PPP for NORTi

User's Guide

2016 年 12 月版

株式会社 ミスポ

2016 年 12 月版で改訂された項目

ページ	更新内容
8	リソースに PPP 接続タスクを追加
9	リソース ID に ID_PPP_CON_TSK(PPP 接続タスク用 ID)を追加
18	ppp_set_opt 関数の解説を修正
21	mdm_ini 関数の解説を修正
22	mdm_rcv_dat 関数を追加

2014 年 2 月版で改訂された項目

ページ	更新内容
14	EV_PPP_ECHO_REP を追記
20	ppp_echo_req 関数を追加

2012 年 8 月版で改訂された項目

ページ	更新内容
6	ファイル構成に nonchap.c を追記
33	ppp_read_frm の解説の誤った記述を削除

2010 年 8 月版で改訂された項目

ページ	更新内容
7	PPP 用固定長メモリプールを 1 個追加
8	リソース ID に ID_PPP_TXBUF を追加
10,11	dbg_ch と dbg_param を削除
11	ダイアルリトライ間隔の単位の説明を変更
15	LITTLE_ENDIAN マクロを削除
16,17	ppp_ini と ppp_ext を削除

2010 年 3 月版で改訂された項目

ページ	更新内容
6	RFC1662 準拠を追記
7,8	フレーム送受信用メモリプールを削除
7	PPP のプロトコルスタックは TCP/IP スタックで利用するリソースも確保する必要がある旨を追記
16,20	E_TMOUT が E_TMO となっていた誤記を修正
21	STS_ATC_IN が STS_MDM_IN となっていた誤記を修正

2009 年 7 月版で改訂された項目

ページ	更新内容
15	コンパイル時のマクロ定義を PPP の全てのソースファイルに対して指定するように変更
15	VJ 圧縮を使用しない場合は noipcmp.c をプロジェクトから外せる旨を追記
15	CHAP の機能を利用しない場合は nonencr.c をプロジェクトから外せる旨を追記
22	初期化の手順で mdm_ini が必要となっていた誤記を修正

2008 年 11 月版で改訂された項目

ページ	更新内容
13	受信キャリアを取得できない場合は自動接続の使用を禁止する旨を追記

2006 年 8 月版で改訂された項目

ページ	更新内容
11	回線種別(line)に LINE_FOMA を追加
15	FOMA_UM01K0_ADAPTER マクロを追加
20	mdm_dial()に E_RESTR を追加
21	「モデム出力関数」を「モデム制御関数」に変更
21	mdm_reset(RELTIM dlytim)を追加

2005 年 4 月版で改訂された項目

ページ	更新内容
10	変数"ppp"のメンバ sio_param,dbg_param,pnumber,user, passwd,user_s ,passwd_s を B 型から char 型に変更
24	param の型を B*から char*に変更

2005 年 3 月版で改訂された項目

ページ	更新内容
8	「3.2 IP アドレス」にマルチチャネル対応プロトコルスタックの記述を追加
10	「3.4 PPP コンフィグレーション」パケットトレース(dump)の記述を変更
11	「3.4 PPP コンフィグレーション」デバッグポート番号(dbg_ch)、シリアルパラメータ(dbg_param)の記述を変更
12	「3.4 PPP コンフィグレーション」ダイアルイン動作時のユーザー名(user_s)、ダイアルイン動作時のパスワード(passwd_s)の記述を変更
15	「3.5 コンパイル時のマクロ定義」DUMP マクロの記述を追加
15	「3.6 TCP/IP プロトコルスタック Ver4.08.12 以降を使用する場合」を追加
20	「4.2 ダイアリング関数」に mdm_ini()を追加
22	「4.4 呼出し手順」の記述を変更
36	「7.1 パケットトレース機能」に「パケットトレース機能の使用手順」を追加

2004 年 11 月版で改訂された項目

ページ	更新内容
-	“NORTi Network”という表現を“NORTi TCP/IP”に変更
6	ドライバ・インターフェース・モジュールの説明を追加
6	2.2 リソースに"PPP 用固定長メモリプール"を追加
7	3.3 マクロを追加
9	回線種別(line)を変更
11	イベントコールバック関数に 3)PPP イベントを追加
16	ppp_ext, ppp_set_opt を新規追加
17	ppp_ref_inf を新規追加
29-32	第 6 章ドライバ・インターフェース・モジュールを追加

目　次

第１章　導入

1.1 はじめに

PPP for NORTi は NORTi TCP/IP のデータリンクレイヤで PPP (Point to Point Protocol)を実現します。本書では PPP の使用に関する説明を行っています。NORTi Version 4 に共通の事項については「NORTi Version4 ユーザーズガイド TCP/IP 編」と「NORTi Version4 ユーザーズガイド カーネル編」を参照してください。

1.2 特長

PPP for NORTi は次の RFC に対応しています。

RFC1662	PPP 疑似 HDLC フレーム
RFC1661	PPP リンク制御プロトコル (LCP)
RFC1332	PPP インターネット・プロトコル制御プロトコル (IPCP)
RFC1334	PPP 認証プロトコル (PAP)
RFC1994	チャレンジ・ハンドシェイク認証プロトコル(MD5 CHAP)
RFC2433	Microsoft PPP CHAP
RFC1144	Van Jacobson TCP/IP ヘッダ圧縮

以下の RFC は未サポートです。

RFC1990	PPP マルチリンクプロトコル(MP)
RFC1989	PPP リンク品質監視プロトコル

1.3 制限事項

NORTi の下で動作するよう特化しているため、他の OS へ移植することは困難です。
使用可能な上位プロトコルは IP Version4 のみです。

1.4 ファイル構成

PPP for NORTi は次のファイルから構成されます。

noneppp.c	PPP モジュールソースコード
nondial.c	ダイアリングモジュールソースコード
nonchap.c	CHAP 用共通ソース
nonencr.c	CHAP 用 暗号化ソース
noipcmp.c	TCP/IP 圧縮プロトコルソース
noneppp.h	PPP 標準ヘッダ
nonpppc.h	PPP コンフィグレーションヘッダ コンフィグレーションを行うファイル(nonetc.h をインクルードしているソースファイル)で１度だけインクルードしてください。
nonppps.h	PPP 内部定義ヘッダ
nonencr.h	CHAP 用 暗号化ヘッダ

