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。

ユーザーズ・マニュアル

RENESAS

RX-NET

ネットワーク・ライブラリ

SMTP

対象デバイス

V850 ファミリ ™

対象リアルタイム ＯＳ

RX850 Pro Ver.3.13 以上

対象 TCP/IP ライブラリ

RX-NET(TCP/IP) Ver.1.20 以上

資料番号　U15505JJ2V0UM00（第 2 版）
発行年月　August 2001 CP(K)

商標等について

・V800 シリーズ，V850 ファミリは，日本電気株式会社の商標です。
・IBM，AT は，米国 International Business Machines, Inc. の登録商標です。
・MS-DOS，Windows，Windows NT は，米国 Microsoft Corporation の米国及びその他の国における登録商標です。
・Green Hills Software，MULTI 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。

・本資料の内容は，後日変更する場合があります。
・文書による当社の許諾なしに本資料の転載複製を禁じます。
・本資料に掲載された製品の使用もしくは本資料に記載された情報の使用に際して，当社は当社もしくは第三者の知的所有権その他の権利に対する保証，または，実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利に関わる問題が生じた場合，当社はその責を負うものではありませんので御了承ください。
・本資料に記載された回路，ソフトウエア，および，これらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これらの回路，ソフトウエア，情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責任を負いません。
・本製品が外国為替，および，外国貿易管理法の規定による規制貨物等 (または，役務) に該当するか否かは，お客様 (仕様を決定した者) が判定してください。

履歴表

版数	発行年月日	改定内容（理由）
1.00	2001 年 4 月 23 日	RX-NET(SMTP) V1.10 対応 新規作成
2.00	2001 年 8 月 24 日	RX-NET(SMTP) V1.20 対応 改版 **1.1　概要** 「図 1-1　RX-NET(SMTP) の位置付け」から“ネットワーク・ライブラリ TCP/IP”，“ネットワーク・ライブラリ SMTP”，および，“リアルタイム OS”を削除。。 **1.2　特徴** 「図 1-2　RX-NET(SMTP) の階層的位置付け」の記述形式を TCP/IP モデルから OSI 参照モデルに変更。 **1.3　実行環境** RX-NET(SMTP) が処理を実行するうえで必要となるプロセッサを“V800 シリーズ V850 ファミリ V850E シリーズ”から“V800 シリーズ V850 ファミリ V850E/xxx”に変更。 **1.4　開発環境** “SunOS”に関する記述を削除。 RX-NET(TCP/IP) の対応バージョンを“V1.10.”から“V1.20”に変更。 C コンパイラ・パッケージ (CA850，CCV850E) の対応バージョンに関する記述を追加。 **2.2.1　Windows ベース** セットアップ・プログラム SETUP.EXE が格納されているディレクトリを“rx-net¥smtp¥disk1”から“RX-NET_SMTP_V850E_NEC¥DISK1”に変更。 **第 3 章　システム構築** 「3.8　ライブラリ・ファイルの生成」で記述されているデバイス・ドライバ・オブジェクトの生成方法を「3.7　オブジェクト・ファイルの生成」に移動。 **図 3-1　システム構築手順** リンク・エディタを起動して，デバイス・ドライバ・オブジェクトを生成する旨の記述を追加。 **3.2　CF 定義ファイルの記述** システム情報において 2 タスク分確保するものを“タスクの ID 番号の保護範囲”から“タスクの自動割り付け ID 番号”に修正。 システム情報において 1 セマフォ分確保するものを“セマフォの ID 番号の保護範囲”から“セマフォの自動割り付け ID 番号”に修正。 RX-NET(TCP/IP) が各種機能を実現するうえで必要となるシステム・コールとして“get_tim″を追加。 **3.3　情報ファイルの生成** cf850pro を実行する際の入力例として記述されているシステム情報テーブルのファイル名を“sitfile.tbl”から“sitfile.s”に，システム・コール・テーブルのファイル名を“sctfile.tbl”から“sctfile.s”に変更。 **3.9　リンク・ディレクティブ・ファイルの記述** CA850 対応版で提供しているサンプル・ソース・ファイルのファイル名を“sample.lnk”から“sample.dir”に変更。

目次

図目次

表目次

第 1 章　　概説

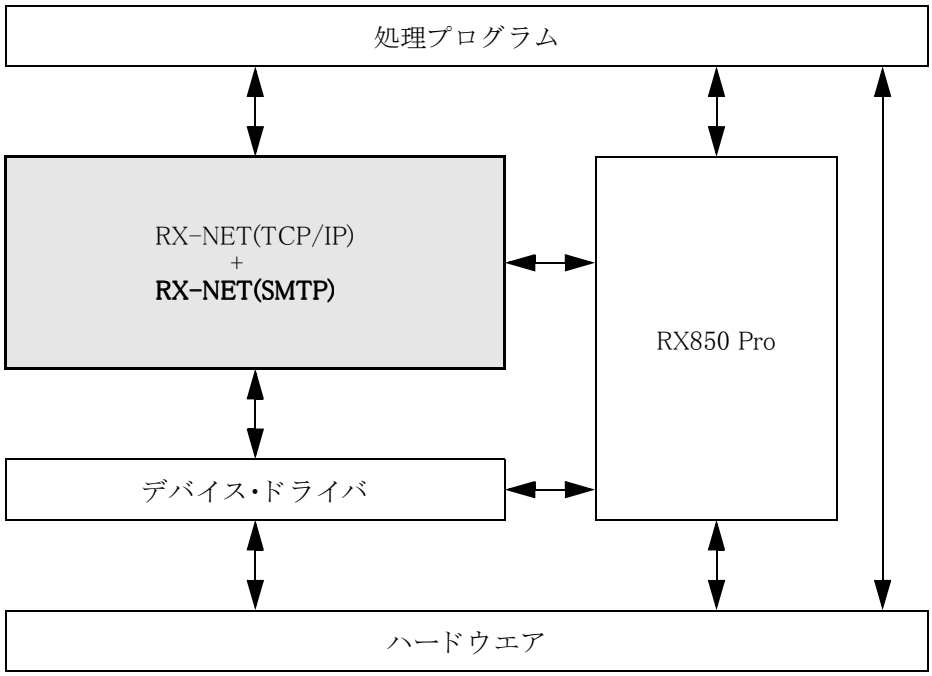
1.1　概要

RX-NET(SMTP) は，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 製）上で動作する TCP/IP ライブラリ RX-NET に対し，SMTP (Simple Mail Transfer Protocol) による電子メール送信を行う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-NET(SMTP) が提供する API 関数を利用することにより，電子メールの送信が可能となります。

図 1-1 に，RX-NET(SMTP) の位置付けを示します。

図 1-1　RX-NET(SMTP) の位置付け



注意　RX-NET(SMTP) では，電子メール・クライアント内部で起動される sender-SMTP（送信 - SMTP) 機能のみを提供しています。このため，RX-NET(SMTP) には，receive-SMTP（受信 - SMTP) 機能が含まれていません。

1.2　特徴

以下に，RX-NET(SMTP) の特徴を示します。

- **RFC に準拠**

  RX-NET(SMTP) では，RFC に準拠した設計が行われています。

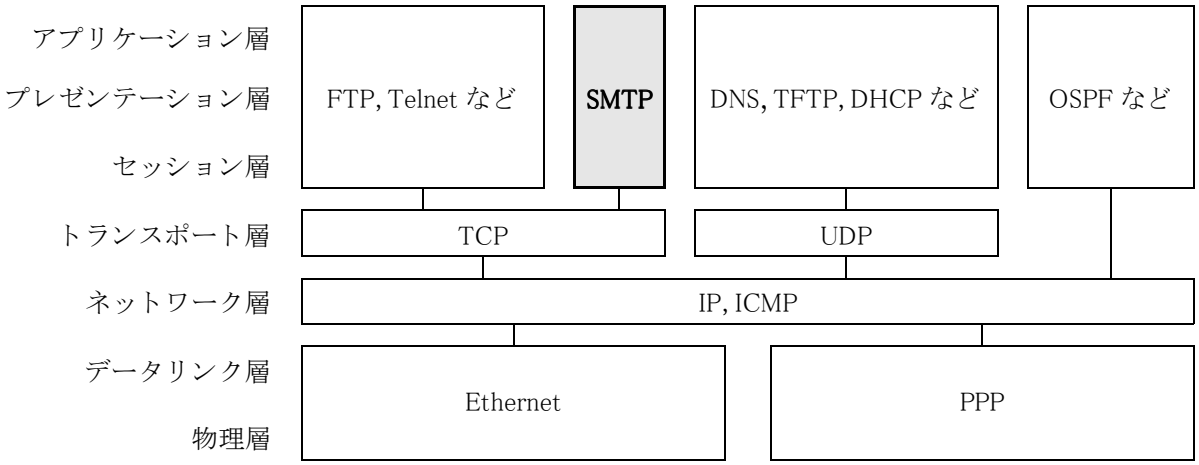
  なお，RFC とは，インターネットに関する研究開発期間 IETF (Internet Engineering Task Force) が取りまとめた公開技術文書であり，電子メール，ファイル転送などのプロトコル仕様 (情報交換を行う際に必要な手順，および，規約) の他にも，各種サービス，ガイドラインといった多岐に渡った情報 (ネットワーク技術の実装と運用に主眼を置いた情報) が記載されています。

