MeshNetics

ZigBit™

ZigBit 開発キット 2.0

ユーザース ガイド

文書番号. S-ZDK-451~01 v.2.0 J0.1

注意

- このマニュアルはMeshNetics社の「ZigBit™Development Kit 2.0 User's Guide」(文書番号 S-ZDK-451~01 v.2.0) をKenConsulting Inc.が翻訳したものです。
- この原文の版権はMeshNetics社に、翻訳の版権はKenConsulting Inc.に帰属します。
- 翻訳は原文に沿って行っていますが、翻訳の正確性を保証するものではありません。翻訳内容に疑義が生じた場合は原文の表現をもって正確な表現とします。
- KenConsultingは、このマニュアルの内容を将来予告無しに変更することがあります。
- このマニュアルの内容の一部、または全部をKenConsulting からの書面による許可無く無断で転載することは禁止します。
- KenConsultingはここで説明する製品の使用についていかなる責任も負わないものとします。
- **このマニュアルは表紙に表記のある部署だけでお使いください。それ以外の部署、団体だお使いになる場合は別途お買い求めください。**
- このマニュアル、製品に関するご質問は *support@kenconsul.com* へメールしてください。

翻訳の履歴

版番号	変更内容	年/月/日
S-ZDK-451~01 v.2.0 J0.1	最初の試訳	08/5/01

免責事項

MeshNetics はこのマニュアルの原文内のすべての情報が配付時には正確であることを信じます。MeshNetics は本書の内容に関して、将来予告無しに変更することがあります。 最新のバージョンの原文は MeshNetics 社の Web サイトから入手してください。

MeshNetics はここで説明する製品の使用についていかなる責任も負わないものとし、また、MeshNetics 社の特許権を含むいかなる知的資産やソフトウェアのライセンスを提供するものではありません。

MeshNetics は MeshNetics の保証基準に従い、提供するハードウェア製品に対して販売時に適用可能な製品仕様通りの性能を保証します。 MeshNetics はこの保証をサポートする為に必要と見做すテストと他の品質管理技術を使います。 しかし、政府機関が要求するものを除いて、各製品のパラメータを全てテストするわけではありません。

商標

MeshNetics[R]、ZigBit、ZigBeeNet、ZigBeeNet、SensiLink、MeshNetics、ZigBit ロゴは MeshNetics の商標です。

他の全ての製品名、商品名、商標、ロゴ、サービス名はそれらのそれぞれの所有者の資産です。

技術サポート

MeshNetics の専門家及び KenConsulting Inc.がご質問にお答えし、適切なサポートを提供します。MeshNetics からのサポートについては開発キット添付の契約書をご覧ください。

開発サポート

ソフトウェアのカスタマイズサービスは KenConsulting Inc.がお客さまのご要望をお聞きし、その詳細を取り纏めたうえで MeshNetics 社と合意して頂き、その条件のもとで提供致します。

コンタクト情報

KenConsulting Inc.

メール: *support@kenconsul.com*

ホームページ: www.kenconsul.com

目次

図の一覧

表の一覧

1. はじめに

対象とする読者と目的

この文書はZigBit™ Development Kit (ZDK)を使って作業するエンジニアやソフトウェア開発者を対象としています。このキットはZigBitモジュールとZigBeeNet組込ソフトウェアの性能と機能を評価しZigBeeNetAPIの上にZigBeeアプリケーションを開発して頂くためのものです。

安全上のご注意

本製品は電気に対して敏感なエレクトロニクス機器を含んでいます。 そのような機器を使う時に必要な予防策を取ってください。 MeshNetics は静電放電現象から製品コンポーネントを保護するためにベストを尽くしますが、弊社はお客さまに、適切な接地などにより静電気の被害を避けるための一般的なガイドラインに従うようお勧めしています。

本製品は自由な環境で電波を放射する機器に適用される FCC(パート 15), IC, ETSI(CE)規則に準拠します。 本製品がお客さまのご使用の地域の規制に適合することを確かめてください。

本製品のハードウェア、そのコンポーネントに対する分解、改造、不適切な使用により受信、送信の為の電波のレベルが制御不能になることがあります。 その為に電波の放出レベルが許容限度を超えた場合、結果として違反行為となることがあります。

予防措置

本製品はマイクロ波帯の電波を放射します。 この放射は低レベル(2mW 以下)と考えられますが、電磁界による有害なインパクトからオペレータを保護することは妥当と思われます。 本製品に電源を投入する時、オペレータは PCB アンテナとボード本体に触れない様にしてください。 オペレータとアンテナ間の推奨距離は 20 センチメートル以上をお勧めします。

この製品に使うことができる AC/DC アダプタは高電圧回路を含んでいます。この製品に電力を投入する時は電気ショックに対する一般的な注意が必要です。

ZigBit™ 開発キットは壊れ易いコンポーネントを含んでいます。 注意して扱ってください。

関連文書

[1] ZigBit™ OEM Modules. Product Datasheet. MeshNetics Doc. M-251~01

[2] ZigBeeNet™ IEEE802.15.4/ZigBee Software. Product Datasheet. MeshNetics Doc.

M-251~06

[3] ZigBeeNet™ Software 1.0. SerialNet. Reference Manual. AT-Command Set. MeshNetics Doc. P-ZBN-452~03

[4] ZigBeeNet™ Software 1.0. ZigBeeNet™ API. Reference Manual. MeshNetics Doc. P-ZBN-452~02

[5] ZigBeeNet Developer's Guide. MeshNetics Doc. P-ZBN-452~01

[6] ZigBeeNet™ Software 1.7. ZBNDemo Messaging Protocol Description. MeshNetics Doc. P-ZBN-452~04.

[7] ZigBit™ OEM Module. Application Note. ZigBit Power Consumption Testing. MeshNetics Doc. AN-481~01

[8] ZigBit™ OEM Module. Application Note. Serial Bootloader. MeshNetics Doc. AN-481~04

[9] ZigBit™ OEM Module. Application Note. Using ZigBit Module with Analog Sensors. MeshNetics Doc. AN-481~06

[10] ZigBeeNet™ Software 1.0. Range Measurement Tool User's Guide. MeshNetics Doc. P-ZBN-451

[11] ZigBee Document 053474r14, November 03, 2006

[12] Serial asynchronous automatic dialing and control. ITU-T Recommendation V.250, 05/99

[13] IEEE Std 802.15.4-2003 IEEE Standard for Information technology – Part 15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)

[14] TSL2550 Ambient Light Sensor With Smbus Interface. TAOS Datasheet TAOS029E. February 2006 http://www.taosinc.com/images/product/document/tsl2550-e67.pdf

[15] LM73 2.7V, SOT-23, 11-to-14 Bit Digital Temperature Sensor with 2-Wire Interface. National Semiconductor Corporation Datasheet DS201478. July 2006 http://www.national.com/pf/LM/LM73.html#Datasheet

[16] CP2102, Single-Chip USB to UART Bridge, Rev. 1.1 9/05. www.silabs.com

[17] AVR Studio. User Guide. Available in HTML Help within the product.

[18] JTAGICE mkII Quick Start Guide. http://www.atmel.com/dyn/resources/prod_documents/doc2562.pdf

[19] avr-libc Reference Manual 1.4.3

[20] WinAVR User Manual – 20070525/ By Eric B. Weddington

[21] Using the GNU Compiler Collection/ By Richard M. Stallman and the GCC Developer Community

省略形と頭文字

API Application Programming Interface、アプリケーションプログラムインタフェース

BOM	Bill of Materials
チャンネルマスク	機能するチャンネルのセットを定義する数
コーディネータ	ZigBee コーディネータは、ZigBee 網内で網を開始し、重要な網パラメータを選ぶ責任があります。 ZigBee 網は ZigBee ルータを使って拡張できます。
DIP	Dual In-line Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
終端装置	ZigBee 網では ZigBee 終端装置はセンサデータをルータに送信します。 終端装置は消費電力管理の制限を受けることがあり、ほとんどスリープモードにあります。
ESD	Electrostatic Discharge、静電放電
GUI	グラフィカル・ユーザ・インタフェース
HAL	ハードウェア・アブストラクション・層
IDE	Integrated Development Environment、統合開発環境
JTAG	IEEE1149.1 標準インターフェイスとしても知られている、組込み機器のデバッグ用デジタルインタフェース
LED	Light Emitting Diode、発光ダイオード
LQI	Link Quality Indicator、リンク品質指標
MAC	Medium Access Control 層、メディア・アクセス制御層
MCU	Microcontroller Unit、マイクロコントローラユニット。 この文書の中ではプロセッサを意味します。これは ZigBit モジュールのコアです
MIPS	Million Instructions per Second、毎秒百万命令を実行
NWK	Network layer、ネットワーク層
PAN ID	Personal Area Network Identifier.、パーソナルエリア網識別子。 ZigBee では、同じ周波数チャンネルで機能する複数の網ごとにユニークでなければならない 16 ビット番号
PCB	Printed Circuit Board、プリント基板
PHY	Physical layer、物理層
RF	Radio Frequency、無線周波数
ルータ	ZigBee 網ではルータは階層型ルーティング法を使っている網を通してデータを転送し、メッセージを制御します。 ZigBee コーディネータはルーティングにも責任があります。
RS-232	バイナリ・データ相互接続インタフェース。コンピュータのシリアルポートでは広く使われている。
RSSI	Received Signal Strength Indicator、受信信号強度
SMA	Surface Mount Assembly、ボード搭載部品
TOS	オープンソースオペレーティングシステム TinyOS

UID	Unique Identifier、ユニーク識別子
USB	Universal Serial Bus、ユニバーサルシリアルバス
VCP	Virtual Com Port
WSN	Wireless Sensor Network、無線センサ網
ZDK	ZigBit Development Kit 、ZigBit 開発キット
ZigBee	低出力のセンサアプリケーション専用無線ネットワーキング標準
802.15.4	低速の無線パーソナルエリア網に適する IEEE 802.15.4-2003 標準[12]
ZigBeeNet	ZigBit モジュール上で機能する IEEE802.15.4/ZigBee ソフトウェア。ネットワーキング機能を提供する。
SerialNet	ZigBeeNet のコンフィギュレーション。これを使って直接モジュールのプログラムを作ったりカスタマイズされた WSN アプリケーションを開発できる。

2. 開発キット概要

ZigBit™ 開発キット(ZDK)は、無線センサ網(WSN)のプロトタイプ作成と開発するために設計されたシンプルで箱から出してそのまま使えるソリューションです。この開発キットに含まれているZigBit モジュールを搭載したMeshBean2ボードと多様なツールを使って無線機能や性能を試験し、ZigBeeNet ソフトウエアでカスタマイズした無線ソリューションを開発して頂けます。

ZigBit 評価キットは以下を含みます:.

1. ZigBit モジュールと PCB アンテナを搭載した MeshBean2 ボード(1個)
2. ZigBit モジュールと外部アンテナコネクタを搭載した MeshBean2 ボード(1個)
3. ZigBit モジュールとデュアルチップアンテナを搭載した MeshBean2 ボード (1個)
4. USB2．0A/ミニ-B ケーブル(3個)
5. 片端がIDC－20ピンソケットコネクタの外部インタフェース用リボンケーブル(2 個)
6. Swivel アンテナ (1/2 wave アンテナ)
7. ソフトウェア＋ドキュメンテーション CD(1 個)

以下のZigBit開発キットの写真を参照。

図 1 開発キットセット

ZigBit 開発キットは2種類あり、付いているサポートパッケージによる異なります。(表1を参照)

- ZigBit開発キット Lite は標準の開発ツールと45日間の全般的なサポート付で提供します。

このオプションは製品のデモンストレーション、プラットフォームの評価、短期間のアプリケーションプロトタイプ作成に適しています。

- ZigBit開発キット Complete は1年間のプロフェッショナルサポート付です。このサポートはお客さまにソフトウエアの継続的更新、設計に関する真摯なサポート、RF設計の補助を含みます。これはMeshNetics社のZigBit 無線プラットフォームを使った設計、プロトタイピング、製品の市場投入に至る開発の全サイクルをなさるお客さまに適しています。このキットは新規リリースのソフトウエアへの早期のアクセスと、ZENDemo アプリケーション,API 利用例などを含む追加のサンプルアプリケーションが特徴です。

表 1 ZDKサポートパッケージ

ZDKの版	Lite	Complete
商品番号	ZDK-A1281-LTE	ZDK-A1281-CPT
サポート期間	45日間	1年間
ハード設計サポート	+	+
RF設計サポート	+	+
ソフト開発サポート	+	+
リリースソフトウエアの早期アクセス[1]	—	+
Gerber ファイルへのアクセス[2]	—	+
bootloader ソースコードへのアクセス[3]	—	+
サンプルアプリケーションの追加提供[4]	—	+
応答時間	72時間 (平日)	72時間 (平日)
サポート手段	メール	メール

[1] リリースソフトウエアの早期アクセスは技術のプレビューとデモ、暫定版のデータシート、正式発表前の製品情報を含みます。

[2] MeshBean Gerber ファイル使うとカスタム PCB のデザイン･インを大幅に促進し、ZigBit モジュールを使った顧客の独自製品や周辺装置を市場に投入するまでの時間を加速できます。USB 拡張、センサー適応他の MeshBean 開発プラットフォームが使えるからです。

[3] シリアルブートローダのソースコードへのアクセスはシリアル用のカスタムツールをビルドしたり OTA のアップグレードには必須です。

[4] 追加のサンプルアプリケーションは (1) 典型的なデータ収集シナリオの最も包括的なが特徴な ZBNDemo の組み込み部分、(2) アプリケーション“ビルディングブロック”として使える API を使った小さな例、(3) ZigBit をサードパーティーのセンサーとの統合が特徴のサンプルアプリケーションのソースを含みます。

2.1. ハードウェアの一般仕様

MeshBean2ボードはZigBitモジュールの性能評価用です。 また、ZigBeeNet ソフトウエアを組み込んだZigBitモジュールはMeshBean2ボードに無線接続機能を提供します。これにより MeshBean2ボードはZigBee網中でノードとして機能します。 MeshBeanは無線通信用にZigBitモジュールを使って開発する顧客の装置の為の基準ハードウエアプラットフォームとしても使えます。

MeshBean2 ボードは、DIP スイッチ(セクション 2.3.2 参照)をセットし、(又は) AT コマンドを送ることによって網コーディネータ、ルータ、終端装置として機能する様に設定できます。 ボードの役割はその中に組み込んだアプリケーションにより決まります。

ボードはそのZigBitモジュール内にシリアルブートローダやSerialNet アプリケーションファームウェアをプリプログラムして提供します。デモアプリケーションの全リストはセクション2.4を参照してください。Gerberファイルは Completeサポートパッケージのお客さま専用です。

MeshBeam2 の基本パラメータを表 2 に示します。

表 2 MeshBean2 ボード仕様

パラメータ	値
RF	
準拠	2.4 GHz IEEE 802.15.4～2003[13]
動作バンド	2400～2483.5 MHz
TX 出力	-17dBm から+3dBm まで
RF トランシーバ	AT86RF230
アンテナ	2.4GHz (PCB オンボードアンテナ、50 Ohm アンバランスアンテナ、デュアルチップアンテナ)
MCU	
マイクロコントローラ	ATmega1281V A,
RAM	8 kBytes
フラッシュメモリ	128 kBytes
EEPROM	4 kBytes
性能	4MHz のクロックで最高 4MIPS のスループット
電源	
電源供給	デュアル単三電池。USB、AC/DC アダプタに自動切換可能
過電圧保護	はい
逆極性保護	はい
動作電圧範囲	1.8～3.6 V.

