参 考 資 料

SimpliciTI：簡易モジュール型RF（無線）ネットワーク Developers Note

Version 1.10
August 31, 2007
Literature No. : JAJA170

TEXAS INSTRUMENTS

内　容

説明図

説明表

1. はじめに

このドキュメントでは、SimpliciTIのプロトコル·サポートを効果的に利用するために必要な情報を提供します。

このサポートにはソース·コードがあるため、本文中ではソース·コード·ファイルが頻繁に参照されます。

このドキュメント内のハードウェア関連の記述は、現時点ではCC2500/CC1100無線通信のものに限定されています。ただしプロトコル自体をサポートしているファームウェアについての記述は、ハードウェア依存ではありません。

2. 参照文献

[1] SimpliciTI Specification, Texas Instruments, 2007

3. 概要

SimpliciTIでは、2つの基本的なトポロジーをサポートします。その一方がスター型トポロジーであり、ハブとして使用されるアクセス·ポイントです。アクセス·ポイントは主に、ネットワーク管理のために使用されます。スリープ状態エンド·デバイス用の蓄積転送や、メンバーシップの許可、リンキングの許可、セキュリティ鍵などに関するネットワーク·デバイス管理といった機能をサポートします。

アクセス·ポイントは、エンド·デバイスの機能もサポートすることができます。つまりネットワークのセンサまたはアクチュエータのインスタンスを、アクセス·ポイント自体が作成(instantiate)することが可能です。

プロトコルのサポートは、少数のAPI呼び出しで実現されます。これらのAPIでは顧客アプリケーションのピア·ツー·ピア·メッセージングをサポートします。2つのアプリケーション間の接続関係(association)は「リンク」と呼ばれ、ランタイムで行われます。リンク処理により、コネクション·ベースのオブジェクトが作成されます。このオブジェクト経由で、アプリケーション·ピアがメッセージを送信できます。確立されたコネクションは、双方向コネクションとなります。

4. ハードウェア構成

4.1 MCUのインターフェイス

CC1100/CC2500の無線通信では、無線～MCU間のインターフェイスの仕様を定めています。インターフェイスではSPI通信の他に、最大2つの追加ラインが提供されます。この追加ラインはMCUの汎用IOピンに接続すると、割り込みを生成するように構成できます。参照設計の多くではこれらが両方とも接続されると規定していますが、本ドキュメントでは片方(GDO2)だけが接続されると仮定しています。

4.2 無線の構成

無線用の初期値のソース·ファイルは、TIのSmartRF04ツールによって生成されたものです。レジスタ構成はエクスポートされて、配布コードに組み込まれています。

さらに、便宜上いくつかの微調整が追加されています。これらは、SmartRF04ツールによって明示的にエクスポートされない設定レジスタから構成されます。

5. アーキテクチャの概要

5.1 プロトコル層

プロトコルは、ピア·ツー·ピア通信を主な目的とするアプリケーション層で機能します。ピアは通常、センサのコントロール·オブジェクトとアクチュエータのコントロール·オブジェクトになります。

直接センサ·アクチュエータのピアもサポートされています。プロトコルではどのピアも平等に扱います。

実装という観点からのプロトコルの目的は、様々な任意のピア·アプリケーション同士のリンクを簡単にすることです。

図1は階層化の概略図です。

正式なPHY層(物理層)やデータリンク層(MAC/LLC)はありません。データはすでにフレーム化された状態で無線から直接受信されるため、物理層やデータリンク層の機能は無線によって実行されます。

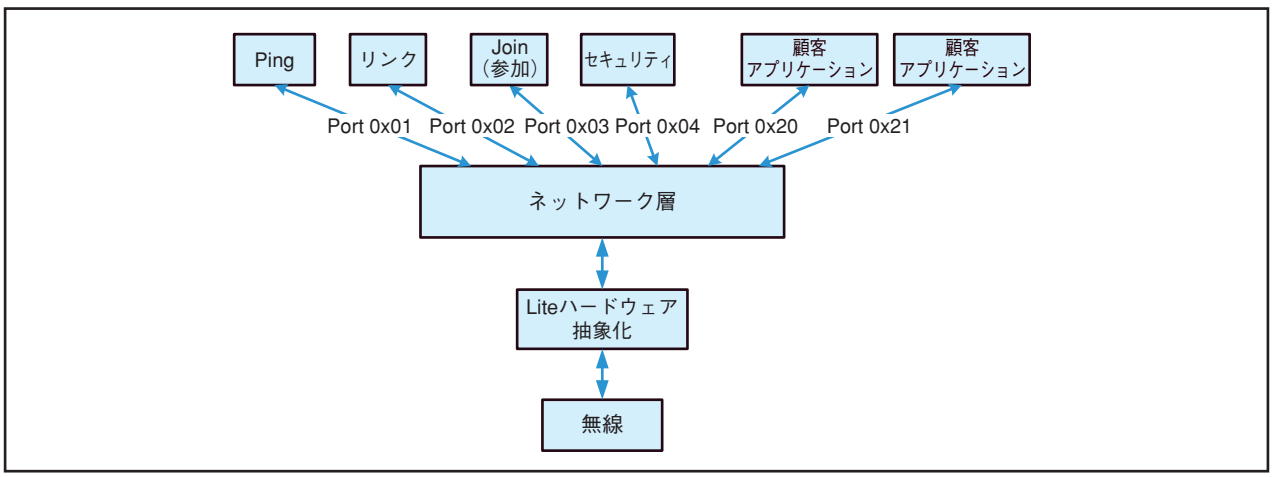

図 1. SimpliciTI の論理層

無線と情報をやり取りするNWK層呼び出しからSPIインターフェイスを抽象化する、BSP（Board Support Package）というエンティティがあります。BSPは、アプリケーションで使用する一般的なハードウェア抽象化のサポートを意図したものではありません。サポート対象となるのは、（SPIインターフェイスのように）NWK～無線間のインターフェイスを直接サポートするようなサービスのみです。BSPでは便宜的に、汎用IOピンに接続したLEDやボタン/スイッチ等の周辺装置もサポートしますが、UARTドライバ、LCDドライバ、タイマ·サービス等には対応していません。

NWK層ではRx（受信）キューとTx（送信）キューを管理し、フレームを宛先に送出（dispatch）します。宛先は常に、ポート番号によって指定されたアプリケーションです。NWK層がアプリケーションに変わってフレーム処理を行うことはありません。

ポートの発想はTCP/IPポートと似ており、概念的にはアドレスの延長です。ネットワークのフレーム·オーバーヘッドは除去され、残存したペイロードは、指定されたポートに常駐するアプリケーションに渡されます。

NWK層のアプリケーションは「ウェルノン（よく知られている）」ポートにあります。これらの持つ値はすべて、0x1Fです。これらはNWK層自体によって、ネットワークの管理に使用されます。これらのポートでは、顧客アプリケーションに直接アクセスされることを意図していません。

顧客アプリケーションのポートは、リンク処理の間にNWKによって割り当てられます。アプリケーション側では、これらのポートはリンクIDとして認識されます。リンクIDからアドレスへのマッピングはNWKによって行われます。これはソケットを使用したアプローチに似ています。ポート·オブジェクトの割り当てと保守に関して、アプリケーションが責任を持つことはありません[1]。

[1] ただし、Pingアプリケーションは例外です。このアプリケーションの主な目的はデバッグであり、それ以外の目的ではうまく動作しません。（ちょうどIP Pingの場合と同様に）このポートを使用するには宛先アドレスが分かっている必要がありますが、たいていの場合アプリケーション層では宛先アドレスが分かっていません。

5.2 NWKアプリケーション

NWKアプリケーションは、ネットワーク管理をサポートします。NWKアプリケーションを顧客の開発環境の一部にすることは意図されていません。可能だとしてもPingくらいでしょう。SimpliciTIプロトコルがどのように通信環境をサポートするかについて理解を深めるために、この章ではNWKアプリケーションについて説明します。

