ITRON TCP/IP API 仕様準拠
プロトコルスタック

TCP/IP 編

ユーザーズガイド

MiSPO

株式会社ミスポ

第 8 版(本版)で改訂された項目

ページ	内容
-	各 API 関数の引数に const が抜けていた点を修正
1	特長の説明で、具体的なプロセッサ名や Ethernet コントローラ名を削除(対応状況は頻繁に更新されるため)
1	特長の説明にマルチホームに関する記述を追加、SNTP クライアントサンプルを追加
2-3	ファイル構成に nonloop.h、noneraw.c、nonloop.c を追加
6-7	プロトコルスタックの構成の一覧と階層構造の図に IGMP を追加
7	IP 受信タスク/IP 送信タスクの"全体で 1 個"を"ネットワークインターフェース毎に 1 個"に変更(マルチチャネルの場合を考慮)
8	送信パケットキュー/送信リトライキューの"全体で 1 個"を"ネットワークインターフェース毎に 1 個"に変更(同上)
16	ARP テーブルの説明に ARP_FLUSH_TOUT についての記述を追加
17	icmp_def_cbk の解説で"ECHO 以外の"を削除、引数に関する解説追加
18	コールバックの形式で T_IP4 構造体のメンバ修正、解説で"ECHO 以外の"を削除
20	キープアライブの説明で戻値の訂正(0 または E_CLS→E_TMOUT)、TCP_KTIME_PRO と TCP_KTIME_SUC についての説明と図を追加
21	再送信でもエラーの場合の戻値を訂正(E_CLS→E_TMOUT)
22,23	23 ページにあった脚注 1 を移動し、設定値の計算例を追加。脚注 2 は削除
23	コンフィグレーション可能な項目(定義の一覧)に ARP_FLUSH_TOUT を追加
23	キープアライブ設定上の注意事項(ルール)を追加
28	メインエラーコードの一覧に E_LNK, E_ADDR を追加
29	htons, ntohl, ntohs の引数の説明の誤記を訂正(hl→hs/nl/ns)
32,42,62	tcp_cre_rep, tcp_con_cep, udp_cre_cep の解説にマルチホームの利用方法を追加
32	tcp_cre_rep の補足で、自分の持つ全 IP アドレスへの接続要求を待ち受ける機能を未サポートからサポートに
36	TCP 通信端点の送受信バッファサイズの制限事項(64KB 未満で 2 のべき乗)を追記
39	tcp_acp_cep の戻値に E_PAR と E_LNK を追加
41	tcp_con_cep の戻値に E_PAR と E_LNK を追加
44	tcp_cls_cep の戻値に E_CLS を追加
65	udp_snd_dat の戻値に E_PAR の説明を変更
65	udp_snd_dat の戻値に E_LNK を追加
67	udp_rcv_dat の戻値に E_LNK を追加
67	udp_rcv_dat の解説で、"パケット未受信の場合"を"タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合"に訂正
68	udp_rcv_dat の解説で、受信要求のキューイング数に制限がないと訂正(E_QOVR エラーにならない)

68	udp_rcv_dat の解説で、受信バッファの最小バイト数を 20byte から 24byte に変更
70	udp_set_opt の解説で、"接続前にのみ変更ができる"の記述を削除
71	udp_set_opt の例で、引数に sizeof(mreq)を渡していたのを、sizeof(addr)に訂正
72	udp_get_opt の戻値に E_LNK を追加
73	ノンブロッキングコールの完了通知で、"サービスコールの機能コード"の説明の誤記を修正(udp_rcv_dat 完了通知と udp_snd_dat ｖを入れ替え)
76	tcp_ini の戻値に E_ID,E_SYS,E_NOMEM を追加
76	tcp_ini()前に TCP/UDP の各 API を呼び出すとエラーになることを追記
77	net_get_opt のページを追加
78	net_set_opt のページを追加
79	net_chg_ipa のページを追加
80	arp_add_entry のページを追加
80	arp_add_byname のページを追加
81	arp_del_entry のページを追加
81	arp_del_byname のページを追加

第 7 版で改訂された項目

ページ	内容
11	lan_ext_dev の説明を追加
22	ARP_CACHE_TOUT のデフォルト値を 120 に変更
39	tcp_acp_cep の戻値に E_CLS を追加
59	tcp_set_opt の記述を変更
60	tcp_get_opt の記述を変更
62	udp_cre_cep の戻値に E_PAR を追加
63	udp_vcre_cep の戻値に E_PAR を追加
65	udp_snd_dat の戻値に E_PAR を変更
67	udp_rcv_dat の戻値に E_OBJ を追加
70	udp_set_opt の記述を変更
71	udp_get_opt の記述を変更
76	tcp_ini の戻値を変更
76	tcp_ext の説明を追加

第 6 版で改訂された項目

ページ	内容
-	"NORTi Network"という表現を"NORTi TCP/IP"に変更
2	noneipf.c の説明を追加
17	ARP モジュール「ARP テーブル」の内容に追記
18	ICMP モジュール「Echo 処理」の"ICMP を活用していません... "を削除
19	UDP モジュール「UDP パケット受信」に注意事項を追記
21	TCP モジュール「キープアライブ」「ストリーム型通信方式」「再送信」に関する記述を追記
23	コンフィグレーション「定義」に新たに追加されたコンフィグレーションを追記
33	tcp_cre_rep の解説の"キューイング"に関する記述を削除
34	tcp_vcre_rep の戻値に E_PAR を追加
36	tcp_cre_cep の解説に"またバッファに..."を追記 tcp_cre_cep の補足の"2920～8760"を"2048,8192"に変更 tcp_cre_cep の補足に"1 回のコネクションで..."を追記
40	tcp_acp_cep の解説の"ポーリングで本サービスコールを..."を削除
42	tcp_con_cep の戻値の E_CLS の"接続失敗"を"接続が拒否された"に変更
43	tcp_con_cep の補足の"E_OBJ でリターン..."を"E_CLS でリターン..."に変更
56	tcp_rel_buf の解説に"取り出したバッファは..."を追記
59	tcp_can_cep の解説の"現バージョンではサポートされていません"を削除
66	udp_snd_dat の解説の"udp_cre_cep で指定した..."を"UDP_QCNT"で指定したに変更
68	udp_rcv_dat の戻値の E_QOVR を削除 udp_rcv_dat の解説の"キューイングできる送信パケット..."の記述を削除
71	udp_set_opt に戻値を追加 udp_set_opt に例を追加
索引	索引を追加

第 5 版で改訂された項目

ページ	内容
1	現バージョンは、IP フラグメント機能をサポートしていません。...を削除
18	ICMP の機能説明を追加
29	メインエラーコードを修正
30	ユーティリティマクロの説明にユーティリティ関数の説明を追加
66	udp_snd_dat の戻値の E_PAR の説明を修正
68	udp_rcv_dat の戻値の E_PAR の説明を修正

第 4 版で改訂された項目

ページ	内容
1	制限事項の内容を一部変更
3	nonigmp.c の説明を追加
33	tcp_cre_rep の解説を一部変更
37	tcp_cre_cep の TCP 送受信バッファについての補足を追加
43	tcp_con_cep の IPV4_ADDRANY,TCP_PORTANY の説明を変更
43	tcp_con_cep の戻値の説明を変更
43	tcp_con_cep の補足を追加
63	udp_cre_cep の IPV4_ADDRANY,UDP_PORTANY の説明を変更
66	udp_snd_dat の戻値の E_PAR の内容を一部変更
74	コールバックの機能コードを追加

第 3 版で改訂された項目

ページ	内容
9	NORTi TCP/IP では次のようなパラメータをタイムアウトに設定できます。を追記
13	lan_wai_snd の説明を修正
13	lan_get_pkt の引数を修正
40	"tcp_acp_cep の補足にノンブロッキング（tmout =TMO_NBLK ）で、... "を追記
45	tcp_cls_cep に補足を追加
47	"tcp_snd_dat の補足にパラメータで指定した、送信したいデータの長さ... "を追記
66	udp_snd_dat の補足で奇数番地を偶数番地に修正
68	udp_rcv_dat の補足で奇数番地を偶数番地に修正
68	udp_rcv_dat の解説の一部を変更
69	"udp_rcv_dat の補足にノンブロッキング（tmout =TMO_NBLK ）で、... "を追記
71	udp_set_opt を記載

第 2 版で改訂された項目

ページ	内容
1	"現バージョンでは、リトルエンディアンの処理系に対応していません。"を削除
1	"現バージョンでは、IP フラグメント機能をサポートしていません... "を追記
23	タスクの優先度に関する補足事項を追記
33	tcp_cre_rep の戻値に E_PAR を追加
36	tcp_cre_cep の補足にバッファサイズの制限を追記
42	tcp_con_cep の補足にタイムアウトなし指定時の注意事項を追記
47	tcp_snd_dat の戻値 の E_RLWAI の説明を追記
49	tcp_rcv_dat の戻値 に E_PAR を追加

53	tcp_snd_buf の戻値 に E_PAR を追加
54	tcp_rcv_buf の戻値 に E_WBLK を追加
66	udp_snd_dat の戻値 に E_PAR を追加
66	udp_snd_dat の戻値 の E_RLWAI の説明を追記
66	"udp_snd_dat の補足に送信パケットのポインタは... "を追記
68	udp_rcv_dat の戻値 に E_PAR を追加
69	"udp_rcv_dat の補足に受信パケットを入れるバッファは... "を追記

目次

第 1 章 導入

1.1 はじめに

NORTi Version 4 は、μITRON4.0 仕様準拠のリアルタイム OS「NORTi カーネル」に、TCP/IP プロトコルスタック「NORTi TCP/IP」を追加した製品です。本書では、TCP/IP 部分の説明を行っていますので、NORTi カーネルについては「NORTi Version 4 ユーザーズガイド・カーネル編」を参照してください。また、「NORTi Version4 ユーザーズガイド補足説明書」には本書に記載しきれなかった説明が記載されています。本書と併せてご覧ください。

1.2 特長

いち早くμITRON 仕様 OS に TCP/IP プロトコルスタックを標準で取り込んだ NORTi は、組込みシステムで TCP/IP が初めてというお客様も含め、数多くのユーザー様で採用されている実績があります。

NORTi は数多くのプロセッサに対応しています。そして、同じ CPU ファミリであれば、CPU 型番や応用機種の制限無く使えるコンパイラと同様のユーザーライセンス制が基本のため、導入が容易です。組込み製品の台数に応じたロイヤリティ支払いは一切不要で、お客様の製品コストの削減に貢献します。

リソースの少ない組込みシステムに適した ITRON TCP/IP API 仕様を基に、TCP、UDP、IP、ARP、ICMP、IGMP をサポートした本格的なプロトコルスタックとして実装されています。また、転送速度制御としてスライディングウィンドウ方式を採用。輻輳制御として、スロースタート、Fast Retransmit, Fast Recovery アルゴリズムを採用し、高速性と高信頼性を実現しています。マルチホームにも対応しています。マルチホームとは一つのネットワークインターフェースに複数の IP アドレスを割り当てる機能です。

NORTi TCP/IP は、各コンパイラ/バージョンで動作確認済みのライブラリとして提供されます。さらにライブラリを構成する全てのソースコードが付属し、デバッガでプロトコルスタック内部の動作を追跡することも可能です。

アプリケーション層として、Telnet、FTP、TFTP の各サーバ/クライアント、DHCP クライアント、DNS レゾルバ、SNTP クライアントのソースがサンプルとして付属しています。また、各種 Ethernet コントローラに対応した LAN ドライバを無償サンプルとして(未収録のドライバのいくつかは有償サンプルとして)提供しています。

一部のプロセッサ向けには、IPv6/IPv4 デュアルスタック「NORTi TCP/IPv6」も用意しています。

1.3 制限事項

内部では NORTi カーネルの機能を活用しており、それ以外の OS へ移植した場合の動作は保証できません。

1.4 ファイル構成

NORTi TCP/IP プロトコルスタックを構成するファイルについて説明します。

NORTi¥INC¥

nonet.h	NORTi TCP/IP 標準ヘッダ
nonetc.h	NORTi TCP/IP コンフィグレーションヘッダ
nonets.h	NORTi TCP/IP 内部定義ヘッダ
nonitod.h	数値/文字列変換関数ヘッダ
nonloop.h	ループバックドライバヘッダ

NORTi¥LIB¥(CPU)¥(CCC)¥

n4nxxx.lib	NORTi TCP/IP ライブラリ
n4dxxx.lib	NORTi TCP/IP ライブラリ(デバッグ情報付き)

NORTi¥SRC¥

nonearp.c	ARP モジュールのソース
noneipf.c	IP フラグメントモジュールのソース
nonelan.c	ネットワーク・ドライバ・インターフェースのソース
noneraw.c	RAW モジュールのソース
nonet.c	NORTi TCP/IP サービスコールのソース
nonetip.c	IP モジュールのソース
noneudp.c	UDP モジュールのソース
nonicmp.c	ICMP モジュールのソース
nonigmp.c	IGMP モジュールのソース
nonitod.c	数値/文字列変換関数のソース
nonloop.c	ループバックドライバのソース
nonslib.c	C 標準ライブラリに不足の関数のソース
nontcp1.c	TCP モジュールのソース 1/2
nontcp2.c	TCP モジュールのソース 2/2

