お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

ユーザーズ・マニュアル

RX-FS850

ファイル・システム

対象デバイス

V850 シリーズ

対象リアルタイム OS

RX850 Pro Ver.3.15

資料番号　U15637JJ4V0UM00 (第 4 版)
発行年月日　September 2003　CP(K)

〔メ　モ〕

目次要約

- TRON は，“ The Real-time Operating System Nucleus ” の略称です。
- ITRON は，“ Industrial TRON ” の略称です。
- μITRON は，“ Micro Industrial TRON ” の略称です。
- TRON，ITRON，および，μITRON は，特定の商品ないしは商品群を指す名称ではありません。
- μITRON4.0 仕様は，トロン協会 ITRON 部会が中心となって策定されたオープンなリアルタイムカーネル仕様です。
- μITRON4.0 仕様の仕様書は，ITRON プロジェクトホームページ (http://www.tron.org/) から入手が可能です。

- V850 シリーズは，NEC エレクトロニクス株式会社の商標です。
- IBM，PC/AT は，米国 International Business Machines, Inc. 社の登録商標です。
- Windows，Windows NT は，米国 Microsoft Corporation の米国，および，その他の国における登録商標，または，商標です。
- Sun，Sun Microsystems，Solaris は，米国 Sun Microsystems, Inc. の米国，および，その他の国における登録商標，または，商標です。
- SPARC は，米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国，および，その他の国における登録商標，または，商標です。
- UNIX は，X/Open Company Limited が独占的にライセンスしている米国，および，その他の国における登録商標です。
- Green Hills Software は，米国 Green Hills Software, Inc. の商標です。
- その他，記載の会社名 / 製品名は，各社の商標，または，登録商標です。

- 本資料に記載されている内容は2003年9月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

〔メ　モ〕

はじめに

対 象 者　このマニュアルは，V850 シリーズ応用システムを設計，開発するユーザを対象としています。

目　　的　このマニュアルは，RX-FS850 の機能の全般を理解していただくことを目的としています。

構　　成　このマニュアルは，次の内容で構成されています。

- **概説**
- **インストレーション**
- **システム構築**
- **入出力管理機能**
- API **関数**
- RX-FS850 **依存部**

読 み 方　このマニュアルの読者には，マイクロコンピュータ，C 言語，アセンブリ言語，リアルタイム OS，ファイル・システムに関する一般知識を必要とします。

V850 シリーズのハードウエア機能，命令機能を知りたいとき
→　各製品のユーザーズ・マニュアルを参照してください。

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2 進数　…XXXX または XXXXB
	10 進数　…XXXX
	16 進数　…XXXXH

2 のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）：$2^{10} = 1024$
M（メガ）：$2^{20} = 1024^2$
G（ギガ）：$2^{30} = 1024^3$

関連資料　　このマニュアルを使用する場合は，次の資料もあわせてご覧ください。
関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。
あらかじめご了承ください。

○ V850 **シリーズに関する資料（ユーザーズ・マニュアル）**

資料名		資料番号	
		和文	英文
CA850 Ver.2.50 C **コンパイラ・パッケージ**	**操作編**	U16053J	U16053E
	C **言語編**	U16054J	U16054E
	アセンブリ言語編	U16042J	U16042E
PM plus Ver.5.10		U16569J	U16569E
RX850 Pro Ver.3.13 **リアルタイム** OS	**基礎編**	U13773J	U13773E
	インストレーション編	U13774J	U13774E
	テクニカル編	U13772J	U13772E

目次

図目次

表目次

第 1 章　概説

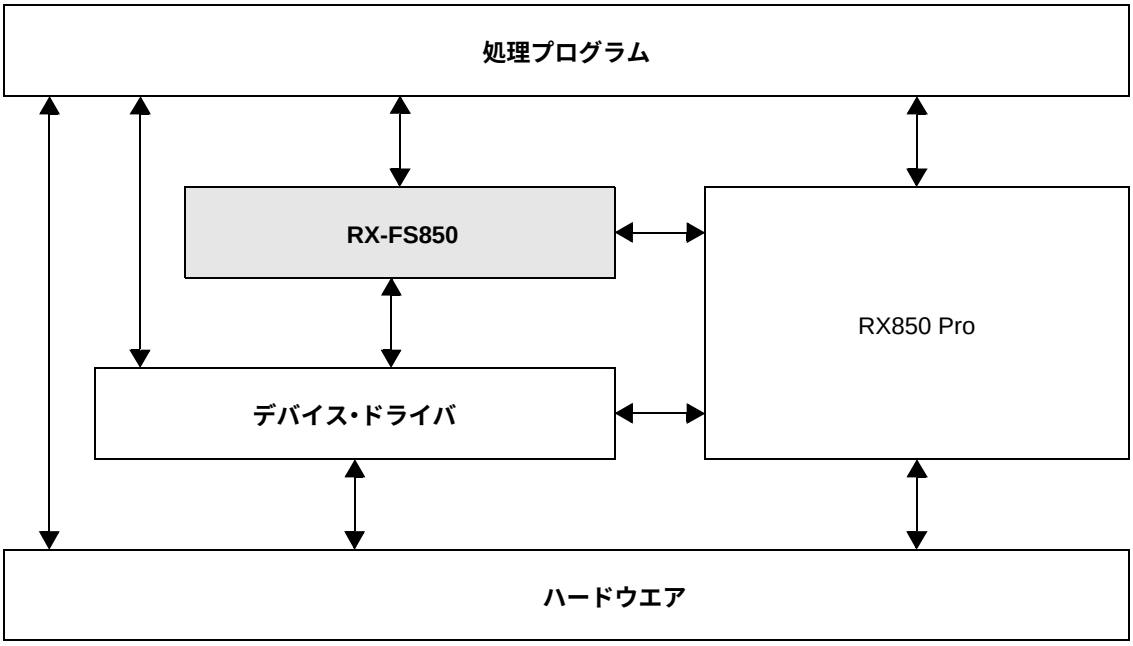
1.1　概要

RX-FS850 は，FAT ファイル・システム機能，および，CD-ROM ファイル・システム機能をサポートしたファイル・システムであり，組み込み型制御用リアルタイム・オペレーティング・システム RX850 Pro (μITRON3.0 仕様準拠，NEC エレクトロニクス製) 上で動作する処理プログラムに対し，ブロック型デバイス (ATA フラッシュ・カード，IDE ハード・ディスク，RAM ディスクなど) への高速な入出力処理を実現するためのアプリケーション・プログラム・インタフェース関数 (Application Program Interface 関数) を提供しています。

したがって，ユーザは，RX-FS850 が提供する API 関数を利用することにより，ブロック型デバイスへの高速な入出力処理を容易に実現することが可能となります。

図 1-1 に，RX-FS850 の位置付けを示します。

図 1-1　RX-FS850 の位置付け



1.2　特徴

以下に，RX-FS850 の特徴を示します。

- FAT ファイル・システムをサポート

  FAT12，FAT16，FAT32 といった FAT ファイル・システムをサポートしています。
  なお，RX-FS850 では，VFAT(最大 254 文字のロング・ファイル・ネーム) についてもサポートしています。

- CD-ROM ファイル・システムをサポート

  Joliet，ISO9660 Level1 といった CD-ROM ファイル・システムをサポートしています。
  なお，RX-FS850 では，マルチセッション CD (“ リードイン～データ～リードアウト ” で構成されるセッションを 1 メディア内に複数持った CD) についてもサポートしています。

- 自動バッファリング機能をサポート

  各種データの入出力処理においてバッファリングが行われています。このため，ユーザが処理プログラムを記述する際，入出力処理に伴うバッファリングを意識する必要がありません。
  なお，RX-FS850 では，入出力処理で利用可能な “ バッファの総数 ” を初期化時に指定させているため，ユーザの実行環境 (メモリ容量) に応じたサイズを割り当てることができます。

- ファイル・ロック機能をサポート

  書き込み可能モードでオープンされているファイルに対する資源の競合を避けるための機能 (ファイル・ロック機能) を提供しています。
  これにより，同一ファイルに対する同時書き込みによるデータ破壊を防止しています。

- プラグ・アンド・プレイ機能をサポート

  動的にデバイスの接続 / 取り外しを可能とするための機能 (プラグ・アンド・プレイ機能) を提供しています。

- データの書き込みタイミング

  データの書き込み要求が発生した際，実際に該当ファイルへ書き込むタイミングとして非同期モード，同期モードの 2 種類を提供しています。

  - 非同期モード

    書き込み要求が発生した際，ただちに該当ファイルへの書き込みを行わず，いったん，バッファに対して書き込みを行います。
    なお，該当ファイルに対する書き込み処理は，バッファがフラッシュされた際，または，該当ファイルがクローズされた際に行われます。

  - 同期モード

    書き込み要求が発生した際，ただちに該当ファイルへの書き込みを行います。

- 多様なファイル・アクセス方式

  ルート・ディレクトリからの絶対パス，および，カレント・ディレクトリからの相対パスによるツリー構造アクセスをサポートしています。

- 高い移植性

  RX-FS850 が処理を実行するうえで必要となるハードウエアの初期化処理，および，ユーザの実行環境 / アプリケーション・システムに依存した処理については，ユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。
  これにより，実行環境への移植性を向上させるとともに，カスタマイズ化を容易なものとしています。

- ROM 化の実現

  実行環境上に組み込んで使用することを想定したファイル・システムであるため，ROM 化を意識し，コンパクトな設計が行われています。
  また，RX-FS850 が提供する各種機能 (API 関数) のうち，システムで使用する機能のみをシステム構築時 (ロード・モジュール生成時) にリンクさせているため，コンパクトでありながら，ユーザのニーズに最適なファイル・システムを構築することができます。

- マルチタスク処理を意識した設計

  RX-FS850 が提供する API 関数では，マルチタスク処理を考慮した設計が行われています。このため，ユーザが処理プログラムを記述する際，API 関数の発行に伴うタスク間の排他制御などを意識する必要がありません。

1.3　実行環境

以下に，RX-FS850 が処理を実行するうえで必要となるハードウエアを示します。

- プロセッサ
  以下に，RX-FS850 が処理を実行するうえで必要となるプロセッサを示します。

  V850 シリーズ V850E/*xxx*，または，V850ES/*xxx*

- 周辺コントローラ
  RX-FS850 では，処理を実行するうえで，特定の周辺コントローラは必要ありません。

1.4　開発環境

以下に，RX-FS850 を使用した処理プログラムを開発するうえで必要となるハードウエア，および，ソフトウエアを示します。

- ハードウエア
  - ホスト・マシン (下記のいずれか)

PC98-NX シリーズ	：Windows 2000，98，Me，XP，NT 4.0
IBM PC/AT 互換機	：Windows 2000，98，Me，XP，NT 4.0
SPARC station	：Solaris Rel.2.5.*x*

- ソフトウエア
  - リアルタイム OS

RX850 Pro Ver.3.15 以上	：NEC エレクトロニクス製

  - C コンパイラ・パッケージ (下記のいずれか)

CA850 Ver.2.41 以上	：NEC エレクトロニクス製
CCV850E Ver.1.8.9 Rel.4.0.2 以上	：米国 Green Hills Software, Inc. 製

第 2 章　インストレーション

本章では，RX-FS850 の提供媒体に格納されているファイル群をユーザの開発環境 (ホスト・マシン) 上にインストールする際の手順について解説しています。

2.1　概要

RX-FS850の提供媒体は，ホスト・マシンの種類(Windowsベース，UNIXベース)に合わせて計2種類が用意されています。表 2-1 に，RX-FS850 の提供形式一覧を示します。

表 2-1　RX-FS850 の提供形式

ホスト・マシン	提供形式	提供媒体
Windows ベース • PC98-NX シリーズ • IBM PC/AT 互換機	オブジェクト・ファイル形式 • CA850 対応版 • GHS 製コンパイラ対応版	CD-ROM
UNIX ベース • SPARC station	オブジェクト・ファイル形式 • CA850 対応版 • GHS 製コンパイラ対応版	CD-ROM

注意　ホスト・マシンの種類別に用意された提供媒体には，2 種類 (CA850 対応版オブジェクト・ファイル形式，GHS 製コンパイラ対応版オブジェクト・ファイル形式) の RX-FS850 が格納されています。したがって，提供媒体からホスト・マシン上にファイル群をインストールする際には，ユーザが使用する C コンパイラ・パッケージに対応した RX-FS850 をインストールする必要があります。

2.2　インストール手順

RX-FS850 の提供媒体に格納されているファイル群のインストール手順は，ホスト・マシンの種類 (Windows ベース，UNIX ベース) により異なります。

そこで，以降に，ホスト・マシンが Windows ベースの場合，UNIX ベースの場合のインストール手順をそれぞれに示します。

2.2.1　Windows ベース

以下に，RX-FS850 の提供媒体に格納されているファイル群をホスト・マシン (Windows ベース：PC98-NX シリーズ，IBM PC/AT 互換機) 上にインストールする際の手順を示します。

1) Windows の起動
ホスト・マシン，および，周辺機器などの電源を投入し，Windows を起動します。

2) 提供媒体のセット
RX-FS850 の提供媒体をホスト・マシンの該当デバイス装置 (CD-ROM ドライブ) にセットすることにより，セットアップ・プログラムが自動実行します。
以降，モニタ画面に表示されるメッセージに従ってインストール作業を実行します。

注意　セットアップ・プログラムが自動実行しない場合には，RX-FS850 の提供媒体のディレクトリ RX-FS_V850E_NEC\DISK1 に格納されている SETUP.EXE を起動します。

3) ファイル群の確認
Windows の標準アプリケーション Explorer などを用いて，RX-FS850 の提供媒体に格納されていたファイル群がホスト・マシン上にインストールされたことを確認します。
なお，各ディレクトリについての詳細は，「2.3　ディレクトリ構成」を参照してください。

2.2.2　UNIX ベース

以下に，RX-FS850 の提供媒体に格納されているファイル群をホスト・マシン (UNIX ベース：SPARC station) 上にインストールする際の手順を示します。
ただし，入力例中の “ % ” はシェル・プロンプトを，“ △ ” はスペース・キーの入力を，“ <Enter> ” はエンター・キーの入力を表しています。

1) ホスト・マシンへのログイン
ホスト・マシンにログインします。

```
%
```

2) ディレクトリの移動
cd コマンドを実行し，インストール用ディレクトリに移動します。
なお，下記入力例では，インストール用ディレクトリとして /usr/local を指定しています。

注意　インストール用ディレクトリのパーミッション (read，write，execute) はインストール作業者に対して許可状態である必要があります。そこで，インストール用ディレクトリのパーミッションが不許可状態であった場合には，chmod コマンドを実行し，パーミッションを不許可状態から許可状態に変更します。

```
% cd △ /usr/local <Enter>
```

3) 提供媒体のセット
RX-FS850 の提供媒体をホスト・マシンの該当デバイス装置 (CD-ROM ドライブ) にセットします。

4) デバイスのマウント
mount コマンドを実行し，該当デバイス装置に対応したデバイスをマウントします。
なお，下記入力例では，該当デバイス装置のデバイス名 (スペシャル・ファイル名) として /dev/rst8 を，マウント・ディレクトリとして /cdrom を指定しています。

注意　ホスト・マシンによっては，“ デバイスのマウント ” が自動的に行われるものがあります。このような場合，mount コマンドを実行する必要はありません。

```
% mount △ /dev/rst8 △ /cdrom <Enter>
```

5) ファイル群のインストール
tar コマンドを実行し，マウント・ディレクトリ /cdrom 下の圧縮ファイルをインストール用ディレクトリに展開します。
ただし，提供媒体には，以下に示した 2 種類の圧縮ファイルが格納されています。

- CA850 対応版
- GHS 製コンパイラ対応版

そこで，ユーザが使用する C コンパイラ・パッケージが CA850 の場合は圧縮ファイル nec/rxfs.tar を，GHS 製コンパイラの場合は圧縮ファイル ghs/rxfs.tar を展開します。

【 CA850 対応版の場合 】

```
% tar △ -xvof △ /cdrom/RXFS/nec/rxfs.tar <Enter>
```

【 GHS 製コンパイラ対応版の場合 】

```
% tar △ -xvof △ /cdrom/RXFS/ghs/rxfs.tar <Enter>
```

6) ファイル群の確認
ls コマンドを実行し，RX-FS850 の提供媒体に格納されていたファイル群がホスト・マシン上にインストールされたことを確認します。
なお，各ディレクトリについての詳細は，「2.3　ディレクトリ構成」を参照してください。

```
% ls △ -CFR △ /usr/local/nectools32 <Enter>
```

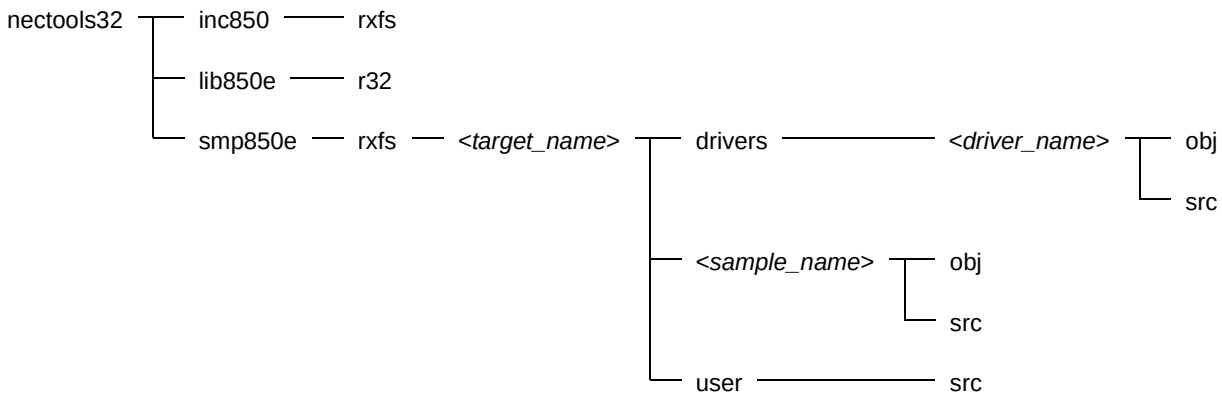
2.3　ディレクトリ構成

RX-FS850 の提供媒体に格納されているファイル群のディレクトリ構成は，ユーザが使用する C コンパイラ・パッケージの種類 (CA850，GHS 製コンパイラ) により異なります。
そこで，以降に，C コンパイラ・パッケージが CA850 の場合，GHS 製コンパイラの場合のディレクトリ構成をそれぞれに示します。

2.3.1　CA850 対応版

図 2-1 に，RX-FS850 の提供媒体 (CA850 対応版) に格納されているファイル群をホスト・マシン上にインストールした際に生成されるディレクトリ構成を示します。

図 2-1　ディレクトリ構成 (CA850 対応版)



以下に，各ディレクトリの概要を示します。

1) nectools32\inc850
RX-FS850 の標準ヘッダ・ファイルが格納されているディレクトリです。

rxfs.h	：RX-FS850 用標準ヘッダ・ファイル

2) nectools32\inc850\rxfs
RX-FS850 のヘッダ・ファイルが格納されているディレクトリです。

3) nectools32\lib850e\r32
RX-FS850 のライブラリ・ファイル (32 レジスタ・モード) が格納されているディレクトリです。

librxfs.a	：API 関数処理部ライブラリ
libfscmn.a	：共通処理部ライブラリ
libfsfat.a	：FAT 処理部ライブラリ
libfsnofat.a	：FAT 処理部ダミー・エントリ・ライブラリ
libfscd.a	：CD-ROM 処理部ライブラリ
libfsnocd.a	：CD-ROM 処理部ダミー・エントリ・ライブラリ

4) nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\obj
デバイス・ドライバを生成するためのコマンド・ファイルが格納されているディレクトリです。
なお，本ディレクトリのコマンド・ファイルを用いることにより，デバイス・ドライバ lib<*driver_name*>.a が本ディレクトリに生成されます。

【 Windows ベース 】

lib<*driver_name*>.prj	：デバイス・ドライバ用プロジェクト・ファイル
lib<*driver_name*>.prw	：デバイス・ドライバ用ワークスペース・ファイル

【 UNIX ベース 】

Makefile	：デバイス・ドライバ用メイク・ファイル

5) nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src
デバイス・ドライバのソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。

6) nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\obj
ロード・モジュールを生成するためのコマンド・ファイルが格納されているディレクトリです。
なお，本ディレクトリのコマンド・ファイルを用いることにより，ロード・モジュール sample.out **が本ディレクトリに生成されます。**

【 Windows **ベース】**
sample.prj　　**：ロード・モジュール用プロジェクト・ファイル**
sample.prw　　**：ロード・モジュール用ワークスペース・ファイル**

【 UNIX **ベース】**
Makefile　　**：ロード・モジュール用メイク・ファイル**

7) nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\src
サンプル・プログラムのソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。

8) nectools32\smp850e\rxfs\<*target_name*>\user\src
RX-FS850 **依存部のソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。**

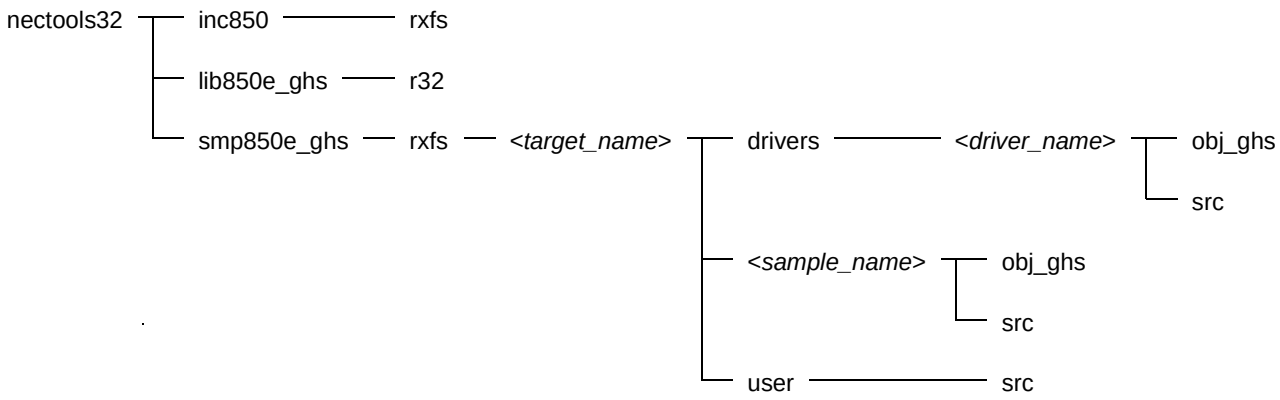
注意 <*target_name*>，<*driver_name*>，<*sample_name*> **についての詳細は，下記に示したテキスト・ファイルを参照してください。**

<*target_name*>　：nectools32\smp850e\rxfs\README.TXT
<*driver_name*>　：nectools32\smp850e\rxfs\README.TXT
<*sample_name*>　：nectools32\smp850e\rxfs\README.TXT

2.3.2　GHS 製コンパイラ対応版

図 2-2 に，RX-FS850 の提供媒体 (GHS 製コンパイラ対応版) に格納されているファイル群をホスト・マシン上にインストールした際に生成されるディレクトリ構成を示します。

図 2-2　ディレクトリ構成 (GHS 製コンパイラ対応版)



以下に，各ディレクトリの概要を示します。

1) nectools32\inc850
RX-FS850 の標準ヘッダ・ファイルが格納されているディレクトリです。

rxfs.h	：RX-FS850 用標準ヘッダ・ファイル

2) nectools32\inc850\rxfs
RX-FS850 のヘッダ・ファイルが格納されているディレクトリです。

3) nectools32\lib850e_ghs\r32
RX-FS850 のライブラリ・ファイル (32 レジスタ・モード) が格納されているディレクトリです。

librxfs.a	：API 関数処理部ライブラリ
libfscmn.a	：共通処理部ライブラリ
libfsfat.a	：FAT 処理部ライブラリ
libfsnofat.a	：FAT 処理部ダミー・エントリ・ライブラリ
libfscd.a	：CD-ROM 処理部ライブラリ
libfsnocd.a	：CD-ROM 処理部ダミー・エントリ・ライブラリ

4) nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\obj_ghs
デバイス・ドライバを生成するためのコマンド・ファイルが格納されているディレクトリです。
なお，本ディレクトリのコマンド・ファイルを用いることにより，デバイス・ドライバ lib<*driver_name*>.a が本ディレクトリに生成されます。

lib<*driver_name*>.bld	：デバイス・ドライバ用ビルド・ファイル

5) nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src
デバイス・ドライバのソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。

6) nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\obj_ghs
ロード・モジュールを生成するためのコマンド・ファイルが格納されているディレクトリです。
なお，本ディレクトリのコマンド・ファイルを用いることにより，ロード・モジュール sample.out が本ディレクトリに生成されます。

sample.bld	：ロード・モジュール用ビルド・ファイル

7) nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\src
サンプル・プログラムのソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。

8) nectools32\smp850e_ghs\rxfs\<*target_name*>\user\src
RX-FS850 **依存部のソース・ファイル，および，ヘッダ・ファイルが格納されているディレクトリです。**

注意 <*target_name*>，<*driver_name*>，<*sample_name*> **についての詳細は，下記に示したテキスト・ファイルを参照してください。**

<*target_name*>　：nectools32\smp850e_ghs\rxfs\README.TXT
<*driver_name*>　：nectools32\smp850e_ghs\rxfs\README.TXT
<*sample_name*>　：nectools32\smp850e_ghs\rxfs\README.TXT

第 3 章　システム構築

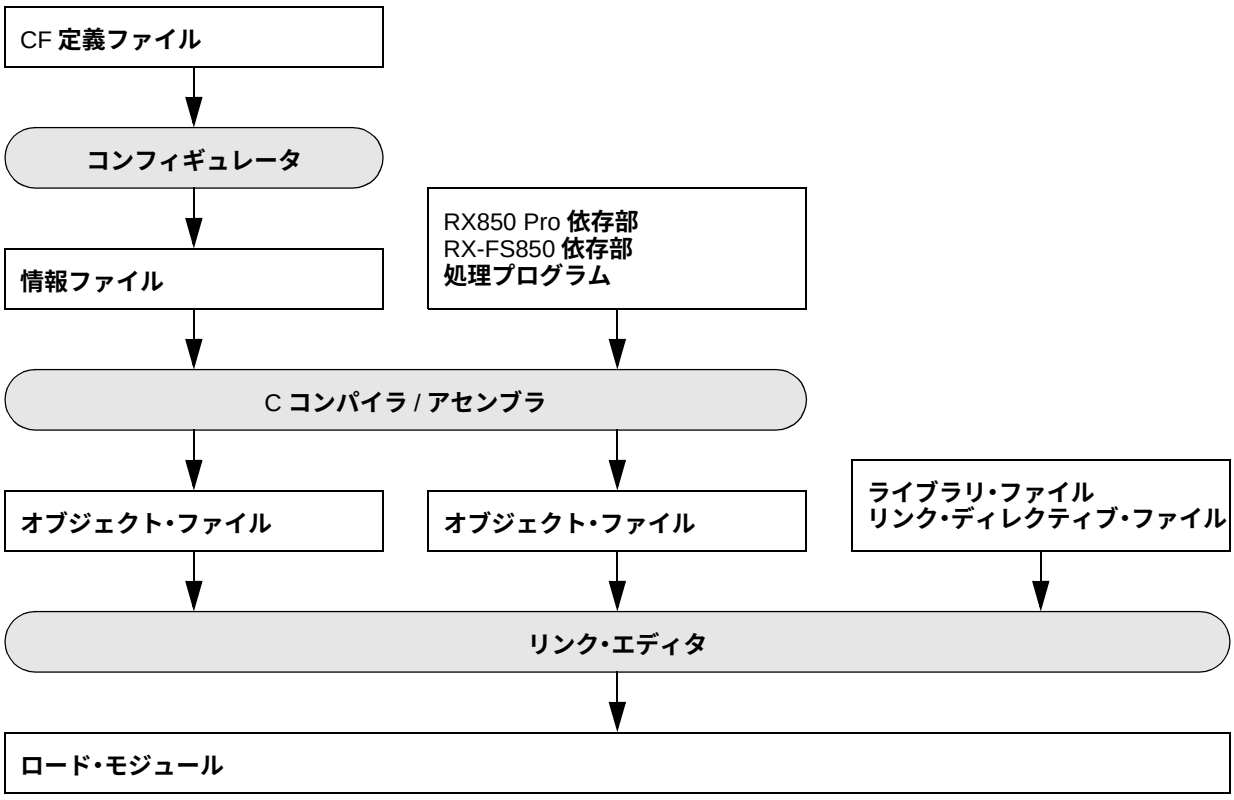
本章では，RX-FS850 を使用したアプリケーション・システム (ロード・モジュール) の構築手順を解説しています。