アプリケーション	ポート	説　明
Ping	0x01	TCP/IPアプリケーションのPingと同じものです。受信されたペイロードを送信側にエコーバックします。直接アドレシングのみ。
Link（リンク）	0x02	異なるデバイス上の2つのピアを接続するために使用します。
Join（参加）	0x03	アクセス·ポイントが存在する場合に、ピアへのアクセス権を取得するために使用します。
Security（セキュリティ）	0x04	鍵などのセキュリティ情報を交換するために使用します。
Freq（周波数）	0x05	周波数移行を管理して、周波数アジリティをサポートするために使用します。
Mgmt（管理）	0x06	汎用NWK管理アプリケーションです。ポーリング·ポート等に使用します。

表 1. NWK アプリケーション

5.3 ピア層の特徴

このアーキテクチャには、NWK層とアプリケーション層という2つの基本的なソフトウェア·ピア層が存在します。図1に示すように、アプリケーション層自体がNWKアプリケーション部と顧客アプリケーション部の2つに区分けされています。それぞれの特徴を下の表に記載してあります。

開発という目的を考えると、SimpliciTIではNWK層の応答（acknowledgement）をサポートしないということに注意する必要があります。このことの主な結果として、次のようなことに対するサポートをアプリケーション自体が行う必要が生じます。

- アプリケーションの最大ペイロードよりも大きいメッセージ用のセグメンテーションとリアセンブリ
- 失われたデータ（NWKの保証を受けられない、トランスポート層形式でのデリバリ）
- 冗長なデータ（重複フレームがNWKに認識されない）

階層	コネクションの種類	応答（Acknowledgment）
NWK	コネクションレス型	なし
NKWアプリケーション部	コネクションレス型	あり[2]
顧客アプリケーション部	コネクション型。 各コネクションは双方向。	オプション。 顧客側の実装により異なる。

表 2. ピア層の特徴

[2] これにはいくつか例外があります。NWKアプリケーションのフレームには応答が返されるのが一般的ですが、応答を返されないフレームもあります。いずれにしても、これらの交換は顧客アプリケーションからは見えません。

6. プロトコルの概要

プロトコルの提供する機能は、コネクション·ベースのピア·ツー·ピア通信のサポートのみを意図したものです。その目的は、基本的な無線通信部（radio portion）を包み隠すことにより、開発中に顧客がその領域について考えなくても済むようにすることです。

機能は、顧客アプリケーションで使用可能なシンプルで少ないAPI呼び出しで実現されます。簡易性を優先すれば、柔軟性を犠牲にすることになります。裏返せば、あまり高い柔軟性を必要としない状況で、この簡易でフットプリントの小さいプロトコルを使用するという状況を想定することが可能です。

次の説明では、プロトコルをサポートするために実装されたメカニズムを要約しています。

6.1 トポロジー

プロトコルは、簡易スター型トポロジーに対してのみ送信を行います。ネットワーク管理に必要なサポートは、ハブが提供します。このサポートにはネットワークのメンバーシップや、周波数アジリティやセキュリティといったネットワーク管理機能などが含まれます。

プロトコルはまた、ハブのないネットワークもサポートします。この場合のネットワークは、ブロードキャスト·ネットワーク、またはピア·デバイスのアドレスをあらかじめ設定してあるネットワークのどちらかとして実現されます。

スター型トポロジーをシンプルにしておくために、このプロトコルでは正式な経路指定（routing）メカニズムをサポートしません。

6.2 デバイスのオブジェクト

デバイス·オブジェクトは、ソフトウェア·オブジェクトです。サポートされているデバイス·オブジェクトは、エンド·デバイス、アクセス·ポイント、レンジ·エクステンダの3つです。各オブジェクトは論理的に構成されるため、単体のハードウェア·プラットフォーム上で複数のデバイス·オブジェクトが実現できるようになっています。例えば、アクセス·ポイントを含むプラットフォームがエンド·デバイスになることも可能です。ただし、ハードウェア·デバイスひとつを単独で占めるか、または他のエンド·デバイスと共有するエンド·デバイスが大部分になると思われます。

6.2.1 エンド·デバイス

これらはもっとも簡易性の高いデバイスであり、ネットワーク上のすべてのセンサ/アクチュエータの中心（locus）です。エンド·デバイスではアプリケーション·ピアのホスティングを行います。エンド·デバイスのみのホスティングを行うハードウェア·プラットフォームは、バッテリーから電源を取ることもあります。

6.2.2 アクセス·ポイント

アクセス·ポイントは、存在する場合にはネットワーク内のスター型ハブとして機能します。これは常時接続デバイスです。1つのネットワークにつき、1つのアクセス·ポイントのみが許可されます。これは、下記のコンストラクトを使用して実現されます。

アクセス·ポイントは同じハードウェア·プラットフォーム上のエンド·デバイスと共存することが可能です。アクセス·ポイントでは、センサまたはアクチュエータ（あるいは両方）をネットワーク上で実現するピア·アプリケーションのホスティングを行うことができます。

プロミスキュアス·モードで稼動するアクセス·ポイントでは、レンジ内のすべてのパケットを受信します。インフラのサポートに加えて、アクセス·ポイントではフレームの再送を行うことにより、エンド·デバイスのレンジの拡張を促進します。

6.2.3 レンジ·エクステンダ

アレンジ·エクステンダの目的は、ネットワーク上の無線通信の範囲を拡張することです。これらは常時接続デバイスです。主な機能は、フレーム送信側の影響の範囲を拡張して効率的にフレームをリピートすることです。現時点では、ネットワークの範囲にはレンジ·エクステンダ4つ分までという制限があります。

一般的な使用方法とはみなされていませんが、レンジ·エクステンダは同じハードウェア·プラットフォーム上で他のエンド·デバイスと共存することが可能です。このようなレンジ·エクステンダでは、センサまたはアクチュエータ（あるいは両方）をネットワーク上で実現するピア·アプリケーションのホスティングを行うことができます。

プロミスキュアス·モードで稼動するレンジ·エクステンダでは、レンジ内のすべてのパケットを受信します。

6.3 アドレス名前空間

1つのネットワーク·アドレスは、プラットフォーム·ハードウェア·アドレス部と、アプリケーション·アドレス（ポート）部の2つで構成されています。

各デバイスのハードウェア·アドレスは、ランタイムではなくビルドタイムに割り当てられます。このアドレスは各デバイスごとに一意である必要があります。ハードウェア·アドレス空間の管理は顧客側の担当になります。アドレス解決プロトコルはありません。

現時点のハードウェア·アドレス空間には、符合なしの文字列として処理される4バイト分の広さがあります。CC1100/CC2500の無線通信では、先頭のアドレス·バイトを0x00または0xFFにすることはできません。フレーム·レイアウトに基づき、先頭バイトはこれらの無線通信時にRx（受信）でチェックしたロケーションに置かれます。

値0x00と値0xFFは、無線のフィルタリング機能によってブロードキャスト·アドレスと解釈されます。この制限を実行するためのチェックは行われません。

アプリケーション·アドレス(ポート)はよく知られている固定ポートか、デバイスのリンク·プロシージャ中にランタイムで割り当てられたポートです。顧客ソフトウェアの制御下にはありません。

6.4 ネットワーク規律

この章では、ピア·ツー·ピア接続がどのように実行され、ネットワーク·メンバーシップがどのように制御されるかについての概要を述べます。

下記のプロセスに関連するフレームの詳細については、[1]を参照してください。

6.4.1 リンキング

リンキングは、SimpliciTIのAPIを使用してアプリケーション·ピア·ツー·ピアのコネクションを確立するための手段です。リンクはペア単位で確立されます。あるペアの片方のアプリケーションではリスニングを行って、ペアとなるべきもう一方のアプリケーションからのリンク·メッセージを探します。結果として成立するコネクションが双方向[3]のため、どちらのアプリケーションでリスニングを行い、どちらからリンク·メッセージを送信するかについての決まりはありません。リンク·セッションは通常の場合、エンド·ユーザによる物理的な介入(ボタン押下など)によって起こります。