フォルダ名の(CPU)は CPU 名略称、(CCC)はコンパイラ名略称を示します。ファイル名の xxx は、対応プロセッサコアを示します。処理系によっては、拡張子が lib でない場合があります。NORTi¥SRC フォルダの各ソースは、コンパイルされてライブラリ(n4nxxx.lib や n4dxxx.lib)へ結合されています。したがって、通常は、これらの各ソースをユーザーがコンパイル/リンクする必要はありません。

nonet.h ・・・・・NORTi TCP/IP の機能を使う全てのソースファイルでインクルードしてください。本ヘッダには、NORTi TCP/IP の全サービスコールのプロトタイプ宣言と、サービスコール呼び出しのために必要な構造体や定数が定義されています。

nonetc.h ・・・・ アプリケーションの１つのソースファイルでのみインクルードしてください。本ヘッダには、プロトコルスタック内部で使用する管理ブロックの実体が定義されています。#include ″nonetc.h″ の上に、通信端点の最大数や各種パラメータの#define を記述することで、NORTi TCP/IP のコンフィグレーションを行うことができます。

nonets.h ・・・・ nonetc.h、および、NORTi¥SRC フォルダの各ソースファイルからインクルードされており、これをアプリケーションから直接インクルードする必要はありません。本ヘッダには、プロトコルスタック内部で使用する構造体や定数などが定義されています。

nonitod.h ･･･ C の標準ライブラリ関数を使わない数値/文字列変換関数の宣言が記述されています。

nonloop.h ･･･ ループバックドライバに関する宣言が記述されています。

n4?xxx.lib ･･ NORTi¥LIB¥(CPU)¥(CCC)フォルダのライブラリ n4nxxx.lib を、ユーザープログラムとリンクしてください。あるいは、プロトコルスタック内部をデバッガで追うためには、デバッグ情報付きの n4dxxx.lib の方をリンクしてください。これらのライブラリには、LAN ドライバとアプリケーション層のサンプルを除く NORTi TCP/IP の機能がすべて結合されています。

nonearp.c ･･･ ARP(アドレス解決プロトコル)の制御部分が記述されています。

noneipf.c ･･･ IP フラグメントの制御部分が記述されています。

nonelan.c ･･･ LAN ドライバインターフェースに関連する部分が記述されています。

noneraw.c ･･･ RAW 端点の制御部分が記述されています。

nonet.c ･････ 以下のソースから共通に呼び出される関数や、いずれにも含まれないその他の関数が記述されています。

nonetip.c ･･･ プロトコルスタックの最も基本的な部分である IP(インターネット・プロトコル)の制御部分が記述されています。

noneudp.c ･･･ UDP(ユーザ・データグラム・プロトコル)の制御部分が記述されています。

nonicmp.c ･･･ ICMP(インターネット・コントロール・メッセージ・プロトコル)の制御部分が記述されています。Echo 処理を行うだけの簡単なものなので、必要ならユーザーにて機能拡張してください。

nonigmp.c ･･･ IGMP(インターネット・グローバル管理プロトコル)の制御部分が記述されています。

nonitod.c ･･･ C の標準ライブラリ関数を使わない数値/文字列変換関数が記述されています。

nonloop.c ･･･ ループバックドライバ処理が記述されています。

nonslib.c ･･･ コンパイラによって C の標準ライブラリに不足している関数が記述されています。

nontcp?.c ･･･ TCP(トランスミッション・コントロール・プロトコル)の制御部分が記述されています。

1.5 用語

通信端点

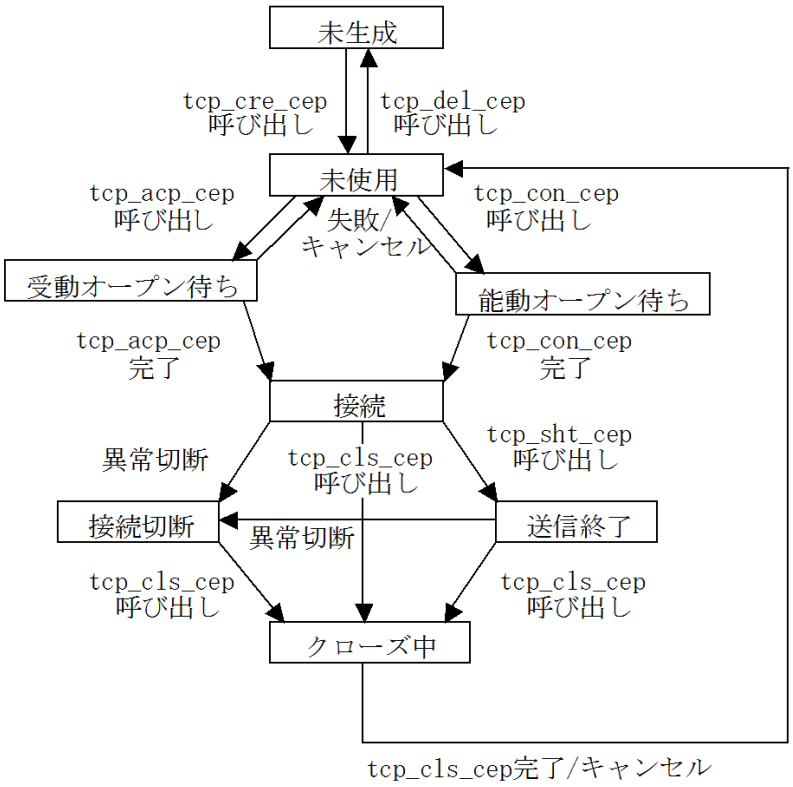
TCP のためのオブジェクト(サービスコールの操作対象)として、「TCP 受付口」と「TCP 通信端点」の 2 種類が用意されています。TCP 受付口は、受動オープンで相手側からの接続要求を待ち受ける際に、TCP 通信端点と共に使用されます。

UDP のためのオブジェクトとしては、「UDP 通信端点」が用意されています。

サービスコールやパラメータの名称において、TCP 受付口(TCP Reception Point)は rep と略されています。TCP 通信端点(TCP Communication End Point)や UDP 通信端点(UDP Communication End Point)は cep と略されています。各通信端点は、1 から始まる ID 番号で区別されます。

TCP 通信端点の状態

TCP 通信端点は、下記の「未生成」～「クローズ中」の 8 状態を遷移します。



「未生成」以外の 7 状態のいずれかにある場合を「生成済み」、「未生成」と「未使用」以外の 6 状態のいずれかにある場合を「使用中」と呼びます。

また、「接続」と「送信終了」と「接続切断」を除く 5 状態のいずれかにある場合を「未接続」と呼びます。

サービスコール

ITRON TCP/IP API 仕様に定義されている、アプリケーションから呼び出される関数群を、サービスコールといいます。

タイムアウト

発行したタスクが待ち状態になる可能性のあるサービスコールには、タイムアウト機能が用意されています。タイムアウトとして指定された時間を経過してもサービスコールが完了しない場合、内部処理が中止されてサービスコールはエラーでリターンします。

ノンブロッキング

発行したタスクが待ち状態になる可能性のあるサービスコールには、ノンブロッキング機能も用意されています。ノンブロッキング指定では、サービスコールは即座にリターンしますが、処理はプロトコルスタック内部で継続しています。そのため、ノンブロッキング指定のサービスコール(ノンブロッキングコール)では、処理の完了をタスクが知ることができません。そこで次のコールバック機能が用意されています。

コールバック

アプリケーションがあらかじめ指定しておいた関数(コールバックルーチン)を、システム側(この場合にはプトロコルスタック側)から呼び出すことを、コールバックといいます。
ノンブロッキングコールで開始した処理の完了は、コールバックで通知されます。また、UDP パケット受信といったイベントも、コールバックで通知されます。

省コピーAPI

プロトコルスタック内での送受信データのコピー回数を減らすことのできる「省コピーAPI」が、TCP のサービスコールには用意されています。省コピーAPI では、プロトコルスタック内部で管理するメモリ領域に対してアプリケーションが直接読み書きを行うため、データのコピー回数を１回だけ省略できます。

パケット

ネットワーク上のデータの固まりは、パケットやデータグラム(TCP ではセグメント、Ethernet はフレーム)といった言葉で表現されますが、本書では、パケットに統一してあります。UDP パケットは、UDP データグラムと同じものです。TCP パケットは、TCP セグメントや TCP データグラムと同じものです。Ethernet パケットと Ethernet フレームは同じものです。

第 2 章 プロトコルスタックの構成

NORTi TCP/IP プロトコルスタックは下記のモジュールから構成され、本章では IGMP を除くモジュールについて説明しています。(IGMP については「NORTi Version 4 ユーザーズガイド補足説明書」の方をご覧下さい)

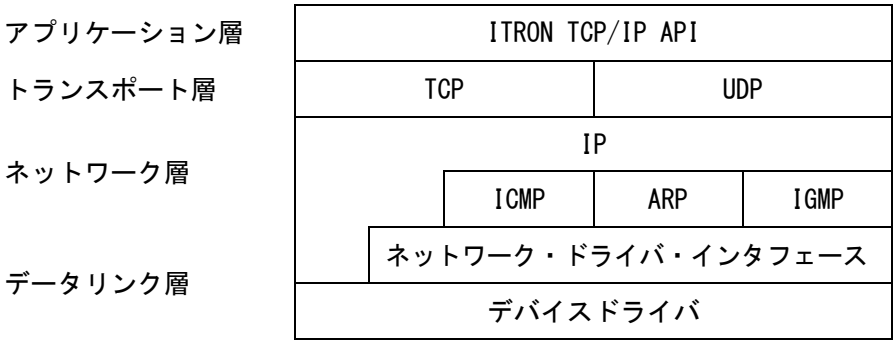
ネットワーク・ドライバ・インタフェース
デバイスドライバ
IP モジュール
ARP モジュール
ICMP モジュール
IGMP モジュール
UDP モジュール
TCP モジュール

ここで説明する内容は、ITRON TCP/IP API 仕様には記載されていない NORTi TCP/IP 独自のものです。特に、デバイスドライバのインターフェースや ARP や ICMP や IGMP については、ITRON TCP/IP API 仕様では規定されていません。

2.1 概要

階層構造

NORTi TCP/IP プロトコルスタックは以下のようなモジュールで構成されています。



プロトコル制御タスク

NORTi TCP/IP プロトコルスタックは次の 2 つのタスクと 1 つの周期ハンドラから構成されています。

IP 受信タスク	ネットワークインターフェース毎に 1 個
IP 送信タスク	ネットワークインターフェース毎に 1 個
TCP/IP タイマ(周期ハンドラ)	全体で 1 個

TCP、UDP、IP、ICMP、IGMP、ARP、ネットワーク・ドライバ・インタフェース、デバイスドライバの全てが、IP 受信タスク/IP 送信タスクと TCP/IP タイマで制御されています。つまり、各モジュール毎のタスクや周期ハンドラは存在しません。

プロトコルスタックのメモリプール

プロトコルスタック内部で使用されるメモリプールには、次のものがあります。いずれも固定長で、メモリブロック数はコンフィグレーション可能です。

Ethernet パケット用メモリプール	全体で 1 個
UDP ヘッダ用メモリプール	UDP 通信端点毎

Ethernet パケット用メモリプールは、送受信する Ethernet パケット用の領域を提供します。このメモリプールは、UDP の送信を除く全てのプロトコルの送信と受信で共有されており、全体で 1 個だけ存在します。このメモリプールから獲得できるメモリブロック数のデフォルトは 16 個です。各メモリブロックのサイズは 1560 バイトに固定で、各プロトコルのヘッダとデータを格納でき、データリンク層での Ethernet パケットの送受信だけでなく各層でのパケットの解析や組立ての処理にもそのまま使われます。つまり、各プロトコル層間でデータを転送する際に、新たな領域の獲得やコピーは行われません。

UDP 送信のみが特殊で、アプリケーションから渡されたデータ領域をそのまま Ethernet パケット領域とするために、別に UDP ヘッダ用のメモリプールを設けてあります。このメモリプールは生成した UDP 通信端点毎に存在し、メモリブロックのサイズは 80 バイト、メモリブロック数のデフォルトは 2 個です。

プロトコルスタックのメールボックス

プロトコルスタック内部で使用されるメールボックスには、次のものがあります。メールボックスでは、領域へのポインタのみが受け渡されるため、データのコピーが発生せず高速に通信させることが可能です。

送信パケットキュー	**ネットワークインターフェース毎に 1 個**
送信リトライキュー	**ネットワークインターフェース毎に 1 個**
UDP 受信キュー	**UDP 通信端点毎**

送信パケットキューとして使用されるメールボックスには、各モジュールから送信されるパケットが一旦キューイングされ、順にデータリンク層へ渡されます。他に、送信を保留したり再送したりするための送信リトライキューとして使用されるメールボックスがあります。

UDP 受信のみが特殊で、アプリケーションから指定された受信バッファ領域へ直接受信データをコピーするために、UDP 受信バッファをキューイングするためのメールボックスを設けてあります。このメールボックスは生成した UDP 通信端点毎に存在します。

タイムアウトとキャンセル

タイムアウトは、サービスコールを発行したアプリケーションのタスクレベルで監視しています。タイムアウトが起きた場合には、キューイングされているパケットがサーチされ、取り外されます。ペンディング中処理のキャンセルや通信端点削除のサービスコールの場合も同様です。

NORTi TCP/IP では次のようなパラメータをタイムアウトに設定できます。

タイムアウトなし (tmout = TMO_FEVR)
タイムアウトあり (tmout = 1～0x7fffffff)
ポーリング (tmout = TMO_POL)
ノンブロッキング (tmout = TMO_NBLK)

タイムアウトなし(tmout = TMO_FEVR)を使用すると、接続先がダウンした場合に、長時間あるいは永久にサービスコールから戻らない可能性があります。なるべくタイムアウトありをパラメータに使用して下さい。

2.2 ネットワーク・ドライバ・インターフェース

構成

ネットワーク・ドライバ・インターフェースは LAN コントローラを抽象化した関数群でデバイスドライバとネットワーク層との間に入るモジュールです。ネットワーク層は、送受信待ち関連の関数を除き、デバイスドライバの関数を直接コールせずにネットワーク・ドライバ・インターフェースを中継します。ネットワーク・ドライバ・インターフェースをカスタマイズすることで、Ethernet 以外のデバイスにも対応できます。
送信パケットの Ethernet ヘッダの設定(MAC アドレス等)を行うのは、本来、データリンク層の仕事ですが、NORTi TCP/IP プロトコルスタックでは、デバイスドライバの実装をシンプルにするため、IP 層が行っています。また、パフォーマンス向上のため、受信したパケット全体を一度に読み出すのではなく、必要最小限のデータだけを読み出せるインターフェースとなっています。

受信パケット長を得る

[形式] UH lan_received_len(void)

[戻値] 受信したバイト数

デバイスドライバで受信しているパケット全体のサイズを返します。lan_read_pkt 関数に先だってコールされます。

受信パケット読み出し

[形式] BOOL lan_read_pkt(void *buf, int len)
buf 読み出すためのバッファ
len 読み出すバイト数

[戻値] TRUE 正常終了
FALSE エラー

デバイスのバッファメモリから、指定サイズだけの受信データを読み出します。1 つのパケットを読み出すために、この関数が繰り返し呼ばれる場合があります(1 度に全データを読み出さない)。

受信パケット末尾まで読み出し

[形式] BOOL lan_read_pkt_end(void *buf, int len, int bufsz)
buf 読み出すためのバッファ
len 読み出すバイト数
bufsz 読み出しバッファのサイズ

[戻値] TRUE 正常終了
FALSE エラー

ドライバが実際に受信したサイズよりも bufsz が小さければ bufsz のサイズを読み出しし、残りのデータは破棄されます。受信したサイズよりも bufsz が大きければ、受信した全てのパケットを読み出します。lan_read_pkt 関数がコールされた後に、残りのデータを全て読み出すためにコールされます。

受信パケット破棄

[形式] void lan_ignore_pkt(void)

[戻値] なし

lan_read_pkt 関数で読み出し途中のパケットを破棄します。不要な受信パケットのためにこれ以上データを読み出す必要がない場合、lan_read_pkt 関数や lan_read_pkt_end 関数に代わって、この関数がコールされます。

