Bluetooth LE マイコン／モジュール

Windows 10 Bluetooth LEアプリケーション

要旨

本アプリケーションノートは、ルネサスエレクトロニクス製 Bluetooth® Low Energy 技術対応マイコンやモジュールを搭載した評価ボードと無線通信することができるWindows® 10 Bluetooth LEアプリケーションを通して、Windows 10 の Bluetooth LE API の動作や処理の流れを解説します。

Windows 10 で動作する Bluetooth LE アプリケーションは、Microsoft Visual Studio Express 2017 for Windows Desktop で.NET Framework の Windows Presentation Foundation (WPF)アプリケーションとして作成され、ペアリング、コネクション、サービスやキャラクタリスティックの検索、データ通信を行います。

動作確認デバイス

Target Board for RX23W
Target Board for RX23W module
EK-RA4W1
EB-RE01B
RL78/G1D 評価ボード (RTK0EN0001D01001BZ)

関連ドキュメント

RX23W グループ

- RX23W グループ Target Board for RX23W クイックスタートガイド (R20QS0014)
- RX23W グループ Target Board for RX23W ユーザーズマニュアル (R20UT4634)
- RX23W グループ Target Board for RX23W module クイックスタートガイド (R20QS0022)
- RX23W グループ Target Board for RX23W module ユーザーズマニュアル (R20UT4890)

RA4W1 グループ

- RA4W1 Group Evaluation Kit for RA4W1 Microcontroller Group EK-RA4W1 Quick Start Guide (R20QS0015)
- RA4W1 グループ EK-RA4W1 ユーザーズマニュアル (R20UT4683)

RE01B グループ

- EB-RE01B ハードウェアマニュアル (TS-TUM09734) (TESSERA TECHNOLOGY INC.)
- RE01B グループ Bluetooth Low Energy サンプルコード (using CMSIS Driver Package) (R01AN5606)

RL78/G1D グループ

- Bluetooth® Low Energy プロトコルスタック 仮想 UART アプリケーション (R01AN3130)
- Bluetooth® Low Energy プロトコルスタック クイックスタートガイド (R01AN2767)
- RL78/G1D 評価ボード ユーザーズマニュアル (R30UZ0048)

目次

1. 概要

本アプリケーションノートは、ルネサスエレクトロニクス製 Bluetooth® Low Energy 技術対応マイコンやモジュールを搭載した評価ボードと無線通信することができる Windows 10 Bluetooth LE アプリケーション(以降、Windows アプリケーション)を通して、Windows 10 の Bluetooth LE API の動作や処理の流れを解説します。

評価ボードと通信する 2 つの Windows アプリケーションを以下に示します。

- LED Switch Service Client

  RX23W グループ／RA4W1 グループ／RE01B グループの評価ボードと通信します。

- Virtual UART Client

  RL78/G1D グループの評価ボードと通信します。

Windows アプリケーションと通信する評価ボードやソフトウェアを以下に示します。

- RX23W グループ
  - 評価ボード
    - Target Board for RX23W
    - Target Board for RX23W module
  - ソフトウェア
    - LED Switch サービス (評価ボードの出荷時に書き込まれているソフトウェア)

- RA4W1 グループ
  - 評価ボード
    - EK-RA4W1
  - ソフトウェア
    - LED Switch サービス (評価ボードの出荷時に書き込まれているソフトウェア)

- RE01B グループ
  - 評価ボード
    - EB-RE01B
  - ソフトウェア
    - GATT Server
      (LED Switch サービスと同機能)
      (「RE01B グループ Bluetooth Low Energy サンプルコード (using CMSIS Driver Package)」(R01AN5606)に同梱されているソフトウェア)

- RL78/G1D グループ
  - 評価ボード
    - RL78/G1D 評価ボード(RTK0EN0001D01001BZ)
  - ソフトウェア
    - 仮想 UART アプリケーション
      (「Bluetooth® Low Energy プロトコルスタック 仮想 UART アプリケーション」 (R01AN3130) に同梱されているソフトウェア (文字データ送受信を使用します))

1.1 LED Switch Service Client

LED Switch Service Client の Windows アプリケーションは、RX23W グループ／RA4W1 グループ／RE01B グループの評価ボードで動作する LED Switch Service と通信を行うことができます。Pairing で Windows のシステムに評価ボードの情報を登録し、評価ボード上の LED 点滅間隔の制御や、スイッチが押された回数のカウントを行います。

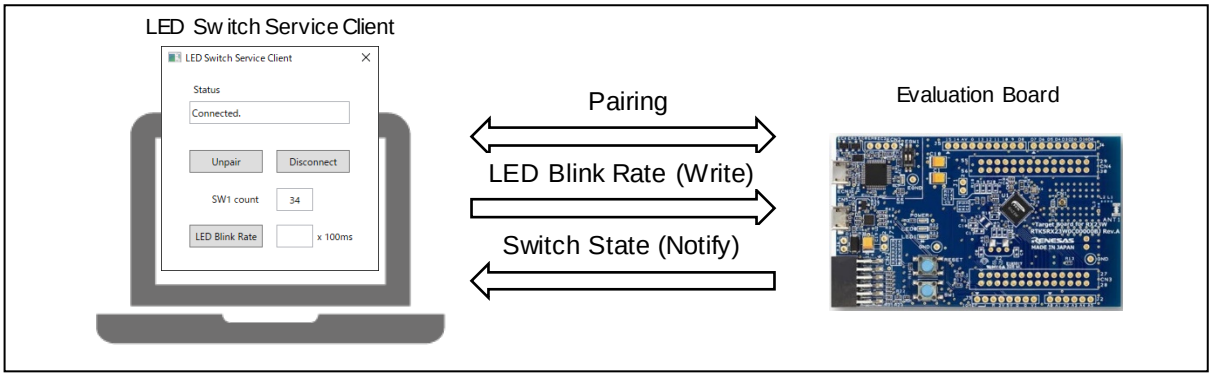

図 1-1 LED Switch Service Client

1.2 Virtual UART Client

Virtual UART Client の Windows アプリケーションは、RL78/G1D グループの評価ボードで動作する仮想 UART アプリケーションと通信を行うことができます。Windows アプリケーションと、評価ボードに USB 接続された PC 上のターミナルソフトで文字データの送受信を行います。

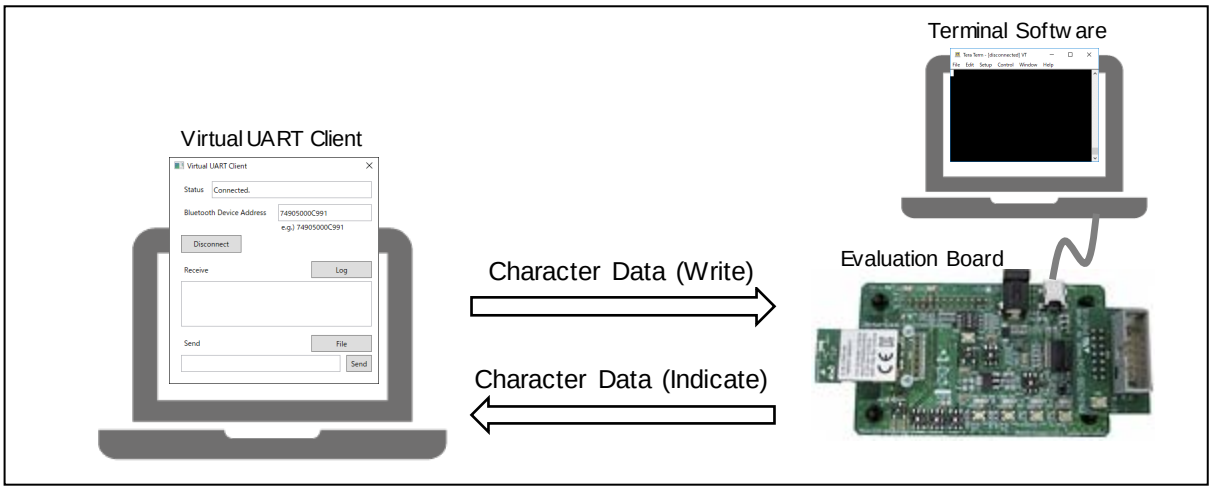

図 1-2 Virtual UART Client

2. 動作確認

2.1 実行ファイル

本アプリケーションノートに含まれる Windows アプリケーションの実行ファイルを使用して、評価ボードとの通信動作を確認することができます。

2.1.1 LED Switch Service Client

LED Switch Service Client の実行ファイルを表 2-1に、使用できる評価ボードを表 2-2に示します。

表 2-1 LED Switch Service Client 実行ファイル

フォルダ名	ファイル名
r01an6004jj0100-ble-mcu-win¥LED_Switch_Service_Client¥exe	LED_Switch_Service_Client.exe

表 2-2 LED Switch Service Client で使用できる評価ボード

Windows アプリケーションファイル名	評価ボード
LED_Switch_Service_Client.exe	・Target Board for RX23W ・Target Board for RX23W module ・EK-RA4W1 ・EB-RE01B

2.1.2 Virtual UART Client

Virtual UART Client の実行ファイルを表 2-3に、使用できる評価ボードを表 2-4に示します。

表 2-3 Virtual UART Client 実行ファイル

フォルダ名	ファイル名
r01an6004jj0100-ble-mcu-win¥Virtual_UART_Client¥exe	Virtual_UART_Client.exe

表 2-4 Virtual UART Client で使用できる評価ボード

Windows アプリケーションファイル名	評価ボード
Virtual_UART_Client.exe	・RL78/G1D 評価ボード(RTK0EN0001D01001BZ)

2.2 Windows アプリケーション仕様

LED Switch Service Client と Virtual UART Client の仕様について説明します。

2.2.1 LED Switch Service Client

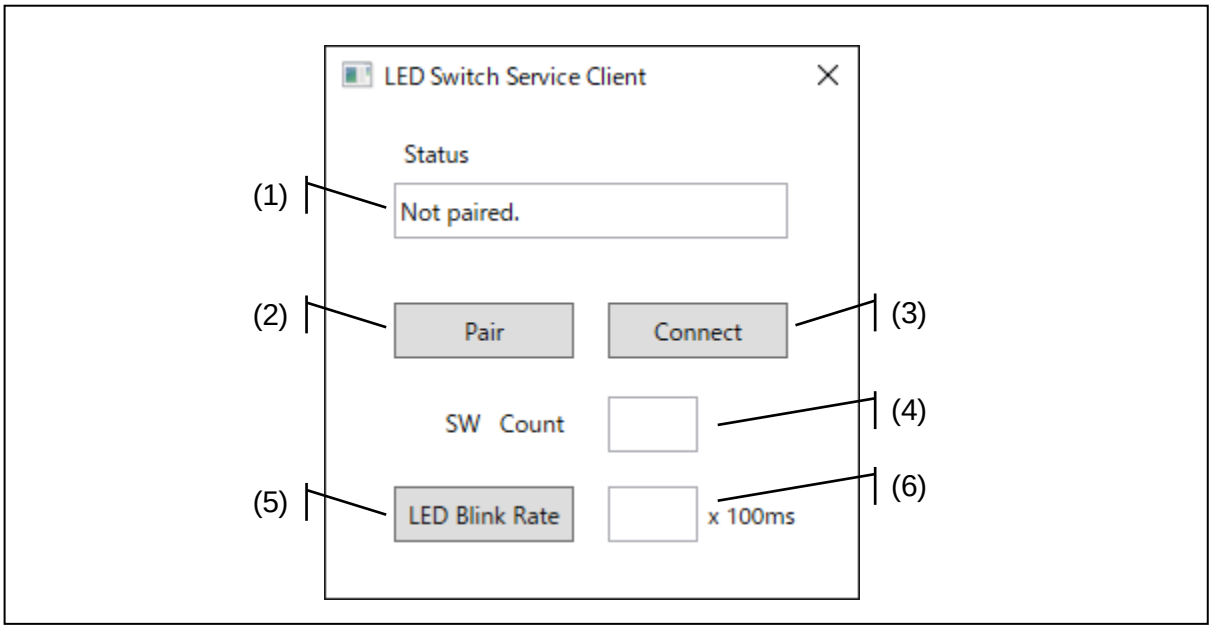

図 2-1 LED Switch Service Client 仕様

(1) ステータス表示テキストボックス

アプリケーションの動作状態を表示します。

(2) Pair/Unpair ボタン

評価ボードとペアリングされていない時、Pair ボタンとして動作し「Pair」と表示されます。ボタンをクリックすると評価ボードとペアリングします。

評価ボードとペアリングされている時、Unpair ボタンとして動作し「Unpair」と表示されます。ボタンをクリックすると評価ボードとのペアリングを解除します。

(3) Connect/Disconnect ボタン

評価ボードと未接続の時、Connect ボタンとして動作し「Connect」と表示されます。ボタンをクリックすると評価ボードと接続します。

評価ボードと接続されている時、Disconnect ボタンとして動作し「Disconnect」と表示されます。ボタンをクリックすると評価ボードとの接続を切断します。

(4) SW Count テキストボックス

接続している評価ボードが送信した Notification の受信回数を表示します。

(5) LED Blink Rate 送信ボタン

LED Blink Rate 入力テキストボックスに入力された時間を、接続している評価ボードに送信します。

(6) LED Blink Rate 入力テキストボックス

LED Blink Rate の時間を入力します。

2.2.2 Virtual UART Client

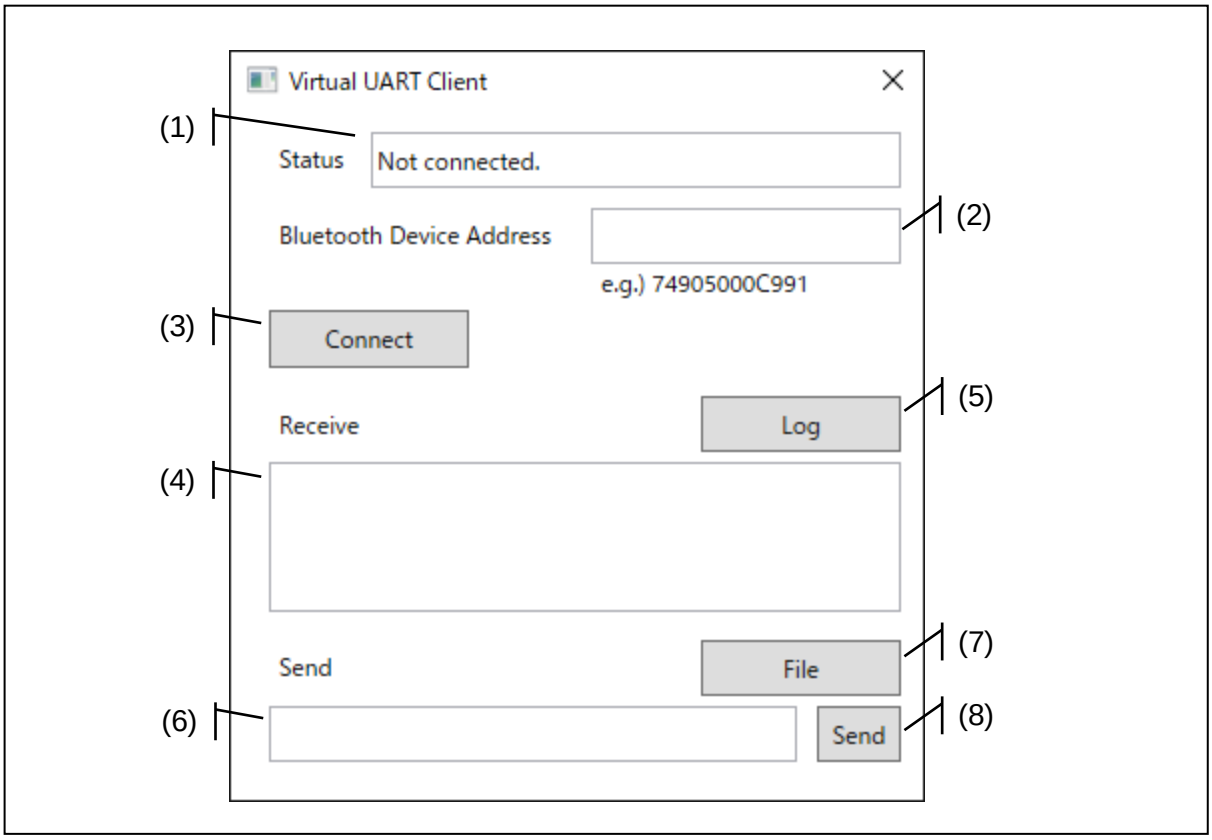

図 2-2 Virtual UART Client 仕様

(1) ステータス表示テキストボックス

アプリケーションの動作状態を表示します。

(2) Bluetooth Device Address テキストボックス

空欄: Connect ボタンをクリックし仮想 UART プロファイルの UUID を持つ評価ボードと接続した後、Bluetooth Device Address を表示します。

入力: Connect ボタンをクリックすると入力されている Bluetooth Device Address の評価ボードと接続します。

(3) Connect/Disconnect ボタン

評価ボードと未接続の時、Connect ボタンとして動作し「Connect」と表示されます。ボタンをクリックすると評価ボードと接続します。

評価ボードと接続している時、Disconnect ボタンとして動作し「Disconnect」と表示されます。ボタンをクリックすると評価ボードとの接続を切断します。

Bluetooth Device Address テキストボックスが空欄:

Connect ボタンをクリックすると仮想 UART プロファイルの UUID を持つ評価ボードと接続します。

Bluetooth Device Address テキストボックスに Bluetooth Device Address が記入されている:

Connect ボタンをクリックすると記入されている Bluetooth Device Address の評価ボードと接続します。

(4) Receive データテキストボックス

接続している評価ボードが送信した文字データを受信し表示します。

(5) Receive データ保存ボタン

接続している評価ボードが送信した文字データを受信しファイルに保存します。

ファイルに保存中は「Logging」と表示されます。

(6) Send データテキストボックス

接続している評価ボードに送信する文字データを入力します。

入力可能な文字データは 20 文字(20 バイト)です。

(7) Send ファイルボタン

ファイルの中の文字データを接続している評価ボードに送信します。

(8) Send データボタン

Send データ入力テキストボックスに入力された文字データを、接続されている評価ボードに送信します。

2.3　動作確認環境の準備

動作確認を行なう環境として、Windows 10 が動作する PC と表 2-2、表 2-4で示す評価ボードの中のいずれか一つを準備します。また、各評価ボードに合わせて使用するソフトウェアを書き込んでください。

2.3.1　Windows PC

Windows アプリケーションの動作を確認した Windows 10 PC の情報を表 2-5に示します。Windows 10 PC は Bluetooth 4.0 以降の Bluetooth 機能を搭載した PC を使用してください。

Windows 10 の状態によってはWindows アプリケーションが動作しないことがあります。この場合は「5. 注意事項」を参照してください。

表 2-5 Windows 情報

エディション	Windows 10 Home
バージョン	21H1
OS ビルド	19043.1110

2.3.2　Target Board for RX23W

Target Board for RX23W では「出荷時ソフトウェア(LED Switch サービス)」を使用します。Target Board for RX23W に他のプログラムを書き込んでいる場合は、以下のドキュメントを参照して「出荷時ソフトウェア」へ復元してください。

- RX23W グループ Target Board for RX23W クイックスタートガイド (R20QS0014)
  - 5.1 出荷時ソフトウェアへの復元

2.3.3　Target Board for RX23W module

Target Board for RX23W module では「出荷時ソフトウェア(LED Switch サービス)」を使用します。Target Board for RX23W module に他のプログラムを書き込んでいる場合は、以下のドキュメントを参照して「出荷時ソフトウェア」へ復元してください。

- RX23W グループ Target Board for RX23W module クイックスタートガイド (R20QS0022)
  - 5.1 出荷時ソフトウェアへの復元

2.3.4　EK-RA4W1

EK-RA4W1 では「factory software (LED Switch Service)」を使用します。EK-RA4W1 に他のプログラムを書き込んでいる場合は、以下のドキュメントを参照して「factory software」へ復元してください。

- RA4W1 Group Evaluation Kit for RA4W1 Microcontroller Group EK-RA4W1 – Quick Start Guide (R20QS0015)
  - 6. Restoring Factory Settings

2.3.5　EB-RE01B

EB-RE01B では「GATT Server」を使用します。以下のドキュメントを参照して「GATT Server」のプログラムを書き込んで下さい。

- RE01B グループ Bluetooth Low Energy サンプルコード (using CMSIS Driver Package) (R01AN5606)
  - 2.4 ファームウェア書き込み

2.3.6　RL78/G1D 評価ボード(RTK0EN0001D01001BZ)

RL78/G1D 評価ボードでは「仮想 UART アプリケーション(文字データ送受信プログラム)」を使用します。以下のドキュメントを参照して「仮想 UART アプリケーション(文字データ送受信プログラム)」のプログラムを書き込んで下さい。

- Bluetooth® Low Energy プロトコルスタック 仮想 UART アプリケーション (R01AN3130)
  - 7.3 実行環境の準備
- Bluetooth® Low Energy プロトコルスタック クイックスタートガイド (R01AN2767)
  - 5. プログラムを書き込む

仮想 UART アプリケーションの文字データ送受信は「レスポンス有り通信」で行います。評価ボード上のディップスイッチ 6-4 を表 2-6のように設定してください。ディップスイッチの詳細については以下のドキュメントを参照してください。

- Bluetooth® Low Energy プロトコルスタック 仮想 UART アプリケーション (R01AN3130)
  - 3.2 データ送受信におけるレスポンス有無の選択

表 2-6 RL78/G1D 評価ボードのディップスイッチ設定

番号	スイッチ	説明
SW6-4	OFF (左側)	レスポンス有り通信 クライアント→サーバ: Write Request サーバ→クライアント: Indication

送信する文字データの入力や受信した文字データの表示は、評価ボードに接続した PC 上のターミナルソフトで行います。以下のドキュメントを参照して、評価ボードと PC の接続と、ターミナルソフトの設定を行ってください。(以下のドキュメントでは 2 台の評価ボードを準備するように記述されていますが、本アプリケーションノートでは 1 台のみ準備してください。)

- Bluetooth® Low Energy プロトコルスタック 仮想 UART アプリケーション (R01AN3130)
  - 7.3 実行環境の準備

2.4　動作確認

2.4.1　LED Switch Service Client を使用した動作確認

(1) 評価ボードに電源を投入します。電源を投入すると自動的にアドバタイジングが開始されます。評価ボードへの電源投入方法は、使用する評価ボードに合わせて以下のドキュメントを参照してください。

- RX23W グループ Target Board for RX23W クイックスタートガイド (R20QS0014)
  - 2.1 電源投入
- RX23W グループ Target Board for RX23W module クイックスタートガイド (R20QS0022)
  - 2.1 電源投入
- RA4W1 Group Evaluation Kit for RA4W1 Microcontroller Group EK-RA4W1 Quick Start Guide (R20QS0015)
  - 3.1 Connecting and Powering Up the EK-RA4W1 Board
- RE01B グループ Bluetooth Low Energy サンプルコード (using CMSIS Driver Package) (R01AN5606)
  - 2.2 起動方法

(2) 表 2-1の LED Switch Service Client を起動します。起動すると評価ボードとのペアリング情報が Windows のシステムに登録されているかチェックします。

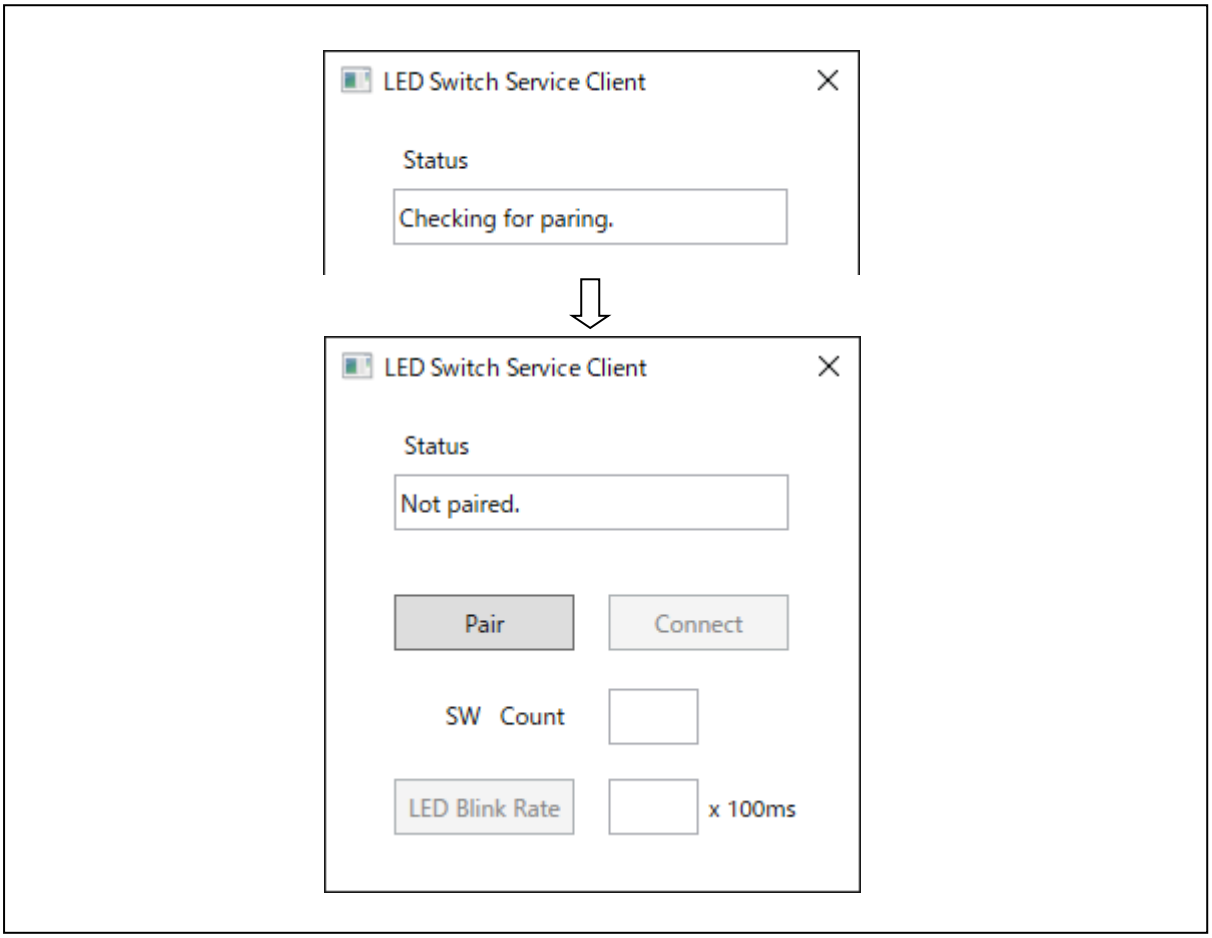

図 2-3 LED Switch Service Client の動作確認 (1)

(3) [Pair]ボタンをクリックし評価ボードとペアリングを行います。
ペアリングが成功すると[Connect]ボタンが有効になります。また、Windows の[設定] - [デバイス] - [Bluetooth とその他のデバイス]に、評価ボード(RBLE-DEV)がペアリング済みのデバイスとして表示されます。
評価ボードが見つからない場合やペアリングが失敗する場合、評価ボードに電源を投入してから一定時間が経過しアドバタイジングの送信間隔が長くなることで発見しにくくなっていることがあります。アドバタイジングの送信間隔を短くするために評価ボードの Reset Switch を押して再起動してください。