3.1　概要

システム構築とは，RX-FS850 の提供媒体からユーザの開発環境 (ホスト・マシン) 上にインストールしたファイル群を用いてアプリケーション・システム (ロード・モジュール) を生成することです。

図 3-1 に，RX-FS850 のシステム構築手順を示します。

図 3-1　システム構築手順



3.2　CF 定義ファイルの記述

組み込み型制御用リアルタイム・オペレーティング・システム RX850 Pro(μITRON3.0 仕様準拠，NEC エレクトロニクス製) の管理下で動作する処理プログラムを作成する場合，RX850 Pro に提供するコンフィギュレーション情報 (リアルタイム OS 情報，SIT 情報，SCT 情報) を保持した CF 定義ファイルが必要となります。

なお，RX-FS850 では 1 個のセマフォ，1 個のメールボックス，1 個のメモリ・プール，14 種類のシステム・コールを利用して各種機能を実現しています。

このため，CF 定義ファイルを記述する際には，RX-FS850 依存部，または，処理プログラムで使用する資源に関するコンフィギュレーション情報の他に，以下に示したコンフィギュレーション情報を RX-FS850 用として追加する必要があります。

- SIT 情報
  - システム情報
    RX-FS850 用に “ セマフォの自動割り付け ID 番号 ” として 1 個分を追加。
    RX-FS850 用に “ メールボックスの自動割り付け ID 番号 ” として 1 個分を追加。
    RX-FS850 用に “ メモリ・プールの自動割り付け ID 番号 ” として 1 個分を追加。
  - システム最大値情報
    RX-FS850 用に “ セマフォの最大生成数 ” として 1 個分を追加。
    RX-FS850 用に “ メールボックスの最大生成数 ” として 1 個分を追加。
    RX-FS850 用に “ メモリ・プールの最大生成数 ” として 1 個分を追加。
- SCT 情報
  - タスク管理機能情報
    RX-FS850 用に “ dis_dsp，ena_dsp ” を定義。
  - 同期通信 (セマフォ) 機能情報
    RX-FS850 用に “ cre_sem，del_sem，sig_sem，wai_sem ” を定義。
  - 同期通信 (メールボックス) 機能情報
    RX-FS850 用に “ cre_mbx，del_mbx，snd_msg，rcv_msg ” を定義。
  - メモリ・プール管理機能情報
    RX-FS850 用に “ cre_mpl，del_mpl，pget_blk，rel_blk ” を定義。

注意 1　CF 定義ファイルを記述する際の注意事項，および，コンフィギュレーション情報についての詳細は，「RX850 Pro ユーザーズ・マニュアル　インストレーション編」を参照してください。

注意 2　RX-FS850 では，CF 定義ファイルのサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\src
sys.cf　　：CF 定義ファイル

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\src
sys.cf　　：CF 定義ファイル

3.3　情報ファイルの生成

「3.2　CF 定義ファイルの記述」で作成された CF 定義ファイルに対して RX850 Pro が提供するユーティリティ・ツール (コンフィギュレータ cf850pro) を実行し，情報ファイル (システム情報テーブル，システム・コール・テーブル，システム情報ヘッダ・ファイル) を生成します。

以下に，シェル・プロンプトのコマンド・ラインから cf850pro を実行する際の入力例 (CF 定義ファイル sys.cf を読み込んだのち，システム情報テーブル sit.s，システム・コール・テーブル svc.s，システム情報ヘッダ・ファイル sys.h を出力) を示します。

ただし，入力例中の “ C> ” はシェル・プロンプトを，“ △ ” はスペース・キーの入力を，“ <Enter> ” はエンター・キーの入力を表しています。

```
C> cf850pro △ -i △ sit.s △ -c △ svc.s △ -d △ sys.h △ sys.cf △ <Enter>
```

注意　コンフィギュレータ cf850pro の起動オプション，および，実行方法についての詳細は，「RX850 Pro ユーザーズ・マニュアル　インストレーション編」を参照してください。

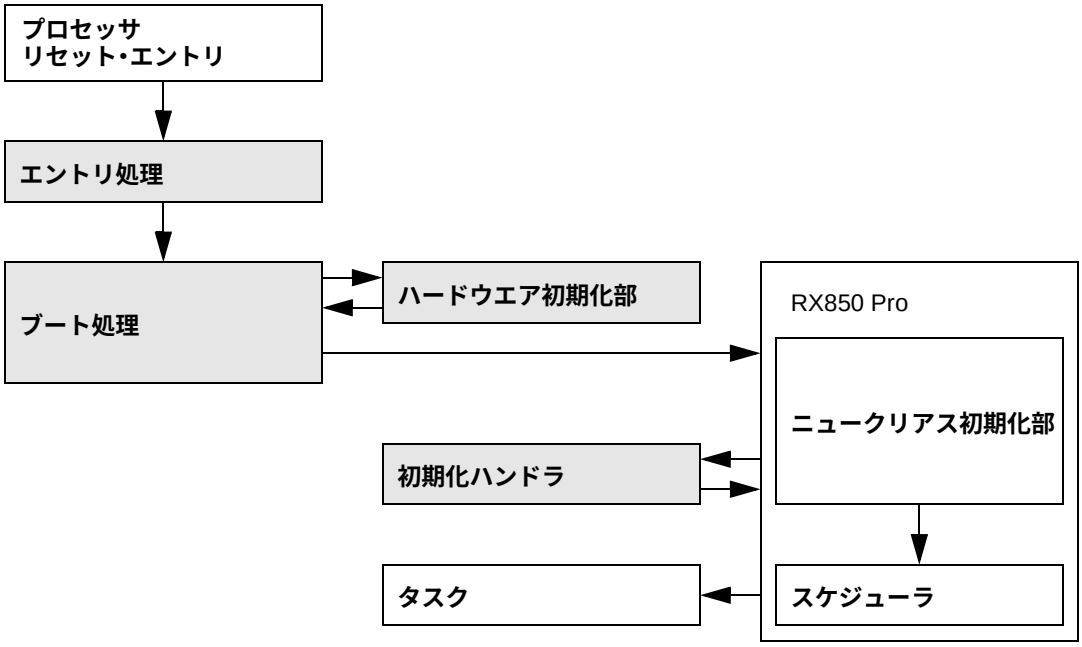
3.4　RX850 Pro 依存部の記述

RX-FS850 では，RX850 Pro が提供する機能を利用して各種機能を実現しています。また，ユーザが記述した処理プログラムは，RX850 Pro の管理下でその処理を実行することになります。

したがって，RX850 Pro を正常に動作させるうえで必要となる RX850 Pro 依存部 (ユーザ・オウン・コーディング部) の記述が必要となります。

図 3-2 に，RX850 Pro 依存部の処理の流れを示します。

図 3-2　RX850 Pro 依存部の処理の流れ



以下に，RX850 Pro 依存部の一覧を示します。

- エントリ処理
  割り込みが発生した際にプロセッサが強制的に制御を移すハンドラ･アドレスに対して該当処理(ブート処理,RX850 Pro が提供する割り込み処理管理機能，直接起動割り込みハンドラ) への分岐処理を割り付けるために用意された処理ルーチンです。

- ブート処理
  RX850 Pro が処理を実行するうえで必要となる最低限の初期化処理を行うために用意された処理ルーチンであり,エントリ処理 (プロセッサのリセット・エントリに割り付けられた分岐処理) から呼び出されます。

- ハードウエア初期化部
  RX850 Pro が処理を実行するうえで必要となるハードウエアの初期化処理を行うために用意された処理ルーチンであり，ブート処理から呼び出されます。
  なお，RX850 Pro では，一定周期で発生するタイマ割り込みを利用して時間管理を行っています。そこで，RX850 Pro が時間管理に利用するタイマ割り込みを発生するハードウエア (リアルタイム・パルス・ユニット，または，タイマ・コントローラ) に対しては，CF 定義ファイル作成時にシステム情報で定義した基本クロック周期でタイマ割り込みが発生するような設定を行う必要があります。

- 初期化ハンドラ
  ユーザの実行環境/アプリケーション･システムに依存した初期化処理を行うために用意された処理ルーチンであり，ニュークリアス初期化部から呼び出されます。

なお，RX850 Pro では，初期化ハンドラを “ タスク ” として位置付けています。

注意 1　RX850 Pro 依存部を記述する際の注意事項についての詳細は，「RX850 Pro ユーザーズ・マニュアル　インストレーション編」を参照してください。

注意 2　RX-FS850 では，初期化ハンドラのサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\src
varfunc.c　　：初期化ハンドラ

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\src
varfunc.c　　：初期化ハンドラ

3.5　RX-FS850 依存部の記述

RX-FS850 では，RX-FS850 が提供する機能を実現する際に必要となるユーザ・オウン関数，および，ドライバ関数を RX-FS850 依存部 (ユーザ・オウン・コーディング部) として切り出しています。
したがって，RX-FS850 が提供する機能を利用した処理プログラムを作成する場合，RX-FS850 を正常に動作させるうえで必要となる RX-FS850 依存部 (ユーザ・オウン・コーディング部) の記述が必要となります。
以下に，RX-FS850 依存部の一覧を示します。

- ユーザ・オウン関数

  RX-FS850 では，RX-FS850 が提供する機能を実現する際に必要となるユーザ・オウン関数をユーザ・オウン・コーディング部として切り出しています。
  表 3-1 に，ユーザ・オウン・コーディング部として切り出されるユーザ・オウン関数の一覧を示します。

表 3-1　ユーザ・オウン関数

ユーザ・オウン関数名	機能概要
fx_GetCurTime	現在時刻の獲得関数

- ドライバ関数

  RX-FS850 では，RX-FS850 が提供する機能を実現する際に必要となるドライバ関数をユーザ・オウン・コーディング部として切り出しています。
  表 3-2 に，ユーザ・オウン・コーディング部として切り出されるドライバ関数の一覧を示します。

表 3-2　ドライバ関数

ドライバ関数名	機能概要
<driver_name>Init	デバイス・ドライバの初期化関数
<driver_name>Term	デバイス・ドライバの終了関数
<driver_name>Task	デバイス・ドライバ用常駐タスク
<driver_name>Interrupt	デバイス・ドライバ用間接起動割り込みハンドラ
<driver_name>GetInfo	デバイス情報の獲得関数
<driver_name>ClearInfo	デバイス情報のクリア関数
<PnP_name>Init	プラグ・アンド・プレイ機能の初期化関数
<PnP_name>Term	プラグ・アンド・プレイ機能の終了関数
<PnP_name>Task	プラグ・アンド・プレイ機能用常駐タスク
<PnP_name>Interrupt	プラグ・アンド・プレイ機能用間接起動割り込みハンドラ
<PnP_name>Cychdr	プラグ・アンド・プレイ機能用周期起動ハンドラ

注意 1　RX-FS850 依存部についての詳細は，「第 6 章　RX-FS850 依存部」を参照してください。

注意 2　RX-FS850 では，RX-FS850 依存部のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\user\src
gettime.c　　：ユーザ・オウン関数

nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src
<*driver_name*>.c　　：ドライバ関数

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\user\src
gettime.c　　：ユーザ・オウン関数

nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src
<*driver_name*>.c　　：ドライバ関数

3.6　処理プログラムの記述

ブロック型デバイスへの入出力処理として実現すべき処理 (処理プログラム) を記述します。
なお，処理プログラムは，用途別に以下のように分類 / 区別されています。

- タスク
  RX850 Pro の管理下で実行可能な処理プログラムの最小単位です。

- 直接起動割り込みハンドラ
  割り込みが発生した際，RX850 Pro を介在させることなく起動される割り込み処理専用ルーチンです。
  なお，RX850 Pro では，直接起動割り込みハンドラを “ タスク ” とは独立したもの (非タスク) として位置付けています。このため，割り込みが発生した際には，システム内で最高優先度を持つタスクが実行中であっても，その処理は中断され，直接起動割り込みハンドラに制御が移ります。

- 間接起動割り込みハンドラ
  割り込みが発生した際に RX850 Pro による割り込み前処理 (レジスタの退避，スタックの切り替えなど) を行わせたのちに起動される割り込み処理専用ルーチンです。
  なお，RX850 Pro では，間接起動割り込みハンドラを “ タスク ” とは独立したもの (非タスク) として位置付けています。このため，割り込みが発生した際には，システム内で最高優先度を持つタスクが実行中であっても，その処理は中断され，間接起動割り込みハンドラに制御が移ります。

- 周期起動ハンドラ
  一定の時間が経過した際に起動される周期処理専用ルーチンです。
  なお，RX850 Pro では，周期起動ハンドラを “ タスク ” とは独立したもの (非タスク) として位置付けています。このため，一定の時間が経過した際には，システム内で最高優先度を持つタスクが実行中であっても，その処理は中断され，周期起動ハンドラに制御が移ります。

- 拡張 SVC ハンドラ
  ユーザが記述した関数を拡張システム・コールとして RX850 Pro に登録した処理ルーチンです。
  なお，RX850 Pro では，拡張 SVC ハンドラを “ 拡張 SVC ハンドラを呼び出した処理プログラム (タスク，非タスク) の延長線 ” として位置付けています。

- 拡張 SVC ハンドラ用インタフェース・ルーチン
  処理プログラム (タスク，非タスク) から 4 個以上の引き継ぎデータを持った拡張 SVC ハンドラを呼び出す際に必要となるインタフェース・ルーチンです。

注意 1　処理プログラムを記述する際の注意事項についての詳細は，「RX850 Pro ユーザーズ・マニュアル　基礎編」を参照してください。

注意 2　RX-FS850 では，処理プログラムのサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\src
task.c　　：タスク

【 GHS 製コンパイラ対応版の場合 】

nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\src
task.c　　：タスク

3.7　リンク・ディレクティブ・ファイルの記述

リンク・エディタが行うアドレス割り付けをユーザが固定化するためのファイル (リンク・ディレクティブ・ファイル) を記述します。

注意 1　リンク・ディレクティブ・ファイルを記述する際の注意事項についての詳細は，使用する C コンパイラ・パッケージのユーザーズ・マニュアルを参照してください。

注意 1　RX-FS850 では，機能単位にモジュール化されたオブジェクトの割り付け先 (セクション名) を規定しています。したがって，規定されたセクション名をリンク・ディレクティブ・ファイルに定義することは必須となります。以下に，RX-FS850 が規定しているセクション名一覧を示します。

【 CA850 対応版の場合 】

セクション名	属性	タイプ	意味
.fs.bss	AW	NOBITS	RX-FS850 の管理オブジェクト
.fs.data	AW	PROGBITS	RX-FS850 のデータ情報
.fs.const	A	PROGBITS	RX-FS850 のデータ情報
.fs.text	RX	PROGBITS	RX-FS850 の本体処理部

【 GHS 製コンパイラ対応版の場合 】

セクション名	意味
.fs.bss	RX-FS850 の管理オブジェクト
.fs.data	RX-FS850 のデータ情報
.fs.const	RX-FS850 のデータ情報
.fs.text	RX-FS850 の本体処理部

注意 2　RX-FS850 では，リンク・ディレクティブ・ファイルのサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\src
sample.dir　　：リンク・ディレクティブ・ファイル

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\src
sample.lx　　：リンク・ディレクティブ・ファイル

3.8　ロード・モジュールの生成

「3.2　CF 定義ファイルの記述」～「3.7　リンク・ディレクティブ・ファイルの記述」で作成した各種ファイルの他に，C コンパイラ・パッケージ，RX850 Pro，RX-FS850 などが提供しているライブラリ・ファイルを用いて，ロード・モジュールを生成します。

注意 1　ロード・モジュールの生成方法についての詳細は，使用する C コンパイラ・パッケージのユーザーズ・マニュアルを参照してください。

注意 2　RX-FS850 が提供しているライブラリ・ファイルについては，処理プログラムにおいて利用する機能の種類により，リンクすべきものが異なります。
なお，API 関数処理部ライブラリ librxfs.a のリンク順序は，“ RX-FS850 が提供しているライブラリ・ファイル中の先頭 ”，共通処理部ライブラリ libfscmn.a のリンク順序は，“ RX-FS850 が提供しているライブラリ・ファイル中の最後尾 ” に行う必要があります。

RX-FS850 の提供機能	ライブラリ・ファイル名
FAT ファイル・システム機能	librxfs.a + libfsfat.a + libfsnocd.a + libfscmn.a
CD-ROM ファイル・システム機能	librxfs.a + libfsnofat.a + libfscd.a + libfscmn.a
FAT ファイル・システム機能 + CD-ROM ファイル・システム機能	librxfs.a + libfsfat.a + libfscd.a + libfscmn.a

注意 3　RX-FS850 **では，ロード・モジュールを生成するためのサンプル・コマンド・ファイルを提供しています。**

【 CA850 **対応版** Windows **ベースの場合 】**
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\obj
sample.prw　　**：ロード・モジュール用ワーク・スペース・ファイル**
sample.prj　　**：ロード・モジュール用プロジェクト・ファイル**

【 CA850 **対応版** UNIX **ベースの場合 】**
nectools32\smp850e\rxfs\<*target_name*>\<*sample_name*>\obj
Makefile　　**：ロード・モジュール用メイク・ファイル**

【 GHS **製コンパイラ対応版の場合 】**
nectools32\smp850e_ghs\rxfs\<*target_name*>\<*sample_name*>\obj_ghs
sample.bld　　**：ロード・モジュール用ビルド・ファイル**

第 4 章　入出力管理機能

本章では，RX-FS850 が提供している入出力管理機能について解説しています。

4.1　概要

RX-FS850 では，ブロック型デバイス (ATA フラッシュ・カード，IDE ハード・ディスク，RAM ディスクなど) への高速な入出力処理の他に，エラー・コードの獲得処理などを “ 入出力管理機能 ” として提供しています。

4.2　処理の流れ

RX-FS850 では,ブロック型デバイスへの高速な入出力処理を実現する際に必要となる各種 API 関数の発行順序を規定しています。

図 4-1 に，RX-FS850 が提供している API 関数を利用した“ ブロック型デバイスのファイルに対するデータの書き込み / 読み込み ” の実現例を示します。

図 4-1　入出力処理の実現例

4.3　メモリ管理機能

4.3.1　RX-FS850 用ヒープ領域の確保 / 解放

RX-FS850 用ヒープ領域の確保 / 解放は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_HeapInit

  パラメータ *heapsz* で指定されたサイズのメモリ領域を RX-FS850 用ヒープ領域として確保します。
  これにより，RX-FS850 用ヒープ領域は，RX-FS850 の管理対象となります。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                unsigned long     heapsz = 0x100000 ;           /* 変数の宣言，初期化 */

                rxfs_HeapInit ( heapsz ) ;      /* RX-FS850 用ヒープ領域の確保 */

                ..........................
                ..........................
                ..........................

                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　RX-FS850 用ヒープ領域のサイズ *heapsz* については，以下に示した計算式から求めることができます。

heapsz = FATdriveNum × { (SctSz + 32) × CtrBuffNum + (ClsSz + 32) × DataBuffNum + 80 }
+ CDdriveNum × { (SctSz + 32) × (CtrBuffNum + DataBuffNum) + 80 }
+ align16 (TaskNum × 20)
+ align16 (FmngNum × 340)
+ align16 (FstmNum × 28)
+ align16 (DrvNum × 16)
+ ChdirTask × 270
+ MaxOpenDir × 272
+ 3K

FATdriveNum　：FAT ファイル・システム用ドライブの総数
rxfs_mount の発行によりマウントする FAT ファイル・システム用ドライブの総数と同値。

CDdriveNum　：CD-ROM ファイル・システム用ドライブの総数
rxfs_mount の発行によりマウントする CD-ROM ファイル・システム用ドライブの総数と同値。

SctSz　：セクタ・サイズ (単位：バイト)

CtrBuffNum　：コントロール用バッファの総数
rxfs_SystemInit の発行時に指定するドライバ管理構造体 _FX_DRV_MNG のメンバ CtrBuffNum と同値。

ClsSz　：クラスタ・サイズ (単位：バイト)

DataBuffNum　：データ用バッファの総数
rxfs_SystemInit の発行時に指定するドライバ管理構造体 _FX_DRV_MNG のメンバ DataBuffNum と同値。

TaskNum　：RX-FS850 が提供している API 関数を利用するタスクの総数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ TaskNum と同値。

FmngNum　：ファイル管理構造体の総数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同値。

FstmNum　：ストリームの最大生成数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ FstmNum と同値。

DrvNum　：ドライバの総数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ DrvNum と同値。

ChdirTask　：rxfs_chdir を発行するタスクの総数

MaxOpenDir　：rxfs_opendir の発行により同時にオープンするディレクトリの最大数

注意 2　RX-FS850 用ヒープ領域の確保は，RX850 Pro が管理しているシステム・メモリ UPOL0 から行われます。ただし，RX-FS850 では，RX-FS850 用ヒープ領域を確保する際，該当メモリ領域のクリア処理を行っていないため，確保されたメモリ領域の内容は不定となります。
なお，システム・メモリ UPOL0 についての詳細は，「**RX850 Pro ユーザーズ・マニュアル インストレーション編**」を参照してください。

- rxfs_HeapTerm

  API 関数 rxfs_HeapInit の発行により確保した RX-FS850 用ヒープ領域を解放します。
  これにより，RX-FS850 用ヒープ領域は，RX-FS850 の管理対象から除外されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                unsigned long     heapsz = 0x100000 ;           /* 変数の宣言，初期化 */

                rxfs_HeapInit ( heapsz ) ;      /* RX-FS850 用ヒープ領域の確保 */

                ..........................
                ..........................
                ..........................

                rxfs_HeapTerm ( ) ;             /* RX-FS850 用ヒープ領域の解放 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　本 API 関数の発行は，API 関数 rxfs_SystemTerm の処理完了後に行う必要があります。

注意 2　RX-FS850用ヒープ領域の解放は,RX850 Proが管理しているシステム・メモリUPOL0に対して行われます。
ただし，RX-FS850 では，RX-FS850 用ヒープ領域を解放する際，該当メモリ領域のクリア処理を行っていないため，解放されたメモリ領域の内容は不定となります。
なお，システム・メモリ UPOL0 についての詳細は，「RX850 Pro ユーザーズ・マニュアル インストレーション編」を参照してください。

4.4　システム管理機能

4.4.1　RX-FS850 の初期化 / 終了

RX-FS850 の初期化 / 終了は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_SystemInit

  パラメータ *pInitPack* で指定された情報をもとに，RX-FS850 が提供する機能を実現するうえで必要となる各種初期化処理を実行します。
  なお，RX-FS850 では，RX-FS850 の初期化処理として，RX-FS850 が提供している機能を実現する際に必要となる各種資源の生成，および，メモリ領域の確保を行っています。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                unsigned long   heapsz = 0x100000 ;         /* 変数の宣言，初期化 */
                FX_DRV_MNG      pDrvMng [ 0x1 ] ;           /* データ構造体の宣言 */
                FX_INIT_PACK    pInitPack ;                 /* データ構造体の宣言 */

                rxfs_HeapInit ( heapsz ) ;      /* RX-FS850 用ヒープ領域の確保 */

                                                /* データ構造体の初期化 */
                strcpy ( pDrvMng [ 0x0 ].DriverName , "RAMDISK" ) ;

                                                /* デバイス・ドライバの初期化 */
                <driver_name>Init ( &( pDrvMng [ 0x0 ] ) , ... ) ;

                pInitPack.DrvMng = pDrvMng ;    /* データ構造体の初期化 */
                pInitPack.DrvNum = 1 ;          /* データ構造体の初期化 */
                pInitPack.RFU1 = 0x0 ;          /* データ構造体の初期化 */
                pInitPack.TaskNum = 5 ;         /* データ構造体の初期化 */
                pInitPack.FmngNum = 15 ;        /* データ構造体の初期化 */
                pInitPack.FstmNum = 10 ;        /* データ構造体の初期化 */
                pInitPack.TmpFilNum = 100 ;     /* データ構造体の初期化 */

                rxfs_SystemInit ( &pInitPack ) ;    /* RX-FS850 の初期化 */

                ..........
                ..........
                ..........