送信パケットの書き込み

[形式] BOOL lan_write_pkt(const void *head, int hlen, const void *data, int dlen)
head 書き込むデータ前半部
hlen データ前半部のバイト数
data 書き込むデータ後半部
dlen データ後半部のバイト数

[戻値] TRUE 正常終了
FALSE エラー

デバイスのバッファメモリへ、送信データを書き込みます。UDP 送信ではヘッダ部とデータ部が別々の領域に格納されているために、head/hlen と data/dlen に別れていますが、それ以外では、data = NULL, dlen = 0 として、head と hlen のみが使われます。

LAN ドライバのエラー処理

[形式] ER lan_error(ER ercd)

[戻値] エラーコード

デバイスドライバ関数がエラーを返したときに、この関数が呼ばれます。標準では何も処理していません。

2.3 デバイスドライバ

構成

NORTi TCP/IP の標準的な LAN ドライバは、タスクを使用しません。デバイスドライバは、ネットワーク・ドライバ・インターフェースから呼び出される関数群と、IP 受信タスク/IP 送信タスクを起床する割込みハンドラから構成されます。デバイスドライバでは、送受信するパケットがどのプロトコルのものであるかは関知していません。
lan_wai_snd 関数と lan_wai_rcv 関数のみが IP 層から直接呼ばれます。それ以外の関数は全てネットワーク・ドライバ・インターフェースから呼ばれます。デバイスドライバの関数群はリエントラントに実装する必要はありません。

デバイスの初期化

[形式] ER lan_ini(UB *macaddr)
macaddr MAC アドレス(6 バイトの配列)

[戻値] エラーコード

プロトコルスタックの初期化関数 tcp_ini から呼び出され、割込みハンドラ定義やデバイスのレジスタ初期設定が行われます。

デバイスの終了

[形式] ER lan_ext_dev(void)
[戻値] エラーコード

プロトコルスタックの終了関数 tcp_ext から呼び出され、初期化時に取得したデバイスドライバで使用するリソースの開放などを行います。

割込みハンドラ

[形式] INTHDR lan_int(void)

受信割込みの場合、受信完了を lan_wai_rcv 関数で待っている IP 受信タスクを起床します。送信割込みの場合、送信可を lan_wai_snd 関数で待っている IP 送信タスクを起床します。
割込みハンドラ定義はデバイスドライバ自身で行うため、lan_int 以外の関数名でも構いません。割込みサービスルーチンとしても実装できます。

受信割込み待ち

[形式] ER lan_wai_rcv(TMO tmout)
tmout タイムアウト指定

[戻値] エラーコード

デバイスのバッファメモリに受信パケットがない場合は、これをコールした IP 受信タスクが、受信完了割込み待ち状態に入ります。割込みハンドラで、この IP 受信タスクの待ちは解除されます。なお、タイムアウトなし(tmout = TMO_FEVR)として IP 受信タスクからコールされているので、タイムアウト機能は、現状では使用されていません。

送信割込み待ち

[形式] ER lan_wai_snd(TMO tmout)
tmout タイムアウト指定

[戻値] エラーコード

デバイスのバッファメモリに空きがなくパケットを送信できない場合は、これをコールした IP 送信タスクが送信可割込み待ち状態に入ります。割込みハンドラで、この IP 送信タスクの待ちは解除されます。なお、タイムアウトなし(tmout = TMO_FEVR)として IP 送信タスクからコールされているので、タイムアウト機能は、現状では使用されていません。

受信パケット長を得る

[形式] ER lan_get_len(UH *len)
len 受信したパケットのバイト数

[戻値] エラーコード

ネットワーク・ドライバ・インターフェースの lan_received_len 関数からコールされます。これからデバイスのバッファメモリより読み出すべきデータ長を返します。

受信パケット読み出し

[形式] ER lan_get_pkt(void *buf, int len)
buf 読み出すためのバッファ
len 読み出すバイト数

[戻値] エラーコード

ネットワーク・ドライバ・インターフェースの lan_read_pkt 関数からコールされます。デバイスのバッファ

メモリから、指定サイズだけのデータを読み出します。1 つのパケットを読み出すために、この関数が繰り返し呼ばれる場合があります（1 度に全データを読み出さない）。渡される buf のアドレスは 2 の倍数(ワード境界)です。lan_read_pkt 関数が繰り返しコールされる場合、最後を除いて len は必ず偶数です。デバイスによって、lan_get_pkt 関数がコールされた都度、受信データを読み出せるタイプと、一気に全データを読み出す必要のあるタイプとがあります。後者の場合、lan_get_pkt 関数がコールされた段階で一気に受信パケットの全データを読み出し、それを lan_get_pkt がコールされるまで保存するための受信バッファをデバイスドライバ内部に持つ必要があります。

受信パケット読み出し終了

[形式] ER lan_get_end(void)

[戻値] エラーコード

ネットワーク・ドライバ・インターフェースの lan_read_pkt 関数から lan_get_pkt 関数が(繰り返し)コールされて、ちょうど最後のデータが読み出された後に、この関数がコールされます。デバイスドライバでは、必要ならば、受信パケットの読み出し完了処理、すなわち、次の受信パケットを読み出せる準備を行います。

受信パケット読み飛ばし

[形式] ER lan_skp_pkt(int len)
len 読み飛ばすバイト数

[戻値] エラーコード

ネットワーク・ドライバ・インターフェースの lan_ignore_pkt 関数からコールされます。不要な受信パケットのため、これ以上データを読み出す必要がない場合、lan_get_end 関数に代わってこの関数がコールされます。len は、ネットワーク・ドライバ・インターフェースで管理している破棄されるべきデータ長ですが、デバイスのバッファメモリからの空読みを行う必要のない場合は無視して構いません。多くのデバイスで、lan_get_end 関数と同じ処理となります。

送信パケット長を設定

[形式] ER lan_set_len(int len)
len 送信するバイト数

[戻値] エラーコード

デバイスドライバの lan_set_len 関数は、これから送信するパケットのバイト数を指定するために、ネットワーク・ドライバ・インターフェースの lan_write_pkt 関数から呼び出されます。

送信パケットの書き込み

[形式] ER lan_put_pkt(const void *data, int len)
data 書き込むデータ
len データのバイト数 (> 0)

[戻値] エラーコード

ネットワーク・ドライバ・インターフェースの lan_write_pkt 関数から呼び出されます。デバイスのバッファメモリへ、データを書き込みます。１つのパケットを送信するために、この関数が繰り返し呼ばれる場合があります(１度に全データを書き込まない)。
渡される data のアドレスは 2 の倍数(ワード境界)です。lan_put_pkt 関数が繰り返しコールされる場合、最後を除いて len は必ず偶数です。
デバイスによって、lan_set_len 関数がコールされた段階で送信パケットの書き込み動作を開始できるものと、続く一連の lan_put_pkt 関数(の繰り返し)と lan_put_end 関数を待って書き込み動作を開始しなければならないものとがあります。後者の場合、lan_put_pkt で渡されるデータを保存するための送信バッファをデバイスドライバ内部に持つ必要があります。

送信パケットが 60 バイト未満の場合のダミー書き込み

[形式] ER lan_put_dmy(int len)
len ダミーデータのバイト数 (> 0)

[戻値] エラーコード

lan_put_pkt 関数で書き込んだデータサイズが 60 バイトに満たない場合、lan_put_end 関数の前に、この lan_put_dmy 関数が呼び出されます。デバイスドライバでは指定バイト数分のゼロの値のデータを、デバイスのバッファメモリへ書き込みます。デバイスが自動的に 60 バイトに足りないデータを補うパディング機能を持っている場合、この関数は何もしなくて構いません。

送信パケット書き込み終了

[形式] ER lan_put_end(void)

[戻値] エラーコード

lan_put_pkt 関数や lan_put_dmy 関数が呼ばれた最後に、この lan_put_end 関数が呼び出されます。LAN コントローラに送信を開始させます。より上位でタイムアウトやキャンセルが発生した場合でも、すでにデバイスドライバに渡された送信パケットは送出されてしまいます。

2.4 IP モジュール

使用する資源

IP モジュールを構成するタスクには、IP 送信タスクと IP 受信タスクの 2 つが存在します。また、送信パケットキューと呼ぶメールボックスが存在します。IP モジュールの IP 送信タスク/IP 受信タスクでは、IP だけでなく、ARP、ICMP、IGMP、UDP、TCP の制御も行っています。

パケット受信

IP 受信タスクは、受信割込みハンドラから起床され、デバイスドライバ関数を利用して、パケットを受信します。受信したパケットのヘッダはこの場で解釈され、ARP、ICMP、IGMP、UDP、TCP の各プロトコル別の処理を行います。

パケット送信

送信パケットキューへ送られてきたパケットにより、IP 送信タスクは起床されます。送られてくるパケットには、ARP、ICMP、IGMP、UDP、TCP の各プロトコルのパケットがあります。IP 送信タスクでは、受け取った送信パケットに、IP ヘッダの設定を行います。さらに、Ethernet ヘッダを設定して、デバイスドライバ関数を利用して送信を行います。すぐに、送信できない場合は、送信割込みハンドラから起床されるのを待ちます。

2.5 ARP モジュール

使用する資源

ARP 専用のタスクは存在しません。NORTi TCP/IP では、IP 受信タスク/IP 送信タスクが、ARP 処理も受け持っています。ARP とは、IP アドレスから相手局の MAC アドレスを検索するためのプロトコルで、アプリケーションからは見えないところで実行されています。

ARP テーブル

プロトコルスタック内部には、ARP テーブルと呼ばれる、IP アドレスに対応する MAC アドレスを記憶するための配列があります。ARP テーブルには、自分宛の受信パケットから得た、宛先の IP アドレスと MAC アドレスの対応が記憶されます。ARP テーブルが一杯になった場合は、古いものから捨てられます。ARP テーブルに登録可能な数はコンフィグレーション ARP_TABLE_CNT で決まります。接続するホストがこれよりも大きい場合は、このコンフィグレーションを変更してください。ARP テーブルのリフレッシュタイミングは ARP_CACHE_TOUT と ARP_FLUSH_TOUT によって決定されます。もし、ARP テーブルの個々のエントリが ARP_CACHE_TOUT で指定した時間使用されない場合、エントリは ARP テーブルから削除されます。また、ARP_FLUSH_TOUT 時間間隔で ARP テーブルの全エントリは強制的にクリアされます。

ARP 問い合わせ

IP 送信タスクで送信しようとしたパケットの宛先 MAC アドレスが、この ARP テーブルに記憶されていない場合、IP 送信タスクは、先に ARP パケットを送信します。具体的には、ARP パケットを作成して、送信パケットキューへ送ります。本来のパケットの送信は、ARP 応答を受信するまで保留されます。
そして、IP 受信タスクが、ARP 応答のパケットを受信してアドレスが解決すると、IP 送信タスクは、保留していたパケットを送信します。

ARP 応答

一方、自分のアドレスを問い合わせる ARP パケットを受信した IP 受信タスクは、その応答パケットを作成し、送信パケットキューへ送ります。この ARP パケット領域には、受信した ARP パケットの領域をそのまま使います。

ARP のタイムアウト

アドレス未解決で保留したパケットが溜まるとメモリを圧迫します。そこで、一定時間が経過しても、アドレスが解決しない場合は、再度 ARP 問い合わせを行い、最終的には、送信保留したパケットは破棄されます。破棄されたパケットを送信したアプリケーションのタスクには、エラーが返ります。ノンブロッキングコールだった場合は、キャンセルされたことをコールバックでアプリケーションへ通知します。

2.6 ICMP モジュール

使用する資源

ICMP 専用のタスクは存在しません。IP 受信タスク/IP 送信タスクが、ICMP 処理も受け持っています。ICMP とは、主にネットワークの診断を目的としたプトロコルで、アプリケーションからは見えないところで実行されています。

Echo 処理

ping コマンドで使用する Echo の ICMP パケットを受信した IP 受信タスクは､その応答パケットを作成し、送信パケットキューへ送ります。この ICMP パケット領域には、受信した ICMP パケットの領域をそのまま使います。

icmp_def_cbk

機 能 ICMP 受信コールバック関数の登録

形 式 ER icmp_def_cbk(T_ICMP_CB *b, ICMP_CALLBACK callback);

b	ICMP 制御ブロックへのポインタ
callback	登録するコールバック関数

戻 値

E_OK	正常終了
負の値	異常終了

解 説 ICMP パケットを受信した時に呼び出される関数を登録できます。
ICMP 制御ブロックはチェーン構造で管理され、複数のコールバック関数を登録できます。ICMP 制御ブロック領域は、ユーザーが変数としてこの領域を確保して渡すだけで、初期設定の必要はありません。

コールバック

機 能 ICMP の受信コールバック

形 式 VP *callback (T_ICMP_CB *b, T_IP4 *ip, T_ICMP_MSG *icmp, INT len);

b	ICMP 制御ブロックへのポインタ(通常、使用しない)
ip	IP パケットへのポインタ
icmp	ICMP メッセージへのポインタ
len	パケット長

```
typedef struct t_ip4 {
    struct t_ip4 *next;   /* Message Pointer for NORTi Mailbox */
    T_CTL_HEADER ctl;     /* Header for Internal Control */
```

```
    T_ETH_HEADER eth;     /* Ethernet Header */
    B data[PATH_MTU4];    /* Data (Variable Size)*/
    T_IP4_HEADER ip;      /* IP4 Header (without option data) */
} T_IP4;

typedef struct t_icmp_msg {
    UB type;                  /* ICMP Type */
    UB code;                  /* ICMP Code */
    UH cs;                    /* Checksum */
    UH id;                    /* Identifier */
    UH seq;                   /* Sequence Number */
    UB data[IP_HEADER_SZ+8]; /*Option Data (Variable Size)*/
} T_ICMP_MSG;
```

解 説 ICMP受信コールバック関数で指定する関数です。関数名は任意です。
ICMP パケットを受信すると、この関数が呼び出されます。

icmp_snd_dat

機 能 ICMP パケット送信

形 式 ER icmp_snd_dat(UW ipaddr, UW dstaddr, T_ICMP_HEADER *icmp, VP data, INT len);

ipaddr	自局の IP アドレス
dstaddr	相手の IP アドレス
icmp	送信する ICMP ヘッダへのポインタ
data	送信する ICMP データへのポインタ
len	データ長

```
typedef struct t_icmp_header {
    UB type;              /* ICMP Type */
    UB code;              /* ICMP Code */
    UH cs;                /* Checksum */
    UH id;                /* Identifier */
    UH seq;               /* Sequence Number */
} T_ICMP_HEADER;
```

