組込み向けファイルシステム
GR-FILE
ポーティングマニュアル

第 1.04 版

2020 年 3 月

株式会社グレープシステム

[注意事項]

・すべての著作権は、株式会社グレープシステムにあります。
・本ドキュメントの内容の一部または全部を無断で転載、複写、複製する事を禁じます。
・本製品の仕様は予告なく変更される事があります。
・本ドキュメントに記載されている会社名、製品名は各社の商標または登録商標です。

はじめに

本書は、組込みシステムでファイルアクセスを実現するためのミドルウェア「**GR-FILE**」のポーティング手順について記述します。

本書はポーティングを行う際の補助的な情報を記載しています。本書を読む際には先に **GR-FILE** 取扱説明書を事前に参照いただき理解してください。

改訂履歴

Rev.	日付	改訂内容
1.00	2008 年 7 月	初版
1.01	2010 年 11 月	必要な資源について追記
1.02	2012 年 6 月	フットプリントについて追記。 パラメータの最小値を追記
1.03	2017 年 9 月	全面見直し
1.04	2020 年 3 月	非キャッシュ領域に配置するメモリ領域について、使用する API 名などを追記

参照ドキュメント

以下のドキュメントも併せて参照ください。

『**GR-FILE** 取扱説明書.pdf』
『**GR-FILE** アプリケーションノート.pdf』（※１）

※１**GR-FILE** アプリケーションノート.pdf は、弊社技術サポートページの技術資料からダウンロードすることで入手できます。

目次

１．概要

本書は **GR-FILE** を使用するために必要なポーティング項目について記載します。

２．GR-FILE との I/F 部分のポーティング方法について

本章では **GR-FILE** のポーティングについて以下の項目に分けて説明します。

- 必要な資源について
- 環境依存処理関数の I/F について
- **GR-FILE** のパラメータについて
- ポーティング手順
- ポーティングの注意事項

２.１ 必要な資源について

以下に **GR-FILE** で必要となる各種資源を示します。

２.１.１ フットプリントについて

GR-FILE のフットプリントは、おおよそ以下の通りです。

#	設定等	ROM	RAM
1	出荷時	約 100KB	約 100KB
2	GRP_FS_MINIMIZE_LEVEL=2	約 60KB	－
3	２.３章のパラメータによる設定	－	メモリプール概算.xls の値を参照ください

表中の「－」は設定に影響されないことを意味します。例えば「GRP_FS_MINIMIZE_LEVEL=2」は ROM のフットプリントには影響しますが RAM には大きな影響はありません。

２.１.２ セマフォア

GR-FILE では、全体の制御用に１つ、可変長メモリプール管理用に１つ、**GR-FILE** を使用する 1 つのアプリケーションタスクに対して１つずつ必要です。

したがって以下の式で必要数を求められます。

```
必要数　＝　1　＋　　// 全体の制御用
　　1　＋　　// 可変長メモリプール管理用
　　GRP_FS_MAX_TASK　　// GR-FILE を使用するタスク数
```

各セマフォアは **GR-FILE** の初期化時、および可変長メモリプール初期化時に作成しますので、必要分が作成できるように OS 側の設定を行ってください。

GRP_FS_MAX_TASK に関しては **GR-FILE** 取扱説明書の「実行時に変更可能なパラメータ」を参照ください。

２.１.３ タスクスタック

GR-FILE は、**GR-FILE** を呼び出すアプリケーションタスクの延長線上として動作し、**GR-FILE** の下位ドライバへもアプリケーションタスクとして実行されます。

その為、アプリケーションタスクのスタックはこの事を考慮する必要があります。

パラメータ、動作環境、コンパイラ等にも依存しますが、**GR-FILE** を使用するタスクには、8KB のアプリケーションタスクスタックが必要になります。

２.２ 環境依存処理関数の I/F について

以下に **GR-FILE** の持つ各環境依存部分との、関数 I/F を示します。

各依存部のソースファイルは、mdep_xxx ディレクトリ下にあります。

ソースファイルのディレクトリ構成などは **GR-FILE** 取扱説明書の「ソースファイルの構成」を参照ください。

- OS 依存部
  ファイル名：mdep_xxx / base / grp_fs_mdep_if.c
- OS 依存部（mdep_itron 使用時）
  ファイル名：mdep_xxx / lib / grp_itron_id.h
- デバイスドライバ依存部
  ファイル名：（mdep_xxx / base / grp_fs_dev_io.c）
- 可変長メモリプール管理依存部
  ファイル名：（mdep_xxx / lib / grp_mem / grp_mem.c）
- 時刻設定/取得依存部
  ファイル名：（mdep_xxx / lib / grp_time / grp_time_get.c、grp_time_set.c）
- その他のファイルシステム依存部

２.２.１ OS 依存部

GR-FILE は、動作する上でセマフォアの作成/取得/解放、タスク番号の取得、メモリのコピー、文字コードの変換/取得、時刻取得、ファイル名比較、ショート名の作成、文字列の出力を行います。

これらの機能を、ターゲットに合わせてポーティングする必要があります。それぞれ以下の関数のポーティングを行ってください。それぞれの関数仕様に関しては **GR-FILE** 取扱説明書を参照ください。時刻取得に関しては grp_fs_get_current_time()と後述の grp_time_get()のどちらかをポーティングしてください。

- セマフォアの作成/取得/解放
  grp_fs_create_sem()、grp_fs_get_sem()、grp_fs_release_sem()
- タスク番号の取得
  grp_fs_get_taskid()
- メモリのコピー
  grp_fs_copyin()、grp_fs_copyout()、grp_fs_get_str()
- 文字コードの変換/取得
  grp_fs_char_cnt()、grp_fs_char_to_unicode()、grp_fs_unicode_to_char()、grp_fs_to_upper()
- 時刻取得
  grp_fs_get_current_time()
- ファイル名比較
  grp_fs_cmp_fname()
- ショート名の作成
  grp_fs_make_sname_another_method()
- 文字列の出力
  grp_fs_printf()

