Bluetooth® Low Energy プロトコルスタック

仮想 UART アプリケーション

要旨

このマニュアルは、Bluetooth LE の無線通信技術を使用した仮想 UART アプリケーションのソフトウェア構成、機能、動作確認手順、実装の詳細について記載しています。

仮想 UART アプリケーションは、RL78/G1D 上で Embedded 構成の Renesas Bluetooth® Low Energy プロトコルスタックを制御して動作します。

アプリケーションは、以下の機能を提供します。

- Bluetooth LE 接続の制御や設定を行うための簡易 AT コマンド実行機能
- Bluetooth LE 通信を用いて文字やバイナリデータの送受信を行う仮想 UART 機能
- Bluetooth LE 通信でレスポンス有無の通信を選択する機能

動作確認デバイス

RL78/G1D

関連資料

資料名	資料番号	
	和文	英文
Bluetooth® Low Energy プロトコルスタック		
ユーザーズマニュアル	R01UW0095J	R01UW0095E
API リファレンスマニュアル　基本編	R01UW0088J	R01UW0088E
クイックスタートガイド	R01AN2767J	R01AN2767E
GUI ツール	R01AN2469J	R01AN2469E
Bluetooth® Specification		
Vol 6. Low Energy Controller volume	-	Core_v4.2

目次

1. 概要

このマニュアルは、Bluetooth LE の無線通信技術を使用した仮想 UART アプリケーション（以下、アプリケーション）のソフトウェア構成、機能、動作確認手順、実装の詳細について記載しています。

アプリケーションは、RL78/G1D 上で Embedded 構成の Renesas Bluetooth® Low Energy プロトコルスタック（以下、BLE プロトコルスタック）を制御して動作します。

アプリケーションは、以下の機能を提供します。

- Bluetooth LE 接続の制御や設定を行うための簡易 AT コマンド実行機能
- Bluetooth LE 通信を用いてデータの送受信を行う仮想 UART 機能
    - 文字データの送受信動作
    - バイナリデータの送受信動作
- Bluetooth LE 通信でレスポンス有無の通信を選択する機能
    - レスポンス有り: Write Request, Indication
    - レスポンス無し: Write Command, Notification

1.1 動作の概要

アプリケーション使用時の構成を図 1-1 に示します。アプリケーションを書き込んだ 2 台の RL78/G1D 評価ボード（以降、デバイス）を、それぞれ USB ケーブルで PC に接続します。ユーザは、PC 上に起動したターミナルソフト経由でアプリケーションを操作します。

Bluetooth LE 接続の開始や切断などの制御、アドレスの設定などを行うために簡易 AT コマンドを使用します。Bluetooth LE 接続を確立後、ローカルデバイス側のターミナルソフトに入力したデータは、リモートデバイスに送信され、リモートデバイス側のターミナルソフトに表示されます。これとは逆に、リモートデバイス側のターミナルソフトに入力したデータは、ローカルデバイスに送信され、ローカルデバイス側のターミナルソフトに表示されます。

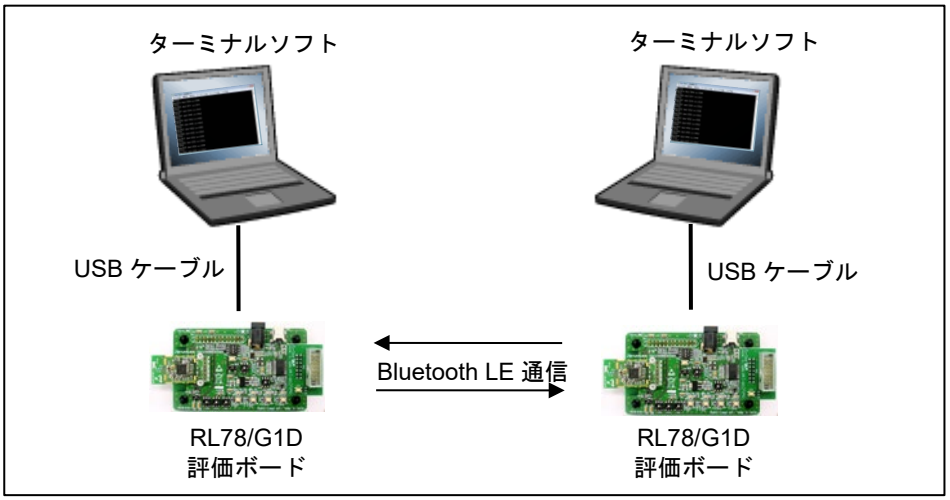


図 1-1 アプリケーション使用時の構成図

2. 構成

2.1 ソフトウェア構成

アプリケーションのソフトウェア構成を図 2-1 に示します。

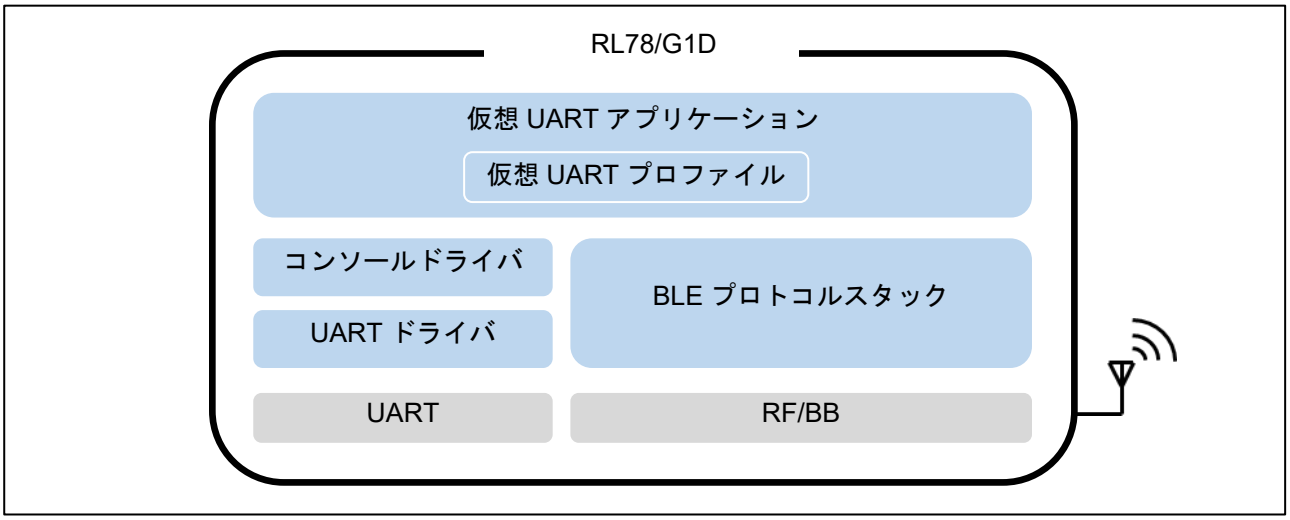


図 2-1 ソフトウェアの構成

ソフトウェアの各コンポーネントの説明を表 2-1 に示します。

表 2-1 各ソフトウェアコンポーネントの説明表

コンポーネント	説明
仮想 UART アプリケーション	簡易 AT コマンドの実行や、文字データ/バイナリデータの送受信を行います。データの送受信を実現するため、内部に仮想 UART プロファイルを定義しています。
コンソールドライバ	UART ドライバを使用して、ターミナルソフトからのデータ受信や、ターミナルソフトへのデータ送信を行います。
UART ドライバ	RL78/G1D に搭載されている UART の制御を行うドライバです。
BLE プロトコルスタック	Renesas Bluetooth® Low Energy プロトコルスタックです。詳細は、「Bluetooth® Low Energy プロトコルスタックユーザーズマニュアル」(R1UW0095J)を参照してください。

2.2　ファイル構成

アプリケーションのファイル構成を以下に示します。アプリケーションは、BLE プロトコルスタックを含む BLE ソフトウェアを基に作成されています。ここでは、BLE ソフトウェアに対して差分のあるファイルのみを表示します。なお、(M)は、BLE ソフトウェアに含まれるソースコードを変更したファイルを示し、(A)は追加したファイルを示します。

アプリケーションでの変更点は、BLE ソフトウェアとの差分を確認してください。

```
r01an3130xx0114-rl78g1d-ble-vuart/
    ├─── Project_Source/
    │    ├─── rBLE/
    │    │    └─── src/
    │    │         ├─── sample_app/
    │    │         │    ├─── r_vuart_app.c              (A)  ┐ 仮想 UART アプリケーション
    │    │         │    ├─── r_vuart_app.h              (A)  │
    │    │         │    ├─── r_vuart_app_param.c       (A)  ┘
    │    │         │    ├─── r_vuart_console.c         (A)  ┐ コンソールドライバ
    │    │         │    └─── r_vuart_console.h         (A)  ┘
    │    │         └─── sample_profile/
    │    │              └─── vuart/
    │    │                   ├─── vuart.h               (A)  ┐ 仮想 UART プロファイル
    │    │                   ├─── vuartc.c              (A)  │
    │    │                   ├─── vuartc.h              (A)  │
    │    │                   ├─── vuarts.c              (A)  │
    │    │                   └─── vuarts.h              (A)  ┘
    │    └─── renesas/
    │         ├─── src/
    │         │    ├─── arch/
    │         │    │    └─── rl78/
    │         │    │         ├─── arch_main.c          (M)  省電力機能を有効にするために変更
    │         │    │         ├─── db_handle.h          (M)  ┐ 仮想 UART プロファイルに関する
    │         │    │         ├─── ke_conf.c            (M)  │ 各種定義の追加に伴う変更
    │         │    │         ├─── prf_config.c         (M)  │
    │         │    │         └─── prf_config.h         (M)  ┘
    │         │    └─── driver/
    │         │         ├─── dataflash/
    │         │         │    ├─── eel_descriptor_t02.c (M)  ┐ アドレスを Data Flash に
    │         │         │    └─── eel_descriptor_t02.h      ┘ 保存するための変更
    │         │         └─── uart/
    │         │              └─── uart.c               (M)  省電力機能を有効にするために変更
    │         └─── tools/
    │              └─── project/
    │                   ├─── CS_CCRL/                  (M)  ┐
    │                   ├─── CS_CACX/                  (M)  │各開発環境のプロジェクトファイル
    │                   └─── e2studio/                 (M)  ┘
    ├─── ROM_File/
    │    ├─── ccrl/                                         ┐ ccrl で生成した
    │    │    ├─── RL78_G1D_CCE(VUART_CHAR).hex            │ アプリケーションのファームウェア
    │    │    └─── RL78_G1D_CCE(VUART_BIN).hex       (A)  ┘
    │    └─── ca78k0r/                                      ┐ ca78k0r で生成した
    │         ├─── RL78_G1D_CE(VUART_CHAR).hex             │ アプリケーションのファームウェア
    │         └─── RL78_G1D_CE(VUART_BIN).hex        (A)  ┘
```

```
└──── Macro/                                          ┐
    ├──── teraterm_debug_mode_on.ttl                  │
    ├──── teraterm_debug_mode_off.ttl                 │Tera Term マクロ
    ├──── tt_send_bin_1.ttl                           │
    └──── tt_send_bin_2.ttl                    (A)    ┘
```

3. 機能

3.1 動作モード

アプリケーションには表 3-1 に示す 2 つの動作モード、簡易 AT コマンドモードと仮想 UART モードの動作モードがあります。

表 3-1 動作モード一覧

モード名	説明
簡易 AT コマンドモード	簡易 AT コマンドを使用して、Bluetooth LE 接続の制御や設定を行うモード。このモード中に入力された文字は、接続状態・非接続状態にかかわらず、送信されません。 詳細は「4.簡易 AT コマンドモード」を参照してください。
仮想 UART モード	Bluetooth LE 通信による文字データやバイナリデータの送受信を行うモード。このモード中に、非接続状態でデータを入力した場合、エラーが表示され、送信は行われません。 詳細は「5.仮想 UART モード」を参照してください。

3.2 データ送受信におけるレスポンス有無の選択

仮想 UART モードでデータの送受信を行う際に、レスポンス有りまたはレスポンス無しの通信を RL78/G1D 評価ボード上のディップスイッチで選択することができます。

表 3-2 レスポンス有無の設定

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
	ON(右側)	レスポンス無し通信 クライアント → サーバ: Write Command サーバ → クライアント: Notification

3.3 送信するデータの選択

アプリケーションでは、マクロ設定で「3.3.1 文字データ送受信」または「3.3.2」を選択することができます。マクロ設定については「8.8.1 文字データ/バイナリデータ送受信設定」を参照してください。

3.3.1 文字データ送受信

文字データ送受信では、アスキーコードの図形文字と改行文字を送信することができます。

表 3-1 に示す動作モードの切り替えは、ターミナルソフトにエスケープキー（アスキーコード：0x1B）を入力することで行うことができます。

文字データ送受信の使用例は、「7 ビルドと動作確認」-「7.4.1 文字データ送受信」を参照してください。

3.3.2 バイナリデータ送受信

バイナリデータ送受信では、文字データを含む全てのデータを送信することができます。

表 3-1 に示す動作モードの切り替えは、RL78/G1D 評価ボード上のディップスイッチで行うことができます。

バイナリデータ送受信の使用例は、「7 ビルドと動作確認」-「7.4.2 バイナリデータ送受信」を参照してください。

表 3-3 バイナリデータ送受信での動作モード切り替え

番号	スイッチ	説明
SW6-1	OFF(左側)	簡易 AT コマンドモード
	ON(右側)	仮想 UART モード ※バイナリデータ送受信の仮想 UART モードでは、ターミナルソフトに入力したデータのローカルエコーは禁止となります。

4. 簡易 AT コマンドモード

簡易 AT コマンドを実行するモードです。ユーザは、簡易 AT コマンドを使用して、Bluetooth LE 接続の制御や設定を行うことができます。

アプリケーションがサポートする簡易 AT コマンドの一覧を表 4-1 に示します。

表 4-1 簡易 AT コマンド一覧

簡易 AT コマンド	概要
AT-C	AT-AP=<addr>で設定したアドレスへの接続を開始します。
AT-C=<addr>	<addr>で指定したアドレスへの接続を開始します。
AT-R	接続中の場合は接続を切断し、アドバタイジングを開始します。非接続中の場合は、アドバタイジングを開始します。
AT-AS=<addr>	ローカルデバイスのパブリックデバイスアドレスを<addr>に設定します。
AT-AS?	ローカルデバイスのパブリックデバイスアドレスを表示します。
AT-AP=<addr>	AT-C で接続を開始するアドレスを設定します。
AT-AP?	AT-C で接続を開始するアドレスを表示します。
AT-DS	周辺でアドバタイジングを行っている仮想 UART プロファイルをサポートしているデバイスのアドレスを表示します。
AT-S?	ローカルデバイスの接続状態（接続状態もしくは非接続状態）を表示します。
AT-CI=<con_intv>	Connection Interval の設定を行います。
AT-CI	Connection Interval の設定値を表示します。
ATE0	ローカルエコーを無効化します。
ATE1	ローカルエコーを有効化します。

4.1 簡易 AT コマンドの詳細

4.1.1 AT-C

動作	AT-AP=<addr>で設定したアドレスへの接続を開始します。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
	CONNECT	接続完了
コマンド例	AT-C OK CONNECT	

4.1.2 AT-C=<addr>

動作	<addr>で指定したアドレスへの接続を開始します。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
	CONNECT	接続完了
コマンド例	AT-C=CBA987654321 （CB:A9:87:65:43:21 を設定） OK CONNECT	

4.1.3 AT-R

動作	接続中の場合は、接続を切断し、アドバタイジングを開始します。非接続中の場合は、アドバタイジングを開始します。	
レスポンス	OK	正常終了
	DISCONNECT	切断完了
コマンド例	（接続中の場合） AT-R OK DISCONNECT （非接続中の場合） AT-R OK	

4.1.4 AT-AS=<addr>

動作	ローカルデバイスのパブリックデバイスアドレスを<addr>に設定します。このコマンドにより設定したアドレスは、電源を OFF にした後も保持されます。 本コマンドで設定したパブリックアドレスは、再起動後に有効になります。本コマンドを実行した後、評価ボード上のリセットボタンを押下するなどして、再起動してください。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
コマンド例	AT-AS=CCCCBBBBAAAA （CC:CC:BB:BB:AA:AA を設定） OK	

4.1.5 AT-AS?

動作	ローカルデバイスのパブリックデバイスアドレスを表示します。
レスポンス	OK
コマンド例	AT-AS? -AS: CCCCBBBBAAAA （アドレスは CC:CC:BB:BB:AA:AA） OK

4.1.6 AT-AP=<addr>

動作	AT-C により、接続を開始するパブリックデバイスアドレスを<addr>に設定します。 このコマンドにより設定したアドレスは、電源を OFF にした後も保持されます。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
コマンド例	AT-AP=CCCCBBBBAAAA （CC:CC:BB:BB:AA:AA を設定） OK	

4.1.7 AT-AP?

動作	AT-C により、接続を開始するパブリックデバイスアドレスを表示します。	
レスポンス	OK	正常終了
コマンド例	AT-AP? -AP: CCCCBBBBAAAA　（アドレスは CC:CC:BB:BB:AA:AA） OK	

4.1.8 AT-DS

動作	周辺でアドバタイジングを行っている仮想 UART プロファイルをサポートしているデバイスのアドレスを表示します。周辺のデバイスの検索を行う期間は、7.68 秒です。 アドバタイジングデータのサービスリストに仮想 UART サービスの 128bit UUID が含まれている場合、そのデバイスが仮想 UART プロファイルをサポートしていると判断します。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
コマンド例	AT-DS -DS: CBA987654321　（アドレスは CB:A9:87:65:43:31） -DS: CCCCBBBBAAAA　（アドレスは CC:CC:BB:BB:AA:AA） OK	

4.1.9 AT-S?

動作	ローカルデバイスの接続状態（接続状態もしくは非接続状態）を表示します。 接続中の場合は CONNECT を、非接続中の場合は DISCONNECT を表示します。	
レスポンス	OK	正常終了
コマンド例	（接続中） AT-S? CONNECT OK （非接続中） AT-S? DISCONNECT OK	

4.1.10 AT-CI=<con_intv>

動作	Connection Interval の設定を行います。 本コマンドを非接続中に実行した場合、アプリケーション内部に con_intv の値を保持します。以降の接続時に保持している設定値を使用します。本コマンドを使用して、接続中に Connection Interval の変更ができません。 Connection Interval の値は、以下の式により計算します。 Connection Interval = con_intv * 1.25[ms] 例）Connection Interval を 20[ms]に設定したい場合、AT-CI=16 を実行 アプリケーションの Connection Interval の初期値は 30[ms]です。 Connection Interval の設定シーケンスは、図 8-3 の「Connection Parameter Update」部分を参照してください。接続確立後、Peripheral 側のデバイスは、本コマンドで設定した Connection Interval への変更要求を Central 側デバイスに対して行います。Central 側のデバイスは、設定可能な Connection Interval に制限を課している場合があり、要求された Connection Interval への変更を拒否する場合があります。要求した Connection Interval への変更が受け入れられたかどうかは、「AT-CI?」を使用して確認してください。	
レスポンス	OK	正常終了
	ERROR	接続状態のため実行不可
		con_intv に設定可能範囲外の値を指定（設定可能範囲：6 ~ 3200）
コマンド例	（非接続中） AT-CI=20 OK AT-CI=3201 ERROR （接続中） AT-CI=20 ERROR	

4.1.11 AT-CI?

動作	Connection Interval の設定値を表示します。 接続中に実行した場合、その接続の Connection Interval を表示します。非接続中に実行した場合、アプリケーションが内部で保持している Connection Interval を表示します。 Connection Interval の値は、本コマンドにより取得した値に 1.25[ms]を乗じた値になります。 例）レスポンスが「-CI: 20」の場合、Connection Interval は、20 * 1.25[ms] = 25[ms]となります。	
レスポンス	OK	正常終了
コマンド例	AT-CI? -CI: 20 OK	

4.1.12 ATE0

動作	ローカルエコーを無効化します。	
レスポンス	OK	正常終了
コマンド例	ATE0 OK	

4.1.13 ATE1

動作	ローカルエコーを有効化します。	
レスポンス	OK	正常終了
コマンド例	ATE1 OK	

5. 仮想 UART モード

Bluetooth LE 接続を確立後、無線通信を介してデータの送受信を行うことができます。

文字データ送受信動作の場合、送信可能な文字はアスキーコードの図形文字と改行文字です。バイナリデータ送受信動作の場合、文字データを含む全てのデータを送信することができます。

5.1 仮想 UART プロファイル

データの送受信は、GATT ベースの仮想 UART プロファイルによって行います。仮想 UART プロファイルの詳細は、「8.1 仮想 UART プロファイル」を参照してください。

AT-C を実行したデバイスが GATT クライアント（以下、クライアント）、その対向デバイスが GATT サーバ（以下、サーバ）として動作し、以下のようにデータの送受信を行います。

