ＩＴＲＯＮ ＴＣＰ／ＩＰ　ＡＰＩ仕様準拠
プロトコルスタック

ネットワーク編

ユ ー ザ ー ズ ガ イ ド

MiSPO
株式会社ミスポ

ＮＯＲＴｉ　ユーザーズガイド
ネットワーク編

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

株式会社ミスポ
〒 213-0012 川崎市高津区坂戸 3-2-1
かながわサイエンスパーク 西 300-G

第２版で訂正された項目

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
3	LAN コントローラのデバイスドライバを除く ... → LAN コントローラのデバイスドライバとサンプルを除く ...
21	タスクの優先度に関する補足事項を追記
30	tcp_cre_rep の戻値 E_PAR を追加
33	tcp_cre_cep の補足にバッファサイズの制限を追記
37	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
40	tcp_snd_dat の戻値 E_RLWAI の説明を追記
41	tcp_rcv_dat の戻値 E_PAR を追加
43	tcp_snd_buf の戻値 E_PAR を追加
44	tcp_rcv_buf の戻値 E_WBLK を追加
55	udp_snd_dat の戻値 E_PAR を追加
55	udp_snd_dat の戻値 E_RLWAI の説明を追記
55	udp_snd_dat の補足に送信パケットのポインタは ... を追記
56	udp_rcv_dat の戻値 E_PAR を追加
57	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

第３版で訂正された項目

ページ	内容
9	NORTi Network 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
36	tcp_acp_cep の補足にノンブロッキング（tmout =TMO_NBLK ）で、... を追記
39	tcp_cls_cep に補足を追加
40	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
55	udp_snd_dat の補足で奇数番地を偶数番地に修正
56	udp_rcv_dat の補足で奇数番地を偶数番地に修正
56	udp_rcv_dat の解説の一部を変更
57	udp_rcv_dat の補足にノンブロッキング（tmout =TMO_NBLK ）で、... を追記
59	udp_set_opt を記載

第４版で訂正された項目

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
30	tcp_cre_rep の解説を一部変更
33	TCP 送受信バッファについての補足を追加
37	IPV4_ADDRANY,TCP_PORTANY の説明を変更
37	tcp_con_cep の戻り値の説明を変更
37	tcp_con_cep の補足を追加
52	IPV4_ADDRANY,UDP_PORTANY の説明を変更
55	udp_snd_dat の戻り値 E_PAR の内容を一部変更
61	コールバックの機能コードを追加

第５版（本版）で訂正された項目

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
26	メインエラーコードを修正
27	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
55	udp_snd_dat の E_PAR の説明を修正
56	udp_rcv_dat の E_PAR の説明を修正

目　次

（余白）

第１章 導入

１．１ はじめに

NORTi Version4 は、μITRON4.0 仕様 OS に、ITRON TCP/IP API 仕様を実現するプロトコルスタック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。OS 部分については、「NORTi version4 ユーザーズガイド カーネル編」を参照してください。

１．２ 特長

組み込み用途に適した ITRON TCP/IP API 仕様に準拠しています。

プロトコルスタックの全ソースコードが付属しています（評価版を除く）。

コンパイラ等と同様に、開発者に対するユーザーライセンス制をとっており、組み込みロイヤリティが無料です。

１．３ 制限事項

TCP, UDP, ICMP, IP, ARP のプロトコルまでしか実装されていません。上位のプロトコルや他のプロトコルは、ユーザーにて追加する必要があります。

μITRON4.0 仕様 OS NORTi Version4 上で動作するよう特化しているため、他の OS へ移植することは困難です。

製品にはいくつかのLANコントローラのデバイスドライバがサンプルとして添付されています。収録されていないドライバがありましたら、弊社までお問い合わせください。新たに対応している可能性があります。

