Panasonic

PAN1760A

Bluetooth Low Energy Module

Software Guide

Rev. 1.0

Wireless Modules

By purchase of any of the products described in this document the customer accepts the document's validity and declares their agreement and understanding of its contents and recommendations. Panasonic Industrial Devices Europe GmbH (Panasonic) reserves the right to make changes as required at any time without notification.

This Software Guide does not lodge the claim to be complete and free of mistakes.

The information contained herein is presented only as guidance for Product use. No responsibility is assumed by Panasonic for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.

Description of hardware, software, and other information in this document is only intended to illustrate the functionality of the referred Panasonic product. It should not be construed as guaranteeing specific functionality of the product as described or suitable for a particular application.

Any provided (source) code shall not be used or incorporated into any products or systems whose manufacture, use or sale is prohibited under any applicable laws or regulations.

Any outlined or referenced (source) code within this document is provided on an “as is” basis without any right to technical support or updates and without warranty of any kind on a free of charge basis according to § 516 German Civil Law (BGB) including without limitation, any warranties or conditions of title, non-infringement, merchantability, or fitness for a particular purpose. Customer acknowledges that (source) code may bear defects and errors.

The third-party tools mentioned in this document are offered by independent third-party providers who are solely responsible for these products. Panasonic has no responsibility whatsoever for the performance, product descriptions, specifications, referenced content, or any and all claims or representations of these third-party providers. Panasonic makes no warranty whatsoever, neither express nor implied, with respect to the goods, the referenced contents, or any and all claims or representations of the third-party providers.

To the maximum extent allowable by Law Panasonic assumes no liability whatsoever including without limitation, indirect, consequential, special, or incidental damages or loss, including without limitation loss of profits, loss of opportunities, business interruption, and loss of data.

Table of Contents

1 About This Document

1.1 Purpose and Audience

This software guide is intended as a quick start guide and explains how to setup the PAN1760A USB stick, describes the basic usage modes and gives an introduction to the software that is provided.

The document is intended for software engineers.

1.2 Revision History

Revision	Date	Modifications/Remarks
1.0	25.10.2017	Initial version

1.3 Use of Symbols

Symbol	Description
	**Note** Indicates important information for the proper use of the product. Non-observance can lead to errors.
	**Attention** Indicates important notes that, if not observed, can put the product's functionality at risk.
⇨ [chapter number] [chapter title]	**Cross reference** Indicates cross references within the document. **Example:** Description of the symbols used in this document ⇨ 1.3 Use of Symbols.

1.4 Related Documents

[1] PAN1760A Design Guide

Please refer to the Panasonic website for more information as well as related documents ⇨ 8.2.2 Product Information.

[2] Tohsiba Bluetooth SDK developers guide

[3] Toshiba BLE AT command specification

2 Overview

The PAN1760A USB stick is a development platform for the Bluetooth Low Energy PAN1760A module.

It is based on the Bluetooth Low Energy chipset TC35678 from Toshiba.

For further information please visit ⇨ https://toshiba.semicon-storage.com/eu/product/wireless-communication/bluetooth/tc35678.html

Toshiba provides a Bluetooth software development kit (SDK) that is available after registration.

For further information please visit ⇨ https://apps.toshiba.de/web/SDKRegistration/

The Toshiba Bluetooth SDK includes a straight-forward serial cable replacement protocol called the SPPoverBLE profile.

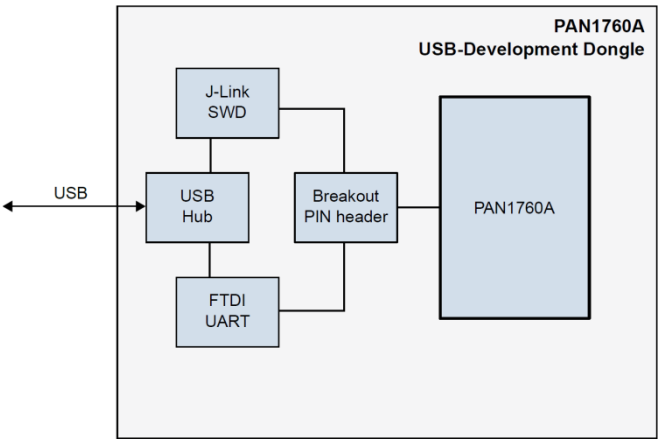
This software guide explains the usage of the SPPoverBLE profile in system configurations that are possible with the PAN1760A module in order to give customers a quick start for their designs.

The guide refers to the software package release version 1.0.

To download the latest version please visit
⇨ https://pideu.panasonic.de/products/bluetooth/PAN1760A-Bluetooth-Low-Energy-Module.html

Please refer to the Panasonic website for related documents ⇨ 8.2.2 Product Information.

3 Installation



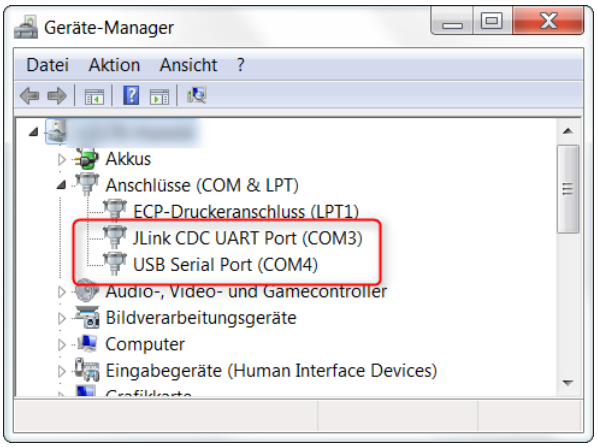
The PAN1760A USB stick consists of the following components:

- PAN1760A module
- PIN header breakout section
- *Segger J-Link* SWD debugger
- *FTDI* USB UART

It might be necessary to install drivers for some of the components.

3.1 Device drivers

Please note that both the *FTDI* USB UART and the *Segger J-Link* SWD debugger will provide a COM port to the system.



In this example only the COM port labeled as USB Serial Port (COM4) can be used to interface directly with the PAN1760A module.

3.1.1 *FTDI* USB UART

Depending on the operating system that is used, drivers for the *FTDI USB UART* might or might not be installed automatically.

Having the drivers installed correctly is mandatory for all the examples mentioned in this guide.

If in doubt, please check the FTDI website and install the drivers manually.

For further information please visit ⇨ http://www.ftdichip.com/Drivers/VCP.htm

3.1.2 *Segger J-Link* SWD debugger

Depending on the operating system that is used, drivers for *Segger J-Link* SWD debugger might or might not be installed automatically.

Having the drivers installed correctly is not strictly mandatory for the basic examples mentioned in this guide.

If in doubt, please check the *Segger* website and install the drivers manually.

For further information please visit ⇨ https://www.segger.com/downloads/jlink/

3.2 Toshiba Bluetooth SDK

Toshiba provides a software development kit (SDK) that is available after registration.

For further information please visit ⇨ https://apps.toshiba.de/web/SDKRegistration/

All examples shown in this document are based on the Bluetooth SDK version 3.2.0.1, so it is mandatory to have that version downloaded.

This document only explains the specific features of the PAN1760A module. If in doubt or if you need more details concerning the underlying chipset, please refer to the documentation from the Toshiba Bluetooth SDK.

The Bluetooth SDK contains extensive documentation as well, for example the *Bluetooth SDK developers guide* or the *BLE AT command specification*.