パラメータ	値
電圧スーパーバイザ	はい
その他	
センサ	デジタル：環境光と周囲の気温
LED 表示	3 つのプログラム可能なカラーステータス LED、 外部電源供給表示用 LED
スイッチ	3 つのディップスイッチ
ボタン	2つのプログラム可能なボタン
サイズ	60 x 63 x 24 mm
動作温度範囲	-40°C から 85°C。 -20°C から +70°Cの範囲外ではクロックの安定性が少々損なわれることがあります。.

2.2 MeshBean2 ボードの機能コンポーネント

2.2.1 ZigBit モジュール

ZigBitモジュールはMeshNeticsが提供する超コンパクトで、低消費電力で、高感度の2.4GHz 802.15.4/ZigBee OEMモジュールです。 ZigBit-UモジュールはAtmelのZ-Link2.4GHzプラットフォームを使っています。これはATmega1281Vマイクロコントローラ、AT86RF230 RFトランシーバを含みます。

ZDKでは、これをMeshBean2ボードにインストールして提供します。 2タイプの ZigBitモジュールがご利用頂けます。[1] 1つはPCBや外付けアンテナのメリットが活かせるバランス型RFポートのバージョン、もう1つはボードの大きさが問題になる場合に有効なデュアルチップアンテナ付のバージョンです。

モジュールインタフェース、電源電圧、消費電力などのZigBitモジュールの詳細仕様はZigBitデータシートを参照してください。[1]

2.2.2 センサ

ボードは、TAOS社製光センサTSL2550Tとナショナルセミコンダクタ社製温度センサLM73CIMKを使います。 どちらのセンサもI²Cバスと平行に接続してあります。 センサの詳細については、メーカーの対応するウェブサイトwww.taosinc.com とwww.national.com 上の関連するデータシート [14], [15] を参照してください。

注;
ビルトイン、ボード搭載のセンサの他に開発者が選んだ外部センサも使えます。外部センサはボードの拡張スロットに接続した外部インタフェースケーブルの片端に接続できます。対応するピン割り当てについては表2を参照してください。外部センサへの接続例をアプリケーションノート[7]に

示します。

2.2.3 USB-UART ブリッジ

Silicon Labs製USBからRS-232へのブリッジコントローラCP2102[16] がボードに搭載してあります。これはシームレスなUSBインタフェースをどの様なRS-232レガシー装置にも提供します。 もし開発キット全体を導入している間にこのコントローラのドライバが既にインストールしてあるなら、(セクション3．3参照) ボード上のUSBポートは特定の数のジェネリックCOMポートとしてＰＣ上で見ることができます。

2.2.4 UID ストレージ用の Silicon シリアル

UID (Unique Identifier,ユニーク識別子) は１６進値で８ビット長です。UIDはノードのユニークなＭＡＣアドレスを設定するのに使います。また、UIDはハードウエア定義の値であり、チップにプログラムされます (Maxim/Dallas シリコンシリアル番号はDS2411R+です。)

UIDの値はユニークであり、上書きできません。ボード上のUIDを確認する最良の方法はハードウエア試験アプリケーションを実行することです。(詳細はセクション３．７を参照)

2.3 MeshBean2 ボードの設計

MeshBean2 ボードは ZigBit モジュールを含んでいます。これは ZigBee/802.15.4 トランシーバとして機能します。 これはセンサ、ボタン、DIP スイッチ、いくつかのインタフェースを含みます。

このボードは以下のインタフェースを含みます:

- USB2.0 ポート
- 光センサと温度センサ
- ソフトウェアを制御する 2 つの押しボタン
- リセットボタン
- 3 ソフトウェア制御 LED
- 対称型双極 PCB アンテナ (ＰＣＢアンテナ付は MeshBean2 だけ)
- ＳＭＡコネクタ (外部アンテナ付は MeshBean2 だけ)
- ソフトウエアのアップロードとデバッグ用のＪＴＡＧ コネクタ
- 電源コネクタ(3 V)。ＡＣ／ＤＣ変換機用 (ＺＤＫでは提供しない)
- 外部ZigBitインタフェースを含む２０ピン拡張スロット (表3を参照) 。これは以下を含む
  - シリアルポートインタフェース (RS-232)
  - ＵＳＡＲＴ
  - ESD 保護と電圧レベル変換付のバッファ I^2C インタフェース
  - ＡＤＣ／ＧＰＩＯ

- 単三電池用電池収納部
- 3 コンフィギュレーションジャンパ
- 電力消費量計測用の 3 つのクランプ

また、3.6Vをほとんどのコンポーネントに供給するために、ボードは内部の電圧調整器を含んでいます。 ZigBit の MCU を 8MHz で動かすならこれは必要です。 [5]

注:
通常、ZigBit モジュールは直接電池、USB、AC/DC アダプタ(保護電気回路経由の)によって駆動します; しかし、ジャンパ J2(表6を参照)を使うと ZigBit を 3.6V の電源供給に切り換えることができます。

統合ＰＣＢアンテナ付の MeshBean2 の外観は図2を、ボードの機能図は図3を参照してください。外部アンテナケース用の上向きのサイドバーに注目。

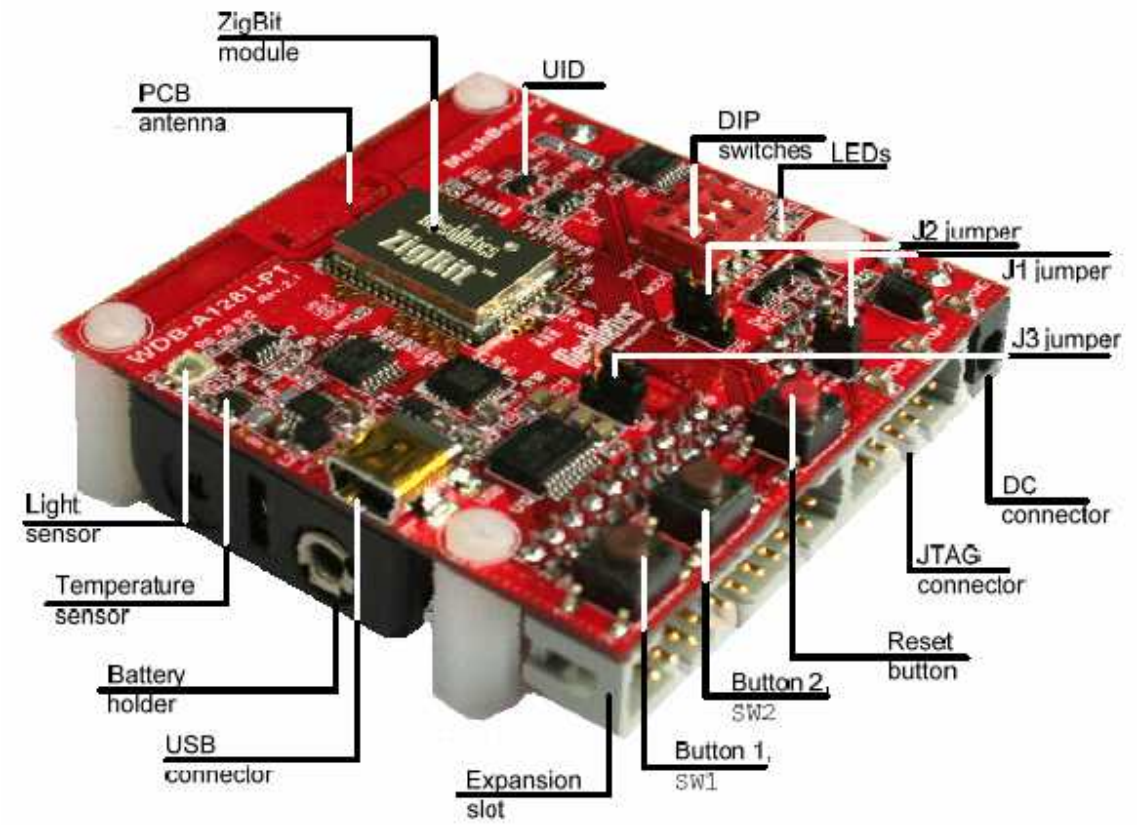

図 2 統合ＰＣＢアンテナ、UIDシリコンシリアル付 MeshBean2

[5]通常 8MHz で動く ZigBeeNet ソフトウェアを、電圧範囲を拡張し、電力消費量を減少させるには ZigBeeNet ソフトウェアに変更を加え、4MHz で動かす必要があります。

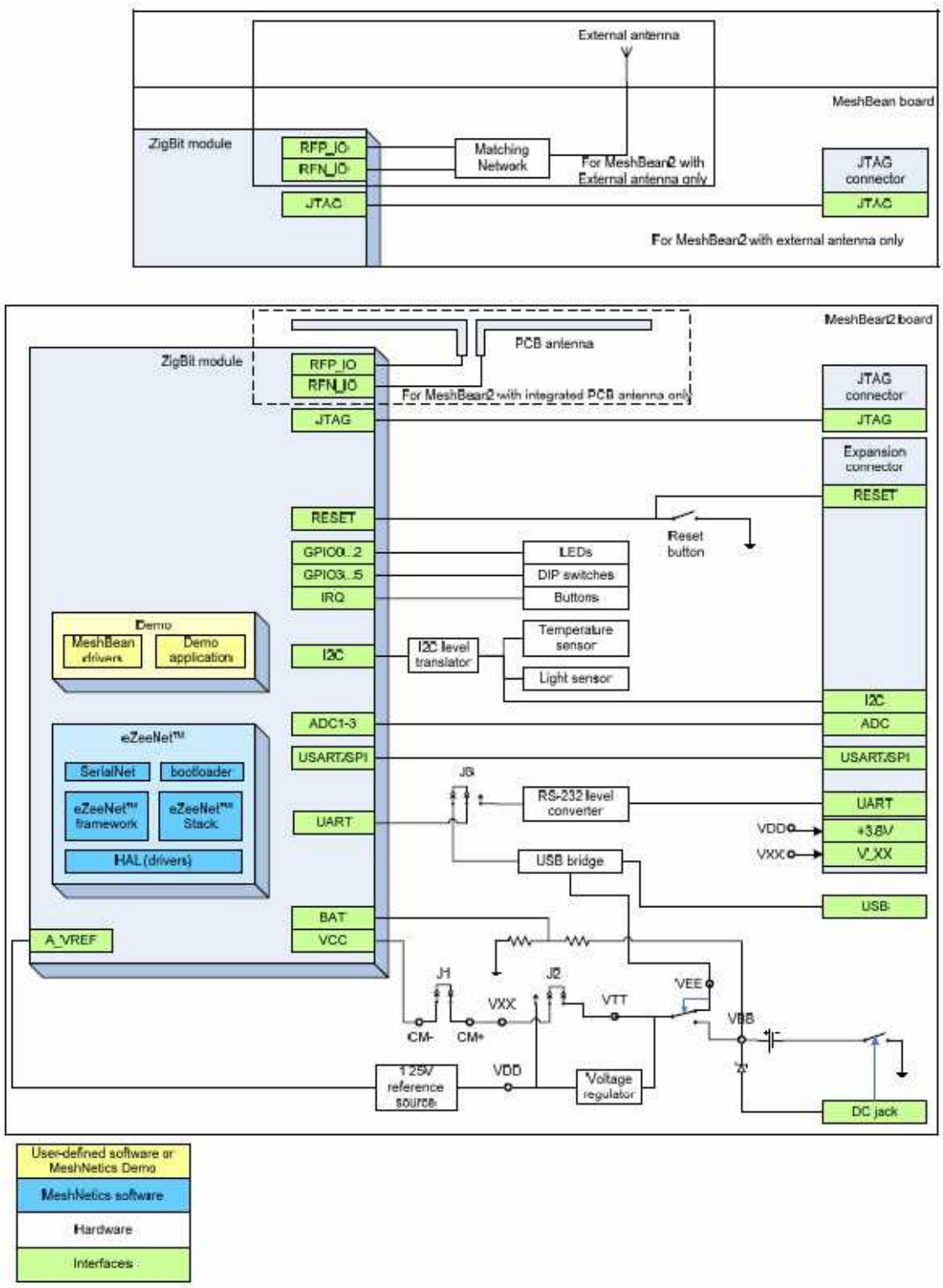

図 3 MeshBean2 機能図

2.3.3 コネクタとジャンパ

ボードコネクタのピンとジャンパの設定を表 3 から表8に示します。

重要な注:
コネクタやジャンパの操作はすべてボードに電力を供給していない時に行ってください！

表 3 拡張スロットピン配置図

ピン	名前	I／O	説明
1	UART_RTS	入力	Request to Send ピン。 RS232レベル
2	UART_TXD	入力	Transmit Data ピン(ホスト機器がこのラインにデータを送信することを意味します)。 RS232レベル
3	UART_CTS.	出力	モジュールからの Clear to Send を示す。 アクティブ時、ロウ。RS232レベル
4	UART_RXD	出力	Receive Data ピン(ホスト機器がこのラインからデータを受け取ることを意味します)。 RS232レベル
5	GND		デジタル/アナログ接地
6	GND		デジタル/アナログ接地
7	I2C_CLK	入力	I²C クロック。 これは低電圧レベル変換器経由でモジュールの I2C_CLK ピンに接続。 詳細は ZigBit データシート[1]を参照
8	I2C_ DATA	双 方向	I²C データ。これは低電圧レベル変換器経由でモジュールの I2C_DATA ピンに接続。 詳細は ZigBit データシート[1]を参照
9	+3.6V.	出力	内部の電圧調整器の出力。 通常この電圧は 3.6V
10	V_XX.	出力	ZigBit 供給電圧
11	RESET	入力	Reset ピン。 アクティブ時、ロウ。 このピンはボード上の RESET ボタンと並列に接続されます。
12	USART_TXD	出力	ZigBit モジュールの USART0 インタフェース用 Transmit Data ピン。 これはモジュールの USART0_TXD ピンに直接接続。 詳細は ZigBit データシート[1]を参照
13	USART_RXD	入力	ZigBit モジュールの USART0 インタフェース用 Receive Data ピン。 これはモジュールの USART0_RXD ピンに直接接続。 詳細は ZigBit データシート[1]を参照
14	USART_CLK	入力	ZigBit モジュールの USART0 インタフェース用 clock data ピン。 これはモジュールの USART0_EXTCLK ピンに直接接続。 詳細は ZigBit データシート[1]を参照
15	GND		デジタル/アナログ接地
16	ADC_INPUT1	入力	ADC 入力。 このピンはモジュールの ADC_INPUT_1 ピンに直接接続。 詳細は ZigBit データシート[1]を参照