１．４ ファイル構成

NORTi Network は次のファイルから構成されます。

nonet.h	**NORTi Network 標準ヘッダ**
nonetc.h	**NORTi Network コンフィグレーションヘッダ**
nonets.h	**NORTi Network 内部定義ヘッダ**
n4nxxx.lib	**NORTi Network ライブラリ**
n4nxxx.mak	**NORTi Network ライブラリ生成用 makefile**
nonet.c	**NORTi Network サービスコールのソース**
nontcp1.c	**NORTi Network TCP モジュールのソース 1/2**
nontcp2.c	**NORTi Network TCP モジュールのソース 2/2**
noneudp.c	**NORTi Network UDP モジュールのソース**
nonicmp.c	**NORTi Network ICMP モジュールのソース**
nonearp.c	**NORTi Network ARP モジュールのソース**
nonetip.c	**NORTi Network IP モジュールのソース**
nonigmp.c	**NORTi Network IGMP モジュールのソース**
nonelan.c	**NORTi Network LAN コントローラ上位部分のソース**
mb86964.c	**富士通 MB86964 用サンプルドライバのソース**
mb86964.h	**富士通 MB86964 用サンプルドライバのヘッダ**
dp8390.c	**ナショナルセミコンダクタ DP8390x 用サンプルドライバソース**
dp8390.h	**ナショナルセミコンダクタ DP8390x 用サンプルドライバヘッダ**

NORTi Network にはこれらのファイル以外にも有用なサンプルファイルがいくつか含まれています。詳細は tcpip.txt を参照して下さい。

nonet.h

nonet.h は、NORTi Network 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 Network 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

nonetc.h は、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、NORTi Network の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。本ヘッダをインクルードする前に、ID 先頭値等の #define を記述することで、NORTi Network のコンフィグレーションを行うことができます。

nonets.h

nonets.h は、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、NORTi Network 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。nonets.h は、nonetc.h および、NORTi Network を構成する各ソースファイルからインクルードされています。

n4nxxx.lib

n4nxxx.lib は、ユーザープログラムとリンクしてください。LAN コントローラのデバイスドライバとサンプルを除く、NORTi Network のプロトコルスタック機能がすべて結合されています。xxx は、対応 MPU 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。

nonet.c

nonet.c は、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、以下のソースから共通に呼び出される関数や、「その他」の関数が記述されています。

nontcp?.c

nontcp?.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、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。

noneudp.c

noneudp.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、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。

nonearp.c

nonearp.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、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。

nonicmp.c

nonicmp.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、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。だたし、Echo 処理するだけの簡単なものですので、必要ならユーザーにて機能拡張してください。

nonetip.c

nonetip.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースは、プロトコルスタックの最も基本的な部分で、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。

nonigmp.c

nonigmp.c も、コンパイルされて n4nxxx.lib へ結合されています。本ソースには、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。

