NORTi TCP/IP 対応 無線 LAN ドライバ

NORTiDD-WLAN

ユーザーズガイド

2005 年 9 月 2 日 第 8 版

株式会社ミスポ

目 次

1. 概要

NORTi TCP/IP 対応 無線 LAN ドライバは IEEE802.11b 準拠、PRISM™チップセット搭載の無線 LAN カードに対応し、インフラストラクチャネットワークまたはアドホックネットワークの設定、さらに 64bit/128bit WEP 暗号化の設定が可能です。これらの設定は本ドライバの API により変更することができます。
また、本ドライバは動作確認済みのライブラリで提供され、無線 LAN カードの制御部、つまり PC カードコントローラのドライバなどを追加実装することで、各種ハードウェアに移植することが可能です。また、移植の際に参考となる各種評価ボード用サンプルプログラムと、本無線 LAN ドライバの全ソースファイルが付属します。

2. 制限事項

- 本ドライバは無線 LAN カード（PC カード）を I/O モードで制御します。よって、PC カードの動作モードとして I/O モードが必要です。たとえば、TrueIDE モードで結線されているハードウェアでは本ドライバは動作しません。
- 本ドライバは無線 LAN カードにアクセスポイントの機能を持たせることはできません。

3. ファイル構成

本ドライバのファイル構成は次のようになります。ライブラリはプロセッサ/コンパイラ別に、サンプルプログラムは評価ボード別に異なるサブディレクトリに格納されています。これらのサブディレクトリに格納されたファイルの一覧は、NORTi¥WLAN¥DOC に格納されているプロセッサ/コンパイラ別の補足説明書を参照してください。

```
NORTi
  │
  ├ WLAN
  │   │
  │   ├ DOC
  │   │   ├ nowlan.pdf ·····  NORTi TCP/IP 対応 無線 LAN ドライバユーザーズガイド
  │   │   └ wlan_xx_xxx.txt···  プロセッサ/コンパイラ別補足説明書
  │   │
  │   ├ INC
  │   │   ├ hfa3841.h·····  PRISM 制御部 1 ヘッダファイル
  │   │   ├ prism2.h·····  PRISM 制御部 2 ヘッダファイル
  │   │   ├ wlanutil.h·····  ユーティリティヘッダファイル
  │   │   ├ wlanapi.h·····  制御用 API ヘッダファイル
  │   │   └ wlancom.h·····  各種定義
  │   │
  │   ├ SRC
  │   │   ├ hfa3841.c·····  PRISM 制御部 1 ソースファイル
  │   │   ├ prism2.c·····  PRISM 制御部 2 ソースファイル
  │   │   ├ wlanutil.c·····  ユーティリティソースファイル
  │   │   ├ wlanapi.c·····  制御用 API ソースファイル
  │   │   └ wlannet.c·····  TCP/IP プロトコルスタックインターフェース部
  │   │
  │   └ LIB
  │       └ CPU
  │           └ COMPILER·····  ライブラリ
  │
  └ SMP
      └ CPU
          └ BOARD·····  評価ボード用サンプルプログラム
```

4. ライブラリ

本ドライバのライブラリはプロセッサ、エンディアン、デバッグ情報の有無などにより、異なるファイル名で格納されています。さらに、マルチチャネル用にビルドされたライブラリが格納されている場合もあります。ライブラリのファイル名の一覧は補足説明書を参照してください。

4.1. ライブラリのファイル名

ライブラリのファイル名の大まかな規則を説明します。次のように、ファイル名の先頭４文字はデバッグ情報の有無により異なります。

n4wdxxxx.lib　：　デバッグ情報あり

n4wnxxxx.lib　：　デバッグ情報なし

通常は、デバッグ情報付きの n4wdxxx.lib をリンクしてください。デバッガで無線 LAN ドライバの内部を C ソースコードレベルでトレースすることができます。ロードモジュールファイルのサイズを小さくするため、デバッグ情報を含めない場合は n4wnxxx.lib をリンクしてください。ファイル名の先頭４文字以降はプロセッサに関する文字になります。さらに、エンディアンを区別する場合は、ビッグエンディアンを示すｂ、またはリトルエンディアンを示すｌがファイル名の末尾になります。

4.2. ライブラリの例

例として SuperH/SHC 用のライブラリのファイル名を示します。

デバッグ情報あり	デバッグ情報なし	説明
n4wdsh1.lib	n4wnsh1.lib	SH-1 ビックエンディアン用
n4wdsh2.lib	n4wnsh2.lib	SH-2 ビックエンディアン用
n4wdsh3b.lib	n4wnsh3b.lib	SH-3 ビックエンディアン用
n4wdsh3l.lib	n4wnsh3l.lib	SH-3 リトルエンディアン用
n4wdsh4b.lib	n4wnsh4b.lib	SH-4 ビックエンディアン用
n4wdsh4l.lib	n4wnsh4l.lib	SH-4 リトルエンディアン用

さらに、SuperH/SHC 用のパッケージでは、マクロ DIF_NUM=2 を定義してビルドしたマルチチャネル用のライブラリを格納しています。例えば、有線 LAN と無線 LAN の２つのインターフェースを使用する場合は、このマルチチャネル用のライブラリを使用してください。マルチチャネル用のライブラリのファイル名には m2 が追加されます。

デバッグ情報あり	デバッグ情報なし	説明
n4wdsh1m2.lib	n4wnsh1m2.lib	SH-1 ビックエンディアン用(DIF_NUM=2)
n4wdsh2m2.lib	n4wnsh2m2.lib	SH-2 ビックエンディアン用(DIF_NUM=2)
n4wdsh3m2b.lib	n4wnsh3m2b.lib	SH-3 ビックエンディアン用(DIF_NUM=2)
n4wdsh3m2l.lib	n4wnsh3m2l.lib	SH-3 リトルエンディアン用(DIF_NUM=2)
n4wdsh4m2b.lib	n4wnsh4m2b.lib	SH-4 ビックエンディアン用(DIF_NUM=2)
n4wdsh4m2l.lib	n4wnsh4m2l.lib	SH-4 リトルエンディアン用(DIF_NUM=2)

5. サンプルプログラム

NORTi¥SMP のサブディレクトリには本ドライバを各種評価ボード上に移植したサンプルプログラムを格納しています。さらに、マルチチャネル用のサンプルプログラムを格納している場合もあります。サンプルプログラムの一覧は補足説明書を参照してください。

