CYW20819/CYW20820 機能およびペリフェラルガイド

本書について

適用範囲と目的

このドキュメントでは、ModusToolbox™ソフトウェアを使用してアプリケーションを作成する開発ツールとともに CYW20819/CYW20820 デバイスアーキテクチャを探索することにより、Arm® Cortex®-M4 CPU を備えた CYW20819/CYW20820 超低消費電力デュアルモード Bluetooth® v5.2 ワイヤレス MCU を紹介します。これらのアプリケーションの開発を支援するために、このテクニカルガイドでは、デバイスのビルディングブロックとアプリケーションの開発中に役立つツールの使用に焦点を当てています。このテクニカルガイドは、CYW20819/CYW20820 の機能とペリフェラルの概要を説明することを目的とします。

目次

本書について **1**
目次 **1**
1 はじめに **3**
2 リソース **4**
2.1 ハードウェア EVB 4
3 CYW20819/CYW20820 Bluetooth® SoC **5**
4 CYW20819/CYW20820 SoC ペリフェラル **6**
4.1 汎用入出力 (GPIO) 6
4.2 アナログ-デジタルコンバーター (ADC) 7
4.2.1 DC 測定モード 7
4.2.1.1 ADC 出力データとクロック 8
4.2.1.2 DC 測定モードでの ADC 入力 MUX クロッキング 8
4.2.1.3 ADC サンプリングとゲイン計算 8
4.3 クロック 9
4.3.1 高周波数クロックソース 9
4.3.2 低周波数クロックソース 9
4.4 リアルタイムクロック (RTC) 10
4.5 ウォッチドッグタイマー (WDT) 11
4.6 アプリケーションタイマー 11
4.7 パルス幅変調器 (PWM) 12
4.8 乱数発生器 (RNG) 13
4.9 Inter-Integrated Circuit (I^2C) 13
4.9.1 I2C クロック 13
4.9.2 I2C 初期化 14
4.9.3 I2C 動作 14
4.9.3.1 I2C マスター書き込み 14
4.9.3.2 I2C マスター読み出し 14
4.9.3.3 I2C マスターの書き込みと読み出しの組合せ 14
4.10 シリアルペリフェラルインターフェース (SPI) 14
4.10.1 SPI マスターTX のみの動作 15
4.10.2 SPI マスターRX のみの動作 15
4.10.3 SPI マスター全二重動作 16

目次

4.10.4 SPI スレーブ TX のみの動作 16
4.10.5 SPI スレーブ RX のみの動作 16
4.10.6 SPI スレーブ全二重動作 16
4.11 UART 17
4.11.1 ペリフェラル UART (PUART) 17
4.11.1.1 PUART 初期化 17
4.11.1.2 PUART ボーレート 17
4.11.1.3 PUART 送信 17
4.11.1.4 PUART 受信 17
4.11.1.5 PUART フロー制御 18
4.11.2 ホストコントローラーインターフェース UART 18
4.12 キーボードスキャナー (Keyscan) 18
5 Bluetooth® Low Energy 無線の機能 20
5.1 TX 電力制御 20
5.2 Co-existence - SECI 20
6 ソフトウェア 21
6.1 アプリケーションコード 21
6.2 低電力機能 22
6.3 ファームウェアのアーキテクチャ 22
6.4 メモリレイアウト 23
6.4.1 *.btp ファイルの重要性 24
6.5 プログラム 24
6.5.1 ファイルフォーマット 24
6.5.1.1 *.hcd 形式の詳細 25
6.5.1.2 *.hex 形式の詳細 26
6.5.1.3 *.cgs 形式の詳細 26
6.5.2 ビルド中のファイル生成 26
6.5.3 リカバリプロセス 27
6.5.4 ミニドライバー 27
6.6 プログラミング用ツール 27
6.6.1 ClientControl 27
6.6.2 ChipLoad 28
6.6.3 DetectAndId 28
6.6.4 インフィニオンプログラマ 28
7 まとめ 29
8 関連資料 30
改訂履歴 31
免責事項 32

1 はじめに

CYW20819/CYW20820 は、Bluetooth® Basic Rate (BR), Enhanced Data Rate (EDR), および Bluetooth® Low Energy (BLE) をサポートする Bluetooth® 5.2 準拠の SoC です。CYW20819/CYW20820 は、Bluetooth®コア仕様 v4.2 に基づくすべてのオプションの LE 機能と、仕様 v5.0 に基づく LE 2 Mbps 機能に対応します。最先端の 40 nm CMOS 低消費電力プロセスを使用して製造された CYW20819/CYW20820 は、最高レベルの統合を採用して、すべての今まで外部コンポーネントとして重要だったものを削除できます。これによってデバイスのフットプリントと Bluetooth®ソリューションの実装に関連するコストを最小限に抑えます。

CYW20819/CYW20820 は、96 MHz の Arm® Cortex®-M4 CPU を統合し、優れた処理能力を提供しながら、より小さなフットプリントを維持します。CYW20819/CYW20820 は、Bluetooth®アプリケーション用の 176 KB の RAM, 8 KB のキャッシュ, 256 KB のオンチップセキュアフラッシュ, および低レベル Bluetooth®スタックとドライバー専用の 1 MB の ROM スペースを提供します。CYW20819/CYW20820 は、ADC, PWM, I^2C, SPI, UART およびキーボードスキャナーなどのいくつかのハードウェア周辺ブロックもホストし、他のコンポーネントとのインターフェースを支援します。このガイドは、CYW20819/CYW20820「デバイスの機能と SoC ペリフェラルの使用」を探索するために役立ちます。

2 リソース

https://www.infineon.com で豊富なデータを提供し、適切なデバイスを選択し、デバイスを設計に迅速かつ効果的に統合するために役立ちます。以下は、このアプリケーションノートに関連するリソースの簡略リストです。

- 概要: **AIROC™ Bluetooth® & Multiprotocol** ポートフォリオは、Bluetooth® Low Energy のみ、および Bluetooth® Classic と Bluetooth® LE をサポートするデュアルモード Bluetooth® ソリューションで構成され、アプリケーションに最も信頼性が高く、最高のパフォーマンスを発揮する接続を提供します。
- **製品セレクター:** ワイヤレス製品セレクター
- **製品ドキュメント**
  - **CYW20819 データシート**: CYW20819 Bluetooth® SoC の電気的仕様を説明および提供します。
  - **CYW20820 データシート**: CYW20820 Bluetooth® SoC の電気的仕様を説明および提供します。CYW20820 には、10 dBm の送信電力で送信できるパワーアンプが統合されています。
  - **アプリケーションノート**: 基本的なレベルから高度なレベルまで、幅広いトピックをカバーします。

これらは、それぞれの **CYW20819** および **CYW20820** 製品ページにあります。

- **IDE: ModusToolbox™ IDE for Bluetooth®-SDK**
- **サンプルコード:** GitHub リポジトリで入手可能な **Bluetooth®-SDK サンプルコード**。
- **Bluetooth®-SDK ドキュメント:** ファームウェア開発, デバッグガイド, テストツール, ライブラリガイド, API ドキュメントを含む最新の Bluetooth® SDK ドキュメント。
- **開発キット:** いくつかの例は次のとおりです。
  - **CYW920819M2EVB-01 evaluation kit**
  - **CYW920820M2EVB-01 evaluation kit**

2.1 ハードウェア EVB

CYW20819/CYW20820 デバイスの機能および SoC ペリフェラルの使用のために 2 つの評価キットが利用可能です。以下を使用できます。

- **CYW920819M2EVB-01** Evaluation Kit は、超低消費電力のデュアルモード Bluetooth® 5.2 ワイヤレス MCU デバイスである CYW20819 を使用してシングルチップ Bluetooth®アプリケーションを評価および開発します。
- **CYW920820M2EVB-01** Evaluation Kit は、超低消費電力のデュアルモード Bluetooth® 5.2 ワイヤレス MCU デバイスである CYW20820 を使用してシングルチップ Bluetooth®アプリケーションを評価および開発します。CYW20820 には、10 dBm の送信電力で送信できるパワーアンプが統合されています。

3 CYW20819/CYW20820 Bluetooth® SoC

Figure 1 に、CYW20819/CYW20820 デバイスのブロックダイヤグラムを示します。CYW20819/CYW20820 SoC 機能の詳細については、**CYW20819 データシート**と **CYW20820 データシート**を参照してください。



Figure 1 CYW20819/CYW20820 Bluetooth® SoC のブロックダイヤグラム

4 CYW20819/CYW20820 SoC ペリフェラル

CYW20819/CYW20820 で ADC や I2C などのいくつかの SoC ペリフェラルを利用するために、Bluetooth® SDK で Hardware Abstraction Layer (HAL) ドライバーを利用できます。Bluetooth® SDK の HAL ドライバーはブロッキング関数です。Bluetooth® SDK の API 実装の詳細については、wiced_btsdk の *README.md* ファイルの **API Documentation** リンクをクリックするか、**Github ドキュメント**にアクセスしてください。一部の SoC ペリフェラルには、ハードウェアブロックの機能を示す専用のサンプルコードがあり、Bluetooth® SDK で使用できます。**GitHub repository** でより多くのサンプルコードを入手できます。

4.1 汎用入出力 (GPIO)

GPIO は、ユーザーによる実行時の設定に応じて入力または出力として機能できるデジタル信号ピンです。Bluetooth® SDK は、*wiced_hal_gpio.h* で GPIO ドライバーを提供します。CYW20819/CYW20820 は、LHL GPIO (最大 40) と Arm® GPIO (最大 10) の 2 種類の GPIO をサポートしています。デバイスで使用可能な LHL GPIO と Arm® GPIO の数は、パッケージによって異なります。

- CYW20819A1KFBG (62-ball FBGA) デバイスパッケージには、22 個の LHL GPIO と 2 つの Arm® GPIO (DEV_WAKE と HOST_WAKE) があります。LHL と Arm® GPIO の両方が `wiced_hal_gpio.h` の `wiced_bt_gpio_numbers_t()` リストに列挙されています。
- CYW20820A1KFBG (62-ball FBGA) デバイスパッケージには、22 個の LHL GPIO と 2 つの Arm® GPIO (DEV_WAKE および HOST_WAKE) があります。LHL と Arm® GPIO の両方が `wiced_hal_gpio.h` の `wiced_bt_gpio_numbers_t()`リストに列挙されています。

LHL GPIO は、いくつかの点で Arm® GPIO と異なります。

- LHL GPIO は低電力モードで動作し、多重化可能なペリフェラルをサポートできます。I2C、SPI、PUART、ACLK などのペリフェラルは、LHL GPIO にその機能を引き出すことができ、スーパーマルチプレクサと呼ばれています。Super-Muxable のペリフェラルのリストについては、データシートを参照してください。LHL GPIO は、ほとんどのユーザーアプリケーションで通常使用されるものです。
- Arm® GPIO はアクティブモードでのみ動作でき、マルチプレクサ可能なペリフェラルをサポートしていません。DEV_WAKE、HOST_WAKE、および 4 つの BT_GPIO は、CYW20819A1KFB1G パッケージで利用可能な Arm® GPIO ですが、DEV_WAKE および HOST_WAKE は、CYW20819A1KFBG パッケージで利用可能な Arm® GPIO です。DEV_WAKE を使用して、ホストからの信号で Bluetooth®デバイスをウェイクさせることができます。HOST_WAKE 信号を使用して、Bluetooth®デバイスからの信号でホストデバイスをウェイクアップできます。BT_GPIO は、GCI (Global Co-existence Interface) ピンとして設定できます。