第２章　モジュール構成

2.1 概要

本システムは次のような各モジュールで構成されています。

シリアルドライバ

RS232C ドライバです。PPP ではモデムと通信を行うチャンネルに 1 チャネル、デバッグ用に 1 チャネルを使用しています。

ダイアリング・モジュール

モデムを制御するためのモジュールで、通信先との接続／切断を行います。

PPP モジュール

PPP のプロトコルをサポートするモジュールです。

NORTi TCP/IP

TCP/IP プロトコルスタックです。

ドライバ・インターフェース・モジュール

物理層のデバイスドライバと PPP の間に入るモジュールです。このインターフェースをカスタマイズすることで、RS232C＋モデム以外のインターフェースを PPP で使用できるようになります。

2.2 リソース

PPP for NORTi のプロトコルスタックは次のリソースを必要とします。

PPP 送信タスク	1 個
PPP 接続タスク	1 個
PPP 送受信メールボックス	2 個
PPP 用固定長メモリプール	2 個
PPP 用イベントフラグ	2 個

また、PPP for NORTi プロトコルスタックは TCP/IP スタックも利用しますので、TCP/IP スタックで利用するリソースも確保する必要があります。詳細は、TCP/IP 編のユーザーズガイドを参照してください。なお、PPP 接続タスクは着信動作のみで使用されますので、着信動作を利用されない場合は、不要になります。

第３章　コンフィグレーション

3.1 リソース ID

次の各リソース ID をユーザーアプリケーション内で定義してください。

```
const ID ID_PPP_SND_TSK   = ?;     PPP 送信タスク ID
const ID ID_PPP_CON_TSK   = ?;     PPP 接続タスク
const ID ID_PPP_SND_MBX   = ?;     PPP 送信用メールボックス
const ID ID_PPP_RCV_MBX   = ?;     PPP 受信用メールボックス
const ID ID_PPP_MPF       = ?;     PPP の送受信用固定長メモリプール
const ID ID_PPP_TXBUF     = ?;     PPP の送信用固定長メモリプール
const ID ID_PPP_EFLG      = ?;     PPP 用イベントフラグ
const ID ID_MDM_EFLG      = ?;     モデム制御用イベントフラグ
```

各リソース ID を 0 で指定すると、ID は内部で自動的に割り当てられます。

3.2 IP アドレス

IP アドレスはグローバル変数 default_ipaddr をユーザーアプリケーション内で定義してください。全て 0 の場合 PPP はサーバーから IP を取得します。

```
UB default_ipaddr[]  = { 0, 0, 0, 0 };
```

※1 IP アドレスを毎回サーバーから取得したい場合は接続前にこの値を 0 クリアしてください。

※2 tcp_acp_cep を使用して相手から接続を待つ場合は必ず IP を指定して下さい。指定をせずに 0.0.0.0 と設定した場合は正しく動作しません。

※3 マルチチャネル対応プロトコルスタック(Ver4.08.12 以降では標準)を使用している場合で IP アドレスを固定で使用している場合、tcp_ini()呼出し後に IP アドレスの値を変更する場合は net_chg_ipa()もしくは netif_chg_ipa()を使って IP アドレスを変更してください。

3.3 コンフィグレーションマクロ

nonpppc.h は PPP のコンフィグレーションを行うためのヘッダファイルです。nonpppc.h を#include する前に次のような定数を記述することにより、コンフィグレーションを行うことができます。()はデフォルト値で、指定がない場合に使用されます。これらのコンフィグレーションは特に問題が無い限りデフォルトの値のままご使用ください。

#define DIAL_TMOUT	(30000)30 秒	ダイアリング・タイムアウト
#define PPP_LCP_RETRY	(5)回	LCP の送信リトライ回数
#define PPP_LCP_TMOUT	(3000)3 秒	LCP の受信タイムアウト
#define PPP_AUTH_RETRY	(5)回	認証フェーズの送信リトライ回数
#define PPP_AUTH_TMOUT	(10000)10 秒	認証フェーズの受信タイムアウト
#define PPP_IPCP_RETRY	(5)回	IPCP の送信リトライ回数
#define PPP_IPCP_TMOUT	(3000)3 秒	IPCP の受信タイムアウト

3.4 PPP コンフィグレーション

PPP の各種設定情報はグローバル変数"ppp"をユーザーアプリケーション内で定義してください。

PPP コンフィグレーションテーブルの構造

```
typedef struct t_ppp_ini{
    BOOL    dump;           パケットトレース
    INT     line;           回線種別
    INT     retry;          ダイアルリトライ
    INT     rttime;         リトライ間隔
    INT     sio_chk;        シリアルステータスチェック項目
    INT     sio_ch;         モデムの回線ポート番号
    char    sio_param[20];  シリアルパラメータ
    BOOL    auto_dial;      自動接続
    BOOL    auto_answer;    自動応答
    char    pnumber[35];    接続番号
    char    user[50];       ユーザー名（認証なしの場合は NULL）
    char    passwd[50];     パスワード（認証なしの場合は NULL）
    char    user_s[50];     ユーザー名（サーバー動作時）
    char    passwd_s[50];   パスワード（サーバー動作時）
    PPP_CALLBACK  callback; イベントコールバック
} T_PPP;
```

パケットトレース（dump）

PPP パケットのフレームダンプを行うか否かのスイッチ
指定されたデバッグポート番号にダンプされます。
以下の設定が OR 指定可能です。

0	パケットトレースの機能を使用しません
DUMP_PPP	PPP パケットを出力します
DUMP_IP	IP パケットを出力します

Ver3.51 以降のバージョンではパケットトレースの機能を使用する場合、この設定以外にいくつかのコンフィグレーション設定が必要になります。
使用手順の詳細は 7.1 パケットトレース機能を参照してください。

