BSDソケットAPI 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

このマニュアルではTCP/IPマネージャのBSDソケットAPIとその使い方および関連事項を説明します。

本BSDソケットAPIはUNIXのBSDソケットと互換を保証するものではありません。本マニュアルを熟読いただき、仕様をよくご理解いただいた上で使用してください。

目次

図表目次

1. 概要

1.1 概略

本プログラムは、TCP/IP マネージャにおいて BSD ソケットをサポートした API 関数を提供します。

1.2 関連するプログラム

本プログラムは、TCP/IP マネージャの ITRON TCP/IP API の上で動作する関数群です。API として BSD ソケット API のみを使う場合でも TCP/IP マネージャ本体は必要になります。

1.3 互換性について

本プログラムは、UNIXのBSDソケットと互換を保証するものではありません。マニュアルを熟読いただき、仕様をよくご理解いただいた上で使用してください。

1.4　構成

本 BSD ソケット API は、TCP/IP マネージャの ITRON TCP/IP API の上で動作する関数群です。TCP/IP プロトコルスタックと BSD ソケット API の構成を図 1-1に示します。

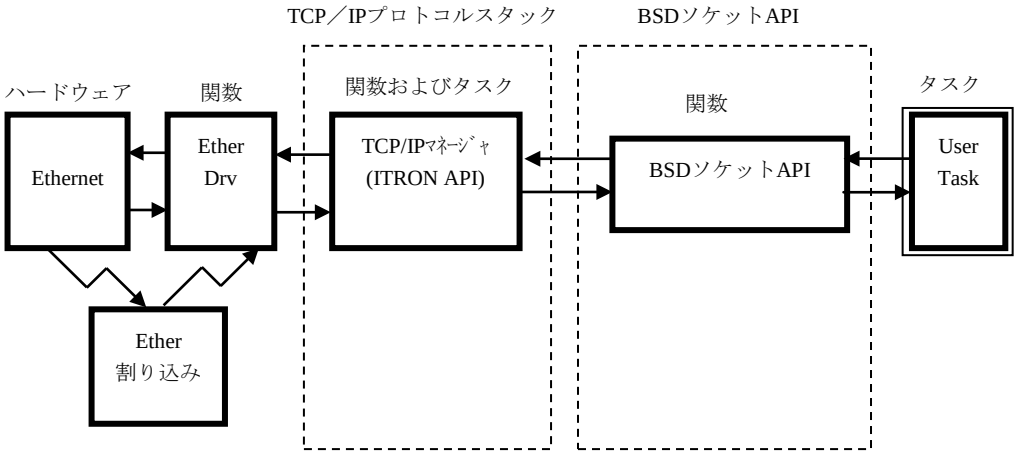


図 1-1　BSDソケットAPI使用時の基本構成

本BSDソケットAPIを使用中であっても、同時にITRON TCP/IP APIを使用することができます。ただし、**BSDソケットAPIが生成した通信端点や受付口を直接ITRON TCP/IP APIで操作した場合は、BSDソケットAPIの正常な動作は保証されません**。ITRON TCP/IP APIを用いて作成したもの以外の通信端点と受付口に対してはITRON TCP/IP APIを用いて操作をしないで下さい。

2. BSD ソケット API

2.1 初期化インタフェース関数

本 BSD ソケット API を使用する際には、まず初期化インターフェイス関数をコールする必要があります。表 2-1に BSD ソケット API の初期化、終了を行う関数を示します。

表 2-1 初期化インタフェース関数

関数種別	関数名	機能
初期化インタフェース	BSD_init	BSDソケットAPI準備
	init_socket	BSDソケットAPI初期化
	stop_socket	BSDソケットAPI終了

2.2 インタフェース関数

本 BSD ソケットには、アプリケーションソフトとのインタフェースとなる関数があります。表 2-2にインタフェース関数を示します。なお、表 2-2の関数の内、closesocket (),selectsocket()は、それぞれソケット識別子のみに対応している close(),select()の機能を持つ関数です。close(),select()はソケット以外（ファイルシステムなど）で使用する関数のため本製品ではサポートしません。

表 2-2 インタフェース関数

関数種別	関数名	機能
インタフェース	socket	ソケット生成
	bind	通信アドレス情報（IPアドレスとポート番号）の指定
	listen	受動モード設定およびコネクション要求数設定
	accept	ソケットに対するコネクションの受け入れ
	connect	ソケットの接続の開始
	getpeername	相手側の通信アドレス情報取得
	getsockname	対象ソケットの通信アドレス情報取得
	getsockopt	プロトコルのオプション取得
	setsockopt	プロトコルのオプション変更
	recv	ソケットからのデータ取得
	recvfrom	データ受信と送信元アドレス取得
	send	データ送信
	sendto	宛て先アドレス指定によるデータ送信
	closesocket	ソケットのクローズ
	shutdown	コネクション閉鎖
	selectsocket	ソケットの利用可能待ち

2.3　拡張インタフェース関数

拡張インタフェース関数では、インタフェース関数で実現していない機能を提供します。
表 2-3に拡張インタフェース関数を示します。

表 2-3　拡張インタフェース関数

関数種別	関数名	機能
拡張インタフェース	get_bsdapi_ver	BSDソケットAPIバージョン情報の参照
	set_blocking_socket	ブロッキング、ノンブロッキングモードの設定
	get_errno	発生した最新のエラー番号の取得
	get_thred_errno	タスク毎で発生した最新のエラー番号の取得
	set_sock_timewait	TIMEWAIT時間の変更
	get_sock_recvlen	受信データ長の取得
	get_socket_cep	ソケットに対する通信端点の参照
	set_sock_keepalive	ソケットに対するキープアライブの設定
	set_mds_mode	ソケットに対するMDSモードの設定

2.4　データの型

本 BSD ソケット API では、ユーザインタフェースで使用する変数の型を次表の様に規定しています。
int 型については対象コンパイラのマニュアルを参照してください。

表 2-4　BSDソケットAPIで使用する変数の型

型名	型	サイズ
int	short または long	符号付き16ビット または 符号付き32ビット
uint8_t	unsigned char	符号無し8ビット
sa_family_t	unsigned char	符号無し8ビット
socklen_t	unsigned long	符号無し32ビット
in_addr_t	unsigned long	符号無し32ビット
in_port_t	unsigned short	符号無し16ビット
size_t	unsigned long	符号無し32ビット
ssize_t	long	符号付き32ビット
ID	short	符号付き16ビット

2.5　エラー番号

インタフェース関数は、エラーが発生したとき異常の種類を示す番号を大域変数 errno に設定します。
本 BSD ソケット API では大域変数 errno として、次の３つの情報を取得することができます。

(1) 各ソケット毎の最新の errno
　　インタフェース関数 getsockopt()の SO_ERROR オプションで取得

(2) BSD ソケット API 全体で発生した最新の errno（***※独自機能***）
　　拡張インタフェース関数 get_errno()で取得

(3) BSD ソケット API を使用している各タスク毎に発生したの最新の errno（***※独自機能***）
　　拡張インタフェース関数 get_thred_errno()で取得

尚、**BSD ソケット API に付属のインクルードファイル（bsdsock.h）では各エラー番号の前に「BSD_」を付けた形で提供しています。**

表 2-5　エラー番号一覧

名称	値	内容
EPERM	1	関数実行不可
EINTR	4	割り込みエラー
EBADF	9	識別子無効
ENOMEM	12	メモリ不足
EACCES	13	アクセス拒否
EFAULT	14	ポインタのアドレス不正
EINVAL	22	引数不正
ENFILE	23	ファイルテーブルオーバーフロー
EPIPE	32	送信もしくは受信が閉じられている
EWOULDBLOCK	35	ノンブロッキングエラー
EINPROGRESS	36	処理開始
EALREADY	37	すでに処理実行中
ENOTSOCK	38	識別子ソケット未指定
EDESTADDRREQ	39	相手側通信アドレス未指定
EMSGSIZE	40	送信データサイズエラー
EPROTOTYPE	41	プロトコル種別異常
ENOPROTOOPT	42	プロトコル利用不可
EPROTONOSUPPORT	43	プロトコル未サポート
EOPNOTSUPP	45	未サポートソケット
EAFNOSUPPORT	47	アドレスファミリ未サポート
EADDRINUSE	48	指定アドレス使用中
EADDRNOTAVAIL	49	指定アドレス利用不可
ECONNRESET	54	コネクションリセット
ENOBUFS	55	利用可能エリア不足
EISCONN	56	接続済みソケット指定
ENOTCONN	57	ソケット未接続
ETIMEDOUT	60	タイムアウトエラー
ECONNREFUSED	61	接続拒否
EPROCLIM	67	プロセス超過
EUSERS	68	ユーザオペレーションの問題
EDEADLK	78	デッドロック条件
EPROTO	86	プロトコルエラー

3. BSD ソケット API での送受信の方法

3.1 ソケットインタフェース

ソケットインタフェースはネットワーク通信に対してソケットを使用します。通信を行う際、アプリケーションプログラムは socket()によりソケットを生成し、生成されたソケットを参照するためのソケット識別子を取得します。

アプリケーションプログラムは connect()によってソケットから外部の終点への TCP コネクションを生成し recv()や send()を使って送受信を行うことができます。

3.2 TCP/IP でのソケットの生成

ソケットの生成は、以下のsocket()関数により行います。

retcode = socket(family, type, protocol);

family : プロトコルファミリ
type : 通信の型
protocol: プロトコル番号
retcode : 0以上 ソケット識別子、 -1 エラー

familyは、ソケットで使用されるプロトコルファミリまたはアドレスファミリを指定します。これは通信アドレス情報が与えられたときどのように解釈するかを指定するものです。本製品では、PF_INET（2）またはAF_INET（2）に対応しています。

typeは通信の型を指定します。本製品では信頼性のあるストリーム型（SOCK_STREAM（1））、コネクションレスのデータグラム型（SOCK_DGRAM（2））に対応しています。TCPの場合はSOCK_STREAM（1）であり、UDPの場合はSOCK_DGRAM（2）です。

protocolは、family、typeの２つのパラメータにより定まるプロトコルが複数存在する場合に、どちらのプロトコルを使用するかを指定するものです。protocolには指定するプロトコルのプロトコル番号（TCPの場合は6、UDPの場合は17）を設定します。このときにfamily、typeにより定まるプロトコル以外のプロトコル番号を設定するとsocket()は-1（エラー）を返します。

protocolにはプロトコル番号以外に0を設定することも可能です。この場合は、family、typeにより定まるプロトコルのうち、仕様によりデフォルトと定められたプロトコルになります。

protocolの取りうる値は、TCPの場合には0またはTCPプロトコルのプロトコル番号(6)であり、UDPの場合には0またはUDPプロトコルのプロトコル番号（17）です。

なお、socket()関数により生成された直後のソケットは自通信アドレスや相手側通信アドレスの情報を持っていません（ポート番号やIPアドレスが指定されていない）。このためソケットを生成したあと、他のインタフェース関数（bind()など）によりこれらの情報を設定する必要があります。