                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　本 API 関数の発行は，API 関数 rxfs_HeapInit の処理完了後に行う必要があります。

注意 2　システム初期化情報 _FX_INIT_PACK についての詳細は「5.4.1　システム初期化情報」を，ドライバ管理構造体 _FX_DRV_MNG についての詳細は「5.4.2　ドライバ管理構造体」を参照してください。

- rxfs_SystemTerm

  RX-FS850 の終了処理を実行します。
  なお，RX-FS850 では，RX-FS850 の終了処理として，RX-FS850 が提供している機能を実現する際に必要となる各種資源の削除，および，メモリ領域の解放を行っています。
  これにより，すべての処理プログラムから RX-FS850 が提供している機能を利用することができなくなります。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                unsigned long   heapsz = 0x100000 ;             /* 変数の宣言，初期化 */
                FX_DRV_MNG      pDrvMng [ 0x1 ] ;               /* データ構造体の宣言 */
                FX_INIT_PACK    pInitPack ;                     /* データ構造体の宣言 */

                rxfs_HeapInit ( heapsz ) ;          /* RX-FS850 用ヒープ領域の確保 */

                                                    /* データ構造体の初期化 */
                strcpy ( pDrvMng [ 0x0 ].DriverName , "RAMDISK" ) ;

                                                    /* デバイス・ドライバの初期化 */
                <driver_name>Init ( &( pDrvMng [ 0x0 ] ) , ... ) ;

                pInitPack.DrvMng = pDrvMng ;        /* データ構造体の初期化 */
                pInitPack.DrvNum = 1 ;              /* データ構造体の初期化 */
                pInitPack.RFU1 = 0x0 ;              /* データ構造体の初期化 */
                pInitPack.TaskNum = 5 ;             /* データ構造体の初期化 */
                pInitPack.FmngNum = 15 ;            /* データ構造体の初期化 */
                pInitPack.FstmNum = 10 ;            /* データ構造体の初期化 */
                pInitPack.TmpFilNum = 100 ;         /* データ構造体の初期化 */

                rxfs_SystemInit ( &pInitPack ) ;    /* RX-FS850 の初期化 */

                ..........
                ..........
                ..........

                rxfs_SystemTerm ( ) ;               /* RX-FS850 の終了 */
                <driver_name>Term ( ... ) ;         /* デバイス・ドライバの終了 */
                rxfs_HeapTerm ( ) ;                 /* RX-FS850 用ヒープ領域の解放 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

4.4.2　タスクの登録 / 登録解除

タスクの登録 / 登録解除は,以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_TaskInit

  本 API 関数を発行したタスクを “ RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を利用するタスク ” として RX-FS850 に登録します。
  なお，RX-FS850 では，タスクの登録処理として，RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を利用するタスク毎に必要となる管理領域の獲得，および，初期化を行っています。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                    /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                rxfs_TaskInit ( ) ;         /* タスクの登録 */

                ..........
                ..........
                ..........

                ext_tsk ( ) ;               /* タスクの終了処理 */
}
```

注意　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ TaskNum と同数のタスクが既に登録されていた場合には，タスクの登録処理は行わず，戻り値として FXSE_TASKMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

- rxfs_TaskTerm

  本 API **関数を発行したタスクを “** RX-FS850 **が提供している** API **関数 (**rxfs_mount**，**rxfs_fopen **など**) **を利用しないタスク ” として** RX-FS850 **から登録解除します。**
  なお，RX-FS850 **では，タスクの登録解除処理として，**API **関数** rxfs_TaskInit **の発行により獲得した管理領域の返却を行っています。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>            /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                rxfs_TaskInit ( ) ;     /* タスクの登録 */

                .........
                .........
                .........

                rxfs_TaskTerm ( ) ;     /* タスクの登録解除 */
                ext_tsk ( ) ;           /* タスクの終了処理 */
}
```

4.5　ドライブ制御管理機能

4.5.1　ドライブのマウント / アンマウント

ドライブのマウント / アンマウントは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_mount

  パラメータ *drvname* で指定されたドライブをパラメータ *driver*，*u_id*，*p_id*，*mode* で指定された情報をもとにマウントします。
  これにより，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が可能となります。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                fx_u8           drvname = 'A' ;             /* 変数の宣言，初期化 */
                char            driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8           u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           mode = FX_MNT_ASYNC ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ドライブのマウント */
                rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　デバイス・ドライバ名 *driver* には，API 関数 rxfs_SystemInit を発行した際に指定したドライバ管理構造体 _FX_DRV_MNG のメンバ DriverName を設定します。
なお，ドライバ管理構造体 _FX_DRV_MNG についての詳細は，「5.4.2　ドライバ管理構造体」を参照してください。

注意 2　RX-FS850 では，マウント・モード *mode* として，以下に示した 3 種類を提供しています。

- FX_MNT_ASYNC (0x0) ：非同期モード

  書き込み要求 (API 関数 rxfs_fwrite，rxfs_write などの発行) が発生した際，ただちに該当ファイルへの書き込みを行わず，いったん，バッファに対して書き込みを行います。
  なお，該当ファイルに対する書き込み処理は，バッファがフラッシュ (API 関数 rxfs_flushall，rxfs_fflush，rxfs_fsync の発行) された際，または，該当ファイルがクローズ (API 関数 rxfs_fclose，rxfs_close の発行) された際に行われます。

- FX_MNT_SYNC (0x80) ：同期モード

  書き込み要求 (API 関数 rxfs_fwrite，rxfs_write などの発行) が発生した際，ただちに該当ファイルへの書き込みを行います。

- FX_MNT_FORMAT (0x40) **：フォーマット・モード**

  パラメータ *drvname* **で指定された** FAT **ファイル・システムのドライブをクイック・フォーマットする際の専用モードです。したがって，本モードでマウントされたドライブに対する操作は，フォーマット処理** (API **関数** rxfs_format **の発行**)**，または，アンマウント処理** (API **関数** rxfs_umount**，**rxfs_umountforce **の発行**) **に限定されます。**

- rxfs_umount

  パラメータ *drvname* **で指定されたドライブをアンマウントします。**
  したがって，本 API **関数の発行後，該当ドライブに対する操作** (API **関数** rxfs_fopen**，**rxfs_open **などの発行**) **が禁止されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                fx_u8           drvname = 'A' ;             /* 変数の宣言，初期化 */
                char            driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8           u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           mode = FX_MNT_ASYNC ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                                                    /* ドライブのマウント */
                rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;

                .....
                .....
                .....

                rxfs_umount ( drvname ) ;           /* ドライブのアンマウント */
                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意　ドライブ名 *drvname* **には，**API **関数** rxfs_mount **を発行した際に指定したドライブ名を設定します。**

- rxfs_umountforce

  パラメータ *drvname* **で指定されたドライブを強制的にアンマウントします。**
  したがって，本 API **関数の発行後，該当ドライブに対する操作** (API **関数** rxfs_fopen**，**rxfs_open **などの発行**) **が禁止されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                fx_u8           drvname = 'A' ;             /* 変数の宣言，初期化 */
                char            driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8           u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           mode = FX_MNT_ASYNC ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                                                    /* ドライブのマウント */
                rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;

                .....
                .....
                .....

                rxfs_umountforce ( drvname ) ;      /* ドライブの強制アンマウント */
                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意 1　**ドライブ名** *drvname* **には，**API **関数** rxfs_mount **を発行した際に指定したドライブ名を設定します。**

注意 2　**本** API **関数では，パラメータ** *drvname* **で指定されたドライブに対する操作が行われている最中であっても，該当ドライブのアンマウント処理を強制的に実行します。したがって，ファイルが " 書き込み可能なモード " でオープンされていた際には，該当ファイルに対する書き込み要求が反映されない可能性があります。**

注意 3　RX-FS850 **では，該当ドライブに一時ディレクトリが設定されていた場合，一時ディレクトリの設定解除処理を実行します。**

4.5.2　マウント情報の獲得

マウント情報の獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_mountinfo

  パラメータ *drvname* で指定されたドライブのマウント情報 (デバイス・ドライバ名，ユニット ID，パーティション番号，マウント・モード) をパラメータ *driver*，*u_id*，*p_id*，*mode* で指定された領域に格納します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include      <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include      <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
              fx_u8       drvname = 'A' ;                 /* 変数の宣言，初期化 */
              char        driver [ 0x9 ] ;                /* 変数の宣言 */
              fx_u8       u_id ;                          /* 変数の宣言 */
              fx_u8       p_id ;                          /* 変数の宣言 */
              fx_u8       mode ;                          /* 変数の宣言 */

              rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                              /* マウント情報の獲得 */
              rxfs_mountinfo ( drvname , driver , &u_id , &p_id , &mode ) ;

              ..........................
              ..........................
              ..........................

              rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
              ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。
注意 2　*driver* で指定されたデバイス・ドライバ名を格納する領域には，9 バイトの大きさが必要となります。

4.5.3　ドライブの強制フラッシュ

ドライブの強制フラッシュは,以下に示したAPI関数を処理プログラム(タスク)から発行することにより実現されます。

- rxfs_flushall

  パラメータ *drvname* で指定されたドライブにおいて，書き込み要求 (API 関数 rxfs_fwrite，rxfs_fputc などの発行) が行われているファイルのバッファを強制的にフラッシュします。
  これにより，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。
  なお,本 API 関数では,フラッシュ処理が完了した際には,該当ドライブをアクセス禁止状態へと遷移させています。
  したがって，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が禁止されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                fx_u8           drvname = 'A' ;             /* 変数の宣言，初期化 */
                char            driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8           u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           mode = FX_MNT_ASYNC ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                                                    /* ドライブのマウント */
                rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;

                ..........
                ..........
                ..........

                rxfs_flushall ( drvname ) ;         /* ドライブの強制フラッシュ */
                rxfs_umountforce ( drvname ) ;      /* ドライブの強制アンマウント */
                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意 1　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。

注意 2　該当ドライブが CD-ROM ファイル・システムの際には，フラッシュ処理は実行されず，状態遷移処理 (該当ドライブをアクセス禁止状態へと遷移) のみが実行されます。

注意 3　本 API 関数では，API 関数 rxfs_tmpfile の発行により生成 / オープンされた一時ファイルのクローズ処理，および，削除処理は行われません。
したがって，該当ドライブの一時ファイルに対するクローズ処理，および，削除処理 (API 関数 rxfs_fclose の発行) は，本 API 関数を発行する以前に行う必要があります。

4.5.4　ドライブのクイック・フォーマット

ドライブのクイック・フォーマットは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_format

  パラメータ *drvname* で指定されたドライブをクイック・フォーマット (フォーマット済みドライブの再フォーマット) します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                fx_u8           drvname = 'A' ;             /* 変数の宣言，初期化 */
                char            driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8           u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8           mode = FX_MNT_FORMAT ;      /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                                                    /* ドライブのマウント */
                rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;
                rxfs_format ( drvname ) ;           /* ドライブのクイック・フォーマット */
                rxfs_umount ( drvname ) ;           /* ドライブのアンマウント */
                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意 1　デバイス・ドライバ名 *driver* には，API 関数 rxfs_SystemInit を発行した際に指定したドライバ管理構造体 _FX_DRV_MNG のメンバ DriverName を設定します。
なお，ドライバ管理構造体 _FX_DRV_MNG についての詳細は，「5.4.2　ドライバ管理構造体」を参照してください。

注意 2　本 API 関数でフォーマット可能なドライブは，フォーマット・モード FX_MNT_FORMAT でマウントされたドライブに限定されます。

4.6　ストリーム入出力管理機能

4.6.1　ファイルのオープン / クローズ

ファイルのオープン / クローズは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fopen

  パラメータ *filename* で指定されたファイルをパラメータ *mode* で指定されたストリーム・モードでオープンします。
  なお，RX-FS850 では，ファイルのオープン処理として，API 関数 (rxfs_fclose，rxfs_fflush など) を発行する際，オープンしたファイル毎に必要となるストリームの獲得，および，初期化を行っています。
  また，本 API 関数の処理が正常終了した際には，“ オープンしたファイルのストリームへのポインタ ” が戻り値として返されます。
  以下に，パラメータ *mode* に指定可能なキー・ワード一覧を示します。

  r　：読み込み専用
  ファイルの先頭からデータの読み込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在しない場合，異常終了 FXE_FILENOTEXIST となります。

  w　：書き込み専用
  ファイルの先頭からデータの書き込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在した場合には “ 該当ファイルの削除処理，および，該当ファイルの新規生成処理 ” が，パラメータ *filename* で指定されたファイルが存在しない場合には “ 該当ファイルの新規生成処理 ” があわせて行われます。

  a　：書き込み専用
  ファイルの末尾からデータの書き込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在しない場合には“ 該当ファイルの新規生成処理 ”があわせて行われます。

  r+　：読み込み / 書き込み両用
  ファイルの先頭からデータの読み込み / 書き込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在しない場合，異常終了 FXE_FILENOTEXIST となります。

  w+　：読み込み / 書き込み両用
  ファイルの先頭からデータの読み込み / 書き込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在した場合には “ 該当ファイルの削除処理，および，該当ファイルの新規生成処理 ” が，パラメータ *filename* で指定されたファイルが存在しない場合には “ 該当ファイルの新規生成処理 ” があわせて行われます。

  a+　：読み込み / 書き込み両用
  ファイルの先頭からデータの読み込みを，ファイルの末尾からデータの書き込みを行います。
  なお，パラメータ *filename* で指定されたファイルが存在しない場合には“ 該当ファイルの新規生成処理 ”があわせて行われます。

以下に，本 API 関数の記述例を示します。
なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include      <stdrx85p.h>                  /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include      <rxfs.h>                      /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
              FX_FILE       *stream ;                    /* データ構造体の宣言 */
              char          filename [ ] = "C:\\tmp\\smp.txt" ;   /* 変数の宣言，初期化 */
              char          mode [ ] = "w+" ;            /* 変数の宣言，初期化 */

              rxfs_TaskInit ( ) ;                        /* タスクの登録 */
```

```
                                        /* ファイルのオープン */
        stream = rxfs_fopen ( filename , mode ) ;

        ..........
        ..........
        ..........

        rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
        ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。**
なお，RX-FS850 **では，パラメータ *filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (**270 **文字) を超えるような相対パスが設定されていた場合には，**FXE_PATHTOOLONG **を返しています。**

注意 2　RX-FS850 **では，パラメータ *filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。**

注意 3　**本** API **関数では，**API **関数** rxfs_SystemInit **を発行した際に指定したシステム初期化情報** _FX_INIT_PACK **のメンバ** FmngNum **と同数のファイル管理構造体，または，**FstmNum **と同数のストリームが既に使用されていた場合には，ファイルのオープン処理は行わず，エラー・コードとして** FXE_FILEMAX**，または，**FXE_OPENMAX **を返しています。**
なお，システム初期化情報 _FX_INIT_PACK **についての詳細は，「**5.4.1　**システム初期化情報」を参照してください。**

- rxfs_tmpfile

  一時ファイルを生成したのち，読み込み / 書き込み両用でオープンします。
  なお，本 API 関数の処理が正常終了した際には，“ オープンした一時ファイルのストリームへのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                FX_FILE         *stream ;                       /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_settmpdir ( dirname ) ;    /* 一時ディレクトリの設定 */
                stream = rxfs_tmpfile ( ) ;     /* 一時ファイルの生成 / オープン */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　RX-FS850 **では，**API **関数** rxfs_settmpdir **の発行により設定された一時ディレクトリに対して，一時ファイルの生成処理を行います。**

注意 2　RX-FS850 **では，本** API **関数の発行により生成 / オープンされた一時ファイルに対して** API **関数** rxfs_fclose **が発行された際には，該当ファイルのクローズ処理を実行した後，該当ファイルの削除処理もあわせて行っています。**

- rxfs_fclose

  パラメータ *stream* **で指定されたファイルをクローズします。**
  なお，RX-FS850 **では，ファイルのクローズ処理として，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の発行により獲得したストリームの返却を行っています。**
  したがって，本 API **関数の発行後，該当ファイルに対する操作 (**API **関数** rxfs_fflush**，**rxfs_fread **などの発行) が禁止されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;

                .........................
                .........................
                .........................

                rxfs_fclose ( stream ) ;        /* ファイルのクローズ */
                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　**パラメータ** *stream* **で指定されたファイルが “ 書き込み可能なモード ” でオープンされていた場合，**RX-FS850 **は該当バッファをフラッシュしたのち，ファイルのクローズ処理を実行します。**
ただし，該当ファイルのドライブが API **関数** rxfs_umountforce **の発行によりアンマウントされていた際，および，**API **関数** rxfs_flushall **の発行によりアクセス禁止状態へと遷移していた際には，フラッシュ処理は実行されず，クローズ処理のみが実行されます。**

注意 3　RX-FS850 **では，該当ファイルが** API **関数** rxfs_tmpfile **の発行によりオープンされていた場合，該当ファイルのクローズ処理を実行した後，該当ファイルの削除処理もあわせて行っています。**

4.6.2　バッファのフラッシュ

バッファのフラッシュは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fflush

  パラメータ *stream* で指定されたファイルのバッファをフラッシュします。これにより，書き込み要求 (API 関数 rxfs_fwrite，rxfs_fputc などの発行) を行った際，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                unsigned int    cnt ;                           /* 変数の宣言 */
                unsigned char   ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = sizeof ( char ) ;        /* 変数の宣言，初期化 */
                long            number = 0x100 ;                /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン *
                stream = rxfs_fopen ( filename , mode ) ;

                for ( cnt = 0x0 ; cnt < 0x100 ; cnt++ ) {
                                                                /* 変数の初期化 */
                        ptr [ cnt ] = ( unsigned char ) cnt & 0xff ;
                }

                                                /* データの書き込み */
                rxfs_fwrite ( ptr , size , number , stream ) ;
                rxfs_fflush ( stream ) ;        /* バッファのフラッシュ */

                .........................
                .........................
                .........................

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

4.6.3　データの読み込み / 書き込み

データの読み込み / 書き込みは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fread

  パラメータ *stream* で指定されたファイルのファイル・ポインタの位置からパラメータ *size* * *number* で算出されたサイズのデータ，または，ファイルの末尾までのデータをパラメータ *ptr* で指定された領域に格納します。
  なお，本 API 関数の処理が正常終了した際には，“ データの読み込み個数 ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;  /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                char            ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 ;                    /* 変数の宣言，初期化 */
                long            number = 0x10 ;                 /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                /* データの読み込み */
                rxfs_fread ( ptr , size , number , stream ) ;

                .........................
                .........................
                .........................

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの読み込み処理が完了した際には，ファイル・ポインタの位置を本 API 関数発行以前の位置から “ 読み込んだデータの合計サイズ (単位：バイト) ” だけ進めています。

注意 3　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

- rxfs_fwrite

  パラメータ *stream* **で指定されたファイルのファイル・ポインタの位置からパラメータ** *size * number* **で算出されたサイズのデータを書き込みます。**
  なお，パラメータ *ptr* **で指定された領域には，該当ファイルに書き込むデータを設定します。**
  また，本 API **関数の処理が正常終了した際には，“ データの書き込み個数 ” が戻り値として返されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                unsigned int    cnt ;                           /* 変数の宣言 */
                unsigned char   ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 ;                    /* 変数の宣言，初期化 */
                long            number = 0x10 ;                 /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;

                for ( cnt = 0x0 ; cnt < 0x100 ; cnt++ ) {
                                                                /* 変数の初期化 */
                        ptr [ cnt ] = ( unsigned char ) cnt & 0xff ;
                }

                                                        /* データの書き込み */
                rxfs_fwrite ( ptr , size , number , stream ) ;

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　RX-FS850 **では，データの書き込み処理が完了した際には，ファイル・ポインタの位置を，本** API **関数発行以前の位置から “ 書き込んだデータの合計サイズ (単位：バイト) ” だけ進めています。**

4.6.4　文字の読み込み / 書き込み

文字の読み込み / 書き込みは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fgetc

  パラメータ *stream* で指定されたファイルのファイル・ポインタの位置から文字 (1 文字) を読み込みます。
  なお，本 API 関数の処理が正常終了した際には，“ 読み込んだ文字 (1 文字) を int 型に拡張した値 ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include      <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include      <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
              FX_FILE       *stream ;                       /* データ構造体の宣言 */
              char          filename [ ] = "C:\\tmp\\smp.txt" ;  /* 変数の宣言，初期化 */
              char          mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
              int           character ;                     /* 変数の宣言 */

              rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                              /* ファイルのオープン */
              stream = rxfs_fopen ( filename , mode ) ;
                                              /* 文字の読み込み */
              character = rxfs_fgetc ( stream ) ;

              .....
              .....
              .....

              rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
              ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだ文字のサイズ (0x1 バイト) ” だけ進めています。

注意 3　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

- rxfs_ungetc

パラメータ *stream* **で指定されたファイルのストリームが確保している返却用バッファにパラメータ** *character* **で指定された文字** (1 **文字**) **を格納します。**
なお，本 API **関数の処理が正常終了した際には，“返却した文字** (1 **文字**) **を** int **型に拡張した値”が戻り値として返されます。**
以下に，本 API **関数の記述例を示します。**
なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;   /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                int             character ;                     /* 変数の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                /* 文字の読み込み */
                character = rxfs_fgetc ( stream ) ;
                                                /* 文字の返却 */
                rxfs_ungetc ( character , stream ) ;

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　**返却用バッファに格納された文字** *character* **は，**API **関数** rxfs_fflush**，**rxfs_fgetc**，**rxfs_fseek **などが発行された際に返却用バッファからクリアされます。**
なお，RX-FS850 **では，返却用バッファがクリアされることなく本** API **関数を再発行した際には，戻り値として** FXE_ALREADYUNGET **を返しています。**

注意 3　**本** API **関数における返却処理は，返却用バッファに対してのみ行われます。したがって，パラメータ** *stream* **で指定されたファイルに対する文字** *character* **の書き込みは行われません。**

- rxfs_fputc

  パラメータ *stream* **で指定されたファイルのファイル・ポインタの位置からパラメータ** *character* **で指定された文字** (1 **文字**) **を書き込みます。**
  なお，パラメータ *character* **で指定された領域には，該当ファイルに書き込む文字を設定します。**
  また，本 API **関数の処理が正常終了した際には，“ 書き込んだ文字** (1 **文字**) **を** int **型に拡張した値 ” が戻り値として返されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;   /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                int             character = 'H' ;               /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                rxfs_fputc ( character , stream ) ;     /* 文字の書き込み */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　RX-FS850 **では，文字の書き込み処理が完了した際には，ファイル・ポインタの位置を，本** API **関数発行以前の位置から “ 書き込んだ文字のサイズ** (0x1 **バイト**) **” だけ進めています。**

4.6.5　文字列の読み込み / 書き込み

文字列の読み込み / 書き込みは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fgets

  パラメータ *stream* で指定されたファイルのファイル・ポインタの位置からパラメータ *number* - 1 で算出された数の文字列，または，改行文字 LF まで，または，ファイルの末尾までの文字列をパラメータ *string* で指定された領域に格納します。
  なお，本 API 関数の処理が正常終了した際には，“ 読み込んだ文字列を格納した領域へのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                char            string [ 0x100 ] ;              /* 変数の宣言 */
                int             number = 0x100 ;                /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                        /* 文字列の読み込み */
                rxfs_fgets ( string , number , stream ) ;

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだ文字の合計サイズ (単位：バイト) ” だけ進めています。

注意 3　RX-FS850 では，文字の読み込み処理が “ 改行文字 LF の検出 ” により完了した際には，パラメータ *string* で指定された領域に “ 改行文字 LF を含まない文字列 ” を格納してます。

注意 4　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

- rxfs_fputs

  パラメータ *stream* **で指定されたファイルのファイル・ポインタの位置からパラメータ** *string* **で指定された文字列を書き込みます。**
  なお，パラメータ *string* **で指定された領域には，該当ファイルに書き込む文字列を設定します。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                char            string [ ] = "Hello World" ;    /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                rxfs_fputs ( string , stream ) ;        /* 文字列の書き込み */

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　RX-FS850 **では，文字の書き込み処理が完了した際には，ファイル・ポインタの位置を，本** API **関数発行以前の位置から“書き込んだ文字の合計サイズ (単位：バイト)”だけ進めています。**

4.6.6　ファイル・ポインタの位置変更 / 位置獲得

ファイル・ポインタの位置変更 / 位置獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fseek

  パラメータ *stream* で指定されたファイルのファイル・ポインタの位置を，本 API 関数発行時の位置からパラメータ *offset*，および，*whence* で指定された位置に変更します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                long            offset = 0x0 ;                  /* 変数の宣言，初期化 */
                int             whence = FX_SEEK_SET ;          /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;

                .....
                .....
                .....

                                                        /* ファイル・ポインタの位置変更 */
                rxfs_fseek ( stream , offset , whence ) ;

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　本 API 関数が正常終了した際には，RX-FS850 はパラメータ *stream* で指定されたファイルの EOF フラグ，および，エラー・フラグのクリア処理を実行します。

- rxfs_rewind

  パラメータ *stream* **で指定されたファイルのファイル・ポインタの位置を，本** API **関数発行時の位置からファイルの先頭に変更します。**
  したがって，本 API **関数は，**API **関数** rxfs_fseek **を以下のように発行した場合と同等と意味を持ちます。**

  (void) rxfs_fseek (*stream* , 0x0 , FX_SEEK_SET) ;

  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;

                ..........
                ..........
                ..........

                rxfs_rewind ( stream ) ;        /* ファイル・ポインタの位置変更 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　**本** API **関数が正常終了した際には，**RX-FS850 **はパラメータ** *stream* **で指定されたファイルの** EOF **フラグ，および，エラー・フラグのクリア処理を実行します。**

- rxfs_ftell

  パラメータ *stream* **で指定されたファイルのファイル・ポインタの位置を獲得します。**
  なお，本 API **関数の処理が正常終了した際には，“ 獲得したファイル・ポインタの位置 ”が戻り値として返されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;   /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                long            offset ;                        /* 変数の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;

                ..........
                ..........
                ..........

                offset = rxfs_ftell ( stream ) ;        /* ファイル・ポインタの位置獲得 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ストリーム** *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　**獲得したファイル･ポインタの位置は,該当ファイルの先頭からのオフセット値 (単位:バイト) となります。**

4.6.7　フラグの状態獲得 / クリア

フラグの状態獲得 / クリアは，以下に示した API 関数を処理プログラム（タスク）から発行することにより実現されます。

- rxfs_feof

  パラメータ *stream* で指定されたファイルの EOF フラグの状態を獲得します。
  なお，本 API 関数の処理が正常終了した際には，“ 獲得した EOF フラグの状態 ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                char            ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 ;                    /* 変数の宣言，初期化 */
                long            number = 0x10 ;                 /* 変数の宣言，初期化 */
                int             ret ;                           /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                        /* データの読み込み */
                rxfs_fread ( ptr , size , number , stream ) ;
                ret = rxfs_feof ( stream ) ;            /* EOF フラグの状態獲得 */

                if ( ret != 0x0 ) {
                        /* EOF 検出時の処理 */
                        ……
                        ……
                        ……
                } else {
                        /* EOF 未検出時の処理 */
                        ……
                        ……
                        ……
                }

                ……
                ……
                ……

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

- rxfs_ferror

  パラメータ *stream* **で指定されたファイルのエラー・フラグの状態を獲得します。**
  なお，本 API **関数の処理が正常終了した際には，“ 獲得したエラー・フラグの状態 ” が戻り値として返されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                               /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;                       /* 変数の宣言，初期化 */
                char            ptr [ 0x100 ] ;                         /* 変数の宣言 */
                long            size = 0x8 ;                            /* 変数の宣言，初期化 */
                long            number = 0x10 ;                         /* 変数の宣言，初期化 */
                int             ret ;                                   /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                        /* データの読み込み */
                rxfs_fread ( ptr , size , number , stream ) ;
                ret = rxfs_ferror ( stream ) ;          /* エラー・フラグの状態獲得 */

                if ( ret != 0x0 ) {
                                /* エラー検出時の処理 */
                                ……
                                ……
                                ……
                } else {
                                /* エラー未検出時の処理 */
                                ……
                                ……
                                ……
                }

                ……
                ……
                ……

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意　ストリーム *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

- rxfs_clearerr

  パラメータ *stream* **で指定されたファイルの** EOF **フラグ，および，エラー・フラグをクリアします。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                FX_FILE         *stream ;                       /* データ構造体の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            mode [ ] = "w+" ;               /* 変数の宣言，初期化 */
                char            ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 ;                    /* 変数の宣言，初期化 */
                long            number = 0x10 ;                 /* 変数の宣言，初期化 */
                int             ret ;                           /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                stream = rxfs_fopen ( filename , mode ) ;
                                                        /* データの読み込み */
                rxfs_fread ( ptr , size , number , stream ) ;
                ret = rxfs_feof ( stream ) ;            /* EOF フラグの状態獲得 */

                if ( ret != 0x0 ) {
                                /* エラー検出時の処理 */
                                                        /* EOF フラグ，および，エラー・フラグのクリア */
                                rxfs_clearerr ( stream ) ;

                                .....
                                .....
                                .....
                } else {
                                /* エラー未検出時の処理 */
                                .....
                                .....
                                .....
                }

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意　ストリーム *stream* **には，**API **関数** rxfs_fopen**，または，**rxfs_tmpfile **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

4.7　低水準入出力管理機能

RX-FS850 では，本機能を実現する際，ストリーム入出力管理機能を利用しています。

4.7.1　ファイルのオープン / クローズ

ファイルのオープン / クローズは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_open

  パラメータ *filename* で指定されたファイルをパラメータ *mode* で指定されたモードでオープンします。
  なお，RX-FS850 では，ファイルのオープン処理として，API 関数 (rxfs_close，rxfs_fsync など) を発行する際，オープンしたファイル毎に必要となるファイル記述子の獲得，および，初期化を行っています。
  また，本 API 関数の処理が正常終了した際には，“ オープンしたファイルのファイル記述子 ” が戻り値として返されます。
  以下に，パラメータ *mode* の代表的な指定形式を示します。

  - FX_O_RDONLY

    パラメータ *filename* で指定された既存ファイルを読み込み専用でオープンします。
    なお，データの読み込み開始位置は，ファイル・ポインタで指定された位置からとなります。

  - FX_O_WRONLY | FX_O_APPEND

    パラメータ *filename* で指定された既存ファイルを書き込み専用でオープンします。
    なお，データの書き込み開始位置は，ファイルの末尾からとなります。

  - FX_O_WRONLY | FX_O_CREATE

    パラメータ *filename* で指定された既存ファイルが存在しなかった場合には，該当ファイルを新規生成したのち，書き込み専用でオープンします。
    なお，データの書き込み開始位置は，ファイル・ポインタで指定された位置からとなります。

  - FX_O_WRONLY | FX_O_TRUNC

    パラメータ *filename* で指定された既存ファイルを書き込み専用でオープンします。
    なお，データの書き込み開始位置は，ファイルの先頭からとなります。

  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include    <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include    <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
            int         fildes ;                        /* 変数の宣言 */
            char        filename [ ] = "C:\\tmp\\smp.txt" ;   /* 変数の宣言，初期化 */
            int         mode = FX_O_RDONLY ;            /* 変数の宣言，初期化 */

            rxfs_TaskInit ( ) ;                         /* タスクの登録 */
                                                        /* ファイルのオープン */
            fildes = rxfs_open ( filename , mode ) ;

            .....
            .....
            .....

