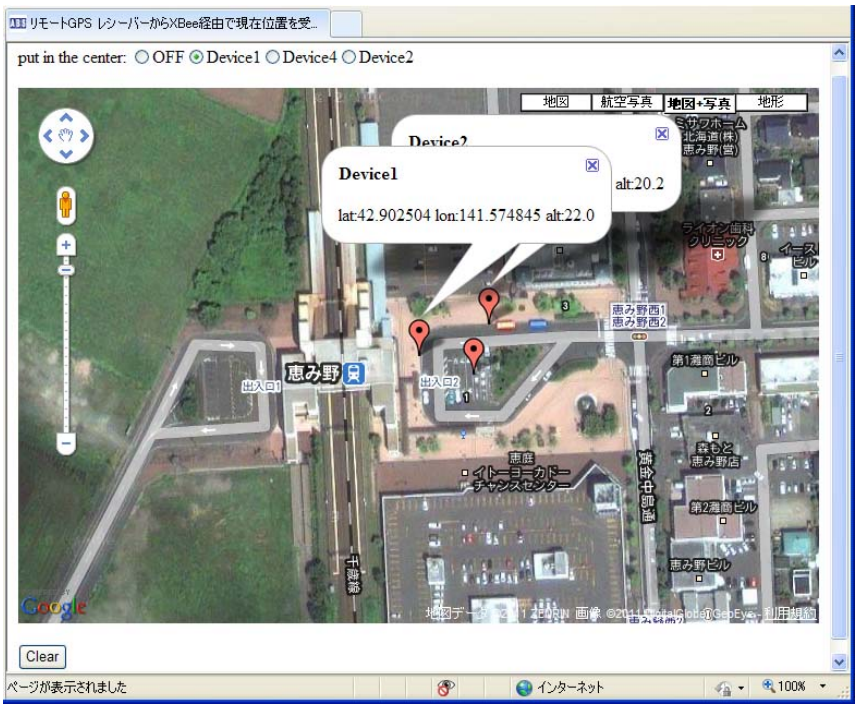
[APNOTE11]

リモートGPS レシーバからXBee経由で現在位置を受信して GoogleMap に表示

ABS-9000 DeviceServer
APNOTE11 Rev A.1.0
2011/01/16



オールブルーシステム (All Blue System)
ウェブページ: www.allbluesystem.com
コンタクト:contact@allbluesystem.com

1 イントロダクション 3

1.1 システム全体構成図 3
1.2 リモートデバイス構成図 5
1.3 リモート GPS イベントデータの処理フロー 5

2 必要な機材・リソース 6

2.1 サーバーPC 6
2.2 リモートデバイス 6

3 セットアップ 7

3.1 XBee デバイス初期設定(サーバー側・リモートデバイス共通) 7
3.2 XBee デバイスをDeviceServer に接続 9
3.3 リモート側デバイス接続 10
3.4 マスター登録とXBee 詳細設定 10
3.5 リモートデバイスTDCP設定 13
3.5.1 XBEE_TDCP_MODE8_CONFスクリプト作成 14
3.5.2 XBee デバイス管理プログラムから TDCP コマンド実行 16
3.6 GPSレシーバ(モジュール)接続 17

4 スクリプト作成 18

4.1 XBEE_TDCP_DATAスクリプト作成 18
4.2 GPS_MAP/GPS_RECEIVEスクリプト作成 21
4.3 GPS_MAP/GPS_REPORTスクリプト作成 22
4.4 GPS_MAP/GPS_CLEARスクリプト作成 23
4.5 PERIODIC_TIMERスクリプト作成 23

5 Webクライアントプログラム設定 24

5.1 DeviceServerのWebProxy セットアップ 25
5.2 GoogleMap 表示用Webページ設定 25

6 アプリケーションを起動する 32

7 このドキュメントについて 34

7.1 著作権および登録商標 34
7.2 連絡先 34
7.3 このドキュメントの使用について 35

8 更新履歴 35

1　　イントロダクション

GPS レシーバと接続された複数のリモートデバイスから、座標データを定期的に無線でサーバーに送信して、クライアントPC で実行する Webブラウザの GoogleMap(Google Maps[1] API で表示した地図) に、リモートデバイスの現在位置をリアルタイムでマーカー表示するシステムについて説明します。

各リモートデバイスでは、GPSレシーバ(GPSモジュール) から NMEA-0183 フォーマットで定期的に座標データが出力されて、マイクロコントローラ(ATmega644P/Atmega1284P) に送信されます。このマイクロコントローラから GPS イベントデータが作成されてサーバーに送信されます。サーバーとリモートデバイス間の通信には XBee[2] デバイスを使用します。イベントデータやリモートデバイスに対するリモートコマンド(コンフィギュレーション変更、ポート操作、マニュアルサンプリング..)などはすべて XBee モジュールのデータパケット中に格納されて送受信されます。マイクロコントローラ上で動作するプログラムは、オールブルーシステム が提供している “TDCP (Tiny Device Control Program)” を使用しています。TDCP は DeviceServer のライセンスと共に使用する場合にはフリーで製品に搭載できます。(*1 参照)

XBee デバイスは、Digi International Inc. 社製の IEEE 802.15.4 RF モジュールを使用します。サーバー側とリモートデバイスの両方で使用しています。

複数のリモートデバイスから送信された GPS 座標情報を含むイベントデータは、定期的に無線(XBee経由)で サーバーPC (DeviceServer) に送られて内部の共有データに格納されます。

クライアントPC の Web ブラウザからはいつでも javascript で記述されたページをアクセスして、現在のリモートデバイス位置を GoogleMap 上にマーカーで表示できます。ページ中に記述された javascript では DeviceServer にアクセスしてリモートデバイスリストや各リモートデバイスの現在の座標情報を取得しています。

(*1) TDCP プログラムについて

リモートデバイス中のマイクロコントローラ上で動作させるプログラムはオールブルーシステムが提供している “TDCP (Tiny Device Control Program)” を搭載しています。DeviceServer のライセンスと共に使用する場合にはフリーで製品に搭載できます。TDCP 詳細とリモートデバイス中に使用したベースボードのマニュアルがフリーで公開しています。下記のWebページを参照して下さい。

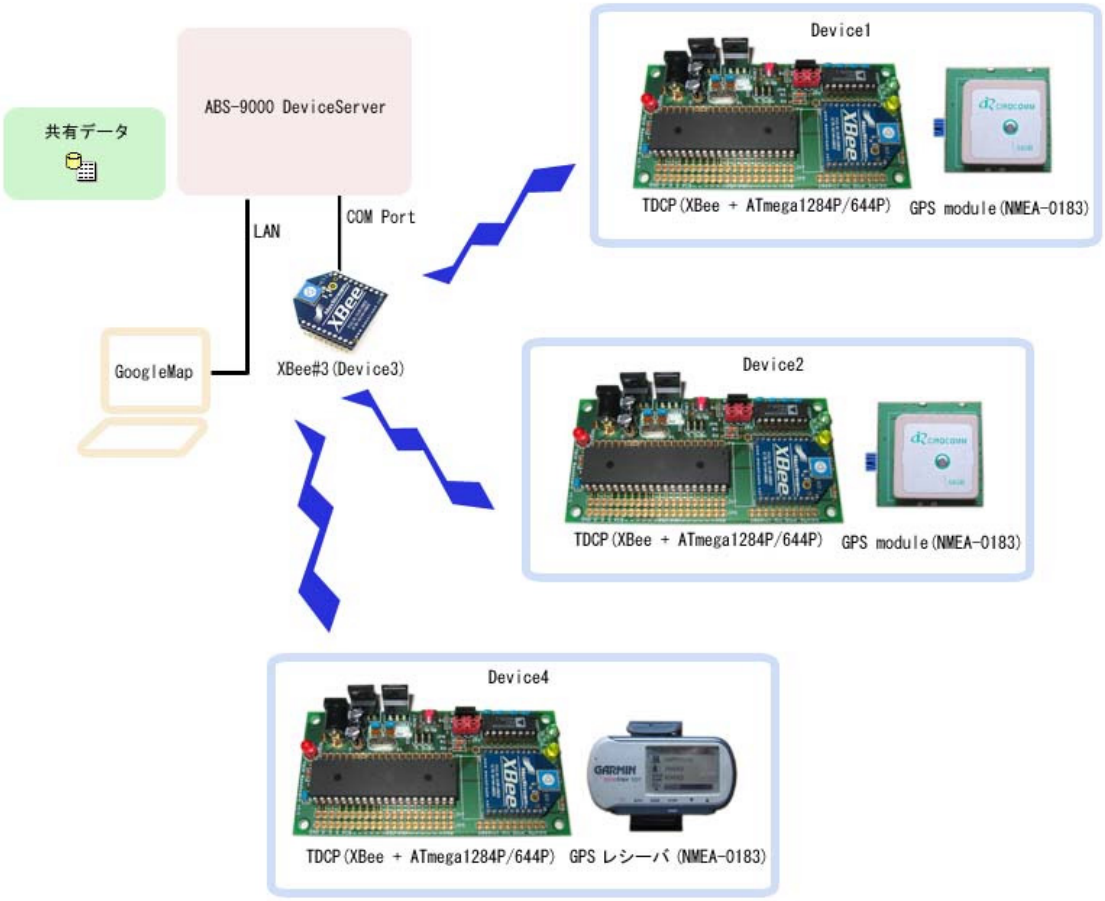
TDCP 紹介とファームウエアダウンロード　http://www.allbluesystem.com/

TDCP マニュアル　http://www.allbluesystem.com/TDCP/TDCP_Users.pdf

1.1　　システム全体構成図

[1] Google Maps は Google Inc. の登録商標です。

[2] XBee XBee® and XBee□PRO® are registered trademarks of Digi, Inc.