２.２.２ OS 依存部（mdep_itron 使用時）

mdep_itron を使用する場合は、「２.２.１OS 依存部」に加えて grp_itron_id.h へのポーティングが必要です。

本ファイルでは以下の定義値を設定します。

- GRP_SEM_ID_START
  GR-FILE で作成するセマフォアの開始 ID 番号を定義します。
  定義された番号から順に使用してセマフォアの作成を行います。
- GRP_MEM_POOL_ID
  可変長メモリプール機能を使用する際のメモリプール ID として pget_mpl/rel_mpl で使用されます。

２.２.３ デバイスドライバ依存部

GR-FILE から、デバイスに対して処理を行う場合、５つの関数を呼び出します。

呼び出す関数は、デバイスのオープン、デバイスのクローズ、デバイスの読み込み、デバイスの書き込み、I/O コントロールです。この内、I/O コントロールのサポートは必須ではありません。ですが、残りの４つの機能は必須となりますので、ターゲットに合わせたポーティングが必要です。それぞれ以下の関数のポーティングを行ってください。それぞれの関数使用に関しては **GR-FILE** 取扱説明書の「デバイスドライバインタフェース」を参照ください。

- デバイスのオープン
  _grp_fs_open_dev()
- デバイスのクローズ
  _grp_fs_close_dev()
- デバイスの読み込み
  _grp_fs_read_dev()
- デバイスの書き込み
  _grp_fs_write_dev()
- I/O コントロール
  _grp_fs_ioctl_dev()

本ポーティングを行った場合は、grp_fs_sysdef.h にある GRP_FS_PORTING_DEVICE_IO 定義を有効にしてください。また、デバイスドライバ依存部に必要な機能については **GR-FILE** 取扱説明書の「デバイスドライバインタフェース」を参照ください。

２.２.４ 可変長メモリプール管理依存部

GR-FILE は、動作時にメモリの取得/解放を行います。

mdep_vos、mdep_vos2.xx を使用する場合では、**GR-FILE** の可変長メモリプール管理を使用できますが、mdep_vos、mdep_vos2.xx を使用せず（mdep_itron を使用する場合など）、OS のライブラリなどで可変長メモリプール管理を使用する場合は、その処理をポーティングする必要があります。それぞれ以下の関数のポーティングを行ってください。それぞれの関数使用に関しては **GR-FILE** 取扱説明書を参照ください。

- メモリの取得
  grp_mem_alloc()
- メモリの解放
  grp_mem_free()

２.２.５ 時刻設定/取得依存部

GR-FILE では、ファイルの作成日、アクセス日、更新日も処理を行っています。

その為、現在時刻を取得する必要があります。ですが、OS やターゲットによって、現在時刻の設定/取得方法が異なる為、設定/取得部分をポーティングする必要があります。

GR-FILE では、時刻の設定/取得部分は、空のスタブ関数を用意しています。（**GR-FILE** で必要な機能は時刻の取得機能のみです。時刻の設定機能は任意でご用意/ご使用ください）それぞれ以下の関数のポーティングを行ってください。

- 現在時刻の取得
  grp_time_get()
- 現在時刻の設定
  grp_time_set()

２.２.６ その他のファイルシステム依存部

GR-FILE の提供している FAT ファイルシステム以外のファイルシステム（例えば ISO9660 など）を、**GR-FILE** と同時に使用する場合にポーティングが必要になります。

ポーティングは、**GR-FILE** のファイルシステムの関数ポインタテーブルに合わせた、関数ポインタテーブルを用意して頂き、grp_fs / base / grp_cfg.c のファイルシステムスイッチテーブルへ登録します。ご用意いただく必要のある関数に関しては **GR-FILE** 取扱説明書の「ファイルシステム抽象化インタフェース」を参照ください。

２.３ GR-FILE のパラメータについて

GR-FILE は、動作に関わる多くのパラメータを用意しています。

パラメータをターゲット環境に合わせ調整することで、システムに合わせたパフォーマンスを得ることが出来ます。

パラメータの設定指針に関しては、弊社技術サポートページの技術資料からダウンロードできる「**GR-FILE** アプリケーションノート」を参照ください。

以下に、**GR-FILE** で設定できるパラメータを示します。

以下のパラメータは grp_fs / include / grp_fs_param.h にて設定します。

- 同時に mount するファイルシステムの最大数（GRP_FS_MAX_MOUNT）
- 同時にオープン可能なファイルの最大数（GRP_FS_MAX_FILE）
- 同時にオープン可能なファイルハンドルの最大数（GRP_FS_MAX_FHDL）
- FAT キャッシュのサイズ（GRP_FS_FBLK_SHIFT）
- DATA キャッシュのサイズ（GRP_FS_DBLK_SHIFT）
- FAT キャッシュ数（GRP_FS_FBLK_CNT）
- DATA キャッシュ数（GRP_FS_DBLK_CNT）
- キャッシュのハッシングバケットの数（GRP_FS_BLK_NHASH）
- ファイルシステムをアクセスするアプリケーションタスクの最大数（GRP_FS_MAX_TASK）
- オープン中のファイルのハッシングバケット数（GRP_FS_FILE_NHASH）
- ファイル名キャッシュの最大数（GRP_FS_FNAME_CACHE_CNT）
- ファイル名キャッシュのハッシングバケットの数（GRP_FS_FNAME_NHASH）
- パス名の最大長（GRP_FS_MAX_PATH）
- ファイル名の各コンポネントの最大文字数（GRP_FS_MAX_COMP）
- デバイス名の最大長（GRP_FS_DEV_NAME_LEN）
- ファイルシステムタイプ名の最大長（GRP_FS_TYPE_LEN）
- マウント先名の各コンポネントの最大長（GRP_FS_MOUNT_COMP）
- マウント先名の最大長（GRP_FS_MOUNT_PATH）
- ボリューム名の最大長（GRP_FS_VOL_NAME_LEN）
- ディレクトリ階層の最大ネスト数（GRP_FS_DIR_NEST）