回線種別（line）

接続する電話回線の種別です

LINE_TONE　　トーン

LINE_PULSE　　パルス

LINE_DIRECT　　モデムを使用せずに RS232C 直結

LINE_NOHDLC　　HDLC のフレームを使用しない

LINE_FOMA　　FOMA 網を使用する

通常のフレーム

FF	03	Protoco	データ	FCS	7E

LINE_NOHDLC 指定のフレーム

Protoco	データ

ダイアルリトライ（retry）

ダイアルのリトライ回数です。リトライしない場合は 0 を指定してください。

ダイアルリトライ間隔（rttime）

ダイアルのリトライを行う間隔を指定します。単位は、システムクロックの割り込み周期です。

シリアルステータスチェック項目（sio_chk）

定期チェックする RS232C(モデム)の制御線を設定します。ここで指定した制御線が変化したときコールバックが呼び出されます。

TSIO_CTS：CTS をチェックする

TSIO_DSR：DSR をチェックする

TSIO_CD　：CD をチェックする

複数をチェックする場合、OR 指定可能です。複数が指定され、同時に複数の制御線が変化した場合は、1 度のみコールバックが呼ばれます。優先度は CTS、DSR、CD の順です。

モデムの回線ポート番号（sio_ch）

モデムを接続するシリアルポートの番号を指定してください。

シリアルパラメータ（sio_param）

モデムを接続するシリアルポートのコンフィグレーションです。

自動接続（auto_dial）

PPP 内部でモデム回線接続の必要がある場合、自動的に接続を行うか否かのスイッチです。このスイッチが TRUE(1)の場合アプリケーションから意識的に接続を行わなくても、TCP/IP の送信が行われると、自動的にダイアル接続し、ppp_start が実行されます。自動接続を有効にする場合は、sio_chk の TSIO_CD をセットし、CD

をチェックするようにしてください。受信キャリアが取得できない場合は自動接続を使用しないでください。

自動応答（auto_answer）

ダイアルインの時に行う認証を設定します。

ANSWER_NONE	0	ダイアルインを行わない
ANSWER_NOAUTH	1	認証を行わない
ANSWER_PAP	2	PAP による認証を行う
ANSWER_CHAP_MD5	3	MD5 の CHAP 認証を行う
ANSWER_CHAP_MSV1	4	MS-CHAP 認証を行う
ANSWER_CHAP_MSV2	5	MS-CHAPv2 認証を行う

接続番号（pnumber）

接続先の電話番号を指定します。

ユーザー名（user）

認証で使用するユーザー名です。認証を使用しない場合は NULL です。

パスワード（passwd）

認証で使用するパスワードです。認証を使用しない場合は NULL です。

ダイアルイン動作時のユーザー名（user_s）

認証で使用するユーザー名です。認証を使用しない場合は NULL です。
この機能を使用する場合は、自動応答を ANSWER_NONE, ANSWER_NOAUTH 以外に設定してください。

ダイアルイン動作時のパスワード（passwd_s）

認証で使用するパスワードです。認証を使用しない場合は NULL です。
この機能を使用する場合は、自動応答を ANSWER_NONE, ANSWER_NOAUTH 以外に設定してください。

イベントコールバック関数

PPP でイベントが発生した場合に呼び出されるコールバック関数です。
シリアル制御線を使用する場合は PPP のコンフィグレーション sio_chk に使用する制御線を設定してください。

[設定例]

```
void callback(int event, VP parblk, int len)
event    イベントコード
parblk   AT コマンドリザルトコード(STS_ATC_IN の場合)
len      AT コマンドリザルトコードの長さ(STS_ATC_IN の場合)

ppp.callback = (PPP_CALLBACK)callback;
でコールバック関数を指定します。
```

イベントコードは次のような種類があります。

１）シリアル制御線の変化

STS_CTS_ON	CTS 信号が OFF から ON に変化した
STS_CTS_OFF	CTS 信号が ON から OFF に変化した
STS_DSR_ON	ケーブルが接続された、モデムが ON になった。(DSR 信号が OFF から ON に変化した)
STS_DSR_OFF	ケーブルが切断された、モデムが OFF になった。(DSR 信号が ON から OFF に変化)
STS_CD_ON	回線が接続された(CD 信号が OFF から ON に変化)
STS_CD_OFF	回線が切断された(CD 信号が ON から OFF に変化)
STS_PPP_RUNN	PPP の接続シーケンス完了通知

２）モデムからの応答

STS_ATC_IN	モデムから AT コマンドの応答コードが入力された。

３）PPP イベント

EV_PPP_RUNN	PPP のネゴシエーションが完了した
EV_PPP_CONT	PPP のネゴシエーションを開始した
EV_PPP_LCP_TRM	LCP TERM ACK が送受信された
EV_PPP_PHY_CLS	ドライバ・インターフェース・モジュールで回線が切断された
EV_PPP_ECHO_REP	LCP エコー応答を受信した

旧式イベントコールバック関数

以下は旧方式のコールバック関数仕様です。旧式のコールバック方式を使用する場合は、nondial.c、noneppp.c をコンパイルする際に、OLD_CB_STYLE マクロを定義してコンパイルしてください。

[設定例]

void ppp_callback(int n)を用意します。
ppp.callback = (PPP_CALLBACK)ppp_callback;
でコールバック関数を指定します。

イベント ID は次のような種類があります。

STS_PPP_DIAL	1	ダイアリング中
STS_PPP_ANSR	2	応答中
STS_PPP_CONT	4	PPP 接続処理中
STS_PPP_RUNN	5	PPP 接続完了
STS_PPP_CLSE	6	回線が切断された

3.5 コンパイル時のマクロ定義

PPP の全てのソースファイルをコンパイルする際に以下のマクロを指定することによって、不要なモジュールを組み込まないようにすることが出来ます。

NOCOMP マクロ

VJ 圧縮プロトコルを使用しない場合はこのマクロを指定してください。又、noipcmp.c をプロジェクトから外すことが出来ます。

NOCHAP マクロ

PPP の認証機能として MD5 CHAP や MS CHAP の機能が利用できますが、利用しない場合は NOCHAP マクロ定義によってプログラムサイズを小さくすることができます。又、nonencr.c をプロジェクトから外すことが出来ます。