戻 値

E_OK	正常終了
負の値	異常終了

解 説 任意の ICMP パケットを送信することができ、上述のコールバック関数の内部からも呼び出せます。ping_command に使用例があります。

2.7 UDP モジュール

使用する資源

UDP 専用のタスクは存在しません。NORTi TCP/IP では、IP 受信タスク/IP 送信タスクが、UDP 処理も受け持っています。UDP 通信端点毎に、UDP 受信キューと呼ぶメールボックスが存在します。

UDP パケット送信

アプリケーションが送信要求した UDP パケットには、サービスコールで、UDP ヘッダが追加されて、送信パケットキューへ送られます。IP 送信タスクでこの UDP パケットの送信が終わると、送信完了を待っていたアプリケーションのタスクの待ちが解除されます。ノンブロッキングコールだった場合は、送信完了したことをコールバックでアプリケーションへ通知します。

UDP パケット受信

UDP の受信では、先にアプリケーションのタスクが、サービスコールを使って、UDP パケットを受け取る領域を指定しておきます。IP 受信タスクが受信した UDP パケットは、その場で UDP ヘッダの解釈まで行われます。そして、該当する UDP 通信端点に受信要求がある場合は、指定された領域へ UDP パケットを格納します。サービスコール内で、受信を待っていたタスクは、この UDP パケットを受け取って待ち解除となります。ノンブロッキングコールだった場合は、受信完了したことをコールバックでアプリケーションへ通知します。UDP 受信要求がなされていない場合は、UDP パケット受信を通知するコールバックを行います。

重要！

UDP は TCP と異なり内部にデータを保管するバッファを持っていません。そのため、プロトコルスタックが UDP のパケットを受信した時点でアプリケーションタスクが udp_rcv_dat で待っていない場合、パケットが破棄されてしまいます。UDP パケットを連続して受信する場合は、コールバック関数を使用してください。コールバック関数は受信があると、プロトコルスタックから呼び出されます。これは割込みハンドラに似ています。

2.8 TCP モジュール

使用する資源

TCP 専用のタスクは存在しません。NORTi TCP/IP では、IP 受信タスク/IP 送信タスクが、TCP 処理も受け持っています。

IP モジュールとの関係

IP 受信タスクでは､受信した TCP パケットの宛先/発信元 IP アドレスと宛先/発信元ポート番号を判別し、コネクションの確立した（確立中を含む）TCP 通信端点があるなら、その TCP 通信端点で TCP パケットを処理します。TCP パケットが接続要求(SYN)の場合、該当する TCP 受付口があるなら、TCP 通信端点のコネクションを確立します。コネクションの確立していないパケットや、受付口のない SYN パケットは破棄されます。IP 送信タスクでは、TCP モジュールから送られた TCP パケットに IP ヘッダを設定して、送信します。TCP モジュールが管理する受信バッファへの TCP データ格納は IP 受信タスクが行います。同じく送信バッファからの TCP データの読み出しは、IP 送信タスクが行います。

キープアライブ

TCP はコネクション確立後､一定時間通信が行われない場合､キープアライブのパケットが送信されます。これはリモートホストが正常に動作しているかを確認するための機能です。キープアライブのパケットは単に ACK フラグのみが ON になったパケットです。
リモートホストが正常に動作していれば、このパケットに対する応答を返しますが、応答が無ければコネクションは切断されます。このとき受信関数で待っている場合は E_TMOUT が返ります。キープアライブは TCP_KTIME_INI,TCP_KTIME_PRO,TCP_KTIME_SUC でコンフィグレーション可能です｡TCP_KTIME_INI(7200 秒)後に最初のキープアライブプローブを送信、無応答の場合、以後は TCP_KTIME_SUC(75 秒)間隔でパケットを送信し､最初に送信したキープアライブプローブからの時間の合計が TCP_KTIME_PRO(600 秒)後にエラー終了します。

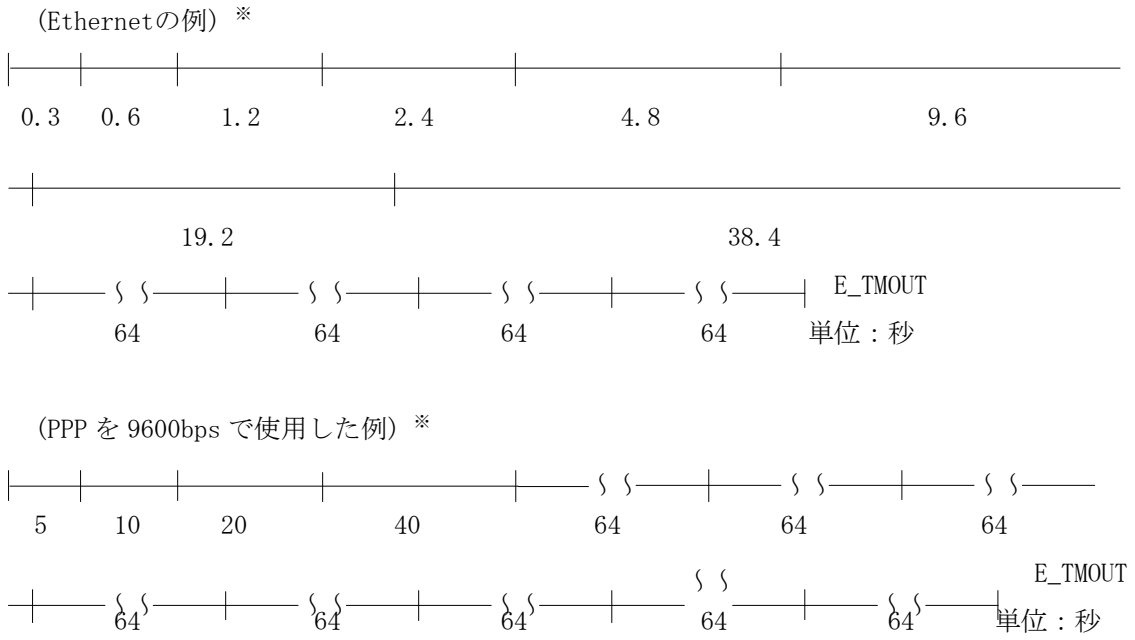
7200 seconds 75sec 75sec = 600 sec

ストリーム型通信方式

UDP や IP がデータグラムというひとかたまりのパケットを送信するのに対し、TCP では連続する区切りのないデータ列を送受信する方式をとっています。これをストリーム型通信方式といいます。これは例えば 1000 バイトのデータを送信した場合、受信側は 1000 バイト全部を一度に受信できる場合もあれば、これが 500 バイト単位でバラバラに受信されたり、直前に送られたデータとくっついて受信されることもありえるということです。これをひとかたまりのデータとして受信したい場合は、パケットの先頭に長さをつけたり、パケットの末尾に区切り文字を付けることで判断する方法があります。

再送信

信頼性が保証されていない IP ネットワークでは、転送中にパケットを喪失することがあります。TCP ではこのようなエラーを回復するために、一定時間経過しても送信先ホストから ACK パケットが得られない場合はタイムアウトにより再度、同一パケットを送信します。このタイムアウト時間はパケットの送信時の相手からの応答時間（ラウンドトリップタイム）を測定して、回線の速度に合わせて動的に変化させる仕組みをとっています。NORTi TCP/IP ではこのタイムアウト時間の上限値を TCP_RTO_UBOUND で下限値を TCP_RTO_LBOUND でコンフィグレーションが可能です。ACK が続けて受信できない場合は再送信を繰り返しますが、その間隔は指数倍されます。例えばデータ送信と FIN 送信の場合、相手から ACK による応答がなければ以下のような間隔で再送信されます。



※ この例は社内で評価を行った代表的な例です。タイムアウトの時間はラウンドトリップタイムを元に決定されますので、実際に使用する回線の速度によって時間が異なります。タイムアウト付きの API を使用した場合、指定した時間内に処理が完了しなければ、E_TMOUT が返りますが、再送信は継続されます。tcp_cls_cep を短いタイムアウト(10MSEC 程度)で呼び出すと RST が送信され再送信が停止します。

第 3 章 コンフィグレーション

定義

コンフィグレーションヘッダ nonetc.h を#include する前に、次の様な定数を記述することにより、コンフィグレーションが行えます。

()はデフォルト値で指定がない場合に使用されます。また?は必ず指定が必要です。

```
#define TCP_REPID_MAX      (4)    TCP 受付口 ID の上限
#define TCP_CEPID_MAX      (4)    TCP 通信端点 ID の上限
#define UDP_CEPID_MAX      (4)    UDP 通信端点 ID の上限

#define TCP_TSKID_TOP      ?      TCP/IP で用いるタスクの先頭 ID
#define TCP_SEMID_TOP      ?      TCP/IP で用いるセマフォの先頭 ID
#define TCP_MBXID_TOP      ?      TCP/IP で用いるメールボックスの先頭 ID
#define TCP_MPFID_TOP      ?      TCP/IP で用いる固定長メモリプール先頭 ID

#define PRI_IP_SND_TSK     (4)    IP 送信タスクの優先度
#define PRI_IP_RCV_TSK     (4)    IP 受信タスクの優先度
```

> 重要！
>
> IP 送受信タスクの優先度は必ず同一にし、プロトコルスタックを使用するアプリケーションのタスクよりも高い優先度で設定してください。IP 送受信タスクがアプリケーションタスクよりも低い場合、予期しない動作を引き起こすことがあります。

```
#define SSZ_IP_SND_TSK     (1024)  IP 送信タスクのスタックサイズ
#define SSZ_IP_RCV_TSK     (1024)  IP 受信タスクのスタックサイズ

#define ETH_QCNT           (16)    Ethernetパケットの最大キューイング数 ※1
#define UDP_QCNT           (2)     UDP パケットの最大キューイング数

#define ARP_TABLE_CNT      (8)     ARP テーブルのエントリ数
#define ARP_CACHE_TOUT     (120)   ARP エントリのリフレッシュ時間間隔(2 分)
#define ARP_RET_INTVAL     (2)     ARP の再送間隔(2 秒)
```

※1 ETH_QCNT は UDP と ICMP の IP フラグメント処理でも使用されますので、一度に送受信したいバイト数を 1480 で割った値を加算してください。(例) 30KB の場合、30KB÷1480≒21 を加算。

#define ARP_FLUSH_TOUT	(1200)	ARP テーブルの強制リフレッシュ時間間隔(1200 秒)
#define IP_DEF_TTL	(32)	TTL(Time To Live)
#define TCP_SYN_RETRY	(3)	TCP SYN 送信の最大リトライ回数
#define TCP_DATA_RETRY	(12)	TCP データ送信の最大リトライ回数
#define TCP_RTO_INI	(3000)	TCP 再送時間の初期値(3 秒)
#define TCP_RTO_UBOUND	(64000)	TCP 再送タイムアウトの上位境界(64 秒)
#define TCP_RTO_LBOUND	(300)	TCP 再送タイムアウトの下位境界(300 ミリ秒)
#define TCP_KTIME_INI	(7200)	キープアライブパケットが送信されるまでの時間 (2 時間) 0 以下の場合、キープアライブは送信されません。
#define TCP_KTIME_PRO	(600)	キープアライブタイムアウト(10 分)
#define TCP_KTIME_SUC	(75)	キープアライブパケットの送信インターバル (75 秒)

```
#include "nonetc.h"
```

キープアライブの設定を行う場合、以下のルールに従ってください。 TCP_RTO_UBOUND < TCP_KTIME_INI TCP_KTIME_PRO < TCP_KTIME_INI TCP_KTIME_SUC = TCP_KTIME_PRO / n ('n'はキープアライブプローブ数)

ID の自動割り当て

NORTi TCP/IPは各資源IDの自動割り当てに対応しています。
各資源の先頭IDを0で設定した場合NORTi TCP/IPは内部のIDは全て自動的に割り当てられます。

```
#define TCP_TSKID_TOP       0     TCP/IPで用いるタスクの先頭ID
#define TCP_SEMID_TOP       0     TCP/IPで用いるセマフォの先頭ID
#define TCP_MBXID_TOP       0     TCP/IPで用いるメールボックスの先頭ID
#define TCP_MPFID_TOP       0     TCP/IPで用いる固定長メモリプール先頭ID
```

端点、受付口の ID 自動割り当て

xxx_vcre_xxx システムコールによりオブジェクトを生成すると、空いていた ID 番号を戻り値として取得できます。xxx_vcre_xxx システムコールは NORTi TCP/IP 独自の拡張機能です。

ローカル IP アドレスと MAC アドレスの設定

ローカル IP アドレスと MAC アドレスは以下の変数で定義します。

```
UB default_ipaddr[]   = { 192, 168, 1, 99 };
UB ethernet_addr[]    = { 0x00, 0x00, 0x12, 0x34, 0x56, 0x78 };
```

これらの変数は各サービスコールが呼ばれる前に必ず設定する必要があります。

デフォルトゲートウェイとサブネットマスクの設定

デフォルトゲートウェイとサブネットマスクは以下の変数で定義します。

```
UB default_gateway[]    = { 0, 0, 0, 0 };
UB subnet_mask[]        = { 255,255,255,0 };
```

これらの変数は各サービスコールが呼ばれる前に必ず設定する必要があります。
ゲートウェイを使用しない場合は全て 0 を設定してください。

第 4 章 共通定義

4.1 バイトオーダ変換

ネットワーク上を流れる TCP/IP ヘッダ部のデータの並び（ネットワークバイトオーダー）は、ビッグエンディアンです。プロセッサのデータの並び（ホストバイトオーダー）がリトルエンディアンの場合は、変換が必要となります。NORTi TCP/IP では、プロトコルスタック内部でこの変換を行っているため、アプリケーションでは、バイトオーダーを気にする必要はありません。アプリケーションで、TCP データ部や UDP データ部のバイトオーダー変換を行う場合には、以下のユーティリティ・マクロが利用できます。

htonl	ホスト→ネットワークバイトオーダ変換（long）
htons	ホスト→ネットワークバイトオーダ変換（short）
ntohl	ネットワーク→ホストバイトオーダ変換（long）
ntohs	ネットワーク→ホストバイトオーダ変換（short）

4.2 エラーコード取り出し

NORTi TCP/IP のサービスコールは、原則としてエラーコードを戻り値として返します。エラーコードの下位バイトには、ITRON 仕様で共通の「メインエラーコード」が入ります。エラーコードの上位バイトには、TCP/IP 特有の「サブエラーコード」が入ります。

