TCP/IPマネージャ　Ver3.0
リファレンスマニュアル

ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社

ご注意

1. 本製品(ソフトウエア製品及びその関連ソフトウエア製品を含む。以下、同じ。)の使用に際しては、「外国為替及び外国貿易法」等、技術輸出に関する日本及び関連諸国の関係法規の遵守が必要となります。
2. 弊社は、本製品の使用に際しては、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に関し、別途、個別の契約書等(マニュアルの記載を含む。以下、同じ。)にて弊社による明示的な許諾がある場合を除き、その保証または実施権の許諾を行うものではありません。また本製品を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。
3. 本製品およびその仕様、またはマニュアルに記載されている事柄については、将来、事前の予告なしに変更することがありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書(マニュアルを含む)をご確認ください。
4. 本製品の使用(マニュアル記載事項に基づくものも含む)により直接または間接に生ずるいかなる損害についても、弊社は一切の責任を負いません。また、本製品の配布に使用される搭載機器や媒体が原因の損害に対しましても、弊社は一切の責任を負いません。
5. 本製品を、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。お客様の用途がこれに該当するかどうか疑問のある場合には、事前に弊社営業担当迄ご相談をお願い致します。
6. 本製品を使用してお客様のシステム製品を設計される際には、通常予測される故障発生率、故障モードをご考慮の上、本製品の動作が原因での事故、その他の拡大損害を生じないようにフェールセーフ等の充分なシステム上の対策を講じて頂きますようお願い致します。
7. 本製品およびマニュアルの著作権は弊社が所有しております。お客様は、弊社から提供された本製品を、別途、個別の契約書等にて定める場合を除き、いかなる場合においても全体的または部分的に複写・解析・改変することはできないものとします。
8. お客様は、別途、個別の契約書等にて定める場合を除き、本製品のマニュアルの一部または全部を無断で使用、複製することはできません。
9. 弊社は、本製品を1台のコンピュータで使用する権利をお客様に対してのみ許諾します。よって、本製品を第三者へ譲渡、貸与、賃借することは許諾しないものとします。但し、別途、個別の契約書等にて定められる場合はその条件に従います。
10. 本製品をはじめ弊社製品およびその関連製品についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

はじめに

このマニュアルは、TCP/IPマネージャVer3.0について説明します。
TCP/IPマネージャは、ITRON TCP/IP API仕様に準拠したTCP/IPプロトコルスタックです。
TCP/IPマネージャは、ドライバプログラムを介してイーサネットコントローラ等の各種ハードウェアを制御し、ネットワークにアクセスします。
このリファレンスマニュアルではTCP/IPマネージャのサービスコールとその使い方および関連事項を説明します。ドライバプログラムについては関連マニュアルを参照してください。

目次

図表目次

1. 概要

1.1 機能

TCP/IPマネージャは、イーサネットコントローラ等を介して、ネットワークにアクセスします。
TCP/IPマネージャの基本機能は次のとおりです。

（1） TCP/IPマネージャはネットワークを経由してネットワークサーバやネットワーククライアントにアクセスします。

（2） マルチタスクプログラムでネットワークを管理することができます。
サービスコールを複数のタスクから同時に発行することができます。

（3） サービスコールを使い、TCPとUDPを管理することができます。
- 最大 128 個の通信端点の生成、削除
- TCP のオープン、クローズ
- データの送信、受信

（4） 複数のドライバとIPアドレスを管理できます。
- 最大 4 つまでのドライバを管理できます。
- 最大 4 つまでの IP アドレスを管理できます。
- 1 つのドライバに対し最大 4 つまでの IP アドレスを登録できます。

（5） MIB Ⅱの以下のグループをサポートしています（ソースコード提供版のみ[*1]）。
- interfaces グループをサポートしています。
- ip グループをサポートしています。
- icmp グループをサポートしています。
- tcp グループをサポートしています。
- udp グループをサポートしています。

（6） IPマルチキャストパケットの送受信が可能です。
- udp パケットの送受信ができます。
- icmp エコー要求／応答ができます。

本バージョンのTCP/IPマネージャには、次の機能的な制限があります。
- **IGMP はサポートしていません。**
- **自 IP アドレスに対するコネクションはできません。**
- **ループバック用の IP アドレス（127.0.0.1）はサポートしていません。**
- **ルーティングプロトコルはサポートしていません。**
- **デフォルト状態での緊急（帯域外）データの送受信動作は独自仕様です。**
- **TCP の MSS の実効値の上限は MTU - 40 バイトです。**

*1：オブジェクト版の製品で本機能が必要な場合はカスタム対応となります。

1.2 関連するドライバ

TCP/IPマネージャはドライバモジュールを介して通信LSIを制御します。アプリケーション（ユーザプログラム）からはTCP/IPマネージャが使用中のドライバモジュールを同時に使うことはできません。

また、このドライバモジュールはユーザが作成しなければなりません。ドライバモジュールの例としてサンプル提供のEthernet用ドライバモジュールがあります。サンプル提供のEthernet用ドライバモジュールは、ドライバ制御モジュールとEthernetドライバの２つのプログラムから構成されています。

1.3　構成

TCP/IP マネージャを使う時の構成をイーサネットの場合を例に図 1-1 を用いて説明します。

ユーザプログラムは上位プロトコルを管理しながら TCP/IP マネージャにサービスコールを発行し、データの送信/受信を行います。TCP/IP マネージャはドライバ制御モジュールを介して Ethernet ドライバにサービスコールを発行し、データの送信/受信を行います。Ethernet ドライバはイーサネットコントローラを介してデータの送信/受信を行います。

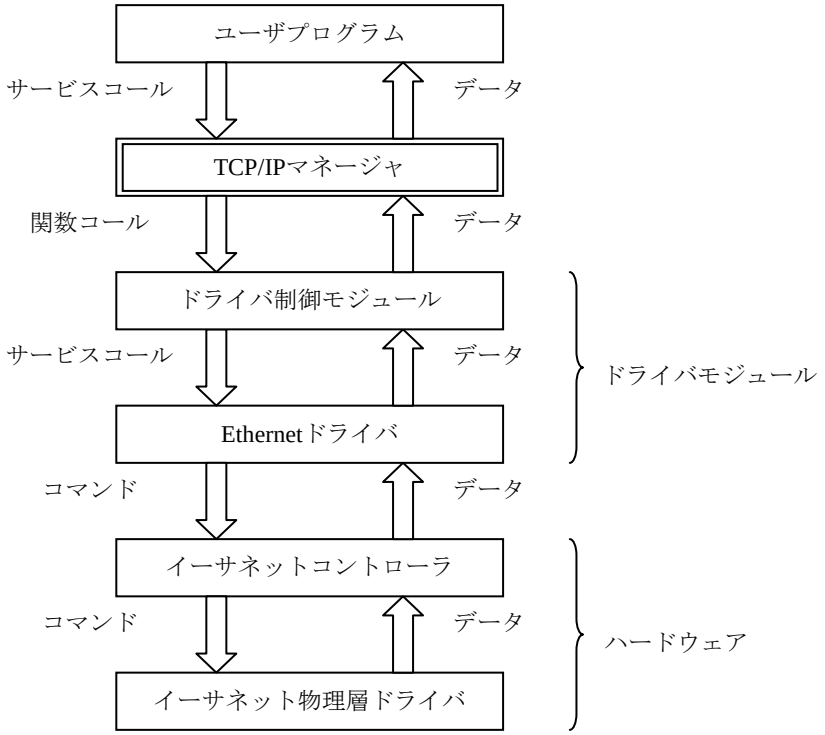


図 1-1　TCP/IPマネージャの構成

2. TCP/IP マネージャの使い方

2.1 TCP/IP マネージャの機能

TCP/IP マネージャは、イーサネット等を経由してネットワークサーバやネットワーククライアントと通信を行うプログラムです。TCP/IP マネージャは、IP、TCP、UDP の各プロトコルを管理しています。TCP/IP マネージャには、次の機能があります。

・通信プロトコル管理機能
・サブネット識別機能

ユーザプログラム（ユーザタスク）は、TCP/IP マネージャのサービスコールによって、これら TCP/IP マネージャの機能を使用することができます。TCP/IP マネージャのサービスコールにおいて IP アドレスやポート番号などの論理的な対象を指定することによって、容易に通信を行うことができます。

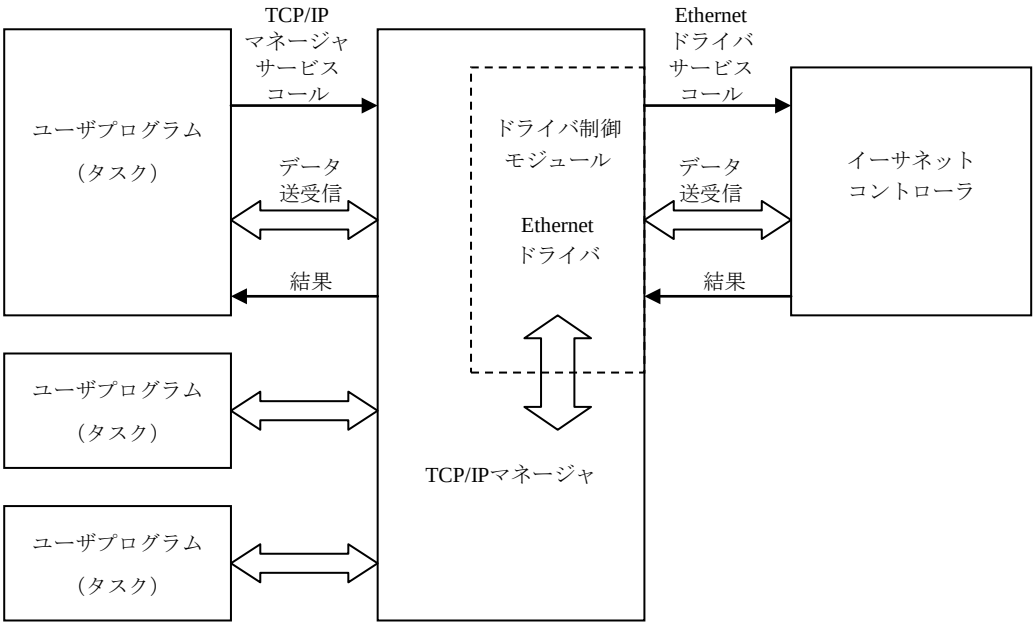


図 2-1 TCP/IPマネージャの機能

2.2 MAC アドレスと IP アドレス

TCP/IPマネージャを使うためには、MACアドレスとIPアドレスを設定しなければなりません。

MACアドレスは48ビット長でハードウェア毎に固有の値でなければなりません。IPアドレスは32ビット長で、装置の論理的なアドレスを表しています。

MACアドレスとIPアドレスの詳細は、一般のTCP/IPに関する技術書をご覧下さい。

2.3　サービスコール

TCP/IPマネージャには以下のサービスコールとコールバックがあります。サービスコールを表 2-1 に、コールバックを表 2-2 に示します。

また、TCP/IPマネージャで使用するサービスコールは基本マネージャコールと拡張マネージャコールに分類されます。なお、コールバックルーチンの名称はユーザ任意であるため、ここで使用している名称は仮称です。

表 2-1　サービスコール

区分	サービスコール名称	サービスコールの機能
初期化コール	man_ip_init	TCP/IPマネージャ初期化
基本マネージャコール	man_ip_start	TCP/IPマネージャ動作開始
	man_ip_stop	TCP/IPマネージャ動作停止
	man_ip_getver	TCP/IPマネージャのバージョン取得
IP拡張マネージャコール	man_ip_creadr	IPアドレス登録
	man_ip_deladr	IPアドレス削除
	man_ip_refact	IP登録状況の参照
	man_ip_refadr	IP登録情報の参照
	man_ip_adrchk	IPアドレス使用状況の確認
	man_ip_ping	pingの実行
TCP拡張マネージャコール	tcp_cre_rep	TCP受付口の生成
	tcp_del_rep	TCP受付口の削除
	tcp_cre_cep	TCP通信端点の生成
	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	接続要求待ち（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）
	tcp_snd_buf	バッファ内のデータの送信（省コピー）
	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）
	tcp_rel_buf	受信用バッファの解放（省コピー）
	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	ペンディングしている処理のキャンセル
	tcp_set_opt	TCP通信端点オプション設定
	tcp_get_opt	TCP通信端点オプション読出し
UDP拡張マネージャコール	udp_cre_cep	UDP通信端点の生成
	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	ペンディングしている処理のキャンセル
	udp_set_opt	UDP通信端点オプション設定
	udp_get_opt	UDP通信端点オプション読出し

表 2-2　コールバック

区分	コールバック名称(仮称)	コールバックの機能
TCPコールバック	callback_wblk	ノンブロッキングコールの完了通知
	callback_rcvoob	緊急データの受信
	callback_rcvdat	データの受信通知 ***(※独自仕様)***
	callback_sndemp	送信バッファの開放通知 ***(※独自仕様)***
	callback_rcvsyn	SYNの受信通知 ***(※独自仕様)***
	callback_rcvicmp	ICMPの受信通知 ***(※独自仕様)***
UDPコールバック	callback_wblk	ノンブロッキングコールの完了通知
	callback_rcvdat	UDPパケットの受信

3. プログラムの書き方

3.1 サービスコールの使い方

TCP/IP マネージャを使うプログラムでは、使用する OS のヘッダファイルを組み込んでください。

TCP/IPマネージャを使うためには、はじめに man_ip_start をコールします。
man_ip_start では、MACアドレスやIPアドレスを設定し、TCP/IPマネージャタスクの登録と起動、登録されたドライバ制御モジュールを経由してEthernetドライバ等の起動やTCP/IPマネージャ内のIPの動作開始処理などを行い、ユーザアプリケーションがTCP/IPマネージャを使うための準備を行います。
TCP/IPマネージャのシステムが使用するメモリ領域は、この man_ip_start のパラメータとして与えなければなりません。

3.2 TCP と UDP

man_ip_startを使ってTCP/IPマネージャを起動すると、TCPとUDPプロトコルが利用できます。
TCPはエラー回復機能を持ったデータ転送プロトコル、UDPはエラー回復機能を持たないデータ転送プロトコルです。
ユーザアプリケーションからは ITRON TCP/IP API仕様に準拠したマネージャコールによって利用できます。

3.3 TCP の使い方

TCP には自分から相手に接続する「能動オープン」と相手からの接続を待つ「受動オープン」があります。
「能動オープン」「受動オープン」共に「TCP 通信端点」が必要です。また「受動オープン」の場合は「TCP 受付口」も必要になります。

3.3.1 TCP の受動オープン

TCPを受動オープンするためにはまず、 tcp_cre_rep で自分のIPアドレスと受け付けるポート番号を指定して、TCP受付口を作成します。

次に、tcp_cre_cep でTCP通信端点を作成します。TCP通信端点はデータの送受信の基準となるものです。ここではTCP/IPマネージャのシステムが送信処理と受信処理で使うためのメモリ領域とユーザがサービスコール実行結果やTCPマネージャからの通知を受け取るコールバックルーチンのアドレスを設定します。

TCP受付口とTCP通信端点を作成したら tcp_acp_cep で受動オープンを行います。受動オープンではTCP受付口とTCP通信端点、および相手からの接続を受け付けたときの相手のIPアドレスとポート番号を受け取る領域のアドレスを指定します。

これでTCPの受動オープンの手続きが完了しました。相手が接続してくれば、データの送受信が可能になります。

3.3.2 TCP の能動オープン

TCPを能動オープンするときは、tcp_cre_cep でTCP通信端点を作成します。能動オープンの場合もTCP通信端点はデータの送受信の基準となるものです。受動オープンと同様にTCP/IPマネージャのシステムが送信処理と受信処理で使うためのメモリ領域とユーザがサービスコール実行結果やTCPマネージャからの通知を受け取るコールバックルーチンのアドレスを設定します。

TCP通信端点さえ作成すれば tcp_con_cep で能動オープンを行うことができます。能動オープンではTCP通信端点と共に、自分のIPアドレスとポート番号、および通信したい相手のIPアドレスとポート番号を指定します。

これでTCPの能動オープンの手続きが完了しました。相手との接続が成功すれば、データの送受信が可能になります。

3.3.3 TCP のデータ送信

TCPでデータを送信するには、能動もしくは受動オープンによって相手と接続済みのTCP通信端点が必要です。送信には２とおりの方法があります。１つは tcp_snd_dat を使って送信する方法、もう１つは送信バッファを tcp_get_buf で取得してバッファにデータをコピーした後、tcp_snd_buf でコピーしたデータ長分を送信する方法です。ここでは tcp_snd_dat で送信する方法を解説します。

tcp_snd_dat では１回のコールで指定したデータ長分を全て送信できるとは限りません。そのTCP通信端点の送信バッファに１バイトでも空きがあれば、その空き領域分までしかコピーせずに、サービスコールからリターンします。そのため、送信したいデータ長分の転送が終わるまで繰り返し tcp_snd_dat をコールする必要があります。

3.3.4 TCP のデータ受信

TCPでデータを受信するには送信の場合と同様に相手と接続済みのTCP通信端点が必要です。受信にも２とおりの方法があります。１つは tcp_rcv_dat を使って受信する方法、もう１つは受信バッファを tcp_rcv_buf で取得してバッファのデータを取り出した後、 tcp_rel_buf で取り出したデータ長分を解放する方法です。ここでは tcp_rcv_dat で受信する方法をを解説します。

tcp_rcv_dat では受信バッファに入っているデータを取り出せるだけ取り出してリターンします。バッファに１バイトしか入っていない場合でも、取り出してすぐにリターンします。そのため、取りだしたいデータ長が決まっている場合、取り出したい長さに達するまで、繰り返し tcp_rcv_dat をコールする必要があります。

3.3.5 TCP のクローズ

通信が終了したらTCPをクローズします。一度接続に成功したTCPを未使用の状態に戻すには tcp_cls_cep をコールしなければなりません。ここではTCPをクローズする方法を解説します。

能動もしくは受動オープンによって相手と接続済みになると、データの送受信が可能になります。

送信用のサービスコールを用いて、送信バッファに設定したデータは異常切断がない限り、相手のTCPに到達することが保証されます。送信バッファに設定したデータの送信が終了したら接続をクローズしたい場合は 予め tcp_cls_cep をコールしておくことでデータの送信終了後に自動的に切断手順を実行できます。ただし、tcp_cls_cep コール後に受信したデータは捨てられてしまうので、受信を継続したい場合は tcp_sht_cep を使ってください。

3.4 UDP の使い方

UDP はオープンの必要がありません。

UDP 通信端点を作成するだけでデータの送受信が可能になります。

3.4.1 UDP 通信端点の作成

UDP通信端点は udp_cre_cep で作成します。UDP通信端点はデータの送受信の基準となるものです。自分のIPアドレスとポート番号､サービスコール実行結果やTCPマネージャからの通知を受け取るコールバックルーチンのアドレスを設定します。

3.4.2 UDP のデータ送信

UDPはUDP通信端点を作成すれば、いつでも送受信が可能です。

UDPでデータを送信するには udp_snd_dat を使います。UDPでは送信時に udp_snd_dat のパラメータで相手のIPアドレスとポート番号を指定します。

udp_snd_dat では空いている送信バッファ長に入る分のデータをコピーした後､コピーしたデータ長を返します。１回でのコールで指定したデータ長分が全て送信できるとは限りません。そのため、送信したいデータ長分の転送が終わるまで繰り返し udp_snd_dat をコールする必要があります。

3.4.3 UDP のデータ受信

UDPでデータを受信するには udp_rcv_dat を使います。UDPでは受信時に udp_rcv_dat のパラメータで指定したアドレスに相手のIPアドレスとポート番号を受け取ります。

udp_rcv_dat では指定した受信バッファに入る分だけデータをコピーします｡指定したバッファに入りきらなかったデータは捨てられてしまうので注意が必要です。本TCP/IPマネージャでは最大1472バイトのUDPデータグラムを受信します。そのため、1472バイト以上の受信バッファを指定すれば１回分の受信データグラムを確実に受け取ることができます。

