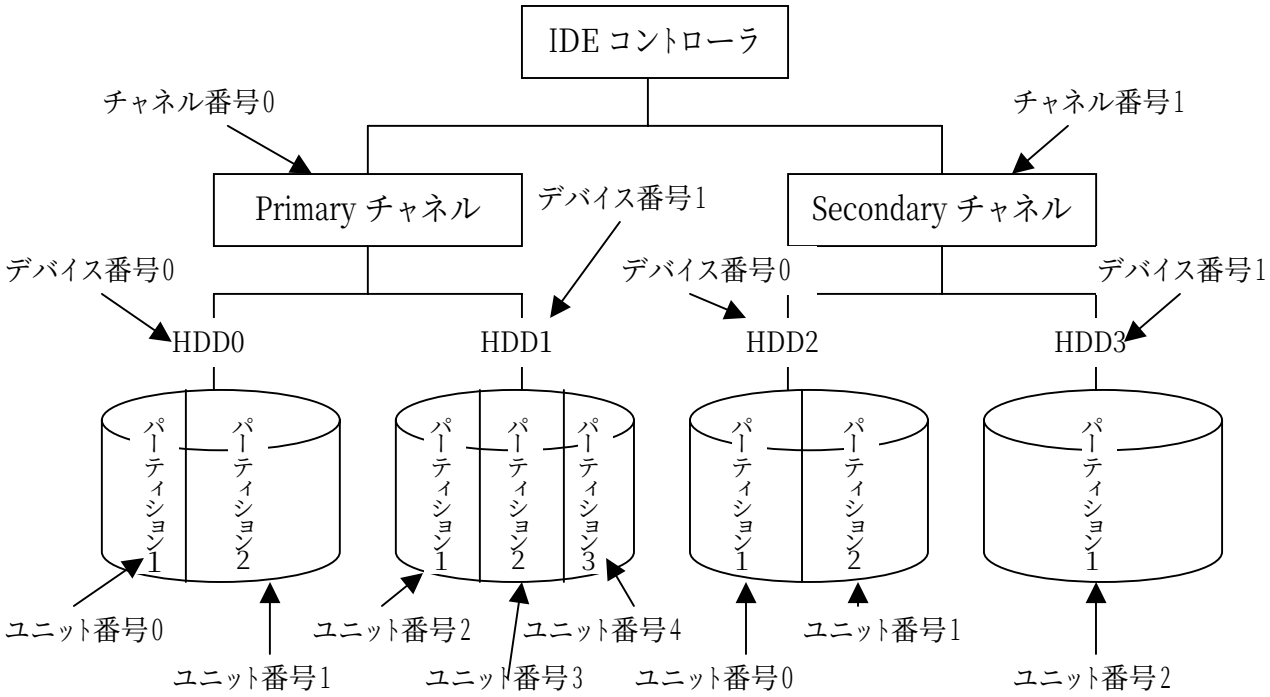
論理ドライブ：本システムでファイルデータを格納する外部記憶領域を論理ドライブと称します。
上位アプリケーション（タスク）は、この論理ドライブを指定して、本ファイルシステムの API を呼び出します。

論理ドライブ番号：論理ドライブを識別する為の番号です。
A ドライブ～Z ドライブまでの指定が可能です。
本ファイルシステムのマウント処理にて、この論理ドライブ番号と、チャネル番号･ユニット番号を対応付けます。

セクタ番号：論理ドライブ内のセクタを識別する為の番号です。
論理ドライブ先頭セクタを 0 とする正の整数。
本ファイルシステムでの有効範囲：0～ユニット全体のセクタ数-1
従って、セクタ番号 0 を Read したら、FAT の Reserved Region の先頭セクタつまり BPB(BIOS Parameter Block)が読込まれます。
また、I/O ドライバに対してのセクタ番号指定も、このセクタ番号で行います。

※チャネル番号、デバイス番号、ユニット番号（複数デバイスが接続されている場合や、ユニットがパーティションに分割されているの場合等）の割当ルールは各I/Oドライバ毎に規定するものとします。

例) IDE コントローラの Primary、Secondary にそれぞれ 2 台ずつの HDD が接続されている場合の、チャネル番号、デバイス番号、ユニット番号の関係



上記の様な構成の場合、

･チャネル数＝2、チャネル番号0を Primary チャネルとします。
チャネル番号1を Secondary チャネルとします。

･チャネル番号0に接続されているデバイス数＝2
デバイス番号0を HDD0、デバイス番号1を HDD1とします。

･チャネル番号1に接続されているデバイス数＝2
デバイス番号0を HDD2、デバイス番号1を HDD3とします。

･チャネル番号0に接続されているユニット数＝5
ユニット番号0を HDD0-パーティション1、ユニット番号1を HDD0-パーティション2、
ユニット番号2を HDD1-パーティション1、ユニット番号3を HDD1-パーティション2、
ユニット番号4を HDD1-パーティション3とします。

･チャネル番号1に接続されているユニット数＝3
ユニット番号0を HDD2-パーティション1、ユニット番号1を HDD2-パーティション2、
ユニット番号2を HDD3-パーティション1とします。

(2)ﾘｴﾝﾄﾗﾝﾄ対応

本ﾌｧｲﾙｼｽﾃﾑでは、ﾏｳﾝﾄ処理にて、論理ﾄﾞﾗｲﾌﾞ番号(”A”～”Z”)単位に、上位ｱﾌﾟﾘｹｰｼｮﾝから指定されたﾒﾓﾘ領域を割当てる事により、論理ﾄﾞﾗｲﾌﾞ単位のﾘｴﾝﾄﾗﾝﾄ構造を実現しています。
ﾏｳﾝﾄ処理以降、本ﾌｧｲﾙｼｽﾃﾑ内では、この論理ﾄﾞﾗｲﾌﾞ単位に割当てられたﾒﾓﾘ領域を使用して、処理を行います。

①複数ﾀｽｸからの異なる論理ﾄﾞﾗｲﾌﾞへの同時ｱｸｾｽ

論理ﾄﾞﾗｲﾌﾞ単位にﾒﾓﾘ領域が割当てられていますので、論理ﾄﾞﾗｲﾌﾞが異なっている場合、それぞれ別々の論理ﾄﾞﾗｲﾌﾞ用ﾒﾓﾘ領域を使用する為、ﾘｴﾝﾄﾗﾝﾄとなり、複数ﾀｽｸからの異なる論理ﾄﾞﾗｲﾌﾞへの同時ｱｸｾｽが可能となります。
但し、異なる論理ﾄﾞﾗｲﾌﾞが同一ﾁｬﾈﾙに接続されている場合、排他制御が必要となります。

②複数ﾀｽｸからの同一論理ﾄﾞﾗｲﾌﾞへの同時ｱｸｾｽ

ⅰ異なるﾌｧｲﾙへの同時ｱｸｾｽ

オープン処理にて、ﾌｧｲﾙﾊﾝﾄﾞﾙを割当て、以降、このﾌｧｲﾙﾊﾝﾄﾞﾙを使用して処理する為、ﾌｧｲﾙが異なれば、それぞれ別のﾌｧｲﾙﾊﾝﾄﾞﾙを使用するので、ﾘｴﾝﾄﾗﾝﾄとなり、複数ﾀｽｸからの同一論理ﾄﾞﾗｲﾌﾞ上の異なるﾌｧｲﾙへの同時ｱｸｾｽが可能となります。
但し、論理ﾄﾞﾗｲﾌﾞが同一なので、同じ論理ﾄﾞﾗｲﾌﾞ用ﾒﾓﾘ領域をｱｸｾｽする事から、排他制御が必要となります。
また、ﾌｧｲﾙﾊﾝﾄﾞﾙを使用しない処理については、ﾌｧｲﾙﾊﾝﾄﾞﾙを検索して、
ｱｸｾｽ対象のﾌｧｲﾙ･ﾃﾞｨﾚｸﾄﾘがｵｰﾌﾟﾝ中であれば、ｴﾗｰ終了する事により、ﾘｴﾝﾄﾗﾝﾄ性を保っています。