- **電子メール送信機能をサポート**

  RX-NET(SMTP) では，インターネットで利用される電子メール送信用の API 関数を提供しています。

  図 1-2 に，RX-NET(SMTP) の階層的位置付けを示します。

図 1-2　RX-NET(SMTP) の階層的位置付け



- **マルチタスク処理を意識した設計**

  RX-NET(SMTP) 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-NET(SMTP) 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-NET(SMTP) 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。

  V800 シリーズ V850 ファミリ V850E/xxx

- **周辺コントローラ**
  RX-NET(SMTP) では，処理を実行するうえで，特定の周辺コントローラは要求しません。

- **メモリ容量**
  以下に，RX-NET(SMTP) が処理を実行するうえで必要となるメモリ容量を示します。

  RX-NET(SMTP) のテキスト領域　：約 5 K バイト
  RX-NET(SMTP) のデータ領域　：約 5 K バイト

1.4 開発環境

以下に，RX-NET(SMTP) 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

    PC-9800 シリーズ　：Windows 2000，95，98，Me，NT 4.*x*
    IBM-PC/AT 互換機　：Windows 2000，95，98，Me，NT 4.*x*
    SPARC station　：Solaris Rel.2.5.*x*

- **ソフトウエア**

  - リアルタイム OS

    RX850 Pro Ver.3.13 以上　：NEC 製

  - ネットワーク・ライブラリ

    RX-NET(TCP/IP) Ver.1.20 以上　：NEC 製

  - C コンパイラ・パッケージ

    CA850 Ver.2.40 以上　：NEC 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-NET(SMTP) 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-NET(SMTP) の提供媒体は，ホスト・マシンの種類 (Windows ベース，UNIX ベース) に併せて計 2 種類が用意されています。
表 2-1 に，RX-NET(SMTP) の提供形式一覧を示します。

表 2-1　RX-NET(SMTP) の提供形式

ホスト・マシン	提供形式	提供媒体
Windows ベース ・ PC-9800 シリーズ ・ IBM-PC/AT 互換機	CA850 対応版オブジェクト・ファイル形式 CCV850E 対応版オブジェクト・ファイル形式	CD-ROM
UNIX ベース ・ SPARC station	CA850 対応版オブジェクト・ファイル形式 CCV850E 対応版オブジェクト・ファイル形式	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，CCV850E 対応版オブジェクト・ファイル形式) の RX-NET(SMTP) 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-NET(SMTP) 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-NET(SMTP) 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。

注意　RX-NET(SMTP) のインストールは，RX-NET(TCP/IP) のインストール完了後に行ってください。

2.2.1　　Windows ベース

以下に，RX-NET(SMTP) 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：PC-9800 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-NET(SMTP) 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-NET(SMTP) の提供媒体のディレクトリ RX-NET_SMTP_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-NET(SMTP) 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-NET(SMTP) 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-NET(SMTP) 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
- CCV850E 対応版

そこで，C コンパイラ・パッケージが NEC 製 CA850 の場合は圧縮ファイル nec/rxnsmtp.tar を，米国 Green Hills Software, Inc. 製 CCV850E の場合は圧縮ファイル ghs/rxnsmtp.tar を展開します。

【 CA850 対応版の場合 】

```
% tar △ -xvof △ /cdrom/RXNET/nec/rxnsmtp.tar <Enter>
```

【 CCV850E 対応版の場合 】

```
% tar △ -xvof △ /cdrom/RXNET/ghs/rxnsmtp.tar <Enter>
```

6） ファイル群の確認
ls コマンドを実行し，RX-NET(SMTP) 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

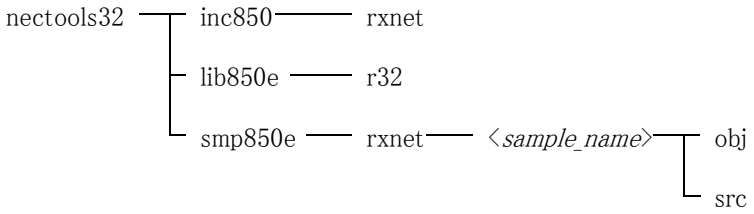
RX-NET(SMTP) の提供媒体に格納されているファイル群のディレクトリ構成は，C コンパイラ・パッケージの種類 (NEC 製 CA850，米国 Green Hills Software, Inc. 製 CCV850E) により異なります。

そこで，以降に，C コンパイラ・パッケージが CA850 の場合，CCV850E 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-NET(SMTP) 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-NET(SMTP) 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。

rxnet_smtp.h　: RX-NET(SMTP) 用標準ヘッダ・ファイル

2) nectools32¥inc850¥rxnet
RX-NET(SMTP) 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
SMTP ライブラリ (32 レジスタ・モード) が格納されているディレクトリです。

libsmtp.a　　: SMTP ライブラリ

4) nectools32¥smp850e¥rxnet¥〈*sample_name*〉¥obj
ロード・モジュールを生成するためのメイク・ファイル Makefile が格納されているディレクトリです。なお，本ディレクトリにおいて，make コマンドを実行することにより，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。

Makefile　　: ロード・モジュール用メイク・ファイル

5) nectools32¥smp850e¥rxnet¥〈*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。

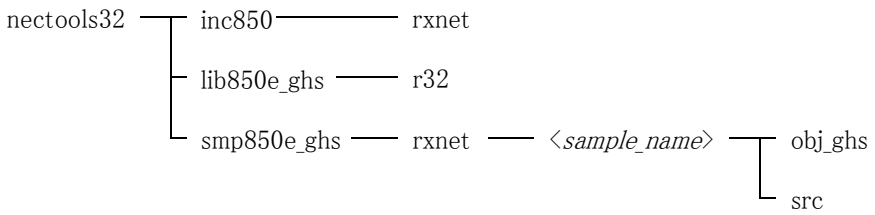
注意　〈*sample_name*〉についての詳細は，下記に示したテキスト・ファイルを参照してください。

〈*sample_name*〉　　: nectools32¥smp850e¥rxnet¥README.SMTP

2.3.2 CCV850E 対応版

図 2-2 に，RX-NET(SMTP) の提供媒体 (CCV850E 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 (CCV850E 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-NET(SMTP) 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。

rxnet_smtp.h ： RX-NET(SMTP) 用標準ヘッダ・ファイル

2 ） nectools32¥inc850¥rxnet
RX-NET(SMTP) 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
SMTP ライブラリ (32 レジスタ・モード) が格納されているディレクトリです。

libsmtp.a ： SMTP ライブラリ

4 ） nectools32¥smp850e_ghs¥rxnet¥〈*sample_name*〉¥obj_ghs
ロード・モジュールを生成するためのビルド・ファイル sample.bld 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 sample.bld 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

5 ） nectools32¥smp850e_ghs¥rxnet¥〈*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。