4. マネージャコール

マネージャコールは、ユーザアプリケーションがTCP/IPプロトコルスタックを利用する場合のインタフェースを提供します。基本マネージャコールでは、TCP/IPプロトコルスタックを起動するAPIを提供します。拡張マネージャコールではTCP/IPプロトコルスタックの機能拡張とITRON TCP/IP APIを提供します。

本節では、マネージャコールについての詳細な説明を以下の形式で行っています。

No.　マネージャコール名　　機能

【発行可能なシステム状態*1】

C言語インタフェース

マネージャコール呼出し形式*2

パラメータ		
型	パラメータ	パラメータの意味
・	・	・
・	・	・
・	・	・

リターンパラメータ		
型	パラメータ	パラメータの意味
・	・	・
・	・	・

パケットの構造

リターン値／エラーコード	
リターン値またはニモニック	リターン値またはエラーコードの意味*3
・	・
・	・
・	・

解　説
・・・・・・・・・・・・

*1 発行可能なシステム状態を以下のアルファベットで示します

Ｔ：タスク実行状態

Ｄ：ディスパッチ禁止状態

Ｌ：ＣＰＵロック状態

Ｉ：非タスク部実行状態

なお、各状態の詳細は各OSのユーザーズマニュアルを参照してください。

発行可能なシステム状態以外の状態でマネージャコールを発行した場合、システムの正常な動作は保証されません。

*2 呼出し形式で、リターン値がvoid型のもの（void ercd=xxx_xxx_xxx）はマネージャコールの処理でリターン値を返さないものです。

*3 エラーコードE_PARの理由として、アドレスが４の倍数以外、アドレスが奇数についてのエラーは、奇数アドレスからの16ビットや32ビットアクセスが可能なマイコン向けの製品では発生しません。

*4 使用するOSに応じて、マネージャコールは拡張SVCまたはサブルーチンコールとなります。サブルーチンコールの場合、拡張SVCに関するエラーは発生しません。また、拡張SVCに関する説明は該当しません。

4.1　初期化マネージャコール

4.1.1　man_ip_init　　TCP/IP マネージャ初期化

【Ｔ／Ｄ／Ｌ／Ｉ】

Ｃ言語インタフェース

void = man_ip_init (void);

パラメータ

無し

リターンパラメータ

無し

解　説

TCP/IPマネージャを初期化する。

TCP/IPマネージャの内部変数を初期化し、TCP/IPマネージャの各マネージャコールを拡張SVCとして登録します。

man_ip_init は、システム初期化時に 1 回だけ実行してください。

man_ip_init をコールしなかったり、OSの構築情報として拡張SVCの最大登録数が不足しているい状態でman_ip_init をコールした場合は、その後に拡張SVCとして登録されていないマネージャコールを呼出した時点で、エラーコードとしてE_RSFNが返ります。

4.2　基本マネージャコール

4.2.1　man_ip_start　　TCP/IP マネージャ動作開始

【T／D】

C言語インタフェース

```
ER ercd = man_ip_start ( T_MAN_ IP_PAR *par);
```

パラメータ

T_MAN_ IP_PAR	*par	IP動作情報の先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_MAN_ IP_PAR	par->memlen	使用したメモリの長さ
T_MAN_ IP_PAR	par->perrno	E_PAR, E_SYS発生時の詳細エラー情報

パケットの構造

```
typedef     struct{
            T_IP_INF    *ipaddrinf[DRVATTMAX];      IP情報の先頭アドレス(DRVATTMAX = 4)
            ID          ip_maxinfid;                最大IP情報ID
            ID          tcp_maxcepid;               最大TCP通信端点ID
            ID          tcp_maxrepid;               最大TCP受付口ID
            ID          udp_maxcepid;               最大UDP通信端点ID
            UW          *memaddr;                   使用するメモリの先頭アドレス
            W           memlen;                     使用可能なメモリの長さ
            H           perrno;                     E_PAR, E_SYS発生時の詳細エラー情報
} T_MAN_ IP_PAR;

typedef     struct{
            UH          type;           IP情報のタイプ
            ID          drvid;          ドライバモジュールID
            H           iftype;         インタフェースのタイプ
            UH          mach;           自MACアドレス上位16ビット
            UW          macl;           自MACアドレス下位32ビット
            UW          ipaddr;         自IPアドレス
            UW          ipmask;         サブネットマスク
            UW          gwaddr;         ゲートウェイのIPアドレス
            H           arptime;        ARPキャッシュタイムアウト時間（秒）
            H           submtu;         サブネットワークのMTU値
            H           extmtu;         外部ネットワークのMTU値
            UB          defaulttos;     デフォルトTOS
            UB          defaulttl;      デフォルトTTL
} T_IP_INF;
```

リターン値／エラーコード

E_OK	正常終了
E_NOMEM	メモリ不足（必要なメモリが確保できない）
E_PAR	パラメータエラー（parが4の倍数以外，ip_maxinfid＞4またはip_maxinfid≦0, tcp_maxcepid＞128またはtcp_maxcepid＜0，tcp_maxrepid＞128またはtcp_maxrepid＜0, udp_maxcepid＞128またはudp_maxcepid＜0, drvid＞4またはdrvidで指定したドライバモジュールが未登録, 同一ドライバモジュールに異なるmach,maclを設定しようとした, ipaddrが0または0xffffffff，gwaddrが0xffffffff，ipmaskが0xffffffffまたは0xfffffffe, submtuおよびextmtuが不正，arptime＜0）
E_OBJ	オブジェクト状態不正（すでに動作開始している）
E_SYS	man_ip_initで拡張SVCの登録に失敗した

詳細エラーコード（E_PAR発生時）

PER_INFID	1	ip_maxinfid が不正
PER_TCEPID	2	tcp_maxcepid が不正
PER_TREPID	3	tcp_maxrepid が不正
PER_UCEPID	4	udp_maxcepid が不正
PER_DRVID	5	drvid が不正
PER_MAC	6	mach, macl が不正
PER_IPADDR	7	ipaddrがマルチキャストアドレス または gwaddr が不正
PER_MTU	8	submtu または extmtu が不正
PER_ARPTIM	9	arptime が不正
PER_SYS	10	tcpipConfTbl の設定値 または apiConfTbl.cTickCnt が不正、コンパイラのマクロ定義の指定が異なる
PER_STADRV	11	ドライバの起動エラー
PER_IP	12	type または iftype が不正、ipaddr が重複
PER_CREEVF	13	イベントフラグ生成エラー
PER_CRETSK	14	タスク生成エラー
PER_STATSK	15	タスク起動エラー
PER_CRECYC	16	周期起動ハンドラ生成エラー
PER_HVER	17	ヘッダファイル(tcpip03.h)バージョン不一致
PER_CVER	18	コンフィギュレーションファイル(tcpip03.c)バージョン不一致

解　説

IPおよびドライバの動作を開始します。

ip_maxinfid 、tcp_maxcepid 、tcp_maxrepid およびudp_maxcepid にはそれぞれシステムで使用する最大IP情報ID 、最大TCP通信端点ID 、最大TCP受付口ID および最大UDP通信端点ID を指定します。本マネージャコールでは、必要となるメモリをmemaddrで指定されたメモリから切り出し、使用したメモリの長さをmemlenに返します。

man_ip_initコール時に拡張SVCの登録に失敗している場合には、エラーコードとしてE_SYSを返し、perrnoに登録に失敗した拡張SVCの機能コードを返します。

指定したパラメータが不正の場合は、エラーコードとしてE_PARを返し、perrnoに詳細エラーコードを返します。ソースコード提供版にて、コンパイラのマクロ定義の指定が異なる場合、詳細エラーコードとして、PER_SYSが返る場合があります。コンパイラのマクロ定義の指定に関しては構築マニュアルを参照してください。

指定したmemlenが必要なメモリサイズに満たない場合には、memlenに必要なメモリの長さを設定し、エラーコードとしてE_NOMEMを返します。

すでに、TCP/IPマネージャが動作を開始していた場合は、エラーコードとしてE_OBJを返します。

IP情報は最大4つ(DRVATTMAX)まで登録できます。

登録領域を確保するIP情報の数をip_maxinfidで指定し、ip_maxinfidで指定した分だけIP情報(ipaddrinf)を設定してください。

起動時にIP情報を登録しないipaddrinfのtypeにはIPTYPE_FREE(H'0000)を設定してください。typeにIPTYPE_FREE(H'0000)を設定した場合、ipaddrinf上の他の情報を無視します。

起動時にIP情報を登録するipaddrinfのtypeにはIPTYPE_USE(H'0002)を指定してください。

- drvidにはシステム構築時にdrvConfTblに登録したドライバモジュールのIDを設定します。
- ifTypeには、次の値を指定します。

  IFTYPE_ETHER　(H'0006)　ethernet（CSMACD）
  IFTYPE_CSMACD　(H'0007)　IEEE802.3（CSMACD）
  IFTYPE_PPP　(H'0017)　PPP
- 同一のドライバモジュールに対し、異なるmach,maclは設定できません。
- サブネットマスクを使用しない場合はipmaskにすべて0を設定して下さい。サブネットマスクを使用する場合は、外部ネットワークに接続するルータのIPアドレスをgwaddrに設定して下さい。gwaddrに0を設定した場合は、ゲートウェイ登録無しとして扱います。

- submtu、extmtuにはネットワーク上、扱う事のできるIPパケットの最大サイズ（MTU：Maximun Transmit Unit）を設定します。設定できる範囲は、ethernetでは46～1500、IEEE802.3では38～1492です。
- defaulttosにはIPパケットのサービスタイプ(Type of Service)を指定します。通常は0を設定してください。
- defaultttlにはIPパケットの生存時間(Time to Live)を指定します。通常は64～128の範囲で設定します。
- arptimeにはARPキャッシュの生存時間を秒単位で指定します。通常は5分(300秒)～15分(900秒)程度に設定します。0を指定した場合、生存時間の経過によるARPキャッシュの削除は行いません。

4.2.2 man_ip_stop TCP/IP マネージャ動作停止

【T／D】

C言語インタフェース

```
void ercd = man_ip_stop( void );
```

パラメータ

なし

リターンパラメータ

なし

解　説

IPおよびドライバの動作を停止します。

本マネージャコールでは、全ての通信端点および受付口に対し、ペンディング中の処理がある場合強制終了します。強制終了されたタスクにはエラーコードとしてEV_IPSTPを返します。

本マネージャコールの発行後、 本マネージャの動作を停止し、各機能は無効となります。そのためman_ip_startとman_ip_info以外のマネージャコールの動作は保証されません。

4.2.3 man_ip_getver TCP/IP マネージャのバージョン取得

【T／D】

C言語インタフェース

```
void ercd = man_ip_getver( T_MAN_VER *ver);
```

パラメータ

T_MAN_VER	*ver	バージョン情報を格納する領域のアドレス

リターンパラメータ

T_MAN_VER	*ver	バージョン情報

パケットの構造

```
typedef     struct  t_man_ver {
            UH      manver;     TCP/IPマネージャのバージョン
            UH      apiver;     TCP/IPマネージャのAPIバージョン
            UH      corever;    TCP/IPマネージャのコアバージョン
            UH      incver;     インクルードファイルのバージョン
            UH      cfgver;     コンフィグレーションファイルのバージョン
} T_MAN_VER;
```

解　説

TCP/IPマネージャのバージョン情報をverの指す領域に返します。
本マネージャコールはman_ip_startの発行より前に使用する事ができます。
バージョン情報には次の情報が格納されています。

TCP/IPマネージャのバージョン　manver
TCP/IPマネージャのバージョンを返します。

TCP/IPマネージャのAPIバージョン　apiver
TCP/IPマネージャのAPIバージョンを返します。

TCP/IPマネージャのコアバージョン　corever
TCP/IPマネージャのコアバージョンを返します。

インクルードファイルのバージョン　incver
TCP/IPマネージャのインクルードファイル（tcpip03.h）のバージョンを返します。

コンフィグレーションファイルのバージョン　cfgver
TCP/IPマネージャのコンフィグレーションファイル（tcpip03.c）のバージョンを返します。

4.3 IP 拡張マネージャコール

4.3.1 man_ip_creadr IP アドレス登録

【T／D】

C言語インタフェース

```
ER ercd = man_ip_creadr(ID ipid, T_IP_INF *par);
```

パラメータ

ID	ipid	IPアドレス登録ID
T_IP_INF	*par	登録するIPアドレス情報の先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef    struct{
           UH          type;          IP情報のタイプ
           ID          drvid;         ドライバモジュールID
           H           iftype;        インタフェースのタイプ
           UH          mach;          自MACアドレス上位16ビット
           UW          macl;          自MACアドレス下位32ビット
           UW          ipaddr;        自IPアドレス
           UW          ipmask;        サブネットマスク
           UW          gwaddr;        ゲートウェイのIPアドレス
           H           arptime;       ARPキャッシュタイムアウト時間（秒）
           H           submtu;        サブネットワークのMTU値
           H           extmtu;        外部ネットワークのMTU値
           UB          defaulttos;    デフォルトTOS
           UB          defaultttl;    デフォルトTTL
} T_IP_INF;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（parが4の倍数以外， drvid＞4またはdrvidで指定したドライバモジュールが未登録， 同一ドライバモジュールに異なるmach,maclを設定しようとした， ipaddrが0または0xffffffff，gwaddrが0xffffffff，ipmaskが0xffffffffまたは0xfffffffe， submtuおよびextmtuが不正，arptime＜0）
E_ID	不正ID番号（ipid≦0，ipid＞4）
E_OBJ	ipidで指定した登録テーブルが登録済み
E_ILUSE	サービスコール不正使用（IPが停止している）

解説

IPアドレス情報を登録します。

ipidで指定したID番号に対応するIP情報を設定しますIP情報が設定できるのはIP情報タイプがIPTYPE_FREE（空き）になっているIDに対してのみです。

typeにはIPTYPE_USE(H'0002)を指定してください。

- drvidにはシステム構築時にdrvConfTblに登録したドライバモジュールのIDを設定します。
- ifTypeには、次の値を指定します。
  - IFTYPE_ETHER (H'0006) ethernet（CSMACD）
  - IFTYPE_CSMACD (H'0007) IEEE802.3（CSMACD）
- 同一のドライバモジュールに対し、異なるmach,maclは設定できません。
- サブネットマスクを使用しない場合はipmaskにすべて0を設定して下さい。サブネットマスクを使用する場合は、外部ネットワークに接続するルータのIPアドレスをgwaddrに設定して下さい。gwaddrに0を設定した場合は、ゲートウェイ登録無しとして扱います。
- submtu、extmtuにはネットワーク上、扱う事のできるIPパケットの最大サイズ（MTU：Maximun

Transmit Unit）を設定します。設定できる範囲は、ethernetでは46～1500、IEEE802.3では38～1492です。

- defaulttosにはIPパケットのサービスタイプ(Type of Service)を指定します。通常は0を設定してください。
- defaultttlにはIPパケットの生存時間(Time to Live)を指定します。通常は64～128の範囲で設定します。
- arptimeにはARPキャッシュの生存時間を秒単位で指定します。通常は5分(300秒)～15分(900秒)程度に設定します。0を指定した場合、生存時間の経過によるARPキャッシュの削除は行いません。

4.3.2 man_ip_deladr IP アドレス削除

【T／D】

C言語インタフェース

```
ER ercd = man_ip_deladr(ID ipid);
```

パラメータ

ID	ipid	IPアドレス登録ID

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（ipid≦0，ipid＞4）
E_ILUSE	サービスコール不正使用（IPが停止している）

解説

ID番号に対応するIP情報を削除します。

ipidで指定したID番号のIPアドレスをTCPまたはUDPで使用中の場合は、そのIPアドレスを使用中の通信端点で待っている全てのサービスコールをキャンセル※し、TCP受付口、UDP通信端点を強制的に削除します。

IP情報削除によってIP情報タイプはIPTYPE_FREE（空き）に変化します。

※　自IPアドレスにIPV4_ADDRANY(0)を指定したTCP受付口にてtcp_acp_cepサービスコールを発行した場合は、tcp_acp_cepサービスコールの待ち状態のキャンセル、およびTCP受付口の削除は行いません。また、自IPアドレスにIPV4_ADDRANY(0)を指定したUDP通信端点にて、udp_rcv_datサービスコールを発行した場合も、udp_rcv_datサービスコールの待ち状態のキャンセル、およびUDP通信端点の削除は行いません。

4.3.3 man_ip_refact IP アドレス登録状態の参照

【T／D】

C言語インタフェース

```
ER ercd = man_ip_refact(T_TIP_ACTINF *par);
```

パラメータ

T_TIP_ACTINF	*par	IPアドレス登録状態を返す領域のアドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_TIP_ACTINF	par	IPアドレス登録状態（IP情報タイプ）

パケットの構造

```
typedef     struct{
            UH          IP1type;        ID=1のIP情報タイプ
            UH          IP2type;        ID=2のIP情報タイプ
            UH          IP3type;        ID=3のIP情報タイプ
            UH          IP4type;        ID=4のIP情報タイプ
} T_TIP_ACTINF;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（parが4の倍数以外）
E_ILUSE	サービスコール不正使用（IPが停止している）

解説

IPアドレスの登録状態を参照します。

IP1type,IP2type,IP3type,IP4typeのIP情報タイプには、次の値を返します。

IPTYPE_NEVERUSE(H'FFFF)	使用禁止
IPTYPE_FREE(H'0000)	空き（未使用）
IPTYPE_SPECIAL(H'0001)	特殊割付（システムが使用中）
IPTYPE_USE(H'0002)	割付（使用中）

IPTYPE_SPECIAL(H'0001)の特殊割付は~~HI.CommunicationEngine の~~システムアプリケーション専用の割付です。

4.3.4 man_ip_refinf IP アドレス登録情報の参照

【T／D】

C言語インタフェース

```
ER ercd = man_ip_refinf(ID ipid, T_IP_INF *par);
```

パラメータ

ID	ipid	IPアドレス登録ID
T_IP_INF	*par	参照するIPアドレス情報の先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_IP_INF	par	IPアドレス登録情報

パケットの構造

```
typedef    struct{
           UH          type;          IP情報のタイプ
           ID          drvid;         ドライバモジュールID
           H           iftype;        インタフェースのタイプ
           UH          mach;          自MACアドレス上位16ビット
           UW          macl;          自MACアドレス下位32ビット
           UW          ipaddr;        自IPアドレス
           UW          ipmask;        サブネットマスク
           UW          gwaddr;        ゲートウェイのIPアドレス
           H           arptime;       ARPキャッシュタイムアウト時間（秒）
           H           submtu;        サブネットワークのMTU値
           H           extmtu;        外部ネットワークのMTU値
           UB          defaulttos;    デフォルトTOS
           UB          defaultttl;    デフォルトTTL
} T_IP_INF;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（parが4の倍数以外）
E_ID	不正ID番号（ipid≦0，ipid＞4）
E_OBJ	IP情報が登録されていない
E_ILUSE	サービスコール不正使用（IPが停止している）

解説

IPアドレス情報を参照します。

ipidで指定したID番号に対応するIP情報を参照します。

IP情報として、MACアドレス、自IPアドレス、サブネットマスク、ゲートウェイのIPアドレス、ARPキャッシュタイムアウト時間、MTU値、デフォルトTOSおよびデフォルトTTLが参照できます。

IP情報が登録されていないIPアドレスIDを参照しようとした場合は、エラーコードとしてE_OBJを返します。

IP情報のタイプtypeには次のものがあります。

IPTYPE_NEVERUSE(H'FFFF)	使用禁止
IPTYPE_FREE(H'0000)	空き（未使用）
IPTYPE_SPECIAL(H'0001)	特殊割付（使用中）
IPTYPE_USE(H'0002)	割付（使用中）

IP情報のタイプtypeがIPTYPE_NEVERUSEとIPTYPE_FREEの場合には、その他のIPアドレス情報の値は不定となります。

IPTYPE_SPECIAL(H'0001)の特殊割付は~~HI.CommunicationEngine~~のシステムアプリケーション専用の割付です。

4.3.5 man_ip_adrchk IP アドレス使用状況の確認

【T】

C言語インタフェース

ER ercd = man_ip_adrchk(ID ipid, UW dstipaddr, H waitcnt);

