Ｍ－８０１　ＴＣＰ/ＩＰプロトコルスタック仕様書

ＴＣＰ/ＩＰプロトコルスタックサンプルは下記のソースファイルから構成されている。

OSI基本参照モデル		ソースコード	定義	動作定義
物理層		rtl8019.c	rtl8019.h	_rtl8019.h
データリンク層	Ethernet	ethernet.c	ethernet.h	
	ARPプロトコル	arp.c	arp.h	
ネットワーク層	IPプロトコル	ip.c	ip.h	
	ICMPプロトコル	icmp.c	icmp.h	
トランスポート層	TCPプロトコル	tcp.c	tcp.h	tcpdef.h
	UDPプロトコル	udp.c	udp.h	udpdef.h
その他		ethergen.c	ethergen.h	
型定義			lp.h	

物理層(rtl8019.c)関数

初期化		
書式	bool　NicInit(void (*waitms)(int))	
引数	void (*waitms)(int)	ウェイト関数へのポインタ
戻り値	初期化成功/失敗 ： TRUE/FALSE	
機能	RTL8019Aを初期化する。 ソフトウェアリセット、CPU割り込み設定、各レジスタの初期化、 Serial EEPROMから動作設定、MACアドレス読み出しを行う。 ソフトウェアリセット時、一定期間過ぎてもPAGE0:ISRのRSTフラグがアクティブに ならない場合FALSEを返す。	

ＭＡＣアドレス取得	
書式	byte　*NicGetMacAddress(void)
引数	なし
戻り値	MACアドレスが格納されているバッファ（６バイト）へのポインタ
機能	RTL8019A初期化時にSerial EEPROMから読み出したMACアドレスを取得する。

送信		
書式	bool　NicTransmit(const byte *ebuf, int psize)	
引数	const byte *ebuf	パケットデータ
	int psize	パケットサイズ
戻り値	送信成功/失敗 ： TRUE/FALSE	
機能	RTL8019Aに送信データを転送する。 送信タイムアウトが起きた場合FALSEを返す。	
注意	この関数はデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	

受信		
書式	int　NicReceive(byte *ebuf)	
引数	byte *ebuf	パケットデータバッファアドレス
戻り値	受信パケットデータサイズ	
機能	RTL8019Aから受信データを転送し、バッファに格納する。	
注意	この関数はデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	

割り込み	
書式	void　NicIntr(void)
引数	なし
戻り値	なし
機能	RTL8019Aの割り込み処理を行う。
注意	この関数はシステムで呼び出す必要がある。

データリンク層Ethernet(ethernet.c)関数、グローバル変数

送信		
書式	bool　EtherTransmit(int size)	
引数	int size	データサイズ
戻り値	送信成功/失敗 ： TRUE/FALSE	
機能	Ethernetヘッダ部にデータをセットし物理層にデータを渡す。 宛て先IPに対するMACアドレスがARPテーブルに登録されていない場合。 ARP要求データを物理層に渡し、FALSEを返す。	
注意	この関数はネットワーク層及びデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
受信		
書式	int　EtherReceive(void)	
引数	なし	
戻り値	受信データサイズ	
機能	RTL8019Aから受信データを転送し、バッファに格納する。 受信データがARPプロトコルの場合ARP受信処理を行い0を返す。	
注意	この関数はネットワーク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
パケットデータバッファ		
書式	byte　ether_buffer[]	
機能	パケットデータ用バッファ。 送信/受信共用。	
IPアドレスの設定		
書式	void　EtherSetIp(dword ip)	
引数	dword ip	IPアドレス
戻り値	なし	
機能	IPアドレスを設定する。	
IPアドレスの取得		
書式	dword　EtherGetIp(void)	
引数	なし	
戻り値	設定されているIPアドレス	
機能	現在の設定されているIPアドレスを取得する。	
Netmaskアドレスの設定		
書式	void　EtherSetNetmask(dword netmask)	
引数	dword netmask	Netmaskアドレス
戻り値	なし	
機能	Netmaskアドレスを設定する。	
Netmaskアドレスの取得		
書式	dword　EtherGetNetmask(void)	
引数	なし	
戻り値	設定されているNetmaskアドレス	
機能	現在の設定されているNetmaskアドレスを取得する。	
Gatewayアドレスの設定		
書式	void　EtherSetGateway(dword gateway)	
引数	dword gateway	Gatewayアドレス
戻り値	なし	
機能	Gatewayアドレスを設定する。	
Gatewayアドレスの取得		
書式	dword　EtherGetGateway(void)	
引数	なし	
戻り値	設定されているGatewayアドレス	
機能	現在の設定されているGatewayアドレスを取得する。	