サービスコール戻り値のエラーコード、メインエラーコード、サブエラーコードのいずれも負の値です。サービスコール戻り値のエラーコードから、メインエラーコードだけ、サブエラーコードだけを取り出すために、次の 2 つのユーティリティ・マクロが用意されています。

mainercd	メインエラーコード取り出し
subercd	サブエラーコード取り出し

※NORTi TCP/IP では、サブエラーコードを定義していていません。

4.3 構造体

```
typedef struct {
        UW ipaddr;          IP アドレス
        UH portno;          ポート番号
} T_IPV4EP;

typedef struct {
        ATR repatr;         TCP受付口属性(未使用, 0)
        T_IPV4EP myaddr;    自分側のIPアドレスとポート番号
} T_TCP_CREP;

typedef struct {
        ATR cepatr;         TCP通信端点属性(未使用, 0)
        VP sbuf;            送信バッファ領域の先頭アドレス
        INT sbufsz;         送信バッファ領域のサイズ
        VP rbuf;            受信バッファ領域の先頭アドレス
        INT rbufsz;         受信バッファ領域のサイズ
        FP callback;        コールバックルーチンのアドレス
} T_TCP_CCEP;

typedef struct t_udp_ccep {
        ATR cepatr;         UDP通信端点属性(未使用, 0)
        T_IPV4EP myaddr;    自分側のIPアドレスとポート番号
        FP callback;        コールバックルーチン
} T_UDP_CCEP;
```

4.4 メインエラーコード

E_OK	0	0		正常終了
E_SYS	0xf..ffb	-5		システムエラー
E_NOSPT	0xf..ff7	-9	(-17)	未サポート機能
E_PAR	0xf..fef	-17	(-33)	パラメータエラー
E_ID	0xf..fee	-18	(-35)	不正 ID 番号
E_NOMEM	0xf..fdf	-33	(-10)	メモリ不足
E_NOEXS	0xf..fd6	-42	(-52)	オブジェクト未生成
E_OBJ	0xf..fd7	-41	(-63)	オブジェクト状態エラー
E_QOVR	0xf..fd5	-43	(-73)	キューイングオーバーフロー
E_RLWAI	0xf..fcf	-49	(-86)	処理のキャンセル、待ち状態の強制解除
E_TMOUT	0xf..fce	-50	(-85)	ポーリング失敗またはタイムアウト
E_DLT	0xf..fcd	-51	(-81)	待ちオブジェクトの削除
E_WBLK	0xf..fad	-83		ノンブロッキングコール受付け
E_CLS	0xf..fa9	-87		コネクションの切断
E_BOVR	0xf..fa7	-89		バッファオーバーフロー
E_LNK	0xf..f42	-190		ネットワークインターフェース未初期化
E_ADDR	0xf..f41	-191		IP アドレスが変更された

()内はμITRON3.0 仕様の値

第5章 ユーティリティ・マクロ

5.1 ユーティリティ・マクロ

htonl

機能 ホスト→ネットワークバイトオーダ変換(long)

形式 UW htonl(UW hl);

hl **ホストバイトオーダーのデータ**

戻値 **ネットワークバイトオーダーに変換したデータ**

解説 ホストバイトオーダのロングワード(32 ビット)データを、ネットワークバイトオーダに変換します。

htons

機能 ホスト→ネットワークバイトオーダ変換(short)

形式 UH htons(UH hs);

hs **ホストバイトオーダーのデータ**

戻値 **ネットワークバイトオーダーに変換したデータ**

解説 ホストバイトオーダのショートワード(16 ビット)データを、ネットワークバイトオーダに変換します。

ntohl

機能 ネットワーク→ホストバイトオーダ変換(long)

形式 UW ntohl(UW nl);

nl **ネットワークバイトオーダーのデータ**

戻値 **ホストバイトオーダーに変換したデータ**

解説 ネットワークバイトオーダのロングワード(32 ビット)データを、ホストバイトオーダに変換します。

ntohs

機能 ネットワーク→ホストバイトオーダ変換(short)

形式 UH ntohs(UH ns);

ns **ネットワークバイトオーダーのデータ**

戻値 **ホストバイトオーダーに変換したデータ**

解説 ネットワークバイトオーダのショートワード(16 ビット)データを、ホストバイトオーダに変換します。

5.2 ユーティリティ関数

byte4_to_long

機能　4バイト配列IPアドレス→long変換

形式　UW byte4_to_long(const UB *b);

const UB *b　変換する4byte配列

戻値　変換された値

解説　4バイトIPアドレス配列をlong型IPアドレスに変換します。

long_to_byte4

機能　long→4バイト配列IPアドレス変換

形式　void long_to_byte4(UB *b, UW d);

UB *b　4byteの配列を格納するポインタ

UW d　変換するlongの値

解説　long型IPアドレスを4バイト配列IPアドレスに変換します。

ascii_to_ipaddr

機能　IP文字列→long型IPアドレス変換

形式　UW ascii_to_ipaddr(const char *s);

const char *s 変換するIPアドレスの文字列（例）“192.168.100.99 ”

戻値　変換された値

解説　IP文字列をlong型IPアドレスに変換します。

ipaddr_to_ascii

機能　long型IPアドレス→IP文字列変換

形式　INT ipaddr_to_ascii(char *s, UW ipaddr);

char *s　変換するIPアドレスの文字列を格納するポインタ

UW ipaddr　変換するIPアドレス

戻値　変換された文字列のサイズ

解説　long型IPアドレスをIP文字列に変換します。

第 6 章 TCP サービスコール

TCP サービスコール一覧

tcp_cre_rep	TCP 受付口の生成
tcp_vcre_rep	TCP 受付口の生成(ID 自動割り当て)
tcp_del_rep	TCP 受付口の削除
tcp_cre_cep	TCP 通信端点の生成
tcp_vcre_cep	TCP 通信端点の生成(ID 自動割り当て)
tcp_del_cep	TCP 通信端点の削除
tcp_acp_cep	接続要求待ち(受動オープン)
tcp_con_cep	接続要求(能動オープン)
tcp_sht_cep	データ送信の終了
tcp_cls_cep	通信端点のクローズ
tcp_snd_dat	データの送信
tcp_rcv_dat	データの受信
tcp_get_buf	送信用バッファの取得(省コピーAPI)
tcp_snd_buf	バッファ内のデータの送信(省コピーAPI)
tcp_rcv_buf	受信したデータの入ったバッファの取得(省コピーAPI)
tcp_rel_buf	受信用バッファの解放(省コピーAPI)
tcp_snd_oob	緊急データの送信
tcp_rcv_oob	緊急データの受信
tcp_can_cep	送受信のキャンセル
tcp_set_opt	TCP 通信端点オプションの設定
tcp_get_opt	TCP 通信端点オプションの参照

tcp_cre_rep

機能 TCP 受付口の生成

形式
```
ER tcp_cre_rep(ID repid, const T_TCP_CREP *pk_crep);
repid      TCP 受付口 ID
pk_crep    TCP 受付口生成情報

typedef struct {
    ATR repatr;        TCP 受付口属性(未使用, 0)
    T_IPEP myaddr;     自分側の IP アドレスとポート番号
} T_TCP_CREP;

typedef struct {
    UW ipaddr;         IP アドレス
    UH portno;         ポート番号
} T_IPEP;
```

戻値
```
E_OK       正常終了
E_ID       不正 ID 番号
E_OBJ      TCP 受付口が生成済み、ポート番号既使用
E_PAR      無効な IP アドレスまたはポート番号を指定した
```

解説 repid で指定された TCP 受付口を生成します。

生成された TCP 受付口は、pk_crep->myaddr.ipaddr で指定された IP アドレスと、pk_crep->myaddr.portno で指定されたポート番号とを宛先とする接続要求のみを待ち受けます。

マルチホームで受動接続を行う場合、追加したい IP アドレスを指定して受付口を生成してください。その場合、指定する IP アドレスのネットワークアドレスはネットワークインターフェースのネットワークアドレスと合わせてください。

補足 自分側の IP アドレスに IPV4_ADDRANY(= 0)を指定すると、全てのネットワークインターフェースに設定されている IP アドレスで待ち受けることが出来ます。

tcp_vcre_rep

機能 TCP 受付口の生成（ID 自動割り当て）

形式 ER tcp_vcre_rep(const T_TCP_CREP *pk_crep);

pk_crep TCP 受付口生成情報

戻値 正の値ならば、割り当てられた受付口 ID

E_ID 受付口 ID が不足

E_OBJ ポート番号既使用

E_PAR 無効な IP アドレスまたはポート番号を指定した

解説 未生成の受付口 ID を検索して割り当てます。受付口 ID が割り当てられない場合は、E_ID エラーを返します。それ以外は tcp_cre_rep と同じです。

補足 NORTi TCP/IP 独自のシステムコールです。

tcp_del_rep

機能 TCP 受付口の削除

形式 ER tcp_del_rep(ID repid);

repid TCP 受付口 ID

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 受付口が未生成

解説 repid で指定された TCP 受付口を削除します。この TCP 受付口にキューイングされた接続待ち処理はキャンセルされるため、tcp_acp_cep サービスコールを発行したタスクへは、E_DLT エラーが返ります。tcp_acp_cep がノンブロッキング指定で呼び出されている場合は、E_DLT エラーをコールバックでアプリケーションへ通知します。

tcp_cre_cep

機能 TCP 通信端点の生成

形式

```
ER tcp_cre_cep(ID cepid, const T_TCP_CCEP *pk_ccep);
```

cepid TCP 通信端点 ID

pk_ccep TCP 通信端点生成情報パケットへのポインタ

```
typedef struct {
    ATR cepatr;         TCP 通信端点属性(未使用, 0)
    VP sbuf;            送信バッファ領域の先頭アドレス
    INT sbufsz;         送信バッファ領域のサイズ
    VP rbuf;            受信バッファ領域の先頭アドレス
    INT rbufsz;         受信バッファ領域のサイズ
    FP callback;        コールバックルーチンのアドレス
} T_TCP_CCEP;
```

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_OBJ	TCP 通信端点が生成済み
E_PAR	送受信バッファ、サイズの指定が正しくない

解説 cepid で指定された TCP 通信端点を生成します。TCP 通信端点を生成しただけでは、何も機能しません。この後、tcp_acp_cep サービスコールによる受動オープン、または、tcp_con_cep サービスコールによる能動オープンを行って初めて機能します。

与えられたバッファ領域は、プロトコルスタック内部で管理されています。省コピーAPI の tcp_get_buf や tcp_rcv_buf サービスコールでは、この領域のいずれかの場所へのポインタを返します。このポインタを使わずに、バッファ領域を直接アクセスすることは避けてください。また、バッファ領域は通信端点毎に独立した領域を使用してください。

送信要求の最大キューイング数と受信パケットの最大キューイング数は、コンフィグレーションで決まります。キューイングできる個数を超える要求や受信は、破棄されます。

補足 バッファ領域先頭アドレスとして NADR を指定した時、バッファをプロトコルスタック内部で確保する機能はサポートしていません。

送受信バッファ領域のサイズは 2byte 以上を指定する必要があります。より効率的な通信

を行う為には可能な限り大きなバッファを使用してください。送受信バッファは 64KB 未満に設定してください。

送受信バッファのサイズが小さすぎると通信の効率が悪くなります。大きなデータを連続して送受信する場合、最適な通信を行うためにはバッファサイズは 2048,8192 程度を下限に設定すると有効です。

1 回のコネクションで連続して総データサイズが 4Gbyte を超えるような通信を行う場合、送受信バッファ領域のサイズは制約上 2 のべき乗サイズを指定してください。その場合の最大サイズは 32KB になります。

tcp_vcre_cep

機能 TCP 通信端点の生成(ID 自動割り当て)

形式 ER tcp_vcre_cep(const T_TCP_CCEP *pk_ccep);

pk_ccep TCP 通信端点生成情報パケットへのポインタ

戻値 正の値ならば、割り当てられた通信端点 ID

E_ID TCP 通信端点 ID が不足

E_PAR 送受信バッファ、サイズの指定が正しくない

解説 未生成の TCP 通信端点 ID を検索して割り当てます。TCP 通信端点 ID が割り当てられない場合は、E_ID エラーを返します。それ以外は tcp_cre_cep と同じです。

補足 NORTi TCP/IP 独自のシステムコールです。

tcp_del_cep

機能 TCP 通信端点の削除

形式 ER tcp_del_cep(ID cepid);

cepid	TCP 通信端点 ID

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が使用中

解説 cepid で指定された TCP 通信端点を削除します。TCP 通信端点が使用中、すなわち、オープン待ち～クローズ中の場合は削除できません。

tcp_acp_cep

機能 接続要求待ち(受動オープン)

形式

```
ER tcp_acp_cep(ID cepid, ID repid, T_IPV4EP *p_dstaddr, TMO tmout);
```

cepid TCP 通信端点 ID

repid TCP 受付口

p_dstaddr 相手側 IP アドレス/ポート番号格納先へのポインタ

tmout タイムアウト指定

```
typedef struct {
    UW ipaddr;      IP アドレス
    UH portno;      ポート番号
} T_IPV4EP;
```

戻値

E_OK 正常終了

E_ID 不正 ID 番号

E_NOEXS TCP 通信端点が未生成

E_OBJ TCP 通信端点が使用中

E_DLT 接続要求を待つ間に TCP 受付口が削除された

E_WBLK ノンブロッキングコール受付け

E_TMOUT ポーリング失敗またはタイムアウト

E_RLWAI 処理のキャンセル、待ち状態の強制解除

E_CLS RST を受信した

E_PAR p_dstaddr が NULL

E_LNK プロトコルスタック未初期化

解説 本サービスコールによって、cepid で指定された TCP 通信端点は受動オープン待ち状態となります。すなわち、repid で指定された TCP 受付口へ受信する接続要求を待ち受けます。SYN を受信したならば、TCP 通信端点は ACK を送信して接続が完了(コネクションが確立)し、接続状態となります。

*p_dstaddr には、相手側の IP アドレスとポート番号が返ります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、接続が完了するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した

時間が経過しても接続要求がない、または、接続が完了しなければ、E_TMOUT エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で、本サービスコールを発行した場合、接続の完了は、コールバックで通知されます。

同じ TCP 受付口に対して、同時に複数の tcp_acp_cep サービスコールを発行することができます。その場合、先に発行された tcp_acp_cep の TCP 通信端点で接続を受け付けます。

タイムアウトや tcp_can_cep によって、本サービスコールの処理がキャンセルされた場合、TCP 通信端点は未使用状態に戻ります。

補足 ノンブロッキング(tmout = TMO_NBLK)で、本サービスコールを発行した場合、指定する相手側 IP アドレス/ポート番号格納先へのポインタ(p_dstaddr)は接続後に書込みが行われる為に、この領域にはスタック空間を使用しないでください。

tcp_con_cep

機能 接続要求(能動オープン)

形式 ER tcp_con_cep(ID cepid, T_IPV4EP *p_myaddr, T_IPV4EP *p_dstaddr, TMO tmout);

cepid TCP 通信端点 ID
p_myaddr 自分側 IP アドレス/ポート番号構造体へのポインタ
p_dstaddr 相手側 IP アドレス/ポート番号構造体へのポインタ
tmout タイムアウト指定

```
typedef struct {
    UW ipaddr;          IP アドレス
    UH portno;          ポート番号
} T_IPV4EP;
```