ⅱ同一ﾌｧｲﾙへの同時ｱｸｾｽ

同一ﾌｧｲﾙに対する多重ｵｰﾌﾟﾝは、本ﾌｧｲﾙｼｽﾃﾑでは禁止しています。
(既にﾌｧｲﾙﾊﾝﾄﾞﾙが割当てられている場合、ｵｰﾌﾟﾝｴﾗｰとしています。)
これにより、複数ﾀｽｸでの同一ﾌｧｲﾙ共有を禁止する為、ﾘｴﾝﾄﾗﾝﾄ性が保たれます。
但し、複数ﾀｽｸで同一ﾌｧｲﾙﾊﾝﾄﾞﾙの共有は可能とします。
この場合、上記「異なるﾌｧｲﾙへの同時ｱｸｾｽ」と同様、同じ論理ﾄﾞﾗｲﾌﾞ用ﾒﾓﾘ領域、及びﾌｧｲﾙﾊﾝﾄﾞﾙをｱｸｾｽする為、排他制御が必要となります。

(3)排他制御

排他制御処理は、セマフォを使用して行い、セマフォ使用中の場合は、本ファイルシステムを呼び出したタスクを待ち状態としています。

①排他制御の単位をチャネル単位とします。

1-2-(2)「リエントラント対応」で述べた通り、論理ドライブ単位で排他制御を行う必要があります。しかし、同一チャネルに接続されている別の論理ドライブは同時 I/O 動作が行えない為、I/O ドライバレベルのリエントラント性を保持する必要性から（同一チャネルの別論理ドライブへの I/O 要求を禁止する為）、チャネル単位の排他制御としています。
従って、同一チャネルに接続されている別の論理ドライブへアクセスするタスクは待ち状態になります。

②排他制御処理は本ファイルシステム内で行います。

この排他制御処理は、本ファイルシステムが論理ドライブ番号単位でのリエントラント対応の為、論理ドライブ用メモリ領域をアクセスする際に必要となることから、I/O ドライバで行わず、本システムにて行うものとしています。
従って、本ファイルシステムの各 API は、スタック領域を使用したローカル変数のみで行える処理（例えばエラーチェック等）を行った後、セマフォを取得します。
また API 終了時は、取得したセマフォを開放して終了します。

これにより、複数タスクからの同一論理ドライブ、同一ファイルへのファイルハンドル共有によるアクセスの場合でも、API 単位で排他制御が行われる為、リエントラント性が保たれます。

(4)エンディアン

FAT ファイルシステムでは、ファイル記憶装置へのデータ格納はリトルエンディアンで行われますが、本ファイルシステムでは、ビッグエンディアン時、エンディアン変換を行い、ファイル記憶装置へのデータ格納をリトルエンディアンで行えるようにしています。

1-3. 仕様

(1)FAT 種類

FAT12、16、32をサポートしています。また VFAT (long name) へも対応しています。

(2)漢字対応

近日中に、対応予定です。

(3)有効パーティションタイプ

0x01：FAT12、0x04：FAT16(32MByte 未満)、0x06：FAT16(32MByte 以上)
0x0B：FAT32(CHS)、0x0C：FAT32(LBA)、0x0E：FAT16(LBA)
(拡張パーティションは I/O ドライバにて、対応します。)

(4)ファイル多重オープン

同一ファイルに対する多重オープンは禁止しています。

(5)パス指定

本ファイルシステムのパス指定は、フルパス指定で行ってください。また、パスの先頭には必ず論理ドライブ (A:～Z:) の指定が必要です。
ドライブ区切り文字は":"、ディレクトリ区切り文字は"\"です。

(6)FAT32に関して

本ファイルシステムでは、FAT32 における FSInfo の、空クラスタ数(FSI_Free_Count)、空クラスタ番号(FSI_Nxt_Free)は常に 0xFFFFFFFF としています。

(7)スタック領域に関して

本ファイルシステムで使用するスタック領域は、上位アプリケーション (タスク) のスタック領域を使用するものとします。
タスクから本ファイルシステムを使用する場合、スタック領域は、タスク毎に割当てられているものとします。これは複数タスク間でのリエントラント性を保つためです。

(8)パス名･ファイル名･ディレクトリ名の制限長

ファイル名･ディレクトリ名の最大長は、Null 含めて 256 バイト、パス名の最大長は Null 含めて 260 バイトです。
ショートファイル名の場合、数字、英大文字、及び次の記号が使用できます。

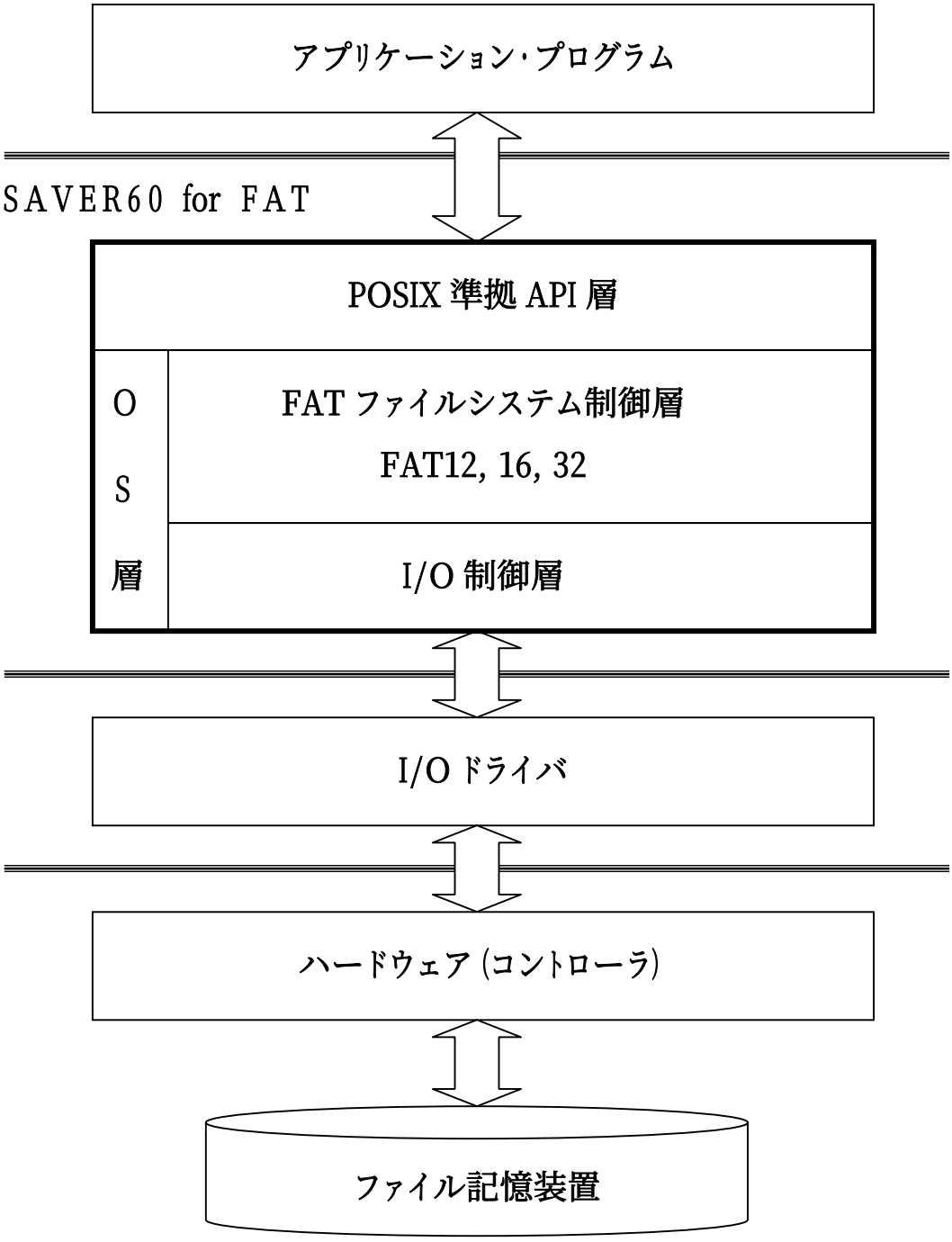
$ % ' - _ @ ~ ` ! ( ) { } ^ # &

ロングファイル名の場合、ショートファイル名で使用できる文字に加えて、英小文字、及び次の記号が使用できます。

+ , ; = [ ]