ピン	名前	I／O	説明
17	ADC_INPUT2	入力	ADC 入力。 このピンはモジュールの ADC_INPUT_2 ピンに直接接続。 詳細は ZigBit データシート[1]を参照
18	ADC_INPUT3	入力	このピンはモジュールの ADC_INPUT_3 ピンに直接接続。詳細は ZigBit データシート[1]を参照
19	GND		デジタル/アナログ接地
20	GND		デジタル/アナログ接地

一般的注意事項:

ピン 12、13、14、16、17、18 はバッファされず、MCU ピンを使って直接ドライブできません。 従って、このインタフェースは、モジュールの損傷を避ける様に、低い供給電圧で使う様に事前警告してください。

ピン 7 と 8 は ESD 保護付きの電圧レベル変換器経由で接続されます。 従って、これらのピンは、余分なロジックなしで余分な I^2C センサに容易に接続して使えます。

V_XX ピン上の電圧はジャンパ J1 の状態とクランプ CM+、CM- 間の電流計接続の状態に依存しません。

表 4 JTAG コネクタピン配置図

ピン	名前	説明
1	JTAG_TCK	スキャンクロック
2	JTAG_GND.	デジタル接地
3	JTAG_TDO.	テストデータ出力
4	JTAG_VCC	コントローラ供給電圧
5	JTAG_TMS	テストモード選択
6	JTAG_RST	リセットコントローラ；アクティブ時、ロウ
7	N_Cont.	接続しない
8	N_Cont.	接続しない
9	JTAG_TDI.	テストデータ入力
10	JTAG_GND	デジタル接地

注:

JTAG ヘッダーピンアウトは ATmega JTAGICE mkII の回路内エミュレーターコネクタと互換性があります。

表 5 J1 ジャンパ設定：電流の測定

ジャンパ位置	説明
J1 は接続	このポジションは通常運用時に使用
J1 は非接続	このポジションでは ZigBit モジュールには電力は供給されませんが、ボードの残りの部分には電力を供給します。このポジションは ZigBit モジュールの電流消費を測定するために使います(セクション3．8を参照)

表 6 J2 ジャンパ設定：ZigBit 電源

Jumper.ポジション	説明
J2 は POWER と BAT のピンを接続	ZigBit は一次電源(電池、USB、AC/DC アダプタ)で稼働
J2 は POWER と DC/DC ピンをブリッジ	ZigBit は 3.6V の内部電圧調整器で稼働

表 7 J3 ジャンパ設定：RS-232/USB 選択

ジャンパ位置	説明
J3 は中央ピンと RS-232 ピンを接続	ボードはホストとの接続用にシリアルポート(拡張スロット上で利用可能)を使用
J3 は中央ピンと USB ピンを接続	ボードはホストとの接続用に USB を使用

重要な注：
ジャンパ J2 と J3 をこれら以外のポジションに設定したり、設定を省略するとハードウェアに永久的な損傷を起こすことがあります。
J1 ジャンパ無しやクランプ CM+と CM- 間をアンメータ接続してボードに電力を供給するとハードウェアに永久的な損傷を起こすことがあります。
拡張スロットを通して PC のシリアルポートに接続する時は以下の表 8 に示すピン割り当てに注意。

表 8 シリアルインタフェース ピン配置図

シグナル	拡張コネクタ	RS-232 コネクタ
RXD	4: UART_RXD.	2
TXD.	2: UART_TXD.	3
CTS	3: UART_CTS	8
RTS	1: UART_RTS	7
GND	5、 6、 15、 19、 20: GND.	5

2.3.4 ボタン、スイッチ、LED

ボードは、2 つのボタン、3 つのディップスイッチ、ハードウェアリセット信号を生成する 1 つのリセットボタン、3 つのソフトウェア定義の LED(緑色、黄色、赤)、USB からボードの電力供給を示す 1 つの青色 LED を含みます。 どのボード上のボタン、DIP スイッチ、LED の状態も ZigBit 上で動くアプリケーションで制御できます。

例えば、SerialNet(セクション5を参照)を実行中は、どの DIP スイッチの状態も無視されます。 しかし、DIP スイッチはハードウェアテストアプリケーション(セクション3．7を参照)を実行時にテストできます。

2.3.3. 外部アンテナ

ＺＤＫで提供する3枚のＭｅshBean2 ボードのうち1枚は外部アンテナ接続用のＳＭＡコネクタが付いています。開発キットに同梱の外部アンテナの仕様を以下の表9に示します。

表 9 外部アンテナ仕様

部品番号	製造業者と説明	利 得 dBi	インピーダンス Ohm	最小セパレーション cm
17010.10	WiMo, ＳＭＡコネクタ付スイベルアンテナ (1/2 wave antenna) 。周波数範囲 2.35～2.5 GHz	2.1	50	20

重要な注
SMA コネクタ付き MeshBean ボードとの組み合わせでは Non-RP SMA コネクタアンテナだけが使えます。

ボードの使用前にＳＭＡコネクタを通して外部アンテナを装着してください。アンテナのメス側にコネクタのオス側を合わせてください。

2.4 ZigBeeNet ソフトウェア

ZigBeeNet は MeshNetics が提供する、全機能を提供する、次世代の組込みソフトウェアスタックです。 このスタックは MeshNetics ZigBit モジュール上で稼働する高信頼の、スケーラブルで、セキュアな無線アプリケーション用のソフトウェア開発プラットホームを提供します。 ZigBeeNet はユーザが設計するアプリケーションによる多様な要求に対して広範なエコシステムをサポートする様に設計してあります。これによりあらゆるタイプのソフトウエアカストマイゼーションが可能になります。

ZigBeeNet はワイヤレスセンシングと制御用の ZigBee PRO と ZigBee 標準に完全に準拠しています。 ZigBeeNet は増加した API セットを提供します。これは１００%標準に準拠し、且つ開発者が快適に感じ、使い易いと思う様な拡張機能も設計して提供します。

コアスタックの最上位層、APS はアプリケーションがインタフェースする最上位レベルのネットワーキング関連 API を提供します。 ZDO は完全互換の ZigBee 装置オブジェクト API セットを提供します。これにより主網管理機能(開始、リセット、形成、参加)が可能になります。ZDO はスタックが実施する ZigBee 装置のプロファイルタイプ、装置、サービス発見コマンドを定義します。

この開発キットは ZigBeeNet API [4]を基にユーザ自身がアプリケーションを開発するのに必要なすべてを提供します。この API を使ったデモをソースコードで提供しています。このソースコードは現実のアプリケーション用に変更し、拡張して使って頂けます。ZigBit 開発キットでは、API を使うことにより対象とする装置向けに多様な WSN アプリケーションのシナリオをプログラミングできます。例えば、終端装置はスリープ状態の間にルータとコミュニケーションし、消費電力を削減するように構成を設定できます。

SerialNet は ZigBeeNet ソフトウェアのもう1つのコンフィギュレーションであり、お客さまはこれを使ってＷＳＮアプリケーションコードを開発せずにカスタマイズしたＷＳＮシナリオを導入することができます。この場合、ＷＳＮノードはＡＴコマンドを使って制御できます。(セクション 5 を参照)

ZigBeeNet ソフトウェアの構造を図 4 に示します。 その詳細はデータシート[2]にあります。

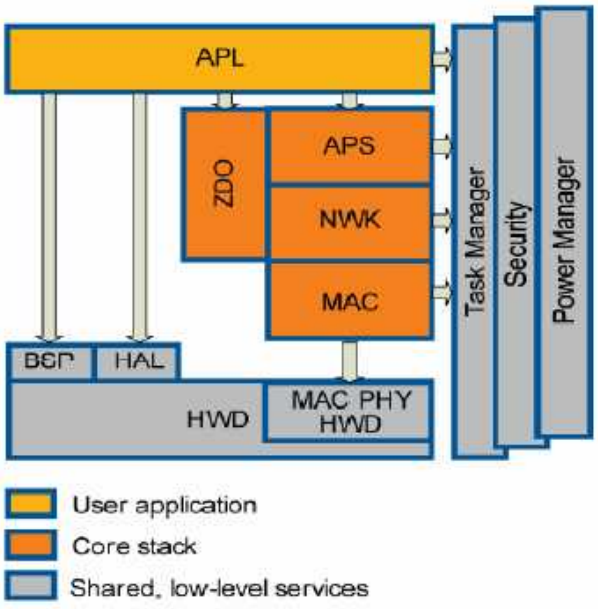

図 4 ZigBeeNet ブロック図

ZigBit 開発キットは2種類のデモアプリケーションを提供します。(付録Aを参照) 評価ツールはバイナリ形式で提供します。サンプルアプリケーションはソースコード形式で提供します。

以下の評価ツールを提供します:

- **SerialNet** アプリケーションにより AT コマン AT コマンドをローカルにインタープリットし、遠隔のノードで実行するために転送できます。
- **ハードウェア試験**(セクション3.7を参照)は MeshBean2 ボードの主要コンポーネントの正しい動きをテストする簡単なアプリケーションです。
- **電波到達距離(Range)測定ツール**は ZigBit を使った装置の無線効率を測定し、他社製品との性能比較をするアプリケーションです。このアプリの利用に必要な情報や使用法は[10]を参照してください。
- WSN モニタ付き **ZBNDemo**
- **シリアルブートローダ**はアプリケーションコードをWSNノードにUSBやシリアルポートを通して、JTAG無しでプログラムする為のソフトウエアユーティリティーです。シリアルブートローダの説明はセクション6を参照してください。

以下のサンプルアプリケーションを提供しています (括弧内は参照名):

- **ZBNDemo**(ZBNDemo)
- **低消費電力ネットワーキング**(Lowpower)
- **ピンポン**(Pibngpong)
- **ピア・ツー・ピア データ交換**(Peer2peer)
- **LED をブリンクさせるミニマム・サンプル・アプリケーション**(Blink)
- **ハードウエア試験** (HardwareTest)

ZBNDemo アプリケーションは開発キットの機能プログラムであり、セクション4で詳細を示す様に、WSN の効率を示します。この詳細をセクション4に示します。ZBNDemo のソースコードは Complete サポートパッケージのお客さま専用です。

残りのプログラムは共通の ZigBeeNet API でトリガーを掛けたサンプル例です。LED のブリンク (セクション7．3参照) はミニマムアプリケーションです。低消費電力、ピンポン、ピア・ツー・ピアアプリケーションはセクション7．4でご紹介します。

3. 使用開始

3.1 概要

このセクションではシステム要件と開発キットの導入方法についてご説明します。また、ボードの扱い方、WSN機能の試験、ローカルなハードウエア試験についてもご説明します。

3.2 システム要件

ユーザはキットを使用する前に最少の必要システム構成を知っておいてください。(表10を参照)

表 10 必要システム構成

パラメータ	値	注
PC		
CPU	インテルペンティアム III かそれ以上。800MHz	
RAM	128 M バイト	
ハードディスクの空き	50 M バイト	
JTAG エミュレータ	ケーブル付 JTAGICE mkII エミュレータ	JTAG を通して MeshBean2 ボードにファームウェアをアップロードするのに必要。(付録 B を参照)
ソフトウェア		
オペレーティングシステム	Windows2000/XP	
USB ドライバ	CP210x USB—UART ブリッジ VCP ドライバ	PC を USB ポート経由で MeshBeam2 に接続するのに必要(セクション3．4を参照)
IDE	AVR Studio 4.13+サービスパック2+ WinAVR	ファームウェアイメージを JTAG を通してアップロード(付録Bを参照)、API を使ったアプリケーション開発に必要。(セクション7を参照)
シリアルブートローダユーティリティー		ファームウェアイメージを JTAG を使わないでアップロードするのに必要(セクション6を参照)
Java マシン	Java ランタイム環境(JRE) 5.0 アップデート8またはそれ以降	WSN モニタアプリケーション実行に必要(セクション 4.5 を参照)

3.3 開発キットのインストール

開発キットをインストールするには、お客さまのPCでZDKソフトウエアとドキュメントのCDを開け、ZDKインストレーションウイザードが自動的に開始するのを確認してください。インストレーションパスを指定し、続く指示に従ってください。

その結果、指定したパスのもとにZDKファイル構成がPC上に生成されます。これを付録Aに示します。

ZDKの導入時に以下の副次的ファイルがオプションとしてインストールできます。

- Windowプラットフォーム用のUSB—UARTブリッジVCPドライバ
- Javaランタイム環境 (JRE)