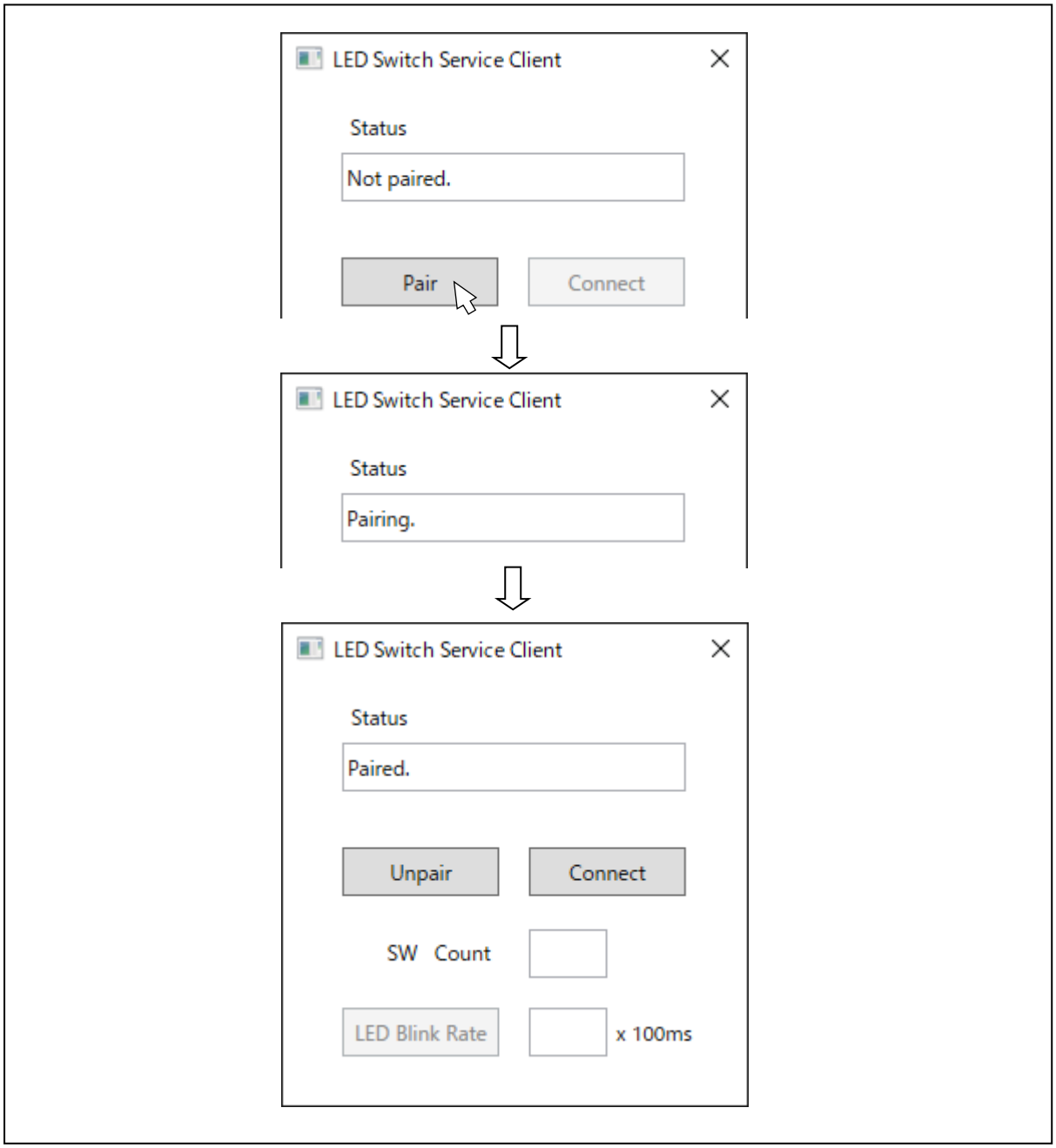

図 2-4 LED Switch Service Client の動作確認 (2)

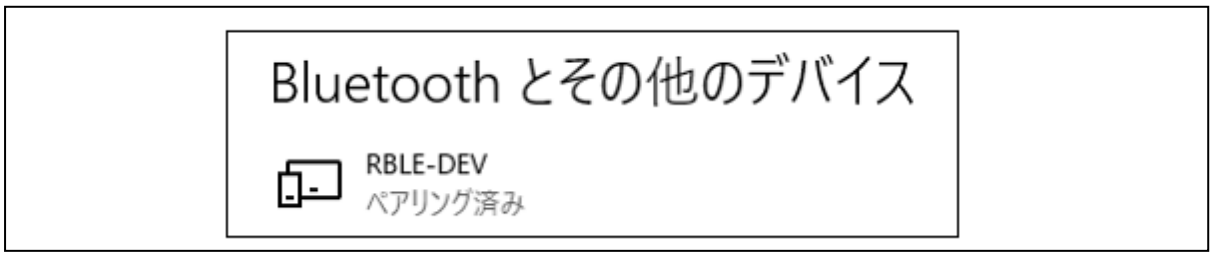

図 2-5 LED Switch Service Client の動作確認 (3)

(4) [Connect]ボタンをクリックし評価ボードと接続します。接続すると、[LED Blink Rate]ボタンが有効になります。

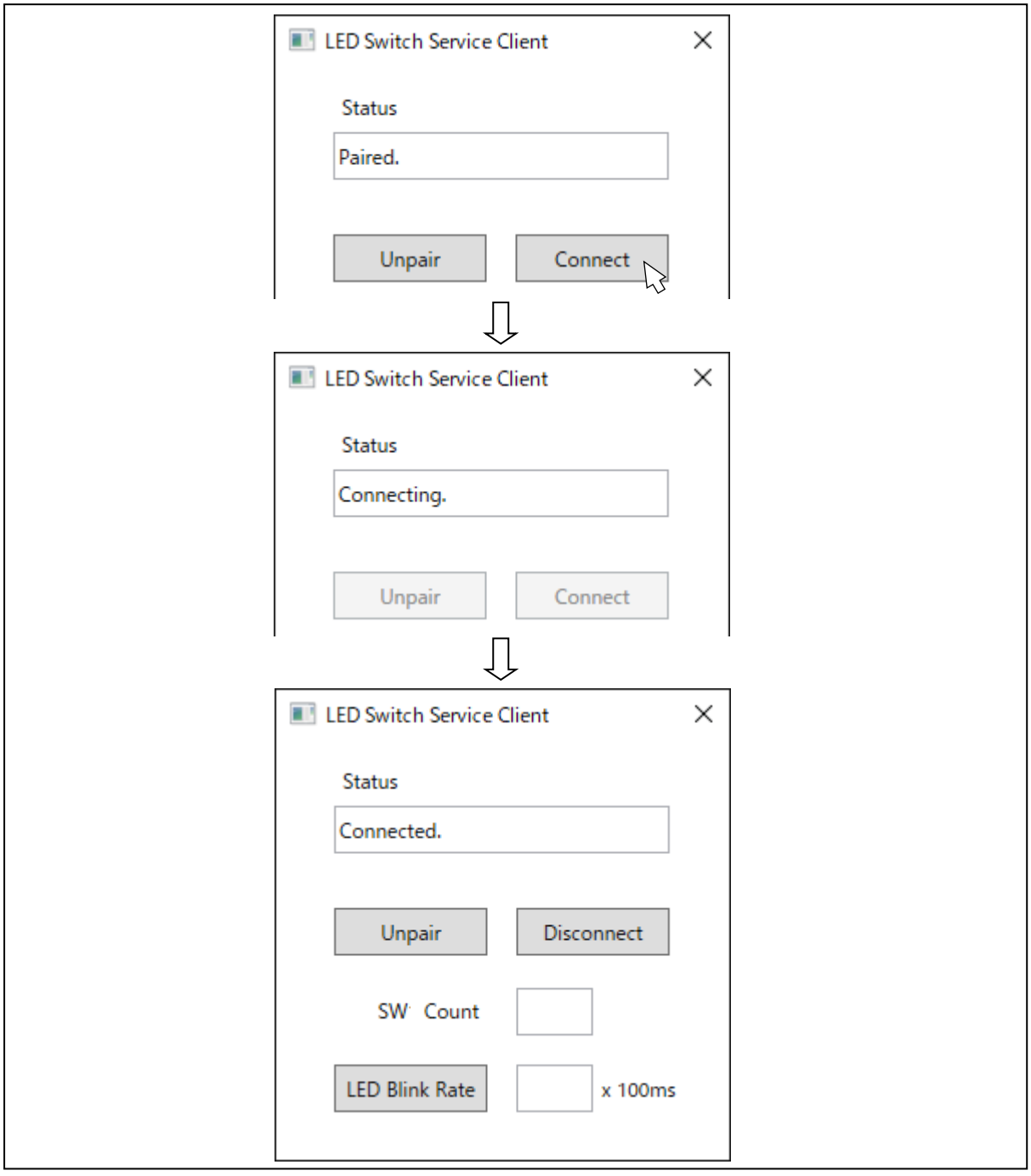

図 2-6 LED Switch Service Client の動作確認 (4)

(5) 評価ボードの SW を押すと Notification が送信され､SW Count テキストボックスにカウントされます。(Target Board for RX23W/Target Board for RX23W module/EK-RA4W1 は SW1、EB-RE01B は SW2 を押下してください。)

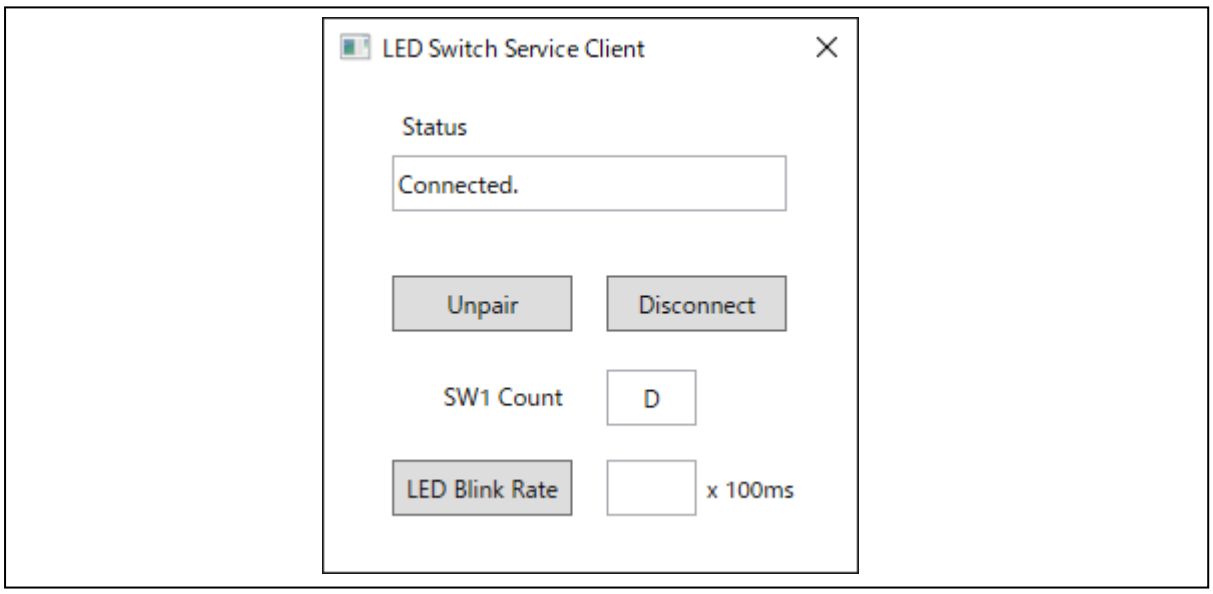

図 2-7 LED Switch Service Client の動作確認 (5)

(6) LED Blink Rate テキストボックスに LED の点滅間隔を入力し、[LED Blink Rate]ボタンをクリックします。評価ボードの LED の点滅速度が変化します。

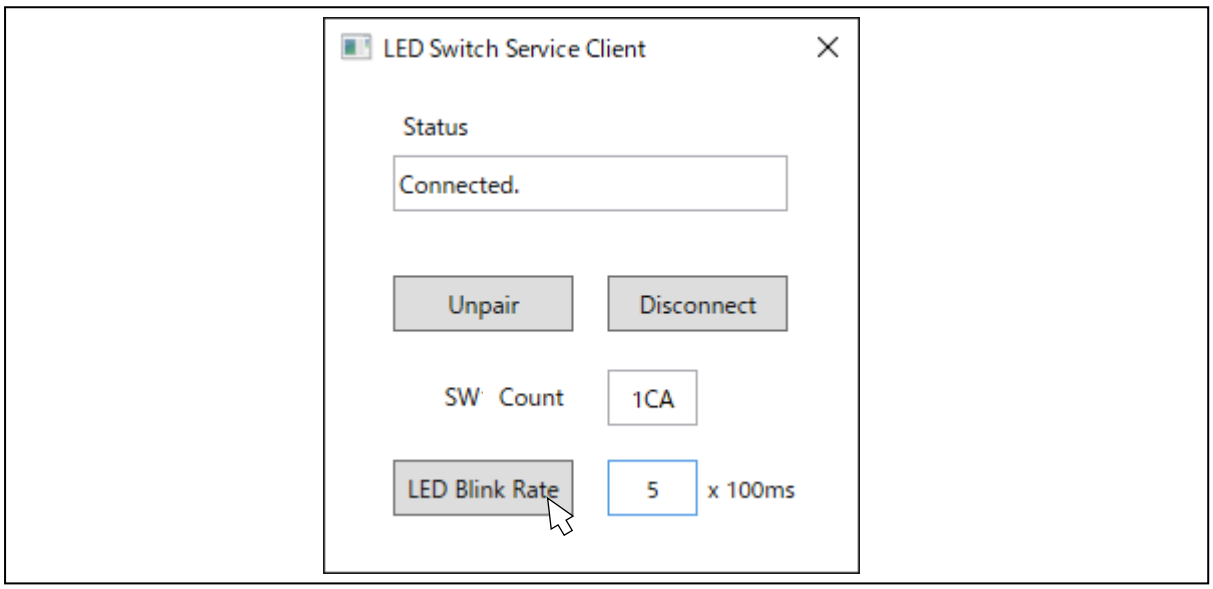

図 2-8 LED Switch Service Client の動作確認 (6)

(7) [Disconnect]ボタンをクリックし、評価ボードとの接続を切断します。

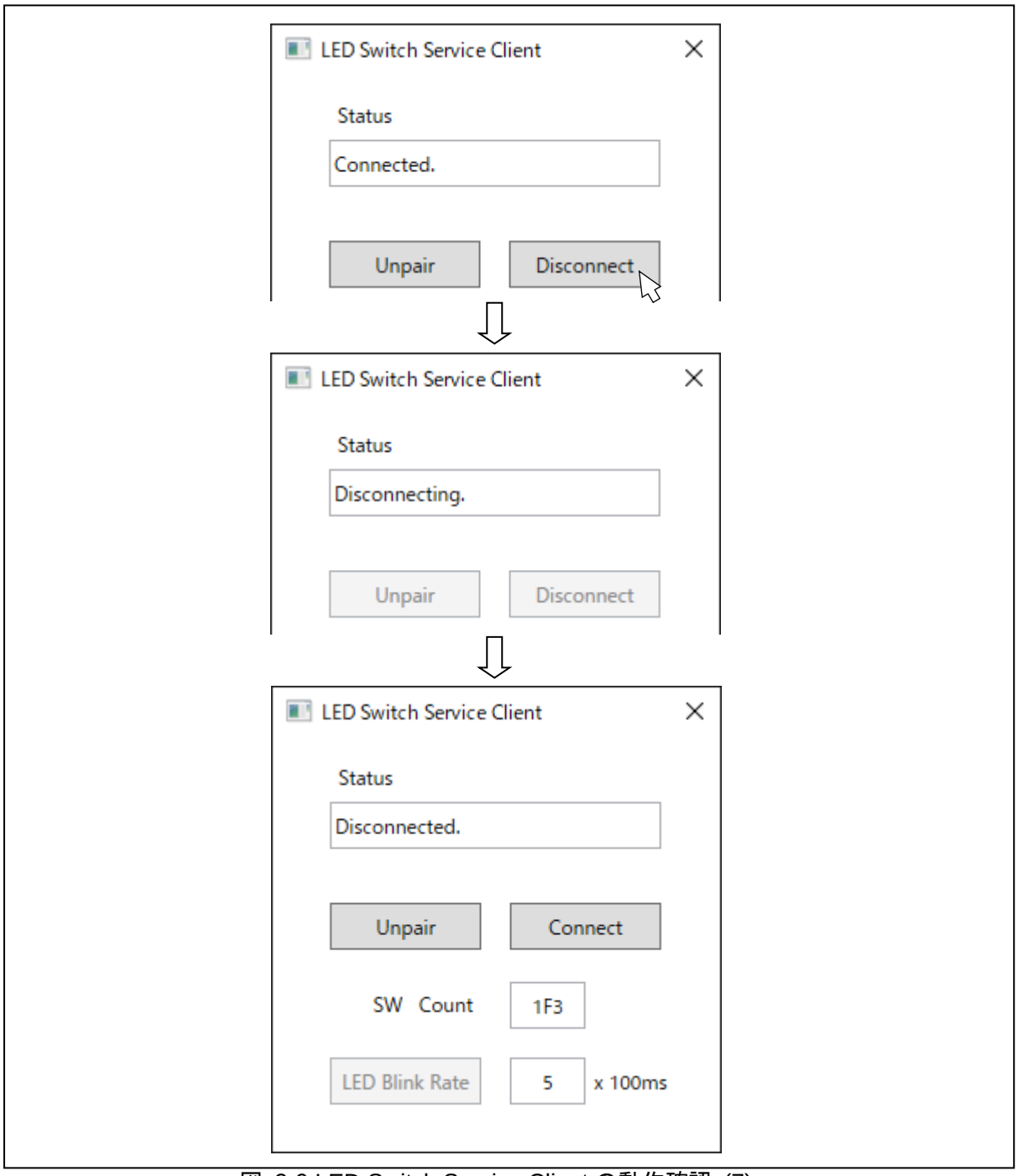

図 2-9 LED Switch Service Client の動作確認 (7)

(8) [Unpair]ボタンをクリックしペアリングを解除します。Windows の[設定] - [デバイス] - [Bluetooth とその他のデバイス]から、評価ボード(RBLE-DEV)が削除されます。

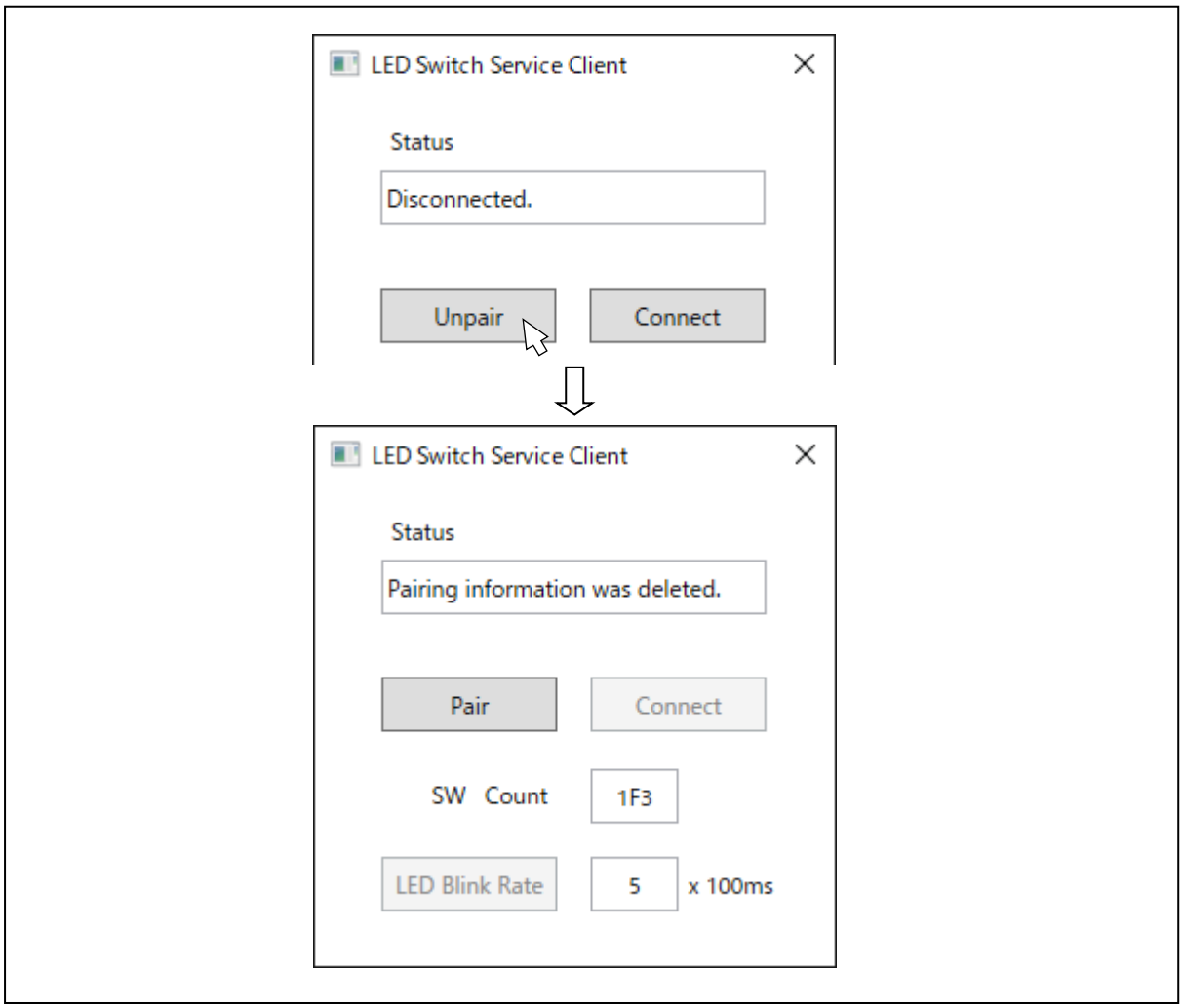

図 2-10 LED Switch Service Client の動作確認 (8)

2.4.2　Virtual UART Client を使用した動作確認

(1)　RL78/G1D 評価ボードの CN3(USB mini-B)と PC を USB ケーブルで接続し、RL78/G1D 評価ボードに電源を投入します。電源を投入すると自動的にアドバタイジングが開始されます。

(2)　RL78/G1D 評価ボードが接続された PC でターミナルソフトを起動します。本動作確認ではターミナルソフトとして Tera Term を使用します。

(3)　表 2-3の Virtual UART Client を起動します。

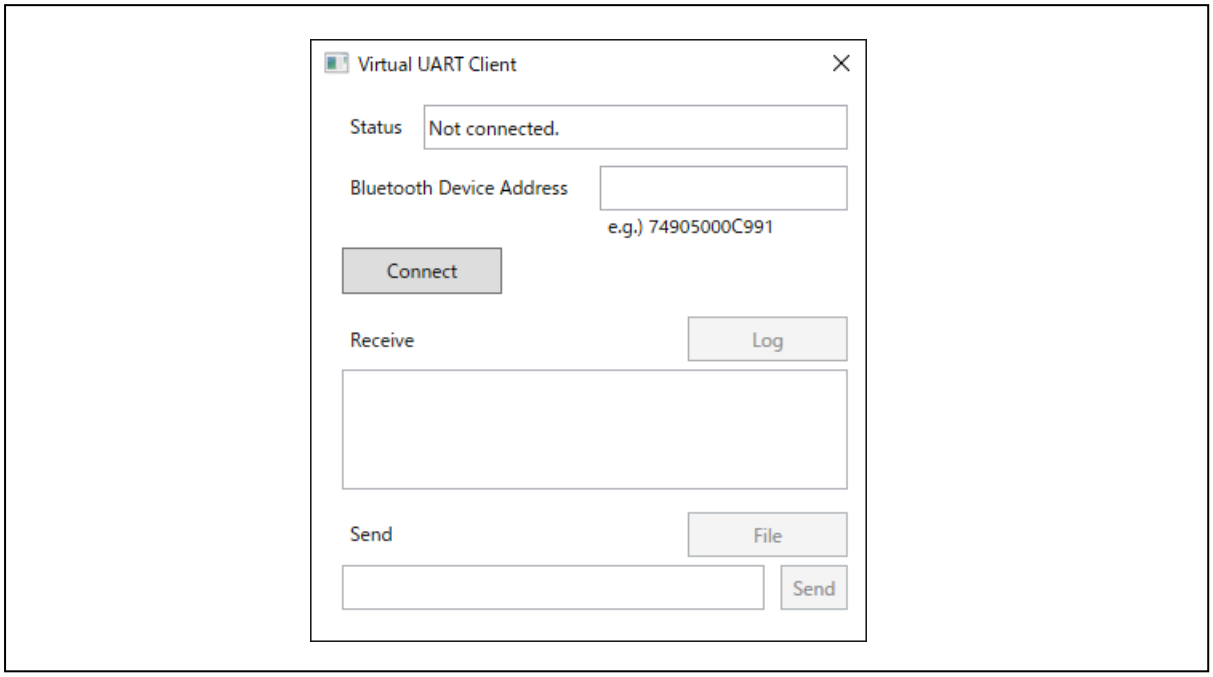

図 2-11 Virtual UART Client の動作確認 (1)

(4) [Connect]ボタンをクリックし評価ボードと接続します。Bluetooth Device Address テキストボックスに接続した評価ボードのアドレスが表示され、[Log]ボタン、[File]ボタン、[Send]ボタンが有効になります。Tera Term に「CONNECT」と表示されます。
評価ボードが見つからない場合、評価ボードに電源を投入してから一定時間が経過しアドバタイジングの送信間隔が長くなることで発見しにくくなっていることがあります。アドバタイジングの送信間隔を短くするために評価ボードの Reset Switch を押して再起動してください。

