AVR 8ビット マイクロ コントローラ

AVR42776：AVRでの小さいFATファイル システムの使い方

応用記述

序説

この応用記述は限定されたメモリのAVRデバイスでの小さいFATファイル システム ライブラリの使い方を記述します。そのような小さなメモリを持つデバイスで完全なSDC/MMCファイル システムのインターフェースは一般的に不可能ですが、これはこのようなデバイスでのファイル システムで限定されたインターフェースを許します。更にもっとメモリ使用量を最小にするためにライブラリ内の不要な関数を禁止することができます。小さいFatFs単位部はFatFs単位部の部分集合で、故にそれは限定された機能です。これはディスク入出力層から完全に独立したどのANSI C適合コンパイラに対しても独立した基盤で、これは更にディスク インターフェース用にデバイス特有関数が提供されることが必要なことを意味します。

特徴

- 制限されたメモリ消費でのSDC/MMCファイル システム アクセス
- デバイス非依存、Atmel® AVR®応用が最適化されたメモリ使用量を必要とするのに理想的
- FAT12、FAT16、FAT32を支援
- 単一ボリュームと単一ファイル専用
- ストリーミング ファイル読み込み
- いくつかの制限付きのファイル書き込み

目次

1. 小さいFATファイル システム

SDカードは大量の不揮発性データを格納する非常に便利な方法です。ファイル システムは一般的に可搬でPCから容易にアクセス可能なことをデータに許すために用いられます。工業標準で広く使用されるファイル システムがFATです。多くのAVR応用は最適化されたフラッシュ メモリとSRAMの使い方が必要で、限定されたインターフェースが必要とされる、または不充分な資源しか利用可能でない時に、標準FATファイル システム単位部は適合せず、故にELM-ChaNから入手可能な小さいFATファイル システム部分集合(Petit FatFs)単位部がン求められるかもしれません。これはAVR 8ビット マイクロ コントローラのような8ビット マイクロ コントローラ用に特に設計され、限定された資源を必要とする一方でFATファイル システムでフォーマットされたMMCまたはSDカードでの基本的なインターフェースを許します。これは基盤独立で故に機能的に低位ディスク入出力層が必要ですが、試供ドライバが利用可能です。

小さいFatFsライブラリによって提供される機能は以下を含みます。

- FATファイル システムを持つボリュームの装着(マウント)
- ファイルを開く
- ファイルを読む
- (いくつかの制限付きで)ファイルを書く
- 読み書きポインタを移動
- ディレクトリを開く
- ディレクトリ項目を読む

どの関数の内包や除外に関連する形態設定はコンパイルの大きさを最小化するために形態設定ヘッダ ファイル(pffconf.h)で定義されます。

2. 小さいFatFsの制限

Petitfsは特に可能な限り少しのフラッシュとスタックを使用します。けれども、これはいくつかの機能を犠牲にしてです。ファイルは作成や大きさを増すことができず、同時に1つのファイルだけをアクセスすることができます。セクタ アクセスの動作は機能の面で制限され、従ってSDC/MMCが多忙な場所で重要な期間があります。この多忙期間は毎回の部分的なセクタ アクセス操作に対して発生し、故に一度に完全なセクタを読みまたは書きすることができない応用に於いて、最大アクセス頻度が更に減ります。

2.1. 書き込み関数

小さいFatFs単位部は必要とするメモリを最小化するために、いくつかの制限付きの書き込み関数を含みます。これらの制限は以下を含みます。

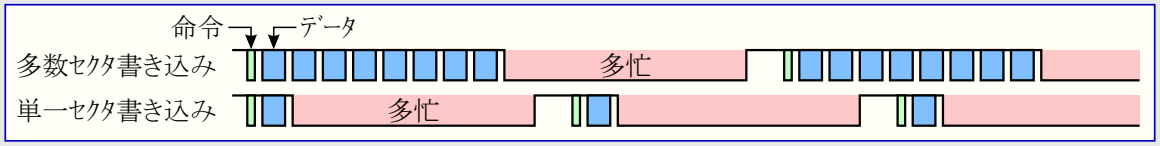
- ファイルを作成できず、既存ファイルにだけ書くことができます。
- ファイルを大きさに於いて拡張することができません。
- ファイルの時刻印を更新することができません。
- 書き込み操作はセクタ境界でだけ開始/停止します。
- 読み込み専用属性は書き込み操作を防ぎません。

これはファイルを書くためにはそれがSDC/MMCで既存で、割り当てられた必要な大きさを持たなければなりません。

2.2. ファイル アクセス効率

読みまたは書きの操作の効率は1操作でアクセスすることができるセクタ数で増されます。必要に迫られて必要なメモリを減らすため、小さいFatFsは多数セクタ アクセスと比べた時にアクセス効率が減らされる単一セクタ アクセスを使用します。例は多数セクタ書き込みと単一セクタ書き込みの両方に関連するカードの多忙時間を描く下図で見ることができます。これは実質的に最大読み書き頻度が減らされることを意味します。