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が使用中、ポート番号既使用
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_CLS	接続要求が拒否された
E_PAR	p_myaddr または p_dstaddr の IP アドレスまたはポート番号が 0
E_LNK	プロトコルスタック未初期化

解説 本サービスコールによって、cepid で指定された TCP 通信端点は能動オープン待ち状態となります。すなわち、*p_dstaddr の IP アドレスとポート番号で指定された相手側へ、TCP 通信端点から SYN を送信して接続を要求します。ACK を受信すると接続が完了(コネクションが確立)し、接続状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、接続が完了するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても接続要求がない、または、接続が完了しなければ、E_TMOUT エラーが返ります。

ポーリング(tmout = TMO_POL)は無意味のため、E_TMOUT エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で、本サービスコールを発行した場合、接続の完了は、コールバックで通知されます。

タイムアウトや tcp_can_cep によって tcp_con_cep の処理がキャンセルされた場合や接続要求が拒否された場合は、TCP 通信端点は未使用状態に戻ります。

自分側の IP アドレスに IPV4_ADDRANY(= 0)を指定した場合、default_ipaddr の値がプロトコルスタックで設定されます。ポート番号に TCP_PORTANY(= 0)を指定した場合、プロトコルスタックで任意の値に設定します。

マルチホームで能動接続を行う場合、追加したい IP アドレスを自分側の IP アドレスに指定して本 API を呼び出してください。その場合、指定する IP アドレスのネットワークアドレスはネットワークインターフェースのネットワークアドレスと合わせてください。

補足 p_myaddr に NADR(= -1)を指定した場合、IP アドレス、ポート番号ともにプロトコルスタックで決定する機能もサポートしていません。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行し、相手側 IP アドレスからの応答がない場合は本サービスコールから戻りません。

E_CLS でリターンした場合、接続先から接続を拒否された(RST 受信)可能性があります。この場合、接続先のポートが準備されていないか、ポート番号が正しいかを確認してください。

tcp_sht_cep

機能 データ送信の終了

形式 ER tcp_sht_cep(ID cepid);

cepid TCP 通信端点 ID

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が接続状態でない

解説 cepid で指定された TCP 通信端点のデータ送信終了を要求します。TCP 通信端点は直ちに送信終了状態となり、送信バッファ中のデータを送信し終わったら、FIN を送ります。本サービスコールでは、TCP 通信端点の状態が変化するだけで、発行元のタスクが待ち状態となることはありません。

本サービスコールは、データ送信終了を相手側に知らせるためのものです。

FIN を受け取った相手側は、これ以上データが送られて来ないことを知って、最後の応答を返してから、コネクションを終了(クローズ)します。

tcp_cls_cep

機能 通信端点のクローズ

形式 ER tcp_cls_cep(ID cepid, TMO tmout);

cepid	TCP 通信端点 ID
tmout	タイムアウト指定

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_CLS	RST による切断

解説 cepid で指定された TCP 通信端点のコネクションを切断します。

TCP 通信端点がまだ送信終了していない場合は、送信バッファ中のデータを送信し終わるのを待って FIN を送ります。そして、送信した FIN に対する ACK 受信を待ちます。一方、相手がまだ送信終了していない場合は、受信したデータを捨てながら FIN の受信を待ち、FIN を受信したら ACK を送信します。

以上の切断手順で、本サービスコールの処理は完了し、TCP 通信端点は未使用状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、切断が完了するまで待ち状態となります。本サービスコールからリターンした後は、TCP 通信端点は未使用状態になっていますので、すぐに再利用することができます。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、切断が完了しなければ、E_TMOUT エラーが返ります。この際、および、tcp_can_cep によって処理がキャンセルされた場合、TCP 通信端点から RST を送信して接続を強制切断します。

ポーリング(tmout = TMO_POL)は無意味のため、E_TMOUT エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で、本サービスコールを発行した場合、切断の完了

は、コールバックで通知されます。

補足 TMO_NBLK を指定したときに、相手側がクラッシュした場合などはコールバックが呼ばれないことがありますので、タイムアウト時間は可能な限り指定してください。E_TMOUT エラーになった場合でも RST の発行を行った後に確実に切断されます。

tcp_snd_dat

機能 データの送信

形式 ER tcp_snd_dat(ID cepid, const VP data, INT len, TMO tmout);

cepid	TCP通信端点ID
data	送信データへのポインタ
len	送信したいデータの長さ
tmout	タイムアウト指定

戻値

正の値	正常終了(送信バッファに入れたデータの長さ)
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続または送信終了、送信がペンディング中
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除 または、指定した IP アドレスから ARP 応答が無い
E_CLS	TCP 接続が切断された
E_PAR	len の長さが不正

解説 cepidで指定されたTCP通信端点からデータを送信します。dataが指している長さlenのデータを送信バッファへコピーするだけで、このサービスコールはリターンします。送信バッファの空きが、送信しようとしたデータ長よりも短い場合、送信バッファが一杯になるまで送信バッファにデータを入れ、送信バッファに入れたデータの長さを返します。最初からまったく送信バッファに空きがない場合には、空きが生じるまで、発行元のタスクは待ち状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、送信バッファへのコピーが完了するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、送信バッファに空きが生じなければ、E_TMOUTエラーが返ります。

ポーリング(tmout = TMO_POL)で本サービスコールを発行した場合、送信バッファに空きが生じなければ、直ぐにE_TMOUTエラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、送信バッファ

に空きがなくても、サービスコールから直ぐにリターンし、送信バッファへのコピー完了は、コールバックで通知されます。

送信処理はキューイングできません。同一のTCP通信端点に対するtcp_snd_datまたはtcp_get_bufの二重発行はE_OBJエラーとなります。

補足 パラメータで指定した、送信したいデータの長さ(len)と戻り値はかならずしも、同じ値になるとは限りません。送信したいデータの長さ(len)よりも送信バッファの空きが少ない場合は、バッファの空きサイズ分がコピーされます。

tcp_rcv_dat

機能 データの受信

形式 ER tcp_rcv_dat(ID cepid, VP data, INT len, TMO tmout);

cepid	TCP 通信端点 ID
data	受信データを入れる領域へのポインタ
len	受信したいデータの長さ
tmout	タイムアウト指定

戻値

正の値	正常終了(取り出したデータの長さ)
0	データ終結(接続が正常切断された)
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続、受信がペンディング中
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_CLS	TCP 接続が切断され、受信バッファが空
E_PAR	len の長さが不正

解説 cepidで指定されたTCP通信端点からデータを受信します。受信バッファに入ったデータを、dataで指し示される領域へコピーした時点で、このサービスコールからリターンします。受信バッファに入っているデータ長が受信しようとしたデータ長lenよりも短い場合、受信バッファが空になるまでデータを取り出し、取り出したデータの長さを戻り値として返します。

受信バッファが空の場合には、データを受信するまで、このサービスコール発行元のタスクは待ち状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、データのコピーが完了するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、データを受信しなければ、E_TMOUTエラーが返ります。

ポーリング(tmout = TMO_POL)で本サービスコールを発行した場合、受信バッファにデータ

がなければ、直ぐにE_TMOUTエラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、受信データがなくても、サービスコールから直ぐにリターンし、データ受信の完了は、コールバックで通知されます。

受信処理はキューイングできません。同一のTCP通信端点に対するtcp_rcv_datまたはtcp_rcv_bufの二重発行はE_OBJエラーとなります。

相手側から接続が正常切断され、受信バッファにデータがなくなると、本サービスコールから 0 が返ります。

tcp_get_buf

機能 送信用バッファの取得(省コピーAPI)

形式 ER tcp_get_buf(ID cepid, VP *p_buf, TMO tmout);

cepid	TCP 通信端点 ID
p_buf	空き領域先頭アドレス格納先へのポインタ
tmout	タイムアウト指定

戻値

正の値	正常終了(空き領域の長さ)
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続または送信終了、送信がペンディング中
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_CLS	TCP 接続が切断された

解説 cepid で指定された TCP 通信端点の送信バッファ中の、次に送信すべきデータを入れることができる空き領域の先頭アドレスを*p_buf へ、連続した空き領域の長さを戻り値に返します。送信バッファに空きがない場合には、本サービスコール発行元のタスクは、空きが生じるまで待ち状態となります。

タイムアウトなし(tmout =TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、空き領域を獲得できるまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、空き領域を獲得できなければ、E_TMOUT エラーが返ります。

ポーリング(tmout = TMO_POL)で本サービスコールを発行した場合、空き領域がなければ、直ぐに E_TMOUT エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、空き領域がなくても、サービスコールから直ぐにリターンし、空き領域を獲得の完了は、コールバックで通知されます。

本サービスコールを発行したことで、プロトコルスタックの内部状態は変化しません。そのため、tcp_get_buf を続けて呼び出すと同じ領域が返されます。逆に、tcp_snd_dat や

tcp_snd_buf を発行するとプロトコルスタックの内部状態は変化します。これらの発行で、それ以前に tcp_get_buf が返した情報は無効となります。

送信処理はキューイングできません。同一の TCP 通信端点に対する tcp_snd_dat または tcp_get_buf との二重発行は E_OBJ エラーとなります。

tcp_snd_buf

機能 バッファ内のデータの送信(省コピーAPI)

形式 ER tcp_snd_buf(ID cepid, INT len);

cepid	TCP 通信端点 ID
len	データの長さ

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続または送信終了、len が長すぎる
E_CLS	TCP 接続が切断された
E_PAR	len の長さが不正

解説 cepid で指定された TCP 通信端点から、tcp_get_buf で取り出したバッファに書き込んだ長さ len のデータを送信します。tcp_snd_buf は、送信を要求するだけのため、本サービスコール発行元のタスクが、待ち状態になることはありません。

tcp_rcv_buf

機能 受信したデータの入ったバッファの取得(省コピーAPI)

形式 ER tcp_rcv_buf(ID cepid, VP *p_buf, TMO tmout);

cepid	TCP 通信端点 ID
p_buf	受信データ先頭アドレス格納先へのポインタ
tmout	タイムアウト指定

戻値

正の値	正常終了(受信データの長さ)
0	データ終結(接続が正常切断された)
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	TCP 通信端点が未接続、受信がペンディング中
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_CLS	TCP 接続が切断され、受信バッファが空
E_WBLK	ノンブロッキングコールの受付け

解説 cepid で指定された TCP 通信端点の受信したデータが入っているバッファの先頭アドレスを *p_buf へ、そこから連続して入っているデータの長さを戻り値として返します。受信バッファが空の場合、本サービスコール発行元のタスクは、データを受信するまで待ち状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、データを受信するまで待ち状態となります。

タイムアウトあり(tmout = 1 ～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、データを受信できなければ、E_TMOUT エラーが返ります。

ポーリング(tmout = TMO_POL)で本サービスコールを発行した場合、受信データがなければ、直ぐに E_TMOUT エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、受信データがなくても、サービスコールから直ぐにリターンし、データ受信は、コールバックで通知されます。

本サービスコールを発行したことにより、プロトコルスタックの内部状態は変化しません。

そのため、tcp_rcv_buf を続けて呼び出すと同じ領域が返されます。逆に、tcp_rcv_dat、tcp_rel_buf を発行すると、プロトコルスタックの内部状態は変化します。これらを発行すると、それ以前に tcp_rcv_buf が返した情報は無効となります。

相手側から接続が正常切断され、受信バッファにデータがなくなると、本サービスコールから 0 が返ります。

異常切断した場合でも、受信バッファ中にデータがある間は、受信データの先頭アドレスと長さを取り出すことができます。

受信処理はキューイングできません。同一の TCP 通信端点に対する tcp_rcv_dat または tcp_rcv_buf との二重発行は E_OBJ エラーとなります。

tcp_rel_buf

機能 受信用バッファの解放(省コピーAPI)

形式 ER tcp_rel_buf(ID cepid, INT len);

cepid	TCP 通信端点 ID
len	データの長さ

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_OBJ	指定した TCP 通信端点が未接続、len が長すぎる
E_PAR	パラメータエラー、len が不正

解説 cepid で指定された TCP 通信端点の tcp_rcv_buf で取り出したバッファ中の長さ len のデータを捨てます。このサービスコールで待ち状態になることはありません。取り出したバッファはこの関数を使って必ず解放してください。

tcp_snd_oob

機能 緊急データの送信

形式 ER tcp_snd_oob(ID cepid, VP data, INT len, TMO tmout);

解説 現バージョンでは、サポートされていません。

tcp_rcv_oob

機能 緊急データの受信

形式 ER tcp_rcv_oob(ID cepid, VP data, INT len);

解説 現バージョンでは、サポートされていません。

tcp_can_cep

機能 送受信のキャンセル

形式 ER tcp_can_cep(ID cepid, FN fncd);

cepid	TCP 通信端点 ID
fncd	キャンセルするサービスコールの機能コード

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	TCP 通信端点が未生成
E_PAR	パラメータエラー(fncd が不正)
E_OBJ	fncd で指定した処理がペンディングしていない

解説 cepid で指定された TCP 通信端点にペンディングしている処理の実行をキャンセルします。キャンセルされた処理のサービスコールを発行していたタスクには、E_RLWAI エラーが返ります。あるいは、ノンブロッキングコールをキャンセルした場合には、それを通知するコールバックルーチンが呼ばれます。

キャンセルする処理は、次の機能コードで指定してください。TFN_TCP_ALL (= 0)を指定すると、すべての処理をキャンセルすることができます。

TFN_TCP_ACP_CEP	tcp_acp_cep の処理をキャンセル
TFN_TCP_CON_CEP	tcp_con_cep の処理をキャンセル
TFN_TCP_CLS_CEP	tcp_cls_cep の処理をキャンセル
TFN_TCP_SND_DAT	tcp_snd_dat の処理をキャンセル
TFN_TCP_RCV_DAT	tcp_rcv_dat の処理をキャンセル
TFN_TCP_GET_BUF	tcp_get_buf の処理をキャンセル
TFN_TCP_RCV_BUF	tcp_rcv_buf の処理をキャンセル
TFN_TCP_SND_OOB	tcp_snd_oob の処理をキャンセル
TFN_TCP_ALL	すべての処理をキャンセル

tcp_set_opt

機能 TCP 通信端点オプションの設定

形式 ER tcp_set_opt(ID cepid, INT optname, const VP optval, INT optlen);

cepid TCP 通信端点 ID

optname オプションの種類

optval オプション値が設定されているバッファのポインタ

optlen オプション値の長さ

optname には以下が設定できます

SET_IP_TTL 通信端点で使用する TTL の値を設定します

SET_IP_TOS 通信端点で使用する TOS の値を設定します

SET_TCP_MTU 通信端点で使用する MTU サイズを設定します

各オプションで指定するタイプは次のようになります

SET_IP_TTL UB 型

SET_IP_TOS UB 型

SET_TCP_MTU UW 型

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	通信端点が未生成
E_PAR	パラメータエラー
E_OBJ	通信端点が使用中