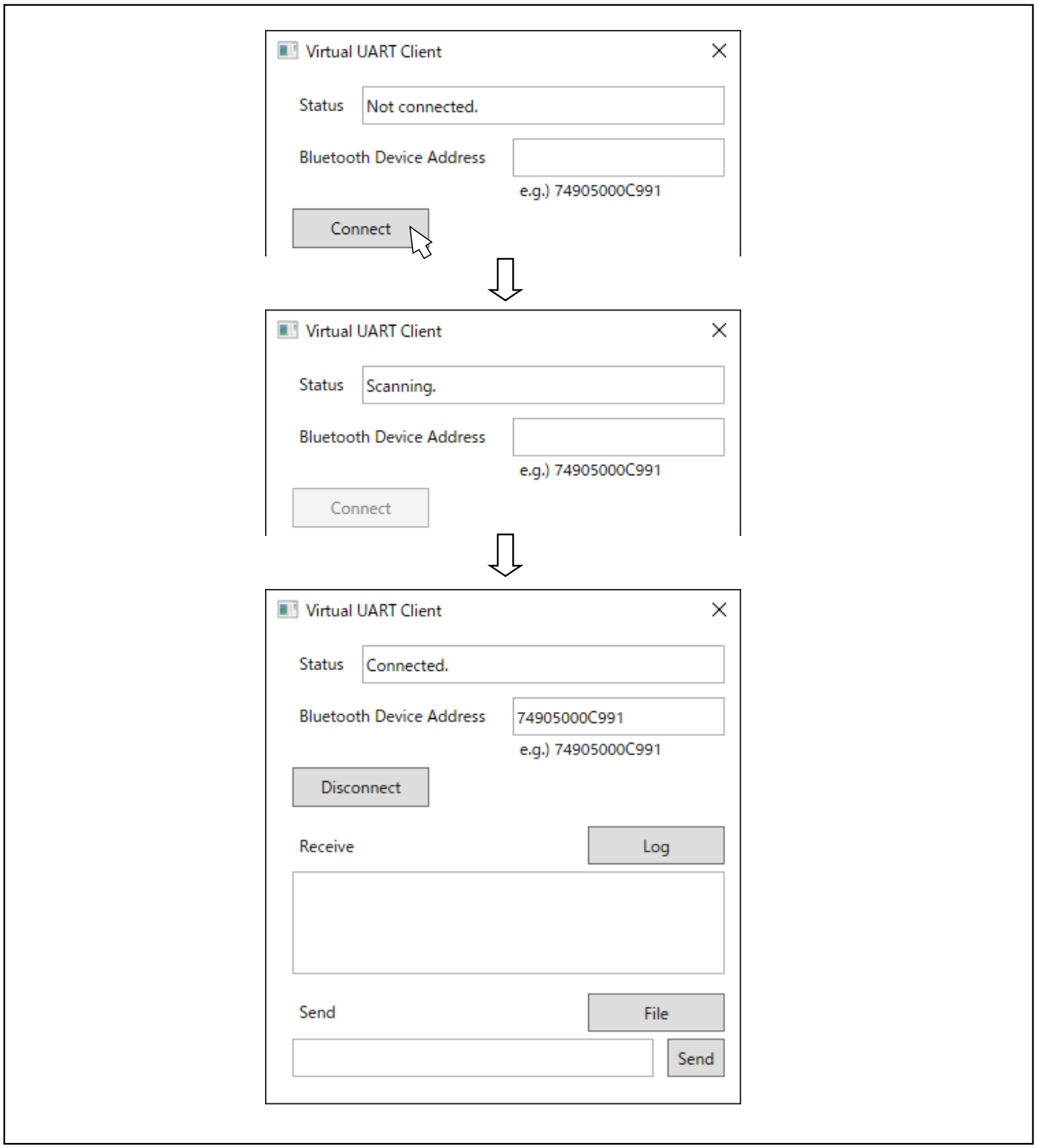

図 2-12 Virtual UART Client の動作確認 (2)

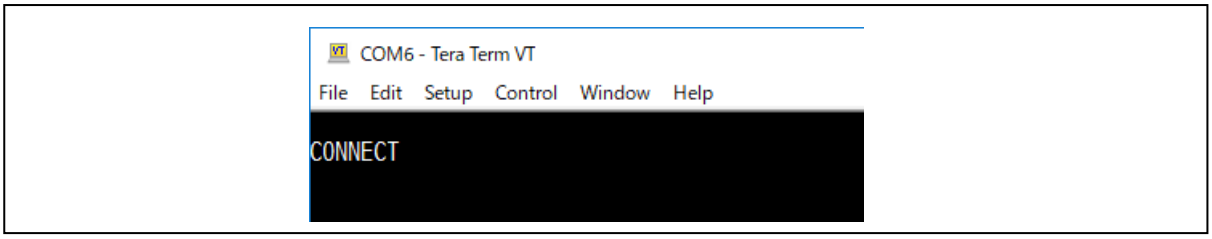

図 2-13 Virtual UART Client の動作確認 (3)

(5) [Log]ボタンをクリックし「名前を付けて保存」ダイアログを表示します。ファイルを作成するフォルダを選択し、[ファイル名]に任意の名前を入力します。[保存]ボタンをクリックするとログファイルが作成されログの取得が開始されます。

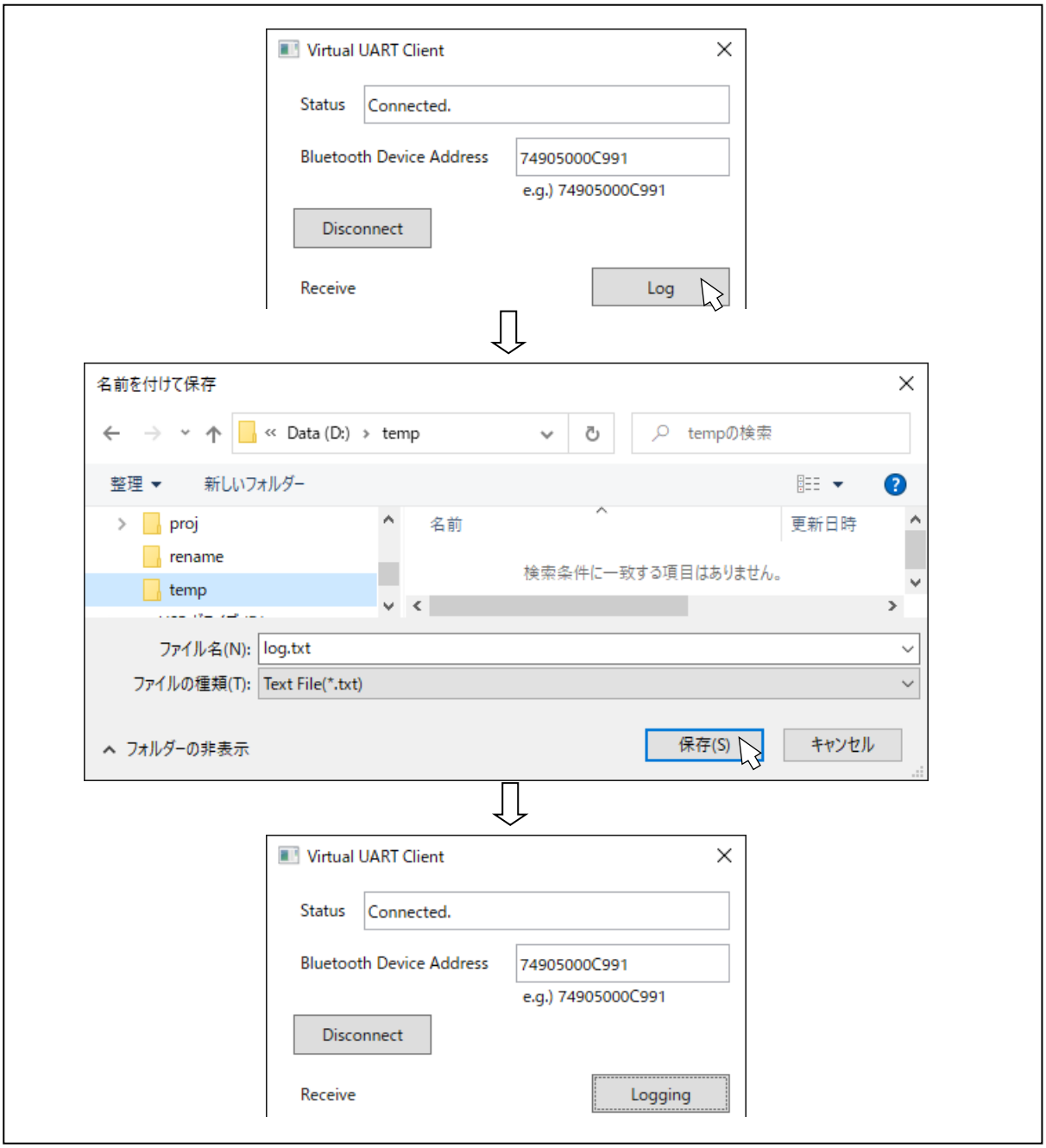

図 2-14 Virtual UART Client の動作確認 (4)

(6) Tera Term で ESC キーを押して仮想 UART モードにした後、文字を入力します。

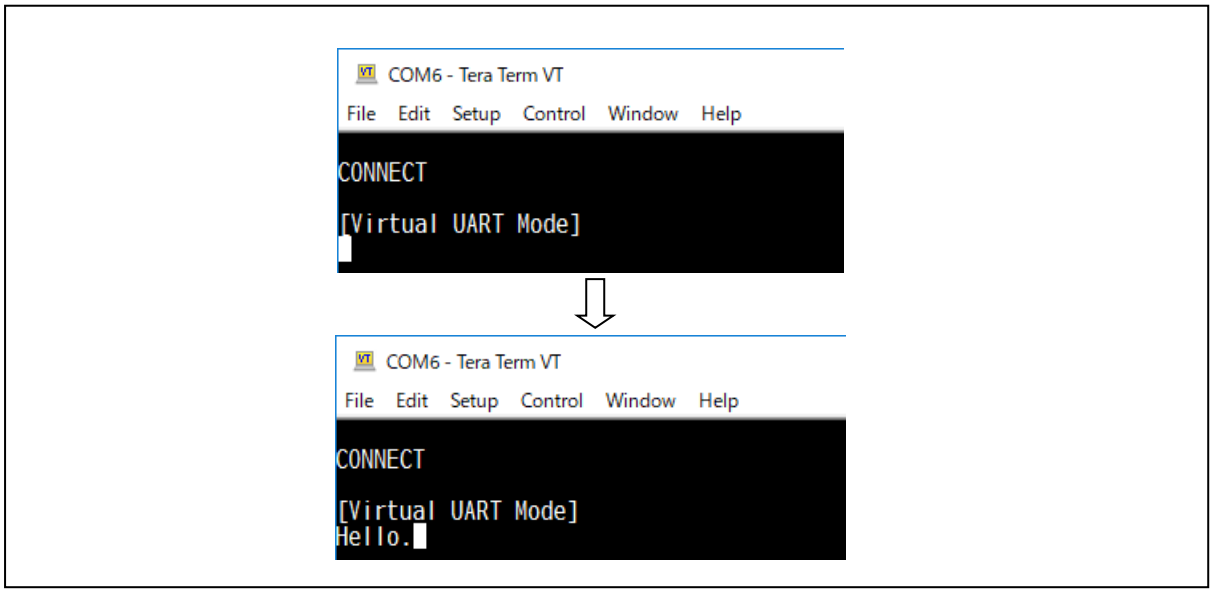

図 2-15 Virtual UART Client の動作確認 (5)

(7) Virtual UART Client の「Receive」テキストボックスに Tera Term に入力した文字が表示されます。

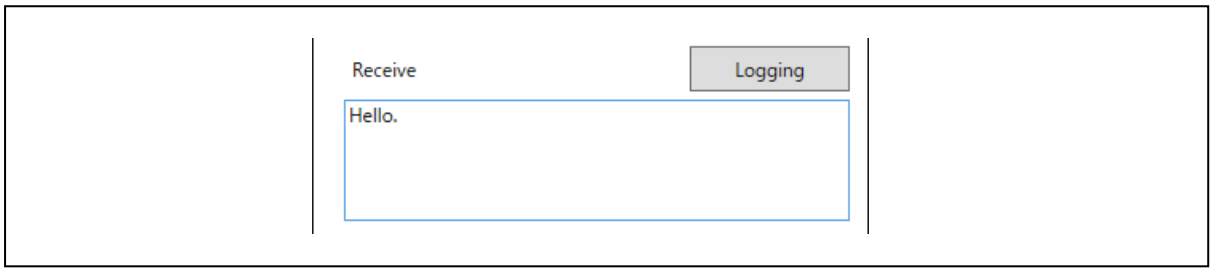

図 2-16 Virtual UART Client の動作確認 (6)

(8) [Logging]ボタンをクリックしログの記録を終了します。ログファイルをテキストエディタで開くと受信した文字が保存されていることが確認できます。

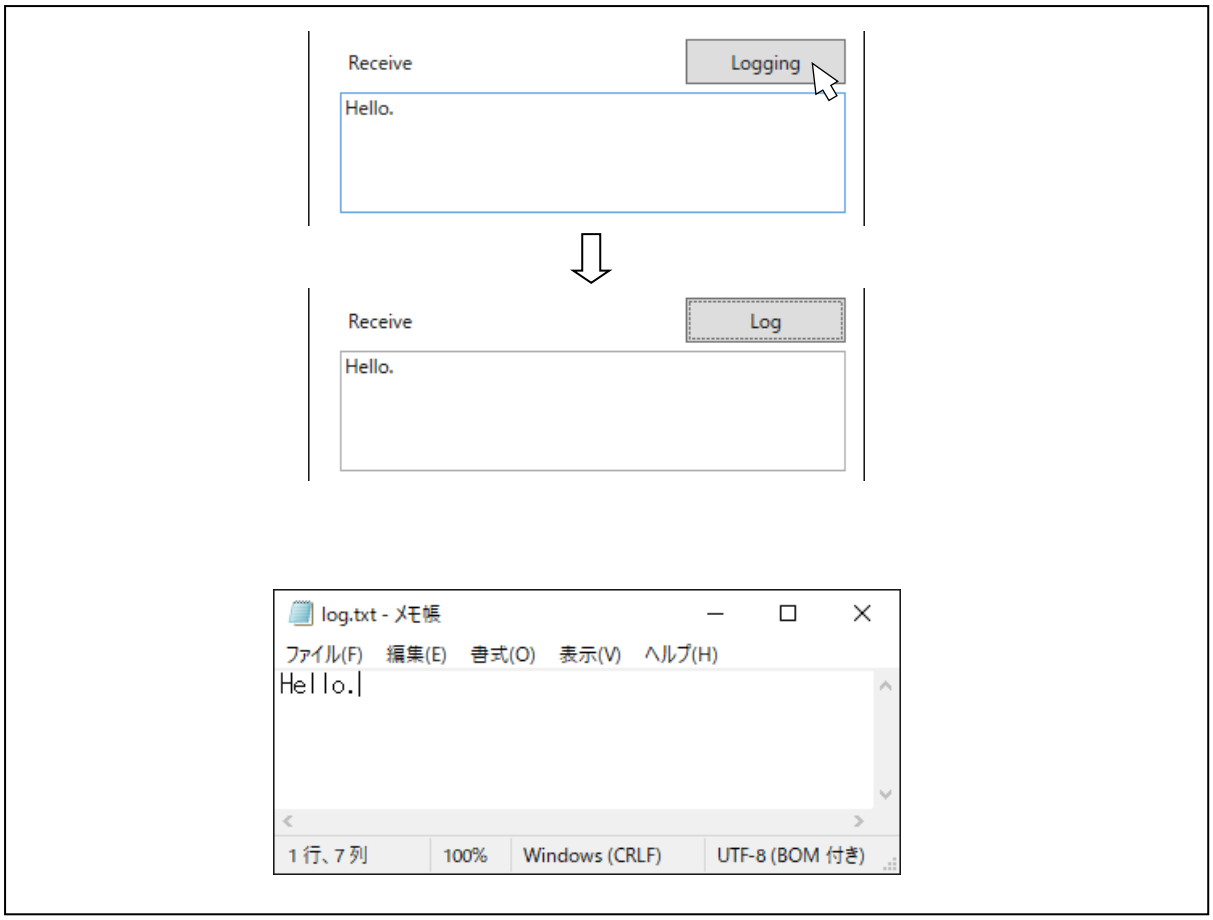

図 2-17 Virtual UART Client の動作確認 (7)

(9) Virtual UART Client で[Send]テキストボックスに文字を入力し[Send]ボタンをクリックします。Tera Term に送信した文字が表示されます。

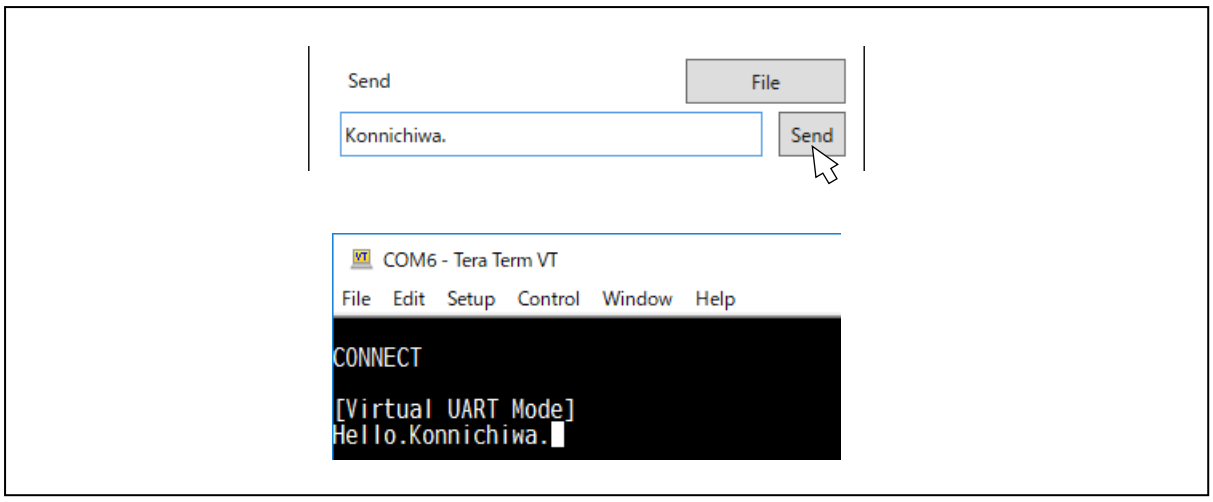

図 2-18 Virtual UART Client の動作確認 (8)

(10) 文字を書き込んだテキストファイルを用意します。[File]ボタンをクリックし「開く」ダイアログを表示します。テキストファイルを選択し[開く]ボタンをクリックするとテキストファイルのデータが送信されます。

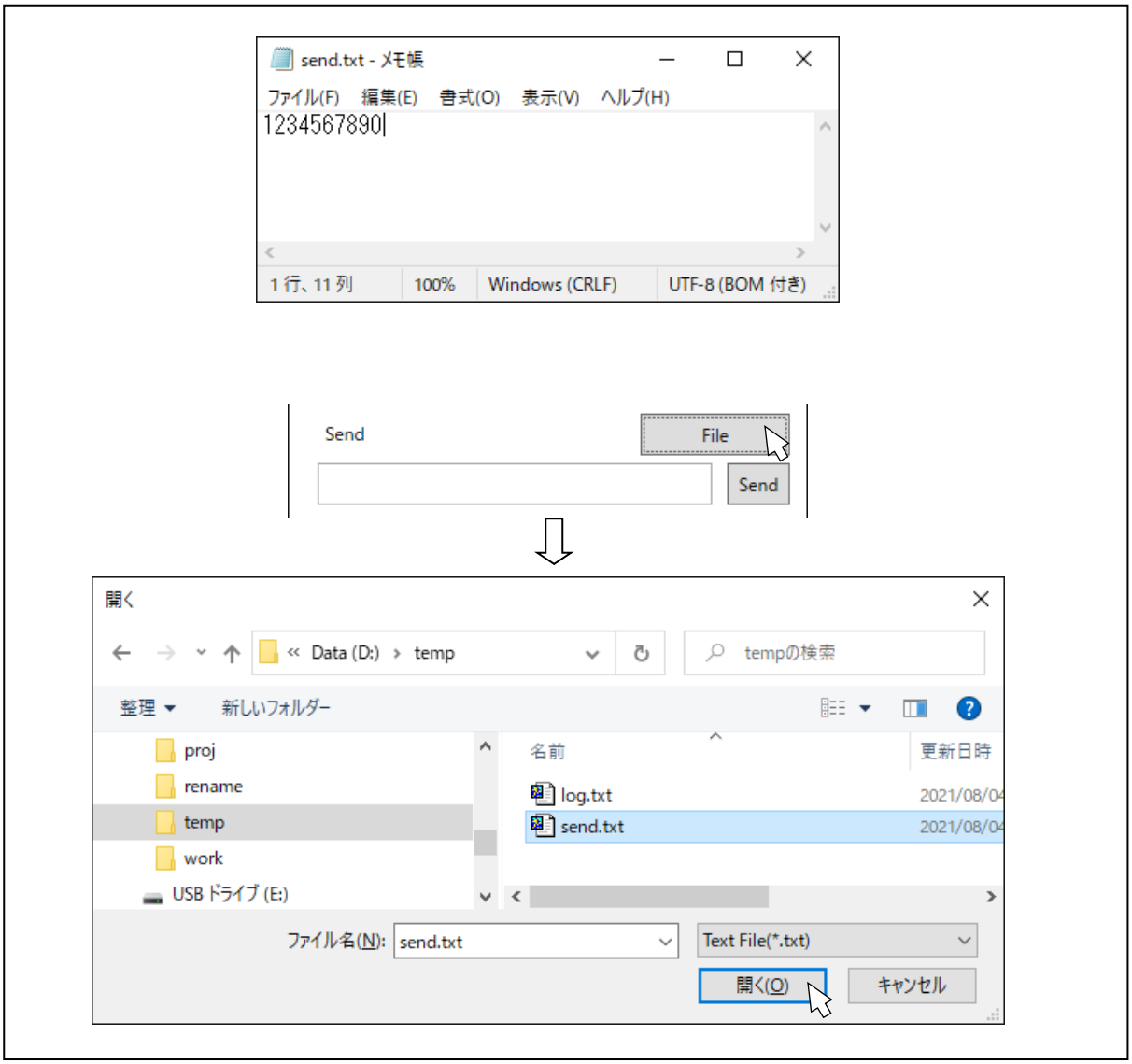

図 2-19 Virtual UART Client の動作確認 (9)

(11) Tera Term にテキストファイルのデータが表示されます。

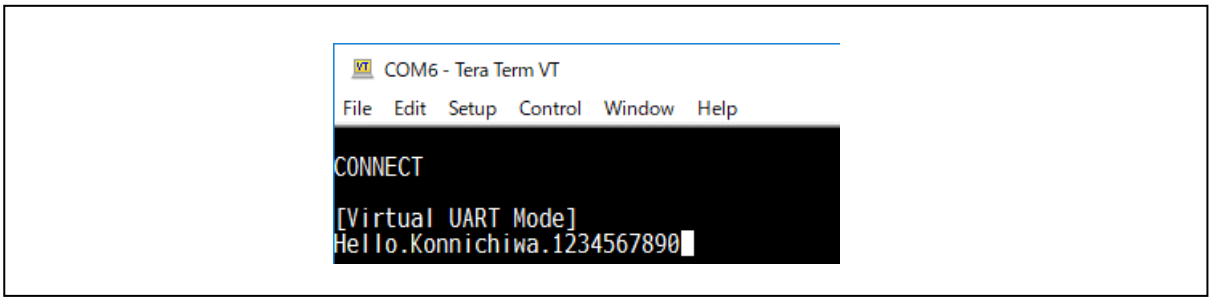

図 2-20 Virtual UART Client の動作確認 (10)

(12) Virtual UART Client の[Disconnect]ボタンをクリックし評価ボードとの接続を切断します。
Tera Term に「DISCONNECT」が表示されます。

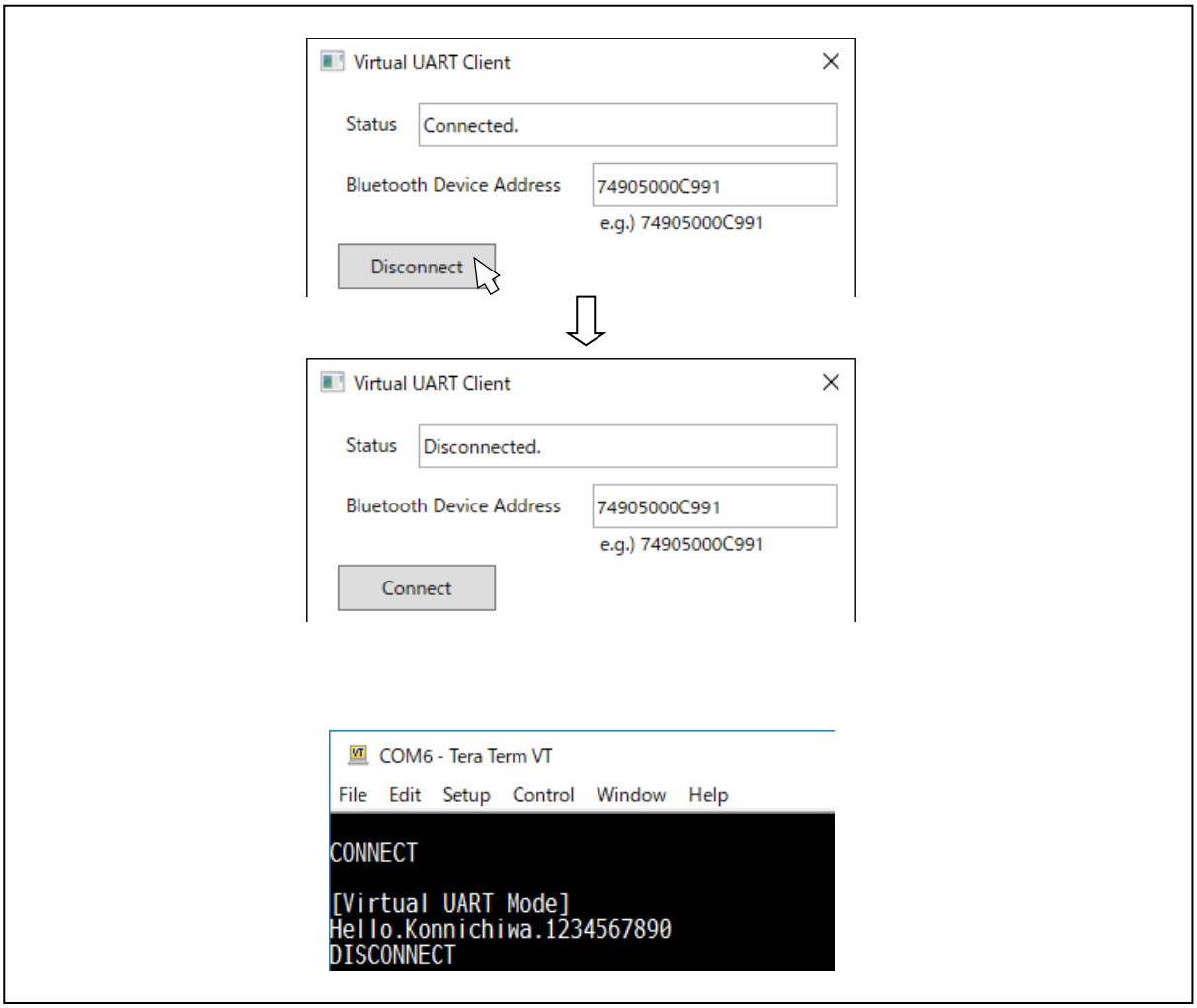

図 2-21 Virtual UART Client の動作確認 (11)

3. ビルド

Windows アプリケーションのビルド環境の構築について説明します。

本アプリケーションノートでは、Windows アプリケーションの動作確認をする目的で「Microsoft Visual Studio Express 2017 for Windows Desktop」(以降、Visual Studio 2017)を使用します。(製品開発においてはご使用条件に合ったエディションの Visual Studio をインストールしてください。)

表 3-1に Windows アプリケーションのビルド環境情報を示します。

表 3-1 ビルド環境情報

Visual Studio バージョン	Microsoft Visual Studio Express 2017 for Windows Desktop Version 15.9.38
.NET Framework バージョン	.NET Framework 4.6.1
プロジェクト テンプレート	WPF アプリ(.NET Framework)
Windows 10 SDK バージョン	10.0.19041.0