OLD_CB_STYLE マクロ

旧式のコールバック関数を使用する場合に指定してください。このマクロを使用する際には nondial.c もコンパイルする必要があります。

DUMP マクロ

7.1 のパケットトレース機能を使用する場合、noneppp.c と nondial.c コンパイル時にこのマクロを定義してコンパイルしてください。

FOMA_UM01K0_ADAPTER マクロ

FOMA UM01-K0 アダプターを使う時にこのマクロと定義して nondial.c をコンパイルしてください。

3.6 TCP/IP プロトコルスタック Ver4.08.12 以降を使用する場合

TCP/IP プロトコルスタック Ver4.08.12 以降、マルチチャネル版プロトコルスタックが標準になりました。そのため、nonetc.h を include しているアプリケーションファイルをコンパイルする際に"PPP"マクロを定義してコンパイルしてください。このマクロ指定で PPP がデフォルトチャネルに設定されます。

Ver4.08.12 より以前のバージョンでは NORTi¥PPP¥LIB のライブラリをリンクしていましたが、Ver4.08.12 以降は NORTi¥LIB のライブラリをリンクしてください。

第４章　外部関数仕様

4.1 PPP 関数

ppp_start

[機 能] PPP のネゴシエーションを開始する

[形 式] ER ppp_start(void);

[戻 値] E_OK　　正常終了
　　E_TMOUT　PPP 接続中タイムアウトした
　　E_PAR　　認証失敗
　　E_OBJ　　回線未接続

[解 説] PPP の一連のリンク処理を行います。LCP、認証、IPCP を実行します

ppp_end

[機 能] PPP のセッションを終了する

[形 式] ER ppp_end(void);

[戻 値] ＝ E_OK　　正常終了

　　≠ E_OK　　内部エラー

[解 説] LCP Terminal Request を送信し PPP のセッションを終了します。

ppp_set_opt

[機 能] LCP、IPCP のオプションを変更します

[形 式] ER ppp_set_opt (T_OPT_PPP *rp, UW optcode);

　　rp　　オプション情報のポインタ
　　optcode オプションコード

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

［解 説］デフォルト以外の LCP、IPCP のオプションを設定したい場合に使用します。

```
typedef struct t_opt_ppp{
UH      l_mru;          ローカルホストの MRU
UW      l_magicn;       ローカルホストのマジックナンバー
UW      l_accm;         ローカルホストの ACCM
UB      r_ip[4];        クライアント動作時に割り当てる IP アドレス
} T_OPT_PPP;
```

オプションコードに OPT_MRU_SET、OPT_MAGIC_SET、OPT_ACCM_SET、OPT_REMOTE_IP_SET のいずれも指定しない場合は、rp に NULL が指定できます。また、LCP 設定要求に MRU オプションを付加したくない場合は、OPT_MRU_SET で 0 を指定してください。

オプションコード	内容	default	備考
OPT_ACFC_ON	Address Field 圧縮を ON	○	LINE_NOHDLC 設定時は OFF
OPT_ACFC_OFF	Address Field 圧縮を OFF		
OPT_ACCM_ON	Async-Control-Character-Map を ON	○	LINE_NOHDLC 設定時は OFF
OPT_ACCM_OFF	Async-Control-Character-Map を OFF		
OPT_PFC_ON	Protocol Field 圧縮を ON	○	
OPT_PFC_OFF	Protocol Field 圧縮を OFF		
OPT_IPCM_ON	Van Jacobson TCP/IP ヘッダ圧縮プロトコルを ON	○	
OPT_IPCM_OFF	Van Jacobson TCP/IP ヘッダ圧縮プロトコルを OFF		
OPT_DNS_ON	サーバーから DNS の IP アドレスを取得する	しない	
OPT_MRU_SET	MRU を設定する	1500	
OPT_MAGIC_SET	マジックナンバーを設定する	ランダム	
OPT_ACCM_SET	ACCM を設定する	0	
OPT_REMOTO_IP_SET	通信相手に割り当てる IP アドレスを指定する	しない	

［使用例］

Address Field 圧縮を OFF、Async-Control-Character-Map を OFF、
Protocol Field 圧縮を OFF 、MRU1000 に設定する場合

```
UW opcode;
T_OPT_PPP rp;

opcode = (OPT_ACFC_OFF | OPT_ACCM_OFF | OPT_PFC_OFF | OPT_MRU_SET);
rp.l_mru =  1000;
ppp_set_opt (&opt, optcode);
```

ppp_ref_inf

[機 能] PPP のネゴシエーション情報を取得します

[形 式] void ppp_ref_inf (T_INF_PPP *inf);

inf　　ネゴシエーション情報を格納するバッファのポインタ

[戻 値] なし

[解 説] リモートホストとネゴシエーションされた PPP の情報を取得できます。
PPP が切断された場合は、前回接続された時の情報が設定されます。

```
typedef struct t_ppp_inf{
  UH     l_mru;          ローカルホストの MRU
  UH     r_mru;          リモートホストの MRU
  UW     l_accm;         ローカルホストの ACCM
  UW     r_accm;         リモートホストの ACCM
  UW     l_magicn;       ローカルホストのマジックナンバー
  UW     r_magicn        リモートホストのマジックナンバー
  BOOL   l_pfc;          ローカルホストの Protocol Field 圧縮(TRUE/FALSE)
  BOOL   r_pfc;          リモートホストの Protocol Field 圧縮(TRUE/FALSE)
  BOOL   l_acfc;         ローカルホストの Address Field 圧縮(TRUE/FALSE)
  BOOL   r_acfc;         リモートホストの Address Field 圧縮(TRUE/FALSE)
  UB     l_auth;         ローカルホストの認証タイプ(サーバー動作時)
  UB     r_auth;         リモートホストの認証タイプ
  UB     l_ip[4];        ローカルホストの IP アドレス
  UB     r_ip[4];        リモートホストの IP アドレス
  BOOL   l_ipcmp;        ローカルホストの VJ TCP/IP ヘッダ圧縮(TRUE/FALSE)
  BOOL   r_ipcmp;        リモートホストの VJ TCP/IP ヘッダ圧縮(TRUE/FALSE)
  UB     dns_ip1[4];     リモートホストから取得したプライマリ DNS アドレス
  UB     dns_ip2[4];     リモートホストから取得したセカンダリ DNS アドレス
} T_PPP_INF;
```

l_pfc と r_pfc, l_acfc と r_acfc, l_ipcmp, r_ipcmp は常に同じになります。

認証タイプには次の値が入ります

```
AUTH_NONE        認証指定なし
AUTH_PAP         PAP による認証
AUTH_CHAP_MD5    MD5 CHAP による認証
AUTH_CHAP_MSV1   Microsoft CHAP Version1
AUTH_CHAP_MSV2   Microsoft CHAP Version2
```

ppp_echo_req

[機 能] LCP エコー要求を送信します

[形 式] ER ppp_echo_req(void);