mb86964.c

デバイスドライバのサンプルソースです。富士通製 LAN コントローラ MB86964 用です。ユーザーのターゲットに合わせてカスタマイズが必要です。

mb86964.h

サンプルドライバのヘッダです。mb86964.c からインクルードされています。MB86964 のレジスタのアドレスやビットの定義が記述されています。

dp8390.c

デバイスドライバのサンプルソースです。ナショナルセミコンダクタ製 LAN コントローラ DP8390x 用

dp8390.h

デバイスドライバのヘッダです。dp8390.c からインクルードされています。

DP8390x のレジスタのアドレスやビットの定義が記述されています。

１．５ 用語

通信端点

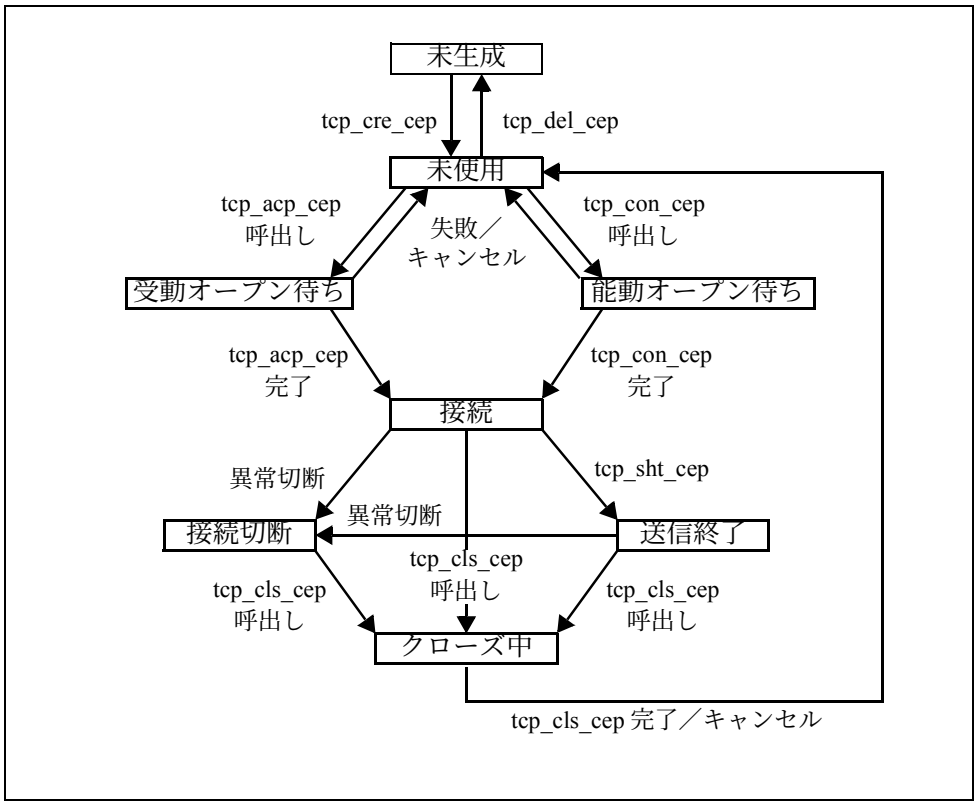
TCP のための通信端点として、TCP 受付口と、TCP 通信端点の２種類が用意されています。TCP 受付口は、相手側からの接続要求を待ち受けるために使用します。コネクションが確立した後のデータ送受信は、TCP 通信端点で行われます。

UDPのための通信端点としては､UDP信端点が用意されています｡TCP受付口(TCP Reception Point）は、サービスコールやパラメータの名称で、rep と略されています。TCP 通信端点（TCP Communication EndPoint）は、cep と略されています。

通信端点は、ITRON 仕様 OS のオブジェクトに相当します。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、サービスコールと言います。サービスコールは、ITRON 仕様 OS のシステムコールに相当します。

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。タイムアウト指定では、指定時間を経過しても処理が完了しない場合に、処理が中止されます。

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終了しますが、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

コールバックとは、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、プトロコルスタック側から呼び出すことです。

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、緊急データ受信や、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、イーサネットはフレーム）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。イーサネットパケットとイーサネット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 Network のプロトコルスタックの構成と動作について、説明しています。

NORTi Network のプロトコルスタックは、μITRON3.0 および 4.0 仕様のメモリプールとメイルボックスの機能を最大限に活用して実現されています。

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 Network 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 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

プロトコル制御タスク

NORTi Network のプロトコルスタックは次のタスクから構成されています。

IP 受信タスク　　全体で１個
IP 送信タスク　　全体で１個

TCP, UDP, ICMP, ARP の全てが、IP 受信タスク /IP 送信タスクで制御されています。

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、全体で１個だけ存在します。

UDP ヘッダ用メモリプールは、送受信する UDP パケットのヘッダ用の領域を提供します。このメモリプールは、UDP 通信端点毎に存在します。

各メモリプールから獲得できるサイズには、イーサネットヘッダと IP ヘッダの分まで含めています。これにより TCP/IP プロトコルスタックの各モジュール間でデータを転送する際に、新たな領域の獲得やコピーの発生を防いでいます。

