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

POP

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　U15539JJ2V0UM00（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 5 月 17 日	RX-NET(POP) V1.10 対応 新規作成
2.00	2001 年 8 月 24 日	RX-NET(POP) V1.20 対応 改版 **1.1　概要** 「図 1-1　RX-NET(POP) 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 POP3”，および，“リアルタイム OS”を削除。 **1.2　特徴** 「図 1-2　RX-NET(POP) 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(POP) 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¥pop¥disk1”から“RX-NET_POP_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** 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。

目次

図目次

表目次

第 1 章　　概説

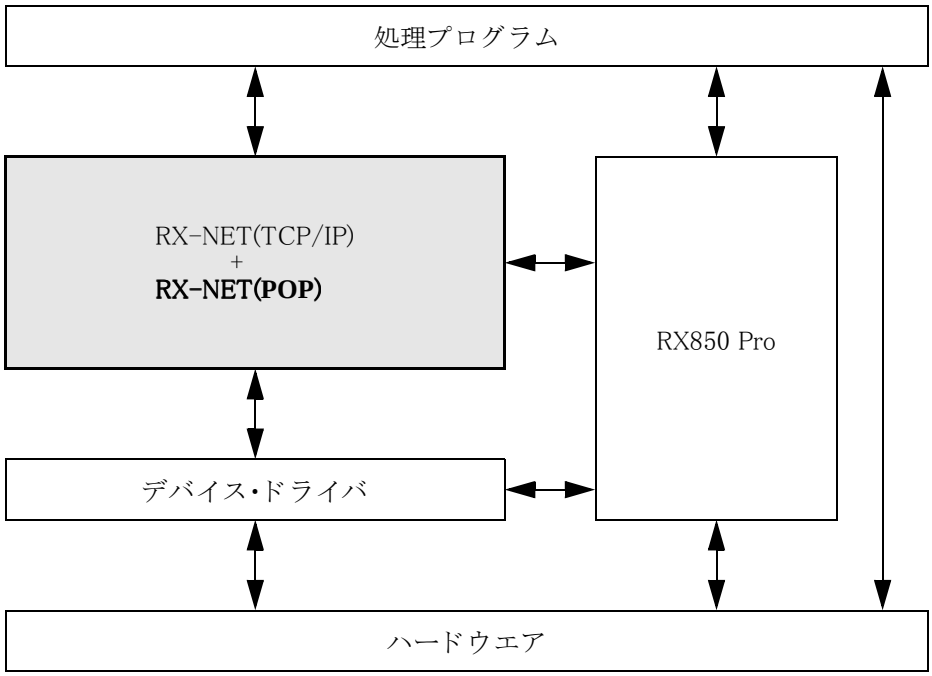
1.1　概要

RX-NET(POP) 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，POP3 (Post Office Protocol 3) による電子メール受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(POP) が提供する API 関数を利用することにより，電子メールの受信が可能となります。

図 1-1 に，RX-NET(POP) の位置付けを示します。

図 1-1　RX-NET(POP) の位置付け



1.2　特徴

以下に，RX-NET(POP) の特徴を示します。

- **RFC に準拠**

  RX-NET(POP) では，RFC に準拠した設計が行われています。

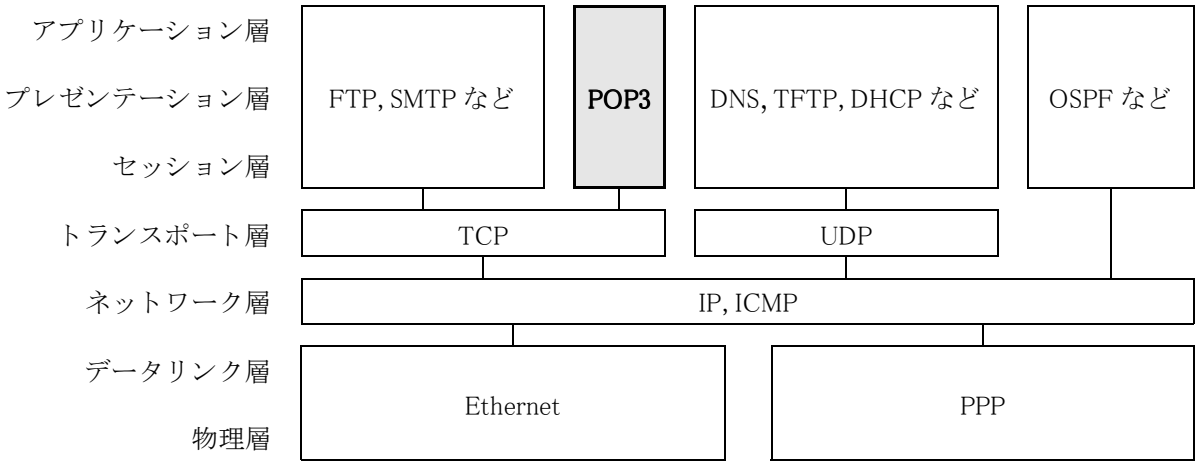
  なお，RFC とは，インターネットに関する研究開発期間 IETF (Internet Engineering Task Force) が取りまとめた公開技術文書であり，電子メール，ファイル転送などのプロトコル仕様 (情報交換を行う際に必要な手順，および，規約) の他にも，各種サービス，ガイドラインといった多岐に渡った情報 (ネットワーク技術の実装と運用に主眼を置いた情報) が記載されています。

- **電子メール受信機能をサポート**

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

  図 1-2 に，RX-NET(POP) の階層的位置付けを示します。

図 1-2　RX-NET(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) のテキスト領域　　：約 5 K バイト
  RX-NET(POP) のデータ領域　　：約 5 K バイト

1.4 開発環境

以下に，RX-NET(POP) 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(POP) 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(POP) 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(POP) の提供形式一覧を示します。

表 2-1　RX-NET(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) の提供媒体のディレクトリ RX-NET_POP_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(POP) 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(POP) 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(POP) 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/rxnetpop.tar を，米国 Green Hills Software, Inc. 製 CCV850E の場合は圧縮ファイル ghs/rxnetpop.tar を展開します。

【 CA850 対応版の場合 】

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

【 CCV850E 対応版の場合 】

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

6） ファイル群の確認
ls コマンドを実行し，RX-NET(POP) 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

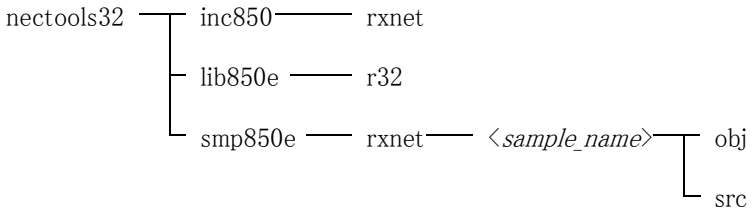
RX-NET(POP) の提供媒体に格納されているファイル群のディレクトリ構成は，C コンパイラ・パッケージの種類 (NEC 製 CA850，米国 Green Hills Software, Inc. 製 CCV850E) により異なります。