[戻 値] ＝ E_OK　　正常終了

　　≠ E_OK　　内部エラー

[解 説] LCP エコー要求を送信します。

4.2 ダイアリング関数

mdm_ini

[機 能] モデムの初期化を行います

[形 式] ER mdm_ini(void);

[戻 値] E_OK　　正常終了
E_TMOUT　タイムアウト
E_OBJ　　モデム使用不可あるいは回線接続不可

[解 説] モデムへ”AT”コマンドを出力し初期化を行います。着信動作が有効な場合は、モデムが着呼に応答するように設定します。本関数をコールせずに mdm_dial をコールした場合は、mdm_dial 内部で本関数が呼ばれます。そのため、発信でしか接続しない場合は、本関数の呼び出しを省くことができます。

mdm_dial

[機 能] 回線の接続を行います

[形 式] ER mdm_dial(void);

[戻 値] E_OK　　正常終了
E_TMOUT　タイムアウト
E_OBJ　　モデム使用不可あるいは回線接続不可
E_RESTR RESTRICTION を受信(FOMA 回線特有のエラー)

[解 説] モデムの回線接続を行います。E_RESTR を受信した場合はアプリケーションで分単位のディレイを行ってから、再度この関数を呼び出してください。

mdm_hung

[機 能] 回線の切断を行います

[形 式] ER mdm_hung(void);

[戻 値] E_OK　　正常終了
E_TMOUT　タイムアウト

[解 説] モデムの回線を切断します。旧バージョン互換性のため dial_connect(), dial_disconnect も使用可能です。

4.3 モデム制御関数

mdm_snd_dat

[機 能] モデムへ AT コマンドを出力します。

[形 式] ER mdm_snd_dat(char *data, INT len, TMO tmout);
char *data　モデムへ出力する文字列
int len　文字数
TMO tmout　タイムアウト

[解 説] モデムへ AT コマンドを出力します。回線接続中の場合は、モデムの仕様にしたがいオンラインコマンドモードに切り替えた後に AT コマンドを出力してください。モデムからのレスポンスはイベントコールバック関数(イベントコード：STS_ATC_IN)で受信できます。

mdm_rcv_dat

[機 能] モデムからレスポンスを受信します。

[形 式] ER mdm_rcv_dat(char *data, INT size, TMO tmout);
char *data　受信した文字列の格納先
int len　格納先のサイズ
TMO tmout　タイムアウト

[戻 値] 0 以上　受信した文字数
E_TMOUT タイムアウト

[解 説] モデムからレスポンスを受信します。mdm_snd_dat で AT コマンドを発行する前に、別のタスクから本関数をコールしてください。別のタスクを用意したくない場合は、イベントコールバック関数(イベントコード：STS_ATC_IN)で受信を行ってください。