3.3　TCP での通信

TCP におけるソケットの状態遷移を以下の図に示します。

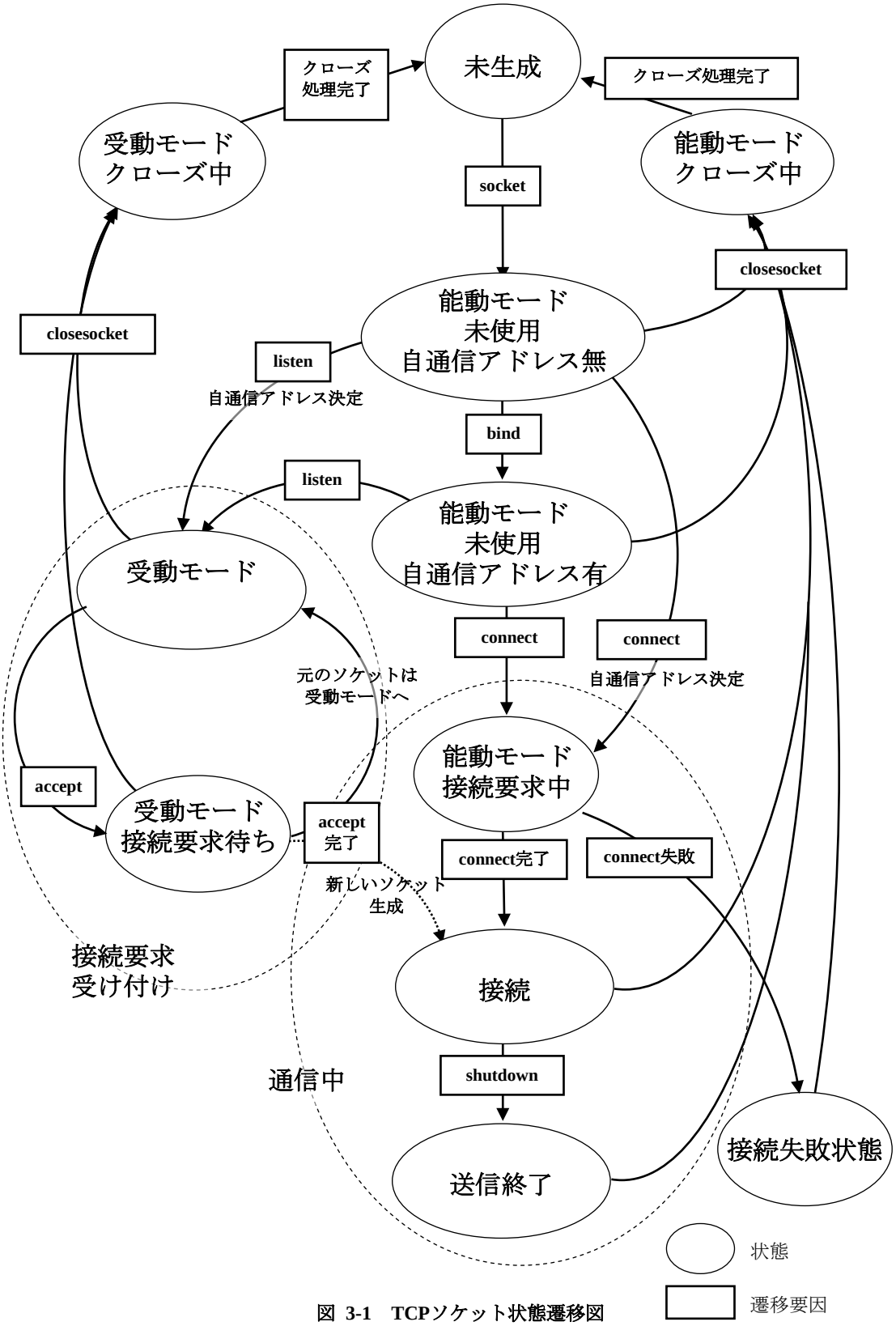
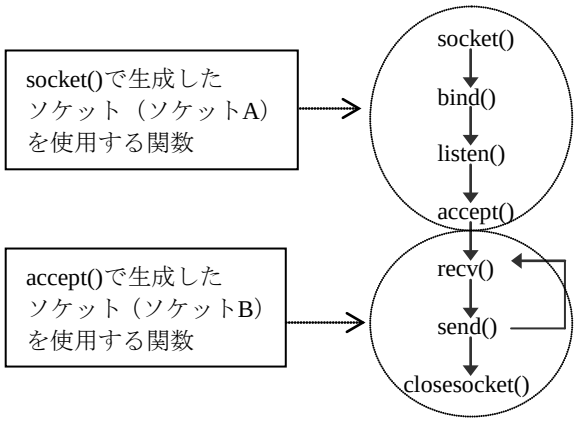


図 3-1　TCPソケット状態遷移図

TCP には自分から相手に接続する場合（能動オープン）と相手からの接続を待つ場合（受動オープン）があります。それぞれの場合における接続の方法と通信の方法について次に示します。

3.3.1 TCP の受動オープン（相手からの接続を待つ場合）

受動オープン時のインタフェース関数呼び出しの例を以下に示します。



TCPを受動オープンするためには、最初にsocket()関数によりTCPプロトコルのソケットを生成します（このsocket()により生成したソケットをソケットAと記します）。

次にbind()関数により、ソケットAに対し自通信アドレス（TCP/IPの場合、ここに自分のIPアドレスとポート番号が含まれます）を指定します。

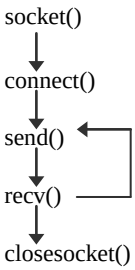
そして、listen()関数により、ソケットAを受動モードにします（つまり、入ってくる接続要求を受け付けるようにします）。また、このとき同時に受け付ける接続要求の数を指定します。

このあと、accept()関数を使って、接続要求してきた相手との接続を行います。接続要求してきた相手がまだいなければ接続要求がくるまで待ちます。接続要求してきた相手がいたなら、新しいソケットを生成し、接続要求した相手と接続します（このaccept()により新しく生成されたソケットをソケットBと記します）。ソケットBは、ソケットAとは別のソケットです。accept()はソケットBのソケット識別子をリターン値として返します。このソケットBのソケット識別子をrecv(),send()等で使用することにより相手との送受信が可能です。

ソケットAは、データの送受信には使用できません。accept()からリターンした後、再度ソケットAにaccept()関数を実行することで、別の新しい接続要求を受け付けることができます。

3.3.2 TCP の能動オープン（自分から相手に接続する場合）

能動オープン時のインタフェース関数呼び出しの例を以下に示します。



TCPを能動オープンするためには、最初にsocket()関数によりTCPプロトコルのソケットを生成します。

その後、必要ならば、bind()関数を使用して自分のIPアドレスとポート番号をソケットに割り付けます。ソケット生成後はconnect()関数により能動オープンを行うことができます。能動オープンでは、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。このとき、すでにbind()関数を使用して自分のIPアドレスとポート番号がソケットに割り付けられている場合は、その値がそのまま使用されますが、まだ割り付け

られていない場合は、システムが選択したIPアドレスとポート番号が割り付けられます。
connect()関数により相手との接続が成功すれば、recv(),send()等によりデータの送受信が可能になります。

3.3.3 TCP のデータ送受信

接続済みのソケットを使ってデータの送受信が可能になります。send(),sendto()等によりデータの送信、recv(),recvfrom()等によりデータの受信を行うことができます。
送信のインタフェース関数では1回の呼び出しで、送信するデータを全て指定することが可能です。
しかし、受信のインタフェース関数では1回の呼び出しで指定したデータ数を全て受信できるとは限りません。受信のインタフェース関数では、受信済みのデータを取得すると関数からリターンするため、受信したいデータ数に達するまで、繰り返し呼び出してください。

> 受信に関して繰り返しが必要なのはTCPプロトコルがブロック指向のプロトコルでないためです。
> TCPプロトコルはストリーム指向のプロトコルであり、送信側が書き込んだバイト列の配送を保証しますが、それらを書き込まれたときと同じ塊で配送することは保証していません。
> 結果として送り側のアプリケーションが1回の送信インタフェース関数の呼び出しでTCPにデータを渡した場合でも、受け取り側のアプリケーションはいくつかの小さな塊のデータを受信するかもしれません。
> あるいは、送り側のアプリケーションがデータを分けて複数回の送信インタフェース関数の呼び出しでTCPにデータを渡した場合でも、大きな1つの塊のデータとして受信するかもしれません。

3.3.4 TCP の通信終了

通信を終了する場合、closesocket()関数により接続を終了しソケットを開放することができます。このときに、ソケットが受信済みでユーザが取得していないデータは破棄されます。そのため、送信のみ終了して受信を継続する場合は、shutdown()関数を使用してください。
closesocket()やshutdown()を使用した場合、未送信のデータを送信後にFINを送ることにより送信の終了を通知します。これにより、相手のアプリケーションは通信の終了を知ることができます。

3.4　UDP での通信

UDP におけるソケットの状態遷移を以下の図に示します。

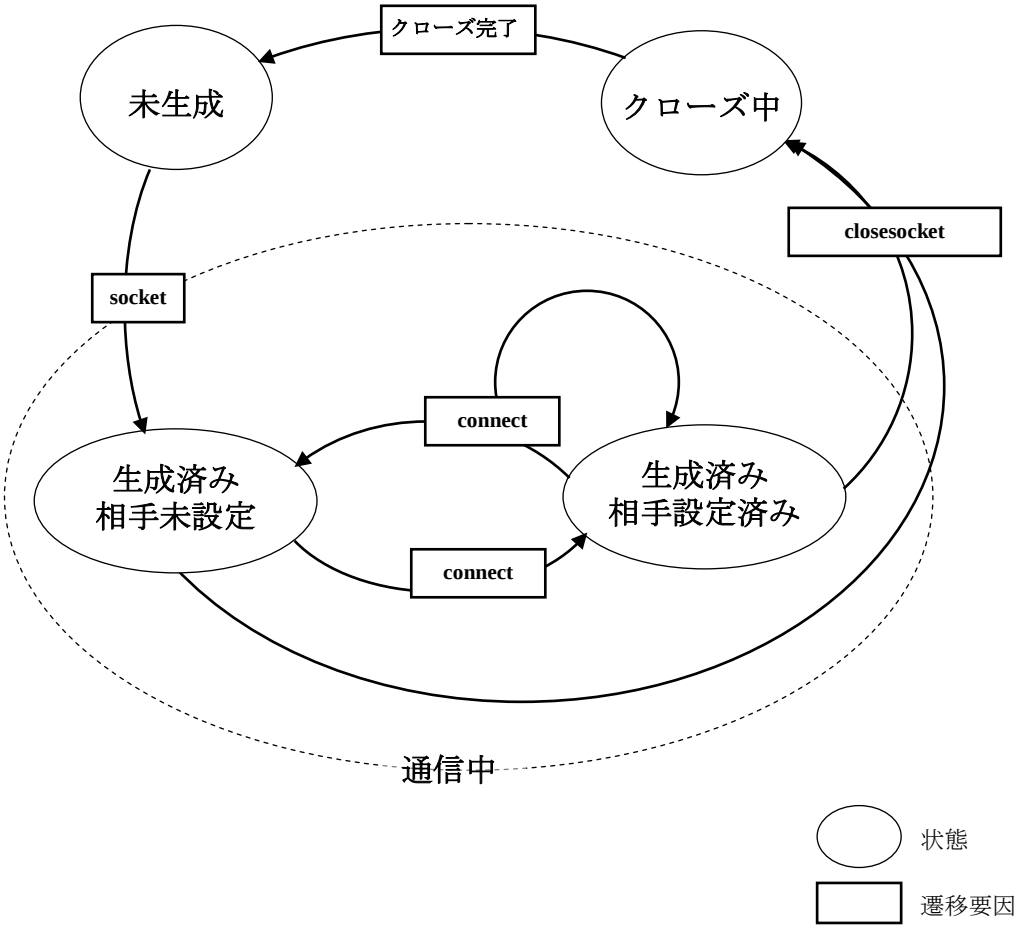


図 3-2　UDPソケット状態遷移図

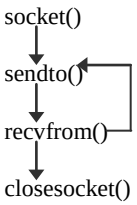
UDP はコネクションレスのプロトコルのため、socket()関数でソケットを生成するだけでデータの送受信が可能になります。

UDP において通信できるソケットには、２つの状態があります。相手側通信アドレスが設定されていない状態（相手未設定）と設定されている状態（相手設定済み）です。

それぞれの場合における接続の方法と通信の方法について次に示します。

3.4.1 UDP の相手未設定のソケット

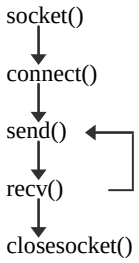
相手が未設定であるソケットのインタフェース関数呼び出しの例を以下に示します。



socket()関数によりUDPプロトコルのソケットを生成します。
通信相手が未設定のため、データを送信するたび相手側通信アドレスを設定する必要があります。このため、送信では相手側通信アドレスを指定するsendto()のみ使用することができます（相手側通信アドレスを指定しないsend()は使用できません）。
受信にはrecv(),recvfrom()の関数を使用します。
送信元の通信アドレスを知るためには、相手側通信アドレスを取得するrecvfrom()関数を使用します。
recv()関数では相手側通信アドレスを取得できないため、どこからのデータなのか判断できません。

3.4.2 UDP の相手設定済みのソケット

相手が設定されているソケットのインタフェース関数呼び出しの例を以下に示します。



socket()関数によりUDPプロトコルのソケットを生成します。
次にconnect()関数により、ソケットに相手側通信アドレスを設定します。
connect()関数の呼び出しが成功すると、相手設定済みのソケットによる送受信が可能になります。
UDPでのconnect()関数の呼び出しでは、単に相手側通信アドレスを記憶します（相手側通信アドレスが有効かどうかの検査は行わないため、実際に送受信を行うまで相手と通信できるかはわかりません）。

送信では、相手側通信アドレスを指定できません。このため、通常は相手先を指定しないsend()関数を使用します。sendto()関数を使用する場合は、相手側通信アドレスを指定せずに使用してください（相手側通信アドレスを指定した場合はエラーが返ります）。