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

2) nectools32¥inc850¥rxnet
RX-NET(POP) 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
POP3 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。

libpop.a　: POP3 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。

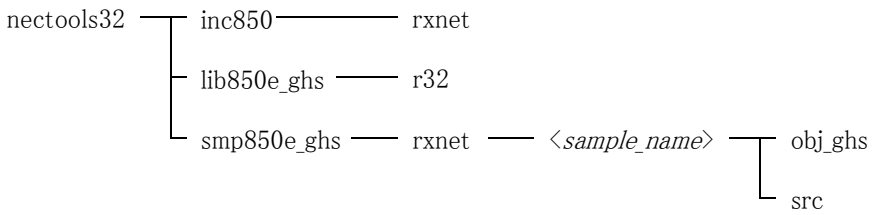
注意　〈*sample_name*〉についての詳細は，下記に示したテキスト・ファイルを参照してください。

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

2.3.2　CCV850E 対応版

図 2-2 に，RX-NET(POP) 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(POP) 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_pop.h　: RX-NET(POP) 用標準ヘッダ・ファイル

2) nectools32¥inc850¥rxnet
RX-NET(POP) 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
POP3 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。

libpop.a　: POP3 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.POP

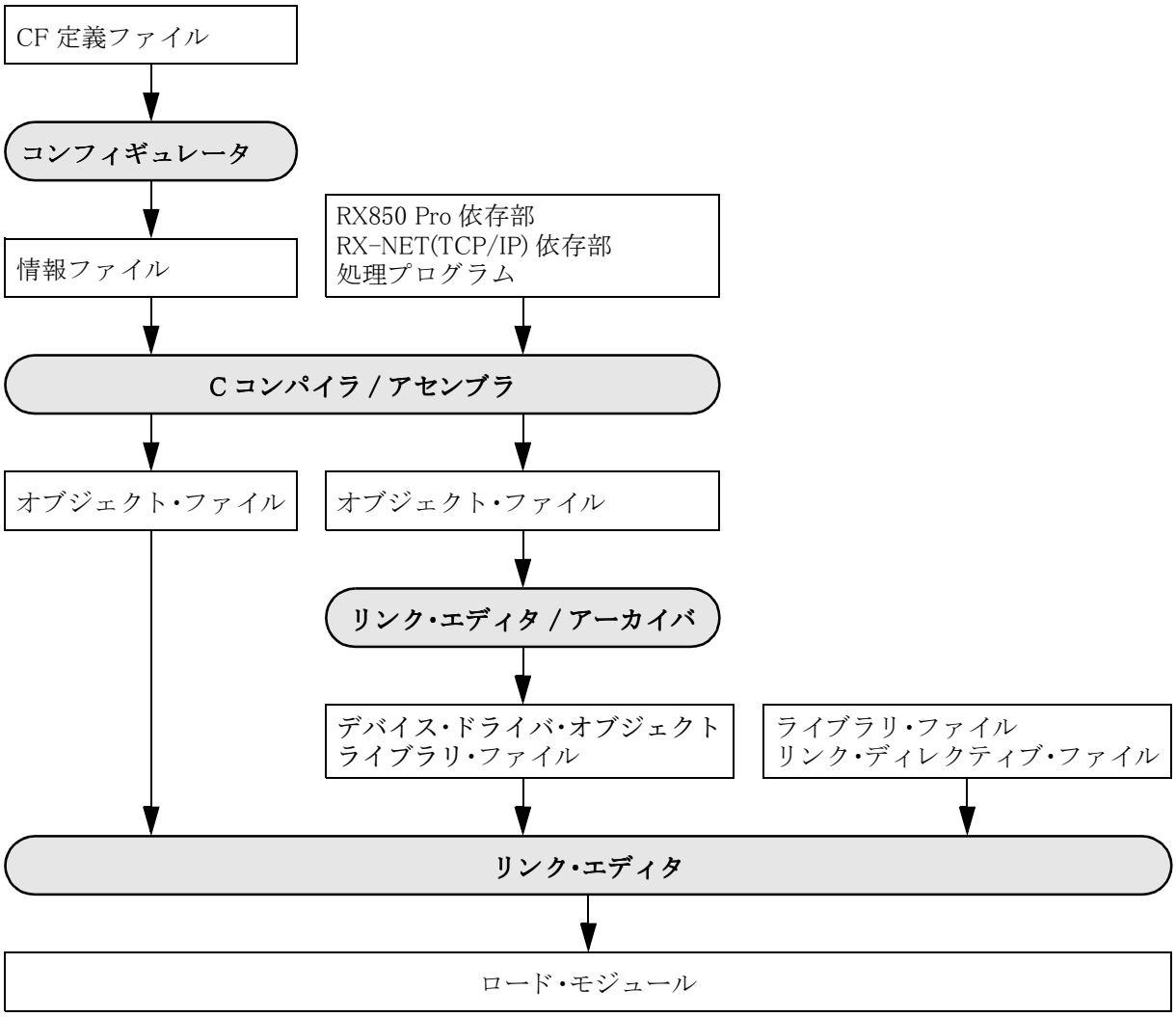
第 3 章　　システム構築

本章では，RX-NET(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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(POP) 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

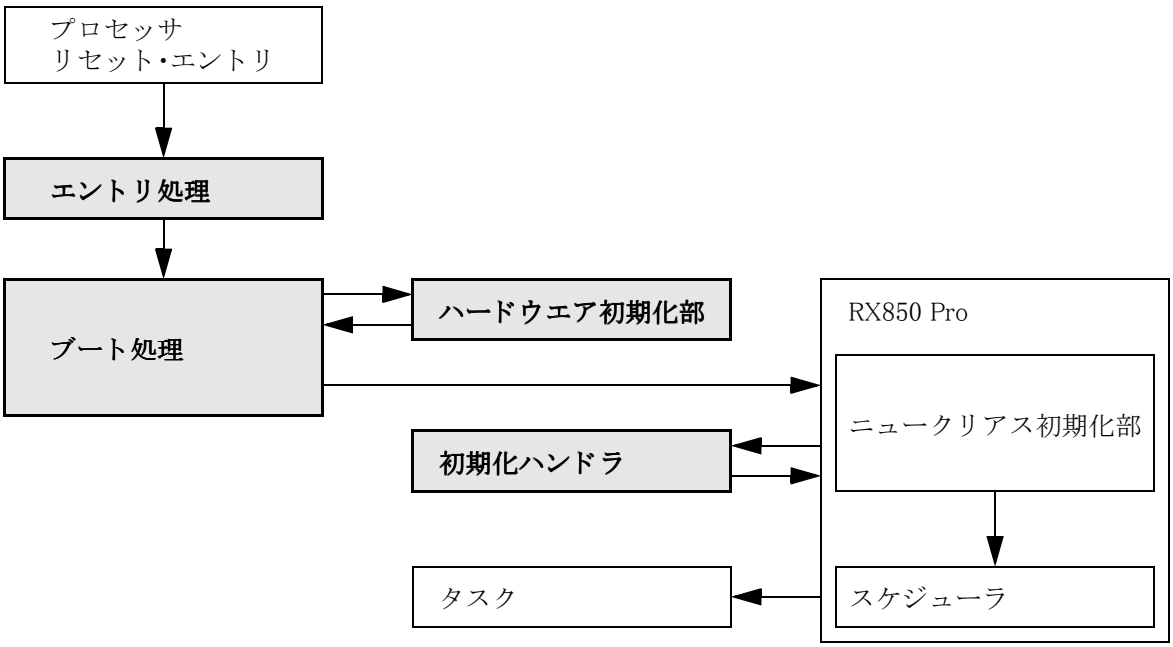
【 CCV850E 対応版の場合 】

・ 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(POP) 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　電子メール受信機能**」を参照してください。
なお，RX-NET(POP) 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