以下のパラメータは grp_fs / include / grp_fat_param.h にて設定します。

- FAT セクタサイズ（FAT_BLK_SHIFT）
- オープン中の各 FAT 領域情報のキャッシュ数（FAT_MAP_CNT）
- フリークラスタキャッシュ数（FAT_FREE_TBL）
- FAT ファイル名の各コンポネントの最大長（FAT_COMP_SZ）
- FAT ファイル名の各コンポネントの最大文字数（FAT_COMP_CHCNT）
- FAT のフリークラスタキャッシュを取得する際のバッファサイズ（FAT_CNT_BUF_SZ）

以下のパラメータは grp_fs / include / grp_fat_format.h にて設定します。

- フォーマット時の I/O バッファサイズ（GRP_FAT_IO_BUF_SZ）

２.３.１ 同時に mount するファイルシステムの最大数

同時に使用するファイルシステムの数を設定します。

最小値は「１」です。

例えば、USB メモリと SD カードを同時にマウントし使用する場合は「２」を設定します。

ですが、USB メモリと SD カードを同時にマウントせず、最初に USB メモリのみマウントし SD カードを使用するときは、USB メモリをアンマウントしてから SD カードをマウントするような場合は「１」となります。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.２ 同時にオープン可能なファイルの最大数

同時にオープンするファイル数を設定します。

最小値は「２」です。(内部管理用に１つ使用するため)

本設定値は、タスク毎にあるのではなく、全てのタスクで同時にオープンするファイル数です。

同じファイルを複数のタスクでオープンした場合は、ファイル数としては「１」となります。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.３ 同時にオープン可能なファイルハンドルの最大数

同時にオープンしているファイルのファイルハンドル数を設定します。

最小値は「１」です。

本設定値は、タスク毎にあるのではなく、全てのタスクで同時にオープンしているファイル数です。

同じファイルを複数のタスクでオープンした場合でも、タスク数の分だけ必要になります。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.４ FAT キャッシュのサイズ、FAT キャッシュ数

FAT のキャッシュとして確保するメモリ量を設定します。

FAT キャッシュサイズの最小値は「９」です。ただし、物理セクタサイズより大きく設定する必要があります。FAT キャッシュ数の最小値は「２」です。

FAT キャッシュサイズは２のべき乗である必要がありますので、２の乗数を設定します。

例えば、2KB のキャッシュを設定する場合は、「１１」を設定します。

FAT キャッシュ数には、FAT キャッシュサイズの領域をいくつ用意するかを設定します。

例えば、２KB のキャッシュを２つ用意する場合は「２」を設定します。

この場合、２KB×２＝４KB のメモリが必要となります。

キャッシュメモリは、FAT キャッシュのサイズ単位で使用されますので、上記の例では一度の FAT 読み込みは２KB となります。

キャッシュメモリは、ヒット数の少ないメモリから再利用されます。

本設定値が増えると、その分 FAT キャッシュバッファ、管理領域としてメモリを必要とします。

２.３.５ DATA キャッシュのサイズ、DATA キャッシュ数

DATA のキャッシュとして確保するメモリ量を設定します。

DATA キャッシュサイズの最小値は「９」です。ただし、物理セクタサイズより大きく設定する必要があります。DATA キャッシュ数の最小値は「２」です。

DATA キャッシュサイズは２のべき乗である必要がありますので、２の乗数を設定します。

例えば、４KB のキャッシュを設定する場合は、「１２」を設定します。

DATA キャッシュ数には、DATA キャッシュサイズの領域をいくつ用意するかを設定します。

例えば、２KB のキャッシュを４つ用意する場合は「４」を設定します。

この場合、２KB×４＝８KB のメモリが必要となります。

キャッシュメモリは、DATA キャッシュのサイズ、または、クラスタサイズの小さい方の値の単位で使用されます。

例えば、DATA キャッシュのサイズが２KB、クラスタサイズが４KB の場合では、DATA キャッシュは２KB 単位で使用されます。

また、DATA キャッシュのサイズが４KB、クラスタサイズが１KB の場合では、DATA キャッシュは１KB 単位で使用されます。（この場合、３KB は使用されません）

キャッシュメモリは、ヒット数の少ないメモリから再利用されます。

本設定値が増えると、その分 DATA キャッシュバッファ、管理領域としてメモリを必要とします。

２.３.６ キャッシュのハッシングバケットの数、オープン中のファイルのハッシングバケット数、ファイル名キャッシュのハッシングバケットの数

ハッシュの数を設定します。

キャッシュのハッシングバケット数の最小値は「２」です。

オープン中のファイルのハッシングバケット数の最小値は「２」です。

ファイル名キャッシュのハッシングバケットの数の最小値は、オープン中のファイルのハッシングバケット数の 2 倍です。本設定を増やした場合、ハッシングによる検索時の深さが浅くなるため検索が高速になります。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.７ ファイルシステムをアクセスするアプリケーションタスクの最大数

GR-FILE を使用するアプリケーションタスクの数を設定します。

最小値は「１」です。（OS 無しの場合はアプリケーションタスクが１つと見なします）

GR-FILE を使用しないアプリケーションタスクは、含みません。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.８ ファイル名キャッシュの最大数

ファイル名キャッシュの最大数を設定します。

最小値はファイル数の 2 倍です。

ファイル名キャッシュは、一度アクセスしたファイルを再度アクセスする際に、高速にアクセスする為のキャッシュです。

本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.９ パス名の最大長

すべてのファイルシステム共通の、パス名の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

最小値は「１」です。使用するすべてのファイルシステムの中で最も長いパス名の最大長以上に設定してください。

本設定値が増えると、その分アプリケーションタスクスタックを必要とします。

２.３.１０ ファイル名の各コンポネントの最大文字数

パス文字列を区切り文字で区切った際の、1 つの文字列の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

本設定はすべてのファイルシステム共通の設定です。

最小値は「１」です。使用するすべてのファイルシステムの中で最も長いコンポネント長以上に設定してください。

本設定値が増えると、その分アプリケーションタスクスタックを必要とします。

２.３.１１ デバイス名の最大長

デバイス名の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

最小値は「１」です。使用するすべてのデバイス名の中の最大長以上に設定してください。

本設定値が増えると、その分管理領域、および、アプリケーションタスクスタックを必要とします。

２.３.１２ ファイルシステムタイプ名の最大長

ファイルシステム名の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

最小値は「１」です。使用するすべてのファイルシステム名の中の最大長以上に設定してください。

本設定値が増えると、その分管理領域、および、アプリケーションタスクスタックを必要とします。

２.３.１３ マウント先名の各コンポネントの最大長

マウント先名文字列を区切り文字で区切った際の、1 つの文字列の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