受信では、connect()関数で指定した相手からのみデータを受け付け、他の相手からのデータは受け付けません。受信にはrecv()またはrecvfrom()関数が使用できます。ただし、recvfrom()関数で返ってくる相手側通信アドレスはconnect()関数によって指定した情報と同じです。
なお、相手設定済みのソケットは、connect()関数により相手未設定のソケットに戻すことも可能です。

3.4.3 UDP のデータ送受信

UDPではTCPと異なり、1回の送信インタフェース関数の呼び出しで、データを分割せずに１つのブロックとして転送します。本製品におけるUDPによる送受信データグラムの最大サイズMDS（Maximum Datagram Size）は1472バイト（イーサネットでMTU（Maximum Transmission Unit）が1500のとき）です[*1]。
受信側では１つのブロックを全て受け取れるようにバッファを用意してください。**用意したバッファより受信したデータの方が大きい場合、超過した分のデータは破棄されます**（MDSより大きい受信バッファを準備してもMDSより大きいデータは受信できません）。また、UDPはプロトコルにフロー制御機能を持

っていません。空きの受信バッファキューが無い状態で受信したデータは全て破棄されます。
送信側ではMDSを超えるデータブロックは送信できません。**MDSを超える分のデータは、送信の際に破棄されます**。

*1 TCP/IPマネージャでフラグメント送信が使用可能に設定されている場合は、set_mds_mode()拡張関数を使用することで、（MTU – 28）を超えたサイズのデータグラム送信が可能です。

3.4.4 UDP の通信終了

通信を終了する場合は、closesocket()関数によりソケットを開放します。このときに、ソケットが受信済みでユーザが取得していないデータは破棄されます。そのため、送信のみ終了して受信を継続する場合は、shutdown()関数を使用してください。
なお、UDPの場合はclosesocket()やshutdown()により送信を停止しても、送信の終了は通知されません。

4. BSD ソケット API 関数

BSDソケットAPI関数はユーザアプリケーションがTCP/IPマネージャを利用する場合のインタフェースをソケット形式で提供します。

本節では、初期化インタフェース関数、インタフェース関数、拡張インタフェース関数についての詳細な説明を以下の形式で行っています。

（No.）関数名　　機能

【発行可能なシステム状態*1】

C言語インタフェース

関数の呼出し形式

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
・	・	・
・	・	・

リターン値

リターン値	リターン値の意味
・	・
・	・

パケットの構造

struct {

・・・

・・・

errno(大域変数)

大域変数名*2	errnoの値	エラーの意味*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BSDソケットAPI関数をコールした場合、システムの正常な動作は保証されません。

*2 **付属のインクルードファイル（bsdsock.h）では各エラー番号（大域変数名）の前に「BSD_」が付いています**

***3 エラーコードEFAULT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。**

4.1　初期化インタフェース関数

4.1.1　BSD_init　　BSD ソケット API 初期化準備

【T／D／L／I】

C言語インタフェース

void BSD_init(void);

パラメータ

無し

リターンパラメータ

無し

解説

BSDソケットAPIを使用する準備を行います。

BSDソケットAPIシステムの内部変数を初期化します。BSD_init()はシステム初期化時に1回だけ実行してください。また、BSD_init()をコールする前は他のBSDソケットAPI関数をコールしないでください。

4.1.2 init_socket BSD ソケット API 初期化

【T】

C言語インタフェース

```
unsigned int errcode = init_socket(INITPRM *p_initprm );
```

パラメータ

INITPRM	*p_initprm;	BSDソケットAPI初期化パラメータ

リターンパラメータ

int	retcode	リターン値
unsigned long	p_initprm->ramlen;	使用したメモリの長さ

パケットの構造

```
struct {
        unsigned long       *ramaddr;           使用可能なメモリの先頭アドレス
        unsigned long       ramlen;             使用可能なメモリの長さ
        unsigned short      portstr;            自動割り当てに使用するポート番号の先頭
        unsigned short      portcnt;            自動割り当てに使用するポート番号の数
        unsigned short      tcpactsockcnt;      TCPで接続するソケットの総数
        unsigned short      tcplstsockcnt;      TCPで接続待ちに使用するソケットの総数
        unsigned short      udpsockcnt;         UDPで使用するソケットの総数
        unsigned short      tcpconnectioncnt;   TCPで同時に受け入れ可能なコネクション要求の総数
        unsigned long       rcvwbufmaxlen;      TCPの受信バッファの最大長
        unsigned long       sndwbufmaxlen;      TCPの送信バッファの最大長
        unsigned short      udprcvdgramlen;     UDPの最大受信データグラム長
        unsigned short      udprcvquecnt;       UDPの受信バッファキューのデフォルト数
        unsigned long       ipaddr[IPADDRMAX]; 使用する自分のIPアドレス（IPADDRMAX=4）
        unsigned long       timeinterval;       OSで使用するtmoutの1カウントあたりの時間
                                                をマイクロ秒に換算した値
} INITPRM;
```

リターン値／エラーコード

0		正常終了
EFAULT	14	ポインタアドレス不正（p_initprm, ramaddrが0または4の倍数以外）
EINVAL	22	引数不正 （①portstr、portcntが0またはportstr+portcnt＞65536 ②rcvwbufmaxlen、sndwbufmaxlenの値が範囲外 ③timeinterval＜1000またはtimeinterval＞100000）
ENOMEM	12	メモリ不足（メモリ資源が足りない）

解説

BSDソケットAPIを初期化します。

init_socket()では、TCP/IPマネージャの初期化は行いません。init_socket()をコールする前に、TCP/IPマネージャのman_ip_start()をコールし、TCP/IPマネージャの初期化を実行しておく必要があります。

各パラメータのチェック後に必要なメモリをramaddrの領域から確保し、使用したメモリの長さをramlenに返します。ramlenが必要なメモリサイズに満たない場合は、ramlenに必要なメモリの長さを設定し、リターン値としてENOMEMを返します。

portstr、portcntには、自動割り当てに使用するポート番号の範囲を指定します。なお、使用可能なポート番号は１～65535です。portstr、portcntに0を指定した場合、またはportstr+portcntが65536を超える場合はリターン値としてEINVALを返します。

BSDソケットAPIで使用するポート番号とITRON TCP/IP APIで使用するポート番号は重複しない様にしてください。重複している場合は、bind(),listen(),connect()関数でエラーとなることがあります。

tcpactsockcntには、BSDソケットAPIにおいてTCPで同時に通信するソケットの数を指定します。
tcplstsockcntには、BSDソケットAPIにおいてTCPで同時に他（クライアント）からのコネクション要求待ちを可能にするソケットの数を指定します。
udpsockcntには、BSDソケットAPIにおいてUDPで同時に使用するソケットの数を指定します。
tcpconnectioncntには、BSDソケットAPIにおいてTCPで（サーバとして）同時に受け入れ可能なコネクション要求数（listen()の受け入れ可能なコネクション要求数）の合計を指定します。

それぞれの値は、TCP/IPマネージャのmain_ip_start()において設定した値に対し以下の条件を満たす必要があります。

tcplstsockcnt ＜ （最大TCP受付口ID）－（ITRON TCP/IP APIで使用するTCP受付口数）
udpsockcnt ＜ （最大UDP通信端点ID）－（ITRON TCP/IP APIで使用するUDP通信端点数）
tcpactsockcnt + tcpconnectioncnt ＜
（最大TCP通信端点ID）－（ITRON TCP/IP APIで使用するTCP通信端点数）

rcvwbufmaxlenには、１つあたりのTCPソケットで使用する受信バッファの最大長を指定します。
sndwbufmaxlenには、１つあたりのTCPソケットで使用する送信バッファの最大長を指定します。
rcvwbufmaxlenおよびsndwbufmaxlenが2048未満の場合は、リターン値としてEINVALを返します。
rcvwbufmaxlenおよびsndwbufmaxlenの推奨値は、共に MSS[*1]×2×N です（Nは2 以上の整数）。
なお、各TCPソケットの受信バッファ、送信バッファの長さはsetsockopt()により指定することが可能です。このとき、rcvwbufmaxlen、sndwbufmaxlenで指定した長さを超える指定はできません。

udprcvdgramlenには、UDPソケットが受信するデータグラムの最大長（通常は576から1472）を指定します。
udprcvquecntには、１つあたりのUDPソケットで用意するデータグラム受信キューのデフォルトの数（通常はrecvまたはrecvfromによって取得する前に受信できるデータグラムの数+1）を指定します。
なお、各UDPソケットのデータグラム受信キューの数はsetsockopt()により指定することが可能です。

ipaddrには、BSDソケットAPIにおいて使用する自分のIPアドレスをホストバイトオーダで指定します。
ipaddrはIPADDRMAX(4)の数だけ指定が必要です。ipaddrの未使用の領域には0を指定してください。
ipaddr に指定するIPアドレスは、TCP/IPマネージャにおいて自IPアドレス（ホストバイトオーダ）として登録されている必要があります。

timeintervalには、OSで使用するtmoutの1カウントあたりの時間をマイクロ秒に換算した値を指定します。

*1：MSS (Maximum Segment Size) 最大セグメントサイズ

4.1.3 stop_socket BSD ソケット API 停止

【T】

C 言語インタフェース

```
int retcode = stop_socket ( void );
```

パラメータ

なし

リターンパラメータ

int	retcode	リターン値

リターン値／エラーコード

0		正常終了
EPERM	1	関数実行不可（BSD_init()が実行されていない）
EALREADY	37	既に実行中

解説

接続中のソケットを全て切断し、BSDソケットAPIを停止します。

4.2　インタフェース関数

4.2.1　socket　　ソケット生成

【T】

C言語インタフェース

int retcode = socket(int family, int type, int protocol);

パラメータ

int	family	アドレスファミリ
int	type	サービス型
int	protocol	使用するプロトコル番号

リターンパラメータ

int	retcode	リターン値

リターン値

正の値	正常終了（生成したソケットのソケット識別子）
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
ENFILE	23	ファイルテーブル不足（ソケット数に空きがない）
EPROTONOSUPPORT	43	プロトコル未サポート（アドレスファミリまたはサービスまたはプロトコルが不正）

解説

ソケットを生成し、ソケット識別子を返します。

本ソケットAPI関数においてサポートしているプロトコルは、TCPとUDPのみです。
familyにはAF_INET(2)を指定してください（PF_INET(2)でも可）。
typeには、TCPではSOCK_STREAM(1)、UDPではSOCK_DGRAM(2)を指定してください（SOCK_RAW(3)は指定できません）。
protocolには、TCPでは0またはTCPのプロトコル番号(6)を、UDPでは0またはUDPのプロトコル番号(17)を指定してください。

空きのソケットがなく、ソケットが生成できない場合は、大域変数errnoにENFILEを設定し-1（エラー）を返します。
family、type、protocolが表にない組み合わせの場合は、大域変数errnoにEPROTONOSUPPORTを設定し-1（エラー）を返します。

使用するプロトコルごとのfamily、type、protocolの設定を下の表に示します。

使用するプロトコル	Family	type	Porotocol
TCPプロトコル	AF_INET(2) または PF_INET(2)	SOCK_STREAM(1)	0 TCPプロトコル番号(6)
UDPプロトコル		SOCK_DGRAM(2)	0 UDPプロトコル番号(17)

4.2.2 bind 通信アドレス情報の指定

【T】

C言語インタフェース

```
int retcode = bind(int sockfd, struct sockaddr *p_localaddr, socklen_t addrlen);
```

パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_ localaddr	自分の通信アドレス情報を格納した領域のアドレス
socklen_t	addrlen	自分の通信アドレス情報を格納した領域の長さ

リターンパラメータ

int	retcode	リターン値

パケットの構造

```
struct  in_addr {                       IPアドレスの構造体
        in_addr_t       s_addr;
} ;

typedef  struct  sockaddr_in {          通信アドレス情報の構造体
        uint8_t         sin_len;        構造体の長さ（16バイト固定）
        sa_family_t     sin_family;     アドレスファミリ（AF_INET(2)）
        in_port_t       sin_port;       TCPあるいはUDPのポート番号
        struct in_addr  sin_addr;       IPアドレス（ネットワークバイトオーダー）
        char            sin_zero[8];    未使用
} ;
```

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
ENOMEM	12	メモリ不足（UDPソケットを生成するための受信バッファ資源が足りない）
EFAULT	14	ポインタアドレス不正（p_localaddrが0または4の倍数以外）
EINVAL	22	引数不正 （①指定したソケットはすでに自通信アドレス情報が決定している ②addrlenが構造体struct sockaddr_in型の長さと異なる）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EAFNOSUPPORT	47	アドレスファミリ未サポート（アドレスファミリがAF_INET(2)以外）
EADDRINUSE	48	通信アドレス情報使用中 （①指定した通信アドレス情報において、自動割り当て以外の指定をしたIPアドレスまたはポート番号が、他のソケットにおいて使用中 ②指定した通信アドレス情報において、自動割り当て指定をしたIPアドレスまたはポート番号に空きが無い）
EADDRNOTAVAIL	49	IPアドレス利用不可（指定したIPアドレスが、INITPRM[*1]に登録されていない）
EPROCLIM	67	プロセス超過（TCP/IPマネージャのUDP通信端点が足りない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）