2. 構成

2-1. システム構成



3. システムへの組込方法

3-1. ユーザー値の定義

下記項目を、システムに合せて FMR_User.h に定義してください。

ラベル名	初期値	内容
DFMR_RWBUFFSIZE	2	システムで使用する R/W バッファサイズをセクタ単位で定義します。 定義できる範囲は、2～クラスタ当りのセクタ数までです。 また定義する値は、必ず偶数を指定してください。
DFMR_USR_MAXFILEHANDLE	32	システムで使用するファイルハンドルの最大数を定義します。 ファイルハンドル1個に付き 56 バイトの RAM を使用します。
DFMR_USR_MAXSEARCHHANDLE	8	システムで使用する検索ハンドルの最大数を定義します。 ファイルハンドル1個に付き 264 バイトの RAM を使用します。
DFMR_USR_MAXSECTSIZE	512	システムで使用するファイル記憶装置の最大セクタサイズ（バイト単位）を定義します。 例)セクタサイズ 512 バイトと 1024 バイトのファイル記憶装置を使用する場合、1024 を定義します。
DFMR_USR_MAXDRVNUM	26	システムで使用する論理ドライブの最大数を定義します。 論理ドライブ1個に付き 60 バイトの RAM を使用しますまたマウント時に、バッファ領域として、論理ドライブ 1 個に付き 3180＋512*DFMR_RWBUFFSIZE バイトの RAM 領域を指定します。

ラベル名	初期値	内容
DFMR_USR_MAXDRIVER	1	システムで使用する I/O ドライバの最大数を定義します。 I/O ドライバ1個に付き(8 バイト＋2×DFMR_USR_MAXCHANNEL) バイトの RAM を使用します。
DFMR_USR_MAXCHANNEL	1	システムで使用する I/O ドライバの最大チャネル数を定義します。 例)チャネル数 1 の I/O ドライバとチャネル数 2 の I/O ドライバを使用する場合、2 を定義します。
DFMR_USR_F32RDCLST	4	FAT32 において、先頭クラスタが不良クラスタの場合ルートディレクトリを作成するクラスタの許容範囲を、総クラスタ数に対するパーセンテージで定義します。 例） 先頭クラスタが不良クラスタの場合、先頭から、(総クラスタ数×(定義値÷100)) クラスタまでの範囲で正常クラスタを検出し、そこに、ルートディレクトリを作成します。先頭から、(総クラスタ数×(定義値÷100)) クラスタまでの範囲で正常クラスタを検出できなければ、フォーマットエラー(エラーコード：DFMR_FORMATERR) とします。
DFMR_ENDIAN_BIG		Big Endian 時のみ当ラベルを定義します。 Little Endian 時は定義不要です。
DFMR_SEMID_AUTO		ITRON 等のセマフォ ID の自動割付け機能を使用する場合、当ラベルを定義します。 自動割付け機能を使用しない場合は、定義不要です。
DFMR_USR_SDRETRY	2	SCANDISK 実行時における、セクタ Read･Write リトライ回数を定義します。

3-2. ドライバ構成情報の定義

※I/O ドライバの構造に関しては、「I/O ドライバ作成マニュアル」を参照してください。

(1)FMR_DriverConf.c に下記のドライバ構成情報を定義してください。

(2)ドライバ構成情報：TFMR_DRVINFO

ドライバ構成情報は、本ファイルシステムで使用する I/O ドライバに関する情報を格納します。

※FMR_DriverConf.c は、ファイルシステム共通 RAM 領域定義ファイル(FMR_ram.c)にてインクルードされます。

データ型	項 目	内 容
unsigned char	driverid	下記のドライバIDを定義します。 0x01=PCMCIAドライバ 0x02=IDEドライバ 0x03=RAMドライバ、 以降I/Oドライバ開発毎に追加予定。 0x40～0x7Fは、ユーザー開発ドライバ用です。 ※Bit7は予約領域の為、使用できません。
unsigned char	channelnum	I/Oドライバで制御するチャネル最大数を定義します。 このチャネル最大数分、セマフォが作成されます。
unsigned char	mediatype	下記のメディアタイプを定義します。 0xf0=removable media 0xf8=non- removable media
int	(* drventry) (unsigned char, unsigned char, void *)	各I/Oドライバ内で定義されるドライバエントリアドレスを定義します。 本ファイルシステムからのドライバ呼出は全てこのエントリアドレス＋I/Oファンクションコードで行われます。
unsigned short	semaphoreid [DFMR_USR_MAXCHANNEL]	セマフォIDを定義します。 ユーザー定義値：DFMR_SEMID_AUTOを定義しない場合、本システムで使用可能なセマフォIDをチャネル最大数、定義します。 ユーザー定義値：DFMR_SEMID_AUTOを定義した場合は、定義不用です。（本システムが、初期化処理で作成したセマフォIDを格納します。）

3-3. O/S 依存部の修正

下記ソースファイルは、O/S に依存した内容になっていますので、使用する O/S に応じて、修正してください。

ソースファイル名	機 能
FMR_osAcreSem.c	セマフォの生成
FMR_osGetTime.c	年月日、時刻の取得 (1985 年 1 月 1 日基準で年月日を求めています。)
FMR_osSigSem.c	セマフォの開放
FMR_osWaiSem.c	セマフォの獲得
FMR_osRAM.c	変数定義
FMR_OS.h	define 定義

3-4. ライブラリ作成方法

(1) 3-1.「ユーザー値の定義」及び 3-2.「ドライバ構成情報の定義」に基づき、必要な定義を行ってください。

(2) 3-3.「O/S 依存部の修正」に基づき、必要な修正を行ってください。

(3)本ファイルシステムソースプログラムをリコンパイルして、ライブラリを作成してください。
※使用開発環境に応じて、リコンパイル時、エンディアンの指定を行ってください。

3-5. アプリケーションへの組込方法

(1)リンク時、本ファイルシステムのライブラリ、使用する I/O ドライバのライブラリをユーザータスクとリンクしてください。

(2)本ファイルシステムを使用するユーザータスクでは、S60FAT.H をインクルードしてください。
S60FAT.H には、API 用各種構造体、エラーコード等の define 値、API 関数のプロトタイプ宣言等が定義されています。