CYW20819/CYW20820 は、Bluetooth® SDK で以下の GPIO 機能をサポートしています。

- 入出力選択
- プルアップ/プルダウン選択
- ドライブ強度の選択
- ヒステリシス制御
- レベル/エッジトリガー割り込み選択

以下は、最も一般的に使用される GPIO API とその説明の一部です。

- `wiced_hal_gpio_configure_pin()` API は、特定の GPIO で GPIO 機能を設定するために使用されます。
- `wiced_hal_gpio_register_pin_for_interrupt()` を使用して GPIO の割込みを登録し、エッジまたはレベルの変更に基づいてコールバック関数 (つまり、割込みサービスルーチンまたは割込みハンドラー) をトリガーします。

- GPIO 割込みコールバック関数で割込みをクリアするためには、`wiced_hal_gpio_clear_pin_interrupt_status()` を使用する必要があります。
- `wiced_hal_gpio_select_function()` API を使用して、機能を特定のピンにマップできます (使用方法の例として、次のプログラムコードを参照してください)。この関数の最初のパラメーターは GPIO ピン (`wiced_bt_gpio_numbers_t`) を参照し、2 番目のパラメーターは GPIO 機能 (`wiced_bt_gpio_function_t`) を参照します。
- `wiced_hal_gpio_select_function (WICED_P03, WICED_PWM0);`

これらの GPIO API の詳細については、**Components > Hardware Drivers > GPIO** をクリックし、**AIROC™ API リファレンスガイド**の GPIO API リファレンスに移動してください。

SDK はすべての I/O の構成を提供していますが、アプリケーションでのピンの選択は、デバイスに関する以下の制約に基づいて行う必要があります。

- 特定の機能 (例えば ADC 入力) は、特定の LHL I/O でのみサポートされており、他のピンにルーティングできません。詳細については、**データシート**の Pin Assignments and GPIOs セクションを参照してください。
- P26, P27, P28, および P29 などの GPIO は、他のピンと比較してより高い電流を駆動できます。詳細については、**データシート**を参照してください。

4.2 アナログ-デジタルコンバーター (ADC)

CYW20819/CYW20820 の ADC ブロックは、オーディオ測定用に 12 ビット、DC 測定用に 10 ビットのシングルエンドスイッチトキャパシタシグマデルタ (Σ-Δ) ADC です。ADC には、28 個の GPIO 入力を含む 32 個の DC 入力チャネルがあります。 ADC は Super-Muxing 機能をサポートしていません。説明されている ADC API の詳細については、**Components** > **Hardware Drivers** > **Analog-to-Digital Converter (ADC)** をクリックし、AIROC™ API リファレンスガイドの ADC API リファレンスに移動してください。

- 次の 15 ピンは、CYW20819A1KFBG/CYW20820A1KFBG (62 ボール FBGA) で ADC 入力としてプログラムできます。
  - P0, P1, P8-P15, P17, P28, P29, P32, P37

Bluetooth® SDK は、*wiced_hal_adc.h* の VDDIO (入力/出力電圧)、VDD_CORE (コア電圧)、ADC_BGREF (バンドギャップリファレンス)、および ADC_REFGND (リファレンスグランド) の電圧/未加工サンプルを読み出すための API を提供します。詳細については、**データシート**の ADC 電気的特性を参照してください。

CYW20819/CYW20820 は、DC 測定モードとオーディオ [1]モードの 2 つの動作モードをサポートしています。オーディオ [1]アプリケーションの場合、ADC は Programmable-Gain Amplifier (PGA)) を使用してゲインを制御します。

4.2.1 DC 測定モード

DC 測定モードは、ADC グランドを基準とした入力ピンでのシングルエンド電圧測定を指します。ADC のブロック図から、28 個の GPIO チャネル、VDDIO、VDD_CORE、ADC_BGREF、および ADC_REFGND が多重化され、switched-capacitor sigma-delta ADC に供給されます。

`wiced_hal_adc_init()` 関数は、DC 測定モードでの ADC 動作に必要なレジスタを初期化します。ADC が特定の低電力モードでパワーダウンされている場合は、ADC を再初期化する必要があります。

1 DMA はオーディオモードでのみ使用できます。DMA 処理は ROM ファームウェアによって行われます。オーディオモードは現在、Bluetooth® SDK の HID プロファイルライブラリでのみサポートされています。ROM ファームウェアは、HID 音声コマンドの処理を処理します。

4.2.1.1 ADC 出力データとクロック

ADC クロック入力は 24 MHz で、ADC ハードウェアでは内部で 12 MHz に分割されます。ADC の静的測定では、標準的な変換レートは 100 kHz であり、最小変換レートは 50 kHz です。ADC の初期化後に DC 電圧を読み出す実際の時間は約 150 µs です。同様に、ADC から DC 生サンプルを読み出す実際の時間は約 50 µs です。変換レートの低下は、生のサンプル読み出し関数がサンプルの後処理を行わないためです。これに対して、電圧を読み出す場合、ファームウェアはサンプリングされた値の平均フィルタリング (low-pass FIR) を移動して精度を高めます。この変換率は、おおよそ 16 ksps から 20 ksps の範囲に変換されます。

4.2.1.2 DC 測定モードでの ADC 入力 MUX クロッキング

ADC は、DC 入力チャネルの切り替え後、セトリングに最大 2 µs を必要とします。ADC の電源投入後、ADC REF が安定するまでに約 20 µs かかります。DC 測定モードの集録時間は 10 µs です。これは、API 関数によって内部的に処理されます。ADC_DC_MUX_clk は、DC 入力チャネルのスイッチングと測定を同期させるためのデジタル側のタイミングクロックです。**Figure 2** に、MUX のクロッキングとタイミングを示します。

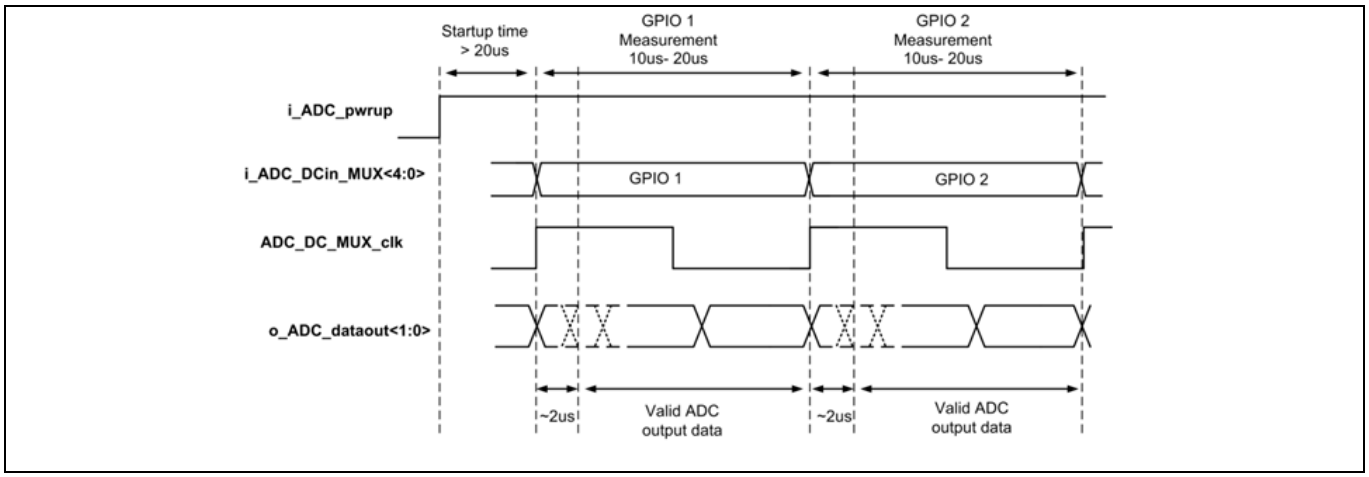

Figure 2 ADC の MUX クロッキングとタイミング図

4.2.1.3 ADC サンプリングとゲイン計算

`wiced_hal_adc_read_voltage()` API 関数を使用すると、プログラムは ADC からアナログ電圧を読み出せます。電圧を読み出す前に ADC を初期化する必要があります。そうしないと、関数は「0」を返します。ADC レジスタへの読み出し関数呼び出しはすべてブロックされます。DC 測定モードでの ADC 入力と ADC 出力ゲインの関係は、電源電圧 (ADC_AVDDBAT, MIC_AVDD, VDDC) により変化する可能性があります。電源電圧が変化した場合、より正確な ADC サンプルを取得するためには、次の API 関数を呼び出す必要があります。これらの関数は、呼び出されると、内部バンドギャップと基準電圧を再調整します。

- 電源電圧が 1.8 V に設定されている場合は、下記を使用します。
  - `wiced_hal_adc_set_input_range (ADC_RANGE_0_1P8V)`
- 電源電圧が 3.3 V に設定されている場合は、下記を使用します。
  - `wiced_hal_adc_set_input_range (ADC_RANGE_0_3P3V)`

4.3 クロック

CYW20819/CYW20820 には、複数の高速および低速ペリフェラルに接続された複数のクロック分周器と PLL があります。**Figure 3** に、CYW20819/CYW20820 の簡略化されたクロックツリーを示します。

CYW20819/CYW20820 のすべての高周波クロックの基本的なソースは、24 MHz の外部クリスタル (xtal) です。これは、周波数を 96 MHz に変換する ADPLL に供給されます。このクロックは、Bluetooth®コアや CPU などの CYW20819/CYW20820 デバイスのほとんどのペリフェラルに供給されます。PWM などの一部のペリフェラルは、24 MHz クロックを直接取得もできます。これらのペリフェラルには、必要に応じて入力クロックを分周できる内部分周器があります。

低周波数クロックは 32 kHz で、複数のソースから生成できます。詳細については、低周波数クロックソースを参照してください。

4.3.1 高周波数クロックソース

高周波数クロックは、内部 PLL を使用して周波数をアップスケールする外部 24 MHz 水晶発振器から生成されます。これらの周波数はハードウェアによって自動的に決定され、ユーザーが設定できません。

- HCLK: このクロックは、Resource Processing Unit (RPU) に電力を供給します。RPU には、CM4 CPU と DMA コントローラーが含まれます。HCLK の周波数の範囲は 1 MHz～96 MHz です。
- BT コアクロック: このクロックは、Bluetooth® RF ハードウェアに電力を供給します。Bluetooth®コアのクロック周波数の範囲は 1 MHz～48 MHz です。
- PTU クロック: このクロックは、SPI、I2C、UART、PCM、および Random Number Generator を含む Peripheral Transport Unit (PTU) に電力を供給します。PTU クロック周波数は、24 MHz または 48 MHz のいずれかです。ペリフェラルは、特定の目的のために内部でクロック周波数を分割します。たとえば、I2C マスタークロック (SCL) の範囲は 100 kHz～1 MHz です。
- タイマークロック: Application Timer (デュアル入力 32 ビットタイマー) は 1 MHz でクロックされます。

4.3.2 低周波数クロックソース

32 kHz の低周波数クロック (次の図での lhl_lpo_32kHz) は、複数のソースから取得できます。LP-LPO および HP-LPO と呼ばれる 2 つの内部低電力発振器 (LPO) と外部水晶 (OSC32K) があります。ファームウェアは、精度と電力要件に応じて、使用可能な LPO 間で使用するクロックソースを決定します。精度が高く、消費電流が最小であるため、推奨されるソースは外部 LPO (OSC32K) です。内部 LP-LPO の消費電流と精度は低く、HP-LPO の精度と消費電流は高くなります。ファームウェアは、外部 LPO のエラーが 250 PPM 未満で、ジッターがほとんどない、または全くないと想定しています。