【 CCV850E 対応版の場合 】

・ nectools32¥smp850e_ghs¥rxnet¥<*sample_name*>¥src
　task.c　　：タスク

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(POP) 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(POP) 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

以前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(POP) が提供するライブラリ・ファイル
　- POP3 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(POP) 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　　電子メール受信機能

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

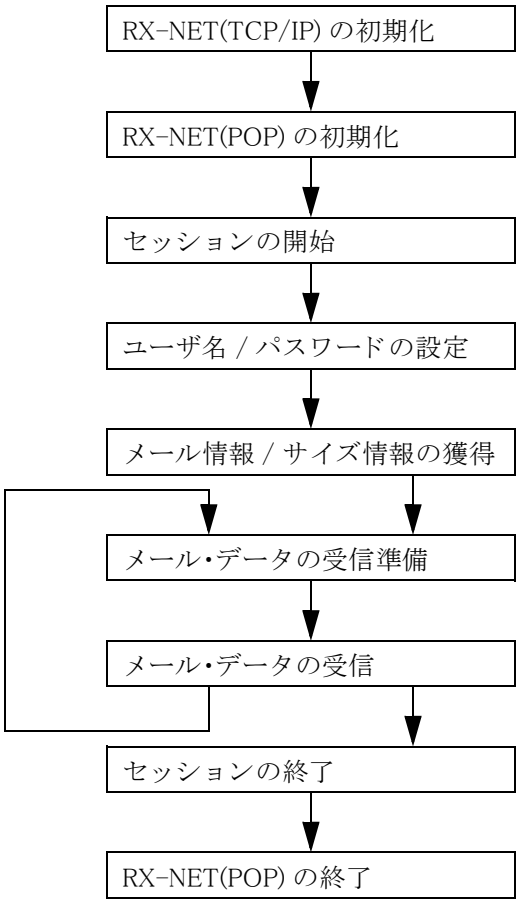
4.1　概要

RX-NET(POP) では，RX-NET(POP) の初期化処理，電子メールの受信処理の他に，エラー情報の獲得処理を“電子メール受信機能”として提供しています。

4.2　処理の流れ

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

図 4-1　電子メールの受信手順



注意　“複数の電子メール”を受信する場合には，上記電子メールの受信手順“メール・データの受信準備”～“メール・データの受信”の処理を繰り返し行うことにより実現されます。

4.3 電子メール受信機能 API 関数

4.3.1 RX-NET(POP) の初期化 / 終了

RX-NET(POP) 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。

- pop_start

  RX-NET(POP) 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_pop.h>              /* RX-NET(POP) 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 */

pop_start ( ) ;                                     /* RX-NET(POP) 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。
  ただし，RX-NET(DHCP) が提供する機能を併せて使用する場合には，API 関数 ll_config の発行が不要となります。

  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。

- pop_end

  RX-NET(POP) 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(POP) 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_pop.h>              /* RX-NET(POP) 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 */
pop_start ( ) ;                                     /* RX-NET(POP) の初期化 */

……
……
```

```
……

pop_end ( ) ;                                   /* RX-NET(POP) の終了 */
```

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

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。

- pop_connect

パラメータ *ipaddr*，*port* で指定されたメール・サーバ (メールをスプールしているホスト・マシン) とのセッションを開始します。

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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                  /* 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 */
pop_start ( ) ;                                 /* RX-NET(POP) の初期化 */

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

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

注意 2　POP 用ポート番号 *port* には，通常，110 番を設定します。

- pop_disconnect

API 関数 pop_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(POP) 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                  /* 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 */
pop_start ( ) ;                                      /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                       /* セッションの開始 */

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

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

注意　本 API 関数の発行以前に API 関数 pop_delete が発行されていた場合には，セッションの終了処理の他に，消去マークが付与されているメールの削除処理も併せて行われます。

4.3.3　ユーザ名 / パスワードの設定

ユーザ名 / パスワードの設定は，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。

- pop_setUserName

API 関数 pop_connect 発行時に指定したメール・サーバに対して USER コマンド (パラメータ *username* で指定されたユーザ名) 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                   /* 変数の宣言，初期化 */
char    *username ;                              /* 変数の宣言 */

username = "samplename" ;                        /* 変数の初期化 */

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 */
pop_start ( ) ;                                  /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                   /* セッションの開始 */

pop_setUserName ( username ) ;                   /* ユーザ名の設定 */
```

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

注意 2　本API関数の異常終了時には，メール・サーバとのセッションの強制終了処理(API関数pop_disconnect 相当の処理) が行われています。このため，メール・データの受信を行う場合は，API 関数 pop_connect の再発行が必要となります。

・pop_setUserPassword

API 関数 pop_connect 発行時に指定したメール･サーバに対して PASS コマンド (パラメータ *password* で指定されたパスワード) 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                      /* 変数の宣言，初期化 */
char    *username ;                                 /* 変数の宣言 */
char    *password ;                                 /* 変数の宣言 */

username = "samplename" ;                           /* 変数の初期化 */
password = "samplepassword" ;                       /* 変数の初期化 */

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 */
pop_start ( ) ;                                     /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                      /* セッションの開始 */
pop_setUserName ( username ) ;                      /* ユーザ名の設定 */

pop_setUserPassword ( password ) ;                  /* パスワードの設定 */
```

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

注意 2　本API関数の異常終了時には，メール･サーバとのセッションの強制終了処理(API関数pop_disconnect 相当の処理) が行われています。このため，メール･データの受信を行う場合は，API 関数 pop_connect の再発行が必要となります。

4.3.4　メール情報 / サイズ情報の獲得

メール情報 / サイズ情報の獲得は，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。

・pop_getMailStatus

API 関数 pop_connect 発行時に指定したメール・サーバに対して STAT コマンドの送信を行います。

これにより，パラメータ *stat_num* で指定された領域にはメール・サーバにスプールされているメールの総数が，パラメータ *stat_size* で指定された領域にはメール・サーバにスプールされているメールの合計サイズ (単位：バイト) が格納されます。

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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                  /* 変数の宣言，初期化 */
char    *username ;                             /* 変数の宣言 */
char    *password ;                             /* 変数の宣言 */
int     stat_num ;                              /* 変数の宣言 */
int     stat_size ;                             /* 変数の宣言 */

username = "samplename" ;                       /* 変数の初期化 */
password = "samplepassword" ;                   /* 変数の初期化 */

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 */
pop_start ( ) ;                                 /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                  /* セッションの開始 */
pop_setUserName ( username ) ;                  /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;              /* パスワードの設定 */

pop_getMailStatus ( &stat_num, &stat_size ) ;   /* メール情報の獲得 */
```