5.1. シングルチャネル用サンプルプログラム

無線 LAN ドライバのサンプルプログラムは、NORTi に標準で付属のネットワークアプリケーション用サンプルプログラム(net7750.c 等)のネットワークインターフェースを無線 LAN に変更し、さらに無線 LAN ドライバ用のコマンドを追加しています。
例として、SuperH/SHC 用のパッケージに格納されている Solution Engine MS7750SSE01 のサンプルプログラムにおいて、無線 LAN 用に使用されるファイルを示します。

```
NORTi
 |
 ├ SMP
 |  └ SH
 |     └ MS7750S
 |        ├ net7750.c·····  サンプルプログラム本体
 |        ├ nonecfg.c·····  コンフィグレーション
 |        ├ wlancmd.c·····  無線 LAN ドライバ用サンプルコマンド(API 使用例)
 |        ├ wlanhw.h·····  無線 LAN ドライバ用ヘッダファイル
 |        ├ wlanhw.c·····  無線 LAN ドライバの初期化と割込みハンドラ
 |        ├ wlanmrshpc.h·····  PC カードコントローラ MR-SHPC 用ドライバヘッダファイル
 |        ├ wlanmrshpc.c·····  PC カードコントローラ MR-SHPC 用ドライバ本体
 |        |
 |        └ SHC9 (SHC Ver.9 用)
 |           ├ wlan7750S.hws·····  HEW ワークスペースファイル
 |           ├ wlan7750S_Hew
 |           |  └ wlan7750S_hew.hwp·····  HEW プロジェクトファイル
 |           ├ wlan7750Sbh.mak·····  メイクファイル      [ビッグエンディアン, ROM 用]
 |           ├ wlan7750Sbh.sub·····  サブコマンドファイル[ビッグエンディアン, ROM 用]
 |           ├ wlan7750Sbhr.mak·····  メイクファイル      [ビッグエンディアン, RAM 用]
 |           ├ wlan7750Sbhr.sub·····  サブコマンドファイル[ビッグエンディアン, RAM 用]
 |           ├ wlan7750Slh.mak·····  メイクファイル      [リトルエンディアン, ROM 用]
 |           ├ wlan7750Slh.sub·····  サブコマンドファイル[リトルエンディアン, ROM 用]
 |           ├ wlan7750Slhr.mak·····  メイクファイル      [リトルエンディアン, RAM 用]
 |           └ wlan7750Slhr.sub·····  サブコマンドファイル[リトルエンディアン, RAM 用]
 |
 └ WLAN
    └ LIB
       └ SH
          └ SHC9
             ├ n4wdsh4b.lib·····  無線 LAN ドライバ・ライブラリ[ビッグエンディアン]
             └ n4wdsh4l.lib·····  無線 LAN ドライバ・ライブラリ[リトルエンディアン]
```

サンプルプログラムでは次のマクロを使用します。

ファイル名	マクロ	説明
net7750.c	WLAN	有線 LAN を無線 LAN に置き換え、さらに無線 LAN 用のコマンドを追加します。
wlancmd.c	BIG_ENDIAN/LITTLE_ENDIAN	エンディアンに合わせて定義します。

無線 LAN ドライバの初期化用関数 wlan_ini は wlanhw.c に実装されています。TCP/IP プロトコルスタックの初期化用システム関数 tcp_ini を発行する前に wlan_ini を呼び出します。本ドライバの初期化方法の詳細な説明は「第 7 章 使用方法」を参照してください。

5.2. マルチチャネル用サンプルプログラム

マルチチャネル用のサンプルプログラムは、NORTi に標準で付属のネットワークアプリケーション用サンプルプログラムに、無線 LAN 用のインターフェースと無線 LAN 用のコマンドを追加し、有線 LAN と無線 LAN の２つのインターフェースを使用可能にしています。
例として、SuperH/SHC 用のパッケージに格納されている Solution Engine MS7750SSE01 のサンプルプログラムにおいて、無線 LAN 用に使用されるファイルを示します。

```
NORTi
 │
 ├ SMP
 │   └ SH
 │       └ MS7750S
 │           ├ net7750.c･･･････････  サンプルプログラム本体
 │           ├ nonecfg.c･･･････････  コンフィグレーション
 │           ├ wlancmd.c･･･････････  無線 LAN ドライバ用サンプルコマンド(API 使用例)
 │           ├ wlanhw.h･･･････････  無線 LAN ドライバ用ヘッダファイル
 │           ├ wlanmhw.c･･･････････  無線 LAN ドライバの初期化と割込みハンドラ
 │           ├ wlanmrshpc.h･･･････  PC カードコントローラ MR-SHPC 用ドライバヘッダファイル
 │           ├ wlanmrshpc.c･･･････  PC カードコントローラ MR-SHPC 用ドライバ本体
 │           │
 │           └ SHC9 (SHC Ver.9 用)
 │               ├ wlan7750S.hws･･･････････  HEW ワークスペースファイル
 │               └ wlanm7750S_Hew
 │                   └ wlanm7750S_hew.hwp･･･････････  HEW プロジェクトファイル
 │
 ├ SRC
 │   └ nonelan1.c･･･････････････････  ネットワーク・ドライバ・インターフェース本体
 │
 └ WLAN
     └ LIB
         └ SH
             └ SHC9
                 ├ n4wdsh4m2b.lib･･･････  無線 LAN ドライバ・ライブラリ
                 │                        [ビッグエンディアン, DIF_NUM=2]
                 └ n4wdsh4m2l.lib･･･････  無線 LAN ドライバ・ライブラリ
                                          [リトルエンディアン, DIF_NUM=2]
```

サンプルプログラムでは次のマクロを使用します。

ファイル名	マクロ	説明
net7750.c	WLAN_MULTI	無線 LAN 用のインターフェースと無線 LAN 用のコマンドを追加します。
nonelan1.c	NIF_NUM = 2	マルチチャネル用に DIF_NUM=2 を定義してビルドされたライブラリに合わせて、インターフェースの識別番号 DIF_NUM=2 を定義します。
	DIF_NUM = 2	
wlancmd.c	BIG_ENDIAN/LITTLE_ENDIAN	エンディアンに合わせて定義します。
	WLAN_MULTI	マルチチャネル用のコマンドを追加します。
wlanmhw.c	DHCP	無線 LAN 用のインターフェースで DHCP を使用します。
	CARD_INS	無線 LAN カードの活線挿抜に対応します。

マルチチャネル用のサンプルプログラムでは nonelan1.c を使用して、無線 LAN 用のネットワーク・ドライバ・インターフェースを追加します。また、無線 LAN ドライバのライブラリは DIF_NUM=2 を定義してビルドされたライブラリを使用します。
無線 LAN ドライバの初期化関数 wlan_nif_ini は wlanmhw.c に実装されています。TCP/IP プロトコルスタックの初期化用システム関数 tcp_ini を発行した後に、wlan_nif_ini を呼び出して無線 LAN 用のインターフェースを追加します。本ドライバの初期化方法の詳細な説明は「第 7 章 使用方法」を参照してください。

