On-Chip Embedded Network Solution

NORTi® Oceans

TCP/IP編

ユーザーズガイド

MiSPO

目次

第 1 章 導入 1
1.1 はじめに 1
1.2 特長 1
1.3 制限事項 1
1.4 ファイル構成 2
nocnet.h 2
nocnetc.h 2
nocnets.h 2
nocnitod.h 2
nocnelan.c 2
1.5 用語 3
通信端点 3
TCP 通信端点の状態 3
サービスコール 4
タイムアウト 4
ノンブロッキング 4
コールバック 4
省コピーAPI 4
パケット 4

第 2 章 プロトコルスタックの構成 5
2.1 概要 5
階層構造 5
プロトコル制御タスク 6
プロトコルスタックのメモリプール 6
プロトコルスタックのメールボックス 7
タイムアウトとキャンセル 7
2.2 IP モジュール 8
使用する資源 8
パケット受信 8
パケット送信 8
2.3 ARP モジュール 9
使用する資源 9
ARP テーブル 9
ARP 問い合わせ 9
ARP 応答 9
ARP のタイムアウト 9
2.4 ICMP モジュール 10
使用する資源 10
Echo 処理 10
icmp_def_cbk 10
コールバック 11
icmp_snd_dat 12
2.5 UDP モジュール 13
使用する資源 13
UDP パケット送信 13
UDP パケット受信 13
2.6 TCP モジュール 14
使用する資源 14
IP モジュールとの関係 14

ARP モジュールとの関係 14
キープアライブ 15
ストリーム型通信方式 16
再送信 16
高速再転送/高速リカバリ 17
ゼロ・ウィンドウ・プローブ 17
2.7 IGMP モジュール 18
使用する資源 18
マルチキャストグループ 18
IGMPv1 ルータ 18
マルチキャストパケット送受信 18

第 3 章 コンフィグレーション 19
定義 19
ID の自動割り当て 21
端点、受付口の ID 自動割り当て 21
ローカル IP アドレスと MAC アドレスの設定 21
デフォルトゲートウェイとサブネットマスクの設定 21

第 4 章 共通定義 22
4.1 バイトオーダー変換 22
4.2 エラーコード取り出し 22
4.3 構造体 23
4.4 メインエラーコード 24

第 5 章 ユーティリティ・マクロ 25
5.1 ユーティリティ・マクロ 25
htonl 25
htons 25
ntohl 25
ntohs 25
5.2 ユーティリティ関数 26
byte4_to_long 26
long_to_byte4 26
ascii_to_ipaddr 26
ipaddr_to_ascii 26

第 6 章 TCP サービスコール 27
TCP サービスコール一覧 27
tcp_cre_rep 28
tcp_vcre_rep 29
tcp_del_rep 30
tcp_cre_cep 31
tcp_vcre_cep 33
tcp_del_cep 34
tcp_acp_cep 35
tcp_con_cep 37
tcp_sht_cep 39
tcp_cls_cep 40
tcp_snd_dat 42
tcp_rcv_dat 44
tcp_get_buf 46
tcp_snd_buf 48

tcp_rcv_buf 49
tcp_rel_buf 51
tcp_can_cep 52
tcp_set_opt 53
tcp_get_opt 56

第 7 章 UDP サービスコール 59
UDP サービスコール一覧 59
udp_cre_cep 60
udp_vcre_cep 61
udp_snd_dat 62
udp_rcv_dat 64
udp_can_cep 66
udp_set_opt 67
udp_get_opt 70

第 8 章 コールバック 71
ノンブロッキングコールの完了 71
UDP パケットの受信 72
ARP コールバック 73

第 9 章 独自システム関数 75
独自システム関数一覧 75
tcp_ini 76
tcp_nif_ini 77
net_get_opt 79
net_set_opt 81
net_chg_ipa 83
netif_get_opt 84
netif_get_byname 85
netif_set_opt 86
netif_set_byname 87
netif_chg_ipa 88
netif_chg_byname 89
getnif_default 90
getnif_from_name 90
getnif_from_ch 91
getnif_addr 91
arp_add_entry 92
arp_add_byname 93
arp_del_entry 94
arp_del_byname 94
arp_snd_req 95
arp_req_byname 95
arp_def_cbk 96
arp_rel_pkt 98
tcp_ref_cep 99

第 1 章 導入

1.1 はじめに

NORTi Oceans は、μITRON4.0 仕様サブセットのリアルタイム OS、ITRON TCP/IP API 仕様サブセットの TCP/IP プロトコルスタック、ミドルウェアから構成される製品です。

本マニュアルは、NORTi Oceans TCP/IP プロトコルスタックのユーザーズガイドです。プロトコルスタックの構成概要､コンフィグレーションの方法や提供するサービスコールについて説明しています。本製品のリアルタイム OS については「NORTi Oceans ユーザーズガイド・カーネル編」を参照してください。ミドルウェアについては「NORTi Oceans ユーザーズガイド・ミドルウェア編」を参照してください。

1.2 特長

NORTi Oceans TCP/IP プロトコルスタックは、リソースが極めて少ない組み込みシステムで利用されることを想定した製品です。NORTi Oceans カーネルと同様に ITRON TCP/IP API 仕様から厳選したサービスコールだけを実装しています。NORTi Version4 でサポートしている TCP、UDP、IP、ARP、ICMP、IGMP は同様にサポートしつつ、プロトコルスタックのためのメモリサイズがより小さくなりました。

メモリリソースが少なく ITRON TCP/IP API 仕様のプロトコルスタックを利用した開発を行うのが難しかった組み込みシステムは、NORTi Oceans を利用するとプロトコルスタックが必要とするメモリサイズがより小さいため、メモリ不足に悩むことが減り、開発が容易になります。

1.3 制限事項

NORTi Version 4 TCP/IP プロトコルスタックでサポートし、本製品でサポートしない機能は次の通りです。

- IP プロトコルのリアセンブル/フラグメンテーション機能
- UDP 通信端点の削除サービスコール(udp_del_cep, TFN_UDP_DEL_CEP)
- プロトコルスタックの終了サービスコール(tcp_ext)

1.4 ファイル構成

NORTi Oceans 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。

NORTiOC¥(CPU)¥INC¥

nocnet.h	NORTi Oceans TCP/IP 標準ヘッダ
nocnetc.h	NORTi Oceans TCP/IP コンフィグレーションヘッダ
nocnets.h	NORTi Oceans TCP/IP 内部定義ヘッダ
nocnitod.h	NORTi Oceans 数値/文字列変換関数ヘッダ

NORTiOC¥(CPU)¥SRC¥

nocnelan.c	ネットワーク・ドライバ・インターフェースのソース

NORTiOC¥(CPU)¥LIB¥(CCC)¥

nocxxxx.a	NORTi Oceans ライブラリ

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。NORTi Oceans では、TCP/IP プロトコルスタックライブラリというものは無く、カーネル、TCP/IP プロトコルスタックとミドルウェアを含む一つのライブラリを提供しています。

nocnet.h

nocnet.h を、NORTi Oceans 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、サービスコールを呼び出す時に必要な構造体や定数が定義されています。

nocnetc.h

nocnetc.h を、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 "nocnetc.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 Oceans TCP/IP のコンフィグレーションを行うことができます。

nocnets.h

nocnetc.h 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。

nocnitod.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。

nocnelan.c

ネットワーク・ドライバ・インターフェース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。

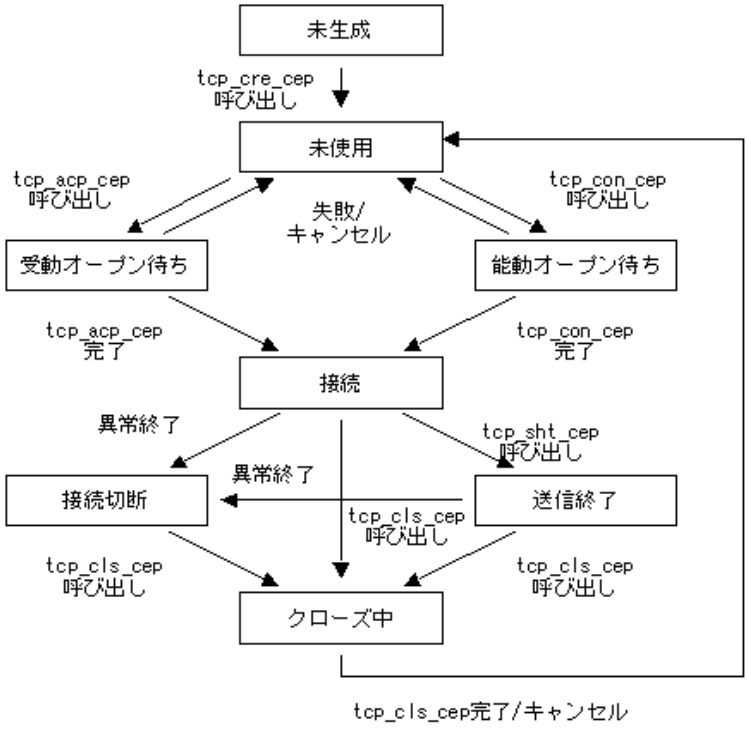

図 1 TCP 通信端点の状態遷移

「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呼びます。(udp_del_cep の ITRON TCP/IP API はサポートしていません。)

タイムアウト

待ちを生じ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。指定されたタイムアウト時間を経過してもサービスコールが完了しない場合、処理が中断され、タイムアウトエラー(E_TMOUT)となります。

ノンブロッキング

待ちを生じ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。ノンブロッキングを指定したサービスコールは、プロトコルスタックに処理を発行し、その処理の完了を待たずに戻ります。サービスコールにより発行された処理は、プロトコルスタックでタイムアウトすること無く、処理が完了するまで継続します。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)の完了通知は、次のコールバック機能により行います。