このシステムはGPSレシーバが接続された複数のリモートデバイスと、DeviceServer が動作するサーバーPC、Webブラウザを使用するクライアントPC から構成されています。クライアントPC を省略してサーバーPC でWebブラウザを起動して Map を表示することもできます。
リモートデバイスは移動する屋外の移動体(車両や人など)に設置されていて、常にGPS から取得した最新の座標情報をサーバーに送信しています。

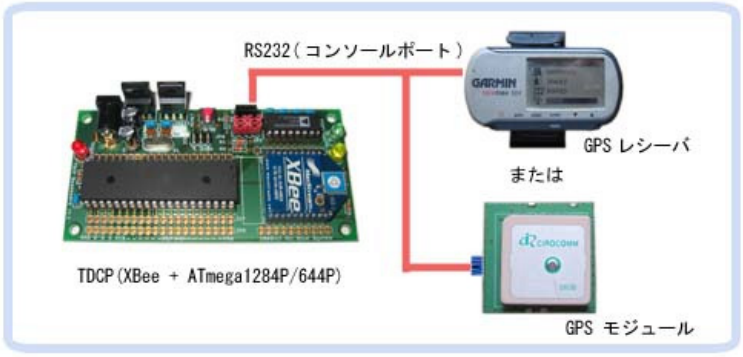
サーバーでは、各リモートデバイスから送信されたGPS イベントデータを内部の共有データに格納して、WebAPI からのリクエストに応じて最新のデバイス情報を提供します。サーバーに接続した PC の Webブラウザから 地図表示用のページにアクセスして Google Map を表示します。ページに記述された javascript によってリモートデバイスの現在位置をマーカーで GoogleMap 上に表示します。

GPS レシーバが接続されたリモートデバイスとサーバーにはそれぞれ XBee 無線モジュールが接続されていて、GPS 座標情報の送信やサーバーからのリモートデバイス操作などはすべて無線でリモートコントロールされます。サーバーとリモートデバイス間をネットワークケーブルやシリアルケーブルで接続する必要がないので、複数のリモートデバイスから同時にイベントを取得したり、移動するリモートデバイスでも問題なく動作させることができます。

リモートデバイスとサーバー間の無線通信がエラーになった場合でも、TDCP が持つパケット通信のリトライ機能に

よって通信安定性を向上させています。またシステムを構成するデバイスの一部がダウンまたは一時的な通信不能になった場合でも、システム全体の動作には影響せず該当デバイスの座標位置が一時的に影響を受けるだけになります。サーバー自身が一時的にダウンした場合でも、特別な設定操作を行わずにサーバー復帰後は自動的に座標情報の取得やクライアントからの表示を継続できます。

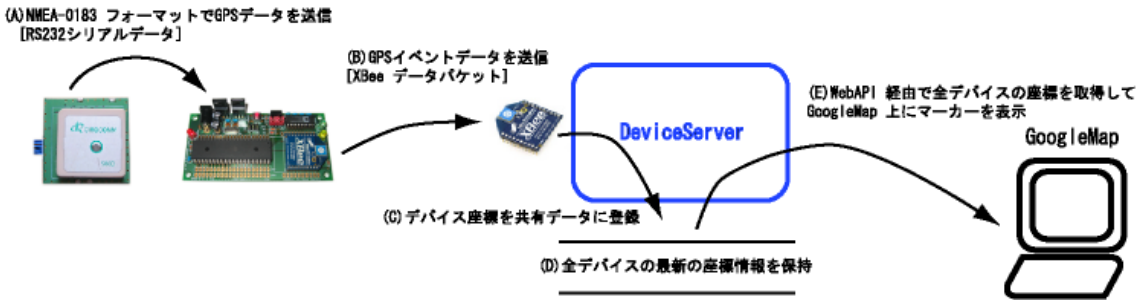
1.2 リモートデバイス構成図



リモートバイスはマイクロコントローラとXBee デバイスから成る TDCP ベースボードに、GPS レシーバもしくは GPS モジュールを RS232 ポート（コンソールポート）で接続した構成になっています。

TDCP ベースボードの コンソールポートから、NMEA-0183 フォーマットの測位情報を取得して、XBee デバイスからデバイスと座標情報から構成される GPS イベントデータパケットがサーバーに送信されます。

1.3 リモート GPS イベントデータの処理フロー



GPS 座標データがクライアントPC でマップ表示されるまでの動作フローを説明します。

リモートデバイスでは、GPS レシーバで決められた間隔(1 Hz ぐらいの更新スピード)で、NMEA-0183 フォーマットで常に測位情報を受信しています。(A)

NMEA-0183 GPS 測位情報の中で $GPRMC, $GPGGA センテンスを受信すると、リモートデバイスから XBee 無線モジュ

ールを経由してサーバーに GPSイベントデータが送信されます。GPSイベントデータ中には、デバイスのアドレスや位置、速度等の情報が格納されています。このイベントの送信間隔は GPS レシーバの出力するNMEA-0183 測位情報更新間隔と同一になります。(B)
イベントデータは リモートデバイス内の XBee モジュールからXBee データパケットとして送信され、サーバー(DeviceServer)側に接続されたXBee デバイスで受信されます。(B),(C)

サーバー側では XBee データパケット受信時にイベントハンドラスクリプト(XBEE_TDCP_DATA)が実行されます。イベントハンドラスクリプト中で各デバイスの座標情報を DeviceServer の共有データに登録します。リモートデバイスが複数ある場合には、同様に全てのデバイスの座標情報が共有データに登録されます。(D)

クライアントの Webブラウザから、GoogleMap 表示用ページを開きます。ページ中に記述された javascript で定期に DeviceServer の WebAPI をコールして、最新のデバイスの座標情報を共有データから取得します。
その後GoogleMap にマーカーを表示します。(E)

2 必要な機材・リソース

2.1 サーバーPC

必要なシステムやデバイス	説明
ABS-9000 DeviceServerの動作しているPC	DeviceServer の動作する PCが１台必要です。
XBee デバイス	サーバー PC 用に Digi International Inc. 社製 XBee IEEE 802.15.4 デバイスが 1台必要です。DeviceServer の COM ポートに接続して各リモートデバイス間との通信を行います。 DeviceServer は、XBee デバイスのファームウエアバージョンの “10CD” にのみ対応しています。(必要に応じて XBee ファームウエアの更新を行ってください)

2.2 リモートデバイス

必要なシステムやデバイス	説明
TDCP プログラムが動作している CPU ボード	“TDCP ユーザーマニュアル” のTDCP リモートコントローラボード作成例を参照してください。 (http://www.allbluesystem.com/TDCP/TDCP_Users.pdf)
XBee デバイス	サーバー PC 用に Digi International Inc. 社製 XBee IEEE 802.15.4 デバイスが 1台必要です。DeviceServer の COM ポートに接続して各リモートデバイス間との通信を行います。 DeviceServer は、XBee デバイスのファームウエアバージョンの “10CD” にのみ対応しています。(必要に応じて XBee ファームウエ

	アの更新を行ってください)
GPS レシーバもしくは GPS モジュール	NMEA-0183 フォーマットで RS232 ポートから出力可能なこと。 測位情報の更新スピードが設定できる場合には 1Hz 以下にしてください。

3 セットアップ

3.1 XBee デバイス初期設定(サーバー側・リモートデバイス共通)

XBee デバイスを DeviceServer とリモートデバイスの TDCPで使用するために初期設定が必要です。
使用するすべての XBee デバイスで下記の設定を行ってください。

最初に DeviceServerとの接続に必要な最低限の設定を COM ポート経由で行います。Sparkfun Electoronics 社製の XBee Explorer USB などを使用して、仮想 USB ポート経由で接続して設定してください。(これ以外の方法で COM ポート接続する場合も手順は同じです)

リモートデバイスの XBee デバイスについては TDCP ベースボードのコンソールポートのジャンパーを設定して、コンソールポートが RS-232C ラインバッファを経由して直接 XBee デバイスのシリアルポートと接続された状態で操作を行います。TDCP ベースボードのコンソールジャンパー設定の詳しい手順については、"TDCP ユーザーマニュアル"を参照して下さい。http://www.allbluesystem.com/TDCP/TDCP_Users.pdf

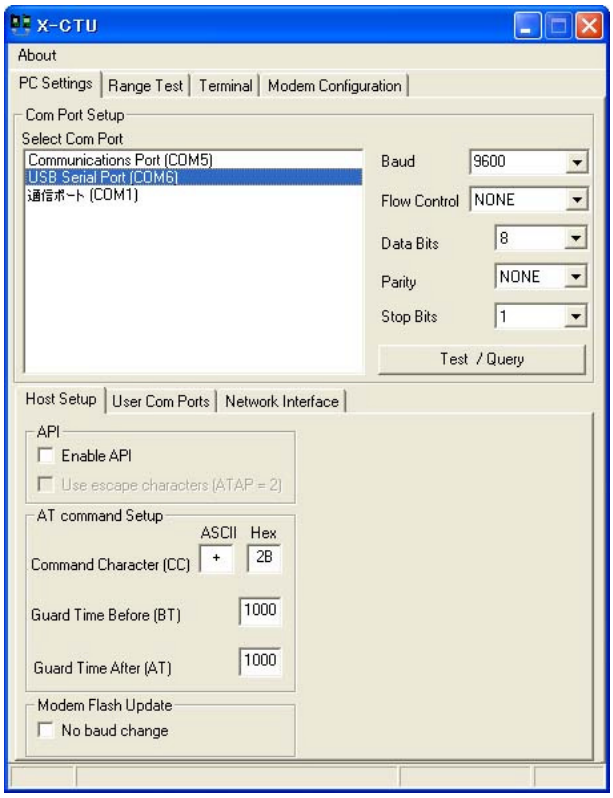
リモートデバイスの XBee デバイスの設定を、DeviceServer に接続する XBee デバイスを設定するときに使用した XBee Explorer USBを使用することもできます。この時には、XBee Explorer USB のソケットから一時的に XBee デバイスを入れ替えて設定してください。

XBee デバイス デフォルト値から変更が必要な設定値	
API モード	1 (default は 0)
PAN(Personal Area Network) ID	任意の値 (default はOx3332) デフォルトの値のままだと、予期しないデバイスからのフレームを受信したり、間違ってデバイスを操作する恐れがありますので、適当な任意の値を設定するようにしてください。このマニュアルではPAN にOxAB90 を使用しています。
16bit Source Address	同一PAN ID 内でユニークな値 (default は 0x0000) この値は、ここで設定しなくても後から XBee 管理プログラムで設定可能ですが、デバイス一覧から選択したデバイスがどのデバイスであるかを見分けることが容易になるように便宜的にここで設定します。

	すべてのデバイス間で違った値を設定してください。 (0x0000,0xFFFF,0xFFFE を除く) 例えば、0x0001, 0x0002,0x0003 など

上記３つの 初期設定のコマンドをXBee に送信するために、XBee デバイスを PC のCOM ポートに接続して Digi international Inc. 社製の X-CTU プログラム、または汎用のターミナルエミュレータプログラムなどを使用します。XBee とCOM ポートのボーレートは初期設定の 9600 bps にして下さい。（ターミナルエミュレータを使用するときは、ローカルエコー ON, 受信時の改行 CR + LF にするとコマンド実行の結果が見やすくなります）