5.1.1 レスポンス有りの通信

- クライアントからサーバへのデータの送信は、仮想 UART プロファイルが提供する Characteristic に対して Write Request を送信することにより行います。サーバはデータを受信すると、Response を返却します。
- サーバからクライアントへのデータの送信は、仮想 UART プロファイルが提供する Characteristic の更新を Indication することにより行います。クライアントはデータを受信すると、Confirmation を返却します。

このように、Write Request や Indication を送信後、Response や Confirmation の受信待ちをするため、データの到達が保証されます。

図 5-1 に、ローカルデバイスがクライアントになった場合のデータ送受信時のシーケンスを示します。

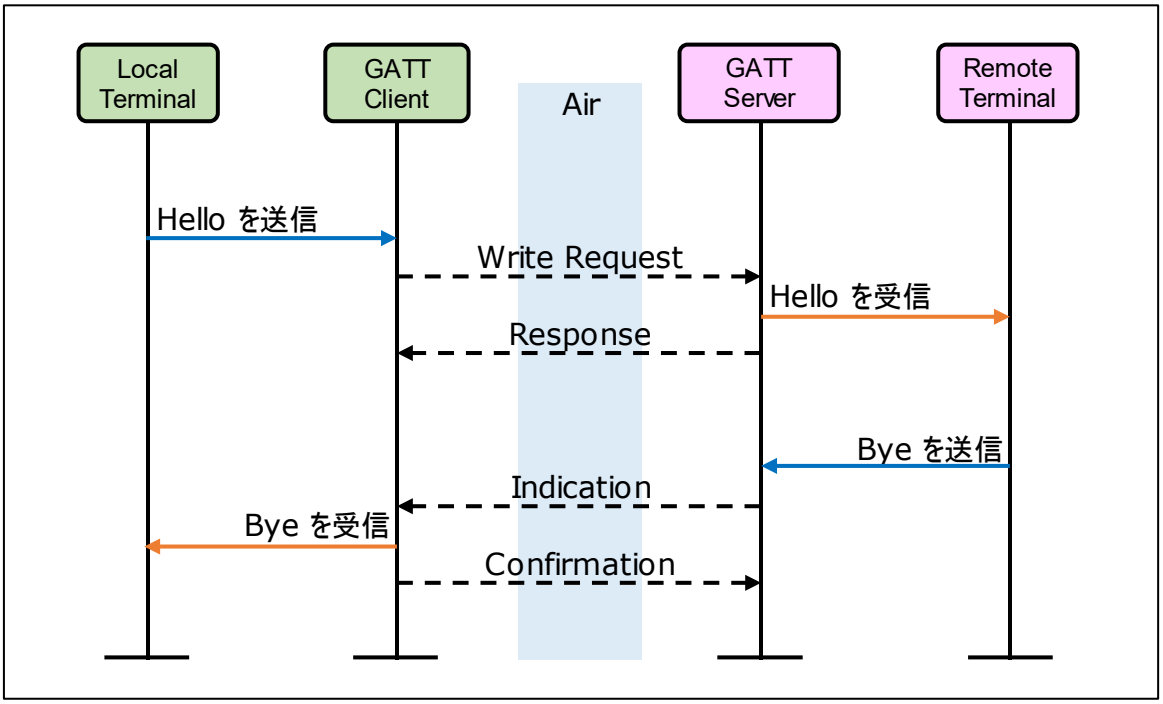


図 5-1 データ送受信時のシーケンス（レスポンス有り）

5.1.2 レスポンス無しの通信

- クライアントからサーバへのデータの送信は、仮想 UART プロファイルが提供する Characteristic に対して Write Command を送信することにより行います。
- サーバからクライアントへのデータの送信は、仮想 UART プロファイルが提供する Characteristic の更新を Notification することにより行います。

図 5-2 に、ローカルデバイスがクライアントになった場合のデータ送受信時のシーケンスを示します。

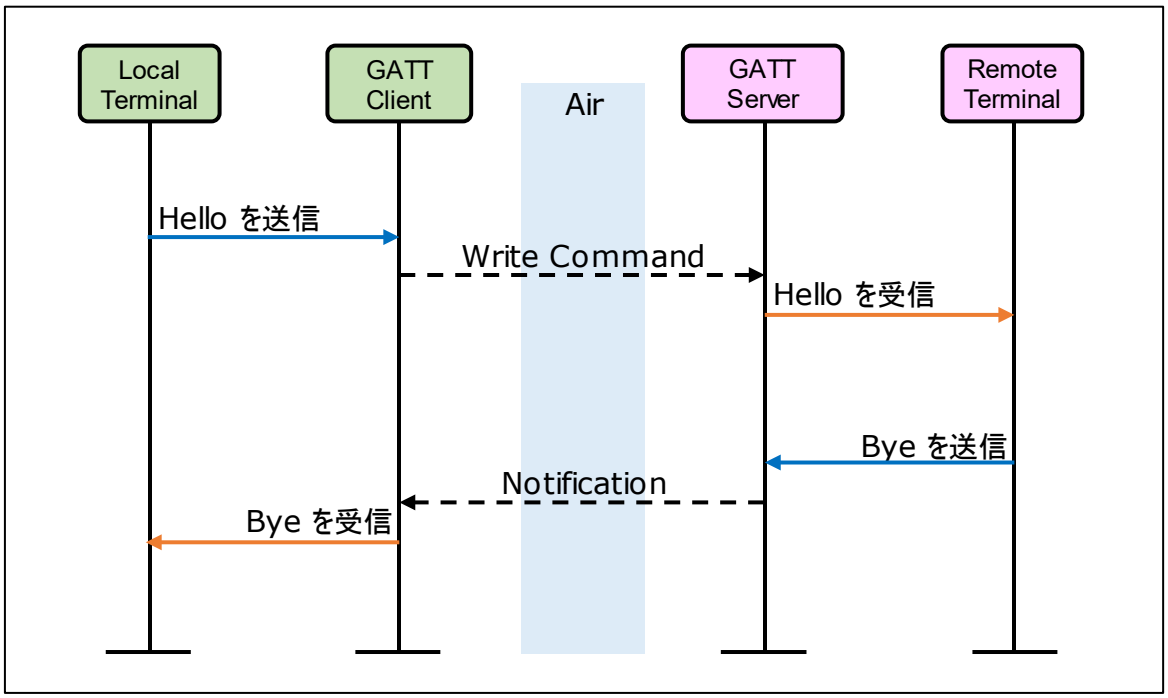


図 5-2 データの送受信時のシーケンス（レスポンス無し）

5.2　送信データのバッファリング

アプリケーションではターミナルソフトに入力したデータの取りこぼしを防ぐために、バッファリングを行います。

5.2.1　レスポンス有りの通信

Write Request や Indication でデータを送信後、Response や Confirmation を受け取るまでの期間は、文字の送信ができません。この期間中に、ユーザが入力したデータの取りこぼしを防ぐために、仮想 UART プロファイル内でバッファリングします。Response や Confirmation を受信時に、バッファ内にデータが格納されている場合は、すぐさまバッファ内のデータの送信を開始します。

以下に、送信データのバッファリングの概念図を示します。

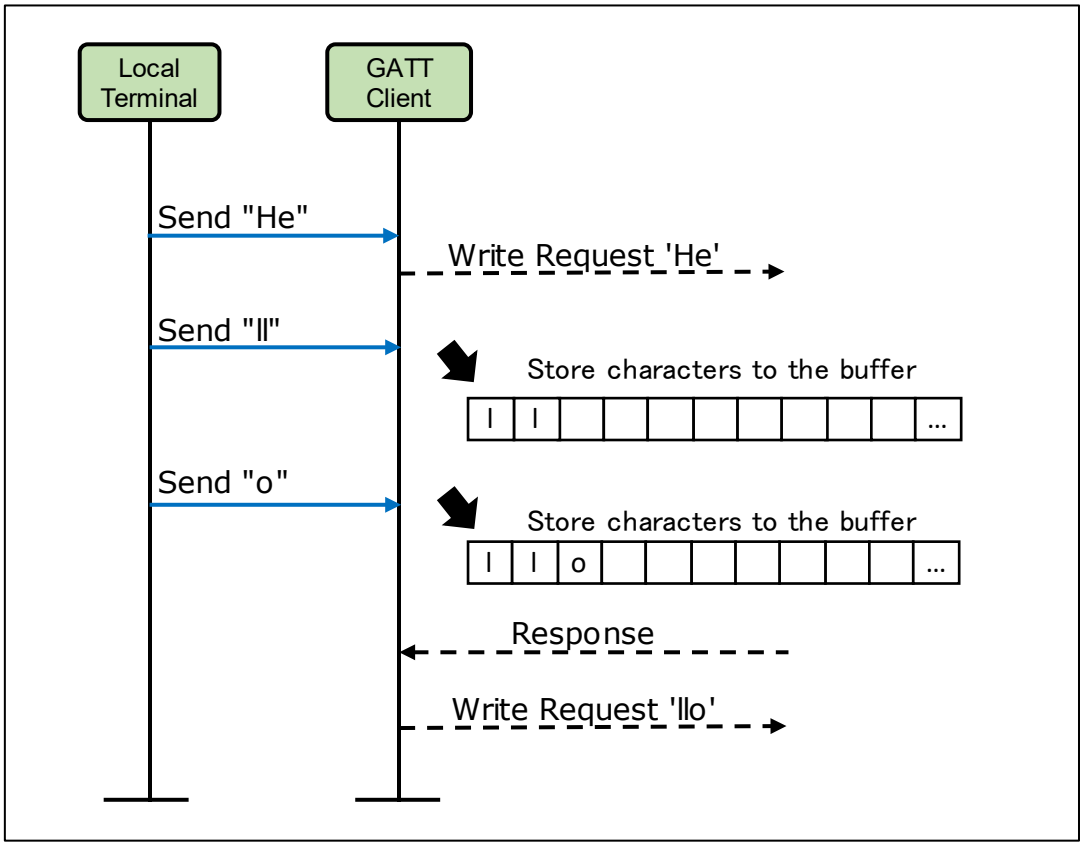


図 5-3 送信データのバッファリング（レスポンス有り）

5.2.2 レスポンス無しの通信

Write Command や Notification でデータを送信後、RWKE のタイマ機能(ke_timer_set)を使用し一定時間待ち、ターミナルソフトからのデータをある程度バッファリングします。そしてタイマ機能経過後に送信を行います。これはターミナルソフトからのデータを受け取るたびに送信を行うと、小さい単位(1 バイトや 2 バイト)で送信されてしまい通信効率が落ちることを防ぎます。また、ユーザが入力したデータの取りこぼしを防ぎます。

以下に、送信データのバッファリングの概念図を示します。

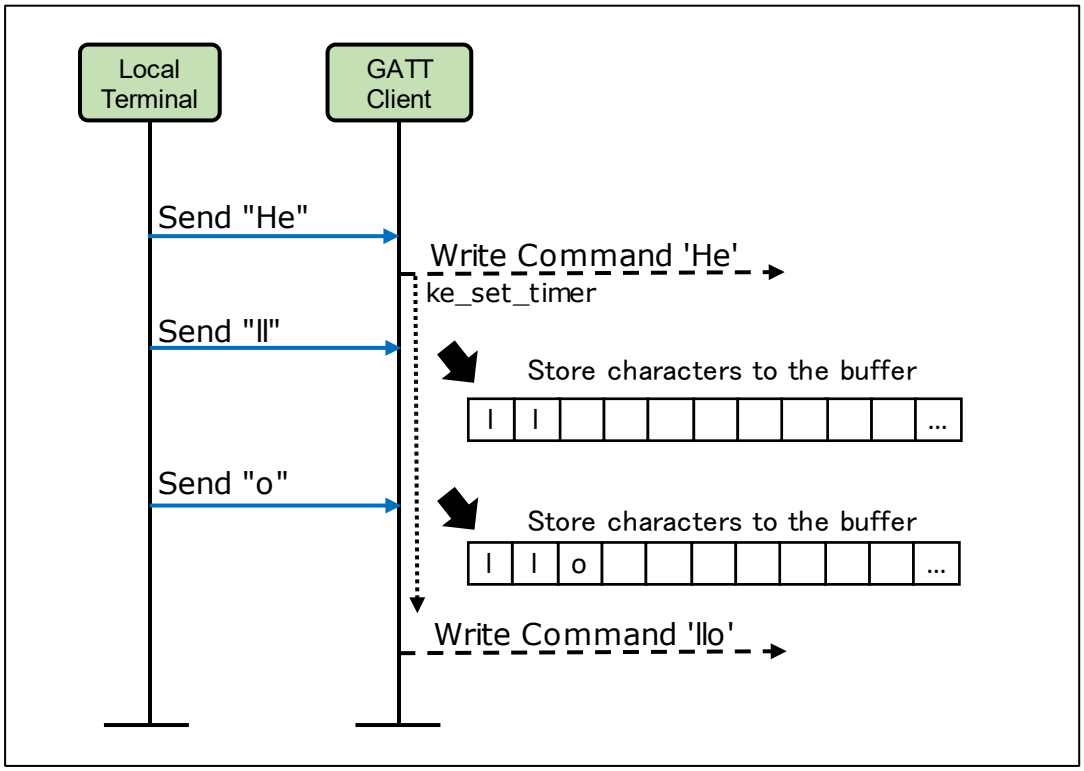


図 5-4 送信文字のバッファリング (レスポンス無し)

5.3 Bluetooth LE 通信の暗号化

Bluetooth LE 通信の暗号化を実施します。アプリケーションはペアリング情報を保持しないため、ペアリングは接続毎に行います。

6. 省電力機能

6.1 CPU の STOP モードへの移行

ユーザからのデータ入力やリモートデバイスからのデータの受信が 3 秒間継続してない場合、CPU を STOP モードに移行させることで、消費電力を低減します。

CPU が STOP モード中に、ユーザからの入力やリモートデバイスからの文字の受信があった場合は、すぐさま STOP モードから復帰します。

CPU が STOP モードに移行している場合、RL78/G1D 評価ボードの LED1/LED2 の点滅が停止、もしくは、点滅の周期が長くなります。

6.2 アドバタイジング間隔の変更

アドバタイジングを開始後、30 秒間継続して接続が開始されない場合は、アドバタイジング間隔を長くすることにより、消費電力を低減します。

アドバタイジング間隔が長くなった後、接続・切断をして、再度アドバタイジングを開始する際には、通常のアドバタイジング間隔に戻ります。または、AT-R を実行した場合も通常のアドバタイジング間隔に戻ります。

通常のアドバタイジング間隔は 30 ミリ秒、省電力時のアドバタイジング間隔は 3 秒です。

アドバタイジング間隔を変更しないようにする場合は、下記コードをコメントアウトしてください。

r_vuart_app.c, line 60, line1009

```
60:   #define APP_ADV_LOW_POWER_DURATION (3000)

1009: ke_timer_set(RBLE_APP_ADV_EVT, TASK_CON_APPL, APP_ADV_LOW_POWER_DURATION);
```

7. ビルドと動作確認

7.1 環境

アプリケーションのビルドと動作確認で使用する環境を以下に示します。

- ハードウェア環境
  - ホストマシン
    - PC/AT™互換機
    - プロセッサ　: 1.6GHz 以上
    - メインメモリ　: 1G Byte 以上
    - インタフェース : USB2.0 (E1 および RL78/G1D 評価ボードとの接続用)
  - デバイス
    - RL78/G1D 評価ボード (RTK0EN0001010001BZ)

- 使用ツール
  - 以下のオンチップデバッギングエミュレータを使用することができます。

    ・E1 エミュレータ
    ・E2 エミュレータ
    ・E2 エミュレータ Lite

    ※ 本アプリケーションノートでは E1 エミュレータを使用しています。また、E1 エミュレータは使用部品の EOL（生産終了）に伴い、生産を終了させていただくことになりました。詳細及び後継品につきましては下記よりご確認ください。
    ≫【お知らせ】E1 エミュレータ生産中止予告（ツールニュース）

- ソフトウェア環境
  - Windows® 7 以降
  - 以下の環境に対応
    ・e^2 studio 2020-07 (64 ビット版) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ・e^2 studio V7.8.0 (32 ビット版)　/ RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ・Renesas CS+ for CC V8.04.00 / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ・Renesas CS+ for CA, CX V4.04.00 / Renesas CA78K0R V1.72
  - Renesas Flash Programmer v3.06.02
  - Tera Term Version 4.105
  - UART-USB 変換デバイスドライバ（※）

    ※ 評価ボードと PC を接続する際に、UART-USB 変換 IC「FT232RL」のデバイスドライバを要求される場合があります。その際にはドライバを以下から入手してください。
    - FTDI(Future Technology Devices International) – Drivers
    https://ftdichip.com/drivers/d2xx-drivers/

7.2 ビルド手順

アプリケーションは、以下の環境を用いてビルドすることができます。

ビルドに必要なファイルは全て本アプリケーションノートに含まれているため BLE プロトコルスタックや EEPROM エミュレーションライブラリのダウンロードは必要ありません。

- e² studio 2020-07 (64 ビット版) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- e² studio V7.8.0 (32 ビット版) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- Renesas CS+ for CC V8.04.00 / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- Renesas CS+ for CA, CX V4.04.00 / Renesas CA78K0R V1.72

※ビルドする前に「8.8 マクロ設定」を参照しアプリケーションの設定を行ってください。

7.2.1 e² studio (32 ビット版/64 ビット版)

1. 32 ビット版または 64 ビット版の Renesas e² studio を起動します。
2. [ファイル] → [インポート]を選択し、インポートダイアログを表示します。
3. [一般] → [既存プロジェクトをワークスペースへ]を選択し、[次へ]をクリックします。
4. [ルートディレクトリの選択]で表 7-1 の Project Folder に示すのパスを選択し、[プロジェクト]に rBLE_Emb が表示され、チェックが入っていることを確認し、[終了]をクリックします。
5. [プロジェクト・エクスプローラー]の rBLE_Emb の上で右クリックすると表示されるメニューから[プロジェクトのビルド]を選択し、ビルドを行います。
6. 表 7-1 の Firmware に示すパスにファームウェアが生成されます。

表 7-1 プロジェクトファイルとファームウェアのパス (e2 studio)

e² studio with CC-RL	
Project Folder	¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Embedded¥rBLE_Emb
Firmware	¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Embedded¥rBLE_Emb¥DefaultBuild¥ rBLE_Emb_CCRL.hex

7.2.2 CS+ for CC / CS+ for CA, CX

1. 表 7-2 の Project Folder に示すプロジェクトファイルをダブルクリックします。これにより CS+が起動します。
2. 「プロジェクトツリー」内の「BLE_Emb（サブプロジェクト）」を右クリックし、ドロップダウンメニューから「BLE_Emb をビルド」を選択して、ビルドを開始します。
3. 表 7-2 の Firmware に示すパスにファームウェアが生成されます。

表 7-2 プロジェクトファイルとファームウェアのパス (CS+)

CS+ for CC	
Project File	¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Embedded¥BLE_Embedded.mtpj
Firmware	¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Embedded¥rBLE_Emb¥DefaultBuild¥ rBLE_Emb_CCRL.hex
CS+ for CA, CX	
Project File	¥Project_Source¥renesas¥tools¥project¥CS_CACX¥BLE_Embedded¥BLE_Embedded.mtpj
Firmware	¥Project_Source¥renesas¥tools¥project¥CS_CACX¥BLE_Embedded¥BLE_Emb¥DefaultBuild¥ BLE_Emb.hex

7.2.3 BLE プロトコルスタック/EEPROM エミュレーションライブラリ

本アプリケーションで使用されている BLE プロトコルスタックと EEPROM エミュレーションライブラリは、それぞれルネサスの Web ページから入手することができます。

- BLE プロトコルスタック
  - https://www.renesas.com/software-tool/bluetooth-low-energy-protocol-stack-rl78-family
- EEPROM エミュレーションライブラリ
  - RL78 ファミリ EEPROM エミュレーションライブラリ Pack02 パッケージ Ver.2.00(CA78K0R/CC-RL コンパイラ用)
    - https://www.renesas.com/software-tool/data-flash-libraries

    ※ 上記に示すダウンロードリンクは、Web サインのリニューアルなどにより、予告なく変更になる場合があります。

7.3 実行環境の準備

1. ファームウェアを RL78/G1D 評価ボードに書き込みます。書き込みの手順は「Bluetooth® Low Energy プロトコルスタック クイックスタートガイド」(R01AN2767)の「5.プログラムを書き込む」を参照してください。ファームウェアの書き込みは、2 台の RL78/G1D 評価ボードに対して行います。

   本アプリケーションノートに含まれているビルド済みの HEX ファイルを利用することもできます。ここでは CC-RL の HEX ファイルを使用します。

表 7-3 ビルド済み HEX ファイル

機能	ファイル
文字データ送受信	¥ROM_File¥ccrl¥RL78_G1D_CCE(VUART_CHAR).hex
バイナリデータ送受信	¥ROM_File¥ccrl¥RL78_G1D_CCE(VUART_BIN).hex