本設定はすべてのファイルシステム共通の設定です。

最小値は「１」です。階層化マウントを使用する場合は、使用するすべてのファイルシステムの中で最も長いコンポネント長以上に設定してください。階層化マウントを使用しない場合はマウント先名長以上を設定してください。階層化マウントに関しては **GR-FILE** 取扱説明書の「階層化マウント」を参照ください。

本設定値が増えると、その分管理領域、および、アプリケーションタスクスタックを必要とします。

２.３.１４ マウント先名の最大長

すべてのファイルシステム共通の、マウント先名の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

本設定はすべてのファイルシステム共通の設定です。

最小値は「１」です。使用するすべてのファイルシステムの中で最も長いマウント先名の最大長以上に設定してください。

本設定値が増えると、その分アプリケーションタスクスタックを必要とします。

２.３.１５ ボリューム名の最大長

すべてのファイルシステム共通の、ボリューム名の最大長をバイト数で指定します。

この値には文字列最後の NULL 文字も含まれます。

本設定はすべてのファイルシステム共通の設定です。

最小値は「１」です。使用するすべてのファイルシステムの中で最も長いボリューム名の最大長以上に設定してください。

本設定値が増えると、その分アプリケーションタスクスタックを必要とします。

２.３.１６ ディレクトリ階層の最大ネスト数

grp_fs_get_cwd()使用時のディレクトリの最大ネスト数を指定します。

本設定はすべてのファイルシステム共通の設定です。

最小値は「１」です。

本設定値が増えると、その分アプリケーションタスクスタックを必要とします。

２.３.１７ FAT セクタサイズ

１セクタのサイズを指定します。

設定値は、２のべき乗である必要がありますので、２の乗数を設定します。

GR-FILE は、BPB、FSINFO、ボリュームチェックの際にこの値を使用し、リードバイト数が少ない場合にエラーとします。

通常は 512 バイト/セクタのメディアが多い為、「９」を設定します。

２.３.１８ オープン中の各 FAT 領域情報のキャッシュ数

GR-FILE は、オープンしているファイルの FAT チェインを先読みしています。
これにより、実際に FAT を読み込むことなく、次のクラスタ領域を特定できます。
この先読みする FAT のキャッシュ数を設定します。
最小値は「１」です。
本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.１９ フリークラスタキャッシュ数

GR-FILE は、クラスタの割り当て用にフリークラスタ番号のキャッシュを持っています。
これにより、ファイル/ディレクトリの書き込みでクラスタの割り当てが発生した場合でも、FAT を読み込むことなくクラスタを割り当てることが出来ます。
最小値は「１」です。
本設定値が増えると、その分管理領域としてメモリを必要とします。

２.３.２０ FAT ファイル名の各コンポネントの最大長

パス文字列を区切り文字で区切った際の、１つの文字列の最大長をバイト数で指定します。
この値には文字列最後の NULL 文字も含まれます。
最小値は「１３」です。ロングファイル名を使用する場合は使用するファイル名の最大長に合わせて設定してください。

２.３.２１ FAT ファイル名の各コンポネントの最大文字数

パス文字列を区切り文字で区切った際の、１つの文字列の最大長を文字数で指定します。
この値には文字列最後の NULL 文字も含まれます。
最小値は「１３」です。ロングファイル名を使用する場合は使用するファイル名の最大長に合わせて設定してください。

２.３.２２ FAT のフリークラスタキャッシュを取得する際のバッファサイズ

フリークラスタを検索する際に使用するバッファサイズを指定します。
指定したサイズのバッファが取得できない場合はキャッシュバッファが使用されます。
大きなバッファを使用することで効率よく検索ができます。
最小値は FAT のセクタサイズです。２の乗数ではなくサイズで設定します。

２.３.２３ フォーマット時の I/O バッファサイズ

フォーマットを行う際の使用する IO バッファのサイズを指定します。
フォーマット時に動的に取得/解放します。
最小値は FAT のセクタサイズです。２の乗数ではなくサイズで設定します。

２.４ ポーティング手順

以下、弊社製 USB マスストレージ統合キットを使用した場合のポーティング方法を示します。

２.４.１ ターゲット環境の確認

ポーティングの対象となる、ターゲット環境を確認します。

ここでは、以下のようなターゲットへのポーティングを行います。

- CPU　SH7727(SH3-DSP)
- メモリ　32MB
- OS　MIMOS（弊社製 iTRON4.0 互換 OS）
- 下位ドライバとして USB マスストレージ統合キットを使用

２.４.２ USB マスストレージ統合キットをポーティング

USB マスストレージ統合キットに付属している、各ドキュメントを参照し、USB マスストレージ統合キットに含まれる FSIF モジュールまでをポーティングします。

GR-FILE は、FSIF を下位のデバイスドライバとして動作しますので、FSIF までのプロトコルスタックをポーティングする必要があります。

２.４.３ GR-FILE のポーティングの準備

GR-FILE の src ディレクトリより、grp_fs ディレクトリと、mdep_vos ディレクトリを、ポーティング作業を行うディレクトリへコピーします。

２.４.４ 空のスタブファイルの実装

コピーを行ったファイルの内、下記ファイルはポーティング対象の為、処理が実装されていませんので、ターゲットに合わせてポーティングを行う必要があります。

- src / mdep_vos / base / grp_fs_dev_io.c
- src / mdep_vos / lib / grp_time / grp_time_get.c
- src / mdep_vos / lib / grp_time / grp_time_set.c