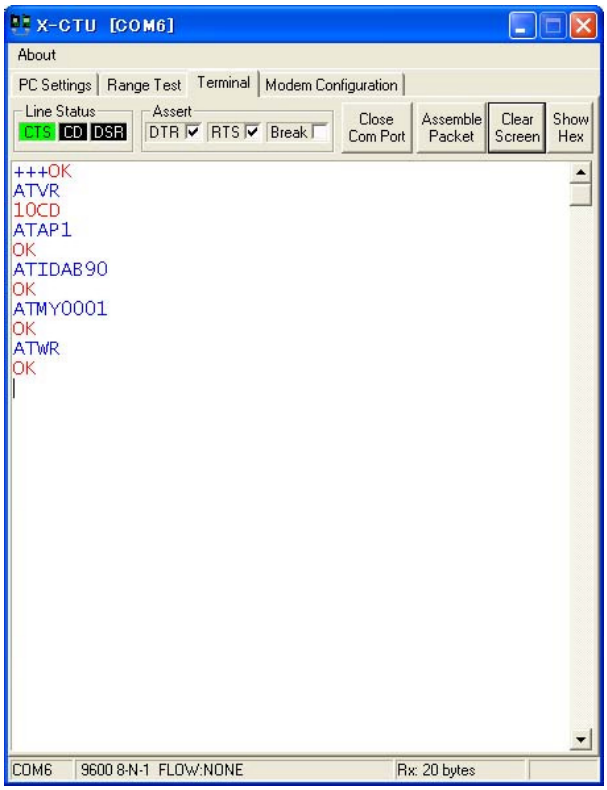
X-CTU プログラムを起動して、COM ポートを選択します。ここでは、USB Serial Port(COM6) を選択しています。



Terminal タブを選択してターミナル画面を表示します。キーボードから、”+++” を入力して、コマンドモードに入ります。コマンドモードに入ると “OK” が表示されますので、続けて以下のコマンド文字列を入力してください。コマンド入力の時間がかかりすぎると、自動的にコマンドモードから抜けてしまいますので、そのときには、”+++” を入力して最初からコマンドを入力し直して下さい。

```
ATVR
ATAP1
ATIDAB90
ATMY0001
ATWR
```

最初に ATVR でファームウエアバージョンを表示しています。"10CD" 以降になっていることを確認してください。
ATAP1 は、API モードを "1" に設定しています。ATIDAB90 は PAN_ID を 0xAB90 に設定しています。もし別の PAN_ID を使用するときには適宜変更してください。次に、ATMY0001 で、デバイスの16 bit Source Address を "0x0001" に設定しています。この部分は、デバイスごとにユニークな値になるように変更して下さい。
最後に、ATWR で、設定値を不揮発メモリに書き込みます。入力時の画面表示は以下のようになります。



X-CTU プログラムを終了します。 その後、XBee Explorer USB に接続する XBee デバイスを切り替えて、 使用するすべての XBee デバイスについて同様に初期設定を行って下さい。
このときに、設定した16bit Source Address の値をデバイス機器にマーキングしておくと、後で XBee デバイス管理プログラムでデバイスを選択するときに、識別し易くなります。

システム上の XBee デバイスの設定値例は、以下のようになります。

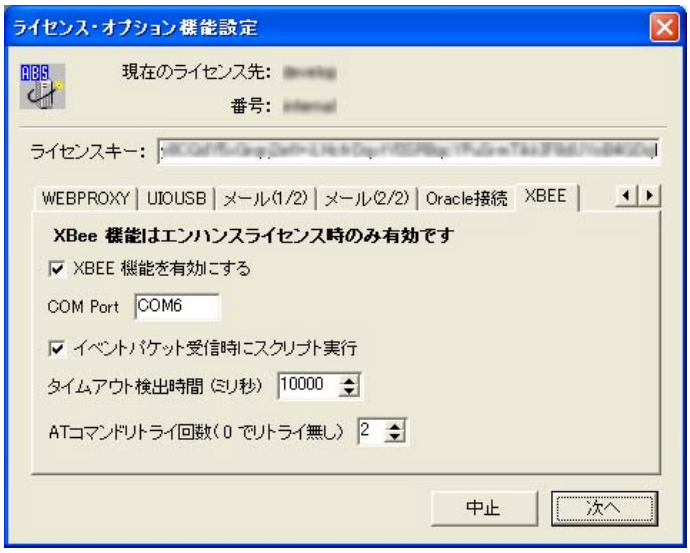
デバイス番号(用途)	API モード(ATコマンド)	PAN_ID(ATコマンド)	16bit Address(ATコマンド)
XBee#1(リモートデバイス)	1(ATAP1)	0xAB90(ATIDAB90)	0x0A01(ATMY0A01)
XBee#2(リモートデバイス)	1(ATAP1)	0xAB90(ATIDAB90)	0x0B02(ATMY0B02)
XBee#3(DeviceServer接続)	1(ATAP1)	0xAB90(ATIDAB90)	0x0C03(ATMY0C03)
XBee#4(リモートデバイス)	1(ATAP1)	0xAB90(ATIDAB90)	0x0D04(ATMY0D04)

3.2 XBee デバイスを DeviceServer に接続

XBee デバイスの初期設定後に、XBee#3(サーバー) を PC の COM ポートに接続します。

デバイスが PC に接続されたら、DeviceServer から XBee デバイスを使用するために COM ポートの設定を行います。サーバー設定プログラム(ServerInit.exe)を起動して、XBEE タブを選択して COM ポート番号を設定して “XBEE 機能を有効にする” にチェックをつけてください。

また、XBee データパケット受信時に XBEE_IO_DATAや XBEE_EVENT_DATA イベントハンドラを実行するために、”イベントパケット受信時にスクリプト実行” にもチェックを付けてください。
サーバー設定プログラムの “次へ” を押して”完了”ボタンが表示されるまで進めて設定を完了して下さい。



3.3 リモート側デバイス接続

リモートデバイスに内蔵する XBee デバイスを、一時的にサーバー PC の XBee Explorer USB などに接続して初期設定した場合には、取り出してリモートデバイスの TDCP ボードにセットします。
リモートデバイスの電源を接続して、サーバー側の XBee と通信できる状態にしておきます。

これ以降の リモートデバイス上の XBee デバイスの設定変更や、リモートデバイスの設定(TDCP プログラムのコンフィギュレーション)はすべて サーバー PC からリモートコマンドで操作できます。

3.4 マスター登録と XBee 詳細設定

リモートデバイスの XBee デバイスと、サーバー PC に接続した XBee デバイスを DeviceServer のマスターファイルに登録します。XBee デバイスの初期設定で設定しなかった NodeIdentifierなどの詳細設定もここで行います。

XBee デバイス管理プログラムを使用して、同一 PAN ID をもつ XBee デバイスを DeviceServer に登録したり、デバイス自身の設定内容を変更します。プログラムメニューから “ALL BLUE SYSTEM” -> “クライアント起動”を選択・実行します。ログインするときは、管理者特権をもったユーザー（例えば DeviceServer セットアップ時に管理者アカウントとして登録したユーザーなど）でログインしてください。デスクトッププログラムが起動したら、“XBee” ツールボタンを選択してXBee デバイス管理プログラムを起動します。

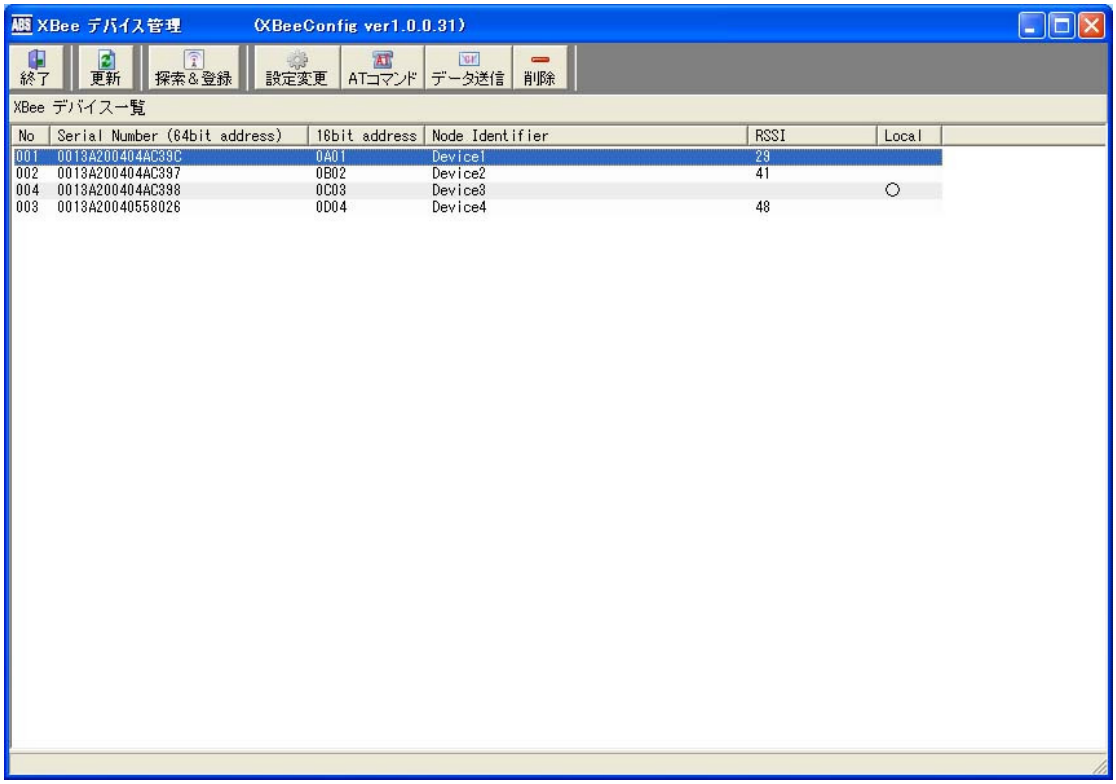
DeviceServer では 登録済みの XBee デバイスをマスターファイルに記録しています。