2. 図 1-1 に示すように、2 台の RL78/G1D 評価ボードをそれぞれ USB ケーブル経由で PC に接続します。
3. PC 上でターミナルソフトを起動し、表 7-4 にしたがって設定します。このマニュアルではターミナルソフトとして、Tera Term を使用します。

表 7-4 ターミナルソフトの設定値

設定項目	設定値
改行コード 受信	LF
改行コード 送信	CR
ボー・レート	4800 [bps]
データ長	8 [bit]
パリティ	none
ストップビット	1 [bit]
フロー制御	none

4. 2 台の RL78/G1D 評価ボードのディップスイッチを表 7-5 にしたがって設定します。

表 7-5 ディップスイッチの設定

ディップスイッチ	設定
SW6-1	OFF (左側 ： 簡易 AT コマンドモード) ※「7.4.2 バイナリデータ送受信」で使用します。
SW6-4	OFF (左側 ： レスポンス有り通信)

7.4　使用例

7.4.1　文字データ送受信

この例では、アドレスの設定を行い、Bluetooth LE 接続を確立後、文字データの送受信、最後に Bluetooth LE 接続の切断を行っています。

下記使用例を実行した場合のターミナルの実行結果を図 7-1 に示します。図 7-2, 図 7-3 には使用例のシーケンス図を示します。図内の赤字の番号は、以下の手順の番号に対応します。

1. ローカルデバイスおよびリモートデバイスのデバイスアドレスを設定します。デバイスアドレスの設定を行う場合は、「AT-AS=<addr>」コマンドを使用します。例えば、デバイスアドレスを「12:34:56:78:9A:BC」に設定する場合は、「AT-AS=123456789ABC」を実行します。現状のデバイスアドレスを確認する場合は、「AT-AS?」を実行します。デバイスアドレスが、00:00:00:00:00:00 になっている場合や、ローカルデバイスとリモートデバイスに同じデバイスアドレスが設定されている場合は変更が必要です。以下では、ローカルデバイスのデバイスアドレスを 12:34:56:78:9A:BC に、リモートデバイスを CB:A9:87:65:43:21 に設定した場合について記載します。
2. 「AT-AS=<addr>」によりデバイスアドレスを設定した場合、変更を反映するためにデバイス上の RESET ボタン(SW5)を押下してリセットを行う必要があります。
3. ローカルデバイスのターミナル上で簡易 AT コマンド「AT-AP=CBA987654321」を実行することで、接続を行いたいデバイスのアドレスを設定します。
4. ローカルデバイスのターミナル上にて、簡易 AT コマンド「AT-C」 を実行し、アドレスが「CB:A9:87:65:43:21」であるリモートデバイスへの接続を開始します。接続が完了すると、ローカルデバイスおよびリモートデバイスのターミナル上に接続が完了したことを示す「CONNECT」が表示されます。
5. ローカルデバイスのターミナル上で、エスケープキーを押下し、簡易 AT コマンドモードから仮想 UART モードに切り替えます。
6. ローカルデバイスのディップスイッチ(SW6-4)を OFF(左側:レスポンス有り通信)に設定します。
7. ローカルデバイスのターミナル上で「Hello」と入力します。リモートデバイスのターミナル上に、「Hello」と表示されます。
8. リモートデバイスのターミナル上で、エスケープキーを押下し、簡易 AT コマンドモードから仮想 UART モードに切り替えます。
9. リモートデバイスのディップスイッチ(SW6-4)を OFF(左側:レスポンス有り通信)に設定します。
10. リモートデバイスのターミナル上で「Bye」と入力します。ローカルデバイスのターミナルに「Bye」と表示されます。
11. ローカルデバイスのディップスイッチ(SW6-4)を ON(右側:レスポンス無し通信)に設定します。
12. ローカルデバイスのターミナル上で「Hello」と入力します。リモートデバイスのターミナル上に、「Hello」と表示されます。
13. リモートデバイスのディップスイッチ(SW6-4)を ON(右側:レスポンス無し通信)に設定します。
14. リモートデバイスのターミナル上で「Bye」と入力します。ローカルデバイスのターミナルに「Bye」と表示されます。
15. ローカルデバイスのターミナル上で、エスケープキーを押下し、仮想 UART モードから簡易 AT コマンドモードに切り替えます。
16. ローカルデバイスのターミナル上で、簡易 AT コマンド「AT-R」を実行し、接続を切断します。切断が完了すると、ローカルデバイスおよびリモートデバイスのターミナル上に切断が完了したことを示す「DISCONNECT」が表示されます。

ローカルデバイスのターミナル

```
AT-AS=123456789ABC      1.

OK                      2. Reset this device
AT-AP=CBA987654321      3.

OK
AT-C                    4.

OK

CONNECT

[Virtual UART Mode]     5. Press Esc key
Hello                   7.
Bye                     10.
Hello                   12.
Bye                     14.

[AT Command Mode]       15. Press Esc key
AT-R                    16.

OK

DISCONNECT
```

リモートデバイスのターミナル

```
AT-AS=CBA987654321      1.

OK                      2. Reset this device

CONNECT                 4.

Hello                   7.

[Virtual UART Mode]     8. Press Esc key
Bye                     10.
Hello                   12.
Bye                     14.

DISCONNECT              16.
```

図 7-1 ターミナルの実行結果

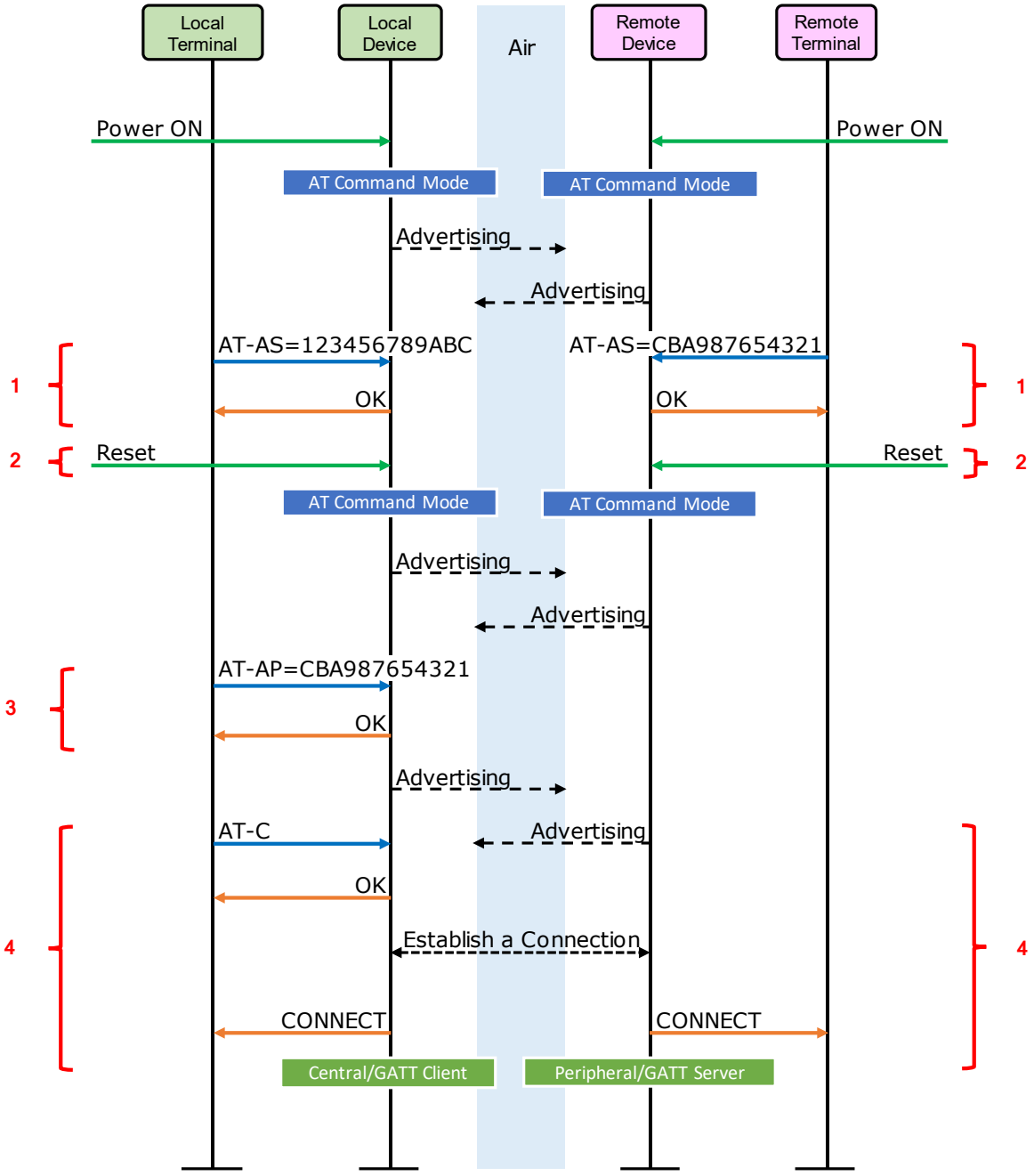


図 7-2 文字データ送受信 実行例のシーケンス (1/2)

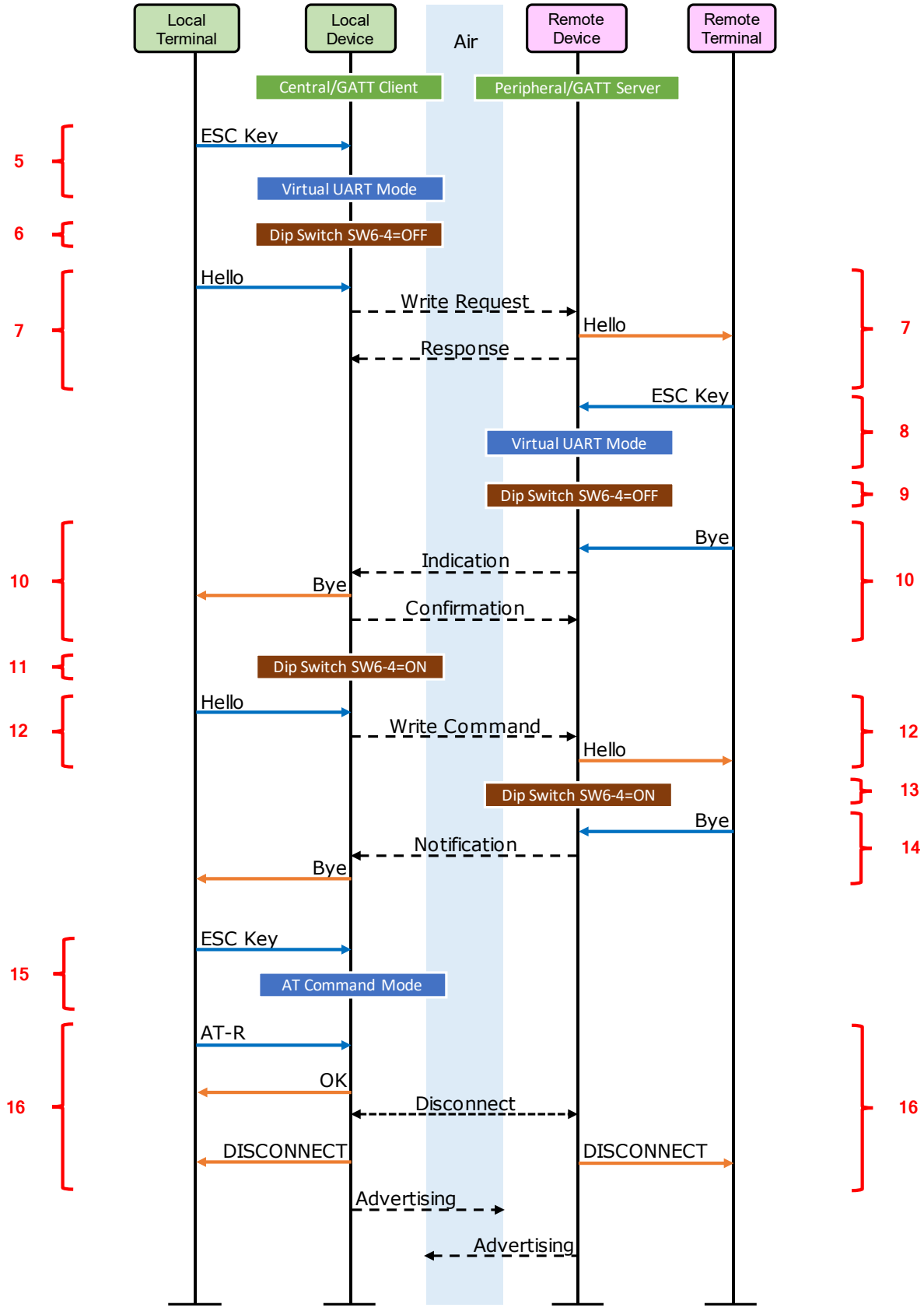


図 7-3 文字データ送受信 実行例のシーケンス (2/2)

7.4.2　バイナリデータ送受信

この例では、アドレスの設定を行い、Bluetooth LE 接続を確立後、バイナリデータの送受信、最後に Bluetooth LE 接続の切断を行っています。

また、バイナリデータの送受信を行う際にターミナルのマクロ機能を利用します。ターミナルの設定と使用するマクロファイルを下記に示します。

- ターミナル設定

ターミナルのメニューから、「設定」→「全般」を選択し「Tera Term: 全般設定」のダイアログを表示します。「言語」を「English」に設定します。

表 7-6 マクロファイル(¥r01an3130xx0114-rl78g1d-ble-vuart¥Macro¥)

ファイル名	説明
tt_debug_mode_on.ttl	Tera Term のデバッグモードを ON にします。 ターミナルの表示が 16 進数になります。
tt_debug_mode_off.ttl	Tera Term のデバッグモードを OFF にします。 ターミナルの表示が初期状態になります。
tt_send_bin_1.ttl	バイナリデータ(0x00 0x01 0x02 0x03 0x04)を送信します。
tt_send_bin_2.ttl	バイナリデータ(0xF0 0xF1 0xF2 0xF3 0xF4)を送信します。

下記使用例を実行した場合のターミナルの実行結果を図 7-4 に示します。図 7-5, 図 7-6 には使用例のシーケンス図を示します。図内の赤字の番号は、以下の手順の番号に対応します。

1. ローカルデバイスとリモートデバイスのディップスイッチ(SW6-1)を OFF(左側:簡易 AT コマンドモード)に設定します。
2. ローカルデバイスおよびリモートデバイスのデバイスアドレスを設定します。デバイスアドレスの設定を行う場合は、「AT-AS=<addr>」コマンドを使用します。例えば、デバイスアドレスを「12:34:56:78:9A:BC」に設定する場合は、「AT-AS=123456789ABC」を実行します。現状のデバイスアドレスを確認する場合は、「AT-AS?」を実行します。デバイスアドレスが、00:00:00:00:00:00 になっている場合や、ローカルデバイスとリモートデバイスに同じデバイスアドレスが設定されている場合は変更が必要です。以下では、ローカルデバイスのデバイスアドレスを 12:34:56:78:9A:BC に、リモートデバイスを CB:A9:87:65:43:21 に設定した場合について記載します。
3. 「AT-AS=<addr>」によりデバイスアドレスを設定した場合、変更を反映するためにデバイス上の RESET ボタン(SW5)を押下してリセットを行う必要があります。
4. ローカルデバイスのターミナル上で簡易 AT コマンド「AT-AP=CBA987654321」を実行することで、接続を行いたいデバイスのアドレスを設定します。
5. ローカルデバイスのターミナル上にて、簡易 AT コマンド「AT-C」 を実行し、アドレスが「CB:A9:87:65:43:21」であるリモートデバイスへの接続を開始します。接続が完了すると、ローカルデバイスおよびリモートデバイスのターミナル上に接続が完了したことを示す「CONNECT」が表示されます。
6. ローカルデバイスとリモートデバイスのディップスイッチ(SW6-1)を ON(右側:仮想 UART モード)に設定します。
7. ローカルデバイスとリモードデバイスのディップスイッチ(SW6-4)を OFF(左側:レスポンス有り通信)に設定します。

8. ローカルデバイスとリモートデバイスのターミナルでマクロを実行し、ターミナルをデバッグモードにします。
   ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_debug_mode_on.ttl"を選択し「開く」ボタンを押します。
9. ローカルデバイスからバイナリデータを送信します。
   ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_send_bin_1.ttl"を選択し「開く」ボタンを押します。リモートデバイスのターミナル上に「00 01 02 03 04」と表示されます。
10. リモートデバイスからバイナリデータを送信します。
    ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_send_bin_2.ttl"を選択し「開く」ボタンを押します。ローカルデバイスのターミナル上に「F0 F1 F2 F3 F4」と表示されます。
11. ローカルデバイスとリモードデバイスのディップスイッチ(SW6-4)を ON(右側:レスポンス無し通信)に設定します。
12. ローカルデバイスからバイナリデータを送信します。
    ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_send_bin_2.ttl"を選択し「開く」ボタンを押します。リモートデバイスのターミナル上に「F0 F1 F2 F3 F4」と表示されます。
13. リモートデバイスからバイナリデータを送信します。
    ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_send_bin_1.ttl"を選択し「開く」ボタンを押します。ローカルデバイスのターミナル上に「00 01 02 03 04」と表示されます。
14. ローカルデバイスとリモートデバイスのターミナルでマクロを実行し、ターミナルのデバッグモードを終了します。
    ターミナルのメニューから、「コントロール」→「マクロ」を選択し、Open macro のダイアログを表示します。表 7-6 のマクロファイル"tt_debug_mode_off.ttl"を選択し「開く」ボタンを押します。
15. ローカルデバイスのディップスイッチ(SW6-1)を OFF(左側:簡易 AT コマンドモード)に設定します。
16. ローカルデバイスのターミナル上で、簡易 AT コマンド「AT-R」を実行し、接続を切断します。切断が完了すると、ローカルデバイスおよびリモートデバイスのターミナル上に切断が完了したことを示す「DISCONNECT」が表示されます。