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

・ pop_getMailList

API 関数 pop_connect 発行時に指定したメール・サーバに対して LIST コマンドの送信を行います。

これにより，パラメータ *list_rslt* で指定された構造体にはパラメータ *number* で指定されたメールに対応したサイズ情報 (メール番号，メール・データのサイズなど) が，パラメータ *count* で指定された領域には実際に獲得したサイズ情報の総数が格納されます。

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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                  /* 変数の宣言，初期化 */
char    *username ;                             /* 変数の宣言 */
char    *password ;                             /* 変数の宣言 */
int     stat_num ;                              /* 変数の宣言 */
int     stat_size ;                             /* 変数の宣言 */
int     number ;                                /* 変数の宣言 */
struct  LIST    list_rslt [ 1024 ] ;            /* データ構造体の宣言 */
int     count ;                                 /* 変数の宣言 */

username = "samplename" ;                       /* 変数の初期化 */
password = "samplepassword" ;                   /* 変数の初期化 */

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 */
pop_start ( ) ;                                 /* RX-NET(POP) の初期化 */
```

```
pop_connect ( ipaddr, port ) ;                          /* セッションの開始 */
pop_setUserName ( username ) ;                          /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;                      /* パスワードの設定 */
pop_getMailStatus ( &stat_num, &stat_size ) ;           /* メール情報の獲得 */

if ( stat_num < 1024 ) {
        number = 0x0 ;                                  /* 変数の初期化 */
        count = stat_num ;                              /* 変数の初期化 */
} else {
        number = 0x1 ;                                  /* 変数の初期化 */
        count = 0x1 ;                                   /* 変数の初期化 */
}

pop_getMailList ( number, list_rslt, &count ) ;         /* サイズ情報の獲得 */
```

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

注意 2　本 API 関数では，パラメータ *number* に 0x0 が指定された場合には，メール・サーバにスプールされている全メールのサイズ情報の獲得処理を行います。
このため，パラメータ *number* に 0x0 を指定する場合は，API 関数 pop_getMailStatus の発行により獲得される“メールの総数 *stat_num*”と等しい数の LIST 構造体を配列で用意し，等しい値をパラメータ *count* に設定する必要があります。

注意 3　本 API 関数が正常終了した際，パラメータ *count* で指定された領域には，実際に獲得したサイズ情報の総数が格納されます。

注意 4　サイズ情報についての詳細は，「**5.4.1　サイズ情報**」を参照してください。

4.3.5　メール・データの受信準備

メール・データの受信準備は，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。

- pop_sendRetrRequest

  API 関数 pop_connect 発行時に指定したメール・サーバに対して RETR コマンド (パラメータ *number* で指定されたメール番号) 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                          /* 変数の宣言，初期化 */
char    *username ;                                     /* 変数の宣言 */
char    *password ;                                     /* 変数の宣言 */
int     number = 0x1 ;                                  /* 変数の宣言，初期化 */

username = "samplename" ;                               /* 変数の初期化 */
password = "samplepassword" ;                           /* 変数の初期化 */

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 */
pop_start ( ) ;                                           /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                            /* セッションの開始 */
pop_setUserName ( username ) ;                            /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;                        /* パスワードの設定 */

pop_sendRetrRequest ( number ) ;                          /* メール・データの受信要求 */
```

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

注意 2　本 API 関数では，メール・サーバに対するメール・データの受信要求を送信するだけであり，メール・サーバから送られてくるメール・データの受信処理は，本 API 関数発行後に API 関数 pop_recvMailData を発行することにより行われます。

・pop_sendTopRequest

API 関数 pop_connect 発行時に指定したメール・サーバに対して TOP コマンド (パラメータ *number* で指定されたメール番号，および，パラメータ *lines* で指定された行数) 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                    /* 変数の宣言，初期化 */
char    *username ;                               /* 変数の宣言 */
char    *password ;                               /* 変数の宣言 */
int     number = 0x1 ;                            /* 変数の宣言，初期化 **/
int     lines = 0x2 ;                             /* 変数の宣言，初期化 **/

username = "samplename" ;                         /* 変数の初期化 */
password = "samplepassword" ;                     /* 変数の初期化 */

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 */
pop_start ( ) ;                                           /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                            /* セッションの開始 */
pop_setUserName ( username ) ;                            /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;                        /* パスワードの設定 */

pop_sendTopRequest ( number, lines ) ;                    /* メール・データの受信要求 */
```

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

注意 2　本 API 関数では，メール・サーバに対するメール・データの受信要求を送信するだけであり，メール・サーバから送られてくるメール・データの受信処理は，本 API 関数発行後に API 関数 pop_recvMailData を発行することにより行われます。

4.3.6 メール・データの受信

メール・データの受信は，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。

- pop_recvMailData

  API 関数 pop_connect 発行時に指定したメール・サーバから API 関数 pop_sendRetrRequest，または，pop_sendTopRequest 発行時に指定したメール番号に対応したメール・データを受信し，パラメータ *msg* で指定された領域に格納します。

  なお，パラメータ *recv_size* には，パラメータ *msg* 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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                      /* 変数の宣言，初期化 */
char    *username ;                                 /* 変数の宣言 */
char    *password ;                                 /* 変数の宣言 */
int     number = 0x1 ;                              /* 変数の宣言，初期化 */
char    msg[ 1024 ] ;                               /* 変数の宣言 */
int     *recv_size ;                                /* 変数の宣言 */
int     retcd ;                                     /* 変数の宣言 */

username = "samplename" ;                           /* 変数の初期化 */
password = "samplepassword" ;                       /* 変数の初期化 */
*recv_size = 1024 ;                                 /* 変数の初期化 */

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 */
pop_start ( ) ;                                     /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                      /* セッションの開始 */
pop_setUserName ( username ) ;                      /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;                  /* パスワードの設定 */
pop_sendRetrRequest ( number ) ;                    /* メール・データの受信要求 */

retcd = pop_recvMailData ( msg , recv_size ) ;      /* メール・データの受信 */
while ( retcd == EPOP_RECVNEXT ) {
        *recv_size = 1024 ;                         /* 変数の初期化 */
        retcd = pop_recvMailData ( msg , recv_size ) ;  /* メール・データの受信 */
}
```

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

注意 2　本 API 関数が正常終了 (EPOP_RECVEND，EPOP_RECVNEXT) した際，パラメータ *recv_size* で指定された領域には，実際に受信したメール・データのサイズが格納されます。