コールバック

アプリケーションが登録した関数(コールバックルーチン)をシステム側(この場合にはプロトコルスタック側)から呼び出すことをコールバックと呼びます。

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)などの言葉で表現されます。本書ではパケットに統一しています。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

本章では、NORTi Oceans TCP/IP プロトコルスタックの構成と動作について、下記のモジュール別に説明しています。

IP モジュール
ARP モジュール
ICMP モジュール
UDP モジュール
TCP モジュール
IGMP 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 Oceans TCP/IP 独自のものです。ARP や ICMP 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 Oceans TCP/IP プロトコルスタックは以下のモジュールで構成されています。

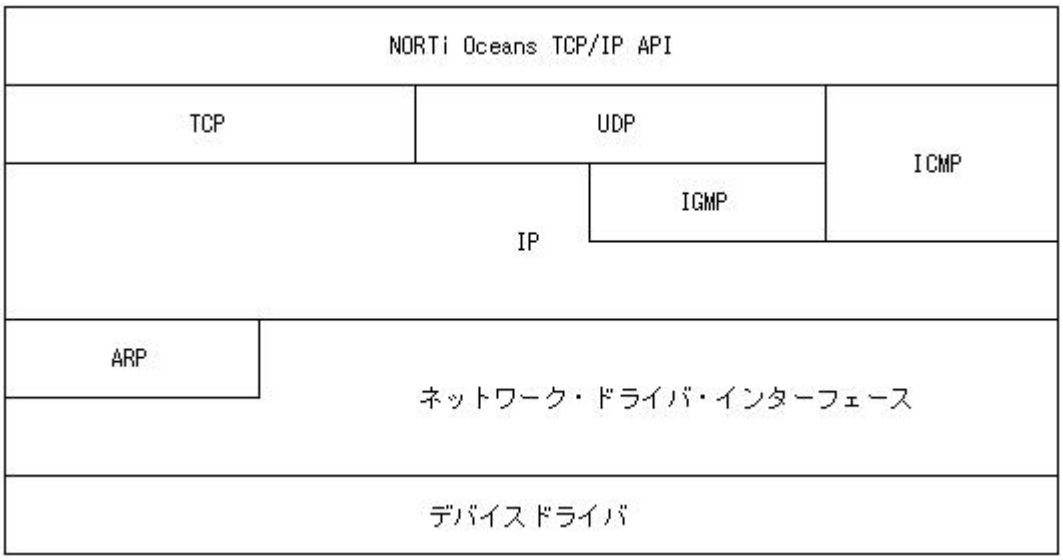

図 2 モジュールの階層構造

プロトコル制御タスク

NORTi Oceans TCP/IP プロトコルスタックは、ネットワーク・ドライバ・インターフェース毎に存在する次の 2 つのタスクと全体で 1 つの周期ハンドラから構成されています。

TCP/IP 受信タスク	ネットワーク・ドライバ・インターフェース毎に 1 個
TCP/IP 送信タスク	ネットワーク・ドライバ・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、TCP/IP 受信タスク、TCP/IP 送信タスクと TCP/IP タイマで制御されます。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で使用するコンフィグレーション可能なメモリプールには、次のものがあります。いずれも固定長のメモリプールです。

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利用されます。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 1 個です。メモリブロックのサイズは 1556 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ます。また、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 送信は他のプロトコルと仕様が異なります。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 84 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が使用す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・ドライバ・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・ドライバ・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するメールボックスには、各モジュールから送信したパケットがキューイングされます。キューから取り出したパケットは、順次、データリンク層へ渡されます。他に、送信を保留するための送信リトライキューとして使用するメールボックスがあります。

UDP 受信は他のプロトコルと仕様が異なります。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キューより取り外されます。ペンディング中処理のキャンセルのサービスコールを発行した場合も同様です。

NORTi Oceans TCP/IP では次の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ください。

ノンブロッキングを使用した場合も同様に、ノンブロッキングコールの完了通知が長時間あるいは永久に無い可能性があります。

2.2 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、TCP/IP 送信タスクと TCP/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送信リトライキューと呼ぶメールボックスが 2 つ存在します｡IP モジュールの TCP/IP 送信タスクと TCP/IP 受信タスクでは、IP だけでなく、ARP、ICMP、IGMP、UDP、TCP の制御も行います。

パケット受信

TCP/IP 受信タスクは、ネットワーク・ドライバ・インターフェースを通してデバイスドライバ関数を呼出し、パケットの受信を行います。取得するパケットが無いとデバイスドライバの処理で起床待ち状態に移行し、デバイスドライバの受信割り込み処理から起床されます。

TCP/IP 受信タスク上の IP モジュール部分では、受信したパケットの Ethernet ヘッダ、若しくは IP ヘッダの解析を行い、受信したパケットに応じた 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

TCP/IP 送信タスクは、ネットワーク・ドライバ・インターフェースを通してデバイスドライバ関数を呼出し、パケットの送信を行います。送信するパケットは送信パケットキューから取り出します。送信パケットキューが空の場合は送信パケットが取得できるまで待ち続けます。

キューから取り出した送信パケットは、ARP、ICMP、IGMP、UDP、TCP のパケットで、TCP/IP 送信タスクにより不足している IP ヘッダ、Ethernet ヘッダを設定されます。

デバイスドライバの送信バッファに空きが無いとデバイスドライバの処理で起床待ちに移行し、デバイスドライバの送信割り込み処理から起床されます。

2.3 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 Oceans TCP/IP では、TCP/IP 受信タスクと TCP/IP 送信タスクが ARP 処理を行い、アプリケーションがそれを意識する必要はありません。

ARP テーブル

プロトコルスタックには、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配列があります。ARP テーブルは、自 MAC 宛の ARP パケットの IP アドレスと MAC アドレスの対応関係を元に構築されます。ARP テーブルに空きが無くなると、古いものから順に上書きされます。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。接続するホスト数がこれよりも多い場合は、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 により設定します。ARP_CACHE_TOUT は、個々の情報の有効時間となります。その間、未参照の情報はテーブルより削除されます。また、エントリ参照の有無に関わらず、ARP テーブルの全エントリを定期的に削除する間隔となります。

ARP 問い合わせ

TCP/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、TCP/IP 送信タスクは、そのパケットの換わりに ARP パケットを送信し、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、TCP/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、TCP/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 TCP/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の場合は、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。

2.4 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。NORTi Oceans では、TCP/IP 受信タスクと TCP/IP 送信タスクが ICMP 処理を行い、アプリケーションがそれを意識する必要はありません。

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 TCP/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 */
    T_IP4_HEADER ip;     /* IP4 Header (without option data) */
    B data[PATH_MTU4];   /* Data (Variable Size)*/
} 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。

2.5 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 Oceans TCP/IP では、TCP/IP 受信タスクと TCP/IP 送信タスクが UDP 処理を行い、UDP 通信端点毎に UDP 受信キューと呼ぶメールボックスと UDP ヘッダ用メモリプールを利用します。

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。TCP/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。TCP/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。これは割り込みハンドラに似ています。

2.6 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 Oceans TCP/IP では、TCP/IP 受信タスクと TCP/IP 送信タスクで TCP 処理を行います。

IP モジュールとの関係

TCP/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/発信元ポート番号からそのパケットを処理する TCP 通信端点を探します。TCP 通信端点が存在しない場合はパケットを破棄します。TCP 通信端点が存在し、何かしらの応答を送信する時は、同タスク上で TCP パケット生成し、パケット送信キューに登録します。

TCP/IP 送信タスクでは、送信パケットキューを利用して TCP モジュールから渡された TCP パケットに IP ヘッダを設定し、ネットワーク上に送信します。再送信、キープアライブ・プローブ、ゼロ・ウィンドウ・プローブ、遅延確認応答の TCP パケットは本タスク上で生成しますが、他のタスクで作成した TCP パケットと同様、送信パケットキューを通して処理されます。

ARP モジュールとの関係

TCP モジュールに ARP モジュールの機能を組み込んであります。TCP パケットを生成する時、ARP モジュールを利用して IP アドレスに対応する MAC アドレスが参照できないと、TCP パケットの換わりに ARP パケットを送信キューに登録します。

ARP パケットを送信した TCP 通信端点は、ARP パケットにより MAC アドレスが解決すると、TCP パケットを生成し送信キューに登録します。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 問い合わせを行います。複数回の ARP 問い合わせを行っても解決できなかった場合は、TCP 通信端点が保持する再送信回数をインクリメントした後、TCP として再送信を試みるか、送信回数リトライオーバーの処理が行われます。

キープアライブ

NORTi Oceans TCP/IP では、キープアライブ・プローブはウィンドウ範囲外のシーケンス番号を持つ TCP パケットです。確立しているコネクション上で一定時間通信が行われない場合、コネクションの有効/無効を確認するためにキープアライブ・プローブを送信します。リモート TCP からそれに対する確認応答を受信しないと、インターバルを空けながら複数回送信します。一度も確認応答を受信しない場合、コネクションは無効と判断し閉じます。その時、処理待ちサービスコールは E_TMOUT エラーを返し､登録されているノンブロッキングコールに対しては E_TMOUT エラーの完了通知が行われます。また、コールバック関数を登録している場合は、キープアライブのタイムアウト通知が行われ、その関数の機能コードに TEV_TCP_TMO_KTIME(0x202)が設定さたコールバック通知されます。

キープアライブ設定は、次のマクロによりコンフィグレーショが行えます。TCP_KTIME_INI を 0 と設定するとキープアライブ機能が無効になります。

```
#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(sec) */
#define TCP_KTIME_PRO    600  /* キープアライブタイムアウト(sec) */
#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(sec) */
```