3.1 ビルド環境構築

3.1.1 Visual Studio 2017 のインストール

以下のホームページからインストーラーをダウンロードし Visual Studio 2017 をインストールしてください。

Visual Studio Express | 今すぐ Visual Studio Community

(1) ホームページから「Express 2017 for Windows Desktop」のインストーラーをダウンロードします。

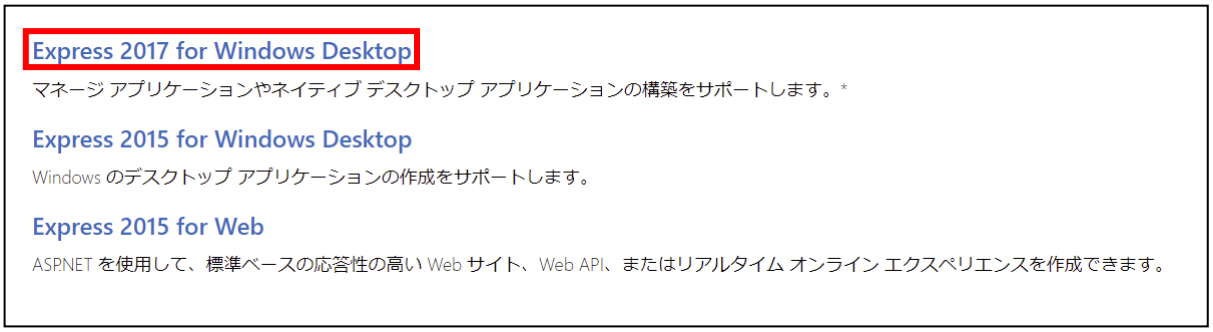

図 3-1 Visual Studio 2017 のインストール (1)

(2) インストーラーを実行し、[続行]ボタンをクリックします。

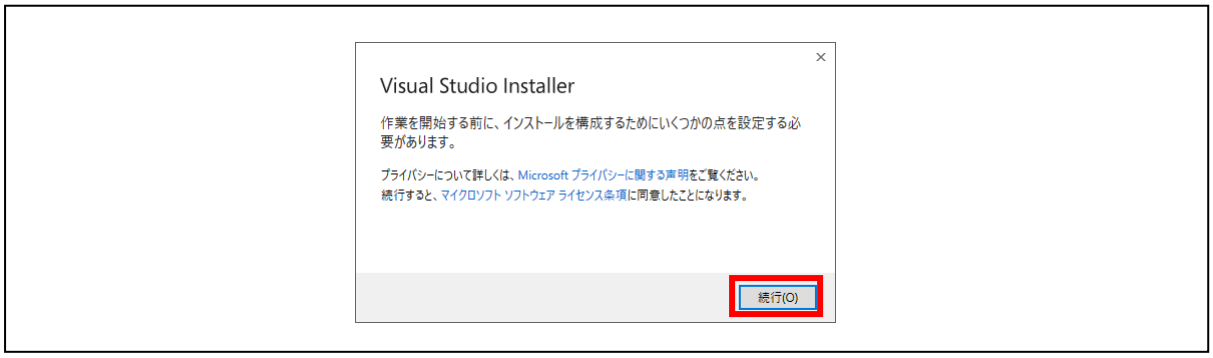

図 3-2 Visual Studio 2017 のインストール (2)

(3) Visual Studio Installer に必要なファイルのダウンロードが完了すると、以下のウインドウが起動します。[インストール]ボタンをクリックしてインストールを開始します。

図 3-3 Visual Studio 2017 のインストール (3)

(4) インストールが完了するとアカウントにサインインするウインドウが起動します。ここでは「後で行う。」をクリックします。Visual Studio 2017 が起動した場合は、右上の[×]ボタンをクリックして Visual Studio 2017 を終了します。

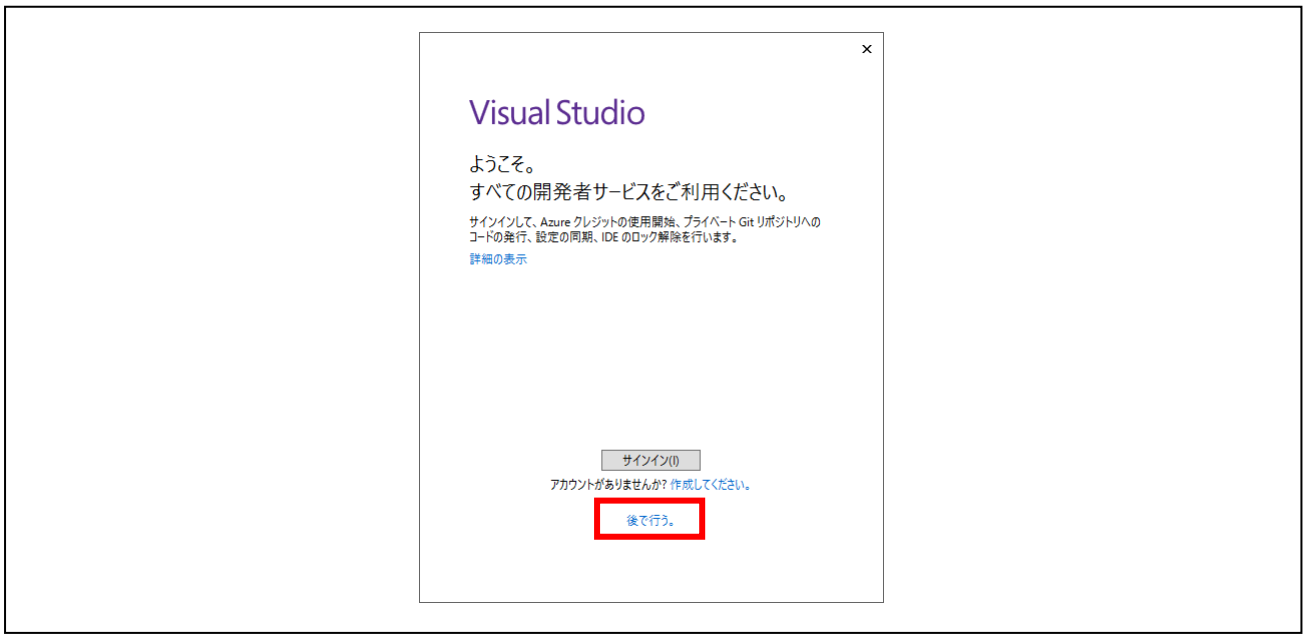

図 3-4 Visual Studio 2017 のインストール (4)

(5) インストーラーの右上の[×]ボタンをクリックしてウインドウを閉じます。

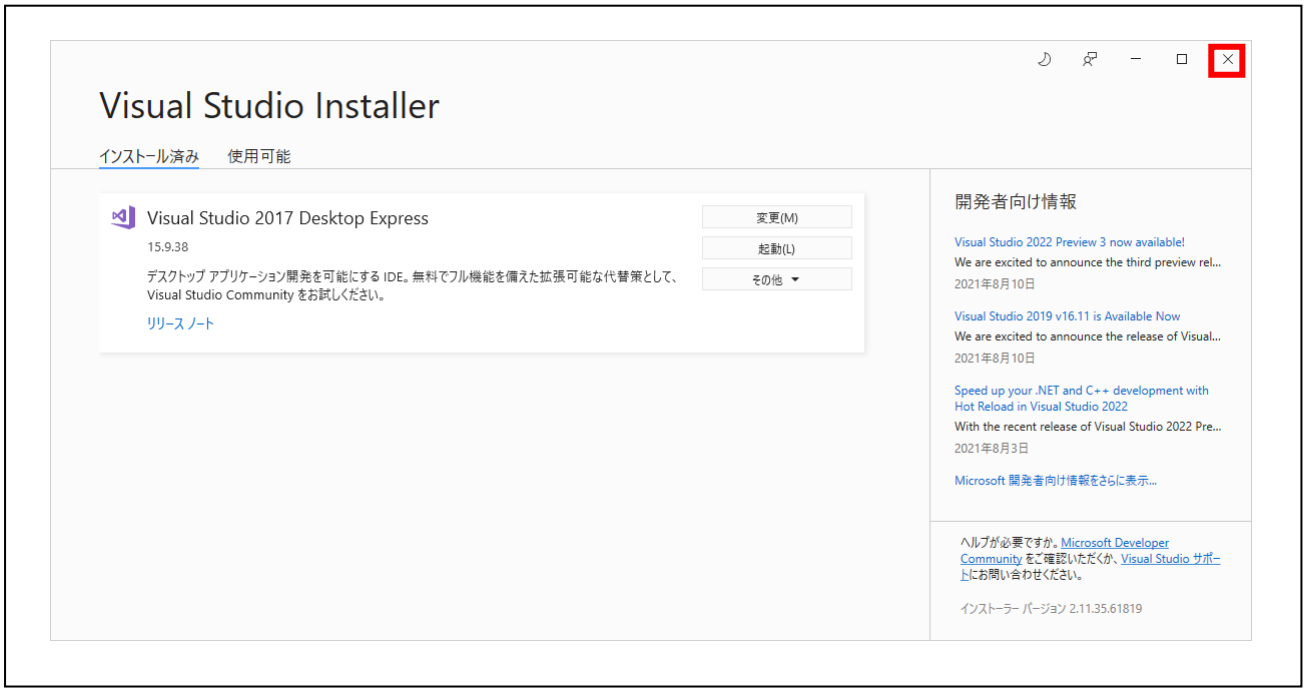

図 3-5 Visual Studio 2017 のインストール (5)

3.1.2　Windows 10 SDK のインストール

以下のホームページからインストーラーをダウンロードし Windows 10 SDK をインストールしてください。

本 Windows アプリケーションでは、「2020 年 12 月 16 日更新」のインストーラーを使用しています。

Windows 10 SDK - Windows アプリ開発

(1)　Windows 10 SDK のホームページから赤線で示すインストーラーをダウンロードします。

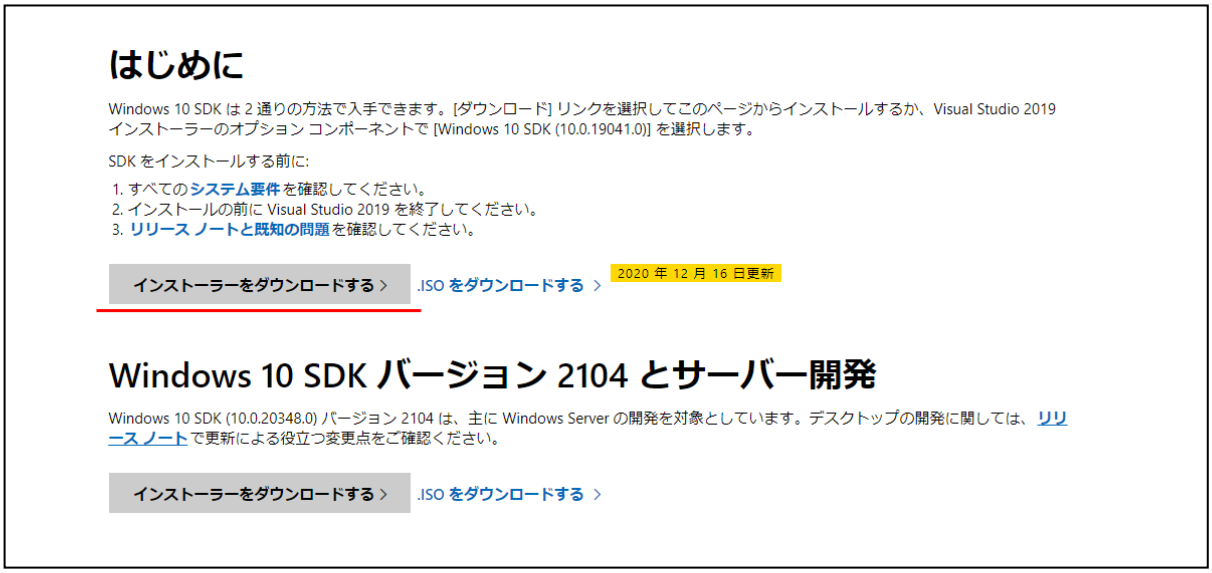

図 3-6 Windows 10 SDK のインストール (1)

(2)　インストーラーを実行し、「Install new or update ...」のラジオボタンを選択して[Next]ボタンをクリックします。

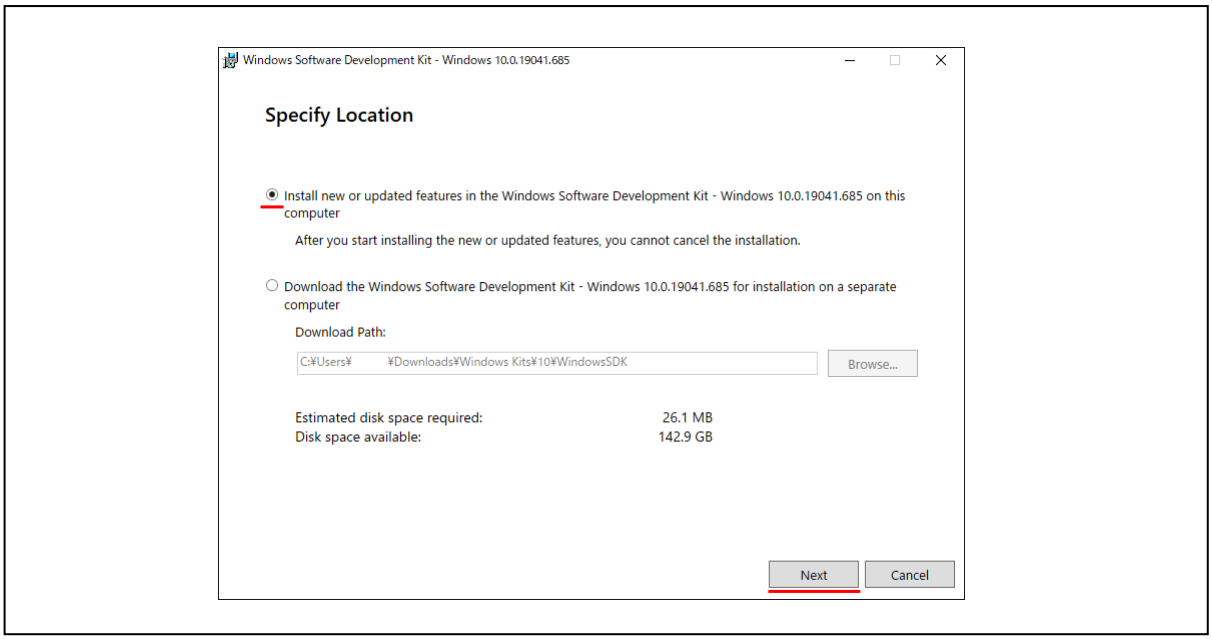

図 3-7 Windows 10 SDK のインストール (2)

(3) ライセンス契約に同意し[Accept]ボタンをクリックします。

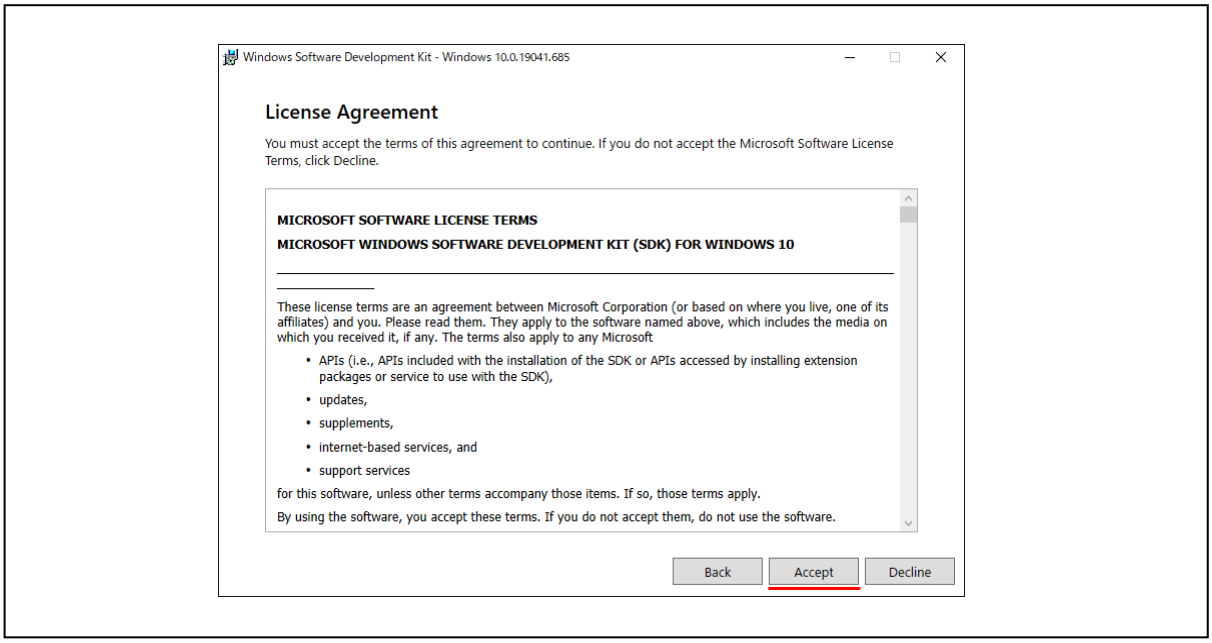

図 3-8 Windows 10 SDK のインストール (3)

(4) インストールする機能を選択します。デフォルト状態で全ての機能が選択されていますので、そのままの状態で[Install]ボタンをクリックします。

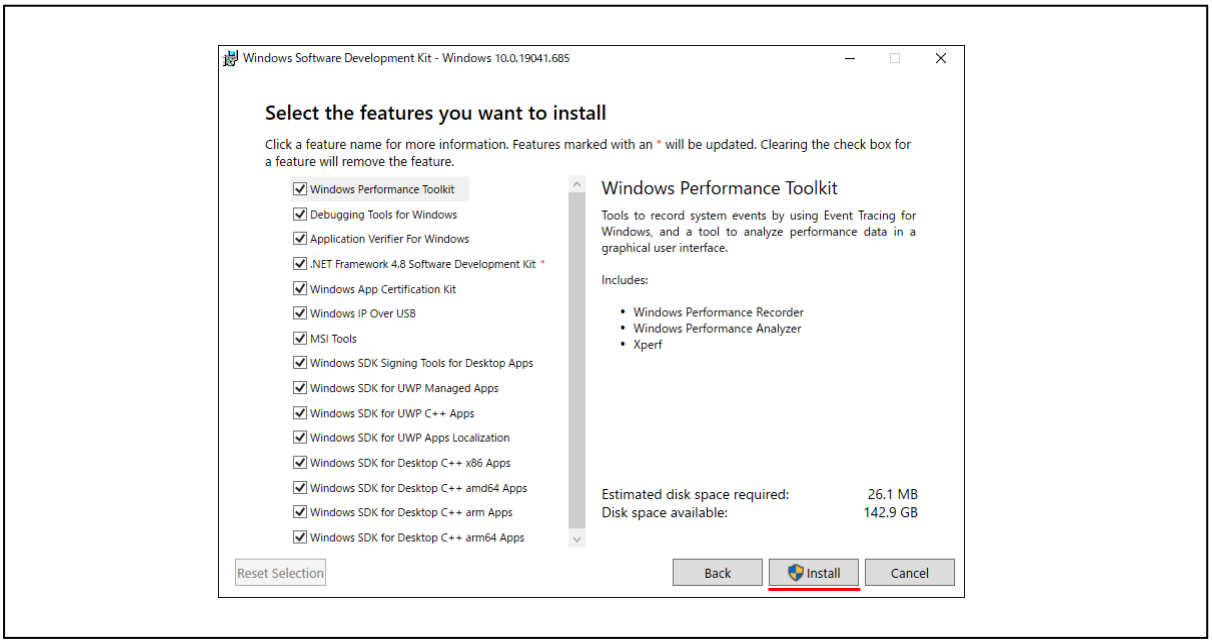

図 3-9 Windows 10 SDK のインストール (4)

(5) インストールが完了し、[Close]ボタンをクリックしてウインドウを閉じます。

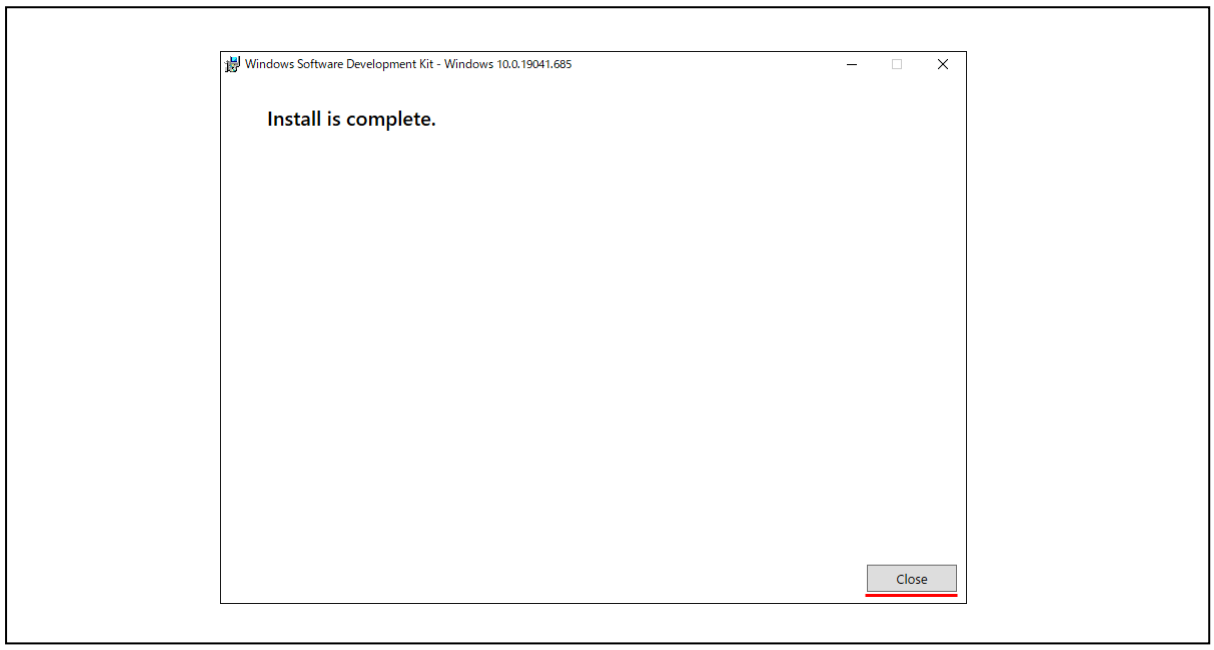

図 3-10 Windows 10 SDK のインストール (5)

3.2　ファイル構成

3.2.1　LED Switch Service Client

LED Switch Service Client のファイル構成を以下に示します。

```
r01an6004jj0100-ble-mcu-win
 │
 └────LED_Switch_Service_Client
      │   App.config                          アプリケーション構成ファイル
      │   App.xaml                            App クラスファイル
      │   App.xaml.cs                         App クラスファイル
      │   LED_Switch_Service_Client.csproj    LED Switch Service Client プログラムファイル
      │   LED_Switch_Service_Client.sln       LED Switch Service Client プログラムファイル
      │   MainWindow.xaml                     プロジェクトファイル
      │   MainWindow.xaml.cs                  ソリューションファイル
      │
      ├────exe
      │       LED_Switch_Service_Client.exe   ビルド済み実行ファイル
      │
      └────Properties
           AssemblyInfo.cs                    アセンブリ情報
           Resources.Designer.cs              リソースファイル
           Resources.resx                     リソースファイル
           Settings.Designer.cs               アプリケーション設定ファイル
           Settings.settings                  アプリケーション設定ファイル
```

3.2.2　Virtual UART Client

Virtual UART Client のファイル構成を以下に示します。

```
r01an6004jj0100-ble-mcu-win
 │
 └────Virtual_UART_Client
      │   App.config                          アプリケーション構成ファイル
      │   App.xaml                            App クラスファイル
      │   App.xaml.cs                         App クラスファイル
      │   MainWindow.xaml                     仮想 UART Client プログラムファイル
      │   MainWindow.xaml.cs                  仮想 UART Client プログラムファイル
      │   Virtual_UART_Client.csproj          プロジェクトファイル
      │   Virtual_UART_Client.sln             ソリューションファイル
      │
      ├────exe
      │       Virtual_UART_Client.exe         ビルド済み実行ファイル
      │
      └────Properties
           AssemblyInfo.cs                    アセンブリ情報
           Resources.Designer.cs              リソースファイル
           Resources.resx                     リソースファイル
           Settings.Designer.cs               アプリケーション設定ファイル
           Settings.settings                  アプリケーション設定ファイル
```

3.3 Windows アプリケーションのビルド

Visual Studio 2017 で、LED Switch Service Client または Virtual UART Client の実行ファイルをビルドします。

(1) 「3.2 ファイル構成」で示されるソリューションファイル(LED_Switch_Service_Client.sln、または Virtual_UART_Client.sln)をダブルクリックして Visual Studio 2017 を起動します。

(2) ソリューション構成を[Release]に設定します。

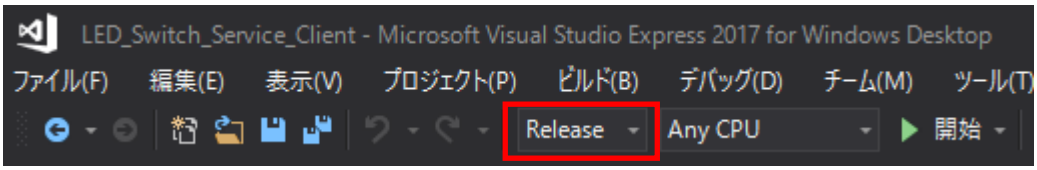

図 3-11 Release ビルド設定

(3) LED Switch Service Client の場合、メニューバーの[ビルド] - [ソリューションのビルド]を選択しビルドを行います。
Virtual UART Client の場合、メニューバーの[ビルド] - [ソリューションのビルド]を選択しビルドを行います。

(4) 実行ファイルが以下のフォルダに作成されます。

表 3-2 実行ファイル

フォルダ名	ファイル名
r01an6004jj0100-ble-mcu-win¥LED_Switch_Service_Client¥bin¥Release	LED_Switch_Service_Client.exe
r01an6004jj0100-ble-mcu-win¥Virtual_UART_Client¥bin¥Release	Virtual_UART_Client.exe

4. 実装の詳細

4.1 Windows アプリケーションのプログラムファイル

LED Switch Service Client と Virtual UART Client のユーザーインターフェース(以降、UI)やアプリケーションの処理は以下の 2 つのファイルに記述されています。ファイルが格納されているフォルダは「3.2 ファイル構成」を参照してください。

- MainWindow.xaml

  Window のユーザーインターフェースを記述する XML 形式の XAML コードファイルです。Window の外観、ボタンやテキストボックス等コントールの配置が定義されています。

- MainWindow.xaml.cs

  アプリケーションの処理を記述する C#コードファイルです。ペアリング、接続、データ通信等を処理するプログラムが記述されています。

4.2 WPF で UWP の Bluetooth LE API を使用する

本アプリケーションノートでは、LED Switch Service Client と Virtual UART Client を WPF のアプリケーションとして作成していますが、Windows の Bluetooth LE API は UWP で利用できる API です。通常、WPF アプリケーションからは利用できませんが、UWP API を構成している Windows メタデータファイル(winmd)と.NET アセンブリ(dll)をプロジェクトから参照することで UWP API を利用しています。