注意 3　本 API 関数の戻り値 EPOP_RECVNEXT は，メール・データの受信処理が途中終了 (指定されたメール・データの格納領域が小さいため，全メール・データを格納することができない) していることを意味しています。このため，全メール・データの受信を完了 (未受信部分のメー

ル・データをパラメータ *msg* で指定された領域に格納）するためには，本 API 関数の再発行が必要となります。

4.3.7 Unique-ID 情報の獲得

Unique-ID 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。

- **pop_getUniqueId**

  API 関数 pop_connect 発行時に指定したメール・サーバに対して UIDL コマンドの送信を行います。

  これにより，パラメータ *uidl_rslt* で指定された構造体にはパラメータ *number* で指定されたメールに対応した Unique-ID 情報（メール番号，メールの Unique-ID など）が，パラメータ *count* で指定された領域には実際に獲得した Unique-ID 情報の総数が格納されます。

  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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                      /* 変数の宣言，初期化 */
char    *username ;                                 /* 変数の宣言 */
char    *password ;                                 /* 変数の宣言 */
int     stat_num ;                                  /* 変数の宣言 */
int     stat_size ;                                 /* 変数の宣言 */
int     number ;                                    /* 変数の宣言 */
struct  UIDL    uidl_rslt [ 1024 ] ;                /* データ構造体の宣言 */
int     count ;                                     /* 変数の宣言 */

username = "samplename" ;                           /* 変数の初期化 */
password = "samplepassword" ;                       /* 変数の初期化 */

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 */
pop_start ( ) ;                                     /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                      /* セッションの開始 */
pop_setUserName ( username ) ;                      /* ユーザ名の設定 */
pop_setUserPassword ( password ) ;                  /* パスワードの設定 */
pop_getMailStatus ( &stat_num, &stat_size ) ;       /* メール情報の獲得 */

if ( stat_num < 1024 ) {
        number = 0x0 ;                              /* 変数の初期化 */
        count = stat_num ;                          /* 変数の初期化 */
} else {
        number = 0x1 ;                              /* 変数の初期化 */
        count = 0x1 ;                               /* 変数の初期化 */
}

pop_getUniqueId ( number, uidl_rslt, &count ) ;     /* Unique-ID 情報の獲得 */
```

注意 1　本 API 関数では，パラメータ *number* に 0x0 が指定された場合には，メール・サーバにスプールされている全メールの Unique-ID 情報の獲得処理を行います。
このため，パラメータ *number* に 0x0 を指定する場合は，API 関数 pop_getMailStatus の発行により獲得される“メールの総数 *stat_num*”と等しい数の UIDL 構造体を配列で用意し，等しい値をパラメータ *count* に設定する必要があります。

注意 2　本API関数が正常終了した際,パラメータ *count* で指定された領域には,実際に獲得したUnique-ID 情報の総数が格納されます。

注意 3　Unique-ID 情報についての詳細は，「**5.4.2　Unique-ID 情報**」を参照してください。

4.3.8　消去マークの付与 / 解除

消去マークの付与 / 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。

- **pop_delete**

API 関数 pop_connect 発行時に指定したメール・サーバに対して DELE コマンド (パラメータ *number* で指定されたメール番号) の送信を行います。
これにより，パラメータ *number* で指定されたメール番号に対応したメールには，消去マークが付与されます。
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_pop.h>              /* RX-NET(POP) 用標準ヘッダ・ファイルの定義 */

int       number = 0x1 ;                        /* 変数の宣言，初期化 */

pop_delete ( number ) ;                         /* 消去マークの付与 */
```

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

注意 2　消去マークが付与されているメールの削除処理は，API 関数 pop_disconnect が発行された際に行われます。

- **pop_reset**

API 関数 pop_connect 発行時に指定したメール・サーバに対して RSET コマンドの送信を行います。
これにより，API 関数 pop_delete 発行時に付与された全消去マークの解除が行われます。
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_pop.h>              /* RX-NET(POP) 用標準ヘッダ・ファイルの定義 */

int       number = 0x1 ;                        /* 変数の宣言，初期化 */

pop_delete ( number ) ;                         /* 消去マークの付与 */

……
……
……

pop_reset ( ) ;                                 /* 消去マークの解除 */
```

注意　本 API 関数では，解除要求のキューイングが行われません。このため，既に本 API 関数が発行され，消去マークの解除処理が実行されていた場合には，何も処理は行わず，エラーとして扱いません。

4.3.9　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。

- pop_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_pop.h>              /* RX-NET(POP) 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 ( 110 ) ;                      /* 変数の宣言，初期化 */
char    *username ;                                 /* 変数の宣言 */
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 */
ll_config ( devname, ipaddress, ipmask, 0x0, 0x0 ) ;/* ネットワーク・インタフェースの起動 */
pop_start ( ) ;                                     /* RX-NET(POP) の初期化 */
pop_connect ( ipaddr, port ) ;                      /* セッションの開始 */

username = "samplename" ;                           /* 変数の初期化 */

pop_setUserName ( username ) ;                      /* ユーザ名の設定 */
pop_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 (pop_connect，pop_disconnect，pop_setUserName 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 (状態 +OK，-ERR 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。

4.3.10　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。

- pop_abort

本 API 関数を発行した際, 処理の完了していない API 関数 (pop_connect, pop_disconnect, pop_setUserName 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EPOP_SENDNFAIL, EPOP_RECVFAIL 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_pop.h>              /* RX-NET(POP) 用標準ヘッダ・ファイルの定義 */

pop_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 (A PI 関数 pop_disconnect 相当の処理) も併せて行われます。このため，メール・データの受信を行う場合は，API 関数 pop_connect の再発行が必要となります。

第 5 章　　API 関数

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

pop_start	pop_end	pop_connect	pop_disconnect
pop_setUserName	pop_setUserPassword	pop_getMailStatus	pop_getMailList
pop_sendRetrRequest	pop_sendTopRequest	pop_recvMailData	pop_getUniqueId
pop_delete	pop_reset	pop_getErrorLine	pop_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(POP) 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(POP) 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(POP) 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_pop.h	: RX-NET(POP) 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(POP) 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_ppp.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥poperrno.h で行われています。

表 5-3　戻り値

マクロ	数値	意味
EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です
EPOP_RUNNING	0x1201	既に pop_start が発行されています
EPOP_DOWN	0x1202	pop_start が発行されていません
EPOP_CRESEM	0x1203	RX-NET(POP) 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
EPOP_DELSEM	0x1204	RX-NET(POP) 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
EPOP_WAISEM	0x1205	RX-NET(POP) 用セマフォが削除されています
EPOP_APIRUNNING	0x1206	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
EPOP_CONNECT	0x1207	既に pop_connect が発行されています
EPOP_NOTCONN	0x1208	pop_connect が発行されていません
EPOP_SOCKET	0x1209	RX-NET(POP) 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
EPOP_CONNFAIL	0x120a	ネットワーク接続の確立が失敗しました
EPOP_SENDFAIL	0x120b	コマンドの送信処理が失敗しました
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました
EPOP_BADRES	0x120d	送信したコマンド（QUIT，STAT，LIST など）に対するメール・サーバからの応答がエラーです
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_RECVEND	0x120f	メール・データの受信処理が正常終了しました
EPOP_RECVNEXT	0x1210	メール・データの受信処理が途中終了しました
EPOP_SENDTMDOUT	0x1211	コマンドの送信処理がタイムアウトしました
EPOP_AUTH	0x1212	送信したコマンド（USER，PASS など）に対するメール・サーバからの応答がエラーです

5.4　データ構造体

RX-NET(POP) 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 (メール･サイズ情報, Unique-ID 情報など) について以下に示します。

5.4.1　サイズ情報

以下に，メール・サイズ情報の構造を示します。
なお，メール・サイズ情報の定義は，標準ヘッダ・ファイル nectools32¥inc850¥rxnet_pop.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥popaux.h で行われています。