キープアライブのパラメータの関係は以下となります。

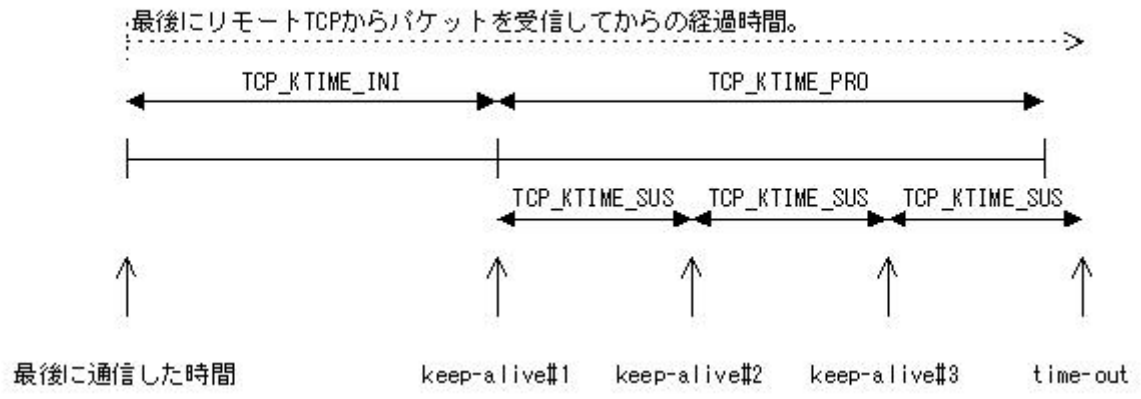

図 3 キープアライブ

キープアライブの最大送信回数は、TCP_KTIME_PRO と TCP_KTIME_SUC の設定値により決まり、次の式となります。

キープアライブの最大送信回数 ＝ TCP_KTIME_PRO ／ TCP_KTIME_SUC（端数切り上げ）

なお、キープアライブ・プローブ及びそれに対する確認応答は、TCP により信頼できる送信を行わないため、キープアライブの最大送信回数が小さくならないよう考慮し、コンフィグレーションしてください。

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固まりのデータを通信するのに対し、TCP では連続する区切りのないデータ列を通信する方式となります。これをストリーム型通信方式と言います。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することもあれば、500 バイト単位で分割されて受信したり、直前に送信されていたデータと一緒に受信することもありえると言うことです。これをひと固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つ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リモート TCP 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リモート TCP からの応答時間（ラウンドトリップタイム）を測定することで、回線の速度に合わせて動的に変化します。

NORTi Oceans TCP/IP ではこのタイムアウト時間の上限値(TCP_RTO_UBOUND)と下限値(TCP_RTO_LBOUND)の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。再送信を行う回数は、コンフィグレーションにより指定された回数となります。

コンフィグレーションに従い、再送信を繰り返しても確認応答を受信しない場合は、送信回数リトライオーバーと判断してコネクションを閉じます。 その時、処理待ちのサービスコールは E_TMOUT エラーを返し、登録されているノンブロッキングコールに対しては、E_TMOUT エラーの完了通知が行われます。

再送信の設定は、次のマクロによりコンフィグレーショが行えます。

```
#define TCP_SYN_RETRY    3      /* SYN 送信の再送信回数 */
#define TCP_DATA_RETRY   12     /* データ送信, FIN 送信の再送回数 */
#define TCP_RTO_INI      (3000/MSEC)
                        /* ラウンドトリップタイム未測定時のタイムアウト時間(msec) */
#define TCP_RTO_UBOUND   (64000/MSEC) /* タイムアウト時間の上限値(msec) */
#define TCP_RTO_LBOUND   (300/MSEC)   /* タイムアウト時間の下限値(msec) */
```

図 4 SYN 送信(ラウンドトリップタイム未測定時)

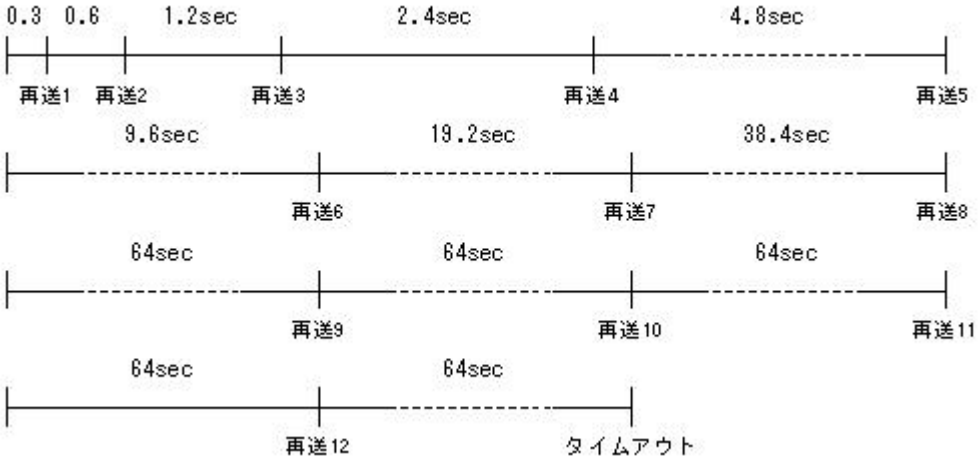

図 5 データ送信(ラウンドトリップタイム時間から計測したタイムアウト時間=100msec)

高速再転送/高速リカバリ

リモート TCP から既に確認応答済みの ACK 番号を持ち、データを含まず、且つ、広告されたウィンドウサイズが更新されていない ACK を受信した場合は duplicate ACK として認識します。連続した duplicate ACK の受信回数がコンフィグレーションにより設定した値以上になった時、確認応答が取れないセグメントが消失したものと見なし、そのセグメントから再送信を行います。

高速再転送/高速リカバリの設定は、次のマクロによりコンフィグレーショが行えます。

```
#define TCP_DUP_ACK      3       /* 3 Duplicate ACK's */
```

ゼロ・ウィンドウ・プローブ

NORTi Oceans TCP/IP では、データ送信時にリモート TCP のウィンドウに空きが無い場合、ゼロ・ウィンドウ・プローブとして、1 バイト分のデータを含む TCP パケットを送信します。ゼロ・ウィンドウ・プローブに対する ACK を受信する間は、コネクションを切断しません。

ゼロ・ウィンドウ・プローブを送信する間隔は、再送信の間隔と同じく送信毎に指数倍に変化し、間隔の上限値は TCP_RTO_UBOUND、下限値は TCP_RTO_LBOUND のコンフィグレーションに従います。

2.7 IGMP モジュール

使用する資源

IGMP 専用のタスクは存在しません。NORTi Oceans では、TCP/IP 受信タスクと TCP/IP 送信タスクが IGMP 処理を行います。

マルチキャストグループ

NORTi Oceans TCP/IP では、マルチキャストグループの参加・脱退は UDP 通信端点毎に設定します。UDP 通信端点をマルチキャストグループに参加させるには、IP_ADD_MEMBERSHIP オプションを指定した udp_set_opt サービスコールを発行する必要があります。脱退させるには、IP_DROP_MEMBERSHIP オプションを指定した udp_set_opt サービスコールを発行する必要があります。同時に参加可能なマルチキャストグループ数には制限があり、次のマクロによりコンフィグレーションが行えます。

```
#define MULTICAST_GROUP_CNT 10
```

IGMPv1 ルータ

NORTi Oceans TCP/IP では、通常、IGMPv2 で動作します。同一ネットワーク上に IGMPv1 で動作するルータが存在し、プロトコルスタックが IGMPv1 ルータからの IGMP クエリを受信すると、IGMPv1 で動作するように切り替わります。そして、そのクエリを一定期間受信しないと IGMPv2 の動作に戻ります。この期間は、次のマクロによりコンフィグレーションが行えます。

```
#define IGMPV1_PRESENT_TMO 400
```

マルチキャストパケット送受信

NORTi Oceans TCP/IP では、マルチキャストパケットの送受信を UDP 通信端点で行います。マルチキャストアドレスの送信は、宛先 IP アドレスをマルチキャスト IP アドレスし、UDP 通信端点を利用してパケット送信を行うだけです。マルチキャストパケットの受信は、UDP 通信端点をマルチキャストグループに参加させた後、UDP 通信端点を利用してパケット受信処理を行う必要があります。

なお、デフォルトの設定ではマルチキャストパケットの受信を行わないデバイスドライバが存在します。マルチキャストパケットの送受信が行えない場合は、デバイスドライバの仕様を確認してください。

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 nocnetc.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無い場合に使用されます。

```
#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       (AUTO_ID) TCP/IP で用いるタスクの先頭 ID
#define TCP_MBXID_TOP       (AUTO_ID) TCP/IP で用いるメールボックスの先頭 ID
#define TCP_MPFID_TOP       (AUTO_ID) TCP/IP で用いる固定長メモリプール先頭 ID

#define PRI_IP_SND_TSK      (4)       TCP/IP 送信タスクの優先度
#define PRI_IP_RCV_TSK      (4)       TCP/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以上の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      (692)     TCP/IP 送信タスクのスタックサイズ
#define SSZ_IP_RCV_TSK      (648)     TCP/IP 受信タスクのスタックサイズ

#define ETH_QCNT            (1)       Ethernet パケットの最大キューイング数
#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_NIF_CNT         (1)       ARP を利用するネットワーク I/F 数
#define ARP_RETRY           (3)       ARP リクエストの再送信する回数
#define ARP_RET_INTVAL      (2)       ARP の再送間隔(2 秒)
```