パラメータ

ID	ipid	IPアドレス登録ID
UW	dstipaddr	応答を確認する相手のIPアドレス
H	waitcnt	相手からの応答を待つ時間カウント値

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（dstipaddr＝0またはdstipaddr＝0xffffffff，waitcnt≦0,または waitcnt＞100）
E_ID	不正ID番号（ipid≦0，ipid＞4）
E_OBJ	ipidで指定したIP情報のtypeがIPTYPE_USE(H'0002)以外，または、IPがビジーのため一時的に送信が行えない
E_QOVR	man_ip_adrchk使用中
E_TMOUT	指定時間カウント内に応答が無い
E_ILUSE	サービスコール不正使用（IPが停止している）

解説

ネットワーク上でIPアドレスが使用中かを確認します。

ipidで指定したIPアドレスからdstipaddrで指定した相手にARPを送信して応答を待ちます。登録してあるサブネットマスク値にかかわらず、指定したIPアドレスに対してARPを送信します。ゲートウェイを超えた問合せは行いません。

dstipaddrで指定したIPアドレスから応答があった場合、正常終了としてE_OKを返します。

dstipaddrで指定したIPアドレスから応答が無かった場合、エラーコードとしてE_TMOUTを返します。

waitcntに指定できるカウント値は1～100の範囲で、1カウント当たり100ミリ秒です。

dstipaddrに自IPアドレスを指定して本マネージャコールを使用することで GratuitousARPを送信することができます。GratuitousARPを送信した場合は、エラーコードとしてE_TMOUTが返った場合が正常です。もし、正常終了としてE_OKが返った場合は、自IPアドレスと同じIPアドレスを持つ相手がネットワーク上に存在する事になります。

4.3.6 man_ip_ping ping の実行

【T】

C言語インタフェース

ER ercd = man_ip_ping(ID ipid, UW dstipaddr, H waitcnt, T_ECHO *par);

パラメータ

ID	ipid	IPアドレス登録ID
UW	dstipaddr	pingを送信する相手のIPアドレス
H	waitcnt	相手からの応答を待つ時間カウント値
T_ECHO	*par	ping情報テーブルの先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
H	par->result	実行結果
TMO	par->time	応答時間

パケットの構造

typedef	struct{		
	UB	*sndaddr;	送信するデータが格納された領域のアドレス
	H	sndlen;	送信するデータの長さ
	H	result;	実行結果
	TMO	time;	応答時間

} T_ECHO;

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（parが4の倍数以外，dstipaddr＝0またはdstipaddr＝0xffffffff, waitcnt≦0またはwaitcnt＞100，sndlen＜0）
E_ID	不正ID番号（ipid≦0，ipid＞4）
E_OBJ	ipidで指定したIP情報のtypeがIPTYPE_USE(H'0002)以外，または、IPがビジーのため一時的に送信が行えない
E_QOVR	man_ip_ping使用中
E_OACV	指定したdstipaddrに送信するパスが見つからない
E_TMOUT	指定時間カウント内に応答が無い
E_ILUSE	サービスコール不正使用（IPが停止している）
E_BOVR	指定したsndlenがMTU－28を超えている

解説

pingコマンドを実行します。
ipidで指定したIPアドレスからdstipaddrで指定した相手にICMPエコー要求を送信して応答を待ちます。
dstipaddrで指定したIPアドレスからICMPエコー応答があった場合は、正常終了としてE_OKを返します。
正常終了した場合、実行結果 result には次の値が設定されます。

0	：ICMPエコー要求で送信したデータとICMPエコー応答で受信したデータが完全一致
-1	：ICMPエコー要求で送信したデータよりICMPエコー応答で受信したデータの方が短い
-2	：ICMPエコー要求で送信したデータよりICMPエコー応答で受信したデータの方が長い
正の値	：ICMPエコー要求で送信したデータとICMPエコー応答で受信したデータを比較した結果、先頭からresult 番目のデータが不一致

dstipaddrで指定したIPアドレスにマルチキャストアドレス（224.0.0.0～239.255.255.255）を指定した場合は、Ethernetアドレスがマルチキャストアドレス（01:00:5e:00:00:00～01:00:5e:7f:ff:ff）に変換されて送信されます。

dstipaddrで指定したIPアドレスに到達するパスが見つからない場合、エラーコードとしてE_OACVを返します。このエラーはゲートウェイを設定しない状態で、dstipaddrに送信元IPアドレスと異なるサブネットワークに属するIPアドレスを指定した場合に発生します。

dstipaddrで指定したIPアドレスからICMPエコー応答が無かった場合、エラーコードとしてE_TMOUTを返します。

waitcntに指定できるカウント値は1～100の範囲で、1カウント当たり100ミリ秒です。

sndlenに指定できるデータ長は、MTUー（IPヘッダサイズ（20）＋ICMPヘッダサイズ（8））までです。この値を超えている場合、エラーコードとしてE_BOVRを返します。（MTUはman_ip_startのパラメータにて指定した値で、dstipaddrがサブネットワーク内か、外部ネットワークかにより異なります。）

4.4　TCP 拡張マネージャコール

以下に、TCP拡張マネージャコール発行によるTCP通信端点の状態遷移を示します。

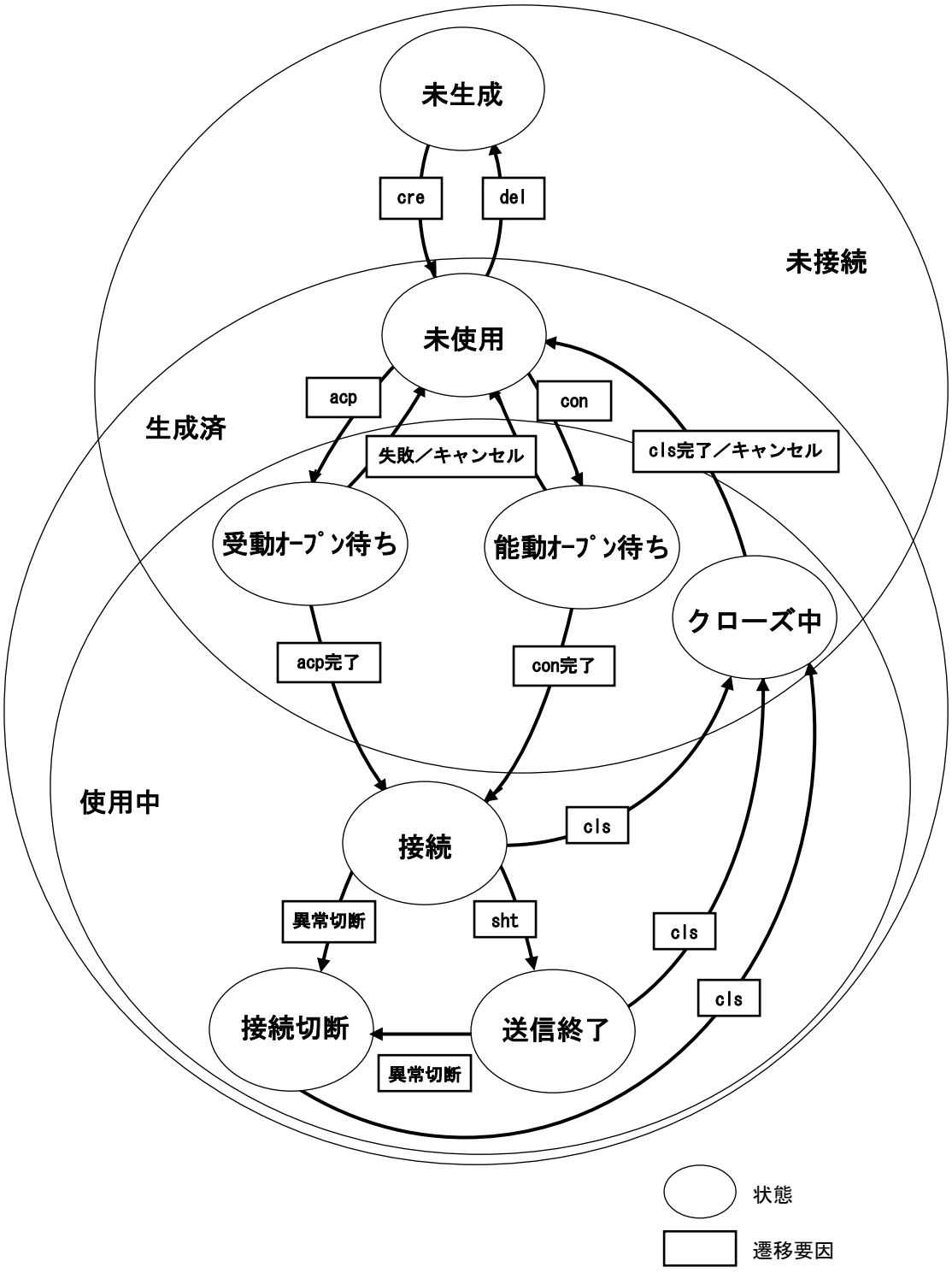


図 4-1　TCP通信端点の状態遷移図

4.4.1 tcp_cre_rep TCP 受付口の生成

【T／D】

C言語インタフェース

```
ER ercd = tcp_cre_rep(ID repid, T_TCP_CREP *pk_crep);
```

パラメータ

ID	repid	TCP受付口ID
T_TCP_CREP	*pk_crep	TCP受付口生成情報の先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_TCP_CREP	pk_crep->crerepid	生成したTCP受付口ID

パケットの構造

```
typedef     struct{
            ATR             repatr;         TCP受付口属性（未使用）
            T_IPV4EP        myaddr;         自IPアドレスおよびポート番号
            ID              crerepid ;      生成したTCP受付口ID※独自機能
} T_TCP_CREP;

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

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（pk_crepが4の倍数以外，ipaddrが動作を開始しているIPアドレスと異なる）
E_ID	不正ID番号（repid＜0，repid＞tcp_maxrepid[*1]）
E_OBJ	オブジェクト状態エラー（指定したTCP受付口IDが生成済み，ポート番号既使用）
E_NOID	ID番号不足（repidに0を指定した場合）
E_ILUSE	サービスコール不正使用（IPが停止している）

*1 tcp_maxrepid : man_ip_startで指定した最大TCP受付口ID

解説

repidで示されたIDを持つTCP受付口を、pk_crepで示された内容で生成します。

*repidにTCP_REPIDANY(0)を指定した場合、未登録の受付口IDを検索して生成します。**※独自機能***

repatrは、将来拡張用であり、本マネージャコールでは使用しません。

自IPアドレスにIPV4_ADDRANY(0)を指定した場合、動作を開始しているIPアドレスに対する接続要求を待ち受けます。

4.4.2 tcp_del_rep TCP 受付口の削除

【T／D】

C言語インタフェース

```
ER ercd = tcp_del_rep(ID repid);
```

パラメータ

ID	repid	TCP受付口ID

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（repid≦0，repid＞tcp_maxrepid*1）
E_NOEXS	オブジェクト未生成（repidのTCP受付口が存在していない）
E_ILUSE	サービスコール不正使用（IPが停止している）

*1 tcp_maxrepid : man_ip_startで指定した最大TCP受付口ID

解説

repidで示されたTCP受付口を削除します。

削除されたTCP受付口で接続要求を待っている（tcp_acp_cep発行）タスクには、接続待ち状態を解除し、エラーコードとしてE_DLTを返します。

4.4.3 tcp_cre_cep TCP 通信端点の生成

【T／D】

C言語インタフェース

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

パラメータ

ID	cepid	TCP通信端点ID
T_TCP_CCEP	*pk_ccep	TCP通信端点生成情報の先頭アドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_TCP_CCEP	pk_ccep->crecepid	生成したTCP通信端点ID

パケットの構造

```
typedef     struct{
            ATR     cepatr;         TCP通信端点属性（未使用）
            VP      sbuf;           送信用ウィンドウバッファの先頭アドレス
            INT     sbufsz;         送信用ウィンドウバッファのサイズ
            VP      rbuf;           受信用ウィンドウバッファの先頭アドレス
            INT     rbufsz;         受信用ウィンドウバッファのサイズ
            FP      callback;       コールバックルーチンのアドレス
            ID      crecepid ;      生成したTCP通信端点ID※独自機能
} T_TCP_CCEP;
```

リターン値／エラーコード

E_OK	正常終了
E_NOSPT	未サポート（sbufまたはrbufにNULL(0)を指定した）
E_PAR	パラメータエラー（pk_ccepが 4 の倍数以外，callbackが奇数）
E_ID	不正ID番号（cepid＜0，cepid＞tcp_maxcepid[*1]）
E_OBJ	オブジェクト状態エラー（指定したIDのTCP通信端点が生成済み）
E_NOID	ID番号不足（cepidに0を指定した場合）
E_ILUSE	サービスコール不正使用（IPが停止している）
E_NOMEM	ウィンドウバッファサイズ不足（sbufsz＜2, rbufsz＜2）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたIDを持つTCP通信端点を、pk_ccepで示された内容で生成します。
生成したTCP通信端点は、受動オープン、能動オープンの両方に使うことができます。
*cepidにTCP_CEPIDANY(0)を指定した場合、未登録のTCP通信端点IDを検索して生成します。**※独自機能***

sbufおよびrbufで指定するには、できる限り偶数アドレスを指定してください。また、sbufszおよびrbufszで指定するウィンドウバッファのサイズには偶数バイト数を指定してください。

sbufおよびrbufに奇数アドレスを指定した場合や、sbufszおよびrbufszに奇数バイト数を指定した場合は、TCP/IPマネージャ内部で各々のウィンドウバッファの先頭アドレスとバイト数補正し、指定された領域内で偶数アドレスから偶数バイト数を使用するよう補正して利用するため、有効なウィンドウバッファ長が短くなります。

sbufszおよびrbufszで指定するウィンドウバッファのサイズの推奨値は、共に MSS×2×N です（Nは2 以上の整数）。一般的なEthernet環境での最小推奨値はN=1の場合の2920バイトとなりますが、大容量のデータ通信のスループットを確保する必要がある場合にはNの値を大きくとってください。指定されたウィンドウバッファの領域が不正（無効なメモリ領域など）な場合、システムの正常な動作は保証されません。

尚、先頭アドレスを奇数、サイズに2を指定すると、補正後の有効なウィンドウバッファ長が０バイトになりますが、本マネージャコールではエラーを返しません。このウィンドウバッファ長が０バイトの通信端点を使用した場合は、TCPコネクションは完了しますが、データの送受信はできません。

rbuf にNULL(0)を指定した場合、受信用ウィンドウバッファ用のメモリをマネージャコール内部（man_ip_startで指定したmemaddrの領域）から確保します。そのため、本機能を使用する場合は、予め構築情報として tcpipConfTbl の tcpRcvQueMaxおよびrcvBufAddCnt に 受信用ウィンドウバッファのサイズ／MSS ＋1 個分の追加バッファ個数を設定しておいてください。

sbuf にNULL(0)を指定した場合、送信用ウィンドウバッファ用のメモリをマネージャコール内部から確保する機能は未サポートであるため、エラーコードとしてE_NOSPTを返します。

4.4.4 tcp_del_cep TCP 通信端点の削除

【T／D】

C言語インタフェース

ER ercd = tcp_del_cep(ID cepid);

パラメータ

ID	cepid	TCP通信端点ID

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が使用中）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点を削除します。
使用中のTCP通信端点を削除しようとした場合、エラーコードとしてE_OBJを返します。

4.4.5 tcp_acp_cep 接続要求待ち（受動オープン）

【T】

C言語インタフェース

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

パラメータ

ID	cepid	TCP通信端点ID
ID	repid	TCP受付口ID
T_IPV4EP	*p_dstaddr	相手IPアドレスおよびポート番号を返す領域の先頭アドレス
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	ercd	リターン値またはエラーコード
T_IPV4EP	p_dstaddr	相手IPアドレスおよびポート番号

パケットの構造

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

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（p_dstaddrが4の倍数以外，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が使用中）
E_DLT	接続要求を待っている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

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

repidで示されたTCP受付口IDに対する接続要求を待ちます。

接続要求があった場合には、cepidで示されたTCP通信端点IDを用いて接続を行い、相手のIPアドレスおよびポート番号を返します。接続が完了するまで待ち状態となります。

tmoutには、接続が完了するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

ただし、本マネージャコールは相手との接続処理を行うことから必ず待ち状態になります。そのため、ポーリング指定では接続動作を行わず、ただちにエラーコードとしてE_TMOUTを返します。

同じTCP受付口IDに対して、同時に複数のtcp_acp_cepを発行することができます。

本マネージャコールが発行されることによって、cepidで示されたTCP通信端点は未使用状態から受動オープン待ち状態（使用中）になり、接続が成功した時点で接続状態となります。タイムアウトやtcp_can_cepによってtcp_acp_cepの処理がキャンセルされた場合、 未使用状態に戻ります。

本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

4.4.6 tcp_con_cep 接続要求待ち（能動オープン）

【T】

C言語インタフェース

```
ER ercd = tcp_con_cep(ID cepid, T_IPV4EP *p_myaddr, T_IPV4EP *p_dstaddr, TMO tmout);
```

パラメータ

ID	cepid	TCP通信端点ID
T_IPV4EP	*p_myaddr	自IPアドレスおよびポート番号を格納した領域の先頭アドレス
T_IPV4EP	*p_dstaddr	相手IPアドレスおよびポート番号を格納した領域の先頭アドレス
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

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

リターン値／エラーコード

E_OK	正常終了
E_NOSPT	未サポート（自portnoにTCP_PORTANY(0)を指定，p_myaddrにNULL(0)を指定）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_PAR	パラメータエラー（p_myaddrまたはp_dstaddrが4の倍数以外，自ipaddrが動作を開始しているIPアドレスと異なる，相手ipaddrが0または0xffffffff，tmout＜-2）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が使用中，ポート番号既使用）
E_WBLK	ノンブロッキングコール受付け
E_TMOUT	ポーリング失敗またはタイムアウト
E_RLWAI	処理のキャンセル，待ち状態の強制解除
E_CLS	接続が要求が拒否された

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点を用いて、p_myaddrで示された自IPアドレスおよびポート番号とp_dstaddrで示された相手IPアドレスおよびポート番号の接続を行います。

自IPアドレスにIPV4_ADDRANY(0)を指定した場合、動作を開始しているIPアドレスを使用します。

tmoutには、接続が完了するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

ただし、本マネージャコールは相手との接続処理を行うことから必ず待ち状態になります。そのため、ポーリング指定では接続動作を行わず、ただちにエラーコードとしてE_TMOUTを返します。本マネージャコールを発行することによって、TCP通信端点は能動オープン待ち状態（使用中）となります。そのため、その状態で同一のTCP通信端点にtcp_con_cepが発行された場合、エラーコードとしてE_OBJを返します。

接続が成功した時点で接続状態となります。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、未使用状態に戻ります。

本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

自ポート番号にTCP_PORTANY(0)を指定する機能（ポート番号の自動設定）、およびp_myaddrにNULL(0)を指定する機能（IPアドレスおよびポート番号の自動設定）は、本製品では未サポートのためエラーコードとしてE_NOSPTを返します。

4.4.7 tcp_sht_cep　　データ送信の終了

【T／D】

C言語インタフェース

ER ercd = tcp_sht_cep(ID cepid);

パラメータ

ID	cepid	TCP通信端点ID

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が接続状態でない）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点の送信バッファ中のデータを送信後に、FINが送られるように接続の切断手順を手配します。

tcp_sht_cepは、切断の手配を行うだけであるため待ち状態になることはありません。

本マネージャコールを発行することによって、TCP通信端点は送信終了状態となります。それ以降データの送信はできません。送信しようとするとエラーコードとしてE_OBJを返します。なお、データの受信は可能です。また、同一のTCP通信端点に対して本マネージャコールを発行した場合は、２回目以降の発行ではエラーコードとしてE_OBJを返します。

4.4.8 tcp_cls_cep 通信端点のクローズ

【T】

C言語インタフェース