データリンク層ARPプロトコル(arp.c)関数

検索		
書式	bool　ArpSearch(byte *mac, dword ip, dword nmask, dword gway)	
引数	byte *mac	MACアドレス格納バッファアドレス
	dword ip	IPアドレス
戻り値	MACアドレス発見 : TRUE/FALSE	
機能	ipに対するMACアドレスをテーブルから検索し、発見した場合はTRUEを返す。 テーブルに登録されていない場合は、ARP要求送信を行いFALSEを返す。	
注意	この関数はデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
受信		
書式	void　ArpInput(void)	
引数	なし	
戻り値	なし	
機能	ARPパケット受信に対する処理を行う。	
注意	この関数はデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
登録		
書式	void　ArpCache(void)	
引数	dword ip	IPアドレス
戻り値	なし	
機能	ARPテーブルにIPアドレスとそれに対するMACアドレスを登録する。	
注意	この関数はデータリンク層から呼び出される。 ユーザーが直接呼び出すことはできない。	

ネットワーク層IPプロトコル（ip.c）

送信		
書式	bool　IpTransmit(const IP_INF0 *ip, int size)	
引数	const IP_INF0 *ip	IP_INF0構造体 typedef struct{ 　byte protocol　プロトコル値 　dword srcip　送信元IPアドレス 　dwrod dstip　送信先IPアドレス }IP_INF0;
	int size	データサイズ
戻り値	EthterTransmit()の戻り値	
機能	IPヘッダ部にデータをセットし、下層の送信関数を呼び出す。	
注意	この関数はトランスポート層及び、ネットワーク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
受信		
書式	int　IpReceive(byte *iphlen)	
引数	byte *iphlen	IPヘッダ長へポインタ
戻り値	受信したプロトコル	
機能	下層の受信関数を呼び出し、受信データからプロトコルを判別する。 マクロ　意味 RECEIVE_NON　データ無し/不明プロトコル RECEIVE_UDP　UDPプロトコル RECEIVE_TCP　TCPプロトコル	
注意	この関数はトランスポート層から呼び出される。 ユーザーが直接呼び出すことはできない。	

ネットワーク層ICMPプロトコル（icmp.c）

受信		
書式	void　IcmpReceive(void)	
引数	なし	
戻り値	なし	
機能	ICMPパケット受信時の処理を行う。	
注意	ネットワーク層から呼び出される。 ユーザーが直接呼び出すことはできない。	
Echo Request送信		
書式	bool　IcmpSendEchoReq(dword ip, int size)	
引数	dword ip	宛て先IPアドレス
	int size	データサイズ
戻り値	IpTransmit()の戻り値	
機能	Echo Requestを送信する。	
Echo応答確認		
書式	bool　IcmpIsEchoReply(void)	
引数	なし	
戻り値	Echo Requestに対する応答があればTRUEを返す。 なければFALSEを返す。	
機能	IcmpSendEchoReq()呼び出しからIcmpIsEchoReply()の戻り値がTRUEになった場合の 経過時間が指定IPアドレス機器の応答時間となる。	

トランスポート層UDPプロトコル(udp.c)

初期化	
書式	void　udp_init(void)
引数	なし
戻り値	なし
機能	モジュール内の変数を初期化する。
注意	同モジュール内の他の関数が呼ばれる前に必ず一回呼ぶ。
受信	
書式	void　udp_input(void)
引数	なし
戻り値	なし
機能	UDPパケットの受信処理を行う。
注意	UDPソケット関数を利用する場合、定期的にこの関数を呼び出す必要がある。
UDPソケット関数群	
	TCP、UDPソケット関数を参照

トランスポート層TCPプロトコル（tcp.c）

初期化		
書式	void　tcp_init(const long *mseccount)	
引数	const long *mseccount	msecタイマー変数のアドレス
戻り値	なし	
機能	モジュール内の変数を初期化する。	
注意	同モジュール内の他の関数が呼ばれる前に必ず一回呼ぶ。	
受信		
書式	void　tcp_input(void)	
引数	なし	
戻り値	なし	
機能	TCPパケットの受信処理を行う。	
注意	TCPソケット関数を利用する場合、定期的にこの関数を呼び出す必要がある。	

再送信処理	
書式	void　　tcp_retry(void)
引数	なし
戻り値	なし
機能	TCPパケットの再送信処理を行う。
注意	TCPソケット関数を利用する場合、定期的にこの関数を呼び出す必要がある。
TCP、UDP受信処理タスク	
書式	void　　tcpudp_input(void)
引数	なし
戻り値	なし
機能	UDPパケット受信処理、TCPパケット受信、再送信処理を行う。 この関数内で udp_input()、tcp_input()、tcp_retry()関数を呼び出す。
注意	TCP、UDPソケット関数を利用する場合、定期的にこの関数を呼び出す必要がある。
TCPソケット関数群	
	TCP、UDPソケット関数を参照