解説 TCP 通信端点のオプションを設定します。

TCP の通信端点の生成後で、接続前にのみ変更ができます。この API はプロトコルスタック Ver4.08.11 以降でサポートされました。

tcp_get_opt

機能 TCP 通信端点オプションの参照

形式 ER tcp_get_opt(ID cepid, INT optname, VP optval, INT optlen);

cepid TCP 通信端点 ID

optname オプションの種類

optval オプション値を取得するバッファのポインタ

optlen オプション取得するバッファのサイズ

optname には以下が設定できます

SET_IP_TTL 通信端点で設定されている TTL の値を取得します

SET_IP_TOS 通信端点で設定されている TOS の値を取得します

SET_TCP_MTU 通信端点で設定されている MTU サイズを取得します

各オプションで指定するタイプは次のようになります

SET_IP_TTL UB 型

SET_IP_TOS UB 型

SET_TCP_MTU UW 型

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	通信端点が未生成
E_PAR	パラメータエラー
E_OBJ	通信端点が使用中

解説 設定されている TCP 通信端点のオプション値を取得します。この API はプロトコルスタック Ver4.08.11 以降でサポートされました。

第 7 章 UDP サービスコール

UDP サービスコール一覧

udp_cre_cep	UDP 通信端点の生成
udp_vcre_cep	UDP 通信端点の生成(ID 自動割り当て)
udp_del_cep	UDP 通信端点の削除
udp_snd_dat	パケットの送信
udp_rcv_dat	パケットの受信
udp_can_cep	送受信のキャンセル
udp_set_opt	UDP 通信端点オプションの設定
udp_get_opt	UDP 通信端点オプションの参照

udp_cre_cep

機能 UDP 通信端点の生成

形式 ER udp_cre_cep(ID cepid, const T_UDP_CCEP *pk_ccep);

cepid UDP 通信端点 ID

pk_ccep UDP 通信端点生成情報パケットへのポインタ

```
typedef struct t_udp_ccep {
    ATR cepatr;          UDP 通信端点属性(未使用, 0)
    T_IPEP myaddr;       自分側の IP アドレスとポート番号
    FP callback;         コールバックルーチン
} T_UDP_CCEP;

typedef struct {
    UW ipaddr;           IP アドレス
    UH portno;           ポート番号
} T_IPEP;
```

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_OBJ	UDP 通信端点が生成済み、ポート番号既使用
E_PAR	pk_ccep が不正

解説 cepid で指定された UDP 通信端点を生成します。UDP 通信端点は生成するだけで受信可能となります。udp_rcv_dat サービスコールが発行される前に UDP パケットを受信した場合は、コールバックで通知されます。

送信パケットの最大キューイング数と受信要求の最大キューイング数は、コンフィグレーションで決まります。キューイングできる個数を超える送信や要求は、破棄されます。

自分側の IP アドレスに IPV4_ADDRANY(= 0)を指定した場合、default_ipaddr の値がプロトコルスタックで設定されます。ポート番号に UDP_PORTANY (= 0) を指定した場合、プロトコルスタックで任意の値に設定します。

マルチホームで利用する場合、追加したい IP アドレスを指定して端点を生成してください。その場合、指定する IP アドレスのネットワークアドレスはネットワークインターフェースのネットワークアドレスと合わせてください。

udp_vcre_cep

機能 UDP 通信端点の生成(ID 自動割り当て)

形式 ER udp_vcre_cep(const T_UDP_CCEP *pk_ccep);

pk_ccep UDP 通信端点生成情報パケットへのポインタ

戻値 正の値ならば、割り当てられた通信端点 ID

E_ID UDP 通信端点 ID が不足

E_OBJ ポート番号既使用

E_PAR pk_ccep が不正

解説 未生成の UDP 通信端点 ID を検索して割り当てます。UCP 通信端点 ID が割り当てられない場合は、E_ID エラーを返します。それ以外は udp_cre_cep と同じです。

補足 NORTi TCP/IP 独自のシステムコールです。

udp_del_cep

機能 UDP 通信端点の削除

形式 ER udp_del_cep(ID cepid);

cepid	UDP 通信端点 ID

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成

解説 cepid で指定された UDP 通信端点を削除します。削除された UDP 通信端点に対して、UDP 送受信のサービスコールを発行して待ち状態になっているタスクがあれば、待ちを解除して E_DLT エラーを返します。キューイングされている送受信パケットは破棄されます。

udp_snd_dat

機能 パケットの送信

形式 ER udp_snd_dat(ID cepid, const T_IPV4EP *p_dstaddr, const VP data, INT len, TMO tmout);

cepid	UDP 通信端点 ID
p_dstaddr	相手側 IP アドレス/ポート番号構造体へのポインタ
data	送信パケットへのポインタ
len	送信パケットの長さ
tmout	タイムアウト指定

```
typedef struct {
    UW ipaddr;          IP アドレス
    UH portno;          ポート番号
} T_IPV4EP;
```

戻値

正の値	正常終了(送信バッファに入れたデータの長さ)
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成
E_QOVR	キューイングオーバーフロー
E_DLT	送信完了を待つ間に UDP 通信端点が削除された
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除、送信先が無応答
E_PAR	パラメータが不正、または送信パケットへのポインタが奇数番地に設定されている
E_LNK	プロトコルスタック未初期化

解説 cepidで指定されたUDP通信端点から、dataで指し示される長さlenのパケットを、*p_dstaddrで指定された相手へ送信します。NORTi TCP/IPでは、プロトコルスタック内部に、UDPのための送信バッファを持っていません。パケットがデバイスのバッファメモリへコピーされるまで待ち状態となります。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタスクは、バッファメモリへのコピーが完了するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した

時間が経過しても、バッファメモリが空かなければ、E_TMOUTエラーが返ります。

ポーリング(tmout = TMO_POL)で本サービスコールを発行した場合、バッファメモリに空きがなければ、ただちにE_TMOUTエラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、バッファメモリに空きがなくても、サービスコールから直ぐにリターンし、バッファメモリへのコピー完了は、コールバックで通知されます。

送信パケットはキューイング可能です。すなわち、同じUDP通信端点に対して、同時に複数の本サービスコールを発行することができます。ただし、バッファメモリへのコピーが完了するまでは、dataで指し示される領域は使用できないことに注意してください。キューイングできる送信パケット数の上限は、UDP_QCNTマクロで指定した値で決まります。上限を越えてudp_snd_datを発行するとE_QOVRエラーとなります。

ここで言う送信パケットとは、UDP パケットのデータ部です。UDP ヘッダは、プロトコルスタック内部で追加されます。

補足 送信パケットのポインタは内部管理上必ず偶数番地になっている必要があります。

udp_rcv_dat

機能 パケットの受信

形式 ER udp_rcv_dat(ID cepid, T_IPV4EP *p_dstaddr, VP data, INT len,TMO tmout);

cepid UDP 通信端点 ID
p_dstaddr 相手側 IP アドレス/ポート番号構造体へのポインタ
data 受信パケットを入れる領域へのポインタ
len 受信パケットを入れる領域の長さ
tmout タイムアウト指定

```
typedef struct {
    UW ipaddr;          IP アドレス
    UH portno;          ポート番号
} T_IPV4EP;
```

戻値

正の値	正常終了(取り出したデータの長さ)
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成
E_DLT	受信を待つ間に UDP 通信端点が削除された
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル、待ち状態の強制解除
E_PAR	受信パケットを入れる領域の長さが小さすぎる、または受信パケットへのポインタが奇数番地に設定されている
E_OBJ	受信データでチェックサムエラー
E_LNK	プロトコルスタック未初期化

解説 cepidで指定されたUDP通信端点から、dataで指し示される領域へパケットを受信します。受信したパケットの長さを戻り値として返します。

*p_dstaddr には、相手側の IP アドレスとポート番号が返ります。

受信パケットを入れる領域の長さlenが、受信したパケットの長さよりも短い場合には、領域いっぱいまでデータを取り出し、入りきらないデータを捨て、E_BOVRを返します。

タイムアウトなし(tmout = TMO_FEVR)で本サービスコールを発行した場合、発行元のタス

クは、パケットを受信するまで待ち状態となります。

タイムアウトあり(tmout = 1～0x7fffffff)で本サービスコールを発行した場合、指定した時間が経過しても、パケットを受信できなければ、E_TMOUTエラーが返ります。

コールバック関数以外からポーリング(tmout = TMO_POL)で本サービスコールを発行できません。この場合 E_PAR エラーが返ります。

ノンブロッキング(tmout = TMO_NBLK)で本サービスコールを発行した場合は、受信パケットがなくても、サービスコールから直ぐにリターンし、パケット受信完了は、コールバックで通知されます。

本サービスコールによる受信要求はキューイングできます。すなわち、同じ UDP 通信端点に対して、同時に複数の udp_rcv_dat を発行することができます。受信要求のキューイング数に制限はありません。

ここで言う受信パケットとは、UDP パケットのデータ部です。UDP ヘッダは、プロトコルスタック内部で除去されます。

補足 受信パケットを入れるバッファはプロトコルスタック内部で一部使用するため、必ず24byte以上必要です。また、同一端点を使って複数のタスクから受信を行う場合はキューイングされた順に受信されます。この場合受信バッファに同じ領域を指定しないでください。

受信パケットのポインタは内部管理上、必ず偶数番地になっている必要があります。

ノンブロッキング(tmout = TMO_NBLK)で、本サービスコールを発行した場合、指定する相手側 IP アドレス/ポート番号格納先へのポインタ(p_dstaddr)は受信後に書込みが行われる為に、この領域にはスタック空間を使用しないでください。

udp_can_cep

機能 送受信のキャンセル

形式 ER udp_can_cep(ID cepid, FN fncd);

cepid UDP 通信端点 ID

fncd キャンセルするサービスコールの機能コード

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成
E_PAR	パラメータエラー(fncd が不正)
E_OBJ	fncd で指定した処理がペンディングしていない

解説 cepid で指定された UDP 通信端点にペンディングしている処理の実行をキャンセルします。キャンセルされた処理のサービスコールを発行していたタスクには、E_RLWAI エラーが返ります。あるいは、ノンブロッキングコールをキャンセルした場合には、それを通知するコールバックルーチンが呼ばれます。

キャンセルする処理は、次の機能コードで指定してください。TFN_UDP_ALL (= 0)を指定すると、すべての処理をキャンセルすることができます。

TFN_UDP_SND_DAT	udp_snd_dat の処理をキャンセル
TFN_UDP_RCV_DAT	udp_rcv_dat の処理をキャンセル
TFN_UDP_ALL	すべての処理をキャンセル

udp_set_opt

機能 UDP 通信端点オプションの設定

形式 ER udp_set_opt(ID cepid, INT optname, const VP optval, INT optlen);

cepid UDP 通信端点 ID
optname オプションの種類
optval オプション値を入れたバッファへのポインタ
optlen オプション値の長さ

optname は以下が使用できます。

IP_BROADCAST
ブロードキャストパケットの送受信を許可する
型 BOOL optval TRUE:許可/FALSE:不許可

IP_ADD_MEMBERSHIP
マルチキャストグループへ参加する
型 UW optval マルチキャストアドレス

IP_DROP_MEMBERSHIP
マルチキャストグループから脱退する
型 UW optval マルチキャストアドレス

SET_IP_TTL
通信端点で使用する TTL の値を設定する
型 UB optval TTL 値

SET_IP_TOS
通信端点で使用する TOS の値を設定する
型 UB optval TOS 値

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成
E_PAR	無効なオプションの種類
E_OBJ	optval の値が不正
E_NOSPT	未サポート機能

解説 UDP 端点で使用するオプションを設定します。SET_IP_TTL と SET_IP_TOS はプロトコルスタック Ver4.08.11 以降でサポートされました。

例　ブロードキャスト IP パケット送受信について

ブロードキャストパケットの送受信を行うためには、upd_set_opt 関数を使用してブロードキャストの送受信を有効にします。送信は IP アドレスを 255.255.255.255（リミテッドブロードキャスト）に送信します。

```
T_UDP_CCEP udp_cep = {0, {IPV4_ADDRANY, UDP_PORTANY}, NULL};
BOOL optval;
/* UDP 端点生成 */
udp_cre_cep(cepid, &udp_cep);

/* ブロードキャストの送受信を有効に */
udp_set_opt(cepid, IP_BROADCAST, (VP)TRUE, sizeof(BOOL));

/* ブロードキャストパケットの送信 */
dstaddr.ipaddr = 0xFFFFFFFF;
dstaddr.portno = UDP_PORT_NO;
udp_snd_dat(cepid, &dstaddr, &pkt, sizeof(pkt), TMO_FEVR);
```

マルチキャスト IP パケットの送受信について

マルチキャストはクラス D IP アドレスの送信、受信と IGMP をサポートします。送信を行うには単純に送信先にマルチキャストアドレスを指定するだけです。受信を行うにはマルチキャストアドレスのグループへの参加が必要です。

```
T_UDP_CCEP udp_cep = {0, {IPV4_ADDRANY, UDP_PORTANY}, NULL};
UB ipaddr[] = { 225, 6, 7, 8 }; /* マルチキャストアドレス */
T_IPV4EP addr;

/* UDP 端点生成 */
udp_cre_cep(cepid, &udp_cep);

/* マルチキャストパケットの送信 */
dstaddr.ipaddr = byte4_to_long(ipaddr);
dstaddr.portno = UDP_PORT_NO;
udp_snd_dat(cepid, &dstaddr, &pkt, sizeof(pkt), TMO_FEVR);

/* マルチキャストアドレス 225.6.7.8 への参加 */
addr.ipaddr = byte4_to_long(ipaddr);
addr.portno = 1100;
udp_set_opt(cepid, IP_ADD_MEMBERSHIP, (VP)&addr, sizeof(addr));

/* マルチキャストパケットの受信 */
udp_rcv_dat(cepid, &dstaddr, &pkt, sizeof(pkt), TMO_FEVR);

/* マルチキャストアドレス 225.6.7.8 への解除 */
udp_set_opt(cepid, IP_DROP_MEMBERSHIP, (VP)&addr, sizeof(addr));
```