```
struct    LIST {
          int       list_num ;          /* メール番号 */
          int       list_size ;         /* メール・データのサイズ ( 単位 : バイト ) */
} ;
```

list_num　: list_num で指定された領域には，メール番号が格納されます。
list_size　: list_size で指定された領域には，メール・データのサイズ (単位 : バイト) が格納されます。

5.4.2　Unique-ID 情報

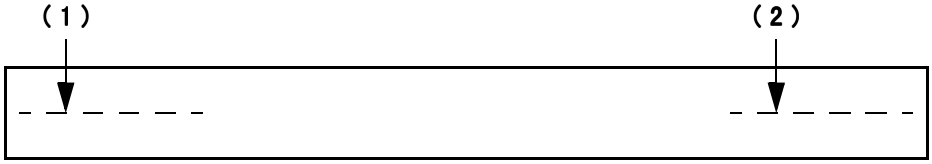
以下に，Unique-ID 情報の構造を示します。
なお，Unique-ID 情報の定義は，標準ヘッダ・ファイル nectools32¥inc850¥rxnet_pop.h から呼び出されるヘッダ・ファイル nectools32¥inc850¥rxnet¥popaux.h で行われています。

```
struct    UIDL {
          int       uidl_num ;          /* メール番号 */
          char      uidl_id [ 71 ] ;    /* メールの Unique-ID */
} ;
```

uidl_num　: uidl_num で指定された領域には，メール番号が格納されます。
uidl_id　: uidl_id で指定された領域には，メールの Unique-ID が格納されます。

5.5 API 関数解説

次項から RX-NET(POP) 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(POP) への入力パラメータ
O … RX-NET(POP) 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。

5.5.1 外部インタフェース仕様

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

表 5-4 RX-NET(POP) の API 関数

API 関数名	機能概要
pop_start	RX-NET(POP) の初期化
pop_end	RX-NET(POP) の終了
pop_connect	セッションの開始
pop_disconnect	セッションの終了 (QUIT コマンドの送信)
pop_setUserName	ユーザ名の設定 (USER コマンドの送信)
pop_setUserPassword	パスワードの設定 (PASS コマンドの送信)
pop_getMailStatus	メール情報の獲得 (STAT コマンドの送信)
pop_getMailList	サイズ情報の獲得 (LIST コマンドの送信)
pop_sendRetrRequest	メール・データの受信要求 (RETR コマンドの送信)
pop_sendTopRequest	メール・データの受信要求 (TOP コマンドの送信)
pop_recvMailData	メール・データの受信
pop_getUniqueId	Unique-ID 情報の獲得 (UIDL コマンドの送信)
pop_delete	消去マークの付与 (DELE コマンドの送信)
pop_reset	消去マークの解除 (RSET コマンドの送信)
pop_getErrorLine	エラー情報の獲得
pop_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。

pop_start

タスク

概要

RX-NET(POP) の初期化

C 言語形式

```
int             pop_start ( void ) ;
```

パラメータ

なし

機能

RX-NET(POP) 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。
ただし，RX-NET(DHCP) が提供する機能を併せて使用する場合には，API 関数 ll_config の発行が不要となります。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_RUNNING	0x1201	既に pop_start が発行されています
EPOP_CRESEM	0x1203	RX-NET(POP) 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

pop_end

タスク

概要

RX-NET(POP) の終了

C 言語形式

```
int             pop_end ( void ) ;
```

パラメータ

なし

機能

RX-NET(POP) 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(POP) が提供する機能を利用することができなくなります。

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

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_DOWN	0x1202	pop_start が発行されていません
EPOP_DELSEM	0x1204	RX-NET(POP) 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
EPOP_APIRUNNING	0x1206	他の API 関数が処理を完了していないため，該当処理“RX-NET(POP) の終了処理”を実行することができません
EPOP_CONNECT	0x1207	pop_disconnect が発行されていません

pop_connect

タスク

概要

セッションの開始

C 言語形式

int　　pop_connect (u32 *ipaddr* , u16 *port*) ;

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 POP 用ポート番号 (ネットワーク・バイト・オーダ)

機能

ipaddr, *port* で指定されたメール・サーバ (メールをスプールしているホスト・マシン) とのセッションを開始します。

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

注意 2　POP 用ポート番号 *port* には，通常，110 番を設定します。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	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 – POP 用ポート番号の指定が不正 (*port* = 0x0) です
EPOP_APIRUNNING	0x1206	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
EPOP_CONNECT	0x1207	既に pop_connect が発行されています
EPOP_SOCKET	0x1209	RX-NET(POP) 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
EPOP_CONNFAIL	0x120a	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pop_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 POP 用ポート番号は *port* で指定された値ではありません
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました – 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 pop_abort が発行されました – 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
EPOP_BADRES	0x120d	メール・サーバからの応答がエラーです

pop_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             pop_disconnect ( void ) ;
```

パラメータ

なし

機能

API 関数 pop_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(POP) 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。

注意　本 API 関数の発行以前に API 関数 pop_delete が発行されていた場合には，セッションの終了処理の他に，消去マークが付与されているメールの削除処理も併せて行われます。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_APIRUNNING	0x1206	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
EPOP_NOTCONN	0x1208	pop_connect が発行されていません
EPOP_SENDFAIL	0x120b	QUIT コマンドの送信処理が失敗しました – QUIT コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました – QUIT コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	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 pop_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 – QUIT コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	QUIT コマンドの送信処理がタイムアウトしました

pop_setUserName

タスク

概要

ユーザ名の設定 (USER コマンドの送信)

C 言語形式

int　　pop_setUserName (char *_username_) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　*_username_ ;	ユーザ名を格納した領域へのポインタ

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して USER コマンド (*username* で指定されたユーザ名) の送信を行います。