XBee デバイス登録は、”探索＆登録” ボタンを押すことで、同一 PAN ID のリモートデバイスを見つけて、自動的にマスターファイルに登録します。既に、登録済みのXBee デバイスの項目は最新の情報でマスターファイルの内容が更新されます。DeviceServer に COMポートで直接接続された XBee デバイスについても同様に自動登録されます。

XBee デバイス管理プログラムの “探索＆登録” ツールボタンを押します。

登録確認ダイアログが表示されますので、”OK” を押します。

DeviceServer の COM ポートに直接接続された XBee デバイスで “Node discover” が実行され、付近にある同一 PAN ID の XBee デバイス情報を取得して、自動的にマスターファイルに登録が行われます。XBee デバイス管理プログラムのデバイス一覧には、登録済みのXBee デバイスが表示されます。



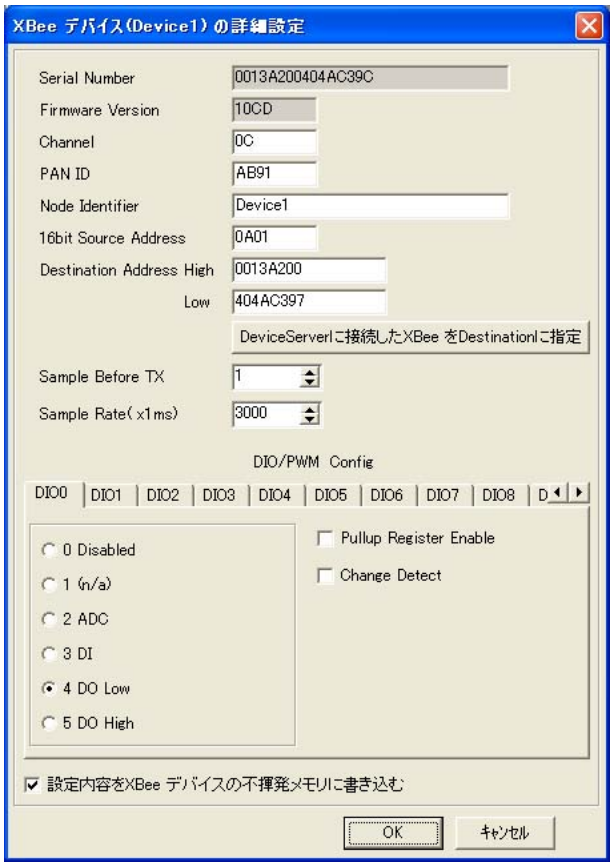
(Node Identifier 項目は、詳細設定修正後に 再度 “探索＆登録” ボタンを押すことで上記のように表示されます)

XBee デバイスの詳細設定を変更するために、XBee デバイス管理プログラムのデバイス一覧から対象デバイスを選択して、”設定変更” ツールボタンを押します。初期設定時に 16 bit Source Address を設定したときは、その値がデバイス一覧に表示されていますので、変更対象の XBee デバイスを確認できます。

選択したXBee デバイスと通信を行って、現在のデバイス情報を取り込みます。
もしエラーが発生したときは、選択したXBee デバイスとの間で通信ができない状態になっていますので、通信経路や電源を確認してください。

XBee デバイスの現在の設定値を取り込んだ後、詳細設定変更ダイアログが表示されます。
ここでは各 XBee デバイスの Node Identifier の設定を行って下さい。Node Identifier に指定できる文字は ASCII で 20文字までです。また、ダイアログに表示されている 16 bit Source Address が、対象のデバイスであるかどうかの確認も行ってください。ここで 16 bit Source Address を任意の値に変更することもできます。

デバイス番号 (16 bit Source Address)	デバイス名 (Node Identifier)
XBee#1 (0x0A01) リモートデバイス	Device1
XBee#2 (0x0B02) リモートデバイス	Device2
XBee#3 (0x0C03) DeviceServer 接続	Device3
XBee#4 (0x0D04) リモートデバイス	Device4



項目を修正した後、 ”設定内容を XBee デバイスの不揮発メモリに書き込む” にチェックを付けて “OK” を押してください。
今回のシステムでは XBee デバイス上の I/O や ADC、サンプリング機能は使用しませんので、”Node Identifier” 以外の項目を設定する必要はありません。

XBeeデバイスの Node Identifier を変更したときは、XBee デバイス管理プログラムのデバイス一覧に表示されている、マスターファイルも更新しておく必要があります。XBee デバイス管理プログラムの “探索＆登録” ツールボタンを押します。

デバイスのNode Identifier または 16bit Source Address以外の詳細設定を変更する場合には、マスターファイルの更新は必要ありません。

3.5 リモートデバイス TDCP 設定

リモートデバイスの TDCP ボードの設定を行います。GPS イベントデータをサーバーに送信するために、下記のTDCPコンフィギュレーション値を設定します。

TDCP プログラム設定項目

項目名称	設定内容	設定用 TDCPコマンド
TDCP 動作モード	app_mode=8 に設定する TDCP では全ての app_mode で GPS レシーバの接続が可能ですのでこれ以外の値を設定しても構いません。	“app_mode,8”
イベントデータ送信先 XBee アドレス	サーバーPC に接続した XBee デバイスのシリアル番号(64bitアドレス)	“server_addr,<64bitAddr(*1)>”
コンソールポートボーレート	接続する GPS レシーバの NMEA-0183 出力時の通信条件に合わせてください。TDCP では “9600” または “4800” が設定可能です。	“uart0_baud,9600”
コンソールポートローカルエコー停止	コンソールポートのローカルエコーバックを停止します。	“uart0_echo,0”
GPS イベント送信	コンソールポートから NMEA-0183 センテンスを受信する毎に、GPS イベントデータをサーバーに送信します。	“position_report,2”
コンフィギュレーション保存	TDCP 設定内容を CPU 内蔵の EEPROM に保存	“config_save”
CPU リセット	TDCP プログラムのリセット。 コンフィギュレーションで保存された新しい app_modeで再起動する。	“reset”(*2)

(*1) サーバーPC に接続した XBee のシリアル番号は XBee デバイス管理プログラムからデバイス一覧で確認するか、スクリプト中から xbee_my_serial_number() 関数を使用して取得できます。

(*2) リセットコマンド実行時は、リモートデバイスからのリプライパケットは常に返らないので、スクリプトから xbee_tdcp_command() 関数を使用してリセットコマンドを送信する場合には、no_result パラメータに true を指定

して下さい。

設定用 TDCP コマンドは、スクリプト中にリモートコマンドを記述して実行する方法と、１コマンドごとに XBee デバイス管理プログラムから送信する方法のどちらか方法で実行できます。

下記に、それぞれの方法で設定を行う場合について記述しますが、どちらかの方法で設定してください。
複数のリモートデバイスがある場合には対象デバイスを切り替えてすべてのリモートデバイス中の TDCP プログラムの設定を行ってください。

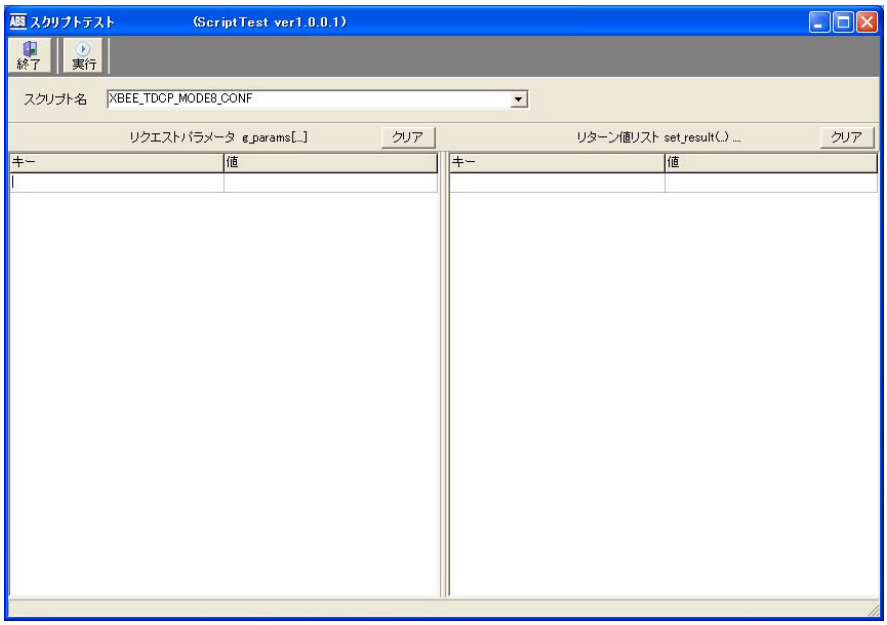
3.5.1 XBEE_TDCP_MODE8_CONF スクリプト作成

下記のスクリプトを作成して、リモートデバイスの TDCP プログラムの設定を行います。

```
file_id = "XBEE_TDCP_MODE8_CONF"
--[[
          TDCP デバイス初期設定用スクリプト
          スクリプト中の device = "xxxxx" 部分を 使用するXBee デバイスの
          NodeIndentifier に変更してからスクリプトを実行してください。
          複数の TDCP デバイスを使用する場合には、全ての XBee デバイスの
          NodeIndentifier についてこのスクリプトを修正・実行してください。
          スクリプトを実行すると、TDCP デバイスに下記の設定が行われた後
          EEPROM に保存された後、デバイスがリセットされます。
]]
local device = "Device1"

log_msg("start..",file_id)
local svr_addr,key,val,stat,result
stat,svr_addr = xbee_my_serial_number()
if (not stat) or (svr_addr == "") then error() end
log_msg("DeviceServer's XBee address is " .. svr_addr,file_id)

stat,result = xbee_tdcp_command(device,"app_mode,8")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"server_addr," .. svr_addr)
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device," uart0_baud,9600")
if (not stat) or (result[2] ~= "1") then error() end
```

```
stat,result = xbee_tdcp_command(device," uart0_echo,0")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"position_report,2")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"config_save")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"reset",true)
if (not stat) then error() end
```

ファイル名(XBEE_TDCP_MODE8_CONF.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。スクリプト実行は、Web の “ScriptControl プログラム” または DeviceServerのクライアントプログラムから実行します。下記は、DeviceServer のクライアントプログラムから実行した画面例です。



“実行”ボタンを押すとスクリプトがサーバーで実行され、リモートデバイスの TDCP プログラムの設定が行われます。ログには、下記の様なコマンド実行ごとのリモートデバイスからの応答パケット受信が記録されます。”$$$<文字列>” の後のカンマに続けて “1” が返っていればコマンド実行が成功したことを示しています。