5.3. サンプルプログラムのコマンド

本無線 LAN ドライバ用のサンプルプログラムは、NORTi に標準で付属のネットワークアプリケーション用サンプルプログラムと同様に、ターミナルから ping、ftp、telnet などのコマンドを使用できます。さらに、無線 LAN 用のコマンドとして wlan コマンドを追加しています。

5.3.1. wlan コマンド概要

サンプルプログラムの実行後、?を入力するとコマンドの一覧が次のように表示されます。

```
>?
ip  ping  ftp  telnet  techo  uecho  dns  dhcp  mac  wlan
```

引数なしで wlan を入力すると簡単なヘルプが表示されます。wlan コマンドの機能とオプションの一覧は次のようになります。

No.	オプション	説明
1.	-p port_type	ポートタイプの設定
2.	-ch channel_number	使用チャネルの設定
3.	-id ssid	SSID の設定
4.	-wep wep_type	WEP の設定
5.	-f fragment_threshold	Fragment Threshold の設定
6.	-r rts_threshold	RTS/CTS Threshold の設定
7.	-cfg	設定データを無線 LAN カードに反映
8.	-wepkey wep_type	WEP Key の設定
9.	-info	設定とリンク状態の表示
10.	-apmac	接続先 MAC アドレスの表示
11.	-fver	ファームウェアバージョンの表示
12.	-ip	IP アドレスなどの表示（マルチチャネル用）
13.	-dhcp	DHCP の実行（マルチチャネル用）
14.	-start	無線 LAN の開始（マルチチャネル用）
15.	-stop	無線 LAN の停止（マルチチャネル用）
16.	-help	ヘルプの表示

1～6 のオプションはデータ設定用のオプションで、同時に複数の指定が可能です。設定内容は cfg オプションの指定後に無線 LAN カードに反映されます。例えば、SSID を"WLANSSID"、WEP 暗号化を 128bit に設定し、設定内容を無線 LAN カードに反映させる場合は、次のようにオプションを指定します。

```
>wlan -id WLANSSID -wep 2 -cfg
```

8～16 のオプションは主にドライバや無線 LAN カードの情報を表示するオプションです。これらのオプションは同時に複数指定することはできません。

5.3.2. wlan コマンド詳細

wlan コマンドのオプションの詳細を説明します。

port オプション

［機能］ポートタイプを設定します。

［形式］wlan -p port_type

［解説］port_type には、次の 0,1 のいずれかを指定してください。

0: アドホックネットワーク

1: インフラストラクチャネットワーク

設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

ch オプション

［機能］使用チャネルを設定します。

［形式］wlan -ch channel_number

［解説］channel_number には 1～14 の数値を指定してください。

設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

ssid オプション

［機能］ssid を設定します。

［形式］wlan -id ssid_string

［解説］ssid_string には、32 文字までの SSID 文字列を指定してください。

設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

wep オプション

［機能］WEP の設定を行います。

［形式］wlan -wep wep_type

［解説］wep_type には、次の 0,1,2 のいずれかを指定してください。

0: WEP を OFF にします。

1: 64bit WEP を設定します。

2: 128bit WEP を設定します。

設定する WEP Key は wepkey オプションで指定します。

設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

frag オプション

［機能］Fragment Threshold を設定します。

［形式］wlan -f fragment_threshold

［解説］fragment_threshold には 256～2346 の偶数を指定してください。
設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

rts オプション

［機能］RTS/CTS Threshold を設定します。

［形式］wlan -r rts_threshold

［解説］rts_threshold には 0～3000 の偶数を指定してください。
設定内容は cfg オプション指定後に無線 LAN カードに反映されます。

cfg オプション

［機能］設定データをカードに反映します。

［形式］wlan -cfg

［解説］設定データを無線 LAN カードに反映します。一度、無線 LAN カードを無効にしてからデータをカードに設定し、再度カードを有効にします。

wepkey オプション

［機能］WEP Key を登録します。

［形式］wlan -wepkey wep_type

［解説］wep オプションで設定する WEP Key を登録します。wep_type には次の 1 か 2 のいずれか指定してください。
1: 64bit WEP 用の WEP Key を登録します。
2: 128bit WEP 用の WEP Key を登録します。
登録は次の例のように 16 進で WEP Key を入力してください。

```
>wlan -wepkey 1
64bit WEP key setting.
  Key 0(10 hex digits) ->1112131415
  Key 1(10 hex digits) ->2122232425
  Key 2(10 hex digits) ->3132333435
  Key 3(10 hex digits) ->4142434445
```

info オプション

［機能］ドライバの設定とリンクの状態を表示します。

［形式］wlan -info

［解説］wlanapi_Get_configurations、wlanapi_GetBSSID、wlanapi_GetLinkStatus、wlanapi_GetLinkQuality を発行し、取得した情報を表示します。実行例は次のようになります。

```
>wlan -info
Type    : Infrastructure
Channel : 11
SSID    : WLANSSID
WEP     : OFF
Fragment: 2346
Rts     : 2432
AP MAC  : xx-xx-xx-xx-xx-x
Link Status : 1
Link Quality: 92
```

apmac オプション

［機能］接続先アクセスポイントの MAC アドレスを表示します。

［形式］wlan -apmac

［解説］API wlanapi_ GetBSSID を発行し、接続先アクセスポイントの MAC アドレスを表示します。

fver オプション

［機能］ファームウェアのバージョンを表示します。

［形式］wlan -fver

［解説］無線 LAN カードのファームウェアのバージョンを表示します。

ip オプション

［機能］IP アドレスなどを表示します。

［形式］wlan -ip

［解説］無線 LAN 用のネットワークインターフェースの MAC アドレス、IP アドレス、ゲートウェイのアドレス、サブネットマスクを表示します。本オプションはマルチチャネル用のサンプルプログラムで使用できます。

dhcp オプション

［機能］DHCP を実行します。

［形式］wlan -dhcp

［解説］無線 LAN 用のネットワークインターフェースに対して DHCP を実行します。本オプションはマルチチャネル用のサンプルプログラムで使用できます。

start オプション

［機能］無線 LAN を開始します。

［形式］wlan -start

［解説］無線 LAN 用のネットワークインターフェースを有効にします。本オプションはマルチチャネル用のサンプルプログラムで使用できます。

stop オプション

［機能］無線 LAN を停止します。

［形式］wlan -stop

［解説］無線 LAN 用のネットワークインターフェースを無効にしてから、無線 LAN カードの電源を OFF にします。無線 LAN カードを抜く前に、本コマンドを発行してください。本オプションはマルチチャネル用のサンプルプログラムで使用できます。