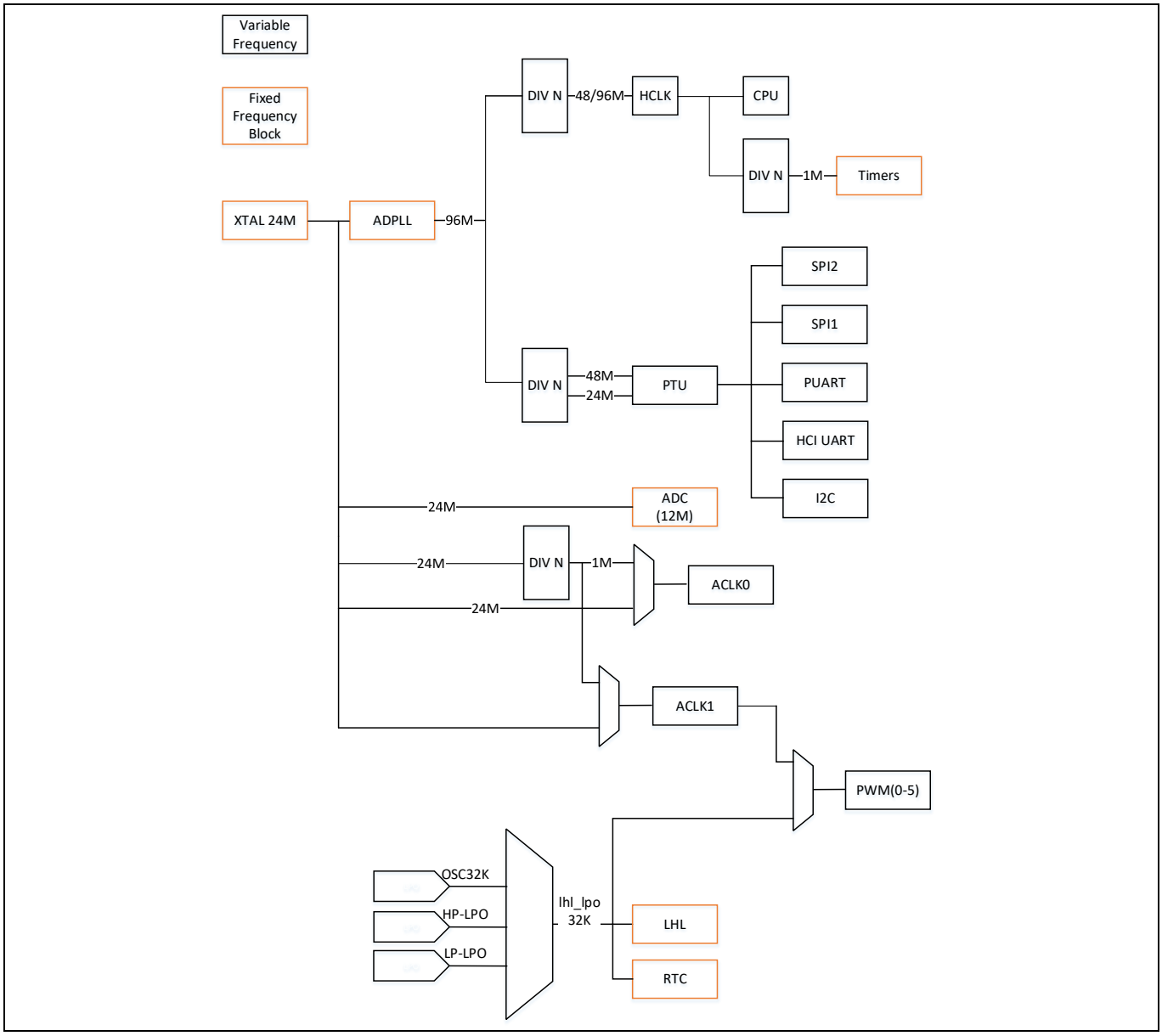

Figure 3　CYW20819/CYW20820 のクロック図

4.4　リアルタイムクロック (RTC)

CYW20819/CYW20820 は、32 kHz クリスタル (XTAL32K) LPO で実行される 48 ビット RTC タイマーをサポートします。RTC は、外部または内部 LPO からのクロック入力をサポートします。外部 LPO が CYW20819/CYW20820 に接続されていない場合、ファームウェアは RTC の内部 LPO からクロック入力を取得します。RTC タイマーは構造 wiced_rtc_time_t で表され、メンバーは秒, 分, 時, 日, 月, および年を表します。デフォルトでは、日付と時刻は 2010 年 1 月 1 日に設定され、時刻は 00:00:00 で、HH：MM：SS を示します。

Bluetooth® SDK は、以下の API 関数を提供します。

- 現在の時刻を設定します (`wiced_set_rtc_time()`)。
- 現在の時刻を取得します (`wiced_rtc_get_time()`)。
- 現在の時刻の値を文字列に変換します (`wiced_rtc_ctime()`)。

上記の RTC API の詳細については、**Components > Hardware Drivers > RTCDriver** をクリックし、**AIROC™ Bluetooth-SDK Documentation ページ**の RTCDriver API リファレンスに移動してください

4.5 ウォッチドッグタイマー (WDT)

CYW20819/CYW20820 のウォッチドッグタイマーモジュールは、誤動作の検出と回復に使用される 32 ビットダウンカウンターに基づいています。通常の動作中、デバイスは、カウント値がゼロに達する前にウォッチドッグタイマーを定期的にリセットして、経過やタイムアウトを防ぎます。ハードウェア障害またはプログラムエラーが原因で、デバイスがウォッチドッグをリセットできなかった場合、タイマーが経過し、タイムアウト時にシステムリセットが生成されます。ウォッチドッグタイマーのカウンターをリセットするプロセスは、「petting the dog」と呼ばれます。デフォルトのウォッチドッグタイムアウトは 2 秒に設定され、ウォッチドッグペッティングはアイドルスレッドで行われます。本番アプリケーションはウォッチドッグを満足させる必要はありません。CPU を 2 秒より長く保持しようとするアプリケーションは、ウォッチドッグをトリガーしてシステムを再起動します。ウォッチドッグタイマーを無効にするために、`wiced_hal_wdog_disable()` API が *wiced_hal_wdog.h* で利用可能です。

Bluetooth® SDK は、*wiced_hal_wdog.h* を使用したウォッチドッグタイマーを介して、制限されたデバッグ機能を提供します。ウォッチドッグタイマーが期限切れになると、システムはコアダンプ後にリセットされます。コアダンプには、デバイスのファームウェアまたはハードウェアのバージョン、警告フラグ、メモリ情報、CPU/HW レジスタ、SRAM、およびパッチ RAM イメージの情報が含まれています。コアダンプは、Bluetooth®-HCI ベンダー固有のイベントとして HCI-UART を介して送信されます。

4.6 アプリケーションタイマー

CYW20819/CYW20820 は、1 つの汎用 32 ビットデュアル入力タイマーを提供します。ファームウェアは、2 x 32 ビットタイマーを 1 x 64 ビットタイマーとして使用します。Bluetooth® SDK は、*wiced_timer.h* のタイマー機能を使用するための API を提供します。最初に、アプリケーションタイマーを特定の秒またはミリ秒のタイマー値に初期化してから、タイマーを開始し、タイムアウトに達すると、コールバック関数が実行されます。このコールバック関数は、LED の点滅, シリアルデータ転送, RTC 値の読み出し, ADC サンプルの読み出しなどのアプリケーションコードを保持できます。タイマーは 2 つの動作モードをサポートします。

- 定期割込みモード
- シングルショットモード

Bluetooth® SDK には、4 つのタイマータイプがあります。

- WICED_SECONDS_TIMER
- WICED_MILLI_SECONDS_TIMER
- WICED_SECONDS_PERIODIC_TIMER
- WICED_MILLI_SECONDS_PERIODIC_TIMER

2 つの定期的なタイマーは、タイマーカウントがゼロ (ダウンカウンター) に達するたびにタイマーコールバック関数をトリガーしますが、他の 2 つのタイマーはシングルショットです。タイマーコールバックは 1 回だけ実行され、タイマーが期限切れになるたびに終了します。タイマーAPI の使用を開始するために役立ついくつかの API は以下のとおりです。

- `wiced_init_timer()` 関数はタイマーを初期化します。タイマーが切れたときに呼び出されるコールバック関数を指定できます。この関数では、使用するタイマーのタイプを指定もできます。この関数はタイマーを開始しません。
- `wiced_start_timer()` 関数は、カウントダウンを開始し、カウント値を指定するために使用されます。`wiced_stop_timer()` 関数を使用して、タイマーを停止できます。

- `wiced_is_timer_in_use()` 関数を使用すると、ブール値を返すことにより、指定したタイマーが現在使用中かどうかを確認できます。

これらのアプリケーションタイマーAPI の詳細については、**Components** > **Hardware Drivers** > **Timer Management Services** をクリックし、**AIROC™ API リファレンスガイド**のタイマーAPI リファレンスに移動してください。

アプリケーションタイマーを使用するサンプルプログラムコードを以下に示します。

```
wiced_timer_t my_timer_handle; /* Typically defined as a global */
/* Typically, inside the BTM_ENABLED_EVT */
wiced_init_timer (&my_timer_handle,
 myTimer,
 0,
 WICED_MILLI_SECONDS_PERIODIC_TIMER);
wiced_start_timer (&my_timer_handle, 100);

/* The timer function */
void myTimer (uint32_t arg)
{
/* Put timer code here */
}
```

4.7 パルス幅変調器 (PWM)

CYW20819/CYW20820 には 6 つの 16 ビットハードウェア PWM チャネルがあり、Super-Muxable 機能に対応します。Bluetooth® SDK の *wiced_hal_pwm.h* ファイルは、PWM 固有のドライバーを定義します。LHL_CLK または PMU_CLK は、各 PWM チャネルのクロックソースとして使用できます。クロックソースが LHL_CLK の場合、クロック周波数は 32 kHz になります。PMU_CLK では、最初に補助クロックを設定する必要があります。補助クロックを設定する場合、ACLK0 は PWM で使用できません。したがって、ACLK1 は、24 MHz または 1 MHz (内部クロック分周を使用) で実行される唯一の利用可能な PMU_CLK です。SuperMUX を使用して、各 PWM チャネルを GPIO ピンにルーティングできます。`wiced_hal_gpio_select_function()`は、LHL GPIO を指定された PWM チャネル (`WICED_PWM<x>`) にマッピングするために使用されます。ここで、「x」は PWM チャネルを表します。PWM 設定パラメーターのランタイム変更中にエラーが発生する可能性があります。この API については、先に汎用入出力 (GPIO) で説明しました。PWM API の詳細については、**Components > Hardware Drivers > Pulse Width Modulation (PWM)** をクリックし、**AIROC™ Bluetooth®-SDK Documentation ページ**の PWM API リファレンスに移動してください。

- *wiced_hal_aclk.h* の `wiced_hal_aclk_enable()` 関数は、補助クロックを有効にし、PWM へのルーティングに 1 MHz または 24 MHz を選択する機能を許可します。

Note: *CYW20820 は、Bluetooth® SDK 1.3 の 1 MHz での ACLK をサポートしていません。*

- `wiced_hal_pwm_start()` 関数は、PWM を設定, 有効化, および開始し、事前設定された GPIO ピンにルーティングします。PWM で使用するために割り当てる前に、目的の GPIO ピンを出力として設定する必要があります。この関数は、5 つのパラメーター (つまり、チャネル, クロックソース, トグルカウント, 初期カウント, および反転信号) を受け取ります。
  - チャネルは、PWM チャネル 0～5 を指します。
  - クロックソースは、LHL_CLK または ACLK1 (PMU_CLK) のいずれかです。
  - トグルカウントは、信号を切り替える前に待機するティックの数を指します。

  - 初期カウントは、レジスタの初期値です。
  - 反転信号は 1 または 0 のいずれかです。反転信号が 1 の場合、PWM 出力は論理 HIGH で開始し、反転信号が 0 の場合、PWM 出力は論理 LOW で開始します。
- `wiced_hal_pwm_change_values()` 関数は、PWM ハードウェアブロックが既に起動された後、特定の PWM チャネルのトグルカウントや初期カウントなどの PWM 設定を変更します。
- `wiced_hal_pwm_get_params()` は、PWM パラメーターを計算するユーティリティ関数です。この関数は、入力クロック周波数, デューティサイクル, および必要な初期出力カウントとトグルカウントを決定するために必要な PWM 出力周波数の 3 つのパラメーターを受け取ります。初期カウントとトグルカウントは、次の式を使用して関数によって決定されます。
  - 初期カウント = 0xFFFF - (入力クロック周波数/PWM 周波数出力)
  - トグルカウント = 0xFFFF - ((デューティサイクルのパーセンテージ) x (入力クロック周波数/PWM 周波数出力)/ 100)