※マルチキャストパケットの送受信を行うにはデバイスドライバがマルチキャストパケットの送受信を行える設定になっている必要があります。

udp_get_opt

機能 UDP 通信端点オプションの参照

形式 ER udp_get_opt(ID cepid, INT optname, VP optval, INT optlen);

cepid UDP 通信端点 ID

optname オプションの種類

optval オプション値を取得するバッファのポインタ

optlen オプション取得するバッファのサイズ

optname は以下が使用できます。

SET_IP_TTL

通信端点に設定されている TTL の値を取得する

型 UB optval TTL 値

SET_IP_TOS

通信端点に設定されている TOS の値を取得する

型 UB optval TOS 値

戻値

E_OK	正常終了
E_ID	不正 ID 番号
E_NOEXS	UDP 通信端点が未生成
E_PAR	無効なオプションの種類
E_OBJ	optval の値が不正
E_NOSPT	未サポート機能
E_LNK	プロトコルスタック未初期化

解説 設定されている UDP 通信端点のオプション値を取得します。この API はプロトコルスタック Ver4.08.11 以降でサポートされました。

第 8 章 コールバック

ノンブロッキングコールの完了

機能 ノンブロッキングコールの完了通知

形式 ER callback(ID cepid, FN fncd, VP parblk);

cepid TCP または UDP 通信端点 ID

fncd 終了したサービスコールの機能コード

parblk エラーコードが格納されている領域へのポインタ

戻値 戻り値は未使用 (0)

解説 ノンブロッキングコールの処理が完了した、あるいは、キャンセルされた場合に、プロトコルスタックから呼び出されます。

parblkは、ER ercd = *(ER *)parblk とキャストしてアクセスしてください。ここには、サービスコールへ返すべきだったエラーコードが格納されています。

上記形式において、callback と表現さているのは、TCP 通信端点生成時にユーザーが定義したコールバックルーチンで、その関数名は任意です

サービスコールの機能コード：

TFN_TCP_ACP_CEP (-0x205) tcp_acp_cep 完了通知
TFN_TCP_CON_CEP (-0x206) tcp_con_cep 完了通知
TFN_TCP_CLS_CEP (-0x208) tcp_cls_cep 完了通知
TFN_TCP_SND_DAT (-0x209) tcp_snd_dat 完了通知
TFN_TCP_RCV_DAT (-0x20a) tcp_rcv_dat 完了通知
TFN_TCP_GET_BUF (-0x20b) tcp_get_buf 完了通知
TFN_TCP_RCV_BUF (-0x20d) tcp_rcv_buf 完了通知

TFN_UDP_SND_DAT (-0x223) udp_snd_dat 完了通知
TFN_UDP_RCV_DAT (-0x224) udp_rcv_dat 完了通知

緊急データの受信

機能 緊急データの受信通知

形式 ER callback(ID cepid, FN fncd, VP parblk);

cepid TCP 通信端点 ID

fncd TEV_TCP_RCV_OOB のみ

parblk 緊急データの長さが格納されている領域へのポインタ

戻値 戻り値は未使用 (0)

解説 緊急データを受信した場合に呼び出されます。コールバックルーチンの中でtcp_rcv_oobを使って緊急データを取り出す必要があります。取り出さずにコールバックルーチンからリターンすると、緊急データは捨てられます。

parblkは、INT len = *(INT *)parblkとキャストしてアクセスしてください。ここには、緊急データ長が格納されています。

上記形式において、callbackと表現さているのは、ノンブロッキングコール完了通知用の関数と同じものです。

※現バージョンでは、tcp_rcv_oob をサポートしていません。

UDP パケットの受信

機能 UDP パケットの受信通知

形式 ER callback(ID cepid, FN fncd, VP parblk);

cepid UDP 通信端点 ID

fncd TEV_UDP_RCV_DAT のみ

parblk パケットの長さが格納されている領域へのポインタ

戻値 戻り値は未使用 (0)

解説 udp_rcv_datがペンディングしていない状態で、すなわち、UDPの受信要求がキューイングされていない状態で、UDP パケットを受信した場合に呼び出されます。コールバックルーチンの中でudp_rcv_datを使って受信パケットを取り出す必要があります。取り出さずにコールバックルーチンからリターンすると、受信パケットは捨てられます。

parblkは、INT len = *(INT *)parblkとキャストしてアクセスしてください。ここには、受信パケットの長さが格納されています。

上記形式において、callback と表現さているのは、UDP 通信端点生成時にユーザーが定義したコールバックルーチンで、その関数名は任意です。

第 9 章 独自システム関数

プロトコルスタックの初期化

機能 プロトコルスタックの初期化

形式 ER tcp_ini();

戻値 0 以上 正常終了
E_ID タスク ID、メールボックス ID、メモリプール ID、アラームハンドラ ID のいずれかが不足
E_SYS 管理ブロック用のメモリが確保できない
E_NOMEM スタック用のメモリが確保できない

解説 プロトコルスタックの内部で使用する資源の生成とデータの初期化を行います。各サービスコールを呼び出す前に必ず一度だけタスク・コンテキストから呼び出してください。tcp_ini()前に TCP/UDP の各 API を呼び出すと E_LNK (-190)がリターンします。

プロトコルスタックの終了

機能 プロトコルスタックの終了

形式 ER tcp_ext(T_NIF *nif);
nif ネットワーク I/F 制御ブロック

戻値 E_OK 正常終了
負の値 内部で OS 資源の生成に失敗した

解説 tcp_ini()で生成したリソースを開放し、プロトコルスタックを終了します。この関数を呼び出した後でプロトコルスタックを使用する場合は、再度 tcp_ini()を呼び出す必要があります。デフォルトのネットワーク I/F を終了するにはネットワーク I/F 制御ブロックは NULL を設定してください。この API はプロトコルスタック Ver4.08.11 以降でサポートされました。

デフォルト・ネットワーク・インターフェースのオプションの取得

機能 デフォルト・ネットワーク・インターフェースのオプション取得

形式 ER net_get_opt(INT optname, const VP optval, INT optlen);

optname オプションの種類

optval オプション値を取得するバッファのポインタ

optlen オプション取得するバッファのサイズ

optname には以下の情報を取得できます

SET_IF_MTU	MTU の値を取得します
SET_IP_TTL	TTL の値を取得します
SET_IP_MTTL	マルチキャストパケットの TTL の値を取得します
SET_IP_REASM_TMO	IP フラグメント再構築タイムアウトを取得します
SET_TCP_SYN_RCNT	SYN の再送回数を取得します
SET_TCP_DAT_RCNT	データの再送回数を取得します
SET_TCP_RTO_INI	TCP 再送時間の初期値を取得します
SET_TCP_RTO_MIN	TCP 再送タイムアウトの下位境界を取得します
SET_TCP_RTO_MAX	TCP 再送タイムアウトの上位境界を取得します
SET_TCP_KEEPALIVE_TMO	キープアライブタイムアウトを取得します
SET_TCP_KEEPALIVE_PRO	キープアライブプローブタイムアウトを取得します
SET_TCP_KEEPALIVE_SUC	キープアライブプローブインターバルを取得します
SET_TCP_DACK_TMO	遅延 ACK の遅延時間を取得します
SET_TCP_DUP_ACK	重複 ACK をエラーとして扱う回数を取得します。

各オプションで指定するタイプは次のようになります

SET_IF_MTU	UH 型
SET_IP_TTL	UB 型
SET_IP_MTTL	UB 型
SET_IP_REASM_TMO	UH 型
SET_TCP_SYN_RCNT	UH 型
SET_TCP_DAT_RCNT	UH 型
SET_TCP_RTO_INI	UW 型
SET_TCP_RTO_MIN	UW 型
SET_TCP_RTO_MAX	UW 型
SET_TCP_KEEPALIVE_TMO	UW 型
SET_TCP_KEEPALIVE_PRO	UW 型
SET_TCP_KEEPALIVE_SUC	UW 型
SET_TCP_DACK_TMO	UH 型
SET_TCP_DUP_ACK	UH 型

戻値 E_OK 正常終了

解説 設定されているデフォルト・ネットワーク・インターフェースのオプションを取得します。

デフォルト・ネットワーク・インターフェースのオプション設定

機能 デフォルト・ネットワーク・インターフェースのオプション設定

形式 ER net_set_opt(INT optname, const VP optval, INT optlen);

optname オプションの種類
optval オプション値が設定されているバッファのポインタ
optlen オプション値の長さ

optname には以下が設定できます

SET_IF_MTU	MTU の値を設定します
SET_IP_TTL	TTL の値を設定します
SET_IP_MTTL	マルチキャストパケットの TTL の値を設定します
SET_IP_REASM_TMO	IP フラグメント再構築タイムアウトを設定します
SET_TCP_SYN_RCNT	SYN の再送回数を設定します
SET_TCP_DAT_RCNT	データの再送回数を設定します
SET_TCP_RTO_INI	TCP 再送時間の初期値を設定します
SET_TCP_RTO_MIN	TCP 再送タイムアウトの下位境界を設定します
SET_TCP_RTO_MAX	TCP 再送タイムアウトの上位境界を設定します
SET_TCP_KEEPALIVE_TMO	キープアライブタイムアウトを設定します
SET_TCP_KEEPALIVE_PRO	キープアライブプローブタイムアウトを設定します
SET_TCP_KEEPALIVE_SUC	キープアライブプローブインターバルを設定します
SET_TCP_DACK_TMO	遅延 ACK の遅延時間を設定します
SET_TCP_DUP_ACK	重複 ACK をエラーとして扱う回数を設定します。

各オプションで指定するタイプは次のようになります

SET_IF_MTU	UH 型
SET_IP_TTL	UB 型
SET_IP_MTTL	UB 型
SET_IP_REASM_TMO	UH 型
SET_TCP_SYN_RCNT	UH 型
SET_TCP_DAT_RCNT	UH 型
SET_TCP_RTO_INI	UW 型
SET_TCP_RTO_MIN	UW 型
SET_TCP_RTO_MAX	UW 型

SET_TCP_KEEPALIVE_TMO UW 型
SET_TCP_KEEPALIVE_PRO UW 型
SET_TCP_KEEPALIVE_SUC UW 型
SET_TCP_DACK_TMO UH 型
SET_TCP_DUP_ACK UH 型

戻値 E_OK 正常終了

解説 デフォルト・ネットワーク・インターフェースへオプションを設定します。

デフォルト・ネットワーク・インターフェースに設定されている IP アドレスの変更

機能 デフォルト・ネットワーク・インターフェースに設定されている IP アドレスの変更

形式 ER net_chg_ipa(T_NIF_ADDR *addr, UB level);
addr ネットワークインターフェースに設定するアドレス情報
level 設定レベル

```
typedef struct t_nif_addr {
  UB *hwaddr;    /* ハードウェアアドレス(未使用) */
  UB *ipaddr;    /* デフォルト IP アドレス */
  UB *gateway;   /* デフォルトゲートウェイ */
  UB *mask;      /* サブネットマスク */
} T_NIF_ADDR;
```

戻値 E_OK 正常終了

解説 デフォルト・ネットワークのアドレス設定を変更します。設定レベルを 0 にすると、デフォルトネットワークインターフェースで設定されている IP アドレス、デフォルトゲートウェイ、サブネットマスク変更します。設定レベルを 1 にするとさらに通信端点で設定されている IP アドレスも変更します。この関数ではハードウェアアドレスは変更されません。設定レベルを 2 に設定すると、処理中の各 API が E_ADDR(-191) でリターンします。待ち状態になっている処理は起床されます。

例

```
T_NIF_ADDR new_addr;

UB ipaddr[4]   = { 192, 168, 0, 99 };
UB gateway[4]  = { 192, 168, 0, 1  };
UB net_mask[4] = { 255, 255, 255, 0 };

new_addr.ipaddr  = ipaddr;
new_addr.gateway = gateway;
```

```
new_addr.mask    = net_mask;

ercd = net_chg_ipa(&new_addr, 0);
```

ARP テーブルに情報を追加する

機能 ARP テーブルに情報を追加する

形式 ER arp_add_entry(UW ipaddr, UB *macaddr, UW type);

ipaddr	登録する IP アドレス
macaddr	登録する MAC アドレスが格納されたバッファへのポインタ
type	タイプ(ARP_STATIC または ARP_DYNAMIC)

戻値

E_OK	正常終了
E_PAR	MAC アドレスへのポインタが NULL
E_OBJ	IP アドレスまたはタイプが不正
E_NOMEM	ARP テーブルが一杯

解説 ARP テーブルに情報を追加します。ARP テーブルはデフォルトで 2 分間、登録アドレスと通信が行われない場合、クリアされます。ARP_STATIC 指定で追加されたアドレスは自動的にクリアが行われません。

ARP テーブルに情報を追加する（ネットワークインターフェース名指定）

機能 ARP テーブルに情報を追加する（ネットワークインターフェース名指定）

形式 ER arp_add_byname(const char *name, UW ipaddr, UB *macaddr, UW type);

name	ネットワークインターフェース名（“eth1”等）
ipaddr	登録する IP アドレス
macaddr	登録する MAC アドレスが格納されたバッファへのポインタ
type	タイプ(ARP_STATIC または ARP_DYNAMIC)

戻値

E_OK	正常終了
E_PAR	MAC アドレスへのポインタが NULL
E_OBJ	インターフェース名、IP アドレスまたはタイプが不正
E_NOMEM	ARP テーブルが一杯

解説 指定したネットワークインターフェースで使用されている ARP テーブルに情報を追加します。ARP テーブルはデフォルトで 2 分間、そのアドレスと通信が行われない、クリアされます。ARP_STATIC 指定で追加されたアドレスは自動的にクリアが行われません。

ARP テーブルから情報を削除する

機能 ARP テーブルから情報を削除する

形式 ER arp_del_entry(UW ipaddr);

ipaddr 削除する IP アドレス

戻値 E_OK 正常終了

E_OBJ IP アドレスが不正か見つからない

解説 ARP テーブルから指定した IP アドレスを削除します

ARP テーブルから情報を削除する（ネットワークインターフェース名指定）

機能 ARP テーブルから情報を削除する（ネットワークインターフェース名指定）

形式 ER arp_del_byname(const char *name, UW ipaddr);

name ネットワークインターフェース名（“eth1”等）

ipaddr 削除する IP アドレス

戻値 E_OK 正常終了

E_OBJ IP アドレスが不正か見つからない

解説 指定したネットワークインターフェースから ARP テーブルから指定した IP アドレスを削除します。

（余白）

索引

NORTi Version 4 ユーザーズガイド

TCP/IP 編

2000 年　4 月 第 1 版
2000 年 11 月 第 2 版
2001 年　5 月 第 3 版
2002 年　4 月 第 4 版
2002 年　6 月 第 5 版
2004 年　4 月 第 6 版
2005 年　3 月 第 7 版
2008 年 12 月 第 8 版
2010 年　1 月 第 8 版 第 2 刷
2015 年　4 月 第 8 版 第 3 刷

株式会社ミスポ http://www.mispo.co.jp/
〒222-0033 横浜市港北区新横浜3-20-8
TEL 044-829-3381 FAX 044-829-3382
一般的なお問い合せ sales@mispo.co.jp
技術サポートご依頼　norti@mispo.co.jp