help オプション

［機能］ヘルプを表示します。

［形式］wlan -help

［解説］wlan コマンドのヘルプを表示します。

6. システム関数

本ドライバの初期化時に使用する構造体とシステム関数を説明します。

6.1. PRISMCFG 構造体

［定義］

```
typedef struct prismcfg {
    UH   NetworkType;      /* 0:Ad-Hoc / 1:Infrastructure / 3:Ad-Hoc Demo                */
    UH   Channel;          /* Channel 1-14                                               */
    UB   bssid[33];        /* ASCII data                                                 */
    UH   BasicRate;        /* Bit0:1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps       */
    UH   SupportedRate;    /* Bit0:1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps       */
    UH   DataRate;         /* Bit0:1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps       */
```

```
    UH   WEPType;            /* 0:WEP Off / 1:64bits / 2:128bits                    */
    UH   WEPDefaultKeyID;    /* Default WEP Key 0-3                                 */
    UB   WEPKey0[14];        /* WEP Key 0                                           */
    UB   WEPKey1[14];        /* WEP Key 1                                           */
    UB   WEPKey2[14];        /* WEP Key 2                                           */
    UB   WEPKey3[14];        /* WEP Key 3                                           */
    UH   AuthenticationType; /* 1:Open system / 2:Shared Key / 3:Open system + Shared Key */
    UH   MaxDataLength;      /* 350-2304                                            */
    UH   FragmentThreshold;  /* 256-2346(Even)                                      */
    UH   RtsCtsThreshold;    /* 0-3000(Even)                                        */
} PRISMCFG;
```

［解説］

コンフィグレーションテーブル（PRISMCFG 構造体）は無線 LAN カードに設定する初期化データを指定します。コンフィグレーションテーブルはユーザー側で用意し、初期化用関数 prism2_init の引数 cfg に本テーブルのポインタを指定します。コンフィグレーションテーブルの内容は次のようになります。

定義	説明
UH NetworkType	アドホックネットワークの場合は 0、インフラストラクチャネットワークの場合は 1 を指定します。 3 を指定して Ad-Hoc Demo に設定することも可能です。
UH Channel	使用チャネルを 1～14 で指定します。
UB bssid[33]	SSID を文字列で指定します。
UH BasicRate	各種転送速度を指定します。設定方法は次の対応ビットを 1 にします。 ビット 0: 1Mbps ビット 1: 2Mbps ビット 2: 5.5Mbps ビット 3: 11Mbps BasicRate/SupportedRate/DataRate は一般にアクセスポイントと接続しやすい値として 0x000F を指定します。
UH SupportedRate	
UH DataRate	
UH WEPType	WEP を次のように 0～2 で指定します。 0: WEP を使用しない 1: 64bits WEP 2: 128bits WEP
UH WEPDefaultKeyID	使用する WEPKey0～3 を 0～3 で指定します。

UB WEPKey0[14] UB WEPKey1[14] UB WEPKey2[14] UB WEPKey3[14]	WEP Key を指定します。64bit WEP の場合は WEPKey*[0]～WEPKey*[4]を指定します。128bit WEP の場合は WEPKey*[0]～WEPKey*[12]を指定します。
UH AuthenticationType	認証方法を次のように 1～3 で指定します。 1: オープンシステム認証 2: シェアードキー認証 3: 1 または 2 のどちらか一方を使用 AuthenticationType は一般には 3 を指定します。
UH MaxDataLength	転送データの最大サイズを 0～2304 で指定します。 MaxDataLength は一般にデフォルト値 2304 を指定します。
UH FragmentThreshold	フラグメント化しきい値を 256～2346 の偶数で指定します。 FragmentThreshold は一般にデフォルト値 2346 を指定します。
UH RtsCtsThreshold	RTS しきい値を 0～3000 の偶数で指定します。 RtsCtsThreshold は一般にデフォルト値 2432 を指定します。

6.2. prism2_init

[形式] prism2_t* prism2_init(UW iobase, PRISMCFG *cfg, PRISM_FP func)

iobase PC カードの I/O 空間のベースアドレス

cfg コンフィグレーションテーブル(PRISMCFG 構造体)のポインタ

func ユーザー定義の初期化関数のポインタ

[戻値]

デバイスディスクリプタのポインタ：正常終了

NULL：失敗

[解説]

無線 LAN カードを初期化します。iobase には PC カードの I/O 空間のベースアドレスを指定してください。cfg にはコンフィグレーションテーブル(PRISMCFG 構造体) のポインタを指定してください。このテーブルはユーザー側で用意してください。func には、ユーザー定義の初期化用関数のポインタを指定してください。ユーザーが独自に定義した初期化用関数をドライバ内部で初期化時(tcp_ini と wlanapi_Set_configuration の発行時)に発行します。必要がない場合は NULL を指定してください。

関数が成功した場合はデバイスディスクリプタ (prism2_t 構造体) のポインタを返しま

す。デバイスディスクリプタは本ドライバの API を発行するときに必要になります。関数が失敗した場合は NULL を返します。

6.3. wlan_get_macaddr

[形式] void wlan_get_macaddr(UB *macaddr);

macaddr MAC アドレス格納先変数のポインタ

[戻値]

なし

[解説]

無線 LAN カードから取得した MAC アドレスを macaddr で指定された変数に格納します。

変数 macaddr は次のように定義します。

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

6.4. probe_cpu_speed/probe_cpu_speed_io/probe_cpu_speed_iol

[形式] void probe_cpu_speed(void);

[戻値]

なし

[形式] void probe_cpu_speed_io(volatile WLAN_DIO_TYP *dummy_io);

dummy_io ダミー読み込み可能な IO ポートなどのアドレス(UH 型)

[戻値]

なし

[形式] void probe_cpu_speed_iol(volatile WLAN_DIO_TYPL *dummy_iol);

dummy_iol ダミー読み込み可能な IO ポートなどのアドレス(UW 型)

[戻値]

なし

[解説]

無線 LAN カードのレジスタへの連続アクセス時にはウェイトが必要です。ウェイトは wlanutil.c に実装されたウェイト用関数 udelay を使用します。ウェイト用関数ではダミーの読み込み処理を必要な時間だけループで繰り返し実行しますが、そのループ回数を本関数で計測します。

probe_cpu_speed を発行すると、ウェイト（ダミーの読み込み処理）はメモリ上の変数を読み込んで実現します。しかし、メモリ上の変数を読み込むだけでは、コンパイラの最

適化などによって、正確なウェイトが実現できない場合があります。この場合は probe_cpu_speed の替わりに probe_cpu_speed_io または probe_cpu_speed_iol を使用します。引数 dummy_io(UH 型)または dummy_iol(UW 型)には読み込み可能な IO ポートやレジスタを指定してください。probe_cpu_speed_io または probe_cpu_speed_iol を発行すると、引数で指定された IO ポートやレジスタをダミーで読み込むことでウェイトを実現します。本関数は無線 LAN ドライバを初期化する前に必ず発行してください。