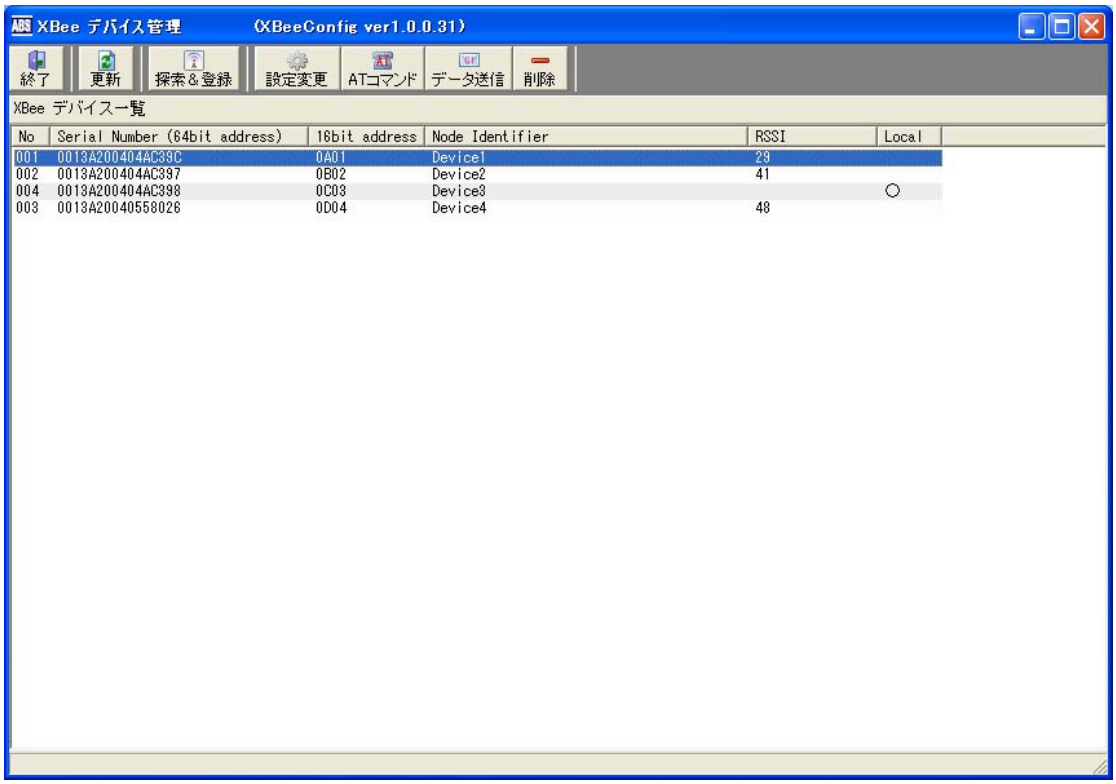
```
2010/03/01 09:30:08 falcon      XBEE_TDCP_MODE8_CONF 0 start..
2010/03/01 09:30:08 falcon      XBEE_TDCP_MODE8_CONF 0 DeviceServer's XBee address is 0013A200404AC398
2010/03/01 09:30:08 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$61052,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$26988,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$16124,1
```

2010/03/01 09:30:09 falcon	XBEE_TDCP_DATA	0 Device1[0A01,0013A200404AC39C] RFData = $$$85710,1
2010/03/01 09:30:09 falcon	XBEE_TDCP_DATA	0 Device1[0A01,0013A200404AC39C] RFData = $$$86327,1
2010/03/01 09:30:09 falcon	XBEE_TDCP_DATA	0 Device1[0A01,0013A200404AC39C] RFData = $$$25659,1

“Device1” の設定が終了したら、スクリプトの内容を編集して設定対象デバイスを “Device4” に変更して同様にスクリプトを実行します。

3.5.2 XBee デバイス管理プログラムから TDCP コマンド実行

DeviceServerのクライアントプログラムを起動して、XBee デバイス管理プログラムを開きます。



“Device1” を選択して、”データ送信” ボタンを押します。下記のダイアログが表示されて、リモートのXBee デバイスに XBee API データパケットを送信できます。

前述の 設定用 TDCP コマンドを入力して “送信ボタン”を押します。

この時、TDCP コマンドプリフィックス文字列の “$$$” とカンマ “,” を 設定用 TDCP コマンドとパラメータの前に付けてください。”$$$” の代わりに “$$$abc” の様な文字列にするとリモートデバイスでコマンドが実行された結果のリプライデータパケットを受け取ることができます。リプライパケット受信時にはログに下記のような内容が出力されます。

```
2010/03/01 22:33:50 falcon      XBEE_TDCP_DATA        0 Device1[0A01,0013A200404AC39C] RFData = $$$abc,1
```

リクエスト時に指定した TDCP コマンドプリフィックスの後のカンマに続けて “1” が返った場合にはコマンド実行が成功したことを示します。コマンド実行が失敗した場合には “0” が返ります。ただし、“reset” コマンド実行時にはリプライパケットは返りませんので注意してください。TDCP コマンドプリフィックスについては ”TDCP ユーザーマニュアル”を参照して下さい。

すべての 設定用 TDCP コマンドを同様に実行して下さい。“Device1” の設定が終了したら”閉じる” ボタンでデバイスリスト画面に戻ります。同様の手順で “Device2”,“Device4” の設定を行ってください。

3.6 GPS レシーバ(モジュール)接続

リモートデバイスの TDCP ボードの設定が完了したら、TDCP ボードのコンソールポートに GPS レシーバを RS232C インターフェースで接続します。TDCP でサポートするGPS の通信条件は下記の様になっています。

シリアルコンソールポートに接続するGPS レシーバ通信条件	
通信フォーマット	NMEA-0183
TDCP で使用するNMEA-0183センテンス(*1)	$GPRMC $GPGGA
通信インターフェース	RS-232
ボーレート	9600 または 4800
ビット長	8
ストップビット	1
パリティ	無し

フロー制御	無し
測位情報更新スピード(回/秒) (*2)	1 以下

(*1) 接続するGPS レシーバの設定で、これら以外のセンテンスが送信された場合は、TDCP はそれらの内容は無視します。これらの TDCP で不用なセンテンスをGPS レシーバ側で抑止することが可能な場合は、できるだけ送信しないようにしてください。

(*2) GPS レシーバの 測位情報更新スピードが 1 Hz よりも大きい場合には、必ず 1 Hz 以下の更新スピードになるように GPS レシーバの設定を変更して下さい。

TDCP に GPS レシーバを接続する場合には、シリアルコンソールポートに RS-232 インターフェースで接続します。マイクロプロセッサのシリアルポート#0 の RX,GND を GPSレシーバ側の TX,GND に接続してください。TDCP からGPS レシーバにデータ送信することはありませんので、シリアルポート#0 の TX 側の接続は不要です。(接続しても構いません) TDCP では H/W フロー制御を使用しませんので、必要に応じてGPSレシーバ側の RTS,CTS の処理を行ってください。

GPS レシーバから、NMEA-0183 センテンスが送信されると TDCP では自動的に最新の測位情報を内部に保存・更新しています。リモートから TDCP コマンドを実行することで、何時でもこれらの測位情報を利用可能です。GPS レシーバが停止していた場合には、再びレシーバから測位情報(NMEA-0183 センテンス)を取得するまでイベントは発生しません。

GPS レシーバ接続や、GPS イベントの詳細については "TDCP ユーザーマニュアル"を参照して下さい。
http://www.allbluesystem.com/TDCP/TDCP_Users.pdf

4 スクリプト作成

システム全体の動作をコントロールするスクリプトを記述します。

⚠ 注意

スクリプト中に日本語を記述するときは、スクリプトファイルを UTF-8N 形式で保存してください。Shift_JISやUTF-8 BOM付き形式などで保存すると、DeviceServer でエラーが発生します。Windows付属のワードパッドやメモ帳ではこの形式で保存できませんので、別途 UTF-8N 形式で保存可能なエディタソフト (*1) を使用してください。
(*1) TeraPad などのソフトウエアがよく使用されています。

4.1 XBEE_TDCP_DATA スクリプト作成

サーバーPC でリモートデバイスからイベントデータ、リクエスト応答パケットなどを受信したときに実行されるスクリプトを作成します。

DeviceServer をインストールしたときに、初期ファイルとして XBee データパケット内容をログに出力する機能のみが記述されていますので、これに追加記述します。