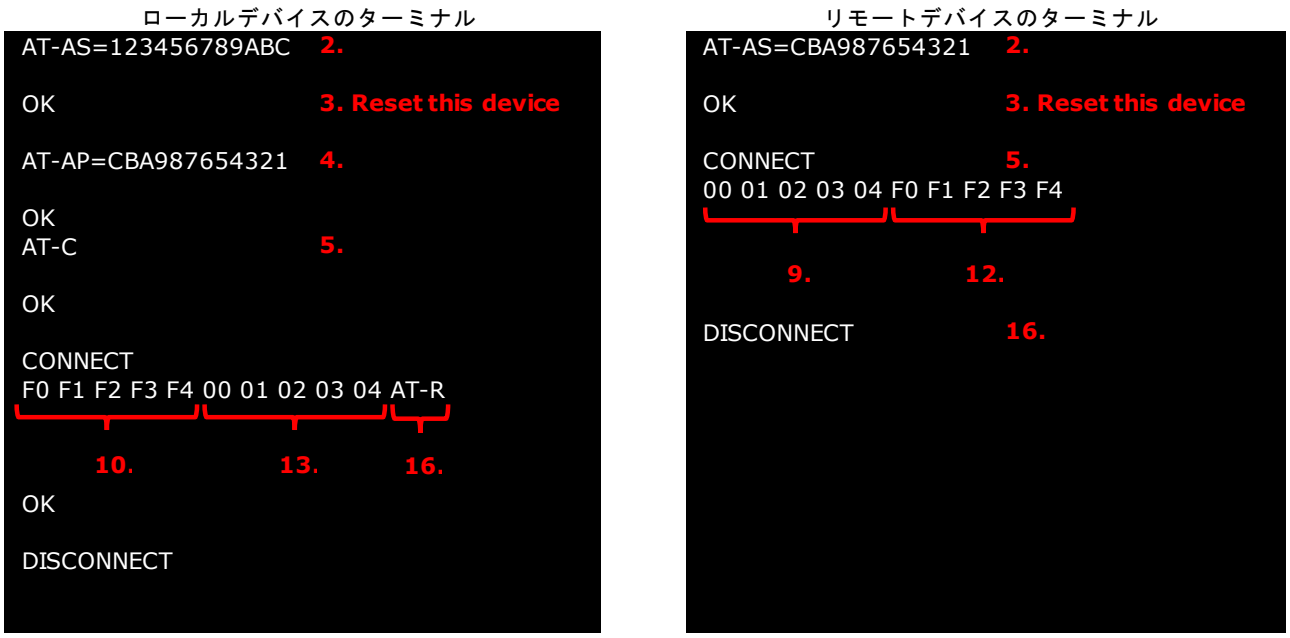


図 7-4 ターミナルの実行結果

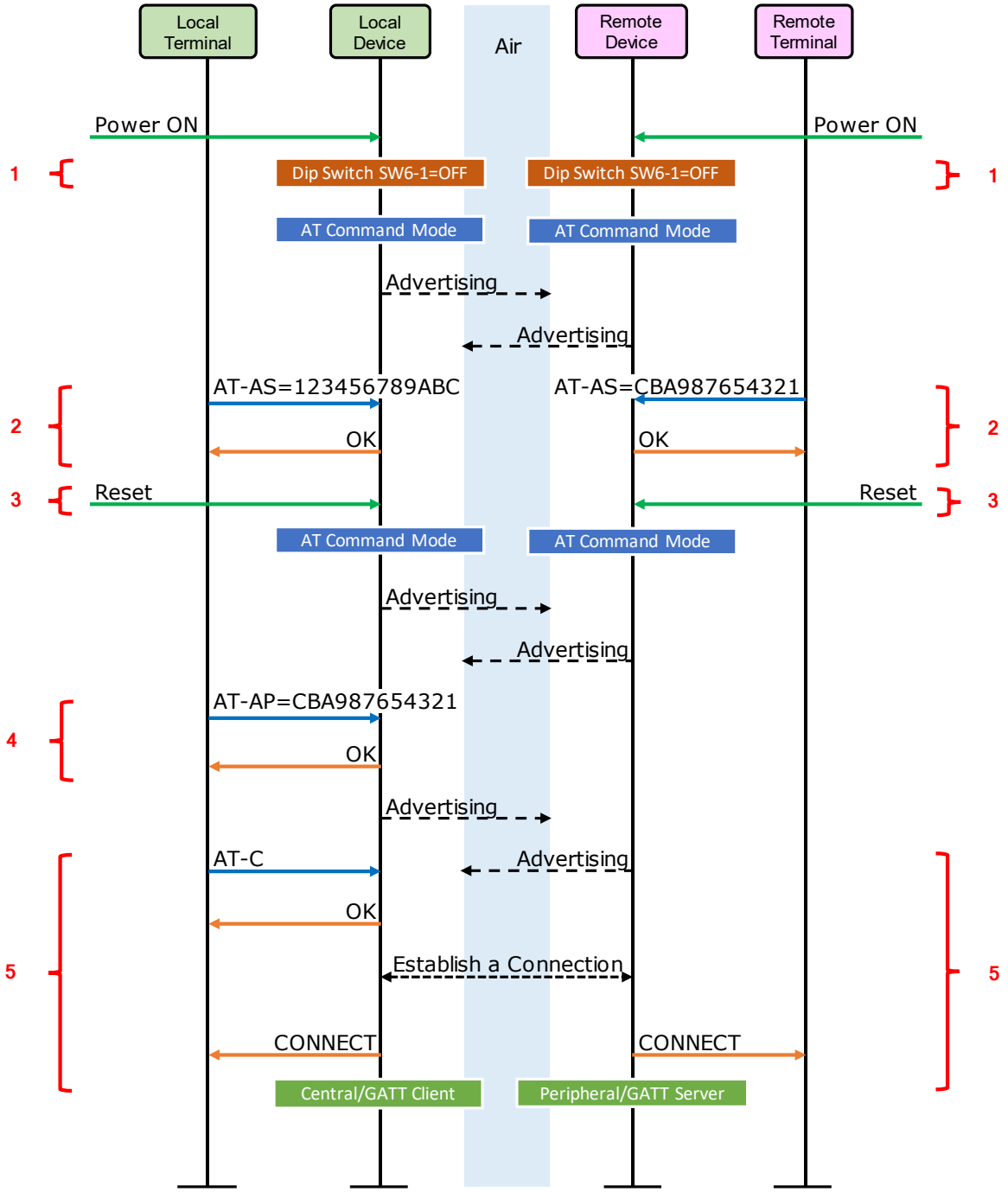


図 7-5 バイナリデータ送受信 実行例のシーケンス (1/2)

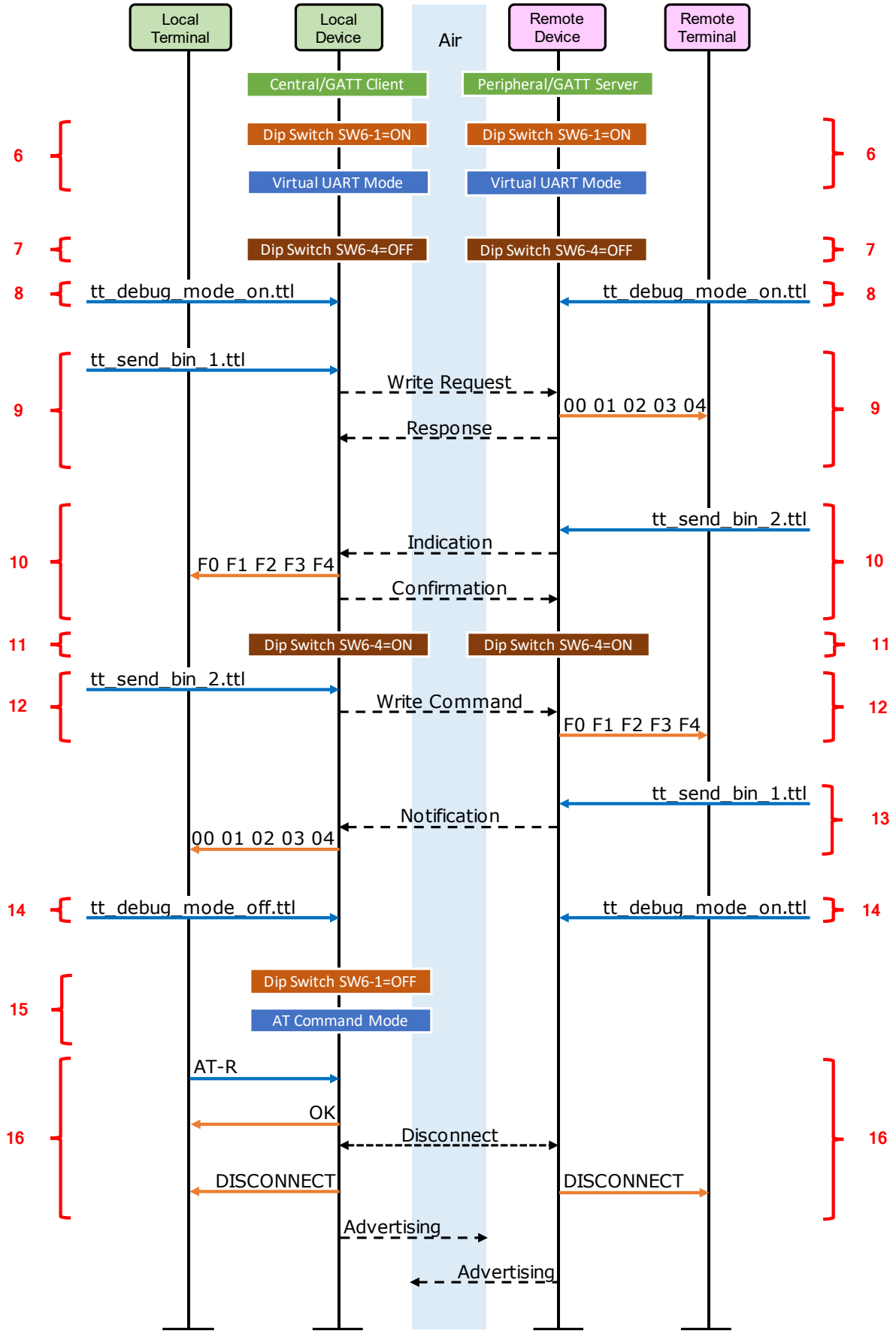


図 7-6 バイナリデータ送受信 実行例のシーケンス (2/2)

8. 実装の詳細

8.1 仮想 UART プロファイル

仮想 UART プロファイルの仕様を表 8-1、表 8-2 に示します。

表 8-1 仮想 UART プロファイルの仕様（1/2）

アトリビュートハンドル	アトリビュートタイプとその値
VUART_HDL_SVC 0x000C （※）	Type: Primary Service Declaration UUID: D68C0001-A21B-11E5-8CB8-0002A5D5C51B 仮想 UART サービス
VUART_HDL_INDICATION_CHAR 0x000D （※）	Type: Characteristic Declaration UUID: D68C0002-A21B-11E5-8CB8-0002A5D5C51B Property: Indicate サーバからクライアントへのデータ送信に使用します。
VUART_HDL_INDICATION_VAL 0x000E （※）	Type: Indication Value 本 Characteristic にデータを設定後、Indication を送信することで、サーバからクライアントへのデータ送信を行います。一度に送信可能なデータ数は、最大 20 バイトです。
VUART_HDL_INDICATION_CFG 0x000F （※）	Type: Client Characteristic Configuration Descriptor クライアントが、サーバの Indication を許可・不許可を設定するために使用します。
VUART_HDL_WRITE_CHAR 0x0010 （※）	Type: Characteristic Declaration UUID: D68C0003-A21B-11E5-8CB8-0002A5D5C51B Property: Write クライアントからサーバへのデータ送信に使用します。
VUART_HDL_WRITE_VAL 0x0011 （※）	Type: Write Value 本 Characteristic に Write Request によりデータを書き込むことで、クライアントからサーバへのデータ送信を行います。一度に送信可能なデータ数は、最大 20 バイトです。

※ アトリビュートハンドルの 16 進値は、ファームウェア内に組み込むプロファイルによって変わります

表 8-2 仮想 UART プロファイルの仕様（2/2）

アトリビュートハンドル	アトリビュートタイプとその値
VUART_HDL_NOTIFICATION_CHAR 0x0012 （※）	Type: Characteristic Declaration UUID: D68C0004-A21B-11E5-8CB8-0002A5D5C51B Property: Notify サーバからクライアントへのデータ送信に使用します。
VUART_HDL_NOTIFICATION_VAL 0x0013 （※）	Type: Notification Value 本 Characteristic にデータを設定後、Notification を送信することで、サーバからクライアントへのデータ送信を行います。一度に送信可能なデータ数は、最大 20 バイトです。
VUART_HDL_NOTIFICATION_CFG 0x0014 （※）	Type: Client Characteristic Configuration Descriptor クライアントが、サーバの Notification を許可・不許可を設定するために使用します。
VUART_HDL_WRITE_NORESP_CHAR 0x0015 （※）	Type: Characteristic Declaration UUID: D68C0005-A21B-11E5-8CB8-0002A5D5C51B Property: Write クライアントからサーバへの文字送信に使用します。
VUART_HDL_WRITE_NORESP_VAL 0x0016 （※）	Type: Write Value 本 Characteristic に Write Command によりデータを書き込むことで、クライアントからサーバへのデータ送信を行います。一度に送信可能なデータ数は、最大 20 バイトです。

※ アトリビュートハンドルの 16 進値は、ファームウェア内に組み込むプロファイルによって変わります。

8.2 アドバタイジング動作

アプリケーションのアドバタイジングの初期設定値を表 8-3 に示します。

表 8-3 アドバタイジングの仕様

Advertising Type		Connectable undirected advertising (ADV_IND)
Advertising Interval Min		通常時: 20 [ms]、省電力時: 1.5 [s]
Advertising Interval Max		通常時: 30 [ms]、省電力時: 3 [s]
Advertising Channel Map		All Channels (37, 38, 39 ch)
Advertising Data		-
	Length of this Data	2 [bytes]
	Data Type	<<Flags>> (0x01)
	Flags	LE General Discoverable Mode BR/EDR Not Supported
	Length of this Data	8 [bytes]
	Data Type	<<Complete Local Name>> (0x09)
	Local Name	REL-BLE
	Length of this Data	17 [bytes]
	Data Type	<< Complete List of 128-bit Service Class UUIDs>> (0x07)
	UUID	D68C0001-A21B-11E5-8CB8-0002A5D5C51B
	Scan Response Data	none

8.3 接続動作

アプリケーションの接続に関する初期設定値を表 8-4 に示します。

表 8-4 接続仕様

Scan Interval	30 [ms]
Scan Window Size	30 [ms]
Initiator Filter Policy	Accept List を使用しない
Peer Address Type	パブリックアドレス
Peer BD Address	AT-C もしくは AT-AP により指定
Own Address Type	パブリックアドレス
Minimum of Connection Interval	30 [ms]
Maximum of Connection Interval	30 [ms]
Connection Latency	0 [ms]
Link Supervision Timeout	5 [s]
Minimum CE Length	0 [ms]
Maximum CE Length	50 [ms]

8.4 ペアリング動作

アプリケーションのペアリングに関する初期設定値を表 8-5 に示します。

表 8-5 ペアリングの仕様

Bonding	Bondable Mode
Security Mode	Unauthenticated pairing with encryption
Pairing Method	Just Works
IO capability	No Input No Output
OOB flag	OOB Data not present
Authentication Requirements	No MITM Bonding
Encryption key size	128 [bit]
Initiator key distribution	None
Responder key distribution	Encryption key

8.5 仮想 UART 機能の API

仮想 UART 機能が提供する定義や API を以下に示します。

8.5.1 仮想 UART 定義

- イベントタイプ列挙型宣言

```
typedef enum {
  // Server Role
  RBLE_VUART_EVENT_SERVER_ENABLE_CMP = 0x01,// Role 有効設定完了イベント
  RBLE_VUART_EVENT_SERVER_WRITE_REQ,         // Write Request 受信イベント
  RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ, // Write Command 受信イベント
  RBLE_VUART_EVENT_SERVER_INDICATION_CFM,    // Indication 送信完了イベント
  RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP,  // Notification 送信イベント
  // Client Role
  RBLE_VUART_EVENT_CLIENT_ENABLE_CMP = 0x81,// Role 有効設定完了イベント
  RBLE_VUART_EVENT_CLIENT_INDICATION,        // Indication 受信イベント
  RBLE_VUART_EVENT_CLIENT_NOTIFICATION,     // Notification 受信イベント
  RBLE_VUART_EVENT_CLIENT_WRITE_RSP,         // Write Request 送信完了イベント
  RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP    // Write Command 送信イベント
} RBLE_VUART_EVENT_TYPE;
```

- イベントコールバック関数型宣言

```
typedef void (*RBLE_VUART_EVENT_HANDLER)(RBLE_VUART_EVENT *event);
```

- イベントパラメータ構造体

```
typedef struct RBLE_VUART_EVENT_t {
  RBLE_VUART_EVENT_TYPE type;          仮想 UART イベントタイプ
  union Event_Vuart_Paramter_u {
```

Server Role 有効設定完了イベント

```
    struct {
      RBLE_STATUS status;              ステータス
    } server_enable_cmp;
```

Server Role データ受信イベント (Write Request)

```
    struct {
      RBLE_STATUS status;              ステータス
      char value[20];                  受信データ
      uint16_t len;                    受信データのサイズ
    } server_write_req;
```

Server Role データ受信イベント (Write Command)

```
    struct {
      RBLE_STATUS status;              ステータス
      char value[20];                  受信データ
      uint16_t len;                    受信データのサイズ
    } server_write_noresp_req;
```

Server Role データ送信完了イベント (Indication)

```
struct {
  RBLE_STATUS status;              ステータス
} server_indicate_cnf;
```

Server Role データ送信完了イベント (Notification)

```
struct {
  RBLE_STATUS status;              ステータス
} server_notify_cmp;
```

Client Role 有効設定完了イベント

```
struct {
  RBLE_STATUS status;              ステータス
} client_enable_cmp;
```

Client Role データ受信イベント (Indication)

```
struct {
  RBLE_STATUS status;              ステータス
  char value[20];                  受信データ
  uint16_t len;                    受信データのサイズ
} client_indication;
```

Client Role データ受信イベント (Notification)

```
struct {
  RBLE_STATUS status;              ステータス
  char value[20];                  受信データ
  uint16_t len;                    受信データのサイズ
} client_notification;
```

Client Role データ送信完了イベント (Write Request)

```
struct {
  RBLE_STATUS status;              ステータス
} client_write_rsp;
```

Client Role データ送信完了イベント (Write Command)

```
struct {
  RBLE_STATUS status;              ステータス
} client_write_norsp;
} param;
} RBLE_VUART_EVENT;
```

8.5.2 関数

8.5.2.1 RBLE_VUART_Server_Enable

```
RBLE_STATUS RBLE_VUART_Server_Enable(
    uint16_t conhdl, RBLE_VUART_EVENT_HANDLER callback)
```

この関数は、仮想 UART 機能の Server Role を有効にします。

結果は、Server Role の有効設定完了イベント RBLE_VUART_EVENT_SERVER_ENABLE_CMP にて通知されます。

Parameters:

conhdl	コネクションハンドル
callback	仮想 UART イベントを通知するコールバック関数を指定

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.2.2 RBLE_VUART_Server_Disable

```
RBLE_STATUS RBLE_VUART_Server_Disable(void)
```

この関数は、仮想 UART 機能の Server Role を無効にします。

Parameters:

_	-

Return:

RBLE_OK	正常終了

8.5.2.3 RBLE_VUART_Server_Send_Indication

```
RBLE_STATUS RBLE_VUART_Server_Send_Indication(
    const char *chars, uint16_t len)
```

この関数は、サーバからクライアントへデータの送信を行います。

送信したデータがクライアントに受信され、クライアントが Confirmation を返却すると、データ送信完了イベント RBLE_VUART_EVENT_SERVER_INDICATION_CFM がサーバに通知されます。

Parameters:

chars	データを格納したバッファ
len	chars が指すデータのバイト数

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.2.4 RBLE_VUART_Server_Send_Notification

```
RBLE_STATUS RBLE_VUART_Server_Send_Notification(
    const char *chars, uint16_t len)
```

この関数は、サーバからクライアントへデータの送信を行います。

この関数を実行すると RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP がサーバに通知されます。

※このイベントは送信が完了したことを保証するものではありません。

Parameters:

chars	データを格納したバッファ
len	chars が指すデータのバイト数

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.2.5 RBLE_VUART_Client_Enable

```
RBLE_STATUS RBLE_VUART_Client_Enable(
    uint16_t conhdl, RBLE_VUART_EVENT_HANDLER callback)
```

この関数は、仮想 UART 機能の Client Role を有効にします。

結果は、Client Role の有効設定完了イベント RBLE_VUART_EVENT_CLIENT_ENABLE_CMP にて通知されます。

Parameters:

conhdl	コネクションハンドル
callback	仮想 UART イベントを通知するコールバック関数を指定

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.2.6 RBLE_VUART_Client_Disable

```
RBLE_STATUS RBLE_VUART_Client_Disable(void)
```

この関数は、仮想 UART 機能の Client Role を無効にします。

Parameters:

–	-

Return:

RBLE_OK	正常終了

8.5.2.7 RBLE_VUART_Client_Send_Chars

```
RBLE_STATUS RBLE_VUART_Client_Send_Chars(
    const char *chars, uint16_t len)
```

この関数は、クライアントからサーバへデータの送信を行います。

送信したデータがサーバに受信され、サーバが Response を返却すると、データ送信完了イベント RBLE_VUART_EVENT_CLIENT_WRITE_RSP がクライアントに通知されます。

Parameters:

chars	データを格納したバッファ
len	chars が指すデータのバイト数

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.2.8 RBLE_VUART_Client_Send_Chars_Noresp

```
RBLE_STATUS RBLE_VUART_Client_Send_Chars_Noresp(
    const char *chars, uint16_t len)
```

この関数は、クライアントからサーバへデータの送信を行います。

この関数を実行すると RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP がサーバに通知されます。

※このイベントは送信が完了したことを保証するものではありません。

Parameters:

chars	データを格納したバッファ
len	chars が指すデータのバイト数

Return:

RBLE_OK	正常終了
RBLE_STATUS_ERROR	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

8.5.3 イベント

仮想 UART 機能で定義されているイベントを以下に示します。

8.5.3.1 RBLE_VUART_EVENT_SERVER_ENABLE_CMP

RBLE_VUART_EVENT_SERVER_ENABLE_CMP		
このイベントは、Server Role の有効化設定の結果を通知します。		
Parameters:		
	status	有効化設定結果

8.5.3.2 RBLE_VUART_EVENT_SERVER_WRITE_REQ

RBLE_VUART_EVENT_SERVER_WRITE_REQ		
このイベントは、クライアントが RBLE_VUART_Client_Send_Chars 関数を使用して送信したデータを、サーバが受信したことを通知します。		
Parameters:		
	status	データ受信結果
	value	受信データ
	len	受信データのサイズ

8.5.3.3 RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ

RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ		
このイベントは、クライアントが RBLE_VUART_Client_Send_Chars_Noresp 関数を使用して送信したデータを、サーバが受信したことを通知します。		
Parameters:		
	status	データ受信結果
	value	受信データ
	len	受信データのサイズ

8.5.3.4 RBLE_VUART_EVENT_SERVER_INDICATION_CFM

RBLE_VUART_EVENT_SERVER_INDICATION_CFM		
このイベントは、サーバが RBLE_VUART_Server_Send_Indication 関数を使用して送信したデータの送信完了を通知します。		
Parameters:		
	status	データ送信結果

8.5.3.5 RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP

RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP		
このイベントは、サーバが RBLE_VUART_Server_Send_Notification 関数を使用してデータ送信したことを通知します。 ※このイベントは送信が完了したことを保証するものではありません。		
Parameters:		
	status	データ送信結果