```
ER ercd = tcp_cls_cep(ID cepid, TMO tmout);
```

パラメータ

ID	cepid	TCP通信端点ID
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定した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

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点の接続を切断します。

指定したTCP通信端点がまだ送信を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。その後受信したデータは捨てられます。切断処理が終了するまで数分かかる場合があります。

tmoutには、切断処理が終了するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

同一のTCP通信端点に対するtcp_ cls _cepがペンディング中にtcp_cls_cepを発行された場合、未接続であるためエラーコードとしてE_OBJを返します。

タイムアウトやtcp_can_cepによってtcp_cls_cepの処理がキャンセルされた場合は、接続を強制切断（RSTを送信）し、それぞれにエラーコードとしてE_TMOUT、E_RLWAIを返します。

tmoutにTMO_POL(0)を指定して本マネージャコールを発行した場合は、接続を強制切断（RSTを送信）し、エラーコードとしてE_TMOUTを返します。

また､本マネージャコール内でタイムアウト待ちになっているタスクに対し､起床要求(wup_tsk, iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

本マネージャコールからリターンすると（ノンブロッキングを指定した場合は、完了を通知するコールバックが呼ばれると）、TCP通信端点は未使用状態となります。

4.4.9 tcp_snd_dat データの送信

【T】

C言語インタフェース

```
ER_UINT sdatlen = tcp_snd_dat(ID cepid, VP data, INT len, TMO tmout);
```

パラメータ

ID	cepid	TCP通信端点ID
VP	data	送信データの先頭アドレス
INT	len	送信したいデータの長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	sdatlen	リターン値またはエラーコード

リターン値／エラーコード

正の値	正常終了（送信バッファに入れたデータの長さ）
E_PAR	パラメータエラー（len≦0，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続または送信終了，tcp_snd_datまたはtcp_get_bufがペンディング中、緊急データ帯域内モードにてtcp_snd_oob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

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点からデータを送信します。

送信データが送信バッファに入った時点で、本マネージャコールからリターンします。

空いている送信バッファ長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。送信バッファに空きがない場合には、空きが生じるまで待ち状態になります。

tmoutには、送信バッファが空くまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

同一のTCP通信端点に対するtcp_snd_datかtcp_get_bufがペンディングしている間にtcp_snd_datが発行された場合、エラーコードとしてE_OBJを返します。また、本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

4.4.10 tcp_rcv_dat　　データの受信

【T】

C言語インタフェース

ER_UINT rdatlen = tcp_rcv_dat(ID cepid, VP data, INT len, TMO tmout);

パラメータ		
ID	cepid	TCP通信端点ID
VP	data	受信データを入れる領域の先頭アドレス
INT	len	受信データを入れる領域の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER_UINT	rdatlen	リターン値またはエラーコード

リターン値／エラーコード	
正の値	正常終了（取り出したデータの長さ）
0	データ終結（接続または送信終了状態でFINを受信し、受信バッファが空）
E_PAR	パラメータエラー（len≦0，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続， tcp_rcv_datまたはtcp_rcv_buf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（受信処理中にtcp_cls_cepが発行された、 または異常切断）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示された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取り出した時点で、本マネージャ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待ち状態となります。

tmoutには、データを受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

同一のTCP通信端点に対するtcp_rcv_datかtcp_rcv_bufがペンディングしている間にtcp_rcv_datが発行された場合、 エラーコードとしてE_OBJを返します。また、本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

TCP接続が異常切断した場合でも、tcp_cls_cep発行前であれば、受信バッファ中にデータがある間、tcp_rcv_datで受信データを取り出すことができます。

接続または送信終了状態でFINを受信し、かつ受信データが無い場合は、リターン値として0を返します。接続切断状態（RSTを受信または送信）で、かつ受信データが無い場合は、リターン値としてE_CLSを返します。

4.4.11 tcp_get_buf 送信用バッファの取得（省コピー）

【T】

C言語インタフェース

```
ER_UINT gbuflen = tcp_get_buf(ID cepid, VP *p_buf, TMO tmout);
```

パラメータ

ID	cepid	TCP通信端点ID
VP	*p_buf	取得したバッファのアドレスを格納する領域のアドレス
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	gbuflen	リターン値またはエラーコード
VP	p_buf	空き領域の先頭アドレス

リターン値／エラーコード

正の値	正常終了（空き領域の長さ）
E_PAR	パラメータエラー（p_bufが４の倍数以外，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続または送信終了，tcp_snd_datまたはtcp_get_bufがペンディング中、緊急データ帯域内モードにてtcp_snd_oob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

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点の送信用バッファを取得します。

次の送信データを入れる空き領域の先頭アドレスと、連続した空き領域の長さを返します。送信バッファに空きがない場合には、空きが生じるまで待ち状態となります。

tmoutには、送信バッファが空くまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

tcp_get_bufが続けて発行されると同じ領域を返します（空き領域の長さは長くなる場合もあります）。

tcp_snd_dat、tcp_snd_bufおよびtcp_cls_cepが発行されると、それ以前にtcp_get_bufが返した情報は無効となります。

同一のTCP通信端点に対するtcp_snd_datかtcp_get_bufがペンディングしている間にtcp_get_bufが発行された場合、エラーコードとしてE_OBJを返します。また、本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

4.4.12 tcp_snd_buf　　バッファ内のデータの送信（省コピー）

【T／D】

C言語インタフェース

```
ER ercd = tcp_snd_buf(ID cepid, INT len);
```

パラメータ

ID	cepid	TCP通信端点ID
INT	len	データの長さ

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（len≦0）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続または送信終了，len＞tcp_get_bufで取得した領域の長さ）
E_CLS	TCP接続が切断された

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点から、tcp_get_bufで取得したバッファに書き込んだデータの送信を手配します。

tcp_snd_bufは、送信する手配を行うだけであるため、待ち状態になることはありません。

tcp_get_bufで取得した空き領域外を書き換えた場合、システムの正常な動作は保証されません。

4.4.13　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）

【T】

C言語インタフェース

```
ER_UINT rbuflen = tcp_rcv_buf(ID cepid, VP *p_buf, TMO tmout);
```

パラメータ

ID	cepid	TCP通信端点ID
VP	*p_buf	取得したバッファのアドレスを格納する領域のアドレス
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	rbuflen	リターン値またはエラーコード
VP	*p_buf	受信データの先頭アドレス

リターン値／エラーコード

正の値	正常終了（受信データの長さ）
0	データ終結（接続または送信終了状態でFINを受信し、受信バッファが空）
E_PAR	パラメータエラー（p_bufが４の倍数以外，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続， tcp_rcv_datまたはtcp_rcv_buf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（受信処理中にtcp_cls_cepが発行された、 または異常切断）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点の受信データが入っているバッファの先頭アドレスと、そこから連続して入っているデータの長さを返します。

受信バッファが空の場合には、データを受信するまで待ち状態となります。

tmoutには、データを受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0）を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

tcp_rcv_bufが続けて発行されると同じ領域を返します（データの長さは長くなる場合もあります）。

tcp_rcv_dat、tcp_rel_bufおよびtcp_cls_cepが発行されると、それ以前にtcp_rcv_bufが返した情報は無効となります。

同一のTCP通信端点に対するtcp_rcv_datかtcp_rcv_bufがペンディングしている間にtcp_rcv_bufが発行された場合、エラーコードとしてE_OBJを返します。また、本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk, iwup_tsk)または強制待ち解除(rel_wai, irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

TCP接続が異常切断した場合でも、tcp_cls_cep発行前であれば、受信バッファ中にデータがある間、tcp_rcv_bufで受信データの先頭アドレスを取り出すことができます。

接続または送信終了状態でFINを受信し、かつ受信データが無い場合は、リターン値として0を返します。接続切断状態（RSTを受信または送信）で、かつ受信データが無い場合は、リターン値としてE_CLSを返します。

4.4.14 tcp_rel_buf 受信用バッファの解放（省コピー）

【T／D】

C言語インタフェース

ER ercd = tcp_rel_buf(ID cepid, INT len);

パラメータ

ID	cepid	TCP通信端点ID
INT	len	データの長さ

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（len＝0またはlen＜-1）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続， len＞tcp_rcv_bufで取得した領域の長さ）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

tcp_rcv_bufで取り出したバッファ中の長さlenのデータを解放します。
本マネージャコール内で待ち状態になることはありません。

*lenにTCP_RELBUFALL(-1)を指定した場合、tcp_rcv_bufで取得したデータ領域の未解放の領域を全て解放します。**※独自機能***

4.4.15 tcp_snd_oob 緊急データの送信 *（※独自仕様）*

【T】

C言語インタフェース

ER_UINT sooblen = tcp_snd_oob(ID cepid, VP data, INT len, TMO tmout);

パラメータ		
ID	cepid	TCP通信端点ID
VP	data	送信データの先頭アドレス
INT	len	送信したいデータの長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER_UINT	sooblen	リターン値またはエラーコード

リターン値／エラーコード	
正の値	正常終了（送信バッファに入れたデータの長さ）
E_PAR	パラメータエラー（len≦0，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続または送信終了，tcp_snd_oobがペンディング中、緊急データ帯域内モードにてtcp_snd_datまたはtcp_get_buf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
E_SYS	緊急データ帯域内モードにて、通常データ送信シーケンスと重複した

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点から緊急データを送信します。

緊急データが送信バッファに入った時点で、本マネージャコールからリターンします。

空いている送信バッファ長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。送信バッファに空きがない場合には、空きが生じるまで待ち状態になります。

tmoutには、送信バッファが空くまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

同一のTCP通信端点に対するtcp_snd_oobがペンディングしている間にtcp_snd_oobが発行された場合、エラーコードとしてE_OBJを返します。また、本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

緊急データ帯域内モードにて、tcp_get_buf～tcp_snd_bufの通常データ送信シーケンスの間にtcp_snd_oobが発行された場合、エラーコードとしてE_SYSを返します。

緊急データ追い越しモードで送信しようとした緊急データは、ウィンドウバッファには格納されず、緊急データ専用バッファ上に格納されます。緊急データ専用バッファ上のデータはウィンドウバッファ上の帯域内データを追い越して緊急モード（URGビットON）で送信されます。

緊急データ帯域内モードで送信しようとした緊急データは、ウィンドウバッファに帯域内データとして格納されます。このとき、ウィンドウバッファ上の未送信のデータの先頭から格納した緊急データの最終バイトまでのデータは全て緊急モード（URGビットON）で送信されます。

※ 緊急（帯域外）データポインタの扱いはRFC1122準拠です。

4.4.16 tcp_rcv_oob 緊急データの受信 ***（※独自仕様）***

【T（コールバックルーチン専用）】

C言語インタフェース

```
ER_UINT rooblen = tcp_rcv_oob(ID cepid, VP data, INT len);
```

パラメータ

ID	cepid	TCP通信端点ID
VP	data	受信データを入れる領域の先頭アドレス
INT	len	受信データを入れる領域の長さ

リターンパラメータ

ER_UINT	rooblen	リターン値またはエラーコード

リターン値／エラーコード

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（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_PAR	パラメータエラー（len≦0）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点が未接続，緊急データを受信していない）
E_BOVR	バッファオーバフロー
E_ILUSE	サービスコール不正使用（コールバックルーチン以外から発行）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点から受信した緊急データを取り出します。

受信データを入れる領域が、受信した緊急データ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は捨て、エラーコードとしてE_BOVRを返します。

緊急データを受信していない場合は、エラーコードとしてE_OBJを返します。

本マネージャコールは、緊急データ受信のコールバックルーチン内でのみ使用できます。

※ 緊急（帯域外）データポインタの扱いはRFC1122準拠です。

4.4.17 tcp_can_cep　　ペンディングしている処理のキャンセル

【T／D】

C言語インタフェース

```
ER ercd = tcp_can_cep(ID cepid, FN fncd);
```

パラメータ

ID	cepid	TCP通信端点ID
FN	fncd	キャンセルするマネージャコールの機能コード

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（fncdが指定可能な機能コードでない）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したTCP通信端点にfncdで指定した処理がペンディングしていない，指定したTCP通信端点のペンディングがキャンセルできない状態）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点に対し、fncdで示されたペンディング中の処理をキャンセルします。

キャンセルされたタスクには、エラーコードとしてE_RLWAIを返します。また、ノンブロッキングコールをキャンセルした場合には、処理の完了を通知するコールバックルーチンが呼ばれます。

キャンセル可能な処理のマネージャコール名と、それを指定する機能コードは次の通りです。また、TFN_TCP_ALL(0)を指定すると、指定したTCP通信端点にペンディングしているすべての処理をキャンセルすることができます。

マネージャコール名	機能コード
tcp_acp_cep	TFN_TCP_ACP_CEP(-0x205)
tcp_con_cep	TFN_TCP_CON_CEP(-0x206)
tcp_cls_cep	TFN_TCP_CLS_CEP(-0x208)
tcp_snd_dat	TFN_TCP_SND_DAT(-0x209)
tcp_rcv_dat	TFN_TCP_RCV_DAT(-0x20a)
tcp_get_buf	TFN_TCP_GET_BUF(-0x20b)
tcp_rcv_buf	TFN_TCP_RCV_BUF(-0x20d)
tcp_snd_oob	TFN_TCP_SND_OOB(-0x20f)
すべて	TFN_TCP_ALL(0)

尚、接続要求待ち（受動オープン）のTCP通信端点から呼出されたSYNの受信通知コールバック（*callback_rcsyn*）の中から、自らの通信端点を指定して本マネージャコールを呼出した場合は、エラーコードとしてE_OBJを返します（ペンディングがキャンセルできない状態）。

4.4.18 tcp_set_opt TCP 通信端点オプション設定

【T／D】

C言語インタフェース

```
ER_UINT optlen = tcp_set_opt(ID cepid, INT optname, VP optval, INT optlen);
```

パラメータ

ID	cepid	TCP通信端点ID
INT	optname	オプションの種類
VP	optval	オプション値を入れた領域のアドレス
INT	optlen	オプション値を入れた領域の長さ

リターンパラメータ

ER_UINT	optlen	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（optnameが未定義，optvalが4の倍数以外，optlenが足りない）
E_ID	不正ID番号（cepid≦0，cepid＞tcp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したオプションが設定できない状態）

*1 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点のオプションを設定します。

未定義のoptnameを指定した場合、optvalが4の倍数でない場合、optlenが設定する情報を格納した長さとして不十分な場合は、エラーコードとしてE_PARを返します。

cepidで指定したID番号がoptnameで指定したオプションで使用できない場合は、エラーコードとしてE_IDを返します。

cepidで指定したID番号のTCP通信端点が存在していない場合は、エラーコードとしてE_NOEXSを返します。（cepid=0は特例としてシステム共通オプションを意味する場合があります）

以下に、各オプションの詳細を示します。

(1) optname = TCP_OPT_CEPMODE(10), optlen = 16 または 24 または 28（TCP通信端点動作モードの設定）

cepidで示されたTCP通信端点の動作モードを設定します。

本オプション（TCP通信端点動作モードの設定）は未使用状態（TCS_NUSE）でしか設定できません。

未使用以外の状態では、エラーコードとしてE_OBJを返します。