GPS イベントデータ(他のTDCPイベントを受信した場合にもコールされます)を受信した場合には、共有データに登録するための GPS_RECEIVE スクリプトを実行します。他のイベント処理をスクリプトに追加した時でも、GPS_RECEIVE スクリプトの処理時間に影響されないように、script_fork_exec() を使用して別スレッドで GPS_RECEIVE スクリプトを実行します。
また、GPS_RECEIVE スクリプトの実際のファイル GPS_RECEIVE.lua を GPS_MAP フォルダの中にまとめて格納する場合にはスクリプト起動時に指定する名前が “GPS_MAP/GPS_RECEIVE” となります。

```
file_id = "XBEE_TDCP_DATA"
--[[
*******************************************************************************
* イベントハンドラスクリプト実行時間について                                   *
*******************************************************************************
一つのスクリプトの実行は長くても数秒以内で必ず終了するようにしてください。
処理に時間がかかると、イベント処理の終了を待つサーバー側でタイムアウトが発生します。
また、同時実行可能なスクリプトの数に制限があるため、他のスクリプトの実行開始が
待たされる原因にもなります。
頻繁には発生しないイベントで、処理時間がかかるスクリプトを実行したい場合は
スクリプトを別に作成して、このイベントハンドラ中から script_fork_exec() を使用して
別スレッドで実行することを検討してください。
*******************************************************************************
XBEE_TDCP_DATA スクリプト起動時に渡される追加パラメータ
-------------------------------------------------------------------------------
キー値              値                                          値の例
-------------------------------------------------------------------------------
APIType             フレームデータ中のAPI Type(16進数2桁)       81
SourceAddress       フレームデータ中のSourceAddress
                    16bit アドレスの場合(16進数4桁)             0A01
                    64bit アドレスの場合(16進数16桁)            0013A200404AC397
SerialNumber        XBee デバイスの SerialNumber
                    DeviceServer に保持されたマスターファイルを使用して、
                    SourceAddress から変換した値が設定される。  0013A200404AC397
NodeIdentifier      XBee デバイスの NodeIdentifier。
                    DeviceServer に保持されたマスターファイルを使用して、
                    SourceAddress から変換した値が設定される。  Device1

RSSI                フレームデータ中のRSSI(16進数2桁)           45
```

```
Options             フレームデータ中Options                              00
TDCP_COUNT          TDCP データカラム数                                  2
TDCP_<Column#>      TDCP データ値(ASCII 文字列)
                    TDCP_1 は常にコマンドプリフィックス文字列を表す      "$$$1234"
                    "$$$" で始まり、0文字以上の任意の文字列が後に続く。
                    TDCP_2 はコマンド実行ステータスを表す                "1"
                    "1" はコマンド実行成功、"0" は失敗を示す
                    イベントデータの場合にはイベント名が入る
                    TDCP_3以降のデータはTDCPコマンド毎に決められた、
                    オプション文字列が入る
                    <Column#> には 最大、TDCP_COUNT まで 1から順番に
                    インクリメントされた値が入る。
]]
------------------
-- BEGIN SCRIPT --
------------------
local rf_data = "";
local tkey,i;
local max_idx = tonumber(g_params["TDCP_COUNT"]);
for i = 1,max_idx,1 do
    tkey = "TDCP_" .. tostring(i);
    if (i == 1) then
        rf_data = g_params[tkey]
    else
        rf_data = rf_data .. "," .. g_params[tkey]
    end;
end;
log_msg(g_params["NodeIdentifier"] .. "[" .. g_params["SourceAddress"] .. "," .. g_params["SerialNumber"] ..
"] RFData = " .. rf_data,file_id)
-------------------------------------------------------------------------
-- GPS イベントを受信した場合は、座標記録用スクリプトを実行する
-------------------------------------------------------------------------
if g_params["TDCP_2"] == "GPS" then
    stat,addr16,serial,nodename = xbee_find_device(g_params["TDCP_3"]);
    if stat then
        local key_list = list_to_csv("SerialNumber","RemoteDeviceName","Status",
                        "Latitude","LatCompass","Longitude","LonCompass","Altitude")
        local val_list = list_to_csv(serial,nodename,g_params["TDCP_5"],g_params["TDCP_6"],
                        g_params["TDCP_7"],g_params["TDCP_8"],g_params["TDCP_9"],g_params["TDCP_12"])
```

```
        if not script_fork_exec("GPS_MAP/GPS_RECEIVE",key_list,val_list) then error() end
        -- if not script_fork_rexec("remote_device_server","GPS_MAP/GPS_RECEIVE",key_list,val_list) then
            error() end
    end
end
------------------
-- END SCRIPT   --
------------------
```

ファイル名(XBEE_TDCP_DATA.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

4.2 GPS_MAP/GPS_RECEIVE スクリプト作成

リモートデバイスから GPS イベントデータを受信したときに XBEE_TDCP_DATA スクリプトから呼び出されて、データベースに登録するためのスクリプトを作成します。
スクリプトパラメータには下記のパラメータが XBEE_TDCP_DATA スクリプトから渡されます。

- g_params["SerialNumber"] リモートデバイスに接続している XBee モジュールのシリアル番号
- g_params["RemoteDeviceName"] リモートデバイス名(RemoteDeviceName)
- g_params["Latitude"] 緯度
- g_params["LatCompass"] 緯度コンパス
- g_params["Longitude"] 経度
- g_params["LonCompass"] 経度コンパス
- g_params["Altitude"] 高度

これらのデータをDeviceServer の共有データに保存します。キー名にデバイス名を含めることで、複数のデバイスからの GPS イベント情報を全て保存します。また、クライアント PC の javascriptで GoogleAPI を扱うことを考慮して座標単位の変換も行います。

クライアント PC や他のスクリプトから、現在 GPS イベントデータを送信している有効なリモートデバイス一覧を取得できるように、データベース(キー名 “GPSDeviceList”) にデバイス名とシリアル番号も登録しておきます。

```
file_id = "GPS_RECEIVE"
local now = os.date "*t";
local timestamp = string.format("%4.4d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d",
                        now["year"],now["month"],now["day"],now["hour"],now["min"],now["sec"]);
-----------------------------------------------------
-- GPS レシーバーステータスが有効な場合のみ処理を行う
-----------------------------------------------------
```

```
if g_params["Status"] == "A" then
    ------------------------------------------------------------
    -- デバイス名とシリアル番号を共有データに登録
    ------------------------------------------------------------
    if not add_shared_strlist("GPSDeviceList",g_params["SerialNumber"] .. ","
                        .. g_params["RemoteDeviceName"],true) then error() end;
    ------------------------------------------------
    -- 取得した座標値の単位を "degree" に変換する
    ------------------------------------------------
    local stat,lat,lon = gps_coordinate_deg("dmm",g_params["Latitude"],
                        g_params["Longitude"],g_params["LatCompass"],g_params["LonCompass"]);
    ------------------------------------------------------------------------
    -- デバイス毎に最後に取得した座標を共有データに保存する
    -- "<DeviceName>_Position,<latitude>,<longitude>,<alititude>,<timestamp>"
    ------------------------------------------------------------------------
    if not set_shared_data(g_params["RemoteDeviceName"] .. "_Position",
                          tostring(lat) .. "," .. tostring(lon) .. "," .. g_params["Altitude"]
                          .. "," .. timestamp) then error() end;
end;
```

ファイル名(GPS_RECEIVE.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts¥GPS_MAP” に保管します。

4.3 GPS_MAP/GPS_REPORT スクリプト作成

クライアントPCで動作している Webブラウザの javascript プログラムから、現在 GPS イベントデータを送信している有効なリモートデバイスと座標データー覧を取得するためのスクリプトを作成します。

このスクリプトでは、GPS_MAP/RECEIVE スクリプトで登録された共有データをアクセスして、最新の全てのデバイスリストと座標データをスクリプトリターンパラメータに設定して、クライアント PC の javascript プログラムに渡します。

```
file_id = "GPS_REPORT"
------------------------------------------------------------------------
-- 各デバイス毎の GPS データ(node_identifier,latitude,longitude,timestamp) を
-- リターンパラメータに設定
------------------------------------------------------------------------
local node_identifier;
local pos_data;
local cnt = 0;
```

```
local stat,strlist = get_shared_strlist("GPSDeviceList");
if not stat then error() end;
for key,val in ipairs(strlist) do
node_identifier = string.gsub(val,"(%w+),(%w+)","%2");
    local stat,pos_data = get_shared_data(node_identifier .. "_Position");
    if not stat then error() end;
    if not script_result(g_taskid,tostring(cnt),node_identifier .. "," .. pos_data) then error() end;
    cnt = cnt + 1;
end
```

ファイル名(GPS_REPORT.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts¥GPS_MAP” に保管します。

4.4 GPS_MAP/GPS_CLEAR スクリプト作成

クライアントPCで動作している Webブラウザで表示したマップ上の “Clear” ボタンを押した時に、サーバー上の全てのデバイス情報を削除するためのスクリプトを作成します。

デバイス毎のGPS 座標情報とデバイス一覧を共有データから消去します。

```
file_id = "GPS_CLEAR"

local node_identifier;
local stat,strlist = get_shared_strlist("GPSDeviceList");
if not stat then error() end;
for key,val in ipairs(strlist) do
    node_identifier = string.gsub(val,"(%w+),(%w+)","%2");
    if not set_shared_data(node_identifier .. "_Position","") then error() end;
end
if not clear_shared_strlist("GPSDeviceList") then error() end;
```

ファイル名(GPS_CLEAR.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts¥GPS_MAP” に保管します。

4.5 PERIODIC_TIMER スクリプト作成

今回のアプリケーションでは、クライアントPC からDeviceServer をアクセスする場合にユーザー認証を省略しています。Webブラウザで表示するページ中の javascript からは、予め DeviceServer に作成されたセッション情報を利用することで簡単な仕組みにしています。

このためのセッション情報を自動的に作成するための記述を PERIODIC_TIMER スクリプトに作成します。

PERIODIC_TIMER スクリプトは、DeviceServer で１分ごとに起動されますので、一回だけセッションを作成するための動作を記述します。 Webページ上の javascript で使用するセッション情報はここで作成したセッショントークン文字列と同一のものを指定してください。

共有変数 “STARTUP_SCRIPT” は、一回だけの実行をカウントするために使用しています。（別の名前を使用しても構いません）

```
file_id = "PERIODIC_TIMER"
------------------
-- BEGIN SCRIPT --
------------------
local stat,val
-------------------------------------------------------------
-- DeviceServer 起動時に一回だけスクリプトを実行する
-------------------------------------------------------------
stat,val = get_shared_data("STARTUP_SCRIPT")
if not stat then error() end
if val == "" then
    if not inc_shared_data("STARTUP_SCRIPT") then error() end
    ------------------------------
    -- 起動時に一回だけ実行される
    ------------------------------


    ---------------------------------------------------------------
    -- セッショントークンを自動的に作成する
    -- WebAPI でログイン認証を省略してアクセスする場合に利用する
    -- セッショントークン文字列を変更するときは、
    -- WebAPI を使用する側の設定も変更すること
    ---------------------------------------------------------------
    local token
    stat,token= create_session("1234",true)
    if not stat then error() end
end
```

ファイル名(PERIODIC_TIMER.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5 Web クライアントプログラム設定

インターネットまたはLAN 上の Webブラウザから、現在のリモートデバイスの位置を GoogleMap 上に表示するための Webページを設定するための説明をします。

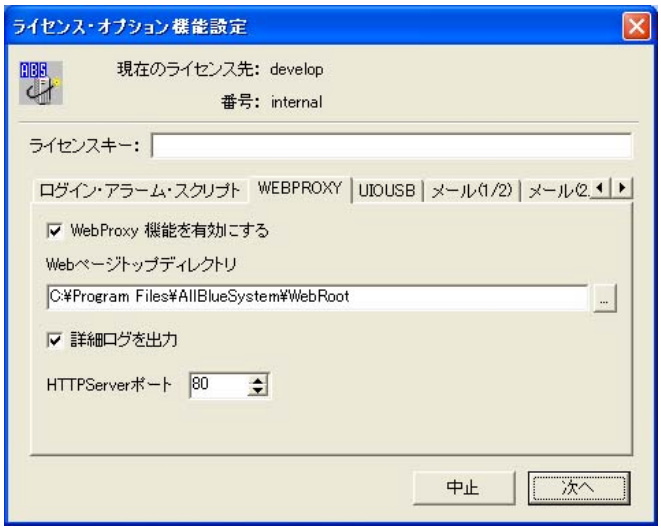
5.1 DeviceServer の WebProxy セットアップ

DeviceServer には HTTP サーバー機能と、Webブラウザ中で動作する javascript やFlash アプリケーションからコマンド操作をするためのWeb API の機能を持った WebProxy があります。DeviceServer インストール時に WebProxy のセットアップを行っていない場合は、下記の手順で設定を行います。