- Windows.winmd

  Windows ランタイム API(WinRT API)を使用できるようにするための Windows メタデータ(WinMD)ファイルです。

  ファイルは以下のフォルダに格納されています。

  C:¥Program Files (x86)¥Windows Kits¥10¥UnionMetadata¥10.0.19041.0

  (10.0.19041.0 は Windows 10 SDK のバージョン番号)

- System.Runtime.WindowsRuntime.dll

  .NET アプリケーションの構成要素である.NET アセンブリファイルです。

  ファイルは以下のフォルダに格納されています。

  C:¥Program Files (x86)¥Reference Assemblies¥Microsoft¥Framework¥.NETCore¥v4.5

  (Windows 10 の.NETCore V4.5 が対象の場合)

4.3 LED Switch Service Client

LED Switch Service Client で定義されるクラスや処理の内容を説明します。

表 4-1 LED Switch Service Client のクラス概要

クラス名	説明
MainWindow	LED Switch Service Client のアプリケーション処理が記述されたクラス。
AdvertisementWatcherInformation	接続対象の評価ボードが送信するアドバタイジングから得た情報を保存するクラス。
DeviceWatcherInformation	DeviceWatcher が列挙する Bluetooth LE 機器情報を保存するクラス。
ViewModel	プログラムと UI の間のデータ交換で使用するクラス。

4.3.1 MainWindow クラス

4.3.1.1 フィールド(メンバ変数)

Guid UUID_LEDSW_SERVICE
LED Switch Service UUID を定義。評価ボードと接続した後、Service の検索で使用します。
Guid UUID_LEDSW_SW_STATE_NOTIFICATION
LED Switch Service の Switch State Characteristic UUID を定義。評価ボードと接続した後、Characteristic の検索で使用します。
Guid UUID_LEDSW_LED_BLINK_RATE_WRITE
LED Switch Service の LED Blink Rate Characteristic UUID を定義。評価ボードと接続した後、Characteristic の検索で使用します。
const string DEVNAME_RBLE
BluetoothLEAdvertisementWatcher で受信したアドバタイジングの Complete Local Name と比較して接続対象の評価ボードの発見に使用します。
BluetoothLEAdvertisementWatcher advertisementWatcher
BluetoothLEAdvertisementWatcher クラスのオブジェクトを保存し、スキャンを実行してアドバタイジングを受信するために使用します。
DeviceWatcher deviceWatcher
DeviceWatcher クラスのオブジェクトを保存し、Windows のシステムに登録されているデバイスや発見した外部デバイスを列挙します。本アプリケーションでは、Windows に登録されているペアリング済みの Bluetooth LE デバイスやアドバタイジングを送信している Bluetooth LE デバイスの列挙に使用します。
AdvertisementWatcherInformation peripheralDevice
AdvertisementWatcherInformation クラスのオブジェクトを保存し、BluetoothLEAdvertisementWatcher クラスのスキャンで受信した、接続対象の評価ボードの情報を格納します。

DeviceWatcherInformation pairedDeviceInfomation
DeviceWatcherInformation クラスのオブジェクトを保存し、ペアリングした評価ボードの DeviceInformation クラス情報を格納します。
ObservableCollection<DeviceWatcherInformation> RBLEDevices
ObservableCollection<T>クラスのオブジェクトを保存するコレクション。DeviceWatcher クラスで列挙されたデバイスの DeviceInformation クラス情報を格納します。
BluetoothLEDevice bluetoothLeConnectedDevice
評価ボードと接続を開始する際に、BluetoothLEDevice クラスのオブジェクトを保存します。
GattDeviceService gattPrimaryService
評価ボードと接続した後、LED Switch Service の BluetoothLEDevice クラスのオブジェクトを保存します。
GattCharacteristic characteristicNotify
評価ボードと接続した後、Switch State Characteristic の GattCharacteristic クラスのオブジェクトを保存します。
GattCharacteristic characteristicWrite
評価ボードと接続した後、LED Blink Rate Characteristic の GattCharacteristic クラスのオブジェクトを保存します。
bool isLunchedApp
DeviceWatcher の停止時に、DeviceWatcher 開始位置の判定に使用します。本フラグはアプリケーション起動した時にセットします。
bool isClickedPairButton
DeviceWatcher の停止時に、DeviceWatcher 開始位置の判定に使用します。本フラグは Pair ボタンをクリックした時にセットします。
bool isClickedConnectButton
DeviceWatcher の停止時に、DeviceWatcher 開始位置の判定に使用します。本フラグは Connect ボタンをクリックした時にセットします。
ViewModel viewModel
ViewModel クラスのオブジェクトを保存し、プログラムと UI の間のデータ交換に使用します。
Timer timer
Timer クラスのオブジェクトを保存し、BluetoothLEAdvertisementWatcher や DeviceWatcher の動作停止(タイムアウト)に使用します。
int notifyCount
評価ボードからの Notification 受信回数のカウントに使用します。

4.3.1.2　コンストラクタ

MainWindow()
UI とのデータ交換を行うことができるようにするため、ViewModel クラスをインスタンス化し DataContext プロパティへセットします。 評価ボードとペアリングされているか調べるため DeviceWatcher を開始します。

4.3.1.3 メソッド/イベントハンドラ

void ButtonPair_Click(object sender, RoutedEventArgs e)	
Pair/Unpair ボタンのクリックイベントハンドラ。 Pair ボタンの時は、評価ボードとのペアリングを開始します。Unpair ボタンの時は、評価ボードとのペアリングを解除します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void ButtonConnect_Click(object sender, RoutedEventArgs e)	
Connect/Disconnect ボタンのクリックイベントハンドラ。 Connect ボタンの時は、ペアリングされている評価ボードと接続します。Disconnect ボタンの時は、評価ボードとの接続を切断します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void ButtonLEDBlinkRate_Click(object sender, RoutedEventArgs e)	
LED Blink Rate ボタンのクリックイベントハンドラ。 LED Blink Rate テキストボックスに入力された値を Write で評価ボードへ送信します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void AdvertisementWatcher_Start(int timeout)	
BluetoothLEAdvertisementWatcher でスキャンを実行します。BluetoothLEAdvertisementWatcher は、接続する評価ボードを発見した場合、または評価ボードがタイムアウト時間内に見つからなかった場合に終了します。	
パラメータ	
int timeout	タイムアウト時間 (単位: ミリ秒)。

void AdvertisementWatcher_Stop()
BluetoothLEAdvertisementWatcher を停止します。

void AdvertisementWatcher_Received(BluetoothLEAdvertisementWatcher watcher, BluetoothLEAdvertisementReceivedEventArgs eventArgs)	
BluetoothLEAdvertisementWatcher のアドバタイジング受信イベントハンドラ。 アドバタイジングを受信するごとに呼び出されます。接続する評価ボードの名前を発見した場合、DeviceWatcher を開始します。	
パラメータ	
BluetoothLEAdvertisementWatcher watcher	イベントを送信した BluetoothLEAdvertisementWatcher のオブジェクト。
BluetoothLEAdvertisementReceivedEventArgs eventArgs	受信したイベントのデータ。

void AdvertisementWatcherTimer_Callback(object state)	
Timer のコールバックメソッド。 BluetoothLEAdvertisementWatcher の動作時間満了で呼び出され、BluetoothLEAdvertisementWatcher を停止します。	
パラメータ	
object state	アプリケーションの情報オブジェクト。

void DeviceWatcher_Start(int timeout)	
DeviceWatcher を開始しシステムに登録されている Bluetooth LE デバイスや外部の Bluetooth LE デバイスを列挙します。タイムアウトで指定された時間 DeviceWatcher を実行します。	
パラメータ	
int timeout	タイムアウト時間 (単位: ミリ秒)。

void DeviceWatcher_Added(DeviceWatcher sender, DeviceInformation deviceInfo)	
DeviceWatcher の Added イベントハンドラ。 DeviceWatcher によってデバイスが追加された時に呼び出され、DeviceWatcherInformation クラスのコレクションに情報を追加します。	
パラメータ	
DeviceWatcher sender	イベントを送信したオブジェクト。
DeviceInformation deviceInfo	追加されたデバイスの DeviceInformation 情報。

void DeviceWatcher_Updated(DeviceWatcher sender, DeviceInformationUpdate deviceInfoUpdate)	
DeviceWatcher の Updated イベントハンドラ。 DeviceWatcher によってデバイスが更新された時に呼び出され、DeviceWatcherInformation クラスのコレクションに格納されている情報を更新します。	
パラメータ	
DeviceWatcher sender	イベントを送信したオブジェクト。
DeviceInformationUpdate deviceInfoUpdate	更新されたデバイスの DeviceInformationUpdate 情報。

void DeviceWatcher_Removed(DeviceWatcher sender, DeviceInformationUpdate deviceInfoUpdate)	
DeviceWatcher の Removed イベントハンドラ。 DeviceWatcher によってデバイスが削除された時に呼び出され、DeviceWatcherInformation class のコレクションに格納されている情報を削除します。	
パラメータ	
DeviceWatcher sender	イベントを送信したオブジェクト。
DeviceInformationUpdate deviceInfoUpdate	削除されたデバイスの DeviceInformationUpdate 情報。

void DeviceWatcher_EnumerationCompleted(DeviceWatcher sender, object e)	
DeviceWatcher の EnumerationCompleted イベントハンドラ。 DeviceWatcher の列挙が完了した時に呼び出されます。本アプリケーションでは処理を行いません。	
パラメータ	
DeviceWatcher sender	イベントを送信したオブジェクト。
object e	イベントのデータ。

void DeviceWatcher_Stopped(DeviceWatcher sender, object e)	
DeviceWatcher の Stopped イベントハンドラ。 DeviceWatcher が停止した時に呼び出されます。 アプリケーションの起動時に DeviceWatcher が開始された場合、既にペアリングされているかチェックを行います。 Pair ボタンのクリックで DeviceWatcher が開始された場合、ペアリングを開始します。 Connect ボタンのクリックで DeviceWatcher が開始された場合、接続を開始します。	
パラメータ	
DeviceWatcher sender	イベントを送信したオブジェクト。
object e	イベントのデータ。

void DeviceWatcherTimer_Callback(object state)	
Timer のコールバックメソッド。 DeviceWatcher の動作時間満了で呼び出され、Timer と DeviceWatcher を停止します。	
パラメータ	
object state	アプリケーションの情報オブジェクト。

DeviceWatcherInformation DeviceWatcherInformation_Search(string id)	
DeviceWatcherInformatio クラスのコレクションの中からデバイスの ID と一致する情報を検索します。	
パラメータ	
string id	デバイスの ID。
リターンパラメータ	
DeviceWatcherInformation	DeviceWatcherInformation クラスのオブジェクト。

DeviceWatcherInformation DeviceWatcherInformation_CheckBDA(string bda)	
DeviceWatcherInformation クラスのコレクションの中から Bluetooth Device Address と一致する情報を検索します。	
パラメータ	
string bda	デバイスの Bluetooth Device Address。
リターンパラメータ	
DeviceWatcherInformation	DeviceWatcherInformation クラスのオブジェクト。

DeviceWatcherInformation DeviceWatcherInformation_CheckPaired()	
DeviceWatcherInformation クラスのコレクションの中からペアリング済みのデバイスを検索します。	
リターンパラメータ	
DeviceWatcherInformation	DeviceWatcherInformation クラスのオブジェクト。

void Connect()
接続処理を行います。評価ボードと接続して、Service や Characteristic を検索し、CCCD へ Notification 許可を設定します。

void Disconnect()
切断処理を行います。

void ConnectionStatusChanged(BluetoothLEDevice sender, object e)	
接続状態の変更を通知するイベントハンドラ。 評価ボードとの切断処理完了で呼び出されます。	
パラメータ	
BluetoothLEDevice sender	イベントを送信したオブジェクト。
object e	イベントのデータ。

void Pair(DeviceInformation deviceInfo)	
ペアリング処理を行います。	
パラメータ	
DeviceInformation deviceInfo	ペアリングするデバイスの DeviceInformation 情報。

void Unpair()
ペアリング解除処理を行います。

void PairingRequestedHandler(DeviceInformationCustomPairing sender, DevicePairingRequestedEventArgs eventArgs)	
ペアリング処理のイベントハンドラ。	
パラメータ	
DeviceInformationCustomPairing sender	イベントを送信したオブジェクト。
DevicePairingRequestedEventArgs eventArgs	イベントデータ。

void ReceiveNotification(GattCharacteristic sender, GattValueChangedEventArgs eventArgs)	
Notification を受信し、受信した回数を SW count テキストボックスに表示します。	
パラメータ	
GattCharacteristic sender	イベントを送信したオブジェクト。
GattValueChangedEventArgs eventArgs	受信データ。

void SendWrite(string str)	
LED Blink Rate テキストボックスの値を Write で送信します。	
パラメータ	
string str	LED Blink Rate 文字列。

4.3.2　AdvertisementWatcherInformation クラス

4.3.2.1　プロパティ

string BluetoothDeviceAddrss
アドバタイジングの Bluetooth Device Address を保存します。DeviceWatcher で列挙されたデバイスから接続対象の評価ボードを検索する時に使用します。
string Name
アドバタイジングの Local Name を保存します。接続対象の評価ボードを判定する時に使用します。

4.3.2.2　メソッド

void Clear()
プロパティを初期化します。

4.3.3　DeviceWatcherInformation クラス

4.3.3.1　コンストラクタ

public DeviceWatcherInformation(DeviceInformation deviceInfo)	
deviceInformation プロパティに DeviceInformation 情報を設定します。	
パラメータ	
DeviceInformation deviceInfo	DeviceInformation 情報。

4.3.3.2　プロパティ

DeviceInformation DeviceInformation
DeviceInformation 情報を保存します。

4.3.3.3　フィールド(メンバ変数)

string Id
列挙されたデバイスの ID を保存します。
string Name
列挙されたデバイスの名前を保存します。
bool IsPaired
列挙されたデバイスがペアリングされているか示します。

4.3.3.4 メソッド

void Update(DeviceInformationUpdate deviceInfoUpdate)	
deviceInformation プロパティを更新します。	
パラメータ	
DeviceInformationUpdate deviceInfoUpdate	更新されたデバイスの DeviceInformationUpdate 情報。

4.3.4 ViewModel クラス

4.3.4.1 フィールド(メンバ変数)

event PropertyChangedEventHandler PropertyChanged
PropertyChanged イベントを宣言します。

4.3.4.2 コンストラクタ

ViewModel()
プロパティを初期化します。

4.3.4.3 プロパティ

string textbox_status
Status テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_status フィールドはデータの保持に使用します。
string button_pair_content
Pair/Unpair ボタンへの文字の表示や、文字の取得に使用します。 string _button_pair_content フィールドはデータの保持に使用します。
string button_connect_content
Connect/Disconnect ボタンへの文字の表示や、文字の取得に使用します。 string _button_connect_content フィールドはデータの保持に使用します。
string textbox_sw_count
SW Count テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_sw_count フィールドはデータの保持に使用します。
string textbox_led_blink_rate
LED Blink Rate テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_led_blink_rate フィールドはデータの保持に使用します。
bool button_pair_isenabled
Pair/Unpair ボタンの有効化/無効化に使用します。 bool _button_pair_isenabled フィールドはデータの保持に使用します。

bool button_connect_isenabled
Connect/Disconnect ボタンの有効化/無効化に使用します。 bool _button_connect_isenabled フィールドはデータの保持に使用します。
bool button_led_blink_rate_isenabled
LED Blink Rate 送信ボタンの有効化/無効化に使用します。 bool _button_led_blink_rate_isenabled フィールドはデータの保持に使用します。

4.3.4.4　メソッド

void NotifyPropertyChanged(string propName)	
PropertyChanged イベントで、各プロパティとバインディングされている UI へデータの変更を通知します。	
パラメータ	
string propName	プロパティ名文字列。

4.3.5　フローチャート

LED Switch Service Client の起動や UI を操作した時の動作をフローチャート用いて説明します。

4.3.5.1　アプリケーション起動

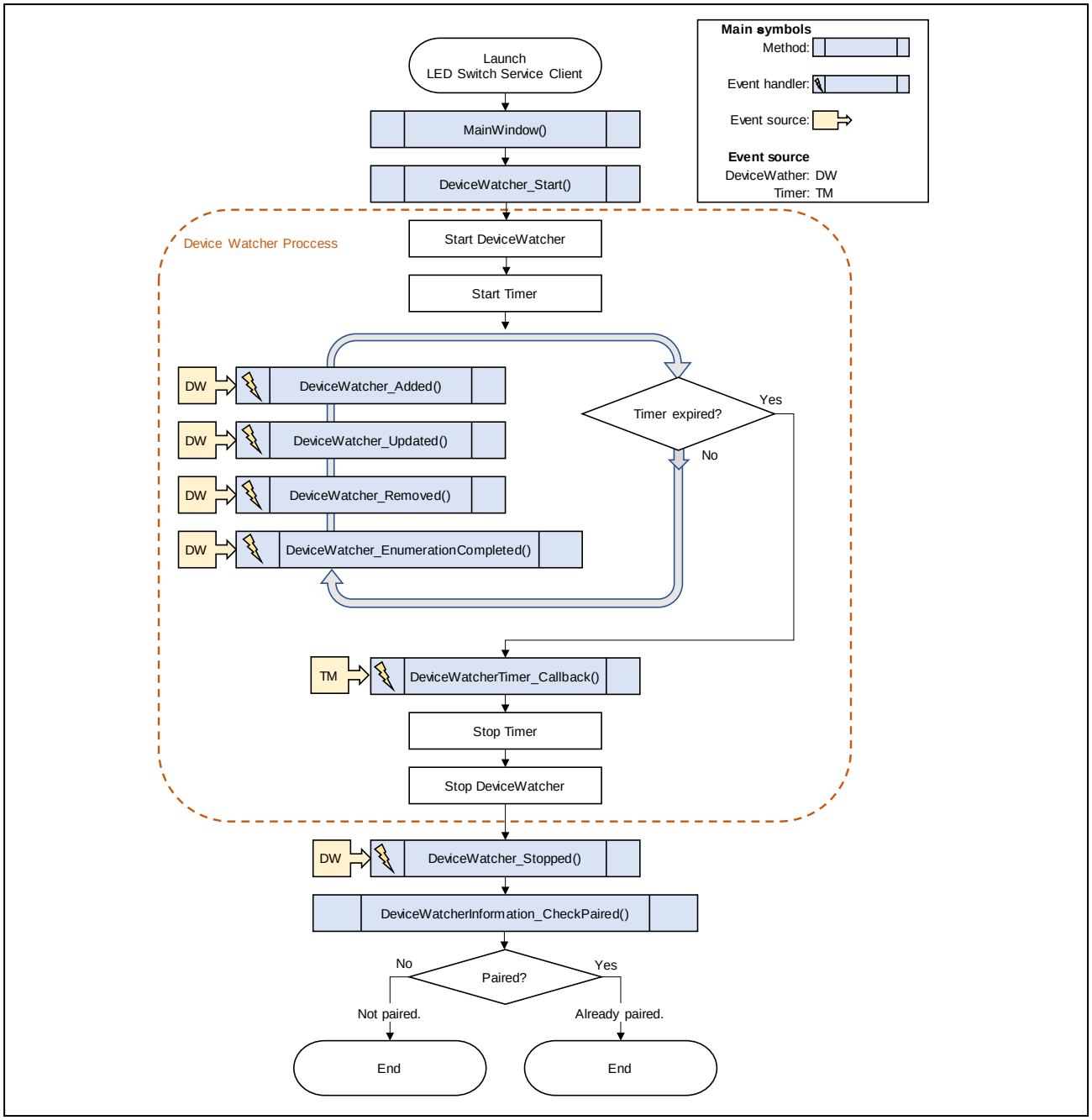

図 4-1 アプリケーション起動フローチャート

アプリケーションを起動すると、評価ボードとペアリング済みかどうかチェックします。DeviceWatcher クラスを使用して Windows のシステムに登録されているデバイスを列挙し、評価ボードとのペアリング情報が保存されているか検索します。

ペアリング情報が無かった場合、ペアリングを実行するための Pair ボタンを有効にします。

ペアリング情報が見つかった場合、評価ボードと接続するための Connect ボタンと、ペアリングを解除するための Unpair ボタンを有効にします。

4.3.5.2 Pair/Unpair ボタン

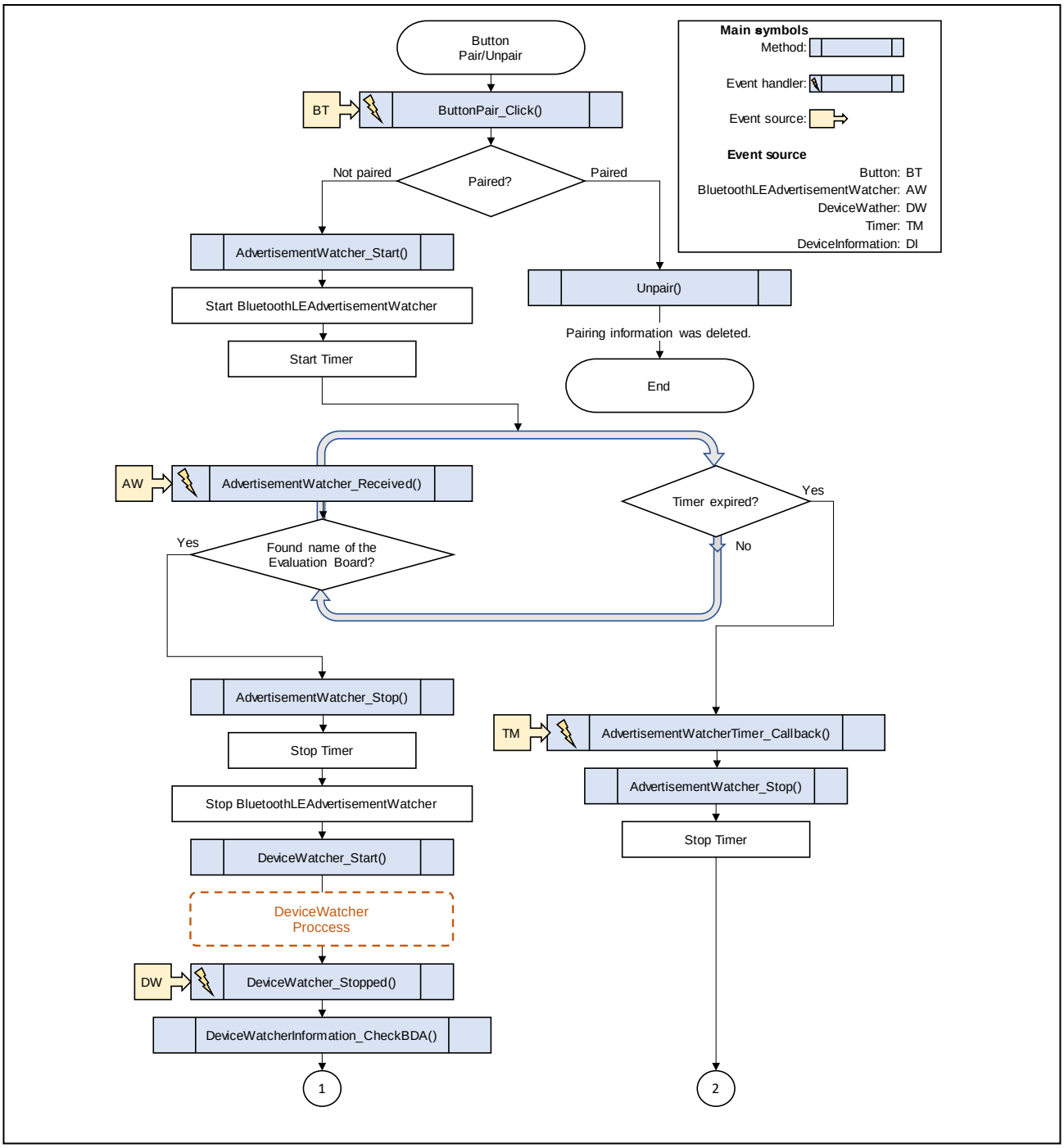

図 4-2 Pair/Unpair ボタンフローチャート(1)

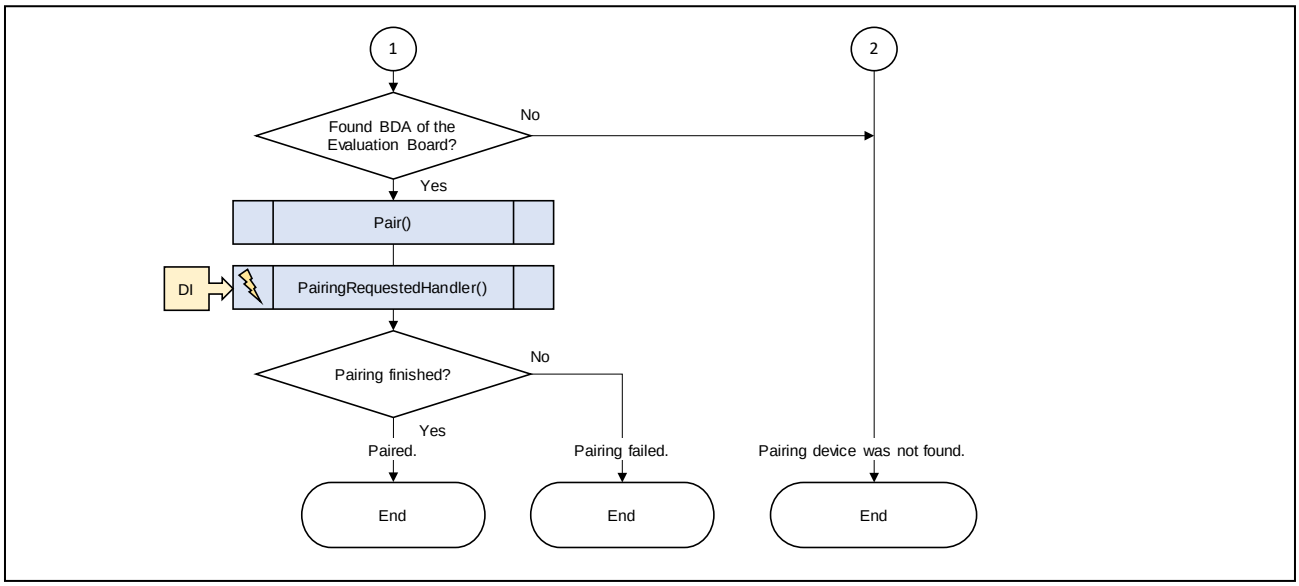

図 4-3 Pair/Unpair ボタンフローチャート(2)

評価ボードとペアリングしていない場合は Pair ボタン、評価ボードとペアリングしている場合は Unpair ボタンとして動作します。

Pair ボタンがクリックされると、評価ボードを見つけるために BluetoothLEAdvertisementWatcher クラスを使用してスキャンを開始し、評価ボードが送信するアドバタイジングを受信します。アドバタイジングデータの中から評価ボードの名前が見つかると、DeviceWatcher クラスを使用して周辺のデバイスや Windows システムに登録されているデバイスを列挙します。列挙されたデバイスの中にスキャンで見つかった評価ボードの Bluetooth Device Address と一致するデバイスが見つかるとペアリングを開始します。ペアリングは DeviceInformation クラスの PairAsync メソッドを使用します。ペアリング処理が進むと PairingRequested イベントが発生し、登録した PairingRequestedHandler イベントハンドラが呼び出されます。イベントのパラメータで渡される DevicePairingRequestedEventArgs クラスの Accept メソッドを実行すると評価ボードとのペアリングが完了し、Windows システムにペアリング情報が登録されます。