8.5.3.6 RBLE_VUART_EVENT_CLIENT_ENABLE_CMP

RBLE_VUART_EVENT_CLIENT_ENABLE_CMP		
このイベントは、Client Role の有効設定の結果を通知します。		
Parameters:		
	status	有効化設定結果

8.5.3.7 RBLE_VUART_EVENT_CLIENT_INDICATION

RBLE_VUART_EVENT_CLIENT_INDICATION		
このイベントは、サーバが RBLE_VUART_Server_Send_Indication 関数を使用して送信したデータを、クライアントが受信したことを通知します。		
Parameters:		
	status	データ受信結果
	value	受信データ
	len	受信データのサイズ

8.5.3.8 RBLE_VUART_EVENT_CLIENT_NOTIFICATION

RBLE_VUART_EVENT_CLIENT_NOTIFICATION		
このイベントは、サーバが RBLE_VUART_Server_Send_Notification 関数を使用して送信したデータを、クライアントが受信したことを通知します。		
Parameters:		
	status	データ受信結果
	value	受信データ
	len	受信データのサイズ

8.5.3.9 RBLE_VUART_EVENT_CLIENT_WRITE_RSP

RBLE_VUART_EVENT_CLIENT_WRITE_RSP		
このイベントは、クライアントが RBLE_VUART_Client_Send_Chars 関数を使用して送信したデータの送信完了を通知します。		
Parameters:		
	status	データ送信結果

8.5.3.10 RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP

RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP		
このイベントは、クライアントが RBLE_VUART_Client_Send_Chars_Noresp 関数を使用してデータ送信したことを通知します。 ※このイベントは送信が完了したことを保証するものではありません。		
Parameters:		
	status	データ送信結果

8.6 アプリケーションの状態遷移

図 8-1 にアプリケーションの状態遷移図を示します。アプリケーションは、接続・切断イベントや簡易 AT コマンドの実行に応じて、状態が遷移します。

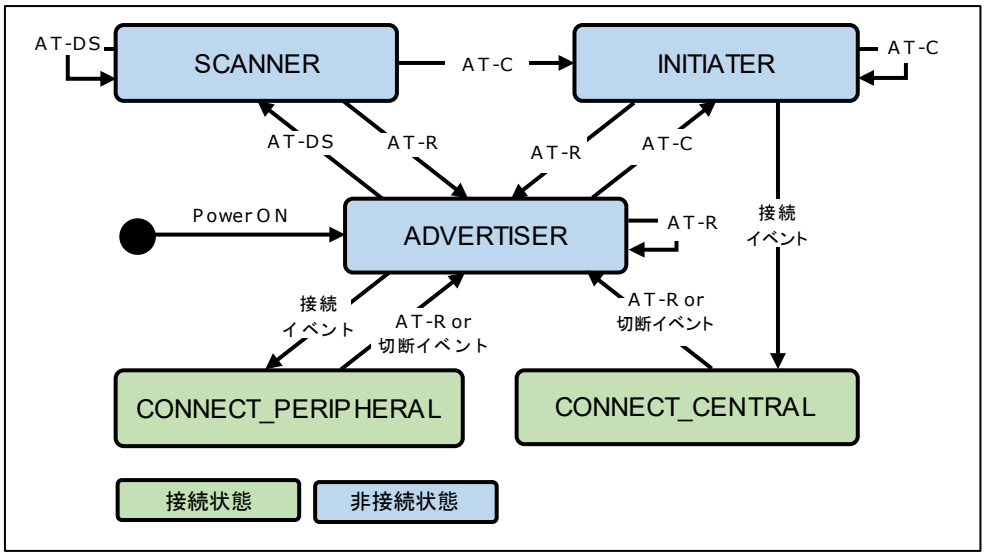


図 8-1 アプリケーションの状態遷移図

アプリケーションの状態の一覧を表 8-6 に示します。

表 8-6 アプリケーションの状態一覧

状態	説明
ADVERTISER	アドバタイジングを行っている状態です。
SCANNER	AT-DS を実行し、周辺デバイスのスキャンを行っている状態です。AT-DS が完了後もこの状態にとどまります。
INITIATER	AT-C を実行し、指定したアドレスへの接続を開始している状態です。
CONNECT_CENTRAL	Bluetooth LE 接続を確立した状態です。CONNECT_CENTRAL 状態の際は、GATT クライアントになります。
CONNECT_PERIPHERAL	Bluetooth LE 接続が確立された状態です。CONNECT_PERIPHERAL 状態の際は、GATT サーバになります。

8.7 アプリケーションの詳細シーケンス

起動、接続、文字の送受信、切断の詳細シーケンスを示します。各関数やイベントの詳細は、「Bluetooth® Low Energy プロトコルスタック API リファレンス 基本編」(R01UW0088J)を参照してください。

8.7.1 起動シーケンス

図 8-2 に起動時のシーケンスを示します。

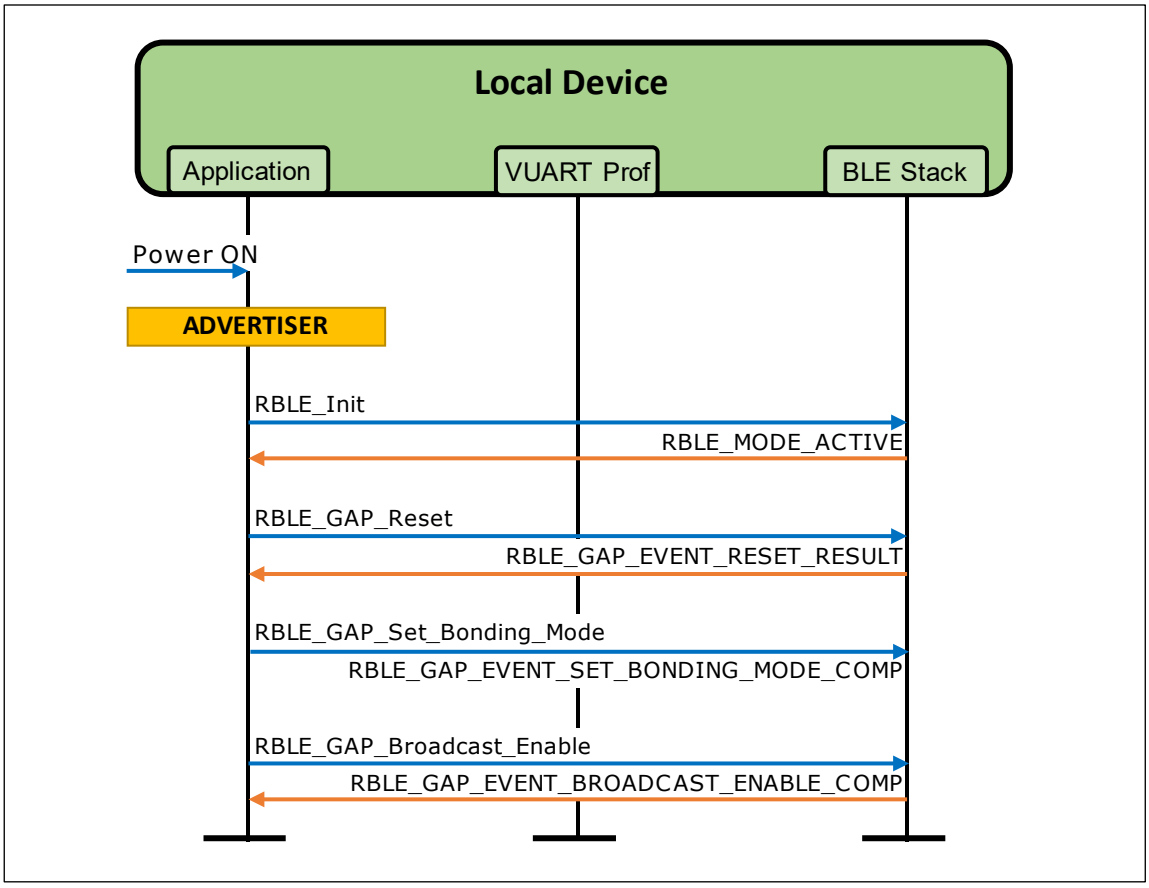


図 8-2 起動シーケンス

8.7.2 接続シーケンス

接続の開始から、接続の確立までのシーケンスを図 8-3 に示します。

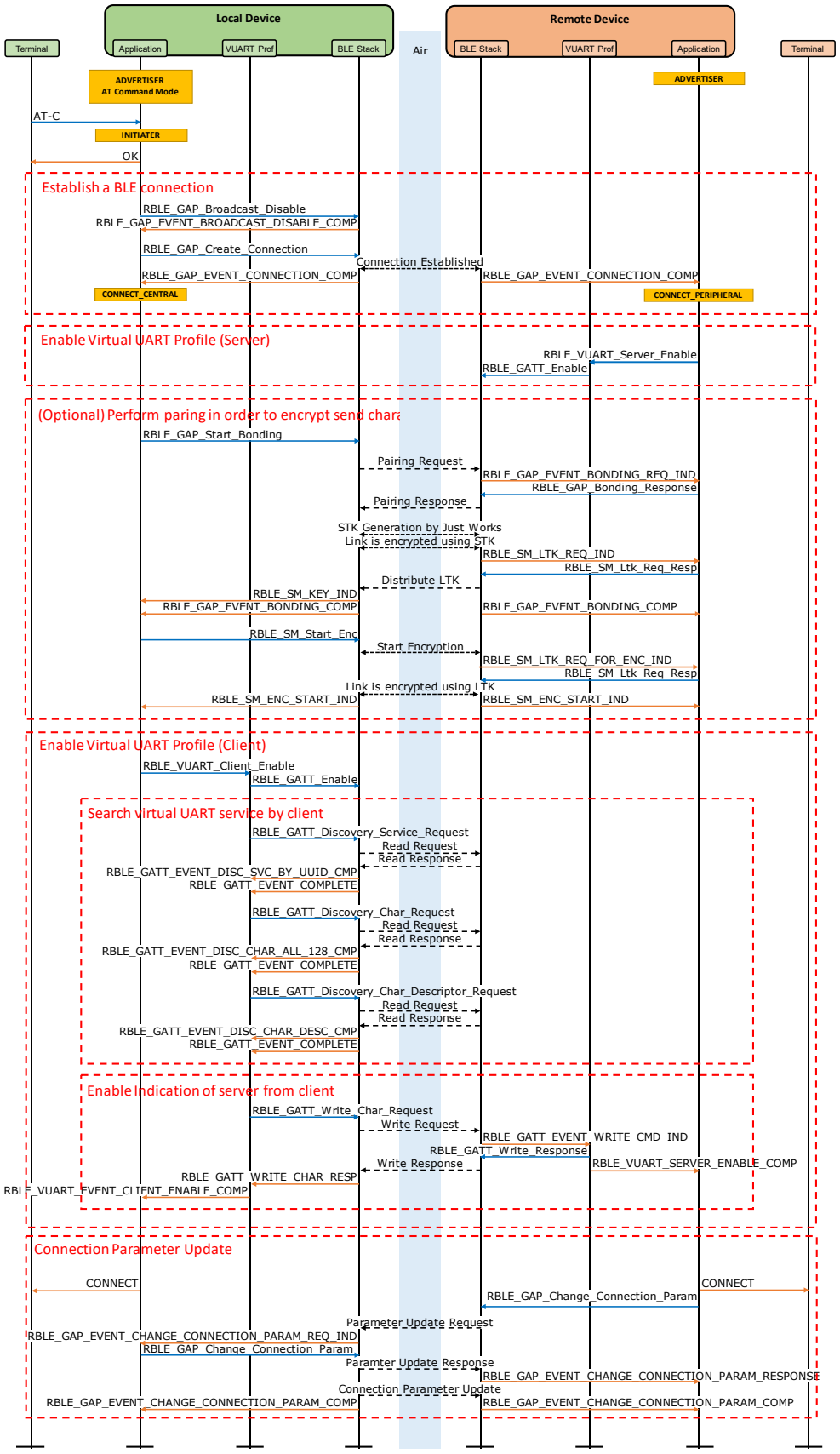


図 8-3 接続シーケンス

8.7.3 データ送受信シーケンス (Write Request/Indication)

図 8-4 にデータ送受信時のシーケンスを示します。

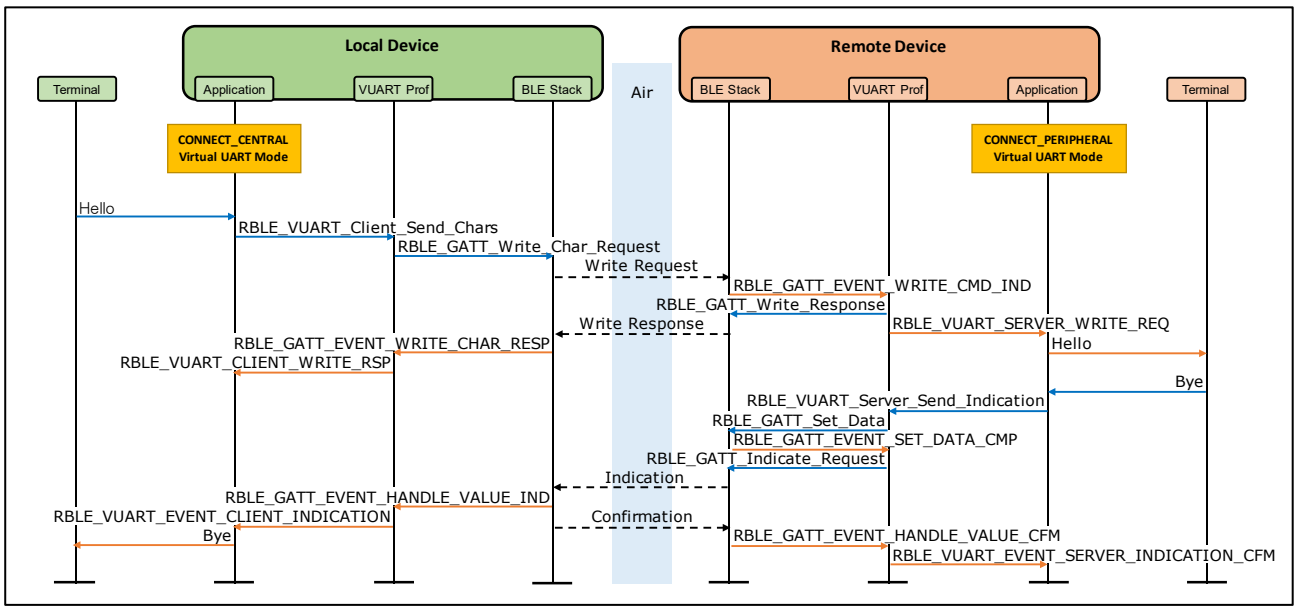


図 8-4 データ送受信シーケンス(Write Request/Indication)

8.7.4 データ送受信シーケンス (Write Command/Notification)

図 8-5 にデータ送受信時のシーケンスを示します。

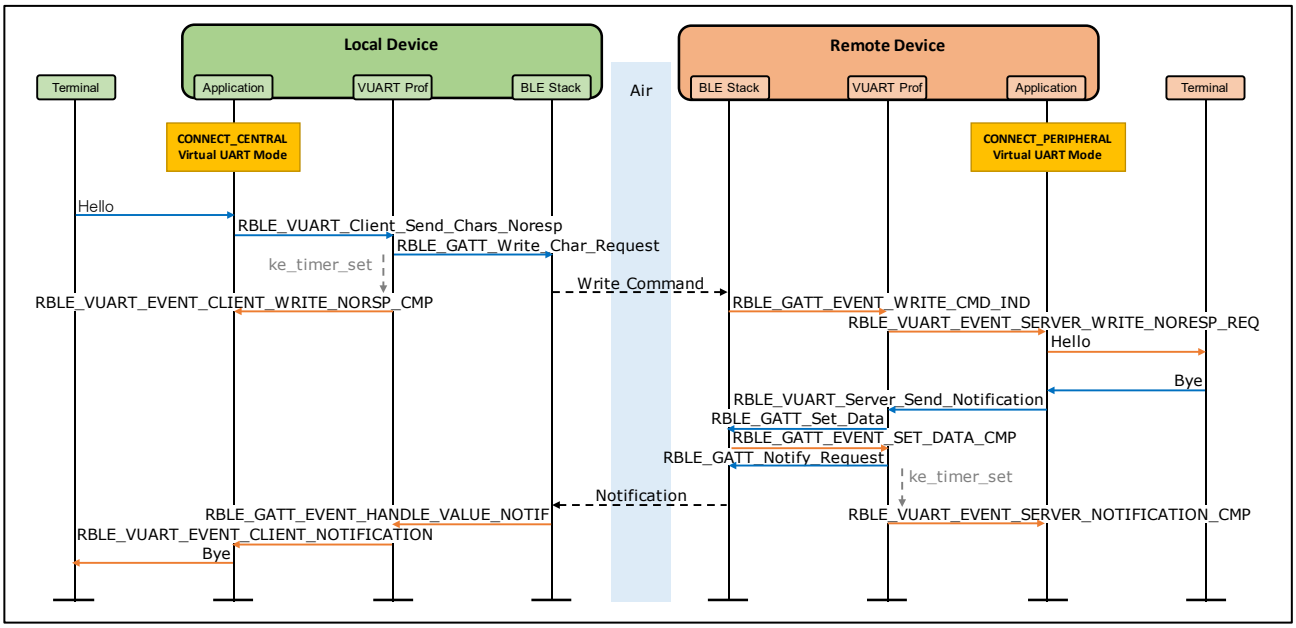


図 8-5 データ送受信シーケンス(Write Command/Notification)

8.7.5 切断シーケンス

図 8-6 に切断時のシーケンスを示します。

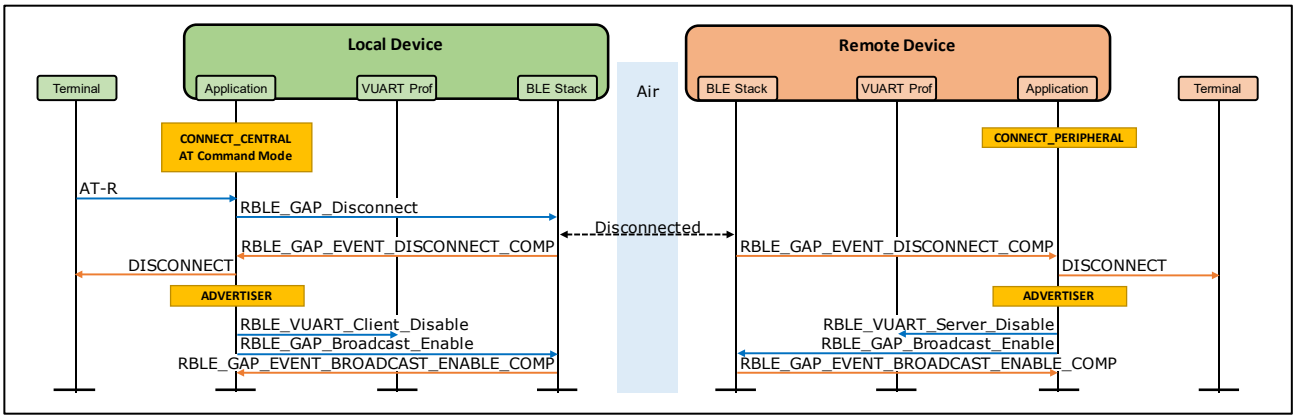


図 8-6 切断シーケンス

8.8 マクロ設定

アプリケーションの動作を設定するマクロを示します。

8.8.1 文字データ/バイナリデータ送受信設定

仮想 UART モードで送受信するデータ(文字データ/バイナリデータ)の種類を設定します。デフォルト設定は 1(文字データ)です。

※バイナリデータ送受信設定の場合、仮想 UART モードでのローカルエコーは無効になります。

r_vuart_app.h, line 47-49

```
47:   #define CFG_VUART_CHAR       (1)                 /* Switch between AT mode and VUART mode. */
48:                                                    /* (0): Binary mode                       */
49:                                                    /* (1): Character mode (default)          */
```

8.8.2 ローカルエコー設定

ローカルエコーの初期設定を「ローカルエコー無効」に設定します。デフォルト設定は 0(ローカルエコー有効)です。

※バイナリデータ送受信設定の場合、仮想 UART モードでのローカルエコーは無効になります。

r_vuart_app.h, line 51-53

```
51:   #define CFG_DISABLE_LOCAL_ECHO_BY_DEFAULT  (0)    /* Disable local echo                 */
52:                                                    /* (0): Enable local echo (default)   */
53:                                                    /* (1): Disable local echo            */
```

8.9 その他

8.9.1 アプリケーションに接続をするプログラムを作成する際の注意点

8.7.2 章に示すように、AT-C コマンドを実行した後、以下の処理が成功した場合に、接続の完了を示す CONNECT が表示されます。

- 接続の確立
- Bluetooth LE 通信の暗号化（ペアリング）
- クライアントによる仮想 UART サービスの検索
- Indication と Notification の有効化設定