最初に、DeviceServer が動作している PCで既に HTTP サーバープログラムが動作していないことを確認してください。ポート番号80(http) でWebProxy が動作しますので、マイクロソフト社製 HTTPサーバー(IIS)や、Apache などのHTTPサーバープログラムを既に使用している場合には、WebProxy で設定するポート番号を変更してください。

DeviceServer と Webサーバープログラムの動作する PC を分離して設置することもできます。

サーバー設定プログラム(ServerInit.exe) プログラムをメニューから選択して実行します。"WEBPROXY" 設定タブの内容を下記のようにして、"次へ" ボタンを続けて押していってサーバーの設定を完了してください。DeviceServer が再起動して WebProxy 機能が有効になります。



5.2 GoogleMap 表示用 Web ページ設定

WebProxy 設定で指定した "Webページトップディレクトリ" 項目が DeviceServer の動作するPC に Webブラウザでアクセスした時のルートディレクトリになります。
デフォルト設定では DeviceServer が動作しているPC の Webブラウザから "http://localhost/index.html" をアクセスすると、"C:¥Program Files¥AllBlueSystem¥WebRoot¥index.html" ファイルにアクセスすることになります。
"HTTPServerポート" を 8080 に設定した場合には、Webブラウザの URL は "http://localhost:8080/index.html" を指定します。

ここでは、Webページトップディレクトリ ("C:¥Program Files¥AllBlueSystem¥WebRoot") 以下に gps_map フォルダ

を作成して、GoogleMap を表示するための HTML ファイル
(”C:¥Program Files¥AllBlueSystem¥WebRoot¥gps_map¥index.html”) を作成します。

DeviceServer 自身の HTTPサーバーのWebページトップディレクトリ以下にファイルを配置しないで、別の HTTP サーバーで管理しているフォルダにページを配置することもできます。このような HTTP サーバーが動作している PC と DevivceServer の PCを分けて設置する場合には、GoogleMap を表示するページ(index.html ファイル) 中の javascript から DeviceServer をアクセスするホスト名指定を変更してください。具体的には DeviceServer の動作している PC のホスト名または IPアドレス(インターネット上に配置する場合にはドメイン名とホスト名)を index.html 中の “server_host_url” 変数に設定してください。

```
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>リモートGPS レシーバからXBee経由で現在位置を受信して地図に表示</title>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">

// サーバーホスト URL 設定
// DeviceServer 自身の HTTP サーバーを使用する場合には空にする
//
//var server_host_url = "http://your_DeviceServer_public_url:80";
var server_host_url = "";

// 位置表示更新間隔(ms)
var updateInterval = 1000;

// DeviceServer設定
var session_token = "1234";

// デバイス毎の最新の座標情報を保存
var DeviceList = new Object();
var DeviceLat = new Object();
var DeviceLon = new Object();
var DeviceAlt = new Object();
var DeviceTimestamp = new Object();
var DevMarkers = new Object();
var DevInfoWindows = new Object();
```

```
// 選択中のセンターマーカー(デバイス名)
var currentCenterMarker;


// 初期化
function initialize() {
    var myOptions = {
        zoom: 15,
        center: new google.maps.LatLng(35.689875642551,139.69165634866),
        mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    map = new google.maps.Map(document.getElementById("map_canvas"),myOptions);
    radio_add("dev_rdo","OFF",true);
    currentCenterMarker = "OFF";

    setInterval("on_timer()",updateInterval);
}

// DeviceServer の LogService にログメッセージを出力
function log(msg){
    var url = server_host_url + "/command/log" +
              "?message=" + encodeURIComponent(msg) +
              "&module=" + encodeURIComponent("GPS_MAP(js)");
    var script  = document.createElement("script");
    script.type = "text/javascript";
    script.src  = url;
    document.body.appendChild(script);
}

// DeviceServer のスクリプトを実行する
// callback パラメータを省略するとリターンパラメータを受け取らない
function script_exec(name,callback){
    if (callback == undefined){
        var url = server_host_url + "/command/json/script" +
                  "?session=" + encodeURIComponent(session_token) +
                  "&resultrecords=0" +
                  "&callback=" + encodeURIComponent("default_callback") +
                  "&name=" + encodeURIComponent(name);
    } else {
        var url =    server_host_url + "/command/json/script" +
```

```
                    "?session=" + encodeURIComponent(session_token) +
                    "&callback=" + encodeURIComponent(callback) +
                    "&name=" + encodeURIComponent(name);
    }
    var script  = document.createElement("script");
    script.type = "text/javascript";
    script.src  = url;
    document.body.appendChild(script);
}

// script_exec() でコールバックを指定しない場合のデフォルトコールバック関数
function default_callback(data){
    if (data.Result != "Success"){
        log("script error!");
        return;
    }
}

// スクリプトリターンパラメータ中の座標情報を配列に格納
function update_gps_data(data){
    if (data.Result != "Success"){
        log("script error!");
        return;
    }
    for(v in data.ResultParams){
        var gps_data = data.ResultParams[v].split(",");
        var dev_name = gps_data[0];
        if(DeviceList[dev_name] == null){
            log("added " + dev_name);
            DeviceList[dev_name] = dev_name;
        }
        DeviceLat[dev_name] = Number(gps_data[1]);
        DeviceLon[dev_name] = Number(gps_data[2]);
        DeviceAlt[dev_name] = Number(gps_data[3]);
        DeviceTimestamp[dev_name] = gps_data[4];
    }

    update_markers();
    center_marker();
```

```
}

// ラジオボタンで選択中のマーカー位置に地図をセンタリングする
function center_marker(){
    if (currentCenterMarker != "OFF"){
        centerLatlon = new google.maps.LatLng(DeviceLat[currentCenterMarker],
                        DeviceLon[currentCenterMarker]);
                        map.setCenter(centerLatlon);
    }
}

// 新しいデバイスからのGPS 情報を取得した場合に、センタリング指示用のラジオボタンを追加
function radio_add(group,name,checked){
    var chk_input = document.getElementById("rdo_" + name)
    if (chk_input == null) {
        var myinput = document.createElement("input");
        myinput.setAttribute('type', "radio");
        myinput.setAttribute('value',name);
        myinput.setAttribute('id',"rdo_" + name);
        myinput.setAttribute('name',group);
        myinput.onclick = function() {on_radio_click(this)};
        document.getElementById("controls").appendChild(myinput);
    }
    var chk_lbl = document.getElementById("lbl_" + name)
    if (chk_lbl == null) {
        var myinput_lbl = document.createElement("label");
        myinput_lbl.setAttribute('for', "rdo_" + name);
        myinput_lbl.setAttribute('id',"lbl_" + name);
        myinput_lbl.appendChild(document.createTextNode(name));
        document.getElementById("controls").appendChild(myinput_lbl);
    }

    if (checked == true) {
        myinput.checked = true;
    }
}

// 指定したラジオボタンをチェック状態にする
function radio_check(name){
```

```
    var chk_input = document.getElementById("rdo_" + name)
    if (chk_input != null) {
        chk_input.checked = true;
    }
}

// センタリング指示用のラジオボタン削除
function radio_del(name){
    var del_rdo = document.getElementById("rdo_" + name)
    if (del_rdo != null) {
        document.getElementById("controls").removeChild(del_rdo);
    }
    var del_lbl = document.getElementById("lbl_" + name)
    if (del_lbl != null) {
        document.getElementById("controls").removeChild(del_lbl);
    }
}

// infowindow に表示する文字列コンテンツの作成
function getCurrentPosInfo(dev_name){
    return ("<p><b>" + dev_name +"</b>" +
            "<p>lat:" + DeviceLat[dev_name].toFixed(6) +
            " lon:" + DeviceLon[dev_name].toFixed(6) +
            " alt:" + DeviceAlt[dev_name].toFixed(1) + "</p>");
}

// 新しいデバイスからのGPS 情報を取得した場合に、マーカーとinfowindow を作成
// 既存のGPS 情報を取得した場合には、マーカー位置を最新の座標に更新
function update_markers(){
    for(dev_name in DeviceList){
        if(DevMarkers[dev_name] == null){
            var myLatLng = new google.maps.LatLng(DeviceLat[dev_name],DeviceLon[dev_name]);
            var newMarker = new google.maps.Marker({
                                position: myLatLng,
                                map: map,
                                title:dev_name
                                });
            DevMarkers[dev_name] = newMarker;
            var newInfowindow = new google.maps.InfoWindow({
```