mdm_reset

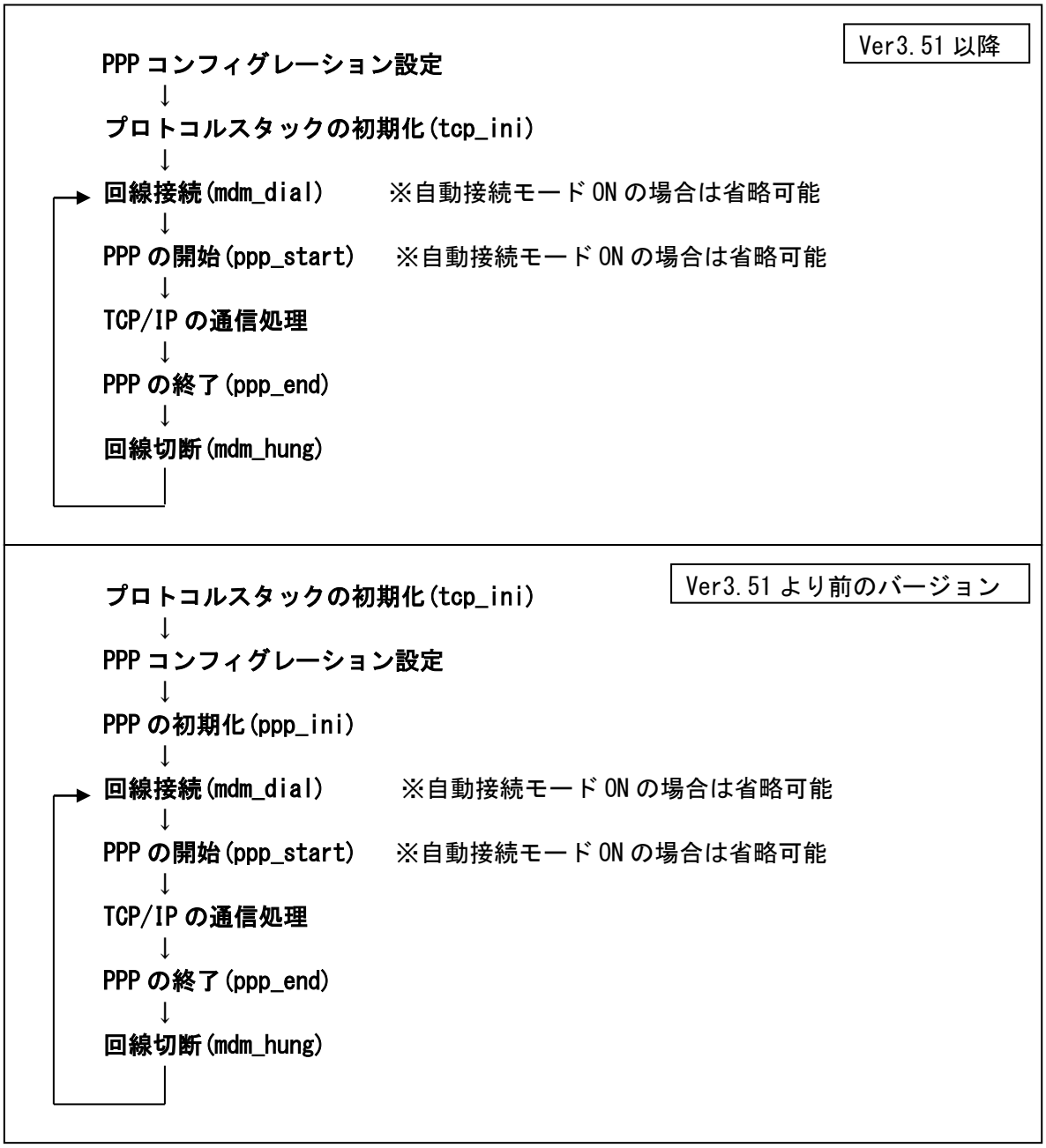
[機 能] モデムをリセットします。

[形 式] ER mdm_reset(RELTIM dlytim);
RELTIM dlytim　　モデムリセット出力後の遅延時間

[解 説] モデムへ AT * DHWRST コマンドを出力します。この API は回線種別(line)が LINE_FOMA 指定時のみ使用できます。モデムリセット後、dlytime で指定した時間サービスコール内部でディレイします。

4.4 呼出し手順

PPP を使用する場合、呼び出しは以下の手順で行います。使用しているバージョンによって初期化の手順が異なりますので、ご注意ください。



※ tcp_ini(), ppp_ini()はシステム起動時にタスクから一度だけ呼び出してください。
ダイアルインの場合は、mdm_dial(),ppp_start()の呼び出しは必要ありません。
※ Ver3.51 以降では ppp_ini()は tcp_ini()内部で自動的に呼び出されます。したがって、呼び出しは不要です。

第５章　シリアル入出力ドライバ

5.1 特徴

PPP は NOSIO をシリアルドライバとして使用しています。
シリアルドライバを作成／移植する場合は本章を参考にしてください。
NOSIO は、送受信とも割り込みを利用した NORTi 対応のシリアル入出力ドライバです。
このドライバでは、RS-232C の１文字送信と１文字受信をタイムアウト付きで実現します。

5.2 機能

・RS-232C 調歩同期式全二重
・送受信共、割り込み駆動
・１文字送受信機能のみ（１行送受信はユーザ作成）
・XON/XOFF キャラクタによるソフトウェアフロー制御を選択可
・DTR-CTS または RTS-CTS 信号によるハードウェアフロー制御を選択可

5.3 コンパイル

NOSIO はソースで提供されます。必要なチャネル数に合わせてコンパイルし、ユーザープログラムとリンクしてください。
コンパイル方法は、各ソースファイルのコメントを参照してください。

5.4 注意事項

本書には標準的な機能を記載していますが、デバイスに依存する事項が多いので、付属ソースファイルのヘッダに必ず目を通してください。

PPP ではモデムの制御信号を使用しています。CD 信号により通信切断中か否かの判断を行っています。これらの制御線を使用しない場合は ref_sio 関数内での siostat に TSIO_CD：受信キャリア検出(bit0),TSIO_DSR：DSR 信号 ON(1)/OFF(0)(bit15)の各ビットが必ず ON になるようにして下さい。

5.5 シリアル入出力関数

ini_sio

[機能] シリアル入出力初期化

[形式] ER ini_sio(ch, param);
INT ch; チャネル番号
const char *param; 通信パラメータ文字列

[解説] ch で指定したチャネルの RS-232C インターフェースを初期化します。
RS-232C のパラメータ param を、次の文字列で指定してください。
大文字、小文字は区別しません。各パラメータの間には、１個以上のスペースを入れてください。

"300"～"38400" ---- ボーレート
"B8","B7" -------- キャラクタ長８，７
"PN","PE","PO" ---- パリティ無，偶，奇
"S1","S2" -------- ストップビット１，２
"XON" ---------- XON/OFF によるソフトウェアフロー制御有り
"DTR" ---------- DTR-CTS によるハードウェアフロー制御有り
"RTS" ---------- RTS-CTS によるハードウェアフロー制御有り

(例) ini_sio(0, "2400 B7 PE S2 XON");

デフォルトは、「9600 ボー、キャラクタ長 8、パリティ無し、ストップビット 1、フロー制御無し」です。
param の不正のチェックは厳密には行っていません。
初期化後は、送受信ともディセーブル、DSR OFF、RTS OFF となります。
従って、初期化後に ctl_sio 関数で、送受信イネーブル、DSR ON、RTS ON とする必要があります。

[戻値] E_OK 正常終了
E_PAR チャネル番号範囲外，通信パラメータ不正
E_NOMEM メモリ不足

ext_sio

[機能] シリアル入出力終了

[形式] void cdecl ext_sio();

[解説] RS-232C インターフェースを初期化前の状態に戻します。
パソコン用のプログラムでは、終了時に必ず呼び出してください。
本関数を実行せずにプログラムを終了すると、RS-232C の割り込みが生き残っており、後で暴走の原因となります。

[戻値] 戻値はありません。

get_sio

[機能] シリアル１文字入力

[形式] ER get_sio(ch, p_c, tmout);
INT ch; チャネル番号
UB *p_c; 入力した文字
TMO tmout タイムアウト時間

[解説] ch で指定したチャネルから受信したキャラクタを *p_c に返します。
受信したキャラクタが無ければ、受信するまで待ちます。
tmout でタイムアウト時間を指定します。tmout = TMO_FEVR ならばタイムアウトを監視しません。
tmout = 0 で、受信したキャラクタが無い場合は、タイムアウトエラーとして即座にリターンします。
rel_wai システムコールにより、受信待ちを解除できます。

(例) get_sio(0, &c, TMO_FEVR);

[戻値] E_OK 正常終了
E_TMOUT 受信タイムアウト
E_RLWAI 受信待ち解除
E_PAR チャネル番号範囲外
EV_SIOOVF 受信バッファオーバフロー
EV_SIOPTY パリティエラー
EV_SIOORN オーバーランエラー
EV_SIOFRM フレーミングエラー

put_sio

[機能] シリアル１文字出力

[形式] ER put_sio(ch, c, tmout);
INT ch; チャネル番号
UB c; 出力する文字
TMO tmout タイムアウト時間

[解説] ch で指定したチャネルへキャラクタ c を送信します。
送信にもバッファを設けていますので、実際に送信が行われた時点ではなく、送信バッファに格納した時点で送信完了と見なされます。真の送信完了を待つ場合は、続けて fls_sio を実行して下さい。
すでに送信バッファが一杯ならば、空きが出るまで待ちます。
tmout でタイムアウト時間を指定します。tmout = TMO_FEVR ならばタイムアウトを監視しません。
tmout = 0 で、送信バッファに空きがない場合は、タイムアウトエラーとして即座にリターンします。
rel_wai システムコールにより、送信待ちを解除できます。