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

注意 2　本API関数の異常終了時には, メール・サーバとのセッションの強制終了処理(API関数pop_disconnect 相当の処理) が行われています。このため, メール・データの受信を行う場合は, API 関数 pop_connect の再発行が必要となります。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です - ユーザ名を格納した領域の指定が不正 (NULL ポインタ) です - ユーザ名の文字総数が不正 (0 文字) です - ユーザ名の文字総数が最大文字数 (40 文字) を越えています
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	USER コマンドの送信処理が失敗しました - USER コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました - USER コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました - USER 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 pop_abort が発行されました - USER 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 - USER コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	USER コマンドの送信処理がタイムアウトしました
EPOP_AUTH	0x1212	USER 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

pop_setUserPassword

タスク

概要

パスワードの設定 (PASS コマンドの送信)

C 言語形式

int　　pop_setUserPassword (char **password*) ;

パラメータ

I/O	パラメータ	説　　明
I	char　　**password* ;	パスワードを格納した領域へのポインタ

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して PASS コマンド (*password* で指定されたパスワード) の送信を行います。

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

注意 2　本API関数の異常終了時には, メール･サーバとのセッションの強制終了処理(API関数pop_disconnect 相当の処理) が行われています。このため, メール･データの受信を行う場合は, API 関数 pop_connect の再発行が必要となります。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です – パスワードを格納した領域の指定が不正 (NULL ポインタ) です – パスワードの文字総数が不正 (0 文字) です – パスワードの文字総数が最大文字数 (40 文字) を越えています
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	PASS コマンドの送信処理が失敗しました – PASS コマンドをメール･サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました – PASS コマンドをメール･サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました – PASS 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 pop_abort が発行されました – PASS 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 – PASS コマンドに対するメール･サーバからの応答待ち処理がタイムアウトしました
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	PASS コマンドの送信処理がタイムアウトしました
EPOP_AUTH	0x1212	PASS 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

pop_getMailStatus

タスク

概要

メール情報の獲得 (STAT コマンドの送信)

C 言語形式

int　　pop_getMailStatus (int *stat_num , int *stat_size) ;

パラメータ

I/O	パラメータ	説　　明
O	int　*stat_num ;	メールの総数を格納する領域へのポインタ
O	int　*stat_size ;	メールの合計サイズ (単位 : バイト) を格納する領域へのポインタ

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して STAT コマンドの送信を行います。
これにより，*stat_num* で指定された領域にはメール・サーバにスプールされているメールの総数が，*stat_size* で指定された領域にはメール・サーバにスプールされているメールの合計サイズ (単位 : バイト) が格納されます。

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

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です - 総数を格納する領域の指定が不正 (NULL ポインタ) です - 合計サイズを格納する領域の指定が不正 (NULL ポインタ) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	STAT コマンドの送信処理が失敗しました - STAT コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました - STAT コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました - STAT 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 pop_abort が発行されました - STAT 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 - STAT コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	STAT 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	STAT コマンドの送信処理がタイムアウトしました

pop_getMailList

タスク

概要

サイズ情報の獲得 (LIST コマンドの送信)

C 言語形式

```
int         pop_getMailList ( int number , struct LIST *list_rslt , int *count ) ;
```

パラメータ

I/O	パラメータ	説　　明
I	int *number* ;	サイズ情報を獲得するメールのメール番号
O	struct LIST **list_rslt* ;	サイズ情報を格納する構造体へのポインタ
I，O	int **count* ;	*list_rslt* で指定されたサイズ情報を格納する構造体の総数を格納した領域へのポインタ

サイズ情報 LIST の構造

```
struct  LIST {
        int     list_num ;          /* メール番号 */
        int     list_size ;         /* メール・データのサイズ ( 単位 : バイト ) */
} ;
```

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して LIST コマンドの送信を行います。
これにより，*list_rslt* で指定された構造体には *number* で指定されたメールに対応したサイズ情報 (メール番号，メール・データのサイズなど) が，*count* で指定された領域には実際に獲得したサイズ情報の総数が格納されます。

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

注意 2　本 API 関数では，*number* に 0x0 が指定された場合には，メール・サーバにスプールされている全メールのサイズ情報の獲得処理を行います。
このため，*number* に 0x0 を指定する場合は，API 関数 pop_getMailStatus の発行により獲得される“メールの総数 *stat_num*”と等しい数の LIST 構造体を配列で用意し，等しい値を *count* に設定する必要があります。

注意 3　本 API 関数が正常終了した際，*count* で指定された領域には，実際に獲得したサイズ情報の総数が格納されます。

注意 4　サイズ情報についての詳細は，「**5.4.1　サイズ情報**」を参照してください。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です – メール番号の指定が不正 (0x0 以下の値) です – サイズ情報を格納する構造体の指定が不正 (NULL ポインタ) です – 総数を格納する領域の指定が不正 (NULL ポインタ) です

		- 総数の指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	LIST コマンドの送信処理が失敗しました - LIST コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました - LIST コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました - LIST 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 pop_abort が発行されました - LIST 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 - LIST コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	LIST 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	LIST コマンドの送信処理がタイムアウトしました

pop_sendRetrRequest

タスク

概要

メール・データの受信要求 (RETR コマンドの送信)

C 言語形式

int　　pop_sendRetrRequest (int *number*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*number* ;	受信要求するメールのメール番号

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して RETR コマンド (*number* で指定されたメール番号) の送信を行います。

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

注意 2　本 API 関数では，メール・サーバに対するメール・データの受信要求を送信するだけであり，メール・サーバから送られてくるメール・データの受信処理は，本 API 関数発行後に API 関数 pop_recvMail Data を発行することにより行われます。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です – メール番号の指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	RETR コマンドの送信処理が失敗しました – RETR コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました – RETR コマンドをメール・サーバが受け付けるの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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	RETR コマンドの送信処理がタイムアウトしました

pop_sendTopRequest

タスク

概要

メール・データの受信要求 (TOP コマンドの送信)

C 言語形式

int　　pop_sendTopRequest (int *number* , int *lines*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　*number* ;	受信要求するメールのメール番号
I	int　*lines* ;	受信要求するメールの行数

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して TOP コマンド (*number* で指定されたメール番号，および，*lines* で指定された行数) の送信を行います。

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

注意 2　本 API 関数では，メール・サーバに対するメール・データの受信要求を送信するだけであり，メール･サーバから送られてくるメール･データの受信処理は, 本API関数発行後にAPI関数pop_recvMail Data を発行することにより行われます。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です - メール番号の指定が不正 (0x0 以下の値) です - 行数の指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	TOP コマンドの送信処理が失敗しました - TOP コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました - TOP コマンドをメール・サーバが受け付けるの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
EPOP_BADRES	0x120d	TOP 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	TOP コマンドの送信処理がタイムアウトしました

pop_recvMailData

タスク

概要

メール・データの受信

C 言語形式

int　　pop_recvMailData (char **msg* , int **recv_size*) ;

パラメータ

I/O	パラメータ	説　　明
O	char　　**msg* ;	メール・データを格納する領域へのポインタ
I, O	int　　**recv_size* ;	*msg* で指定された領域のサイズ (単位：バイト)