            rxfs_TaskTerm ( ) ;                         /* タスクの登録解除 */
            ext_tsk ( ) ;                               /* タスクの終了処理 */
}
```

注意 1　**ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ相対パスを設定します。**
なお，RX-FS850 **では，パラメータ *filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，**FXE_PATHTOOLONG **を返しています。**

注意 2　RX-FS850 **では，パラメータ *filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。**

注意 3　**オープンするファイルのモード *mode* については，同時に指定することが禁止されているもの，組み合わせて指定するものがあります。**

- FX_O_RDONLY，FX_O_WRONLY，FX_O_RDWR **は，同時に指定することが禁止**
- FX_O_APPEND **は，**FX_O_WRONLY，**または，**FX_O_RDWR **と組み合わせて指定**
- FX_O_CREATE **は，**FX_O_WRONLY，**または，**FX_O_RDWR **と組み合わせて指定**
- FX_O_TRUNC **は，**FX_O_WRONLY，**または，**FX_O_RDWR **と組み合わせて指定**
- FX_O_EXCL **は，**FX_O_CREATE **と組み合わせて指定**

注意 4　**本** API **関数では，**API **関数** rxfs_SystemInit **を発行した際に指定したシステム初期化情報** _FX_INIT_PACK **のメンバ** FmngNum **と同数のファイル管理構造体，または，**FstmNum **と同数のストリームが既に使用されていた場合には，ファイルのオープン処理は行わず，エラー・コードとして** FXE_FILEMAX**，または，**FXE_OPENMAX **を返しています。**
なお，システム初期化情報 _FX_INIT_PACK **についての詳細は，「**5.4.1　**システム初期化情報**」**を参照してください。**

- rxfs_close

  パラメータ *fildes* で指定されたファイルをクローズします。
  なお，RX-FS850 では，ファイルのクローズ処理として，API 関数 rxfs_open の発行により獲得したファイル記述子の返却を行っています。
  したがって，本 API 関数の発行後，該当ファイルに対する操作 (API 関数 rxfs_fsync，rxfs_read などの発行) が禁止されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int             fildes ;                        /* 変数の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                int             mode = FX_O_FX_O_RDONLY ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ファイルのオープン */
                fildes = rxfs_open ( filename , mode ) ;

                .....
                .....
                .....

                rxfs_close ( fildes ) ;         /* ファイルのクローズ */
                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　パラメータ *fildes* で指定されたファイルが “ 書き込み可能なモード ” でオープンされていた場合，RX-FS850 は該当バッファをフラッシュしたのち，ファイルのクローズ処理を実行します。
ただし，該当ファイルのドライブが API 関数 rxfs_umountforce の発行によりアンマウントされていた際，および，API 関数 rxfs_flushall の発行によりアクセス禁止状態へと遷移していた際には，フラッシュ処理は実行されず，クローズ処理のみが実行されます。

4.7.2　バッファのフラッシュ

バッファのフラッシュは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_fsync

  パラメータ *fildes* で指定されたファイルのバッファをフラッシュします。
  これにより，書き込み要求 (API 関数 rxfs_write などの発行) を行った際，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include      <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include      <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int             fildes ;                                /* 変数の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                                                                        /* 変数の宣言，初期化 */
                int             mode = FX_O_WRONLY | FX_O_APPEND ;
                unsigned int    cnt ;                                   /* 変数の宣言 */
                unsigned char   ptr [ 0x100 ] ;                         /* 変数の宣言 */
                long            size = 0x8 * 0x10 ;                     /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                fildes = rxfs_open ( filename , mode ) ;

                for ( cnt = 0x0 ; cnt < 0x100 ; cnt++ ) {
                                                                        /* 変数の初期化 */
                        ptr [ cnt ] = ( unsigned char ) cnt & 0xff ;
                }

                rxfs_write ( fildes , ptr , size ) ;    /* データの書き込み */
                rxfs_fsync ( fildes ) ;                 /* バッファのフラッシュ */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

4.7.3　データの読み込み / 書き込み

データの読み込み / 書き込みは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_read

  パラメータ *fildes* で指定されたファイルのファイル･ポインタの位置からパラメータ *size* で指定されたサイズのデータ，または，ファイルの末尾までのデータをパラメータ *ptr* で指定された領域に格納します。
  なお，本 API 関数の処理が正常終了した際には，“ 読み込んだデータのサイズ (単位：バイト) ”が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int             fildes ;                        /* 変数の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                int             mode = FX_O_RDONLY ;            /* 変数の宣言，初期化 */
                char            ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 * 0x10 ;             /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                fildes = rxfs_open ( filename , mode ) ;
                rxfs_read ( fildes , ptr , size ) ;     /* データの読み込み */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだデータのサイズ (単位：バイト) ” だけ進めています。

- rxfs_write

  パラメータ *fildes* **で指定されたファイルのファイル･ポインタの位置からパラメータ** *size* **で指定されたサイズのデータを書き込みます。**
  なお，パラメータ *ptr* **で指定された領域には，該当ファイルに書き込むデータを設定します。**
  また，本 API **関数の処理が正常終了した際には，“ 書き込んだデータのサイズ (単位：バイト) ”が戻り値として返されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int             fildes ;                        /* 変数の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                                                                /* 変数の宣言，初期化 */
                int             mode = FX_O_WRONLY | FX_O_APPEND ;
                unsigned int    cnt ;                           /* 変数の宣言 */
                unsigned char   ptr [ 0x100 ] ;                 /* 変数の宣言 */
                long            size = 0x8 * 0x10 ;             /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                fildes = rxfs_open ( filename , mode ) ;

                for ( cnt = 0x0 ; cnt < 0x100 ; cnt++ ) {
                                                                /* 変数の初期化 */
                        ptr [ cnt ] = ( unsigned char ) cnt & 0xff ;
                }

                rxfs_write ( fildes , ptr , size ) ;    /* データの書き込み */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　**ファイル記述子** *fildes* **には，**API **関数** rxfs_open **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたファイル記述子の管理を行っているため，本** API **関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。**

注意 2　RX-FS850 **では，データの書き込み処理が完了した際には，ファイル・ポインタの位置を，本** API **関数発行以前の位置から “ 書き込んだデータのサイズ (単位：バイト) ” だけ進めています。**

4.7.4　ファイル・ポインタの位置変更

ファイル・ポインタの位置変更は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_lseek

  パラメータ *fildes* で指定されたファイルのファイル・ポインタの位置を，本 API 関数発行時の位置からパラメータ *offset*，および，*whence* で指定された位置に変更します。
  なお，本 API 関数の処理が正常終了した際には，“ 変更後のファイル・ポインタの位置 ”が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int             fildes ;                        /* 変数の宣言 */
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                                                                /* 変数の宣言，初期化 */
                int             mode = FX_O_WRONLY | FX_O_APPEND ;
                long            offset = 0x0 ;                  /* 変数の宣言，初期化 */
                int             whence = FX_SEEK_SET ;          /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイルのオープン */
                fildes = rxfs_open ( filename , mode ) ;

                .....
                .....
                .....

                                                        /* ファイル・ポインタの位置変更 */
                rxfs_lseek ( fildes , offset , whence ) ;

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　変更後のファイル･ポインタの位置は,該当ファイルの先頭からのオフセット値 (単位:バイト) となります。

4.8　ファイル・アクセス管理機能

4.8.1　ファイル名の変更

ファイル名の変更は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_rename

  パラメータ *oldname* で指定されたファイルのファイル名をパラメータ *newname* で指定されたファイル名に変更します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            oldname [ ] = "C:\\tmp\\smp.txt" ;      /* 変数の宣言，初期化 */
                                                                        /* 変数の宣言，初期化 */
                char            newname [ ] = "C:\\tmp\\sample.txt" ;

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                                                        /* ファイル名の変更 */
                rxfs_rename ( oldname , newname ) ;

                .........................
                .........................
                .........................

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　ファイル名 *oldname*，*newname* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
ただし，RX-FS850 では，パラメータ *oldname*，*newname* に相対パスが設定された際には，相対パスから絶対パスへの変換処理を行うため,変換結果が制限文字数(270文字)を超えた場合には,FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *oldname*，*newname* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，パラメータ *oldname*，*newname* に異なるパス名が指定されていた場合には，ファイル名の変更処理は行わず，ファイルの移動処理を行います。

注意 4　本 API 関数では，パラメータ *oldname*，*newname* に異なるドライブ名が指定されていた場合には，ファイル名の変更処理は行わず，戻り値として FXE_CROSSDRIVE を返しています。

注意 5　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ファイル名の変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

4.8.2　ファイルの削除

ファイルの削除は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_remove

  パラメータ *filename* で指定されたファイルを削除します。
  したがって，本 API 関数は，API 関数 rxfs_unlink を以下のように発行した場合と同等と意味を持ちます。

  rxfs_unlink (*filename*) ;

  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_remove ( filename ) ;      /* ファイルの削除 */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ファイルの削除処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

- rxfs_unlink

  パラメータ *filename* **で指定されたファイルを削除します。**
  したがって，本 API **関数は，**API **関数** rxfs_remove **を以下のように発行した場合と同等と意味を持ちます。**

  rxfs_remove (*filename*) ;

  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_unlink ( filename ) ;      /* ファイルの削除 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ファイル名** *filename* **には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ相対パスを設定します。**
なお，RX-FS850 **では，パラメータ** *filename* **に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数** (270 **文字**) **を超えるような相対パスが設定されていた場合には，**FXE_PATHTOOLONG **を返しています。**

注意 2　RX-FS850 **では，パラメータ** *filename* **で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。**

注意 3　**本** API **関数では，**API **関数** rxfs_SystemInit **を発行した際に指定したシステム初期化情報** _FX_INIT_PACK **のメンバ** FmngNum **と同数のファイル管理構造体が既に使用されていた場合には，ファイルの削除処理は行わず，エラー・コードとして** FXE_FILEMAX **を返しています。**
なお，システム初期化情報 _FX_INIT_PACK **についての詳細は，「**5.4.1　**システム初期化情報」を参照してください。**

4.8.3　ステータス情報の獲得 / 変更

ステータス情報の獲得 / 変更は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_stat

  パラメータ *filename* で指定されたファイルのステータス情報 (サイズ，アクセス日時，更新日時，作成日時など) をパラメータ *pStat* で指定されたパケットに格納します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                        /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                            /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                FX_STAT         pStat ;                                 /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;                     /* タスクの登録 */
                rxfs_stat ( filename , &pStat ) ;       /* ステータス情報の獲得 */

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;                     /* タスクの登録解除 */
                ext_tsk ( ) ;                           /* タスクの終了処理 */
}
```

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ステータス情報の獲得処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 4　ステータス情報 _FX_STAT についての詳細は，「5.4.3　ステータス情報」を参照してください。

- rxfs_chstat

  パラメータ *filename* **で指定されたファイルのステータス情報 (サイズ，アクセス日時，更新日時，作成日時など) をパラメータ** *flag*，*pStat* **で指定された状態に変更します。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            filename [ ] = "C:\\tmp\\smp.txt" ;     /* 変数の宣言，初期化 */
                char            flag = FX_CHSTAT_ATTR ;                 /* 変数の宣言，初期化 */
                FX_STAT         pStat ;                                 /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */

                                                /* データ構造体の初期化 */
                pStat.st_mode = FX_DEATR_READ ;

                                                /* ステータス情報の変更 */
                rxfs_chstat ( filename , flag , &pStat ) ;

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　**ファイル名** *filename* **には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を“\”でつないだ相対パスを設定します。**
なお，RX-FS850 **では，パラメータ** *filename* **に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (**270 **文字) を超えるような相対パスが設定されていた場合には，**FXE_PATHTOOLONG **を返しています。**

注意 2　RX-FS850 **では，パラメータ** *filename* **で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。**

注意 3　**本** API **関数では，**API **関数** rxfs_SystemInit **を発行した際に指定したシステム初期化情報** _FX_INIT_PACK **のメンバ** FmngNum **と同数のファイル管理構造体が既に使用されていた場合には，ステータス情報の変更処理は行わず，エラー・コードとして** FXE_FILEMAX **を返しています。**
なお，システム初期化情報 _FX_INIT_PACK **についての詳細は，「**5.4.1　**システム初期化情報」を参照してください。**

注意 4　**ステータス情報** _FX_STAT **についての詳細は，「**5.4.3　**ステータス情報」を参照してください。**

4.8.4　一時ディレクトリの設定

一時ディレクトリの設定は,以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_settmpdir

  一時ディレクトリをパラメータ *dirname* で指定されたディレクトリに設定します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_settmpdir ( dirname ) ;    /* 一時ディレクトリの設定 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，一時ディレクトリの変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 4　本 API 関数を発行した際，*dirname* で指定される領域に NULL を設定した場合，一時ディレクトリの設定解除が行われます。

4.8.5　一時ファイル名の生成

一時ファイル名の生成は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_tmpnam

  パラメータ *filename* で指定された領域に生成した一時ファイル名を格納します。
  なお，本 API 関数の処理が正常終了した際には，“ 生成した一時ファイル名 (*dirname*\TEMP*xxx*.TMP：*dirname* は一時ディレクトリ名，*xxxx* は任意 0000 ～ 9999 の数字) を格納した領域へのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                char            filename [ 271 ] ;              /* 変数の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_settmpdir ( dirname ) ;    /* 一時ディレクトリの設定 */
                rxfs_tmpnam ( filename ) ;      /* 一時ファイル名の生成 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意　RX-FS850 では，API 関数 rxfs_settmpdir の発行により設定された一時ディレクトリに対して，一時ファイル名の生成処理を行います。

4.9　ディレクトリ制御管理機能

4.9.1　ディレクトリの生成 / 削除

ディレクトリの生成/削除は,以下に示したAPI関数を処理プログラム(タスク)から発行することにより実現されます。

- rxfs_mkdir

  パラメータ *mkname* で指定されたディレクトリを生成します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            mkname [ ] = "C:\\newdir" ;     /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_mkdir ( mkname ) ;         /* ディレクトリの生成 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ディレクトリ名 *mkname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *mkname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *mkname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ディレクトリの生成処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

- rxfs_rmdir

  パラメータ *rmname* で指定されたディレクトリを削除します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            rmname [ ] = "C:\\newdir" ;         /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_rmdir ( rmname ) ;         /* ディレクトリの削除 */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ディレクトリ名 *rmname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *rmname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *rmname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ディレクトリの削除処理は行わず，エラー・コードとして FXE_FILEMAX 返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

4.9.2　カレント・ディレクトリの設定 / 獲得

カレント・ディレクトリの設定 / 獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_chdir

  カレント・ディレクトリをパラメータ *dirname* で指定されたディレクトリに設定します。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                rxfs_chdir ( dirname ) ;        /* カレント・ディレクトリの設定 */

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数の変更対象はカレント・ディレクトリの他に，カレント・ドライブも含まれています。このため，本 API 関数の処理が完了した際には，カレント・ドライブもパラメータ *dirname* で指定されたドライブに変更されます。

注意 4　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，カレント・ディレクトリの変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 5　本 API 関数を発行した際，*dirname* で指定される領域に NULL を設定した場合，カレント・ディレクトリの設定解除が行われます。

- rxfs_getwd

  本 API 関数発行時のカレント・ディレクトリをパラメータ *dirname* で指定された領域に格納します。
  なお，本 API 関数の処理が正常終了した際には，“ 獲得したカレント・ディレクトリ名を格納した領域へのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            currentdir [ ] = "C:\\tmp" ;        /* 変数の宣言，初期化 */
                char            dirname [ 272 ] ;                   /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                rxfs_chdir ( currentdir ) ;         /* カレント・ディレクトリの設定 */

                ..........
                ..........
                ..........

                rxfs_getwd ( dirname ) ;            /* カレント・ディレクトリの獲得 */

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意 1　RX-FS850 では，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パスを *dirname* で指定された領域に格納しています。

注意 2　RX-FS850 では，*dirname* で指定された領域に格納するカレント・ディレクトリ名の最大文字数を 270 文字に規定しています。このため，ユーザは，獲得したカレント・ディレクトリ名の格納領域として，文字列終端の区切り文字 “ \ ”，および，NULL 文字 “ \0 ” を加えた 272 文字分の領域 (272 バイト) を確保する必要があります。

4.9.3　絶対パスの獲得

絶対パスの獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_realpath

  パラメータ *relpath* で指定されたディレクトリの絶対パスをパラメータ *abspath* で指定された領域に格納します。
  なお，本 API 関数の処理が正常終了した際には，“ 獲得した絶対パスを格納した領域へのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char        dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                char        relpath [ ] = "..\\newdir" ;    /* 変数の宣言，初期化 */
                char        abspath [ 271 ] ;               /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                 /* タスクの登録 */
                rxfs_chdir ( dirname ) ;            /* カレント・ディレクトリの設定 */
                                                    /* 絶対パスの獲得 */
                rxfs_realpath ( relpath , abspath ) ;

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;                 /* タスクの登録解除 */
                ext_tsk ( ) ;                       /* タスクの終了処理 */
}
```

注意 1　相対パス *relpath* には，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだパスを設定します。
ただし，RX-FS850 では，パラメータ *relpath* の変換結果が制限文字数 (270 文字) を超えた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *relpath* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　RX-FS850 では，パラメータ *relpath* で指定された相対パスにショート・ファイル名が含まれていた場合，該当ショート・ファイル名をロング・ファイル名に変換したのち，パラメータ *abspath* で指定された領域に格納しています。

注意 4　RX-FS850 では，パラメータ *abspath* で指定された領域に格納する絶対パスの最大文字数を 270 文字に規定しています。このため，ユーザは，獲得した絶対パスの格納領域として，文字列終端の NULL 文字 “ \0 ” を加えた 271 文字分の領域 (271 バイト) を確保する必要があります。

注意 5　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，絶対パスの獲得処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

4.9.4　ディレクトリのオープン / クロース

ディレクトリのオープン / クローズは，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_opendir

  パラメータ *dirname* で指定されたディレクトリをオープンします。
  なお，RX-FS850 では，ディレクトリのオープン処理として，API 関数 (rxfs_closedir，rxfs_readdir) を発行する際，オープンしたディレクトリ毎に必要となるストリームの獲得，および，初期化を行っています。
  また，本 API 関数の処理が正常終了した際には，“ オープンしたディレクトリのストリームへのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                FX_DIR          *stream ;                       /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ディレクトリのオープン */
                stream = rxfs_opendir ( dirname ) ;

                ..........................
                ..........................
                ..........................

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，パラメータ *dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，パラメータ *dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体，または，FstmNum と同数のストリームが既に使用されていた場合には，ディレクトリのオープン処理は行わず，エラー・コードとして FXE_FILEMAX，または，FXE_OPENMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

- rxfs_closedir

  パラメータ *stream* **で指定されたディレクトリをクローズします。**
  なお，RX-FS850 **では，ディレクトリのクローズ処理として，**API **関数** rxfs_opendir **の発行により獲得したストリームの返却を行っています。**
  したがって，本 API **関数の発行後，該当ディレクトリに対する操作 (**API **関数** rxfs_readdir **の発行) が禁止されます。**
  以下に，本 API **関数の記述例を示します。**
  なお，記述例中の ext_tsk **は，**RX850 Pro **が提供しているシステム・コールです。**

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                FX_DIR          *stream ;                       /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ディレクトリのオープン */
                stream = rxfs_opendir ( dirname ) ;

                ..........
                ..........
                ..........

                rxfs_closedir ( stream ) ;      /* ディレクトリのクローズ */
                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意　ストリーム *stream* **には，**API **関数** rxfs_opendir **の戻り値を設定します。**
ただし，RX-FS850 **では，タスクを単位としたストリームの管理を行っているため，本** API **関数の操作対象ディレクトリは，同一タスク内でオープン処理が行われたディレクトリに限定されます。**

4.9.5　ディレクトリ項目情報の獲得

ディレクトリ項目情報の獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_readdir

  パラメータ*stream*で指定されたディレクトリのディレクトリ項目情報(該当ディレクトリの配下に存在しているディレクトリ名，ファイル名など) を獲得します。
  なお，本 API 関数の処理が正常終了した際には，“ 獲得したディレクトリ項目情報へのポインタ ” が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include        <stdrx85p.h>                    /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include        <rxfs.h>                        /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                char            dirname [ ] = "C:\\tmp" ;       /* 変数の宣言，初期化 */
                FX_DIR          *stream ;                       /* データ構造体の宣言 */
                FX_DIRENT       *pDirent ;                      /* データ構造体の宣言 */

                rxfs_TaskInit ( ) ;             /* タスクの登録 */
                                                /* ディレクトリのオープン */
                stream = rxfs_opendir ( dirname ) ;
                                                /* ディレクトリ項目情報の獲得 */
                pDirent = rxfs_readdir ( stream ) ;

                .....
                .....
                .....

                rxfs_TaskTerm ( ) ;             /* タスクの登録解除 */
                ext_tsk ( ) ;                   /* タスクの終了処理 */
}
```

注意 1　ストリーム *stream* には，API 関数 rxfs_opendir の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ディレクトリは，同一タスク内でオープン処理が行われたディレクトリに限定されます。

注意 2　ディレクトリ項目情報 _FX_DIRENT についての詳細は，「5.4.4　ディレクトリ項目情報」を参照してください。

4.10　エラー管理機能

4.10.1　エラー・コードの獲得

エラー・コードの獲得は，以下に示した API 関数を処理プログラム (タスク) から発行することにより実現されます。

- rxfs_geterrno

  RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を発行した際に返却されるエラー・コードを獲得します。
  なお，本 API 関数の処理が正常終了した際には，" 獲得したエラー・コード (FXE_OK，FXE_SYSTEMNOTINIT など) " が戻り値として返されます。
  以下に，本 API 関数の記述例を示します。
  なお，記述例中の ext_tsk は，RX850 Pro が提供しているシステム・コールです。

```
#include      <stdrx85p.h>                  /* RX850 Pro 用標準ヘッダ・ファイルの定義 */
#include      <rxfs.h>                      /* RX-FS850 用標準ヘッダ・ファイルの定義 */


void
func_task ( INT stacd ) {
                int           err ;                       /* 変数の宣言 */
                fx_u8         drvname = 'A' ;             /* 変数の宣言，初期化 */
                char          driver [ ] = "RAMDISK" ;    /* 変数の宣言，初期化 */
                fx_u8         u_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8         p_id = 0x0 ;                /* 変数の宣言，初期化 */
                fx_u8         mode = FX_MNT_ASYNC ;       /* 変数の宣言，初期化 */
                int           ercd ;                      /* 変数の宣言 */

                rxfs_TaskInit ( ) ;                       /* タスクの登録 */
                                                          /* ドライブのマウント */
                err = rxfs_mount ( drvname , driver , u_id , p_id , mode ) ;

                if ( err == -1 ) {
                                                          /* エラー・コードの獲得 */
                        ercd = rxfs_geterrno ( ) ;

                        if ( ercd == FXE_MOUNTED ) {
                                                          /* 異常終了 (FXE_MOUNTED) した場合の処理
                                ..........
                                ..........
                                ..........
                        } else {
                                                          /* 異常終了 (FXE_MOUNTED 以外 ) した場合の処理
                                ..........
                                ..........
                                ..........
                        }
                }

                ..........
                ..........
                ..........

                rxfs_TaskTerm ( ) ;                       /* タスクの登録解除 */
                ext_tsk ( ) ;                             /* タスクの終了処理 */
}
```

注意　RX-FS850 **では，タスクを単位としたエラー・コードの管理を行っているため，本** API **関数の獲得対象エラー・コードは，同一タスク内で直前に発行された** API **関数のエラー・コードに限定されます。**

第 5 章　API 関数

本章では，RX-FS850 が提供しているアプリケーション・プログラム・インタフェース関数 (API 関数) について解説しています。

5.1　概要

RX-FS850 が提供している API 関数は，ユーザが記述した処理プログラムから RX-FS850 が直接管理している資源を間接的に操作するために用意されたサービス・ルーチンです。
以下に，RX-FS850 が提供している API 関数を管理モジュール別に示します。

- **メモリ管理機能** (2 種類)

  rxfs_HeapInit　rxfs_HeapTerm

- **システム管理機能** (4 種類)

  rxfs_SystemInit　rxfs_SystemTerm　rxfs_TaskInit　rxfs_TaskTerm

- **ドライブ制御管理機能** (6 種類)

  rxfs_mount　rxfs_umount　rxfs_umountforce　rxfs_mountinfo
  rxfs_flushall　rxfs_format

- **ストリーム入出力管理機能** (17 種類)

  rxfs_fopen　rxfs_tmpfile　rxfs_fclose　rxfs_fflush
  rxfs_fread　rxfs_fwrite　rxfs_fgetc　rxfs_ungetc
  rxfs_fputc　rxfs_fgets　rxfs_fputs　rxfs_fseek
  rxfs_rewind　rxfs_ftell　rxfs_feof　rxfs_ferror
  rxfs_clearerr

- **低水準入出力管理機能** (6 種類)

  rxfs_open　rxfs_close　rxfs_fsync　rxfs_read
  rxfs_write　rxfs_lseek

- **ファイル・アクセス管理機能** (7 種類)

  rxfs_rename　rxfs_remove　rxfs_unlink　rxfs_stat
  rxfs_chstat　rxfs_settmpdir　rxfs_tmpnam

- **ディレクトリ制御管理機能** (8 種類)

  rxfs_mkdir　rxfs_rmdir　rxfs_chdir　rxfs_getwd
  rxfs_realpath　rxfs_opendir　rxfs_closedir　rxfs_readdir

- **エラー管理機能** (1 種類)

  rxfs_geterrno

5.2　API 関数の呼び出し

API 関数を C 言語，および，アセンブリ言語で記述された処理プログラムから発行する場合の呼び出し方法を以下に示します。

- C 言語

  API 関数を C 言語で記述された処理プログラムから発行する場合，通常の C 言語関数と同様の方法で呼び出しを行うことにより，API 関数のパラメータは RX-FS850 に引き数として渡され，該当処理が実行されます。

- アセンブリ言語

  API 関数をアセンブリ言語で記述された処理プログラムから発行する場合，ユーザが開発環境として使用する C コンパイラ・パッケージの関数呼び出し規約に従ったパラメータ，および，戻り番地の設定を行ったのち，jarl 命令による呼び出しを行うことにより，API 関数のパラメータは RX-FS850 に引き数として渡され，該当処理が実行されます。

注意　RX-FS850 が提供する API 関数を処理プログラムから発行する場合，以下に示したヘッダ・ファイルの定義 (インクルード処理) を行う必要があります。

stdrx85p.h　：RX850 Pro 用標準ヘッダ・ファイル
rxfs.h　：RX-FS850 用標準ヘッダ・ファイル

5.3　データ・マクロ

RX-FS850 が提供する API 関数を発行する際に使用する各種データ・マクロ (データ・タイプ，基準点など) について以下に示します。

5.3.1　データ・タイプ

表 5-1 に，API 関数を発行する際に使用する各種パラメータのデータ・タイプ一覧を示します。
なお，データ・タイプのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_types.h で行われています。

表 5-1　データ・タイプ

マクロ	型	意味
fx_s8	char	符号付き 8 ビット整数
fx_s16	short	符号付き 16 ビット整数
fx_s32	long	符号付き 32 ビット整数
fx_u8	unsigned char	符号無し 8 ビット整数
fx_u16	unsigned short	符号無し 16 ビット整数
fx_u32	unsigned long	符号無し 32 ビット整数
FX_INIT_PACK	struct _FX_INIT_PACK	システム初期化情報
FX_DRV_MNG	struct _FX_DRV_MNG	ドライバ管理構造体
FX_FILE	struct _FX_FILE	ストリーム情報
FX_STAT	struct _FX_STAT	ステータス情報
FX_DIR	struct _FX_FILE	ストリーム情報
FX_DIRENT	struct _FX_DIRENT	ディレクトリ項目情報

5.3.2　ドライバ・タイプ

表 5-2 に，API 関数 rxfs_SystemInit を発行する際に使用するドライバ・タイプ一覧を示します。
なお，ドライバ・タイプのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

表 5-2　ドライバ・タイプ

マクロ	数値	意味
FX_DRVTYPE_FAT	0x1	FAT ファイル・システム
FX_DRVTYPE_CDROM	0x2	CD-ROM ファイル・システム

5.3.3　マウント・モード

表 5-3 に，API 関数 rxfs_mount を発行する際に使用するマウント・モード一覧を示します。
なお，マウント・モードのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

表 5-3　マウント・モード

マクロ	数値	意味
FX_MNT_ASYNC	0x0	非同期モード
FX_MNT_SYNC	0x80	同期モード
FX_MNT_FORMAT	0x40	フォーマット・モード

5.3.4　基準点

表 5-4 に，API 関数 rxfs_fseek，rxfs_lseek を発行する際に使用する基準点一覧を示します。
なお，基準点のマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

表 5-4　基準点

マクロ	数値	意味
FX_SEEK_SET	0x0	ファイルの先頭
FX_SEEK_CUR	0x1	現在のファイル・ポインタの位置
FX_SEEK_END	0x2	ファイルの末尾

5.3.5　オープン・モード

表 5-5 に，API 関数 rxfs_open を発行する際に使用するオープン・モード一覧を示します。
なお，オープン・モードのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

表 5-5　オープン・モード

マクロ	数値	意味
FX_O_RDONLY	0x0	読み込み専用
FX_O_WRONLY	0x1	書き込み専用
FX_O_RDWR	0x2	読み込み / 書き込み両用
FX_O_APPEND	0x8	データの追加
FX_O_CREATE	0x200	ファイルの新規生成
FX_O_TRUNC	0x400	データの上書き
FX_O_EXCL	0x800	ファイル存在時エラー