*1：INITPRMについては init_socket()を参照してください。

解説

ソケットに対し自分の通信アドレス情報を割り当てます。

socket()を用いて作成された直後のソケットには、通信アドレス情報が割当てられていません。
bind()はsockfdが指すソケットに、p_localaddrで示す領域の通信アドレス情報を割り当てます。
通信アドレス情報はstruct sockaddr_in型です。この領域のアドレスを関数に渡す際にはstruct sockaddrへのポインタ型にキャストしてください。

指定した通信アドレス情報のうち、IPアドレスまたはポート番号が他のソケットにおいて使用中の場合は、大域変数errnoにEADDRINUSE*2を設定し-1（エラー）を返します。

sockfdが指定したソケットにすでに通信アドレス情報が割り当てられている場合は、通信アドレス情報を割り当てることはできないため、大域変数errnoにEINVALを設定し-1（エラー）を返します。
addrlenが構造体struct sockaddr_in型の長さと異なる場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。
sin_portに0が指定された場合には登録されているポート番号の範囲から未使用のポートを検索して割り当てます。未使用のポートが無い場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。
sin_addrで指定したIPアドレスが登録されていない場合は、大域変数errnoにEADDRNOTAVAILを設定し-1（エラー）を返します。
sin_portに0が指定され、且つ、sin_addrに0が指定された場合には登録されているIPアドレスから未使用のIPアドレスを検索して割り当てます。未使用のIPアドレスが無い場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。
sin_portにポート番号が指定され、且つ、sin_addrに0が指定された場合には、登録されている全てのIPアドレスで指定したポート番号へのTCPの接続待ち、および、UDPの受信待ちが可能です。

TCP/IPマネージャで確保したUDP通信端点に空きがなく処理ができない場合は、大域変数errnoにEPROCLIMを設定し-1（エラー）を返します。

UDPソケットが送受信に使用するバッファが確保できない場合は、大域変数errnoにENOMEMを設定し-1（エラー）を返します。

*2 ソケットオプションSO_REUSEADDRがOFFの時のものです。詳細は、getsockopt()の解説を参照してください。

4.2.3 listen 受動モード設定

【T】

C言語インタフェース

int retcode = listen(int sockfd, int queuelen);

パラメータ

int	sockfd	ソケット識別子
int	queuelen	受け入れ可能なコネクション要求の数

リターンパラメータ

int	retcode	リターン値

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
ENOMEM	12	メモリ不足（ソケット用のバッファが不足している）
EINVAL	22	引数不正 （①queuelen≦0 ②指定したソケットを、受動モードへ変更できない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EOPNOTSUPP	45	未サポートソケット（指定したソケットがTCPのソケットでない）
EADDRINUSE	48	通信アドレス情報使用中 （①登録されている通信アドレス情報のポート番号が、他のソケットにおいて使用中（自動割り当て指定でない場合） ②通信アドレス情報に割り当てるポート番号に空きが無い（自動割り当て指定の場合））
EPROCLIM	67	プロセス超過（TCP/IPマネージャのTCP受付口が足りない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された ③BSDで生成した受付口が他のプロセスによって削除された）
EPROTO	86	プロトコルエラー（使用するIPアドレスが動作していない）

解説

ソケットを受動オープンし、受動モードに設定します。

queuelenには、このソケットが受け付けることができるコネクション要求[*1*2]の最大数を設定します。

queuelenには1以上を指定してください。0以下の値を指定した場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

対象ソケットが下記に示す(1)と(2)の状態以外の場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

(1) socket()で生成した直後のソケット

(2) bind()が成功した直後のソケット

listen()は、TCPのソケットが対象です。指定されたソケットがTCPでない場合は、大域変数errnoにEOPNOTSUPPを設定し-1（エラー）を返します。

ソケットに登録されている通信アドレス情報のポート番号が、他のソケットにおいて既に使用中の場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。

通信アドレス情報のポート番号を自動割り当てに指定し、割り当てるポート番号に空きが無い場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。

ソケットが送受信に使用するバッファが確保できない場合は、大域変数errnoにENOMEMを設定し-1（エラー）を返します。
TCP/IPマネージャで確保したTCP受付口に空きがなく処理が行えない場合は、大域変数errnoにEPROCLIMを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

自分の通信アドレス情報として設定されているIPアドレスがTCP/IPマネージャ上で動作していない場合は、大域変数errnoにEPROTOを設定し-1（エラー）を返します。

*1　受け付けることができるコネクション要求とは、3-WAYハンドシェイクが完了し、accept()されていない状態のコネクション要求[*3]のことを指し、同時に複数のコネクション要求を受け付けできることを指すのではありません[*4]。1つの3-WAYハンドシェイクが完了するまでは、他のSYNを受け付けられません。

*2　受け付けることができるコネクション要求とは、3-WAYハンドシェイクが完了し、accept()されていない状態のコネクション要求のことを指し、ひとつのソケットでaccept()可能な数を制限することを指すのではありません。3-WAYハンドシェイク完了後にaccept()を実行することでlisten()を実行したソケットはまた次のコネクション要求の受け付けを開始します。その際にTCP/IPマネージャのリソースを使用することを十分に理解したうえでご使用願います。

*3　ソケットにaccept()されていないコネクション要求が存在する状態でclosesocket()を実行した場合は、accept()されていないコネクションをRSTによって切断します。状況によっては相手からの接続が成功しているのにRSTされているように見える場合がありますので、listen()を実行したソケットをclosesocket()する場合は、このことを十分に理解したうえでご使用願います。

*4　同時に複数のコネクション要求を受け付ける必要がある場合は、ソケットを複数用意し、bind()する際にsin_addrに0を、sin_portに同一ポート番号を指定し、それぞれのソケットでlisten()を実行してください。ただしこの場合においては、3-WAYハンドシェイクが完了した際にどのソケットでaccept()が可能になるかは保証できないため、listen()中のソケットをselectsocket()で監視し、accept()が可能かを判断したうえでaccept()を実行してください。

4.2.4 accept ソケットに対するコネクションの受入れ

【T】

C言語インタフェース

```
int retcode = accept(int sockfd, struct sockaddr *p_addr, socklen_t *addrlen);
```

パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	接続相手の通信アドレス情報を格納する領域のアドレス
socklen_t	*addrlen	通信アドレス情報の長さを格納した領域のアドレス

リターンパラメータ

int	retcode	リターン値
struct sockaddr_in	p_addr	格納した接続相手の通信アドレス情報
socklen_t	addrlen	格納した接続相手の通信アドレス情報の長さ

パケットの構造

```
struct    in_addr {                          IPアドレスの構造体
            in_addr_t              s_addr;
} ;

typedef   struct   sockaddr_in {             通信アドレス情報の構造体
            uint8_t                sin_len;          構造体の長さ（16バイト固定）
            sa_family_t            sin_family;       アドレスファミリ（AF_INET(2)）
            in_port_t              sin_port;         TCPあるいはUDPのポート番号
            struct in_addr         sin_addr;         IPアドレス（ネットワークバイトオーダー）
            char                   sin_zero[8];      未使用
} ;
```

リターン値

0以上	新しく生成されたソケット識別子（正常終了）
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_addr, addrlenが0または4の倍数以外）
EINVAL	22	引数不正 （①指定したソケットが受動モードでない ②addrlenが構造体struct sockaddr_in型の長さと異なる）
ENFILE	23	ファイルテーブル不足（ソケット数に空きがない）
EWOULDBLOCK	35	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつキュー上に待ち状態のコネクションがない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EOPNOTSUPP	45	未サポートソケット（指定したソケットがTCPのソケットではない）
EPROCLIM	67	プロセス超過（TCP/IPマネージャのTCP通信端点が足りない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された ③BSDで生成した受付口が他のプロセスによって削除された）

解説

確立済みコネクションから接続済みのソケットを生成します。

確立済みコネクションから接続済みのソケットを生成してデータの送受信を可能にした後、p_addrの指し示す領域に接続相手の通信アドレス情報を返します。

p_addrが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

accept()はコネクション待ちキューにある確立済みコネクションを引き出し、確立済みコネクションから接続済みの能動モードのソケットを作成して、そのソケットに新しいソケット識別子を割り当てリターン値として返します。新しく生成された接続済みソケットは、sockfdの示すソケットとは別のソケットです。
キュー上に待ち状態のコネクションがない場合、コネクションができるまでaccept()からリターンしません（ソケットのノンブロッキングモード*1がOFFの場合）。ソケットのノンブロッキングモード*1がONの場合は、大域変数errnoにEWOULDBLOCKを設定しリターン値に-1を返します。
accept()により新しく生成された接続済みのソケットは、能動モードであり、データの送受信のために使用されます。
sockfdが示すソケットは、accept()からリターンしたあとも受動モードのまま存在するため、再びaccept()を実行することが可能です。

accept()は、TCPのソケットが対象です。指定されたソケットがTCPでない場合は、大域変数errnoにEOPNOTSUPPを設定し-1（エラー）を返します。
sockfdの示すソケットは、受動モードかつshutdown()が実行されていない状態である必要があります。この条件を満たしていないソケットが指定された場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。
addrlenが構造体struct sockaddr_in型の長さと異なる場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。
ソケットに空きが無く、新しいソケットが生成できない場合は、大域変数errnoにENFILEを設定し-1（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

*1：ノンブロッキングモードとそのON/OFFの切り替えについては4.3.2（set_blocking_socket()の項）を参照してください。

4.2.5 connect ソケットの接続の開始

【T】

C言語インタフェース

```
int retcode = connect(int sockfd, struct sockaddr *p_addr, socklen_t addrlen);
```

パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	相手通信アドレス情報を格納する領域のアドレス
socklen_t	addrlen	相手通信アドレス情報の長さを格納した領域のアドレス

リターンパラメータ

int	retcode	リターン値

パケットの構造

```
struct   in_addr {                          IPアドレスの構造体
            in_addr_t           s_addr;
} ;

typedef  struct   sockaddr_in {             通信アドレス情報の構造体
            uint8_t             sin_len;        構造体の長さ（16バイト固定）
            sa_family_t         sin_family;     アドレスファミリ
            in_port_t           sin_port;       TCPあるいはUDPのポート番号
            struct in_addr      sin_addr;       IPアドレス（ネットワークバイトオーダー）
            char                sin_zero[8];    未使用
} ;
```

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
ENOMEM	12	メモリ不足（ソケット用のバッファが不足している）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_addrが0または4の倍数以外）
EINVAL	22	引数不正 （①指定したソケットが受動モード ②指定したソケットが接続失敗状態 ③addrlenが構造体struct sockaddr_in型の長さと異なる）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
EINPROGRESS	36	処理開始（ノンブロッキング設定で、かつ接続がすぐに完了しない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EAFNOSUPPORT	47	アドレスファミリ未サポート （①TCPにおいてsin_family≠AF_INET(2) ②UDPにおいてsin_family≠AF_INET(2)かつ sin_family≠AF_UNSPEC(0)）
EADDRINUSE	48	通信アドレス情報割り当て不可（自分のポート番号に空きがない）
EADDRNOTAVAIL	49	通信アドレス情報利用不可（相手のIPアドレスまたはポート番号が0）
EISCONN	56	接続済みソケット指定
ETIMEDOUT	60	タイムアウトエラー（プロトコルがコネクション確立前に75秒経過）
ECONNREFUSED	61	接続失敗（コネクションが相手によって拒絶された）
EPROCLIM	67	プロセス超過（TCP/IPマネージャのUDP通信端点が足りない）

EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された）
EPROTO	86	プロトコルエラー（使用するIPアドレスが動作していない）

解説

通信相手とのコネクションを確立します。

p_addrが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

ソケットが送受信に使用するバッファが確保できない場合は、大域変数errnoにENOMEMを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

自分の通信アドレス情報として設定されているIPアドレスがTCP/IPマネージャ上で動作していない場合は、大域変数errnoにEPROTOを設定し-1（エラー）を返します。

次にプロトコルごとの説明を行います。

(1) TCPのソケット

TCPのソケットの場合には、connect()は３ウェイハンドシェイクを使ってp_addrの示す相手に対しコネクションを確立します。コネクションの確立に成功すると、0（正常終了）を返します。