All documentation can be found in the directory
BT_SDK_v_3_2_0_1.zip\BT_SDK_v_3_2_0_1\documentation.zip

3.3 Software Package

This software guide refers to the software package version 1.0.

To download the latest version please visit
⇨ https://pideu.panasonic.de/products/bluetooth/PAN1760A-Bluetooth-Low-Energy-Module.html

The software package comes as a ZIP file and contains the following components:

- Software Guide (this file)
- Software Package ZIP file
- *Unzipper* tool

3.3.1 Software package ZIP file

This part contains all sample projects and the pre-compiled binaries that are referenced in this guide.

The sample project refers to certain files from the Toshiba Bluetooth SDK, so the SDK must be present in a specific location.

3.3.2 *Unzipper* tool

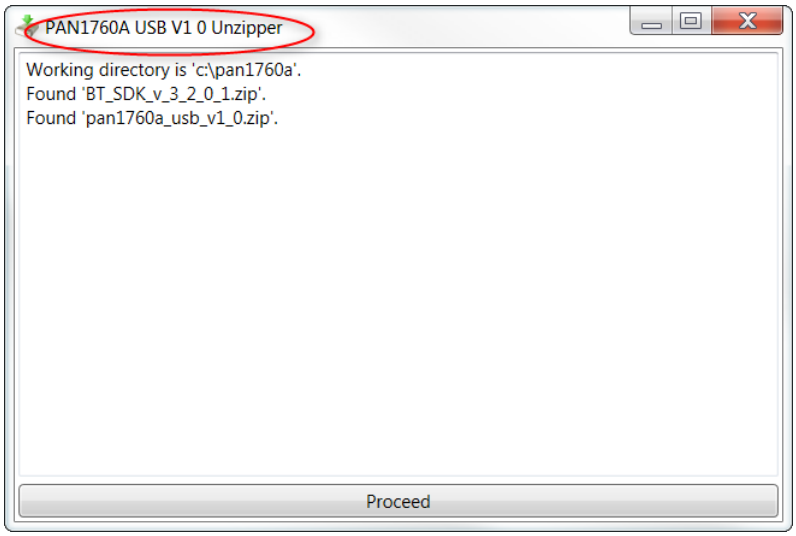
Setting up the correct directory layout is tedious and error prone. In order to facilitate this process, an *unzipper* tool is available that will execute all necessary steps in the correct order.

> For the rest of this document it is assumed that the installation of the software package was done to *c:\pan1760a*.
>
> If a different path was used, please make sure to adopt the path names in the examples accordingly.

For the *unzipper* to work correctly, the following files must be present in the same directory.

- *pan1760a_usb_v1_0_unzipper.exe*
- *pan1760a_usb_v1_0.zip*
- *BT_SDK_v_3_2_0_1.zip*

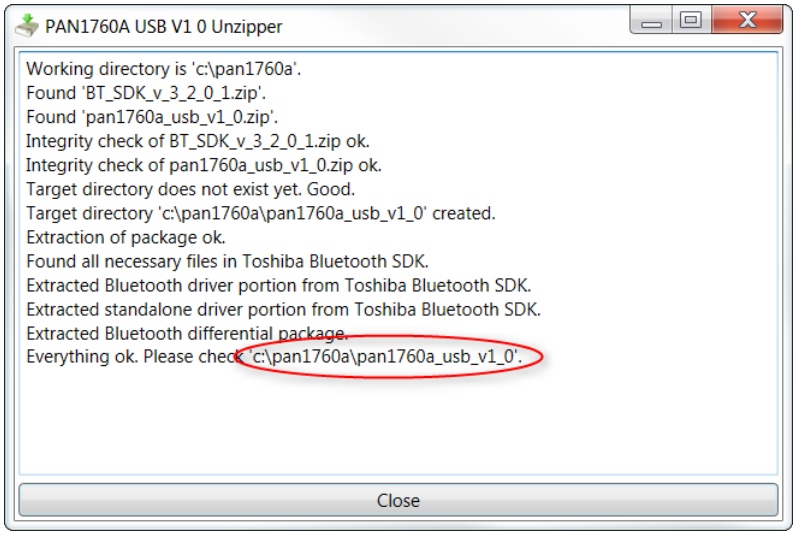
Afterwards *pan1760a_usb_v1_0_unzipper.exe* can be executed.



First make sure that version number of then *unzipper* matches the version number of the software package you want to process.

If any of the necessary files is not present, the tool will immediately display an error message.

Continue the process by clicking *Proceed*.

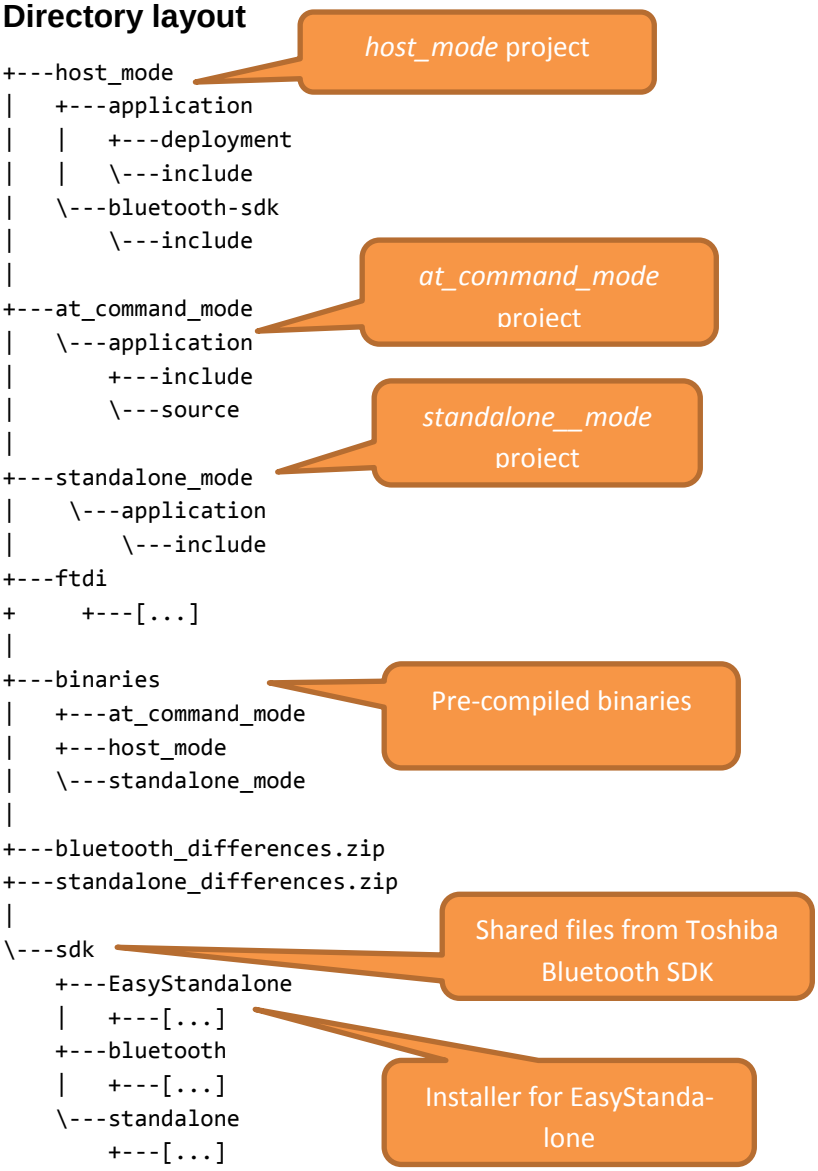


If all goes well the pre-configured software package will be present in the shown location afterwards.