リンク·メッセージには、Link(リンク)トークン(現時点では4バイトのオブジェクト)が含まれます。Link(リンク)トークンはリスナーによるピアの認証に使用されます。顧客によってビルドタイムに設定されたデフォルトのLink(リンク)トークンもあります。ネットワークにアクセス·ポイントがある場合は、さらに制約を追加することもできます。6.4.2.を参照してください。

リンクは、論理的なエンティティです。単一のプラットフォームでも、複数のピア·ツー·ピアのリンクをサポートできます。これらは同じ2つのアプリケーション間の複数リンクにすることも、単一のリンクを持つ複数のアプリケーション·ペアにすることも、その他どのような組み合わせにもできます。ペアは、明確に区別できる任意の2つのプラットフォーム上に存在することが可能です。SimpliciTIでは、同一プラットフォーム上にある2つのアプリケーション間のリンクはサポートしていません。

デバイス上のリンクの数は、RAMとポート·アドレス空間の容量のみに制限されます。[4]

6.4.2 Join(参加)

Join(参加)は、ネットワークにアクセス·ポイントがある場合のみサポートされます。Join(参加)はSimpliciTIのAPIでサポートするわけではなく、初期化呼び出しのサイド·エフェクトです。Join(参加)は、アクセス·ポイントからネットワークにアクセスできる権利をプラットフォームが得るためのプロセスです。また初期化後、他のアクションが取られる前にデバイスで行われる最初のアクションでもあります。

ネットワークへのJoin(参加)を希望するプラットフォームは、Join(参加)メッセージを送信します。メッセージには、顧客がビルド時に設定するJoin(参加)トークン(現時点では4バイトのオブジェクト)が含まれています。Join(参加)トークンは、2つのアクセス·ポイントが両方とも、ネットワークに参加しようとする新しいデバイスに応答することを避けるために使用されます。Join(参加)トークンがアクセス·ポイントの想定するトークンと合致すれば、プラットフォームがネットワーク上で正しく情報をやりとりするために必要なネットワーク情報がアクセス·ポイントから返信されます。現時点では、この情報にはネットワーク用のLink(リンク)トークンが含まれます(6.4.1参照)。また、セキュリティ鍵や他のネットワーク·パラメータ(周波数アジリティ用のチャネル·テーブルなど)も含まれます。これらの機能は、最初のリリースには実装されていません。

一意のJoin(参加)トークンを使用していると、市販のデバイスをネットワークに付加する場合に面倒なことになるかもしれません。インストールの時点で、その新しいデバイスが何らかの方法でネットワークのJoin(参加)トークンを認識する必要があるからです。このような状況に対応するために、アクセス·ポイント内のJoin(参加)コンテクストをアクティブまたは非アクティブに設定するための追加の機能があります。このコンテクストは、Link(リンク)コンテクストを設定するのと同じ方法(ボタン押下など)を使用して、アクセス·ポイント上で起動することもできます。デフォルトのJoin(参加)トークンを使用することもできます。デフォルトでは、Join(参加)コンテクストは常にアクティブです。

ネットワークに参加(Join)するという行為からは、Join(参加)デバイスに対するLink(リンク)トークン等のインフラ情報以外のものは発生しません。アクセス·ポイントでは参加デバイスの追跡(track)は行いません。ただし例外として、参加デバイスがポーリング·デバイスである場合には追跡を行います。ポーリング·デバイスにはアクセス·ポイントから蓄積転送サポートを提供する必要があるためです。(6.4.3参照)

[3] アプリケーションがTx(送信)専用プラットフォームにある場合、役割の割り当てには制約が生じます。

[4] 現時点では、ポート·アドレス空間には30リンク分の余裕があります。

6.4.3 スリープ状態エンド・デバイス

スリープ状態エンド・デバイスでは、メッセージを取得するために次の2つ方法のうち1つを使用できます。第1の方法では、アクセス・ポイントに対してポーリングを行い、待機中のメッセージがあるかどうかを調べます。第2の方法ではリスニングを行ってアクティビティを探し、アクティビティがあった場合には起動(awake)状態を継続して、そのエンド・デバイス自体を宛先とするフレームを探します。

スリープ状態デバイスは、ビルドタイムに構成されます。

6.4.3.1 ポーリング・デバイス

スリープ状態エンド・デバイスがポーリング・デバイスとして構成されている場合は、そのデバイスがネットワークに参加(Join)する際に、アクセス・ポイントによってポーリング・デバイスとして認識されます。この時点で、アクセス・ポイントはこのデバイスをサポートするためのリソースを確保(予約)します。ネットワーク外のポートにあるスリープ状態デバイスに宛てたすべてのメッセージは、アクセス・ポイントによって保持されます。ブロードキャスト・メッセージは保持されません。

スリープ状態デバイスが起動して呼び出しを受信すると、呼び出しの結果ポーリング・メッセージがアクセス・ポイントに送信されます。このポーリング・メッセージはクエリ対象となるポートを指定し、管理ポートに送信されます。アクセス・ポイントはクエリ対象のポート上で最も古いフレームをポーリング・デバイスに宛てて送信します。何も保持されていなければ、アクセス・ポイントではクエリ対象のポートに対してペイロードのないフレームを送信します。

現時点の実装では、各ポートは個別にポーリングされる必要があります。また各ポートに対して、ポートにメッセージがなくなるまでポーリングを行う必要があります。

6.4.3.2 センシング・デバイス

TBD.

7. アプリケーション・プログラミング

この章では、SimpliciTIの機能を使用して顧客アプリケーションを適切に構築するために役立つ、SimpliciTIの実装に関する情報を提供します。

7.1 開発環境

SimpliciTIの開発には「IAR Embedded Workbench」を使用します。IDE(統合開発環境)依存のファイル(ターゲット用のデフォルト・リンカ・スクリプトなど)を除けば、ソース・コード内にはIDE依存コンストラクトまたはキーワードはありません。

SimpliciTIのソース・コードには、アセンブラ・ファイルは含まれていません。

7.2 ハードウェア抽象化

ディストリビューションでは、最小限のハードウェア抽象化しか提供しません。この最小限の抽象化(BSP)では、CC1100/CC2500の無線通信インターフェイスをサポートします。このインターフェイスにはSPIのサポートや、割り込みの生成に使用できる、無線(GDO0とGDO2)からの外部接続(最大2つ)が含まれています。

また、最大8つのLEDと8つのスイッチをターゲット・ボード上でサポートするための抽象化も含まれています。

UART、タイマ、LCD、およびその他のペリフェラルやオンチップ・リソースはサポートされていません。

これらを実装するかどうかは、完全に顧客側の判断に任されています。

7.3 MCUのリソース

コード空間と定数用のフラッシュ・メモリ、およびRAMの他には、SimpliciTIの実装で使用するリソースはありません。現時点では、SimpliciTIはタイマ・リソースや動的メモリ割り当てに依存しないため、ヒープ・メモリも使用されません。すべてのメモリ割り当ては静的であるか、自動変数として表されるため、メモリ割り当てではスタックを使用します。RAMを節約するために、可能な場合はconstメモリ型を使用します。

ランタイム・コンテクストは、MCUの不揮発メモリにも無線の不揮発メモリにも保存されません。

コードで処理される少数の例外を除いて、最初の無線ターゲット(CC1100/CC2500)はスリープ・モードでもランタイム・コンテクストを失いません。SRAMの大部分は維持され、維持されない重要な値も復元されます。

7.4 スレッド・モデル

全体として、SimpliciTIのAPIは顧客アプリケーションのスレッド・モデルと連動して実行される設計になっています。プロトコルの動作は、独自のコンテクストを必要とする独立したタスクとしては構成されていません。コードは既存のスレッド・コンテクストで実行されるため、スケジューラ等のOSは、実装には必要ありません。

SimpliciTIでは、受信されたフレームの汎用コールバック機能を提供します。7.6.1.を参照してください。このコールバックは受信ISRスレッドで実行されるため、コールバックの効率的な使用が促進されます。