TCP パケットはメモリプールを使用しません。TCP はユーザーが指定したバッファ領域に直接コピーされます。

プロトコルスタックのメイルボックス

アプリケーションのタスクや各プロトコル制御タスクの間の通信には、メイルボックスを使用しています。メイルボックスでは、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。

受信したパケットのキューイング、あるいは、送信するパケットのキューイングだけでなく、送信を保留したり再送したりするための保存場所にも、メイルボックスの機能を利用しています。

プロトコルスタックで使用するメイルボックス（キュー）には、次のものがあります。これらの機能については、それぞれのモジュールで説明します。

送信パケットキュー　全体で１個
UDP 受信キュー　　UDP 通信端点毎

TCP パケットはメイルボックスを使用しません。TCP はユーザーが指定したバッファ領域に直接コピーされます。

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 Network 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。

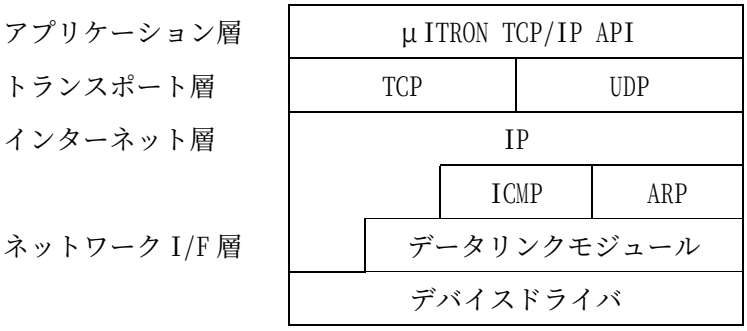
タイムアウトなし（tmout = TMO_FEVR ）
タイムアウトあり（tmout = 1 ～ 0x7fffffff)
ポーリング（tmout = TMO_POL)
ノンブロッキング（tmout = TMO_NBLK)

接続先がクラッシュした場合、タイムアウトなし（tmout = TMO_FEVR ）を使用すると関数によっては永久に関数から戻らない可能性があります。可能な限りタイムアウトありをパラメータに使用して下さい。

μITRON の標準のメイルボックスでは、キューイングしてある途中のパケットだけを取り出すことができず、キャンセル処理に不向きです。そこで、NORTi Network では、メイルボックスを独自に機能拡張しています。

階層構造

NORTi Network は以下のようなレイヤーで構成されています。



２．２ データリンクモジュール

構成

データリンクモジュールは 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インターネット層との間に入るモジュールです。送受信待ち関連の関数以外はインターネット層からはデバイスドライバを直接コールせずに、データリンクモジュールを中継します。ネットワーク I/F を Ethernet 以外のデバイスで使用する場合はデータリンクモジュールとデバイスドライバのインプリメントが必要です。イーサネットヘッダのイーサネットアドレス（MAC アドレス）の設定を行うのは、本来、リンク層の仕事ですが、NORTi Network の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 モジュールが行っています。また、受信したパケット全体を一度に読み出すのではなく、ヘッダ部とデータ部とに分けてあります。

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。

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 書き込むヘッダデータ
hlen ヘッダデータのバイト数
data 書き込むデータ
dlen データのバイト数

[戻 値] TRUE 正常終了
FALSE エラー

デバイスのバッファメモリへ、ヘッダとデータを書き込みます。

受信パケット長を得る

[形 式] UH lan_received_len(void)

[戻 値] 受信した Ethernet フレームのバイト数

Ethernet ドライバで受信しているフレーム数を返します。

受信パケット破棄

[形 式] void lan_ignore_pkt(void)