Please note that the necessary parts of the Toshiba Bluetooth SDK have been copied to the *sdk* subdirectory.

The installer for the *EasyStandalone* tool has been extracted from the Toshiba Bluetooth SDK and copied to the *sdk* directory as well for later installation.

3.3.3 Directory layout




```
+---host_mode
|   +---application
|   |   +---deployment
|   |   \---include
|   \---bluetooth-sdk
|       \---include
|
+---at_command_mode
|   \---application
|       +---include
|       \---source
|
+---standalone_mode
|    \---application
|        \---include
+---ftdi
+     +---[...]
|
+---binaries
|   +---at_command_mode
|   +---host_mode
|   \---standalone_mode
|
+---bluetooth_differences.zip
+---standalone_differences.zip
|
\---sdk
    +---EasyStandalone
    |   +---[...]
    +---bluetooth
    |   +---[...]
    \---standalone
        +---[...]
```

The file `bluetooth_differences.zip` contains all files that are usually present in the Toshiba Bluetooth SDK, but have been modified for the projects in this package.

The same is true for the file `standalone_differences.zip`.

These changes are explained in ⇨ 8.1 Changes to the Toshiba Bluetooth SDK.

3.3.4 Manual installation

If due to whatever reasons the *unzipper* tool cannot be used, the following steps have to be done manually to create the necessary directory structure.

- Create a destination directory and change into it
- Unzip *pan1760a_usb_v1_0.zip*
- Create a *sdk* subdirectory and change into it
- Copy *bluetooth* directory from *BT_SDK_v_3_2_0_1.zip**BT_SDK_v_3_2_0_1**software.zip**software*\ *bluetooth_driver.zip**bluetooth_driver**bluetooth*\

- Go into *bluetooth* directory *and u*nzip *bluetooth_differences.zip* over existing files
- Change into *sdk* directory again
- Copy *standalone* directory from *BT_SDK_v_3_2_0_1.zip**BT_SDK_v_3_2_0_1**software.zip**software*\ *standalone_deployment.zip**standalone_deployment**standalone*\
- Go into *standalone* directory *and u*nzip *standalone_differences.zip* over existing files

3.4 Microsoft Visual Studio 2017

All the sample application projects that run on the PC are based on *Visual Studio 2017.*

It is sufficient to install the *Visual Studio 2017 Community* edition.

For further information please visit ⇨ https://www.visualstudio.com/downloads/

For the first steps it is **not** necessary to have *Visual Studio 2017* installed, because the software package comes with pre-compiled binaries that can be used for testing.

3.5 IAR Embedded Workbench

All the sample application projects that run on the embedded Cortex-M0 microcontroller are based on *IAR Embedded Workbench.*

Because the code sizes of any of the projects exceed 32kB you either need to have a licensed version or use the 30-day trial version.

For further information please visit ⇨ https://www.iar.com/iar-embedded-workbench/

For the first steps it is **not** necessary to have *IAR Embedded Workbench* installed, because the software package comes with pre-compiled binaries that can be used for testing.

4 Background information

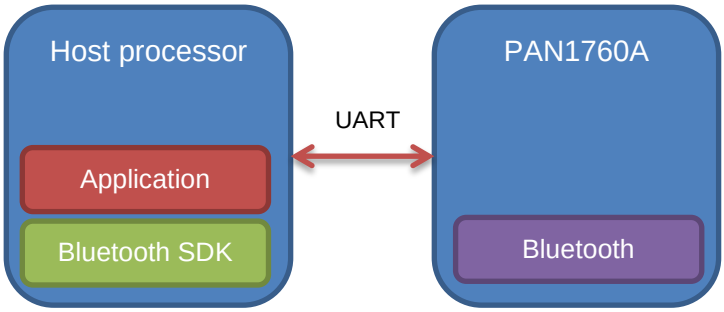
4.1 Operating Modes

The PAN1760A module allows different operating modes which make different product configurations possible.

The main operating modes are:

- Host mode
- AT command mode
- Standalone mode

4.1.1 Host mode

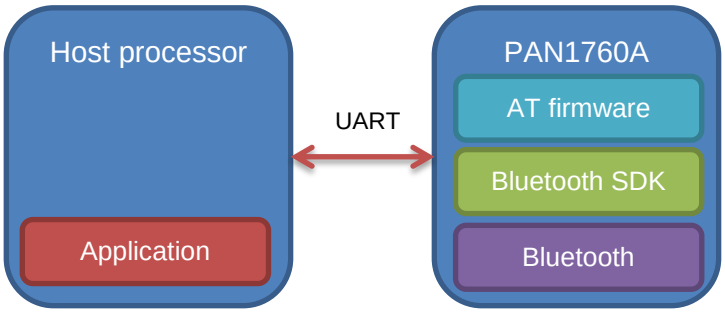


The host mode operating mode can be used in a dual chip product configuration, where the chips are connected via UART.

The PAN1760A module provides the necessary Bluetooth functionality to a host processor, which needs to run the Toshiba Bluetooth SDK. No software needs to be run on the PAN1760A module itself.

This setup gives the most flexibility to the system designer, but the host processor needs the necessary resources to run the Toshiba Bluetooth SDK.

4.1.2 AT Command Mode



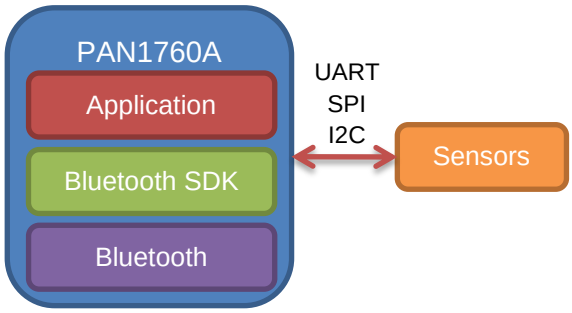
The AT command mode operating mode can be used in a dual chip product configuration as well and again the chips are connected via UART.

The PAN1760A module provides the necessary Bluetooth functionality to a host processor as well, but in a simplified form by using an *AT command* like interface.

The Toshiba Bluetooth SDK is run on the PAN1760A module inside a special AT command firmware.

This setup gives good flexibility to the system designer and the host processor does not need many resources.

4.1.3 Standalone Mode



The standalone operating mode is a single chip configuration, where all processing is done by the PAN1760A module itself.