Unpair ボタンがクリックされると、DeviceInformation クラスの UnpairAsync メソッドを使用してペアリング情報を Windows システムから削除します。

4.3.5.3 Connect/Disconnect ボタン

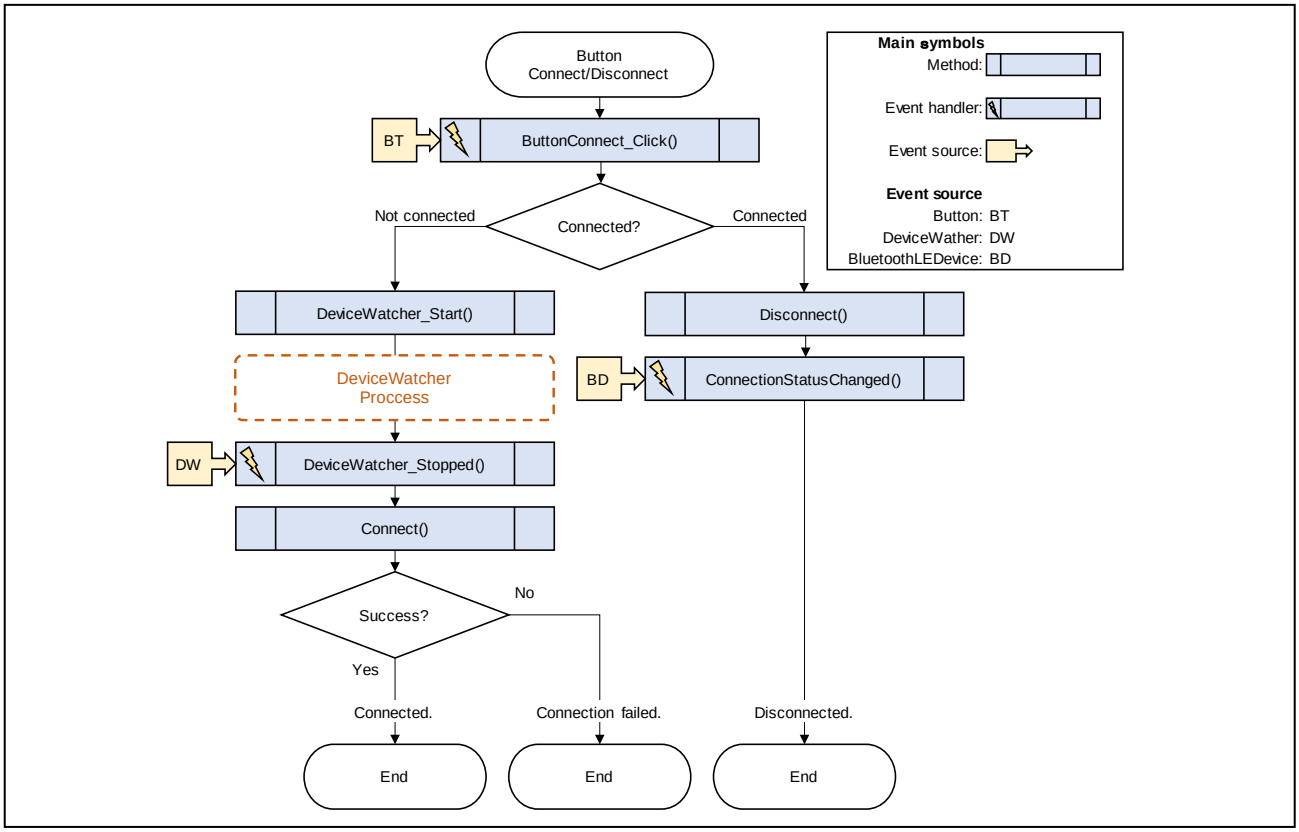

図 4-4 Connect/Disconnect ボタンフローチャート

評価ボードと接続していない場合は Connect ボタン、評価ボードと接続している場合は Disconnect ボタンとして動作します。

Connect ボタンがクリックされると、DeviceWatcher を開始して DeviceWatcherInformation クラスのペアリング情報を更新し、更新されたペアリング情報を使用して評価ボードと接続します。接続できると LED Switch Service、Notify characteristic、Write characteristic を検索し、データ通信で使用するために Service や Characteristic のオブジェクトをフィールドに保存します。また、評価ボードが Notification を送信できるようにするために、Client Characteristic Configuration Descriptor へ Notification 許可を書き込みます。

Disconnect ボタンがクリックされると、Characteristic、Service、BluetoothLEDevice の順番に Dispose を実行しリソースを解放します。リソースを解放すると BluetoothLEDevice クラスの ConnectionStatusChanged イベントが発生し評価ボードとの接続が切断されます。切断後の処理として ConnectionStatusChanged イベントハンドラの登録を解除し、Service や Characteristic を保存しているフィールドを初期化します。

4.3.5.4　LED Blink Rate ボタン(Write 送信)

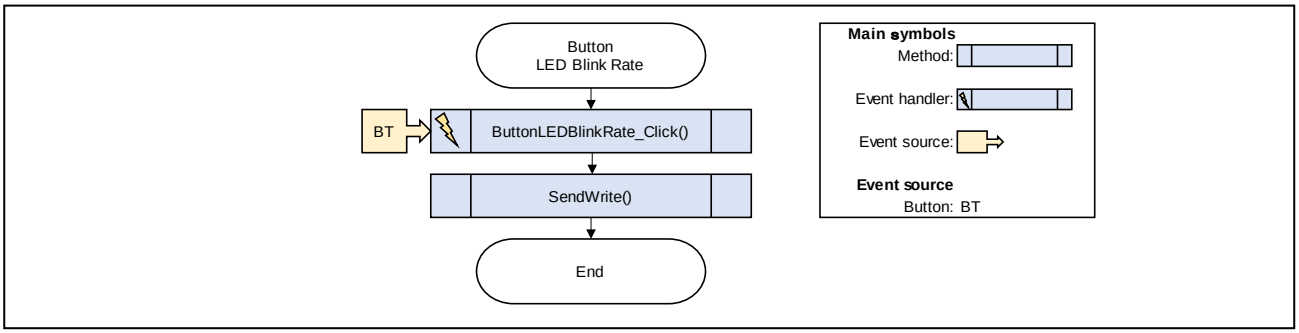

図 4-5 LED Blink Rate ボタン(Write 送信)フローチャート

LED Blink Rate テキストボックスに入力されたデータを評価ボードへ送信します。データは GattCharacteristic クラスの WriteValueAsync メソッドで送信します。

4.3.5.5　SW State 受信(Notification 受信)

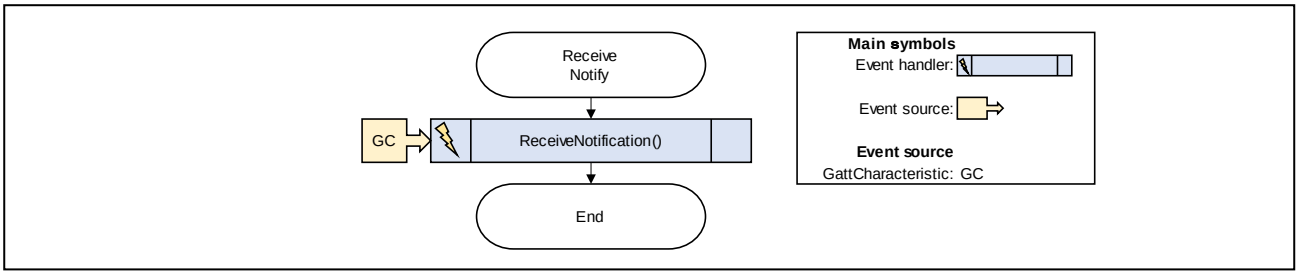

図 4-6 SW State 受信(Notification 受信)フローチャート

評価ボードからの Notification を受信すると、GattCharacteristic クラスの ValueChanged イベントが発生します。イベントのパラメータで SW State のデータを受け取り、SW Count テキストボックスに Notification の受信回数を表示します。

4.3.5.6　評価ボードからの切断

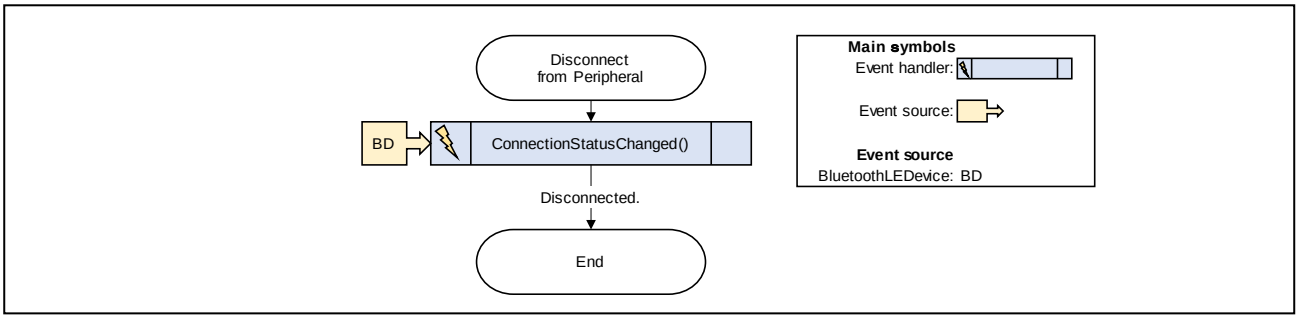

図 4-7 評価ボードからの切断フローチャート

評価ボードの電源断や、評価ボードをリセットすることで通信できない状態となった場合に BluetoothLEDevice クラスの ConnectionStatusChanged イベントが発生します。

切断処理として、ConnectionStatusChanged イベントハンドラの登録を解除し、Service や Characteristic を保存しているフィールドを初期化します。

4.4 Virtual UART Client

Virtual UART Client で定義されるクラスや処理の内容を説明します。

表 4-2 Virtual UART Client のクラス概要

クラス名	説明
MainWindow	LED Switch Service Client のアプリケーション処理が記述されたクラス。
AdvertisementWatcherInformation	接続対象評価ボードのアドバタイジングから得た情報を保存するクラス。
ViewModel	プログラムと UI の間のデータ交換で使用するクラス。

4.4.1 MainWindow クラス

4.4.1.1 フィールド(メンバ変数)

Guid UUID_VUART_SERVICE
仮想 UART プロファイルの Primary Service UUID を定義。評価ボードと接続した後、Service の検索で使用します。
Guid UUID_VUART_INDICATION
仮想 UART プロファイルの Indication Characteristic UUID を定義。評価ボードと接続した後、Characteristic の検索で使用します。
Guid UUID_VUART_WRITE
仮想 UART プロファイルの Write Characteristic UUID を定義。評価ボードと接続した後、Characteristic の検索で使用します。
const int VUART_WRITE_LENGTH
Virtual UART Client から評価ボードへの送信データ数の判定に使用します。
BluetoothLEAdvertisementWatcher advertisementWatcher
BluetoothLEAdvertisementWatcher クラスのオブジェクトを保存し、スキャンを実行してアドバタイジングを受信するために使用します。
AdvertisementWatcherInformation vuartDevice
AdvertisementWatcherInformation クラスのオブジェクトを保存し、仮想 UART プロファイルを搭載した評価ボードの情報を格納します。
BluetoothLEDevice bluetoothLeConnectedDevice
評価ボードと接続を開始する際に、BluetoothLEDevice クラスのオブジェクトを保存します。
GattDeviceService gattPrimaryService
評価ボードと接続した後、仮想 UART プロファイル Primary Service の BluetoothLEDevice クラスのオブジェクトを保存します。
GattCharacteristic characteristicIndicate
評価ボードと接続した後、仮想 UART プロファイル Indication Characteristic の GattCharacteristic クラスのオブジェクトを保存します。

GattCharacteristic characteristicWrite
評価ボードと接続した後、仮想 UART プロファイル Write Characteristic の GattCharacteristic クラスのオブジェクトを保存します。
bool isFoundVuartDevice
BluetoothLEAdvertisementWatcher 停止時に、接続処理への移行判定に使用します。本フラグは Connect ボタンをクリックした時にセットします。
ViewModel viewModel
ViewModel class のオブジェクトを保存し、プログラムと UI の間のデータ交換に使用します。
Timer timer
Timer クラスのオブジェクトを保存し、BluetoothLEAdvertisementWatcher の動作停止(タイムアウト)に使用します。
StreamWriter streamWriter
ログファイルに書き込みを行う StreamWriter クラスのオブジェクトを保存します。
string pathLogFile
ログファイルのパス情報を保存します。
StreamWriter streamReader
送信ファイルから読み出しを行う StreamReader クラスのオブジェクトを保存します。
string pathSendFile
送信ファイルのパス情報を保存します。

4.4.1.2　コンストラクタ

MainWindow()
UI とのデータ交換を行うことができるようにするため、ViewModel class をインスタンス化し DataContext プロパティへセットします。

4.4.1.3　メソッド/イベントハンドラ

void ButtonConnect_Click(object sender, RoutedEventArgs e)	
Connect/Disconnect ボタンのクリックイベントハンドラ。 Connect ボタンの時は、接続する評価ボードを検索するため BluetoothLEAdvertisementWatcher を開始します。Disconnect ボタンの時は、評価ボードとの接続を切断します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void ButtonLog_Click(object sender, RoutedEventArgs e)	
Log ボタンのクリックイベントハンドラ。 ファイル保存ダイアログを表示し、指定されたファイルの StreamWrite クラスのインスタンスを作成します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void ButtonFile_Click(object sender, RoutedEventArgs e)	
File ボタンのクリックイベントハンドラ。 ファイルを開くダイアログを表示し、指定されたファイルの文字データを評価ボードへ送信します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void ButtonSend_Click(object sender, RoutedEventArgs e)	
Send ボタンのクリックイベントハンドラ。 Send データ入力テキストボックスに入力された文字データを評価ボードへ送信します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void TextBoxSend_KeyDown(object sender, RoutedEventArgs e)	
Send データ入力テキストボックスのキーダウンイベントハンドラ。 Send データ入力テキストボックスの中で Enter(Return)キーを押すと、テキストボックスに入力された文字データを評価ボードへ送信します。	
パラメータ	
object sender	イベントを送信したオブジェクト。
RoutedEventArgs e	イベントのパラメータ。

void SendTextBoxData()
Send データ入力テキストボックスに入力されたデータの文字数をチェックして送信します。

void AdvertisementWatcher_Start(int timeout)	
BluetoothLEAdvertisementWatcher でスキャンを実行します。BluetoothLEAdvertisementWatcher は、接続する評価ボードを発見した場合、または評価ボードがタイムアウト時間内に見つからなかった場合に終了します。	
パラメータ	
int timeout	タイムアウト時間 (単位: ミリ秒)。

void AdvertisementWatcher_Stop()
BluetoothLEAdvertisementWatcher を停止します。接続する評価ボードが見つかった場合は、評価ボードと接続を開始します。

void AdvertisementWatcher_Received(BluetoothLEAdvertisementWatcher watcher, BluetoothLEAdvertisementReceivedEventArgs eventArgs)	
BluetoothLEAdvertisementWatcher のアドバタイジング受信イベントハンドラ。 アドバタイジングを受信するごとに呼び出されます。Bluetooth Device Address テキストボックスに記入されたアドレスの評価ボード、または仮想 UART プロファイルの UUID を持つ評価ボードを発見した場合、BluetoothLEAdvertisementWatcher を停止します。	
パラメータ	
BluetoothLEAdvertisementWatcher watcher	イベントを送信した BluetoothLEAdvertisementWatcher のオブジェクト。
BluetoothLEAdvertisementReceivedEventArgs eventArgs	受信したイベントのデータ。

void AdvertisementWatcherTimer_Callback(object state)	
Timer のコールバックメソッド。 BluetoothLEAdvertisementWatcher の動作時間満了で呼び出され、BluetoothLEAdvertisementWatcher を停止します。	
パラメータ	
object state	アプリケーションの情報オブジェクト。

void Connect()
接続処理を行い、Service や Characteristic を検索し、CCCD へ Indicate 許可を設定します。

void Disconnect()
切断処理を行います。

void ConnectionStatusChanged(BluetoothLEDevice sender, object e)	
接続状態の変更を通知するイベントハンドラ。 評価ボードとの切断処理完了で呼び出されます。	
パラメータ	
BluetoothLEDevice sender	イベントを送信したオブジェクト。
object e	イベントのデータ。

void ReceiveIndiation(GattCharacteristic sender, GattValueChangedEventArgs eventArgs)	
Indicate を受信し、受信した文字列データを Receive データ表示テキストボックスに表示します。	
パラメータ	
GattCharacteristic sender	イベントを送信したオブジェクト。
GattValueChangedEventArgs eventArgs	受信データ。

void SendWrite(string str)	
Write で文字データを送信します。	
パラメータ	
string str	文字データ。

void FileAppend(string str)	
文字列をファイルに追加します。	
パラメータ	
string str	ファイルに追加する文字列。

void FileSend()
ファイルから文字列データを読み出し、Write で評価ボードへ送信します。

4.4.2　AdvertisementWatcherInformation クラス

4.4.2.1　プロパティ

string BluetoothDeviceAddrss
アドバタイジングの Bluetooth Device Address(文字列)を保存します。接続対象の評価ボードを判定するときに使用します。
string Name
アドバタイジングの Local Name を保存します。
Guid ServiceUuid
アドバタイジングの 128bit UUID を保存します。接続対象の評価ボードを判定するときに使用します。

ulong BluetoothDeviceAddrss_ulong
アドバタイジングの Bluetooth Device Address(数値)を保存します。接続開始時に接続対象を指定するパラメータとして使用します。

4.4.2.2　メソッド

void Clear()
プロパティを初期化します。

4.4.3　ViewModel クラス

4.4.3.1　フィールド(メンバ変数)

event PropertyChangedEventHandler PropertyChanged
PropertyChanged イベントを宣言します。

4.4.3.2　コンストラクタ

ViewModel()
プロパティを初期化します。

4.4.3.3　プロパティ

string textbox_status
Status テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_status フィールドはデータの保持に使用します。
string textbox_bdaddress
Bluetooth Device Address テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_bdaddress フィールドはデータの保持に使用します。
string textbox_receive_text
Rceive データ表示テキストボックスへの文字の表示や、文字の取得に使用します。 string _textbox_receive_text フィールドはデータの保持に使用します。
string button_connect_content
Connect/Disconnect ボタンへの文字の表示や、文字の取得に使用します。 string _button_connect_content フィールドはデータの保持に使用します。
string button_log_content
Receive データ保存ボタンへの文字の表示や、文字の取得に使用します。 string _button_log_content フィールドはデータの保持に使用します。
bool button_connect_isenabled
Connect/Disconnect ボタンの有効化/無効化に使用します。 bool _button_connect_isenabled フィールドはデータの保持に使用します。

bool button_log_isenabled
Receive データ保存ボタンの有効化/無効化に使用します。 bool _button_log_isenabled フィールドはデータの保持に使用します。
bool button_file_isenabled
Send ファイルボタンの有効化/無効化に使用します。 bool _button_file_isenabled フィールドはデータの保持に使用します。
bool button_send_isenabled
Send データボタンの有効化/無効化に使用します。 bool _button_send_isenabled フィールドはデータの保持に使用します。

4.4.3.4　メソッド

void NotifyPropertyChanged(string propName)	
PropertyChanged イベントで、各プロパティとバインディングされている UI へデータの変更を通知します。	
パラメータ	
string propName	プロパティ名文字列。

4.4.4　フローチャート

Virtual UART Client の起動や UI を操作した時の動作をフローチャート用いて説明します。

4.4.4.1　アプリケーション起動

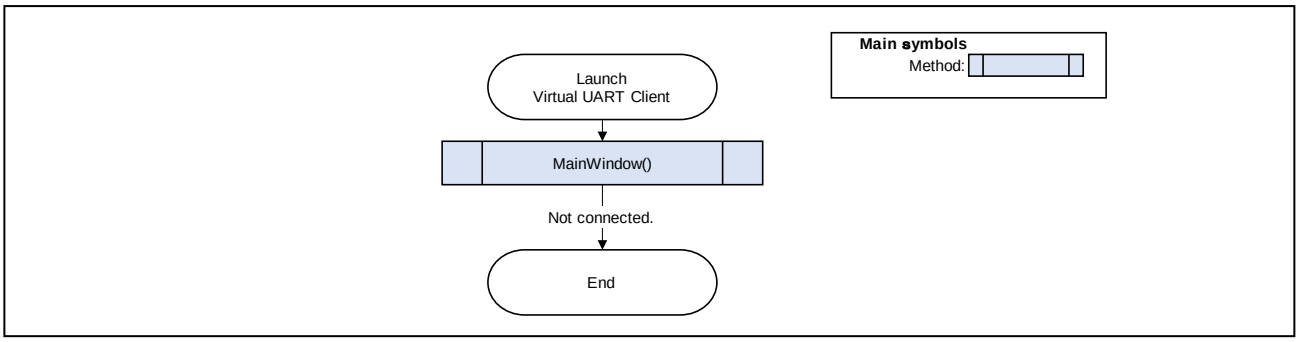

図 4-8 アプリケーション起動フローチャート

アプリケーションを起動すると Connect ボタンがクリックされることを待つ状態となります。

4.4.4.2 Connect/Disconnect ボタン

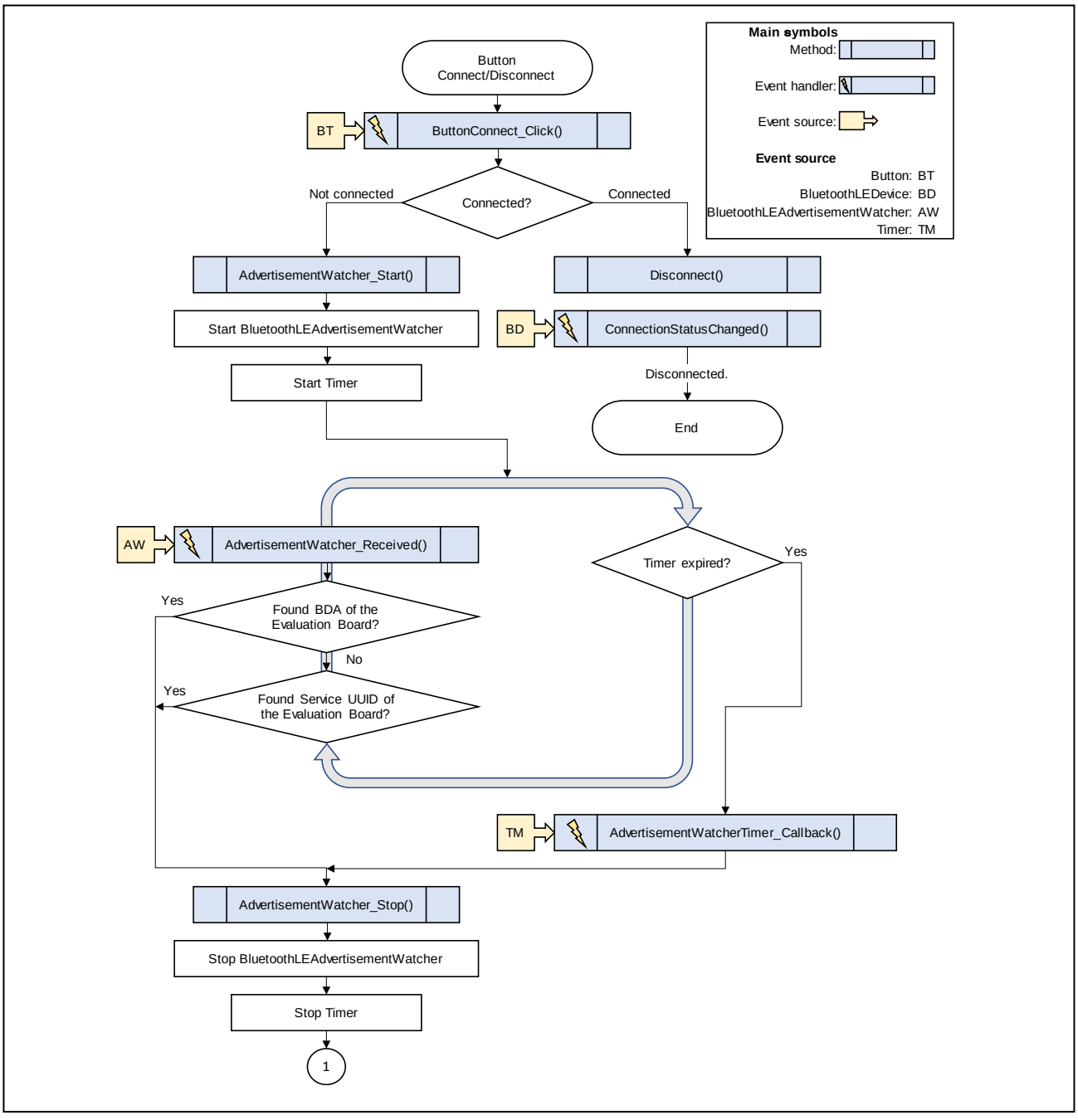

図 4-9 Connect/Disconnect ボタンフローチャート(1)

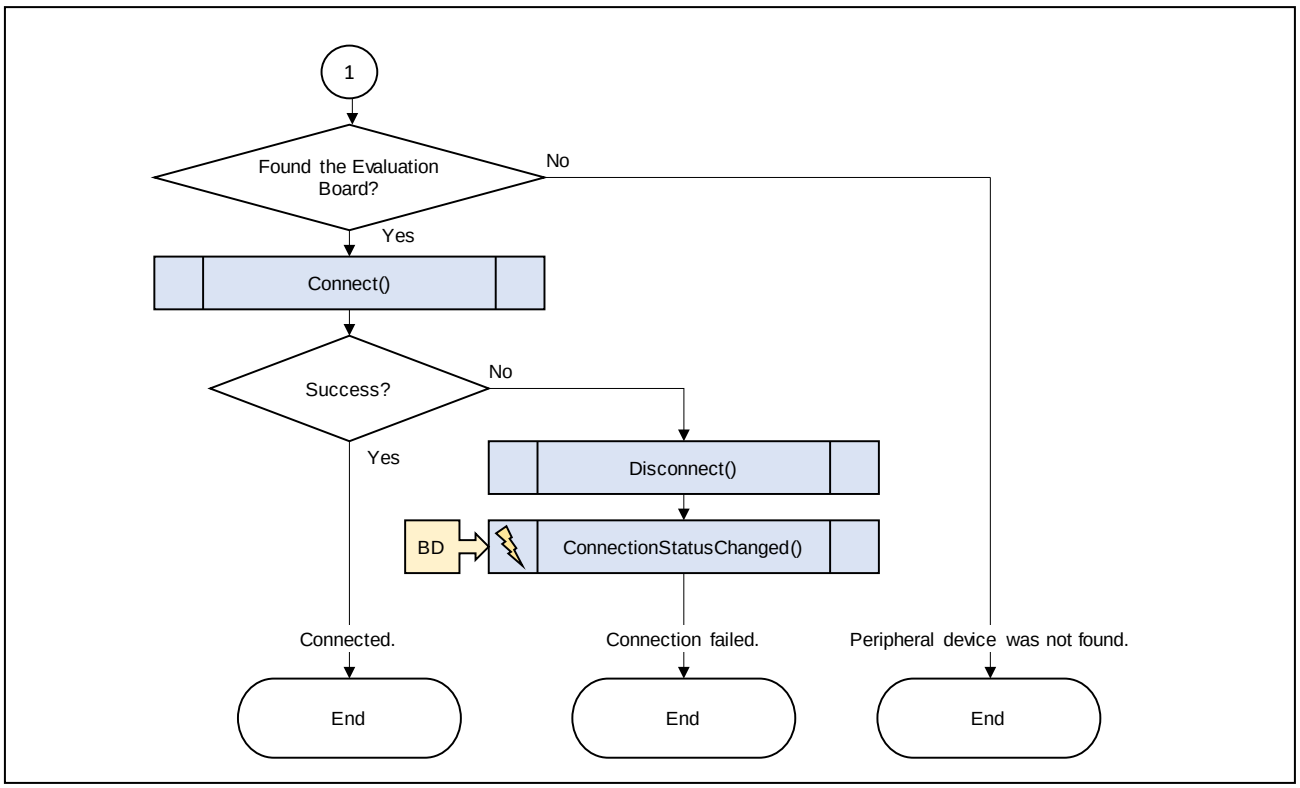

図 4-10 Connect/Disconnect ボタンフローチャート(2)