#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_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 IGMPV1_PRESENT_TMO	(400)	IGMPv1 Query 受信後、送信するメッセージを IGMPv1 形式にする期間(400 秒)
#define MULTICAST_GROUP_CNT	(10)	マルチキャストグループの数
#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/MSEC)	TCP 再送時間の初期値(3 秒)
#define TCP_RTO_UBOUND	(64000/MSEC)	TCP 再送タイムアウトの上位境界(64 秒)
#define TCP_RTO_LBOUND	(300/MSEC)	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)
#define TCP_DACK_TMO	(200/MSEC)	ACK 遅延時間(200 ミリ秒)
#define TCP_DUP_ACK	(3)	再送信までの Duplicate ACK 受信回数
#define TCP_TIMEWAIT_TMO	(0/MSEC)	TIME-WAIT からのタイムアウト時間
#define TCP_PORT_EPHEMERAL	(49152)	エフェメラルポートの初期値※1

```
#include "nocnetc.h"
```

※1 ウェルノウンポート(well-known port)より小さい値を設定しないでください。

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_KTIME_PRO ＞ 0

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 Oceans 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 AUTO_ID で設定した場合 NORTi Oceans TCP/IP は内部の ID は全て自動的に割り当てられます。(デフォルト)

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 Oceans 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、プロトコルスタックを初期化す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 バイトオーダー変換