注意 〈*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。

〈*sample_name*〉 ： nectools32¥smp850e_ghs¥rxnet¥README.SMTP

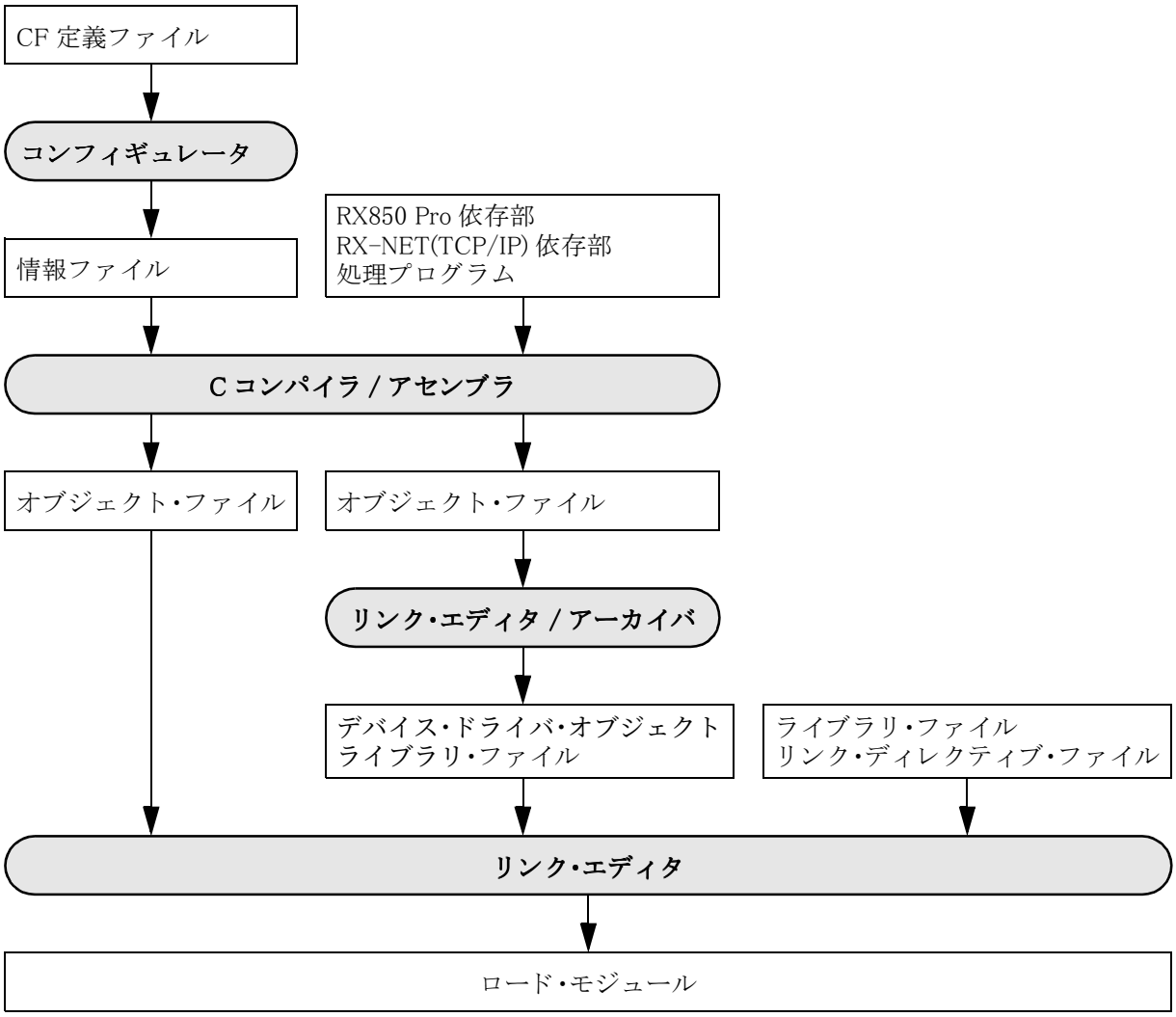
第 3 章　　システム構築

本章では，RX-NET(SMTP) を使用したネットワーク・アプリケーション (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-NET(SMTP) 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ネットワーク・アプリケーション (ロード・モジュール) を生成することです。
図 3-1 に，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 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-NET(TCP/IP) では，2 個のタスク，1 個の間接起動割り込みハンドラ，1 個の周期起動ハンドラ，12 種類のシステム・コールを，RX-NET(SMTP) では 1 個のセマフォ，5 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，ユーザが記述した処理プログラムを動作させるうえで必要となる情報の他に，以下に示した情報を RX-NET(TCP/IP) 用，および，RX-NET(SMTP) 用に確保 / 定義する必要があります。

- **SIT 情報**
  - システム情報
    RX-NET(TCP/IP) 用に“タスクの自動割り付け ID 番号”として 2 タスク分を確保。
    RX-NET(SMTP) 用に“セマフォの自動割り付け ID 番号”として 1 セマフォ分を確保。
  - システム最大値情報
    RX-NET(TCP/IP) 用に“タスクの最大生成数”として 2 タスク分を確保。
    RX-NET(SMTP) 用に“セマフォの最大生成数”として 1 セマフォ分を確保。
    RX-NET(TCP/IP) 用に“間接起動割り込みハンドラの最大登録数”として 1 間接起動割り込みハンドラ分を確保。
    RX-NET(TCP/IP) 用に“周期起動ハンドラの最大登録数”として 1 周期起動ハンドラ分を確保。
- **SCT 情報**
  - タスク管理機能情報
    RX-NET(TCP/IP) 用に“cre_tsk，sta_tsk，dis_dsp，ena_dsp，get_tid”を定義。
  - タスク付属同期機能情報
    RX-NET(TCP/IP) 用に“slp_tsk，wup_tsk”を定義。
  - 同期通信（セマフォ）機能情報
    RX-NET(SMTP) 用に“cre_sem，del_sem，sig_sem，wai_sem，preq_sem”を定義。
  - 割り込み処理管理機能情報
    RX-NET(TCP/IP) 用に“def_int”を定義。
  - 時間管理機能情報
    RX-NET(TCP/IP) 用に“get_tim，dly_tsk，def_cyc”を定義。
  - システム管理機能情報
    RX-NET(TCP/IP) 用に“ref_sys”を定義。

注意　CF 定義ファイルを記述するうえでの注意事項，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。
なお，RX-NET(SMTP) 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¥rxnet¥<*sample_name*>¥src
  sys.cf　　：CF 定義ファイル

【 CCV850E 対応版の場合 】

- nectools32¥smp850e_ghs¥rxnet¥<*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 cffile.cf を読み込んだのち，システム情報テーブル sitfile.s，システム・コール・テーブル sctfile.s，システム情報ヘッダ・ファイル hfile.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 △ sitfile.s △ -c △ sctfile.s △ -d △ hfile.h △ cffile.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。
なお，RX-NET(SMTP) では，情報ファイ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 対応版の場合 】

・ nectools32¥smp850e¥rxnet¥<*sample_name*>¥obj
　　Makefile　：ロード・モジュール用メイク・ファイル

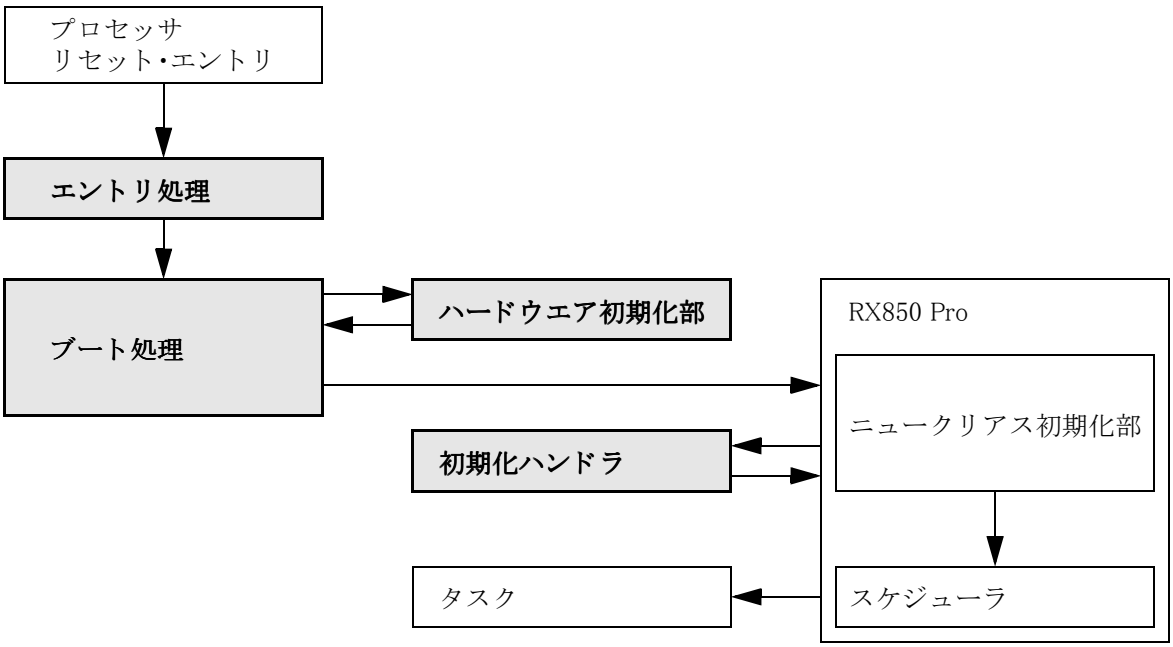
【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥obj_ghs
　　sample.bld ：ロード・モジュール用ビルド・ファイル

3.4　RX850 Pro 依存部の記述

RX-NET(TCP/IP) 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 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 を正常に動作させるうえで必要となる各種処理プログラム (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。

以下に，ブート処理で実行すべき処理を示します。

- テキスト・ポインタ TP の設定
- グローバル・ポインタ GP の設定
- エレメント・ポインタ EP の設定
- スタック・ポインタ SP の設定
- メモリ拡張モード・レジスタ MM の設定
- 初期値無しメモリ領域の初期化
- ハードウエア初期化部の呼び出し
- システム情報テーブルの先頭アドレスの設定
- ニュークリアス初期化部に制御を移す

なお，TP，GP，EP，SP の設定は，ブート処理の先頭で行う必要があります。

・**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。

以下に，ハードウエア初期化部で実行すべき処理を示します。

- 内部ユニットの初期化
- 周辺コントローラの初期化
- タイマ割り込みの許可
- ブート処理に制御を戻す

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 が時間管理用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 / 処理プログラ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。

以下に，初期化ハンドラで実行すべき処理を示します。

- 初期化データのコピー
- 時刻の初期化
- 評価ボード依存処理部(CPU ボード初期化部，Mother ボード初期化部)の呼び出し
- ニュークリアス初期化部に制御を戻す

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　RX850 Pro 依存部についての詳細は，「**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。

なお，RX-NET(SMTP) では，RX850 Pro 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¥rxnet¥<*sample_name*>¥src

entry.c	： エントリ処理(割り込みエントリ)
reset.s	： エントリ処理(リセット・エントリ)
boot.s	： ブート処理
inithw.c	： ハードウエア初期化部
inihdr.c	： 初期化ハンドラ

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥src

entry.850	： エントリ処理(割り込みエントリ)
reset.850	： エントリ処理(リセット・エントリ)
boot.850	： ブート処理
inithw.c	： ハードウエア初期化部
inihdr.c	： 初期化ハンドラ

3.5 RX-NET(TCP/IP) 依存部の記述

RX-NET(TCP/IP) では，RX-NET(TCP/IP) 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，ユーザの実行環境 / 処理プログラムに依存した初期化処理，および，RX-NET(TCP/IP) 用コンフィギュレーション情報については，RX-NET(TCP/IP) 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-NET(TCP/IP) 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-NET(TCP/IP) を正常に動作させるうえで必要となる各種処理プログラム (RX-NET(TCP/IP) 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-NET(TCP/IP) 依存部の一覧を示します。

- **評価ボード依存処理部**

  RX-NET(TCP/IP) では，CPU ボード初期化部，Mother ボード初期化部，I/O ポート操作処理については，ユーザ・オウン・コーディング部として切り出しています。