[戻 値] なし

Ethernet ドライバで受信している全てのフレーム数を破棄します。

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

LAN 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 Network のイーサネット用 LAN コントローラ・デバイスドライバは、タスクを使用しません。デバイスドライバは、データリンクモジュールから呼び出される関数群と、IP モジュールの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、上位プロトコルのことを知っている必要がありますが、できるだけ、メモリを使わない実装のための工夫です。

デバイスドライバ関数群はリエントラントではありません。

lan_wai_snd、lan_wai_rcv のみ IP 層から直接呼ばれます。それ以外の関数は全てデータリンクモジュールから呼ばれます。

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）
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

TCP/IP 初期化関数 tcpini から呼び出され、デバイスのレジスタ初期設定や、割込みハンドラ定義が行われます。

割込みハンドラ

[形 式] INTHDR lan_int(void)

受信割込みの場合、受信を lan_wai_rcv 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、送信を lan_wai_snd で待っている IP 送信タスクを起床します。

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

受信したパケットがない場合は、IP 受信タスクが、lan_wai_rcv 関数で、受信割込み待ち状態に入ります。デバイスの受信割込みで、この IP 受信タスクの待ちは解除されます。

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

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、IP 送信タスクが、lan_wai_snd 関数で、送信割込み待ち状態に入ります。デバイスの送信完了割込みで、この IP 送信タスクの待ちは解除されます。

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

デバイスドライバの lan_get_len 関数は、lan_received_len から呼び出されます。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

デバイスドライバの lan_get_pkt 関数は、lan_read_pkt から呼び出されます。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)。

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

lan_get_pkt 関数が繰り返し呼ばれた後の最後に、lan_get_end 関数が呼び出されます。

受信パケット読み飛ばし

[形 式] ER lan_skp_pkt(int len)

[戻 値] エラーコード

不要な受信パケットのため、lan_get_pkt でデータを読み出さない場合、この lan_skp_pkt 関数が代わりに呼び出されます。

送信パケット長を設定

[形 式] ER lan_set_len(int len)

[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、lan_write_pkt 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

デバイスドライバの lan_put_pkt 関数は、lan_write_pkt 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）。

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、続けて、この lan_put_dmy 関数が呼び出されます。

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。

タイムアウトについて

デバイスドライバのレベルでは、現状、タイムアウトは監視していません。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、すでにデバイスのバッファメモリに書き込まれた送信パケットは送信されてしまいます。

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イ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、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、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、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、イーサネットヘッダを設定して、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 Network 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。TCP や UDP 受信でのタイムアウトエラーで、ARP テーブルの該当エントリはクリアされます。

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

NORTi Network では、ping コマンドに応答する Echo を除き、ICMP を活用していません。本プロトコルスタックから Echo 応答以外の ICMP パケットを発信することはありません。

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。

ECHO 以外のパケットの受信を行うためには icmp_def_cbk でユーザー側で作成したコールバック関数を登録することで受信ができます。送信は icmp_snd_dat を使用します。

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)**

戻　値　**E_OK　正常終了**
負の値　異常終了

解　説　ECHO 以外の 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_IP *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_ip {
    struct t_ip *next;      /* Message Pointer for NORTi Mailbox */
    T_CTL_HEADER ctl;       /* Header for Internal Control */
    T_ETH_HEADER eth;       /* Ethernet Header */
```

```
    T_IP_HEADER ip;         /* IP Header (without option data) */
    B data[1];              /* Data (Variable Size) */
} T_IP;

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。
ECHO 以外の 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 Network 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イ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。

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 Network 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。

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 で用いるメイルボックスの先頭 ID
#define TCP_MPFID_TOP      ?    TCP/IP で用いる固定長メモリプール先頭 ID

#define ARP_TABLE_CNT     (8)   ARP テーブルのエントリ数

#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  (512)   IP 送信タスクのスタックサイズ
#define SSZ_IP_RCV_TSK  (512)   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
#define UDP_QCNT          (2)   UDP パケットの最大キューイング数