アプリーションと接続を行うプログラムを作成する場合も、上記処理を実施してください。Bluetooth LE 通信の暗号化に関しては、省略することも可能です。

8.9.2 バイナリデータ送受信におけるモードを選択するディップスイッチの読み出し処理

アプリケーションでバイナリデータ送受信におけるモードの選択で使用する、ディップスイッチ状態の読み出し処理を以下に示します。

r_vuart_app.c, line 1916-1921

```
1916: BOOL read_dipsw1(void)
1917: {
1918:    /* TRUE (1) : AT command mode */
1919:    /* FALSE(0) : VUART mode    */
1920     return ((BOOL)read1_sfr(P1, 0));
1921: }
```

8.9.3 データ送受信におけるレスポンス有無を選択するディップスイッチの読み出し処理

アプリケーションでデータ送受信におけるレスポンス有無の選択に使用する、ディップスイッチ状態の読み出し処理を以下に示します。

r_vuart_app.c, line 1931-1936

```
1931: BOOL read_dipsw4(void)
1932: {
1933:    /* TRUE (1) : with response.   Indication, Write              */
1934:    /* FALSE(0) : without response. Notification, Write without response */
1935     return ((BOOL)read1_sfr(P0, 2));
1936  }
```

8.9.4 CFG_CON マクロ

CFG_CON マクロは最大同時接続台数の設定と同時に RL78/G1D の RF 部が持つヒープメモリも設定します。CFG_CON の設定 4(デフォルト)を推奨します。

8.9.5 ターミナルやホストマイコンからのデータ送信

PC のターミナルやホストマイコン等から UART で送信するデータのサイズは最大 20 バイトで送信してください。

8.9.5.1、8.9.5.2 にターミナルから送信されたデータを RL78/G1D が UART で受信し、Bluetooth LE 通信で送信する例を示します。本アプリケーションの設定は、ボー・レート 4800bps、データ長 20 バイト、コネクションインターバル 30ms です。

8.9.5.1 レスポンス有り通信時のデータ送信例

レスポンス有り通信(Write Request/Indication)として Indication の送信例を図 8-7 に示します。

Indication 通信では、Indication によるデータ送信と Confirmation によるレスポンス受信を一組とする通信で、Confirmation を受信するまで次の Indication でデータを送信することはできません。

最初の Bluetooth LE 通信でターミナルから受信できた分のデータ(2 バイト)を Indication で送信します。Confirmation を受信するまでは次の Indication を送信することができないため、ターミナルから受信した残りのデータ(18 バイト)の送信は保留されます。Confirmation が受信されると次の Bluetooth LE 通信タイミングで残りのデータ(18 バイト)を Indication で送信します。

この時、21 バイト以上のデータが送られてくると RL78/G1D のパケットサイズ(20 バイト)を超えてしまい不正なデータを送信する可能性があります。ターミナルからデータを送る際は、20 バイト単位で、コネクションインターバル x3 の時間より大きい間隔(この例では 100ms)で送ってください。

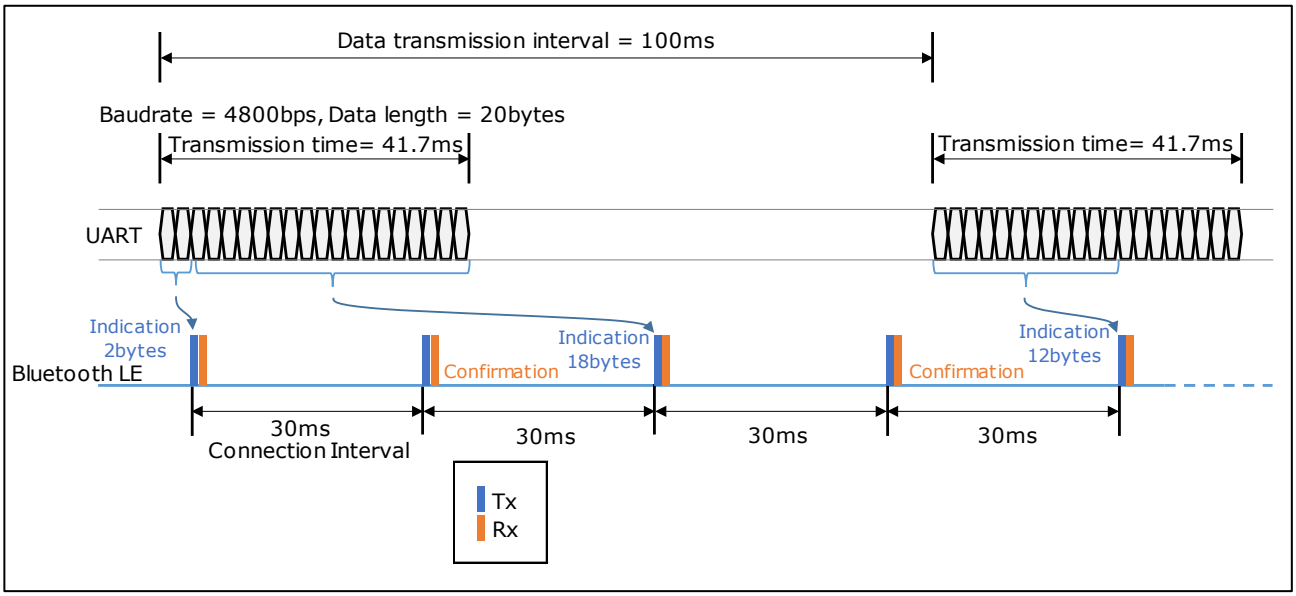


図 8-7 ターミナルからのデータ受信と Indication 通信例

8.9.5.2 レスポンス無し通信時のデータ送信例

レスポンス無し通信(Write Command/Notification)として Notification の送信例を図 8-8 に示します。

Notification 通信では、Bluetooth LE 通信のタイミングごとにデータを送信することができます。

最初の Bluetooth LE 通信でターミナルから受信できた分のデータ(2 バイト)を Notification で送信します。次の Bluetooth LE 通信タイミングまでにターミナルから受信したデータ(15 バイト)があると Notification で送信します。このように、Notification では送信するデータがあると Bluetooth LE 通信タイミングで送信することができます。

ターミナルから UART でデータを送る際は、レスポンス有り通信と同様に 20 バイト単位で送ってください。また隙間なく連続でデータを送ると UART でエラーが発生する可能性があります。データを送る間隔はコネクションインターバルよりも大きい間隔(この例では 50ms)で送ってください。

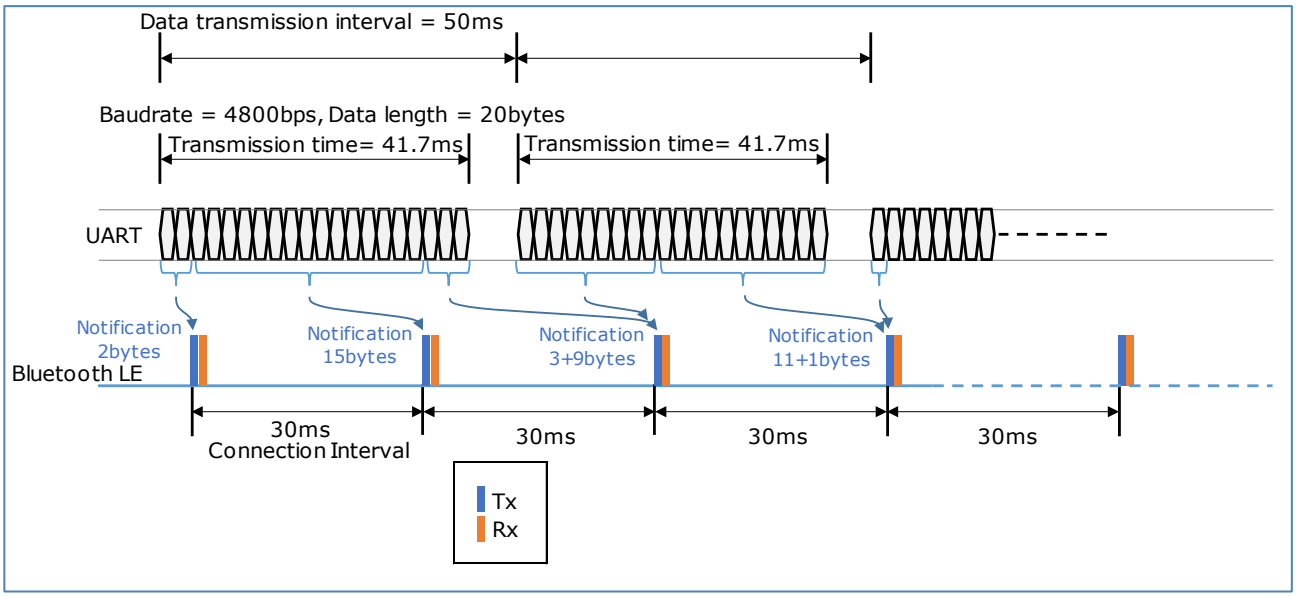


図 8-8 ターミナルからのデータ受信と Notification 通信例

9. 付録

9.1 ROM サイズ・RAM サイズ

仮想 UART アプリケーションを BLE プロトコルスタック V1.21 で使用した場合の ROM サイズ、RAM サイズを表 9-1、表 9-2 に示します。

表 9-1 ROM サイズ・RAM サイズ (文字データ送受信)

Compiler	ROM size	RAM size
RL78 Family C Compiler Package V1 V1.09.00	127,002	11,535
Renesas CA78K0R V1.72	155,849	11,609

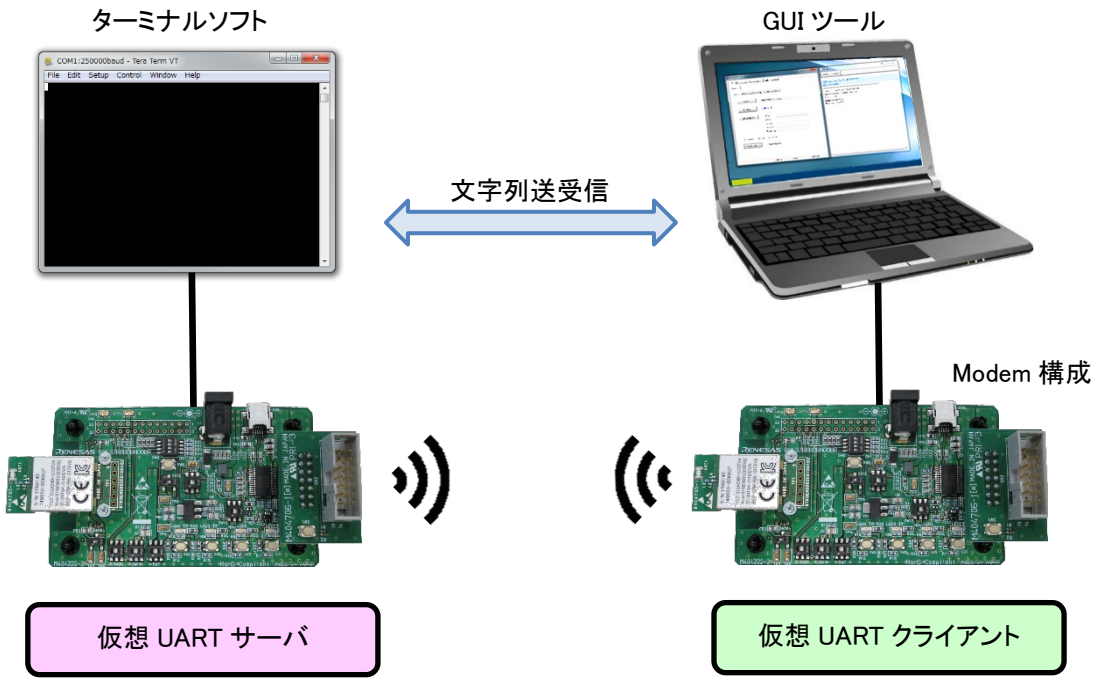
表 9-2 ROM サイズ・RAM サイズ (バイナリデータ送受信)

Compiler	ROM size	RAM size
RL78 Family C Compiler Package V1 V1.09.00	126,684	11,535
Renesas CA78K0R V1.72	155,676	11,609

9.2　GUI ツールを使用した動作確認方法

本節では、GUI ツール（R01AN2469）を使用した仮想 UART の動作確認方法について説明します。

図 9-1 に GUI ツールを使用した動作確認構成図を示します。本アプリケーションが仮想 UART サーバ、GUI ツールが仮想 UART クライアントとして動作し、お互いに文字列を送受信することが可能です。



【注】　動作確認のためには GUI ツール Version1.12（R01AN2469XX0112）以降が必要です。
別途入手してください。

図 9-1 GUI ツールを使用した動作確認構成

以降では、仮想 UART アプリケーションを書き込んだ評価ボードとターミナルソフトの組み合わせを「仮想 UART サーバ」、Modem 構成の Hex ファイルを書き込んだ評価ボードと GUI ツールの組み合わせを「仮想 UART クライアント」として記載します。

9.2.1　準備

- 仮想 UART サーバ
  本アプリケーションノートの下記節の手順に従って、RL78/G1D 評価ボードに仮想 UART アプリケーションを書き込み、PC 上でターミナルソフトを起動します。
  - 7.2 ビルド手順
  - 7.3 実行環境の準備
    ここでは、文字データ送受信（RL78_G1D_CCE(VUART_CHAR).hex）を使用します。

- 仮想 UART クライアント
  BLE プロトコルスタックのパッケージに含まれる任意のビルド環境の Modem 構成の Hex ファイルを書き込み、GUI ツールを起動します。

【注】 1. GATT API を使用して仮想 UART サーバのサービスにアクセスするため、書き込む Hex ファイルのプロファイル種別は問いません。
2. GUI ツールの起動方法については、「Bluetooth Low Energy プロトコルスタック GUI ツール」（01AN2469）の「6. 起動方法」をご確認ください。

9.2.2　動作確認

仮想 UART サーバ、仮想 UART クライアントをそれぞれ下記手順で操作することで文字列の送受信を行うことができます。

1. 発見可能モード（仮想 UART サーバ）
   仮想 UART サーバとして動作する RL78/G1D 評価ボードの RESET スイッチ(SW5)を押下します。
   リセット後、仮想 UART アプリケーションは自動的に発見可能モードに遷移し、Advertising を開始します。

2. デバイス検索（仮想 UART クライアント）
   GUI ツールを操作し、発見可能モードのデバイスを検索します。
   (1) [GAP]タブの中の[Scanning]タブをアクティブにします。
   (2) Discovery グループで、“General Discovery”を選択します。
   (3) [Discover]ボタンを押下します。

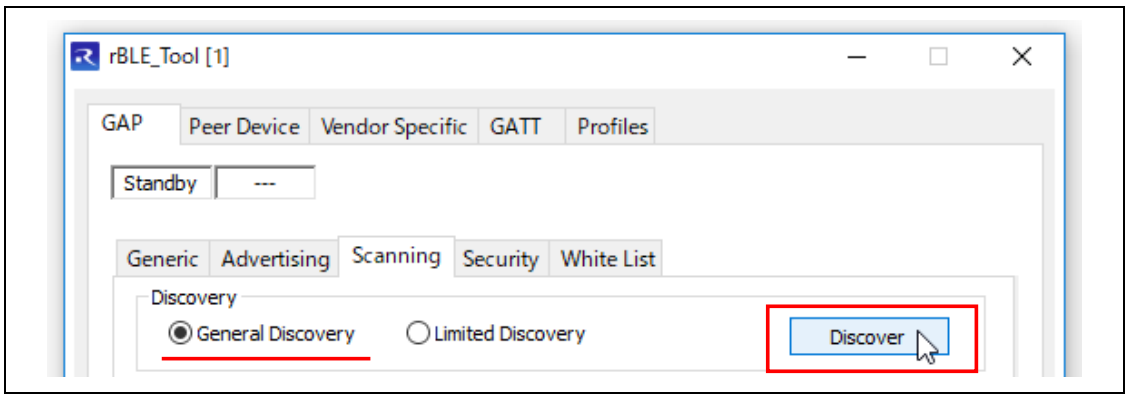


図 9-2 デバイス検索

(4) 発見可能モードのデバイスが、受信 Advertising データリスト表示部に表示されます。

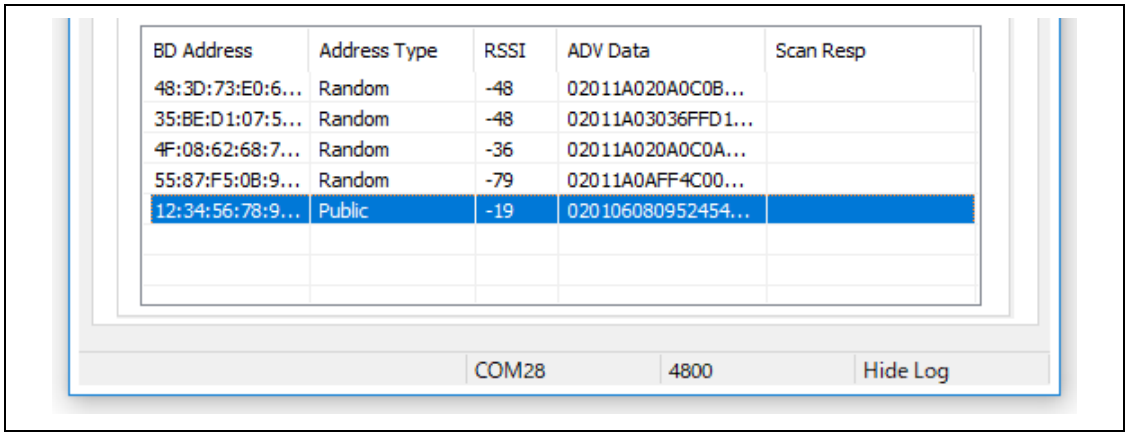


図 9-3 デバイス検索結果

3. 接続（仮想 UART クライアント）
GUI ツールを操作し、仮想 UART サーバのデバイスに接続を開始します。
(1) ログダイアログで RBLE_GAP_EVENT_DEVICE_SEARCH_COMP イベントが発生していることを確認します。

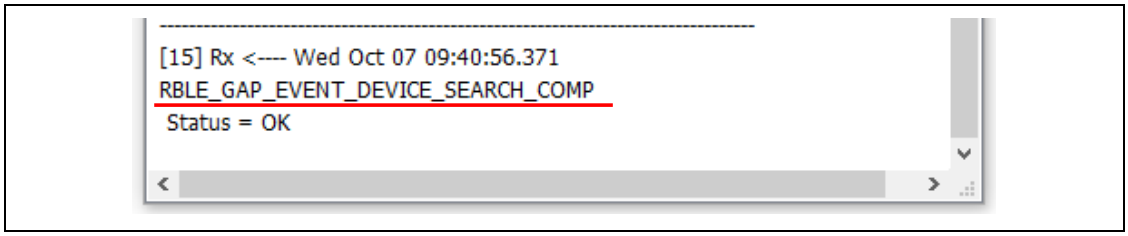


図 9-4 RBLE_GAP_EVENT_DEVICE_SEARCH_COMP イベントの確認

(2) [Scanning]タブの受信 Advertising データリスト表示部で、接続対象のデバイスをダブルクリックします。

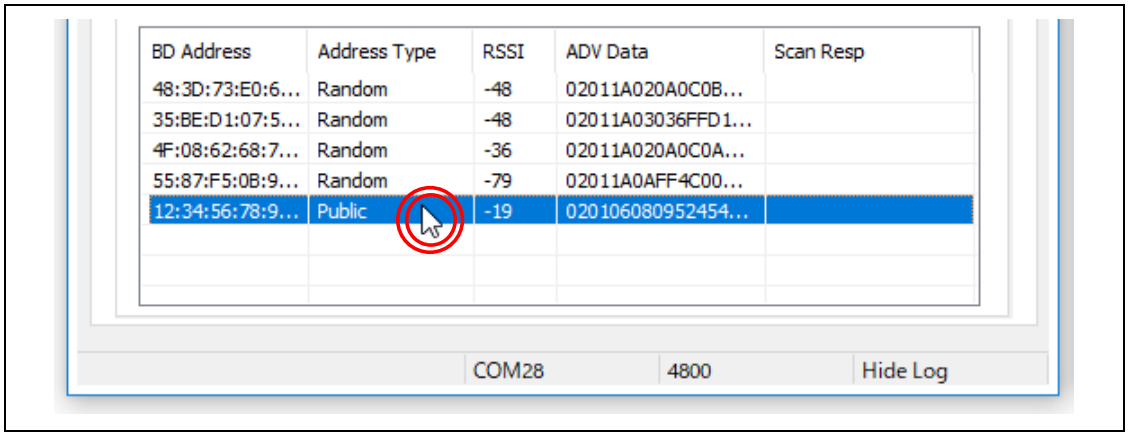


図 9-5 デバイス選択

Tips: 受信 Advertising データリスト表示部で任意の行を[Ctrl]キーを押しながらダブルクリックすると Advertising データ解析ダイアログが表示されます。
≪Complete List of 128-bit Service UUIDs≫に“Renesas Virtual UART Service”が含まれるデバイスが仮想 UART サーバとして動作するデバイスです。