ZDKを使う前にVCPドライバのインストレーションを完了するには以下を行います。

- MeshBean2ボードをUSBポートに接続します。Windowsは新しいハードウエアを発見します。ドライバのインストレーションウイザードによる支持に従います。
- ドライバがちゃんとインストールしてあり、新COMポートがデバイスリストに載っていることを確認します。図5に示すデバイスマネージャのウインドウを、スタート/設定/コントロールパネル/システム/ハードウエア/デバイスマネージャにより開きます。

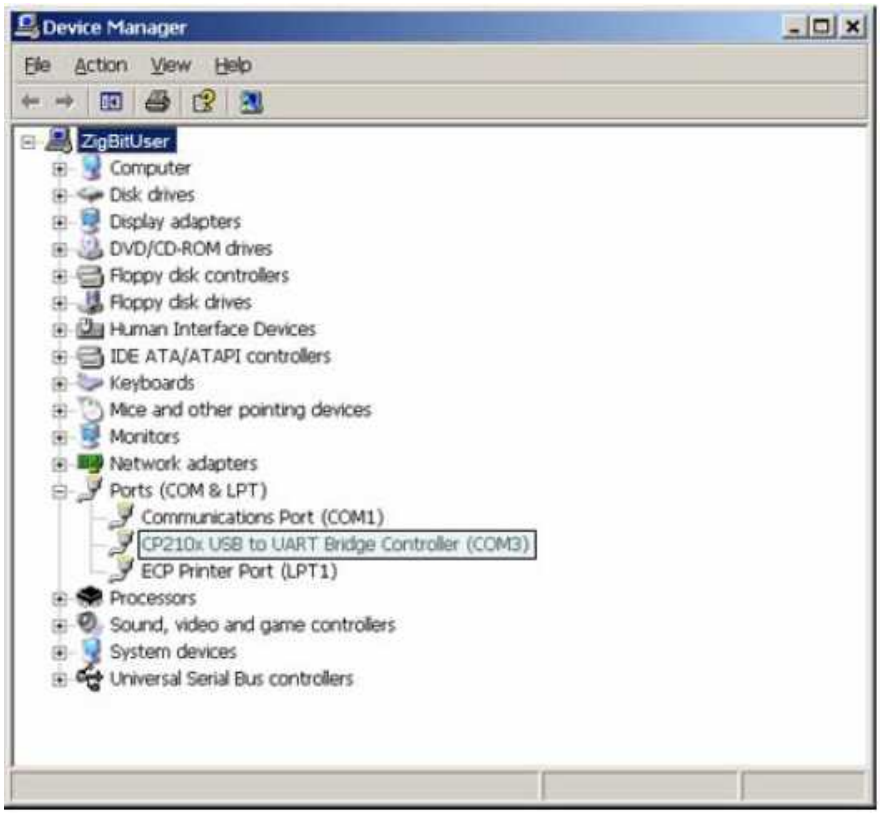

図 5 Windows デバイス・マネジャー・ウィンドウ中の COM ポートドライバ

問題が発生したらセクション3．4を参照してください。

注:
Windows プラットフォーム用のＵＳＢ―ＵＡＲＴブリッジＶＣＰドライバはメーカのサイトからアップロードできます。
http://www.silabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/mcu_vcp.htm
Javaランタイム環境（ＪＲＥ）も以下から利用できます。
http://java.sun.com/javase/downloads/index.jsp

お客さまのコンピュータには別の java インスタンスがインストールされているかもしれません。 混乱を避けるために、WSN モニタを含んでいるサブディレクトリ ./Evaluation Tools/ZBNDemo (WSN Monitor)の中の start.bat ファイルを編集してください。Java 実行ファイルへのパス全体を指定し、そのファイル名の拡張子(.exe)をはっきりと指定してください。

現在のサービスパック付きの AVR Studio[16]のバージョンは Atmel のウェブサイト(http://www.atmel.com)から無料でダウンロードできます。 ダウンロードしたインストーラプログラムを実行し、指定された手順に従ってください。

開発ツールである WinAVR スイートは http://sourceforge.net/projects/winavr からダウンロードできます。 WinAVR をインストールするにはセットアップ指示に従ってください。

3.4 ボードを PC に接続

ボードは開発キットに同梱の USB 2.0 A/mini-B ケーブルを使って USB ポート経由で PC と接続できます。 ＵＳＢは一般的な接続手段です。ＵＳＢを使って複数のボードを１台のＰＣに快適にリンクする可能性を提供します。また、ＵＳＢを使ってボードに給電すると電池は不要です。

無線アプリケーションは通常ＣＯＭポートを通してホストとの接続を行うので、ボード上のＵＳＢ―ＵＡＲＴブリッジコントローラをＰＣにリンクするにはＵＳＢ―ＵＡＲＴブリッジＶＣＰドライバのインストレーションが必要です（セクション３．３、セクション２．２．３を参照）。この結果、ジェネリックＣＯＭポートはＵＳＢ経由でボードにアクセスするのに使えます。

重要な注:
USB 接続を使う時、ボードを再接続すると Windows システムは COM ポート番号を変更することがあります。混乱を避ける為に、Windows コントロールパネルを使って実際に使っているポート番号

をチェックしてください。
ある状況下では、ボードは既にインストールしてある他の USB デバイスと矛盾することがあります。 そのような場合、Windows デバイスマネージャはプラグ･アンド･プレイ手順の間に起こった問題を報告するか、USB-UART ブリッジコントローラを検出しないかもしれません。 これに対する解決法は、USB コントローラメーカから入手可能な特別なユーティリティを使ってボードに対して USB ID を変更することです。 詳細はセクション8を参照。

代替案として、シリアルケーブルを使ってボードをシリアルポート経由でＰＣに接続できます。シリアルケーブルはＺＤＫには同梱していません。シリアルケーブルのピン割当てを表8に示します。

重要な注:
ＵＳＢとシリアルポート（ＲＳ232）のどちらのポートもボード上の物理的に同じポートを共有するので同時には使えません。

この接続モードはジャンパＪ3の設定を使って制御します。（表7を参照）

3.5 ボードへの電力供給

ボードへの電源供給は2個の単三電池から、または USB ホストポート経由で（データ伝送用に接続した場合）、または AC/DC アダプタ経由で行います。 公称電圧は 3V です。 AC/DC アダプタを使うと単 3 電池は自動的に非接続になります。 USB ポートを使うと AC/DC アダプタソースは非接続になります。

センサパラメータを正確に測定するには電池からの電力供給を勧めます。 USB 電力供給は十分に安定していません。送信出力レベルの送信や RF パラメータに影響することがあります。

重要な注:
ボードのプログラムを作る前に電源電圧をシリアルブートローダや JTAG を使ってチェックする様強く勧めます。 プログラミングプロセスの間に電源断が発生すると ZigBit は機能しなくなるか永久的な損傷が発生することがあります。

放電した電池(電圧が指定した限界の下にある時)を使うとフラッシュメモリや EEPROM の損傷を起こすことがあります。 それが起こったら、シリアルブートローダを使ったプログラミングは失敗するかもしれません。この場合の唯一の手段は JTAG エミュレータを使うことです。(付録Bを参照)

ニッケルカドミウム充電式電池を使ってもかまいませんが、事前の注意がいくつかあります。 この電池の1セルの電圧は約 1.2V です。 これを 1 ペアで使うと 2.4V になり、従って、動作電圧範囲

(セクション2.1を参照)にまだ適合しますが、最もポピュラーな1対のアルカリ電池が与える3Vより低いです。 それゆえ、ニッケルカドミウム充電式電池はどんなアプリケーション用にも使えるアルカリ電池の適切な代替案ではないかもしれません。

3.6 SerialNetを使ったWSN機能の試験

WSNに同梱のボードは全てSerialNet ファームウエアをプリプログラムして提供します。(セクション3を参照)

ボードをPCに接続してください。(セクション3．4を参照)

Windows 2000/XP の一部である標準の Hyper ターミナルユーティリティを実行し、以下の COM ポートパラメータを設定してください:

スタート/プログラム/アクセサリ/通信/ハイパーターミナル

システムが提供する COM ポートの論理値を選択してください (セクション3．4を参照) COMポートパラメータは表11に示す値に設定します。

表 11 ハードウェア試験用 COM ポート設定

オプション	値
データ転送速度	38400 bps
データビット	8
パリティ	なし
ストップビット	1
フロー制御	なし。ボード間のデータ転送を計画していないならハードウエアフロー制御オプションを選んでください

“AT”コマンドを入力し、Enter キーを押してください。

ボードはハイパーターミナルに“OK”を応答します。

ユーザはいろいろのネットワークシナリオをATコマンドを送信することにより、ボードを再プログラミングしなくても実行できます。ATコマンドの全般的な説明は[3]にあります。

WSNを構築し、WSNノード間でデータを送信し、ノードのインタフェースにアクセスするという単純なドキュメントを[3]の Examples セクションに示してあります。

3.7 ボードの制御とセンサの試験

ボードの制御とセンサの試験をするにはハードウエア試験アプリケーションを使います。

ボードをＰＣに接続してください。

ハードウエア試験イメージをボードにアップロードしてください。ハードウエア試験イメージは附録Ａにリストしてあります。

上に説明した方法でハイパーターミナルユーティリティーを実行してください（表１１を参照）。

ハードウエア試験を実行中は、ボードのすべての LED はブリンクしています。ボタン、ディップスイッチ、センサの状態、ユニーク ID 番号を含むレポートが 5 秒 (図 6 を参照) 毎に生成されます。ハードウェアを試験するには、ボードを使って簡単な操作を実行します：ボタンを押し、DIP スイッチを動かし、手のひらで光センサを覆い、指を使って温度センサに触れるなどをします。 パラメータの変化が、Hyper ターミナルウィンドウ上で報告されるのを見てください。(図6を参照)

注:
操作の間に、ボードを USB と再接続したりボードの電源を切るなら、オペレーティングシステムはこの特定の USB 接続のポート番号を勝手に変更することがあります。 Hyper ターミナルはそのような変化を認識しません。 これが起こったなら、Hyper ターミナルを適切なポートに再接続する必要があります。 ファイル/新しい接続（N） メニュー項目を選び、最初に行ったのと同じ手続を繰り返してください。

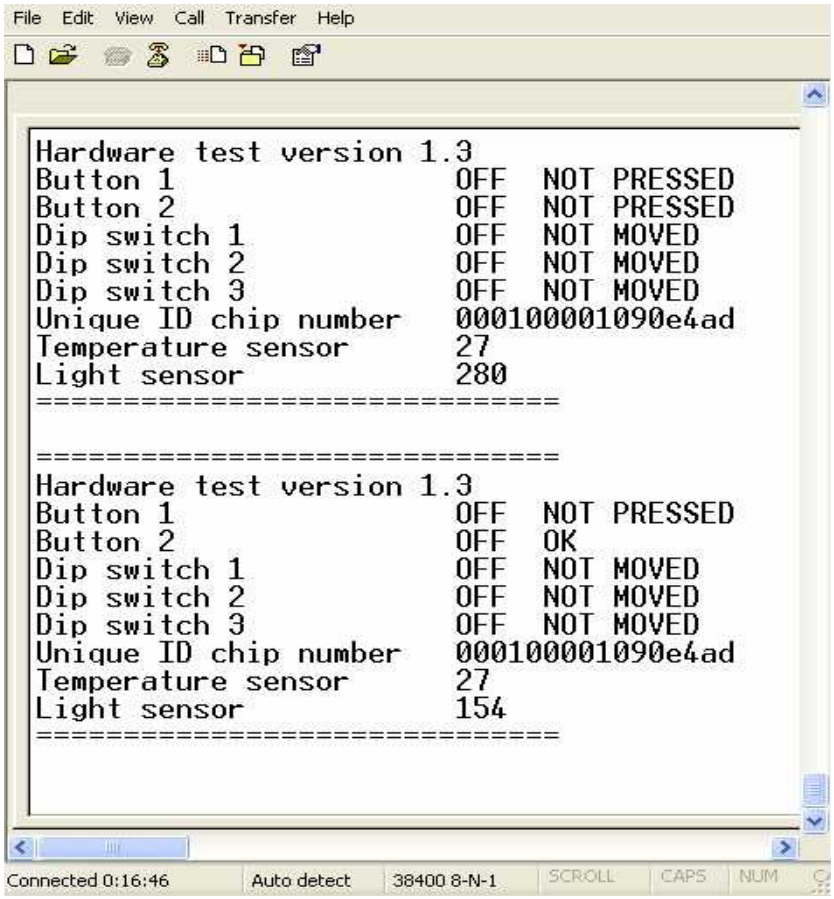

図 6 ハイパーターミナル ハードウェア試験リポート

3.8 消費電力の計測

ボードを使って ZigBit モジュールの電力消費量が測定できます。 測定するには、電流計をクランプ CM+と CM- に接続しジャンパ J1 を外してください。 ボードが電池だけで動いていることを確かめてください。 しかし、この様な方法では ZigBit と接続されたインタフェースや周辺装置が電力を消費するので、計測値は厳密には正しくないでしょう。 正確に測定するには、RF ポートを除くすべてのインタフェースを ZigBit モジュールから切り離すべきです。 詳細はアプリケーションノート[7]を参照してください。

3.9 アンテナに関する注意

各タイプのアンテナ：PCB アンテナとデュアルチップアンテナと外部アンテナはアンテナと隣接したすべてのコンポーネント（ZigBit モジュールシールド、電池収納部、プラスチックの脚を含む）を考慮に入れ、技術条件に適合する様に調整してあります。 アンテナの近くに置いたオブジェクトはどの様なものでもその性能に影響します。 モジュールをエンクロージャに入れないでください。 ボードを金属表面上に搭載しないでください。 ボードを固定するためにその脚に 5mm より長い金属ねじを使わないてください。 これらの要因は性能に影響します。

プラスチックの脚の取り付けは底の方(電池収納部の隣)からだけ行ってください。 足を固定するためプラスチックのねじを使ってください。 異なったプラスチック素材で作った脚を使わないでください。 プラスチックの脚を除くとアンテナ性能に大きく影響することがあります。