7.4.1 ピア・ツー・ピア I/O

初期化後、ピア・アプリケーションはAPI内の連続呼び出しのペアを使用してリンクされます。

各々の側でこれらの呼び出しを行った結果、リンクIDが生成されます。これはハンドルであり、ソケットに似ていますが、バインディングが自動的に行われる点が異なります。リンクIDは、後でコネクションを参照するために使用されます。

リンキング·ステップの後に使用可能な機能は、リンクIDからの読み出しとリンクIDへの書き込みだけです。この機能では、リンクされたピア·アプリケーションからのメッセージを読み取り、リンクされたピア·アプリケーションへメッセージを送信します。基本的に、APIは読み取り(Receive(受信))呼び出しと書きこみ(Send(送信))呼び出しから構成されます。

7.4.1.1 入力/受信

スリープ状態にならないデバイスでは、無線によるフレームの受信はMCUへの割り込みが発生するきっかけとなります。フレームは無線の受信(Rx)FIFOから読み出され、MCU内のユーザー空間にある受信フレーム用のキューに蓄積(store)されます。フレームが受信された時点で入力フレーム·キューが満杯の場合は、キュー内の最も古いフレームが削除されます。その後、割り込みスレッドがリリースされます。これは比較的効率の良いプロセスです。

指定されたリンクIDでアプリケーション·レベルの読み取りが行われた場合は、そのリンクIDを待機しているフレームが入力フレーム·キュー内にあるかどうかが調べられます。待機中のフレームが見つかった場合、アプリケーションのペイロードは呼び出し側に返されます。そうでない場合、呼び出し側ではデータが存在しないという通知を受信します。基本的に読み取りとは、指定されたリンクIDを待機しているフレームを探すために入力フレーム·キューが行うポーリングです。ペイロードはFIFOの順序に従って返されます。

デバイスがスリープ状態あるいはポーリング中のデバイスである場合、アプリケーション層の読み取りではアクセス·ポイントに対するポーリング·クエリを開始します。これは呼び出し側には見えません。その後スレッドは保持され、アクセス·ポイントからの応答を待機します。アクセス·ポイントがフレームを転送した場合、フレームは転送後に呼び出し側に戻ります。アクセス·ポイントがペイロードのないフレームを送信した場合、そのフレームは単なる確認応答と解釈され、呼び出し側への戻りはペイロードなしで返るポーリングのように見えます。この状況ではアクセス·ポイントから応答があるまでスレッドが保持されます。そのため、スリープ状態以外の場合に行われる、入力キュー内のデータの有無を調べるだけの方式よりも効率は落ちます。

7.4.1.2 出力/送信

送信という状況は、フレームが無線通信で送信されるまでは返らない同期呼び出しとして実装されます。この設計では、フレームが送信される前に無線電力(radio power)を止めて、意図されない端末による送信シーケンスを防止します。

Tx(送信)スレッドが無線チャネルを利用してフレームを送信できない場合、呼び出し側ではその後戻りコードを受信して、後でリトライできます。チャネルへのアクセス権が即座に獲得できない場合には回復(リカバリ)がある程度試みられるため、ネットワーク層には多少の堅牢性(ロバストネス)がありますが、これは最小限のものです。SimpliciTIのNWKでは保証付きのデリバリをサポートしていないため、この状況をどう処理するかは顧客アプリケーションの判断に任されています(表2参照)。フレーム送信の重要性の度合は、アプリケーションのみが認識しています。消費電力と通信の信頼性のどちらを優先させるかは、顧客アプリケーション側で判断されます。

7.4.2 NWKアプリケーションのスレッド

NWKアプリケーションのスレッド(NWKアプリケーションのポートにサービスを提供するスレッド)は顧客アプリケーション·コンテクストの一部ではなく、またスケジューリング·サービスを使用しないため、個別に処理する必要があります。

Join(参加)、Link(リンク)、Mgmt(管理)の各ポーリング·フレームは、ターゲット·デバイスからの応答を要求します。アプリケーション層では、この要求により各フレームが確認応答済みフレームにされます。アプリケーションがタスク(Join(参加)やリンキングなど)を完了する機会を確実に作るために、アプリケーションでは確認応答が到着するのを待ちます。コールバックが使用されず、関連するスケジューラもないため、送信側アプリケーションでは応答が受信されるまで呼び出し側スレッドを保持します。したがって、送信側デバイスが参加、リンキング、ポーリングを完了するまでに「長い時間」(ミリ秒単位)がかかることもあります。

これらの同じアプリケーション用の受信デバイスも、ひとつのスレッドですべてを処理します。この場合、受信と確認応答(送信(Tx))は同一のスレッドで行われます。この場合のスレッドはISRスレッドであるため、上記のようなフレームの処理時に割り込みが「長時間」(ミリ秒単位)にわたって無効(disabled)になることもあります。

注意する必要があるのは、このようなスレッドは「長く」なる可能性があるものの、発生の頻度はそれほど高くないということです。参加はスタートアップ時のみに発生します。リンキングはいつでも発生する可能性がありますが、通常はスタートアップ時に発生します。ポーリングの発生頻度は前記2つより高くなる場合もありますが、それでも比較的低いほうです。

7.5 オブジェクト・モデル

SimpliciTIには正式なオブジェクト・モデルが存在しません。ピア・アプリケーションの特性を定義するための、プロファイル等の抽象化はありません。プロトコルの主な目的は、ピア・ツー・ピアのコネクションを設定して、各ピア間での無線によるメッセージングをサポートすることです。ピア間に設定されたアプリケーション層のプロトコルでのオブジェクト・モデルは「暗黙オブジェクト」であるため、顧客側で作成されます。

7.6 API

APIの目的は、アプリケーション側の手間をほとんどかけずに機能的で信頼性のあるネットワークを形成できるように、ネットワーク機能をカプセル化することです。このアプローチの主な影響は、結果として生成されるネットワークでは簡易性のために柔軟性が犠牲になるということです。

APIでは、次の機能をサポートします。

- 初期化
- リンキング
- アプリケーション・ピア・ツー・ピアのメッセージング
- デバイス管理

NWK層では、(すべてではないにしても)ほとんどの交換が確認応答を持っています。APP層では、確認応答ステータスがアプリケーション・ピア・ツー・ピアの規律によって制御されます。

7.6.1 smplStatus_t SMPL_Init (uint8 (*pCB) (linkID))

この呼び出しを使用すると、すべてのNWKの初期化が行われます。初期化のサイド・エフェクトのひとつは、ネットワークへのJoin(参加)の試みです。アクセス・ポイントのサポートするネットワークの場合は、Join(参加)リクエスト(サイレント発行)がアクセス・ポイントと引き換えにネットワーク・パラメータを獲得します。

またこの呼び出しは、11章で定義されるシステム・パラメータに基づいて、無線通信の設定やすべてのNWKコンストラクトの初期化を行います。デバイスがAPの場合は、暗号化鍵やLink(リンク)トークンの生成といった他の処理も行われます。

引き数は、リンクID引き数を取ってuint8を返す関数へのポインタです。この引数により、アプリケーションがコールバック関数ポインタを、受信されたフレームのロジックを処理するフレームに提供することが可能になります。次に示す条件が満たされれば、受信されたフレームのリンクIDとともにコールバックが引き数として呼び出されます。

1. 供給された関数ポインタがヌルではない。
2. 受信されたフレームの宛先ポートが、ユーザー側のポートである。
3. フレームの宛先アドレスが、デバイスの宛先アドレスである。
4. コネクションが有効である(つまり、ポートがコネクション・テーブルに記載されている)。

関数がゼロ以外の値を返すと、フレームの処理が完了したとみなされ、フレームのリソースが解放されて再利用できるようになります。それ以外の場合、フレームは入力フレーム・キューに残り、アプリケーションの処理を待機します。アプリケーションではどちらの場合にも、コールバック引き数に含まれるリンクIDをリンクID引き数としてSMPL_Receive()への呼び出しを実行し、受信フレームを検索します。このような条件でのこの呼び出しは、必ず成功するようになっています。

デバイスがアクセス・ポイントであり、そのAPがデータ・ハブとして設計されていれば、参加しているデバイスがエンド・デバイス・オブジェクトをサポートしている場合にもコールバックが呼び出されます。リンクIDでは0x00という値を持ちます。