```
typedef    struct{
        UW          cepmode;        通信端点制御オプション
        H           keeptime;       キープアライブパケット開始タイマ（秒）
        H           kretrytime;     キープアライブパケットのリトライ間隔（秒）
        H           kretrycnt;      キープアライブパケットのリトライ回数
        H           sockMSL;        最大セグメント生存時間タイマ（秒）
        H           sockTIMEWAIT;   TIMEWAIT（時間待機）状態タイマ（秒）
        UH          optTYPE;        オプションタイプ（ 0、1 または 2 ）
        H           minRTO;         最小セグメント再送時間タイマ（秒）
        H           maxRTO;         最大セグメント再送時間タイマ（秒）
        UB          sockTOS;        このソケットのType Of Service
        UB          sockTTL;        このソケットのTime To Live
        H           synRTO;         SYNセグメント再送時間タイマ初期値（秒）
        H           datRTO;         データセグメント再送時間タイマ初期値（秒）
        UH          reserve;        システム予約（ 0 固定）
} T_TCEP_MODE;
```

tcp_cre_cepによりTCP通信端点生成時の cepmode は0に初期化されます。
その際、keeptime は 120秒、kretrytime は 30秒、kretryctn は 4 回に初期化され、sockMSL, sockTIMEWAITは構築情報としてtcpipConfTbl に設定したtcpMSL（最大セグメント寿命）を基準に、sockMSL = tcpMSL（秒）、sockTIMEWAIT = tcpMSL×2（秒）に、minRTO,maxRTOはtcpipConfTbl に設定したminRTO,maxRTOの値に初期化されます。

optTYPEには、オプション指定構造体のタイプ情報を設定します。**通常はoptTYPEには、2 を設定してください。**
optTYPE = 0のときは、optTYPE以降の構造がないものとみなされ、optlen は 16 になります。
optTYPE = 1のときは、optTYPE以降のminRTO, maxRTO, sockTOS, sockTTL, reservEがあるものとみなされるためoptlen は 24 になります（この場合、synRTO, datRTOはありません）。
optTYPE = 2のときは、sockTTLとreservEの間にsynRTO, datRTOがあるものとみなされるためoptlen は 28 になります。
optTYPEが0の場合は、cepmodeのビット0～2以外のビットに関するオプションは無効になります（以前のバージョンとの互換性確保の為、optTYPE以降のメンバに対するアクセスは行いません）。
また、optTYPEが1の場合は、cepmodeのビット0～6以外のビットに関するオプションは無効になります（以前のバージョンとの互換性確保の為、synRTO以降のメンバに対するアクセスは行いません）。

通信端点制御オプション(cepmode)には、次の値を設定します。

ビット0,1:シリーウィンドウ回避機能
TMD_ACKAUTO（H'00000000) 独自制御
TMD_ACKNAGLE（H'00000001) Nagleアルゴリズムによる制御
TMD_ACKNODLY（H'00000002) 遅延制御なし
ビット2:キープアライブ機能
TMD_KEEPOFF（H'00000000) キープアライブ機能を使用しない
TMD_KEEPON（H'00000004) キープアライブ機能を使用する
ビット3:RTO指定機能
TMD_DEFRTO（H'00000000） 現在のRTOを使用する
TMD_REQRTO（H'00000008) 指定したRTOを使用する
ビット4:TOS指定機能
TMD_DEFTOS（H'00000000） 現在のTOSを使用する
TMD_REQTOS（H'00000010) 指定したTOSを使用する
ビット5:TTL指定機能
TMD_DEFTTL（H'00000000） 現在のTTLを使用する
TMD_REQTTL（H'00000020) 指定したTTLを使用する
ビット6:緊急データモード
TMD_URGPAS（H'00000000) 緊急データ追い越しモード
TMD_URGINL（H'00000040) 緊急データ帯域内モード
ビット7:PSHフラグ制御モード
TMD_PSHAUTO（H'00000000) PSHフラグ自動設定モード
TMD_PSHMANU（H'00000080) PSHフラグ手動設定モード
ビット8:TIMEWAIT禁止機能
TMD_TIMWEN（H'00000000) TIMEWAIT状態有効モード
TMD_TIMWDS（H'00000100) TIMEWAIT状態禁止モード
ビット9:クロース通知遅延機能
TMD_CLSNFDL（H'00000000) クローズ通知遅延モード
TMD_CLSNFFS（H'00000200) クローズ通知即時モード
ビット10:SYN再送時間初期値指定機能
TMD_SYNRTODEF（H'00000000） 現在のSYN再送時間初期値を使用する
TMD_SYNRTOREQ（H'00000400) 指定したSYN再送時間初期値を使用する
ビット11:データ再送時間初期値指定機能
TMD_DATRTODEF（H'00000000） 現在のデータ再送時間初期値を使用する
TMD_DATRTOREQ（H'00000800) 指定したデータ再送時間初期値を使用する
ビット12:再送時間スケール選択機能
TMD_RTOSCLSEC（H'00000000) RTOスケール秒基準モード
TMD_RTOSCLCYC（H'00001000) RTOスケール周期タイマイベント基準モード

ビット13:スロースタート機能
　TMD_SLOWON（H'00000000)　　スロースタート機能有効モード
　TMD_SLOWOFF（H'00002000)　　スロースタート機能無効モード
ビット14:Selective-ACK機能
　TMD_SACKOFF（H'00000000)　　Selective-ACK無効モード
　TMD_SACKON（H'00004000)　　Selective-ACK有効モード

cepmodeのビット6が0の場合（緊急データ追い越しモード）は、緊急データを帯域外（out-of-band）データとして扱い、ウィンドウバッファ上の未送信のデータを追い越して緊急データビットをONにして送信します。
緊急データ追い越しモードで受信した緊急データは、ウィンドウバッファに格納せずコールバックルーチンで通知されます。このモードでは緊急データをウィンドウバッファに格納しない為、コールバックルーチン内で取り出さない（コピーしない）データは失われます。

cepmodeのビット6が1の場合（緊急データ帯域内モード）は、緊急データを帯域内（in-band）データとして扱い、ウィンドウバッファ上の未送信のデータと合せて、緊急データビットをONにして送信します。
緊急データ帯域内モードで受信した緊急データは、ウィンドウバッファに帯域内データとして格納し、コールバックルーチンでは、ウィンドウバッファ内にある緊急データの終わりまでの未取得受信データ長を通知します。
緊急データ帯域内モードでは、通常データ送信と緊急データ送信で同一のバッファ領域を使用するため、お互いの送信シーケンスが重ならないようにしてください。シーケンスが重なった場合、送信するデータの内容と長さは保証されません。

cepmodeのビット7が0の場合（PSHフラグ自動設定モード）は、その送信によって送信ウィンドウバッファが空になる場合に、送信するTCPパケットのPSHフラグがセットされます。それ以外の場合は送信するTCPパケットのPSHフラグはセットされません。

cepmodeのビット7が1の場合（PSHフラグ手動設定モード）は、特に指定しない限り送信するTCPパケットのPSHフラグはセットされません。
PSHフラグ手動設定モードで送信するTCPパケットのPSHフラグをセットするには、送信用TCP拡張マネージャコール（tcp_snd_dat(), tcp_snd_buf(), tcp_snd_oob()）を呼び出す直前に、TCP_OPT_SETPSHを指定してTCP通信端点オプション設定マネージャコール（tcp_set_opt()）を実行してください。
この操作を行うことにより、それまでに送信ウィンドウバッファに設定したデータが全て送出されるまでは、途中のTCPパケットのPSHフラグはセットされず、送信ウィンドウバッファ最後のTCPパケットのPSHフラグはセットされるため、受信相手のTCP は1プロックのデータとして認識することができます。ただし、TCP_OPT_SETPSHが設定されたあとは一度送信ウィンドウバッファが空になるまで次の送信は行えません（tcp_snd_dat(), tcp_get_buf(), tcp_snd_oob()は送信バッファに空きがない時と同様の動作になります）。
なお、TCP_OPT_SETPSH の要求後にtcp_snd_dat()で送信する場合、送信バッファに入れたデータの長さが送信を要求したデータの長さより短いときはPSHフラグはセットされません。PSHをセットしたいデータが全て送信バッファに入るまでデータのアドレスと長さを更新しながら繰り返しtcp_snd_dat()を実行してください。

cepmodeのビット8が0（TIMEWAIT状態有効モード）の場合は、FINWAIT-2状態やCLOSING状態からCLOSED状態に移行する場合はTIMEWAIT状態を経由します。
cepmodeのビット8が1（TIMEWAIT状態禁止モード）の場合は、FINWAIT-2状態やCLOSING状態からCLOSED状態に移行する場合にTIMEWAIT状態を経由しません。

cepmodeのビット9が0（クローズ通知遅延モード）の場合は、TCP/IPマネージャはCLOSED状態に移行後0.5秒経過後にクローズを認識します。
cepmodeのビット9が1（クローズ通知即時モード）の場合は、TCP/IPマネージャはCLOSED状態に移行後直ちにクローズを認識します。

cepmodeのビット10が0（SYN再送時間デフォルトモード）の場合は、現在の値を使用します。通信端点生成直後のSYN再送時間の初期値にはデフォルトの値として3秒を使用します。
cepmodeのビット10が1（SYN再送時間指定モード）の場合は、SYN再送時間の初期値にはsynRTOに指定した値を使用します。

cepmodeのビット11が0（セグメント再送時間デフォルトモード）の場合は、現在の値を使用します。通信端点生成直後のセグメント再送時間の初期値にはデフォルトの値として3秒を使用します。
cepmodeのビット11が1（セグメント再送時間指定モード）の場合は、セグメント再送時間の初期値にはdatRTOに指定した値を使用します。

cepmodeのビット12が0（RTOスケール秒基準モード）の場合は、セグメント再送時間の単位として秒が用いられます。
cepmodeのビット12が1（RTOスケール周期タイマイベント基準モード）の場合は、セグメント再送時間の単位としてタイマイベント周期※が用いられます（例えば10ミリ秒毎にタイマイベントが発生する場合、単位は10ミリ秒となります）。RTOスケール周期タイマイベント基準モードを指定した場合は、ビット3、ビット10、ビット11を1にし、タイマイベント周期※を基準にしてminRTO, maxRTO, synRTO, datRTOを指定してください。cepmodeのビット12に1を指定した場合は、ビット3、ビット10、ビット11のいずれかのビットに0を指定すると、エラーコードとしてE_PARを返します。

※ タイマイベント周期 ＝ OSのタイムテック周期(通常1ms) × apiConfTbl.cCycCnt 設定値

cepmodeのビット13が0（スロースタート機能有効モード）の場合は、受信確認（ACK）無しに送信できるパケットの個数をコネクション直後に1とし、受信確認を受け取るごとに増加させます。そのため、送信のパフォーマンスはコネクション直後は低速ですが、受信確認（ACK）を受け取るごとに加速します。尚、送信できるパケットの個数は、送信ウィンドウバッファサイズ／MSSを超える事はありません。
cepmodeのビット13が1（スロースタート機能無効モード）の場合は、前述のスロースタート機能を使わないため、コネクション直後からある程度の送信速度が得られますが、ネットワーク上で輻輳が発生する場合は、このモードは使用しないでください。

cepmodeのビット14が0（Selective-ACK無効モード）の場合は、Selective-ACK機能が無効になります。
cepmodeのビット14が1（Selective-ACK有効モード）の場合は、Selective-ACK機能が有効になります。
尚、tcpipConfTbl. extConfTcp に EXCNFTCP_SACKALL(0x00000002)が設定されている場合は、本ビット指定に係わらず Selective-ACK機能が有効になります。
ただし、Selective-ACKが実際に使われるかどうかについては、コネクション先のTCPがコネクション時のネゴシェーションでSelective-ACKの使用を認めた場合に限られます。

keeptime, kretrytime, kretrycnt, sockMSL, sockTIMEWAITは通信端点ごとの設定値です。通信端点ごとに個別に設定することができます。
keeptimeに0を指定すると、以前に設定された（または初期値）のキープアライブパケット開始タイマをそのまま使用します（書き換えしません）。
kretrytimeに0を指定すると、以前に設定された（または初期値）のキープアライブパケット繰返しタイマをそのまま使用します（書き換えしません）。

sockMSL に0を指定すると、以前に設定された（または初期値）の最大セグメント寿命タイマをそのまま使用します（書き換えしません）。

sockTIMEWAIT に0を指定すると、以前に設定された（または初期値）のTIME-WAIT（時間待機：TCPSTAT_TIMEWAIT）状態タイマをそのまま使用します（書き換えしません）。
ただし、sockMSL に0以外の値を指定し、sockTIMEWAITに0を指定すると、sockTIMEWAITにsockMSL×2の値を自動で設定します。

minRTOとmaxRTOは、cepmodeにTMD_REQRTOがセットされているときだけ有効です。セグメント再送時間の計算結果がminRTOより小さくなる場合はminRTOを、maxRTOより大きくなる場合はmaxRTOをセグメント再送時間として使用します。
minRTOはmaxRTO以下でなければなりません。minRTOがmaxRTOより大きい場合は、エラーコードとしてE_PARを返します。
また、maxRTOはsockMSL以下でなければなりません。maxRTOがsockMSLより大きい場合は、エラーコードとしてE_PARを返します。
なお、minRTOに0を指定した場合は、最小セグメント再送時間を0.5秒に設定します。
（実際の時間は、設定値マイナス0.1秒から設定値の間の時間になります）

sockTOSは、cepmodeにTMD_REQTOSがセットされているときだけ有効です。sockTOSに指定したTOS値を、送信するIPパケットヘッダに設定します。本バージョンでは下位8ビットのみ有効です。
sockTTLは、cepmodeにTMD_REQTTLがセットされているときだけ有効です。sockTTLに指定したTTL値を、送信するIPパケットヘッダに設定します。ただし、sockTTLに0を指定した場合は、TMD_REQTTLオフとして扱い、デフォルトのTTLを、送信するIPパケットヘッダに設定します。
TOSおよびTTLのデフォルト値としては、man_ip_startもしくはman_ip_creadrで指定したT_IP_INFのdefaulttosとdefaulttlの値を使用します。

synRTOは、cepmodeにSYNRTOREQがセットされているときだけ有効です。
尚、synRTOはminRTO以上でsockMSL以下でなければなりません。synRTOに指定した値がこの範囲に無い場合は、エラーコードとしてE_PARを返します。

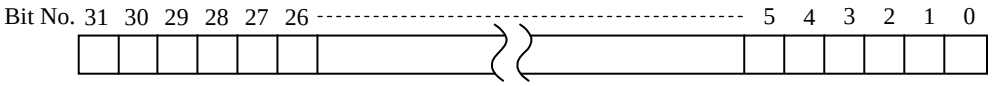
datRTOは、cepmodeにDATRTOREQがセットされているときだけ有効です。
尚、datRTOはminRTO以上でmaxRTO以下でなければなりません。datRTOに指定した値がこの範囲に無い場合は、エラーコードとしてE_PARを返します。

reserveはシステム予約領域です。必ず0を設定してください。

(2) optname = TCP_OPT_CBREXT(0), optlen = 4 （TCP拡張コールバックルーチンの設定）
cepidで示されたTCP通信端点のextcbrswで指定した各ビットに対応した拡張コールバックルーチンを有効にします。 ***(※独自仕様)***
tcp_cre_cepにより通信端点の拡張コールバックルーチンスイッチは全てOFF(無効)に初期化されます。

```
typedef   struct{
        UW        extcbrsw;                    TCP拡張コールバックルーチンスイッチ
} T_TEXTCBR_INF;
```

extcbrswのフォーマットを以下に示します。



ビットの意味付けは、'0'が「OFF(無効)」、'1'が「ON(有効)」です。

ビット位置	コールバックルーチンの機能コード	コールバックルーチンの機能
0	TEV_TCP_RCV_DAT(0x211)	データの受信通知
		callback_ rcvdat（仮称）
1	TEV_TCP_SND_EMP(0x212)	送信バッファの開放通知
		callback_sndemp（仮称）
2	TEV_TCP_RCV_SYN(0x213)	SYNの受信通知
		callback_rcvsyn（仮称）
3 ～ 31	－	予約(0固定)

拡張コールバックルーチンは関連する事象が発生したときに呼び出されますが、関連する事象を待つサービスコールがペンディングされている状態では呼び出されません。その場合は事象発生によって、ペンディングされているサービスコールの待ちが解除されます。

コールバックルーチンの機能	関連する事象	事象を待つサービスコール
データの受信通知 callback_ rcvdat（仮称）	有効セグメント受信 FIN受信 または RST受信	tcp_rcv_dat tcp_rcv_buf
送信バッファの開放通知 callback_sndemp（仮称）	送信セグメント到達確認	tcp_snd_dat tcp_get_buf
SYNの受信通知 callback_rcvsyn（仮称）	無し	－

TCPに関するサービスコールの何れかがペンディングされている場合は、RSTを受信してもデータの受信通知（callback_ rcvdat）は呼び出されません。

(3) optname = TCP_OPT_SETPSH(30), optlen = 0 （データのPSH要求）

cepidで示されたTCP通信端点において、PSHフラグ手動設定モードが設定されている場合に、現在の送信ウィンドウ内の最後のセグメントにPSHフラグをセットします。

本オプション実行後は、PSH付きのセグメントが送信されるまで、TCP/IPマネージャは送信バッファに空きが無い場合と同じふるまいをします。

(4) optname = TCP_OPT_ARPDEL(201), optlen = 4 （ARPキャッシュの削除）

cepidで示されたIPアドレスID上にある相手IPアドレスがdstipaddrのARPキャッシュを削除します。

cepidには、ARPキャッシュを削除したいIPアドレスのIDを指定します。

dstipaddrに0を指定すると、指定したIPアドレスのID上のARPキャッシュを全て削除します。

```
typedef struct {
        UW              dstipaddr;      通信相手のIPアドレス
} T_ARPDEL_INF;
```

cepidで指定したIDにIPアドレスが登録されていない場合は、エラーコードとしてE_NOEXSを返します。

(5) optname = TCP_OPT_IPMODE(202), optlen = 4 （IP動作モードの設定）

cepidで示されたIPアドレスIDで使用されるIP動作モードを設定します。

cepidには、IP動作モードを設定したいIPアドレスIDを指定します。

```
typedef struct {
        UW              ipmode;         IP制御オプション
} T_IPMODE_INF;
```

IP動作モード(ipmode)には、次の値が指定できます。
デフォルトでは、ipmodeは0（IMD_ARPCACRST｜IMD_MYARPREQ）に設定されます。

ビット0:ARPキャッシュ生存時間のリセット
- IMD_ARPCACRST（H'00000000） ARPキャッシュ生存時間をパケット送信時にリセットする
- IMD_ARPCACNORST（H'00000001） ARPキャッシュ生存時間をパケット送信時にリセットしない

ビット1〜2:ARPキャッシュの更新
- IMD_MYARPRCV（H'00000000） 自分宛のARP要求および応答でのみ更新する
- IMD_GARPRCV（H'00000002） GratuitousARPの受信で更新する
- IMD_ALLARPRCV（H'00000004） 全てのARPの受信で更新する

ARPキャッシュは、自分宛のARP要求を受信した場合、および送信したARP要求に対する応答を受信した場合に追加登録されます。ARPキャッシュの更新は、該当するIPアドレスがARPキャッシュ上に存在する場合のみ行います。ARPキャッシュ上のIPアドレスが使用されずにARPキャッシュ生存時間が経過することで、ARPキャッシュから削除されます。

ipmodeのビット0が0の場合（ARPキャッシュ生存時間をパケット送信時にリセットする）は、ARPキャッシュ生存時間を、パケットの送信毎にリセットします。

ipmodeのビット0が1の場合（ARPキャッシュ生存時間をパケット送信時にリセットしない）は、ARPキャッシュ生存時間を自動的にリセットしません。ただしARPキャッシュの更新時はリセットします。

ipmodeのビット1〜2が0の場合（自分宛のARP要求および応答でのみ更新する）は、ARPキャッシュの更新は、自分宛のARP要求を受信した場合にのみ行います。

ipmodeのビット1が1の場合（GratuitousARPの受信で更新する）は、ARPキャッシュの更新は、自分宛のARP要求を受信した場合、およびGratuitousARP（要求パケット）を受信した場合に行います。

ipmodeのビット2が1の場合（全てのARPの受信で更新する）は、ARPキャッシュの更新は、宛先にかかわらず全てのARP要求を受信した場合に行います。

cepidで指定したIDにIPアドレスが登録されていない場合は、エラーコードとしてE_NOEXSを返します。
cepidで指定したIDのインターフェースタイプが、IFTYPE_ETHERまたはIFTYPE_CSMACD以外の場合は、エラーコードとしてE_OBJを返します。

(6) optname = TCP_OPT_ICMPEXP (203), optlen = 4 （ICMP受信コールバックの設定）
cepidで示されたIPアドレスIDに対してICMP受信コールバックルーチンを設定します。
本機能で設定したIPアドレスIDに対してICMPパケットを受信した場合、callbackで指定したコールバックルーチンを呼び出します。
設定したICMP受信コールバックルーチンを無効にしたい場合は、本機能で callback に 0 を設定してください。

```
typedef struct {
        FP                      callback;               コールバックルーチンアドレス
} T_ICMPCBR_INF;
```

callbackの値が奇数の場合は、エラーコードとしてE_PARを返します。

4.4.19 tcp_get_opt TCP 通信端点オプション読出し

【T／D】

C言語インタフェース

```
ER_UINT optlen = tcp_get_opt(ID cepid, INT optname, VP optval, INT optlen);
```

パラメータ

ID	cepid	TCP通信端点ID または ドライバモジュールID
INT	optname	オプションの種類
VP	optval	オプション値を入れる領域のアドレス
INT	optlen	オプション値を入れる領域の長さ

リターンパラメータ

ER_UINT	optlen	リターン値またはエラーコード

リターン値／エラーコード

正の値	正常終了（読み出したオプション値の長さ）
E_PAR	パラメータエラー（optnameが未定義，optvalが4の倍数以外，optlenが足りない）
E_ID	不正ID番号（cepid≦0[*1]，cepid＞tcp_maxcepid[*2]）
E_NOEXS	オブジェクト未生成（cepidのTCP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したオプションが取得できない状態）

*1 TCP_OPT_CEPSTATの場合は0を許可する
*2 tcp_maxcepid : man_ip_startで指定した最大TCP通信端点ID

解説

cepidで示されたTCP通信端点の情報を読み出します。
未定義のoptnameを指定した場合、optvalが4の倍数でない場合、optlenが読み出す情報を格納する長さとして不十分な場合は、エラーコードとしてE_PARを返します。
cepidで指定したID番号がoptnameで指定したオプションで使用できない場合は、エラーコードとしてE_IDを返します。
cepidで指定したID番号のTCP通信端点が存在していない場合は、エラーコードとしてE_NOEXSを返します。（cepid=0は特例としてシステム共通オプションを意味する場合があります）

以下に、各オプションの詳細を示します。

(1) optname = TCP_OPT_CEPSTAT(0), optlen = 4 （TCP通信端点状態の参照）
cepidで示されたTCP通信端点の状態を読み出します。（cepid≠0）

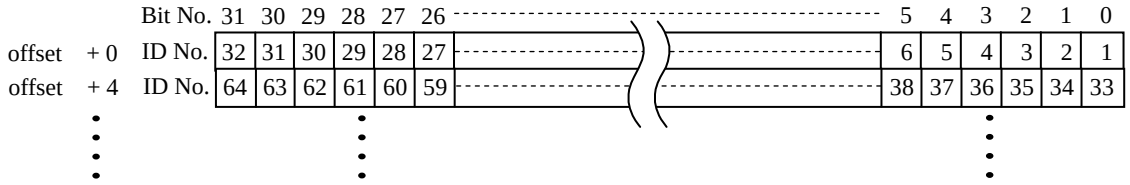
```
typedef    struct{
        STAT                cepstat;            通信端点状態
} T_TCEP_STAT;
```

通信端点状態(cepstat)には、次の値を返します。
- TCS_NUSE（H'00000001） 未使用状態
- TCS_WACP（H'00000002） 受動オープン待ち状態
- TCS_WCON（H'00000003） 能動オープン待ち状態
- TCS_CONN（H'00000004） 接続状態
- TCS_DISC（H'00000005） 接続切断状態
- TCS_ENDS（H'00000006） 送信終了状態
- TCS_WCLS（H'00000007） クローズ中状態

cepidに0を指定した場合、TCP登録情報（最大TCP通信端点IDと最大TCP受付口ID）を読み出します。
この場合、optlen = 16 となります。
maxcepid, maxrepidにはそれぞれ生成可能な通信端点、受付口の最大IDを返します。
usecepcnt, userepcntにはそれぞれ生成中の通信端点、受付口の数を返します。
usecepmap, userepmapにはそれぞれ通信端点、受付口の生成状態をビットマップで返します。

```
typedef   struct{
          ID        maxcepid;          最大TCP通信端点ID
          H         usecepcnt;         TCP通信端点生成数
          UW        *usecepmap;        TCP通信端点生成マップを返す領域のアドレス
          ID        maxrepid;          最大TCP受付口ID
          H         userepcnt;         TCP受付口生成数
          UW        *userepmap;        TCP受付口生成マップを返す領域のアドレス
} T_TREG_INF;
```

usecepmap, userepmapのフォーマットを以下に示します。
usecepmap は (maxcepid + 31) / 32 個分、userepmap は (maxrepid + 31) / 32 個分の領域を使用します。



ビットの意味付けは、'0'が「未生成」、'1'が「既生成」です。

(2) optname = TCP_OPT_ CEPINF(1), optlen = 52 （TCP通信端点情報の参照）
cepidで示されたTCP通信端点の状態、自IPアドレスおよびポート番号、相手IPアドレスおよびポート番号、およびペンディングの情報を読み出します。