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

4.2 エラーコード取り出し

NORTi Oceans 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ません。

4.3 構造体

```
typedef struct t_ipv4ep {
      UW ipaddr;          IP アドレス
      UH portno;          ポート番号
} T_IPV4EP;
typedef struct t_tcp_crep {
      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 t_tcp_ccep {
      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	未サポート機能
E_PAR	0xf..fef	-17	パラメータエラー
E_ID	0xf..fee	-18	不正 ID 番号
E_NOMEM	0xf..fdf	-33	メモリ不足
E_NOEXS	0xf..fd6	-42	オブジェクト未生成
E_OBJ	0xf..fd7	-41	オブジェクト状態エラー
E_QOVR	0xf..fd5	-43	キューイングオーバーフロー
E_RLWAI	0xf..fcf	-49	処理のキャンセル、待ち状態の強制解除
E_TMOUT	0xf..fce	-50	ポーリング失敗またはタイムアウト
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

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ー変換(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ーのロングワード(32 ビット)データを、ネットワークバイトオーダーに変換します。

htons

機能 ホスト→ネットワークバイトオーダー変換(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ーのショートワード(16 ビット)データを、ネットワークバイトオーダーに変換します。

ntohl

機能 ネットワーク→ホストバイトオーダー変換(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ーのロングワード(32 ビット)データを、ホストバイトオーダーに変換します。

ntohs

機能 ネットワーク→ホストバイトオーダー変換(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ーのショートワード(16 ビット)データを、ホストバイトオーダーに変換します。

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_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_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_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 t_tcp_crep {
        ATR repatr;       TCP 受付口属性(未使用, 0)
        T_IPV4EP myaddr; TCP 受付口の IP アドレスとポート番号
} T_TCP_CREP;

typedef struct t_ipv4ep {
        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_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 TCP 受付口の 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できます。

ポート番号 TCP_PORTANY(= 0)は無効なポート番号として扱っております。指定すると E_PAR エラーを返します。

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 Oceans 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 t_tcp_ccep {
        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。コールバックルーチンを
                        使用しない時には NULL を指定すること
} 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、サイズが不正
E_NOMEM	ウィンドウバッファの確保に失敗

解説 cepid で指定した ID の TCP 通信端点を生成します。生成された TCP 通信端点は、tcp_acp_cep サービスコールによる受動オープン、tcp_con_cep サービスコールによる能動オープンを行なうことで機能します。

送受信バッファ領域は、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、送受信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。

補足 バッファ領域先頭アドレスとして NULL を指定した時、バッファをプロトコルスタック内部で自動に確保します。一度自動確保されたバッファは tcp_del_cep()で通信端点を削除しても保持されます。tcp_cre_cep()で再生成する場合は、前回生成した時と同じサイズにし、バッファ先頭アドレスを NULL にしてください。自動確保せずに、バッファ先頭アドレスに有効なアドレスを指定した場合の通信端点の再生成では、アドレスやサイズを変

更できます。

TCP 通信端点の利用用途がデータ送信、もしくはデータ受信の一方だけでも、必ず両バッファ領域を指定してください。送受信バッファ領域のサイズは 1byte で指定することも可能ですがサイズが小さすぎると通信の効率が悪くなります。

受信バッファ領域は受信用のウィンドウバッファとして使われます。バッファ内の受信データサイズが 0 の場合、受信バッファ領域のサイズが接続先 TCP に広告するウィンドウサイズとなります。そのサイズがデバイスドライバ(LAN ドライバなど)の受信バッファサイズと比べて著しく大きい場合、一時的にデバイスとデバイスドライバのバッファに納まりきらないデータを受信し、データを消失することがあります。データの一部が消失すると受信性能が悪化しますので、過度に大きな受信バッファを利用しないことをお勧めします。

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。

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
E_NOMEM ウィンドウバッファの確保に失敗

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 Oceans 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

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	リモート TCP 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 t_ipv4ep{
        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
	TCP 受付口が未生成
E_OBJ	TCP 通信端点が使用中
	ポート番号既使用
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	TCP 受付口の IP アドレスが指定した TCP 通信端点には不適切
E_LNK	プロトコルスタック未初期化
E_ADDR	ネットワーク I/F の IP アドレスが変更された

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 及び SYN を送信します。SYN の ACK が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。

*p_dstaddr には、リモート TCP 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無い、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。

タイムアウトや tcp_can_cep などにより正常終了しなかった場合、TCP 通信端点は未使用状態に戻ります。

補足 リモート TCP 側 IP アドレス/ポート番号格納先へ書き込みは、SYN を受信した時に行ないます。ノンブロッキング(tmout = TMO_NBLK)により本サービスコールを発行した場合、書き込むタイミングで開放されているメモリをリモート TCP 側 IP アドレス/ポート番号格納先へのポインタに指定しないでください。

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)は接続後に書き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 通信端点側 IP アドレス/ポート番号構造体へのポインタ
p_dstaddr リモート TCP 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 t_ipv4ep{
        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_PAR	IP アドレスまたはポート番号が不正
E_CLS	接続要求が拒否された
E_ADDR	ネットワーク I/F の IP アドレスが変更された

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リモート TCP 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)指定ではリモート TCP との接続は確立できず、直ちに 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、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。

TCP 通信端点側の IP アドレスに IPV4_ADDRANY(= 0)を指定した場合、リモート TCP と同じサブネットに属するネットワーク I/F が存在するなら、そのネットワーク I/F の IP アドレスが設定されます。同じサブネットに属するネットワーク I/F が存在しない場合は、デフォルト・ネットワーク I/F が設定されます。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 NULL(= -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。

ノンブロッキング(tmout = TMO_NBLK)、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、リモート TCP からの SYN を受信できないとコールバック通知やタイムアウトが行われません。タイムアウト指定は、可能な限りタイムアウトする値を設定してください。(リモート TCP に送信した SYN に対する ACK を受信しない場合は、タイムアウトします。)

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
E_ADDR ネットワーク I/F の IP アドレスが変更された

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、通信端点のデータ送信が終了したことをリモート TCP 側に知らせるためのものです。

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 TCP 接続が異常切断した

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。

処理待ちしているサービスコールは全てキャンセルされます。ペンディングしているサービスコールからはE_RLWAIエラーが返り、登録されているノンブロッキングコールはエラーコード E_RLWAI が通知されます。

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。

TCP 通信端点から先に FIN を送信した場合、リモート TCP から FIN の ACK と FIN を受信した後、リモート TCP に送信した FIN の ACK が受信されていると確信できるまでの時間を待ちます。

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)で本サービスコールを発行した場合、リモート TCP に FIN の送信と RST の送信を行うと、直ぐに E_TMOUT エラーが返ります。リモート TCP から応答を待たずに接続が異常切断します。

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。

本サービスコール発行後、リモート TCP から RST などを受信し、接続が異常切断した場合、クローズ処理を直ちに終えます。

補足 TMO_FEVR を指定したときに、リモート TCP 側のクラッシュなどにより FIN を受信できないと本サービスコールから戻りません。

TMO_NBLK を指定したときも、同様に FIN を受信できないとコールバックが呼ばれないことがあります。タイムアウト時間は可能な限り指定してください。

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 通信端点が接続以外、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 接続が異常切断した

E_PAR len の長さが不正

E_ADDR ネットワーク I/F の IP アドレスが変更された

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し、データ送信が可能なら送信を行います。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 接続が異常切断し、受信バッファが空
E_PAR	len の長さが不正
E_ADDR	ネットワーク I/F の IP アドレスが変更された

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

タが無ければ、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、受信データが無くても、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。

リモート TCP から FIN フラグを含む TCP パケットを受信した後、受信済みのデータが無くなると、接続が正常切断されたとして、本サービスコールから 0 が返ります。

RST 受信や同じ TCP 通信端点の送信処理が送信回数リトライオーバーすると、接続は異常切断します。異常切断以後でも受信済みのデータは本サービスコールより取得することができます。受信済みのデータが無くなると本サービスコールから E_CLS 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 通信端点が接続状態以外、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 接続が異常切断した

E_ADDR ネットワーク I/F の IP アドレスが変更された

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 通信端点が接続状態以外、len が長すぎる

E_CLS TCP 接続が異常切断した

E_PAR len の長さが不正

E_ADDR ネットワーク I/F の IP アドレスが変更された

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 接続が異常切断され、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

E_ADDR ネットワーク I/F の IP アドレスが変更された

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、受信データが無ければ、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。

リモート TCP から FIN フラグを含む TCP パケットを受信した後、受信済みのデータが無くなると本サービスコールから 0 が返ります。

RST 受信や同じ TCP 通信端点の送信処理が送信回数リトライオーバーすると、接続は異常切断します。異常切断以後でも受信済みのデータは本サービスコールより取得することができます。受信済みのデータがなくなると本サービスコールから E_CLS が返ります。

受信処理はキューイングされません。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。

補足 本サービスコールで取得する領域は TCP 通信端点が管理する受信バッファ領域です。受信バッファはリングバッファとして扱っております。受信バッファが一杯になった状態で本サービスコールを発行しても、必ず一度に全データを含む領域を取得できるものではありません。

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 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

E_ADDR ネットワーク I/F の IP アドレスが変更された

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_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
TCP 通信端点の状態が未使用
E_ADDR ネットワーク I/F の IP アドレスが変更された

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_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
SET_TCP_TIMEWAIT_TMO	通信端点で使用する TIME-WAIT 時間を設定します。(単位：TMO(=ミリ秒/MSEC))
IP_IF_NAME	ネットワーク I/F 名指定で TCP 通信端点が利用するチャネルを指定します。
IP_IF_CH	チャンネル番号を指定で TCP 通信端点が利用するチャネルを指定します。
SET_TCP_DAT_RCNT	データの再送回数を設定します(単位：回)
SET_TCP_RTO_INI	TCP 再送時間の初期値を設定します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_RTO_MIN	TCP 再送タイムアウトの下位境界を設定します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_RTO_MAX	TCP 再送タイムアウトの上位境界を設定します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_KEEPALIVE_TMO	キープアライブタイムアウトを設定します(単位：秒)
SET_TCP_KEEPALIVE_PRO	キープアライブプローブインターバルを設定します(単位：秒)
SET_TCP_KEEPALIVE_SUC	キープアライブプローブタイムアウトを設定します(単位：秒)

各オプションで指定する optval の型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
SET_TCP_TIMEWAIT_TMO	UH 型
IP_IF_NAME	ネットワーク I/F 名へのポインタ(optlen は未使用)

IP_IF_CH	チャンネル番号(optlen は未使用)
SET_TCP_DAT_RCNT	UB 型
SET_TCP_RTO_INI	UH 型
SET_TCP_RTO_MIN	UH 型
SET_TCP_RTO_MAX	UW 型
SET_TCP_KEEPALIVE_TMO	UH 型
SET_TCP_KEEPALIVE_PRO	UH 型
SET_TCP_KEEPALIVE_SUC	UB 型

下記の optname は、右側に記したコンフィグレーション値を変更します。

SET_TCP_DAT_RCNT	TCP_DAT_RCNT
SET_TCP_RTO_INI	TCP_RTO_INI
SET_TCP_RTO_MIN	TCP_RTO_LBOUND
SET_TCP_RTO_MAX	TCP_RTO_UBOUND
SET_TCP_KEEPALIVE_TMO	TCP_KTIME_INI
SET_TCP_KEEPALIVE_PRO	TCP_KTIME_PRO
SET_TCP_KEEPALIVE_SUC	TCP_KTIME_SUC
SET_TCP_TIMEWAIT_TMO	TCP_TIMEWAIT_TMO

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 指定したネットワークインターフェースが存在しない
E_OBJ	通信端点が使用中
E_LNK	ネットワークインターフェースが見つからない
E_NOSPT	未サポートのパラメータが指定された

解説

SET_IP_TTL、SET_IP_TOS、SET_TCP_MTU、SET_TCP_TIMEWAIT_TMO は、TCP 通信端点の設定値の変更を行います。TCP 通信端点の状態が未使用の時、設定値の変更が可能です。

IP_IF_NAME, IP_IF_CH は、TCP 通信端点で使用するネットワーク I/F（チャネル）の変更を行います。TCP 通信端点の状態が未使用の時、設定値の変更が可能です。

SET_TCP_DAT_RCNT、SET_TCP_RTO_INI、SET_TCP_RTO_MIN、SET_TCP_RTO_MAX、SET_TCP_KEEPALIVE_TMO、SET_TCP_KEEPALIVE_PRO、SET_TCP_KEEPALIVE_SUC は、TCP 通信端点の設定値の変更を行います。設定値が変更可能になるのは、TCP 通信端点が通信先と接続後で且つデータ通信を始める前となります。

キープアライブタイムアウト値(SET_TCP_KEEPALIVE_TMO)を 0 に設定するとキープアライ

ブタイマーは停止(無効)します。

キープアライブ関連のパラメータ(SET_TCP_KEEPALIVE_TMO、 SET_TCP_KEEPALIVE_PRO、SET_TCP_KEEPALIVE_SUC)を変更すると、キープアライブ関連の内部変数とキープアライブタイマーは関連パラメータで初期化されます。

TCP 再送関連のパラメータ(SET_TCP_DAT_RCNT、SET_TCP_RTO_INI、SET_TCP_RTO_MIN、SET_TCP_RTO_MAX)を変更すると、TCP 再送関連の内部変数は関連パラメータで初期化されます。

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
SET_TCP_TIMEWAIT_TMO	通信端点で設定されている TIME-WAIT 時間の値を取得します (単位：TMO(=ミリ秒/MSEC))
IP_IF_NAME	通信端点で設定されている利用するネットワーク I/F 名を取得します
SET_TCP_DAT_RCNT	通信端点で設定されているデータの再送回数の値を取得します(単位：回)
SET_TCP_RTO_INI	通信端点で設定されている TCP 再送時間の初期値を取得します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_RTO_MIN	通信端点で設定されている TCP 再送タイムアウトの下位境界の値を取得します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_RTO_MAX	通信端点で設定されている TCP 再送タイムアウトの上位境界の値を取得します(単位：TMO(=ミリ秒/MSEC))
SET_TCP_KEEPALIVE_TMO	通信端点で設定されているキープアライブタイムアウトの値を取得します(単位：秒)
SET_TCP_KEEPALIVE_PRO	通信端点で設定されているキープアライブプローブインターバルの値を取得します(単位：秒)
SET_TCP_KEEPALIVE_SUC	通信端点で設定されているキープアライブプローブタイムアウトの値を取得します。(単位：秒)

各オプションで指定する optval の型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
SET_TCP_TIMEWAIT_TMO	UH 型
IP_IF_NAME	ネットワーク I/F 名へのポインタ
SET_TCP_DAT_RCNT	UB 型
SET_TCP_RTO_INI	UH 型
SET_TCP_RTO_MIN	UH 型
SET_TCP_RTO_MAX	UW 型
SET_TCP_KEEPALIVE_TMO	UH 型
SET_TCP_KEEPALIVE_PRO	UB 型
SET_TCP_KEEPALIVE_SUC	UH 型

下記の optname は、右側に記したコンフィグレーション値を取得します。

SET_TCP_DAT_RCNT	TCP_DAT_RCNT
SET_TCP_RTO_INI	TCP_RTO_INI
SET_TCP_RTO_MIN	TCP_RTO_LBOUND
SET_TCP_RTO_MAX	TCP_RTO_UBOUND
SET_TCP_KEEPALIVE_TMO	TCP_KTIME_INI
SET_TCP_KEEPALIVE_PRO	TCP_KTIME_PRO
SET_TCP_KEEPALIVE_SUC	TCP_KTIME_SUC
SET_TCP_TIMEWAIT_TMO	TCP_TIMEWAIT_TMO

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_NOSPT	未サポートのパラメータが指定された

解説 SET_IP_TTL、SET_IP_TOS、SET_TCP_MTU、SET_TCP_TIMEWAIT_TMO、SET_TCP_DAT_RCNT、SET_TCP_RTO_INI、SET_TCP_RTO_MIN、SET_TCP_RTO_MAX、SET_TCP_KEEPALIVE_TMO、SET_TCP_KEEPALIVE_PRO、SET_TCP_KEEPALIVE_SUC は、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。

IP_IF_NAME は、TCP 通信端点が使用するネットワーク I/F(チャネル)のネットワーク I/F 名を取得します。tcp_set_opt()により使用するネットワーク I/F(チャンネル)の設定を行なわなかった TCP 通信端点に対して行なっても、ネットワーク I/F 名は取得できません。

ネットワーク I/F 名の文字数よりサイズが小さいバッファを指定すると、E_PAR エラーを返します。

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_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, 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_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;
typedef struct t_ipv4ep {
        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_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を行うと E_QOVR エラーが返され、パケットは破棄されます。

自分側の IP アドレスに IPV4_ADDRANY(=0)を指定した場合、UDP パケットの送信は、デフォルト・ネットワーク I/F から行われます。受信は、 全てのネットワーク I/F に設定されている IP アドレス宛の UDP パケットが対象となります。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(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 Oceans TCP/IP 独自のシステムコールです。

udp_snd_dat

機能 UDP 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 t_ipv4ep {
        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_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 Oceans 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 UDP 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 t_ipv4ep {
        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_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。 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)は受信後に書き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 UDP 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 (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
IP_DROP_MEMBERSHIP	マルチキャストグループから脱退する
SET_IP_TTL	通信端点で使用する TTL の値を設定する
SET_IP_TOS	通信端点で使用する TOS の値を設定する
IP_IF_NAME	ネットワーク I/F 名指定で UDP 通信端点が利用するチャネルを指定します。
IP_IF_CH	チャンネル番号を指定で UDP 通信端点が利用するチャネルを指定します。

各オプションで指定するタイプは次のようになります。

IP_BROADCAST	BOOL
IP_ADD_MEMBERSHIP	UW
IP_DROP_MEMBERSHIP	UW
SET_IP_TTL	UB
SET_IP_TOS	UB
IP_IF_NAME	ネットワーク I/F 名へのポインタ(optlen は未使用)
IP_IF_CH	チャンネル番号(optlen は未使用)

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。

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(mreq));
```

※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
SET_IP_TOS	通信端点に設定されている TOS の値を取得する
IP_IF_NAME	通信端点で設定されている利用するネットワーク I/F 名を取得します

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
SET_IP_TOS	UB
IP_IF_NAME	ネットワーク I/F 名へのポインタ

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。

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、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