TCP、UDPソケット関数（tcp.c、udp.c）

関数名はWinsockに似せているが、組み込みシステムにおけるシングルタスクでの使用を想定
各関数は内部で待ち処理を行わず必ず即結果を返す。
また、ユーザーの実装するソケットプログラムとは別にTCPまたはUDPの受信処理を
定期的に呼ぶ必要がある。

生成（TCP、UDP）		
書式	int　　tcp_socket(void)、udp_socket(void)	
引数	なし	
戻り値	ソケットの利用が可能な場合ソケットID（0以上）を返す。 ソケットが利用できない場合TCP_NO_ENTRY(UDP_NO_ENTRY)を返す。	
終了（TCP、UDP）		
書式	int　　tcp_closesocket(int id)、udp_closesocket(int id)	
引数	int id	ソケットID
戻り値	正常の場合0を返す。 不正なIDの場合はTCP_NO_ENTRY(UDP_NO_ENTRY)を返す。	
ポート結合（TCP、UDP）		
書式	int　　tcp_bind(int id, word port)、udp_bind(int id, word port)	
引数	int id	ソケットID
	word port	ポート番号
戻り値	ソケットの利用が可能な場合ソケットID（0以上）を返す。 ソケットが利用できない場合TCP_NO_ENTRY(UDP_NO_ENTRY)を返す。	
接続キュー作成（TCP）		
書式	int　　tcp_listen(int id, int backlog)	
引数	int id	ソケットID
	int backlog	接続キュー数
戻り値	接続キュー数を返す。 ソケットが利用できない場合TCP_NO_ENTRYを返す。	
接続受け入れ（TCP）		
書式	int　　tcp_accept(int id)	
引数	int id	ソケットID
戻り値	接続要求がクライアントからあった場合割り当てられたソケットIDを返す。 接続要求がない場合TCP_NO_ACCEPTを返す。 ソケットが利用できない場合TCP_NO_ENTRYを返す。	

接続（TCP）

書式	int　　tcp_connect(int id, const sockaddr *dst)	
引数	int id	ソケットID
	const sockaddr *dst	typedef struct{ 　dword ip;　　接続先IPアドレス 　word port;　　接続先ポート番号 }sockaddr;
戻り値	接続できた場合0を返す。 ソケットが利用できない場合、または未接続の場合TCP_NO_ENTRYを返す。	

状態取得（TCP）

書式	byte　　tcp_status(int id)	
引数	int id	ソケットID
戻り値	ソケットの状態を返す。 ソケットが利用できない場合0xffを返す。	

状態	値	状態	値	状態	値
LISTEN		ESTABLISHED		LAST_ACK	
SYN_RCVD		FIN_WAIT_1		FIN_WAIT_2	
SYN_SENT		CLOSE_WAIT		CLOSED	

TCPデータ送信（TCP）

書式	int　　tcp_send(int id, const byte *data, int len)	
引数	int id	ソケットID
	const byte *data	データバッファ
	int len	データサイズ
戻り値	送信データ数を返す。 ソケットが利用できない場合、または未接続の場合TCP_NO_ENTRYを返す。 前回の送信データが残っている場合TCP_TX_BUSYを返す。	

TCPデータ受信（TCP）

書式	int　　tcp_recv(int id, byte *data, int len)	
引数	int id	ソケットID
	byte *data	データバッファ
	int len	バッファサイズ
戻り値	受信データ数を返す。 受信データが無く、送信データが送信済みの場合0を返す。 ソケットが利用できない場合、または未接続の場合TCP_NO_ENTRYを返す。 送信データが残っている場合TCP_TX_BUSYを返す。	

UDPデータ送信（UDP）

書式	int　　udp_sendto(int id, const byte *buf, int len, 　　word sport, const sockaddr *dst)	
引数	int id	ソケットID
	const byte *data	データバッファ
	int len	データサイズ
	word sprot	送信元ポート
	const sockaddr *dst	送信先情報（tcp_connect()参照）
戻り値	送信データ数を返す。 ソケットが利用できない場合、または未接続の場合UDP_NO_ENTRYを返す。	

UDPデータ受信（UDP）

書式	int　　udp_recfrom(int id, byte *buf, int max, word *dport, sockaddr *src)	
引数	int id	ソケットID
	byte *data	データバッファ
	int max	バッファサイズ
	word *dport	送信先ポートデータへのポインタ
	sockaddr *src	送信元情報（tcp_connet()参照）

戻り値	受信データ数を返す。 ソケットが利用できない場合UDP_NO_ENTRYを返す。

イーサネット関連汎用関数（ethergen.c）

データ加算		
書式	dword　add_dataw(const byte *addr, int size, dword init)	
引数	const byte *addr	データバッファアドレス
	int size	データサイズ（byte）
	dword init	初期値
戻り値	加算値	
機能	バッファデータをワード単位で加算する。	