評価ボードと接続していない場合は Connect ボタン、評価ボードと接続している場合は Disconnect ボタンとして動作します。

Connect ボタンがクリックされると、BluetoothLEAdvertisementWatcher でスキャンを開始しアドバタイジングを受信します。Bluetooth Device Address テキストボックスに記入されたアドレス、または仮想 UART プロファイルの UUID を持つアドバタイジングを発見すると接続を行います。

Disconnect ボタンがクリックされると、Characteristic、Service、BluetoothLEDevice の順番に Dispose を実行しリソースを解放します。リソースを解放すると BluetoothLEDevice クラスの ConnectionStatusChanged イベントが発生し評価ボードとの接続が切断されます。切断後の処理として ConnectionStatusChanged イベントハンドラの登録を解除し、Service や Characteristic を保存しているフィールドを初期化します。

4.4.4.3 Log ボタン

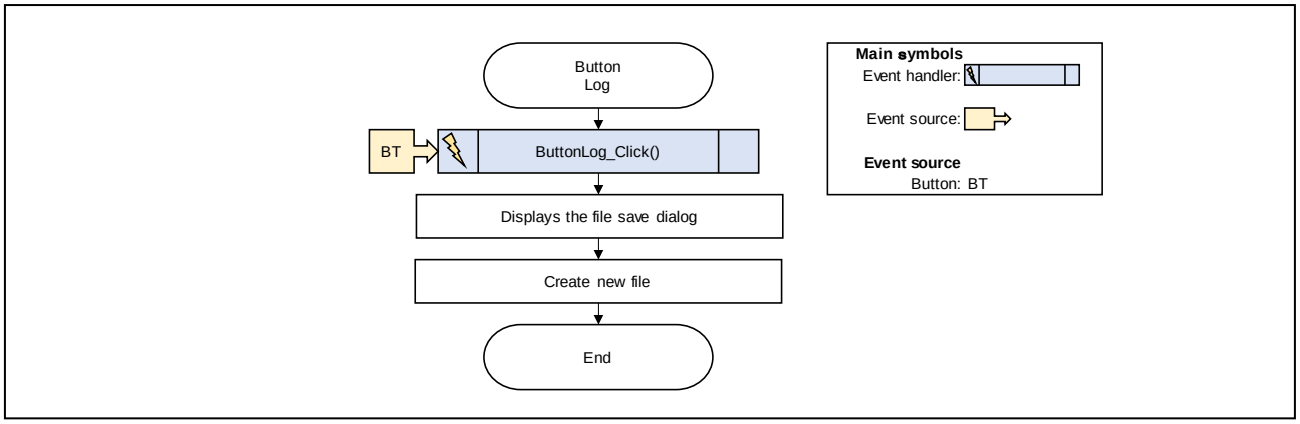

図 4-11 Log ボタンフローチャート

ログを記録していない場合は Log ボタン、ログを記録している場合は Logging ボタン(ログ停止ボタン)として動作します。

Log ボタンがクリックされると、ファイル保存ダイアログを表示してファイルのパスをフィールドに保存し、新しいファイルを作成します。

Logging ボタンがクリックされると、文字データを受信してもログが記録されないようにファイルのパスを保存しているフィールドを初期化します。

4.4.4.4 File ボタン(Write 送信)

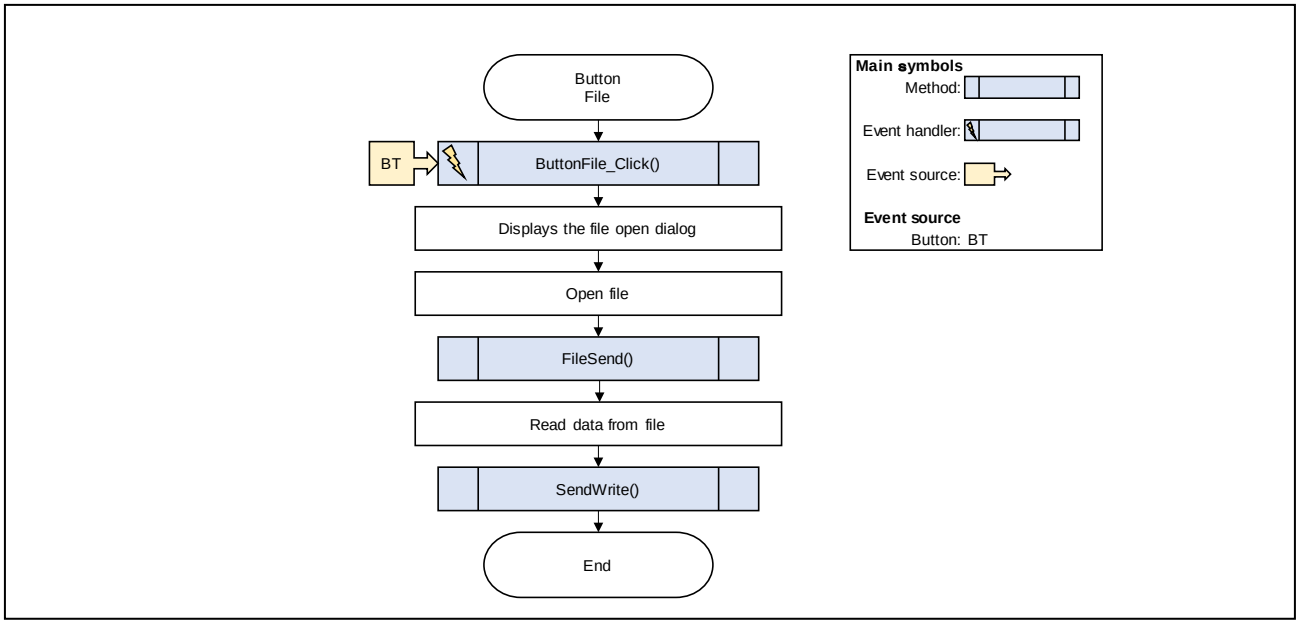

図 4-12 File ボタンフローチャート

ファイルを開くダイアログを表示し、指定されたファイルの文字データを評価ボードへ送信します。データは GattCharacteristic クラスの WriteValueAsync メソッドで 20 バイトごとに送信します。

4.4.4.5 Send ボタン(Write 送信)

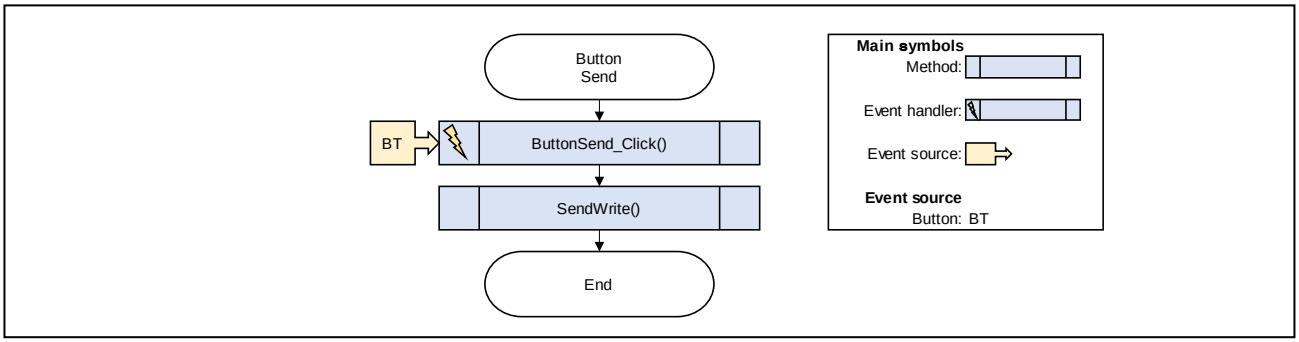

図 4-13 Send ボタンフローチャート(Write 送信)

Send データテキストボックスに入力されたデータを評価ボードへ送信します。データは GattCharacteristic クラスの WriteValueAsync メソッドで送信します。送信できるデータ長は 20 バイトです。

4.4.4.6 Send テキストボックス・キーダウン(Write 送信)

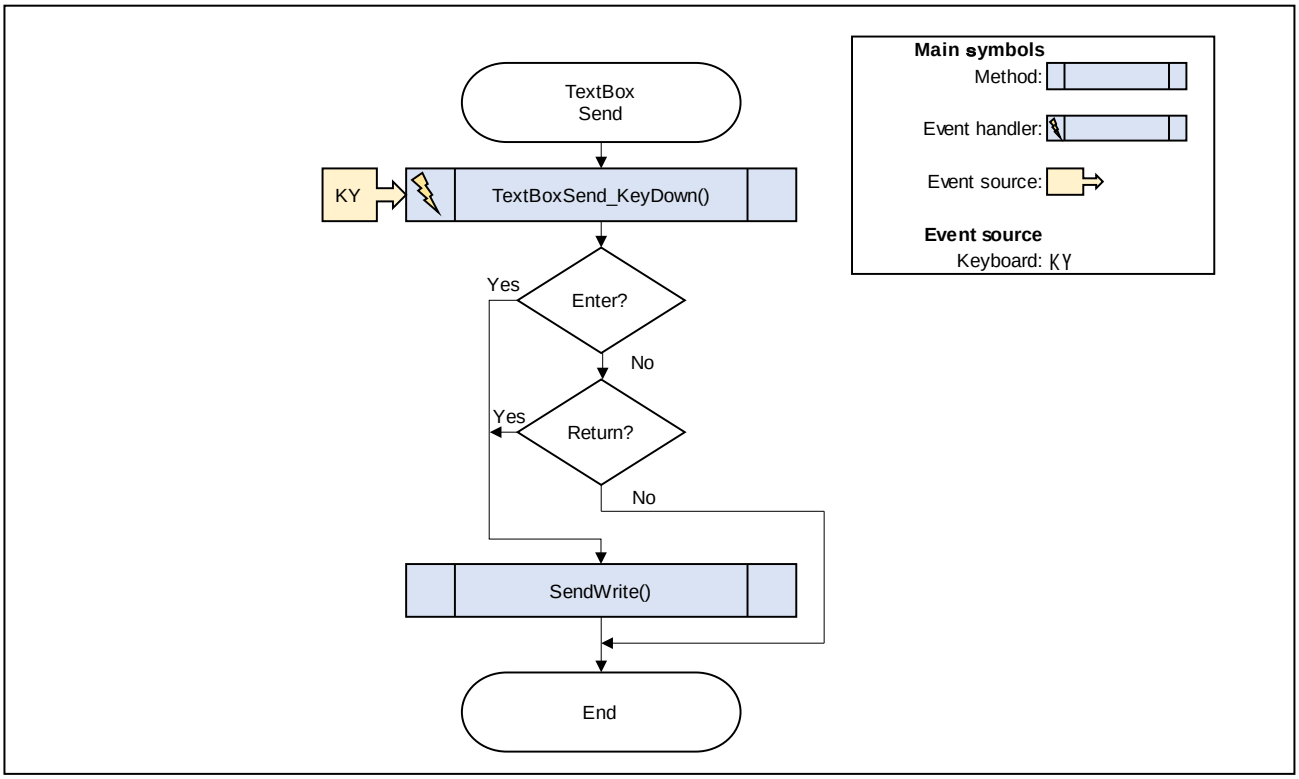

図 4-14 Send テキストボックス・キーダウン(Write 送信)

Send データテキストボックスに入力されたデータを Enter キー、または Return キーの押下で評価ボードへ送信します。データは GattCharacteristic クラスの WriteValueAsync メソッドで送信します。送信できるデータ長は 20 バイトです。

4.4.4.7　文字データ受信(Indication 受信)

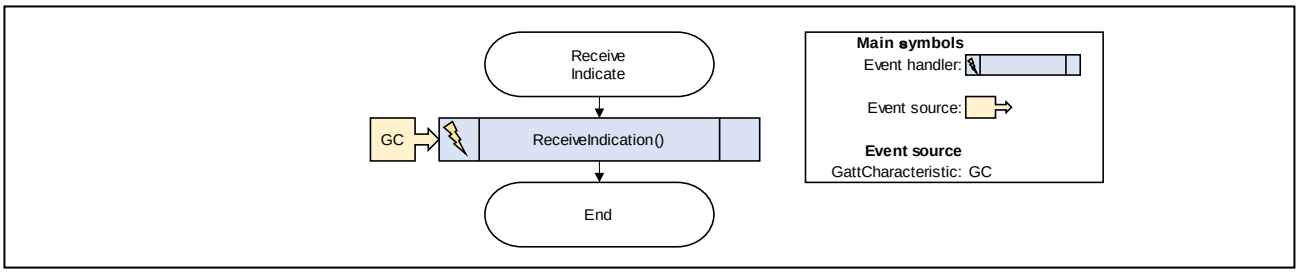

図 4-15 文字データ受信(Indication 受信)

評価ボードからの Indication を受信すると、GattCharacteristic クラスの ValueChanged イベントが発生します。イベントのパラメータで文字データを受け取り、Receive テキストボックスに受信した文字データを表示します。

4.4.4.8　評価ボードからの切断

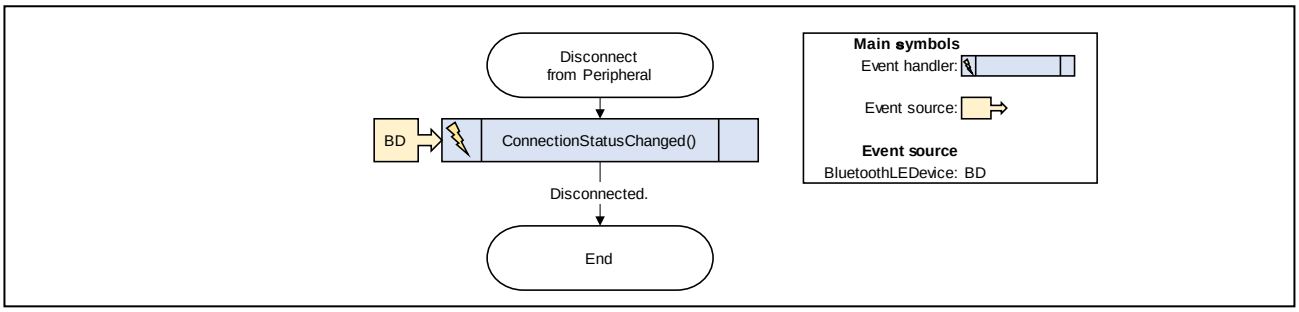

図 4-16 評価ボードからの切断

評価ボードの電源断や、評価ボードをリセットすることで通信できない状態となった場合に BluetoothLEDevice クラスの ConnectionStatusChanged イベントが発生します。

切断処理として、ConnectionStatusChanged イベントハンドラの登録を解除し、Service や Characteristic を保存しているフィールドを初期化します。

4.5　Windows 10 の Bluetooth LE 通信

LED Switch Service Client のソースコードから一部抜粋して、Windows10 の Bluetooth LE 通信に関わるソースコードについて説明します。Virtual UART Client の Bluetooth LE 通信に関わるソースコードは、LED Switch Service Client のソースコードとほぼ同一であることからここでの説明は省略します。

本節に掲載しているソースコードは、デバッグ出力や Bluetooth LE 通信に不要なコードを削除しています。

4.5.1　BluetoothLEAdvertisementWatcher

BluetoothLEAdvertisementWatcher はスキャンを実行して Bluetooth LE 機器が送信するアドバタイジング(ビーコン)を受信することができるクラスです。

スキャンを開始するには、BluetoothLEAdvertisementWatcher のオブジェクトを作成し、スキャンを行うモード(Active モード)を設定、アドバタイジングを受信した時に呼び出されるイベントハンドラ(AdvertisementWatcher_Received)を登録します。そして Start メソッドを呼び出すとスキャンが開始されます。

```
private void AdvertisementWatcher_Start(int timeout)
{
    advertisementWatcher = new BluetoothLEAdvertisementWatcher();
    advertisementWatcher.ScanningMode = BluetoothLEScanningMode.Active;
    advertisementWatcher.Received += AdvertisementWatcher_Received;
    advertisementWatcher.Start();
}
```

図 4-17 BluetoothLEAdvertisementWatcher 開始

アドバタイジングを受信した時に呼び出されます。ここでは、受信したアドバタイジングのデータから接続対象の評価ボードの名前を判定し、接続対象である場合は必要なパラメータを保存しています。

```
public void AdvertisementWatcher_Received(BluetoothLEAdvertisementWatcher watcher,
                                           BluetoothLEAdvertisementReceivedEventArgs eventArgs)
{
    if (advertisementWatcher != null)
    {
        if (eventArgs.Advertisement.LocalName == DEVNAME_RBLE)
        {
            peripheralDevice.BluetoothDeviceAddrss = String.Format("{0:X}", eventArgs.BluetoothAddress);
            peripheralDevice.Name = eventArgs.Advertisement.LocalName;
        }
    }
}
```

図 4-18 アドバタイジング受信イベントハンドラ

スキャンを停止する時は Stop メソッドを呼び出します。イベントハンドラを解除し、BluetoothLEAdvertisementWatcher のオブジェクトを保存していたフィールドを初期化します。

```
private void AdvertisementWatcher_Stop()
{
    advertisementWatcher.Stop();
    advertisementWatcher.Received -= AdvertisementWatcher_Received;
    advertisementWatcher = null;
}
```

図 4-19 BluetoothLEAdvertisementWatcher 停止

4.5.2 DeviceWatcher

DeviceWatcher は、Windows のシステムに登録されているデバイスやシステムに登録するデバイスを検知して動的に列挙するクラスです。BluetoothLEAdvertisementWatcher との違いは、ペアリング済みでシステムに登録されているデバイスも列挙することができます。BluetoothLEAdvertisementWatcher のように何度もアプリケーションへ通知するのではなく 1 度だけ通知し、通知した情報に変化があった場合は再度通知します。

DeviceWatcher を開始するには、DeviceWatcher を Bluetooth LE 機器を対象として使用するためのパラメータを設定し、CreateWatcher メソッドに渡してオブジェクトを作成します。そして通知を受け取るイベントハンドラを登録した後、Start メソッドを呼び出すと DeviceWatcher が開始されます。

```
private void DeviceWatcher_Start(int timeout)
{
    // DeviceWatcher Bluetooth LE setting.
    string aqsAllBluetoothLEDevices =
        "(System.Devices.Aep.ProtocolId:=¥"{bb7bb05e-5972-42b5-94fc-76eaa7084d49}¥")";

    string[] requestedProperties = {"System.Devices.Aep.DeviceAddress",
                                    "System.Devices.Aep.IsConnected",
                                    "System.Devices.Aep.Bluetooth.Le.IsConnectable"};

    deviceWatcher = DeviceInformation.CreateWatcher(aqsAllBluetoothLEDevices,
                                                    requestedProperties,
                                                    DeviceInformationKind.AssociationEndpoint);

    // Register event handlers before starting the watcher.
    deviceWatcher.Added   += DeviceWatcher_Added;
    deviceWatcher.Updated += DeviceWatcher_Updated;
    deviceWatcher.Removed += DeviceWatcher_Removed;
    deviceWatcher.EnumerationCompleted += DeviceWatcher_EnumerationCompleted;
    deviceWatcher.Stopped += DeviceWatcher_Stopped;

    // Start DeviceWatcher.
    deviceWatcher.Start();
}
```

図 4-20 DeviceWatcher 開始

デバイスの情報が Windows のシステムに追加された時に呼び出されます。通知された情報から接続対象の評価ボードの名前を判定し、接続対象の評価ボード名かつ RBLEDevices コレクションに同一の Id を持つ情報が入っていない場合、RBLEDevices コレクションに追加します。

```
private void DeviceWatcher_Added(DeviceWatcher sender, DeviceInformation deviceInfo)
{
    if (deviceInfo.Name == DEVNAME_RBLE)
    {
        DeviceWatcherInformation device = DeviceWatcherInformation_Search(deviceInfo.Id);
        if (device == null)
        {
            // The deviceInfo.Id was not found in the RBLEDevices ObservableCollection.

            // Add the found RBLE device to the RBLEDevices ObservableCollection.
            RBLEDevices.Add(new DeviceWatcherInformation(deviceInfo));
        }
    }
}
```

図 4-21 DeviceWatcher 情報追加イベントハンドラ

デバイスの情報が Windows のシステムで更新された時に呼び出されます。通知された情報から RBLEDevices コレクションに同一の Id を持つ情報が入っている場合、RBLEDevices コレクションを更新します。

```
private void DeviceWatcher_Updated(DeviceWatcher sender, DeviceInformationUpdate deviceInfoUpdate)
{
    DeviceWatcherInformation device = DeviceWatcherInformation_Search(deviceInfoUpdate.Id);
    if (device != null)
    {
        // The deviceInfo.Id was found in the RBLEDevices ObservableCollection.

        // Update the RBLE device information in the RBLEDevices ObservableCollection.
        device.Update(deviceInfoUpdate);
    }
}
```

図 4-22 DeviceWatcher 情報更新イベントハンドラ

デバイスの情報が Windows のシステムから削除された時に呼び出されます。通知された情報から RBLEDevices コレクションに同一の Id を持つ情報が入っている場合、RBLEDevices コレクションから削除します。

```
private void DeviceWatcher_Removed(DeviceWatcher sender, DeviceInformationUpdate deviceInfoUpdate)
{
    DeviceWatcherInformation device = DeviceWatcherInformation_Search(deviceInfoUpdate.Id);
    if (device != null)
    {
        // The deviceInfo.Id was found in the RBLEDevices ObservableCollection.

        // Remove the RBLE device information in the RBLEDevices ObservableCollection.
        RBLEDevices.Remove(device);
    }
}
```

図 4-23 DeviceWatcher 情報削除イベントハンドラ

DeviceWatcher の列挙が完了した時に呼び出されます。本アプリケーションでは処理を行いません。

```
private void DeviceWatcher_EnumerationCompleted(DeviceWatcher sender, object e)
{
    // do nothing.
}
```

図 4-24 DeviceWatcher 列挙完了イベントハンドラ

DeviceWatcher を停止する時は Stop メソッドを呼び出します。本アプリケーションでは Timer を使用して、一定時間 DeviceWatcher を動作させた後タイムアウトにより停止させています。

Stop メソッドを呼び出すと DeviceWatcher 停止イベントハンドラが呼び出されます。イベントハンドラを解除し、DeviceWatcher のオブジェクトを保存していたフィールドを初期化します。

```
private void DeviceWatcherTimer_Callback(object state)
{
    deviceWatcher.Stop();
}

private void DeviceWatcher_Stopped(DeviceWatcher sender, object e)
{
    // Unregister the event handlers.
    deviceWatcher.Added -= DeviceWatcher_Added;
    deviceWatcher.Updated -= DeviceWatcher_Updated;
    deviceWatcher.Removed -= DeviceWatcher_Removed;
    deviceWatcher.EnumerationCompleted -= DeviceWatcher_EnumerationCompleted;
    deviceWatcher.Stopped -= DeviceWatcher_Stopped;

    deviceWatcher = null;
```

図 4-25 DeviceWatcher 停止

4.5.3 Pairing

Pairing は Windows のシステムに評価ボードとの Pairing 情報を登録することができます。

Pairing は Pairing リクエスト通知を受け取るイベントハンドラ(PairingRequestedHandler)を登録し、DeviceInformation クラスの PairAsync メソッドを呼び出すことで実行されます。そしてイベントハンドラを解除して終了します。

```
private async void Pair(DeviceInformation deviceInfo)
{
    deviceInfo.Pairing.Custom.PairingRequested += PairingRequestedHandler;
    DevicePairingResult result =
            await deviceInfo.Pairing.Custom.PairAsync(DevicePairingKinds.ConfirmOnly,
                                                      DevicePairingProtectionLevel.Encryption);
    deviceInfo.Pairing.Custom.PairingRequested -= PairingRequestedHandler;
}
```

図 4-26 Pairing

Pairing を実行すると呼び出されるイベントハンドラです。イベントパラメータの DevicePairingRequestedEventArgs クラスの Accept メソッドを呼び出すことで Windows のシステムに Pairing 情報が登録されます。

```
private static void PairingRequestedHandler(DeviceInformationCustomPairing sender,
                                            DevicePairingRequestedEventArgs eventArgs)
{
    switch (eventArgs.PairingKind)
    {
        case DevicePairingKinds.ConfirmOnly:
            eventArgs.Accept();
            break;
        default:
            break;
    }
}
```

図 4-27 Pairing リクエストイベントハンドラ

4.5.4 Unpairing

Pairing を解除する時は DeviceInformation クラスの UnpairAsync メソッドを呼び出します。UnpairAsync メソッドを呼び出すと Windows のシステムから Pairing 情報が削除されます。

```
private async void Unpair()
{
    DeviceUnpairingResult result =
            await pairedDeviceInfomation.DeviceInformation.Pairing.UnpairAsync();
}
```

図 4-28 Unpairing

4.5.5 Connection

Connection の開始は BluetoothLEDevice クラスの FromIdAsync メソッドを呼び出すことで実行されます。FromIdAsync メソッドは Connection のトリガのみで、Connection(Connection request の発行)は Service 検索の GetGattServicesForUuidAsync メソッドを呼び出した時に実行されます。

```
private async void Connect()
{
    bluetoothLeConnectedDevice = await BluetoothLEDevice.FromIdAsync(pairedDeviceInfomation.Id);
```

図 4-29 Connection

Connection の処理が全て完了(本アプリケーションでは Client Characteristic Configuration Descriptor への Notification 許可)で ConnectionStatusChanged イベントハンドラを登録します。このイベントハンドラは Connection の状態が変化した時(Disconnection した時)に呼び出されます。

```
bluetoothLeConnectedDevice.ConnectionStatusChanged += ConnectionStatusChanged;
```