4.8 乱数発生器 (RNG)

Bluetooth® SDK は、RNG ハードウェアモジュールを使用するための API を提供します。単一の 32 ビット乱数を生成するか、特定の配列に 32 ビット乱数を入力する関数を提供します。これらの機能は、認証などのセキュリティ関連のアプリケーションに役立ちます。これらの関数は、「seed」と呼ばれる入力から乱数を生成します。この入力は、生成ハードウェアブロックの温度が高い場合はその温度から、Bluetooth®クロックが低い場合はそのクロックから取得されます。*wiced_hal_rand.h* ファイルは、32 ビットのランダムな整数を生成する `wiced_hal_rand_gen_num()` や、ランダムに生成された 32 ビットの整数で特定の配列を埋める `wiced_hal_rand_gen_num_array()` などの関数を定義します。 RNG ハードウェアブロックは、ウォームアップ時間を含む乱数 (単一の数値または配列) を生成するために 1 ミリ秒かかります。 RNG API の詳細については、**Components > Hardware Drivers > Random Number Generator (RNG)** をクリックし、**AIROC™ Bluetooth®-SDK Documentation ページ**の PWM API リファレンスに移動してください。

4.9 Inter-Integrated Circuit (I^2C)

CYW20819/CYW20820 は、I2C スレーブデバイスとシリアル通信するための 1 つの I2C 互換マスターインターフェースを提供します。I2C ブロックは Super-Muxable 機能をサポートします。

I2C モジュールの機能には、以下のサポートを含みます。

- 7 ビットアドレス指定モード
- マスターとしてのクロックストレッチ
- SCL クロック周波数: 100 kHz (標準モード), 400 kHz (高速モード), 800 kHz (独自仕様), および 1 MHz (高速モード+)
- マルチスレーブ動作

4.9.1 I2C クロック

I2C クロック (SCL) は、I2C バス上のマスターによって提供されます。I2C モジュールがマスターモードの場合、シリアルクロックジェネレーターは 24 MHz のトランスポートクロックから SCL クロックを生成します。`wiced_hal_i2c_set_speed()` 関数を使用すると、SCL のクロックレートを概要にリストされている速度のいずれかに設定できます。I2C ブロックには、トランザクションバッファ用に 64 バイトの RX および TX FIFO があるため、各 I2C トランザクションペイロードは最大 64 バイトにできます。

必要な SCL クロック周波数を設定するために、次の列挙定数が *wiced_hal_i2c.h* で提供されています。

- `I2CM_SPEED_100KHZ`

- I2CM_SPEED_400KHZ
- I2CM_SPEED_800KHZ
- I2CM_SPEED_1000KHZ

例えば、I2CM_SPEED_100KHz の列挙値は、除算パラメーターである 240 に設定されます。つまり、24,000,000 を 240 で除算すると、100 kHz になります。マルチマスターを使用しようとすると、2 つ以上のマスターからのクロックが同期されないため、未定義の動作が発生します。また、独自仕様の 800 kHz モードは、すべての I2C スレーブで動作するとは限りません。wiced_hal_i2c_get_speed() 関数を使用すると、I2C ブロックの現在のクロック周波数を取得できます。

4.9.2 I2C 初期化

I2C モジュールレジスタは、*wiced_hal_i2c.h* で定義される wiced_hal_i2c_init()を呼び出すことによって初期化されます。この関数呼び出しは、I2C ドライバーとそのプライベート値を初期化します。一部のモジュールは節電のためにブロックをオフにするため、I2C を使用する前にこの関数を呼び出す必要があります。この関数呼び出しにより、ハードウェアブロックへのクロック信号がデフォルトの SCL クロック周波数 100 kHz で確実にオンになります。したがって、100 kHz 以外の速度を使用する場合は、wiced_hal_i2c_init() 関数の後に wiced_hal_i2c_set_speed() 関数を呼び出す必要があります。

4.9.3 I2C 動作

I2C マスターモジュールは、マスター書き込み, マスター読み出し, および読み出しに続く結合書き込みの 3 つの動作をサポートしています。これらの動作については、以下のセクションで説明します。

4.9.3.1 I2C マスター書き込み

wiced_hal_i2c_write() 関数は、特定のスレーブアドレスにデータを書き込みます。任意の長さのデータをスレーブに書き込めますが、ハードウェアの能力を超えるアトミックトランザクションは不可能であり、データは複数のトランザクションに分割されます。これはブロッキングコールです。I2C スレーブデバイスが成功した I2C マスター書き込み操作を確認すると、関数は I2CM_SUCCESS を返し、I2C マスターが I2C スレーブから確認を受信しない場合、関数は I2CM_OP_FAILED を返します。

4.9.3.2 I2C マスター読み出し

wiced_hal_i2c_read() 関数は、特定のスレーブアドレスレジスタから指定されたバッファにデータを読み出します。これはブロッキング呼び出しです。この I2C 関数は、I2C スレーブデバイスからの NACK (確認応答なし) がある場合、または I2C スレーブが 0 バイトを返す場合、I2CM_OP_FAILED を返します。詳細については、**AIROC™ Bluetooth®-SDK Documentation ページ**を参照してください。

4.9.3.3 I2C マスターの書き込みと読み出しの組合せ

wiced_hal_i2c_combined_read() 関数は、書き込みトランザクションを実行し、続いて読み出しトランザクションが実行され、トランザクション間に開始条件が繰り返されます。最初のトランザクションでは、データはスレーブアドレスに書き込まれ、繰り返し開始した後、2 番目のトランザクションでスレーブからデータが読み出されます。この操作は通常、スレーブデバイスのレジスタを読み出すために使用されます。最初の書き込みトランザクションでは、読み出す必要があるスレーブのレジスタアドレスを指定します。この I2C 関数は、I2C スレーブデバイスからの NACK (確認応答なし) がある場合、または I2C スレーブが 0 バイトを返す場合、I2CM_OP_FAILED を返します。

4.10 シリアルペリフェラルインターフェース (SPI)

CYW20819/CYW20820 には、他の SPI マスターおよびスレーブデバイスとの通信に使用される 2 つの SPI ブロックが含まれます。CYW20819/CYW20820 SPI インターフェースは、Super-Muxable 機能に対応しま

す。このブロックは、温度センサーやモーションセンサーなどの SPI ベースのセンサーとの通信に使用できます。SPI ブロックは以下の機能に対応します。

- 3 線式 (マスター) および 4 線式 (マスターおよびスレーブ) SPI インターフェース
- マスターモードとスレーブモード
- 設定可能な SCK 極性と位相
- 設定可能な LSB または MSB ファースト転送
- SPI 1[2]用の 1024 バイトの送信バッファと 1024 バイトの受信バッファ (HCI UART と共有)
- SPI 2[2] 用の 64 バイトの送信バッファと 64 バイトの受信バッファ (PUART と共有)

CYW20819/CYW20820 は、2 つの SPI ハードウェアブロックから選択する機能を提供します。`spi_interface_t` では、`SPI1` と `SPI2` のどちらかを選択できます。各 SPI ユーティリティ関数には、2 つの SPI ブロックから選択する引数があります。

SPI ブロックは、汎用マスターまたは汎用スレーブとして使用できます。SPI インターフェースには、どの LHL GPIO も使用できます。Bluetooth® SDK は、SPI ドライバーにアクセスするためのパラメーターと関数のリストを提供します。 SPI API の詳細については、**Components > Hardware Drivers > PeripheralSpiDriver** をクリックし、**AIROC™ Bluetooth®-SDK Documentation ページ**の SPI API リファレンスに移動してください。

SPI ブロックは、`wiced_hal_pspi_init()` 関数を使用して、必要な設定で初期化できます。汎用マスターモードでは、CS をさまざまなペリフェラルにアサートするのはファームウェア次第です。マスターモードでは、CS ピンと INT ピンを GPIO として使用できます。SPI クロックとピンは、SPI ブロックにアクティビティがない場合、アプリケーションの実行中に変更できます。以下に説明するように、3 種類のマスタートランザクションと 2 種類のスレーブトランザクションがあります。

4.10.1 SPI マスターTX のみの動作

データのみを送信する必要がある場合、SPI ブロックは TX 専用モードです。このモードでは以下のとおりです。

- TxFIFO を有効にする必要があります
- RxFIFO を無効にする必要があります

この設定では、SPI ブロックは TxFIFO に配置されたすべてのデータを送信します。`wiced_hal_pspi_tx_data()` 関数を使用して、データを TxFIFO に配置し、データをマスターとして送信します。

TxFIFO が空になると、`wiced_hal_pspi_tx_data()` 関数を使用して TxFIFO にさらにデータが配置されるまで、バイト境界で SPI_CLK が一時停止します。RxFIFO でデータが受信されません。

4.10.2 SPI マスターRX のみの動作

データの受信のみが必要な場合、SPI ブロックは RX-Only モードです。このモードでは以下のとおりです。

- TxFIFO を無効にする必要があります
- RxFIFO を有効にする必要があります

2 SPI1 ブロックと HCIUART は同じバッファを使用するため、同時に使用できません。SPI 2 ブロックと PUART は同じバッファを使用するため、同時に使用できません。

この設定では、SPI ブロックはスレーブから指定されたバイトのデータを取得します。
`wiced_hal_pspi_rx_data()` 関数は、マスターとしてデータを受信するために使用されます。

RxFIFO がいっぱいになると、データストリームが一時停止します。RxFIFO にスペースがあると、データストリームが再開されます。このモードではデータは送信されません。

4.10.3 SPI マスター全二重動作

データを同時に送受信する必要がある場合、SPI ブロックは全二重モードです。このモードでは以下のとおりです。

- TxFIFO が有効になっている
- RxFIFO が有効になっている

この設定では、次の場合、SPI ブロックは両方の FIFO との間でデータを転送します。

- RxFIFO のスペース
- TxFIFO のデータ

これらの条件のいずれかが偽になると、両方の条件が再び真になるまで SPI_CLK が一時停止します。
`wiced_hal_pspi_exchange_data()` 関数を使用して、データを同時に送受信できます。

4.10.4 SPI スレーブ TX のみの動作

汎用スレーブとして、次の場合にデータがホストに転送されます。

- TxFIFO にデータがある
- TxFIFO が有効になっている
- ホストが SPI_CLK を切り替え、SPI_CSN がアサートされている

条件 (1) が偽で、条件 (2) および (3) が真の場合、データアンダーフロー条件が発生します。
`wiced_hal_pspi_slave_tx_data()` 関数を使用して、データを TxFIFO に配置できます。

4.10.5 SPI スレーブ RX のみの動作

一般的なスレーブとして、次の場合にデータがホストから受信されます。

- RxFIFO にスペースがある
- RxFIFO が有効になっている
- ホストが SPI_CLK を切り替え、SPI_CSN がアサートされている

条件 (1) が偽で、条件 (2) および (3) が真の場合、データオーバーフロー条件が発生します。
`wiced_hal_pspi_slave_rx_data()` 関数を使用して、データを受信できます。

4.10.6 SPI スレーブ全二重動作

データを同時に送受信する必要がある場合、SPI ブロックは全二重モードです。このモードでは以下のとおりです。

- TxFIFO が有効になっている
- RxFIFO が有効になっている

この設定では、次の場合、SPI ブロックは両方の FIFO との間でデータを転送します。

- RxFIFO のスペース

- TxFIFO のデータ

データを同時に送受信するためには、Tx バッファと Rx バッファを有効にし、`wiced_hal_pspi_slave_tx_data()` API を使用してデータをバッファに配置し、マスターがデータを転送した後に `wiced_hal_pspi_slave_rx_data()` を使用して Rx バッファからデータを読み出します。