(3)アプリケーションからの利用方法

下記に、サンプルプログラムを示します。

```
#include      "S60fat.h"                  /* SAVER60 for FAT用インクルードファイル      */

unsigned char           userram[4204];    /* 論理ドライブ用RAM領域                      */
TFMR_MOUNTINFO          l_mountinfo;      /* マウント情報                               */
int                     handle;           /* ファイルハンドル                           */
unsigned char           write_data[512];  /* 書込みデータ                               */

FMR_Init();                               /* ファイルシステム初期化                     */
l_mountinfo.driverid = DFMR_DRVID_PCATA;        /* マウント情報.ドライバID指定          */
l_mountinfo.chanel_unit = 0;              /* マウント情報.チャネル番号/ユニット番号指定 */

FMR_Mount(                                /* マウント                                   */
          'A',                            /* 論理ドライブ番号                           */
          &userram,                       /* RAM領域アドレス                            */
          4204,                           /* RAM領域サイズ                              */
          &l_mountinfo        );          /* マウント情報領域アドレス                   */

handle = FMR_Open(                        /* オープン                                   */
          “A:\\SAMPLE.BIN”,               /* ファイルパス名(フルパス指定)               */
          ( DFMR_O_CREAT ¦ DFMR_O_RDWR ), /* オープンフラグ                             */
          DFMR_ATTR_ARC  );               /* 属性                                       */
FMR_Write(                                /* 書込み                                     */
          handle,                         /* ファイルハンドル                           */
          &write_data,                    /* 書込みデータ領域                           */
          512);                           /* 書込みサイズ                               */

FMR_Close(                                /* クローズ                                   */
          handle);                        /* ファイルハンドル                           */
```

4. API 関数

4-1. API 一覧

関数名	機 能
FMR_Init	本ファイルシステムを初期化します。
FMR_Mount	論理ﾄﾞﾗｲﾌﾞをﾏｳﾝﾄします。
FMR_Unmount	論理ﾄﾞﾗｲﾌﾞをｱﾝﾏｳﾝﾄします。
FMR_Open	ﾌｧｲﾙをｵｰﾌﾟﾝします。
FMR_Close	ﾌｧｲﾙをｸﾛｰｽﾞします。
FMR_Read	ﾌｧｲﾙからﾃﾞｰﾀを読込みます。
FMR_Write	ﾌｧｲﾙへﾃﾞｰﾀを書込みます。
FMR_Seek	ﾌｧｲﾙﾎﾟｲﾝﾀを移動します。
FMR_Tell	ﾌｧｲﾙﾎﾟｲﾝﾀ値を取得します。
FMR_Remove	ﾌｧｲﾙを削除します。
FMR_SetAtr	ﾌｧｲﾙ属性を設定します。
FMR_Rename	ﾌｧｲﾙ名･ﾃﾞｨﾚｸﾄﾘ名を変更します。
FMR_Mkdir	ﾃﾞｨﾚｸﾄﾘを作成します。
FMR_Rmdir	ﾃﾞｨﾚｸﾄﾘを削除します。
FMR_FindFirst	ﾌｧｲﾙ･ﾃﾞｨﾚｸﾄﾘの検索を開始します。
FMR_FindNext	ﾌｧｲﾙ･ﾃﾞｨﾚｸﾄﾘの検索を継続します。
FMR_FindClose	ﾌｧｲﾙ･ﾃﾞｨﾚｸﾄﾘの検索を終了します。
FMR_Stat	ﾌｧｲﾙ状態を取得します。
FMR_Info	ﾄﾞﾗｲﾌﾞ情報を取得します。
FMR_Format	ﾕﾆｯﾄをﾌｫｰﾏｯﾄします。
FMR_ScanDisk	ﾌｧｲﾙｼｽﾃﾑを検査します。

FMR_Init

機能

本ファイルシステムを初期化します。

形式

int FMR_Init( );

引数

なし

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

本 API にて、ドライバの初期化、及び必要なセマフォの作成を行います。電源投入時には必ず１度のみ本 API を実行してください。本 API 実行後、「FMR_Mount」API にて、論理ドライブをマウントする事により、論理ドライブに対しての各種 API が実行可能となります。

その他

*1 本システムの各 API は、本 API 実行後に実行可能となり、本 API 実行前に他の API を実行した場合、動作は保障されません。

*2 本 API では、排他制御は行われていないので、本 API は１タスクからのみの実行に限定してください。

FMR_Mount

機能

論理ドライブをマウントします。

形式

```
int  FMR_Mount( char drvnum,  char * raminfo,  unsigned long ramlen,
                TFMR_MOUNTINFO * mountinfo );
```

引数

drvnum	下記マウント情報で指定したユニットに割当てる論理ドライブ番号を指定します。 論理ドライブ番号：0x41(A)～0x5a(Z)
raminfo	RAM 領域アドレス 論理ドライブに割当てる RAM 領域の先頭アドレスを指定します。
ramlen	RAM 領域サイズ 論理ドライブに割当てる RAM 領域のサイズを指定します。
mountinfo	マウント情報を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

マウント情報で指定されたI/Oドライバのチャネル･ユニットと、指定された論理ドライブ番号を対応付けます。
また、指定されたRAM領域に、論理ドライブ毎に必要となるデータ領域を割当てます。
本APIより、マウント情報で指定されたI/Oドライバに対して、「マウント」要求を行います。
本API実行後、論理ドライブに対する各種APIが実行可能となります。

その他

*1 論理ドライブ番号は、ユーザー定義値：DFMR_USR_MAXDRVNUM の範囲内の事。
例) DFMR_USR_MAXDRVNUM = 3 の場合、指定可能な論理ドライブ番号は、
0x41(A)～0x43(C)までです。

FMR_Mount

その他

*2 マウント情報：TFMR_MOUNTINFO

データ型	項 目	内 容
unsigned char	driverid	Bit7：指定論理ドライブが接続されているデバイスのMBRの有無を指定します。 ON:MBR無し、OFF:MBR有り Bit6～0:指定論理ドライブを制御するI/Oドライバのドライバを指定します。 ドライバIDは3-2.「ドライバ構成情報の定義」を参照してください。 ※MBR:Master Boot Record
unsigned char	chanel_unit	Bit7～4：チャネル番号 Bit3～0：ユニット番号

※チャネル番号、ユニット番号に関しては、使用するI/Oドライバの規定を参照してください。

*3 割当メモリに関して

①論理ドライブ毎に異なるメモリ領域を割当ててください。複数論理ドライブに同一メモリ領域を割当てると、本ファイルシステムのリエントラント性が保障できません。

②割当メモリ量

論理ドライブ1個に付き、各種バッファ用に3180＋512*DFMR_RWBUFFSIZEバイト必要です。

③メモリバウンダリ

割当メモリは、必ず使用CPUに応じたメモリバウンダリ上に配置してください。

例) 32BitCPUの場合、4バイトバウンダリ上に配置してください。

FMR_Unmount

機能

論理ドライブをアンマウントします。

形式

int FMR_Unmount(char drvnum);

引数

drvnum	アンマウントする論理ドライブ番号を指定します。 論理ドライブ番号：0x41(A)～0x5a(Z)

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

I/Oドライバで管理しているチャネル･ユニットと、指定された論理ドライブとの対応付けを解除します。

以降、アンマウントされた論理ドライブは再びマウントするまで使用できなくなります。

またマウント時、指定されたRAM領域は、未使用となり、ユーザー側で再利用可能となります。

本APIより、マウント情報で指定されたI/Oドライバに対して、「アンマウント」要求を行います。

その他

FMR_Open

機能

ファイルをオープンします。

形式

int FMR_Open(const char * path, int flag, unsigned char atrb);

引数

path	オープンするファイルへのパス名をフルパスで指定します。
flag	オープンフラグを指定します。
atrb	ファイル属性を指定します。

戻り値

ファイルハンドル(≧0)、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたファイルをオープンします。
オープンフラグ には、ファイルへのアクセスモードやオープンモードを指定します。
ファイル新規作成時、指定されたファイル属性でファイルを作成します。
ファイル新規作成時以外では、ファイル属性は無視されます。
ファイルが正常にオープンされたら、ファイルハンドル（≧0）が返されます。以降、ファイルからのデータ読込み、ファイルへのデータ書込み等では、このファイルハンドルを各APIで指定します。