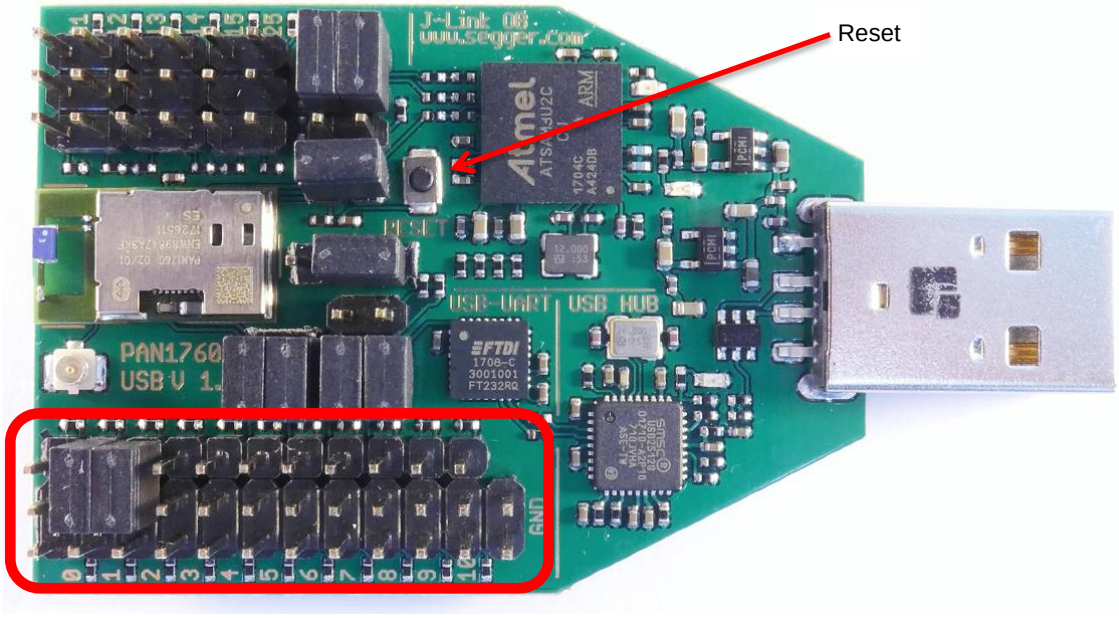
The Toshiba Bluetooth SDK as well as the customer application is run in the context of the Bluetooth processing.

The application has full access to the built-in peripherals and can use them for data input or output.

This setup provides the most cost-efficient solution for a new Bluetooth-enabled product.

4.2 Mode Selection

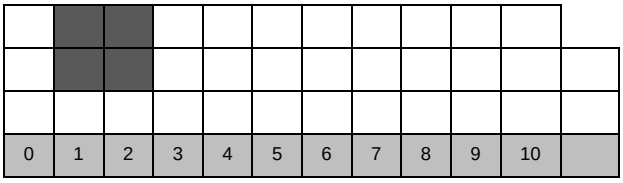
The operating mode of the PAN1760A module is determined by 2 GPIO jumpers in the PIN header breakout section.



Regardless of the mode to be used GPIO 2 must always be set in the upward position, connecting the GPIO to GND.

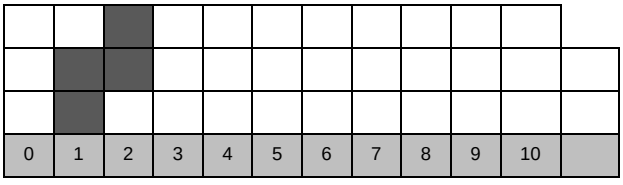
> When the mode has been changed the PAN1760A module needs to be reset by pressing the reset button.

4.2.1 Host Mode



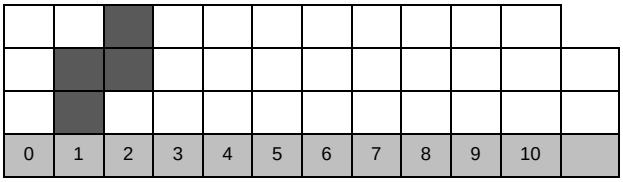
- Both GPIO jumpers must be set to the upward position, connecting the GPIOs to GND.

4.2.2 Standalone Mode



- GPIO jumper 1 must be set to the downward position, connecting GPIO 1 to VCC
- GPIO jumper 2 must be set to the upward position, connecting GPIO 2 to GND

4.2.3 AT Command Mode



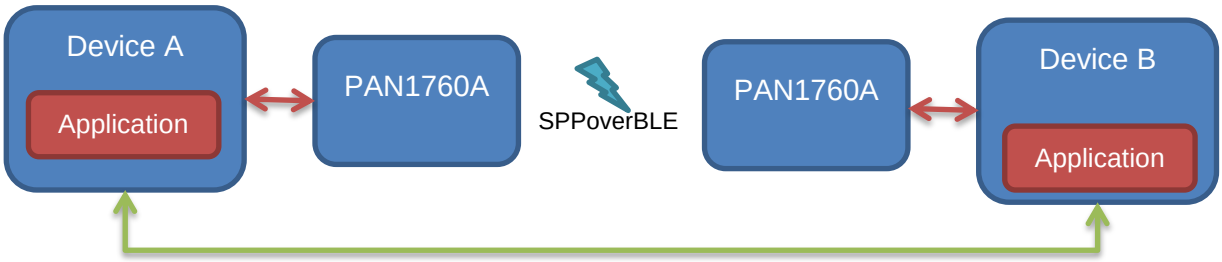
- GPIO jumper 1 must be set to the downward position, connecting GPIO 1 to VCC
- GPIO jumper 2 must be set to the upward position, connecting GPIO 2 to GND

Please note that the AT command firmware must be written to the in-device flash as explained in chapter ⇨ 6.1 Programming the AT Command Firmware

4.3 SPPoverBLE profile

The concept of Bluetooth Low Energy encourages developers to write custom and specific Bluetooth Low Energy profiles for their special hardware.

But even if the Bluetooth SDK in conjunction with the PAN1760A module makes this very easy, sometimes all you want to have is some sort of wireless serial cable replacement and transfer some data easily between two devices.



Virtual COM port connection

The Bluetooth SDK already contains such a wireless serial cable replacement profile which is called the SPPoverBLE profile.

If used on two devices A and B the SPPoverBLE profile basically provides a virtual COM port connection between these 2 devices.

For more information please check out the SPPoverBLE profile client and server specifications which are part of the Bluetooth SDK:

- BT_SDK_v_3_2_0_1.zip\BT_SDK_v_3_2_0_1\documentation.zip\documentation\client_profiles\spp_over_ble_profile.pdf
- BT_SDK_v_3_2_0_1.zip\BT_SDK_v_3_2_0_1\documentation.zip\documentation\server_profiles\spp_over_ble_profile.pdf

4.4 Bluetooth Low Energy Basics

The following Bluetooth Low energy basics will be briefly explained:

- Central and Peripheral device
- Master and slave
- Client and server

For more information please check out the Bluetooth Core Specification
⇨ https://www.bluetooth.com/specifications/bluetooth-core-specification

The *peripheral device* is the device which announces a service via advertising, while the *central device* discovers peripheral devices via scanning.

The *master* is the device which initiates a connection, while the *slave* device is accepting an incoming connection.

If a device is providing any information to a remote device it is acting as a *server*, while the other device using this information is called the *client*.

The most common setup is where a mobile phone acts as a *central device*, finds a peripheral device via scanning, then acts as a *master* to establish a connection and then works as a *client* to retrieve information.

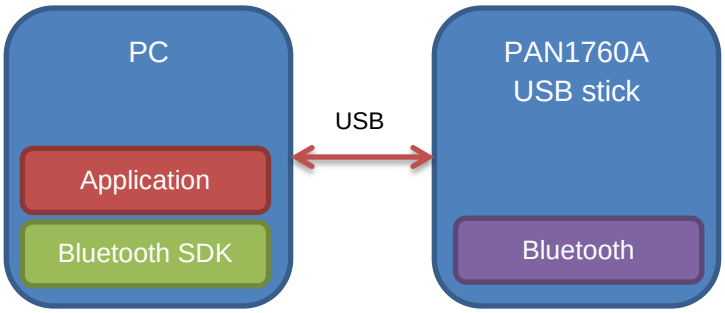
Then there might be a sensor device, which acts as a *peripheral device* and waits for an incoming connection while advertising. When a connection comes in, it acts as a *slave* and accepts the connection, then works as a *server* to provide its sensor information.