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。

ARP コールバック

機能 ARP モジュールのコールバック

形式 VP arp_callback(T_ARP *pkt);
pkt ARP 受信パケットへのポインタ

戻値 !NULL(==pkt) コールバックで処理を行わず、処理を NORTi Oceans に任せる。
NULL コールバックで ARP パケットを処理し、ARP パケットを解放する。
(NORTi Oceans に処理を任せない)

解説 ARP モジュールは ARP コールバック関数が登録されていると、次のような ARP パケットを受信した時にそのコールバック関数を呼び出します。

自身以外の IP アドレスを問合せる ARP 要求パケットを受信した時
送信元の IP アドレスと自身の IP アドレスが一致する ARP パケットを受信した時

コールバック関数は、T_ARP *pkt により受信した ARP パケットを参照することができます。そのパケットへの応答判断を NORTi Oceans TCP/IP に任せる場合は、戻り値に T_ARP *pkt の値を設定してください。

NORTi Oceans TCP/IP に処理させず、アプリケーションで処理させる場合は、戻り値に NULL を設定してください。受信パケットの処理が終わり、メモリを解放する時は arp_rel_pkt() を発行してください。

[例]

```
VP arp_callback(T_ARP *pkt)
{
     : (略)
    if (flg) {
        /* コールバックで受信 ARP パケットを処理する */
         : (略)
        arp_rel_pkt(pkt);
        pkt = NULL;
    } else {
        /* NORTi Oceans 仕様に任せる */
    }
   return pkt;
}
```

このコールバック関数はユーザーが定義するもので、任意の関数名で定義してください。

補足 NORTi Oceans TCP/IP 独自のコールバックです。

ARP モジュールにコールバック関数を登録する方法については arp_def_cbk()を参照してください。

第 9 章 独自システム関数

独自システム関数一覧

tcp_ini	プロトコルスタックの初期化
tcp_nif_ini	ネットワーク I/F の初期化
net_get_opt	デフォルト・ネットワーク I/F のオプション取得
net_set_opt	デフォルト・ネットワーク I/F のオプション設定
net_chg_ipa	デフォルト・ネットワーク I/F の IP アドレス設定
netif_get_opt	ネットワーク I/F のオプション取得
netif_get_byname	ネットワーク I/F のオプション取得(ネットワーク I/F 名指定)
netif_set_opt	ネットワーク I/F のオプション設定
netif_set_byname	ネットワーク I/F のオプション設定(ネットワーク I/F 名指定)
netif_chg_ipa	ネットワーク I/F の IP アドレス変更
netif_chg_byname	ネットワーク I/F の IP アドレス変更(ネットワーク I/F 名指定)
getnif_default	デフォルト・ネットワーク I/F 制御ブロックの取得
getnif_from_name	ネットワーク I/F 制御ブロックの取得(ネットワーク I/F 名指定)
getnif_from_ch	ネットワーク I/F 制御ブロックの取得(チャネル番号指定)
getnif_addr	ネットワーク I/F 制御ブロックの取得(IP アドレス指定)
arp_add_entry	デフォルト・ネットワーク I/F の ARP テーブルに情報を追加
arp_add_byname	ネットワーク I/F の ARP テーブルに情報を追加
arp_del_entry	デフォルト・ネットワーク I/F の ARP テーブルから情報を削除
arp_del_byname	ネットワーク I/F の ARP テーブルから情報削除
arp_snd_req	デフォルト・ネットワーク I/F から ARP 要求パケット送信
arp_req_byname	ネットワーク I/F から ARP 要求パケット送信
arp_def_cbk	ARP コールバック関数の登録/削除
arp_rel_pkt	ARP パケットのメモリブロック解放

tcp_ini

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、周期ハンドラ 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、及び、デフォルトネットワークインターフェース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。

デフォルトネットワークインターフェースは、ユーザーが定義した次の変数の設定値を利用します。

```
UB ethernet_addr[]   = { 0, 0, 0, 0, 0, 0 };  MAC アドレス
UB default_ipaddr[]  = { 0, 0, 0, 0 };        ローカル IP アドレス
UB subnet_mask[]     = { 255, 255, 255, 0 };  サブネットマスク
UB default_gateway[] = { 0, 0, 0, 0 };        デフォルトゲートウェイ
```

tcp_nif_ini

機能 ネットワーク I/F の初期化

形式 ER tcp_nif_ini(T_NIF *nif, const char *name, NIF_FP func, T_NIF_ADDR *addr);

nif	ネットワークインターフェース制御ブロックへのポインタ
name	ネットワークインターフェース名
func	ネットワークインターフェース関数
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

戻値

0 以上	チャンネル番号
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
E_PAR	ネットワークインターフェースの識別番号が不正 タスクの優先度が不正
E_ID※1	タスク ID、メールボックス ID が範囲外
E_NOID※1	タスク ID、メールボックス ID が不足
E_OBJ	ARP テーブル不足 タスク、メールボックスが既に生成されている
E_CTX	割り込みハンドラから発行

OS 資源の ID 自動割り当てを利用している場合、OS 資源の ID が不足するとエラーコード E_NOID が返ります。OS 資源の ID 自動割り当てを利用してない場合、OS 資源の ID が範囲外の値を指定するとエラーコード E_ID が返り、OS 資源が既に生成されていると E_OBJ が返ります。OS 資源の割り当て方法については、『第 3 章 コンフィグレーション ID の自動割り当て』を参照して下さい。

解説 NORTi Oceans TCP/IP プロトコルスタックに追加するネットワークインターフェースの初期化を行います。

ネットワークインターフェース制御ブロックはネットワークインターフェース毎に必要な情報が保存されます。ネットワークインターフェース毎に領域を確保し、アプリケーション側で管理してください。中身は変更しないでください。

戻り値として返るネットワーク I/F のチャンネル番号は、ネットワーク I/F のチャンネル番号です。

例
```
T_NIF nif_eth1;
ER MainTask(void)
{
    T_NIF_ADDR addr;
     :
    /* プロトコルスタックに"eth1"ネットワーク I/F を追加する */
    addr.hwaddr  = {0x00,0x00,0x12,0x34,0x56,0x78}; /* MAC アドレス     */
    addr.ipaddr  = {192,168,  0,100};               /* IP アドレス      */
    addr.gateway = {192,168,  0,  1};               /* ゲートウェイ     */
    addr.mask    = {255,255,255,  0};               /* サブネットマスク */
    ercd = tcp_nif_ini(&nif_eth1, "eth1", lan_nif_dev1, &addr);
     :
```

補足 プロトコルスタックを初期化する前は呼び出さないでください。

ネットワークインターフェース名は 7 文字以内で設定してください。8 文字以上の名称を指定した場合は、8 文字目以降を無視されます。

net_get_opt

機能 デフォルト・ネットワーク I/F 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 の設定値を取得します
SET_IP_TTL	TTL の設定値を取得します
SET_TCP_SYN_RCNT	SYN の再送回数の設定値を取得します
SET_TCP_DAT_RCNT	データの再送回数の設定値を取得します
SET_TCP_RTO_INI	TCP 再送時間の初期値の設定値を取得します
SET_TCP_RTO_MIN	TCP 再送タイムアウトの下位境界の設定値を取得します
SET_TCP_RTO_MAX	TCP 再送タイムアウトの上位境界の設定値を取得します
SET_TCP_KEEPALIVE_TMO	キープアライブタイムアウトの設定値を取得します
SET_TCP_KEEPALIVE_PRO	キープアライブプローブタイムアウトの設定値を取得します
SET_TCP_KEEPALIVE_SUC	キープアライブプローブインターバルの設定値を取得します
SET_TCP_DACK_TMO	遅延 ACK の遅延時間の設定値を取得します
SET_TCP_DUP_ACK	重複 ACK をエラーとして扱う回数の設定値を取得します
SET_TCP_TIMEWAIT_TMO	TIME-WAIT からのタイムアウト時間の設定値を取得します

各オプションで指定する optval の型は次のようになります

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

SET_TCP_DUP_ACK UB 型
SET_TCP_TIMEWAIT_TMO UH 型

戻値 E_OK 正常終了
E_PAR デフォルト・ネットワーク I/F が存在しない
パラメータエラー(バッファへのポインタ不正、バッファサイズ不正)
E_NOSPT 未サポートのパラメータが指定された

解説 設定されているデフォルト・ネットワーク I/F のオプションを取得します。

net_set_opt

機能 デフォルト・ネットワーク I/F 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_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
SET_TCP_TIMEWAIT_TMO	TIME-WAIT からのタイムアウト時間を設定します

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_TCP_SYN_RCNT	UB 型
SET_TCP_DAT_RCNT	UB 型
SET_TCP_RTO_INI	UH 型
SET_TCP_RTO_MIN	UH 型
SET_TCP_RTO_MAX	UW 型
SET_TCP_KEEPALIVE_TMO	UH 型
SET_TCP_KEEPALIVE_PRO	UH 型
SET_TCP_KEEPALIVE_SUC	UB 型
SET_TCP_DACK_TMO	UH 型
SET_TCP_DUP_ACK	UB 型
SET_TCP_TIMEWAIT_TMO	UH 型

戻値 E_OK 正常終了
E_PAR デフォルト・ネットワーク I/F が存在しない
パラメータエラー(バッファへのポインタ不正、バッファのサイズ不正、
設定値の 不正)
E_NOSPT 未サポートのパラメータが指定された