4.11 UART

このデバイスには、2 つの UART ブロックがあります。周辺 UART (PUART) と HCI UART (Bluetooth® UART)

4.11.1 ペリフェラル UART (PUART)

CYW20819/CYW20820 の PUART は、デバッグ印刷機能に加えて、汎用 Tx/Rx UART ブロックとして使用できます。CYW20819/CYW20820 は、ペリフェラル UART を任意の LHL GPIO にマップできます。PUART は 24 MHz でクロックされ、ボーレートとパリティを設定できます [3]。PUART RX と PUART TX の両方に 256 バイトの FIFO があります。

PUART には次の機能があります。

- 最大 3 Mbps の 8 ビット転送
- 最大 2.5 Mbps のストップビットを含む 9 ビット転送
- 設定可能なフロー制御
- 受信操作時の割込み機能

4.11.1.1 PUART 初期化

wiced_hal_puart.h で定義されている `wiced_hal_puart_init()`関数は、PUART ブロックを初期化するために使用されます。

`wiced_hal_puart_select_uart_pads()`関数を使用して、PUART ハードウェアが使用する TX/RX およびオプションの CTS/RTS ピンを選択できます。

4.11.1.2 PUART ボーレート

`wiced_hal_puart_set_baudrate()`関数は、ボーレートを設定するために使用されます。デフォルトの PUART ボーレートは 115200 です。標準的なボーレートには 115200、921600、1500000、および 3000000 bps が含まれますが、中間速度も利用できます。ボーレートの詳細については、データシートを参照してください。

4.11.1.3 PUART 送信

`wiced_hal_puart_enable_tx()`および `wiced_hal_puart_disable_tx()`関数は、それぞれ PUART の送信機能を有効または無効にします。PUART を介してデータを送信するためには、送信を有効にする必要があります。

`wiced_hal_puart_print()`関数は、TX ラインを介して文字列を送信するために使用されます。

`wiced_hal_puart_write()`関数は、TX ラインを介して 1 バイトを送信するために使用されます。

4.11.1.4 PUART 受信

`wiced_hal_puart_enable_rx()`関数は、受信機能を有効にするために使用されます。

3 BT SDK のドライバーが利用できるかによります。Bluetooth® SDK v1.3 以降、SDK はパリティ選択をサポートしていません。

wiced_hal_puart_rx_fifo_not_empty()関数は、RX FIFO に使用可能なデータがあるかどうかを確認するために使用されます。

wiced_hal_puart_register_interrupt()関数は、データの受信時に割込みコールバックを提供します。

データを受信するためには、割込みコールバック関数を登録し、ウォーターマークを設定して、割込みがトリガーされる前に受信するバイト数を決定し、Rx を有効にします。PUART でデータを受信するためのプログラムコードを以下に示します。

```
wiced_hal_puart_register_interrupt(rx_interrupt_callback);
/* Set watermark level to 1 to receive interrupt up on receiving each byte */
wiced_hal_puart_set_watermark_level (1);
wiced_hal_puart_enable_rx ();
```

Rx 処理は、割込みコールバック関数内で行われます。追加の文字を受信するためには、コールバック関数内で PUART 割込みをクリアする必要があります。

```
void rx_interrupt_callback (void* unused)
{
        uint8_t readbyte;
        /* Read one byte from the buffer and then clear the interrupt */
        wiced_hal_puart_read (&readbyte);
        wiced_hal_puart_reset_puart_interrupt ();

        /* Add your processing here */
}
```

4.11.1.5 PUART フロー制御

wiced_hal_puart_flow_on()および wiced_hal_puart_flow_off()関数は、それぞれ PUART のフロー制御を有効または無効にします。

4.11.2 ホストコントローラーインターフェース UART

この UART は、コントローラーモードの HCI トランスポートに使用され、デバイスのプログラミングにも使用されます。HCI UART 信号は特定の I/O パッドに固定されています。DMA は、受信するデータの長さが約 64 バイトを超える場合に HCI UART パケットに使用されます。

HCI UART インターフェースを介したプログラミングシーケンスを理解するためには、GitHub **btsdk-docs** リポジトリの *WICED-HCI-Control-Protocol.pdf* を参照してください。

4.12 キーボードスキャナー (Keyscan)

キーボードスキャナーは、GPIO からのキープレスおよびリリースイベントを処理できるマトリクススキャナーハードウェアペリフェラルです。キーボードスキャナーは、ハードウェア FIFO の上にファームウェアキューとして実装されます。キーボードスキャナーのアーキテクチャについては、**Figure 4** を参照してください。

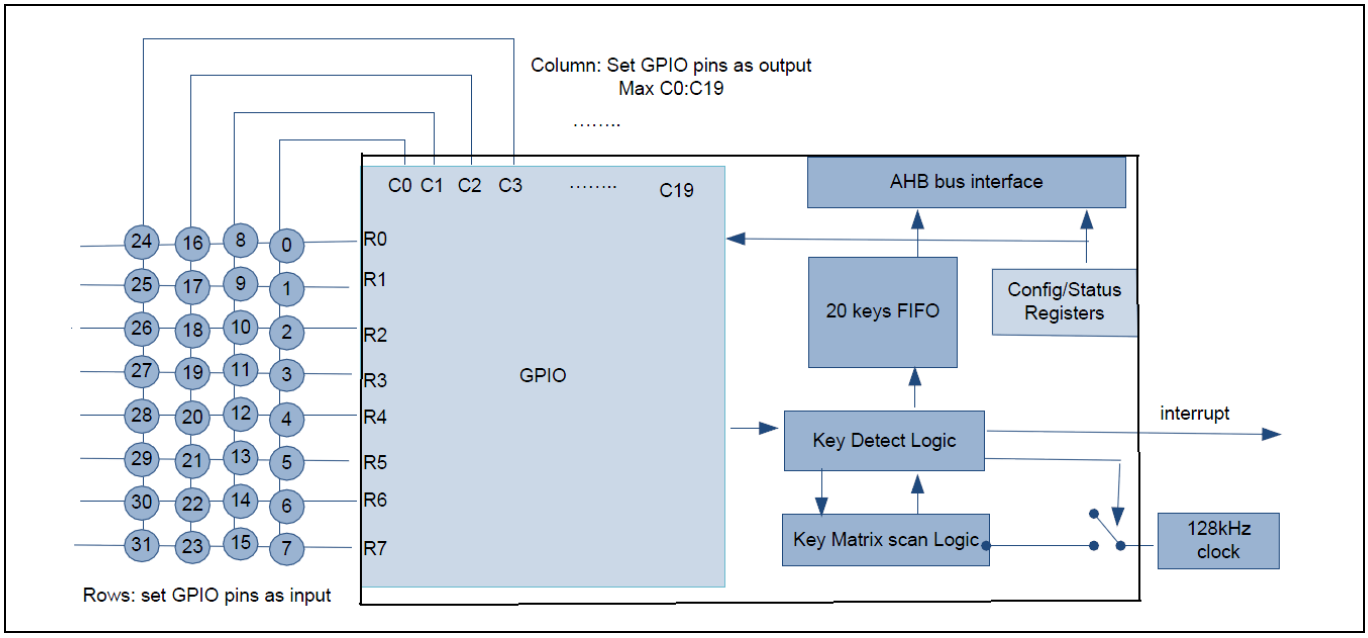

Figure 4　CYW20819/CYW20820 用キーボードスキャナー

キーボードスキャナーハードウェアブロックの仕様は次のとおりです。

- 設定できる最大行数 (Key Scan Inputs (KSI)) は 8 です。
- 設定できる列 (Key Scan Outputs (KSO)) の最大数は 20 です。
- キーボードスキャナーへのすべての列ピンはスーパーマルチプレクサです。
- キーボードスキャナーブロックは、128 KHz で 160 キー全体をサンプリングします。
- ハードウェアデバウンスとゴーストキー検出が利用可能です。

CYW20819/CYW20820 のキーボードスキャナーは、ホストマイクロコントローラーが介入する必要なく、キーを自律的にサンプリングしてバッファレジスターに格納するように設計されています。キーアップ/ダウンイベントは、API ドライバーからのバイトストリームです。ROM ファームウェアは、キーボードスキャナーハードウェアのステートマシン処理を処理します。ステートマシンはアイドル状態で始まり、いずれかのキーが押されるとスキャン状態になります。スキャン状態の間、行カウンターは 0 からプログラム可能な行数-1 およびプログラム可能な列数-1 までカウントします。特定のパッケージで利用可能な行と列の最大数については、CYW20819/CYW20820 データシートを参照してください。

キーボードスキャナーの構成と利用に必要な API は、*wiced_hal_keyscan.h* にあります。キーボードスキャナーAPI は、以下の機能のための関数を提供します。

- キーボードスキャナーの行と列の数を設定
- 新しいキープレスイベントと保留中のキープレスイベントをチェック
- ゴーストキーの検出を有効化/無効化
- キーボードスキャナーハードウェアブロックをリセット
- デバイスのウェイクアップソースを提供

関数のプロトタイプとパラメーターの詳細については、API リファレンスガイドを参照してください。アプリケーション開発には、ModusToolbox™のデバイスコンフィギュレーターツールを使用して、キーボードスキャナーハードウェアを設定できます。

5 Bluetooth® Low Energy 無線の機能

ここでは、CYW20819/CYW20820 で利用可能ないくつかの Bluetooth® Low Energy 無線機能について説明します。CYW20819/CYW20820 デバイスは、高度に統合された低消費電力の 2.4 GHz 無線トランシーバを提供し、複数の変調およびパケットフォーマットをサポートします。CYW20820 は、10 dBm の送信電力で送信できる統合パワーアンプを追加します。MCU サブシステムは、ペリフェラルと無線の間のインターフェースを提供し、コマンドの発行、ステータスの読み出し、無線イベントの自動化とシーケンス化を可能にします。Arm Cortex-M4 プロセッサは、アナログ RF およびベースバンド回路との間のデータを処理し、情報ビットを所定のパケット構造にアセンブルします。

5.1 TX 電力制御

CYW20819 の最大出力は、クラス 2 動作で+4dbm です。+4dBm～-16dBm の TX 電力をサポートしています。解像度は 4dBm です。つまり、設定可能な TX 電力レベルは{+4 dBm, 0 dBm, -4 dBm, -8 dBm, -12 dBm, -16 dBm}です。

CYW20820 には、クラス 2.5 で動作する最大電力出力が+10dbm の内部パワーアンプがあります。+4dBm から-24dBm までの TX 電力をサポートします。解像度は 4dBm です。つまり、設定可能な TX 電力レベルは{+4 dBm, 0 dBm, -4 dBm, -8 dBm, -12 dBm, -16 dBm, -20 dBm, -24 dBm}です。

Bluetooth® SDK は、TX 電力を制御するために次の機能を提供します。

- `wiced_bt_set_tx_power()`: *wiced_bt_dev.h* で定義されるこの関数を使用して、特定の接続のデータチャネルの TX 電力を設定できます。ピアデバイスの BD (Bluetooth®デバイス) アドレスを入力パラメーターとして受け取ります。この API を使用して、`BD_ADDR` を 0 として渡すことにより、アドバタイズメントチャネルの TX 電力を設定することもできます。

5.2 Co-existence - SECI

Serial Enhanced Coexistence Interface (SECI) は、Infineon-to-Infineon Coexistence ソリューションにおける独自のインフィニオンインターフェースです。`WICED_GCI_SECI_IN` および `WICED_GCI_SECI_OUT` 機能は、CYW20819/CYW20820 で利用可能な GPIO のいずれかにルーティングする必要があります。*wiced_bt_dev.h* で定義されている `wiced_bt_coex_enable()`および `wiced_bt_coex_disable()`関数を使用すると、Coexistence インターフェースを有効および無効にできます。Coexistence インターフェースとそれらを CYW20819/CYW20820 で使用する方法については、**AN214852** を参照してください。