その他

*1 オープンフラグ

DFMR_O_RDONLY	リードオンリーモードでファイルをオープンします。
DFMR_O_WRONLY	ライトオンリーモードでファイルをオープンします。
DFMR_O_RDWR	リード･ライトモードでファイルをオープンします。
DFMR_O_APPEND	ファイルオープン後、ファイルポインタをファイル終端へ移動します。
DFMR_O_CREAT	ファイルを新規作成します。DFMR_O_EXCLが指定されていなければ、既に指定されたファイルが存在している場合既存ファイルをそのままオープンします。この場合、ファイルポインタは、ファイル先頭に位置します。
DFMR_O_EXCL	指定されたファイルが既に存在している場合、エラーとします。DFMR_O_CREATと同時に指定されている場合のみ有効です。

※オープンフラグには、DFMR_O_RDONLY、DFMR_O_WRONLY、DFMR_O_RDWRのいずれかを必ず指定してください。
※オープンフラグを複数指定する場合、各フラグの論理和の値を指定してください。

FMR_Open

その他

*2 ファイル属性

DFMR_ATTR_RONLY	読込み専用ファイルとして新規作成します。 新規作成後、当ファイル属性が指定されたファイルは、 ﾘｰﾄﾞｵﾝﾘｰﾓｰﾄﾞでのみオープン可能となります。
DFMR_ATTR_HIDE	隠しファイルとして新規作成します。
DFMR_ATTR_SYS	システムファイルとして新規作成します。
DFMR_ATTR_ARC	アーカイブファイルとして新規作成します。

※本ファイルシステムでは、DFMR_ATTR_HIDE、DFMR_ATTR_SYS、
DFMR_ATTR_ARCの各ファイル属性は使用していません。
※属性を複数指定する場合、各属性の論理和の値を指定してください。

*3 ファイルを新規作成する場合、ファイルを作成しようとしているﾃﾞｨﾚｸﾄﾘ直下のサブﾃﾞｨﾚｸﾄﾘに、新規作成するファイル名と同一名のサブﾃﾞｨﾚｸﾄﾘが有る場合、エラーとなります。

*4 同一ファイルに対する多重オープンは禁止されています。

*5 同時オープン可能数は、ユーザー定義値：DFMR_USR_MAXFILEHANDLEに従います。

FMR_Close

機能

ファイルをクローズします。

形式

int FMR_Close(int filehandle);

引数

filehandle　　クローズするファイルのファイルハンドルを指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

ファイルハンドルで指定されたファイルをクローズし、ファイルハンドルを開放します。

その他

FMR_Read

機能

ファイルからデータを読込みます。

形式

```
int   FMR_Read(   int filehandle,     void * readbuff,     unsigned int size   );
```

引数

filehandle	読込みファイルのファイルハンドルを指定します。
readbuff	読込みデータ格納領域を指定します。
size	読込みバイト数を指定します。

戻り値

実際に読込んだバイト数、またはエラーコード(エラーコード一覧参照)を返します。

解説

ファイルハンドルで指定されたファイルのファイルポインタ位置から、読込みデータ格納領域へ読込みバイト数分のデータを読込みます。
読込み後、ファイルポインタは読込みバイト数分進みます。

その他

*1 読込み最大バイト数は、int 型の整数範囲に依存します。
※ int 型が 2 バイトの場合、読込み最大バイト数は 64,512 バイトです。

*2 ファイルポインタ位置がファイル終端にある場合、エラーコード:DFMR_EACCES を返します。

*3 読込み中にファイル終端に達した場合、その時点での読込みバイト数を返して、読込み終了します。

*4 読込みデータ格納領域は、必ず使用 CPU に応じたメモリバウンダリ上に配置してください。
例) 32BitCPU の場合、4 バイトバウンダリ上に配置してください。

FMR_Write

機能

ファイルへデータを書込みます。

形式

int FMR_Write(int filehandle, void * writebuff, unsigned int size);

引数

filehandle	書込みファイルのファイルハンドルを指定します。
writebuff	書込みデータ格納領域を指定します。
size	書込みバイト数を指定します。

戻り値

実際に書込んだバイト数、またはエラーコード(エラーコード一覧参照)を返します。

解説

ファイルハンドルで指定されたファイルのファイルポインタ位置へ、書込みデータ格納領域から書込みバイト数分のデータを書込みます。
書込み後、ファイルポインタは書込みバイト数分進みます。

その他

*1 書込み最大バイト数は、int 型の整数範囲に依存します。
※ int 型が 2 バイトの場合、書込み最大バイト数は 64,512 バイトです。

*2 ファイルオープン時、オープンフラグに DFMR_O_APPEND が指定されている場合、常にファイル終端からデータが書込まれ、ファイルポインタは書込み後のファイル終端へ移動します。

*3 書込みデータ格納領域は、必ず使用 CPU に応じたメモリバウンダリ上に配置してください。
例) 32BitCPU の場合、4 バイトバウンダリ上に配置してください。

FMR_Seek

機能

ファイルポインタを移動します。

形式

long FMR_Seek(int filehandle, long offset, int origin);

引数

filehandle	ファイルポインタを移動させるファイルのファイルハンドルを指定します。
offset	初期位置からの移動バイト数を指定します。 (マイナスの指定も可です。)
origin	初期位置を指定します。

戻り値

移動後のファイルポインタ値、またはエラーコード(エラーコード一覧参照)を返します。

解説

ファイルハンドルで指定されたファイルのファイルポインタを指定された初期位置から、指定された移動バイト数分、移動します。

ファイルポインタはファイル終端またはファイル先頭を越えて移動させる事はできません。

以降のファイルの読込み/書込みは、この移動後のフィルポインタ位置から行われます。

その他

*1 初期位置

DFMR_SEEK_SET	ファイル先頭
DFMR_SEEK_CUR	ファイルポインタの現在位置
DFMR_SEEK_END	ファイル終端

※初期位置にファイル先頭を指定する場合、移動バイト数には、正の値を、ファイル終端を指定する場合は、移動バイト数には、負の値を指定してください。

FMR_Tell

機能

ファイルポインタ値を取得します。

形式

long FMR_Tell(int filehandle);

引数

filehandle　　ファイルポインタ値を取得するファイルのファイルハンドルを指定します。

戻り値

現在のファイルポインタ値、またはエラーコード(エラーコード一覧参照)を返します。

解説

ファイルハンドルで指定されたファイルのファイルポインタ値（ファイル先頭からのバイト数）を取得します。

その他

FMR_Remove

機能

ファイルを削除します。

形式

int FMR_Remove(const char * path);

引数

path 削除するファイルへのパス名をフルパスで指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたファイルを削除します。
パス名で指定されたファイルがオープン中の場合は、エラーとなります。
また、パス名で指定されたファイルが FMR_FindFirst、FMR_FindNext による検索中の場合、検索ハンドルの開放を行った後、ファイルの削除を行います。

その他

FMR_SetAtr

機能

ファイル属性を設定します。

形式

int FMR_SetAtr(const char * path, unsigned char attr);

引数

path	ファイル属性を設定するファイルへのパス名をフルパスで指定します。
attr	ファイル属性を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたファイルに指定されたファイル属性を設定します。
パス名で指定されたファイルがオープン中の場合は、エラーとなります。

その他

*1 ファイル属性に関しては、「FMR_Open」を参照してください。

FMR_Rename

機能

ファイル名･ディレクトリ名を変更します。

形式

int FMR_Rename(const char * oldpath, const char * newpath);

引数

oldpath	変更対象のファイル･ディレクトリへのパス名をフルパスで指定します。
newpath	新しい名前のファイル･ディレクトリへのパス名をフルパスで指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