解説 デフォルト・ネットワーク I/F のオプションを設定します。デフォルト・ネットワーク I/F で通信が行なわれる前に使用してください。

net_chg_ipa

機能 デフォルト・ネットワーク I/F の IP アドレス変更

形式 ER net_chg_ipa(T_NIF_ADDR *addr, UB level);

addr ネットワーク I/F 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

E_OBJ デフォルト・ネットワーク I/F が存在しない

解説 デフォルト・ネットワーク I/F 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 I/F 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);
```

netif_get_opt

機能 ネットワーク I/F のオプション取得

形式 ER netif_get_opt(T_NIF *nif, INT optname, const VP optval, INT optlen);

nif	ネットワーク I/F 制御ブロック
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

戻値

E_OK	正常終了
E_PAR	ネットワーク I/F 制御ブロックが不正 パラメータエラー(バッファへのポインタ不正、バッファサイズ不正)
E_NOSPT	未サポートのパラメータが指定された

解説 net_get_opt システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 制御ブロックにより指定します。それ以外は net_get_opt と同じ仕様です。 ▪

netif_get_byname

機能 ネットワーク I/F のオプション取得(ネットワーク I/F 名指定)

形式 ER netif_get_opt(const char *name, INT optname, const VP optval, INT optlen);

name	ネットワーク I/F 名("eth1"など)
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

戻値

E_OK	正常終了
E_PAR	ネットワーク I/F が存在しない パラメータエラー(バッファへのポインタ不正、バッファサイズ不正)
E_NOSPT	未サポートのパラメータが指定された

解説 net_get_opt システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は net_get_opt と同じ仕様です。▪

netif_set_opt

機能 ネットワーク I/F のオプション設定

形式 ER netif_set_opt(T_NIF *nif, INT optname, const VP optval, INT optlen);

nif	ネットワーク I/F 制御ブロック
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

戻値

E_OK	正常終了
E_PAR	ネットワーク I/F 制御ブロックが不正 パラメータエラー(バッファへのポインタ不正、バッファのサイズ不正、設定値の 不正)
E_NOSPT	未サポートのパラメータが指定された

解説 net_set_opt システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 制御ブロックにより指定します。それ以外は net_set_opt と同じ仕様です。 ▪

netif_set_byname

機能 ネットワーク I/F のオプション設定(ネットワーク I/F 名指定)

形式 ER netif_set_byname(const char *name, INT optname, const VP optval, INT optlen);

name	ネットワーク I/F 名("eth1"など)
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

戻値

E_OK	正常終了
E_PAR	ネットワーク I/F が存在しない パラメータエラー(バッファへのポインタ不正、バッファのサイズ不正、 設定値の 不正)
E_NOSPT	未サポートのパラメータが指定された

解説 net_set_opt システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は net_set_opt と同じ仕様です。 ▪

netif_chg_ipa

機能 ネットワーク I/F の IP アドレス変更

形式 ER netif_chg_ipa(T_NIF *nif, T_NIF_ADDR *addr, UB level);

nif ネットワーク I/F 制御ブロック

addr ネットワーク I/F 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

E_OBJ ネットワーク I/F 制御ブロックが不正

解説 net_chg_ipa システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 制御ブロックにより指定します。それ以外は net_chg_ipa と同じ仕様です。 ▪

例

```
T_NIF *nif;
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 };
nif = getnif_from_name("eth0");
new_addr.ipaddr  = ipaddr;
new_addr.gateway = gateway;
new_addr.mask    = net_mask;
ercd = netif_chg_ipa(nif, &new_addr, 0);
```

netif_chg_byname

機能 ネットワーク I/F の IP アドレス変更(ネットワーク I/F 名指定)

形式 ER netif_chg_ipa(const char *name, T_NIF_ADDR *addr, UB level);

name ネットワーク I/F 名("eth1"など)

addr ネットワーク I/F 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

E_OBJ ネットワーク I/F が存在しない

解説 net_chg_ipa システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は net_chg_ipa と同じ仕様です。▪

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 = netif_chg_byname("eth0", &new_addr, 0);
```

getnif_default

機能 デフォルト・ネットワーク I/F 制御ブロックの取得

形式 T_NIF *getnif_default(void);

戻値 !NULL デフォルト・ネットワーク I/F 制御ブロック
NULL デフォルト・ネットワーク I/F が存在しない

解説 デフォルト・ネットワーク I/F のネットワーク I/F 制御ブロックを取得します。

getnif_from_name

機能 ネットワーク I/F 制御ブロックの取得(ネットワーク I/F 名指定)

形式 T_NIF *getnif_from_name(const char *name);
name ネットワーク I/F 名("eth1"など)

戻値 !NULL ネットワーク I/F 制御ブロック
NULL ネットワーク I/F が存在しない

解説 ネットワーク I/F 名からネットワーク I/F 制御ブロックを取得します。

getnif_from_ch

機能 ネットワーク I/F 制御ブロックの取得(チャネル番号指定)

形式 T_NIF *getnif_from_ch(int ch)
ch チャネル番号

戻値 !NULL ネットワーク I/F 制御ブロック
NULL ネットワーク I/F が存在しない

解説 ネットワーク I/F のチャネル番号からネットワーク I/F 制御ブロックを取得します。

getnif_addr

機能 ネットワーク I/F 制御ブロックの取得(IP アドレス指定)

形式 T_NIF *getnif_addr(UW ipaddr);
ipaddr IP アドレス

戻値 !NULL ネットワーク制御ブロック
NULL ネットワーク I/F が存在しない

解説 IP アドレスが一致するネットワーク I/F 制御ブロックを取得します。IP アドレスが一致するネットワーク I/F 制御ブロックが無い場合、同じネットワークアドレスを持つネットワーク I/F 制御ブロックを取得します。

arp_add_entry

機能 デフォルト・ネットワーク I/F の ARP テーブルに情報を追加

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 デフォルト・ネットワーク I/F が存在しない

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 IP アドレスに対応する MAC アドレスの情報を追加します。

ARP テーブルは定期的にリフレッシュされます。そのリフレッシュ動作により ARP_DYNAMIC を指定し登録した情報を削除されます。永続的な情報を登録する場合は ARP_STATIC を指定し登録してください。その情報はリフレッシュ対象外となり、arp_del_entry サービスコールか arp_del_byname サービスコールを発行しない限り、削除されることはありません

arp_add_byname

機能 ネットワーク I/F の ARP テーブルに情報を追加

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 arp_add_entry システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は arp_add_entry と同じ仕様です。

arp_del_entry

機能 デフォルト・ネットワーク I/F の ARP テーブルから情報を削除

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 デフォルト・ネットワーク I/F が存在しない
IP アドレスが不正か見つからない

解説 デフォルト・ネットワーク I/F の ARP テーブルから指定した IP アドレスを削除します。

arp_del_byname

機能 ネットワーク I/F の ARP テーブルから情報を削除

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 ネットワーク I/F が存在しない
IP アドレスが不正か見つからない

解説 arp_del_entry システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は arp_del_entry と同じ仕様です。

arp_snd_req

機能 デフォルト・ネットワーク I/F から ARP 要求パケット送信

形式 ER arp_snd_req(UW ipaddr);
ipaddr 問合せる IP アドレス

戻値 E_OK 正常終了
E_PAR IP アドレスが不正
E_OBJ デフォルト・ネットワーク I/F が存在しない
ゲートウェイが未設定
メモリブロックが確保できない

解説 デフォルト・ネットワーク I/F より IP アドレスの MAC アドレスを問い合わせる ARP 要求パケットを送信します。▪

arp_req_byname

機能 ネットワーク I/F から ARP 要求パケット送信

形式 ER arp_req_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 問合せる IP アドレス

戻値 E_OK 正常終了
E_PAR IP アドレスが不正
E_OBJ ネットワーク I/F が存在しない
ゲートウェイが未設定
メモリブロックが確保できない

解説 arp_snd_req システム関数はデフォルト・ネットワーク I/F が対象ですが、本システム関数は対象となるネットワーク I/F をネットワーク I/F 名により指定します。それ以外は arp_snd_req と同じ仕様です。▪

arp_def_cbk

機能　ARP コールバック関数の登録/削除

形式　ER arp_def_cbk(T_ARP_CB *arp_cb, ARP_CALLBACK callback);

arp_cb　ARP コールバック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

NULL を指定すると ARP コールバック関数の登録が削除される。

戻値　E_OK　　正常終了

解説　ARP モジュールのコールバックルーチンの登録や削除を行います。

コールバックルーチンが呼び出される条件は、次のような ARP パケットを受信した時です。

自身以外の IP アドレスを問合せる ARP 要求パケットを受信した時

送信元の IP アドレスと自身の IP アドレスが一致する ARP パケットを受信した時

ARP コールバック制御ブロックはチェーン構造で管理されています。ARP コールバック制御ブロックへのポインタが重複しない限り、複数のコールバック関数を登録できます。ARP コールバック制御ブロック領域は、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。

ARP コールバック制御ブロックは、ARP コールバック関数が登録されている間、領域を開放しないでください。ARP コールバック関数を削除する時は、ARP コールバック制御ブロックへのポインタにその領域を指定し、且つ、コールバック関数に NULL を指定して本 API を発行してください。

例

```
VP arp_callback(T_ARP *pkt);     /* ユーザー定義のコールバックルーチン */
T_ARP_CB arp_cb;                 /* ARP コールバック制御ブロック */

TASK MainTask(void)
{
    :
   /* ARP コールバック関数を登録します */
   ercd = arp_def_cbk(&arp_cb, arp_callback);       --- ①
    :
```

```
        /* ①で登録した ARP コールバック関数を削除する */
        ercd = arp_def_cbk(&arp_cb, NULL);
         :
    }
```

arp_rel_pkt

機能 ARP パケットのメモリブロック解放

形式 ER arp_rel_pkt(T_ARP *pkt);

pkt ARP 受信パケットへのポインタ

戻値 E_OK 正常終了

E_PAR パケットの制御ブロックの情報が壊れている
異なるメモリプールへの返却

E_ID 固定長メモリプール ID が範囲外

解説 ARP モジュールからコールバックで渡された ARP パケットのメモリブロックを解放します。

ARP コールバック関数で NULL を返す場合は、必ず ARP コールバック関数内にて本サービスコールを使用して ARP パケットのメモリブロックを解放してください。

tcp_ref_cep

機能 TCP 通信端点に設定された IP アドレス、ポート番号と状態を取得

形式 ER tcp_ref_cep(ID cepid, T_IPV4EP *p_myaddr);

cepid TCP 通信端点 ID

p_myaddr IP アドレス/ポート番号構造体へのポインタ

```
typedef struct t_ipv4ep {
        UW ipaddr;      IP アドレス
        UH portno;      ポート番号
} T_IPV4EP;
```