コネクションの確立に失敗した場合、そのソケットは使用不可になり、同じソケットに対し再度connect()が呼び出された場合は必ず異常終了となります。

使用可能なIPアドレスやポート番号に空きがない場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。

コネクションが確立しないで一定時間（75秒）経過した場合は、大域変数errnoにETIMEDOUTを設定し-1（エラー）を返します。

コネクションが相手によって拒絶された場合は、大域変数errnoにECONNREFUSEDを設定し-1（エラー）を返します。

ノンブロッキングモードがOFFの場合、コネクションが確立するかエラーが発生するまでは接続中（TCP/IPマネージャによる接続を待つ）の状態となりリターンしません。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

ノンブロッキングモードがONの場合は、大域変数errnoにEINPROGRESSを設定し-1（エラー）を返しますが異常ではありません。対象となっていたソケットはコネクションを開始し、接続中の状態になります。

コネクションが終了する前に再度connect()を実行した場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

コネクションが正常に終了した後に再度connect()を実行した場合は、大域変数errnoにEISCONNを設定し-1（エラー）を返します。

(2) UDPのソケット

UDPのソケットの場合には相手側の通信アドレス情報を記録し、すぐにリターンします。これにより、UDPソケットを相手設定済みの状態にします。

UDPのソケットの場合、同じソケットに対し何度もconnect()が呼び出せます。このとき、別の相手側通信アドレスを設定すると、登録してある相手側通信アドレスが変更されます。

相手側通信アドレスのp_addrにNULLを、addrlenに0を設定しconnect()を呼び出した場合は、大域変数errnoにEADDRNOTAVAILを設定し-1（エラー）を返しますが異常ではありません。この場合、ソケットは相手未設定の状態になります。

送信側および受信側が共にshutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

TCP/IPマネージャで確保したUDPの通信端点に空きがなく処理が行えない場合は、大域変数errnoにEPROCLIMを設定し-1（エラー）を返します。

4.2.6 getpeername 相手側通信アドレス情報取得

【T】

C言語インタフェース

```
int retcode = getpeername(int sockfd, struct sockaddr *p_remaddr, socklen_t *addrlen);
```

パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_remaddr	相手通信アドレス情報を格納する領域
socklen_t	*addrlen	相手通信アドレス情報を格納する領域の長さ

リターンパラメータ

int	retcode	リターン値
struct sockaddr_in	p_remaddr	相手側の通信アドレス情報
socklen_t	addrlen	相手側の通信アドレス情報の長さ

パケットの構造

```
struct  in_addr {                       IPアドレスの構造体
        in_addr_t           s_addr;
};

typedef  struct  sockaddr_in {          通信アドレス情報の構造体
        uint8_t             sin_len;        構造体の長さ（16バイト固定）
        sa_family_t         sin_family;     アドレスファミリ
        in_port_t           sin_port;       TCPあるいはUDPのポート番号
        struct in_addr      sin_addr;       IPアドレス（ネットワークバイトオーダー）
        char                sin_zero[8];    未使用
};
```

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_remadd, addrlenが0または4の倍数以外）
EINVAL	22	引数不正（*addrlenが構造体struct sockaddr_in型の長さより小さい）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ENOTCONN	57	ソケット未接続 （①TCPにおいて通信相手と接続されていない ②UDPにおいて通信相手が設定されていない）

解説

接続されているソケットの相手側通信アドレス情報を取得します。

p_remaddrが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

対象ソケットが接続されておらず、相手がいない場合は、大域変数errnoにはENOTCONNを設定し-1（エラー）を返します。

4.2.7 getsockname 自通信アドレス情報取得

【T】

C言語インタフェース

```
int retcode = getsockname(int sockfd, struct sockaddr *p_addr, socklen_t *addrlen);
```

パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	自通信アドレス情報を格納する領域
socklen_t	*addrlen	自通信アドレス情報の長さを格納する領域の先頭アドレス

リターンパラメータ

int	retcode	リターン値
struct sockaddr_in	p_addr	指定ソケットの通信アドレス情報
socklen_t	addrlen	指定ソケットの通信アドレス情報の長さ

パケットの構造

```
struct  in_addr {                       IPアドレスの構造体
        in_addr_t           s_addr;
} ;

typedef  struct  sockaddr_in {          通信アドレス情報の構造体
        uint8_t             sin_len;        構造体の長さ（16バイト固定）
        sa_family_t         sin_family;     アドレスファミリ
        in_port_t           sin_port;       TCPあるいはUDPのポート番号
        struct in_addr      sin_addr;       IPアドレス（ネットワークバイトオーダー）
        char                sin_zero[8];    未使用
} ;
```

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_addr, addrlenが0または4の倍数以外）
EINVAL	22	引数不正（*addrlenが構造体struct sockaddr_in型の長さより小さい）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）

解説

ソケット識別子が示すソケットの通信アドレス情報を取得します。

p_addrが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

指定のソケットに通信アドレス情報が設定されていない場合でも、正常終了になります。ただし、p_addrに設定するパラメータの値は保証しません。

4.2.8　recv　　ソケットからの受信データ取得

【T】

C言語インタフェース

ssize_t retcode = recv(int sockfd, void *buffer, size_t length, int flags);

パラメータ

int	sockfd	ソケット識別子
void	*buffer	受信データを格納する領域の先頭アドレス
size_t	length	受信データを格納する領域の長さ
int	flags	受信動作を指定するフラグ

リターンパラメータ

ssize_t	retcode	リターン値
char	buffer[]	ソケットからの受信データ

リターン値

0または正の値	正常終了（受信データの長さ）
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
EBADF	9	識別子無効（sockfd＜0）
EINVAL	22	引数不正 （①指定したソケットが受動モード ②length＝0またはlength≧0x80000000 ③flagsが不正）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
EWOULDBLOCK	35	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつ受信済みのデータがない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ECONNRESET	54	コネクションリセット（コネクションがリセットされた）
ENOTCONN	57	未接続エラー（sockfdの示すソケットが未接続）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された）

解説

ソケットからデータを受信します。

ソケット内に読み取られていない受信データがありその長さがlengthより小さい場合は、全てのデータを取り出してその取り出したデータの長さを返します。lengthより大きい場合は、length分のデータを取り出してその長さを返します。

ノンブロッキングモードがOFFの場合は、読み取られていない受信データが無いときは、データを受信するかエラーが発生するまでリターンしません。

ノンブロッキングモードがONで受信データがない場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。この場合は異常ではありません。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。

指定したソケットの受信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

指定したソケットがTCPで既にFINを受信しており、かつ受信バッファが空の場合は、正常終了として受信データ長0を返します（FINを受信しても受信データがあるうちは正常に受信できます）。

指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

TCPソケットの受信待ち状態でFINを受信し、かつ受信データが無い場合は、正常終了として受信データ長0を返します。
TCPソケットの受信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。
他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。また、値の論理和を設定することで２つ以上の動作を指定することができます。
flagsに不正な値が設定された場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

表 4-1　recv(),recvfrom()でflagsに設定する値

define名	値	内容	TCP	UDP
MSG_PEEK	0x0002	バッファリングデータを保持したままで読み込み実行	○	○
MSG_DONTWAIT	0x0040	1回の操作毎のノンブロッキング指定	○	○
MSG_WAITALL	0x0100	全データ到着待ち	○	×

注意：MSG_PEEKとMSG_WAITALLは同時に設定できません

MSG_PEEK　　：受信バッファの状態とデータを変化させずに、受信データの内容を取り出します。
MSG_PEEKが設定されていない場合は、受信したデータ分、受信バッファの内容が変化します。
MSG_PEEKが設定されている場合は、受信バッファの状態とデータを変化させません（再度、受信を実行したとき、同じデータを読み込みます）。
TCPの場合は、再度受信を実行するとデータの長さが増える場合があります。
UDPの場合は、再度受信を実行してもまったく同じデータを取得します。

MSG_DONTWAIT ：ノンブロッキングモードONとして動作します。
MSG_DONTWAITが設定されていない場合は、ソケット自身のノンブロッキングモードに従って動作します。
MSG_DONTWAITが設定されている場合は、関数内の処理はノンブロッキングモードONとして動作します。

MSG_WAITALL　：要求された長さ分のデータを受信します。
MSG_WAITALLが設定されていない場合は、受信データがあればその長さがlengthより小さい場合でもデータを取り出してリターンします。
MSG_WAITALLが設定されている場合は、要求された長さ分のデータを受信するまでリターンしません。ただし、次の場合は要求より少ないデータでリターンすることがあります。
(1) コネクションが切断された場合
(2) ソケットにエラーが発生した場合

4.2.9 recvform 受信データと送信者アドレス取得

【T】

C言語インタフェース

```
ssize_t retcode =
    recvform(int sockfd, void *buffer, size_t buflen, int flags, struct sockaddr *p_from, socklen_t *fromlen);
```

パラメータ

int	sockfd	ソケット識別子
void	*buffer	受信データを格納した領域の先頭アドレス
size_t	buflen	受信データを格納した領域の長さ
int	flags	各種の特殊な受信動作を指定するフラグ
struct sockaddr	*p_from	送信者のアドレス情報を格納する領域
socklen_t	*fromlen	送信者のアドレス情報を格納する領域の長さを格納した領域の先頭アドレス

リターンパラメータ

ssize_t	retcode	リターン値
char	buffer[]	ソケットからの入力データ（配列）
struct sockaddr_in	p_from	送信者のアドレス情報
socklen_t	fromlen	送信者のアドレス情報の長さ

パケットの構造

```
struct  in_addr {                       IPアドレスの構造体
        in_addr_t           s_addr;
} ;

typedef  struct  sockaddr_in {          通信アドレス情報の構造体
        uint8_t             sin_len;        構造体の長さ（16バイト固定）
        sa_family_t         sin_family;     アドレスファミリ
        in_port_t           sin_port;       TCPあるいはUDPのポート番号
        struct in_addr      sin_addr;       IPアドレス（ネットワークバイトオーダー）
        char                sin_zero[8];    未使用
} ;
```

リターン値

0以上	bufferに設定した受信データの長さ（正常終了）
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_fromまたはfromlenが4の倍数以外、またはfromlenのみが0）
EINVAL	22	引数不正 （①指定したソケットが、受動モード ②fromlenが構造体struct sockaddr_in型の長さと異なる ③buflen＝0またはbuflen≧0x80000000 ④flagsが不正）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
EWOULDBLOCK	35	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつ受信済みのデータがない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ECONNRESET	54	コネクションリセット（コネクションがリセットされた）

ENOTCONN	57	未接続エラー（sockfdの示すソケットが未接続）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された）

解説

受信したデータと、その送信者の通信アドレス情報を取得します。

recvform()は相手未設定のUDPソケットを対象としていますが、相手設定済みのUDPソケットやTCPソケットの場合にも使用できます。

p_fromが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。
p_fromの示す領域には送信者の通信アドレス情報を、fromlenの示す領域には送信者の通信アドレス情報の長さを返します。
p_fromにNULLを指定するとrecv()と同じ動作を行います。その場合は、*p_fromおよび*fromlenの領域には設定を行いません。

p_fromがNULL以外でfromlenが構造体struct sockaddr_in型の長さと異なる場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

ソケット内に読み取られていない受信データがありその長さがlengthより小さい場合は、全てのデータを取り出してその取り出したデータの長さを返します。lengthより大きい場合は、length分のデータを取り出してその長さを返します。
ノンブロッキングモードがOFFの場合は、読み取られていない受信データが無いときは、データを受信するかエラーが発生するまでリターンしません。
ノンブロッキングモードがONで受信データがない場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。この場合は異常ではありません。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。
指定したソケットの受信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。
指定したソケットがTCPで既にFINを受信しており、かつ受信バッファが空の場合は、正常終了として受信データ長0を返します（FINを受信しても受信データがあるうちは正常に受信できます）。
指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

TCPソケットの受信待ち状態でFINを受信し、かつ受信データが無い場合は、正常終了として受信データ長0を返します。
TCPソケットの受信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。
他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。詳細については、recv()を参照してください。

4.2.10　send　　ソケットへの送信データ設定

【T】

C言語インタフェース

ssize_t retcode = send(int sockfd, void * buffer, size_t len, int flags);

パラメータ

int	sockfd	ソケット識別子
void	*buffer	送信データを格納した領域の先頭アドレス
size_t	len	送信データの長さ
int	flags	各種の特殊な送信動作を指定するフラグ