アンテナの放射パターンは広範囲です。 以下の要件を検討してください。見通しの良い屋外では、電磁放射線は双極子に対してノーマルな方向で水平面に、より強く現れます。 しかし、このパターンは数センチメートルの距離ではもっと複雑です。 近似のフィールドパターンは ZigBit データシート[1]で提供しています。

外部アンテナはメカニカルな損傷を生じない様に扱ってください。

4. ZBNDemo アプリケーション

4.1 概要

ZigBitプラットフォームを使ったネットワーキングの効率はZigBeeNetソフトウェアAPIを基にしたZBNDemoアプリケーションで示すことができます。このアプリケーションは組込みソフトウェアとコーディネータ、ルータ、終端装置の機能サポートとGUI部分（PC上で実行するWSNモニタ）から構成します。

ZBNDemoアプリケーションは組込みソフトなので、MeshBean2ボードは無線網を構成するノード群として構成されます。ボードのＬＥＤはボードの現在の状態と動きを示します。デューティーサイクルでは終端装置やルータはボード上のセンサの計測値を受け、それをパケットにしてコーディネータに送ります。このデータはWSNモニタパネル上に温度、照度、電池電圧の計測値として表示します。

終端装置はほとんどの間スリープ状態にあり、その間の消費電力はとても低く、１０秒間隔で起きて何らかの作業を行います。スリープ状態の間SW1ボタンを押して終端装置を強制的に起こすことができます。ルータは1秒間隔でデータを送ります。コーディネータはUARTを使って受信したパケットを自身のセンサデータと共にGUIアプリケーション（ＷＳＮモニタ）に送ります。

WSNモニタは網トポロジをツリー形式でリアルタイムで表示します。また、ノードアドレス、ノードセンサ情報、ノードのリンク品質データの様なノードパラメータも表示します。

リンクの現状を示すRSSIはｄBmで計測します。この精度は3dBmです。LQはリンク品質の計測値であり、0から２５５の範囲で定義します。この値が大きいとより良いリンク品質を意味し、0に近いとノード間の接続品質が劣悪であることを示します。

WSNモニタ制御を使って網チャネルマスク、ノードのタイムアウトを変更し、ノードを遠隔からリセットできます。

ZBNDemo に関するボードの利用方法はセクション４．３に記述します。GUI はセクション４．５に記述します。操作手順はセクション４．６に示します。

このアプリケーションはソースコードで開発キットに同梱しています。（付録 A を参照）これは ZigBeeNet API の上に導入します。また、セクション7に示す様に再構築できます。

ZBNDemo を使うと多くのルータや終端装置を網パラメータの制限の範囲で表示できます（セクショ

ン7．4を参照)。

4.2 ボードのプログラミング

ZBNDemo イメージをボードにアップロードしてください。ZBNDemo イメージファイルは附録Aにリストしてあります。

ZBNDemo イメージファイルは 2 つの方法でアップロードできます：シリアル ブートローダ ユーティリティ(セクション 4.2.1 を参照)を使うか、JTAG エミュレータを使って、AVR Studio の下で行います。例えば、Atmel[17][6]製の JTAGICE mkII を使います。(セクション 4.2.2 を参照)

重要な注：
ノードをプログラミングするための選択は慎重に行ってください。各 MeshBean2 ボードは ZigBit の MCU 内にブートストラップをアップロードして出荷します。これはシリアルブートローダを実行するのに必要です。JTAG を使っているなら、ボード上にブートストラップをリロードしない限りシリアルブートローダを使うことはできません。

各ノードはWSN網内で相互接続するにはユニークなMACアドレスで識別しなければなりません。ハードウェアでMACアドレスを定義していないなら、そのノード アドレッシングIDをプログラムしなければなりません。MeshBean2ボードにMACアドレスをプログラミングする方法は4つあります。

1. MACアドレスはシリアルブートローダを使ってボード上にアップロードできます。シリアルブートローダはコマンド行で指定したキーと共に実行します。([7]を参照)
2. これはアプリケーションのコンパイルを定義する時にMakefile内で指定できます。(詳細はセクション7.4を参照) この成果物のイメージファイルはユニークなMACアドレスを含み、JTAGまたはシリアル ブートローダを使ってボードにアップロードできます。
3. そうでなければ、MACアドレスはSerialNetのATコマンドを使ってプログラムできます。([3]を参照)
4. UID内に保存した値をＭＡＣアドレスとして使います。

ＭＡＣアドレスは網内のノードの特定に使います。ＭＡＣアドレスの既定値はゼロです。モジュールはそのＭＡＣアドレスが0xFFFFFFFFFFFFFFFFと等価でない非ゼロの値を設定されない限り網には参加しません。

ZigBeeNetソフトウエアは次の方法でＭＡＣアドレスを発見します。スタートアップ時には、

[6] もう1つの JTAG プログラマも使えますが、これは Atmel 1281 MCU と互換性が必要です。

ZigBeeNetソフトウエアはＥＥＰROMからＭＡＣアドレスをロードしようとします。もしＥＥＰＲＯＭ内の値が0もしくは0xFFFFFFFFFFFFFFFFなら、ZigBeeNetはＵＩＤからＭＡＣアドレスをロードしようとします。

4.2.1 シリアルブートローダの使用

シリアルブートローダを使ってボードをプログラムするには以下のステップを実行します：

1. ジャンパ J3 の設定に従って USB または RS-232 ポート経由で、ボードを PC と接続します。(表7を参照)
2. シリアルブートローダを実行してください。このコマンドラインではイメージファイル ZBNDemo.srec (付録Aを参照)、COM ポート、オプションのキーを指定します。(詳細は[8]を参照)
3. ボード上のリセットボタンを押下します。
4. ボード上のリセットボタンをリリースします。 シリアルブートローダは30秒以内にボタンがリリースされるのを待っています。 この間にこのリリースが無ければブート手順は停止します。

注：
ノードが終端装置として設定してあり、それが現在実行中のアプリケーションによって制御されているなら、このノードは再プログラミングする前に電源を切るべきです。

ボード上の J3 の設定が現実の RS-232/USB 接続に対応していることを確かめてください。

シリアルブートローダは操作の進展状況を示します。 アップロードが問題なく完了したらボードは自動的に再スタートします。 アップロードが失敗したらシリアルブートローダはその理由を表示します。 ブートプロセスはボードと PC 間の通信エラーのため失敗することがまれにあります。 これが起こったら、ブートを繰り返すか、USB の代わりに正常なシリアルポートを使ってみてください。ブートが失敗したら、ボードに直前に書かれたプログラムが破壊されることがありますが、ボードは再プログラムできます。

4.2.2 JTAG の使用

JTAG エミュレータを MeshBean2のオンボード JTAG コネクタとリンクしてください。(図 2 を参照)
AVR Studio の元で、[17]から[18]の指示に従ってアップロード手順を始めてください。
ZBNDemo.hex としてアップロードするイメージファイルを選択してください。(付録Aを参照)

JTAGを通してイメージをアップロードする前に、Fuses Tab 中の以下のオプションがONになっていることをチェックしてください。：

Brown-out detection disabled; [BODLEVEL=111]

JTAG Interface Enabled; [JTAGEN=0]
Serial program downloading (SPI) enabled; [SPIEN=0]
Boot Flash section size=1024 words Boot start address=$FC00;[BOOTSZ=10]
Divide clock by 8 internally; [CKDIV8=0]
Int. RC Osc.; Start-up time: 6 CK + 65 ms; [CKSEL=0010 SUT=01]

残りのオプションを解除してください。 以下の16進値の文字列がFuses Tab の下面に現われているのを確かめてください:

```
0xFF, 0x9D, 0x62.
```

さらに、ノードをシリアルブートローダでプログラムするなら、以下のオプションが ON になっていることをチェックしてください。:

```
Boot Reset vector Enabled (default address=$0000);
[BOOTRST=0]
```

以下の16進値の文字列がヒューズタブの下面に現われているのを確かめてください:

```
0xFF, 0x9C, 0x62.
```

各ボード(MCU)は、既定値として、上に示す方法でプリプログラムできます。

その他に、JTAG はシリアルブートローダコマンドに応答する装置の能力を修復する為にも使えます。シリアルブートローダコードは JTAG を使い、開発キットCDにある bootloader.hex イメージを選び、それを装置に移すことにより再プログラムできます。

4.3 ボードの使用

通常、ノードの開始時には現在のチャネルマスクをEEPROMから読みます。もしチャネルマスクがシリアルブートローダを使ってEEPROMに事前にアップロードしてあるなら、ZBNDemoを始める前には以下に記述する特別な操作は必要ありません。それにも拘わらず、フラッシュ(イメージファイル)からEEPROMにチャネルマスクをロードする必要があるなら、ノードを初めてイニシャライズする時は以下の様に行わなければなりません。

先ずボード上の SW1 ボタンを押下します。(図 2 を参照)このボタンを最低1秒押下したままボードの電源を入れます。この LED2 は 3 回フラッシュを始めます。次に全 LED がフラッシュを始め、各ノードの役割を表示します。ルータは1回、終端装置は2回、コーディネータは3回フラッシュします。

LED1, LED2, LED3は2秒間ブリンクし、EEPROM内のチャンネルマスクが受け入れられたことを示します。

注:
上の操作が完了すると EEPROM に事前にロードしてあったチャネルマスクは失われます。

ZBNDemo を始めるには以下を行います。

1. 1つのノードをコーディネータとして、他をルータや終端装置として構成します。
   (表12を参照) どのボードもコーディネータ、ルータ、終端装置として構成できます。
2. コーディネータノードをコーディネータのボードのUSBポートを使ってPCと接続します。
3. コーディネータノードの電源を入れます
4. WSNモニタを実行します(セクション 4.6.1を参照)
5. 残りのボードの電源を入れ、リセットします

注:
ZBNDemo の実行中には、チャネルマスクはWSNモニタからコマンドを送ることにより、後でいつでも変更できます。(セクション 4.6.4 を参照) WSNモニタから送信し、ノードが受信したチャネルマスクは、電源のオン、オフにかかわらずEEPROM 内にずっと保存されます。EEPROM 内の既定値のチャネルマスクを回復するには、このセクションの上に示す再初期化手順を繰り返すかシリアルブートローダを使ってください。

表 12 ZBNDemo で使用する DIP スイッチ

DIP スイッチ			説明
1	2	3	
ON	OFF	OFF	ボードはコーディネータとして設定します
OFF	ON	OFF	ボードはルータとして設定します
OFF	OFF	ON	ボードは終端装置として設定します

コーディネータは無線網を自動的に構成します。開始時にはどのノードも網に対して自分の役割を知らせます。この時、LED1, LED2, LED3がルータでは1回フラッシュし、終端装置では2回フラッシュし、コーディネータでは3回フラッシュします。

網に参加した後ノードはコーディネータにデータを送信し始め、この状態はLEDで示します。
WSNの活動状況は2つの方法で見ることができます。

- ボード上のLED表示による (表13のLED表示の解説を参照)
- PCにインストールしたWSNモニタの網情報による

表 13 ZBNDemo で示す LED 表示の意味

ノード状態	LED 状態		
	LED1(赤)	LED2(黄)	LED3(緑)
スタンバイ	同期してブリンク		
網を検索中	ブリンク	オフ	オフ

ノード状態	LED 状態		
	LED1(赤)	LED2(黄)	LED3(緑)
網に参加	オン		
＋メッセージの受信中		ブリンク	
＋メッセージの送信中 (コーディネータだけ)			ブリンク
スリープ状態 (終端装置だけ)	オフ	オフ	オフ

コーディネータは電源を入れると、子ノードが無くてもアクティブ状態に変わります。これは正常な動作であり、コーディネータはコーディネータのPAN IDを持つ網に子ノードが参加するのを受け入れる準備ができていることを示します。

既定値では、コーディネータは事前に定義したPAN IDとして D170 を使います。ルータはこれを認識します。

注:
コーディネータが存在しないか、その電源がオンになっていないならば、ルータは網検索モードのままです。 このモードでは、ルータは選ばれた周波数のチャンネルを走査し続け、選んだ PAN ID の網を捜します。

選んだ周波数チャンネルの無線チャンネルがビジーなら、コーディネータノードは網検索モードのままなことがまれにあります。 これが起こったら、WSN モニタを使ってチャンネルマスクを変更して使用チャンネルを変更するべきです。

4.4 センサデータと電池レベルの表示

各ボードは温度、明度、電池のレベルを測定します。 各ボードはデータ値をコーディネータに送り、コーディネータはデータを PC に送ります。 WSN モニタアプリケーションはその値をノードアイコンの隣に表示し、それをグラフでも表示します。(セクション 4.5 を参照)

温度センサは周囲の温度を測定します。 センサの実際の精度は 1 度以上ですが、センサデータは 1 度単位で、WSN モニタチャート中で表示します。光センサは周囲の照明を Lux.単位で測定します。 電池電圧インジケータの通常の精度は約 0.1V です。これはほとんどのアプリケーションと自動監視用には十分です。

注:
ボードに USB ポート経由で電力供給している場合、電池レベルの表示は不適切です。 一般に、それは電源保護電気回路により 0.6V を表示します。 しかし、電池を電池収納部に収納してある

ならば、ボードを USB に接続している時、電池レベルの表示は正確になります。

ボードに USB ポート経由で電源供給している場合、 温度センサの隣にある電圧調整器は発熱するのでセンサの測定値をゆがめることがあります。 より正確な測定のために電池駆動のボードを使ってください。

4.5 WSN モニタ

WSN モニタは PC で動く ZBNDemo 用の GUI アプリケーションで、WSN 網についての WSN トポロジーや他の情報を表示します。 図 7 の WSN モニタスクリーンを参照してください。これは網トポロジー表示枠、センサデータグラフ枠、ノードデータ表枠、ツールバーを表示しています。