5.3.6　ファイル属性

表 5-6 に，API 関数 rxfs_stat，rxfs_chstat を発行する際に使用するファイル属性一覧を示します。
なお，ファイル属性のマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

表 5-6　ファイル属性

マクロ	数値	意味
FX_DEATR_READ	0x1	読み込み専用
FX_DEATR_HIDDEN	0x2	隠しファイル
FX_DEATR_SYSTEM	0x4	システム・ファイル
FX_DEATR_LABEL	0x8	ボリューム・ラベル
FX_DEATR_DIR	0x10	サブディレクトリ
FX_DEATR_ARCHIVE	0x20	アーカイブ・ファイル
FX_DEATR_REL	0x40	関連ファイル
FX_DEATR_EXTENTRY	0x80	拡張レコード
FX_DEATR_PROTECT	0x100	アクセス制限
FX_DEATR_NOLASTENTRY	0x200	非最終ディレクトリ・レコード

5.3.7　ステータス変更フラグ

表 5-7 に，API 関数 rxfs_chstat を発行する際に使用するステータス変更フラグ一覧を示します。
なお，ステータス変更フラグのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

表 5-7　ステータス変更フラグ

マクロ	数値	意味
FX_CHSTAT_ATTR	0x1	ファイルの属性
FX_CHSTAT_ATIME	0x4	ファイルのアクセス日時
FX_CHSTAT_MTIME	0x8	ファイルの更新日時
FX_CHSTAT_CTIME	0x10	ファイルの作成日時

5.3.8　戻り値

表 5-8 に，API 関数からの戻り値一覧を示します。
なお，戻り値のマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_syserrno.h で行われています。

表 5-8　戻り値

マクロ	数値	意味
FXSE_OK	0x0	正常終了
FXSE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXSE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXSE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXSE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXSE_SYSTEMINIT	0x70	既に rxfs_SystemInit が発行されています
FXSE_SEM	0x71	セマフォの生成処理，または，削除処理が失敗しました
FXSE_MBX	0x72	メールボックスの生成処理，または，削除処理が失敗しました
FXSE_TASKEXIST	0x73	rxfs_TaskTerm を発行していないタスクが存在しています
FXSE_TASKINIT	0x74	既に rxfs_TaskInit が発行されています
FXSE_TASKMAX	0x75	管理領域に空きがありません
FXSE_FILEEXIST	0x76	クローズ処理の実行されていないファイルが存在しています
FXSE_PARAMETER	0x77	パラメータの指定が不正です
FXSE_NOTUMOUNTED	0x78	rxfs_umount の発行されていないドライブが存在しています

5.3.9　エラー・コード

表 5-9 に，API 関数を発行した際に返却されるエラー・コード一覧を示します。
なお，エラー・コードのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_errno.h で行われています。

表 5-9　エラー・コード

マクロ	数値	意味
FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_OPENMAX	0x5	ストリームに空きがありません
FXE_MOUNTED	0x10	既に rxfs_mount が発行されています
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません，または，既に操作対象ドライブがアンマウントされています
FXE_DRIVEBUSY	0x12	該当ドライブに対する操作が行われています
FXE_DISKFULL	0x13	デバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブのマウント・モードが不正です
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_FILEEXIST	0x21	既に該当ファイルが存在しています
FXE_FILEBUSY	0x22	該当ファイルがオープンされています
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_NOTDIR	0x24	操作対象にファイルを指定しています
FXE_DIR	0x25	操作対象にディレクトリを指定しています
FXE_DIRNOTEMPTY	0x26	該当ディレクトリの配下にディレクトリ，または，ファイルが存在しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です
FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_NOOPEN	0x30	該当ファイル，または，該当ディレクトリがオープンされていません
FXE_READOPEN	0x31	該当ファイルは読み込み専用でオープンされています
FXE_WRITEOPEN	0x32	該当ファイルは書き込み専用でオープンされています
FXE_DUPWRITE	0x33	既に該当ファイルは書き込み専用，または，読み込み / 書き込み両用でオープンされています

マクロ	数値	意味
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_FINDEOF	0x35	ファイルの末尾まで読み込みました
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_ILLMODE	0x42	モードの指定が不正です
FXE_TMPNAMMAX	0x43	一時ファイルの生成数が最大生成数を超えています
FXE_NOSETTMPDIR	0x44	一時ディレクトリの設定が行われていません
FXE_ALREADYUNGET	0x45	既に rxfs_ungetc が発行されています
FXE_NOUNGET	0x46	ファイル・ポインタが該当ファイルの先頭を指しています
FXE_EOFUNGET	0x47	返却する文字に EOF を指定しています
FXE_CROSSDRIVE	0x48	ドライブの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

注意　返却されたエラー・コードは，API **関数** rxfs_geterrno **の発行により獲得することができます。**

5.4　データ構造体

RX-FS850 が提供する API 関数を発行する際に使用する各種データ構造体 (システム初期化情報，ドライバ管理構造体など) について以下に示します。

5.4.1　システム初期化情報

以下に，API 関数 rxfs_SystemInit を発行する際に使用するシステム初期化情報 _FX_INIT_PACK を示します。
なお，システム初期化情報 _FX_INIT_PACK の定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

```
struct          _FX_INIT_PACK {
                FX_DRV_MNG    *DrvMng ;     /* ドライバ管理構造体配列へのポインタ */
                fx_u8         DrvNum ;      /* ドライバの総数 */
                fx_u8         RFU1 ;        /* システム予約領域 */
                fx_u16        TaskNum ;     /* rxfs_TaskInit を発行するタスクの総数 */
                fx_u16        FmngNum ;     /* ファイル管理構造体の総数 */
                fx_u16        FstmNum ;     /* ストリームの最大生成数 */
                fx_u16        TmpFilNum ;   /* 一時ファイルの最大生成数 */
} ;
```

以下に，システム初期化情報 _FX_INIT_PACK の詳細を示します。

- DrvMng

  ドライバ管理構造体 _FX_DRV_MNG の配列へのポインタを設定します。

- DrvNum

  ドライバの総数を設定します。
  なお，DrvNum に設定可能な値は，“ DrvMng で指定されたドライバ管理構造体 _FX_DRV_MNG の有効配列要素数 ” に限られます。

- RFU1

  システム予約領域です。
  なお，RFU1 に設定可能な値は “ 0x0 ” に限られます。

- TaskNum

  RX-FS850 が提供している API 関数を利用するタスクの総数 (rxfs_TaskInit を発行するタスクの総数) を設定します。

- FmngNum

  ファイル管理構造体の総数を設定します。
  なお，FmngNum に設定可能な値は，“ 下記計算式から得られた結果 ” に限られます。

  FmngNum = fopen_num + open_num + opendir_num + tmpdir_use + 4

  fopen_num　：rxfs_fopen の発行によりオープンするファイルの総数
  open_num　：rxfs_open の発行によりオープンするファイルの総数
  opendir_num　：rxfs_opendir の発行によりオープンするディレクトリの総数
  tmpdir_use　：rxfs_settmpdir の発行有無 (未発行：0，発行：1)

- FstmNum

  ストリームの最大生成数を設定します。
  なお，FstmNum に設定可能な値は，“ 下記計算式から得られた結果 ” に限られます。

  FstmNum = fopen_num + open_num + opendir_num

  fopen_num　：rxfs_fopen の発行によりオープンするファイルの総数
  open_num　：rxfs_open の発行によりオープンするファイルの総数
  opendir_num　：rxfs_opendir の発行によりオープンするディレクトリの総数

- TmpFilNum

  一時ファイルの最大生成数を設定します。

5.4.2　ドライバ管理構造体

以下に，API 関数 rxfs_SystemInit を発行する際に“ システム初期化情報 _FX_INIT_PACK ”のメンバとして使用するドライバ管理構造体 _FX_DRV_MNG を示します。

なお，ドライバ管理構造体 _FX_DRV_MNG の定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

```
struct      _FX_DRV_MNG {
            char        DriverName [ 0x9 ] ;    /* デバイス・ドライバ名 */
            fx_u8       DriverType ;            /* ドライバ・タイプ */
            fx_s16      ReqMbxID ;              /* メールボックスの ID 番号 */
            fx_u16      CtrlBuffNum ;           /* コントロール用バッファの総数 */
            fx_u16      DataBuffNum ;           /* データ用バッファの総数 */
} ;
```

以下にドライバ管理構造体 _FX_DRV_MNG の詳細を示します。

- DriverName [0x9]

  9 文字以下のユニークなデバイス・ドライバ名を設定します。
  なお，DriverName に設定されたデバイス・ドライバ名は，rxfs_mount を発行する際に利用されます。

- DriverType

  ドライバ・タイプ (ファイル・システムの種別) を設定します。
  なお，DriverType に設定可能な値は，“ 下記に示した値 ” に限られます。

FX_DRVTYPE_FAT	(0x1)	： FAT ファイル・システム
FX_DRVTYPE_CDROM	(0x2)	： CD-ROM ファイル・システム

- ReqMbxID

  リクエスト・パケット _FX_REQ_PACK を受信する際に利用するメールボックスの ID 番号を設定します。

- CtrlBuffNum

  コントロール用バッファの総数を設定します。
  なお，CtrlBuffNum に設定可能な値は，“ 0x0 以外の値 ” に限られます。

- DataBuffNum

  データ用バッファの総数を設定します。
  なお，DataBuffNum に設定可能な値は，“ 0x0 以外の値 ” に限られます。

5.4.3　ステータス情報

以下に，API 関数 rxfs_stat，rxfs_chstat を発行する際に使用するステータス情報 _FX_STAT を示します。
なお，ステータス情報 _FX_STAT の定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_stdio.h で行われています。

```
struct      _FX_STAT {
            fx_u32      st_size ;     /* ファイルのサイズ ( 単位：バイト ) */
            fx_u32      st_atime ;    /* ファイルのアクセス日時 */
            fx_u32      st_mtime ;    /* ファイルの更新日時 */
            fx_u32      st_ctime ;    /* ファイルの作成日時 */
            fx_u8       st_cmsec ;    /* ファイルの作成日時 ( 単位：10 ミリ秒 )*/
            fx_u8       RFU1 ;        /* システム予約領域 */
            fx_u16      st_mode ;     /* ファイルの属性 */
            fx_u16      RFU2 ;        /* システム予約領域 */
} ;
```

以下に，ステータス情報 _FX_STAT の詳細を示します。

- st_size

  ファイルのサイズ (単位：バイト) を設定します。

- st_atime

  ファイルのアクセス日時を設定します。
  以下に，st_atime の構成を示します。

  ビット 0 〜 15　：システム予約領域
  ビット 16 〜 20　：日
  ビット 21 〜 24　：月
  ビット 25 〜 31　：年 - 1980

- st_mtime

  ファイルの更新日時を設定します。
  以下に，st_mtime の構成を示します。

  ビット 0 〜 4　：秒
  ビット 5 〜 10　：分
  ビット 11 〜 15　：時
  ビット 16 〜 20　：日
  ビット 21 〜 24　：月
  ビット 25 〜 31　：年 - 1980

  注意　ビット 0 〜 4 の単位は，2 ミリ秒となります。

- st_ctime

  ファイルの作成日時を設定します。
  以下に，st_ctime の構成を示します。

  ビット 0 〜 4　：秒
  ビット 5 〜 10　：分
  ビット 11 〜 15　：時
  ビット 16 〜 20　：日
  ビット 21 〜 24　：月
  ビット 25 〜 31　：年 - 1980

  注意　ビット 0 〜 4 の単位は，2 ミリ秒となります。

- st_cmsec

  ファイルの作成日時 (単位：10 ミリ秒) を設定します。

- RFU1

  システム予約領域です。

- st_mode

  ファイルの属性を設定します。
  なお，st_mode **に設定可能な値は，“ 下記に示した値 ” に限られます。**

FX_DEATR_READ	(0x1)	**：読み込み専用**
FX_DEATR_HIDDEN	(0x2)	**：隠しファイル**
FX_DEATR_SYSTEM	(0x4)	**：システム・ファイル**
FX_DEATR_LABEL	(0x8)	**：ボリューム・ラベル**
FX_DEATR_DIR	(0x10)	**：サブディレクトリ**
FX_DEATR_ARCHIVE	(0x20)	**：アーカイブ・ファイル**
FX_DEATR_REL	(0x40)	**：関連ファイル**
FX_DEATR_EXTENTRY	(0x80)	**：拡張レコード**
FX_DEATR_PROTECT	(0x100)	**：アクセス制限**
FX_DEATR_NOLASTENTRY	(0x200)	**：非最終ディレクトリ・レコード**

- RFU2

  システム予約領域です。

5.4.4　ディレクトリ項目情報

以下に，API **関数** rxfs_readdir **の戻り値として返却されるディレクトリ項目情報** _FX_DIRENT **を示します。**
なお，ディレクトリ項目情報 _FX_DIRENT **の定義は，標準ヘッダ・ファイル** nectools32\inc850\rxfs.h **から呼び出されるヘッダ・ファイル** nectools32\inc850\rxfs\fx_stdio.h **で行われています。**

```
struct      _FX_DIRENT {
                char        d_name [ 256 ] ;        /* ロング・ファイル名 */
                char        s_name [ 13 ] ;         /* ショート・ファイル名 */
} ;
```

以下に，ディレクトリ項目情報 _FX_DIRENT **の詳細を示します。**

- d_name [256]

  ロング・ファイル名 (**ディレクトリ名，ファイル名など** **) が格納されます。**

- s_name [13]

  ショート・ファイル名 (**ディレクトリ名，ファイル名など** **) が格納されます。**

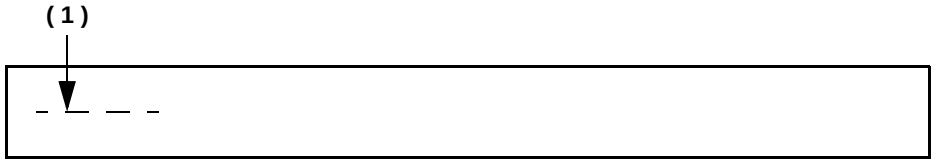
注意 1　**対象ディレクトリが** FAT **ファイル・システムの場合，**d_name [256]**，**s_name [13] **の両メンバに該当ディレクトリ項目情報が格納されます。**

注意 2　**対象ディレクトリが** CD-ROM **ファイル・システムの場合，**d_name [256]，s_name [13] **の何れかのメンバに該当ディレクトリ項目情報が格納されます。**

Joliet　→d_name [256] **に該当ディレクトリ項目情報を格納**
Joliet **以外** →s_name [13] **に該当ディレクトリ項目情報を格納**

5.5　API 関数解説

次項から RX-FS850 が提供している API 関数について，以下の記述フォーマットに従って解説します。
次項から RX-FS850 が提供しているユーザ・オウン関数，および，ドライバ関数について，以下の記述フォーマットに従って解説します。



(2) ⟶ **概要**

(3) ⟶ **C 言語形式**

(4) ⟶ **パラメータ**

I/O	パラメータ	説　　明

(5) ⟶ **機能**

(6) ⟶ **戻り値**

(7) ⟶ **エラー・コード**

（1） 名称

API 関数の名称を示しています。

（2） 概要

API 関数の機能概要を示しています。

（3） C 言語形式

API 関数を C 言語で記述された処理プログラムから発行する際の記述形式を示しています。

（4） パラメータ

API 関数のパラメータを以下の形式で示しています。

I/O	パラメータ	説　　明
A	B	C

A ） パラメータの種類

I　　…RX-FS850 への入力パラメータ
O　　…RX-FS850 からの出力パラメータ

B ） パラメータのデータ・タイプ

C ） パラメータの説明

（5） 機能

API 関数の機能詳細を示しています。

（6） 戻り値

API 関数からの戻り値をデータ・マクロ，および，数値で示しています。

（7） エラー・コード

API 関数を発行した際に返却されるエラー・コードをデータ・マクロ，および，数値で示しています。
なお，エラー・コードは，API 関数 rxfs_geterrno の発行により獲得することができます。

5.5.1　メモリ管理機能

表 5-10 に，RX-FS850 がメモリ管理機能として提供している API 関数の一覧を示します。

表 5-10　メモリ管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_HeapInit	RX-FS850 用ヒープ領域の確保	○	○
rxfs_HeapTerm	RX-FS850 用ヒープ領域の解放	○	○

rxfs_HeapInit

概要

RX-FS850 用ヒープ領域の確保

C 言語形式

int　　rxfs_HeapInit (unsigned long *heapsz*) ;

パラメータ

I/O	パラメータ	説　　明
I	unsigned long　*heapsz* ;	RX-FS850 用ヒープ領域のサイズ (単位：バイト)

機能

heapsz で指定されたサイズのメモリ領域を RX-FS850 用ヒープ領域として確保します。
これにより，RX-FS850 用ヒープ領域は，RX-FS850 の管理対象となります。

注意 1　RX-FS850 用ヒープ領域のサイズ *heapsz* については，以下に示した計算式から求めることができます。

heapsz　= FATdriveNum × { (SctSz + 32) × CtrBuffNum + (ClsSz + 32) × DataBuffNum + 80 }
+ CDdriveNum × { (SctSz + 32) × (CtrBuffNum + DataBuffNum) + 80 }
+ align16 (TaskNum × 20)
+ align16 (FmngNum × 340)
+ align16 (FstmNum × 28)
+ align16 (DrvNum × 16)
+ ChdirTask × 270
+ MaxOpenDir × 272
+ 3K

FATdriveNum　：FAT ファイル・システム用ドライブの総数
rxfs_mount の発行によりマウントする FAT ファイル・システム用ドライブの総数と同値。

CDdriveNum　：CD-ROM ファイル・システム用ドライブの総数
rxfs_mount の発行によりマウントする CD-ROM ファイル・システム用ドライブの総数と同値。

SctSz　：セクタ・サイズ (単位：バイト)

CtrBuffNum　：コントロール用バッファの総数
rxfs_SystemInit の発行時に指定するドライバ管理構造体 _FX_DRV_MNG のメンバ CtrBuffNum と同値。

ClsSz　：クラスタ・サイズ (単位：バイト)

DataBuffNum　：データ用バッファの総数
rxfs_SystemInit の発行時に指定するドライバ管理構造体 _FX_DRV_MNG のメンバ DataBuffNum と同値。

TaskNum　：RX-FS850 が提供している API 関数を利用するタスクの総数

rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ TaskNum と同値。

FmngNum　：ファイル管理構造体の総数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同値。

FstmNum　：ストリームの最大生成数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ FstmNum と同値。

DrvNum　：ドライバの総数
rxfs_SystemInit の発行時に指定するシステム初期化情報 _FX_INIT_PACK のメンバ DrvNum と同値。

ChdirTask　：rxfs_chdir を発行するタスクの総数

MaxOpenDir　：rxfs_opendir の発行により同時にオープンするディレクトリの最大数

注意 2　RX-FS850 用ヒープ領域の確保は，RX850 Pro が管理しているシステム・メモリ UPOL0 から行われます。ただし，RX-FS850 では，RX-FS850 用ヒープ領域を確保する際，該当メモリ領域のクリア処理を行っていないため，確保されたメモリ領域の内容は不定となります。
なお，システム・メモリ UPOL0 についての詳細は，「RX850 Pro ユーザーズ・マニュアル インストレーション編」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

rxfs_HeapTerm

概要

RX-FS850 用ヒープ領域の解放

C 言語形式

```
int             rxfs_HeapTerm ( void ) ;
```

パラメータ

なし

機能

API 関数 rxfs_HeapInit の発行により確保した RX-FS850 用ヒープ領域を解放します。
これにより，RX-FS850 用ヒープ領域は，RX-FS850 の管理対象から除外されます。

注意 1　本 API 関数の発行は，API 関数 rxfs_SystemTerm の処理完了後に行う必要があります。

注意 2　RX-FS850 用ヒープ領域の解放は，RX850 Pro が管理しているシステム・メモリ UPOL0 に対して行われます。
ただし，RX-FS850 では，RX-FS850 用ヒープ領域を解放する際，該当メモリ領域のクリア処理を行っていないため，解放されたメモリ領域の内容は不定となります。
なお，システム・メモリ UPOL0 についての詳細は，「RX850 Pro ユーザーズ・マニュアル インストレーション編」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

5.5.2　システム管理機能

表 5-11 に，RX-FS850 がシステム管理機能として提供している API 関数の一覧を示します。

表 5-11　システム管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_SystemInit	RX-FS850 の初期化	○	○
rxfs_SystemTerm	RX-FS850 の終了	○	○
rxfs_TaskInit	タスクの登録	○	○
rxfs_TaskTerm	タスクの登録解除	○	○

rxfs_SystemInit

概要

RX-FS850 の初期化

C 言語形式

```
int         rxfs_SystemInit ( FX_INIT_PACK *pInitPack ) ;
```

パラメータ

I/O	パラメータ	説　　明
I	FX_INIT_PACK　*pInitPack ;	システム初期化情報を格納したパケットへのポインタ

【 システム初期化情報 _FX_INIT_PACK の構造 】

```
struct      _FX_INIT_PACK {
            FX_DRV_MNG    *DrvMng ;     /* ドライバ管理構造体配列へのポインタ */
            fx_u8         DrvNum ;      /* ドライバの総数 */
            fx_u8         RFU1 ;        /* システム予約領域 */
            fx_u16        TaskNum ;     /* rxfs_TaskInit を発行するタスクの総数 */
            fx_u16        FmngNum ;     /* ファイル管理構造体の総数 */
            fx_u16        FstmNum ;     /* ストリームの最大生成数 */
            fx_u16        TmpFilNum ;   /* 一時ファイルの最大生成数 */
} ;
```

【 ドライバ管理構造体 _FX_DRV_MNG の構造 】

```
struct      _FX_DRV_MNG {
            char          DriverName [ 0x9 ] ;   /* デバイス・ドライバ名 */
            fx_u8         DriverType ;           /* ドライバ・タイプ */
            fx_s16        ReqMbxID ;             /* メールボックスの ID 番号 */
            fx_u16        CtrlBuffNum ;          /* コントロール用バッファの総数 */
            fx_u16        DataBuffNum ;          /* データ用バッファの総数 */
} ;
```

機能

pInitPack で指定された情報をもとに，RX-FS850 が提供する機能を実現するうえで必要となる各種初期化処理を実行します。

なお，RX-FS850 では，RX-FS850 の初期化処理として，RX-FS850 が提供している機能を実現する際に必要となる各種資源の生成，および，メモリ領域の確保を行っています。

注意 1　本 API 関数の発行は，API 関数 rxfs_HeapInit の処理完了後に行う必要があります。

注意 2　システム初期化情報 _FX_INIT_PACK についての詳細は「5.4.1　システム初期化情報」を，ドライバ管理構造体 _FX_DRV_MNG についての詳細は「5.4.2　ドライバ管理構造体」を参照してください。

戻り値

FXSE_OK	0x0	**正常終了**
FXSE_MEMORYSMALL	0x3	RX-FS850 **用ヒープ領域が不足しています**
FXSE_SYSTEMINIT	0x70	**既に本** API **関数が発行されています**
FXSE_SEM	0x71	**セマフォの生成処理が失敗しました**
FXSE_MBX	0x72	**メールボックスの生成処理が失敗しました**
FXSE_PARAMETER	0x77	**パラメータの指定が不正です**

rxfs_SystemTerm

概要

RX-FS850 の終了

C 言語形式

```
int                 rxfs_SystemTerm ( void ) ;
```

パラメータ

なし

機能

RX-FS850 の終了処理を実行します。
なお，RX-FS850 では，RX-FS850 の終了処理として，RX-FS850 が提供している機能を実現する際に必要となる各種資源の削除，および，メモリ領域の解放を行っています。
これにより，すべての処理プログラムから RX-FS850 が提供している機能を利用することができなくなります。

戻り値

FXSE_OK	0x0	正常終了
FXSE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXSE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXSE_SEM	0x71	セマフォの削除処理が失敗しました
FXSE_MBX	0x72	メールボックスの削除処理が失敗しました
FXSE_TASKEXIST	0x73	rxfs_TaskInit を発行していないタスクが存在しています
FXSE_NOTUMOUNTED	0x78	rxfs_umount の発行されていないドライブが存在しています

rxfs_TaskInit

概要

タスクの登録

C 言語形式

```
int                 rxfs_TaskInit ( void ) ;
```

パラメータ

なし

機能

本 API 関数を発行したタスクを “ RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を利用するタスク ” として RX-FS850 に登録します。

なお，RX-FS850 では，タスクの登録処理として，RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を利用するタスク毎に必要となる管理領域の獲得，および，初期化を行っています。

注意　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ TaskNum と同数のタスクが既に登録されていた場合には，タスクの登録処理は行わず，戻り値として FXSE_TASKMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

FXSE_OK	0x0	正常終了
FXSE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXSE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXSE_TASKINIT	0x74	既に本 API 関数が発行されています
FXSE_TASKMAX	0x75	管理領域に空きがありません

rxfs_TaskTerm

概要

タスクの登録解除

C 言語形式

```
int                 rxfs_TaskTerm ( void ) ;
```

パラメータ

なし

機能

本 API 関数を発行したタスクを “ RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を利用しないタスク ” として RX-FS850 から登録解除します。

なお，RX-FS850 では，タスクの登録解除処理として，API 関数 rxfs_TaskInit の発行により獲得した管理領域の返却を行っています。

戻り値

FXSE_OK	0x0	正常終了
FXSE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXSE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXSE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXSE_FILEEXIST	0x76	クローズ処理の実行されていないファイルが存在しています

5.5.3　ドライブ制御管理機能

表 5-12 に，RX-FS850 がドライブ制御管理機能として提供している API 関数の一覧を示します。

表 5-12　ドライブ制御管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_mount	ドライブのマウント	○	○
rxfs_umount	ドライブのアンマウント	○	○
rxfs_umountforce	ドライブの強制アンマウント	○	○
rxfs_mountinfo	マウント情報の獲得	○	○
rxfs_flushall	ドライブの強制フラッシュ	○	○
rxfs_format	ドライブのクイック・フォーマット	○	−

rxfs_mount

概要

ドライブのマウント

C 言語形式

int　　rxfs_mount (fx_u8 *drvname* , char **driver* , fx_u8 *u_id* , fx_u8 *p_id* , fx_u8 *mode*) ;

パラメータ

I/O	パラメータ		説　　明
I	fx_u8	*drvname* ;	マウントするドライブのドライブ名 ('A' 〜 'Z')
I	char	**driver* ;	デバイス・ドライバ名を格納した領域へのポインタ
I	fx_u8	*u_id* ;	*driver* で指定されたデバイス・ドライバが管理しているユニット ID
I	fx_u8	*p_id* ;	*driver* で指定されたデバイス・ドライバが管理しているパーティション番号
I	fx_u8	*mode* ;	マウント・モード FX_MNT_ASYNC (0x0)　：非同期モード FX_MNT_SYNC (0x80)　：同期モード FX_MNT_FORMAT (0x40)　：フォーマット・モード

機能

drvname で指定されたドライブを *driver*，*u_id*，*p_id*，*mode* で指定された情報をもとにマウントします。
これにより，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が可能となります。

注意 1　デバイス・ドライバ名 *driver* には，API 関数 rxfs_SystemInit を発行した際に指定したドライバ管理構造体 _FX_DRV_MNG のメンバ DriverName を設定します。
なお，ドライバ管理構造体 _FX_DRV_MNG についての詳細は，「5.4.2　ドライバ管理構造体」を参照してください。

注意 2　RX-FS850 では，マウント・モード *mode* として，以下に示した 3 種類を提供しています。

- FX_MNT_ASYNC (0x0) ：非同期モード

  書き込み要求 (API 関数 rxfs_fwrite，rxfs_write などの発行) が発生した際，ただちに該当ファイルへの書き込みを行わず，いったん，バッファに対して書き込みを行います。
  なお，該当ファイルに対する書き込み処理は，バッファがフラッシュ (API 関数 rxfs_flushall，rxfs_fflush，rxfs_fsync の発行) された際，または，該当ファイルがクローズ (API 関数 rxfs_fclose，rxfs_close の発行) された際に行われます。