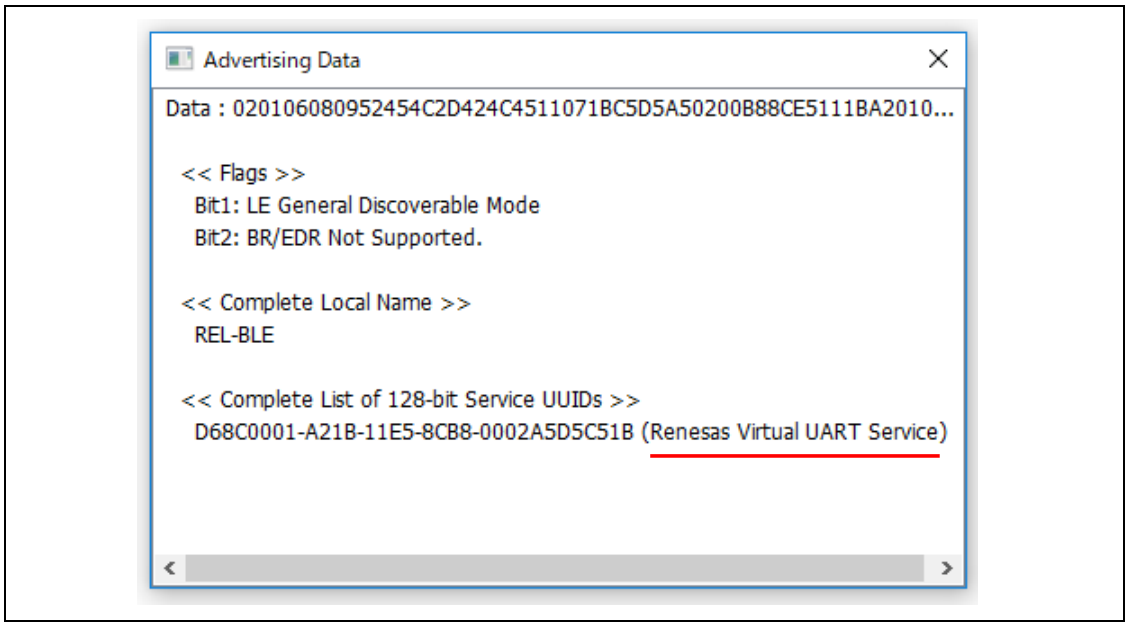


図 9-6 Advertising データ解析表示

(3) [Peer Device]タブの中の[Connection]タブをアクティブにします。
この時、[Peer Device]タブ上部の“Peer Addr”欄に接続対象のデバイスが入力されていることを確認してください。

(4) [Connect]ボタンを押下し、仮想 UART サーバへの接続を開始します。

図 9-7 接続

(5) 接続が完了すると、[Peer Device]タブ上部の状態表示部が“Connected”に変化します。

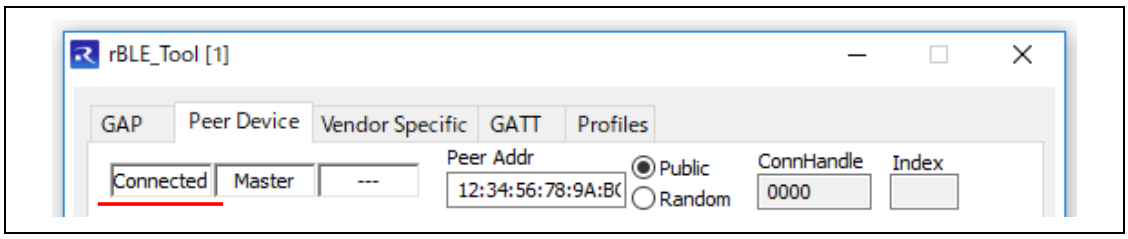


図 9-8 接続完了

4. サービス検索（仮想 UART クライアント）
GUI ツールを操作し、仮想 UART サーバの情報を取得します。
- サービス検索
— 仮想 UART サーバの保有する全てのサービスを検索します。
(1) [GATT]タブ→[Client]タブ→[Service Discovery]タブをアクティブにします。
(2) Discovery Type を“Discover All Primary Services”に設定し、[Discover]ボタンを押下します。

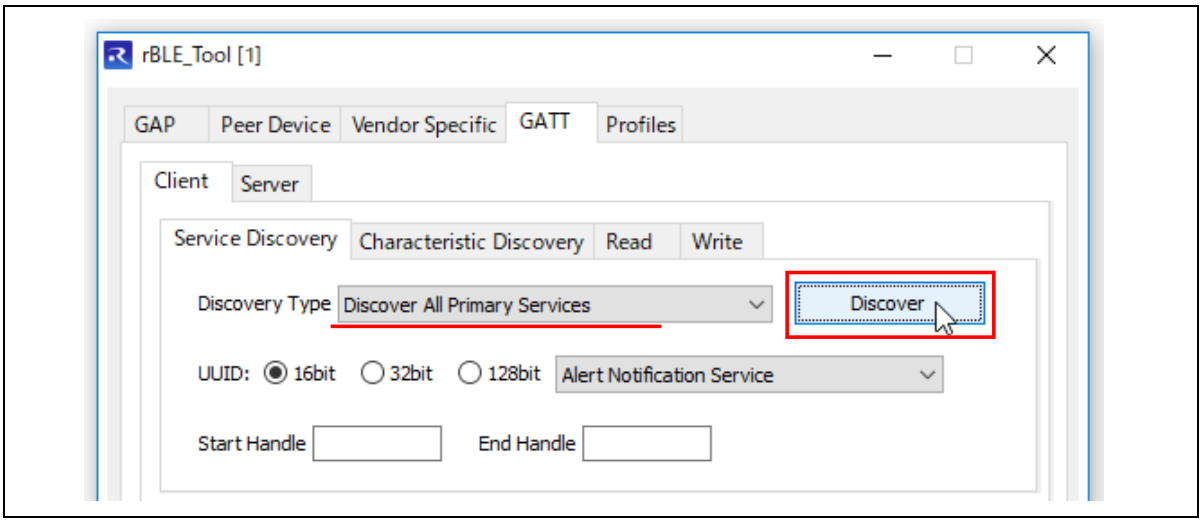


図 9-9 サービス検索

(3) 取得したサービス情報が“Remote GATT Database”部に表示されます。

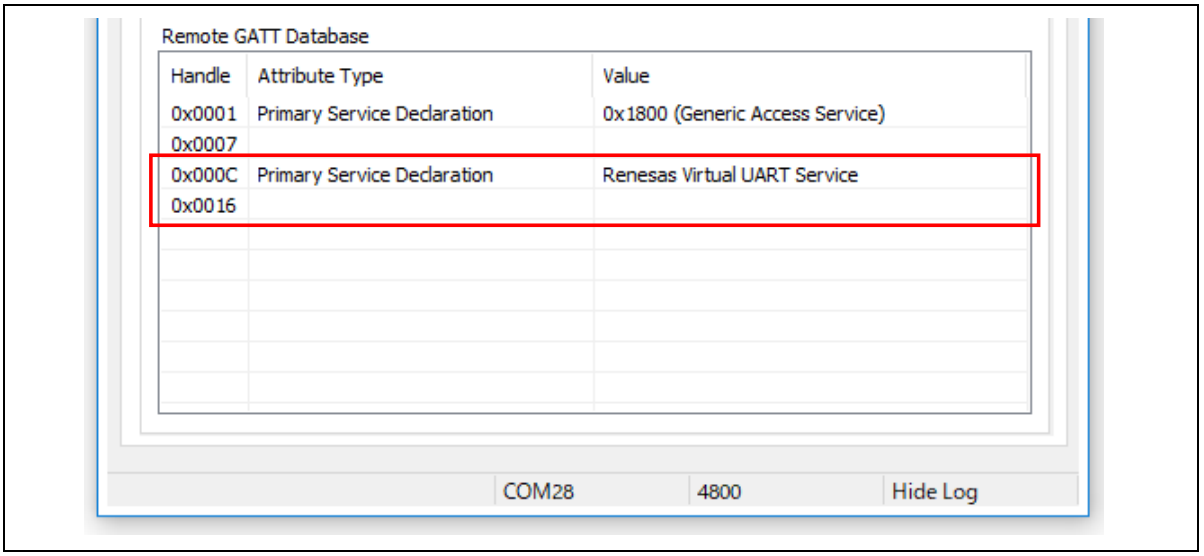


図 9-10 サービス検索結果

- キャラクタリスティック検索

— 仮想 UART サーバの保有するサービスのキャラクタリスティックを検索します。

(1) [GATT]タブ→[Client]タブ→[Characteristic Discovery]タブをアクティブにします。

(2) Discovery Type を“Discover Characteristics of a Service”に設定し、[Discover]ボタンを押下します。

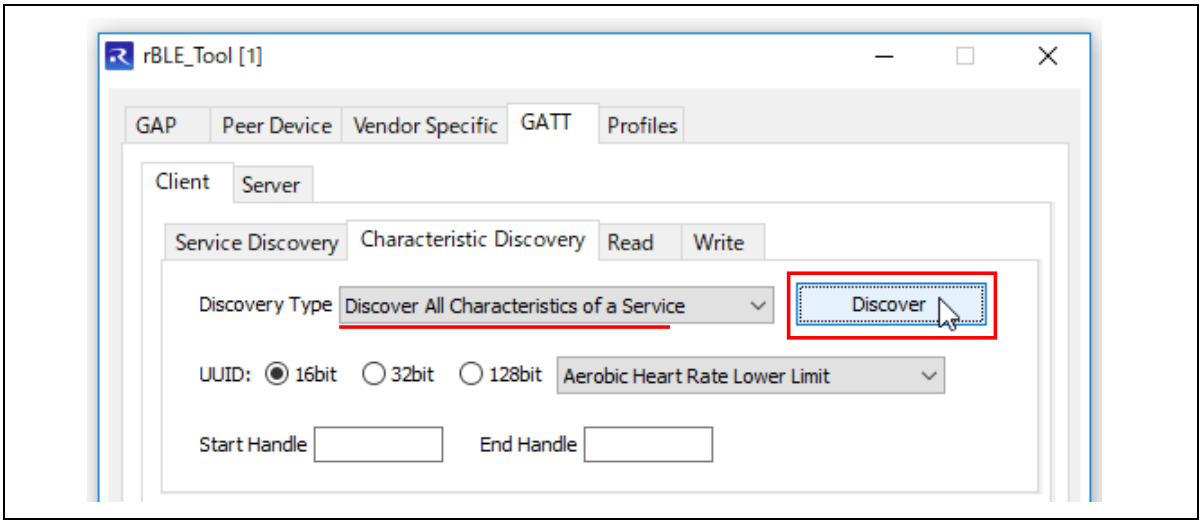


図 9-11 キャラクタリスティック検索

(3) 取得したキャラクタリスティック情報が“Remote GATT Database”部に表示されます。

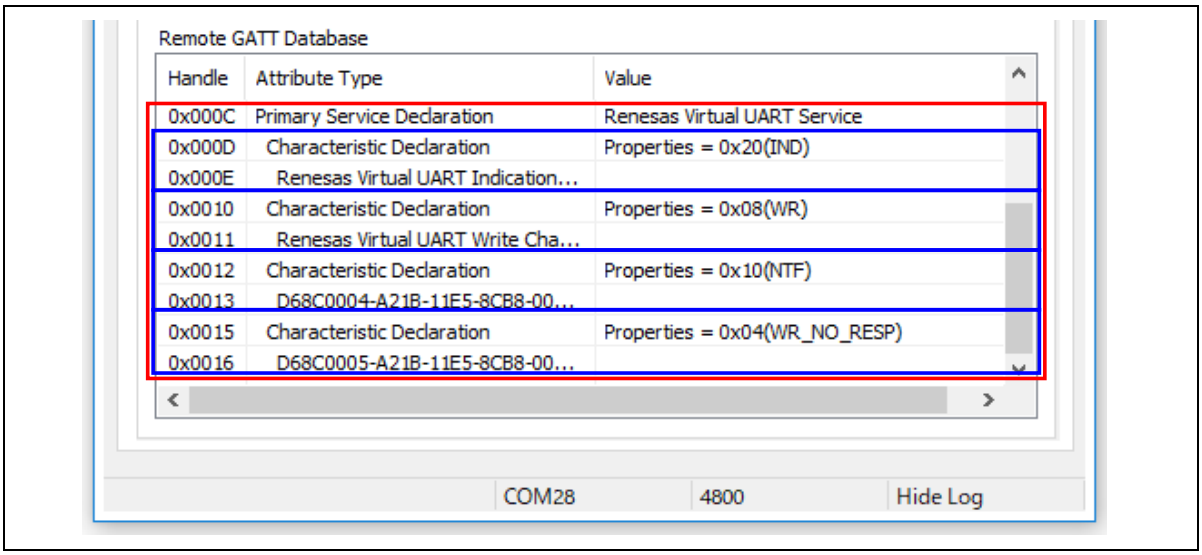


図 9-12 キャラクタリスティック検索結果

- キャラクタリスティックディスクリプタ検索

— 仮想 UART サーバの保有するキャラクタリスティックのディスクリプタを検索します。

(1) [GATT]タブ→[Client]タブ→[Characteristic Discovery]タブをアクティブにします。

(2) Discovery Type を“Discover All Characteristic Descriptors”に設定し、[Discover]ボタンを押下します。

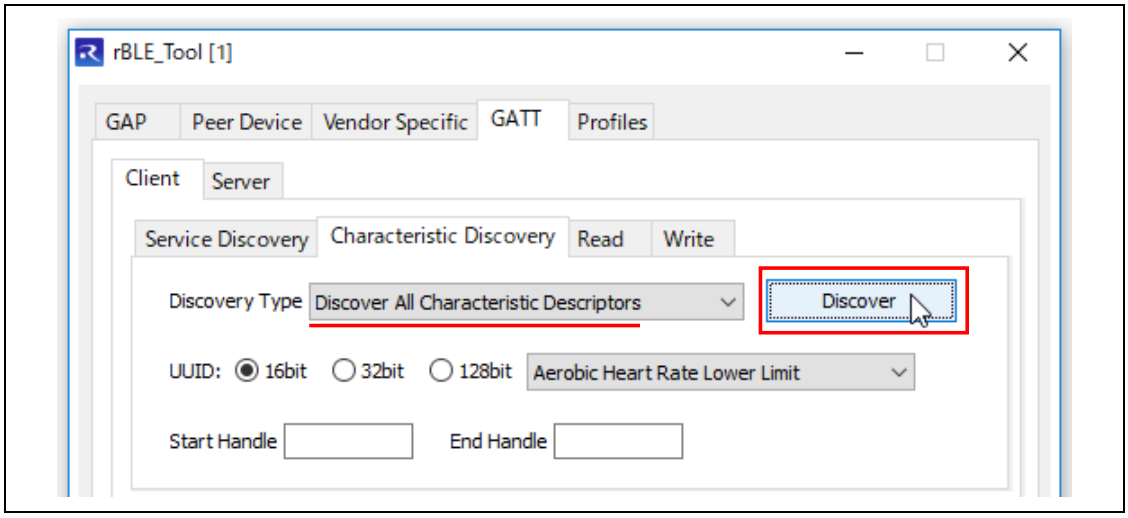


図 9-13 ディスクリプタ検索

(3) 取得したキャラクタリスティックディスクリプタ情報が“Remote GATT Database”部に表示されます。

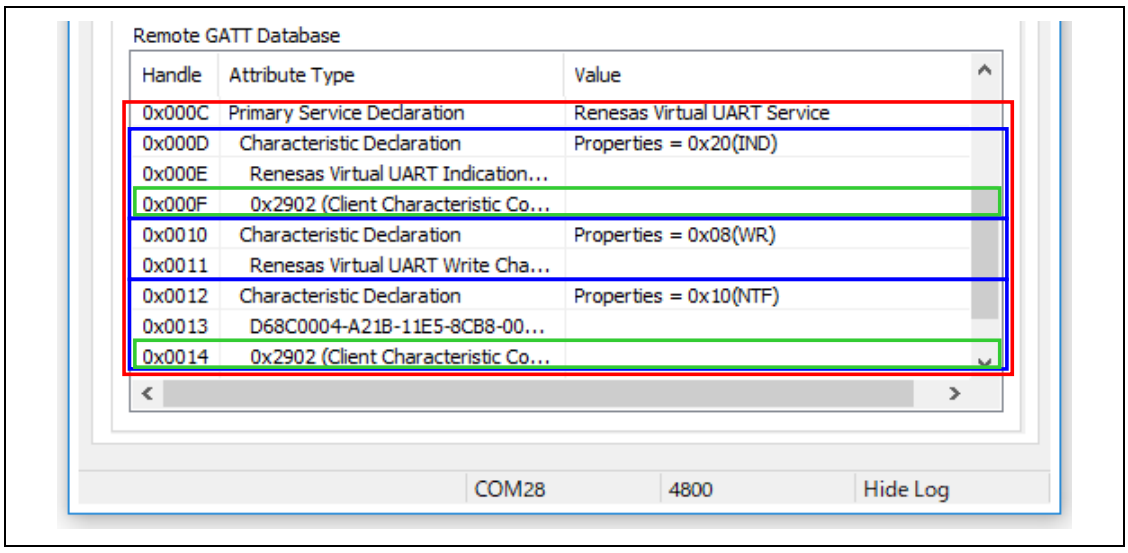


図 9-14 ディスクリプタ検索結果

5. Indication と Notification の有効化（仮想 UART クライアント）
   GUI ツールを操作し、仮想 UART サーバからの文字列送信（Indication, Notification）を有効にします。
   (1) [GATT]タブ→[Client]タブ→[Write]タブをアクティブにします。
   (2) Write Type を“Write Characteristic Descriptors”に設定します。
   (3) “Remote GATT Database”部において、“Renesas Virtual UART Indication Characteristic”の Client Characteristic Configuration Descriptor をダブルクリックします。
   ダブルクリックをすると、[Write]タブのの“Handle”欄にハンドルが入力されます。
   (4) Write Data に“Indication Enable”を意味する“0002”を入力します。
   (5) [Write]ボタンを押下し、仮想 UART サーバの Client Characteristic Configuration Descriptor に Write します。

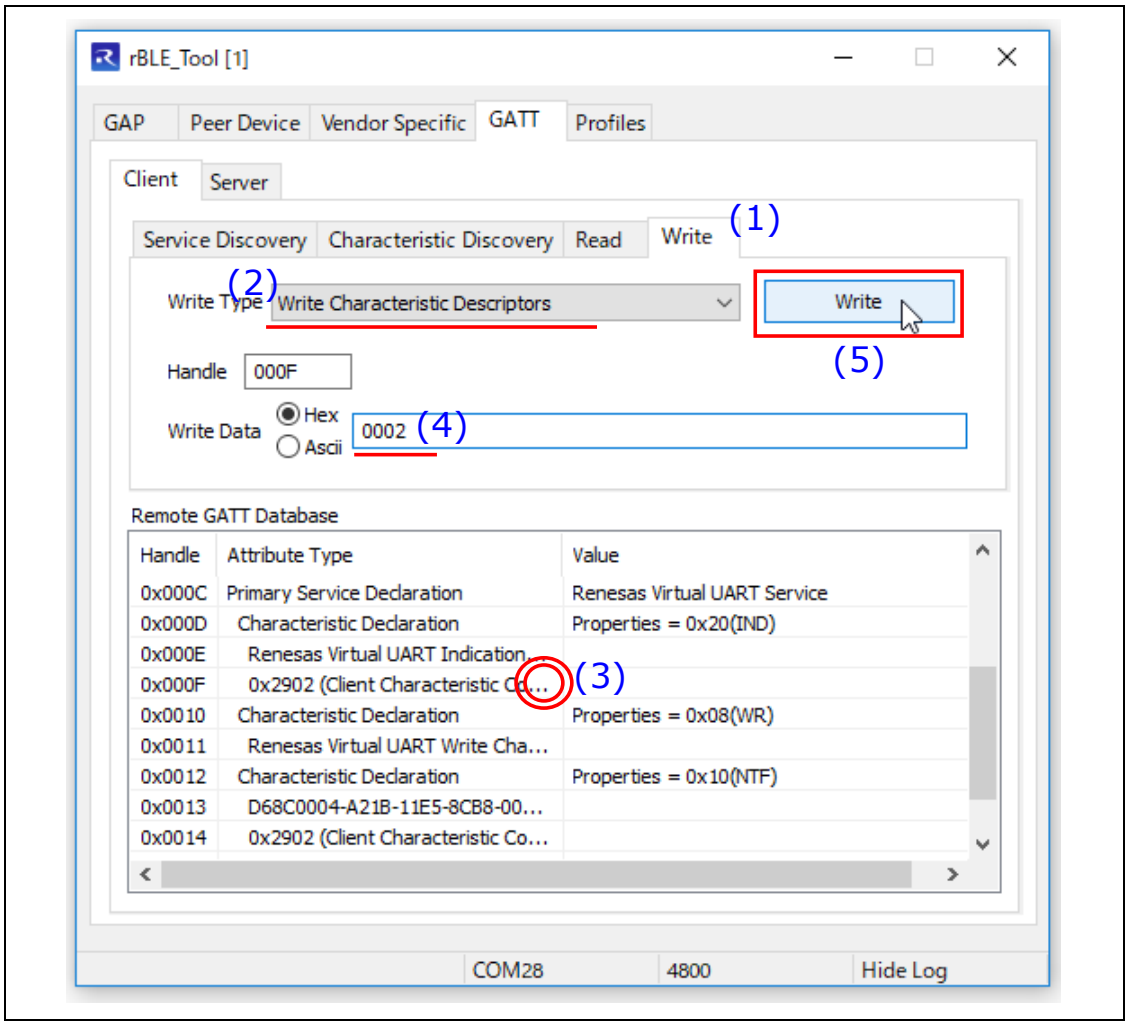


図 9-15 Indication の有効化

(6) “Remote GATT Database”部において、“D68C0004-A21B-11E5-8CB8-0002A5D5C51B”(Renesas Virtual UART Notification Characteristic)の Client Characteristic Configuration Descriptor をダブルクリックします。
ダブルクリックをすると、[Write]タブのの“Handle”欄にハンドルが入力されます。