6 ソフトウェア

ここでは、CYW20819/CYW20820 Bluetooth®デバイスでのアプリケーションコードのビルドおよびダウンロードプロセスについて説明します。

6.1 アプリケーションコード

すべての Bluetooth®ワイヤレスシステムは同じ基本的な方法で動作します。プログラマは、ネットワークスタックで Bluetooth® API を呼び出すアプリケーションファームウェアを記述します。次に、スタックは無線ハードウェアと通信し、無線ハードウェアはデータを送受信します。無線ハードウェアでイベントが発生すると、スタックはイベントを作成することにより (例えば、Bluetooth®接続の反対側からメッセージを受信したとき)、アプリケーションファームウェアでアクションを開始します。アプリケーションコードは、スタックによってプッシュされたこれらのイベントを処理し、各イベントに必要な後処理を行います。ROM には、ブートシーケンス, デバイスドライバー機能, 低レベルのプロトコルスタックコンポーネント, およびブートローダーがあらかじめプログラムされています。CYW20819/CYW20820 の ROM ファームウェアは、Bluetooth®スタックアクティビティを処理し、無線イベントを処理します。ROM ファームウェアは、Bluetooth®イベントの処理において Finite-State Machine (FSM) のように機能します。アプリケーション開発者は、低レベルの Bluetooth®機能を無視できます。

アプリケーションの最小 C ファイルは次のようになります。

```
#include "wiced.h"
#include "wiced_platform.h"
#include "sparcommon.h"
#include "wiced_bt_dev.h"
/***************************** Function Prototypes ******************/
wiced_result_t bt_cback (wiced_bt_management_evt_t event, wiced_bt_management_evt_data_t
*p_event_data);
/***************************** Functions ****************************/
/* Main application. This just starts the BT stack and provides the callback function.
* The actual application initialization will happen when stack reports that BT device is
* ready.
*/
APPLICATION_START ()
{
        /* Add initialization required before starting the BT stack here */
        wiced_bt_stack_init (bt_cback, NULL, NULL); /* Register BT stack callback */
}
/* Callback function for Bluetooth® events */
wiced_result_t bt_cback (wiced_bt_management_evt_t event, wiced_bt_management_evt_data_t
*p_event_data)
{
        wiced_result_t result = WICED_SUCCESS;
        switch (event)
        {
                /* Bluetooth® stack enabled */
                case BTM_ENABLED_EVT:
                /* Initialize and start your application here once the BT stack is running */
                        break;
```

```
            default:
                    break;
        }
        return result;
}
```

RTOS の目的は、リソース要件が重複している複数の非同期応答時間クリティカルタスクを持つ組込みファームウェアの記述の複雑さを軽減することです。RTOS は、アクティブ, 非アクティブ, またはスケジュールなどのさまざまな状態にあるスレッドと呼ばれるタスクのリストを維持し、これらのタスクを優先度に基づいて実行します。SoC ペリフェラルを利用するマルチスレッドアプリケーションは、共有メモリ, 相互排他, ホールドアンドウェイト, および循環ウェイトなどのリソースの競合がある場合に、アプリケーションのスレッド同期を処理する必要があります。

ROM ファームウェアの RTOS は、Bluetooth®機能を処理するための起動直後に複数のスレッドを作成し、アプリケーションスレッドに制御を与えます。現在、ModusToolbox™は複数の RTOS をサポートしています。デバイス ROM には、Express Logic による ThreadX が組み込まれており、ライセンスが含まれているため、CYW20819/CYW20820 を使用したアプリケーションの開発に最適です。複数の RTOS の使用を簡素化するために、Bluetooth® SDK には、基本的な RTOS 機能への統一されたインターフェースを提供する組込みの抽象化レイヤーがあります。**Components > RTOS** の AIROC™ RTOS API のドキュメントは、**AIROC™ Bluetooth®-SDK Documentation ページ**にあります。

6.2 低電力機能

AN225270 CYW20819/CYW20820 BLE low power guideline アプリケーションノートは、CYW20819/CYW20820 を使用した低電力アプリケーションの設計と開発のガイドラインと、電力効率の良い設計を検討するためのパラメーターを提供します。このドキュメントは、**CYW920819**/**CYW920820** 製品ページにあります。

6.3 ファームウェアのアーキテクチャ

CYW20819/CYW20820 デバイスは、コントローラーモードまたは組込みモードで使用できます。

コントローラーモードでは、CYW20819/CYW20820 デバイスは Bluetooth®コントローラースタックを実行し、Bluetooth®ホストスタックは外部ホスト MCU で実行されます。CYW20819/CYW20820 デバイスは、Host Controller Interface (HCI) を使用してホストコントローラーと通信します。

組込みモードでは、Bluetooth®ホストスタックとコントローラースタックの両方が CYW20819/CYW20820 デバイスで実行されます。コントローラースタックのすべてのコンポーネントとホストスタックのほとんどのコンポーネントは、デバイス ROM にあります。API を呼び出して ROM コードにアクセスできるユーザーアプリケーションは、フラッシュにプログラムされています。フラッシュにコードがプログラムされておらず、デバイスの電源が入っている場合、HCI によって制御される Bluetooth®コントローラーのように動作します。

CYW20819/CYW20820 デバイスは、ROM コードに「パッチ」を使用して動作を変更し、バグを修正します。これらのパッチは、すべての ModusToolbox™ビルドに含まれるコアパッチファイル (このドキュメントの残りの部分では一般パッチと呼ばれます) に組み込むことも、オプションのパッチライブラリ (このドキュメントの残りの部分ではパッチライブラリと呼ばれる) に組み込むこともできます。オーディオまたはメッシュネットワークアプリケーションのみに関連するパッチなどのように、それらはアプリケーション固有のものです。このアプローチでは、特定のアプリケーションに必要なパッチライブラリのみを使用して、RAM の使用を最適化します。

インフィニオンの Bluetooth®デバイスは、アプリケーションコードの Over-The-Air (OTA) 更新をサポートしています。OTA の詳細については、GitHub **btsdk-docs** リポジトリの *WICED-Firmware-Upgrade-Library.pdf* および *WICED-Secure-Over-the-Air-Firmware-Upgrade.pdf* を参照してください。

ソフトウェア

デバイスのプログラミングは、パッチとアプリケーションコードをフラッシュにコピーするだけではありません。また、コマンドを使用して構造化データを設定し、CPU に関数呼び出しを行わせたり、ハードウェアレジスタを設定したりします。ファームウェアをダウンロードするデフォルトの方法は、HCI UART を介したブートロードです。デバイス ROM には、外部デバイス (PC または MCU) から HCI コマンドを介してデータを受け取り、フラッシュに格納できるコードが含まれます。アプリケーションコードがダウンロードされると、ホスト MCU (または PC) は最初にミニドライバーと呼ばれるコードを SRAM に転送します。次に、ミニドライバーは HCI コマンドを介して追加のデータを受け入れ、それを on-chip flash (OCF) に保存します。ミニドライバーの詳細については、このドキュメントで後ほど説明します。

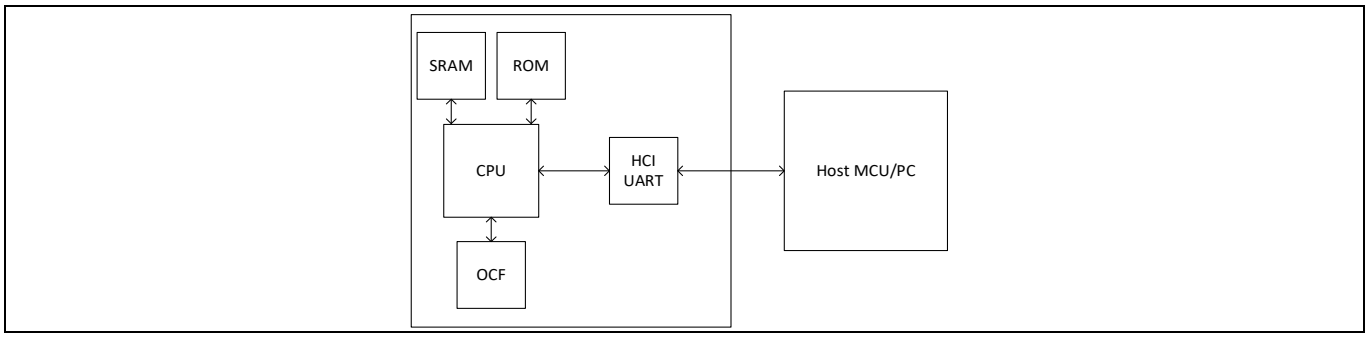

Figure 5　CYW20819/CYW20820 ファームウェアのダウンロードフロー

6.4　メモリレイアウト

はじめにで説明したように、CYW20819/CYW20820 SoC はさまざまな物理メモリで構成されています。ここでは、CYW20819/CYW20820 SoC のさまざまなメモリについて詳しく説明します。ROM とフラッシュは不揮発性の物理メモリですが、SRAM は CYW20819/CYW20820 の揮発性メモリの大部分を提供します。ROM コードを変更するためのパッチ RAM の使用については、ファームウェアのアーキテクチャで説明しました。

Bluetooth®デバイスの不揮発性メモリ (オンチップフラッシュまたは外部シリアルフラッシュ) は、**Figure 6** に示すように 5 つの論理セクションに分かれています。**Table 1** に、メモリレイアウトの各セクションの詳細を示します。Bluetooth®スタックのアプリケーションバッファプール要件の詳細については、**AN216403 - ModusToolbox™ AIROC™ Application Buffer** を参照してください。

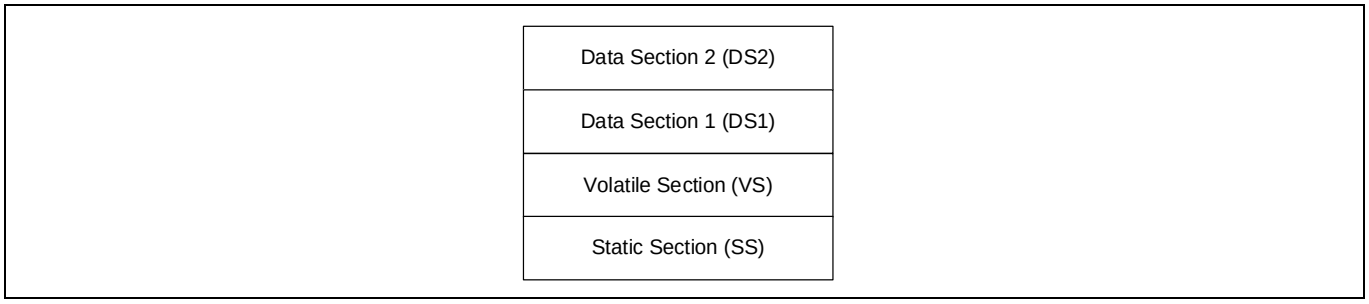

Figure 6　不揮発性メモリのメモリ構成

Table 1　不揮発性メモリ構成、オンチップフラッシュ

セクション名	オフセット	長さ	変更内容
Static Section (SS)	0x500000	0x2000	チップファームウェアによって内部的に使用される静的セクション。これには通常、出荷時のプログラミング後に変更されない設定レコードが含まれます。

ソフトウェア

セクション名	オフセット	長さ	変更内容
Volatile Section (VS)	0x500400	0x1000	アプリケーションまたはスタックがデータを外部またはオンチップフラッシュメモリに格納するために使用する最初の揮発性セクション。
Data Section (DS1)	0x501400	0x1F600	最初のパーティション。アプリケーションコードを保存するために使用されます。
Data Section (DS2)	0x520A00	0x1F600	2 番目のパーティション。アプリケーションコードを保存するために使用されます。