この場合にはNWK層で受信フレームが処理されるため、アプリケーションで受信フレームを直接検索することはできません。コールバックは、参加デバイスから次のリンク・フレームを受信するためにSMPL_LinkListen()を実行する必要があることを、APへ伝えるための信号として使用されます。

コールバックの各条件が満たされれば、ハンドラ(処理プログラム)が受信ISRコンテクスト内で呼び出されることに注意してください。設計者は、このスレッドでの処理が多くなりすぎないように注意する必要があります。

7.6.2 smplStatus_t SMPL _Link (linkID_t *linkID)

別のリスン状態デバイス(サーバー)にクライアントとしてリンクします。

リスン状態のデバイスは、応答を返します。一度応答が受信されると、リンクIDとともに呼び出しが返ります。このリンクIDは、リンクの設定されたデバイスで送受信される、後続のすべてのメッセージングで使用される必要があります。これはブロッキング呼び出しではなく、リンクが成功したかどうかを示すステータスを返します。成功しなかった場合は、アプリケーションから再試行することが可能です。

7.6.3 smplStatus_t SMPL _LinkListen (linkID_t *linkID)

これはリンク呼び出しに伴うものであり、クライアントのリンク･メッセージを待機します。

この呼び出しの結果受信されるのは、最初の有効なリンク･メッセージのみです。また、任意の時間に実施できるのは、1つの未解決リスンのみです。リスンが実行される前に到着したリンク･メッセージは破棄されます。

リスン中に、重複したリンク･フレームを受信しないようにするための方策はあります。この方策は、再送信されたフレームや送信の遅れたフレームを検知する仕組みになっています。ただし、リセットされたデバイスが毎回同じリンク･フレームを生成する場合には、リセットされたデバイスから来るリンク･フレームも検知します。例えば、毎回ランダムに新規デバイス･アドレスを生成するデバイスは、重複したリンキング･デバイスとしては検出されず、リソースを消費します。

これはブロッキング･コールであり、有効なリンク･フレームが受信されるまでは返りません。

7.6.4 smplStatus_t SMPL _Send (linkID_t lid, uint8 *msg, uint8 len)

この呼び出しは、長さlenのメッセージmsgを、リンクIDlidとともにデバイスへ送信します。未解決になる送信呼び出しは、一度にひとつだけしか存在できません。lidは、SMPL_Link()呼び出しまたはSMPL_LinkListen()呼び出しのうち、デバイス上で実行された方によって返されます。

この呼び出しは、メッセージが送信されたあとで返ります。これは、MCUと無線の送信コンテクストと電源コンテクストに関して規律（discipline）が保たれるようにするためです。

メッセージ･サイズは最大50バイトに制限されます。セグメンテーションとリアセンブリが必要な場合は、APPの担当となります。

7.6.5 smplStatus_t SMPL _Receive (linkID_t lid, uint8 *msg, uint8 *len)

この呼び出しは、リンクIDlidからのメッセージをチェックします。メッセージがある場合は、メッセージとその長さを返します。ポーリング･モードでのみ動作するため、非ブロッキングとなります。呼び出し側（caller）は、メッセージをコピーするためのバッファのアドレスを供給する必要があります。

所要の受信メッセージのリンクIDは、クライアントの呼び出しによって供給されます。これにより、複数リンクのサポートが可能になります。

7.6.6 void SMPL_Ioctl(ioctlObject_t object, ioctlAction_t action, void *val)

これは、APPがランタイムでNWKを構成できるようにする手段です。現在使用可能な値については、10章で説明しています。

7.6.7 void BSP_Init(void)

この初期化呼び出しは、厳密にはSimpliciTI APIではありません。SimpliciTIが実行されている個々のボードに対応するハードウェア抽象化の初期化に使用されます。無線のSPIインターフェイスを初期化し、ターゲット･ボード上に存在する可能性のあるLEDとスイッチに基本的なサポートを提供します。

この呼び出しは、SimpliciTIの初期化呼び出しの前に行われる必要があります。SimpliciTIの初期化呼び出しは、この呼び出しによって準備されたSPIインターフェイスを使用するためです。

7.6.8 疑似コードの例

次に示す疑似コードの抜粋は、デバイスがどのようにしてAPのあるSimpliciTIネットワークに参加（Join）し、2つの他のデバイスにリンクして、様々なメッセージを送信するかについての、APIのシーケンスを示しています。

```
void main()
{
  linkID_t linkIDLow, linkIDHigh;
  uint32   temp;

  // Initialize the board's HW
  BSP_Init();

  // Initialize SimpliciTI. The initialization will cause the
  // device to Join as a side effect. The Join will return
  // security key and a token to be used in linking
  // sequences for this network. This is all hidden from
  // the application.

  SMPL_Init(0);  // no callback supplied

  // Establish links to two peers. They may or may not be
  // the same application. If they are not they may or may
  // not be on the same device. One will get a message
  // if the sampled temperature is to low. The other gets
  // a message if it is too high. The listening device(s) must
  // associate these calls correctly. The listeners need
  // to know that the first link message is associated with
  // messages when the temperature is low and the second
  // when it is high.
  SMPL_Link(&linkIDLow);
  SMPL_Link(&linkIDHigh);
  while (TRUE)
  {
    // put board to sleep until timer wakes it up
    // to read the temperature sensor
    MCU_Sleep();
    HW_ReadTempSensor(&temp);
    if (temp > TOO_HIGH)
    {
      SMPL_Send(linkIDHigh, "Hot!", 4);
    }
    if (temp < TOO_LOW)
    {
      SMPL_Send(linkIDLow, "Cold!", 5);
    }
  }
}
```

このデバイスがリンクしている時は、ピア·アプリケーションではSMPL_LinkListen()を最初に実行済みである必要があります。これが、リンクされたペアを認識する方法です。2つのリンクが同じ物理デバイス上にある必要はありません。

アプリケーションが応答を実装している場合は、各SMPL_Send()ステートメントの後にSMPL_Receive(リンクID…)が続くことになります。それが意図されている場合には、受信のアクティビティをタイムアウト規律(discipline)で調節して、メッセージを再送信する必要があるかどうかを知る必要があるかもしれません。ただしこの例では、行方不明のメッセージが重要なものでないかぎりは、単なる応答の返されないメッセージになる可能性の方が高いと思われます。

次の章からは、SimpliciTIセッションのサンプルを紹介します。上記の疑似コードを具体的に再現したものではなく、SimpliciTIのピア同士がどのようにリンクされるかについての一般的な意味を伝えることが意図されています。アクセス·ポイントが置かれるかどうかはオプションです。アクセス·ポイントが存在しない場合は、2つのエンド·デバイスのデフォルト·リンク·トークンが一致している必要があります。そうでない場合は、応答しないという単純な方法によって、リスナーがリンク·メッセージを拒否します。