図2-1. 多数対単一のセクタ書き込み



SDC/MMC記憶装置がバイト アクセス不可で、故に毎回の部分的な(1バイトからセクタの大きさの512バイトまでの)セクタ アクセス操作に対して、アクセスされるのに完全なセクタが必要なことに注意してください。例えば、バイト単位でのセクタ読み込みでセクタは512回読まれます。Petitfsで達成可能な最小多忙時間は同時に(セクタの大きさの)512バイトの読みまたは書きによってで、故にセクタは一度だけでアクセスされます。従ってデータは可能な限り長い塊で読みまたは書きされるべきです。

3. 関数

ﾎﾞﾘｭｰﾑ装着(ﾏｳﾝﾄ)

```
FRESULT pf_mount (
    FATFS*  fs      /* [IN] 作業領域への位置指示子 */
);
```

pf_mount()関数はどのｲﾝﾀｰﾌｪｰｽを試みるのに先立って使用されるべきです。これは記録ﾎﾞﾘｭｰﾑを装着(ﾏｳﾝﾄ)して小さいFatFs単位部に作業領域を与えます。

- ﾊﾟﾗﾒｰﾀ
  - fs：登録されるﾌｧｲｽ ｼｽﾃﾑ対象物(作業領域)への位置指示子
- 戻り値
  - FR_OK(0)　　：機能成功
  - FR_NOT_READY　　：ﾊｰﾄﾞ的な異常またはﾎﾞﾘｭｰﾑが存在しないため、ﾎﾞﾘｭｰﾑを装着(ﾏｳﾝﾄ)できません。
  - FR_DISK_ERR　　：ﾃﾞｨｽｸ読み込み機能で異常発生
  - FR_NO_FILESYSTEM：ﾄﾞﾗｲﾌﾞに有効なFATﾊﾟｰﾃｨｼｮﾝがありません。

ﾌｧｲﾙを開く

```
FRESULT pf_open (
    const char* path      /* [IN] ﾌｧｲﾙ名への位置指示子 */
);
```

pf_open()関数はﾌｧｲﾙに読み書きするのに先立って使用されるべきです。開いたﾌｧｲﾙは別のﾌｧｲﾙに対して再びopen関数が使用されるまで有効です。

- ﾊﾟﾗﾒｰﾀ
  - path：開くﾌｧｲﾙのﾌｧｲﾙ名を指定するﾇﾙ終端された文字列への位置指示子
- 戻り値
  - FR_OK(0)　　：機能成功
  - FR_NOT_FILE　　：ﾌｧｲﾙまたはﾊﾟｽを見つけることができません。
  - FR_DISK_ERR　　：ﾊｰﾄﾞ異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_ENABLED：ﾎﾞﾘｭｰﾑが装着(ﾏｳﾝﾄ)されていません。

ﾌｧｲﾙから読み込み

```
FRESULT pf_read (
    void* buff,    /* [OUT] 読み込み緩衝部への位置指示子 */
    UINT btr,      /* [IN]  読むべきﾊﾞｲﾄ数 */
    UINT* br       /* [OUT] 読んだﾊﾞｲﾄ数への位置指示子 */
);
```

pf_read()関数は開いたﾌｧｲﾙからﾃﾞｰﾀを読むのに使用されます。これはpffconf.hﾌｧｲﾙで_USE_READが1に書かれた時に利用可能です。ﾌｧｲﾙ ｼｽﾃﾑ対象物内のﾌｧｲﾙ位置指示子(fptr)は読んだﾊﾞｲﾄ数分増えます。機能成功で、*brはﾌｧｲﾙの最後を検出したことを調べられるべきで、関数の最後で*brがbtrよりも少ない場合、読み込み操作中にﾌｧｲﾙの最後に到達しました。buff引数がNULLとして与えられる場合、読み込みﾃﾞｰﾀはﾒﾓﾘ緩衝部に蓄えられるよりもむしろ出て行くｽﾄｰﾘｰﾑに転送されます。ｽﾄﾘｰﾐﾝｸﾞ関数は各ﾌﾟﾛｼﾞｪｸﾄに依存し、代表的にdisk_readp()関数で実装されるでしょう。