(7) Write Data に“Notification Enable”を意味する“0001”を入力します。

(8) [Write]ボタンを押下し、仮想 UART サーバの Client Characteristic Configuration Descriptor に Write します。

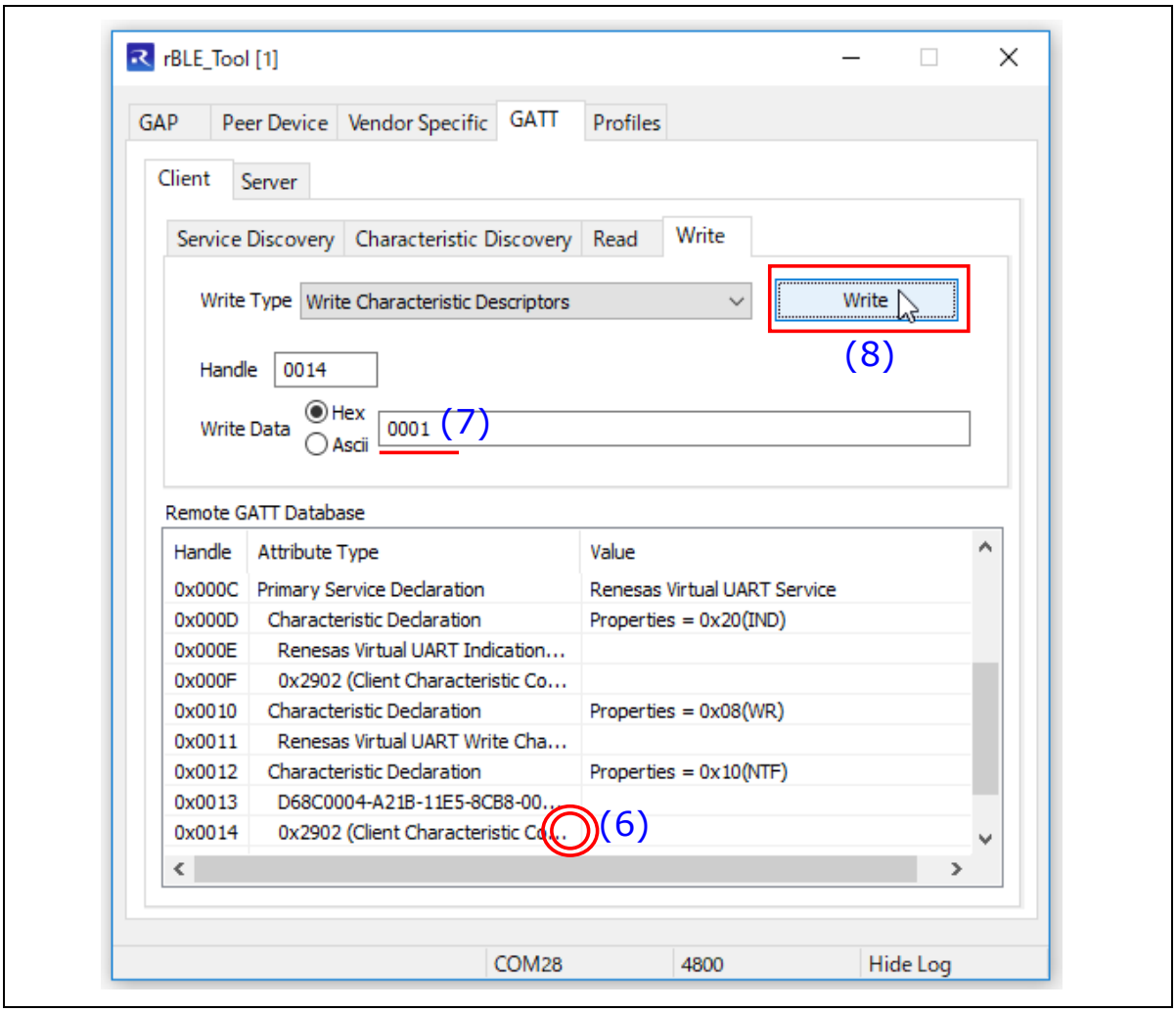


図 9-16 Notification の有効化

(9) Parameter Update Request のダイアログが表示されるので"Accept"を押します。

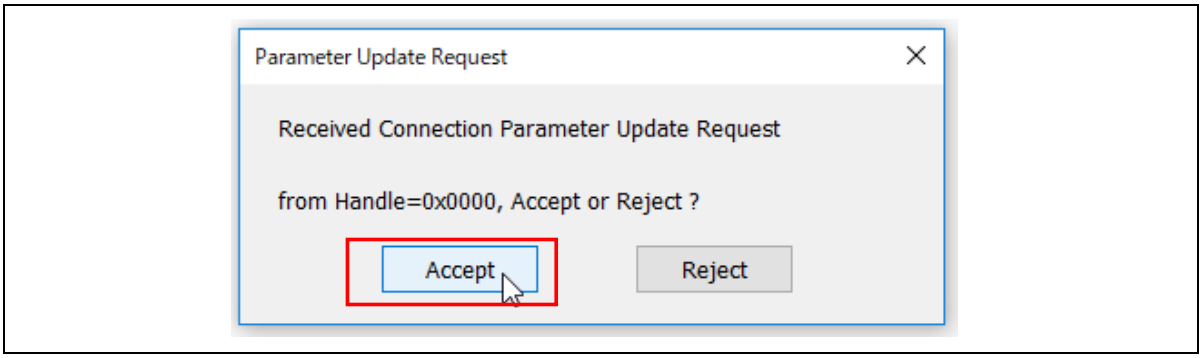


図 9-17 Parameter Update Request の受け入れ

(10) 仮想 UART サーバから Write に対する応答を受信するとコンソールウィンドウが表示されます。

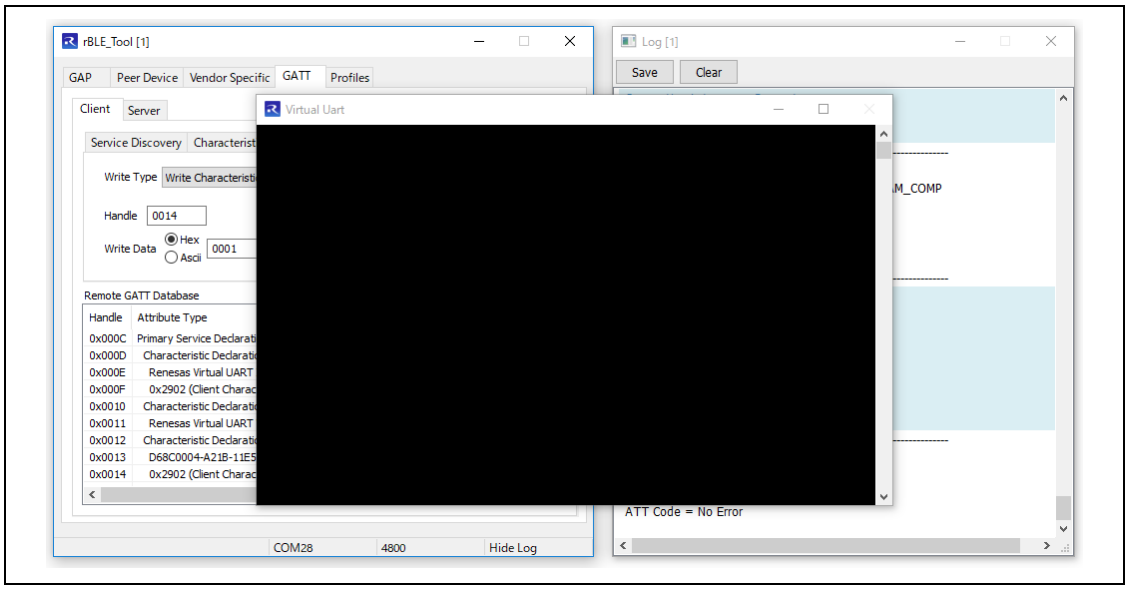


図 9-18 コンソールウィンドウ表示

6. 文字列の送受信（仮想 UART サーバ／仮想 UART クライアント）
   - 仮想 UART サーバからの送信
     (1) ターミナルソフト上で、エスケープキーをタイプし仮想 UART モードに遷移します。
     (2) ターミナルソフト上で、任意の文字列（例：“Hello!”）を入力します。
     (3) 入力した文字列が仮想 UART クライアントのコンソールウィンドウに黄色い文字で表示されます。

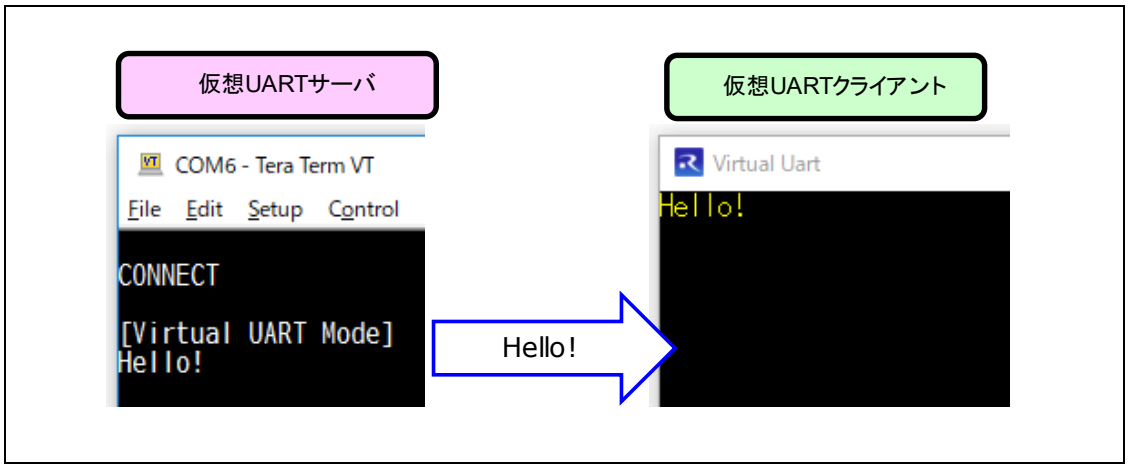


図 9-19 文字列送信（サーバ→クライアント）

   - 仮想 UART クライアントからの送信
     (1) コンソールウィンドウ上で、任意の文字列（例：“Bye”）を入力します。
     (2) 入力した文字列が仮想 UART サーバのターミナルソフトに表示されます。

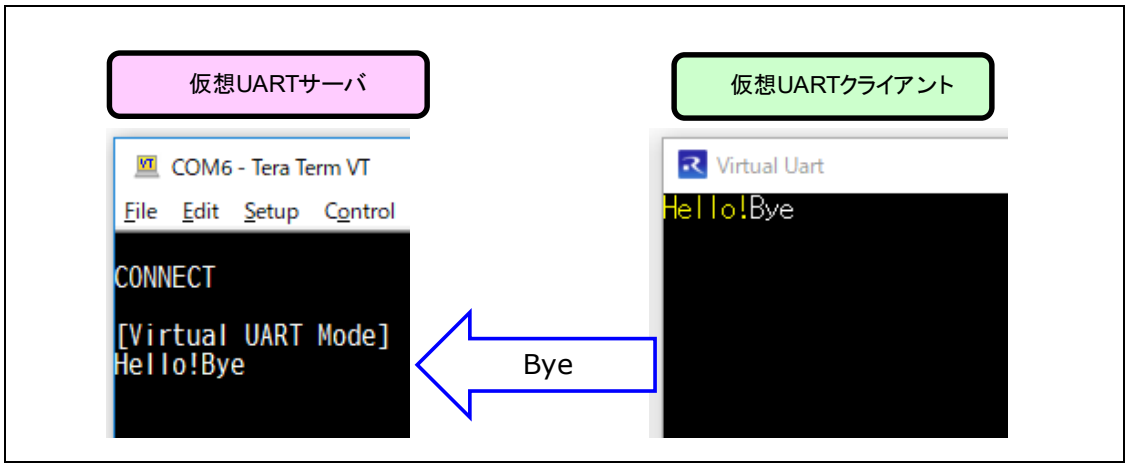


図 9-20 文字列送信（クライアント→サーバ）

7. 切断（仮想 UART サーバ／仮想 UART クライアント）

- 仮想 UART サーバから切断する場合
  (1) ターミナルソフト上で、エスケープキーをタイプし簡易 AT コマンドモードに遷移します。
  (2) ターミナルソフト上で、切断コマンド“AT-R”を入力し、仮想 UART クライアントとの接続を切断します。
  (3) 切断が完了すると、ターミナルソフト上に“DISCONNECT”が表示されます。

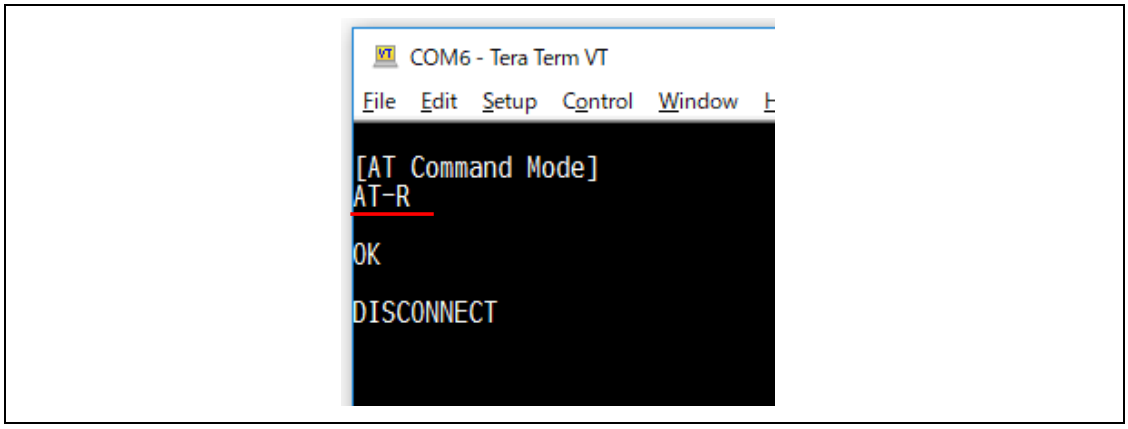


図 9-21 仮想 UART サーバからの切断

- 仮想 UART クライアントから切断する場合
  (1) [Peer Device]タブの中の[Connection]タブをアクティブにします。
  (2) [Disconnect]ボタンを押下し、仮想 UART サーバとの接続を切断します。
  (3) 切断が完了すると、[Peer Device]タブ上部の状態表示部が“Standby”に変化します。

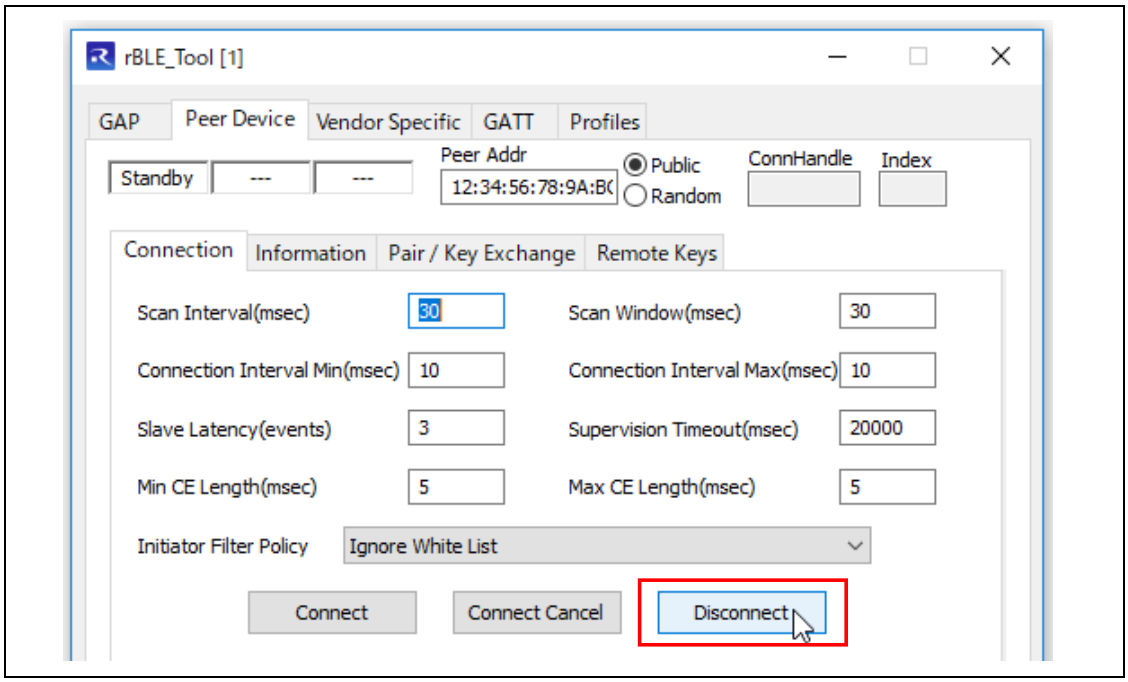


図 9-22 仮想 UART クライアントからの切断

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.02.24	-	初版発行
1.10	2016.10.07	7	2.2 ファイル構成 : 各開発環境およびファームウェアのファイル構成の追加
		11	4 簡易 AT コマンドモード : AT-CI=<con_intv>、AT-CI?、ATE0、ATE1 の追加
		22	7 ビルドと動作確認 : 各開発環境の準備・ビルド方法の追加
		26	7.4 使用例: デバイスアドレスの設定方法の追記
		37	8.3 接続動作 : Connection Interval の設定値の変更
		49	8.7.2 接続シーケンス : Connection Interval の設定シーケンスの追加、BLE プロトコルスタックの暗号化シーケンスへの対応
		57	9.2GUI ツールを使用した動作確認方法 : 新規追加
1.11	2016.11.22	-	ソースコードの不具合修正に伴う改版（文章の修正なし）
1.12	2017.10.20	24	7.2.3 BLE プロトコルスタック/EEPROM エミュレーションライブラリ : EEPROM エミュレーションライブラリのファイル名を更新
1.13	2019.07.12	-	AT-AS コマンドの不具合修正に伴う改版（文章の修正なし）
1.20	2020.10.23	-	バイナリデータの送受信と、レスポンス無し通信(Write Command, Notification)に対応したことに伴う改版。
		5	1. 概要 : バイナリデータの送受信と、レスポンス無し通信について概要を追加。
		7	2.2 ファイル構成 : バイナリデータ送受信の HEX ファイルを追加。Tera Term のマクロファイルを追加。
		9	3. 機能 : バイナリデータの送受信と、レスポンス無し通信の機能説明を追加。
		17	5. 仮想 UART モード : バイナリデータの送受信と、レスポンス無し通信の説明を追加。
		21	6.2 アドバタイジング間隔の変更 : アドバタイジング間隔を変更するコードを掲載。
		22	7. ビルドと動作確認 : ビルド環境を更新。バイナリデータの送受信と、レスポンス無し通信のビルド手順や使用例を追加。
		35	8. 実装の詳細 : バイナリデータの送受信と、レスポンス無し通信のプロファイル、関数、イベントを追加。
		56	9. 付録 : ROM サイズ、RAM サイズを更新。レスポンス無し通信に対応したことによる GUI ツール操作方法を更新。
1.20	2022.01.31	-	Bluetooth Low Energy プロトコルスタックでの IAR サポート終了に伴う修正。

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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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.4.0-1　2017.11)

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。