7. 使用方法

本ドライバの使用方法を説明します。本ドライバの使用にあたっては、はじめに PC カードの制御部、つまり PC カードコントローラなどを初期化し、無線 LAN カードを IO モードに設定します。次に、ユーザーで定義した初期化用データを使用して、無線 LAN カードを初期化します。さらに、TCP/IP プロトコルスタックの初期処理の中で、無線 LAN カードの使用を有効にします。無線 LAN カードを有効にした後は、API を使用して無線 LAN カードの設定を変更し、さらに無線 LAN カードの状態を取得することが可能です。

7.1. 無線 LAN ドライバが使用する資源

本無線 LAN ドライバの内部では OS の資源としてセマフォを 1 個使用します。また、無線 LAN カード用の割り込みサービスルーチンが必要になります。カーネルのコンフィグレーションでは、これらの無線 LAN ドライバ用の資源を確保してください。

7.2. 初期設定データ

無線 LAN カードを初期化する前に初期化用のデータをユーザーで用意します。具体的な実装は次のようになります。デバイスディスクリプタ（prism2_t 構造体）のポインタを格納する変数を定義してください。さらに、コンフィグレーションテーブル（PRISMCFG 構造体）を定義してください。コンフィグレーションテーブルのデータは接続先に合わせて指定してください。

```
prism2_t *prism_desc;
PRISMCFG prism_config = {
  /* 基本設定 */
  1,                  /* 0:Ad-Hoc / 1:Infrastructure / 3:Ad-Hoc Demo */
  11,                 /* チャネル  1～14 */
  {"WLANSSID"},  /* SSID */
  0x000F,    /* Basic Rate        Bit0: 1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps */
  0x000F,    /* Supported Rate   Bit0: 1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps */
  0x000F,    /* Data Rate         Bit0: 1Mbps / Bit1:2Mbps / Bit2:5.5Mbps / Bit3:11Mbps */
  /* WEP 設定 */
  0,          /* WEP Type              0: WEP 無し  /  1: 64bits WEP  /  2: 128bits WEP */
  0,          /* Default WEP Key     0～3 */
  {0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d},  /*WEP Key 1*/
```

```
    {0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d},   /*WEP Key 2*/
    {0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d},   /*WEP Key 3*/
    {0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d},   /*WEP Key 4*/
    3,          /* Authentication Type   1:Open system   /   2:Shared Key   /   3:Open+Shared */

    2304,     /* Max Data Length         350～2304 */
    2346,     /* Fragment Threshold     256～2346(偶数を指定) */
    2432      /* RTS Threshold           0～3000(偶数を指定) */
};
```

7.3. 無線 LAN カードの初期化

はじめに PC カードコントローラ部と無線 LAN カードを初期化します。具体的な実装は次のようになります。

```
ER wlan_ini(void)
{
      ER ercd;

    probe_cpu_speed_io((volatile WLAN_DIO_TYP*)MR_CHIPINFO);   /* CPU 速度計測 */

    ercd = PCCard_init();   /* PC カードコントローラの初期化 */
    if (ercd != E_OK)
        goto ERROR;

    port_outw(ATR_WIN+0x03e0L, 0x41);   /* 無線 LAN カードを I/O モードに設定 */
    prism_desc = prism2_init(IO_WIN, &prism_config, NULL);    /* 無線 LAN カードの初期化 */
    if (prism_desc == NULL)
        goto ERROR;
    wlan_get_macaddr(ethernet_addr);     /* 無線 LAN カードの MAC アドレスを格納 */

    return E_OK;
}
```

関数 probe_cpu_speed_io または probe_cpu_speed は本ドライバのユーティリティ関数であり、ディレイ用関数のためにシステムの速度を測定します。関数 PCCard_init は PC カード

コントローラを初期化する関数で、ハードウェアに合わせてユーザー側で実装する必要があります。一般に、この関数ではPCカードコントローラを初期化し、無線LANカードの電源をONにします。次にPCカードのアトリビュートメモリ空間 のコンフィグレーションオプションレジスタに値をセットして、無線 LAN カードを I/O モードにします。さらに、初期化用関数 prism2_init を発行し、無線 LAN カードを初期化します。最後に関数wlan_get_macaddr(ethernet_addr)を発行し、無線LANカードから取得したMACアドレスをethernet_addr に格納します。

7.4. TCP/IP プロトコルスタックの初期化

次に関数 tcp_ini を発行しTCP/IP プロトコルスタックを初期化します。関数 tcp_ini は内部で本ドライバの関数 lan_ini を発行します。関数 lan_ini では、PC カードの割込みハンドラを登録し､コンフィグレーションテーブルのデータを無線LANカードに設定してから、無線LANカードを有効にします。関数 lan_ini の実装は次のようになります。

```
ER lan_ini(UB *macaddr)
{
    ER      err;
    Int     result;

    prism2_desc = prism2_get_pdesc();   /* デバイスディスクリプタを取得 */
    if (prism2_desc == (prism2_t *)0)
        return E_SYS;

    snd_len   = 0;
    psnd_pkt = &snd_pkt[0];      /* 送信パケットの初期化 */

    err = wlan_def_int();     /* 割込みハンドラ登録関数 */
    if (err != E_OK)
        return err;

    result = wlan_Setcfg_Defaults(prism2_desc);   /* コンフィグレーションテーブルを設定 */
    if(result)
        return err;

    err = prism2_cmd_enable(prism2_desc);   /* 無線 LAN カードを有効にする */
```

```
    tslp_tsk(real_tick(20));

    return err;
}
```

関数 lan_ini では関数 wlan_def_int を発行し、無線 LAN カード（PC カード）の割込みハンドラを登録します。関数 wlan_def_int はハードウェアに合わせてユーザー側で実装する必要があります。割込みハンドラの登録後に関数 wlan_Setcfg_Defaults を発行し、コンフィグレーションテーブルのデータを無線 LAN カードにセットします。この関数の内部では、関数 prism2_init の引数 func で指定した“ユーザー定義の初期化用関数”を発行します。最後に関数 prism2_cmd_enable を発行し、無線 LAN カードの機能を有効にします。有効にした時点で、無線 LAN カードと接続先とのリンクが確立します。無線 LAN カードによってはカードの LED によってリンクの確立を示す場合があります。

7.5. ユーザー実装関数

本ドライバの移植にあたって、ハードウェアに大きく依存する部分は PC カードの制御部です。一般に、ユーザーは PC カードコントローラの初期化関数と割込みハンドラ登録関数をハードウェアに合わせて実装する必要があります。