Please note that the device configuration does not need to be fixed during the runtime of a device.

Devices may freely change between being a peripheral device or a central device and then act as a master or a slave. When in a connection, each device can be both a server and a client and use the resources of the remote device freely.

5 Host Mode

With the PAN1760A USB stick a standard PC can act as the host processor.



The on-board *FTDI USB UART* will transparently convert the UART connection to a USB connection for easier use.

The Toshiba Bluetooth SDK has to be run in the context of the host processor.

5.1 EasyPAN

EasyPAN is a tool that lets you explore most of the features of the PAN1760A USB stick with a nice graphical user interface.

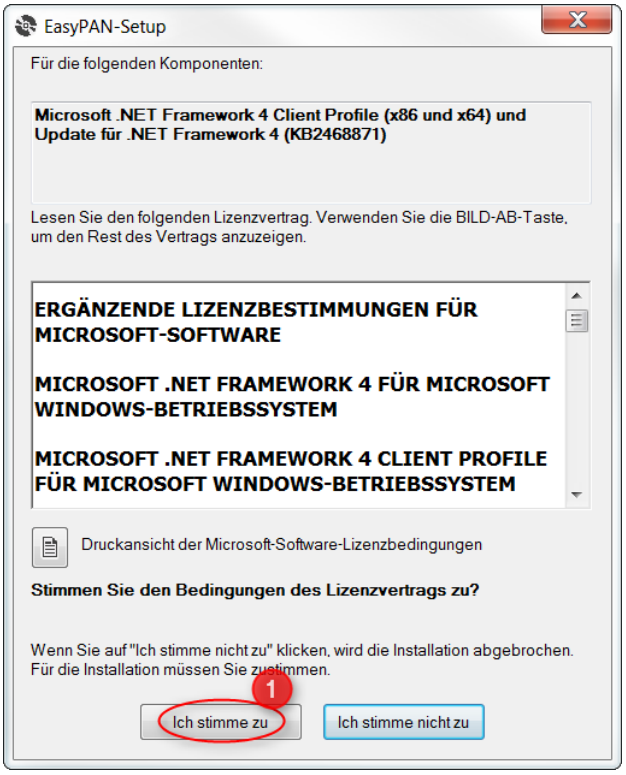
EasyPAN is available as a separate download from
⇨ https://pideu.panasonic.de/products/bluetooth/PAN1760A-Bluetooth-Low-Energy-Module.html

5.1.1 Installation

Please unzip the *EasyPAN* ZIP archive to a local folder on your PC, because installing *EasyPAN* from a network drive will not work.

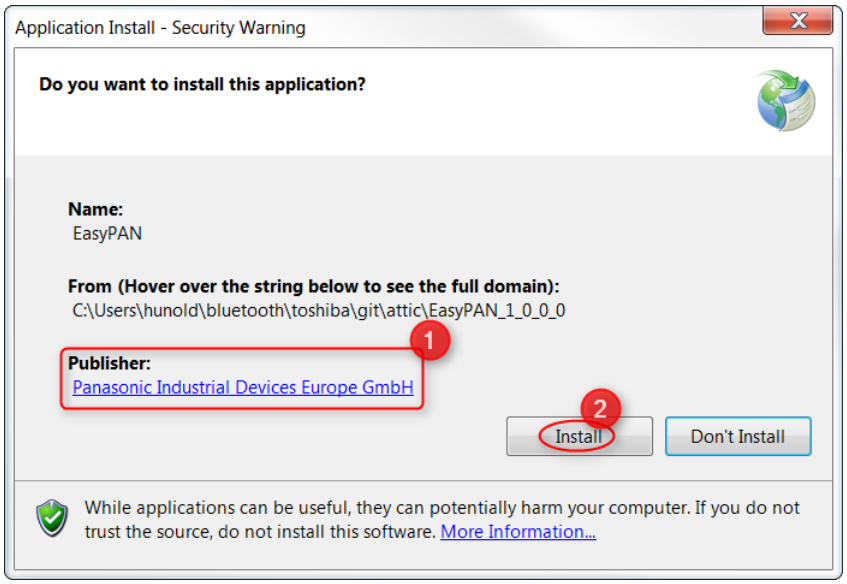
The password for the ZIP archive is *easypan*.

Afterwards please execute *setup.exe.*



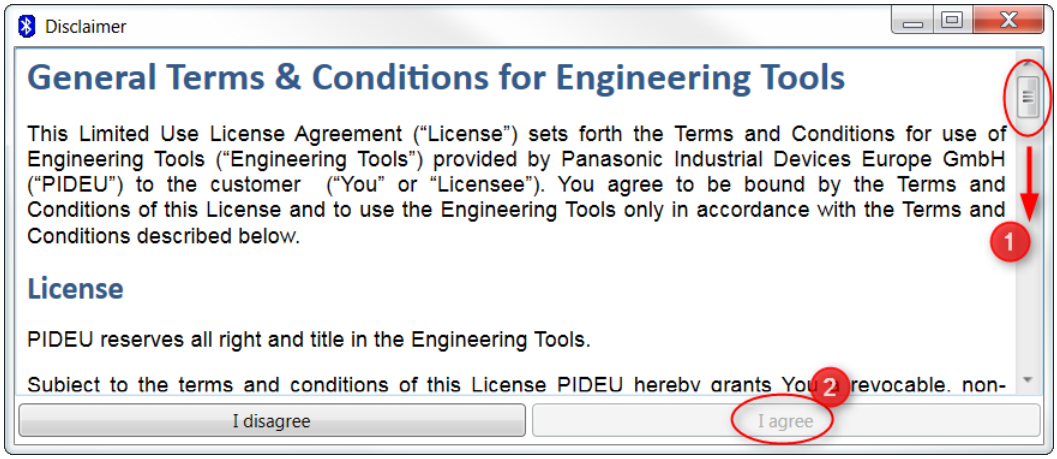
EasyPAN is based on the *Microsoft .NET framework 4 Client profile*. If this is not available on your PC the installer wants to install it automatically.

In this case please confirm the installation and wait until the installation is finished.



Please check the authenticity of the application by verifying the publisher.