- FX_MNT_SYNC (0x80) ：同期モード

  書き込み要求 (API 関数 rxfs_fwrite，rxfs_write などの発行) が発生した際，ただちに該当ファイルへの書き込みを行います。

- FX_MNT_FORMAT (0x40) ：**フォーマット・モード**

  drvname **で指定された** FAT **ファイル・システムのドライブをクイック・フォーマットする際の専用モードです。したがって，本モードでマウントされたドライブに対する操作は，フォーマット処理** (API **関数** rxfs_format **の発行**)**，または，アンマウント処理** (API **関数** rxfs_umount**，**rxfs_umountforce **の発行**) **に限定されます。**

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	**正常終了**
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit **が発行されていません**
FXE_TASKNOTINIT	0x2	rxfs_TaskInit **が発行されていません**
FXE_MEMORYSMALL	0x3	RX-FS850 **用ヒープ領域が不足しています**
FXE_MOUNTED	0x10	**既に本** API **関数が発行されています**
FXE_DRVNOTSUPPORT	0x15	**ドライバ・タイプが不正です**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_ILLMODE	0x42	**マウント・モードの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_umount

概要

ドライブのアンマウント

C 言語形式

int　　rxfs_umount (fx_u8 *drvname*) ;

パラメータ

I/O	パラメータ	説　　明
I	fx_u8　　*drvname* ;	アンマウントするドライブのドライブ名 ('A' ～ 'Z')

機能

drvname で指定されたドライブをアンマウントします。
したがって，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が禁止されます。

注意　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DRIVEBUSY	0x12	該当ドライブに対する操作が行われています
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_umountforce

概要

ドライブの強制アンマウント

C 言語形式

int　　rxfs_umountforce (fx_u8 *drvname*) ;

パラメータ

I/O	パラメータ	説　　明
I	fx_u8　　*drvname* ;	アンマウントするドライブのドライブ名 ('A' 〜 'Z')

機能

drvname で指定されたドライブを強制的にアンマウントします。
したがって，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が禁止されます。

注意 1　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。

注意 2　本 API 関数では，*drvname* で指定されたドライブに対する操作が行われている最中であっても，該当ドライブのアンマウント処理を強制的に実行します。したがって，ファイルが“ 書き込み可能なモード ”でオープンされていた際には，該当ファイルに対する書き込み要求が反映されない可能性があります。

注意 3　RX-FS850 では，該当ドライブに一時ディレクトリが設定されていた場合，一時ディレクトリの設定解除処理を実行します。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

rxfs_mountinfo

概要

マウント情報の獲得

C 言語形式

int　　rxfs_mountinfo (fx_u8 *drvname* , char **driver* , fx_u8 **u_id* , fx_u8 **p_id* , fx_u8 **mode*) ;

パラメータ

I/O	パラメータ		説　　明
I	fx_u8	*drvname* ;	マウント情報を獲得するドライブのドライブ名 ('A' 〜 'Z')
O	char	**driver* ;	デバイス・ドライバ名を格納する領域へのポインタ
O	fx_u8	**u_id* ;	ユニット ID を格納する領域へのポインタ
O	fx_u8	**p_id* ;	パーティション番号を格納する領域へのポインタ
O	fx_u8	**mode* ;	マウント・モードを格納する領域へのポインタ

機能

drvname で指定されたドライブのマウント情報 (デバイス・ドライバ名，ユニット ID，パーティション番号，マウント・モード) を *driver*，*u_id*，*p_id*，*mode* で指定された領域に格納します。

注意 1　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。

注意 2　*driver* で指定されたデバイス・ドライバ名を格納する領域には，9 バイトの大きさが必要となります。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

rxfs_flushall

概要

ドライブの強制フラッシュ

C 言語形式

int　　rxfs_flushall (fx_u8 *drvname*) ;

パラメータ

I/O	パラメータ	説　　明
I	fx_u8　　*drvname* ;	フラッシュするドライブのドライブ名 ('A' 〜 'Z')

機能

drvname で指定されたドライブにおいて，書き込み要求 (API 関数 rxfs_fwrite，rxfs_fputc などの発行) が行われているファイルのバッファを強制的にフラッシュします。
これにより，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。
なお，本 API 関数では，フラッシュ処理が完了した際には，該当ドライブをアクセス禁止状態へと遷移させています。
したがって，本 API 関数の発行後，該当ドライブに対する操作 (API 関数 rxfs_fopen，rxfs_open などの発行) が禁止されます。

注意 1　ドライブ名 *drvname* には，API 関数 rxfs_mount を発行した際に指定したドライブ名を設定します。

注意 2　該当ドライブが CD-ROM ファイル・システムの際には，フラッシュ処理は実行されず，状態遷移処理 (該当ドライブをアクセス禁止状態へと遷移) のみが実行されます。

注意 3　本 API 関数では，API 関数 rxfs_tmpfile の発行により生成 / オープンされた一時ファイルのクローズ処理，および，削除処理は行われません。
したがって，該当ドライブの一時ファイルに対するクローズ処理，および，削除処理 (API 関数 rxfs_fclose の発行) は，本 API 関数を発行する以前に行う必要があります。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です

FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_format

概要

ドライブのクイック・フォーマット

C 言語形式

int　　rxfs_format (fx_u8 *drvname*) ;

パラメータ

I/O	パラメータ	説　　明
I	fx_u8　　*drvname* ;	クイック･フォーマットするドライブのドライブ名 ('A' 〜 'Z')

機能

drvname で指定されたドライブをクイック・フォーマット (フォーマット済みドライブの再フォーマット) します。

注意 1　デバイス・ドライバ名 *driver* には，API 関数 rxfs_SystemInit を発行した際に指定したドライバ管理構造体 _FX_DRV_MNG のメンバ DriverName を設定します。
なお，ドライバ管理構造体 _FX_DRV_MNG についての詳細は，「5.4.2　ドライバ管理構造体」を参照してください。

注意 2　本 API 関数でフォーマット可能なドライブは，フォーマット・モード FX_MNT_FORMAT でマウントされたドライブに限定されます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブが同期モード，または，非同期モードでマウントされています
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

5.5.4　ストリーム入出力管理機能

表 5-13 に，RX-FS850 がストリーム入出力管理機能として提供している API 関数の一覧を示します。

表 5-13　ストリーム入出力管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_fopen	ファイルのオープン	○	○
rxfs_tmpfile	一時ファイルの生成 / オープン	○	−
rxfs_fclose	ファイルのクローズ	○	○
rxfs_fflush	バッファのフラッシュ	○	−
rxfs_fread	データの読み込み	○	○
rxfs_fwrite	データの書き込み	○	−
rxfs_fgetc	文字の読み込み	○	○
rxfs_ungetc	文字の返却	○	○
rxfs_fputc	文字の書き込み	○	−
rxfs_fgets	文字列の読み込み	○	○
rxfs_fputs	文字列の書き込み	○	−
rxfs_fseek	ファイル・ポインタの位置変更 (指定された位置に変更)	○	○
rxfs_rewind	ファイル・ポインタの位置変更 (ファイルの先頭に変更)	○	○
rxfs_ftell	ファイル・ポインタの位置獲得	○	○
rxfs_feof	EOF フラグの状態獲得	○	○
rxfs_ferror	エラー・フラグの状態獲得	○	○
rxfs_clearerr	EOF フラグ，および，エラー・フラグのクリア	○	○

rxfs_fopen

概要

ファイルのオープン

C 言語形式

FX_FILE　　*rxfs_fopen (char **filename* , char **mode*) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　**filename* ;	オープンするファイルのファイル名を格納した領域へのポインタ
I	char　　**mode* ;	オープンするファイルのストリーム・モードを格納した領域へのポインタ

機能

filename で指定されたファイルを *mode* で指定されたストリーム・モードでオープンします。
なお，RX-FS850 では，ファイルのオープン処理として，API 関数 (rxfs_fclose，rxfs_fflush など) を発行する際，オープンしたファイル毎に必要となるストリームの獲得，および，初期化を行っています。
また，本 API 関数の処理が正常終了した際には，“ オープンしたファイルのストリームへのポインタ ” が戻り値として返されます。
以下に，*mode* に指定可能なキー・ワード一覧を示します。

r　：読み込み専用
ファイルの先頭からデータの読み込みを行います。
なお，*filename* で指定されたファイルが存在しない場合，異常終了 FXE_FILENOTEXIST となります。

w　：書き込み専用
ファイルの先頭からデータの書き込みを行います。
なお，パラメータ *filename* で指定されたファイルが存在した場合には“ 該当ファイルの削除処理，および，該当ファイルの新規生成処理 ”が，パラメータ *filename* で指定されたファイルが存在しない場合には“ 該当ファイルの新規生成処理 ” があわせて行われます。

a　：書き込み専用
ファイルの末尾からデータの書き込みを行います。
なお，パラメータ *filename* で指定されたファイルが存在しない場合には “ 該当ファイルの新規生成処理 ” があわせて行われます。

r+　：読み込み / 書き込み両用
ファイルの先頭からデータの読み込み / 書き込みを行います。
なお，*filename* で指定されたファイルが存在しない場合，異常終了 FXE_FILENOTEXIST となります。

w+　：読み込み / 書き込み両用
ファイルの先頭からデータの読み込み / 書き込みを行います。
なお，パラメータ *filename* で指定されたファイルが存在した場合には“ 該当ファイルの削除処理，および，該当ファイルの新規生成処理 ”が，パラメータ *filename* で指定されたファイルが存在しない場合には“ 該当ファイルの新規生成処理 ” があわせて行われます。

a+　：読み込み / 書き込み両用
ファイルの先頭からデータの読み込みを，ファイルの末尾からデータの書き込みを行います。
なお，パラメータ *filename* で指定されたファイルが存在しない場合には “ 該当ファイルの新規生成処理 ” があわせて行われます。

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体，または，FstmNum と同数のストリームが既に使用されていた場合には，ファイルのオープン処理は行わず，エラー・コードとして　FXE_FILEMAX，または，FXE_OPENMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	オープンしたファイルのストリームへのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_OPENMAX	0x5	ストリームに空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DISKFULL	0x13	デバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_DIR	0x25	操作対象にディレクトリを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です
FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_DUPWRITE	0x33	既に該当ファイルは書き込み専用，または，読み込み / 書き込み両用でオープンされています
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_ILLMODE	0x42	ストリーム・モードの指定が不正です

FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_tmpfile

概要

一時ファイルの生成 / オープン

C 言語形式

```
FX_FILE          *rxfs_tmpfile ( void ) ;
```

パラメータ

なし

機能

一時ファイルを生成したのち，読み込み / 書き込み両用でオープンします。
なお，本 API 関数の処理が正常終了した際には，“ オープンした一時ファイルのストリームへのポインタ ” が戻り値として返されます。

注意 1　RX-FS850 では，API 関数 rxfs_settmpdir の発行により設定された一時ディレクトリに対して，一時ファイルの生成処理を行います。

注意 2　RX-FS850 では，本 API 関数の発行により生成 / オープンされた一時ファイルに対して API 関数 rxfs_fclose が発行された際には，該当ファイルのクローズ処理を実行した後，該当ファイルの削除処理もあわせて行っています。

戻り値

正常終了	オープンした一時ファイルのストリームへのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_OPENMAX	0x5	ストリームに空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DISKFULL	0x13	デバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILEEXIST	0x21	既に該当ファイルが存在しています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています

FXE_TMPNAMMAX	0x43	**一時ファイルの生成数が最大生成数を超えています**
FXE_NOSETTMPDIR	0x44	**一時ディレクトリの設定が行われていません**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_fclose

概要

ファイルのクローズ

C 言語形式

int　　rxfs_fclose (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	クローズするファイルのストリームへのポインタ

機能

stream で指定されたファイルをクローズします。
なお，RX-FS850 では，ファイルのクローズ処理として，API 関数 rxfs_fopen，または，rxfs_tmpfile の発行により獲得したストリームの返却を行っています。
したがって，本 API 関数の発行後，該当ファイルに対する操作 (API 関数 rxfs_fflush，rxfs_fread などの発行) が禁止されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　*stream* で指定されたファイルが“ 書き込み可能なモード ”でオープンされていた場合，RX-FS850 は該当バッファをフラッシュしたのち，ファイルのクローズ処理を実行します。
ただし，該当ファイルのドライブが API 関数 rxfs_umountforce の発行によりアンマウントされていた際，および，API 関数 rxfs_flushall の発行によりアクセス禁止状態へと遷移していた際には，フラッシュ処理は実行されず，クローズ処理のみが実行されます。

注意 3　RX-FS850 では，該当ファイルが API 関数 rxfs_tmpfile の発行によりオープンされていた場合，該当ファイルのクローズ処理を実行した後，該当ファイルの削除処理もあわせて行っています。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLSTREAM	0x34	ストリームの状態が不正です

FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_fflush

概要

バッファのフラッシュ

C 言語形式

int　　rxfs_fflush (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	バッファをフラッシュするファイルのストリームへのポインタ

機能

stream で指定されたファイルのバッファをフラッシュします。
これにより，書き込み要求 (API 関数 rxfs_fwrite，rxfs_fputc などの発行) を行った際，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_READOPEN	0x31	該当ファイルは読み込み専用でオープンされています
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_fread

概要

データの読み込み

C 言語形式

long　　rxfs_fread (void *_ptr_ , long _size_ , long _number_ , FX_FILE *_stream_) ;

パラメータ

I/O	パラメータ		説　　明
O	void	*_ptr_ ;	読み込んだデータを格納する領域へのポインタ
I	long	_size_ ;	1 個当たりのデータの読み込みサイズ (単位：バイト)
I	long	_number_ ;	データの読み込み個数
I	FX_FILE	*_stream_ ;	データを読み込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から _size_ * _number_ で算出されたサイズのデータ，または，ファイルの末尾までのデータを _ptr_ で指定された領域に格納します。
なお，本 API 関数の処理が正常終了した際には，“ データの読み込み個数 ” が戻り値として返されます。

注意 1　ストリーム _stream_ には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの読み込み処理が完了した際には，ファイル・ポインタの位置を本 API 関数発行以前の位置から “ 読み込んだデータの合計サイズ (単位：バイト) ” だけ進めています。

注意 3　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

戻り値

正常終了	データの読み込み個数
異常終了	0x0

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています

FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_WRITEOPEN	0x32	**該当ファイルは書き込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_FINDEOF	0x35	**ファイルの末尾まで読み込みました**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_fwrite

概要

データの書き込み

C 言語形式

long　　rxfs_fwrite (void *_ptr_ , long _size_ , long _number_ , FX_FILE *_stream_) ;

パラメータ

I/O	パラメータ		説　　明
I	void	*_ptr_ ;	書き込むデータを格納した領域へのポインタ
I	long	_size_ ;	1 個当たりのデータの書き込みサイズ (単位：バイト)
I	long	_number_ ;	データの書き込み個数
I	FX_FILE	*_stream_ ;	データを書き込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から _size_ * _number_ で算出されたサイズのデータを書き込みます。
なお，_ptr_ で指定された領域には，該当ファイルに書き込むデータを設定します。
また，本 API 関数の処理が正常終了した際には，“ データの書き込み個数 ” が戻り値として返されます。

注意 1　ストリーム _stream_ には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの書き込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 書き込んだデータの合計サイズ (単位：バイト) ” だけ進めています。

戻り値

正常終了	データの書き込み個数
異常終了	0x0

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DISKFULL	0x13	該当ファイルのデバイスに空き領域がありません

FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_READOPEN	0x31	**該当ファイルは読み込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_fgetc

概要

文字の読み込み

C 言語形式

int　　rxfs_fgetc (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	文字を読み込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から文字 (1 文字) を読み込みます。
なお，本 API 関数の処理が正常終了した際には，“ 読み込んだ文字 (1 文字) を int 型に拡張した値 ” が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだ文字のサイズ (0x1 バイト) ” だけ進めています。

注意 3　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

戻り値

正常終了	読み込んだ文字 (1 文字) を int 型に拡張した値
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_WRITEOPEN	0x32	該当ファイルは書き込み専用でオープンされています
FXE_ILLSTREAM	0x34	ストリームの状態が不正です

FXE_FINDEOF	0x35	**ファイルの末尾まで読み込みました**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_ungetc

概要

文字の返却

C 言語形式

int　　rxfs_ungetc (int *character* , FX_FILE **stream*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*character* ;	返却する文字を int 型に拡張した値
I	FX_FILE　　**stream* ;	文字を返却するファイルのストリームへのポインタ

機能

stream で指定されたファイルのストリームが確保している返却用バッファに *character* で指定された文字 (1 文字) を格納します。
なお,本API関数の処理が正常終了した際には,“ 返却した文字(1文字)をint型に拡張した値 ”が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　返却用バッファに格納された文字 *character* は，API 関数 rxfs_fflush，rxfs_fgetc，rxfs_fseek などが発行された際に返却用バッファからクリアされます。
なお，RX-FS850 では，返却用バッファがクリアされることなく本 API 関数を再発行した際には，戻り値として FXE_ALREADYUNGET を返しています。

注意 3　本 API 関数における返却処理は，返却用バッファに対してのみ行われます。したがって，*stream* で指定されたファイルに対する文字 *character* の書き込みは行われません。

戻り値

正常終了	返却した文字 (1 文字) を int 型に拡張した値
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です

FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_WRITEOPEN	0x32	**該当ファイルは書き込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_ALREADYUNGET	0x45	**既に本** API **関数が発行されています**
FXE_NOUNGET	0x46	**ファイル・ポインタが該当ファイルの先頭を指しています**
FXE_EOFUNGET	0x47	**返却する文字に** EOF **を指定しています**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**

rxfs_fputc

概要

文字の書き込み

C 言語形式

int　　rxfs_fputc (int *character* , FX_FILE **stream*) ;

パラメータ

I/O	パラメータ		説　　明
I	int	*character* ;	書き込む文字を int 型に拡張した値
I	FX_FILE	**stream* ;	文字を書き込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から *character* で指定された文字 (1 文字) を書き込みます。
なお，*character* で指定された領域には，該当ファイルに書き込む文字を設定します。
また，本 API 関数の処理が正常終了した際には，“ 書き込んだ文字 (1 文字) を int 型に拡張した値 ” が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の書き込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 書き込んだ文字のサイズ (0x1 バイト) ” だけ進めています。

戻り値

正常終了	書き込んだ文字 (1 文字) を int 型に拡張した値
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DISKFULL	0x13	該当ファイルのデバイスに空き領域がありません
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープンされていません

FXE_READOPEN	0x31	**該当ファイルは読み込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_fgets

概要

文字列の読み込み

C 言語形式

```
char        *rxfs_fgets ( char *string , int number , FX_FILE *stream ) ;
```

パラメータ

I/O	パラメータ		説　　明
O	char	*string ;	読み込んだ文字列を格納する領域へのポインタ
I	int	number ;	最大読み込み文字数 + 1
I	FX_FILE	*stream ;	文字列を読み込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から *number* - 1 で算出された数の文字列，または，改行文字 LF まで，または，ファイルの末尾までの文字列を *string* で指定された領域に格納します。

なお，本 API 関数の処理が正常終了した際には，“ 読み込んだ文字列を格納した領域へのポインタ ” が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだ文字の合計サイズ (単位：バイト) ” だけ進めています。

注意 3　RX-FS850 では，文字の読み込み処理が “ 改行文字 LF の検出 ” により完了した際には，*string* で指定された領域に “ 改行文字 LF を含まない文字列 ” を格納してます。

注意 4　EOF フラグが “ EOF 検出 ” のファイルに追加書き込みされたデータは，EOF フラグのクリア処理 (API 関数 rxfs_clearerr などの発行) を実行した後に読み込む必要があります。

戻り値

正常終了	読み込んだ文字列を格納した領域へのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません

FXE_NOTMOUNT	0x11	**既に操作対象ドライブがアンマウントされています**
FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_WRITEOPEN	0x32	**該当ファイルは書き込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_FINDEOF	0x35	**ファイルの末尾まで読み込みました**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_fputs

概要

文字列の書き込み

C 言語形式

int　　rxfs_fputs (char *_string_ , FX_FILE *_stream_) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*_string_ ;	書き込む文字列を格納した領域へのポインタ
I	FX_FILE　　*_stream_ ;	文字列を書き込むファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置から _string_ で指定された文字列を書き込みます。
なお，_string_ で指定された領域には，該当ファイルに書き込む文字列を設定します。

注意 1　ストリーム _stream_ には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，文字の書き込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 書き込んだ文字の合計サイズ (単位：バイト) ” だけ進めています。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DISKFULL	0x13	該当ファイルのデバイスに空き領域がありません
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_READOPEN	0x31	該当ファイルは読み込み専用でオープンされています
FXE_ILLSTREAM	0x34	ストリームの状態が不正です

FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_fseek

概要

ファイル・ポインタの位置変更 (指定された位置に変更)

C 言語形式

int　　rxfs_fseek (FX_FILE *_stream_ , long _offset_ , int _whence_) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　*_stream_ ;	ファイル・ポインタの位置を変更するファイルのストリームへのポインタ
I	long　_offset_ ;	_whence_ で指定された基準点からのオフセット値 (単位：バイト)
I	int　_whence_ ;	基準点 FX_SEEK_SET　(0x0)　：ファイルの先頭 FX_SEEK_CUR　(0x1)　：現在の位置 FX_SEEK_END　(0x2)　：ファイルの末尾

機能

stream で指定されたファイルのファイル・ポインタの位置を，本 API 関数発行時の位置から _offset_，および，_whence_ で指定された位置に変更します。

注意 1　ストリーム _stream_ には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　本 API 関数が正常終了した際には，RX-FS850 は _stream_ で指定されたファイルの EOF フラグ，および，エラー・フラグのクリア処理を実行します。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています

FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_rewind

概要

ファイル・ポインタの位置変更 (ファイルの先頭に変更)

C 言語形式

void　　rxfs_rewind (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	ファイル・ポインタの位置を変更するファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置を，本 API 関数発行時の位置からファイルの先頭に変更します。
したがって，本 API 関数は，API 関数 rxfs_fseek を以下のように発行した場合と同等と意味を持ちます。

(void) rxfs_fseek (*stream* , 0x0 , FX_SEEK_SET) ;

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　本 API 関数が正常終了した際には，RX-FS850 は *stream* で指定されたファイルの EOF フラグ，および，エラー・フラグのクリア処理を実行します。

戻り値

なし

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_ftell

概要

ファイル・ポインタの位置獲得

C 言語形式

long　　rxfs_ftell (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	ファイル・ポインタの位置を獲得するファイルのストリームへのポインタ

機能

stream で指定されたファイルのファイル・ポインタの位置を獲得します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得したファイル・ポインタの位置 ” が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。
注意 2　獲得したファイル・ポインタの位置は，該当ファイルの先頭からのオフセット値 (単位：バイト) となります。

戻り値

正常終了	獲得したファイル・ポインタの位置
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープンされていません
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

rxfs_feof

概要

EOF フラグの状態獲得

C 言語形式

int　　rxfs_feof (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	EOF フラグの状態を獲得するファイルのストリームへのポインタ

機能

stream で指定されたファイルの EOF フラグの状態を獲得します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得した EOF フラグの状態 ” が戻り値として返されます。

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

戻り値

EOF 検出	0x0 以外
EOF 未検出	0x0

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

rxfs_ferror

概要

エラー・フラグの状態獲得

C 言語形式

int　　rxfs_ferror (FX_FILE *_stream_) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*_stream_ ;	エラー・フラグの状態を獲得するファイルのストリームへのポインタ

機能

stream で指定されたファイルのエラー・フラグの状態を獲得します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得したエラー・フラグの状態 ” が戻り値として返されます。

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

戻り値

エラー検出	0x0 以外
エラー未検出	0x0

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

rxfs_clearerr

概要

EOF フラグ，および，エラー・フラグのクリア

C 言語形式

void　　rxfs_clearerr (FX_FILE *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_FILE　　*stream ;	EOF フラグ，および，エラー・フラグをクリアするファイルのストリームへのポインタ

機能

stream で指定されたファイルの EOF フラグ，および，エラー・フラグをクリアします。

注意　ストリーム *stream* には，API 関数 rxfs_fopen，または，rxfs_tmpfile の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

戻り値

なし

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました

5.5.5　低水準入出力管理機能

表 5-14 に，RX-FS850 が低水準入出力管理機能として提供している API 関数の一覧を示します。

表 5-14　低水準入出力管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_open	ファイルのオープン	○	○
rxfs_close	ファイルのクローズ	○	○
rxfs_fsync	バッファのフラッシュ	○	−
rxfs_read	データの読み込み	○	○
rxfs_write	データの書き込み	○	−
rxfs_lseek	ファイル・ポインタの位置変更 (指定された位置に変更)	○	○

rxfs_open

概要

ファイルのオープン

C 言語形式

int　　rxfs_open (char *_filename_ , int _mode_) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*_filename_ ;	オープンするファイルのファイル名を格納した領域へのポインタ
I	int　　_mode_ ;	オープンするファイルのモード FX_O_RDONLY　(0x0)　：読み込み専用 FX_O_WRONLY　(0x1)　：書き込み専用 FX_O_RDWR　(0x2)　：読み込み / 書き込み両用 FX_O_APPEND　(0x8)　：データの追加 FX_O_CREATE　(0x200)　：ファイルの新規生成 FX_O_TRUNC　(0x400)　：データの上書き FX_O_EXCL　(0x800)　：ファイル存在時エラー

機能

filename で指定されたファイルを _mode_ で指定されたモードでオープンします。
なお，RX-FS850 では，ファイルのオープン処理として，API 関数 (rxfs_close，rxfs_fsync など) を発行する際，オープンしたファイル毎に必要となるファイル記述子の獲得，および，初期化を行っています。
また,本 API 関数の処理が正常終了した際には,“ オープンしたファイルのファイル記述子 ”が戻り値として返されます。
以下に，_mode_ の代表的な指定形式を示します。

- FX_O_RDONLY

  filename で指定された既存ファイルを読み込み専用でオープンします。
  なお，データの読み込み開始位置は，ファイル・ポインタで指定された位置からとなります。

- FX_O_WRONLY | FX_O_APPEND

  filename で指定された既存ファイルを書き込み専用でオープンします。
  なお，データの書き込み開始位置は，ファイルの末尾からとなります。

- FX_O_WRONLY | FX_O_CREATE

  filename で指定された既存ファイルが存在しなかった場合には，該当ファイルを新規生成したのち，書き込み専用でオープンします。
  なお，データの書き込み開始位置は，ファイル・ポインタで指定された位置からとなります。

- FX_O_WRONLY | FX_O_TRUNC

  filename で指定された既存ファイルを書き込み専用でオープンします。
  なお，データの書き込み開始位置は，ファイルの先頭からとなります。

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　オープンするファイルのモード *mode* については，同時に指定することが禁止されているもの，組み合わせて指定するものがあります。

- FX_O_RDONLY，FX_O_WRONLY，FX_O_RDWR は，同時に指定することが禁止
- FX_O_APPEND は，FX_O_WRONLY，または，FX_O_RDWR と組み合わせて指定
- FX_O_CREATE は，FX_O_WRONLY，または，FX_O_RDWR と組み合わせて指定
- FX_O_TRUNC は，FX_O_WRONLY，または，FX_O_RDWR と組み合わせて指定
- FX_O_EXCL は，FX_O_CREATE と組み合わせて指定

注意 4　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体，または，FstmNum と同数のストリームが既に使用されていた場合には，ファイルのオープン処理は行わず，エラー・コードとして FXE_FILEMAX，または，FXE_OPENMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	オープンしたファイルのファイル記述子
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_OPENMAX	0x5	ストリームに空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DISKFULL	0x13	デバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_FILEEXIST	0x21	該当ファイルが存在しています
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_DIR	0x25	操作対象にディレクトリを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です

FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_DUPWRITE	0x33	既に該当ファイルは書き込み専用，または，読み込み / 書き込み両用でオープンされています
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_ILLMODE	0x42	モードの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_close

概要

ファイルのクローズ

C 言語形式

int　　rxfs_close (int *fildes*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*fildes* ;	クローズするファイルのファイル記述子

機能

fildes で指定されたファイルをクローズします。
なお，RX-FS850 では，ファイルのクローズ処理として，API 関数 rxfs_open の発行により獲得したファイル記述子の返却を行っています。
したがって,本API関数の発行後,該当ファイルに対する操作(API関数rxfs_fsync,rxfs_readなどの発行)が禁止されます。