  以下に，評価ボード依存処理部のユーザ・オウン・コーディング部として切り出されている処理ルーチンの一覧を示します。

  - Mother ボード初期化部
  - I/O ポート操作処理
  - 周辺コントローラ操作処理

- **デバイス依存処理部，コンフィギュレーション情報依存処理部**

  RX-NET(TCP/IP) では，Ethernet コントローラ初期化部，RX-NET(TCP/IP) 用コンフィギュレーション情報については，ユーザ・オウン・コーディング部として切り出しています。

  以下に，デバイス依存処理部，コンフィギュレーション情報依存処理部のユーザ・オウン・コーディング部として切り出されている処理ルーチンの一覧を示します。

  - RX-NET(TCP/IP) 用タスク
  - RX-NET(TCP/IP) 用間接起動割り込みハンドラ
  - RX-NET(TCP/IP) 用周期起動ハンドラ
  - デバイス・ドライバ・エントリ・テーブル
  - デバイス・ドライバ API 関数
  - デバイス・ドライバ API 関数用内部関数
  - ディバグ用 printf 関数

注意　RX-NET(TCP/IP) 依存部のサンプル・ソース・ファイルについては，RX-NET(TCP/IP) が提供しています。

3.6 処理プログラムの記述

ネットワーク・アプリケーション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，RX850 Pro では，処理プログラムを用途別に以下のように分類 / 区別しています。

- **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に 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**

  一定の起動時間に達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，起動時間に達した際には，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　処理プログラムを記述するうえでの注意事項，および，記述形式についての詳細は，**「RX850 Pro ユーザーズ・マニュアル　基礎編」**，および，「**第 4 章　電子メール送信機能**」を参照してください。
　なお，RX-NET(SMTP) 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¥rxnet¥<*sample_name*>¥src
　task.c　　：タスク
　testsmtp.h ：処理プログラム用ヘッダ・ファイル

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥src
　task.c　　：タスク
　testsmtp.h ：処理プログラム用ヘッダ・ファイル

3.7　オブジェクト・ファイルの生成

「**3.2　CF 定義ファイルの記述**」～「**3.6　処理プログラムの記述**」で作成された C 言語ソース・ファイル / アセンブリ言語ソース・ファイルに対して C コンパイラ / アセンブラを実行し，リロケータブルなオブジェクト・ファイルを生成します。

注意　C コンパイラ / アセンブラの起動オプション，および，実行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。
　なお，RX-NET(SMTP) では，オブジェクト・ファイ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 対応版の場合 】

・ nectools32¥smp850e¥rxnet¥<*sample_name*>¥obj
　Makefile　：ロード・モジュール用メイク・ファイル

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥obj_ghs
　sample.bld ：ロード・モジュール用ビルド・ファイル

3.7.1　デバイス・ドライバ・オブジェクト

以下に示したファイル群に対してリンク・エディタを実行し，リロケータブルなオブジェクト・ファイル (デバイス・ドライバ・オブジェクト) を生成します。

・「**3.7　オブジェクト・ファイルの生成**」で作成されたリロケータブルなオブジェクト・ファイル
　- RX-NET(TCP/IP) 依存部
　　＊ デバイス依存処理部，コンフィギュレーション情報依存処理部
　　RX-NET(TCP/IP) 用タスク
　　RX-NET(TCP/IP) 用間接起動割り込みハンドラ
　　RX-NET(TCP/IP) 用周期起動ハンドラ
　　デバイス・ドライバ・エントリ・テーブル
　　デバイス・ドライバ API 関数
　　デバイス・ドライバ API 関数用内部関数
　　ディバグ用 printf 関数

注意　リンク・エディタの起動オプション，および，実行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。

なお，デバイス・ドライバ・オブジェクトを生成するためのサンプル・コマンド・ファイルについては，RX-NET(TCP/IP) が提供しています。

3.8 ライブラリ・ファイルの生成

3.8.1 BSP ライブラリ

以下に示したファイル群に対してアーカイバを実行し，ライブラリ・ファイル (BSP ライブラリ) を生成します。

・「**3.7 オブジェクト・ファイルの生成**」で作成されたリロケータブルなオブジェクト・ファイル
  - RX-NET(TCP/IP) 依存部
    * 評価ボード依存処理部
      Mother ボード初期化部
      I/O ポート操作処理
      周辺コントローラ操作処理

注意 アーカイバの起動オプション，および，実行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。
なお，BSP ライブラリを生成するためのサンプル・コマンド・ファイルについては，RX-NET(TCP/IP) が提供しています。

3.9 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。
以下に，割り付け先のセクション名が規定されている領域の一覧を示します。

・ **RX850 Pro 標準処理部 (.system セクション)**
RX850 Pro の本体処理 (タスク管理機能，タスク付属同期機能など)，および，CF 定義ファイルに対してコンフィギュレータ cf850pro を実行した際に生成されるシステム・コール・テーブルが割り付けられる領域です。
なお，.system セクションが必要とする領域のサイズは，内蔵命令 RAM のサイズよりも大きくなるため，.system セクションを内蔵命令 RAM に割り付けることはできません。

・ **RX850 Pro スケジューリング処理部 (.system_cmn セクション)**
RX850 Pro が提供するスケジューリング機能のうち，タスクの起床処理，および，タスクのスケジューリング処理が割り付けられる領域です。
したがって，.system_cmn セクションを内蔵命令 RAM に割り付けることにより，タスクの起床処理，および，タスクのスケジューリング処理が高速化される他に，スケジューリング処理を伴ったシステム・コールの処理も高速化されます。

・ **RX850 Pro 割り込み処理部 (.system_int セクション)**
RX850 Pro が提供する割り込み処理管理機能のうち，割り込みハンドラに制御を移す際に行われる割り込み前処理，および，割り込みの発生した処理プログラムに制御を戻す際に行われる割り込み後処理が割り付けられる領域です。
したがって，.system_int セクションを内蔵命令 RAM に割り付けることにより，割り込みハンドラに対する応答性が向上します。

・ **システム情報テーブル (.sit セクション)**
CF 定義ファイルに対してコンフィギュレータ cf850pro を実行した際に生成されるシステム情報テーブルが割り付けられる領域です。
なお，システム情報テーブルは，RX850 Pro のニュークリアス初期化部 (システム・メモリの確保，管理オブジェクトの生成 / 初期化など) を実行する際に必要となる各種データから構成されています。

・ **AZ850 予約領域 (.azwork セクション)**
システム・パフォーマンス・アナライザ AZ850 (NEC 製) がワーク・エリアとして使用する領域です。
なお，.azwork セクションの定義は，AZ850 使用の有無に関わらず必要となります。

・ **MULTI 予約領域 (.syscall セクション)**
ディバッガ MULTI (米国 Green Hills Software, Inc. 製) がワーク・エリアとして使用する領域です。

なお，.syscall セクションの定義は，MULTI 使用の有無に関わらず必要となります。

注意　.syscall セクションの定義を行う際には，4 バイト・アライン指定も併せて行う必要があります。

・コピー情報格納領域（.secinfo セクション）

リンク・ディレクティブ・ファイルで ROM 識別子の指定が行われているセクションのプログラム（テキスト，または，データ）を ROM から RAM に転送する際に必要となる情報（先頭アドレス，サイズ）をリンク・エディタが出力するための領域です。

なお，ROM 識別子の指定は，ロード・モジュールを ROM 化する際に必要となるものです。したがって，ROM 化を行わない場合には，.secinfo セクションの定義は不要となります。

注意　リンク・ディレクティブ・ファイル（別名称：セクション・マップ・ファイル）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。
なお，RX-NET(SMTP) 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¥rxnet¥<*sample_name*>¥src
　sample.dir　：リンク・ディレクティブ・ファイル

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥src
　sample.lx　：リンク・ディレクティブ・ファイル

3.10　ロード・モジュールの生成

以下に示したファイル群に対してリンク・エディタを実行し，ロード・モジュールを生成します。

・「**3.7　オブジェクト・ファイルの生成**」で作成されたリロケータブルなオブジェクト・ファイル
　- 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
　- RX850Pro 依存部
　　＊ エントリ処理
　　＊ ブート処理
　　＊ ハードウエア初期化部
　　＊ 初期化ハンドラ
　- 処理プログラム
　　＊ タスク
　　＊ 直接起動割り込みハンドラ
　　＊ 間接起動割り込みハンドラ
　　＊ 周期起動ハンドラ
　　＊ 拡張 SVC ハンドラ
　　＊ 拡張 SVC ハンドラ用インタフェース・ルーチン
　- デバイス・ドライバ・オブジェクト

・「**3.8　ライブラリ・ファイルの生成**」で作成されたライブラリ・ファイル
　- BSP ライブラリ

・ RX850 Pro が提供するライブラリ・ファイル
　- ニュークリアス共通部
　- ニュークリアス・ライブラリ
　- システム・コール用インタフェース・ライブラリ

・ RX-NET(TCP/IP) が提供するライブラリ・ファイル
　- TCP/IP ライブラリ
　- DHCP ダミー・エントリ・ライブラリ

・ RX-NET(SMTP) が提供するライブラリ・ファイル
　- SMTP ライブラリ

・ C コンパイラ・パッケージが提供すライブラリ・ファイル
　– ランタイム・ライブラリ（標準ライブラリ，数学ライブラリなど）

・「**3.9　リンク・ディレクティブ・ファイルの記述**」で作成されたリンク・ディレクティブ・ファイル

注意　リンク・エディタの起動オプション，および，実行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。
なお，RX-NET(SMTP) 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 対応版の場合 】