戻値

CEP_UNUSED	未使用
CEP_WAIT_P_OPEN	受動オープン待ち
CEP_WAIT_A_OPEN	能動オープン待ち
CEP_CONNECTING	接続
CEP_TERMINATED	送信終了
CEP_DISCONNECTED	接続切断
CEP_CLOSING	クローズ中
E_ID	不正 ID 番号、または、非タスクコンテキストで実行
E_NOEXS	TCP 通信端点が未生成
E_LNK	ネットワークインタフェースが未初期化

解説 TCP 通信端点に設定された IP アドレス、ポート番号と状態を取得します。TCP 通信端点の状態につきましては、「1.5 用語」の「TCP 通信端点の状態」を参照ください。p_myaddr には、端点に割り当てられている IP アドレスとポート番号が入ります。

索引

A

arp_add_byname 93
arp_add_entry 92
ARP_CACHE_TOUT 20
arp_def_cbk 96
arp_del_byname 94
arp_del_entry 94
ARP_FLUSH_TOUT 20
ARP_NIF_CNT 19
arp_rel_pkt 98
arp_req_byname 95
ARP_RET_INTVAL 19
ARP_RETRY 19
arp_snd_req 95
ARP_TABLE_CNT 19
ARP コールバック関数の登録/削除 96
ARP テーブル 9
ARP のタイムアウト 9
ARP パケットのメモリブロック解放 98
ARP モジュール 9
ARP モジュールのコールバック 73
ARP 応答 9
ARP 問い合わせ 9
ascii_to_ipaddr 26

B

byte4_to_long 26

E

Echo 処理 10
ETH_QCNT 19
Ethernet パケット用メモリプール 6

G

getnif_addr 91
getnif_default 90
getnif_from_ch 91
getnif_from_name 90

H

htonl 22, 25
htons 22, 25

I

ICMP 10
icmp_def_cbk 10
icmp_snd_dat 12
ICMP の受信コールバック 11
ICMP パケット送信 12
ICMP モジュール 10
ICMP 受信コールバック関数の登録 10
ID の自動割り当て 21
IGMPV1_PRESENT_TMO 18, 20
IGMPv1 ルータ 18
IP_ADD_MEMBERSHIP 18
IP_DEF_TTL 20
IP_DROP_MEMBERSHIP 18
ipaddr_to_ascii 26
IP モジュール 8

L

long_to_byte4 26

M

MAC アドレス 21
mainercd 22
MULTICAST_GROUP_CNT 20

N

net_chg_ipa 83
net_get_opt 79
net_set_opt 81
netif_chg_byname 89
netif_chg_ipa 88
netif_get_byname 85
netif_get_opt 84
netif_set_byname 87
netif_set_opt 86
nocnelan.c 2
nocnet.h 2
nocnetc.h 2
nocnets.h 2
nocnitod.h 2
NORTi Oceans ライブラリ 2
ntohl 22, 25
ntohs 22, 25

P

PRI_IP_RCV_TSK 19
PRI_IP_SND_TSK 19

S

SSZ_IP_RCV_TSK 19
SSZ_IP_SND_TSK 19
subercd 22

T

T_IPV4EP 23
T_TCP_CCEP 23
T_TCP_CREP 23
T_UDP_CCEP 23
TCP 14
TCP/IP タイマ(周期ハンドラ) 6
TCP/IP 受信タスク 6, 8
TCP/IP 送信タスク 6, 8
tcp_acp_cep 35
tcp_can_cep 52
TCP_CEPID_MAX 19
tcp_cls_cep 40
tcp_con_cep 37
tcp_cre_cep 31
tcp_cre_rep 28
TCP_DACK_TMO 20
TCP_DATA_RETRY 16, 20
tcp_del_cep 34
tcp_del_rep 30
TCP_DUP_ACK 17, 20
tcp_get_buf 46
tcp_get_opt 56
tcp_ini 76
TCP_KTIME_INI 15, 20
TCP_KTIME_PRO 15, 20
TCP_KTIME_SUC 15, 20
TCP_MBXID_TOP 19
TCP_MPFID_TOP 19
tcp_nif_ini 77
TCP_PORT_EPHEMERAL 20
tcp_rcv_buf 49
tcp_rcv_dat 44
tcp_ref_cep 99
tcp_rel_buf 51
TCP_REPID_MAX 19
TCP_RTO_INI 16, 20

TCP_RTO_LBOUND 16, 17, 20
TCP_RTO_UBOUND 16, 17, 20
tcp_set_opt 53
tcp_sht_cep 39
tcp_snd_buf 48
tcp_snd_dat 42
TCP_SYN_RETRY 16, 20
TCP_TIMEWAIT_TMO 20
TCP_TSKID_TOP 19
tcp_vcre_cep 33
tcp_vcre_rep 29
TCP サービスコール 27
TCP 受付口の削除 30
TCP 受付口の生成 28
TCP 受付口の生成 (ID 自動割り当て) 29
TCP 通信端点オプションの参照 56
TCP 通信端点オプションの設定 53
TCP 通信端点に設定された IP アドレス、ポート番号と状態を取得 99
TCP 通信端点の削除 34
TCP 通信端点の状態 3
TCP 通信端点の生成 31
TCP 通信端点の生成(ID 自動割り当て) 33
TEV_TCP_TMO_KTIME 15

U

UDP 13
udp_can_cep 66
UDP_CEPID_MAX 19
udp_cre_cep 60
udp_get_opt 70
UDP_QCNT 19
udp_rcv_dat 64
udp_set_opt 67
udp_snd_dat 62
udp_vcre_cep 61
UDP サービスコール 59
UDP パケットの受信 64
UDP パケットの受信通知 72
UDP パケットの送信 62
UDP パケット受信 13
UDP パケット送信 13
UDP ヘッダ用メモリプール 6
UDP 受信キュー 7, 13
UDP 送受信のキャンセル 66
UDP 通信端点オプションの参照 70
UDP 通信端点オプションの設定 67
UDP 通信端点の生成 60
UDP 通信端点の生成(ID 自動割り当て) 61

え

エラーコード取り出し 22

き

キープアライブ 15
キャンセル 7

こ

高速再転送/高速リカバリ 17
コールバック 4, 71
コンフィグレーション 19

さ

サービスコール 4
再送信 16
サブネットマスク 21

し

受信したデータの入ったバッファの取得 49
受信用バッファの解放 51
省コピーAPI 4
使用中 3

す

ストリーム型通信方式 16

せ

生成済み 3
セグメント 4
接続 3
接続切断 3
接続要求(能動オープン) 37
接続要求待ち(受動オープン) 35
ゼロ・ウィンドウ・プローブ 17

そ

送受信のキャンセル 52
送信終了 3
送信パケットキュー 7, 8
送信用バッファの取得 46
送信リトライキュー 7

た

タイムアウト 4, 7
端点、受付口の ID 自動割り当て 21

つ

通信端点 3
通信端点のクローズ 40

て

データグラム 4
データ送信の終了 39
データの受信 44
データの送信 42
デフォルト・ネットワーク I/F から ARP 要求パケット送信 95
デフォルト・ネットワーク I/F 制御ブロックの取得 90
デフォルト・ネットワーク I/F の ARP テーブルから情報を削除 94
デフォルト・ネットワーク I/F の ARP テーブルに情報を追加 92
デフォルト・ネットワーク I/F の IP アドレス変更 83
デフォルト・ネットワーク I/F のオプション取得 79
デフォルト・ネットワーク I/F のオプション設定 81
デフォルトゲートウェイ 21

と

独自システム関数 75

ね

ネットワーク I/F から ARP 要求パケット送信 95
ネットワーク I/F の ARP テーブルから情報を削除 94
ネットワーク I/F の ARP テーブルに情報を追加. 93
ネットワーク I/F の IP アドレス変更 88
ネットワーク I/F の IP アドレス変更(ネットワーク I/F 名指定) 89
ネットワーク I/F のオプション取得 84
ネットワーク I/F のオプション設定 86
ネットワーク I/F のオプション取得(ネットワーク I/F 名指定) 85
ネットワーク I/F のオプション設定(ネットワーク I/F 名指定) 87
ネットワーク I/F 制御ブロックの取得(IP アドレス指定) 91
ネットワーク I/F 制御ブロックの取得(チャネル番号指定) 91
ネットワーク I/F 制御ブロックの取得(ネットワーク I/F 名指定) 90

ネットワークインターフェースの初期化 77

の

ノンブロッキング 4
ノンブロッキングコールの完了通知 71

は

バイトオーダー変換 22
パケット 4
パケット受信 8
パケット送信 8
バッファ内のデータの送信 48

ふ

ファイル構成 2
フレーム 4
プロトコルスタック 5
プロトコルスタックの初期化 76
プロトコルスタックのメールボックス 7
プロトコルスタックのメモリプール 6
プロトコル制御タスク 6

ま

マルチキャストグループ 18
マルチキャストパケット送受信 18

み

未使用 3
未生成 3
未接続 3

め

メイルボックス 13

ゆ

ユーティリティ・マクロ 25
ユーティリティ関数 26

ろ

ローカル IP アドレス 21

NORTi Oceans ユーザーズガイド

TCP/IP 編

2012 年　3 月　初版

株式会社ミスポ http://www.mispo.co.jp/
〒213-0002　川崎市高津区二子 5-1-1
TEL 044-829-3381　FAX 044-829-3382
一般的なお問い合わせ sales@mispo.co.jp
技術サポートご依頼　norti@mispo.co.jp