変更対象のパス名で指定されたファイル･ディレクトリの名前を、指定された新しい名前に変更します。

変更対象のパス名で指定されたファイルがオープン中の場合は、エラーとなります。

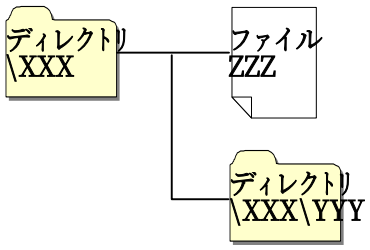
新しい名前と同一のファイル名･ディレクトリ名が同一階層のファイルまたはディレクトリに存在する場合はエラーとなります。

ファイル名の場合、新しい名前が、属しているディレクトリ直下のサブディレクトリと同一名の場合、エラーとなります。

ディレクトリ名の場合、新しい名前が、属しているディレクトリ直下のファイルと同一名の場合、エラーとなります。

本 API ではパス名の変更によるファイルの移動は行えません。

その他



左図のような場合、

①ファイル”ZZZ”は、新しい名前”YYY”には変更できません。

②ディレクトリ”\XXX\YYY”は、新しい名前”\XXX\ZZZ”には変更できません。

FMR_Mkdir

機能

ディレクトリを作成します。

形式

int FMR_Mkdir(const char * path);

引数

path 作成するディレクトリへのパス名をフルパスで指定します。

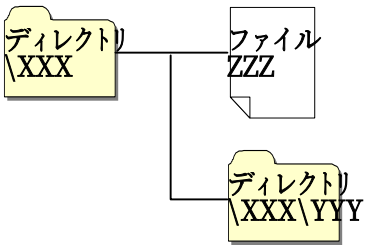
戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたディレクトリを作成します。
作成するディレクトリ名が、属しているディレクトリ直下のファイルと同一名の場合、エラーとなります。

その他



左図のような場合、
①ディレクトリ"\XXX\ZZZ"は、作成できません。

FMR_Rmdir

機能

ディレクトリを削除します。

形式

int FMR_Rmdir(const char * path);

引数

path 削除するディレクトリへのパス名をフルパスで指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたディレクトリを削除します。

指定されたディレクトリ直下にサブディレクトリまたはファイルが存在している場合、エラーとなります。

またルートディレクトリは削除できません。

その他

FMR_FindFirst

機能

ファイル･ディレクトリの検索を開始します。

形式

long FMR_FindFirst(const char * path, TFMR_FILE_INFO * fileinfo);

引数

path	検索するファイル･ディレクトリへのパス名をフルパスで指定します。 パス名には最終要素(検索対象となるファイル名･ディレクトリ名)のみワイルドカード"*"、"?"が使用可能です。
fileinfo	ファイル状態を格納する領域を指定します。

戻り値

検索ハンドル(≧0)、またはエラーコード(エラーコード一覧参照)を返します。

解説

指定されたファイル･ディレクトリの検索を行い、該当する最初のファイル･ディレクトリに関する情報を取得し、指定されたファイル状態格納領域へ格納します。
また同時に、検索ハンドルを返します。
同一パス名で検索を継続する場合（パス名にワイルドカードを指定している場合）、本 API から返される検索ハンドルを用いて、「FMR_FindNext」を実行します。

その他

*1 ワイルドカードに関して

本 API でサポートしているワイルドカードは"*"、"?"です。また意味は下記の通りです。

"*"：0 個以上の文字

* = 全てのファイル、サブディレクトリが検索対象となります。

. = 拡張子付の全てのファイルが検索対象となります。

"?"：任意の 1 文字

FMR_FindFirst

その他

*2 ファイル状態：TFMR_FILE_INFO

データ型	項 目	内 容
unsigned short	crtdate	作成日付 Bit15～9：西暦年 Bit8～5：月 Bit4～0：日
unsigned short	crttime	作成時刻 Bit15～11：時間 Bit10～5：分 Bit4～0：秒
unsigned short	lstaccdate	最終アクセス日付 フォーマットは作成日付と同様です。
unsigned short	wrtdate	更新日付 フォーマットは作成日付と同様です。
unsigned short	wrttime	更新時刻 フォーマットは作成時刻と同様です。
unsigned char	crtmsec	作成時刻の0msec～199msec(作成時刻の秒は2秒単位の為)
unsigned char	attr	ファイル属性（「FMR_Open」を参照してください）
unsigned long	filesize	バイト単位のファイルサイズです。
unsigned char	sfn[11]	ファイル･ディレクトリのショートファイル名(最大8Byte)+拡張子(3Byte)
unsigned char	lfn[256]	ファイル･ディレクトリのロングファイル名(最大254Byte)

FMR_FindNext

機能

ファイル･ディレクトリの検索を継続します。

形式

int FMR_FindNext(long srchhandle, TFMR_FILE_INFO * fileinfo);

引数

srchhandle	検索を継続する検索ハンドルを指定します。 「FMR_FindFirst」からの戻り値を指定してください。
fileinfo	ファイル状態を格納する領域を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

検索ハンドルで指定された検索パス名で「FMR_FindFirst」で開始したファイル･ディレクトリの検索を継続します。

本 API が実行されるごとに、該当するファイル･ディレクトリに関する情報を取得し指定されたファイル状態格納領域へ格納します。

該当するファイル･ディレクトリが無くなった場合、エラーとなります。

本 API 実行前に「FMR_FindFirst」を実行しておく必要があります。

その他

*1 ファイル状態：TFMR_FILE_INFO に関しては、「FMR_FindFirst」を参照してください。

FMR_FindClose

機能

ファイル･ディレクトリの検索を終了します。

形式

int FMR_FindClose(long srchhandle);

引数

srchhandle　　検索を終了する検索ハンドルを指定します。
「FMR_FindFirst」からの戻り値を指定してください。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

検索ハンドルを開放し、「FMR_FindFirst」で開始したファイル･ディレクトリの検索を終了します。

その他

FMR_Stat

機能

ファイル状態を取得します。

形式

int FMR_Stat(const char * path, TFMR_FILE_INFO * fileinfo);

引数

path	ファイル状態を取得するファイルへのパス名をフルパスで指定します。
fileinfo	ファイル状態を格納する領域を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

パス名で指定されたファイルのファイル状態を取得します。

その他

*1 ファイル状態：TFMR_FILE_INFO に関しては、「FMR_FindFirst」を参照してください。

FMR_Info

機能

論理ドライブ情報を取得します。

形式

int FMR_Info(char drvnum, TFMR_DRIVE_INFO * drvinfo);

引数

drvnum	論理ドライブ情報を取得する論理ドライブ番号を指定します。 論理ドライブ番号：0x41(A)～0x5a(Z)
drvinfo	ドライブ情報を格納する領域を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

指定された論理ドライブ番号のドライブ情報を取得します。

その他

*1 ドライブ情報：TFMR_DRIVE_INFO

データ型	項 目	内 容
unsigned char	volume[11]	ボリューム名
unsigned char	mediatype	メディアタイプ 0xf0：removable media 0xf8：non-removable media
unsigned char	fattype	FAT種別 0x01:FAT12 0x02：FAT16 0x03：FAT32
unsigned long	totalsize	総容量(バイト単位)
unsigned long	usedsize;	使用容量(バイト単位)
unsigned long	emptyzise	未使用容量(バイト単位)

FMR_Format

機能

ユニットをフォーマットします。

形式

int FMR_Format(TFMR_FORMATINFO * formatinfo);

引数

formatinfo　　フォーマット情報を指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

指定されたフォーマット情報に従って I/O ドライバで管理しているユニットをフォーマットします。

本 API で行うフォーマットでは、I/O ドライバで管理しているユニット上に、FAT ファイルシステムを構築します。(BPB：BIOS Parameter Block、FAT、FSInfo(FAT32 の場合)が作成され、バッドクラスタの検出が行われます。