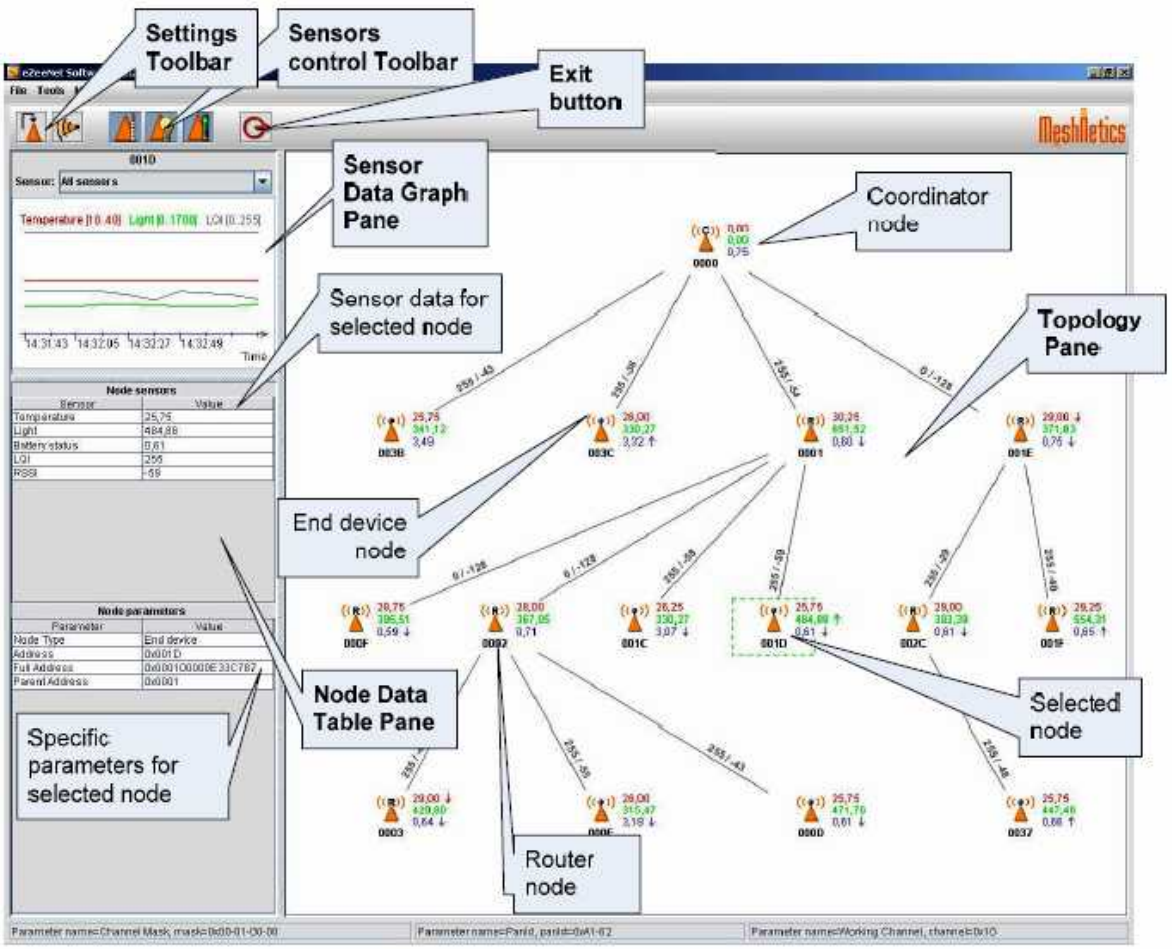

図 7 WSN モニタ GUI

網トポロジー枠はツリー状の網をリアルタイムで表示します。 これはノードが網に参加したり、データを送ったり、網から離脱する様な、網の構成変更や拡張作業に役立ちます。網トポロジー枠の表示内容は、コーディネータを通してノードを網上で発見したりノードが網に参加すると自動的に更新します。 各ノードはその名前で示し、ノードデータ情報付きでアイコンで表示します。 ノー

ドは RSSI と LQI の値により親／子リンク経由で一旦相互接続されると、ツリー構造の一部として構成を表示します。

ノードデータ枠は、ノードセンサデータを表示します。(セクション 4.4 を参照) これはグラフと表で表示します。 表では各ノードの他のパラメータも表示します。 ノードデータ枠にはプルダウンのセンサ選択コンボボックスがあり、センサタイプを切り替えできます。

ノードのタイトルは NodeNames.txt ファイルで定義します。 このファイルは、既定値では以下のサブディレクトリに置かれます:
"./Evaluation Tools/ZBNDemo (WSN Monitor)/resources/ configuration/"

注: ファイルへの全パス表示は開発キットのインストール時に指定したルートディレクトリに依存します。 (セクション 3.3 を参照)

NodeNames.txt は"-- NodeNames --"という文字列をヘッダテキストとして含み、その後に 64 ビットの MAC アドレスとノード名を各行に含む多くの文字列が続きます。 例えば図 8 を参照。

NodeNames.txt ファイルが見つからないか、そのフォーマットが認識されないならば、WSN モニタは既定値の名前を明示します。

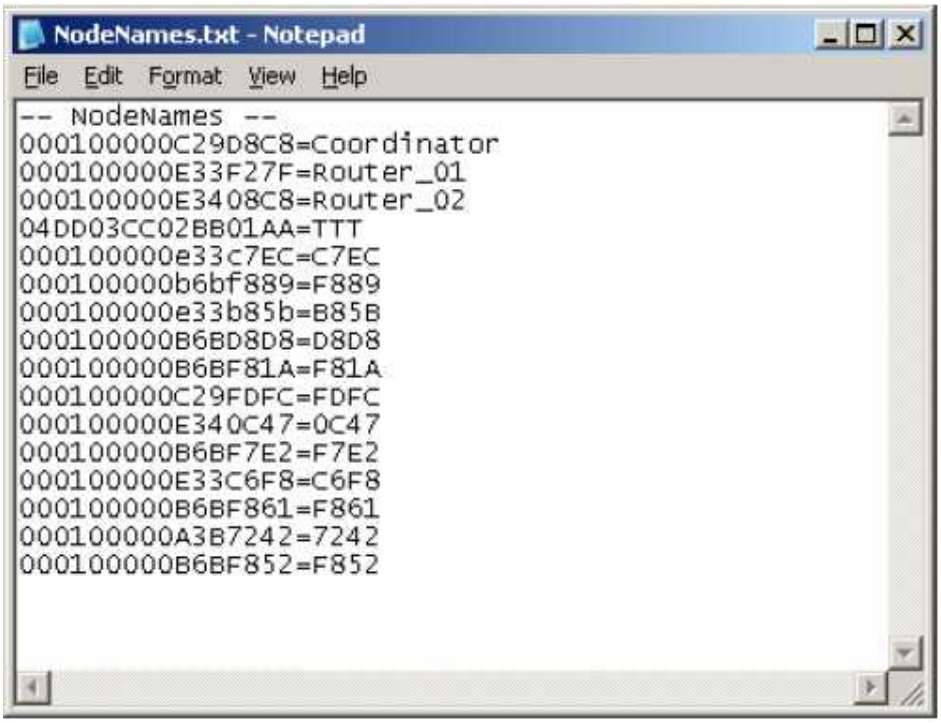

図 8 ノードタイトルを含むファイルの例

4.6 ZBNDemo の操作

4.6.1 MeshBean2 ノード上で ZBNDemo を開始

先ず、J3 ジャンパの設定に従い、コーディネータノードを USB またはシリアルポートに接続します。(表 7 を参照) 次に、WSN モニタアプリケーションを PC 上で実行します。 立ち上がり時には、WSN モニタは既定値の COM ポートを使ってコーディネータとの接続を試みます。 WSN モニタスクリーンがポップアップしますが、コーディネータノードのアイコンは網トポロジー枠には未だ表示されません。(図 7 を参照) Tools/Settings メニュー経由で適切な COM ポートを設定する必要があります。(図 9 を参照) アイコンが表れないならプログラムを再スタートしてください。

4.6.2 ノードタイムアウトの設定

Tools/Settings メニューは多くのパラメータを含んでいます。 タイムアウトはリンクドロップ、電源断、リセットなどの理由で網から離れようとするコーディネータ、ルータ、終端装置の表示の設定に使います。 ノードのタイムアウトは WSN モニタがノードからデータパケットを受け取ろうとして待つ待ち時間を意味します。待ち時間の間にデータを受取るとWSNモニタは表示を更新します。トポロジー表示の変更をスムーズにするには、待ち時間をコーディネータとルータでは3秒、終端装置では30秒に設定することをお勧めします。 この様な設定は3回のパケット送信間隔をカバーします。

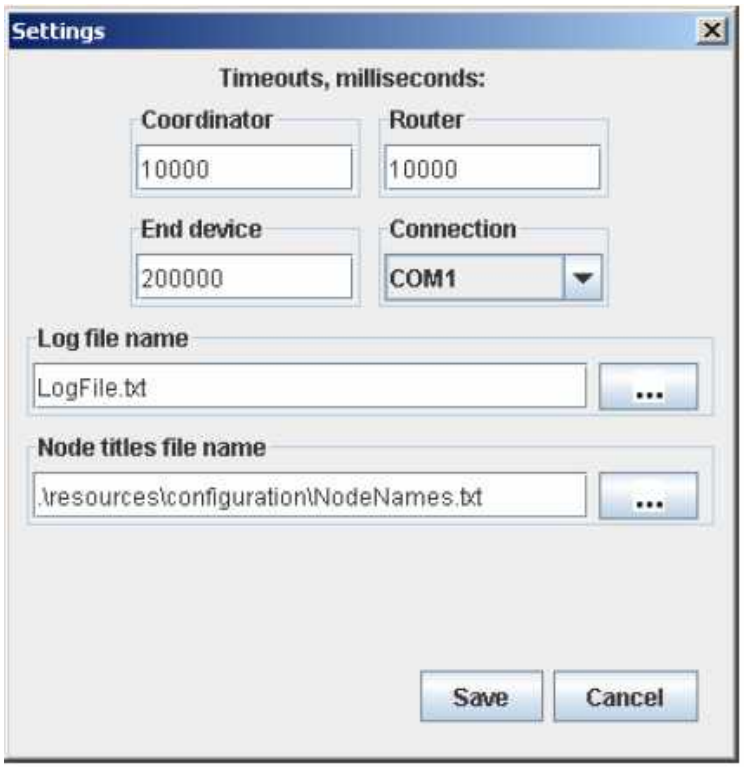

図 9 WSN モニタツール/設定のメニュー

4.6.3 ノードのリセット

ノードは WSN モニタの Tools/Send Command メニューを使ってリセットできます。(図 10 を参照)。ノードはその MAC アドレスで識別するか、トポロジー枠中に表示しているノードのリストから(combo-box を使って)選べます。

図 10 ノードをリセットします

4.6.4 周波数チャンネルの変更

網の運用は 2.4GHz 帯域中の 16 の上位チャンネル(11(0x0B)から 26(0x1A)まで)を使って行います。チャンネルマスクをセットするため、Tools/Send Command ダイアログボックスを使います。現在のチャンネルマスクは既定値で、そこに表示します。(図 10を参照) マスクを16進で直接入力するか、「…」ボタンをクリックしてください。

注:
チャンネルマスクは、利用可能なチャンネルを定義するビットフィールドです。 チャンネルマスクの最上位の5ビット(b_{27},…,b_{31}) は 0 に設定されるべきです。 残りの下位の 27 ビット(b_0, b_1,〜 b_{26}) は各 27 チャンネルの利用可能性状態(1=利用可能、 0=利用不能)を示します。

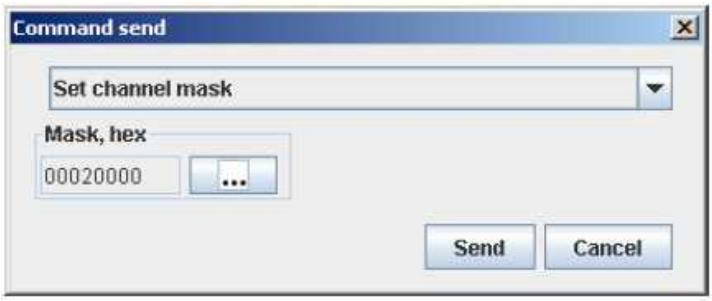

図 11 チャンネルマスクダイアログボックスの設定

さもなければ、"…"ボタンをクリックすることによって別のダイアログボックスを開けます。 チェックボックスを使ってチャンネルを選び、以下のどれかを ON に設定してください(図 12 を参照)。

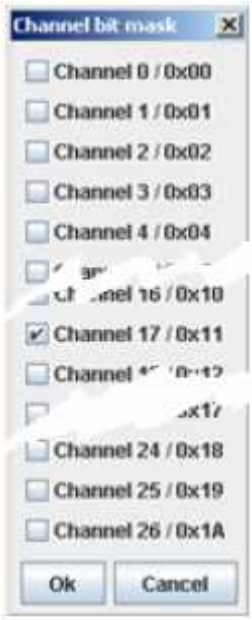

図 12 チェックボックスを使ってチャンネルマスクを設定

コーディネータはチャンネルマスクを変更する時、コマンドを全てのノードに送り、古いチャンネルマスクを使ったパケットを最後に受信した後、更に 1 分間待ちます。 次に、コーディネータは新チャンネル上で網を構成します。

ルータまたは終端装置がチャンネルマスクコマンドを受け入れる時、ノードはパケット送信を 1 分間中止し、LED1、LED2、LED3 のブリンク（点滅）を始めます。 次に、ノードは網をそのままにして新しいチャンネルマスクを使った網に参加する手順を開始します。

ルータが網に再参加する時、網の表示 LED の LED3 はブリンク（点滅）します。 参加が完了すると LED3 はオンになります。

終端装置が網に再参加する時、網の表示 LED の LED3 はブリンク（点滅）します。終端装置の参加が完了すると LED3 はオンになります。 LED1 は短時間フラッシュし、パケット送信を示します。 LED1 は短時間フラッシュし、確認情報(acknowledgement)を受信したことを示します。 次に、終端装置がスリープモードになるとすべての LED は消灯します。

チャンネルマスクを変更する時、トポロジースクリーンは現実のトポロジーツリーを表示しないことがあります。 チャンネルマスクを交換した後、網トポロジーツリーの表示は更新します。

4.6.5 センサデータの可視化

トポロジー系統図を持ち、GUI 制御を使うことにより、どのノードも自由に選んでその活動を監視し、

また 3 つの異なったフォームでノードデータを見られます:

- テキスト表
- チャート
- トポロジー枠上のセンサデータ値。 これらの値の隣に、その相対的な増加や減少を矢印で示します。

トポロジー枠では各々選んだノードの温度と光センサの測定値や電池レベルを上書き表示します(アイコンを点線で囲んで表示)。 また、これらのデータ値はセンサデータグラフ枠上に表示します。これらの値が時間の経過に応じてどの様に変化したかを容易にチェックできます。

このセンサデータグラフ枠はセンサ選択 combo-box を含みます。 センサ制御ツールバー上のボタンを使うと求めるタイプのセンサデータが表示できます。

5 SerialNet

SerialNet は ZigBeeNet ソフトウェアのコンフィギュレーションです。SerialNet を使うと、通信インタフェースを通してほとんどの ZigBit/ZigBeeNet の機能の制御が標準の Heyes AT コマンドに似たコマンドセットを使うことにより可能になります。

このコマンドはシンプルなテキスト形式で RS-232/USB インタフェースを通して与えます。その言語規則は ITU-T V.250 勧告で説明しています。([12]を参照)

注:
厳密に言えば、SerialNet は ZigBeeNet API の"上に"開発したアプリケーションです。

SerialNet アプリケーションを実行する前に、対応するイメージファイルがシリアルブートローダまたは JTAG を使って各ノードのボードにアップロードされてあることを確かめてください。(付録 A を参照)

サポートしている AT コマンド、各コマンドの文法と詳細な説明はリファレンスマニュアル[3]を参照してください。この[3]の「例」の章は、以下を行うためにどのようにコマンドを使うかを示しています:

- LED とディップスイッチを制御
- 網の構築(ノードの役割とアドレスの設定)
- ノード間でデータを送信
- PAN ID と周波数チャンネルを管理
- 遠隔での実行のためにコマンドを転送
- 終端装置の電力消費量を制御

AT コマンドの柔軟性のおかげで、ユーザ自身の特定用途やニーズを満たす自分の通信シナリオが作れます。 この例を SerialNet 評価の開始点としてお勧めします。

さまざまなターミナルプログラムを活用して AT コマンドスクリプトを入力したりボードからの応答を分析できます。 SerialNet アプリケーションを実行するには文書[3]の Examples セクションにあるインストラクションにステップ毎に従ってください。

注:
+IFC コマンドと+IPR コマンドは RS-232/USB ポートの (送受信) レートとフロー制御パラメータを

変更します。 この様なコマンドを使うなら PC 上で実行するターミナルプログラムの COM ポート設定は適切に変更するべきです。

6 シリアルブートローダ

シリアルブートローダは、JTAG を使わずに WSN ノードの中にファームウェアイメージを SREC 形式で焼き付ける為のソフトウェアです。(付録 B を参照) また、各ノードに、そのファームウェアに影響を与えずにネットワークパラメータを設定できます。

シリアルブートローダは 2 つの部分から成ります: Windows プラットフォーム用の PC コンソールアプリケーションと MCU 内にあるブートストラップコードです。 ZigBit 開発キットでは、各 MeshBean2 ボードは、ZigBit MCU 内にあるヒューズビットを設定し、ブートストラップをプリロードして提供します。 ブートストラップ自身は必要に応じて JTAG を使って回復できます。 これは、bootloader.hex イメージファイルの形式で提供します。(付録 A を参照)

シリアルブートローダ使用に関する網羅的な情報は文書 [8]にあります。

7. ZigBeeNet API でプログラミング

7.1 API の概要

ZigBeeNetの内部アーキテクチャは802.15.4及び, ZigBee定義の網スタックと論理層の分離に従います。 ZigBeeNet はプロトコルを含む1スタックの導入の他に、共有サービス (例:タスクマネジャー、セキュリティー、消費電力管理) 、ハードウエア・アブストラクション (例:ハードウエア・アブストラクション層 (HAL) 、ボード・サポート・パッケージ (BSP)) を導入する追加の層も含んでいます。これらの層が関係するAPIはコアスタックの機能ではありません。しかし、APIセットへのこれらの必須の追加機能は、アプリケーションの複雑さを削減し、統合を単純化する上で大きな助けになります。 ZigBeeNet API 参照マニュアルはパブリックAPI全てとその使用方法について詳細に説明しています。[4]

APSはコアスタック層の最上位にあり、アプリケーションが見ることができる、網関連の最高位レベルのAPIを提供します。 ZDOは完全準拠のZigBee 装置オブジェクトAPIを提供します。このAPIにより主網管理機能 (開始、リセット、形成、参加) が可能になります。 ZDOはZigBee装置のプロファイルタイプ、スタックが導入する装置とサービスの発見コマンドも定義します。

タスクマネジャー、セキュリティー、消費電力管理を含むサービスの"面"が3つあります。これらのサービスはユーザアプリケーションが使うことが出来、下位スタック層も利用できます。 タスクマネジャーは内部スタックコンポーネントとユーザアプリケーション間でのMCUの使用を調整するスタックスケジューラです。 タスクマネージャは優先度キューを基にした独自のアルゴリズムを使います。このアルゴリズムはマルチ層スタック環境とタイムクリティカルな網プロセスの要求用に特に調整したものです。電力管理ルーチンはスリープ状態になる準備として全スタックコンポーネントを手際よく終らせ、システム状態を保存し、立ち上げ時にシステム状態を復元する責任があります。

HAL(Hardware Abstraction Layer、ハードウェア・アブストラクション層)は API の完全なセットを含みます。これはモジュール上のハードウェア資源(EEPROM、アプリ、スリープ、監視タイマ)を使う為のものであり、短期間のデザイン・インや一連の外部周辺装置(IRQ、I2C、SPI、UART、無線)との滑らかな統合の為の参照ドライバを含みます。BSP(Board Support Package 、バードサポートパッケージ)は MeshBean 開発ボード上の標準周辺装置(センサ、UID チップ、スライダ、ボタン)を操作するドライバの完全なセットを含みます。

7.2 AVR プログラミングツールの使用

弊社はAtmelのAVR Studio[17]を使ってZigBeeNet APIに基づいたカスタムアプリケーションを開発することをお薦めします。 このマルチプラットフォーム統合開発環境(IDE:Integrated Development Environment)を使ってソースコードを編集し、コンパイルし、オブジェクトモジュールを

ライブラリとリンクし、実行ファイルを自動的に作成する等々ができます。 IDEのインストール手順についてはセクション3.3を参照してください。この製品に同梱してあるAVR Studioユーザマニュアルも参照してください。

AVR Studio は WinAVR と統合できます。これは Atmel AVR シリーズのためのソフトウェア開発ツールスイートです。Atmel AVR シリーズは Windows プラットフォーム[20]がホストする RISC マイクロプロセッサです。 WinAVR は AVR GCC コンパイラ、リンカ、自動 makefile ジェネレータ、システムライブラリなどのユーティリティセットを含んでいます。 AVR GCC プラグインをインストールすることによってこれらのツールを自動的に AVR Studio 内で機能する様にできます。 GCC コンパイラは Windows プラットフォーム上で実行するように設計してあり、C または C++コードをコンパイルする様構成してあります。 GCC コンパイラの解説は WinAVR ドキュメンテーションを参照してください。このコンパイルとリンクのためのコマンドオプションは[21]にあります。

AVR Studioではアプリケーション開発は特定のプロジェクトとして構成します。 このプロジェクトに関する必要な情報は全てプロジェクトファイルの中に持ちます。 AVR Studioにアサインしたそのようなファイルは*.aps という拡張子が付いているので、ダブルクリックするとAVR Studio内で自動的に開きます。

AVRプロジェクトを構成する最も容易な方法は、Makefileを使うことです。Makefileはプレーンテキストファイルで、拡張子がありません。 Makefileはコンパイルとリンクフラグを指定します。Makefileはヘッダファイルを含み、システム・オブジェクト・ライブラリとリンクする為に対応するディレクトリを指定します。

必要な ZigBeeNet ソフトウェアはZDKに同梱の CD の中に入っており、付録 A に提示してある構造化された“ZigBeeNet”サブディレクトリ内にあります。

7.3 ミニマム アプリケーションのビルド方法

プログラミングが直ぐ開始できる様に、ユーザのサンプルアプリケーションは、必要なアプリケーション構成とコーディング上の約束を示す様に設計してあります。 このアプリケーション(“Hello World!“のような)は GPIO インタフェースを使って MeshBean2 ボード上の LED の点滅を常時行います。 この「ミニマム アプリケーション」のソースコードは付録 C にあります。 付録 C には付録 A で指定した構造に対応する Makefile もあります。これらは別の ./Sample Applications/Blink サブディレクトリ内にあります。 結果として生成するイメージファイルも CD 上にあります。 これらは以下に説明する様にいつでも再ビルドできます。

お客さまのハードディスク上で“./Sample Applications/Blink/“サブディレクトリから

blink.apsファイルを開き、AVR Studio のメインメニューからBuild/Rebuild All を実行してください。 blink.hexとblink.srecイメージファイルが生成されます。ブリンクにはEEPROMを使っていないので*.eepイメージファイルは生成されません。「ミニマム アプリケーション」を試験するにはこのどちらかのイメージファイルをMeshBean2ボードに、セクション6または付録Bにある手順に従ってアップロードしてください。

このミニマムコードを修正して、他のZigBeeNet API機能を使ってアプリケーションの機能性を拡張できます。 作成したアプリケーションコードは[4]で指定するプログラミング上の約束を満たしていることを確かめてください。

他の API デモも使ってみて、同様の方法でアプリケーションを新機能で拡張してみてください。(セクション7．4を参照) お使いの前にご自分のアプリケーションがボードにアップロードしてあることを確かめてください。

7.4 サンプルアプリケーション

開発キットでは以下の ZigBeeNet API サンプルアプリケーションセットをソースコードで提供します。括弧内はアプリケーション名です:

- ZBNDemo アプリケーション(ZBNDemo)
- 低消費電力の網アプリケーション(Lowpower)
- ピア･ツー･ピア データ交換アプリケーション (Peer2peer)
- ピンポンアプリケーション(Pingpong)
- ハードウエア試験 (HardwareTest)

ZBNDemoは開発キット独自のアプリケーションであり、ZigBeeNetソフトウェアとMeshBean2ハードウェアを使った網の形成をデモンストレーションします。ZBNDemoでは、ノードは[6]で説明する適切なプロトコルを基に通信します。ZBNDemoの詳細はセクション4に示します。

ZBNDemoアプリケーションのソースコードは“ ./Sample Applications/ZBNDemo“サブディレクトリ (附録Aを参照) 内にあります。開発キットをユーザのPCに事前にインストールしてください。(セクション3．3を参照)

網パラメータ (セキュリティー設定を含む) とその既定値はMakefileに以下の様に定義されます。

```
 ####################################
# Stack configuration parameters
####################################
#CFLAGS += -DSYS_LOG_ON
```

```
CFLAGS += -DSYS_ASSERT_ON
# Enable the security in the stack
CFLAGS += -D_SECURITY_
# Enable the security on NWK only (preconfigured key)
CFLAGS += -D_NWK_SECURITY_
# Pre-configured key which is used by NWK
CFLAGS +=
-DCS_NETWORK_KEY="{0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,
0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC}"
CFLAGS += -DTIMER_SENDING_PERIOD=20000
CFLAGS += -DCS_END_DEVICE_SLEEP_PERIOD=30000
CFLAGS += -DAPP_CHANNEL_MASK=0x01000000
CFLAGS += -DAPP_COORD_UID=0xAAAAAAAAAAAAAAAALL
CFLAGS += -DCS_NEIB_TABLE_SIZE=7
CFLAGS += -DCS_MAX_CHILDREN_AMOUNT=6
CFLAGS += -DCS_MAX_CHILDREN_ROUTER_AMOUNT=2
CFLAGS += -DCS_ROUTE_TABLE_SIZE=35
# Used only for static addressing
#CFLAGS += -DCS_NWK_ADDR=0x0001
#CFLAGS += -DCS_NWK_UNIQUE_ADDR=true
```

ZBNDemoアプリケーションをコンパイルするにはmakeユーティリティを使います。そうでなければ AVR Studioのサブディレクトリ.API/SampleApplication/ZBNDemo/からWSNDemo.apsファイルを開き、メインメニューからBuild/Rebuild Allアイテムを単に実行してください。ZBNDemo.hex, ZBNDemo.eep, ZBNDemo.srecイメージファイルが生成されます。Low-Power, Peer-to-peer, Ping-Pong アプリケーションの詳細説明は[4]にあります。

8. トラブルシューティング

システム運用上のどのような問題の場合でも、先ず電源供給をチェックし、網を構成するすべての機器が正しく接続してあることを確かめてください。

お客さまの PC が最小のシステム要件を満たしているかどうかをチェックしてください。(セクション 3.2 を参照) PC インタフェース(COM、USB)が存在し、ドライバがインストールしてあるかどうかをチェックしてください。

ノードが応答していないか異常な動きをしているなら LED 表示をチェックしてください。 DIP スイッチがボードの上のアプリケーションに従って設定してあることを確かめてください。