7.5.1. PC カードコントローラの初期化

7.3.”無線 LAN カードの初期化”では関数 PCCard_init で PC カードコントローラを初期化していますが、この関数名はユーザー側で自由に決めることができます。関数 prism2_init を発行する前に、PC カードコントローラを初期化し、PC カードのアトリビュートメモリ空間のレジスタにアクセスできるようにしてください。本ドライバは I/O モードで無線 LAN カードを制御します。よって、アトリビュートメモリ空間のレジスタに適切なデータをセットして、PC カードを I/O モードにする必要があります。
一般に関数 PCCard_init の内部では PC カードコントローラを初期化し、カードの電源電圧を調査してからカードの電源を ON にします。さらに、PC カードの割込みに関して設定が必要になる場合があります。実装例は次のようになります。

```
ER PCCard_init(void)
{
    /* PC カードコントローラの初期化 */
    sfr_set(PCC_CTL, CTL_RESET);
```

```
    dly_tsk(100/MSEC);
    sfr_clr(PCC_CTL, CTL_RESET);

    if (PCCard_check() == E_OK) { /* PC カードの有無を確認 */
        PCCard_pwr_ON(); /* PC カードの電源 ON */

      #ifndef LAN_POLL
        sfr_set(PCC_INT, 0x0000); /* PC カードの割込みの設定 */
        sfr_set(PCC_INT, INT_RDY);
      #endif
    } else
        return E_SYS;

    return E_OK;
}
```

7.5.2. 割込みハンドラ登録関数

TCP/IP プロトコルスタックの初期化用システム関数 tcp_ini は、内部で本ドライバの関数 lan_ini を発行します。さらに、関数 lan_ini ではユーザー側で実装した割込みハンドラ登録関数 wlan_def_int を発行します。この割込みハンドラ登録関数は次の形式で実装してください。

[形式] ER　wlan_def_int(void)；

[戻値]

E_OK	正常終了
上記以外	エラー

関数 wlan_def_int は本ドライバ内の関数 wlan_intr を割込みハンドラとして登録します。実装例は次のようになります。

```
extern void wlan_intr(void);
INTHDR wlan_int(void);
const T_DINH dinh_wlan = { TA_HLNG, wlan_int, 5 };

ER wlan_def_int(void)
```

```
{
    ER ercd;

    ercd = def_inh(PCCARD_INT, &dinh_wlan);
    if (ercd == E_OK)
        ercd = ena_int(PCCARD_INT); /* 割込みを有効にする */

    return ercd;
}

INTHDR wlan_int(void)
{
    ent_int();
    wlan_intr();
    ret_int();
}
```

8. API

本章では API について説明します。無線 LAN カードを有効にした後、ユーザーは API を使用して無線 LAN カードの設定を変更し、さらに無線 LAN カードの状態を取得することが可能です。なお、API は必ず使用する必要はありません。無線 LAN カードの初期化後に、設定の変更や状態の取得が必要な場合は API を使用します。

8.1. API 概要

API 名とその機能の一覧は次のようになります。

設定用 API

1. wlanapi_TxRates　送信 Rate の設定
2. wlanapi_SuppotedRates　Supported Rates の設定
3. wlanapi_BasicRates　BroadCast/MultiCast/Mgmt の送信 Rate の設定
4. wlanapi_Wep　WEP キーの設定
5. wlanapi_WepDefaultKeyID　WEP キーID の設定
6. wlanapi_AuthType　認証タイプの設定
7. wlanapi_SSID　SSID の設定
8. wlanapi_PortType　Port タイプの設定
9. wlanapi_OwnChannel　チャネルの設定
10. wlanapi_FragThresh　フラグメントしきい値の設定
11. wlanapi_RtsThresh　RTS/CTS しきい値の設定
12. wlanapi_Set_configurations　設定データをカードに反映
13. wlanapi_Get_configurations　設定データを取得

詳細設定用 API

14. wlanapi_Getcfg_general　指定データの取得
15. wlanapi_Setcfg_general　指定データの設定

状態取得用 API

16. wlanapi_GetLinkStatus　リンクの状態を取得
17. wlanapi_GetLinkQuality　リンクの品質を取得
18. wlanapi_GetBSSID　アクセスポイントの MAC アドレスを取得

各 API の第 1 引数にはデバイススクリプタ(初期化用関数 prism2_init()の戻値)を指定して

ください。1～11 のデータ設定用 API はドライバ内部のデバイスディスクリプタの内容を更新するだけであり、更新内容はすぐに無線 LAN カードに反映されません。更新内容を無線 LAN カードに反映させるためには、12 wlanapi_Set_configurations を発行する必要があります。
一般に無線 LAN カードの制御においては、14 と 15 の詳細設定用の API を使用する必要はありません。これらの API は 1～11 の API で設定できない特別なデータを無線 LAN カードに設定する API です。詳細設定用の API の使用にあたっては、PRISM の詳細を理解している必要があります。
16～18 は無線 LAN カードの状態を取得する API です。

次に API の使用例を示します。例えば Port タイプを変更する場合は次のように API を発行します。

```
prism2_t * prism_desc;
UH porttype;
int result;

porttype = 1;
result = wlanapi_PortType(prism_desc, porttype);   /* Infrastructure モードに設定 */
if ( result != E_OK )
    goto ERROR;
result = wlanapi_Set_configurations (prism_desc);    /* 無線 LAN カードに反映 */
if ( result != E_OK )
    goto ERROR;
```

API の使用例に関しては、サンプルプログラムの wlancmd.c を参考にしてください。

8.2. API 詳細

各 API の詳細は次のようになります。

wlanapi_TxRates

［機能］送信 Rate の設定
［形式］int wlanapi_TxRates(prism2_t* prism_desc, UH rate);
prism_desc　デバイスディスクリプタのポインタ
rate　　送信 Rate