リターンパラメータ

ssize_t	retcode	リターン値

リターン値

0以上	送信用ウインドウバッファに書き込んだ送信データの長さ（正常終了）
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
EBADF	9	識別子無効（sockfd＜0）
ENOMEM	12	メモリ不足（UDPソケットを生成するための受信バッファ資源が足りない）
EACCES	13	許可が無い（SO_BROADCASTがOFFで相手アドレスにブロードキャストを指定）
EINVAL	22	引数不正（①len＝0またはlen≧0x80000000 ②flagsが不正）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
EWOULDBLOCK	35	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつ送信バッファに空きがない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EDESTADDRREQ	39	相手側通信アドレス情報未指定（sockfdの示すソケットが相手未設定のUDPソケット）
EADDRINUSE	48	通信アドレス情報使用中（通信アドレス情報のポート番号が他のソケットにおいて使用されている）
ECONNRESET	54	コネクションリセット（コネクションがリセットされた）
ENOBUFS	55	利用可能エリア不足（UDPの送信バッファが確保できない）
ENOTCONN	57	未接続エラー（sockfdの示すソケットが未接続）
EPROCLIM	67	プロセス超過（TCP/IPマネージャのUDP通信端点が足りない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件（①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された）

解説

ソケットを用いてデータを送信します。

TCPでノンブロッキングモードがOFFの場合は、送信用ウインドウバッファに対しlenの長さ分のデータを書き込み、書き込んだ長さを返します。lenの長さ分のデータ全てを書き込む領域がない場合は、lenの長さ分のデータ全てを書き込むかエラーが発生するまで内部で待ち状態となり、リターンしません。

TCPでノンブロッキングモードがONの場合は、送信用ウインドウバッファに１バイトでも書き込めた場合は、正常終了として書きこんだバイト数を返します。送信用ウインドウバッファに空きがなく、１バイトも書き込めなかった場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。
送信用ウインドウバッファに書き込まれたデータはTCP/IPマネージャによって送信されます。

sockfdに相手未設定のUDPソケットを指定した場合は、大域変数errnoにEDESTADDRREQを設定し-1（エラー）を返します。
UDPでは送信用バッファにlenの長さ分のデータを書き込み、書き込んだ長さを返します。また、送信データが１回で送信可能なデータサイズより大きくて全てを書き込めない場合は、書き込める分のデータを書き込み、書き込んだ長さを返します**(注1)**。いずれの場合も書き込んだ分のデータは送信します。
UDPのソケットの場合、ノンブロッキングモードOFFでもブロックすることはありません。
UDPでノンブロッキングモードがOFFのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにENOBUFSを設定し-1（エラー）を返します。
UDPでノンブロッキングモードがONのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します
指定したソケットの送信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。
指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

TCPソケットの送信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。
他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。ただし、本BSDソケットAPIではMSG_DONTWAITのみをサポートしています。
flagsに不正な値が設定された場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

表 4-2　send(),sendto()でflagsに設定する値

define名	値	内容	TCP	UDP
MSG_DONTWAIT	0x0040	1回の操作毎のノンブロッキング指定	○	○

MSG_DONTWAIT ：ノンブロッキングモードONとして動作します。
MSG_DONTWAITが設定されていない場合は、ソケット自身のノンブロッキングモードに従って動作します。
MSG_DONTWAITが設定されている場合は、関数内の処理はノンブロッキングモードONとして動作します。

注1：EMSGSIZE(40)のエラーは発生しません。また、要求したブロックの全データをIPで分割して送信することもありません。

4.2.11　sendto　　送信データと送信先アドレス設定

【T】

C言語インタフェース

```
ssize_t retcode = sendto(int sockfd, void *buffer, size_t msglen, int flags, struct sockaddr *p_to, int tolen);
```

パラメータ

int	sockfd	ソケット識別子
void	*buffer	送信データを格納した領域のポインタ
size_t	msglen	送信データの長さ
int	flags	各種の特殊な送信動作を指定するフラグ
struct sockaddr	*p_to	送信先の通信アドレス情報を格納した領域の先頭アドレス
int	tolen	送信先の通信アドレス情報を格納した領域の長さ

リターンパラメータ

ssize_t	retcode	リターン値

パケットの構造

```
struct  in_addr {                       IPアドレスの構造体
        in_addr_t       s_addr;
} ;

typedef  struct  sockaddr_in {          通信アドレス情報の構造体
        uint8_t         sin_len;        構造体の長さ（16バイト固定）
        sa_family_t     sin_family;     アドレスファミリ
        in_port_t       sin_port;       TCPあるいはUDPのポート番号
        struct in_addr  sin_addr;       IPアドレス（ネットワークバイトオーダー）
        char            sin_zero[8];    未使用
} ;
```

リターン値

0以上	送信用ウインドウバッファに書き込んだ送信データの長さ（正常終了）
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー（待ち状態が強制的に解除された）
EBADF	9	識別子無効（sockfd＜0）
ENOMEM	12	メモリ不足（UDPソケットを生成するための受信バッファ資源が足りない）
EACCES	13	許可が無い（SO_BROADCASTがOFFで相手アドレスにブロードキャストを指定）
EINVAL	22	引数不正 （①msglen＝0またはmsglen≧0x80000000 ②flagsが不正 ③tolenが構造体struct sockaddr_in型の長さと異なる）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
EWOULDBLOCK	35	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつ送信バッファに空きがない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EDESTADDRREQ	39	相手側通信アドレス情報未指定（sockfdの示すソケットが相手未設定のUDPソケット、かつ指定した相手側通信アドレスが不正）

EADDRINUSE	48	通信アドレス情報使用中（通信アドレス情報のポート番号が他のソケットにおいて使用されている）
ECONNRESET	54	コネクションリセット（コネクションがリセットされた）
ENOBUFS	55	利用可能エリア不足（UDPの送信バッファが確保できない）
EISCONN	56	接続済みソケット指定
ENOTCONN	57	未接続エラー（sockfdの示すソケットが未接続）
EPROCLIM	67	プロセス超過（TCP/IPマネージャのUDP通信端点が足りない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （①BSDで生成した通信端点またはポートを他のプロセスが使用中 ②BSDで生成した通信端点が他のプロセスによって削除された）

解説

指定された通信アドレスに対し送信を行います。

送信先を指定する場合、p_toが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

送信先を指定しない場合、p_toにNULLを指定すると、send()と同じ動作を行います。

p_toがNULL以外でtolenが構造体struct sockaddr_in型の長さと異なる場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

(1) sockfdが相手未設定のUDPソケットを指している場合

送信用バッファにmsglenの長さ分のデータを書き込み、書き込んだ長さを返します。また、送信データが 1 回で送信可能なデータサイズより大きくて全てを書き込めない場合は、書き込める分のデータを書き込み、書き込んだ長さを返します**(注1)**。いずれの場合も書き込んだ分のデータは送信を行います。

ノンブロッキングモードがOFFのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにENOBUFSを設定し-1（エラー）を返します。

ノンブロッキングモードがONのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。

送信先通信アドレス情報（p_to）を指定しない（NULL）場合は、大域変数errnoにEDESTADDRREQを設定し-1（エラー）を返します。

(2) sockfdが相手設定済みのUDPソケットまたはTCPソケットを指している場合

送信先通信アドレス情報（p_to）を指定しない（NULL）場合は、send()と同様の動作を行います。

送信先通信アドレス情報（p_to）を指定した（NULL以外）場合は、大域変数errnoにEISCONNを設定し-1（エラー）を返します。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します

指定したソケットの送信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

TCPソケットの送信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoに

EDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。詳細については、send()を参照してください。

注1：EMSGSIZE(40)のエラーは発生しません。また、要求したブロックの全データをIPで分割して送信することもありません。

4.2.12 closesocket ソケットのクローズ

【T】

C言語インタフェース

```
int retcode = closesocket( int sockfd );
```

パラメータ		
int	sockfd	ソケット識別子

リターンパラメータ		
int	retcode	リターン値

リターン値	
0	正常終了
-1	エラー

errno(大域変数)		
EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EWOULDBLOCK	35	ノンブロッキングリターン （ノンブロッキングモードでかつSO_LINGERがONの場合に、リンガー時間が経過した）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）

解説

通信を終了し、ソケットを閉じます。

送信バッファ内に残っているデータは全て送信し、未読の受信データは破棄します。
TCPソケットの場合は、送信バッファ内のデータを全て送信した後にFINを送ります。
送信データ,受信データに対する処理の終了後、ソケットを未生成の状態にします。

SO_LINGERオプションがOFFの場合は、クローズ処理を開始後直ちに、0（正常終了）を返します。

SO_LINGERオプションがONの場合は、リンガー時間が経過するまでclosesocket()からリターンしません。リンガー時間が経過した場合、ノンブロッキングモードがOFFの場合は、0（正常終了）を返しますが、ノンブロッキングモードがONの場合は、大域変数errnoにEWOULDBLOCK設定し-1（エラー）を返します。
TCPソケットではSO_LINGERオプションがONのとき、リンガー時間に0を指定した場合と、リンガー時間が経過した場合には、送信バッファ内に残っているデータを破棄してRSTを送信します。

4.2.13 shutdown コネクション閉鎖

【T】

C言語インタフェース

```
int retcode = shutdown(int sockfd, int direction);
```

パラメータ

int	sockfd	ソケット識別子
int	direction	切断方向

リターンパラメータ

int	retcode	リターン値

リターン値

0	正常終了
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EINVAL	22	引数不正（directionの値が対象外）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ENOTCONN	57	ソケット未接続
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）
EDEADLK	78	デッドロック条件 （BSDで生成した通信端点が他のプロセスによって削除された）

解説

コネクションを部分的に閉鎖します。

切断方向ごとの動作は以下になります。接続していないソケットに使用した場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。

directionがSHUT_RD（0）の場合：

コネクションの読み出し側をクローズし、受信バッファ内のデータやその後到着したデータを破棄します。TCPソケットの場合、到着データに対しては応答（ACK）を返しますが、データ自体は破棄します。

directionがSHUT_WR（1）の場合：

コネクションの書き込み側をクローズし、それ以後データ送信を行いません。送信バッファに残っているデータがあれば、そのデータは全て送信します。TCPソケットの場合、送信バッファ内データを全て送信した後にFINを送信します。

directionがSHUT_RDWR（2）の場合：

コネクションの読み出し側、書き込み側の双方をクローズし、それ以後、データの送受信を行えなくします。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

4.2.14 getsockopt プロトコルのオプション取得

【T】

C言語インタフェース

int retcode = getsockopt(int sockfd, int level, int opt, void *p_optval, int *optlen);

パラメータ

int	sockfd	ソケット識別子
int	level	プロトコルレベル
int	opt	オプション
void	*p_optval	オプションを格納する領域の先頭アドレス
int	*optlen	オプションを格納する領域の長さを格納する領域の先頭アドレス

リターンパラメータ

int	retcode	リターン値
void	p_optval	オプションデータ
int	optlen	オプションデータの長さ

リターン値

0	正常終了
-1	エラー

errno(大域変数)

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_optval, optlenが0または4の倍数以外）
EINVAL	22	引数不正（optlenが短い）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ENOPROTOOPT	42	プロトコル利用不可（level、optの組み合わせが不正）

解説

プロトコルに対するオプションの値を読み出します。

ソケットから取得したオプションデータをp_optvalの示す領域に返し、オプションデータの長さをoptlenに返します。

オプションには、特定の機能の有効／無効を設定、取得するオプション（フラグ型オプション）と、特定の値の設定、取得を行うオプション（数値型オプション）の2種類に分けられます。
フラグ型オプションの場合は、オプションデータが0ならオプションが無効になっていることを示し、0以外ならオプションが有効になっていることを示します。

表 4-3に、本BSDソケットAPIにおいて使用するソケットオプションを示します。
levelとoptの組み合わせが正しくない場合は、大域変数errnoにENOPROTOOPTを設定し-1（エラー）を返します。

表 4-3　ソケットオプション一覧

level	Opt	使用可能な関数	内容	オプションの型	データ型
SOL_SOCKET (0xffff)	SO_REUSEADDR（0x0004）	getsockopt setsockopt	自通信アドレスの再利用の許可	フラグ型	int型
	SO_KEEPALIVE（0x0008）	getsockopt setsockopt	コネクションが有効か定期的な検査の実行	フラグ型	int型
	SO_LINGER（0x0080）	getsockopt setsockopt	クローズ時のリンガー動作実行	数値型	linger型
	SO_BROADCAST（0x0020）	getsockopt setsockopt	ブロードキャストメッセージの転送を許可（UDPのみ）	フラグ型	int型
	SO_SNDBUF（0x1001）	getsockopt setsockopt	出力バッファの大きさ	数値型	int型
	SO_RCVBUF（0x1002）	getsockopt setsockopt	入力バッファの大きさ	数値型	int型
	SO_TYPE（0x1008）	getsockopt	ソケットの型	数値型	int型
	SO_ERROR（0x1007）	getsockopt	ソケットの最新のエラー（取得によりクリア）	数値型	int型
IPPROTO_IP (0)	IP_TOS（0x02）	getsockopt setsockopt	IPヘッダ上のTOSの値	数値型	int型
	IP_TTL（0x03）	getsockopt setsockopt	IPヘッダ上のTTLの値	数値型	int型
IPPROTO_TCP (6)	TCP_KEEPALIVE（0x02）	getsockopt setsockopt	TCPキープアライブプローブ送信までの待ち時間（単位は秒）	数値型	int型
	TCP_NODELAY（0x01）	getsockopt setsockopt	Nagleアルゴリズムの禁止	フラグ型	int型