・ nectools32¥smp850e¥rxnet¥<*sample_name*>¥obj
　Makefile　：ロード・モジュール用メイク・ファイル

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*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　　電子メール送信機能

本章では，RX-NET(SMTP) が提供している電子メール送信機能について解説しています。

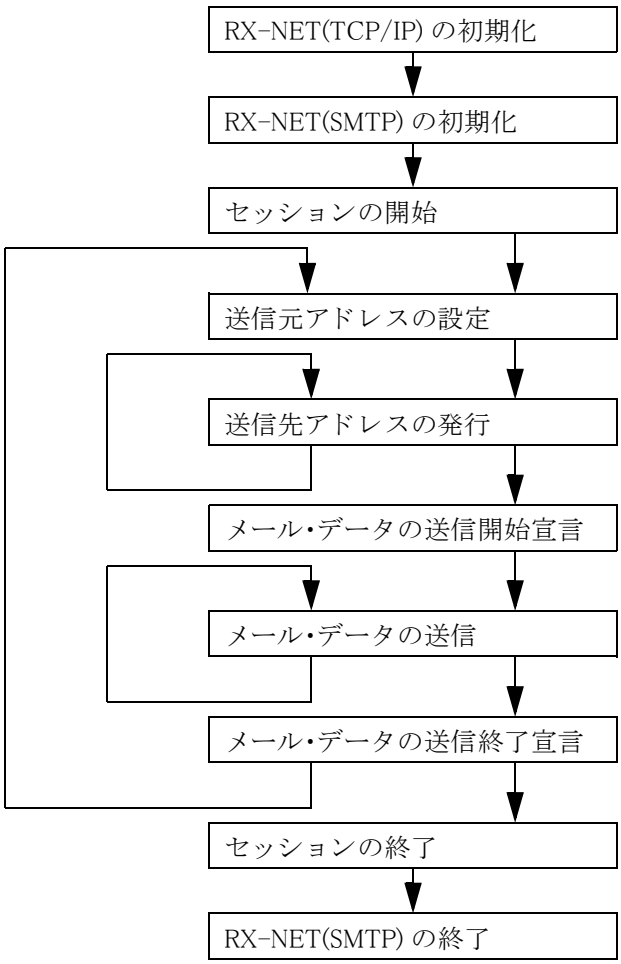
4.1　概要

RX-NET(SMTP) では，RX-NET(SMTP) の初期化処理 / 終了処理，および，電子メールの送信処理を“電子メール送信機能”として提供しています。

4.2　処理の流れ

RX-NET(SMTP) では，電子メールを送信する際に必要となる各種処理の実行順序を規定しています。
図4-1に,RX-NET(SMTP)が提供している電子メール送信機能を利用した電子メールの送信手順を示します。

図 4-1　電子メールの送信手順



注意 1　“1 通の電子メール”に対して複数の送信先が存在する場合には，送信先アドレスの設定 (API 関数 smtp_addToAddress の発行) を送信先の総数分だけ行う必要があります。

注意 2　“1 通の電子メール”に対して複数行のメール・データが存在する場合には，メール・データの送信 (API 関数 smtp_sendMailData の発行) をメール・データの総行数分だけ行う必要があります。

注意 3　“複数の電子メール”を送信する場合には，上記電子メールの送信手順“送信元アドレスの設定”～“メール・データの送信終了宣言”の処理を繰り返し行うことにより実現されます。

4.3 電子メール送信機能 API 関数

4.3.1 RX-NET(SMTP) の初期化 / 終了

RX-NET(SMTP) 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。

- **smtp_start**

  RX-NET(SMTP) 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。
  以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;  /* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
```

  注意 1　本 API 関数の発行は，RX-NET(TCP/IP) が提供する API 関数 so_initialize，ll_config の処理完了後に行う必要があります。

  注意 2　RX-NET(TCP/IP) が提供する API 関数 ll_config 発行時に指定可能なネットワーク・デバイス名 *devname* は，あらかじめデバイス・ドライバ・エントリ・テーブル ndevsw [ ] に登録されているものに限られます。

- **smtp_end**

  RX-NET(SMTP) の終了処理を実行します。
  これにより，すべての処理プログラムから RX-NET(SMTP) が提供す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。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;  /* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */

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

smtp_end ( ) ;                                      /* RX-NET(SMTP) の終了 */
```

4.3.2　セッションの開始 / 終了

セッションの開始 / 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。

- smtp_connect

パラメータ *ipaddr*，*port* で指定されたメール・サーバ (receive-SMTP 機能が動作しているホスト・マシン) とのセッションを開始します。

なお，RX-NET(SMTP) では，セッションの開始処理として，メール・サーバに対する HELO コマンド (パラメータ *domainname* で指定された RX-NET(SMTP) のドメイン名) の送信を行っています。

以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr ( "10.30.178.84" ) ;     /* 変数の宣言，初期化 */
u16     port = htons ( 25 ) ;                       /* 変数の宣言，初期化 */
char    *domainname = “sample1.nec.co.jp" ;         /* 変数の宣言，初期化 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;/* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;         /* セッションの開始 */
```

注意　SMTP 用ポート番号 *port* には，通常，25 番を設定します。

- smtp_disconnect

API 関数 smtp_connect 発行時に指定したメール・サーバとのセッションを終了します。

なお，RX-NET(SMTP) では，セッションの終了処理として，メール・サーバに対する QUIT コマンドの送信を行っています。

以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr ( "10.30.178.84" ) ;     /* 変数の宣言，初期化 */
u16     port = htons ( 25 ) ;                       /* 変数の宣言，初期化 */
char    *domainname = “sample1.nec.co.jp" ;         /* 変数の宣言，初期化 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;/* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;         /* セッションの開始 */
```

```
……
……
……

smtp_disconnect ( ) ;                                   /* セッションの終了 */
```

4.3.3 送信元アドレス / 送信先アドレスの設定

送信元アドレス / 送信先アドレスの設定は，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。

- smtp_setFromAddress

  API 関数 smtp_connect 発行時に指定したメール・サーバに対して MAIL コマンド (パラメータ *mailname* で指定された送信元アドレス) の送信を行います。

  以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr ( "10.30.178.84" ) ;     /* 変数の宣言，初期化 */
u16     port = htons ( 25 ) ;                       /* 変数の宣言，初期化 */
char    *domainname = "sample1.nec.co.jp" ;         /* 変数の宣言，初期化 */
char    *mailname ;                                 /* 変数の宣言 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;/* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;         /* セッションの開始 */

mailname = "myaddress@sample1.nec.co.jp" ;          /* 変数の初期化 */

smtp_setFromAddress () ;                            /* 送信元アドレスの設定 */
```

注意　RX-NET(SMTP) では，本 API 関数の発行から API 関数 smtp_endMailData が発行されるまでに得られた情報 (送信元アドレス，送信先アドレス，メール・データなど) を“1 通の電子メール”として扱っています。
なお，RX-NET(SMTP) では，本 API 関数の発行回数を“1 通の電子メール”に対して 1 回に限定しています。

- smtp_addToAddress

  API 関数 smtp_connect 発行時に指定したメール・サーバに対して RCPT コマンド (パラメータ *mailname* で指定された送信先アドレス) の送信を行います。

  以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */
```

```
char    devname[] = "s91c";                          /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr("10.30.178.84");       /* 変数の宣言，初期化 */
u32     ipmask = inet_addr("255.255.255.0");         /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr("10.30.178.84");          /* 変数の宣言，初期化 */
u16     port = htons(25);                            /* 変数の宣言，初期化 */
char    *domainname = "sample1.nec.co.jp";           /* 変数の宣言，初期化 */
char    *mailname;                                   /* 変数の宣言 */

so_initialize();                                     /* RX-NET(TCP/IP) の初期化 */
ll_config(devname, ipaddress, ipmask, 0x0, 0x0);     /* ネットワーク・インタフェースの起動 */
smtp_start();                                        /* RX-NET(SMTP) の初期化 */
smtp_connect(ipaddr, port, domainname);              /* セッションの開始 */

mailname = "myaddress@sample1.nec.co.jp";            /* 変数の初期化 */

smtp_setFromAddress(mailname);                       /* 送信元アドレスの設定 */

mailname = "youraddress@sample1.nec.co.jp";          /* 変数の初期化 */

smtp_addToAddress(mailname);                         /* 送信先アドレスの設定 */
```

注意　“1 通の電子メール”に対して複数の送信先が存在する場合には，本 API 関数を送信先の総数分だけ発行する必要があります。

4.3.4　メール・データの送信開始宣言 / 送信終了宣言

メール・データの送信開始宣言 / 送信終了宣言は，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。

- **smtp_startMailData**

  API 関数 smtp_connect 発行時に指定したメール・サーバに対して“メール・データの送信開始宣言”を行います。

  なお，RX-NET(SMTP) では，メール・データの送信開始宣言処理として，メール・サーバに対する DATA コマンドの送信を行っています。

  以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[] = "s91c";                          /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr("10.30.178.84");       /* 変数の宣言，初期化 */
u32     ipmask = inet_addr("255.255.255.0");         /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr("10.30.178.84");          /* 変数の宣言，初期化 */
u16     port = htons(25);                            /* 変数の宣言，初期化 */
char    *domainname = "sample1.nec.co.jp";           /* 変数の宣言，初期化 */
char    *mailname;                                   /* 変数の宣言 */

so_initialize();                                     /* RX-NET(TCP/IP) の初期化 */
ll_config(devname, ipaddress, ipmask, 0x0, 0x0);     /* ネットワーク・インタフェースの起動 */
smtp_start();                                        /* RX-NET(SMTP) の初期化 */
```