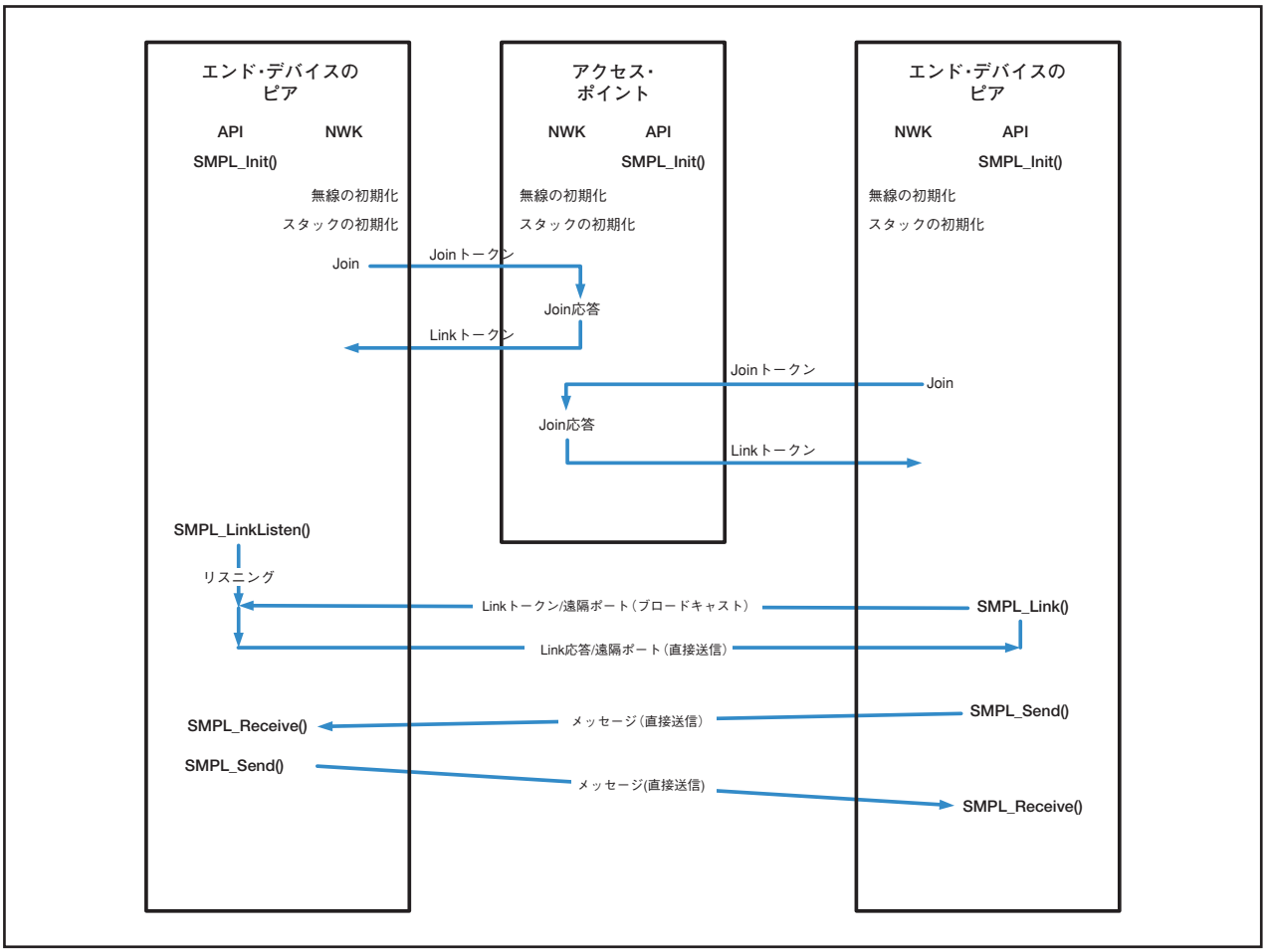

図 2. SimpliciTI のピア･ツー･ピアのセッション例

8. ネットワーク･アクセスのコントロール

具体的なネットワークへのアクセスを制御する手段は様々です。悪意のあるなしにかかわらず、不正なデバイスからネットワークを保護する必要があると思われます。

様々な形態のアクセス･コントロールを可能にするメカニズムが、SimpliciTIには多数組み込まれています。これらのうち2つでは、アクセス･ポイントが構成の一部である場合に可能な追加コントロールを利用しています。次の章では、これらのメカニズムの各々について説明します。

8.1 Joinトークン

アクセス･ポイントが構成の一部である場合は、アクセス･ポイントがJoin（参加）NWKアプリケーションをサポートします。このアプリケーションの主な目的は、ネットワークに参加している他のデバイスに、そのネットワークのパラメータについてのヒントを提供することです。これは、Link（リンク）トークンや暗号化鍵などです。正しい方法で（つまり、Linkトークンと鍵を知った上で）ネットワークに参加することで、不正デバイスがネットワーク上で通信を行うことを不可能にしようという考え方です。他のネットワーク･パラメータを見つけるために、デバイス側では具体的なネットワークのJoin（参加）トークンを知る必要があります。

顧客側ではネットワークを構成して、様々なJoin（参加）トークン、また様々な製品やアプリケーション･タイプ用の様々なトークンを持つことができます。Join（参加）応答も同様に修正できます。例えば、様々なLink（リンク）トークンを様々なネットワーク･デバイスに割り当てることも、アクセス･ポイントでは可能です。Join（参加）NWKアプリケーションのペイロードを修正して、デフォルトよりも豊富で多様な情報を提供するようにもできます。

8.2 アクセス・ポイントのJoin（参加）コンテクスト

アクセス･ポイントはJoin（参加）コンテクストを能動的に呼び出すことも必要になる可能性があります。例えば、互いに関連のない2つのアクセス･ポイントがJoin（参加）デバイスに近接しているような場合（例: 隣接する2つの家屋）には、参加可能なネットワークに制限を設けるために追加のコントロールが必要になることも考えられます。

デフォルトでは、アクセス･ポイントは常に、正しいJoin（参加）トークンを持つJoin（参加）フレームを受け入れます。顧客アプリケーションでは、アクセスの追加コントロールとして、アクセス･ポイントのJoin（参加）コンテクストを設定できます。つまりこのJoin（参加）コンテクストは、Join（参加）フレームを積極的に許可するように設定されていないかぎり、Join（参加）フレームを拒否するように設定できます。これは、SMPL_Ioctl()のAPIを使用して行われます。

8.3 Link（リンク）トークン

アクセス･ポイントが構成の一部である場合は、Join（参加）NWKアプリケーションをサポートします。このアプリケーションの主な目的は、ネットワークに参加（Join）している残存デバイスに、そのネットワークのパラメータについてのヒントを提供することです。これらのネットワーク･パラメータのひとつがLink（リンク）トークンです。これで、各ネットワークについてLink（リンク）トークンが一意であるか、少なくとも制御されているということになります。連続してリンクしている2つのデバイスでは、正しいトークンを使用する必要があります。そうでない場合はリンクできません。

Link（リンク）トークン･スキームを使用すると、不要なデバイスを排除するために必要なセキュリティが2つのデバイスの物理的なコロケーションによって提供される、明示的な物理バインディングに替わるものが提供されます。

8.4 暗号化

暗号化は、ネットワーク･アクセスのコントロールに使用できます。現時点ではまだ実装されていませんが、暗号化を行うことにより、ネットワーク･リソースに対するデバイスの恣意的なアクセスを防止し、ネットワーク･アクセスのコントロールをさらに強化することが検討されています。

9. Tx（送信）専用デバイス

Tx（送信）専用デバイスは、この章で説明するような特殊な動作を示します。顧客側でこの動作を修正することもできますが、デフォルトの動作でも機能的に問題はないと思われます。

9.1 参加

Tx（送信）専用デバイスがネットワークに参加（Join）することに、機能的な理由はありません。Link（リンク）トークン等のネットワーク･パラメータを取得するための応答を受信できないため、またSimpliciTIプロトコルでは経路指定（routing）をサポートしないため、Tx（送信）専用デバイスには自分の存在をアナウンスする必要がないのです。ただし、Tx（送信）専用デバイスがJoin（参加）フレームを送信しても害にはなりません。

9.2 リンキング

Tx（送信）専用デバイスについての基本的な問題は、Tx（送信）専用デバイスでは応答を受信できないため、ネットワークのLink（リンク）トークンを取得する方法を持たないということです。またTx（送信）専用デバイスには、ピア･デバイス上で送信対象となる一意のポートについての情報も伝えられません。このポート割り当てはリンク応答に載せて、受信の可能なリンキング･デバイスに返されます。これに対応するために、Tx（送信）専用デバイスはリンキングの後でデフォルトのTx（送信）専用ポートに対して送信を行います。それでもリンク･シーケンスは、リンク先のデバイス上にコンテクストを設定するために必要な過程です。

TxデバイスではデフォルトのLink（リンク）トークンを使用します。Txデバイスの構築には、顧客側の裁量で様々なデフォルトのLink（リンク）トークンを使用できますが、トークンとはネットワーク･アクセスの制御を補助するための手段に過ぎません。常にデフォルトを使用すれば、このことは無視できます。