図 4-30 ConnectionStatusChanged イベントハンドラ

4.5.6 Service 検索

Service の検索は BluetoothLEDevice クラスの GetGattServicesForUuidAsync メソッドに検索対象の UUID をパラメータとして指定して呼び出すことで実行されます。GetGattServicesForUuidAsync メソッドが正常に実行されると、GattDeviceServicesResult クラスの Services プロパティを読み出し、Service のオブジェクト(GattDeviceService クラスのオブジェクト)をフィールドに保存します。

```
var serviceResult =
    await bluetoothLeConnectedDevice.GetGattServicesForUuidAsync(UUID_LEDSW_SERVICE);
if (serviceResult.Status == GattCommunicationStatus.Success)
{
    gattPrimaryService = serviceResult.Services[0];
```

図 4-31 Service 検索

4.5.7　Characteristic 検索

Characteristic の検索は GattDeviceServicesResult クラスの GetCharacteristicsForUuidAsync メソッドに検索対象の UUID をパラメータとして指定して呼び出すことで実行されます。GetCharacteristicsForUuidAsync メソッドが正常に実行されると、GattCharacteristicsResult クラスの Characteristics プロパティを読み出し、Characteristic のオブジェクト(GattCharacteristic クラスのオブジェクト)をフィールドに保存します。

Characteristic が Notification の場合、Notification を受信した時に呼び出される ReceiveNotification イベントハンドラを登録します。

```
var notificationCharacteristicsResult =
    await serviceResult.Services[0].GetCharacteristicsForUuidAsync(UUID_LEDSW_SW_STATE_NOTIFICATION);
if (notificationCharacteristicsResult.Status == GattCommunicationStatus.Success)
{
    characteristicNotify = notificationCharacteristicsResult.Characteristics[0];
    characteristicNotify.ValueChanged += ReceiveNotification;
```

図 4-32 Characteristic 検索(Notification)

Write の Characteristic も同様に GetCharacteristicsForUuidAsync メソッドに UUID をパラメータとして指定して呼び出します。

```
var writeCharacteristicsResult =
    await serviceResult.Services[0].GetCharacteristicsForUuidAsync(UUID_LEDSW_LED_BLINK_RATE_WRITE);
if (writeCharacteristicsResult.Status == GattCommunicationStatus.Success)
{
    characteristicWrite = writeCharacteristicsResult.Characteristics[0];
```

図 4-33 Characteristic 検索(Write)

4.5.8　Client Characteristic Configuration Descriptor 設定

Notification の許可は、GattCharacteristic クラスの WriteClientCharacteristicConfigurationDescriptorAsync メソッドに Notify 許可パラメータを指定して呼び出すことで実行されます。

```
GattCommunicationStatus status =
    await characteristicNotify.WriteClientCharacteristicConfigurationDescriptorAsync(
                                GattClientCharacteristicConfigurationDescriptorValue.Notify);
```

図 4-34 Client Characteristic Configuration Descriptor 設定

4.5.9 Disconnection

Disconnection は、GattCharacteristic クラスの Dispose メソッド、GattDeviceService クラスの Dispose メソッド、BluetoothLEDevice クラスの Dispose メソッドを呼び出します。

```
private void Disconnect()
{
    if (characteristicNotify != null)
    {
        characteristicNotify.Service.Dispose();
    }
    if (gattPrimaryService != null)
    {
        gattPrimaryService.Dispose();
    }
    if (bluetoothLeConnectedDevice != null)
    {
        bluetoothLeConnectedDevice.Dispose();
    }
}

private void ConnectionStatusChanged(BluetoothLEDevice sender, object e)
{
    if (sender.ConnectionStatus == BluetoothConnectionStatus.Disconnected)
    {
        // Disconnect and clear value.
        if (bluetoothLeConnectedDevice != null)
        {
            bluetoothLeConnectedDevice.ConnectionStatusChanged -= ConnectionStatusChanged;
            bluetoothLeConnectedDevice = null;
        }
        if (gattPrimaryService != null)
        {
            gattPrimaryService = null;
            characteristicNotify.ValueChanged -= ReceiveNotification;
            characteristicNotify = null;
            characteristicWrite = null;
        }
    }
}
```

図 4-35 Disconnection

4.5.10 Notification 受信

Connection した評価ボードから Notification を受信すると登録したイベントハンドラが呼び出されます。受信データはパラメータの GattValueChangedEventArgs クラスで渡されます。

```
private void ReceiveNotification(GattCharacteristic sender, GattValueChangedEventArgs eventArgs)
{

}
```

図 4-36 Notification 受信

4.5.11 Write 送信

Connection した評価ボードに Write でデータを送信します。GattCharacteristic クラスの WriteValueAsync メソッドにパラメータとして送信データを指定して呼び出すことで、送信することができます。

```
private async void SendWrite(string str)
{
    await characteristicWrite.WriteValueAsync(ibuffer, GattWriteOption.WriteWithResponse);
}
```

図 4-37 Write 送信

5. 注意事項

5.1 Client Characteristic Configuration Descriptor 書き込みエラー

ご使用の Windows 10 の状態によっては、Connect ボタンをクリックした時に必ず図 5-1に示すダイアログが表示される場合があります。これは、Client Characteristic Configuration Descriptor へ設定する WriteClientCharacteristicConfigurationDescriptorAsync メソッドの呼び出しで例外が発生することが原因です。Windows 10 をバージョンアップするか、または別の Windows 10 PC をご使用ください。

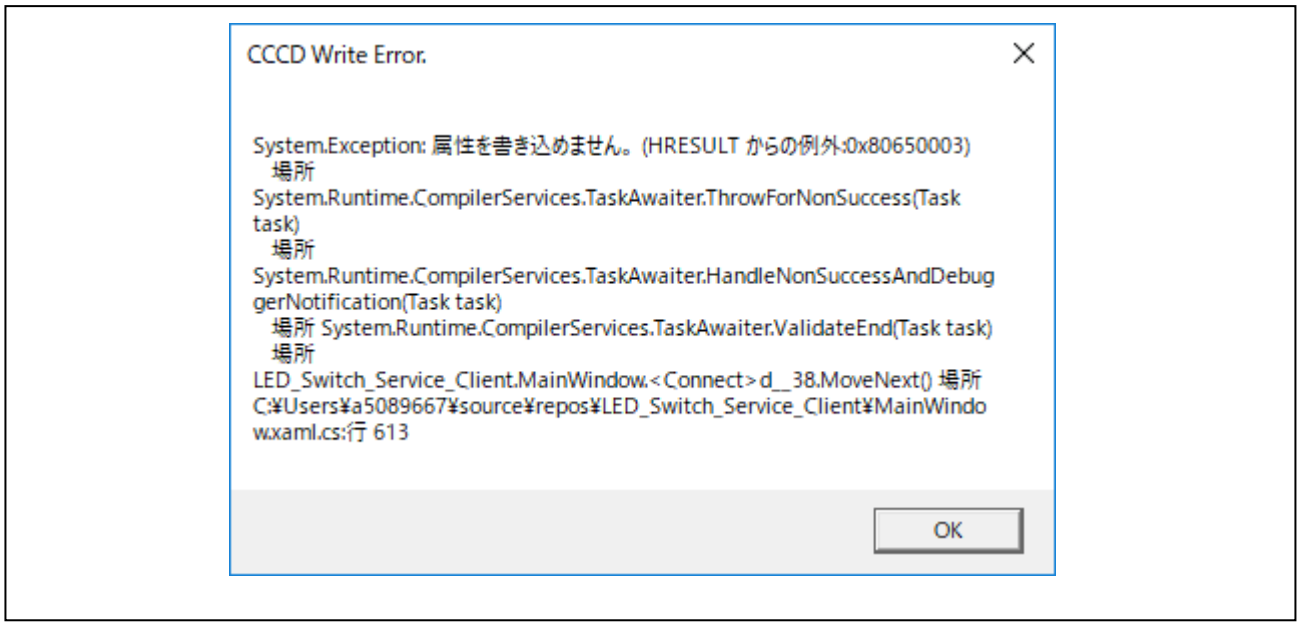

図 5-1 CCCD Write Error

6. 付録

6.1 サービスやキャラクタリスティックのカスタマイズ

ペリフェラルデバイスのサービスやキャラクタリスティックは UUID を使用して検索します。「4.5.6 Service検索」や「4.5.7 Characteristic検索」で示すソースコードとほぼ同一です。差分は UUID、フィールド名(変数名)、イベントハンドラ名となりますので、容易にカスタマイズすることができます。

図 6-1に Indication の Characteristic を取得するサンプルコードを示します。(エラー処理は除いています。)

```
private Guid UUID_SERVICE    = new Guid("5BC1B9F7-A1F1-40AF-9043-C43692C18D7A");
private Guid UUID_INDICATION = new Guid("5BC18D80-A1F1-40AF-9043-C43692C18D7A");


private BluetoothLEDevice bluetoothLeConnectedDevice;
private GattDeviceService gattPrimaryService;
private GattCharacteristic characteristicIndication;


private async void Connect()
{
    // Start Connection
    bluetoothLeConnectedDevice = await BluetoothLEDevice.FromIdAsync(pairedDeviceInfomation.Id);

    // Service
    var serviceResult =
        await bluetoothLeConnectedDevice.GetGattServicesForUuidAsync(UUID_SERVICE);
    gattPrimaryService = serviceResult.Services[0];

    // Characteristic Indication
    var indicationCharacteristicsResult =
        await serviceResult.Services[0].GetCharacteristicsForUuidAsync(UUID_INDICATION);
    characteristicIndication = indicationCharacteristicsResult.Characteristics[0];
    characteristicIndication.ValueChanged += ReceiveIndication;
}

private void ReceiveIndication(GattCharacteristic sender, GattValueChangedEventArgs eventArgs)
{
    // process
}
```

図 6-1 サービスやキャラクタリスティックのカスタマイズ

6.2　新しいプロジェクトの作成

WPF アプリケーションで Bluetooth LE 機能を使用するプロジェクトの作成方法を示します。

(1)　Microsoft Visual Studio Express 2017 for Windows Desktop を起動します。

(2)　メニューバーから[ファイル] - [新しいプロジェクト]を選択し、「新しいプロジェクト」ダイアログを表示します。

(3)　「新しいプロジェクト」のダイアログで、「WPF アプリ(.NET Framework)」を選択します。
[名前]、[場所]に作成するプロジェクト名とフォルダ名を入力します。
[フレームワーク]で使用する.NET Framework のバージョンを選択します。(ここでは「.NET Framework 4.6.1」を選択します。)
[OK]ボタンをクリックしダイアログを閉じます。

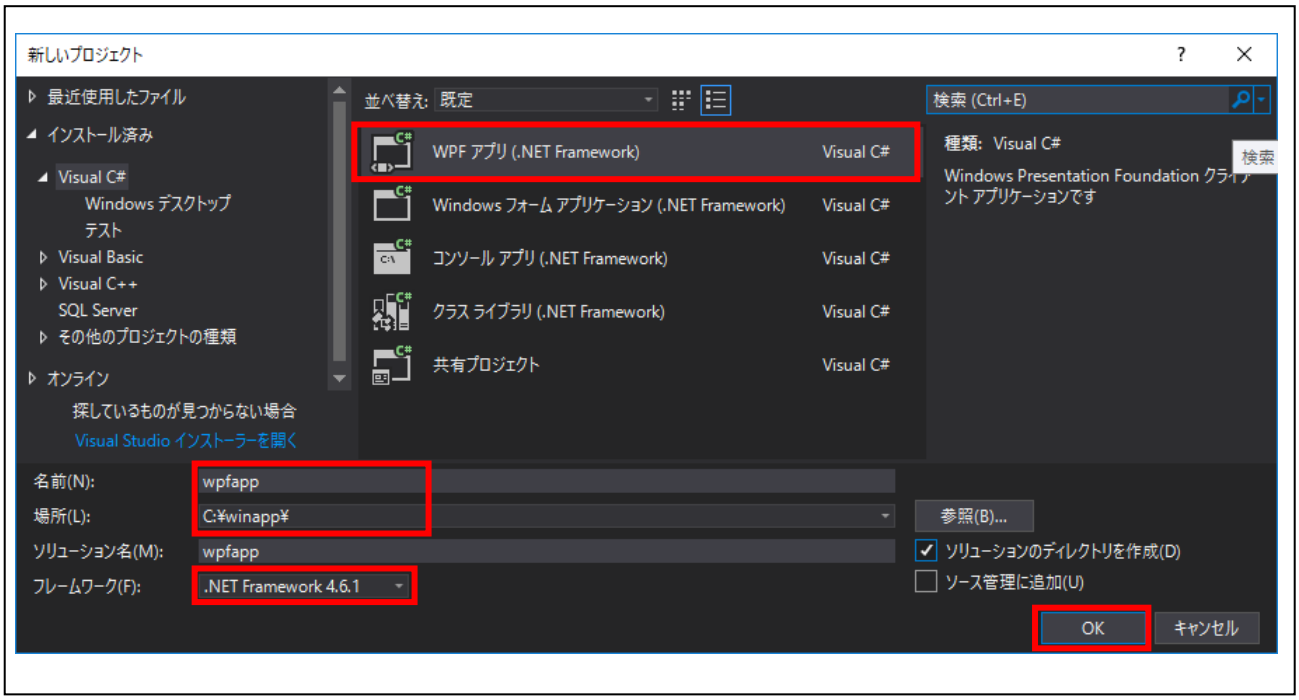

図 6-2 新しいプロジェクトの作成 (1)

(4)　新しいプロジェクトのウインドウが起動します。

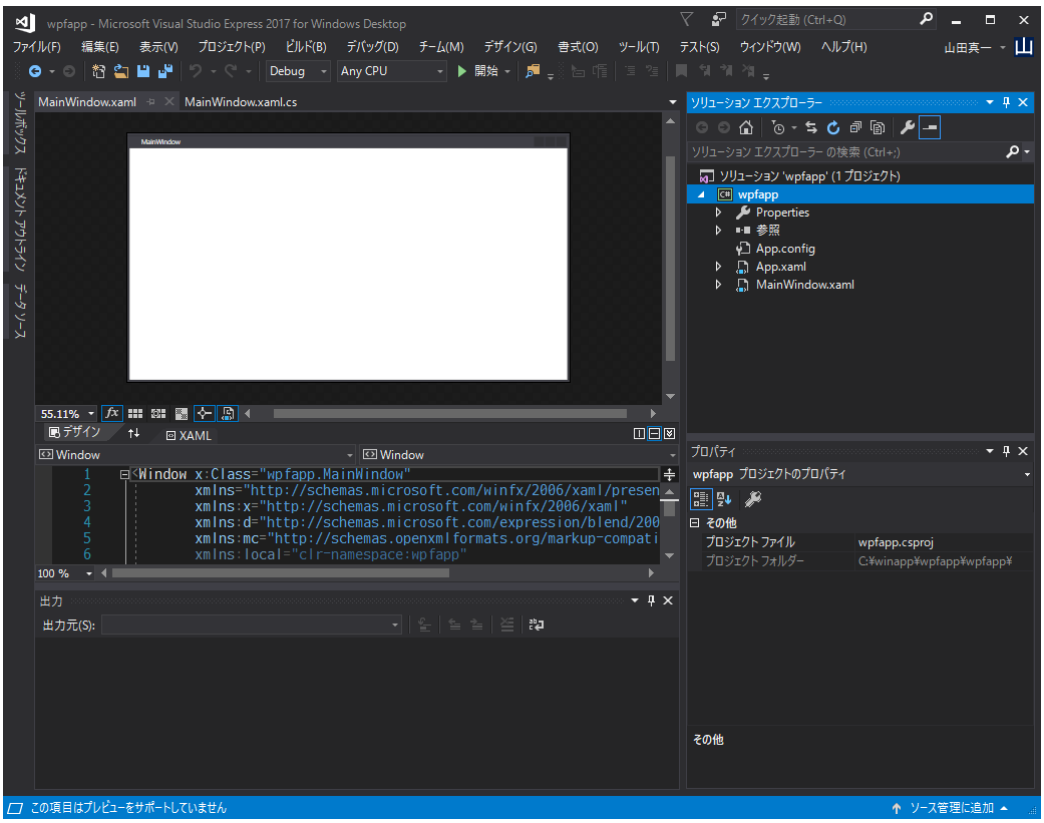

図 6-3 新しいプロジェクトの作成 (2)

(5) 「ソリューション エクスプローラー」のプロジェクト名の上で右クリックしメニューを表示します。
[プロジェクトのアンロード]を選択します。

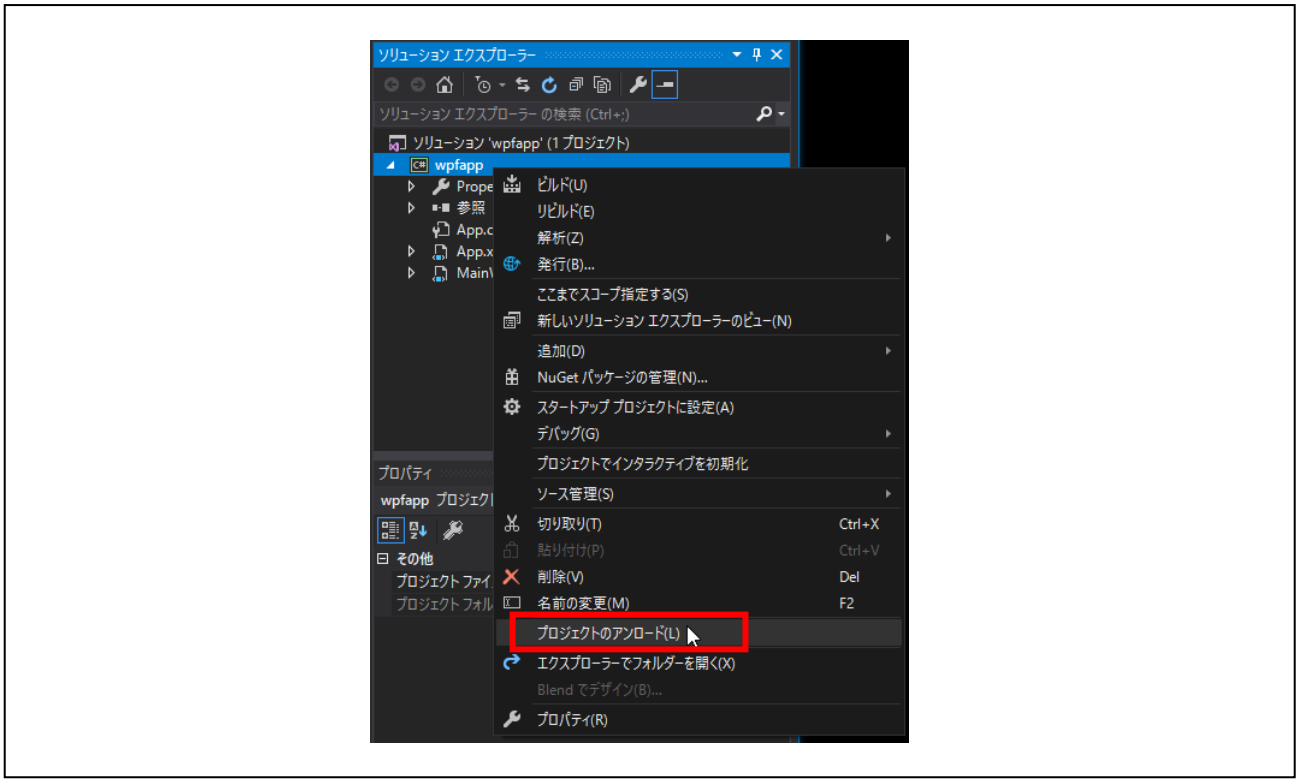

図 6-4 新しいプロジェクトの作成 (3)

(6) 「ソリューション エクスプローラー」のプロジェクト名の上で右クリックしメニューを表示します。
[編集 (wpfapp.csproj)]を選択します。

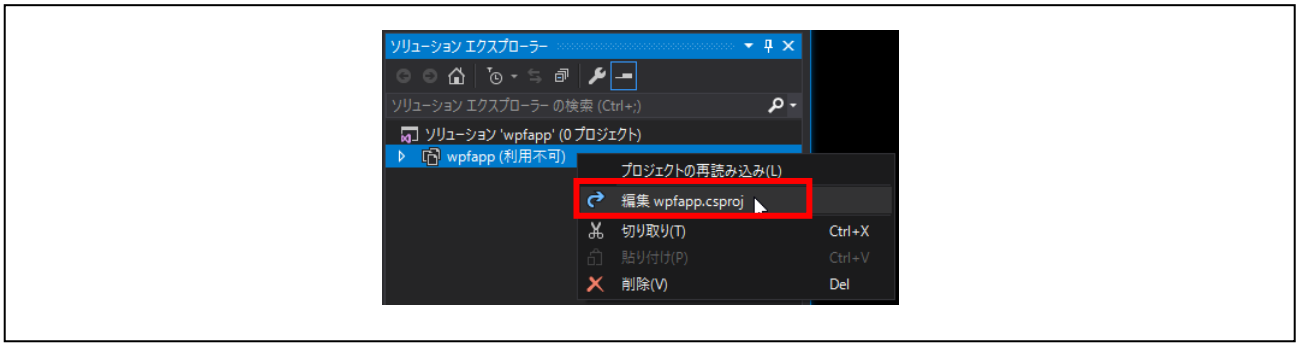

図 6-5 新しいプロジェクトの作成 (4)

(7) wpfapp.csproj に「<TargetPlatformVersion>10.0</TargetPlatformVersion>」を追加します。

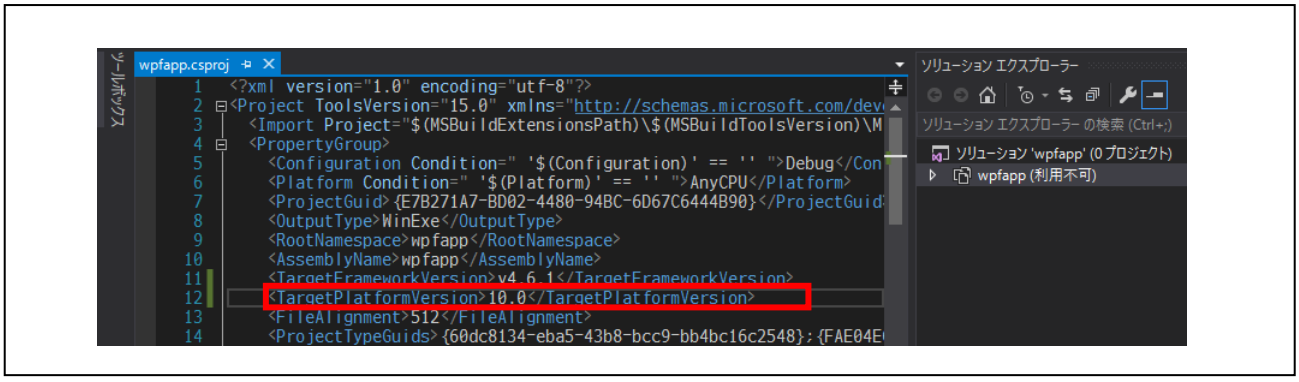

図 6-6 新しいプロジェクトの作成 (5)

(8) 「ソリューション エクスプローラー」のプロジェクト名の上で右クリックしメニューを表示します。[プロジェクトの再読み込み]を選択します。

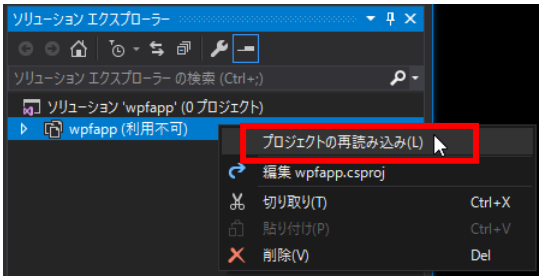

図 6-7 新しいプロジェクトの作成 (6)

(9) 「ソリューション エクスプローラー」の「参照」の上で右クリックしメニューを表示します。[参照の追加]を選択し「参照マネージャー」ダイアログを表示します。

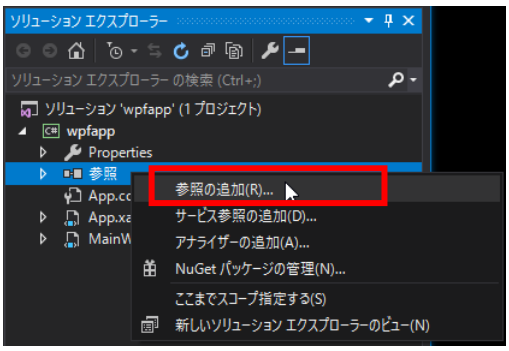

図 6-8 新しいプロジェクトの作成 (7)

(10) 「参照マネージャー」のダイアログで「参照」メニューを選択します。
[参照]ボタンを押し、以下 2 つのファイルを追加します。

- C:¥Program Files (x86)¥Windows Kits¥10¥UnionMetadata¥10.0.19041.0¥Windows.winmd
- C:¥Program Files (x86)¥Reference Assemblies¥Microsoft¥Framework¥.NETCore¥v4.5¥System.Runtime.WindowsRuntime.dll

[OK]ボタンを押してダイアログを閉じると、Bluetooth LE 機能のクラスやメソッドが使用できるようになります。

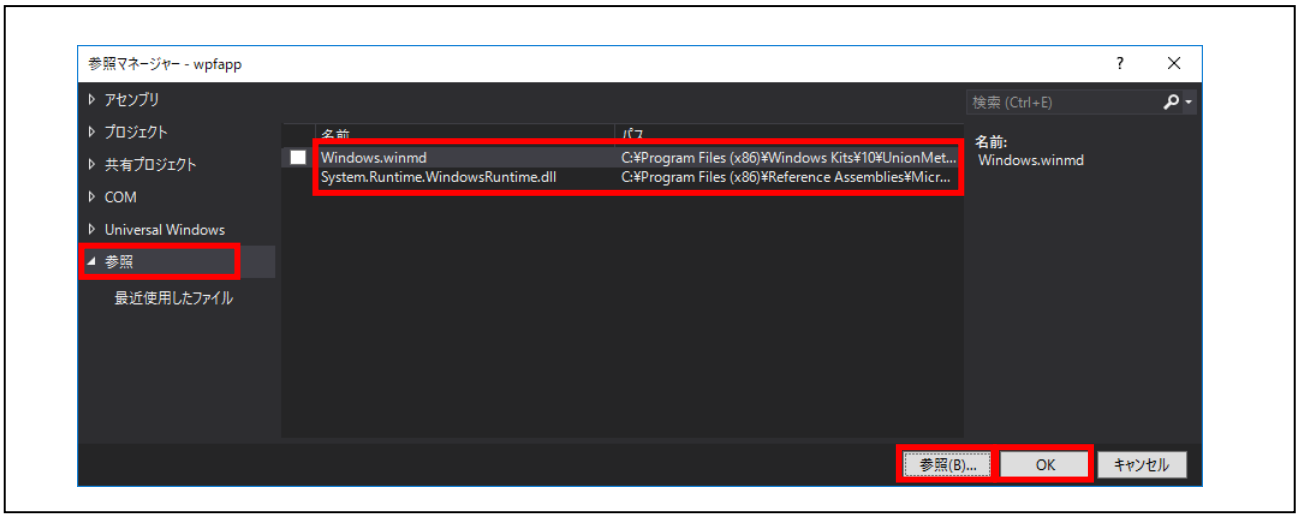

図 6-9 新しいプロジェクトの作成 (8)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.0	2021/08/31	-	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。