```
smtp_connect ( ipaddr, port, domainname ) ;          /* セッションの開始 */

mailname = "myaddress@sample1.nec.co.jp" ;           /* 変数の初期化 */

smtp_setFromAddress ( mailname ) ;                   /* 送信元アドレスの設定 */

mailname = "youraddress@sample1.nec.co.jp" ;         /* 変数の初期化 */

smtp_addToAddress ( mailname ) ;                     /* 送信先アドレスの設定 */
smtp_startMailData ( ) ;                             /* メール・データの送信開始宣言 */
```

注意　RX-NET(SMTP) では, 本 API 関数の発行から API 関数 smtp_endMailData が発行されるまでに得られた情報 (メール・データ) を“電子メールの本文”として扱っています。

・ **smtp_endMailData**

API 関数 smtp_connect 発行時に指定したメール・サーバに対して“メール・データの送信終了宣言”を行います。

なお, RX-NET(SMTP) では, メール・データの送信終了宣言処理として, メール・サーバに対するメール・データ (ピリオド + 改行文字 “.<CR><LF>”) の送信を行っています。

以下に, 本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname [ ] = "s91c" ;                       /* 変数の宣言, 初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;   /* 変数の宣言, 初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;     /* 変数の宣言, 初期化 */
u32     ipaddr = inet_addr ( "10.30.178.84" ) ;      /* 変数の宣言, 初期化 */
u16     port = htons ( 25 ) ;                        /* 変数の宣言, 初期化 */
char    *domainname = “sample1.nec.co.jp" ;          /* 変数の宣言, 初期化 */
char    *mailname ;                                  /* 変数の宣言 */

so_initialize ( ) ;                                  /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ; /* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                     /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;          /* セッションの開始 */

mailname = "myaddress@sample1.nec.co.jp" ;           /* 変数の初期化 */

smtp_setFromAddress ( mailname ) ;                   /* 送信元アドレスの設定 */

mailname = "youraddress@sample1.nec.co.jp" ;         /* 変数の初期化 */

smtp_addToAddress ( mailname ) ;                     /* 送信先アドレスの設定 */
smtp_startMailData ( ) ;                             /* メール・データの送信開始宣言 */

.....
.....
.....
```

```
smtp_endMailData ( ) ;                                  /* メール・データの送信終了宣言 */
```

注意　RX-NET(SMTP) では，API 関数 smtp_startMailData の発行から本 API 関数 smtp_endMailData が発行されるまでに得られた情報 (メール・データ) を“電子メールの本文”として扱っています。

4.3.5　メール・データの送信

メール・データの送信は，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。

- **smtp_sendMailData**

  API 関数 smtp_connect 発行時に指定したメール・サーバに対してパラメータ *line* で指定されたメール・データ (1 行分) の送信を行います。

  なお，パラメータ *line_len* には，パラメータ *line* で指定されたメール・データのサイズ (単位：バイト) を指定します。

  以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

char    devname[ ] = "s91c" ;                       /* 変数の宣言，初期化 */
u32     ipaddress = inet_addr ( "10.30.178.84" ) ;  /* 変数の宣言，初期化 */
u32     ipmask = inet_addr ( "255.255.255.0" ) ;    /* 変数の宣言，初期化 */
u32     ipaddr = inet_addr ( "10.30.178.84" ) ;     /* 変数の宣言，初期化 */
u16     port = htons ( 25 ) ;                       /* 変数の宣言，初期化 */
char    *domainname = “sample1.nec.co.jp" ;         /* 変数の宣言，初期化 */
char    *mailname ;                                 /* 変数の宣言 */
char    *line ;                                     /* 変数の宣言 */
int     line_len ;                                  /* 変数の宣言 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;/* ネットワーク・インタフェースの起動 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;         /* セッションの開始 */

mailname = "myaddress@sample1.nec.co.jp" ;          /* 変数の初期化 */

smtp_setFromAddress ( mailname ) ;                  /* 送信元アドレスの設定 */

mailname = "youraddress@sample1.nec.co.jp" ;        /* 変数の初期化 */

smtp_addToAddress ( mailname ) ;                    /* 送信先アドレスの設定 */
smtp_startMailData ( ) ;                            /* メール・データの送信開始宣言 */

line = "Subject : Test Mail" ;                      /* 変数の初期化 */
line_len = strlen ( line ) ;                        /* 変数の初期化 */

smtp_sendMailData ( line, line_len ) ;              /* メール・データの送信 */
```

注意 1　本 API 関数では，パラメータ *line* で指定されたメール・データの末尾に改行文字“<CR><LF>”を付加したものをメール・サーバに対して送信しています。

注意 2　本 API 関数では，パラメータ *line* で指定されたメール・データの先頭文字がピリオド“.”であった場合には，メール・データの先頭にピリオド“.”を付加したものをメール・サーバに対して送信しています。

注意 3　“1 通の電子メール”に対して複数行のメール・データが存在する場合には，本 API 関数をメール・データの総行数分だけ発行する必要があります。

注意 4　RX-NET(SMTP) では，メール・ヘッダの付加処理を行っていません。したがって，Subject，To 等のメール・ヘッダを送信する場合には，Subject，To 等を本 API 関数のメール・データとして指定する必要があります。

4.3.6　エラー情報の獲得

エラー情報の獲得は，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。

- smtp_getErrorLine

本 API 関数の直前に発行された API 関数に対応したエラー情報をパラメータ *line* で指定された領域に格納します。

なお，パラメータ *line_len* には，パラメータ *line* で指定された領域のサイズ (単位:バイト) を指定します。

以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                    /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>                /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>               /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

u32     ipaddr = inet_addr ( "10.30.178.84" ) ;     /* 変数の宣言，初期化 */
u16     port = htons ( 25 ) ;                       /* 変数の宣言，初期化 */
char    *domainname = "sample1.nec.co.jp" ;         /* 変数の宣言，初期化 */
char    line [ 1000 ] ;                             /* 変数の宣言 */
int     line_len = 1000 ;                           /* 変数の宣言，初期化 */

so_initialize ( ) ;                                 /* RX-NET(TCP/IP) の初期化 */
smtp_start ( ) ;                                    /* RX-NET(SMTP) の初期化 */
smtp_connect ( ipaddr, port, domainname ) ;         /* セッションの開始 */
smtp_getErrorLine ( &line, line_len ) ;             /* エラー情報の獲得 */
```

注意 1　エラー情報とは，API 関数 (smtp_connect，smtp_disconnect，smtp_setFromAddress など) の発行により送信されたコマンド等に対するメール・サーバからの応答文字列 (エラー・コードを含む) を意味しています。

注意 2　本 API 関数の直前に発行された API 関数が異常終了していた場合，パラメータ *line* で指定された領域の内容は不定となります。

4.3.7　セッションの強制終了

セッションの強制終了は，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。

- smtp_abort

本API関数を発行した際，処理の完了していないAPI関数(smtp_connect，smtp_disconnect，smtp_setFromAddress など) を強制的に異常終了させます。

なお，本API関数の発行により異常終了したAPI関数には，戻り値としてESMTP_CONNFAIL，ESMTP_RECVFAIL，ESMTP_SENDTMDOUT などが返されます。

以下に，本 API 関数の記述例を示します。

```
#include <rxnet.h>                  /* RX-NET(TCP/IP) 用標準ヘッダ・ファイルの定義 */
#include <fnsconfig.h>              /* 静的設定情報ヘッダ・ファイルの定義 */
#include <rxnet_smtp.h>             /* RX-NET(SMTP) 用標準ヘッダ・ファイルの定義 */

smtp_abort ( ) ;                                    /* ネットワーク接続の強制遮断 */
```

注意　本 API 関数では，API 関数の強制終了処理の他に，メール・サーバとのセッションの強制終了処理 (API 関数 smtp_disconnect 相当の処理) も併せて行われます。このため，再度，電子メールの送信を行う場合には，API 関数 smtp_connect の発行が必要となります。

第 5 章　　API 関数

本章では，RX-NET(SMTP) 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-NET(SMTP) 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-NET(SMTP) 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-NET(SMTP) が提供している API 関数を示します。

smtp_start	smtp_end	smtp_connect	smtp_disconnect	smtp_setFromAddress
smtp_addToAddress	smtp_startMailData	smtp_sendMailData	smtp_endMailData	smtp_getErrorLine
smtp_abort				

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-NET(SMTP) 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-NET(SMTP) に引き数として渡され，該当処理が実行されます。

注意　RX-NET(SMTP) 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。

```
rxnet.h       : RX-NET(TCP/IP) 用標準ヘッダ・ファイル
fnsconfig.h   : 静的設定情報ヘッダ・ファイル
rxnet_smtp.h  : RX-NET(SMTP) 用標準ヘッダ・ファイル
```

なお，rxnet.h，fnsconfig.h は，RX-NET(TCP/IP) が提供しています。

5.3　データ・マクロ

RX-NET(SMTP) 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，戻り値など) 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¥rxnet.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥ccdep.h で行われています。