受信が可能なデバイスでは、ピアのアドレスをリンク応答から収集します。顧客側ハードウェアがアドレスを指定しないかぎり、Tx（送信）専用デバイスではリンク先のデバイスに送信されたフレームをブロードキャストする必要があります。これは、Tx（送信）専用デバイスにリンクされたデバイスはプロミスキュアス･モードで動作する必要があるということです。

複数のTx（送信）専用デバイスがある場合、受信側のデバイスでは各Tx（送信）専用デバイスの一意性を保つ責任を負います。このことが可能なのは、発信元アドレスが一意であること、またNWK層でプロクシ･マッピングを行って、個々の遠隔Tx（送信）専用デバイス用のローカル受信（Rx）ポートを一意に保つことが可能であることによります。このマッピングはNWK層で行われます。

個々のTx（送信）デバイスは、他の明確なデバイスへのリンクをひとつだけ持つことが可能です。単体のデバイスへの複数の論理リンクは、Tx（送信）専用デバイス用にはサポートされません。複数の論理リンクが必要な場合は、アプリケーション層で多重化（multiplexing）を行う必要があります。

Tx（送信）専用デバイスをリンクする場合、返されたリンクIDはSimpliciTI環境の一部として設定された固定値です。これはアプリケーションにとって重要な問題とはなりません。APIの使用も同様です。RXタイプのビルドタイムでのオプション設定により、特殊な処理コンテクストが設定されます。(8章参照)

10. SimpliciTIのIoctl インターフェイス

Ioctlインターフェイスでは、アプリケーション層がSimpliciTI環境の下位のサポート層にアクセスすることを許可します。これからの記述は、現時点でのサポートをある程度詳細に説明したものです。各ioctlオブジェクトは、使用可能なアクションとともに気記載されています。各オブジェクト/アクションのペアをサポートする値も記載されています。

タイプ(types)となる値は、nwk_types.hで定義されています。

10.1 ロー I/O

このオブジェクトでは、様々な宛先のアドレス/ポートの組み合わせ間での送信および受信を許可します。通常、アプリケーションではリンキング･スキームを使用してピア･コネクションを確立している必要があります。このオブジェクトでは、無条件の通信を許可します。このサポートは、NWK層自体によって広範囲で使用されます。

このインターフェイスでは、アプリケーションからリンクIDを供給するのではなく、呼び出し側からポート番号を供給する必要があることに注意してください。

10.2 無線コントロール

現時点では、いくつかの簡易無線コントロール機能が使用可能です。現在、このインターフェイスでは無線構成レジスタへの直接アクセスをサポートしていませんが、もちろんサポートするように修正することも可能です。

10.3 アクセス･ポイントJoin(参加)コントロール

デバイスがSimpliciTIネットワークにアクセスする能力にいくらかのコントロールを追加するために、プロトコルではトークンを使用して、ネットワークへの参加(Join)とリンキングによるピア作成の両方を行います。(8章参照)

追加のコントロールは、コンテクストがJoinフレーム処理を許可するように設定されていないかぎり、アクセス･ポイントがJoinフレームの処理を行わないことを許可することで提供されます。つまり、Join(参加)できない場合、デバイスはネットワークの正しいLink(リンク)トークンが獲得できないために、他のどのデバイスにもリンクできないことになります。

オブジェクト	アクション	値	コメント
IOCTL_OBJ_RAW_IO	IOCTL_ACT_READ	ioctlRawReceive_t	実行されると、指定されたポート上の最も古いフレームのペイロードを返す。SMPL_Receive()呼び出しに似ているが、受信されたフレームから追加情報が入手可能という点が異なる。
	IOCTL_ACT_WRITE	ioctlRawSend_t	同封されたペイロードを指定されたアドレス/ポートの組み合わせに送信する。

表 3. ローI/Oの ioctl

オブジェクト	アクション	値	コメント
IOCTL_OBJ_RADIO	IOCTL_ACT_RADIO_SLEEP	NULL	MCUをスリープ状態にする前に実行される。無線に対して規律通りのステート変更を行う。必要な無線レジスタをすべて保存する。
	IOCTL_ACT_RADIO_AWAKE	NULL	MCUが復帰した後に実行される。必要な無線レジスタをすべて回復する。
	IOCTL_ACT_RADIO_SIGINFO	ioctlRadioSiginfo_t	指定されたポート上で最後に受信されたフレームの信号強度情報を取得する。

表 4. 無線コントロールの ioctl

オブジェクト	アクション	値	コメント
IOCTL_OBJ_AP_JOIN	IOCTL_ACT_ON	NULL	Joinフレームの処理を許可する。
	IOCTL_ACT_OFF	NULL	Joinフレームを無視する。

表 5. アクセス･ポイントのJoin（参加）コントロールのioctl

オブジェクト	アクション	値	コメント
IOCTL_OBJ_ADDR	IOCTL_ACT_SET	addr_t *	ポイントされた値のアドレスを設定する。
	IOCTL_ACT_GET	addr_t *	ポイントされたアドレスにアドレスを返す。

表 6. デバイス･アドレスのコントロールのioctl

10.4　デバイス･アクセス･コントロール

このインターフェイスでは、ビルドタイム･デバイスのアドレス設定を上書きする許可をアプリケーションに与えます。アプリケーションがランタイムにデバイス･アドレスを生成する場合は、アドレスの設定にこのインターフェイスが使用されます。アドレスの設定は、SMPL_Init()に対する呼び出しが発生する前に行う必要があります。そうでない場合は、ビルドタイム･アドレスが優先されます。この2つの条件（前初期化ioctl呼び出しを行うか、SMPL_Init()を介するか）のどちらが当てはまる場合でも、一度アドレスを設定すると、後から変更することはできません。

10.5　暗号化鍵と周波数のコントロール

暗号化鍵と無線周波数を処理するためのコントロールが予定されています。ただしこれらはまだ実装されていません。

11.　システム構成

参考文献[1]では、SimpliciTIで必要とされる、ビルドタイムに構成可能な値をまとめています。この章では、新たな説明を加えて再度それらの値を紹介します。

SimpliciTIのデバイスを構築する際に使用される構成ファイルは2種類あります。一方のファイルは一般ネットワーク用であり、特にそのネットワーク用に構築されるすべてのデバイスのマクロ値を供給します。もう一方のファイルはデバイス固有です。それぞれのファイル･タイプを次の章で説明します。

11.1　一般ネットワーク用の構成

ファイルsmpl_nwk_config.datには、次のようなネットワーク規模のマクロの定義が入っています。

マクロ	デフォルト値	コメント
MAX_HOPS	3	フレームの再送は最大3回まで。
MAX_HOPS_FROM_AP	1	APからの最大距離。ポーリングおよびポーリング応答によって、ネットワークの再送トラフィックを制限するために使用される。
MAX_APP_PAYLOAD	10	絶対最大定格値は50。
DEFAULT_JOIN_TOKEN	0x01020304	アクセス･セキュリティを提供するには、これを変更すること。
DEFAULT_LINK_TOKEN	0x05060708	APネットワーク上のAPによって変更される必要がある。顧客側では nwk_join.c:generateLinkToken()を変更する必要がある。