Afterwards confirm the installation by clicking *Install.*



After the first start the *General Terms & Conditions for Engineering Tools* are shown. Please carefully check the agreement, scroll down and confirm by clicking *I Agree.*

If you don't agree and click *I disagree* then the application cannot be used. If you change your mind you can always start the application again from the *Start menu.*

5.1.2 Deinstallation

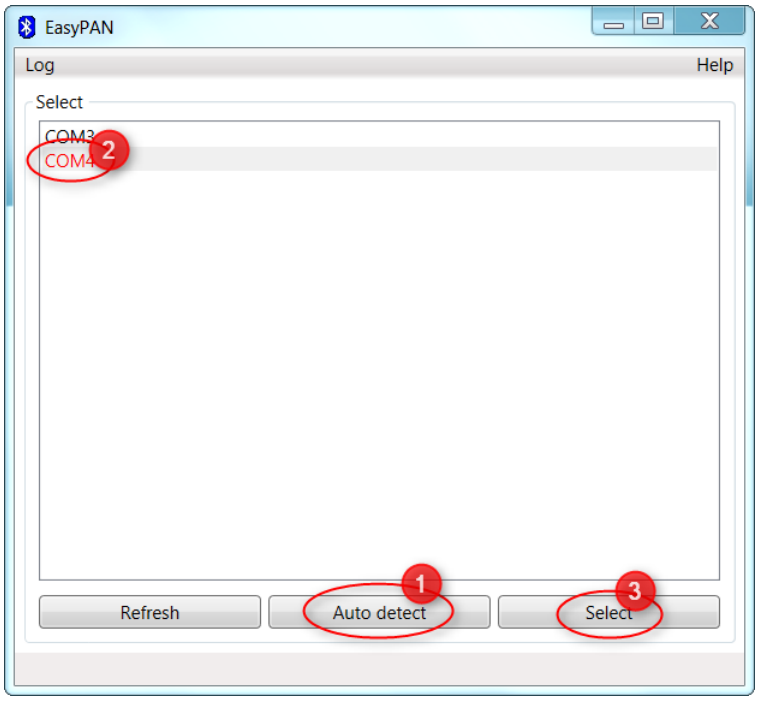
If you ever want to de-install *EasyPAN* you can do so by using the *Remove program* option in the *Program* section of the *Control Panel.*

5.1.3 Usage

In the following example a SPPoverBLE profile connection between two PAN1760A USB sticks shall be established using *EasyPAN.*

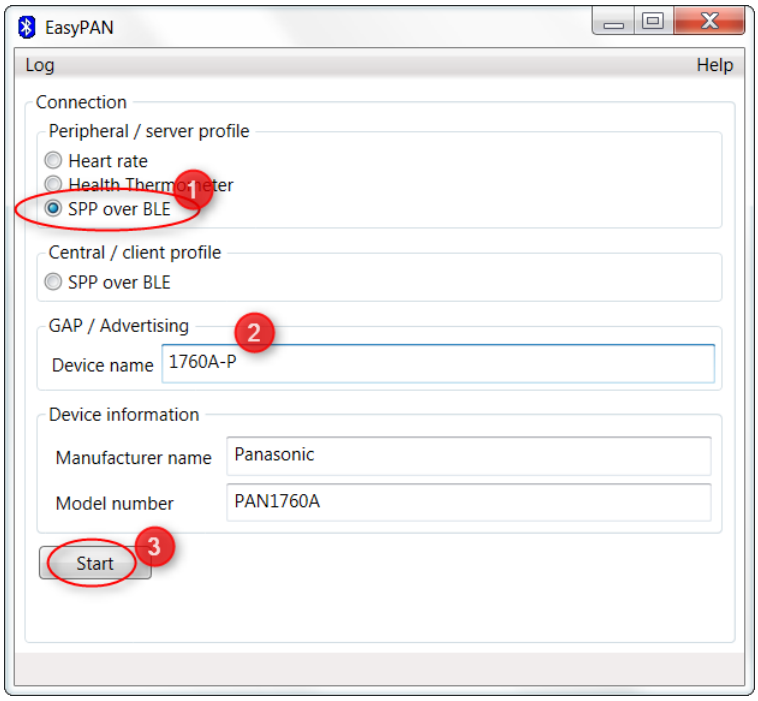
Please attach one of the PAN1760A USB sticks but make sure that it is jumpered to *host mode* as explained in ⇨ 0

Host Mode. Afterwards please launch *EasyPAN* once.



Please select the correct COM port as explained in ⇨ 3.1 Device drivers or use the *Auto detect* button to locate it.

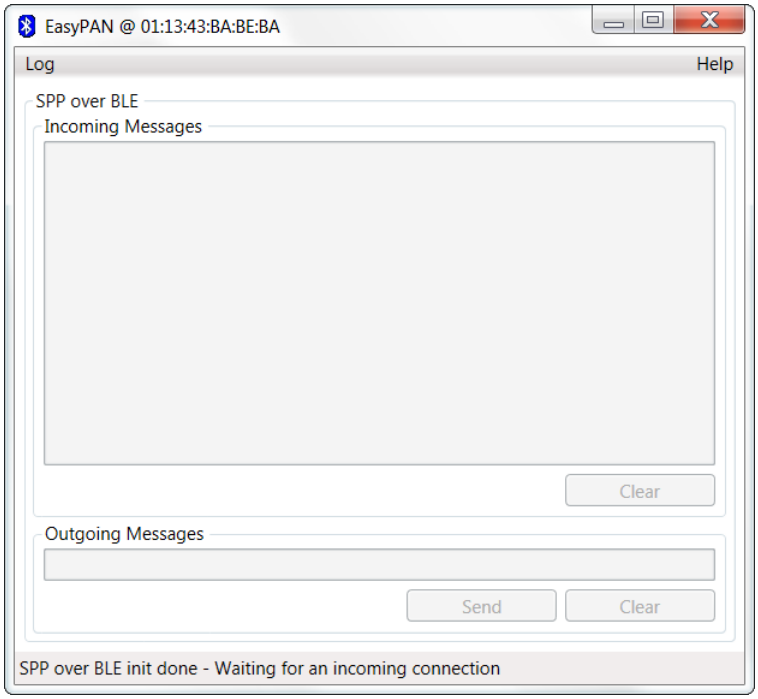
Afterwards select the COM port and proceed by clicking *Select.*



The first device will be configured as a *peripheral device*.

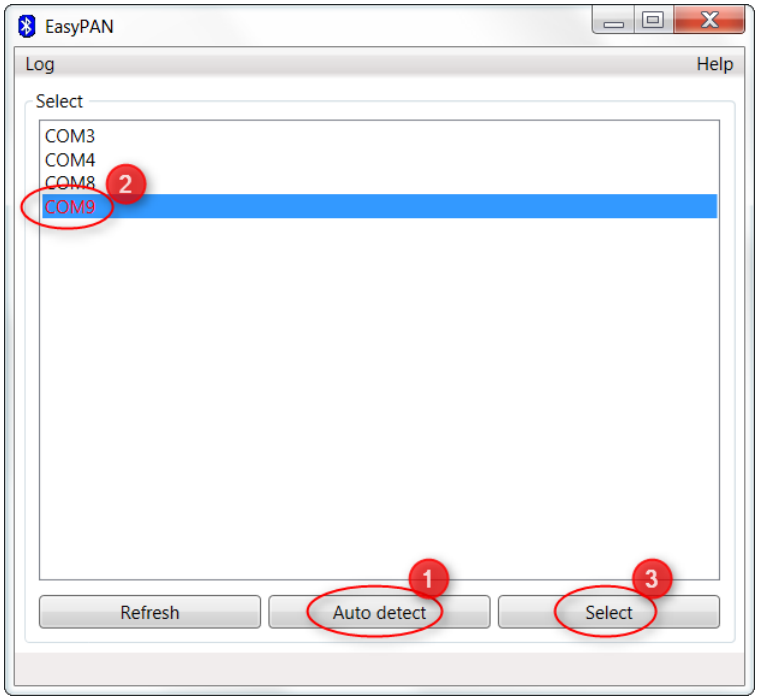
You can customize the configuration by changing the *Device name* used for *Advertising.*

By clicking the *Start* button the device will be initialized accordingly.