#include "nonetc.h"
```

ID の自動割り当て

NORTi Network 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 Network 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 で用いるメイルボックスの先頭 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 Network 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 Network 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 Network 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 Network 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		バッファオーバフロー

() 内は μITRON3.0 仕様の値

第5章 ユーティリティ

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);**
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);**
hl　　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);**
hl　　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　　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(B *s);**
B *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(B *s, UW ipaddr);**
B *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 lP 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, 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_IPV4EP 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_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のみを待ち受けます。

TCP 受付口には、tcp_acp_cep サービスコールによる接続待ち処理をキューイングすることができます。キューイングできる個数に制限はありません。

tcp_acp_cep サービスコールによる接続待ちが行われるまで、TCP 受付口で、通信端点の接続要求をキューイングしておくことができます。キューイングできる個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接続要求は、破棄されます。

補　足　自分側の IP アドレスに IPV4_ADDRANY（= 0）を指定した場合に、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はサポートしていません。

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(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

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 Network 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, 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。

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2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。

送受信バッファのサイズが小さすぎると通信の効率が悪くなります。最適な通信を行うためにはバッファサイズは 2920 ～ 8760 程度を下限に設定すると有効です。

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(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 Network 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 *dstaddr, TMO tmout);
```
cepid　TCP 通信端点 ID
repid　TCP 受付口
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

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。

*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。

ポーリング（tmout = TMO_POL）で本サービスコールを発行した場合、既に接続要求を受信しているならば、それを受け付けて、接続が完了するまで待ち状態となります。接続要求を受信していない場合は、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 (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 *myaddr, T_IPV4EP *dstaddr, TMOtmout);**

cepid　TCP 通信端点 ID
myaddr　自分側 IP アドレス / ポート番号構造体へのポインタ
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、または接続要求が拒否された
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	接続失敗

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、*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。

補　足　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_OBJ 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可能性があります。

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**

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, 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 *buf, TMO tmout);**

cepid	**TCP 通信端点 ID**
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 *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 *buf, TMO tmout);**

cepid	**TCP 通信端点 ID**
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 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 *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**

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。

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。

現バージョンでは、サポートされていません。

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, VP optval, INT optlen);**

解　説　現バージョンでは、サポートされていません。

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);**

解　説　現バージョンでは、サポートされていません。

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, 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 {
    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

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。

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　**E_OK**　**正常終了**

E_ID　**UDP 通信端点 ID が不足**

E_OBJ　**ポート番号既使用**

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 Network 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, T_IPV4EP *dstaddr, VP data, INT len, TMOtmout);
```
cepid　UDP 通信端点 ID
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	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

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、*dstaddrで指定された相手へ送信します。NORTi Network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_cre_cep 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 *dstaddr, VP data, INT len, TMOtmout);

cepid	UDP 通信端点 ID
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_QOVR	キューイングオーバフロー
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_BOVR	バッファオーバフロー
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

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。

*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。

パケット未受信の場合は、パケットを受信するまでの間、このサービスコール発行元のタスクが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、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。キューイングできる送信パケット数の上限は、udp_cre_cep　で指定した値で決まります。上限を越えて

udp_rcv_dat を発行すると E_QOVR エラーとなります。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。また、

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 20byte 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 (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。

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, VP optval, INT optlen);**

cepid　**UDP 通信端点 ID**

optname　**値を設定する UDP 通信端点オプション**

optval　**オプションに指定する値を持つバッファへのポインタ**

optlen　**optval のバッファサイズ**

解　説　**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

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);**

解　説　現バージョンでは、未サポートです。

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、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_RCV_DAT (-0x223) udp_rcv_dat 完了通知
TFN_UDP_SND_DAT (-0x224) udp_snd_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 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初期化

機　能　プロトコルスタック初期化

形　式　`ER tcp_ini();`

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　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呼び出してください。