記憶装置に対する物理フォーマット(物理セクタ単位のフォーマット)及び、パーティションの作成は、本 API ではサポートしていません。

本 API 実行後、「FMR_Mount」を行う事により、フォーマットしたユニットに対して論理ドライブ番号が割当てられ、論理ドライブとして利用可能となります。

その他

*1 フォーマット情報：TFMR_FORMATINFO

データ型	項 目	内 容
unsigned char	driverid	Bit7：指定ユニットが接続されているデバイスのMBRの有無を指定します。 ON:MBR無し、OFF:MBR有り Bit6～0:指定ユニットを制御するI/Oドライバのドライバを指定します。 ※MBR:Master Boot Record
unsigned char	chanel_unit	Bit7～4：チャネル番号 Bit3～0：ユニット番号

※チャネル番号、ユニット番号に関しては、使用する I/O ドライバの規定を参照してください。

FMR_Format

その他

*1 フォーマット情報:TFMR_FORMATINFO (続き)

データ型	項 目	内 容
unsigned short	entry_cnt	ルートディレクトリエントリ数を指定します。 FAT12の場合、ルートディレクトリに登録できるエントリ数を指定します。必ずセクタ単位に収まる数を指定してください。(セクタサイズ512バイトの場合、16の倍数) FAT16/32の場合、0を指定 (FAT16の場合は、512に内部で設定されます。)
unsigned char	FATtype	フォーマットするFATタイプを指定します。 0x00=ユニットの容量より判断 (容量が512MB以下ならFAT16、512MB以上ならFAT32となります。) 0x01=FAT12 0x02=FAT16 0x03=FAT32
char *	vol_name	BPBに設定するボリューム名を指定します。 ボリューム名の終端にはNULLを指定します。

*2 必ずアンマウント状態で実行してください。

*3 メディア容量と FAT タイプ、クラスタ当りのセクタ数は下記の通りです。

FAT12 の場合

ユニット内 総セクタ数	クラスタ当りの セクタ数	クラスタ サイズ	ユニット容量
≦4,084	1	0.5K	～2.0M
≦8,168	2	1K	2.0M～4M
≦16,336	4	2K	4M～8M
≦32,672	8	4K	8M～16M
≦65,344	16	8K	16M～32M
≦130,688	32	16K	32M～64M
≦261,376	64	32K	64M～128M

FMR_Format

その他

FAT16 の場合

ユニット内 総セクタ数	クラスタ当りの セクタ数	クラスタ サイズ	ユニット容量
≦8,400	エラー	エラー	エラー
≦32,680	2	1K	4.1M～16M
≦262,144	4	2K	16M～128M
≦524,288	8	4K	128M～256M
≦1,048,576	16	8K	256M～512M
≦2,097,152	32	16K	512M～1G
≦4,194,304	64	32K	1G～2G

FAT32 の場合

ユニット内 総セクタ数	クラスタ当りの セクタ数	クラスタ サイズ	ユニット容量
≦66,600	エラー	エラー	エラー
≦532,480	1	0.5K	32.5M～260M
≦16,777,216	8	4K	260M～8G
≦33,554,432	16	8K	8G～16G
≦67,108,864	32	16K	16G～32G
≦0xffffffff	64	32K	32G～

FMR_ScanDisk

機能

ファイルシステムを検査します。

形式

```
int  FMR_ScanDisk( char drvnum,    unsigned char * ramaddr,    unsigned long ramsize,
                   unsigned char * logarea,    unsigned long logsize,
                   unsigned char logflg  );
```

引数

drvnum	ファイルシステムを検査する論理ﾄﾞﾗｲﾌﾞ番号を指定します。 論理ﾄﾞﾗｲﾌﾞ番号:0x41(A)～0x5a(Z)
raminfo	RAM 領域アドレス 当 API に割当てる RAM 領域の先頭アドレスを指定します。
ramlen	RAM 領域サイズ 当 API に割当てる RAM 領域のサイズを指定します。
logarea	検査ログ領域アドレス 検査ログに割当てる RAM 領域の先頭アドレスを指定します。
logsize	検査ログ領域サイズ 検査ログに割当てる RAM 領域のサイズを指定します。
logflg	検査ログ動作フラグを指定します。

戻り値

正常終了、またはエラーコード(エラーコード一覧参照)を返します。

解説

指定された論理ﾄﾞﾗｲﾌﾞに対して不良クラスタの検出及び、ﾃﾞｨﾚｸﾄﾘｴﾝﾄﾘ、ファイルエントリのクラスタﾘﾝｸの整合性チェックを行います。不良クラスタ及びクラスタﾘﾝｸ異常を検出した場合、ｸﾗｽﾀﾘﾝｸをたどれる範囲で修復ﾌｧｲﾙとして保存します。

その他

*1 当APIに割当てるRAM領域に関して

①論理ﾄﾞﾗｲﾌﾞ毎に異なるﾒﾓﾘ領域を割当ててください。複数論理ﾄﾞﾗｲﾌﾞに同一ﾒﾓﾘ領域を割当てると、本ﾌｧｲﾙｼｽﾃﾑのﾘｴﾝﾄﾗﾝﾄ性が保障できません。

②割当ﾒﾓﾘ量

論理ﾄﾞﾗｲﾌﾞ1個に付き、下記計算式で求めたﾒﾓﾘ量が必要です。

必要ﾒﾓﾘ量＝(論理ﾄﾞﾗｲﾌﾞの総ｸﾗｽﾀ数÷8)＋32ﾊﾞｲﾄ

③ﾒﾓﾘﾊﾞｳﾝﾀﾞﾘ

割当ﾒﾓﾘは、必ず使用CPUに応じたﾒﾓﾘﾊﾞｳﾝﾀﾞﾘ上に配置してください。

例) 32BitCPUの場合、4ﾊﾞｲﾄﾊﾞｳﾝﾀﾞﾘ上に配置してください。

*2 検査ログに割当てるに割当てるRAM領域に関して

①論理ﾄﾞﾗｲﾌﾞ毎に異なるﾒﾓﾘ領域を割当ててください。複数論理ﾄﾞﾗｲﾌﾞに同一ﾒﾓﾘ領域を割当てると、本ファイルシステムのﾘｴﾝﾄﾗﾝﾄ性が保障できません。

②割当ﾒﾓﾘ量

論理ﾄﾞﾗｲﾌﾞ1個に付き、下記計算式で求めたﾒﾓﾘ量が必要です。

･ファイル名のみのログの場合:必要ﾒﾓﾘ量＝32ﾊﾞｲﾄ×ログ件数

･フルパスでのログの場合:必要ﾒﾓﾘ量＝276ﾊﾞｲﾄ×ログ件数

※ログ件数を超えた場合は、古いログから上書きしていきます。

③ﾒﾓﾘﾊﾞｳﾝﾀﾞﾘ

割当ﾒﾓﾘは、必ず使用CPUに応じたﾒﾓﾘﾊﾞｳﾝﾀﾞﾘ上に配置してください。

例) 32BitCPUの場合、4ﾊﾞｲﾄﾊﾞｳﾝﾀﾞﾘ上に配置してください。

④検査ログ:

ファイル名指定:TFMR_SDLOGTBLSHT

フルパス指定:TFMR_SDLOGTBLLNG

(*3 検査ログ動作フラグ 元ファイル名出力形式を参照下さい。)