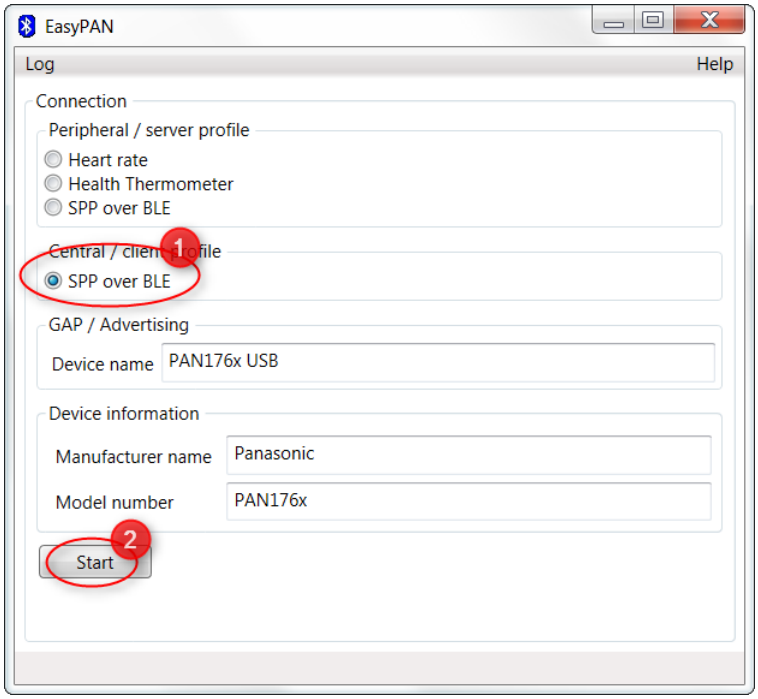
As long as no incoming connection is present all controls will be inaccessible.

Now attach the other PAN1760A USB stick and launch *EasyPAN* again.



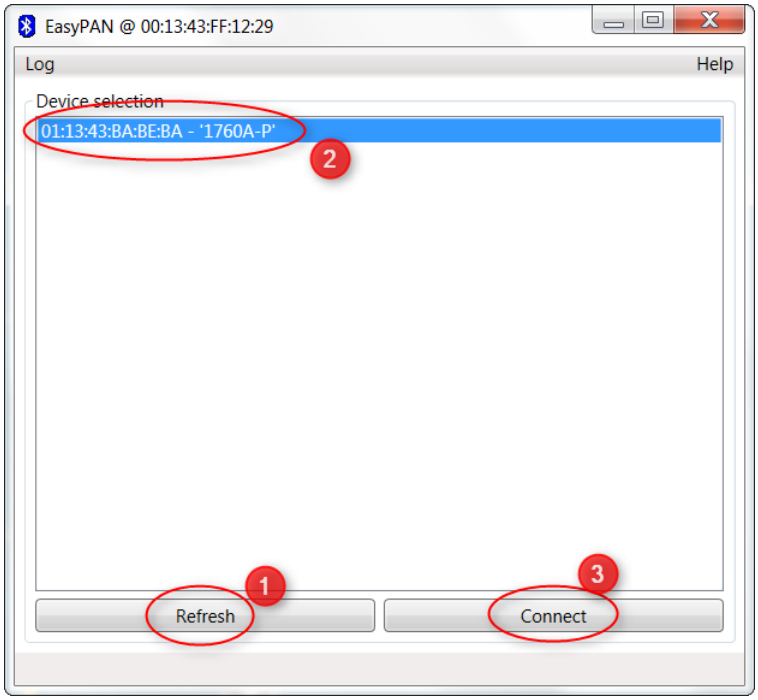
Again select the correct COM port for the new device as explained in ⇨ 3.1 Device drivers or use the *Auto detect* button to locate it.

Afterwards select the COM port and proceed by clicking *Select.*



This device will now be configured as a *central device*.

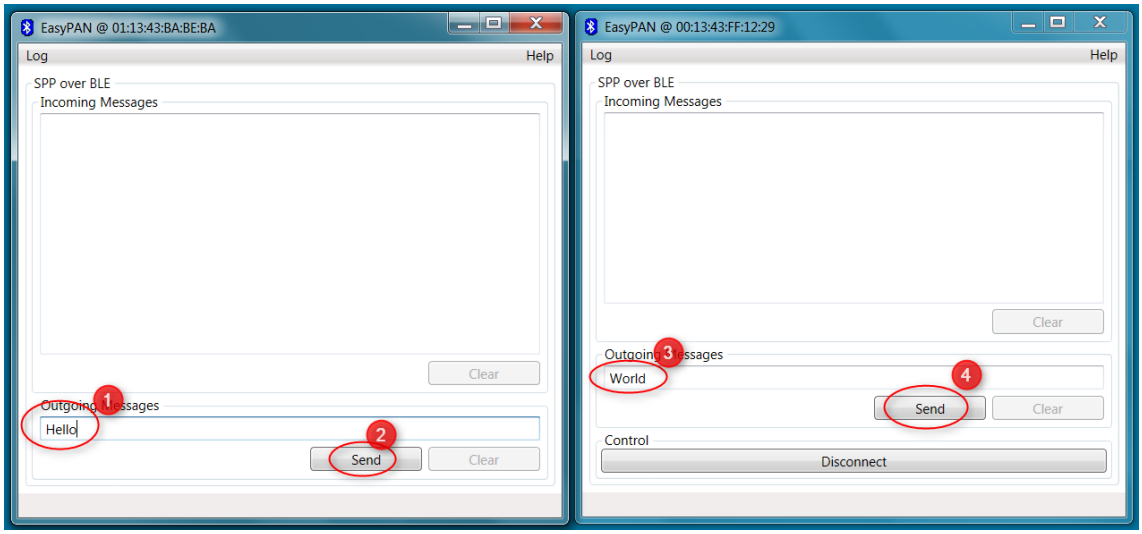
By clicking the *Start* button the device will be initialized accordingly.



The PAN1760A USB stick will immediately scan the environment for peripheral devices and show them in the *Device selection* list.

If in doubt, please use the *Refresh* button to clear the list of found devices and restart the scanning.

If the other device has been found, please select it in the *Device selection* list and click the *Connect* button.



Data can be send now back and forth by entering it into the *Outgoing messages* box and clicking the *Send* button.

The received data will be displayed in the *Incoming messages* section.

The central device acting as a master may terminate the connection – this is done by pressing the *Disconnect* button.

After the connection has been terminated the devices will return to their previous states: the peripheral device will start to advertise again while the central device will scan for peripheral devices.

5.2 PC Application

EasyPAN is a great tool to make the first steps with the PAN1760A USB sticks.

But it cannot be used as a starting point for custom projects, because it has a graphical user interface and is written in a mixture of C# and C. The source code is not available for reference either.

Because of this the software package includes a sample project called *host_mode* that shows how to use the Bluetooth SDK in a plain C application.