- ﾊﾟﾗﾒｰﾀ
  - buff：読んだﾃﾞｰﾀが格納されるべき緩衝部への位置指示子。ﾇﾙ位置指示子は出力ｽﾄｰﾘｰﾑに転送されつつある読み込みﾃﾞｰﾀに帰着します。
  - btr　：読むべきﾊﾞｲﾄ数
  - br　：戻りに於いて読んだﾊﾞｲﾄ数を示す変数への位置指示子
- 戻り値
  - FR_OK(0)　　：機能成功
  - FR_DISK_ERR　　：ﾊｰﾄﾞ異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_OPENED　：ﾌｧｲﾙが開かれていません。
  - FR_NOT_ENABLED：ﾎﾞﾘｭｰﾑが装着(ﾏｳﾝﾄ)されていません。

ファイルに書く

```
FRESULT pf_write (
    const void* buff,     /* [IN]  書かれるべきデータへの位置指示子 */
    UINT btw,             /* [IN]  書かれるべきバイト数 */
    UINT* bw              /* [OUT] 書かれたバイト数を返す変数への位置指示子 */
);
```

pf_write()関数は開いたファイルにデータを書くのに使用されます。これはpffconf.hファイルで_USE_WRITEが1に書かれた時に利用可能です。ファイル システム対象物内のファイル位置指示子(fptr)は書いたバイト数分増えます。機能成功で、*bwはファイルの最後を検出したことを調べられるべきで、関数の最後で*bwがbtwよりも少ない場合、書き込み操作中にファイルの最後に到達しました。

制限

- ファイル作成不可
- 既存ファイルの大きさ増加不可
- ファイルの時刻印、即ち”最終変更”属性の更新不可
- 書き込み操作はセクタ境界でだけ開始と停止ができます。
- 書き込み操作は読み込み専用属性を持つファイルによって妨げられません。

ファイル書き込みは以下の手順で完了されなければなりません。

1. pf_lseek(ofs)：読み書き位置指示子は書き込み操作を始める前にセクタ境界に移動されなければなりません。これが行われなかった場合、位置指示子は最も近いセクタ境界に切り下げで丸められます。
2. pf_write(buff, btw, &bw)：ファイルに対する書き込みデータ。この関数は必要な場合、即ち、データが継続的に生成されつつある場合に繰り返して呼ぶことができます。けれども、次の段階で書き込みが終了されるまでデータは完全に完遂されないでしょう。
3. pf_write(0, 0, &bw)：書き込み操作終了。読み書き位置指示子がセクタ境界でなければ、セクタ内の残りのバイトは0で満たされます。この段階は全ての書き込みデータを完遂するように完了されなければなりません。

- パラメータ
  - buff：書かれるべきデータが格納された緩衝部への位置指示子。ヌル位置指示子は書き込み操作が終了されるべきであることを示します。
  - btw：書くバイト数
  - bw：戻りに於いて書かれたバイト数を示す変数への位置指示子
- 戻り値
  - FR_OK(0)：機能成功
  - FR_DISK_ERR：ハードド異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_OPENED：ファイルが開かれていません。
  - FR_NOT_ENABLED：ボリュームが装着(マウント)されていません。

ファイル位置指示子を移動

```
FRESULT pf_lseek (
    DWORD ofs      /* [IN] バイト単位でのファイル内変位 */
);
```

pf_lseek()関数は開いたファイル内の読み書き位置指示子を移動するのに使用されます。これはpffconf.hファイルで_USE_LSEEKが1に書かれた時に利用可能です。この変位はファイルの先頭に相対して指定されるべきです。ファイル システム対象物のファイル位置指示子(fptr)がこの変更を反映します。

- パラメータ
  - ofs：読み書き位置指示子が移動されるべき場所がファイルの始めからバイトで指定されます。
- 戻り値
  - FR_OK(0)：機能成功
  - FR_DISK_ERR：ハードド異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_OPENED：ファイルが開かれていません。

ディレクトリを開く

```
FRESULT pf_opendir (
    DIR* dp,              /* [OUT] 空ディレクトリ対象物構造体への位置指示子 */
    const char* path      /* [IN]  ディレクトリ名への位置指示子 */
);
```

pf_opendir()関数は既存のディレクトリを開くのに使用され、特別な手続きなしに何時でも破棄することができる後の参照用のディレクトリ対象物を作成します。これはpffconf.hファイルで_USE_DIRが1に書かれた時に利用可能です。

- パラメータ
  - dp　：作成される空のディレクトリ対象物への位置指示子
  - path：開くディレクトリのディレクトリ名を指定するヌル終端された文字列への位置指示子
- 戻り値
  - FR_OK(0)　　：機能成功
  - FR_NOT_FILE　：パスを見つけることができません。
  - FR_NOT_READY　：ボリュームがありません。
  - FR_DISK_ERR　：ハードド異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_ENABLED：ボリュームが装着(マウント)されていません。