```
typedef   struct{
          STAT              cepstat;          通信端点状態
          T_IPV4EP          myaddr;           自IPアドレスおよびポート番号
          T_IPV4EP          dstaddr;          相手IPアドレスおよびポート番号
          STAT              acp_cep;          acp_cepペンディング状態
          STAT              con_cep;          con_cepペンディング状態
          STAT              cls_cep;          cls_cepペンディング状態
          STAT              snd_dat;          snd_datペンディング状態
          STAT              rcv_dat;          rcv_datペンディング状態
          STAT              get_buf;          get_bufペンディング状態
          STAT              rcv_buf;          rcv_bufペンディング状態
          STAT              snd_oob;          snd_oobペンディング状態
} T_TCEP_INF;
```

ペンディング状態には、以下の値を返します。
TCP_NPND（H'00000000） ペンディングなし
TCP_WAIT（H'00000001） 待ち状態（タイムアウト待ちまたは永久待ち）
TCP_WBLK（H'00000002） ノンブロッキング受付中

(3) optname = TCP_OPT_MSS(2), optlen = 4 （最大セグメントサイズ(MSS)の参照）
cepidで示されたTCP通信端点から最大セグメントサイズを取得します。
本オプション（最大セグメントサイズ(MSS)の参照）は接続状態でしか取得できません。接続以外の状態では、エラーコードとしてE_OBJを返します。

```
typedef   struct{
          W                 mss;              最大セグメントサイズ
} T_TCEP_MSS;
```

(4) optname = TCP_OPT_ CONNENTRY(11), optlen = 20 （TCPコネクションテーブルの参照）
cepidで示されたTCPコネクションテーブルを読み出します。
本オプション（TCPコネクションテーブルの参照）は使用中の状態でしか取得できません。使用中以外の状態では、エラーコードとしてE_OBJを返します。

```
typedef   struct{
        UW      tcpConnState;           コネクション状態
        UW      tcpConnLocalAddress;    ローカルIPアドレス
        UH      tcpConnLocalPort;       ローカルポート番号
        UH      reserve1;               予約領域１
        UW      tcpConnRemAddress;      リモートIPアドレス
        UH      tcpConnRemPort;         リモートポート番号
        UH      reserve2;               予約領域２
} T_TCPCONNENTRY;
```

tcpConnStateには、以下の値を返します。

TCPSTAT_CLOSED	（H'00000001)	閉鎖
TCPSTAT_LISTEN	（H'00000002)	聴取
TCPSTAT_SYNSENT	（H'00000003)	SYN送信済み
TCPSTAT_SYNRECEIVED	（H'00000004)	SYN受信済み
TCPSTAT_ESTABLISHED	（H'00000005)	確立済み
TCPSTAT_FINWAIT1	（H'00000006)	FIN待機1
TCPSTAT_FINWAIT2	（H'00000007)	FIN待機2
TCPSTAT_CLOSEWAIT	（H'00000008)	閉鎖待機
TCPSTAT_LASTACK	（H'00000009)	最終ACK
TCPSTAT_CLOSING	（H'0000000A)	閉鎖処理中
TCPSTAT_TIMEWAIT	（H'0000000B)	時間待機

(5) optname = TCP_OPT_SNDWND(21), optlen = 28 （送信ウィンドウバッファ情報の参照）

cepidで示されたTCP通信端点の送信ウィンドウバッファ情報を読み出します。

本オプション（送信ウィンドウバッファ情報の参照）は接続状態でしか取得できません。接続以外の状態では、エラーコードとしてE_OBJを返します。

```
typedef   struct{
        UB      *sndtopadr;     送信ウィンドウバッファ 先頭アドレス
        UB      *sndbtmadr;     送信ウィンドウバッファ 最終アドレス
        UB      *sndputadr;     送信ウィンドウバッファ 書込みアドレス
        UB      *sndgetadr;     送信ウィンドウバッファ 読出しアドレス
        UB      *sndreladr;     送信ウィンドウバッファ 解放アドレス
        W       snddatalen;     送信ウィンドウバッファ 有効データ領域長
        W       sndfreelen;     送信ウィンドウバッファ 空きデータ領域長
} T_TCEP_SNDWND;
```

(6) optname = TCP_OPT_RCVWND(22), optlen = 28 （受信ウィンドウバッファ情報の参照）

cepidで示されたTCP通信端点の受信ウィンドウバッファ情報を読み出します。

本オプション（受信ウィンドウバッファ情報の参照）は接続状態でしか取得できません。接続以外の状態では、エラーコードとしてE_OBJを返します。

```
typedef   struct{
        UB      *rcvtopadr;     受信ウィンドウバッファ 先頭アドレス
        UB      *rcvbtmadr;     受信ウィンドウバッファ 最終アドレス
        UB      *rcvputadr;     受信ウィンドウバッファ 書込みアドレス
        UB      *rcvgetadr;     受信ウィンドウバッファ 読出しアドレス
        UW      reserve;        予約エリア
        W       rcvdatalen;     受信ウィンドウバッファ 有効データ領域長
        W       rcvfreelen;     受信ウィンドウバッファ 空きデータ領域長
} T_TCEP_RCVWND;
```

(7) optname = TCP_OPT_DRVIFINF(101), optlen = 20 （ドライバインタフェース情報の参照）
cepidで示されたドライバのインタフェース情報を読み出します。
cepidには、参照したいドライバのドライバモジュールIDを指定します。

```
typedef struct {
        H               ifType;         インタフェースのタイプ
        UH              ifEnInf;        インタフェース情報の有無ビット
        UW              ifStat;         インタフェースの状態
        UW              ifSpeed;        インタフェースの速度情報（ビット／秒）
        UB              *ifDescrP;      インタフェースの情報を含む文字列のアドレス
        UW              *ifSpecificP;   構成するメディアを表すOIDサブツリーのアドレス
} T_DRVIF_INF;
```

ドライバが動作していない等の理由で、cepidで指定したドライバの情報が参照できない状態の場合は、エラーコードとしてE_OBJを返します。

ifTypeには、インタフェースのタイプを返します。

6	IFTYPE_ETHER	ethernet（CSMACD）
7	IFTYPE_CSMACD	IEEE802.3（CSMACD）
23	IFTYPE_PPP	PPP

ifEnInfには、ifStat、ifSpeed、ifDescrP、ifSpecificPが有効であるか無効であるかを返します。
ifEnInf中の該当する情報に対応するビットが0（無効）の場合は、そのドライバが該当する情報の提供をサポートしていないことを示します。

ビット０：ifStat有効ビット（０：無効／１：有効）
ビット1：ifSpeed有効ビット（０：無効／１：有効）
ビット2：ifDescrP有効ビット（０：無効／１：有効）
ビット3：ifSpecificP有効ビット（０：無効／１：有効）

ifStatには、インタフェースの状態を返します。（ifEnInfのビット0が1のとき）

ビット０：接続切断状態（０：DOWN／１：UP）
ビット1：通信回線状態（０：半２重／１：全２重）
ビット2、3：回線速度状態（00：10Base／01：100Base／10：1000Base）EtherまたはIEEE802.3のみ

ifSpeedには、インタフェースの通信速度（ビット／秒）を返します。（ifEnInfのビット1が1のとき）

ifDescrPには、インタフェースの情報文字列の先頭アドレスを返します。（ifEnInfのビットが2のとき）

ifSpecificPには、インタフェースを構成するメディアのOIDサブツリーを記述した配列の先頭アドレスを返します。（ifEnInfのビットが2のとき）

4.5　UDP 拡張マネージャコール

以下に、UDP拡張マネージャコール発行によるUDP通信端点の状態遷移を示します。

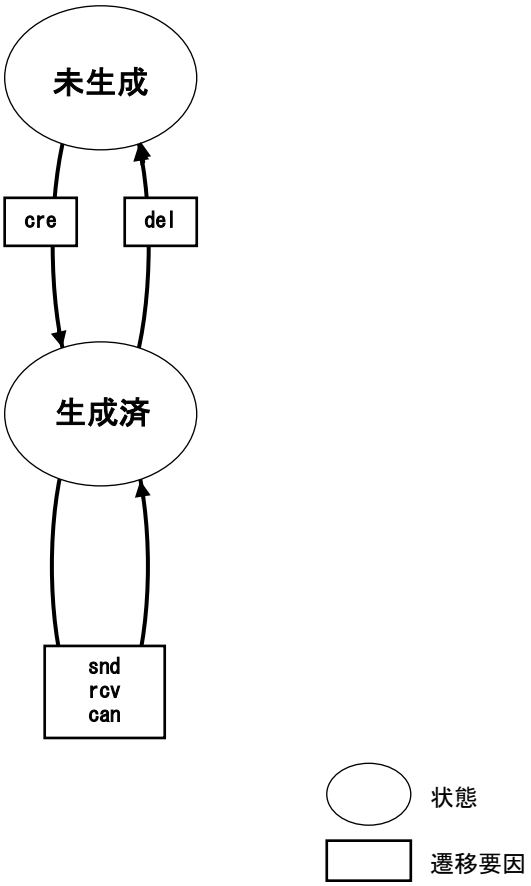


図 4-2　UDP通信端点の状態遷移図

4.5.1 udp_cre_cep UDP 通信端点の生成

【T／D】

C言語インタフェース

ER ercd = udp_cre_cep(ID cepid, T_UDP_CCEP *pk_ccep);

パラメータ

ID	cepid	UDP通信端点ID
T_UDP_CCEP	*pk_ccep	UDP通信端点生成情報の先頭アドレス

リターン値／エラーコード

T_UDP_CCEP	pk_ccep->crecepid	生成したUDP通信端点ID
ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef     struct{
            ATR             cepatr;         通信端点属性(未使用)
            T_IPV4EP        myaddr;         自IPアドレスおよびポート番号
            FP              callback;       コールバックルーチンのアドレス
            ID              crecepid ;      生成したUDP通信端点ID※独自機能
} T_UDP_CCEP;

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

エラーコード

E_OK	正常終了
E_NOSPT	未サポート（portnoにUDP_PORTANY(0)を指定，myaddrにNULL(0)を指定）
E_PAR	パラメータエラー（pk_ccepが 4 の倍数以外，ipaddrが動作を開始しているIPアドレスと異なる，callbackが奇数）
E_ID	不正ID番号（cepid＜0，cepid＞udp_maxcepid[*1]）
E_OBJ	オブジェクト状態エラー（指定したUDP通信端点IDが生成済み，ポート番号既使用）
E_NOID	ID番号不足（cepidに0を指定した場合）
E_ILUSE	サービスコール不正使用（IPが停止している）

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示された指定したIDのUDP通信端点を、pk_ccepで示された内容で生成します。

指定されたポート番号と指定されたIPアドレスで送受信を行います。なお、ブロードキャスト、およびマルチキャストされたUDPパケットも受信します。

*cepidにUDP_CEPIDANY(0)を指定した場合、未登録のUDP通信端点IDを検索して生成します。**※独自機能***

cepatrには、ユーザが指定するUDP通信端点に関する情報を設定するなどの目的で自由に使用できます。マネージャコール内の処理では使用しません。

ipaddrにIPV4_ADDRANY(0)を指定した場合、動作を開始しているIPアドレスとブロードキャスト、およびマルチキャストされたUDPパケットを受信し、送信パケットに入る自IPアドレスは動作を開始しているIPアドレスのうち、一番小さい登録IDを持つIPアドレスを使用します。

自ポート番号にUDP_PORTANY(0)を指定する機能（ポート番号の自動設定）は、本製品では未サポートのためエラーコードとしてE_NOSPTを返します。

callbackには、ユーザが作成したコールバックルーチンのアドレスを指定します。udp_rcv_datがペンディングしていない状態でUDPパケットを受信した場合には、コールバックルーチンが呼び出されるので必ず設定してください。

4.5.2 udp_del_cep UDP 通信端点の削除

【T／D】

C言語インタフェース

ER ercd = udp_del_cep(ID cepid);

パラメータ

ID	cepid	UDP通信端点ID

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（cepid≦0，cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示された指定したIDのUDP通信端点を削除します。

削除されたUDP通信端点でデータの送受信を待っているタスクには、接続待ち状態を解除し、エラーコードとしてE_DLTを返します。

4.5.3 udp_snd_dat パケットの送信

【T】

C言語インタフェース

```
ER_UINT sdatlen = udp_snd_dat(ID cepid, T_IPV4EP *p_dstaddr, VP data, INT len, TMO tmout);
```

パラメータ

ID	cepid	UDP通信端点ID
T_IPV4EP	*p_dstaddr	相手IPアドレスおよびポート番号を格納した領域のアドレス
VP	data	送信パケットの先頭アドレス
INT	len	送信パケットの長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	sdatlen	リターン値またはエラーコード

パケットの構造

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

リターン値／エラーコード

正の値	正常終了（送信バッファに入れたデータの長さ）
E_PAR	パラメータエラー（p_dstaddrが４の倍数以外，ipaddrが0，len＜0，tmout＜-2）
E_ID	不正ID番号（cepid≦0, cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定されたtmoutと異なる指定でペンディング中）
E_QOVR	キューイングオーバフロー
E_DLT	送信を待っている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

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示されたUDP通信端点から、p_dstaddrで示された相手へパケットを送信します。データが送信バッファに入った時点で、本マネージャコールからリターンします。

取得した送信バッファ長[*2]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。送信バッファ取得できない場合には、取得できるまで待ち状態になります。

tmoutには、送信バッファを取得するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

相手IPアドレスにマルチキャストアドレス（224.0.0.0～239.255.255.255）を指定した場合は、Ethernetアドレスがマルチキャストアドレス（01:00:5e:00:00:00～01:00:5e:7f:ff:ff）に変換されて送信されます。

同一のUDP通信端点に対して、同時に複数のudp_snd_datを発行することができます。

ただし、udp_maxsndquecnt(3)を超えてudp_snd_datを発行した場合、エラーコードとしてE_QOVRを返します。また、時間待ち(TMO_FEVRを含む)でペンディング中のUDP通信端点にノンブロッキング(TMO_WBLK)指定でudp_snd_datを発行した場合、またはノンブロッキングでペンディング中のUDP通信端点に時間待ち指定でudp_snd_datを発行した場合、エラーコードとしてE_OBJを返します。

本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*2 取得できる送信バッファ長は、送信相手に対するMDS（最大データグラムサイズ）とUDP通信端点オプションの状態（optname = UDP_OPT_CEPMODEの送信バッファモード）によって決定します。

4.5.4 udp_rcv_dat パケットの受信

【T】

C言語インタフェース

```
ER_UINT rdatlen = udp_rcv_dat(ID cepid, T_IPV4EP *p_dstaddr, VP data,INT len, TMO tmout);
```

パラメータ

ID	cepid	UDP通信端点ID
T_IPV4EP	*p_dstaddr	相手IPアドレスおよびポート番号を返す領域の先頭アドレス
VP	data	受信パケットを入れる領域の先頭アドレス
INT	len	受信パケットを入れる領域の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	rdatlen	リターン値またはエラーコード
T_IPV4EP	p_dstaddr	相手IPアドレスおよびポート番号

パケットの構造

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

リターン値／エラーコード

0または正の値	正常終了（取り出したデータの長さ）
E_PAR	パラメータエラー（p_dstaddrが４の倍数以外，len≦0，tmout＜-2）
E_ID	不正ID番号（cepid≦0，cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定されたtmoutと異なる指定でペンディング中）
E_QOVR	キューイングオーバフロー
E_DLT	受信を待っている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

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示されたUDP通信端点からパケットを受信し、相手のIPアドレスおよびポート番号を返します。

受信バッファに入ったデータを取り出した時点で、本マネージャコールからリターンします。取り出すデータにはUDPのヘッダは含みません。

データが無いUDPヘッダのみのパケットを受信した場合、リターン値としてデータの長さ0を返します。

cepidで示されたUDP通信端点のポート番号宛てのブロードキャストパケット、およびマルチキャストパケットも受信します。

受信バッファに入っているパケット長が受信しようとしたパケット長よりも短い場合、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返します。入りきらないデータは捨て、エラーコードとしてE_BOVRを返します。受信バッファが空の場合には、データを受信するまで待ち状態となります。

tmoutには、データを受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_POL(0)を指定した場合はポーリング、TMO_FEVR(-1)を指定した場合は永久待ち、TMO_WBLK(-2)を指定した場合はノンブロッキングコールとなります。

同一の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。ただし、udp_maxrcvquecnt(3)を超えてudp_rcv_datを発行した場合、エラーコードとしてE_QOVRを返します。また、時間待ち(TMO_FEVRを含む)でペンディング中のUDP通信端点にノンブロッキング(TMO_WBLK)指定でudp_rcv_datを発行した場合、またはノンブロッキングでペンディング中のUDP通信端点に時間待ち指定でudp_rcv_datを発行した場合、エラーコードとしてE_OBJを返します。

本マネージャコール内でタイムアウト待ちになっているタスクに対し、起床要求(wup_tsk，iwup_tsk)または強制待ち解除(rel_wai，irel_wai)が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

4.5.5 udp_can_cep ペンディングしている処理のキャンセル

【T／D】

C言語インタフェース

```
ER ercd = udp_can_cep(ID cepid, FN fncd);
```

パラメータ

ID	cepid	UDP通信端点ID
FN	fncd	キャンセルするマネージャコールの機能コード

リターンパラメータ

ER	ercd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（fncdが指定可能な機能コードでない）
E_ID	不正ID番号（cepid≦0，cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したUDP通信端点にfncdで指定した処理がペンディングしていない）

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示されたUDP通信端点に対し、fncdで示されたペンディング中の処理をキャンセルします。

キャンセルされたタスクには、エラーコードとしてE_RLWAIを返します。また、ノンブロッキングコールをキャンセルした場合には、処理の完了を通知するコールバックルーチンが呼ばれます。

キャンセル可能な処理のマネージャコール名と、それを指定する機能コードは次の通りです。また、TFN_UDP_ALL(0)を指定すると、指定したUDP通信端点にペンディングしているすべての処理をキャンセルすることができます。

マネージャコール名	機能コード
udp_snd_dat	TFN_UDP_SND_DAT(-0x223)
udp_rcv_dat	TFN_UDP_RCV_DAT(-0x224)
すべて	TFN_UDP_ALL(0)

4.5.6 udp_set_opt UDP 通信端点オプション設定

【T／D】

C言語インタフェース

```
ER_UINT optlen = udp_set_opt(ID cepid, INT optname, VP optval, INT optlen);
```

パラメータ

ID	cepid	UDP通信端点ID
INT	optname	オプションの種類
VP	optval	オプション値を入れた領域のアドレス
INT	optlen	オプション値を入れた領域の長さ

リターンパラメータ

ER_UINT	optlen	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（optnameが未定義，optvalが4の倍数以外，optlenが足りない）
E_ID	不正ID番号（cepid≦0，cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）
E_OBJ	オブジェクト状態エラー（指定したオプションが設定できない状態）

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示されたUDP通信端点のオプションを設定します。

未定義のoptnameを指定した場合、optvalが4の倍数でない場合、optlenが設定する情報を格納した長さとして不十分な場合は、エラーコードとしてE_PARを返します。

cepidで指定したID番号がoptnameで指定したオプションで使用できない場合は、エラーコードとしてE_IDを返します。

cepidで指定したID番号のUDP通信端点が存在していない場合は、エラーコードとしてE_NOEXSを返します。（cepid=0は特例としてシステム共通オプションを意味する場合があります）

以下に、各オプションの詳細を示します。

(1) optname = UDP_OPT_CEPMODE(10), optlen = 8（UDP通信端点動作モードの設定）

cepidで示されたUDP通信端点の動作モードを設定します。