チェックサム計算（１の補数）		
書式	word　checksum(const byte *addr, int size, dword init, bool make)	
引数	const byte *addr	データバッファアドレス
	int size	データサイズ（byte）
	dword init	初期値
	bool make	TRUE/FALSE　計算/評価
戻り値	計算値を返す。	
機能	バッファデータをワード単位で加算、チェックサムを行う。	

３２ビットデータ文字列変換		
書式	char*　dword2str(dword num, char *buf)	
引数	dword num	データ
	char *buf	文字列バッファアドレス
戻り値	文字列バッファアドレスへのポインタ	
機能	３２ビットデータを１６進文字列に変換する。	

１６ビットデータ文字列変換		
書式	char*　dword2str(dword num, char *buf)	
引数	word num	データ
	char *buf	文字列バッファアドレス
戻り値	文字列バッファアドレスへのポインタ	
機能	１６ビットデータを１６進文字列に変換する。	

８ビットデータ文字列変換		
書式	char*　dword2str(dword num, char *buf)	
引数	byte num	データ
	char *buf	文字列バッファアドレス
戻り値	文字列バッファアドレスへのポインタ	
機能	８ビットデータを１６進文字列に変換する。	

８ビットデータセット		
書式	void　set_byte(byte *buf, int offset, byte val)	
引数	byte *buf	データバッファアドレス
	int offset	オフセット
	byte val	データ
戻り値	なし	
機能	バッファ先頭からoffsetの位置にデータをセットする。	
注意	マクロ関数	

８ビットデータ取得		
書式	byte　get_byte(const byte *buf, int offset)	
引数	const byte *buf	データバッファアドレス
	int offset	オフセット
戻り値	８ビットデータ	
機能	バッファ先頭からoffsetの位置のデータを取得する。	
注意	マクロ関数	

１６ビットデータセット		
書式	void　set_word(byte *buf, int offset, word val)	
引数	byte *buf	データバッファアドレス
	int offset	オフセット
	word val	データ
戻り値	なし	
機能	バッファ先頭からoffsetの位置にデータをセットする。	
１６ビットデータ取得		
書式	word　get_word(const byte *buf, int offset)	
引数	const byte *buf	データバッファアドレス
	int offset	オフセット
戻り値	１６ビットデータ	
機能	バッファ先頭からoffsetの位置のデータを取得する。	
３２ビットデータセット		
書式	void　set_dword(byte *buf, int offset, dword val)	
引数	byte *buf	データバッファアドレス
	int offset	オフセット
	byte val	データ
戻り値	なし	
機能	バッファ先頭からoffsetの位置にデータをセットする。	
３２ビットデータ取得		
書式	dword　get_dword(const byte *buf, int offset)	
引数	const byte *buf	データバッファアドレス
	int offset	オフセット
戻り値	３２ビットデータ	
機能	バッファ先頭からoffsetの位置のデータを取得する。	
符号なし１６ビットデータ文字列変換		
書式	char*　utoa(word num, char *buf)	
引数	word num	データ
	char *buf	文字列バッファアドレス
戻り値	文字列バッファアドレスへのポインタ	
機能	符号なし１６ビットデータを１０進文字列に変換する。	
IPアドレスデータ文字列変換		
書式	char*　inet_ntoa(dword ipaddr, char *buf)	
引数	dword ipaddr	IPアドレス
	char *buf	文字列バッファアドレス
戻り値		
機能	IPアドレスデータを文字列に変換する。 0xC0A80016 -> "192.168.0.22"	
IPアドレス文字列データ変換		
書式	dword　inet_addr(const char *str)	
引数	const char *str	文字列バッファアドレス
戻り値	IPアドレスデータ値を返す。文字列が不正な場合0を返す。	
機能	IPアドレス文字列を３２ビットデータに変換する。 "192.168.0.123" -> 0xC0A8007B	

参考文献

・CQ出版　トランジスタ技術 2001年1月号
・CQ出版　トランジスタ技術 2001年9月号
・CQ出版　トランジスタ技術 2002年12月号
・Winsock Programmer's FAQ
 http://tangentsoft.net/wskfaq/
・Winsock Programmer's FAQ 日本語訳
 http://www.kt.rim.or.jp/~ksk/wskfaq-ja/index.html

M-801　TCP/IPプロトコルスタック仕様書
初版作成　2005年1月10日

発行　株式会社ロジパック

〒438-0078　静岡県磐田市中泉1803-1
TEL 0538-32-2822　　FAX 0538-34-1082
URL　http://www.enshu-net.or.jp/logicpack/
E-mail logicpack@po1.enshu-net.or.jp