ここでは、USB マスストレージ統合キットに含まれる、FSIF をドライバとして使用するので、grp_fs_dev_io.c のポーティングは不要となります。

grp_time_get.c、grp_time_set.c は SH7727 の CPU マニュアルを参照して、内部を実装します。

(ファイルの日付情報が不要であれば、コメントを削除し、空のままで構いません)

また、下記ソースファイルに実装されている、grp_fs_printf 関数は文字列を出力する関数です。

- src / mdep_vos / base / grp_fs_mdep_if.c

文字の出力先は使用する環境により異なる為、環境に合わせたポーティングが必要となります。

文字の出力が不要であれば、grp_fs_printf 関数のポーティングは不要となりますので、grp_fs_printf 関数内部で使用している、cons_putchar 関数をコメントアウトして下さい。

２.４.５ ポーティングの確認

ポーティング対象ファイル/関数には、ポーティングの見落としを軽減する為、初期状態ではコンパイルエラーとなるように、コメントが記述されています。

環境/動作仕様によりポーティングの必要/不要がありますので、確認/ポーティングを行った後にコメントを削除するか、#ifdef 等でコンパイルしないようにして下さい。

ポーティングの必要なファイルは、以下のようになっています。

表 ２.４-１VOS 環境でのコメントの記述されているファイル

ディレクトリ/ファイル	内容
src / mdep_vos / base / grp_fs_dev_io.c	デバイスドライバインターフェーススタブ関数
src / mdep_vos / base / grp_fs_dev_sw_tbl.c	デバイススイッチテーブル
src / mdep_vos / base / grp_fs_mdep_if.c	grp_fs_printf 関数
src / mdep_vos / lib / grp_time / grp_time_get.c	現在時刻取得関数
src / mdep_vos / lib / grp_time / grp_time_set.c	現在時刻設定関数

表 ２.４-２μITRON 環境でのコメントの記述されているファイル

ディレクトリ/ファイル	内容
src / mdep_itron / base / grp_fs_dev_io.c	デバイスドライバインターフェーススタブ関数
src / mdep_itron / base / grp_fs_dev_sw_tbl.c	デバイススイッチテーブル
src / mdep_itron / base / grp_fs_mdep_if.c	grp_fs_printf 関数
src / mdep_itron / lib / grp_itron_id.h	セマフォア/メモリプール ID
src / mdep_itron / lib / grp_time / grp_time_get.c	現在時刻取得関数
src / mdep_itron / lib / grp_time / grp_time_set.c	現在時刻設定関数

２.４.６ コンパイルを行う

コピーした、grp_fs 以下の全てのファイルと、mdep_vos 以下の grp_fs_dev_io.c を除く、全てのファイルをコンパイルします。

ここでは、デバイスドライバとして FSIF のみを使用するため、grp_fs_dev_io.c のポーティング、コンパイルは行いませんが、FSIF 以外のデバイスドライバ等を、grp_fs_dev_io.c へ実装した場合は、grp_fs_dev_io.c のコンパイル、および、grp_fs_sysdef.h にあるコンパイルスイッチ「GRP_FS_PORTING_DEVICE_IO」を有効にする必要があります。grp_fs_dev_io.c へ実装を行う場合は **GR-FILE** 取扱説明書の「デバイスドライバインタフェース」を参照いただき、必要な機能を実装してください。

また、ご提供時のデバイススイッチテーブルには、grp_fs_dev_io.c を使用した場合のデバイス名は「dev」と定義されていますので、必要に応じて変更してください。

２.４.７ アプリケーションの設定

GR-FILE を使用するには、**GR-FILE** の初期化を行う必要があります。

また、ここでは mdep_vos を使用するので、可変長メモリプール管理は **GR-FILE** の機能を使用します。そのため、可変長メモリプール管理の初期化も行う必要があります。

初期化処理は以下のように行います。

- メモリプールの作成
- メモリプールの取得
- 可変長メモリプール管理の初期化
- **GR-FILE** の初期化

初期化処理については、サンプルアプリケーションを参照ください。gr_file_sample の場合は usb_test.c の GRUSB_Test_Init() を、gr_file_sample_no_console の場合は test_main.c の _usb_test_GrFileInit()を参照ください。

挿抜については、サンプルアプリケーション gr_file_sample の usb_test.c に記述されている test_event_task()を参照ください。また、test_event_task()内で挿抜時にマウント/アンマウント等を行う grp_fs_proc_event.c もサンプルとしてご提供していますので、参考にしてください。（gr_file_sample_no_console には挿抜によるマウント/アンマウント処理のサンプルはありません）

２.４.７.１ メモリプールの作成

GR-FILE で使用するメモリ領域を確保します。

注意事項として、メディアとのデータ通信に DMA などの CPU 以外が転送を行うような環境では、メモリ領域を非キャッシュ領域に確保する必要があります。

以下に使用例を示します。(l_ptPartPool は非キャッシュ領域に存在するものとします)

```
usRet = GRVOS_CreatePartitionPool( &l_ptPartPool, (UINT8*)"FT_PP", POOL_SIZE, 1 );
if( usRet != GRVOS_POS_RESULT ){
    return GRUSB_TEST_ERROR;
}
```

２.４.７.２ メモリプールの取得

２.４.７.１メモリプールの作成で作成したメモリプールより、メモリ領域を取得します。

ここで取得したメモリ領域を、管理情報やキャッシュに使用します。

以下に使用例を示します。

```
usRet = GRVOS_GetPartitionPool( l_ptPartPool, &pvMem, GRVOS_INFINITE );
if( usRet != GRVOS_POS_RESULT ){
    return GRUSB_TEST_ERROR;
}
```

２.４.７.３ 可変長メモリプール管理の初期化

取得したメモリ領域を可変長メモリプール管理用に初期化します。

以下に使用例を示します。

```
if( grp_mem_vl_init( pvMem, POOL_SIZE ) != 0 ){
    return GRUSB_TEST_ERROR;
}
```

２.４.７.４ GR-FILE の初期化

GR-FILE の初期化を行います。

初期化により、各種管理情報、キャッシュバッファの初期化、セマフォアの作成が行われ、GR-FILE を使用することが出来ます。

２.５ ポーティングの注意事項

ポーティングを行う際の注意事項を記します。

２.５.１ 文字列出力関数について(grp_fs_printf 関数)