```
typedef    struct{
        UW              cepmode;        通信端点制御オプション
        UB              sockTOS;        このソケットのType Of Service
        UB              sockTTL;        このソケットのTime To Live
        UH              reserve;        システム予約（ 0 固定）
} T_UCEP_MODE;
```

udp_cre_cepによりUDP通信端点生成時の cepmode は0に初期化されます。

通信端点制御オプション(cepmode)には、次の値を設定します。

ビット0:送信TOS選択

UMD_DEFTOS（H'00000000）	デフォルトのTOSを使用する
UMD_REQTOS（H'00000001)	指定したTOSを使用する

ビット1:送信TTL選択

UMD_DEFTTL（H'00000000）	デフォルトのTTLを使用する
UMD_REQTTL（H'00000002)	指定したTTLを使用する

ビット2:送信バッファモード

UMD_DEFMDS（H'00000000）	MTUからフラグメントされないMDSを取得する
UMD_EXTMDS（H'00000004)	構築時に指定したバッファ長をMDSとする

cepmodeのビット2がUMD_DEFMDSの場合は、IPの現在のMTUを基にUDPパケットがフラグメントされないサイズを送信相手に対するMDSとして求め、udp_snd_datの送信バッファ長として使用します*2（MDSの参照に関しては、udp_get_optの optname = UDP_OPT_MDSの項を参照してください）。
cepmodeのビット2がUMD_EXTMDSの場合は、tcpipConfTblのMaximum Datagram Size(udpMDS)に指定したサイズをudp_snd_datの送信バッファ長として使用します。

sockTOSは、cepmodeにUMD_REQTOSがセットされているときだけ有効です。sockTOSに指定したTOS値の下位8ビットを、送信するIPパケットヘッダに設定します。

sockTTLは、cepmodeにUMD_REQTTLがセットされているときだけ有効です。sockTTLに指定したTTL値を、送信するIPパケットヘッダに設定します。ただし、sockTTLに0を指定した場合は、UMD_REQTTLオフとして扱い、デフォルトのTTLを、送信するIPパケットヘッダに設定します。
TOSおよびTTLのデフォルト値としては、man_ip_startもしくはman_ip_creadrで指定したT_IP_INFのdefaulttosとdefaulttlの値を使用します。

reserveはシステム予約領域です。必ず0を設定してください。

*2 送信バッファ長はtcpipConfTblのMaximum Datagram Size(udpMDS)に指定した値を超える事はありません。

4.5.7 udp_get_opt UDP 通信端点オプション読出し

【T／D】

C言語インタフェース

```
ER_UINT optlen = udp_get_opt(ID cepid, INT optname, VP optval, INT optlen);
```

パラメータ

ID	cepid	UDP通信端点ID
INT	optname	オプションの種類
VP	optval	オプション値を入れる領域のアドレス
INT	optlen	オプション値を入れる領域の長さ

リターンパラメータ

ER_UINT	optlen	リターン値またはエラーコード

リターン値／エラーコード

正の値	正常終了（読み出したオプション値の長さ）
E_PAR	パラメータエラー（optnamが未定義，optvalが4の倍数以外，optlenが足りない）
E_ID	不正ID番号（cepid≦0，cepid＞udp_maxcepid[*1]）
E_NOEXS	オブジェクト未生成（cepidのUDP通信端点が存在していない）

*1 udp_maxcepid : man_ip_startで指定した最大UDP通信端点ID

解説

cepidで示されたUDP通信端点の情報を読み出します。

未定義のoptnameを指定した場合、optvalが4の倍数でない場合、optlenが読み出す情報を格納する長さとして不十分な場合は、エラーコードとしてE_PARを返します。

cepidで指定したID番号がoptnameで指定したオプションで使用できない場合は、エラーコードとしてE_IDを返します。

cepidで指定したID番号のUDP通信端点が存在していない場合は､エラーコードとしてE_NOEXSを返します。（cepid=0は特例としてシステム共通オプションを意味する場合があります）

以下に、各オプションの詳細を示します。

(1) optname = UDP_OPT_CEPSTAT(0), optlen = 8 （UDP登録情報の参照）

cepidに0を指定して、UDP登録情報（最大UDP通信端点ID）を読み出します。

maxcepidには生成可能な通信端点の最大IDを返します。

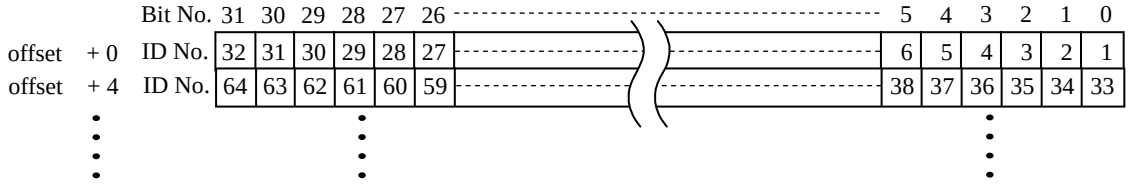
usecepcntには生成中の通信端点の数を返します。

usecepmapには通信端点の生成状態をビットマップで返します。

```
typedef   struct{
          ID        maxcepid;          最大UDP通信端点ID
          H         usecepcnt;         UDP通信端点生成数
          UW        *usecepmap;        UDP通信端点生成マップを返す領域のアドレス
} T_UREG_INF;
```

usecepmap のフォーマットを以下に示します。

usecepmap は (maxcepid + 31) / 32 個分の領域を使用します。



ビットの意味付けは、'0'が「未生成」、'1'が「既生成」です。

(2) optname = UDP_OPT_CEPINF(1), optlen = 20 （UDP通信端点状態の参照）
cepidで示されたUDP通信端点の状態、自IPアドレスおよびポート番号、ペンディングの情報を読み出します。

```
typedef    struct{
        STAT            cepstat;        通信端点状態
        T_IPV4EP        myaddr;         自IPアドレスおよびポート番号
        STAT            snd_dat;        snd_datペンディング状態
        STAT            rcv_dat;        rcv_datペンディング状態
} T_UCEP_INF;
```

通信端点状態(cepstat)には、次の値を返します。
TCS_CRED（H'00000008） 生成済み状態

ペンディング状態には、以下の値を返します。
UDP_NPND（H'00000000) ペンディングなし
UDP_WAIT（H'00000001) 待ち状態（タイムアウト待ちまたは永久待ち）
UDP_WBLK（H'00000002) ノンブロッキング受付中

(3) optname = UDP_OPT_MDS(2), optlen = 8 （最大データグラムサイズ(MDS)の参照）
cepidで示されたUDP通信端点からrmaddrで示されるIPアドレスに対して送信可能なデータグラムのサイズを取得します。
rmaddrには本マネージャコール呼び出し前に、送信相手のIPアドレスを設定してください。
mdsには取得した送信可能なデータグラムのサイズを返します。

```
typedef    struct{
        UW              rmaddr;         送信相手のIPアドレス
        W               mds;            送信可能なデータグラムのサイズ
} T_UCEP_MDS;
```

5. コールバックルーチン

コールバックルーチンは、マネージャ内で起こった事象をアプリケーションプログラムに伝えるために用いられます。コールバックルーチンはユーザが作成し、そのアドレスがマネージャコールに渡されるため、モジュール名はユーザ任意となります。ここでは仮称として***callback_xxxx***とします。

5.1 TCP 用コールバックルーチン

5.1.1 *callback_wblk（仮称）* ノンブロッキングコールの完了通知

C言語インタフェース

ER ercd = *callback_wblk* (ID cepid, FN fncd, T_TCP_CBRWBLK *p_parblk)

パラメータ

ID	cepid	TCP通信端点ID
FN	fncd	機能コード
T_TCP_CBRWBLK	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャコールでは使用しない）

パケットの構造

```
typedef     struct{
              ER_UINT       rtncd;      マネージャコールからの返値
} T_TCP_CBRWBLK;
```

解説

ノンブロッキングコールの処理が完了した（またはキャンセルされた）場合に呼び出されます。

例：ノンブロッキングのtcp_rcv_buf処理が完了した場合、バッファに連続して入っている受信データの長さがrtncdに渡され、バッファの先頭アドレスはtcp_rcv_bufを呼び出した時のパラメータp_bufが指す領域に格納されます。

渡される機能コードは次の通りです。

機能コード	マネージャコール名
TFN_TCP_ACP_CEP(-0x205)	tcp_acp_cep
TFN_TCP_CON_CEP(-0x206)	tcp_con_cep
TFN_TCP_CLS_CEP(-0x208)	tcp_cls_cep
TFN_TCP_SND_DAT(-0x209)	tcp_snd_dat
TFN_TCP_RCV_DAT(-0x20a)	tcp_rcv_dat
TFN_TCP_GET_BUF(-0x20b)	tcp_get_buf
TFN_TCP_RCV_BUF(-0x20d)	tcp_rcv_buf
TFN_TCP_SND_OOB(-0x20f)	tcp_snd_oob

5.1.2 *callback_rcvoob（仮称）* 緊急データの受信

C言語インタフェース

ER ercd = *callback_ rcvoob* (ID cepid, FN fncd, T_TCP_CBRRCVOOB *p_parblk)

パラメータ

ID	cepid	TCP通信端点ID
FN	fncd	TEV_TCP_RCV_OOB(0x201)
T_TCP_CBRRCVOOB	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            INT             len;        緊急データの長さ
} T_TCP_CBRRCVOOB;
```

解説

緊急データを受信した場合に呼び出されます。
緊急データは、コールバックルーチンの中で、tcp_rcv_oobを使って取り出してください。

緊急データ追い越しモードでは、受信した緊急データはウィンドウバッファに格納せず、本コールバックルーチンでのみ通知されます。このモードでは緊急データをウィンドウバッファに格納しない為、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、データは捨てられます。

緊急データ帯域内モードでは、受信した緊急データはウィンドウバッファに帯域内データとして格納し、本コールバックルーチンのlenでは、ウィンドウバッファ内にある緊急データの終わりまでの未取得受信データ長を通知します。この場合、コールバックルーチンの中で、tcp_rcv_oobを使って取り出さなかったデータは、ウィンドウバッファに帯域内データとして残ります（ただし、コールバックルーチンの外でtcp_rcv_oobを使って取り出すことはできません）。

※ 緊急（帯域外）データポインタの扱いはRFC1122準拠です。

5.1.3 *callback_rcvdat（仮称）* データの受信通知*（※独自仕様）*

C言語インタフェース

ER ercd = *callback_ rcvdat* (ID cepid, FN fncd, T_TCP_CBRRCVDAT *p_parblk)

パラメータ

ID	cepid	TCP通信端点ID
FN	fncd	TEV_TCP_RCV_DAT(0x211)
T_TCP_CBRRCVDAT	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            INT             len;        受信データの長さ
} T_TCP_CBRRCVDAT;
```

解説

データを受信した場合に呼び出されます。

TCPが有効なデータを受信するたびに本コールバックを呼び出します。

lenには受信バッファ内の有効データ長を設定します。

相手からFINまたはRSTを受信した場合は受信バッファ内の有効データの長さが0以外でも、lenに0を設定して本コールバックを呼び出します。そのため、本コールバックがlen=0で呼び出されたケースでも受信バッファ内には未だ取り出していない有効データが残っている場合がありますので注意してください。

尚、RSTを受信しても状態が変化しない場合は、本コールバックは呼び出されません。

本コールバックはtcp_set_optでoptnameにTCP_OPT_CBREXT（TCP拡張コールバックルーチンの設定）を指定し、TEV_TCP_RCV_DATを有効にした場合にのみ呼び出されます。

5.1.4 *callback_sndemp（仮称）* 送信バッファの開放通知***（※独自仕様）***

C言語インタフェース

ER ercd = *callback_ sndemp* (ID cepid, FN fncd, T_TCP_CBRSNDEMP *p_parblk)

パラメータ

ID	cepid	TCP通信端点ID
FN	fncd	TEV_TCP_SND_EMP(0x212)
T_TCP_CBRSNDEMP	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            INT             len;        送信バッファの空き領域の長さ
} T_TCP_CBRSNDEMP;
```

解説

送信バッファに空きが発生したことを通知する。

TCPが有効な受信確認（ACK）を受信するたびに本コールバックを呼び出します。

lenには送信バッファ内の空き領域の長さを設定します。

本コールバックはtcp_set_optでoptnameにTCP_OPT_CBREXT（TCP拡張コールバックルーチンの設定）を指定し、TEV_TCP_SND_EMPを有効にした場合にのみ呼び出されます。

5.1.5 *callback_rcvsyn（仮称）* SYNの受信通知 ***(※独自仕様)***

C言語インタフェース

ER ercd = *callback_ rcsyn* (ID cepid, FN fncd, T_TCP_CBRRCVSYN *p_parblk)

パラメータ

ID	cepid	IPアドレスID
FN	fncd	TEV_TCP_RCV_SYN(0x213)
T_TCP_CBRRCVSYN	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	接続要求判定結果

リターン値／エラーコード

0	接続許可
0 以外	接続拒否

パケットの構造

```
typedef     struct{
            T_IPV4EP        srcaddr;        送信元IPアドレスおよびポート番号
            T_IPV4EP        dstaddr;        宛先IPアドレスおよびポート番号
} T_TCP_CBRRCVSYN;
```

解説

SYNを受信したことを通知する。

接続要求待ち（受動オープン）のTCP通信端点に有効な接続要求（SYN）が到着した場合に本コールバックを呼び出します。

srcaddrには接続要求元のIPアドレスとポート番号を設定します。

dstaddrには接続を受け付けるIPアドレスとポート番号を設定します。

接続要求を受け付けたい場合はリターン値として 0 を返してください。

接続要求を拒否したい場合はリターン値として 0以外 を返してください。接続を拒否する場合はSYNに対してRSTを送信します。

本コールバックはtcp_set_optでoptnameにTCP_OPT_CBREXT（TCP拡張コールバックルーチンの設定）を指定し、TEV_TCP_RCV_SYNを有効にした場合にのみ呼び出されます。

5.1.6 *callback_rcvicmp（仮称）* ICMP の受信通知 ***(※独自仕様)***

C言語インタフェース

ER ercd = *callback_ rcvicmp* (ID ipid, FN fncd, T_TCP_CBRRCVICMP *p_parblk)

パラメータ

ID	ipid	IPアドレスID
FN	fncd	TEV_TCP_RCV_ICMP(0x21f)
T_TCP_CBRRCVICMP	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            UB          type;       ICMPパケットのタイプ
            UB          code;       ICMPパケットのコード
            ID          drvid;      ICMPを受信したドライバのID
            UW          srcip;      ICMPの送信元IPアドレス
            UW          dstip;      ICMPの宛先IPアドレス
            VP          data;       ICMPパケットが格納されている領域のアドレス
            INT         len;        ICMPパケットの長さ
} T_TCP_CBRRCVICMP;
```

解説

ICMPパケットを受信したことを通知する。
ICMPパケットを受信するたびに本コールバックを呼び出します。

typeには受信したICMPパケットのタイプを設定します（受信したものをそのまま設定します）。
codeには受信したICMPパケットのコードを設定します（受信したものをそのまま設定します）。
drvidにはICMPパケットを受信したドライバモジュールのIDを設定します。
srcipには受信したICMPパケットの送信元IPアドレスを設定します。
dstipには受信したICMPパケットの宛先IPアドレスを設定します(ICMPパケットの宛先がブロードキャストやマルチキャストの場合はdstipを参照する事で目的の宛先を確認することができます）。
dataには受信したICMPパケットが格納されている領域のICMPヘッダの先頭アドレスを設定し、lenにはICMPヘッダの先頭アドレスからICMPパケットの終端までの長さを設定します。
尚、dataの示す領域はTCP/IPマネージャのシステム領域であるため参照のみ行ってください。この領域を書き換えた場合の動作は保証いたしません。

本コールバックはtcp_set_optでoptnameにTCP_OPT_ICMPEXP（ICMP受信コールバックの設定）を指定し、callbackに有効なコールバックルーチンアドレスを登録した場合にのみ呼び出されます。

※ ICMPエコー要求(type:8)を受信した場合は、本コールバックからリターンした後にICMPエコー応答(type:0)を送信します。

5.2 UDP 用コールバックルーチン

5.2.1 *callback_wblk（仮称）* ノンブロッキングコールの完了通知

C言語インタフェース

ER ercd = *callback_wblk* (ID cepid, FN fncd, T_UDP_CBRWBLK *p_parblk)

パラメータ

ID	cepid	UDP通信端点ID
FN	fncd	機能コード
T_UDP_CBRWBLK	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            ER_UINT         rtncd;      マネージャコールからの返値
            T_IPV4EP        srcaddr;    送信元IPアドレスおよびポート番号
                                        （TFN_UDP_RCV_DATの場合のみ）
            T_IPV4EP        dstaddr;    宛先IPアドレスおよびポート番号
                                        （TFN_UDP_RCV_DATの場合のみ）
} T_UDP_CBRWBLK;

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

解説

ノンブロッキングコールの処理が完了した（またはキャンセル）場合に呼び出さます。

例：ノンブロッキングのudp_rcv_dat処理が完了した場合、取り出した受信データの長さがrtncdに、送信元のIPアドレスおよびポート番号がsrcaddrに、宛先IPアドレスおよびポート番号がdstaddrに渡されます。また、送信元のIPアドレスおよびポート番号は、udp_rcv_datを呼び出した時のパラメータp_dstaddrが指す領域にも格納されます。

渡される機能コードは次の通りです。

機能コード	マネージャコール名
TFN_UDP_SND_DAT(-0x223)	udp_snd_dat
TFN_UDP_RCV_DAT(-0x224)	udp_rcv_dat

ノンブロッキングのudp_snd_dat処理が完了した場合の、srcaddrおよびdftaddrの内容は無効です。

5.2.2 *callback_rcvdat（仮称）* UDP パケットの受信

C言語インタフェース

ER ercd = *callback_ rcvdat* (ID cepid, FN fncd, T_UDP_CBRRCVDAT *p_parblk)

パラメータ

ID	cepid	UDP通信端点ID
FN	fncd	TEV_UDP_RCV_DAT(0x221)
T_UDP_ CBRRCVDAT	*p_parblk	パラメータブロックを格納した領域のアドレス

リターンパラメータ

ER	ercd	任意の値（マネージャでは使用しない）

パケットの構造

```
typedef     struct{
            INT             len;        パケットの長さ
            T_IPV4EP        srcaddr;    送信元IPアドレスおよびポート番号
            T_IPV4EP        dstaddr;    宛先IPアドレスおよびポート番号
} T_UDP_CBRRCVDAT;

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

解説

udp_rcv_datがペンディングしていない状態でUDPパケットを受信した場合に呼び出されます。
データが無いUDPヘッダのみのパケットを受信した場合は、パケットの長さに0が渡されます。

受信データは、コールバックルーチンの中で、udp_rcv_datを使って受信してください。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、データは捨てられます。

6. ドライバモジュールインタフェース

TCP/IPマネージャが使用するドライバモジュールはユーザが準備しなければなりません。この章では、ドライバモジュールの仕様についてトライバモジュール側の立場から説明しています。

表 6-1 ドライバモジュール関数一覧表

項番	サービスコール関数名	機能概要
1	getParam	パラメータを取得する
2	setParam	パラメータを設定する
3	startMdl	モジュールを起動する
4	stopMdl	モジュールを停止する
5	sndPacket	パケットの送信を起動する
6	rcvPacket	受信パケットを取得する

表 6-2 ドライバモジュールコールバック機能一覧表

項番	コールバック機能名	機能概要
1	sndReady	パケット送信の受付準備完了を通知する
2	rcvInf	受信パケットの情報を通知する

6.1　ドライバモジュール関数

6.1.1　getParam 関数　　パラメータの取得

C言語インタフェース

W rtcd = getParam(T_DRV_PRMLSTL2 *param);

パラメータ

T_DRV_PRMLSTL2	*param	パラメータリストを返すテーブルのアドレス

リターンパラメータ

W	rtcd	リターン値
T_DRV_PRMLSTL2	param	パラメータリスト

パケットの構造

typedef struct {　　パラメータリスト

H	ifType;	インタフェースのタイプ
UH	mdlStat	モジュールの状態
H	sndPacketMin;	最小送信パケット長
H	sndPacketMax;	最大送信パケット長
H	rcvPacketMin;	最小受信パケット長
H	rcvPacketMax;	最大受信パケット長
FP	cbrAdr;	コールバックルーチンのアドレス
H	mdlId;	接続した時の接続エントリＩＤ
H	macLen;	ＭＡＣアドレス情報長
UB	macA[MACLEN];	ＭＡＣアドレス
B	sndTime;	送信待ちタイムアウト時間(単位は秒)
UB	prmType;	パラメータリストのタイプ（1固定）
UW	sndEvPtn;	送信イベントフラグパターン
UW	rcvEvPtn;	受信イベントフラグパターン
UW	ifOption;	インタフェースのオプション情報
void	*ifTimeFuncP;	インタフェース上の周期起動関数のアドレス
UB	*ifDescrP;	インタフェースの情報を含む文字列のアドレス
UW	*ifSpecificP;	構成するメディアを表すOIDサブツリーのアドレス
UW	*ifSpeedP;	インタフェースの速度情報領域のアドレス
UW	*ifStatP;	インタフェースの状態格納領域のアドレス