データ型	項 目	内 容
char	pcename[11]	修復ファイル名 ｸﾗｽﾀﾘﾝｸ等に異常があった場合、ﾃﾞｨﾚｸﾄﾘ:“FOUND.XXX”(XXXは001からの連番)下に、このファイル名で修復ファイルが作成されます。
char	errcode	エラーコード 0x01:ﾃﾞｨﾚｸﾄﾘｸﾗｽﾀﾘﾝｸ異常 ﾘﾝｸ先ｸﾗｽﾀが未使用又は既に他のﾃﾞｨﾚｸﾄﾘ又はファイルで使用されています。 0x02:ファイルｸﾗｽﾀﾘﾝｸ異常 ﾘﾝｸ先ｸﾗｽﾀが未使用又は既に他のﾃﾞｨﾚｸﾄﾘ又はファイルで使用されています。 0x03:不良ｸﾗｽﾀ検出 ｸﾗｽﾀﾘﾝｸﾁｪｯｸ中に不良ｸﾗｽﾀを検出しました。

データ型	項 目	内 容
char	errcode	0x04：ファイルサイズエラー ディレクトリ上のファイルサイズとクラスタリンクから得られたファイルサイズが相違しています。 0x05：ディレクトリ先頭クラスタ不良 ディレクトリ先頭クラスタが不良クラスタです。 0x06：不正ディレクトリ 「.」、「..」ディレクトリが存在しません。 0x07：不正ロングファイル名 ロングファイル名の順序又はチェックサムが異常です。 0x08：ディレクトリ先頭クラスタ異常 ディレクトリ先頭クラスタが既に他のディレクトリ又はファイルで使用されています。 0x09：再帰リンク検出 既に使用しているクラスタに再度、リンクされています。
unsigned long	orgtopclstnum	元ディレクトリ/ファイルの先頭クラスタ番号
unsigned short	orgclstnum	元ディレクトリ/ファイルのクラスタ数
char	orgname[12]又は orgname[256]	元ディレクトリ/ファイル名 検査ログ動作フラグで「ファイル名指定」の場合、orgname[12] 「フルパス指定」の場合、orgname[256]

*3 検査ログ動作フラグ

Bit0：検査ログ出力指定

ON＝検査ログを出力する。（但し、当BitがONでも、検査ログ領域アドレスにNULLが指定されている場合は、検査ログは出力されません。）

OFF＝検査ログを出力しない。

Bit1：元ファイル名出力形式

ON＝フルパス指定：元ディレクトリ/ファイル名をフルパスで検査ログに出力します。

OFF＝ファイル名指定：元ディレクトリ/ファイル名のショートファイル名のみを検査ログに出力します。

Bit2：検査ログファイル出力指定

ON＝終了時、検査ログをファイルに出力する。

※検査ログファイルは、ルートディレクトリ直下に、ファイル名：“REPORT.LOG”で出力されます。

OFF＝ファイルへの出力を行わない。

*4 修復ファイルに関して

①修復ファイルはルートディレクトリ直下のディレクトリ：“FOUND.XXX”(XXXは000からの連番)下に作成されます。ルートディレクトリ直下に既に同一ディレクトリ名が存在する場合は、ディレクトリ名の連番部分を＋1した名前のディレクトリを作成し、この作成したディレクトリ下に修復ファイルが作成されます。

例） ディレクトリ：“FOUND.001”が既に存在すれば、ディレクトリ：“FOUND.002”を作成し、このディレクトリ下に修復ファイルが作成されます。

②修復ファイル名は“FILEXXXX.CHK”(XXXXは0000からの連番)です。複数の修復ファイルが作成される場合、連番部分を＋1したファイル名がそれぞれの修復ファイルに付けられます。

③修復ファイルは、クラスタリンクがたどれる範囲で作成されます。

クラスタリンクが途中で未使用の場合、そこまでの内容が修復ファイルに出力されます。

また、クラスタリンクの途中に不良クラスタが存在する場合、先頭クラスタから不良クラスタまでで一つの修復ファイルが作成され、不良クラスタからクラスタリンク終端までが別の修復ファイルとして作成されます。

5. エラーコード

ラベル名	エラー内容
DFMR_SUC	正常終了
DFMR_ENOENT	該当ファイル無し
DFMR_EBADF	ファイルアクセス異常
DFMR_EACCES	オープンモード不一致
DFMR_EEXIST	既存ファイル有り
DFMR_EINVAL	不正な値(オープンフラグ組合わせ異常)
DFMR_EMFILE	オープンファイル数オーバー
DFMR_EFBIG	ファイルサイズオーバー
DFMR_ENOTEMPTY	対象ディレクトリ中身あり
DFMR_MULTIOPEN	2重オープン
DFMR_NMERR	名称異常
DFMR_ERRPARAM	パラメータエラー
DFMR_FATALERR	FATAL ERROR
DFMR_NOENTRY	対象エントリ無し
DFMR_NODIR	対象上位ディレクトリ無し
DFMR_ENOEMPTYENTRY	未使用/無効エントリ領域なし
DFMR_MEMFULL	空容量無し
DFMR_ESEARCHFAIL	検索失敗
DFMR_EIMPFORMAT	フォーマット未完了
DFMR_CLSTNUMERR	クラスタ番号異常
DFMR_NOFATENTRY	FAT エントリなし
DFMR_DRVINITERR	ドライバ初期化エラー
DFMR_CRESENERR	セマフォ生成エラー
DFMR_NODRV	対象ドライバなし
DFMR_CHANNELERR	チャネル番号エラー
DFMR_PARTCNTERR	パーティション数エラー
DFMR_PARTTYPERR	パーティションタイプエラー
DFMR_WAISENERR	セマフォ獲得エラー
DFMR_SIGSENERR	セマフォ返却エラー
DFMR_FATTYPEERR	FAT タイプエラー
DFMR_ENTRYCNTERR	エントリ数エラー
DFMR_RAMSIZEERR	RAM 領域サイズエラー
DFMR_FATERR	FAT サイズエラー
DFMR_DRVNUMERR	論理ドライブ番号エラー

ラベル名	エラー内容
DFMR_DRVCONNECTERR	論理ﾄﾞﾗｲﾌﾞ接続済みエラー
DFMR_DRVNOCONNECTERR	論理ﾄﾞﾗｲﾌﾞ未接続エラー
DFMR_FORMATERR	ルートディレクﾄﾘ作成エラー (FAT32 の場合のみ)
DFMR_SDILLEGALDIR	不正なディレクﾄﾘ(「.」「..」が存在しない)
DFMR_SDILLEGALROOT	ルートディレクﾄﾘ読込みエラー
DFMR_SDROOTENTFULL	修復ファイル作成不能(ルートディレクﾄﾘに空きが存在しない)
DFMR_SDSCRAPERR	修復ファイル作成エラー
DFMR_SDLOGOUTERR	検査ログファイル出力エラー
DFMR_SDALLERRDIR	ルートディレクﾄﾘが全て破損

ドライバエラーコード

ラベル名	エラー内容
DFMR_DRVERR_PARAM	パラメータエラー
DFMR_DRVERR_NOTINIT	未初期化エラー
DFMR_DRVERR_NOCARD	カード抜きエラー
DFMR_DRVERR_ILGALVS	VS1,2 状態エラー
DFMR_DRVERR_MRSHPC	MR-SHPC-01 異常
DFMR_DRVERR_PCCIS	PC カード CIS 異常
DFMR_DRVERR_PCREADYTMO	PC カード Ready TimeOut
DFMR_DRVERR_PCCARD	PC カード異常
DFMR_DRVERR_TSLP	PCATA_osSleep 異常終了
DFMR_DRVERR_UNITOVER	該当ユニット無し
DFMR_DRVERR_NOTMOUNT	未マウントエラー
DFMR_DRVERR_DEFCARD	別カード挿入
DFMR_DRVERR_ILGCARD	カード挿抜発生
DFMR_DRVERR_MOUNT	マウント済みエラー
DFMR_DRVERR_NOPRIMARY	基本領域無し
DFMR_DRVERR_NOEXTEND	拡張領域無し