表 5-1　データ・タイプ

マクロ	型	意味
u8	unsigned char	汎用 8 ビット整数
u16	unsigned short	汎用 16 ビット整数
u32	unsigned int	汎用 32 ビット整数

注意　標準ヘッダ・ファイル nectools32¥inc850¥rxnet.h, および, ヘッダ・ファイル nectools32¥inc850¥rxnet¥ccdep.h は，RX-NET(TCP/IP) が提供しています。

5.3.2　バイト順反転用条件付きマクロ

表 5-2 に，RX-NET(TCP/IP) が提供しているバイト順反転用条件付きマクロ一覧を示します。
なお，バイト順反転用条件付きマクロのマクロ定義は，標準ヘッダ・ファイル nectools32¥inc850¥rxnet.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥bsd_in.h で行われています。

表 5-2　バイト順反転用条件付きマクロ

マクロ	意味
htons (a)	ホスト・バイト・オーダの 16 ビット・データ“a”をネットワーク・バイト・オーダの 16 ビット・データに変換
ntohs (a)	ネットワーク・バイト・オーダの 16 ビット・データ“a”をホスト・バイト・オーダの 16 ビット・データに変換
htonl (a)	ホスト・バイト・オーダの 32 ビット・データ“a”をネットワーク・バイト・オーダの 32 ビット・データに変換
ntohl (a)	ネットワーク・バイト・オーダの 32 ビット・データ“a”をホスト・バイト・オーダの 32 ビット・データに変換

注意　標準ヘッダ・ファイル nectools32¥inc850¥rxnet.h, および, ヘッダ・ファイル nectools32¥inc850¥rxnet¥ccdep.h は，RX-NET(TCP/IP) が提供しています。

5.3.3　戻り値

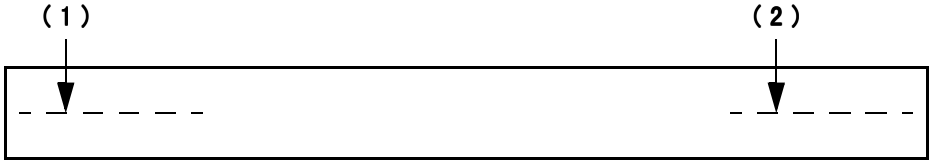
表 5-3 に，API 関数からの戻り値一覧を示します。
なお，戻り値のマクロ定義は，標準ヘッダ・ファイル nectools32¥inc850¥rxnet_smtp.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥smtperrno.h で行われています。

表 5-3　戻り値

マクロ	数値	意味
ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です
ESMTP_RUNNING	0x1101	既に smtp_start が発行されています
ESMTP_DOWN	0x1102	smtp_start が発行されていません
ESMTP_CRESEM	0x1103	RX-NET(SMTP)が排他制御を行う際に用いるセマフォの生成処理が失敗しました
ESMTP_DELSEM	0x1104	RX-NET(SMTP)が排他制御を行う際に用いるセマフォの削除処理が失敗しました
ESMTP_WAISEM	0x1105	RX-NET(SMTP) 用セマフォが削除されています
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理を実行することができません
ESMTP_CONNECT	0x1107	既に smtp_connect が発行されています
ESMTP_NOTCONN	0x1108	smtp_connect が発行されていません
ESMTP_SOCKET	0x1109	RX-NET(SMTP)がネットワーク接続を行う際に用いるソケットの生成処理が失敗しました
ESMTP_CONNFAIL	0x110a	ネットワーク接続の確立処理が失敗しました
ESMTP_SENDFAIL	0x110b	メール・データの送信処理が失敗しました
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました
ESMTP_BADCODE	0x110d	送信したコマンド (HELO，QUIT，MAIL など) に対するメール・サーバからの応答がエラーです
ESMTP_DATAMODE	0x110e	DATA コマンドに対応した処理を完了していないため，該当処理を実行することができません
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています
ESMTP_LONGSTR	0x1111	最大文字数を越えています
ESMTP_SENDTMDOUT	0x1112	メール・データの送信処理が失敗しました

5.4 API 関数解説

次項から RX-NET(SMTP) 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。



（3）→ 概要

（4）→ C 言語形式

（5）→ パラメータ

I/O	パラメータ	説　　明

（6）→ 機能

（7）→ 戻り値

（1） 名称

API 関数の名称を示しています。

（2） 発行有効範囲

API 関数の発行が可能な処理プログラムの種別を示しています。

タスク　　：タスクからのみ発行可能
非タスク　　：非タスクからのみ発行可能
タスク / 非タスク　：タスク，非タスクのどちらからも発行可能

（3） 概要

API 関数の機能概要を示しています。

（4） 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。

（5） 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-NET(SMTP) への入力パラメータ
O … RX-NET(SMTP) 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

（6） 機能

API 関数の機能詳細を示しています。

（7） 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。

5.4.1 外部インタフェース仕様

表 5-4 に，RX-NET(SMTP) が提供している API 関数の一覧を示します。

表 5-4 RX-NET(SMTP) の API 関数

API 関数名	機能概要
smtp_start	RX-NET(SMTP) の初期化
smtp_end	RX-NET(SMTP) の終了
smtp_connect	セッションの開始 (HELO コマンドの送信)
smtp_disconnect	セッションの終了 (QUIT コマンドの送信)
smtp_setFromAddress	送信元アドレスの設定 (MAIL コマンドの送信)
smtp_addToAddress	送信先アドレスの設定 (RCPT コマンドの送信)
smtp_startMailData	メール・データの送信開始宣言 (DATA コマンドの送信)
smtp_sendMailData	メール・データの送信
smtp_endMailData	メール・データの送信終了宣言 (“.<CR><LF>” の送信)
smtp_getErrorLine	エラー情報の獲得
smtp_abort	セッションの強制終了

次頁以降に，各種 API 関数の外部インタフェース仕様詳細を示します。

smtp_start

タスク

概要

RX-NET(SMTP) の初期化

C 言語形式

```
int             smtp_start ( void ) ;
```

パラメータ

なし

機能

RX-NET(SMTP) 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。

注意　本 API 関数の発行は，RX-NET(TCP/IP) が提供する API 関数 so_initialize，ll_config の処理完了後に行う必要があります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_RUNNING	0x1101	既に smtp_start が発行されています
ESMTP_CRESEM	0x1103	RX-NET(SMTP) が排他制御を行う際に用いるセマフォの生成処理が失敗しました

smtp_end

タスク

概要

RX-NET(SMTP) の終了

C 言語形式

```
int             smtp_end ( void ) ;
```

パラメータ

なし

機能

RX-NET(SMTP) の終了処理を実行します。
これにより，すべての処理プログラムから RX-NET(SMTP) が提供する機能を利用することができなくなります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_DOWN	0x1102	smtp_start が発行されていません
ESMTP_DELSEM	0x1104	RX-NET(SMTP) が排他制御を行う際に用いるセマフォの削除処理が失敗しました
ESMTP_APIRUNNING	0x1106	他のAPI関数が処理を完了していないため,該当処理“RX-NET(SMTP)の終了処理”を実行することができません
ESMTP_CONNECT	0x1107	smtp_disconnect が発行されていません

smtp_connect

タスク

概要

セッションの開始 (HELO コマンドの送信)

C 言語形式

int　　smtp_connect (u32 *ipaddr* , u16 *port* , char **domainname*) ;

パラメータ

I/O	パラメータ	説　　明
I	u32　　*ipaddr* ;	メール・サーバの IP アドレス (ネットワーク・バイト・オーダ)
I	u16　　*port* ;	メール･サーバの SMTP 用ポート番号 (ネットワーク･バイト・オーダ)
I	char　　**domainname* ;	RX-NET(SMTP)のドメイン名を格納した領域へのポインタ

機能

ipaddr, *port* で指定されたメール・サーバ (receive-SMTP 機能が動作しているホスト・マシン) とのセッションを開始します。

なお, RX-NET(SMTP) では, セッションの開始処理として, メール･サーバに対する HELO コマンド (*domainname* で指定された RX-NET(SMTP) のドメイン名) の送信を行っています。

注意　SMTP 用ポート番号 *port* には，通常，25 番を設定します。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です - IP アドレスの指定が不正 (*ipaddr* = 0x0) です - SMTP 用ポート番号の指定が不正 (*port* = 0x0) です - ドメイン名を格納した領域の指定が不正 (NULL ポインタ) です - ドメイン名の文字総数が不正 (0 文字) です
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“セッションの開始処理”を実行することができません
ESMTP_CONNECT	0x1107	既に smtp_connect が発行されています
ESMTP_SOCKET	0x1109	RX-NET(SMTP) がネットワーク接続を行う際に用いるソケットの生成処理が失敗しました
ESMTP_CONNFAIL	0x110a	ネットワーク接続の確立処理が失敗しました - ネットワーク接続の確立処理がタイムアウトしました - ネットワーク接続の確立処理を実行中に他タスクから smtp_abort が発行されました - IP アドレス *ipaddr* で指定されたメール・サーバの SMTP 用ポート番号は *port* で指定された値ではありません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました

		– HELO コマンドに対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – HELO コマンドに対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	HELO コマンドに対するメール・サーバからの応答がエラーです
ESMTP_LONGSTR	0x1111	ドメイン名の文字総数が最大文字数 (64 文字) を越えています

smtp_disconnect

タスク

概要

セッションの終了 (QUIT コマンドの送信)