機能

API関数pop_connect発行時に指定したメール･サーバからAPI関数pop_sendRetrRequest, または, pop_sendTopRequest 発行時に指定したメール番号に対応したメール･データを受信し, *msg* で指定された領域に格納します。

なお, *recv_size* には, *msg* 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　本 API 関数の発行は, API 関数 pop_sendRetrRequest, または, pop_sendTopRequest の処理完了後に行う必要があります。

注意 2　本 API 関数が正常終了 (EPOP_RECVEND, EPOP_RECVNEXT) した際, *recv_size* で指定された領域には, 実際に受信したメール・データのサイズが格納されます。

注意 3　本 API 関数の戻り値 EPOP_RECVNEXT は, メール･データの受信処理が途中終了 (指定されたメール･データの格納領域が小さいため, 全メール・データを格納することができない) していることを意味しています。このため, 全メール・データの受信を完了 (未受信部分のメール・データを *msg* で指定された領域に格納) するためには, 本 API 関数の再発行が必要となります。

戻り値

EPOP_INVAL	0x1200	パラメータの指定が不正です - メール･データを格納する領域の指定が不正(NULLポインタ)です - サイズを格納する領域の指定が不正 (NULL ポインタ) です - サイズの指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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･データの受信処理”を実行することができません
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました - RETR コマンド, または, TOP 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pop_abortが発行されました - RETR コマンド, または, TOP 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 - RETR コマンド, または, TOP コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	RETR コマンド, または, TOP 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

EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_RECVEND	0x120f	メール・データの受信処理が正常終了しました
EPOP_RECVNEXT	0x1210	メール・データの受信処理が途中終了しました

pop_getUniqueId

タスク

概要

Unique-ID 情報の獲得 (UIDL コマンドの送信)

C 言語形式

int　　pop_getUniqueId (int *number* , struct UIDL **uidl_rslt* , int **count*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*number* ;	Unique-ID 情報を獲得するメールのメール番号
O	struct　UIDL　**uidl_rslt* ;	Unique-ID 情報を格納する構造体へのポインタ
I, O	int　　**count* ;	*uidl_rslt* で指定された Unique-ID 情報を格納する構造体の総数を格納した領域へのポインタ

Unique-ID 情報 UIDL の構造

```
struct    UIDL {
          int       uidl_num ;          /* メール番号 */
          int       uidl_id [ 71 ] ;    /* メールの Unique-ID */
} ;
```

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して UIDL コマンドの送信を行います。
これにより，*uidl_rslt* で指定された構造体には *number* で指定されたメールに対応した Unique-ID 情報 (メール番号，メールの Unique-ID など) が，*count* で指定された領域には実際に獲得した Unique-ID 情報の総数が格納されます。

注意 1　本 API 関数では，*number* に 0x0 が指定された場合には，メール・サーバにスプールされている全メールの Unique-ID 情報の獲得処理を行います。
このため，*number* に 0x0 を指定する場合は，API 関数 pop_getMailStatus の発行により獲得される“メールの総数 *stat_num*”と等しい数の UIDL 構造体を配列で用意し，等しい値を *count* に設定する必要があります。

注意 2　本 API 関数が正常終了した際，*count* で指定された領域には，実際に獲得した Unique-ID 情報の総数が格納されます。

注意 3　Unique-ID 情報についての詳細は，「**5.4.2　Unique-ID 情報**」を参照してください。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です - メール番号の指定が不正 (0x0 以下の値) です - Unique-ID 情報を格納する構造体の指定が不正 (NULL ポインタ) です - 総数を格納する領域の指定が不正 (NULL ポインタ) です

		- 総数の指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	UIDL コマンドの送信処理が失敗しました - UIDL コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました - UIDL コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました - UIDL 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 pop_abort が発行されました - UIDL 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 - UIDL コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	UIDL 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	UIDL コマンドの送信処理がタイムアウトしました

pop_delete

タスク

概要

消去マークの付与 (DELE コマンドの送信)

C 言語形式

int　　pop_delete (int *number*) ;

パラメータ

I/O	パラメータ	説　　明
I	int　　*number* ;	消去マークを付与するメールのメール番号

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して DELE コマンド (*number* で指定されたメール番号) の送信を行います。
これにより，*number* で指定されたメール番号に対応したメールには，消去マークが付与されます。

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

注意 2　消去マークが付与されているメールの削除処理は，API 関数 pop_disconnect が発行された際に行われます。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	パラメータの指定が不正です – メール番号の指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	DELE コマンドの送信処理が失敗しました – DELE コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました – DELE コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました – DELE 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 pop_abort が発行されました – DELE 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 – DELE コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	DELE 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	DELE コマンドの送信処理がタイムアウトしました

pop_reset

タスク

概要

消去マークの解除 (RSET コマンドの送信)

C 言語形式

```
int             pop_reset ( void ) ;
```

パラメータ

なし

機能

API 関数 pop_connect 発行時に指定したメール・サーバに対して RSET コマンドの送信を行います。
これにより，API 関数 pop_delete 発行時に付与された全消去マークの解除が行われます。

注意　本 API 関数では，解除要求のキューイングが行われません。このため，既に本 API 関数が発行され，消去マークの解除処理が実行されていた場合には，何も処理は行わず，エラーとして扱いません。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_APIRUNNING	0x1206	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
EPOP_SENDFAIL	0x120b	RSET コマンドの送信処理が失敗しました – RSET コマンドをメール・サーバが受け付けるのを待っている際に他タスクから pop_abort が発行されました – RSET コマンドをメール・サーバが受け付けるの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
EPOP_RECVFAIL	0x120c	エラー情報の受信処理が失敗しました – RSET 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 pop_abort が発行されました – RSET 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 – RSET コマンドに対するメール・サーバからの応答待ち処理がタイムアウトしました
EPOP_BADRES	0x120d	RSET 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
EPOP_STATUS	0x120e	API 関数の発行順序が不正です
EPOP_SENDTMDOUT	0x1211	RSET コマンドの送信処理がタイムアウトしました

pop_getErrorLine

タスク

概要

エラー情報の獲得

C 言語形式

int　　pop_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 (pop_connect，pop_disconnect，pop_setUserName 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 (状態 +OK，-ERR 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，*line* で指定された領域の内容は不定となります。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_INVAL	0x1200	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 - サイズの指定が不正 (0x0 以下の値) です
EPOP_APIRUNNING	0x1206	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

pop_abort

タスク

概要

セッションの強制終了

C 言語形式

```
int             pop_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 (pop_connect，pop_disconnect，pop_setUserName 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EPOP_SENDNFAIL, EPOP_RECVFAIL 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 pop_disconnect 相当の処理) も併せて行われます。このため，メール・データの受信を行う場合は，API 関数 pop_connect の再発行が必要となります。

戻り値

EPOP_NOERR	0x0	正常終了
EPOP_WAISEM	0x1205	RX-NET(POP) 用セマフォが削除されています

索引

い

お

か

さ

し

せ

て

は

ひ

め

も

り

ろ