ディレクトリを読む

```
FRESULT pf_readdir (
    DIR* dp,        /* [IN]  開いたディレクトリ対象物への位置指示子 */
    FILINFO* fno    /* [OUT] ファイル情報構造体への位置指示子 */
);
```

pf_readdir()関数はディレクトリ エントリを順番通りに読みます。これはpffconf.hファイルで_USE_DIRが1に書かれた時に利用可能です。この関数を繰り返して呼ぶことによってディレクトリ内の全ての項目を読むことができます。全てのディレクトリ エントリが読まれてしまうと、関数は異常を返すことなく、ファイル情報構造体内のf_name[]メンバ内にヌル文字列を置きます。ヌル位置指示子としてfnoが与えられると、ディレクトリ対象物の読み込み指標が開始位置に戻されます。

- パラメータ
  - dp　：開いたディレクトリ対象物への位置指示子
  - fno：読んだ項目が格納されるファイル情報構造体への位置指示子
- 戻り値
  - FR_OK(0)　　：機能成功
  - FR_DISK_ERR　：ハードド異常、不正なFAT構造体、または内部異常のために機能失敗
  - FR_NOT_OPENED：ディレクトリ対象物が開かれていません。

4. 単位部メモリ使用量と形態設定

AVG-GCCを用いてコンパイルする時の小さいFatFs単位部の大きさは使用される形態設定と最適化の設定に依存して1～4Kバイトの間で変わります。右表は大きさ最適化でAtmel Studio7.0を使用して各種形態設定任意選択でのコンパイルの大きさの例を示します。既定形態設定は_USE_READと_FS_FAT16が許可で、他の全ての任意選択は禁止されます。これらの概算は小さいFatFs単位部だけに対してで、ディスク入出力層を含みません。

表4-1. 小さいFatFs単位部のメモリ使用例

項目	大きさ (バイト)
フラッシュ メモリ (既定形態設定)	2064
フラッシュ メモリ (_USE_READ抜き)	-416
フラッシュ メモリ (_USE_DIR追加)	+688
フラッシュ メモリ (_USE_LSEEK追加)	+556
フラッシュ メモリ (_USE_WRITE追加)	+486
SRAM (bss)	1
SRAM (data)	42

形態設定は応用によって何の機能が必要とされるかに従って独自化することができます。右表は形態設定ヘッダ ファイル(pffconf.h)に含まれる各形態設定任意選択に対してどの関数が取り去られるかを示します。

表4-2. 形態設定任意選択によって取り去られる関数

関数	_USE_READ 0	_USE_DIR 0	_USE_LSEEK 0	_USE_WRITE 0
pf_mount				
pf_open				
pf_read	×			
pf_lseek			×	
pf_opendir		×		
pf_readdir		×		
pf_write				×

5. 参照

これに関する単位部と情報はhttp://elm-chan.org/fsw/ff/00index_p.htmlのELM-ChaNから提供されました。

6. Atmel STARTからのソース コード取得

例のｺｰﾄﾞは図画によるしよ使用者ｲﾝﾀｰﾌｪｰｽ(GUI)を通して応用ｺｰﾄﾞの形態設定を許すｳｪﾌﾞに基づくツールのAtmel STARTを通して入手可能です。このｺｰﾄﾞは下の**例**のﾘﾝｸ、またはAtmel STARTの扉ﾍﾟｰｼﾞの例閲覧(BROWSE EXAMPLES)釦経由でAtmel Studio 7.0とIAR™ IDEの両方に対してﾀﾞｳﾝﾛｰﾄﾞすることができます。

ｳｪﾌﾞﾍﾟｰｼﾞ : http://start.atmel.com/
資料 : http://start.atmel.com/static/help/index.html
例 : http://start.atmel.com/#examples

例閲覧部に於いて、AVR42776 Petit FatFs Exampleを探してください(例ﾌﾟﾛｼﾞｪｸﾄに対する詳細な必要条件についてはAtmel Studioで使用者の手引き(User Guide)を押してください)。

ﾀﾞｳﾝﾛｰﾄﾞした.atzipﾌｧｲﾙをﾀﾞﾌﾞﾙｸﾘｯｸしてください。するとﾌﾟﾛｼﾞｪｸﾄがAteml Studio 7.0にｲﾝﾎﾟｰﾄされるでしょう。

IARにﾌﾟﾛｼﾞｪｸﾄをｲﾝﾎﾟｰﾄする方法の情報については上の**資料**のﾘﾝｸを押し、'Atmel Start Output in External Tools'と'IAR Embedded Workbench®'を選択してください。

7. 改訂履歴

資料改訂	日付	注釈
42776A	2016年10月	初版資料公開

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

---

本応用記述はAtmelのAVR42776応用記述(Rev.42776A-10/2016)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。