これらの値は目安であり、実際の値は次に説明する BTP ファイルにあります。

Table 1 に記載されている論理セクションは、メモリの消去セクター境界に揃えられています。これらの詳細は、*\<mtw_path>\wiced_btsdk\dev-kit\baselib\<device>\platforms\208XX_OCF.btp* にある*.btp ファイルにあります。メモリレイアウトの詳細については、GitHub **btsdk-docs** リポジトリの *WICED-Firmware-Upgrade-Library.pdf* ファイルを参照してください。

6.4.1 *.btp ファイルの重要性

**.btp* ファイルには、ミニドライバーの場所, ボーレート, メモリセクションアドレス, およびその他の詳細など、ダウンロードツールで使用される情報が含まれます。また、デバイスのプログラミング中に変更できる BD ADDRESS などの情報も含まれます。パラメーターの一部は以下のとおりです。

- DLConfigCrystalFreqMHz X 10000 –デバイスに接続されている水晶の周波数
- DLConfigSerialControlBaudRate –プログラミングに使用される UART ボーレート
- ConfigDSLocation – DS1 ロケーションオフセット
- ConfigDS2Location – DS2 ロケーションオフセット
- DLConfigSSLocation – SS ロケーションオフセット
- DLConfigBD_ADDRBase – BD アドレス

6.5 プログラム

前述のように、CYW20819/CYW20820 デバイスは通常、コントローラーモードまたは組込みモードの 2 つの設定で使用できます。

コントローラーモードでは、ホストは通常、Bluetooth®デバイスの SRAM または OCF に一般的なパッチなどのコードをダウンロードします。コードが SRAM にダウンロードされた場合、リセット時に SRAM の内容が失われるため、起動ごとにコードを実行する必要があります。

組込みモードでは、デバイスの組込みスタックが一部のホスト機能に使用され、アプリケーションコードをデバイスの OCF にダウンロードする必要があります。

6.5.1 ファイルフォーマット

アプリケーションコードは通常、*.hex* または*.hcd* 形式で保持されます。通常、**.hcd* ファイルは SRAM のダウンロードに使用され、**.hex* ファイルはフラッシュのダウンロードに使用されます。中間ステップとして、ソースコードといくつかの設定値が*.cgs* (設定ソース) 形式に変換され、次に*.hex* または*.hcd* 形式に変換されます。*.hcd* ファイルは、ファイルサイズが小さく、ペイロード形式が HCI コマンドに似ているため、外部 MCU がコントローラーモードで Bluetooth®デバイスをプログラムする必要がある場合に役立ちます。Intel *.hex* ファイルは、フラッシュへのプログラミング時に使用されます。次の図に、ファイル変換プロセスを示します。

ソフトウェア

デバイスにダウンロードするためには、これらのファイルを解析して HCI コマンドに変換する必要があります。ModusToolbox™では、これは ChipLoad ツールによって行われます。

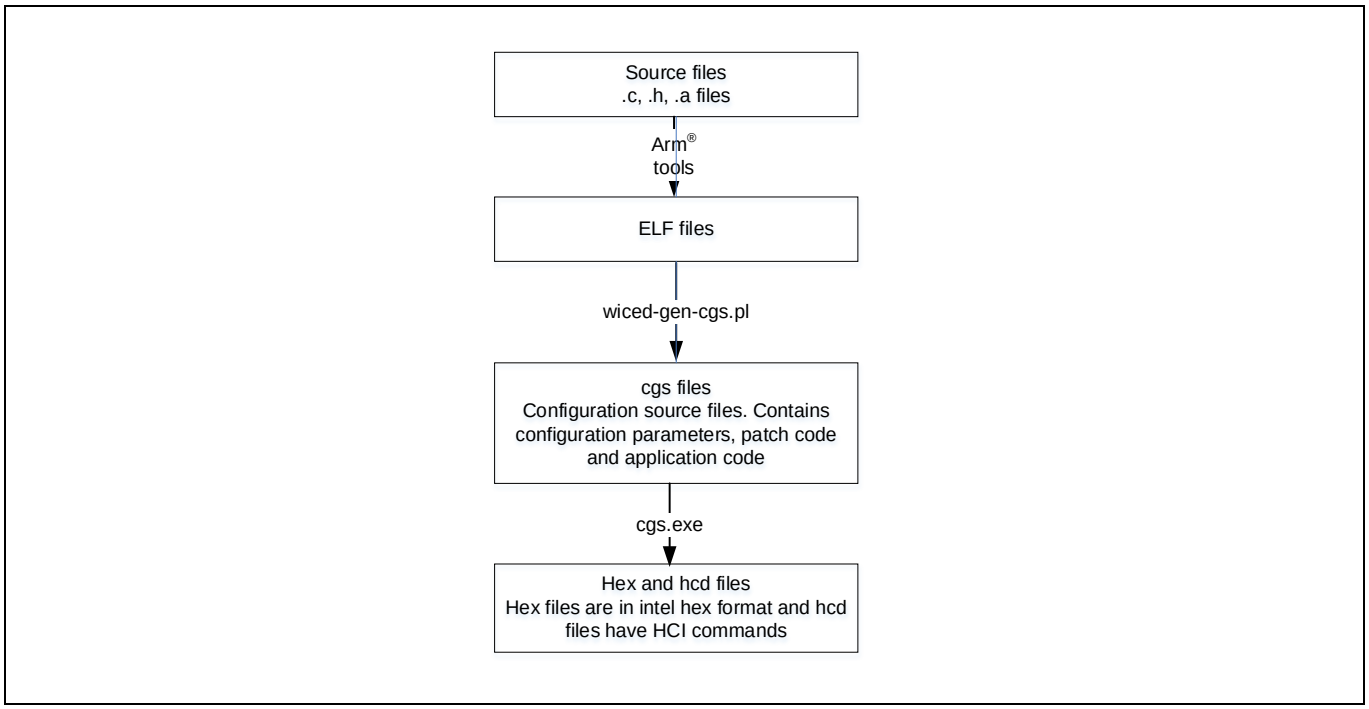

Figure 7　ファイルフォーマット

6.5.1.1　*.hcd 形式の詳細

hcd ファイルは、HCI コマンドとして直接解析および解釈できるバイナリファイルです。hcd ファイルには、ホストが追加する必要のあるパケットタイプを除くすべてのコマンドがバイナリ形式で含まれます。*hcd* ファイルの形式は次のとおりです。

- 最初の 2 バイトはコマンド ID です。例えば、HCI_WRITE コマンドは 0x4C、0xFC で表されます。
- 次のバイトはコマンドペイロードの長さです。例えば、0x06 はコマンドを完了するためにさらに 6 バイトが続くことを示します。
- コマンドのペイロード。

ファイルを HCI コマンドに変換するためには、HCI パケットタイプのみを追加する必要があります。例えば、ホストからコマンドを送信する場合、0x01 は HCI コマンドの前に配置して、イベントではなくコマンドであることを示す必要があります。

次の WRITE_RAM コマンドは 1 つの例です。

01 4C FC nn xx xx xx xx yy yy yy…

この WRITE_RAM コマンドでは以下のとおりです。

- nn は 4 + N で、4 つのアドレスバイトと N ペイロードバイトを表します。
- xx xx xx xx は、4 バイトの絶対 RAM アドレスです。
- yy yy yy…は、アドレス指定された RAM ロケーションにロードされる N ペイロードバイトです。

各 WRITE_RAM コマンドに対する次の応答は、200 ミリ秒以内と予想されます。: 04 0E 04 01 4C FC 00

6.5.1.2 *.hex 形式の詳細

.hex ファイルは、Intel I32HEX の規則に従います。形式は、ASCII キャリッジリターンとラインフィード (0x0D、0x0A) で区切られたレコードで構成されます。*.hex* ファイルの形式は次のとおりです。

- 最初の文字は開始コード 「:」 (ASCII 0x3A) です
- 次のバイトはペイロードの長さを示します。例えば、「FF」は 255 を示します。
- 次の 4 バイトは 16 アドレスビットを示します。例えば、「1000」はアドレス 0x1000 を表します。アドレスのタイプは、コマンドのタイプによって異なります。
- 次のバイトはコマンドのタイプを示します。
  - 「00」はデータ用で、アドレスフィールドは宛先アドレスの下位 16 ビットを表します。
  - 「01」はファイルの終わりを示します。ペイロードは 0 で、アドレスフィールドは「0000」に設定されます。
  - 「04」は拡張アドレスを示します。アドレスフィールドはアドレスの下位 16 ビットを示し、ペイロードはアドレスの上位 16 ビットを表します。
  - 「05」は 32 ビットのアドレスペイロードの場合です。アドレスフィールドは「0000」に設定され、ペイロード長は「04」に設定され、ペイロードは 32 ビットアドレスとして解釈されます。これは、LAUNCH_RAM (後述) の宛先を示すためによく使用されます。

6.5.1.3 *.cgs 形式の詳細

.cgs ファイルには、設定パラメーター, パッチコード, およびアプリケーションコードが含まれます。アプリケーションのソースコードから生成された*.elf* ファイルは*.cgs* 形式に変換され、*\<mtw_path>\wiced_btsdk\dev-kit\baselib\208XXA1\internal\<device>\patch\patch.cgs* にあるパッチ*.cgs* ファイルに追加されます。

.cgs ファイル内の設定パラメーターの定義は、*\<mtw_path>\wiced_btsdk\dev-kit\baselib\208XXA1\internal\<device>\configdef<device>.hdf* にあります。

Note: <device> - ディレクトリパス内のデバイスは、20819A1 または 20820A1 を表します。

水晶周波数, UART 設定, BD_ADDR, ローカル名, および RSSI 設定などのパラメーターは、パッチ.cgs ファイルを使用して変更できます。

6.5.2 ビルド中のファイル生成

ファイル変換プロセスは次のとおりです。

1. ソースファイル (*.c*、*.h*) はコンパイルされ、GNU ツールを使用して*.o* (オブジェクト) ファイルに変換されます。
2. ライブラリのソースは、ライブラリの*.a* ファイルにリンクされています。
3. これらは、事前にビルドされたライブラリファイルとともに、GNU ツールによって.elf (実行可能でリンク可能な形式) ファイルにリンクされます。
4. *.elf* ファイルは、*\<mtw_path>\wiced_btsdk\dev-kit\baselib\<device>\make\scripts\wiced-gen-cgs.pl* スクリプトを使用して*.cgs* (設定ソース) ファイルに変換されます。ROM パッチコードとプラットフォーム*.cgs* ファイルも追加されます。
5. 次に、*\<mtw_path>\wiced_btsdk\dev-kit\btsdk-tools\<OS>\CGS\cgs.exe* ツールを使用して、*.cgs* ファイルを Intel hex 形式に変換します。*cgs.exe* アプリケーションは、前のステップで作成された*.cgs* ファイルや*.btp* ファイルなどの入力を受け取ります。cgs.exe アプリケーションは、BD ADDRESS などのランタイム引数を取り、*.btp* ファイルの BD ADDRESS を上書きできます。これは、異なるアドレスで複数のデバイスをプログラミングするときに役立ちます。

6. hex ファイルは、ツール *\<mtw_path>\wiced_btsdk\dev-kit\btsdk-tools\<OS>\IntelHexToBin\IntelHexToHCD.exe* を使用して *hcd* ファイルに変換されます。

6.5.3 リカバリプロセス

場合によっては、CYW20819/CYW20820 デバイスが unknown な状態になるか、HCI ボーレートがデフォルトから変更された可能性があります。このような場合、HCI UART を使用してデバイスをプログラムできません。デバイスを正しくプログラムするためには、デバイスをリカバリモードにする必要があります。これは、デバイスのリセット中に HCI UART CTS ラインをアサートすることによって行われます。