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　*fildes* で指定されたファイルが “ 書き込み可能なモード ” でオープンされていた場合，RX-FS850 は該当バッファをフラッシュしたのち，ファイルのクローズ処理を実行します。
ただし，該当ファイルのドライブが API 関数 rxfs_umountforce の発行によりアンマウントされていた際，および，API 関数 rxfs_flushall の発行によりアクセス禁止状態へと遷移していた際には，フラッシュ処理は実行されず，クローズ処理のみが実行されます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_fsync

概要

バッファのフラッシュ

C 言語形式

int　　rxfs_fsync (int *fildes*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*fildes* ;	バッファをフラッシュするファイルのファイル記述子

機能

fildes で指定されたファイルのバッファをフラッシュします。
これにより，書き込み要求 (API 関数 rxfs_write などの発行) を行った際，一時的にバッファへと書き込まれていたデータのすべてが該当ファイルに書き込まれます。

注意　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_READOPEN	0x31	該当ファイルは読み込み専用でオープンされています
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_read

概要

データの読み込み

C 言語形式

long　　rxfs_read (int *fildes* , void **ptr* , long *size*) ;

パラメータ

I/O	パラメータ		説　　明
I	int	*fildes* ;	データを読み込むファイルのファイル記述子
O	void	**ptr* ;	読み込んだデータを格納する領域へのポインタ
I	long	*size* ;	読み込むデータのサイズ (単位：バイト)

機能

fildes で指定されたファイルのファイル・ポインタの位置から *size* で指定されたサイズのデータ，または，ファイルの末尾までのデータを *ptr* で指定された領域に格納します。

なお,本API関数の処理が正常終了した際には,“ 読み込んだデータのサイズ(単位:バイト) ”が戻り値として返されます。

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの読み込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 読み込んだデータのサイズ (単位：バイト) ” だけ進めています。

戻り値

正常終了	読み込んだデータのサイズ (単位：バイト)
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープされていません
FXE_WRITEOPEN	0x32	該当ファイルは書き込み専用でオープンされています

FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_FINDEOF	0x35	**ファイルの末尾まで読み込みました**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_write

概要

データの書き込み

C 言語形式

long　　rxfs_write (int *fildes* , void **ptr* , long *size*) ;

パラメータ

I/O	パラメータ		説　　明
I	int	*fildes* ;	データを書き込むファイルのファイル記述子
I	void	**ptr* ;	書き込むデータを格納した領域へのポインタ
I	long	*size* ;	データの書き込みサイズ (単位：バイト)

機能

fildes で指定されたファイルのファイル・ポインタの位置から *size* で指定されたサイズのデータを書き込みます。
なお，*ptr* で指定された領域には，該当ファイルに書き込むデータを設定します。
また,本API関数の処理が正常終了した際には,“ 書き込んだデータのサイズ(単位:バイト) ”が戻り値として返されます。

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　RX-FS850 では，データの書き込み処理が完了した際には，ファイル・ポインタの位置を，本 API 関数発行以前の位置から “ 書き込んだデータのサイズ (単位：バイト) ” だけ進めています。

戻り値

正常終了	書き込んだデータのサイズ (単位：バイト)
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DISKFULL	0x13	該当ファイルのデバイスに空き領域がありません
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ファイルがオープンされていません

FXE_READOPEN	0x31	**該当ファイルは読み込み専用でオープンされています**
FXE_ILLSTREAM	0x34	**ストリームの状態が不正です**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_lseek

概要

ファイル・ポインタの位置変更 (指定された位置に変更)

C 言語形式

long　　rxfs_lseek (int *fildes* , long *offset* , int *whence*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　*fildes* ;	ファイル・ポインタの位置を変更するファイルのファイル記述子
I	long　*offset* ;	*whence* で指定された基準点からのオフセット値 (単位：バイト)
I	int　*whence* ;	基準点 FX_SEEK_SET　(0x0)　：ファイルの先頭 FX_SEEK_CUR　(0x1)　：現在の位置 FX_SEEK_END　(0x2)　：ファイルの末尾

機能

fildes で指定されたファイルのファイル・ポインタの位置を，本 API 関数発行時の位置から *offset*，および，*whence* で指定された位置に変更します。
なお，本 API 関数の処理が正常終了した際には，“ 変更後のファイル・ポインタの位置 ” が戻り値として返されます。

注意 1　ファイル記述子 *fildes* には，API 関数 rxfs_open の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたファイル記述子の管理を行っているため，本 API 関数の操作対象ファイルは，同一タスク内でオープン処理が行われたファイルに限定されます。

注意 2　変更後のファイル・ポインタの位置は，該当ファイルの先頭からのオフセット値 (単位：バイト) となります。

戻り値

正常終了	変更後のファイル・ポインタの位置
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています

FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_NOOPEN	0x30	**該当ファイルがオープされていません**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

5.5.6　ファイル・アクセス管理機能

表 5-15 に，RX-FS850 がファイル・アクセス管理機能として提供している API 関数の一覧を示します。

表 5-15　ファイル・アクセス管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_rename	ファイル名の変更	○	−
rxfs_remove	ファイルの削除	○	−
rxfs_unlink	ファイルの削除	○	−
rxfs_stat	ステータス情報の獲得	○	○
rxfs_chstat	ステータス情報の変更	○	−
rxfs_settmpdir	一時ディレクトリの設定	○	−
rxfs_tmpnam	一時ファイル名の生成	○	−

rxfs_rename

概要

ファイル名の変更

C 言語形式

int　　rxfs_rename (char *oldname , char *newname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*oldname ;	変更前のファイル名を格納した領域へのポインタ
I	char　　*newname ;	変更後のファイル名を格納した領域へのポインタ

機能

oldname で指定されたファイルのファイル名を *newname* で指定されたファイル名に変更します。

注意 1　ファイル名 *oldname*，*newname* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*oldname*，*newname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*oldname*，*newname* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，*oldname*，*newname* に異なるパス名が指定されていた場合には，ファイル名の変更処理は行わず，ファイルの移動処理を行います。

注意 4　本 API 関数では，*oldname*，*newname* に異なるドライブ名が指定されていた場合には，ファイル名の変更処理は行わず，戻り値として FXE_CROSSDRIVE を返しています。

注意 5　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ファイル名の変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了

FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DISKFULL	0x13	デバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_FILEEXIST	0x21	既に該当ファイルが存在しています
FXE_FILEBUSY	0x22	該当ファイルがオープンされています
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です
FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_CROSSDRIVE	0x48	ドライブの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_remove

概要

ファイルの削除

C 言語形式

int　　rxfs_remove (char *filename) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*filename ;	削除するファイルのファイル名を格納した領域へのポインタ

機能

filename で指定されたファイルを削除します。
したがって，本 API 関数は，API 関数 rxfs_unlink を以下のように発行した場合と同等と意味を持ちます。

rxfs_unlink (*filename*) ;

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ファイルの削除処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません

FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_FILEBUSY	0x22	該当ファイルがオープンされています
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_DIR	0x25	パラメータ *filename* にディレクトリを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です
FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_unlink

概要

ファイルの削除

C 言語形式

int　　rxfs_unlink (char *_filename_) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*_filename_ ;	削除するファイルのファイル名を格納した領域へのポインタ

機能

filename で指定されたファイルを削除します。
したがって，本 API 関数は，API 関数 rxfs_remove を以下のように発行した場合と同等と意味を持ちます。

rxfs_remove (_filename_) ;

注意 1　ファイル名 _filename_ には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，_filename_ に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，_filename_ で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ファイルの削除処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません

FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ファイルが存在しません
FXE_FILEBUSY	0x22	該当ファイルがオープンされています
FXE_READONLY	0x23	該当ファイルは読み込み専用です
FXE_DIR	0x25	パラメータ *filename* にディレクトリを指定しています
FXE_ILLPATH	0x27	絶対パスの指定が不正です
FXE_BADNAME	0x28	ファイル名の指定が不正です
FXE_NAMETOOLONG	0x29	ファイル名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_stat

概要

ステータス情報の獲得

C 言語形式

```
int         rxfs_stat ( char *filename , FX_STAT *pStat ) ;
```

パラメータ

I/O	パラメータ	説　明
I	char *filename ;	ステータス情報を獲得するファイルのファイル名を格納した領域へのポインタ
O	FX_STAT *pStat ;	ステータス情報を格納するパケットへのポインタ

【 ステータス情報 _FX_STAT の構造 】

```
struct      _FX_STAT {
            fx_u32      st_size ;     /* ファイルのサイズ ( 単位：バイト ) */
            fx_u32      st_atime ;    /* ファイルのアクセス日時 */
            fx_u32      st_mtime ;    /* ファイルの更新日時 */
            fx_u32      st_ctime ;    /* ファイルの作成日時 */
            fx_u8       st_cmsec ;    /* ファイルの作成日時 ( 単位：10 ミリ秒 )*/
            fx_u8       RFU1 ;        /* システム予約領域 */
            fx_u16      st_mode ;     /* ファイルの属性 */
            fx_u16      RFU2 ;        /* システム予約領域 */
} ;
```

機能

filename で指定されたファイルのステータス情報 (サイズ，アクセス日時，更新日時，作成日時など) を *pStat* で指定されたパケットに格納します。

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ステータス情報の獲得処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 4　ステータス情報 _FX_STAT についての詳細は，「5.4.3　ステータス情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	**正常終了**
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit **が発行されていません**
FXE_TASKNOTINIT	0x2	rxfs_TaskInit **が発行されていません**
FXE_MEMORYSMALL	0x3	RX-FS850 **用ヒープ領域が不足しています**
FXE_FILEMAX	0x4	**ファイル管理構造体に空きがありません**
FXE_NOTMOUNT	0x11	rxfs_mount **が発行されていません**
FXE_ILLMOUNTMODE	0x16	**操作対象ドライブがフォーマット・モードでマウントされています**
FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_FILENOTEXIST	0x20	**該当ファイルが存在しません**
FXE_ILLPATH	0x27	**絶対パス，または，相対パスの指定が不正です**
FXE_BADNAME	0x28	**ファイル名の指定が不正です**
FXE_NAMETOOLONG	0x29	**ファイル名が制限文字数** (FAT：254 **文字**，CD-ROM：64 **文字**) **を超えています**
FXE_PATHTOOLONG	0x2a	**絶対パスが制限文字数** (270 **文字**) **を超えています**
FXE_NOSETWD	0x2b	**カレント・ディレクトリの設定が行われていません**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

rxfs_chstat

概要

ステータス情報の変更

C 言語形式

```
int         rxfs_chstat ( char *filename , int flag , FX_STAT *pStat ) ;
```

パラメータ

I/O	パラメータ	説　明
I	char *filename ;	ステータス情報を変更するファイルのファイル名を格納した領域へのポインタ
I	int flag ;	ステータス変更フラグ FX_CHSTAT_ATTR (0x1)　：ファイルの属性 FX_CHSTAT_ATIME (0x4)　：ファイルのアクセス日時 FX_CHSTAT_MTIME (0x8)　：ファイルの更新日時 FX_CHSTAT_CTIME (0x10)　：ファイルの作成日時
I	FX_STAT *pStat ;	ステータス情報を格納したパケットへのポインタ

【 ステータス情報 _FX_STAT の構造 】

```
struct      _FX_STAT {
            fx_u32      st_size ;     /* ファイルのサイズ ( 単位：バイト ) */
            fx_u32      st_atime ;    /* ファイルのアクセス日時 */
            fx_u32      st_mtime ;    /* ファイルの更新日時 */
            fx_u32      st_ctime ;    /* ファイルの作成日時 */
            fx_u8       st_cmsec ;    /* ファイルの作成日時 ( 単位：10 ミリ秒 )*/
            fx_u8       RFU1 ;        /* システム予約領域 */
            fx_u16      st_mode ;     /* ファイルの属性 */
            fx_u16      RFU2 ;        /* システム予約領域 */
} ;
```

機能

filename で指定されたファイルのステータス情報 (サイズ，アクセス日時，更新日時，作成日時など) を *flag*，*pStat* で指定された状態に変更します。

注意 1　ファイル名 *filename* には，ドライブ名を含むルート・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ファイルまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*filename* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*filename* で指定されたドライブ名，サブディレクトリ名，ファイル名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ステータス情報の変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 4　ステータス情報 _FX_STAT についての詳細は，「5.4.3　ステータス情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	**正常終了**
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit **が発行されていません**
FXE_TASKNOTINIT	0x2	rxfs_TaskInit **が発行されていません**
FXE_MEMORYSMALL	0x3	RX-FS850 **用ヒープ領域が不足しています**
FXE_FILEMAX	0x4	**ファイル管理構造体に空きがありません**
FXE_NOTMOUNT	0x11	rxfs_mount **が発行されていません**
FXE_DRVNOTSUPPORT	0x15	**ドライバ・タイプが不正** (CD-ROM **ファイル・システム**) **です**
FXE_ILLMOUNTMODE	0x16	**操作対象ドライブがフォーマット・モードでマウントされています**
FXE_DRVPROTECTED	0x17	**操作対象ドライブがアクセス禁止状態です**
FXE_FILENOTEXIST	0x20	**該当ファイルが存在しません**
FXE_ILLPATH	0x27	**絶対パス，または，相対パスの指定が不正です**
FXE_BADNAME	0x28	**ファイル名の指定が不正です**
FXE_NAMETOOLONG	0x29	**ファイル名が制限文字数** (FAT：254 **文字**，CD-ROM：64 **文字**) **を超えています**
FXE_PATHTOOLONG	0x2a	**絶対パスが制限文字数** (270 **文字**) **を超えています**
FXE_NOSETWD	0x2b	**カレント・ディレクトリの設定が行われていません**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_settmpdir

概要

一時ディレクトリの設定

C 言語形式

int　　rxfs_settmpdir (char *dirname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*dirname ;	設定する一時ディレクトリのディレクトリ名を格納した領域へのポインタ

機能

一時ディレクトリを *dirname* で指定されたディレクトリに設定します。

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，一時ディレクトリの変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 4　本 API 関数を発行した際，*dirname* で指定される領域に NULL を設定した場合，一時ディレクトリの設定解除が行われます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません

FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ディレクトリが存在しません
FXE_NOTDIR	0x24	パラメータ *dirname* にファイルを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_tmpnam

概要

一時ファイル名の生成

C 言語形式

```
char        *rxfs_tmpnam ( char *filename ) ;
```

パラメータ

I/O	パラメータ	説　明
O	char *filename ;	生成した一時ファイル名を格納する領域へのポインタ

機能

filename で指定された領域に生成した一時ファイル名を格納します。
なお，本 API 関数の処理が正常終了した際には，“ 生成した一時ファイル名 (*dirname*\TEMP*xxx*.TMP：*dirname* は一時ディレクトリ名，*xxxx* は任意 0000 ～ 9999 の数字) を格納した領域へのポインタ ” が戻り値として返されます。

注意　RX-FS850 では，API 関数 rxfs_settmpdir の発行により設定された一時ディレクトリに対して，一時ファイル名の生成処理を行います。

戻り値

正常終了	生成した一時ファイル名を格納した領域へのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_TMPNAMMAX	0x43	一時ファイルの生成数が最大生成数を超えています
FXE_NOSETTMPDIR	0x44	一時ディレクトリの設定が行われていません

FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

5.5.7　ディレクトリ制御管理機能

表 5-16 に，RX-FS850 がディレクトリ制御管理機能として提供している API 関数の一覧を示します。

表 5-16　ディレクトリ制御管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_mkdir	ディレクトリの生成	○	−
rxfs_rmdir	ディレクトリの削除	○	−
rxfs_chdir	カレント・ディレクトリの設定	○	○
rxfs_getwd	カレント・ディレクトリの獲得	○	○
rxfs_realpath	絶対パスの獲得	○	○
rxfs_opendir	ディレクトリのオープン	○	○
rxfs_closedir	ディレクトリのクローズ	○	○
rxfs_readdir	ディレクトリ項目情報の獲得	○	○

rxfs_mkdir

概要

ディレクトリの生成

C 言語形式

int　　rxfs_mkdir (char *mkname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*mkname ;	生成するディレクトリのディレクトリ名を格納した領域へのポインタ

機能

mkname で指定されたディレクトリを生成します。

注意 1　ディレクトリ名 *mkname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*mkname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*mkname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ディレクトリの生成処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています

FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DISKFULL	0x13	該当ディレクトリのデバイスに空き領域がありません
FXE_ROOTFULL	0x14	ルート・ディレクトリに空きエントリがありません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILEEXIST	0x21	既に該当ディレクトリが存在しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_rmdir

概要

ディレクトリの削除

C 言語形式

int　　rxfs_rmdir (char *rmname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*rmname ;	削除するディレクトリのディレクトリ名を格納した領域へのポインタ

機能

rmname で指定されたディレクトリを削除します。

注意 1　ディレクトリ名 *rmname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “\” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “\” でつないだ相対パスを設定します。
なお，RX-FS850 では，*rmname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*rmname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，ディレクトリの削除処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています

FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_DRVNOTSUPPORT	0x15	ドライバ・タイプが不正 (CD-ROM ファイル・システム) です
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ディレクトリが存在しません
FXE_FILEBUSY	0x22	該当ディレクトリがオープンされています
FXE_READONLY	0x23	該当ディレクトリは読み込み専用です
FXE_NOTDIR	0x24	パラメータ *rmname* にファイルを指定しています
FXE_DIRNOTEMPTY	0x26	該当ディレクトリの配下にディレクトリ，または，ファイルが存在しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_chdir

概要

カレント・ディレクトリの設定

C 言語形式

```
int                rxfs_chdir ( char *dirname ) ;
```

パラメータ

I/O	パラメータ	説　　明
I	char　　*dirname ;	変更後のカレント・ディレクトリのディレクトリ名を格納した領域へのポインタ

機能

カレント・ディレクトリを *dirname* で指定されたディレクトリに設定します。

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “\” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “\” でつないだ相対パスを設定します。
なお，RX-FS850 では，*dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数の変更対象はカレント・ディレクトリの他に，カレント・ドライブも含まれています。このため，本 API 関数の処理が完了した際には，カレント・ドライブも *dirname* で指定されたドライブに変更されます。

注意 4　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，カレント・ディレクトリの変更処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

注意 5　本 API 関数を発行した際，*dirname* で指定される領域に NULL を設定した場合，カレント・ディレクトリの設定解除が行われます

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了

FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ディレクトリが存在しません
FXE_NOTDIR	0x24	パラメータ *dirname* にファイルを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_getwd

概要

カレント・ディレクトリの獲得

C 言語形式

```
char            *rxfs_getwd ( char *dirname ) ;
```

パラメータ

I/O	パラメータ	説　明
O	char *dirname ;	獲得したカレント・ディレクトリ名を格納する領域へのポインタ

機能

本 API 関数発行時のカレント・ディレクトリを *dirname* で指定された領域に格納します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得したカレント・ディレクトリ名を格納した領域へのポインタ ” が戻り値として返されます。

注意 1　RX-FS850 では，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パスを *dirname* で指定された領域に格納しています。

注意 2　RX-FS850 では，*dirname* で指定された領域に格納するカレント・ディレクトリ名の最大文字数を 270 文字に規定しています。このため，ユーザは，獲得したカレント・ディレクトリ名の格納領域として，文字列終端の区切り文字“ \ ”，および，NULL 文字“ \0 ”を加えた 272 文字分の領域 (272 バイト) を確保する必要があります。

戻り値

正常終了	獲得したカレント・ディレクトリ名を格納した領域へのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	既にカレント・ドライブがアンマウントされています
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	カレント・ディレクトリが存在しません

FXE_ILLPATH	0x27	**絶対パス，または，相対パスの指定が不正です**
FXE_NOSETWD	0x2b	**カレント・ディレクトリの設定が行われていません**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 ～ 0xff	**ドライバ部の処理が失敗しました**

rxfs_realpath

概要

絶対パスの獲得

C 言語形式

```
char        *rxfs_realpath ( char *relpath , char *abspath ) ;
```

パラメータ

I/O	パラメータ	説　明
I	char *relpath ;	絶対パスを獲得するディレクトリの相対パスを格納した領域へのポインタ
O	char *abspath ;	relpath で指定されたディレクトリの絶対パスを格納する領域へのポインタ

機能

relpath で指定されたディレクトリの絶対パスを *abspath* で指定された領域に格納します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得した絶対パスを格納した領域へのポインタ ” が戻り値として返されます。

注意 1　相対パス *relpath* には，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだパスを設定します。
なお，RX-FS850 では，*relpath* の変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*relpath* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　RX-FS850 では，*relpath* で指定された相対パスにショート・ファイル名が含まれていた場合，該当ショート・ファイル名をロング・ファイル名に変換したのち，*abspath* で指定された領域に格納しています。

注意 4　RX-FS850 では，*abspath* で指定された領域に格納する絶対パスの最大文字数を 270 文字に規定しています。このため，ユーザは，獲得した絶対パスの格納領域として，文字列終端の NULL 文字 “ \0 ” を加えた 271 文字分の領域 (271 バイト) を確保する必要があります。

注意 5　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体が既に使用されていた場合には，絶対パスの獲得処理は行わず，エラー・コードとして FXE_FILEMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	獲得した絶対パスを格納した領域へのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ディレクトリが存在しません
FXE_ILLPATH	0x27	相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_opendir

概要

ディレクトリのオープン

C 言語形式

FX_DIR　　*rxfs_opendir (char *dirname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*dirname ;	オープンするディレクトリのディレクトリ名を格納した領域へのポインタ

機能

dirname で指定されたディレクトリをオープンします。
なお，RX-FS850 では，ディレクトリのオープン処理として，API 関数 (rxfs_closedir，rxfs_readdir) を発行する際，オープンしたディレクトリ毎に必要となるストリームの獲得，および，初期化を行っています。
また，本 API 関数の処理が正常終了した際には，“ オープンしたディレクトリのストリームへのポインタ ” が戻り値として返されます。

注意 1　ディレクトリ名 *dirname* には，ドライブ名を含むルート・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ絶対パス，または，カレント・ディレクトリから該当ディレクトリまでのサブディレクトリ名を “ \ ” でつないだ相対パスを設定します。
なお，RX-FS850 では，*dirname* に相対パスが設定された際には，相対パスから絶対パスへの変換を行っています。このため，変換処理において制限文字数 (270 文字) を超えるような相対パスが設定されていた場合には，FXE_PATHTOOLONG を返しています。

注意 2　RX-FS850 では，*dirname* で指定されたドライブ名，サブディレクトリ名，ディレクトリ名に用いられている英大文字 / 英小文字の区別を行いません。

注意 3　本 API 関数では，API 関数 rxfs_SystemInit を発行した際に指定したシステム初期化情報 _FX_INIT_PACK のメンバ FmngNum と同数のファイル管理構造体，または，FstmNum と同数のストリームが既に使用されていた場合には，ディレクトリのオープン処理は行わず，エラー・コードとして FXE_FILEMAX，または，FXE_OPENMAX を返しています。
なお，システム初期化情報 _FX_INIT_PACK についての詳細は，「5.4.1　システム初期化情報」を参照してください。

戻り値

正常終了	オープンしたディレクトリのストリームへのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了

FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_FILEMAX	0x4	ファイル管理構造体に空きがありません
FXE_OPENMAX	0x5	ストリームに空きがありません
FXE_NOTMOUNT	0x11	rxfs_mount が発行されていません
FXE_ILLMOUNTMODE	0x16	操作対象ドライブがフォーマット・モードでマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_FILENOTEXIST	0x20	該当ディレクトリが存在しません
FXE_NOTDIR	0x24	パラメータ *dirname* にファイルを指定しています
FXE_ILLPATH	0x27	絶対パス，または，相対パスの指定が不正です
FXE_BADNAME	0x28	ディレクトリ名の指定が不正です
FXE_NAMETOOLONG	0x29	ディレクトリ名が制限文字数 (FAT：254 文字，CD-ROM：64 文字) を超えています
FXE_PATHTOOLONG	0x2a	絶対パスが制限文字数 (270 文字) を超えています
FXE_NOSETWD	0x2b	カレント・ディレクトリの設定が行われていません
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 〜 0xff	ドライバ部の処理が失敗しました

rxfs_closedir

概要

ディレクトリのクローズ

C 言語形式

int　　rxfs_closedir (FX_DIR *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_DIR　　*stream ;	クローズするディレクトリのストリームへのポインタ

機能

stream で指定されたディレクトリをクローズします。
なお，RX-FS850 では，ディレクトリのクローズ処理として，API 関数 rxfs_opendir の発行により獲得したストリームの返却を行っています。
したがって，本 API 関数の発行後，該当ディレクトリに対する操作 (API 関数 rxfs_readdir の発行) が禁止されます。

注意　ストリーム *stream* には，API 関数 rxfs_opendir の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ディレクトリは，同一タスク内でオープン処理が行われたディレクトリに限定されます。

戻り値

正常終了	0x0
異常終了	-1

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_NOOPEN	0x30	該当ディレクトリがオープされていません
FXE_ILLSTREAM	0x34	ストリームの状態が不正です
FXE_ILLARG	0x40	パラメータの指定が不正です
FXE_RTOS	0x60	RX850 Pro の処理が失敗しました
FXE_DRIVERERR	0x80 ～ 0xff	ドライバ部の処理が失敗しました

rxfs_readdir

概要

ディレクトリ項目情報の獲得

C 言語形式

FX_DIRENT　　*rxfs_readdir (FX_DIR *stream) ;

パラメータ

I/O	パラメータ	説　　明
I	FX_DIR　　*stream ;	ディレクトリ項目情報を獲得するディレクトリのストリームへのポインタ

機能

stream で指定されたディレクトリのディレクトリ項目情報 (該当ディレクトリの配下に存在しているディレクトリ名，ファイル名など) を獲得します。

なお，本 API 関数の処理が正常終了した際には，“ 獲得したディレクトリ項目情報へのポインタ ” が戻り値として返されます。

注意 1　ストリーム *stream* には，API 関数 rxfs_opendir の戻り値を設定します。
ただし，RX-FS850 では，タスクを単位としたストリームの管理を行っているため，本 API 関数の操作対象ディレクトリは，同一タスク内でオープン処理が行われたディレクトリに限定されます。

注意 2　ディレクトリ項目情報_FX_DIRENT についての詳細は，「5.4.4　ディレクトリ項目情報」を参照してください。

戻り値

正常終了	獲得したディレクトリ項目情報へのポインタ
異常終了	NULL

エラー・コード

FXE_OK	0x0	正常終了
FXE_SYSTEMNOTINIT	0x1	rxfs_SystemInit が発行されていません
FXE_TASKNOTINIT	0x2	rxfs_TaskInit が発行されていません
FXE_MEMORYSMALL	0x3	RX-FS850 用ヒープ領域が不足しています
FXE_NOTMOUNT	0x11	既に操作対象ドライブがアンマウントされています
FXE_DRVPROTECTED	0x17	操作対象ドライブがアクセス禁止状態です
FXE_NOOPEN	0x30	該当ディレクトリがオープンされていません
FXE_ILLSTREAM	0x34	ストリームの状態が不正です

FXE_FINDEOF	0x35	**パラメータ** *stream* **で指定されたディレクトリの末尾までディレクトリ項目情報を獲得ました**
FXE_ILLARG	0x40	**パラメータの指定が不正です**
FXE_RTOS	0x60	RX850 Pro **の処理が失敗しました**
FXE_DRIVERERR	0x80 〜 0xff	**ドライバ部の処理が失敗しました**

5.5.8　エラー管理機能

表 5-17 に，RX-FS850 がエラー管理機能として提供している API 関数の一覧を示します。

表 5-17　エラー管理機能

API 関数名	機能概要	FAT	CD-ROM
rxfs_geterrno	エラー・コードの獲得	○	○

rxfs_geterrno

概要

エラー・コードの獲得

C 言語形式

```
int         rxfs_geterrno ( void ) ;
```

パラメータ

なし

機能

RX-FS850 が提供している API 関数 (rxfs_mount，rxfs_fopen など) を発行した際に返却されるエラー・コードを獲得します。

なお，本 API 関数の処理が正常終了した際には，“ 獲得したエラー・コード (FXE_OK，FXE_SYSTEMNOTINIT など) ” が戻り値として返されます。

注意　RX-FS850 では，タスクを単位としたエラー・コードの管理を行っているため，本 API 関数の獲得対象エラー・コードは，同一タスク内で直前に発行された API 関数のエラー・コードに限定されます。

戻り値

正常終了	獲得したエラー・コード (FXE_OK，FXE_SYSTEMNOTINIT など)