必要ならセクション3．7で説明したように特定のノードを再テストできます。

ノードをリセットする様要求されるかも知れません。

以下に開発キットをご使用の間に発生するかも知れない代表的な問題とその解決法を示します。

表 14 代表的な問題と解決策

問題	解決
ボードの動きが LED に表示されません	ZBNDemo イメージかハードウェアテストイメージのどちらかがロード済みなことを確かめてください。 SerialNet では LED ステータスは AT コマンドで制御しています。
ボードは外部からのコマンドに応答しません (外部アンテナの場合)	外部アンテナが壊れていないか、それがボードにちゃんと取り付けられていることを確認してください。
複数のボードを同じ PC に接続する場合、ボードの検出が ID 認識衝突のため失敗します。	Silicon Laboratories社が提供するUSBView.exe ユーティリティを使って接続したボードのIDを検出してください。このユーティリティは http://www.silabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/USBXpress.htm. からアップロードできます。 Silicon Laboratories からの CP210xSetIDs.exe ユーティリティが使えます。これは AN144SW に含まれています。この説明は http://www.silabs.com/public/documents/tpub_doc/anote/M

	icrocontrollers/Interface/en/an144.pdf にあり、 http://www.silabs.com/public/documents/software_doc/othersoftware/Microcontrollers/Interface/en/an144sw.zip. からアップロードできます。
WSN モニタはスタートできません	Javaマシンが PC に正しくインストールされていることを確かめてください。Javaランタイム環境のインストレーションプログラムは ./Third Party Software/ ディレクトリの中の jre-6u3-windows-i586-p-s.exe file にあります。(付録 A を参照)
WSN モニタのトポロジー枠にノードが全く表示されません	WSN モニタが適切な COM ポートを使っているかチェックしてください。もしそうでなければそれを変更し、プログラムを再スタートしてください。
WSN モニタはセンサデータグラフ枠に NO DATA を表示しています	どのノードも選んでいません。必要なノードの上でマウスをクリックし選んでください。
トポロジー枠上で表示したノードタイトルはノードの宛先を示していません	表示したタイトルは必ずしもノード機能と関連してはいませんが、それらはいつでも再定義できます。 これらの名前は、ノードにマッピングした MAC アドレスと共にノードタイトルファイル中に記憶します(セクション 4．5を参照)。
WSN モニタスタートアップ時に、すべてのノードの LED がブリンクしているか、又は全くフラッシュしていません	ZBNDemo アプリケーションはノードにアップロードされていません。 ノードにこのアプリケーションをアップロードしてください。
シリアルブートローダも他のアプリケーションもハードウェア試験を除いてノードで動きません	ボード上の J3 の設定が実際の接続タイプ(シリアルかUSB)と一致していることを確かめてください。 JTAGを通してノードをプログラムした後、マイクロコントローラのフラッシュメモリが消去されていなか、ブートストラップが失くなっていないかどうかを確認してください。

付録

付録 A　ZDKファイル構造

ZDKのユーザのPCへのインストレーションはZDKソフトウエアとドキュメントCDから行います（セクション3．3を参照）。その結果ユーザが指定した場所に以下のファイル構造が生成されます。

表 15　CD の内容

ディレクトリ/ファイル	説明
Readme.html	ドキュメントファイルへのリンクを含む導入ドキュメント
ZigBit Development Kit Release Notes.txt	開発キットリリースノート
EULA.txt	エンドユーザ　ライセンス契約書
./Documentation	ハードウエア、ソフトウエア、データシート、アプリケーションノート等のドキュメント
./Product Information	Getting Startedドキュメント、製品概要、ケーススタディードキュメント
./Bootloader/Bootloader.exe ./Bootloader/bootloader.hex	シリアルブートローダ実行ファイル ブートストラップコードを含むバイナリイメージファイル
./Evaluation Tools/Hardware Test/ HardwareTest.srec ./Evaluation Tools/Hardware Test/ HardwareTest.hex	ハードウエア試験イメージファイル
./Evaluation Tools/ZBNDemo (Embedded)/ ZBNdemo.srec ./Evaluation Tools/ZBNDemo (Embedded)/ ZBNdemo.hex	ZBNDemo イメージファイル
./Evaluation Tools/ZBNDemo (WSN Monitor)/Start.exe	WSN Monitor 実行ファイル。全リソースファイル付き
./Evaluation Tools/SerialNet/ serialnet.srec ./Evaluation Tools/SerialNet/ serialnet.hex	SerialNet イメージファイル
./ZigBeeNet/Components	ZigBeeNetスタック用ヘッダファイル

ディレクトリ/ファイル	説明
./ZigBeeNet/Components/BSP/	ZigBeeNet BSP用ソース、ヘッダ、ライブラリファイル
./ZigBeeNet/lib	ZigBeeNetスタック用ライブラリファイル
/Sample Applications/ZBNDemo	ZBNDemo アプリケーション用ソースファイルとイメージファイル。 ソースコードは Complete サポートパッケージだけで利用可能
./Sample Applications/Blink	Blinkアプリケーション用ソースファイルとイメージファイル
./Sample Applications/Lowpower	Low Power サンプルアプリケーション用ソースファイルとイメージファイル
./Sample Applications/Peer2peer	Peer-to-Peer サンプルアプリケーション用ソースファイルとイメージファイル
./Sample Applications/Pingpong	Ping-Pong サンプルアプリケーション用ソースファイルとイメージファイル
./Evaluation Tools/Range Measurement Tool/range_tool.vi ./Evaluation Tools/Range Measurement Tool/receiver.hex ./Evaluation Tools/Range Measurement Tool/receiver.srec ./Evaluation Tools/Range Measurement Tool/transmitter.hex ./Evaluation Tools/Range Measurement Tool/transmitter.srec	電波到達範囲測定ツールアプリケーションの GUI とイメージファイル
./Third Party Software/ CP210x_VCP_Win2K_XP_S2K3.exe ./Third Party Software/ jre-6u3-windows-i586-p-s.exe	USB – UARTブリッジ VCP ドライバ インスタレーションプログラム Java 実行環境インスタレーションプログラム

付録 B　JTAG エミュレーターの使用

JTAG を使ってプログラミングするとロードプロセスをより柔軟に管理できますが、特別なハードウェアが必要です。 Windows 環境用に、弊社は AVR Studio 4.13 + Service Pack2 をお勧めます。 VaRICE2.40 は Linux 用に使えます。 どちらの場合にも JTAG エミュレータとして Atmel 社の JTAGICE mkII をお勧めします。 他のプログラミング装置も同様にご利用頂けますが、利用前にお使いのモデルが Atmega1281 MCU のプログラミングをサポートすることを確かめてください。

AVR Studioを使ってボードのフラッシュメモリとEEPROMをインテルHEX形式でプログラムできます。EEPROMイメージは .eep拡張子を持ち、一方フラッシュイメージは .hex拡張子を持っています。ファームウェアをアップロードするには、機器メーカのマニュアル[17], [18], [19]にある手順に従ってください。 ポップアップ・ウィンドウの例を図13に示します。

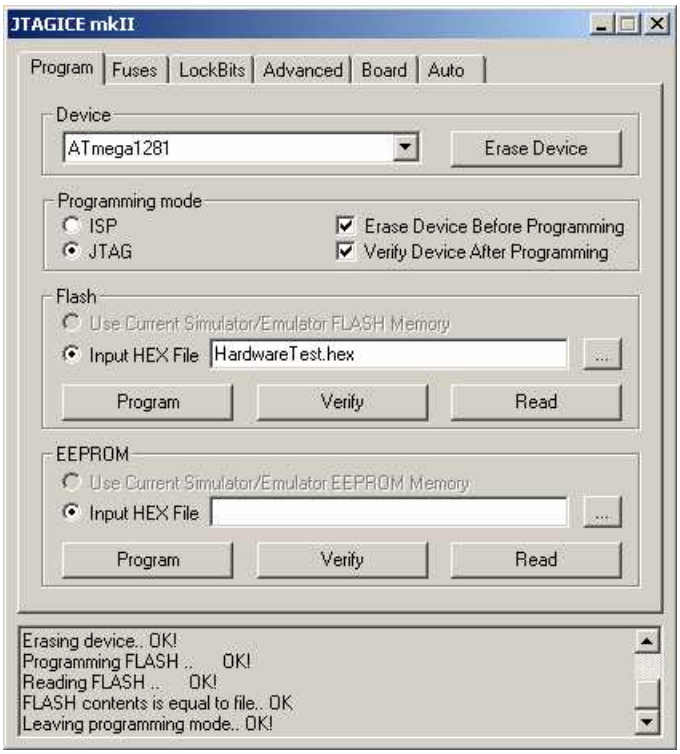

図 13 JTAG ファームウェアアップロード用 AVR Studio ダイアログボックス

WinAVR 環境(http://sourceforge.net/projects/winavr)の一部で、有名なコマンドラインユーティリティである avrdude もアップロード用に使えます。 このユーティリティはインテル HEX とモトローラ HEX 両方の形式を認識します。

重要な注:
シリアルブーティングに必要なブートストラップコードの衝突事故を避ける為に、JTAG使用中は機

器を消去しないでください。
JTAGプログラミングのためにはBoot Reset vector のヒューズビットは使用不可にするべきです。 シリアルブーティングを可能にするには、このヒューズビットは使用可能にするべきです。

付録C. ミニマム アプリケーション

ソースコード

```
#define HALF_PERIOD_BUTTON BSP_KEY0   // Button that reduces
blink interval to a half.
#define DOUBLE_PERIOD_BUTTON BSP_KEY1 // Button that doubles
blink interval.

/*****************************************************************
  blink.c

  Blink application.

  Written by V.Marchenko
*****************************************************************/

#include "appTimer.h"
#include "sliders.h"
#include "buttons.h"
#include "leds.h"
#include "zdo.h"

#ifndef BLINK_PERIOD
#define BLINK_PERIOD 1000 // Initial blink period, ms.
#endif

#ifndef MIN_BLINK_PERIOD
#define MIN_BLINK_PERIOD 100  // Minimum blink period, ms.
#endif

#ifndef MAX_BLINK_PERIOD
#define MAX_BLINK_PERIOD 10000 // Maximum blink period, ms.
#endif

#define BLINK_INTERVAL (BLINK_PERIOD / 2)         // Blink
interval.
#define MIN_BLINK_INTERVAL (MIN_BLINK_PERIOD / 2) // Minimum
blink interval.
#define MAX_BLINK_INTERVAL (MAX_BLINK_PERIOD / 2) // Maximum
blink interval.
```

```
static HAL_AppTimer_t blinkTimer; // Blink timer.

static void buttonsReleased(uint8_t buttonNumber); // Button
release event handler.
static void blinkTimerFired();                      //
blinkTimer fire event indication.

/************************************************************************
  Description: application task handler.

  Parameters: none.

  Returns: nothing.
************************************************************************/
void APL_TaskHandler(void)
{
  BSP_OpenLeds(); // Enable LEDs
  BSP_OpenButtons(NULL, buttonsReleased); // Register button
event handlers

  // Configure blink timer
  blinkTimer.interval = BLINK_INTERVAL;       // Timer
interval
  blinkTimer.mode     = TIMER_REPEAT_MODE; // Repeating mode
(TIMER_REPEAT_MODE or TIMER_ONE_SHOT_MODE)
  blinkTimer.callback = blinkTimerFired;    // Callback
function for timer fire event

  HAL_StartAppTimer(&blinkTimer); // Start blink timer
}

/************************************************************************
  Description: blinking timer fire event handler.

  Parameters: none.

  Returns: nothing.
************************************************************************/
static void blinkTimerFired()
{
```

```
  BSP_ToggleLed(LED_RED);
  BSP_ToggleLed(LED_YELLOW);
  BSP_ToggleLed(LED_GREEN);
}

/*************************************************************
  Description: button release event handler.

  Parameters: none.

  Returns: nothing.
*************************************************************/
static void buttonsReleased(uint8_t buttonNumber)
{
  HAL_StopAppTimer(&blinkTimer); // Stop blink timer

  // Dependent on button being released, update blink
interval
  if (HALF_PERIOD_BUTTON == buttonNumber)
  {
    blinkTimer.interval /= 2;
    if (blinkTimer.interval < MIN_BLINK_INTERVAL)
      blinkTimer.interval = MIN_BLINK_INTERVAL;
  }
  else if (DOUBLE_PERIOD_BUTTON == buttonNumber)
  {
    blinkTimer.interval *= 2;
    if (blinkTimer.interval > MAX_BLINK_INTERVAL)
      blinkTimer.interval = MAX_BLINK_INTERVAL;
  }


  blinkTimerFired();              // Update LED status
immediately.
  HAL_StartAppTimer(&blinkTimer); // Start updated blink
timer.
```

```
}

/*************************************************************
  Description: just a stub.

  Parameters: are not used.

  Returns: nothing.
*************************************************************/
void ZDO_MgmtNwkUpdateNotf(ZDO_MgmtNwkUpdateNotf_t
*nwkParams)
{
  nwkParams = nwkParams;  // Unused parameter warning
prevention
}

/*************************************************************
  Description: just a stub.

  Parameters: none.


  Returns: nothing.
*************************************************************/
void ZDO_WakeUpInd(void)
{
}

/*************************************************************
  Description: just a stub.

  Parameters: none.

  Returns: nothing.
*************************************************************/
void ZDO_SleepInd(void)
{
}

//eof blink.c
```

Makefile

```
PROJNAME = blink

# Path to ZigBeeNet stack
#STACK_PATH =
../../../../ZigBeeNet/tags/release_zbn_1_0/Components
STACK_PATH = ../../ZigBeeNet/Components

include $(STACK_PATH)/../../ZigBeeNet/lib/MakerulesZbnAll
include ../Makerules

#-------------------------------------
# Application parameters
#-------------------------------------
CFLAGS += -DBLINK_PERIOD=1000
CFLAGS += -DMIN_BLINK_PERIOD=100
CFLAGS += -DMAX_BLINK_PERIOD=10000

#-------------------------------------
# Stack parameters being set to Config Server
#-------------------------------------



## app include dirs
INCLUDES += -I.

## app objects to build
OBJ = blink.o
## path to app objects
VPATH += .:

## objects to build with -O0
#DBG_OBJS =

#-------------------------------------
# Build
#-------------------------------------
```

```
all: clean $(PROJNAME).elf $(PROJNAME).srec $(PROJNAME).hex
size

$(OBJ) $(STACK_OBJ): %.o: %.c
   $(CC) -c $(CFLAGS) $(INCLUDES) $^ -o $@

$(DBG_OBJ): %.o: %.c
   $(CC) -c $(CFLAGS) -O0 $(INCLUDES) $^ -o $@

#---------------------------------------
# Link
#---------------------------------------
$(PROJNAME).elf: $(OBJ) $(DBG_OBJ) $(STACK_OBJ) Makefile
   $(CC) $(CFLAGS) $(OBJ) $(DBG_OBJ) $(STACK_OBJ) -o
$(PROJNAME).elf -L$(LIB_PATH) -L$(BSP_PATH)/lib -
l$(STACK_LIB) -lBSP
   rm -f *.o

%.srec: %.elf
   avr-objcopy -O srec --srec-len 128 $< $@

%.hex: %.elf
   avr-objcopy -O ihex $(HEX_FLASH_FLAGS)  $< $@

size:
   avr-size -td $(PROJNAME).elf

clean:
   rm -rf *.elf *.hex *.srec *.o

# eof Makefile
```