Bluetooth®デバイスはリセットされると、電源投入時に HCI UART CTS ピン (リカバリピン) のステータスをチェックします。リセット中にリカバリピンがアサート (LOW) されると、デバイスはいわゆるリカバリモードに入ります。このモードは、HCI_RESET コマンドのビットパターンについて RX ラインをチェックすることにより、UART ボーレートの検出を試みます。検出されると、HCI_RESET 応答は同じボーレートで与えられます。

このモードでは、ほとんどの HCI コマンドは応答しません。このモードでデバイスにダウンロードするためには、HCI_DOWNLOAD_MINIDRIVER に対する「応答なし」を無視して、ダウンロード手順を続行します。リセット後に CTS ピンが High の場合、デバイスは OCF をチェックし、保存されている設定を適用します。通常、デフォルトのボーレートですべての HCI コマンドを受け入れる準備ができたモードで終了します。設定が利用できない場合、デバイスは自動ボーモードに入ります。アプリケーションコードは SRAM に直接ロードして実行することも、シリアルフラッシュに実行して、次回の起動時に実行もできます。

6.5.4 ミニドライバー

ミニドライバーは、ROM コードから独立した別の FW です。アプリケーションコードをオンチップフラッシュにダウンロードするために使用されます。ミニドライバーには、WRITE_RAM や READ_RAM などの特定の HCI コマンドを解釈し、それらのコマンドを実行するためのコードが含まれます。オンチップフラッシュにコードをダウンロードするためには、最初に SRAM からミニドライバーを作成して実行する必要があります。これにより、HCI コマンドがさらに解釈され、コードがフラッシュの正しい場所に書き込まれます。コードのダウンロードが完了するとすぐに、ミニドライバーは不要になったため破棄されます。ミニドライバーは、インターフェースが UART から SPI などに変更された場合に異なります。デフォルトのミニドライバーは*\<mtw_path>\wiced_btsdk\dev-kit\baselib\<device>\platforms\minidriver-<device>-uart-patchram.hex* にあります。

6.6 プログラミング用ツール

ダウンロードプロセスと HCI コマンドフローの詳細を理解するためには、GitHub **btsdk-docs** リポジトリの *WICED-HCI-Control-Protocol.pdf* を参照してください。

6.6.1 ClientControl

ClientControl は、PC で Bluetooth®ホストとして機能する AIROC™アプリケーションです。HCI UART を使用して Bluetooth®デバイスに接続し、AIROC™ HCI プロトコルを介して通信します。このアプリケーションを使用して、アドバタイズの開始/停止, 接続要求の送信, HID プロファイルのテストなどのコマンドを送信し、ファームウェアをデバイスの SRAM にダウンロードすることで、CYW20819/CYW20820 デバイスをテストできます。このツールはデバイスの SRAM にダウンロードするため、ミニドライバーは必要ありません。

Windows, macOS, および Linux 用の ClientControl アプリケーションを起動するためには、ModusToolbox™ で Bluetooth®-SDK アプリケーションを選択し、**Quick Panel** > **Tools** > **ClientControl** をクリックしてください。

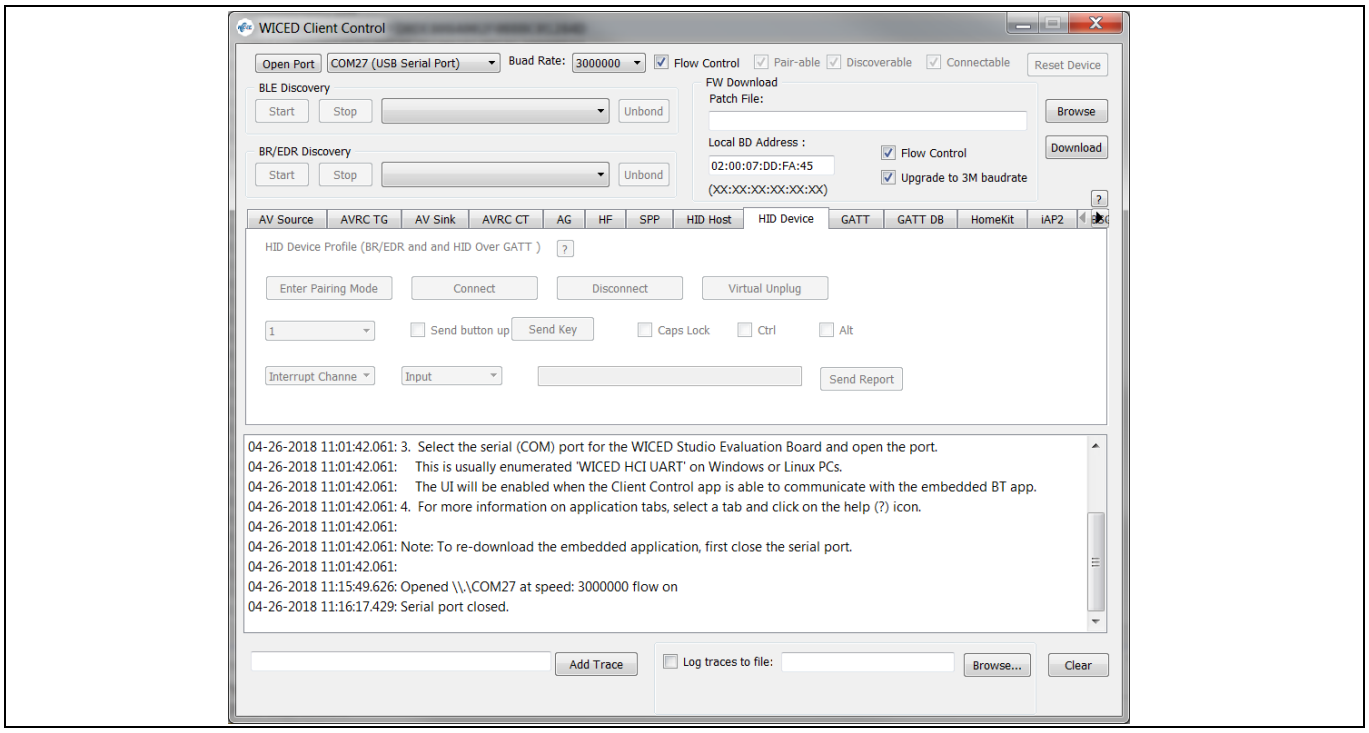

Figure 8 ClientControl ウィンドウ

6.6.2 ChipLoad

ChipLoad は、Bluetooth®デバイスへファームウェアのダウンロードに使用されるコマンドラインユーティリティです。*.hex* ファイルや*.btp* ファイルなどの入力を使用して、ダウンロードデータとダウンロード設定をそれぞれ取得します。*\<mtw_path>\wiced_btsdk\dev-kit\btsdk-tools\<OS>\ChipLoad* にあります。

Note: ChipLoad ツールは Intel hex 形式をサポートしています。

6.6.3 DetectAndId

DetectAndId ツールは、検出と ID の 2 つのタスクを実行します。 最初に、このツールはインフィニオン Bluetooth®デバイスが存在するシリアルポートを検出しようとします。これは、さまざまなボーレートで利用可能なポートに HCI_RESET コマンドを送信し、適切な応答を待つことによって行われます。シリアルポートがこのツールへの入力として既に指定されている場合、ポート検出部分はスキップされます。

2 番目のタスクは、接続されたデバイスの ID です。これは、チップ ID を読み出すために AIROC™ HCI コマンドを送信することによって行われます。受信した ID は、このパス*<mtw_path>\wiced_btsdk\dev-kit\baselib\<device>\platforms\CYW20819/CYW20820A1_IDFILE.txt* のチップ固有 ID ファイルに存在する予期されたチップ ID と比較されます。これにより、適切なデバイスのみがプログラムされます。これは、他のツールがアプリケーションコードをダウンロードするために使用できます。

6.6.4 インフィニオンプログラマ

インフィニオンプログラマは、CYW20819/CYW20820 ベースの評価ボードのプログラミングに使用できる GUI アプリケーションです。CYW20819/CYW20820 のフラッシュをプログラム, 消去, および検証するオプションを提供します。インフィニオンプログラマは、Bluetooth®アプリケーションのビルドプロセスによって生成される**.hex* ファイルを想定しています。HCI UART ポートを選択すると、ボードをプログラムできます。詳細については、インフィニオンプログラマリファレンスガイドとリリースノートを参照してください。

7 まとめ

CYW20819/CYW20820 は、モノのインターネット (IoT) デバイス用のワイヤレス入力, ウェアラブル, 医療, およびビーコンのアプリケーションに最適なソリューションです。コスト重視のデバイスの設計を容易にするために、Arm® Cortex®-M4 プロセッサは、緊密に結合されたシステムコンポーネントを実装し、Bluetooth®パケットの処理と周辺システムの機能を大幅に改善しながら、プロセッサ領域を削減します。同じシャーシを維持しながら、複数の市場に対応できる独自の差別化機能を提供します。このガイドは、CYW20819/CYW20820 デバイスの機能とペリフェラルを探索するプログラマに役立ちます。このガイドでは、さまざまな CYW20819/CYW20820 SoC ペリフェラルの使用方法と、アプリケーションの開発および製品の設計時にハードウェアレベルの制約を克服する方法について説明します。

8 関連資料

アプリケーションノート	
AN225684 - Getting started with CYW20819/CYW20820	CYW20819/CYW20820 Bluetooth® SoC、および ModusToolbox™ IDE のデバイスを使用して最初の Bluetooth® Low Energy アプリケーションを構築する方法について説明します
AN225948 - CYW20819/20820 hardware design guideline	このドキュメントには、ボードスタックアップとインピーダンス制御要件を含む基本的なレイアウトガイドライン, 重要なコンポーネントのコンポーネント配置と推奨部品, ルーティングガイドライン, およびさまざまな電源トレースの推奨トレース幅が含まれます。
AN225270 - CYW20819/CYW20820 low power guideline	このドキュメントは、CYW20819/CYW20820 を使用した低消費電力アプリケーションの設計と開発に関するガイドラインと、電力効率の良い設計を検討するためのパラメーターを提供します。
AN216403 - Application buffer pools	このドキュメントでは、アプリケーションで使用されるバッファと、AIROC™ Bluetooth®スタックの上位層スタックについて説明します。
AN214852 - Collaborative Coexistance interface between Infineon-to-Infineon solution and Infineon-to-third-party chips	このアプリケーションノートでは、CYW43XX のコラボレーティブ共存ハードウェアメカニズムとアルゴリズム、およびインフィニオンからインフィニオンへのソリューションまたはインフィニオン間の COEX 配線をサードパーティのチップに接続する方法について説明します。
サンプルコード	
ModusToolbox™ IDE を使用したサンプルコードの包括的なコレクションについては、**GitHub リポジトリ**にアクセスしてください	
デバイスと評価キットのドキュメント	
CYW20819 デバイスデータシート	**CYW920819M2EVB-01 evaluation kit**
CYW20820 デバイスデータシート	**CYW920820M2EVB-01 evaluation kit**
ツールのドキュメント	
ModusToolbox™ IDE	IoT デザイナー向けの IDE
インフィニオンプログラマのドキュメント	このページでは、インフィニオンプログラマ GUI ユーザーガイドとインフィニオンプログラマのリリースノートを提供します。

改訂履歴

版数	発行日	変更内容
**	2021-02-04	本版は英語版 002-26546 Rev. *A について、CYPRESS DEVELOPER COMMUNITY の参画者によって日本語に翻訳されたドキュメントです。
*A	2021-05-26	テンプレートの変更を実施。 これは英語版 002-26546 Rev. *B を翻訳した日本語版 Rev. *A です。
*B	2023-01-19	これは英語版 002-26546 Rev. *C を翻訳した日本語版 Rev. *B です。

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-01-19
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
002-32491 Rev. *B

重要事項

本手引書に記載された本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。
インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。