［戻値］

E_OK　正常終了

E_PAR　パラメータの不正

［解説］

デバイスディスクリプタに送信Rateを設定します。rateは次のように指定してください。

Bit0: 1Mbps

Bit1: 2Mbps

Bit2: 5.5Mbps

Bit3: 11Mbps

設定内容を無線LANカードに反映させる場合はwlanapi_Set_configurationsを発行してください。

wlanapi_SuppotedRates

［機能］Supported Rateの設定

［形式］int wlanapi_SuppotedRates(prism2_t* prism_desc,UH rate);

prism_desc　デバイスディスクリプタのポインタ

rate　Supported Rate

［戻値］

E_OK　正常終了

E_PAR　パラメータの不正

［解説］

デバイスディスクリプタにSupported Rateを設定します。rateは次のように指定してください。

Bit0: 1Mbps

Bit1: 2Mbps

Bit2: 5.5Mbps

Bit3: 11Mbps

設定内容を無線LANカードに反映させる場合はwlanapi_Set_configurationsを発行してください。

wlanapi_BasicRates

［機能］BroadCast/MultiCast/Mgmtの送信Rateの設定

［形式］int wlanapi_BasicRates(prism2_t* prism_desc, UH rate);

prism_desc　デバイスディスクリプタのポインタ

rate　送信Rate

［戻値］
E_OK　正常終了
E_PAR　パラメータの不正
［解説］
デバイスディスクリプタに BroadCast/MultiCast/Mgmt の送信 Rate を設定します。rate は次のように指定してください。
Bit0: 1Mbps
Bit1: 2Mbps
Bit2: 5.5Mbps
Bit3: 11Mbps
設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_Wep
［機能］WEP キーの設定
［形式］int wlanapi_Wep(prism2_t* prism_desc,UH weptype, UB *Key0, UB *key1, UB *key2, UB *key3);
prism_desc　デバイスディスクリプタのポインタ
weptype　WEP 選択(0:なし / 1:64Bit / 2:128Bit)
*key0～*key3　WEP キーのポインタ
［戻値］
E_OK　正常終了
E_PAR　パラメータの不正
［解説］
デバイスディスクリプタに WEP キーを設定します。weptype には設定する WEP の種類(0:なし / 1:64Bit / 2:128Bit)を指定してください。設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_WepDefaultKeyID
［機能］WEP キーID の設定
［形式］int wlanapi_WepDefaultKeyID(prism2_t* prism_desc, UH keyid);
prism_desc　デバイスディスクリプタのポインタ
keyid　WEP キーID(0～3)
［戻値］
E_OK　正常終了
E_PAR　パラメータの不正

［解説］
デバイスディスクリプタに使用する WEP キーID を設定します。設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_AuthType

［機能］認証タイプの設定
［形式］int wlanapi_AuthType(prism2_t * prism_desc, UH authtype);
prism_desc　デバイスディスクリプタのポインタ
authtype　認証タイプ(1:Open System　2:Shared Key　3:Open System+Shared Key)
［戻値］
E_OK　正常終了
E_PAR　パラメータの不正
［解説］
デバイスディスクリプタに認証タイプを設定します。設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_SSID

［機能］SSID の設定
［形式］int wlanapi_SSID(prism2_t* prism_desc, UB *ssid);
prism_desc　デバイスディスクリプタのポインタ
ssid　SSID の文字列のポインタ(NULL 文字で終端)
［戻値］
E_OK　正常終了
E_PAR　パラメータの不正
［解説］
デバイスディスクリプタに SSID を設定します。ssid に指定する SSID の文字列は NULL 文字で終端してください。設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_PortType

［機能］Port タイプの設定
［形式］int wlanapi_PortType(prism2_t* prism_desc, UH porttype);
prism_desc　デバイスディスクリプタのポインタ
porttype　Port タイプ(0:Ad-Hoc / 1:Infrastructure / 3:Ad-Hoc Demo)
［戻値］
E_OK　正常終了

E_PAR　パラメータの不正

［解説］

デバイスディスクリプタにPortタイプを設定します。設定内容を無線LANカードに反映させる場合はwlanapi_Set_configurationsを発行してください。

wlanapi_OwnChannel

［機能］チャネルの設定

［形式］int wlanapi_OwnChannel(prism2_t* prism_desc, UH channel);

prism_desc　デバイスディスクリプタのポインタ

channel　チャネル(1～14)

［戻値］

E_OK　正常終了

E_PAR　パラメータの不正

［解説］

デバイスディスクリプタにチャネルを設定します。channelには使用するチャネル(1～14)を指定してください。設定内容を無線LANカードに反映させる場合はwlanapi_Set_configurationsを発行してください。

wlanapi_FragThresh

［機能］フラグメントしきい値の設定

［形式］int wlanapi_FragThresh(prism2_t* prism_desc, UH thresh);

prism_desc　デバイスディスクリプタのポインタ

thresh　フラグメントしきい値(256～2346の偶数)

［戻値］

E_OK　正常終了

E_PAR　パラメータの不正

［解説］

デバイスディスクリプタにフラグメントしきい値を設定します。threshには256～2346の偶数を指定してください。設定内容を無線LANカードに反映させる場合はwlanapi_Set_configurationsを発行してください。

wlanapi_RtsThresh

［機能］RTS/CTSしきい値の設定

［形式］int wlanapi_RtsThresh(prism2_t* prism_desc, UH thresh);

prism_desc　デバイスディスクリプタのポインタ

thresh　RTS/CTSしきい値(0～3000の偶数)

［戻値］
E_OK　正常終了
E_PAR　パラメータの不正
［解説］
デバイスディスクリプタに RTS/CTS しきい値を設定します。thresh には 0～3000 の偶数を指定してください。設定内容を無線 LAN カードに反映させる場合は wlanapi_Set_configurations を発行してください。

wlanapi_Set_configurations
［機能］設定データをカードに反映
［形式］int wlanapi_Set_configurations(prism2_t* prism_desc);
prism_desc　デバイスディスクリプタのポインタ
［戻値］
E_OK　　正常終了
E_TMOUT　タイムアウトエラー
［解説］
デバイスディスクリプタに設定したデータを無線 LAN カードに反映します。本 API は無線 LAN カードを無効にしてから設定データをカードにセットし、カードを有効に戻します。また、本 API 内部では初期化関数 prism2_init()で指定したユーザー定義の初期化用関数を発行します。

wlanapi_Get_configurations
［機能］設定データの取得
［形式］int wlanapi_Get_configurations(prism2_t * prism_desc, PRISMCFG* prismcfg);
prism_desc　デバイスディスクリプタのポインタ
prismcfg　　コンフィグレーションテーブルのポインタ
［戻値］
E_OK　　正常終了
E_PAR　　パラメータエラー
［解説］
ドライバ内で保持しているコンフィグレーションテーブルの内容を prismcfg で指定された変数に格納します。prismcfg にはユーザーで用意した PRISMCFG 構造体のポインタを指定してください。取得したデータはドライバ内で保持しているデータであり、必ずしも無線 LAN カードに反映されているとは限りません。

wlanapi_Getcfg_general

［機能］指定データの取得

［形式］int wlanapi_Getcfg_general(prism2_t* prism_desc, UH rid, void* buf, UH len);

prism_desc　デバイスディスクリプタのポインタ

rid　RID

buf　取得データの格納バッファ

bufflen　buf のサイズ(バイト)