（1） SO_REUSEADDR（0x0004）　　自通信アドレスの再利用の許可

このオプションは、自分の通信アドレス情報重複許可のON/OFFを示します。

このオプションがOFFの場合は、同一のIPアドレスの通信アドレス情報は使用できません。bind()により既に存在するIPアドレスを指定すると大域変数errnoにEADDRINUSEを設定して-1（エラー）を返します。

このオプションがONの場合、自分の通信アドレス情報の重複を可能にします。TCPとUDPでは働きが異なります。

TCPソケットの場合、能動モードのソケットにおいて、ポート番号、IPアドレスともに同一の自通信アドレスを複数使用できるようにします。受動モードのソケットは、ポート番号、IPアドレスともに同一にすることはできません。自通信アドレスがすでに存在する受動モードソケットと同一の能動モードのソケットを生成することはできますが、この能動モードのソケットに対しlisten()をコールすると大域変数errnoにEADDRINUSEを設定して-1（エラー）を返します。また、能動モードのソケットの場合でも自通信アドレスと相手側通信アドレスの両方を同じにすることはできません。connect()により自通信アドレスと相手側通信アドレスが同一のソケットが生成される場合は、大域変数errnoにEINVALを設定して-1（エラー）を返します。

UDPソケットの場合、複数の自分の通信アドレス情報を同一のポートで使用できるようにします。ただし、通信アドレス情報毎のIPアドレスは異なっている必要があります。bind()によりポート番号、IPアドレスともに同一の自通信アドレスを指定すると大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。

（２） SO_KEEPALIVE（0x0008）　　コネクションが有効か定期的な検査の実行

このオプションは、TCPソケットのコネクション定期的に検査する機能のON/OFFを示します。このオプションがONかつソケットが接続されている場合、そのソケットによる送信受信ともに行われない状況が２時間以上続いたなら、そのTCPソケットは自動的に相手に向けてキープアライブプローブを送信します。このキープアライブプローブに対しての相手側の対応によって次の処理を行います。

① 相手がACKで応答
正常な状態であるのでアプリケーションに対しなにも通知しません。送信受信ともないまま再び2時間経過したら再びキープアライブプローブを送信します。

② 相手からの応答が無い
この場合、75秒経過するごとにキープアライブプローブの再送をおこないます。8回再送しても応答が無い場合、RSTを送信してコネクションを切断します。

（３） SO_LINGER（0x0080）　　クローズ時のリンガー動作実行

このオプションは、TCPソケットにおけるクローズ処理のリンガー動作の内容を示します。このオプションデータは、構造体linger型（以下に示す）のデータです。この型のデータをp_optvalを通じて参照、設定します。

```
struct linger {
    unsigned short      l_onoff;    /* 0 : リンガーオプションOFF、0以外 : リンガーオプションON */
    unsigned short      l_linger;   /*  リンガー時間　単位は[1/100秒] */
};
```

オプションデータの内容によってclosesocket()の動作が異なります。オプションデータ毎のclosesocket()の動作を次に示します。

① l_onoffが0の場合
l_lingerの値は無視します。
closesocket()からすぐにリターンしますが、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。

② l_onoffが0以外、かつl_lingerが0の場合
closesocket()からすぐにリターンし、直ちにRSTを送信します。未送信のデータは破棄します。
l_onoffが0以外、かつl_lingerが0以外の場合closesocket()からすぐにはリターンしません。送信バッファ中のデータを送信し終わるのを待ってFINを送りますが、l_lingerの指定する時間を経過した場合は、RSTを送信して未送信のデータを破棄します。

（４） SO_BROADCAST（0x0020）　　ブロードキャストメッセージの転送の許可

このオプションは、ブロードキャストメッセージの転送能力の有効／無効を示します。ブロードキャストはUDPでのみサポートします。このオプションがONの時にブロードキャストメッセージの転送が有効となります。

このオプションがOFFの状態でデータの送信を行う（sendto()等の関数を使用する）時、相手側通信アドレス情報としてブロードキャストアドレスが設定されると、大域変数errnoにEACCESを設定して-1（エラー）を返します。

（５） SO_SNDBUF（0x1001）　　出力バッファの大きさ

このオプションは、TCPソケットの送信バッファサイズを示します。このオプションはint型のデータであり、p_optvalを通じて参照、設定します。設定する値は2048以上である必要があります。推奨値は、MSS×2×N です（Nは2 以上の整数）。

UDPのソケットに対してこのオプションを指定することはできません。

（６） SO_RCVBUF（0x1002）　　入力バッファの大きさ

このオプションは、ソケットの受信バッファサイズを示します。このオプションはint型のデータで、p_optvalを通じて参照、設定します。

TCPソケットの場合、ソケットの受信バッファサイズを示します。設定する値は2048以上である必要があります。推奨値は、MSS×2×N です（Nは2 以上の整数）。

UDPソケットの場合、データグラム受信キューの数を示します。

（７） SO_TYPE（0x1008）　　ソケットの型

このオプションを指定してgetsockopt()をコールすることで、ソケットの型の参照を行います。このオプションはint型のデータで、SOCK_STREAM（1）（TCPの場合）、SOCK_DGRAM（2）（UDPの場合）の値を取ります。

このオプションは、setsockopt()では、使用できません。

（８） SO_ERROR（0x1007）　　ソケットの最近のエラー取得

このオプションを指定してgetsockopt()をコールすることで、ソケット内の変数so_errorの値を取得することができます。

このとき、getsockopt()は変数so_errorを0にクリアします。

このオプションは、setsockopt()では、使用できません。

（９） IP_TOS（0x02）　　IPヘッダ内のTOSの値

このオプションは、IPヘッダ内のTOSの値を参照または設定します。

このオプションはint型のデータで、p_optvalを通じて参照、設定します。有効なデータの範囲は8ビットで0～255です。下位8ビット以外の設定データは無視します。

sockfdにより**指定したソケットの送信するIPヘッダのTOSだけが対象となります**。

（10） IP_TTL（0x03）　　IPヘッダ内のTTLの値

このオプションは、IPヘッダ内のTTLの値を参照または設定します。

このオプションはint型のデータで、p_optvalを通じて参照、設定します。有効なデータの範囲は8ビットで1～255です。下位8ビット以外の設定データは無視します。

sockfdにより**指定したソケットの送信するIPヘッダのTTLだけが対象となります**。

（11） TCP_KEEPALIVE（0x02）　　キープアライブプローブ送信までのアイドル時間

このオプションは、TCPキープアライブプローブ送信までのアイドル時間を秒単位で設定します。

このオプションはint型のデータで、p_optvalを通じて参照、設定します。また、SO_KEEPALIVEオプションがOFFのときのは、このオプションは無効です。時間の単位は秒です。

sockfdにより**指定したソケットだけアイドル時間が変更されます**。

（12） TCP_NODELAY（0x01）　　Nagleアルゴリズムの禁止

このオプションは、TCP Nagleアルゴリズムの禁止のON/OFFを示します。

sockfdにより**指定したソケットだけNagleアルゴリズムがON/OFFされます**。

4.2.15 setsockopt プロトコルのオプション変更

【T】

C言語インタフェース

```
int retcode = setsockopt(int sockfd, int level, int opt, void *optval, int optlen);
```

パラメータ

int	sockfd	ソケット識別子
int	level	プロトコル
int	opt	オプション
void	*optval	オプションデータを格納する領域の先頭アドレス
int	optlen	オプションデータの長さ

リターンパラメータ

int	retcode	リターン値

リターン値

0	正常終了
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	アドレス不正（optvalが0または4の倍数以外）
EINVAL	22	引数不正（optlenが短い）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ENOPROTOOPT	42	プロトコル利用不可（optが不正）
ENOBUFS	55	利用可能エリア不足（バッファが確保できない）
EPROTO	86	プロトコルエラー（使用中の通信端点に対し実行できない）

解説

プロトコルに対するオプションの値を設定します。

オプションの設定値をoptvalの示す領域から取得しソケットに設定します。
フラグ型オプションの場合、オプションの設定値が0なら、オプションを無効にし、0以外ならオプションを有効にします。

ソケットオプションについては、getsockopt()を参照してください。

levelとopt組み合わせが正しくない場合、およびオプションとしてSO_TYPEまたはSO_ERRORが指定された場合は、大域変数errnoにENOPROTOOPTを設定し-1（エラー）を返します。

BSDソケットAPIが生成したTCP/IPマネージャの通信端点を既に使用中で、指定したオプションの設定ができない状態の場合は、大域変数errnoにEPROTOを設定し-1（エラー）を返します。（SO_KEEPALIVE、SO_SNDBUF、SO_RCVBUF、IP_TOS、IP_TTL、TCP_KEEPALIVE、TCP_NODELAYの各オプションはlisten()またはconnect()後のTCPソケットには使えません）

4.2.16　selectsocket　　ソケットの利用可能待ち

【T】

C言語インタフェース

```
int retcode = selectsocket(int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *excepset, struct timeval *timeout );
```

パラメータ

int	maxfdp1	検査するディスクリプタの範囲
fd_set	*readset	受信用ディスクリプタを指定した領域のアドレス
fd_set	*writeset	送信用ディスクリプタを指定した領域のアドレス
fd_set	*excepset	例外用ディスクリプタを指定した領域のアドレス
struct timeval	*timeout	タイムアウト時間を格納した領域のアドレス

リターンパラメータ

int	retcode	リターン値
fd_set	readset	受信可能となったディスクリプタを示す情報
fd_set	writeset	送信可能となったディスクリプタを示す情報
fd_set	excepset	例外状態が発生したディスクリプタを示す情報

パケットの構造

```
typedef   struct {                          ディスクリプタ情報の構造体
            fd_mask             fds_bits[4];
} fd_set;

struct   timeval {                          タイムアウト時間の構造体
            long                tv_sec;             秒単位の値
            long                tv_usec;            μ秒単位の値
};
```

リターン値

1以上	正常終了（利用可能なディスクリプタの数）
0	タイムアウト
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EINTR	4	割り込みエラー （①timeoutで指定したタイムアウト時間を経過した ②stop_socket()によりソケットが停止した）
EBADF	9	識別子ソケット未指定（ディスクリプタに対応するソケットが未生成）
EFAULT	14	ポインタアドレス不正（readset, writeset, excepsetが0または4の倍数以外、timeoutが4の倍数以外）
EINVAL	22	引数不正（maxfdp1＜0またはmaxfdp1＞FD_SETSIZE(97)、0＞tv_sec、0≦tv_usec≦999999以外）

解説

ソケットに対する複数のイベントのいずれかが発生するのを待ちます。

イベントの発生が無い場合は、指定時間が経過するまで待ちます。

maxfdp1は、指定するソケット識別子の最大値を指定します（ソケット識別子最大番号プラス１の値）。実際には、maxfdp1の値をソケット識別子の最大値プラス１の値として取得します（ソケット識別子の最大値は、maxfdp1-1の値である）。

readset、writeset、excepsetの領域内のデータは、それぞれ受信可能、送信可能及び例外状態発生を検査するソケットを複数指定します。本BSDソケットAPIでは、例外状態に対応する状態は次の１つのみです。

①ソケットが緊急データを受信した

readset、writeset、excepsetの領域内のデータは、unsigned long型であり各ビットがそれぞれのソケット識別子を指定します(例えば､ソケット識別子が5のソケットを指定する場合はデータのbit5をセットします)。
対象となるビットは、maxfdp1により決まります。maxfdp1の指定するビットより大きいビットは無視します。

selectsocket()がコールされたときには、readset、writeset、excepsetの領域から検査するソケットの指定を受け取ります。また、リターンする際にはreadset、writeset、excepsetの領域に対し、受信可能、送信可能及び例外状態発生となったソケットを指定する値を設定します。

リターン値は、readset、writeset、excepsetの領域に設定されたデータのビットの総計です（例えば、ソケット識別子5のソケットのみ利用可能でかつそのソケットが受信、送信とも可能な場合、リターン値は2となります）。