C 言語形式

```
int             smtp_disconnect ( void ) ;
```

パラメータ

なし

機能

API 関数 smtp_connect 発行時に指定したメール・サーバとのセッションを終了します。
なお，RX-NET(SMTP) では，セッションの終了処理として，メール・サーバに対する QUIT コマンドの送信を行っています。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“セッションの終了処理”を実行することができません
ESMTP_NOTCONN	0x1108	smtp_connect が発行されていません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました – QUIT コマンドに対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – QUIT コマンドに対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	QUIT コマンドに対するメール・サーバからの応答がエラーです
ESMTP_DATAMODE	0x110e	DATA コマンドに対応した処理を完了していないため，該当処理“セッションの終了処理”を実行することができません

smtp_setFromAddress

タスク

概要

送信元アドレスの設定 (MAIL コマンドの送信)

C 言語形式

int　　smtp_setFromAddress (char *mailname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*mailname ;	送信元アドレスを格納した領域へのポインタ

機能

API 関数 smtp_connect 発行時に指定したメール・サーバに対して MAIL コマンド (*mailname* で指定された送信元アドレス) の送信を行います。

注意　RX-NET(SMTP) では, 本 API 関数の発行から API 関数 smtp_endMailData が発行されるまでに得られた情報 (送信元アドレス，送信先アドレス，メール・データなど) を“1 通の電子メール”として扱っています。
なお，RX-NET(SMTP) では，本 API 関数の発行回数を“1 通の電子メール”に対して 1 回に限定しています。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です – 送信元アドレスを格納した領域の指定が不正 (NULL ポインタ) です – 送信元アドレスの文字総数が不正 (0 文字) です
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“MAIL コマンドの送信処理”を実行することができません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました – MAIL コマンドに対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – MAIL コマンドに対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	MAIL コマンドに対するメール・サーバからの応答がエラーです
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています
ESMTP_LONGSTR	0x1111	送信元アドレスの文字総数が最大文字数 (64 文字) を越えています

smtp_addToAddress

タスク

概要

送信先アドレスの設定 (RCPT コマンドの送信)

C 言語形式

int　　smtp_addToAddress (char *mailname) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*mailname ;	送信先アドレスを格納した領域へのポインタ

機能

API 関数 smtp_connect 発行時に指定したメール・サーバに対して RCPT コマンド (*mailname* で指定された送信先アドレス) の送信を行います。

注意　“1 通の電子メール”に対して複数の送信先が存在する場合には，本 API 関数を送信先の総数分だけ発行する必要があります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です – 送信先アドレスを格納した領域の指定が不正 (NULL ポインタ) です – 送信先アドレスの文字総数が不正 (0 文字) です
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“RCPT コマンドの送信処理”を実行することができません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました – RCPT コマンドに対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – RCPT コマンドに対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	RCPT コマンドに対するメール・サーバからの応答がエラーです
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています
ESMTP_LONGSTR	0x1111	送信先アドレスの文字総数が最大文字数 (64 文字) を越えています

smtp_startMailData

タスク

概要

メール・データの送信開始宣言 (DATA コマンドの送信)

C 言語形式

```
int             smtp_startMailData ( void ) ;
```

パラメータ

なし

機能

API 関数 smtp_connect 発行時に指定したメール・サーバに対して“メール・データの送信開始宣言”を行います。

なお，RX-NET(SMTP) では，メール・データの送信開始宣言処理として，メール・サーバに対する DATA コマンドの送信を行っています。

注意　RX-NET(SMTP) では, 本 API 関数の発行から API 関数 smtp_endMailData が発行されるまでに得られた情報 (メール・データ) を“電子メールの本文”として扱っています。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“メール・データの送信開始宣言処理”を実行することができません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました – DATA コマンドに対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – DATA コマンドに対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	DATA コマンドに対するメール・サーバからの応答がエラーです
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています

smtp_sendMailData

タスク

概要

メール・データの送信

C 言語形式

int smtp_sendMailData (char **line* , int *line_len*) ;

パラメータ

I/O	パラメータ	説　　明
I	char **line* ;	メール・データを格納した領域へのポインタ
I	int *line_len* ;	*line* で指定されたメール・データのサイズ (単位 : バイト)

機能

API 関数 smtp_connect 発行時に指定したメール・サーバに対して *line* で指定されたメール・データ (1 行分) の送信を行います。

なお，*line_len* には，*line* で指定されたメール・データのサイズ (単位 : バイト) を指定します。

注意 1　本 API 関数では，*line* で指定されたメール・データの末尾に改行文字“<CR><LF>”を付加したものをメール・サーバに対して送信しています。

注意 2　本 API 関数では，*line* で指定されたメール・データの先頭文字がピリオド“.”であった場合には，メール・データの先頭にピリオド“.”を付加したものをメール・サーバに対して送信しています。

注意 3　“1 通の電子メール”に対して複数行のメール・データが存在する場合には，本 API 関数をメール・データの総行数分だけ発行する必要があります。

注意 4　RX-NET(SMTP) では，メール・ヘッダの付加処理を行っていません。したがって，Subject，To 等のメール・ヘッダを送信する場合には，Subject，To 等を本 API 関数のメール・データとして指定する必要があります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です – メール･データを格納した領域の指定が不正(NULLポインタ)です – サイズ *line_len* の指定が不正 (負数) です
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“メール・データの送信処理”を実行することができません
ESMTP_SENDFAIL	0x110b	メール・データの送信処理が失敗しました – メール・データの送信処理が完了する以前に，ネットワーク接続がメール・サーバ側から遮断されました
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています
ESMTP_LONGSTR	0x1111	メール・データの文字総数が最大文字数 (998 文字) を越えています
ESMTP_SENDTMDOUT	0x1112	メール・データの送信処理が失敗しました

- メール・データの送信処理が完了する以前に,他タスクから smtp_abort が発行されました

smtp_endMailData

タスク

概要

メール・データの送信終了宣言 (“.<CR><LF>” の送信)

C 言語形式

```
int             smtp_endMailData ( void ) ;
```

パラメータ

なし

機能

API 関数 smtp_connect 発行時に指定したメール・サーバに対して“メール・データの送信終了宣言”を行います。

なお，RX-NET(SMTP) では，メール・データの送信終了宣言処理として，メール・サーバに対するメール・データ (ピリオド + 改行文字 “.<CR><LF>”) の送信を行っています。

注意　RX-NET(SMTP) では，API 関数 smtp_startMailData の発行から本 API 関数 smtp_endMailData が発行されるまでに得られた情報 (メール・データ) を“電子メールの本文”として扱っています。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“メール・データの送信終了宣言処理”を実行することができません
ESMTP_RECVFAIL	0x110c	エラー情報の受信処理が失敗しました – メール・データ “.<CR><LF>” に対するメール・サーバからの応答を待っている際に他タスクから smtp_abort が発行されました – メール・データ “.<CR><LF>” に対するメール・サーバからの応答を待っている際にネットワーク接続がメール・サーバ側から遮断されました
ESMTP_BADCODE	0x110d	メール・データ “.<CR><LF>” に対するメール・サーバからの応答がエラーです
ESMTP_TOOEARLY	0x110f	API 関数の発行順序が不正です
ESMTP_TOOLATE	0x1110	API 関数の発行が重複しています

smtp_getErrorLine

タスク

概要

エラー情報の獲得

C 言語形式

int　　smtp_getErrorLine (char *_line_ , int _line_len_) ;

パラメータ

I/O	パラメータ	説　　明
O	char　*_line_ ;	エラー情報を格納する領域へのポインタ
I	int　_line_len_ ;	_line_ で指定された領域のサイズ (単位：バイト)

機能

本 API 関数の直前に発行された API 関数に対応したエラー情報を _line_ で指定された領域に格納します。
なお，_line_len_ には，_line_ で指定された領域のサイズ (単位：バイト) を指定します。

注意 1　エラー情報とは，API 関数 (smtp_connect，smtp_disconnect，smtp_setFromAddress など) の発行により送信されたコマンド等に対するメール・サーバからの応答文字列 (エラー・コードを含む) を意味しています。

注意 2　本 API 関数の直前に発行された API 関数が異常終了していた場合,パラメータ _line_ で指定された領域の内容は不定となります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_INVAL	0x1100	パラメータの指定が不正です － エラー情報を格納する領域の指定が不正 (NULL ポインタ) です － サイズ _line_len_ の指定が不正 (0x0 以下の値) です
ESMTP_APIRUNNING	0x1106	他の API 関数が処理を完了していないため，該当処理“エラー情報の獲得処理”を実行することができません

smtp_abort

タスク

概要

セッションの強制終了

C 言語形式

```
int             smtp_abort ( void ) ;
```

パラメータ

なし

機能

本API関数を発行した際, 処理の完了していないAPI関数(smtp_connect, smtp_disconnect, smtp_setFromAddress など) を強制的に異常終了させます。

なお，本 API 関数の発行により異常終了した API 関数には，戻り値として ESMTP_CONNFAIL，ESMTP_RECVFAIL，ESMTP_SENDTMDOUT などが返されます。

注意　本 API 関数では，API 関数の強制終了処理の他に，メール・サーバとのセッションの強制終了処理 (API 関数 smtp_disconnect 相当の処理) も併せて行われます。このため，再度，電子メールの送信を行う場合には，API 関数 smtp_connect の発行が必要となります。

戻り値

ESMTP_NOERR	0x0	正常終了
ESMTP_WAISEM	0x1105	RX-NET(SMTP) 用セマフォが削除されています

索引

い

お

か

し

せ

て

は

ひ

め

も

り

ろ