} T_DRV_PRMLSTL2;

リターン値

0	正常終了

解説

モジュールのパラメータリストを返します。

リストの取得に成功した場合、リターン値として0を返します。

本関数がコールされた場合、ifType、mdlStat、sndPacketMin、sndPacketMax、rcvPacketMin、rcvPacketMax、macLen、macA、sndTime、ifOption、ifTimeFuncP、ifDescrP、ifSpecificP、ifSpeedP、ifStatPを設定してリターンします。

ifTypeには、次のインタフェースのタイプを返します。

IFTYPE_ETHER	6	ethernet（CSMACD）
IFTYPE_CSMACD	7	IEEE802.3（CSMACD）
IFTYPE_PPP	23	PPP

mdlStatには、次のモジュール状態を返します。

ビット0：動作状態（0:停止中／1:動作中）

sndPacketMinには、最小送信パケット長を返します。TCP/IPマネージャが送信する最小データ長以下でなければなりません。TCP/IPマネージャが送信する最小データ長は次のとおりです。尚、TCP/IPマネージャが送信するデータには、FCSの4バイトは含まれていません。パディングが必要な場合はドライバモジュールで付加する必要があります。

ifType=IFTYPE_ETHERの場合	42バイト
ifType=IFTYPE_CSMACDの場合	50バイト
ifType=IFTYPE_PPPの場合	28バイト

sndPacketMaxには、最大送信パケット長を返します。TCP/IPマネージャが送信する最大データ長以上でなければなりません。TCP/IPマネージャが送信する最大データ長は次のとおりです。尚、TCP/IPマネージャが送信するデータには、FCSの4バイトは含まれていません。

ifType=IFTYPE_ETHERの場合	1514バイト
ifType=IFTYPE_CSMACDの場合	1514バイト
ifType=IFTYPE_PPPの場合	1500バイト

rcvPacketMinには、最小受信パケット長を返します。sndPacketMinと同じ値を返します。

rcvPacketMaxには、最大受信パケット長を返します。sndPacketMaxと同じ値を返します。

cbrAdrには、setParam関数でTCP/IPマネージャが設定したコールバックルーチンのアドレスを返します。未設定（setParam関数が呼ばれる前）の場合はNULL(0)を返します。

mdlIdには、setParam関数でTCP/IPマネージャが設定したドライバモジュールのIDを返します。未設定（setParam関数が呼ばれる前）の場合は0を返します。

macLenには、設定されているMACアドレスの長さを返します。未設定（setParam関数が呼ばれる前）の場合は0を返します。TCP/IPマネージャは本関数により取得したifTypeがIFTYPE_ETHERまたはIFTYPE_CSMACDでmacLenが0のとき、setParam関数でMACアドレスの設定を行います。

macA[MACLEN]には、ドライバモジュールに設定されているMACアドレスを返します。

sndTimeには、送信待ちタイムアウト時間(秒)を返します。TCP/IPマネージャは送信待ちパケットの待ち時間が、ここで返した分の時間だけ経過すると送信を中止して破棄します。

prmTYPEには、パラメータリストの構造として1を返します。**TCP/IPマネージャでは、prmTYPEが0の場合、ifOption以下のメンバが存在しない旧仕様のドライバモジュールとして処理します。**

sndEvPtnには、設定されている送信イベントフラグパターンを返します。未設定（setParam関数が呼ばれる前）の場合は0を返します。

rcvEvPtnには、設定されている受信イベントフラグパターンを返します。未設定（setParam関数が呼ばれる前）の場合は0を返します。

ifOptionには、ドライバモジュールがTCP/IPマネージャに対してオプションを指定する為に使用します。オプション機能がない場合はNULL(0)を返します。
ドライバモジュールがオプションをサポートしている場合、ifOptionにはオプション情報をリンクするためにドライバモジュールが準備したI/Fオプション情報テーブルの先頭アドレスが設定されています。

I/Fオプション情報テーブルの構造

```
typedef struct {
        UH      optRequest;         オプション要求
        UH      optAnswer;          オプション応答
        VP      *optInf[16];        オプション情報アドレス配列
} T_DRV_IFOPTINF;
```

ドライバモジュールは、optRequest上の自分が要求するオプションに相当するビットを1にセットしておきます。TCP/IPマネージャはこの情報を参照し、サポートしているオプションだった場合、optAnswer上の該当ビットを1にセットします。（optAnswer上の該当ビットが0の場合、そのオプションは使用してはいけません。）

その後、TCP/IPマネージャはオプションのビット番号に相当するoptInf上のアドレスにある、そのオプション固有の情報にアクセスしてオプション毎の処理を行います（オプションによってはoptInfを使用しない場合があります）。本リビジョンでは以下のオプションのみサポートしています。

ビット0：PPPマルチセッション（０：ドライバ機能無し／１：ドライバ機能有り）
ビット1：TCP/IPチェックサム・オフロード機能（０：ドライバ機能無し／１：ドライバ機能有り）

PPPマルチセッション情報テーブルの構造

```
typedef struct {
        UH          optType;          オプションタイプ（OPTTYP_PPPMLT：0x0001）
        H           sesCnt;           セッション数
        T_DRV_PPPSES *sesInf;         セッション情報テーブル
} T_DRV_PPPMLTOPT;
```

セッション情報テーブルの構造

```
typedef struct {
        UB          sesNo;            セッション番号
        UB          actFlg;           セッションのアクティブフラグ（1:アクティブ 0:未アクティブ）
        H           sesMRU;           セッションのMRU
        UW          rmIpAddr;         セッションのリモートIPアドレス
} T_DRV_PPPSES;
```

TCP/IPチェックサム・オフロード情報テーブルの構造

```
typedef struct {
        UH          optType;          オプションタイプ（OPTTYP_CSOFFLOAD：0x0002）
        UH          csFunc;           ドライバ機能有無（０：機能無し／１：機能有り）※
                                      ビット0：送信時のIPヘッダチェックサム計算機能
                                      ビット1：受信時のIPヘッダチェックサム計算機能
                                      ビット2：送信時のTCPチェックサム計算機能
                                      ビット3：受信時のTCPチェックサム計算機能
                                      ビット4：送信時のUDPチェックサム計算機能
                                      ビット5：受信時のUDPチェックサム計算機能
                                      ビット6～15：未使用
} T_DRV_CSOFFLOAD;
```

※ドライバの受信時のチェックサム計算機能を１（機能有り）にした場合、TCP/IPマネージャでは受信パケット（フラグメントパケット以外）のチェックサムの計算は行いません。チェックサムエラーとなった受信パケットはドライバにて破棄してください。
ドライバの送信時のチェックサム計算機能を１（機能有り）にした場合、TCP/IPマネージャでは送信パケット（フラグメントパケット以外）のチェックサムの計算は行いません。ドライバにて送信パケットの各チェックサムフィールドに計算したチェックサムを格納してください。

ifTimeFuncPには、ドライバモジュール上の100ミリ秒間隔で実行する関数のエントリアドレスを返します。100ミリ秒間隔で実行する関数がない場合は、ifTimeFuncPに0を返します。

ifDescrPには、インタフェースの情報を含む文字列の先頭アドレスを返します。インタフェースの情報を含む文字列が定義していない場合は、ifDescrPに0を返します。

ifSpecificPには、インタフェースを構成するメディアのOIDサブツリーを記述した配列の先頭アドレスを返します。インタフェースを構成するメディアのOIDサブツリーを定義していない場合は、ifSpecificPに0を返します。
OIDサブツリーの記述は、符号無し32ビットで表したOIDの配列で、最大127個のOIDを記述できます。また、OIDサブツリー記述の終わりには0を設定します。
例：unsigned long「OIDサブツリーの記述」= { 1, 3, 6, 1, 2, 1, 10, 7 ,0 } ・・・ IEEE802.3（1.3.6.1.2.1.10.7）

ifSpeedPには、インタフェースの速度情報を参照する領域のアドレスを返します。ifSpeedPの示す領域の速度情報はリアルタイムに参照されます。速度情報の単位はビット／秒です。インタフェースの速度情報の提供をサポートしていない場合は、ifSpeedPに0を返します。

ifStatPには、インタフェースの現在の状態を参照する領域のアドレスを返します。ifStatPの示す領域のインタフェース状態はリアルタイムに参照されます。

ビット0：接続切断状態（０：DOWN／１：UP）
ビット1：通信回線状態（０：半２重／１：全２重）
ビット2、3：回線速度状態（00：10Base／01：100Base／10：1000Base）EthernetまたはIEEE802.3のみ

インタフェースの現在の状態の提供をサポートしていない場合は、ifStatPに0を返します。

ドライバモジュールが、ifOption、ifTimeFuncP、ifSpecificP、ifSpeedP、ifStatPに不正な値を返した場合、TCP/IPマネージャの正常な動作は保証されません。

6.1.2 setParam 関数 パラメータの設定

C言語インタフェース

W rtcd = setParam(T_DRV_PRMLSTL2 *param);

パラメータ

T_DRV_PRMLSTL2	*param	設定するパラメータリストテーブルのアドレス

リターンパラメータ

W	rtcd	リターン値

パケットの構造

typedef struct { パラメータリスト

H	ifType;	インタフェースのタイプ
UH	mdlStat	モジュールの状態
H	sndPacketMin;	最小送信パケット長
H	sndPacketMax;	最大送信パケット長
H	rcvPacketMin;	最小受信パケット長
H	rcvPacketMax;	最大受信パケット長
FP	**cbrAdr;**	**コールバックルーチンのアドレス**
H	**mdlId;**	**接続した時の接続エントリＩＤ**
H	**macLen;**	**ＭＡＣアドレス情報長**
UB	**macA[MACLEN];**	**ＭＡＣアドレス**
B	sndTime;	送信待ちタイムアウト時間(単位は秒)
UB	prmType;	パラメータリストのタイプ（1固定）
UW	**sndEvPtn;**	**送信イベントフラグパターン**
UW	**rcvEvPtn;**	**受信イベントフラグパターン**
UW	ifOption;	インタフェースのオプション情報
void	*ifTimeFuncP;	インタフェース上の周期起動関数のアドレス
UB	*ifDescrP;	インタフェースの情報を含む文字列のアドレス
UW	*ifSpecificP;	構成するメディアのOIDサブツリーのアドレス
UW	*ifSpeedP;	インタフェースの速度情報領域のアドレス
UW	*ifStatP;	インタフェースの状態格納領域のアドレス

}T_DRV_PRMLSTL2;

注）太字の情報以外は設定しません。

リターン値

0	正常終了
－１	パラメータリスト不正、またはパラメータリスト設定失敗

解説

モジュールにパラメータを設定します。
パラメータの設定に成功した場合は、リターン値として0を返します。

本関数では、cbrAdr、mdlId、macLen、macA、sndEvPtn、rcvEvPtn 以外のパラメータは設定しません。
cbrAdrには、sndReady（パケット送信の受付準備完了通知）および rcvInf（受信パケットの情報通知）のコールバック機能で実行する関数のアドレスが設定されます。

mdlIdには、TCP/IPマネージャが割り当てたドライバモジュールIDが設定されます。
macLenには、設定するMACアドレスの長さ（0または6）が設定されます。
macAには、macLenで示された設定するMACアドレスが設定されます。
sndEvPtnには、送信イベントフラグパターンが設定されます。
rcvEvPtnには、受信イベントフラグパターンが設定されます。

尚、本関数は、ドライバモジュール動作中（startMdlからstopMdlの間）にはコールされません。

6.1.3 startMdl 関数 ドライバモジュールの起動

C言語インタフェース

W rtcd = startMdl (H flgid);

パラメータ

H	flgid	イベント通知用のフラグID

リターンパラメータ

W	rtcd	リターン値

リターン値

0	正常終了
－1	動作開始失敗

解説

ドライバモジュールとハードウェアを初期化し、動作を開始します。

動作の開始に成功した場合、リターン値として0を返します。

setParamを用いてコールバックルーチンのアドレスを設定していない場合は、動作開始失敗として－１を返します。

MACアドレスが必要なシステムでMACアドレスが設定されていない場合は、動作開始失敗として－１を返します。

ハードウェアの初期化に失敗した場合は、動作開始失敗として－１を返します。

6.1.4　stopMdl 関数　　ドライバモジュールの停止

C言語インタフェース

W rtcd = stopMdl (void);

パラメータ

無し

リターンパラメータ

W	rtcd	リターン値

リターン値

0	正常終了
－1	停止失敗

解説

ハードウェアを初期化し、動作を停止します。

動作の停止に成功した場合、リターン値として0を返します。

setParamで設定された情報は全て初期化します。

停止に失敗した場合は、停止失敗として－1を返します。停止に失敗した場合は、以後の動作は保証されません。

6.1.5　sndPacket 関数　　パケットの送信要求

C言語インタフェース

```
W rtcd = sndPacket (UB *sndadr , H sndcnt , H restcnt);
```

パラメータ

UB	*sndadr	送信データの先頭アドレス
H	sndcnt	送信データの長さ
H	restcnt	残りデータの長さ

リターンパラメータ

W	rtcd	リターン値

リターン値

0	正常終了
－1	送信起動失敗

解説

パケットの送信を要求します。
送信の要求処理に失敗した場合は、リターン値として－1を返します。

sndadrには、送信を要求するデータの先頭アドレスが渡されます。尚、ここで渡されるアドレスは、パケットの先頭とは限りません。
sndcntには、送信を要求するデータの長さが渡されます。
restcntには、送信するパケットの残りデータの長さが渡されます。

TCP/IPマネージャでは、1つのパケットを送信する際に1～3回に分けて本関数をコールします。
restcntが0より大きい場合は、現在送信要求中のパケットに続きのデータがあることを示しています。
restcntが0で本関数が呼ばれた場合は1つのパケットのデータ転送が完結したことを示しています。それまでに要求されたデータを1つのパケットとして送信処理を行ってください。

1つのパケットの送信処理が完了して次の送信データを受付けられる状態になったら、setParam関数でcbrAdrに登録されたコールバックルーチンのアドレスを関数としてsndReady（パケット送信の受付準備完了通知）コールバックを発行してください。また、sndReadyコールバック発行に続いて、startMdl関数で受け取ったflgidのイベントフラグIDに対して、setParam関数で受け取ったsndEvPtnのイベントフラグを発行してください。

注）1つのパケットを複数回で送信する場合、送信元のデータは連続しない独立した領域にあります。送信する際は連続したデータからなる1つのパケットとして送信してください。

例として、図 6-1に本関数を使って100バイトずつ3回に分けて送信データを渡す場合を示します。

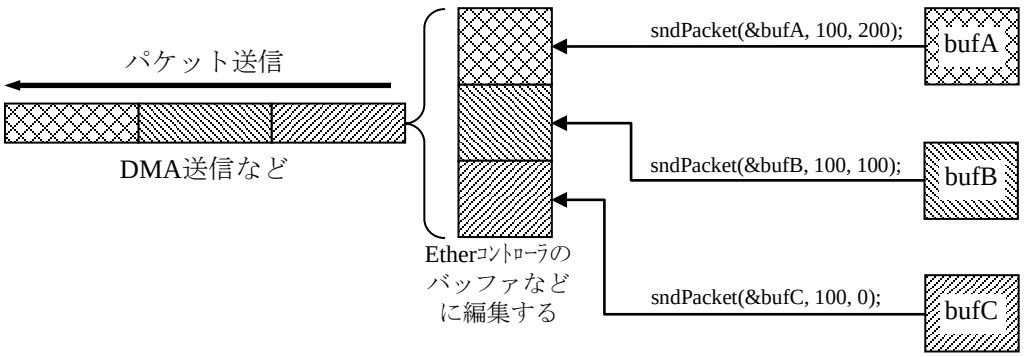


図 6-1　sndPacket関数の使用例

6.1.6 rcvPacket 関数 受信パケットの取得

C言語インタフェース

W rtcd = rcvPacket (UB *bufadr , H *getcnt);

パラメータ

UB	*bufadr	受信パケットを入れる領域のアドレス
H	*getcnt	受信パケットを取り出す長さを格納した領域のアドレス

リターンパラメータ

H	getcnt	取得できた受信パケットの長さ
W	rtcd	リターン値

リターン値

0 または 正の値	未取得の受信パケットデータ長
－1	受信パケット無し

解説

ドライバが受信したパケットのデータを取り出します（rcvInfコールバックで通知された受信パケット）。データを取り出していない受信パケットが無い場合は、リターン値として－1を返します。

bufadrには、取り出したデータを格納する領域のアドレスが渡されます。

getcntの示す領域には、取り出したデータを格納する領域の長さが渡されます。本関数では実際に取り出してコピーできたデータの長さをgetcntの示す領域に設定しなおしてリターンします。

また、コピー先の領域の長さが足りなくて、取り出しきれないデータがある場合[*1]は、リターン値として取り出していない残りのデータ長を返します。取り出していない残りのデータが無い場合は、リターン値として0を返します。

1つの受信パケットの取り出しが完了した後で、まだ取り出していない受信済みのパケットがあった場合は、setParam関数でcbrAdrに登録されたコールバックルーチンのアドレスを関数としてrcvInf（受信パケットの情報通知）コールバックを発行してください。また、rcvInfコールバック発行に続いて、startMdl関数で受け取ったflgidのイベントフラグIDに対して、setParam関数で受け取ったrcvEvPtnのイベントフラグを発行してください。

*1：TCP/IPマネージャの構築時に設定した受信バッファの長さが、ドライバが受信したパケット長よりも短い場合、TCP/IPマネージャは本関数を複数回に分けて連続的にコールし、1つの受信パケットを別々のバッファに分割して取り出します。

6.2　ドライバモジュールコールバック

6.2.1　sndReady コールバック　　パケット送信受付準備完了通知

C言語インタフェース

```
void　cbrAdr (UH cbrfncd , H mdlid , W infcd);
```

パラメータ

UH	cbrfncd	コールバックルーチンの機能コード（sndReady＝0x0001）
H	mdlid	接続エントリＩＤ（パラメータリストのmdlIdに設定された値）
W	infcd	情報コード

リターンパラメータ

無し

情報コード

0	正常終了
－１	前回の送信が失敗した（現バージョンのTCP/IPマネージャでは使用していません）

解説

パケットの送信受付準備の完了を通知します。

本コールバックの呼び出しによって、sndPacket関数によるパケット送信を受付けることができます。

また、本コールバックの呼び出しに続いて、startMdl関数で受け取ったflgidのイベントフラグIDに対して、setParam関数で受け取ったsndEvPtnのイベントフラグを発行してください。

6.2.2　rcvInf コールバック　　受信パケットの情報通知

C言語インタフェース

void　cbrAdr (UH cbrfncd , H mdlid , W infcd);

パラメータ

UH	cbrfncd	コールバックルーチンの機能コード（rcvInf＝0x0002）
H	mdlid	接続エントリＩＤ（パラメータリストのmdlIdに設定された値）
W	infcd	情報コード

リターンパラメータ

無し

情報コード

0 または 正の値　受信したパケットの長さ

解説

パケットを受信した事と受信したパケットの長さを通知します。

本コールバックの呼び出しによって、rcvPacket関数によるパケット受信を受付けることができます。

また、本コールバックの呼び出しに続いて、startMdl関数で受け取ったflgidのイベントフラグIDに対して、setParam関数で受け取ったrcvEvPtnのイベントフラグを発行してください。

TCP/IPマネージャ Ver3.0
リファレンスマニュアル
CM7000TCP03J-19

発行年月　　2014年　8月　　第19版
発　　行　　ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社
編　　集　　ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社