第 6 章　RX-FS850 依存部

本章では，RX-FS850 が提供している RX-FS850 依存部について解説しています。

6.1　概要

RX-FS850 では，RX-FS850 が提供する機能を実現する際に必要となるユーザ・オウン関数，および，ドライバ関数を RX-FS850 依存部 (ユーザ・オウン・コーディング部) として切り出し，サンプル・ソース・ファイルを提供しています。
そこで，システム構築時には，これらユーザ・オウン関数，および，ドライバ関数をユーザの実行環境 / アプリケーション・システムにあわせてカスタマイズ化する必要があります。
以下に，RX-FS850 が RX-FS850 依存部として切り出しているユーザ・オウン関数，および，ドライバ関数を示します。

- **ユーザ・オウン関数** (1 種類)

  fx_GetCurTime

- **ドライバ関数** (11 種類)

  <driver_name>Init　<driver_name>Term　<driver_name>Task　<driver_name>Interrupt
  <driver_name>GetInfo　<driver_name>ClearInfo　<PnP_name>Init　<PnP_name>Term
  <PnP_name>Task　<PnP_name>Interrupt　<PnP_name>Cychdr

また，RX-FS850 では，ドライバ関数をユーザの実行環境 / アプリケーション・システムにあわせてカスタマイズ化する際に有益な RX-FS850 依存部用 API 関数を提供しています。
以下に，RX-FS850 が提供している RX-FS850 依存部用 API 関数を示します。

- RX-FS850 **依存部用** API **関数** (3 種類)

  fx_malloc　fx_free　fx_LockFunction

6.2　データ・マクロ

RX-FS850 が RX-FS850 依存部として切り出しているユーザ・オウン関数，および，ドライバ関数を発行する際に使用する各種データ・マクロ (データ・タイプ，ドライバ・タイプなど) について以下に示します。

6.2.1　データ・タイプ

表 6-1 に，ドライバ関数を発行する際に指定する各種パラメータのデータ・タイプ一覧を示します。
なお，データ・タイプのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_types.h，および，RX850 Pro が提供している標準ヘッダ・ファイル nectools32\inc850\stdrx85p.h から呼び出されるヘッダ・ファイル nectools32\inc850\rx85p\types.h で行われています。

表 6-1　データ・タイプ

マクロ	型	意味
fx_s8	char	符号付き 8 ビット整数
fx_s16	short	符号付き 16 ビット整数
fx_s32	long	符号付き 32 ビット整数
fx_u8	unsigned char	符号無し 8 ビット整数
fx_u16	unsigned short	符号無し 16 ビット整数
fx_u32	unsigned long	符号無し 32 ビット整数

マクロ	型	意味
FX_DRV_MNG	struct _FX_DRV_MNG	ドライバ管理構造体
FX_REQ_PACK	struct _FX_REQ_PACK	リクエスト・パケット
T_MSG	struct t_msg	メッセージ管理構造体
VB	char	データ・タイプが一定しない値 (8 ビット)
VW	long	データ・タイプが一定しない値 (32 ビット)
INT	int	符号付き 32 ビット整数
ID	short	管理オブジェクトの ID 番号
PRI	short	タスク，および，メッセージの優先度

6.2.2　ドライバ・タイプ

表 6-2 に，ライバ関数 <driver_name>Init を発行する際に使用するドライバ・タイプ一覧を示します。
なお，ドライバ・タイプのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

表 6-2　ドライバ・タイプ

マクロ	数値	意味
FX_DRVTYPE_FAT	0x1	FAT ファイル・システム
FX_DRVTYPE_CDROM	0x2	CD-ROM ファイル・システム

6.2.3　ドライバ・コマンド

表 6-3 に，RX-FS850 から送信されたリクエスト・パケットに設定されているドライバ・コマンド一覧を示します。
なお，ドライバ・コマンドのマクロ定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

表 6-3　ドライバ・コマンド

マクロ	型	意味
FX_DRVCMD_BI	0x1	ブロック入力
FX_DRVCMD_BO	0x2	ブロック出力
FX_DRVCMD_GP	0x8	設定パラメータの獲得
FX_DRVCMD_TOC	0x9	セッション情報 (最終セッション先頭トラックの論理ブロック・アドレス) の獲得

6.3　データ構造体

RX-FS850 が提供する機能を実現する際に必要となるデータ構造体 (ドライバ管理構造体，リクエスト・パケットなど) について以下に示します。

6.3.1　ドライバ管理構造体

以下に，ドライバ関数 <driver_name>Init を発行する際に使用するドライバ管理構造体 _FX_DRV_MNG を示します。
なお，ドライバ管理構造体 _FX_DRV_MNG の定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

```
struct      _FX_DRV_MNG {
            char        DriverName [ 0x9 ] ;    /* デバイス・ドライバ名 */
            fx_u8       DriverType ;            /* ドライバ・タイプ */
            fx_s16      ReqMbxID ;              /* メールボックスの ID 番号 */
            fx_u16      CtrlBuffNum ;           /* コントロール用バッファの総数 */
            fx_u16      DataBuffNum ;           /* データ用バッファの総数 */
} ;
```

以下にドライバ管理構造体 _FX_DRV_MNG の詳細を示します。

- DriverName [0x9]

  9 文字以下のユニークなデバイス・ドライバ名を設定します。
  なお，DriverName に格納されたデバイス・ドライバ名は，rxfs_mount を発行する際に利用されます。

- DriverType

  ドライバ・タイプ (ファイル・システムの種別) を設定します。
  なお，DriverType に設定可能な値は，“ 下記に示した値 ” に限られます。

  FX_DRVTYPE_FAT　(0x1)　：FAT ファイル・システム
  FX_DRVTYPE_CDROM　(0x2)　：CD-ROM ファイル・システム

- ReqMbxID

  リクエスト・パケット _FX_REQ_PACK を受信する際に利用するメールボックスの ID 番号を設定します。

- CtrlBuffNum

  コントロール用バッファの総数を設定します。
  なお，CtrlBuffNum に設定可能な値は，“ 0x0 以外の値 ” に限られます。

- DataBuffNum

  データ用バッファの総数を設定します。
  なお，DataBuffNum に設定可能な値は，“ 0x0 以外の値 ” に限られます。

6.3.2　リクエスト・パケット

以下に，ドライバ関数 <driver_name>Task の内部処理で使用するリクエスト・パケット _FX_REQ_PACK を示します。
なお，リクエスト・パケット _FX_REQ_PACK の定義は，標準ヘッダ・ファイル nectools32\inc850\rxfs.h から呼び出されるヘッダ・ファイル nectools32\inc850\rxfs\fx_drvmng.h で行われています。

```
struct      _FX_REQ_PACK {
            T_MSG       MsgCtrl ;       /* メッセージ管理構造体 */
            fx_u16      DriverFunc ;    /* ドライバ・コマンド */
            fx_u8       UnitID ;        /* ユニット番号 */
            fx_u8       PartitionID ;   /* パーティション番号 */
            fx_s16      err ;           /* エラー・コード */
            fx_s16      ResMbxID ;      /* メールボックスの ID 番号 */
            fx_u32      LBA ;           /* 論理ブロックの先頭アドレス */
            fx_u32      iocnt ;         /* 入出力カウント値 */
            fx_u32      DataLen ;       /* データ長 ( 単位：バイト ) */
            char        *BuffAddr ;     /* バッファの先頭アドレスへのポインタ */
} ;
```

以下にリクエスト・パケット _FX_REQ_PACK の詳細を示します。

- MsgCtrl

  メッセージ管理構造体が格納されます。

- DriverFunc

  ドライバ・コマンドが格納されます。
  なお，DriverFunc に格納される値は，“ 下記に示した値 ” となります。

FX_DRVCMD_BI	(0x1)	：ブロック入力
FX_DRVCMD_BO	(0x2)	：ブロック出力
FX_DRVCMD_GP	(0x8)	：設定パラメータの獲得
FX_DRVCMD_TOC	(0x9)	：セッション情報 (最終セッション先頭トラックの論理ブロック・アドレス) の獲得

- UnitID

  ユニット番号が格納されます。
  なお，UnitID に格納される値は，“ rxfs_mount の発行時にパラメータで指定されたユニット番号 ” となります。

- PartitionID

  パーティション ID が格納されます。
  なお，PartitionID に格納される値は，“ rxfs_mount の発行時にパラメータで指定されたパーティション番号 ” となります。

- err

  エラー・コードが格納されます。
  なお，err に格納される値は，“ 0x1 ～ 0x7f ” となります。

- ResMbxID

  処理結果を通知する際に利用するメールボックスの ID 番号が格納されます。

- LBA

  論理ブロックの先頭アドレスが格納されます。
  なお，LBA に格納される値は，DriverFunc に格納されている値により対象となる論理ブロックが異なります。

  - FX_DRVCMD_BI，FX_DRVCMD_BO の場合

    要求パーティションの論理ブロック

  - FX_DRVCMD_TOC の場合

    最終セッションの先頭トラックの論理ブロック

- iocnt

  入出力カウント値 (実際に入出力したデータの総数) が格納されます。

- DataLen

  データ長 (単位：バイト) が格納されます。
  なお，DataLen **に格納される値は，**DriverFunc **に格納されている値により意味が異なります。**

  - FX_DRVCMD_BI **の場合**

    入力データの要求データ長

  - **その他の場合**

    バッファに格納されたデータのデータ長

- BuffAddr

  バッファの先頭アドレスへのポインタが格納されます。

6.3.3　メッセージ管理構造体

以下に，ドライバ関数 <driver_name>Task の内部処理で使用する “ リクエスト・パケット _FX_REQ_PACK ” のメンバとして使用するメッセージ管理構造体 t_msg を示します。

なお，メッセージ管理構造体 t_msg の定義は，RX850 Pro が提供している標準ヘッダ・ファイル nectools32\inc850\stdrx85p.h から呼び出されるヘッダ・ファイル nectools32\inc850\rx85p\packet.h で行われています。

```
struct      t_msg {
            VW          msgrfu ;        /* メッセージの管理領域 ( システム予約領域 ) */
            PRI         msgpri ;        /* メッセージの優先度 */
            VB          msgcont [ 3 ] ; /* メッセージの本体 */
} ;
```

以下にメッセージ管理構造体 t_msg の詳細を示します。

- msgrfu

  システム予約領域です。
  なお，msgrfu に設定可能な値は “ 0x0 ” に限られます。

- msgpri

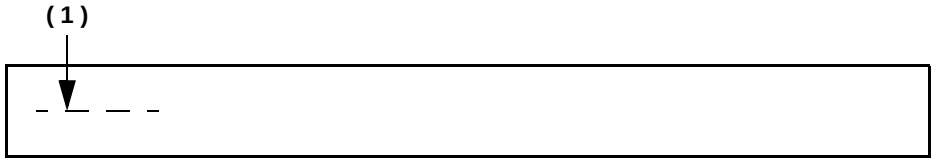
  メッセージの優先度を設定します。
  なお，msgpri に設定可能な値は “ 0x1 ～ 0x7fff ” に限られます。

- msgcont [3]

  メッセージの本体 (内容) を設定します。

6.4　ユーザ・オウン関数 / ドライバ関数解説

次項から RX-FS850 が提供しているユーザ・オウン関数，および，ドライバ関数について，以下の記述フォーマットに従って解説します。



(2) ⟶ 概要

(3) ⟶ C 言語形式

(4) ⟶ パラメータ

I/O	パラメータ	説　　明

(5) ⟶ 機能

(6) ⟶ 戻り値

（1） 名称

ユーザ・オウン関数，または，ドライバ関数の名称を示しています。

（2） 概要

ユーザ・オウン関数，または，ドライバ関数の機能概要を示しています。

（3） C 言語形式

ユーザ・オウン関数，または，ドライバ関数を RX-FS850 から呼び出す際の記述形式を示しています。

（4） パラメータ

ユーザ・オウン関数，または，ドライバ関数のパラメータを以下の形式で示しています。

I/O	パラメータ	説　　明
A	B	C

A ）パラメータの種類

I　　…ユーザ・オウン関数，または，ドライバ関数への入力パラメータ
O　　…ユーザ・オウン関数，または，ドライバ関数からの出力パラメータ

B ）パラメータのデータ・タイプ

C ）パラメータの説明

（5） 機能

ユーザ・オウン関数，または，ドライバ関数の機能詳細を示しています。

（6） 戻り値

ユーザ・オウン関数，または，ドライバ関数からの戻り値を数値で示しています。

6.4.1　ユーザ・オウン関数

表 6-4 に，RX-FS850 が提供しているユーザ・オウン関数の一覧を示します。

表 6-4　ユーザ・オウン関数

ユーザ・オウン関数名	機能概要
fx_GetCurTime	現在時刻の獲得関数

fx_GetCurTime

概要

現在時刻の獲得関数

C 言語形式

```
void        fx_GetCurTime ( fx_u16 *pDate , fx_u16 *pTime , fx_u8 *pMtime ) ;
```

パラメータ

I/O	パラメータ	説　明
O	fx_u16 *pDate ;	現在時刻 (年月日) を格納する領域へのポインタ 年月日　：(年 - 1980) << 9) | (月 << 5) | 日
O	fx_u16 *pTime ;	現在時刻 (時分秒) を格納する領域へのポインタ 時分秒　：(時 << 11) | (分 << 5) | (秒 >> 1)
O	fx_u8 *pMtime ;	現在時刻 (10 ミリ秒) を格納する領域へのポインタ 10 ミリ秒 ：(秒 & 1) * 100 + ミリ秒 / 10)

機能

RX-FS850 がファイル，または，ディレクトリのエントリに記録する現在時刻獲得処理を実行した際に呼び出される関数です。
以下に，本ユーザ・オウン関数で実行すべき処理を示します。

- 現在時刻 (年月日時分秒ミリ秒) を RTC (Real Time Clock) などから獲得
- 現在時刻 (年月日) を *pDate* で指定された領域に設定
- 現在時刻 (時分秒) を *pTime* で指定された領域に設定
- 現在時刻 (10 ミリ秒) を *pMtime* で指定された領域に設定

注意　RX-FS850 では，本ユーザ・オウン関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\user\src\gettime.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\user\src\gettime.c

戻り値

なし

6.4.2　ドライバ関数

表 6-5 に，RX-FS850 が提供しているドライバ関数の一覧を示します。

表 6-5　ドライバ関数

ドライバ関数名	機能概要
<driver_name>Init	デバイス・ドライバの初期化関数
<driver_name>Term	デバイス・ドライバの終了関数
<driver_name>Task	デバイス・ドライバ用常駐タスク
<driver_name>Interrupt	デバイス・ドライバ用間接起動割り込みハンドラ
<driver_name>GetInfo	デバイス情報の獲得関数
<driver_name>ClearInfo	デバイス情報のクリア関数
<PnP_name>Init	プラグ・アンド・プレイ機能の初期化関数
<PnP_name>Term	プラグ・アンド・プレイ機能の終了関数
<PnP_name>Task	プラグ・アンド・プレイ機能用常駐タスク
<PnP_name>Interrupt	プラグ・アンド・プレイ機能用間接起動割り込みハンドラ
<PnP_name>Cychdr	プラグ・アンド・プレイ機能用周期起動ハンドラ

<*driver_name*>Init

概要

デバイス・ドライバの初期化関数

C 言語形式

```
int             <driver_name>Init ( FX_DRV_MNG *pDrvMng , ... ) ;
```

パラメータ

I/O	パラメータ	説　　明
O	FX_DRV_MNG　*pDrvMng ;	ドライバ管理構造体を格納するパケットへのポインタ
I,O	その他	該当デバイス・ドライバに依存したパラメータ

【 ドライバ管理構造体 _FX_DRV_MNG の構造 】

```
struct          _FX_DRV_MNG {
                char        DriverName [ 0x9 ] ;    /* デバイス・ドライバ名 */
                fx_u8       DriverType ;            /* ドライバ・タイプ */
                fx_s16      ReqMbxID ;              /* メールボックスの ID 番号 */
                fx_u16      CtrlBuffNum ;           /* コントロール用バッファの総数 */
                fx_u16      DataBuffNum ;           /* データ用バッファの総数 */
} ;
```

機能

デバイス・ドライバの初期化処理を記述するために用意された初期化処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- 該当デバイスの初期化
- メールボックスの生成
- デバイス・ドライバ用常駐タスク <driver_name>Task の生成 / 起動
- デバイス・ドライバ用間接起動割り込みハンドラ <driver_name>Interrupt の登録
- マスカブル割り込みの受け付け許可
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

注意 1　本ドライバ関数の呼び出しは，API 関数 rxfs_SystemInit の発行以前に行う必要があります。

注意 2　ドライバ管理構造体 _FX_DRV_MNG についての詳細は，「6.3.1　ドライバ管理構造体」を参照してください。

注意 3　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

正常終了	0x0
異常終了	-1

<*driver_name*>Term

概要

デバイス・ドライバの終了関数

C 言語形式

int　　<*driver_name*>Term (...) ;

パラメータ

I/O	パラメータ	説　　明
I, O	その他	該当デバイス・ドライバに依存したパラメータ

機能

デバイス・ドライバの終了処理を記述するために用意された終了処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- マスカブル割り込みの受け付け禁止
- デバイス・ドライバ用間接起動割り込みハンドラ <driver_name>Interrupt の登録解除
- デバイス・ドライバ用常駐タスク <driver_name>Task の強制終了 / 削除
- メールボックスの削除
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

注意 1　本ドライバ関数の呼び出しは，API 関数 rxfs_SystemTerm の処理完了後に行う必要があります。

注意 2　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

正常終了	0
異常終了	-1

<*driver_name*>Task

概要

デバイス・ドライバ用常駐タスク

C 言語形式

void　　<*driver_name*>Task (INT *stacd*) ;

パラメータ

I/O	パラメータ	説　　明
I	INT　　*stacd* ;	起動コード

機能

RX-FS850 からリクエスト・パケットが送信された際に起床するデバイス・ドライバ用常駐タスクです。
以下に，本ドライバ関数で実行すべき処理を示します。

- RX-FS850 から送信されたリクエスト・パケットの受信
- 受信したリクエスト・パケットの内容 (ドライバ・コマンド DriverFunc，エラー・コード err など) に対応した処理
- 本ドライバ関数の処理結果を RX-FS850 に送信
- デバイス・ドライバ用常駐タスク (自タスク) を wait 状態 (メッセージ待ち状態) へと遷移

注意 1　起動コード *stacd* の使用用途については，該当デバイス・ドライバに依存しています。

注意 2　リクエスト・パケット _FX_REQ_PACK についての詳細は，「6.3.2　リクエスト・パケット」を参照してください。

注意 3　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

なし

<*driver_name*>Interrupt

概要

デバイス・ドライバ用間接起動割り込みハンドラ

C 言語形式

ID　　<*driver_name*>Intrrupt (void) ;

パラメータ

なし

機能

マスカブル割り込みが発生した際に呼び出されるデバイス・ドライバ用間接起動割り込みハンドラです。
以下に，本ドライバ関数で実行すべき処理を示します。

- デバイス・ドライバ用常駐タスク <driver_name>Task に対する起床要求の発行
- プロセッサの割り込み終了処理
- 0x0 を本ドライバ関数の戻り値として設定

戻り値

正常終了	0x0

<*driver_name*>GetInfo

概要

デバイス情報の獲得関数

C 言語形式

int　　<*driver_name*>GetInfo (...) ;

パラメータ

I/O	パラメータ	説　　明
I, O	その他	該当デバイス・ドライバに依存したパラメータ

機能

デバイス情報 (パーティション情報など) の獲得処理を記述するために用意された獲得処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- デバイス情報の獲得
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

戻り値

正常終了	0
異常終了	-1

<*driver_name*>ClearInfo

概要

デバイス情報のクリア関数

C 言語形式

int　　<*driver_name*>ClearInfo (...) ;

パラメータ

I/O	パラメータ	説　　明
I, O	その他	該当デバイス・ドライバに依存したパラメータ

機能

デバイス情報 (パーティション情報など) のクリア処理を記述するために用意されたクリア処理専用ルーチンです。以下に，本ドライバ関数で実行すべき処理を示します。

- デバイス情報のクリア
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

戻り値

正常終了	0
異常終了	-1

<*PnP_name*>Init

概要

プラグ・アンド・プレイ機能の初期化関数

C 言語形式

int　　<*PnP_name*>Init (...) ;

パラメータ

I/O	パラメータ	説　　明
I,O	その他	該当デバイス・ドライバに依存したパラメータ

機能

動的にデバイスの接続 / 取り外しを可能とするための機能 (プラグ・アンド・プレイ機能) の初期化処理を記述するために用意された初期化処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- イベントフラグの生成
- プラグ・アンド・プレイ機能用常駐タスク <PnP_name>Task の生成 / 起動
- プラグ・アンド・プレイ機能用間接起動割り込みハンドラ <PnP_name>Interrupt の登録
- マスカブル割り込みの受け付け許可
- プラグ・アンド・プレイ機能用周期起動ハンドラ <PnP_name>Cychdr の登録
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

注意 1　本ドライバ関数の呼び出しは，API 関数 rxfs_SystemInit の発行以前に行う必要があります。

注意 2　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

正常終了	0x0
異常終了	-1

<*PnP_name*>Term

概要

プラグ・アンド・プレイ機能の終了関数

C 言語形式

int　　<*PnP_name*>Term (...) ;

パラメータ

I/O	パラメータ	説　　明
I, O	その他	該当デバイス・ドライバに依存したパラメータ

機能

デバイス・ドライバの終了処理を記述するために用意された終了処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- プラグ・アンド・プレイ機能用周期起動ハンドラ <PnP_name>Cychdr の登録解除
- マスカブル割り込みの受け付け禁止
- プラグ・アンド・プレイ機能用間接起動割り込みハンドラ <PnP_name>Interrupt の登録解除
- プラグ・アンド・プレイ機能用常駐タスク <PnP_name>Task の削除
- イベント・フラグの削除
- 該当処理が正常終了したか否かを本ドライバ関数の戻り値として設定

注意 1　本ドライバ関数の呼び出しは，API 関数 rxfs_SystemTerm の処理完了後に行う必要があります。

注意 2　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

正常終了	0
異常終了	-1

<*PnP_name*>Task

概要

プラグ・アンド・プレイ機能用常駐タスク

C 言語形式

void　　<*PnP_name*>Task (INT *stacd*) ;

パラメータ

I/O	パラメータ	説　明
I	INT　*stacd* ;	起動コード

機能

RX-FS850 からリクエスト・パケットが送信された際に起床するデバイス・ドライバ用常駐タスクです。
以下に，本ドライバ関数で実行すべき処理を示します。

- プラグ・アンド・プレイ機能用常駐タスク (自タスク) を RX-FS850 に登録
- プラグ・アンド・プレイ機能用常駐タスク (自タスク) を wait 状態 (イベントフラグ待ち状態) へと遷移
- プラグ・アンド・プレイ機能用間接起動割り込みハンドラ <PnP_name>Interrupt，または，プラグ・アンド・プレイ機能用周期起動ハンドラ <PnP_name>Cychdr により検出された " デバイスの接続 / 取り外し " に対応した処理

  【" デバイスの接続 " を検出した場合】
  - デバイス情報 (パーティション情報など) の獲得
  - ドライブのマウント

  【" デバイスの取り出し " を検出した場合】
  - ドライブの強制アンマウント
- プラグ・アンド・プレイ機能用常駐タスク (自タスク) を RX-FS850 から登録解除
- プラグ・アンド・プレイ機能用常駐タスク (自タスク) の終了

注意 1　起動コード *stacd* の使用用途については，該当デバイス・ドライバに依存しています。

注意 2　リクエスト・パケット _FX_REQ_PACK についての詳細は，「6.3.2　リクエスト・パケット」を参照してください。

注意 3　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

なし

<*PnP_name*>Interrupt

概要

プラグ・アンド・プレイ機能用間接起動割り込みハンドラ

C 言語形式

void　　<*PnP_name*>Interrupt (void) ;

パラメータ

なし

機能

マスカブル割り込みの発生を利用して，デバイスの接続 / 取り外しが行われた事をプラグ・アンド・プレイ機能用常駐タスク <PnP_name>Task に通知するために用意された通知処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- プラグ・アンド・プレイ機能用常駐タスク <PnP_name>Task に対する起床要求の発行
- プロセッサの割り込み終了処理
- プラグ・アンド・プレイ機能用間接起動割り込みハンドラからの復帰

注意　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

なし

<*PnP_name*>Cychdr

概要

プラグ・アンド・プレイ機能用周期起動ハンドラ

C 言語形式

void　　<*PnP_name*>Cychdr (void) ;

パラメータ

なし

機能

周期的なポーリング処理を利用して，デバイスの接続 / 取り外しが行われたか否かを確認するために用意された確認処理専用ルーチンです。
以下に，本ドライバ関数で実行すべき処理を示します。

- デバイスの接続 / 取り外しが行われたか否かの確認
- プラグ・アンド・プレイ機能用常駐タスク <PnP_name>Task に対する起床要求の発行

注意　RX-FS850 では，本ドライバ関数のサンプル・ソース・ファイルを提供しています。

【 CA850 対応版の場合 】
nectools32\smp850e\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

【 GHS 製コンパイラ対応版の場合 】
nectools32\smp850e_ghs\rxfs\<*target_name*>\drivers\<*driver_name*>\src\<*driver_name*>.c

戻り値

なし

6.4.3　RX-FS850 依存部用 API 関数

表 6-6 に，RX-FS850 が提供している RX-FS850 依存部用 API 関数の一覧を示します。

表 6-6　RX-FS850 依存部用 API 関数

ユーザ・オウン関数名	機能概要
fx_malloc	メモリ領域の獲得
fx_free	メモリ領域の返却
fx_LockFunction	RX-FS850 が提供している API 関数との排他制御

fx_malloc

概要

メモリ領域の獲得

C 言語形式

```
void        *fx_malloc ( unsigned long memsz ) ;
```

パラメータ

I/O	パラメータ	説　明
I	unsigned long　*memsz* ;	メモリ領域のサイズ (単位：バイト)

機能

memsz で指定されたサイズのメモリ領域を API 関数 rxfs_HeapInit の発行により確保した RX-FS850 用ヒープ領域から獲得します。
なお，本 API 関数の処理が正常終了した際には，“ 獲得したメモリ領域の先頭アドレス ” が戻り値として返されます。

注意　RX-FS850 では，本 API 関数の発行をドライバ関数に限定しています。

戻り値

正常終了	獲得したメモリ領域の先頭アドレス
異常終了	NULL

fx_free

概要

メモリ領域の返却

C 言語形式

void　　fx_free (void *p_mem) ;

パラメータ

I/O	パラメータ	説　　明
I	void　　*p_mem ;	返却するメモリ領域の先頭アドレスを格納した領域へのポインタ

機能

API 関数 fx_malloc の発行により獲得したメモリ領域を RX-FS850 用ヒープ領域に返却します。

注意 1　RX-FS850 では，本 API 関数の発行をドライバ関数に限定しています。

注意 2　返却するメモリ領域の先頭アドレスには，API 関数 fx_malloc の戻り値を設定します。
なお，RX-FS850 では，*p_mem* で指定された領域に “ API 関数 fx_malloc の戻り値 ” 以外を設定した際の動作については保証していません。

注意 3　RX-FS850 ではメモリ領域を返却する際，該当メモリ領域のクリア処理を行っていないため，返却されたメモリ領域の内容は不定となります。

戻り値

なし

fx_LockFunction

概要

RX-FS850 が提供している API 関数との排他制御

C 言語形式

fx_u32　　fx_LockFunction (fx_u32 (*p_func) ( ) , fx_u32 *param*) ;

パラメータ

I/O	パラメータ	説　明
I	fx_u32　(*p_func) ( ) ;	排他制御を行って実行する関数の起動アドレスが格納された領域へのポインタ
I	fx_u32　*param* ;	*p_func* のパラメータ

機能

p_func で指定された関数を RX-FS850 が提供している API 関数との排他制御を行って実行します。
なお，*param* で指定された領域には，該当関数 *p_func* のパラメータを格納します。

注意　RX-FS850 では，本 API 関数の発行をプラグ・アンド・プレイ機能用ドライバ関数に限定しています。

戻り値

正常終了	該当関数 *p_func* の戻り値
異常終了	0xffffffff

索引

A

C

D

F

P

R

S

い

え

お

か

き

け

こ

し

す

た

ち

て

と

に

は

ひ

ふ

ま

め

も

ゆ

り

ろ

［メ　モ］

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクス特約店へお申し付けください。