（例）put_sio(0, c, 100/MSEC);

[戻値] E_OK 正常終了
E_TMOUT 送信タイムアウト
E_RLWAI 送信待ち解除
E_PAR チャネル番号範囲外

ctl_sio

[機能] シリアル入出力制御

[形式] ER ctl_sio(ch, fncd);
INT ch; チャネル番号
UH fncd; 機能コード

[解説] 機能コード fncd により、ch で指定したチャネルの制御を行います。
fncd には、nosio.h ファイルで定義されている次の値を指定してください。

TSIO_RXE	受信イネーブル
TSIO_RXD	受信ディセーブル
TSIO_TXE	送信イネーブル
TSIO_TXD	送信ディセーブル
TSIO_RTSON	RTS 信号 ON
TSIO_RTSOFF	RTS 信号 OFF
TSIO_DTRON	DTR 信号 ON
TSIO_DTROFF	DTR 信号 OFF
TSIO_RXCLR	受信バッファクリア
TSIO_TXCLR	送信バッファクリア
TSIO_SBON	ブレーク送信 ON
TSIO_SBOFF	ブレーク送信 OFF

(例) ctl_sio(0, TSIO_RXE|TSIO_TXE|TSIO_DTRON|TSIO_RTSON);

[戻値] E_OK 正常終了
E_PAR チャネル番号範囲外

ref_sio

[機能] シリアル入出力状態参照

[形式] ER ref_sio(ch, pk_sios);
INT ch; チャネル番号
T_SIOS *pk_sios; シリアル入出力状態パケット

[解説] ch で指定したチャネルの状態を *pk_sios に返します。
シリアル入出力状態パケット T_SIOS 構造体は、nosio.h ファイルの中で
次のように定義されています。

```
typedef struct t_sios
{
    UB siostat;      シリアル入出力ステータス
    UB rxchr;        受信バッファの先頭の文字
    UH rxlen;        受信バッファのデータ長
    UH frbufsz;      送信バッファの空きサイズ
    UH eotcnt;       受信バッファの終端文字個数
} T_SIOS;
```

構造体メンバー siostat は次のビットより構成されます。

TSIO_CD	受信キャリア検出
TSIO_CTS	CTS 信号 ON(1)/OFF(0)
TSIO_TXEMP	送信バッファ空
TSIO_PE	パリティエラー
TSIO_OE	オーバランエラー
TSIO_FE	フレーミングエラー
TSIO_BD	ブレーク状態検出
TSIO_DSR	DSR 信号 ON(1)/OFF(0)

(例) T_SIOS sios;

```
ref_sio(0, &sios);
if (sios.siostat & TSIO_DSR)
```

[戻値] E_OK 正常終了
E_PAR チャネル番号範囲外

fls_sio

[機能] シリアル送信バッファ･フラッシュ

[形式] ER fls_sio(ch, tmout);
INT ch; チャネル番号
TMO tmout タイムアウト時間

[解説] ch で指定したチャネルの送信バッファにたまっているデータを、全て送り出します。
tmout でタイムアウト時間を指定します。tmout = TMO_FEVR ならばタイムアウトを監視しません。
rel_wai システムコールにより、送信待ちを解除できます。

(例) fls_sio(0, 1000/MSEC);

[戻値] E_OK 正常終了
E_TMOUT 送信タイムアウト
E_RLWAI 送信待ち解除
E_PAR チャネル番号範囲外

第６章ドライバ・インターフェース・モジュール

6.1 特徴

ドライバ・インターフェース・モジュールは物理層のデバイスドライバと PPP の間に入るモジュールです。このインターフェースをカスタマイズすることで、RS232C＋モデム以外のインターフェースを PPP で使用できるようになります。

6.2 API 一覧

関 数	機 能
ppp_ini_phy	物理層モジュールの初期化
ppp_ext_phy	物理層モジュールの終了
ppp_opn_phy	回線のオープン
ppp_cls_phy	回線のクローズ
ppp_ref_phy	物理層モジュールの状態参照
ppp_write_frm	PPP フレームの書き込み
ppp_read_frm	PPP フレームの読み出し
ppp_cfg_ini	コンフィグレーションの設定値を有効にする

---

ppp_ini_phy

[機 能] 物理層モジュールの初期化

[形 式] ER ppp_ini_phy(void);

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

[解 説] この関数は ppp_ini よりコールされます。
　　この関数ではデバイスの初期化やリソースの生成などをおこないます。
　　E_OK 以外で関数から終了した場合、ppp_ini はエラー終了します。

ppp_ext_phy

[機 能] 物理層モジュールの終了

[形 式] ER ppp_ext_phy(void);

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

[解 説] この関数は ppp_ext よりコールされます。
　　この関数ではリソースの削除や、デバイスの終了処理(割り込みのディセーブルなど)を行います。E_OK 以外で関数から終了した場合、ppp_ext はエラー終了します。

ppp_opn_phy

[機 能] 回線のオープン

[形 式] ER ppp_opn_phy(void);

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

[解 説] 物理層で使用する回線のオープン(例えばモデムの場合の回線接続) を行います。
　　この関数はアプリケーションから直接呼び出されます。内部ではコンフィグレーション設定を有効にするため ppp_cfg_ini をコールする必要があります。

ppp_cls_phy

[機 能] 回線のクローズ

[形 式] ER ppp_cls_phy(void);

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

[解 説] 物理層で使用する回線のクローズ(例えばモデムの場合の回線切断) を行います。この関数はアプリケーションから直接呼び出されます。

ppp_cfg_ini

[機 能] PPP のコンフィグレーション設定を有効にします

[形 式] ER ppp_cfg_ini(BOOL flag);