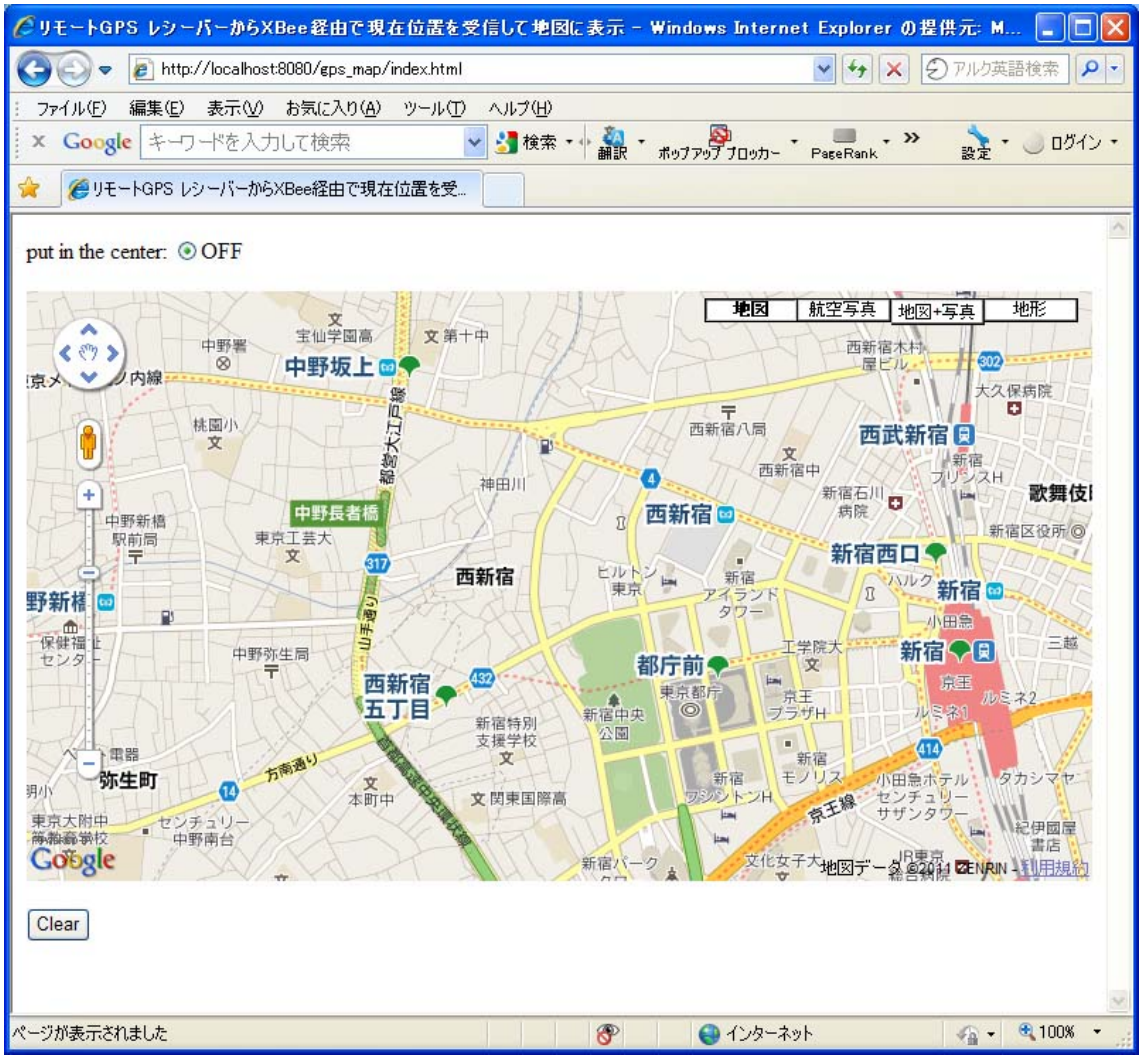
```
                                content:getCurrentPosInfo(dev_name)
                            });
            DevInfoWindows[dev_name] = newInfowindow;
            google.maps.event.addListener(newMarker, 'click',function() {
                        DevInfoWindows[this.title].open(map,DevMarkers[this.title]);
                    });

            radio_add("dev_rdo",dev_name);
        } else {
            var newLatlon = new google.maps.LatLng(DeviceLat[dev_name],DeviceLon[dev_name]);
            DevMarkers[dev_name].setPosition(newLatlon);

            var tmpContent = getCurrentPosInfo(dev_name);
            if (DevInfoWindows[dev_name].getContent() != tmpContent){
                DevInfoWindows[dev_name].setContent(tmpContent);
            }
        }
    }
}

// センタリングラジオボタンがクリックされた時のハンドラ
function on_radio_click(obj){
    currentCenterMarker = obj.value;
}

// 定期的に DeviceServerのスクリプトを実行して、最新のデバイスの GPS座標情報を取得する
function on_timer(){
    script_exec("GPS_MAP/GPS_REPORT","update_gps_data");
};

// 全てのマーカーとinfowindow を削除する
function clear_gps_data(){
    for(dev_name in DeviceList){
        DevMarkers[dev_name].setMap(null);
        delete DevMarkers[dev_name];
        DevInfoWindows[dev_name].close();
        delete DevInfoWindows[dev_name];
        delete DeviceList[dev_name];
        //
```

```
        radio_del(dev_name);
    }
    currentCenterMarker = "OFF";
    radio_check("OFF");
}

// DeviceServer に保存されたGPS座標情報を消去する
function delete_markers(){
    script_exec("GPS_MAP/GPS_CLEAR","clear_gps_data");
}

</script>
</head>
<body onload="initialize()">
    <p id="controls">put in the center: </p>
    <div id="map_canvas" style="width:100%; height:78%"></div>
    <p id="controls">
        <input type="button" onclick="delete_markers()" value="Clear"></input>
    </p>
</body>
</html>
```

6 アプリケーションを起動する

DeviceServer が動作しているPC もしくは、ネットワーク接続されたPC から Webブラウザを起動して、前項で設置したWebページをアクセスします。

例えば、WebProxy のポート番号が 8080 に設定されていた場合には、"http://localhost:8080/gps_map/index.html" を開きます。



（ページを開いた初期画面。デバイスからの座標情報を取得していない状態）

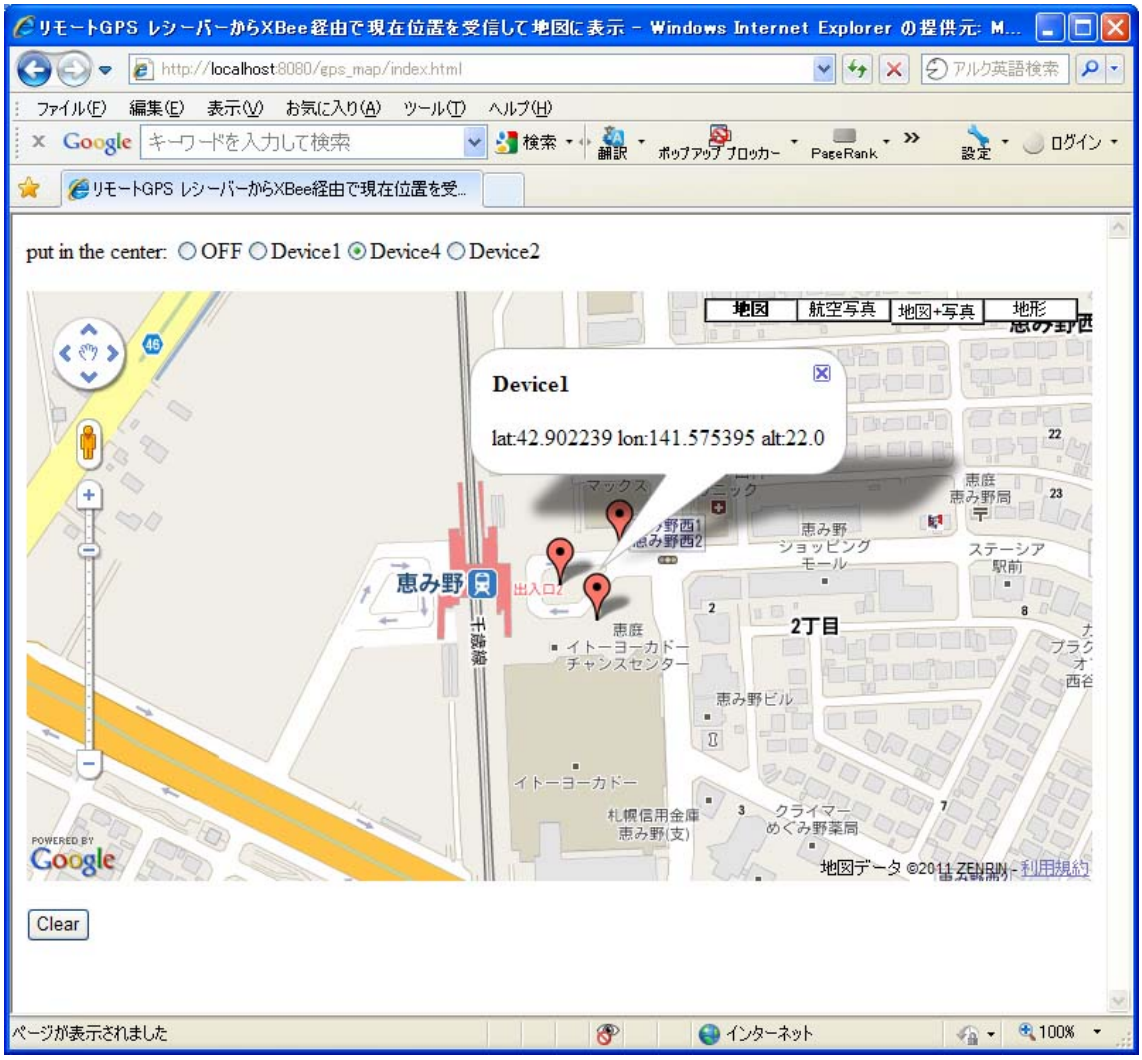
リモートデバイスから GPS イベント受信するとマップ上にマーカーが表示されて、デバイスが移動するとマップ上のマーカーも移動します。

ページ上部に GPS イベントを送信している全デバイス名のラジオボタンが表示されています。

このラジオボタンを選択することで、そのデバイスを中心にマップを表示し続けることができます。選択したデバイスが移動するとマップ全体が相対的に移動します。”OFF” を選択すると現在表示中のマップ位置はそのままで、マーカーが移動します。

マーカーをクリックするとそのデバイスが最後に送信した緯度、経度を InfoWindow に表示することが出来ます。

画面下の “Clear” ボタンを押すと全てのデバイス情報がクリアされて、同時にラジオボタンも削除されます。



(Device4 を中心にマップを表示、Device1 の緯度経度を同時に画面に表示)

7 このドキュメントについて

7.1 著作権および登録商標

7.2 連絡先

オールブルーシステム (All Blue System)

ウェブページ http://www.allbluesystem.com

メール contact@allbluesystem.com

7.3 このドキュメントの使用について

このドキュメントは、ABS-9000 DeviceServer の一般的な使用方法と応用例について解説してあります。お客様の個別の問題について、このドキュメントに記載された内容を実際のシステムに利用するときには、ここに記載されている以外にも考慮する事柄がありますので、ご注意ください。特に安全性やセキュリティ、長期間にわたる運用を想定してシステムを構築する必要があります。

オールブルーシステムでは ABS-9000 DeviceServer の使用や、このドキュメントに記載された内容を使用することによって、お客様及び第三者に損害を与えないことを保証しません。 ABS-9000 DeviceServer を使用したシステムを構築するときは、お客様の責任の下で、システムの構築と運用が行われるものとします。

8 更新履歴

REV A.1.0 2011/1/16

初版作成