［戻値］

E_OK　正常終了

E_PAR　パラメータの不正

E_OBJ　格納バッファの不足

E_TMOUT　タイムアウトエラー

［解説］

無線 LAN カードから RID で指定した設定データを読み出し、buf に読み出したデータを格納します。本 API 内部で buf に対してワードアクセスをするため、buf は偶数番地から配置してください。bufflen は buf のサイズ(バイト、偶数)を指定してください。buf のサイズが不足している場合は、読み出したデータを bufflen で指定されたサイズまで buf に格納し、E_OBJ を返します。バッファには次の形式でデータが格納されます。

buf[0] Record Length(ワード長、RID と Data の長さ)

buf[1] RID

buf[2] Data

buf[3] Data

buf[4] Data

buf[5] Data

…

バッファのデータはカードから読み出した状態のまま格納しています。よって、ビッグエンディアン環境の場合、Stream でない Integer なデータはエンディアンに合わせてスワップをしてから使用してください。リトルエンディアン環境の場合、スワップは不要です。

wlanapi_Setcfg_general

［機能］指定データの設定

［形式］ int wlanapi_Setcfg_general(prism2_t* prism_desc, UH rid, void* buf, UH reclen);

prism_desc　デバイスディスクリプタのポインタ
rid　　RID
buf　　設定データの格納バッファ
reclen　　buf のサイズ(ワード)

[戻値]

E_OK　　正常終了
E_PAR　　パラメータの不正
E_TMOUT　タイムアウトエラー

[解説]

無線 LAN カードへ RID で指定したデータを設定します。設定するデータが Dynamic に分類される場合､本API 発行時に設定内容がカードに反映されます｡設定するデータがStaic に分類される場合、設定内容は本 API 発行時にカードに反映されず、関数 wlanapi_Set_configurations()を発行した時点でカードに反映されます。reclen には設定する RID と Data の合計サイズ(ワード)を指定してください。buf には設定するデータを次の形式で格納してください。

buf[0] Record Length(ワード長、RID と Data の長さ)
buf[1] RID
buf[2] Data
buf[3] Data
buf[4] Data
buf[5] Data
...

buf[0]の Record Length と buf[1]の RID は､本 API 内部でエンディアンに合わせてスワップをしてからカードに書き込みます。buf[2]以降の Data は本 API 内部でスワップをしません。よって、ビッグエンディアン環境の場合、Stream でない Integer なデータはユーザー側でエンディアンにあわせてスワップしてください。リトルエンディアン環境の場合、スワップは不要です。

wlanapi_GetLinkStatus

[機能] リンクの状態を取得

[形式] UH　wlanapi_GetLinkStatus(prism2_t * prism_desc);

[戻値]

1　　リンクが確立している
0　　リンクが確立していない

［解説］
アクセスポイントと無線 LAN カードのリンク状態を返します。
本 API は Infrastructure モードの時のみ使用できます。Ad-Hoc と Ad-Hoc Demo モードの場合はリンクの状態を取得することはできません。
［制限事項］
無線 LAN カードの制限により、次の各場合において、本 API の戻値が正確でない場合があります。いずれの場合も、リンクが確立していない状態ですが、本 API が 1 を返す場合があります。

- アクセスポイントが WEP を使用し、本ドライバが WEP を使用しない設定の場合。
- アクセスポイントと本ドライバが WEP を使用しているが、二つの WEP Key が異なる場合。
- アクセスポイントと本ドライバが WEP を使用して接続した後、アクセスポイントを WEP 無しに変更した場合。

wlanapi_GetLinkQuality
［機能］リンクの品質を取得
［形式］UH wlanapi_GetLinkQuality(prism2_t * prism_desc);
［戻値］
0～92 リンク品質
［解説］
リンクの品質を返します。リンクの品質は PRISM 独自の指標(最小値 0、最大値 92)で、数値が大きいほどリンクの品質が良いことを示します。
本 API は Infrastructure モードの時のみ使用できます。Ad-Hoc と Ad-Hoc Demo モードの場合はリンクの品質を取得することはできません。
また、本 API で取得できるリンク品質はリンクが確立しているときのみ有効です。リンクが確立していないときの戻値は不定です。
本 API は無線 LAN カードにアクセスして情報を読み取るため、無線 LAN カードに負荷がかかります(wlanapi_GetLinkStatus ()は無線 LAN カードにアクセスしません)。よって、連続して発行する場合は 2～3 秒以上の間隔をあけて発行してください。

wlanapi_GetBSSID
［機能］アクセスポイントの MAC アドレスを取得
［形式］int wlanapi_GetBSSID(prism2_t * prism_desc, UB *bssid);
［戻値］
E_OK 正常終了

E_PAR　　パラメータの不正
E_TMOUT　タイムアウトエラー

［解説］

接続先アクセスポイントの MAC アドレスを bssid で指定されたポインタに格納します。bssid には MAC アドレス(6 バイトのデータ)が格納できる変数のポインタを指定してください。

本 API は Infrastructure モードの時のみ使用できます。また、本 API はリンクが確立しているときのみ有効です。リンクが確立していない場合は不定な値が bssid に格納されます。

9. ライブラリ生成用マクロ

本無線 LAN ドライバはライブラリで提供されるため、通常はユーザーでライブラリをビルドする必要はありません。特別なカスタマイズが必要な場合のみ、ユーザーでライブラリを再度ビルドしてください。カスタマイズでは次のマクロが使用できます。

BIG_ENDIAN

ビックエンディアン用のライブラリを作成する場合に指定します。本ドライバはデフォルトではリトルエンディアン用に実装されています。よって、リトルエンディアン用のライブラリ作成ではエンディアンを識別するマクロは不要です。

DIF_NUM

マルチチャネルで TCP/IP プロトコルスタックを使用する場合に、ネットワークドライバインターフェースの識別番号を指定します。DIF_NUM は 0,1,2 のいずれかを定義します。

WLAN_IO

無線 LAN カードへのアクセス用関数をユーザーが独自に実装する場合に指定します。このマクロは無線 LAN カードへのアクセス時に特別な処理(wait 等)が必要な場合に使用します。ユーザー独自の無線 LAN カードへのアクセス関数は次のように実装します。

```
UH wlan_outw(UW adr, UH data)
{
    udelay(10);     /* 特別な処理 */
    *(volatile UH*)(adr) = data;
```

```
    return data;
}

UH wlan_inw(UW adr)
{
    udelay(10);     /* 特別な処理 */
    return *(volatile UH*)(adr);
}
```

NORTi TCP/IP 対応 無線 LAN ドライバ ユーザーズガイド

株式会社ミスポ http://www.mispo.co.jp/

〒213-0012 川崎市高津区坂戸 3-2-1
TEL 044-829-3381 FAX 044-829-3382
一般的なお問い合わせ sales@mispo.co.jp
技術サポートご依頼 norti@mispo.co.jp