GR-FILE には、誤動作や致命的なエラーの際に、エラー情報を出力したり、文字列を出力する機能があります。

この出力時に、grp_fs_printf 関数を使用しておりますが、出力部分はターゲットに依存する為、出力を行う場合は、この部分のポーティングが必要となります。

grp_fs_printf 関数は、grp_fs_mdep_if.c に実装されており、出力関数として cons_putchar 関数を呼ぶように実装されています。

ですので、cons_putchar 関数はアプリケーションでご用意ください。

また、文字列出力を行わない場合は、grp_fs_printf 関数の cons_putchar 関数をコメントアウトして下さい。

GR-FILE からエラー情報を出力する際には「FAT:」、または「GRP_FS:」で始まる文字列を出力します。多くの場合はデータをメディアに書き込めず、メディアのファイルシステムが破損する可能性のある場合などで出力されます。

また、grp_fs_printf 関数はアプリケーションからも使用が可能です。一般的な printf 関数相当の使用ができますが、マルチタスク環境下で複数のタスクが同時に使用する場合は出力される文字列が混ざって出力されることがありますのでご注意ください。

２.５.２ GR-FILE のメモリ領域について

CPU キャッシュが有効な環境で、メディアとのデータの転送に DMA や USB コントローラなどの CPU 以外を使用する場合は、**GR-FILE** の使用するメモリプール領域（grp_mem_vl_init()に指定するアドレス）を非キャッシュ領域に確保する必要があります。（NON_POSIX 定義を無効としている場合や、NON_POSIX を有効とし、メモリアロケートに OS 機能を使用する場合も同様に非キャッシュ領域のアドレスを取得できるようにポーティングを行ってください。）

これは、CPU 以外のコントローラでメディアの読み書きを行った場合、CPU のキャッシュ機能にヒットし、実際のデータとは異なるデータにアクセスしてしまう事があるためです。

同様の理由で **GR-FILE** の DirectIO 機能を使用する場合は、**GR-FILE** の API（read()/write()、grp_fs_read()/grp_fs_write()など）に指定するバッファも非キャッシュ領域に確保する必要があります。

２.５.３ メモリプールサイズについて

GR-FILE の使用するメモリプールのサイズは、「メモリプール概算.xls」で、その値を算出することが出来ます。

３．Version1.20 以前の GR-FILE をご使用のお客様へ

既に、旧バージョンの GR-FILE をご使用の場合は、GR-FILE のカスタムを行っていない場合に限り、設定値の移行と、ターゲット依存部の移行を行うことで、Version1.20 以降の構成へ移行できます。