表 7. 一般ネットワーク用構成のマクロ

11.2 デバイス固有の構成

マクロ	デフォルト値	コメント
NUM_CONNECTIONS	4	サポートされるコネクションの数。各コネクションでは、双方向通信をサポートする。 エンド・デバイスのホスティングを行っていない場合、アクセス・ポイントとレンジ・エクステンダはこれを0に設定できる。
SIZE_INFRAME_Q	2	エンド・デバイスひとつにはおそらく2で十分。REとAP用にはもう少し大きな数値が必要。蓄積転送クライアントをサポートしている場合、転送されたメッセージはここで保持されるため、APではさらに大きな入力フレーム・キューが必要になる。
SIZE_OUTFRAME_Q	2	Tx（送信）が同期的に行われるため、出力フレーム・キューは小さくてもよい。アクセス・ポイントのデバイスがスリープ状態ピアに送信するエンド・デバイスのホスティングも行っている場合、待機中フレームはここで保持されるため、出力キューをより大きくする必要がある。そうでない場合には、実際には1で十分と思われる。
THIS_DEVICE_ADDRESS	{0x78, 0x56,0x34, 0x12}	このデバイスのアドレス。最初のバイトは、CC1100/CC2500での無線通信のフィルタとして使用されるため、最初のバイトを0x00または0xFFにすることは絶対にできない。また、エンド・デバイス上でのこれらの無線通信に関しては、フィルタリングに最大限の効果を持たせるために最初のバイトはLSB（least significant byte）である必要がある。 それ以外の場合は、デバイス用のものでないと認識される前に、フレームをMCUで処理する必要がある。フィルタリングが行われないように、APとREはプロミスキュアス・モードで動作する。 このマクロでは、長さNET_ADDR_SIZE（nwk_types.hにある）である符合なし文字列の静的なconst配列を初期化する。
レンジ・エクステンダ		
RANGE_EXTENDER		レンジ・エクステンダのデバイス型宣言。
エンド・デバイス		
END_DEVICE		エンド・デバイスのデバイス型宣言。
RX_LISTENS RX_POLLS RX_NEVER RX_ALWAYS	RX_ALWAYS	1つだけ選択すること。RX_NEVERはTx（送信）専用デバイス。RX_LISTENSとRX_NEVERはまだサポートされていない。
アクセス・ポイント		
ACCESS_POINT		アクセス・ポイントのデバイス型宣言。
NUM_STORE_AND_FWD_CLIENTS	3	サポートされている蓄積転送クライアントの数。
AP_IS_DATA_HUB	Not defined	このマクロを定義すると、APでは自動的に、エンド・デバイス・オブジェクトに参加してサポートしている、個別のデバイスからのリンク・メッセージをリスンする。参加しているEDは、Join応答を受信した直後にリンクする必要がある。

表 8. デバイス固有構成のマクロ

12. 一般的な情報と警告

以下、各情報の順序に特別な意味はありません。

1. SimiplicTIでは、構成ツールSmartRF Studioからエクスポートされたレジスタ設定値を使用します。このツールでは、制御レジスタ設定用の値と同様に、CC25xxとCC11xx両方の無線RF設定のレジスタ値もエクスポートします。SimpliciTIでは無線RF設定の値を使用しますが、制御レジスタ設定値の大部分を無視する必要があります。制御設定値は変更すると大きな影響のある機能性を持つため、上書きすることはできません。次に示す表に、ソフトウェアに直接制御されるレジスタを記載します。

レジスタ	コメント
IOCFG0	汎用IO構成。ソフトウェアのみで制御される。
IOCFG2	汎用IO構成。ソフトウェアのみで制御される。
MCSM0	ステート・マシン構成。PO_TIMEOUTの値はSmartRF Studioの出力から抽出される。それ以外のすべてのフィールドはソフトウェアで制御される。
MCSM1	ステート・マシン構成。ソフトウェアのみで制御される。
PKTLEN	パケット長。 ソフトウェアのみで制御されます。
PKTCTRL0	パケット制御レジスタ。WHITE_DATAの値はSmartRF Studioの出力から抽出される。それ以外のすべてのフィールドはソフトウェアで制御される。
PATABLE0	SmartRF Studioでは現在この値をエクスポートしていない。今後の改訂版でこの値をエクスポートする場合は、内蔵ソフトウェアのデフォルト値の代わりにこの値が使用される。
CHANNR	チャネル数。SmartRF Studioによりエクスポートされた値を使用する。 ただし、MRFI_CHANをプロジェクト・レベルで「所要のチャネル数に等しい」と定義すれば、この値を上書きすることが可能。

表 9. CC1100/CC2500のレジスタ設定の例外

2. 受信されたフレームの到着順は元のまま保たれます。フレーム·キューが満杯のところへ新しいフレームが到着した場合は、最も古いフレームを削除して空きを作ります。
3. アプリケーションがSMPL_Receive()を行った場合は、指定されたリンクID用の、使用可能な最も古いフレームからペイロードを受信します。
4. 蓄積転送クライアントにフレームを転送する場合、フレームは受信された順番通りに転送されます。
5. ブロードキャストされたフレームとNWKアプリケーション·フレームは、蓄積転送クライアント用には保存されません。
6. ポーリング デバイスに送信を行うエンド·デバイスをアクセス·ポイント自体がホスティングした場合に、ある問題が起こることが知られています。これらのフレームは使用可能にされた順番どおりではなく、他のデバイスから転送されてきたすべてのフレームの後で送信されることになります。
7. アクセス·ポイントでは、参加(Join)したデバイスのアドレスを記憶しません。つまり、アクセス·ポイントでは発信元に関係なく、認識したフレームを再送します(レンジ·エクステンダについても同様です)。不正なメッセージを無視するかどうかは、ピア·アプリケーションの判断に任されます。Join(参加)トークンとLink(リンク)トークンを使用すると、認証されていないデバイスへのコネクションを防止するのに役立つのでメンバ以外のデバイスからの干渉を緩和できますが、必ずそうなるという保証はありません。
8. ユーザー·アプリケーションへのフレームは、宛先ポートと、コネクション·テーブルに入力された発信元アドレスを照合することによって、NWK層で部分的に認証されます。受信されたフレームは、この評価に合格しなければ破棄されます。それ以外には、入力フレーム·キューからこのフレームを除去する方法はありません。これらのフレームを検索しようとするアプリケーションがまずないためです。
9. Tx(送信)専用デバイスのサポートは、完全ではありません。RX_NEVER Rxデバイス·タイプはエンド·デバイス用には使用しないことを推奨します。

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJないしTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIJとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIJの標準販売契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIJとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTI からライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、かつその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、かつ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

TIは、TIの製品が、安全でないことが致命的となる用途ないしアプリケーション（例えば、生命維持装置のように、TI製品に不良があった場合に、その不良により相当な確率で死傷等の重篤な事故が発生するようなもの）に使用されることを認めておりません。但し、お客様とTIの双方の権限有る役員が書面でそのような使用について明確に合意した場合は除きます。たとえTIがアプリケーションに関連した情報やサポートを提供したとしても、お客様は、そのようなアプリケーションの安全面及び規制面から見た諸問題を解決するために必要とされる専門的知識及び技術を持ち、かつ、お客様の製品について、またTI製品をそのような安全でないことが致命的となる用途に使用することについて、お客様が全ての法的責任、規制を遵守する責任、及び安全に関する要求事項を満足させる責任を負っていることを認め、かつそのことに同意します。さらに、もし万一、TIの製品がそのような安全でないことが致命的となる用途に使用されたことによって損害が発生し、TIないしその代表者がその損害を賠償した場合は、お客様がTIないしその代表者にその全額の補償をするものとします。

TI製品は、軍事的用途もしくは宇宙航空アプリケーションないし軍事的環境、航空宇宙環境にて使用されるようには設計もされていませんし、使用されることを意図されておりません。但し、当該TI製品が、軍需対応グレード品、若しくは「強化プラスティック」製品としてTIが特別に指定した製品である場合は除きます。TIが軍需対応グレード品として指定した製品のみが軍需品の仕様書に合致いたします。お客様は、TIが軍需対応グレード品として指定していない製品を、軍事的用途もしくは軍事的環境下で使用することは、もっぱらお客様の危険負担においてなされるということ、及び、お客様がもっぱら責任をもって、そのような使用に関して必要とされる全ての法的要求事項及び規制上の要求事項を満足させなければならないことを認め、かつ同意します。

TI製品は、自動車用アプリケーションないし自動車の環境において使用されるようには設計されていませんし、また使用されることを意図されておりません。但し、TIがISO/TS 16949の要求事項を満たしていると特別に指定したTI製品は除きます。お客様は、お客様が当該TI指定品以外のTI製品を自動車用アプリケーションに使用しても、TIは当該要求事項を満たしていなかったことについて、いかなる責任も負わないことを認め、かつ同意します。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上