[戻 値] ＝ E_OK　　正常終了
　　≠ E_OK　　内部エラー

[解 説] この関数は ppp_opn_phy から呼び出されます。
　　サーバー動作時はリモートホストからの接続に対して ppp.user_s および ppp.pass_s を使い、リモートホストに対して認証動作を行います。
　　クライアント動作時はリモートホストから指定された認証タイプで、ppp.user および ppp.pass を使って認証を行います。

ppp_ref_phy

[機 能] 物理層モジュールの状態参照

[形 式] ER ppp_ref_phy(UH code);

　　code　　参照内容
　　STS_PHY_READY　　物理層モジュールが通信を行える状態にあるか
　　STS_PHY_CONNECT　回線が接続状態にあるか

[戻 値] E_NOSPT　　未サポート
　　E_OK　　可
　　E_NORDY　　否

[解 説] この関数は PPP モジュール内部から物理層の状態を確認するために定期的に呼び出されます。

ppp_read_frm

[機 能] PPP フレームの読み出し

[形 式] ER ppp_read_frm(UB *frm, UH len);

frm　PPP フレームを格納するバッファのポインタ

len　バッファのサイズ

[戻 値] 正の値　正常終了(取り出したデータのサイズ)

＜＝ 0　異常終了

[解 説] この関数は PPP モジュール内部から呼び出されます。受信データが無い場合は待ちになります。

ppp_write_frm

[機 能] PPP フレームの書き込み

[形 式] ER ppp_write_frm(UB *frm, UH len);

frm　PPP フレームを書き込むバッファのポインタ

len　書き込むフレームのサイズ

[戻 値] ＝ E_OK　正常終了

≠ E_OK　内部エラー

[解 説] この関数は PPP モジュール内部から呼び出されます。この関数では PPP のフレームをデバイスに書き出します。

第７章　トラブルシューティング

7.1 パケットトレース機能

問題が起こった場合は別のシリアルチャンネルで PPP のトレースダンプを表示してみると、どのシーケンスで問題が発生しているのかがわかります。以下はダンプの例です。

```
[sent LCP ConfRej] id=1 len=7
    0D 03 06
 [sent LCP ConfReq] id=1 len=28
    01 04 05 DC 03 04 C0 23 02 06 00 00 00 00 05 06
    00 08 09 F8 07 02 08 02
[rcvd LCP ConfRej] id=1 len=7
    0D 03 06
[sent LCP ConfAck] id=2 len=20
    02 06 00 0A 00 00 05 06 01 36 09 A9 07 02 08 02
 [rcvd LCP ConfAck] id=2 len=20
    02 06 00 0A 00 00 05 06 01 36 09 A9 07 02 08 02
[rcvd LCP ConfAck] id=1 len=28
    01 04 05 DC 03 04 C0 23 02 06 00 00 00 00 05 06
    00 08 09 F8 07 02 08 02
[sent PAP ConfAck] id=1 len=5
    00
[rcvd PAP ConfAck] id=1 len=5
    00
[sent IPCP ConfRej] id=1 len=28
    84 06 00 00 00 00 03 06 C0 A8 66 0C 81 06 00 00
    00 00 82 06 00 00 00 00
[sent IPCP ConfReq] id=1 len=16
    02 06 00 2D 0F 01 03 06 C0 A8 66 0B
[rcvd IPCP ConfRej] id=1 len=28
    84 06 00 00 00 00 03 06 C0 A8 66 0C 81 06 00 00
    00 00 82 06 00 00 00 00
[rcvd IPCP ConfAck] id=1 len=16
    02 06 00 2D 0F 01 03 06 C0 A8 66 0B
[sent IPCP ConfRej] id=2 len=22
    83 06 00 00 00 00 03 06 C0 A8 66 0C 81 06 00 00
    00 00
[rcvd IPCP ConfRej] id=2 len=22
    83 06 00 00 00 00 03 06 C0 A8 66 0C 81 06 00 00
    00 00
[sent IPCP ConfAck] id=3 len=16
    02 06 00 2D 0F 01 03 06 C0 A8 66 0C
 [rcvd IPCP ConfAck] id=3 len=16
    02 06 00 2D 0F 01 03 06 C0 A8 66 0C
```

パケットトレース機能の使用手順（Ver3.51 以降）

パケットトレースの機能を使用する場合は､以下のコンフィグレーションを行ってください。

・noneppp.c,nondial.c をコンパイルする際に"DUMP"マクロを定義します。
・NETSMP¥SRC フォルダにある nondump.c をリンクします。
・ tcp_ini()を呼び出す前に landump_ini（ダンプ機能の初期化）を呼び出してください。
・ppp.dump を設定します。

nondump.c の詳細につきましては

NORTi Version4 ユーザーズガイド｢2.8 LAN パケットダンプ機能｣をご覧ください。

7.2 トラブルシューター

モデムが反応しない

ターゲットボードとの接続で使用しているケーブルを確認してください。

ダイアルするが、その後反応が無い

mdm_dial でタイムアウトエラーが返ってきている場合は、login の
受信が無いにも関わらず、login 受信を待っている場合です。
ためしにターミナルソフトを使って RAS に接続してみてください。
ATDT を使って RAS に接続すると CONNECT の後に login を返してくる RAS と
返さない RAS があります。返さない場合は nondial.c を NOLOGIN マクロ付き
でコンパイルして実行してみてください。
実装しているターゲットボードの SIO には CTS,RTS 線が結線されていますか？
モデムを使用する場合この信号線が必要です。

ダイアルするが、BUSY

接続先の番号は正しいですか？

PPP のフレームを送信はしているが、受信が返ってこない

tcp_ini,ppp_ini 関数の戻り値がエラーになっていませんか？
また、4.1 呼出し手順どおりに呼び出していますか？
IP の受信タスクが正常に動作していないと、このような現象になります。

PPP 接続中にエラー

ユーザー名、パスワードは間違っていませんか？
IP を指定した場合は IP アドレスが正しいですか？

PPP for NORTi ユーザーズガイド

株式会社ミスポ http://www.mispo.co.jp/
〒222-0033 神奈川県横浜市港北区新横浜 3-20-8 BENEX S-3 12F
一般的なお問い合せ sales@mispo.co.jp
技術サポートご依頼 norti@mispo.co.jp