以下に、旧バージョンよりどのように構成が変更されたかと、移動された定義、依存部分の移行について説明します。

旧バージョンでポーティングを行われている場合でも、本ポーティングマニュアルに目を通してください。

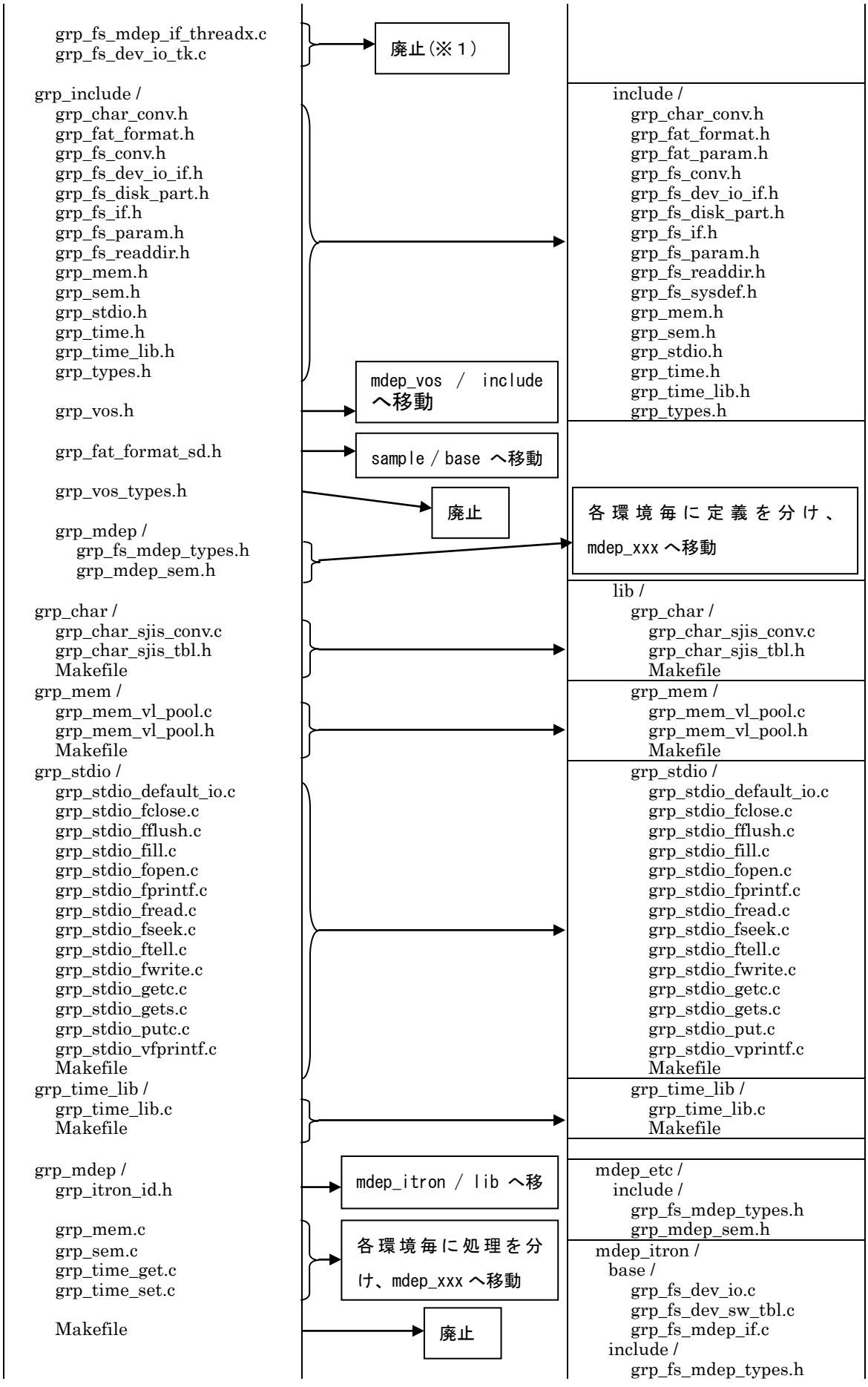
３.１ 旧バージョンよりの構成の変更について

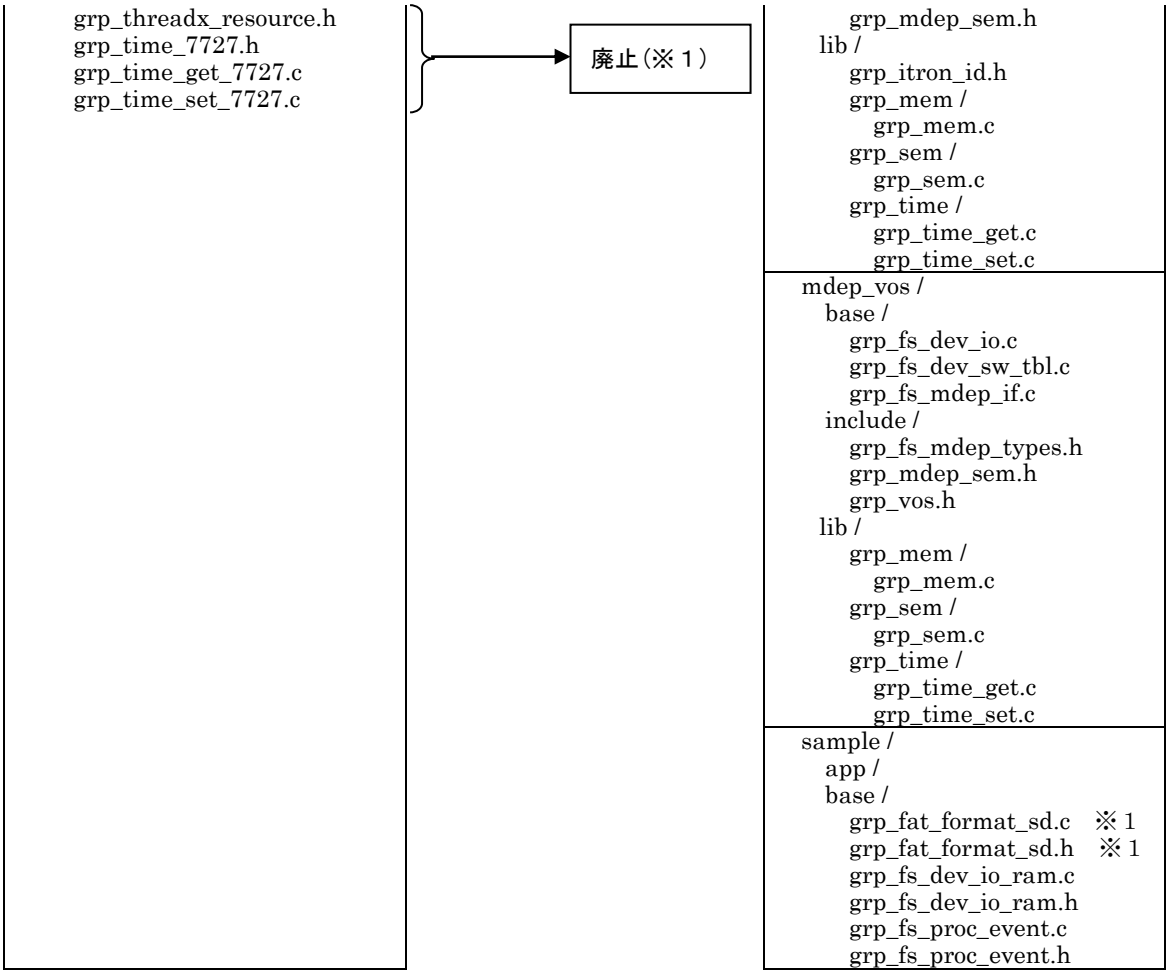
旧バージョンでは、ターゲットに依存する処理と依存しない処理が区別無く、ディレクトリに格納されていました。

また、ポーティングに必要なファイルも判りにくく、不要なファイルもありました。

Version1.20 より、これらの問題点を解消し判りやすくする為、表 ３.１-１構成の比較で示しますように、大きくターゲットに依存する部分と、依存の無い部分に分けました。