*timeoutには、タイムアウト時間を指定します。timeoutがNULLの場合は、利用可能なソケットが発生するまで待ち続けます。

selectsocket()によりコネクション中のTCPソケットを検査すると、受信不可かつ送信不可の状態という結果になります。コネクションの試みが終了すると（正常、異常とわず）、受信可能や送信可能（あるいは、両方とも可能）になります。そのため、selectsocket()により対象のソケットの受信可能／不可と送信可能／不可を検査することでコネクションの試みの終了を確認することができます。
コネクションの確立が成功したかどうかは、ソケットオプションSO_ERRORによりソケットのエラーを調べることで確認できます。ソケットのエラーが正常終了を示している（errno=0）ならコネクションの確立に成功しています。

readset、writeset、excepsetに0または4の倍数以外が指定された場合は、大域変数errnoにEFAULTを設定し-1（エラー）を返します。
timeoutに4の倍数以外が指定された場合は、大域変数errnoにEFAULTを設定し-1（エラー）を返します。

4.3　拡張インタフェース関数

4.3.1　get_bsdapi_ver　　BSD ソケット API バージョン情報の参照

【T】

C言語インタフェース

int retcode = get_bsdapi_ver (void);

パラメータ

無し

リターンパラメータ

int	retcode	リターン値

リターン値／エラーコード

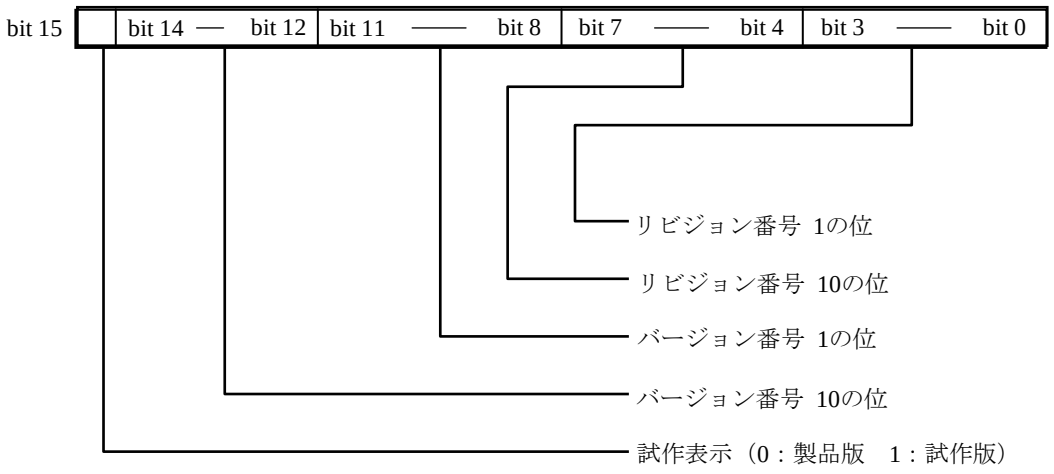
バージョン情報（下位16ビット）

解説

BSDソケットAPIのバージョンを参照します。

バージョン情報

バージョン情報は16ビットのBCD表示です。例えば Ver.03.10ではH'0310となります。



4.3.2 set_blocking_socket ブロッキングモード設定

【T】

C言語インタフェース

int retcode = set_blocking_socket(int sockfd, int blocking);

パラメータ

int	sockfd	ソケット識別子
int	blocking	設定するブロッキングモード （0：ブロッキング、1：ノンブロッキング）

リターンパラメータ

int	retcode	リターン値

リターン値／エラーコード

0	正常終了	
EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EINVAL	22	引数不正（blockingが0，1以外）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）

解説

指定したソケットのブロッキングモードを設定します。

blockingが0，1以外の値の場合は、リターン値としてEINVALを返します。

4.3.3 get_errno エラーコード取得

【T】

C言語インタフェース

int retcode = get_errno(void);

パラメータ

なし

リターンパラメータ

int	retcode	エラーコード

リターン値／エラーコード

errno（大域変数）に設定されていたエラーコード

errno（大域変数）

エラーはなし

解説

BSDソケットAPI全体で発生した最新のエラーコードを取得します。

4.3.4 get_thread_errno スレッドエラーコード取得

【T】

C言語インタフェース

int retcode = get_thread_errno(ID threadid);

パラメータ

ID	threadid	スレッド（タスク）ID

リターンパラメータ

int	retcode	エラーコード

リターン値／エラーコード

正の値	正常終了（タスクの大域変数（エラーコード））
0	正常終了（タスクの大域変数（エラーなし））
-1	エラー（関数実行不可（init_socket()が実行されていない））

errno（大域変数）

エラーはなし

解説

BSDソケットAPIを使用したタスク毎に発生した最新のエラーコードを取得します。
threadidに0を指定した場合は、自タスクのエラーコードを取得します。
取得したタスクのエラーコード（大域変数）は0にクリアされます。

4.3.5 set_sock_timewait TIMEWAIT 時間の変更

【T】

C言語インタフェース

```
int retcode = set_sock_timewait(int sockfd, short socktimewait);
```

パラメータ

int	sockfd	ソケット識別子
short	socktimewait	TIMEWAIT時間（秒）

リターンパラメータ

int	retcode	エラーコード

リターン値／エラーコード

0	正常終了
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EINVAL	22	引数不正（socktimewait＜1 または socktimewait＞240）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EOPNOTSUPP	45	未サポートソケット（指定したソケットがTCPのソケットではない）
EDEADLK	78	デッドロック条件 （BSDで生成した通信端点が他のプロセスによって削除された）
EPROTO	86	プロトコルエラー（使用中の通信端点に対し実行できない）

解説

指定したソケットのTIMEWAIT時間を変更します。

ソケットは、TIMEWAIT機能がデフォルトで無効設定で生成されます。本BSDソケットAPIをコールし、TIMEWAIT時間を設定することで指定したソケットのTIMEWAIT機能が有効になります。

TIMEWAIT時間として指定できる範囲は1秒から4分です。socktimewaitが指定できる範囲の値でない場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

指定されたソケットがTCPでない場合は、大域変数errnoにEOPNOTSUPPを設定し-1（エラー）を返します。

BSDソケットAPIが生成したTCP/IPマネージャの通信端点を既に使用中で、TIMEWAIT時間の変更ができない状態の場合は、大域変数errnoにEPROTOを設定し-1（エラー）を返します。（listen()またはconnect()後は使えません）

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

4.3.6 get_sock_recvlen 受信データ長の取得

【T】

C言語インタフェース

```
int retcode = get_sock_recvlen(int sockfd);
```

パラメータ

int	sockfd	ソケット識別子

リターンパラメータ

int	retcode	エラーコード

リターン値／エラーコード

0以上	正常終了（受信データの長さ）
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EPIPE	32	指定したソケットに対しshutdown()が実行されている
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
ENOTCONN	57	未接続エラー（sockfdの示すソケットが未接続）
EDEADLK	78	デッドロック条件 （BSDで生成した通信端点が他のプロセスによって削除された）

解説

ソケットが受信しているデータ長を取得します。

TCPソケットの場合は、受信用ウィンドバッファ内の有効な受信セグメント長を返します。
UDPソケットの場合は、キューイングされている全てのデータグラムを合計した長さを返します。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。
指定したソケットの受信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

4.3.7 get_socket_cep ソケットに対する通信端点の参照

【T】

C言語インタフェース

```
int retcode = get_socket_cep(int sockfd, T_BSD_CEP *cepinf);
```

パラメータ

int	sockfd	ソケット識別子
T_BSD_CEP	*cepinf	通信端点情報

リターンパラメータ

int	retcode	エラーコード
unsigned short	cepinf->protocol	プロトコル番号
short	cepinf->cepid	通信端点ID

リターン値／エラーコード

0	正常終了
-1	エラー

パケットの構造

```
typedef   struct{
            unsigned short      protocol;       プロトコル番号（TCP＝6，UDP＝17）
            short               cepid;          通信端点ID
} T_BSD_CEP;
```

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（cepinfが0または4の倍数以外）
EPIPE	32	指定したソケットに対する通信端点が存在しない（指定したソケットがTCPでもUDPでもない）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）

解説

ソケットが使用しているTCP/IPマネージャの通信端点を参照します。

指定したソケット指定したソケットがTCPでもUDPでもない場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

protocol には、ソケットのプロトコルを返します。TCPソケットの場合は6、UDPソケットの場合は17です。

cepid には、ソケットが使用しているTCP/IPマネージャの通信端点を返します。TCPソケットの場合はTCP通信端点ID、UDPソケットの場合はUDP通信端点IDを返します。指定したソケットが受動モードの場合と、通信端点が確定していない場合は0を返します。

本関数を用いて取得した通信端点に対して、直接ITRON TCP/IP APIを用いた操作を行った場合は、BSDソケットAPIの正常な動作は保証できません。

4.3.8 set_sock_keepalive ソケットに対するキープアライブの設定

【T】

C言語インタフェース

```
int retcode = set_sock_keepalive(int sockfd, T_BSD_KEEPALIVE *p_ktbl);
```

パラメータ

int	sockfd	ソケット識別子
T_BSD_KEEPALIVE	*p_ktbl	キープアライブ制御情報を格納した領域のアドレス

リターンパラメータ

int	retcode	エラーコード

パケットの構造

```
typedef   struct {                キープアライブ制御情報
            short                 k_onoff;        キープアライブ機能スイッチ（0：OFF，1：ON）
            short                 keeptime;       キープアライブパケット開始タイマ（秒）
            short                 kretrytime;     キープアライブパケットのリトライ間隔（秒）
            short                 kretrycnt;      キープアライブパケットのリトライ回数
} T_BSD_KEEPALIVE;
```

リターン値／エラーコード

0	正常終了
-1	エラー

errno（大域変数）

EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EFAULT	14	ポインタアドレス不正（p_ktblが0または2の倍数以外）
EINVAL	22	引数不正（keeptime＜1、kretrytime＜1、kretrycnt＜1）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EOPNOTSUPP	45	未サポートソケット（指定したソケットがTCPのソケットではない）
EDEADLK	78	デッドロック条件 （BSDで生成した通信端点が他のプロセスによって削除された）
EPROTO	86	プロトコルエラー（使用中の通信端点に対し実行できない）

解説

指定したソケットのキープアライブ機能の設定を変更します。

k_onoffには、キープアライブ機能の有効／無効を設定します。k_onoffに0を指定するとキープアライブ機能は無効になり、keeptime、kretrytime、kretrycntに指定した値は無視します。k_onoffに1（0以外の値）を指定するとキープアライブ機能が有効になります。

k_onoffに1（0以外の値）を指定した場合は、keeptime、kretrytime、kretrycntには1以上の値を指定してください。keeptime、kretrytime、kretrycntに0以下の値を指定した場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

指定されたソケットがTCPでない場合は、大域変数errnoにEOPNOTSUPPを設定し-1（エラー）を返します。

BSDソケットAPIが生成したTCP/IPマネージャの通信端点を既に使用中で、キープアライブ機能の設定変更ができない状態の場合は、大域変数errnoにEPROTOを設定し-1（エラー）を返します。（listen()またはconnect()後は使えません）

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

4.3.9 set_mds_mode ソケットに対する MDS モードの設定

【T】

C言語インタフェース

```
int retcode = set_mds_mode(int sockfd, int flag);
```

パラメータ

int	sockfd	ソケット識別子
int	flag	設定するMDSモード （0：MTUからMDSを取得、1：構築時のバッファ長をMDSとして取得）

リターンパラメータ

int	retcode	リターン値

リターン値／エラーコード

0	正常終了	
EPERM	1	関数実行不可（init_socket()が実行されていない）
EBADF	9	識別子無効（sockfd＜0）
EINVAL	22	引数不正（flagが0，1以外）
ENOTSOCK	38	識別子ソケット未指定（sockfdと同じ識別子のソケットが存在しない）
EOPNOTSUPP	45	未サポートソケット（指定したソケットがUDPのソケットではない）
EUSERS	68	重複操作（指定したソケットを他のBSD関数が使用中）

解説

指定したソケットのMDSモードを設定します。

flagが0の値の場合は、MTUから取得したMDS値をソケットで使用します。
flagが1の値の場合は、TCP/IPマネージャ構築時に指定したUDPバッファ長を、MDS値としてソケットで使用します。
flagが0，1以外の値の場合は、リターン値としてEINVALを返します。

この拡張インターフェースは、TCP/IPマネージャでフラグメント送信が使用可能に設定されている場合に限り、有効になります。

BSDソケットAPI Ver3.0
リファレンスマニュアル
CM7000BSD03J-13

発行年月　　2014年 8月　　第13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