表 ３.１-１構成の比較


grp_fs_mdep_if_threadx.c
grp_fs_dev_io_tk.c
廃止（※１）
grp_include /
grp_char_conv.h
grp_fat_format.h
grp_fs_conv.h
grp_fs_dev_io_if.h
grp_fs_disk_part.h
grp_fs_if.h
grp_fs_param.h
grp_fs_readdir.h
grp_mem.h
grp_sem.h
grp_stdio.h
grp_time.h
grp_time_lib.h
grp_types.h
include /
grp_char_conv.h
grp_fat_format.h
grp_fat_param.h
grp_fs_conv.h
grp_fs_dev_io_if.h
grp_fs_disk_part.h
grp_fs_if.h
grp_fs_param.h
grp_fs_readdir.h
grp_fs_sysdef.h
grp_mem.h
grp_sem.h
grp_stdio.h
grp_time.h
grp_time_lib.h
grp_types.h
grp_vos.h
mdep_vos / include へ移動
grp_fat_format_sd.h
sample / base へ移動
grp_vos_types.h
廃止
grp_mdep /
grp_fs_mdep_types.h
grp_mdep_sem.h
各環境毎に定義を分け、mdep_xxx へ移動
lib /
grp_char /
grp_char_sjis_conv.c
grp_char_sjis_tbl.h
Makefile
grp_char /
grp_char_sjis_conv.c
grp_char_sjis_tbl.h
Makefile
grp_mem /
grp_mem_vl_pool.c
grp_mem_vl_pool.h
Makefile
grp_mem /
grp_mem_vl_pool.c
grp_mem_vl_pool.h
Makefile
grp_stdio /
grp_stdio_default_io.c
grp_stdio_fclose.c
grp_stdio_fflush.c
grp_stdio_fill.c
grp_stdio_fopen.c
grp_stdio_fprintf.c
grp_stdio_fread.c
grp_stdio_fseek.c
grp_stdio_ftell.c
grp_stdio_fwrite.c
grp_stdio_getc.c
grp_stdio_gets.c
grp_stdio_putc.c
grp_stdio_vfprintf.c
Makefile
grp_stdio /
grp_stdio_default_io.c
grp_stdio_fclose.c
grp_stdio_fflush.c
grp_stdio_fill.c
grp_stdio_fopen.c
grp_stdio_fprintf.c
grp_stdio_fread.c
grp_stdio_fseek.c
grp_stdio_ftell.c
grp_stdio_fwrite.c
grp_stdio_getc.c
grp_stdio_gets.c
grp_stdio_put.c
grp_stdio_vprintf.c
Makefile
grp_time_lib /
grp_time_lib.c
Makefile
grp_time_lib /
grp_time_lib.c
Makefile
grp_mdep /
grp_itron_id.h
mdep_itron / lib へ移
mdep_etc /
include /
grp_fs_mdep_types.h
grp_mdep_sem.h
grp_mem.c
grp_sem.c
grp_time_get.c
grp_time_set.c
各環境毎に処理を分け、mdep_xxx へ移動
mdep_itron /
base /
grp_fs_dev_io.c
grp_fs_dev_sw_tbl.c
grp_fs_mdep_if.c
include /
grp_fs_mdep_types.h
Makefile
廃止



※１　ご要望の場合のみご提供致します。

３.２ 設定値の移行について

GR-FILE で設定できる項目の多くは、ヘッダファイルで設定されています。

旧バージョンで設定された設定値を、Version1.20 のヘッダファイルへ反映することで移行が出来ます。

移動された定義を、表 ３.２-１ grp_fs_cfg.h から grp_fs_param.h へ移動された定義と表 ３.２-２ fat.h から grp_fat_param.h へ移動された定義に示します。

これらのファイルへ設定値を反映します。

表 ３.２-１ grp_fs_cfg.h から grp_fs_param.h へ移動された定義

定義	内容
GRP_FS_MAX_FSTYPE	ファイルシステムテーブルの最大エントリ数　※廃止
GRP_FS_MAX_MOUNT	同時に mount するファイルシステムの最大数
GRP_FS_MAX_FILE	同時にオープン可能なファイルの最大数
GRP_FS_MAX_FHDL	同時にオープン可能なファイルハンドルの最大数
GRP_FS_FBLK_SHIFT	ファイル管理ブロックキャッシュのブロックサイズのシフト値
GRP_FS_DBLK_SHIFT	ファイルデータキャッシュのブロックサイズのシフト値
GRP_FS_FBLK_CNT	ファイル管理ブロックキャッシュのブロック数
GRP_FS_DBLK_CNT	ファイルデータキャッシュのブロック数
GRP_FS_BLK_NHASH	キャッシュブロックのハッシングバケットの数
GRP_FS_MAX_TASK	ファイルシステムをアクセスするタスクの最大数
GRP_FS_FILE_NHASH	オープン中のファイルのハッシングバケット数
GRP_FS_FNAME_CACHE_CNT	ファイル名称キャッシュの最大数
GRP_FS_FNAME_NHASH	ファイル名称キャッシュのハッシングバケットの数

※廃止

GRP_FS_MAX_FSTYPE 定義は使用されていない為、Version1.20 より廃止されました。

表 ３.２-２ fat.h から grp_fat_param.h へ移動された定義

定義	内容
FAT_BLK_SHIFT	FAT セクタサイズのシフト値
FAT_MAP_CNT	オープン中の各 FAT ファイルの領域情報のキャッシュ数
FAT_FREE_TBL	フリーブロックキャッシュの数
FAT_COMP_SZ	FAT ファイル名の各コンポネントの最大長（NULL を含む）
FAT_COMP_CHCNT	FAT ファイル名の各コンポネントの最大文字数（NULL を含む）

３.３ 依存部分の移行について

Version1.20 より、ターゲット環境に依存する処理は、mdep_xxx ディレクトリに集められています。

旧バージョンでポーティングされている依存処理ファイルを、Version1.20 のディレクトリへファイル名を変更しながら上書きします。

ここで注意点として、デバイスドライバとの I/F である「grp_fs_dev_io.c」だけは、ご使用の環境に合わせる必要があります。

弊社製 FSIF をご使用の場合は、FSIF に含まれる「grp_fs_dev_io_grusb.c」を使用しますので、Version1.20 で提供される「grp_fs_dev_io.c」は不要となります。

これに合わせ、「grp_fs_cfg.c」で定義される、デバイススイッチテーブルも変更が必要です。

FSIF 以外のデバイスドライバをご使用されている場合でも、デバイススイッチテーブルの変更が必要な場合がありますので、ご確認下さい。

３.４ コンパイル環境の移行について

Version1.20 からは、コンパイルスイッチをヘッダファイル「grp_fs_sysdef.h」に定義しています。

全てのソースファイルは、「grp_fs_sysdef.h」をインクルードするように変更されていますので、「grp_fs_sysdef.h」を変更することで定義を変更できるようになりました。

これにより、**GR-FILE** のコンパイル時に統合環境での設定は不要となります。

組込み向けファイルシステム GR-FILE
ポーティングマニュアル

発行年月：2020 年 3 月　第 1.04 版
発行：株式会社グレープシステム

E-Mail　：　gr@support.grape.co.jp
URL　　：　http://www.grape.co.jp