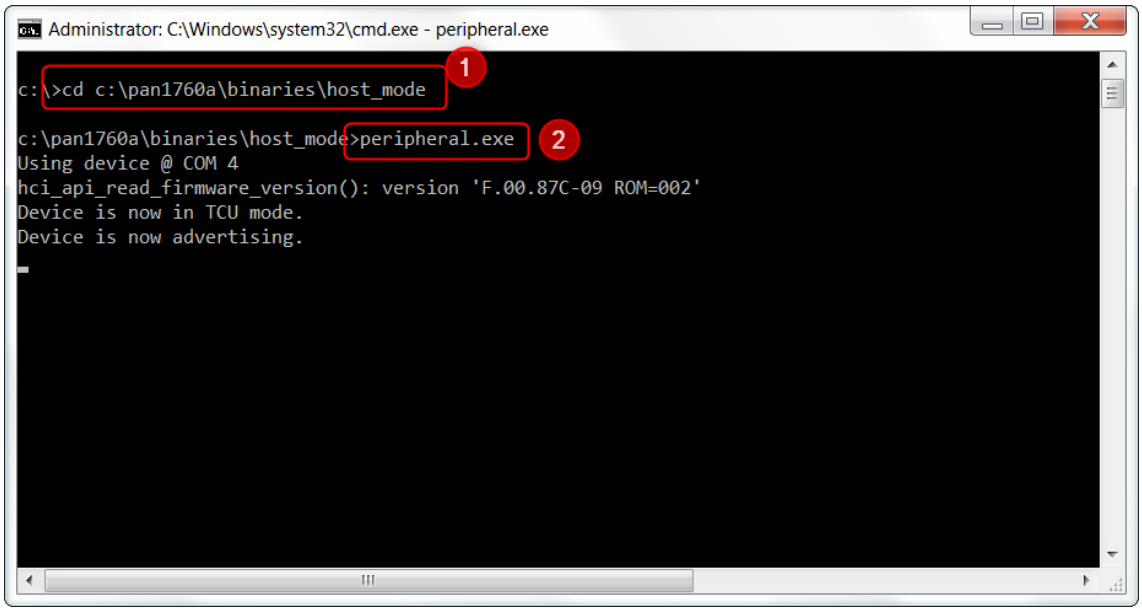
It showcases the use of the SPPoverBLE profile and comes in 2 project configurations: one will compile an application acting as a central device the other will compile an application acting as a peripheral device.

The software package includes pre-compiled binaries that can be used for testing and are available under the *binaries/host_mode* subdirectory in the installation directory.

Please remove all of the attached PAN1760A USB sticks.

Make sure that all the PAN1760A USB sticks are jumpered to *mode* as explained in ⇨ 0 Host Mode.

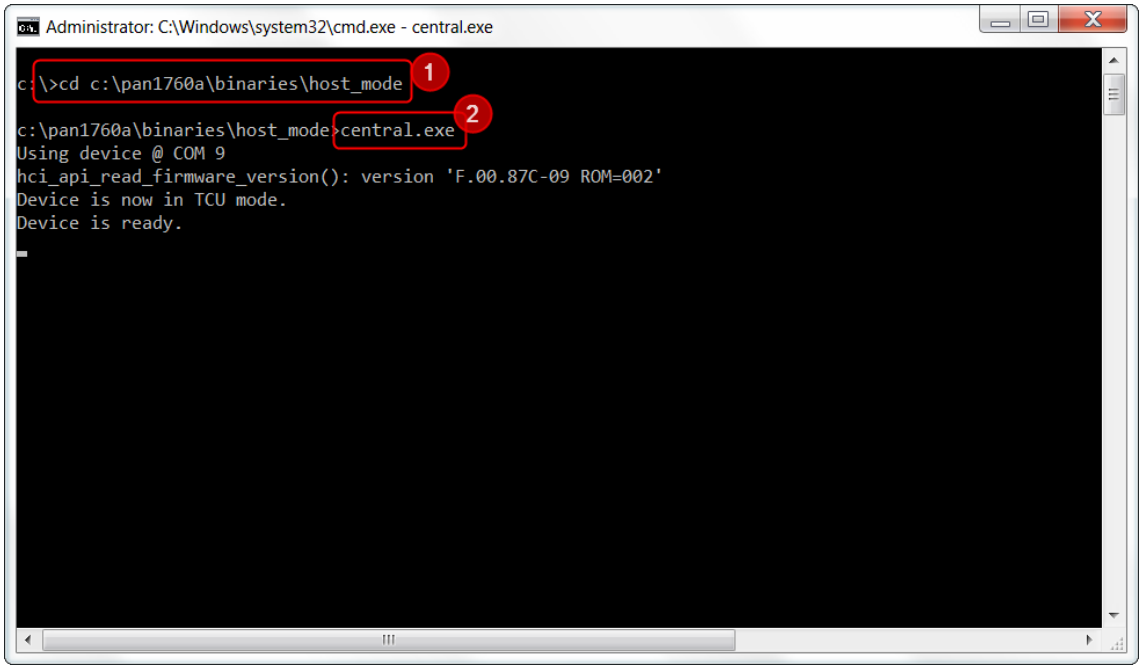
Now attach one of the PAN1760A USB sticks and launch a Windows command line prompt, for example by executing *cmd.exe* from the *Start menu.*



Navigate to the *binaries\host_mode* folder with the *cd* command and execute *peripheral.exe.*

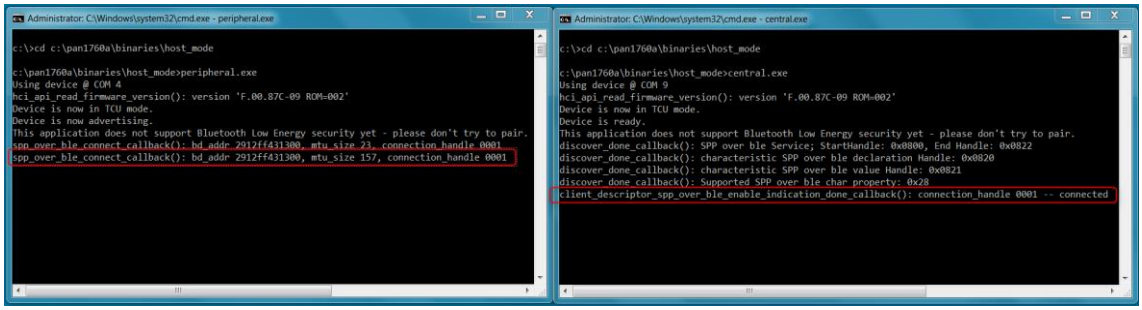
This PAN1760A USB stick will now act as a peripheral device.

Now attach the other PAN1760A USB stick and launch another Windows command line prompt.

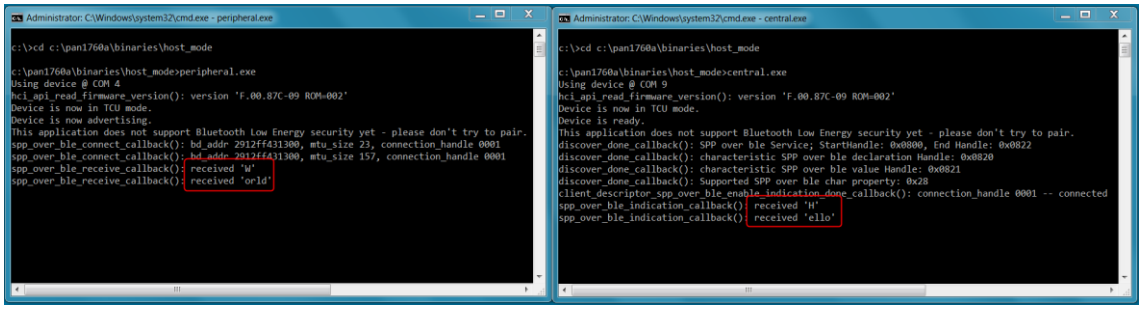


Navigate to the *binaries\host_mode* folder with the *cd* command and execute *central.exe.*

This PAN1760A USB stick will now act as a central device.

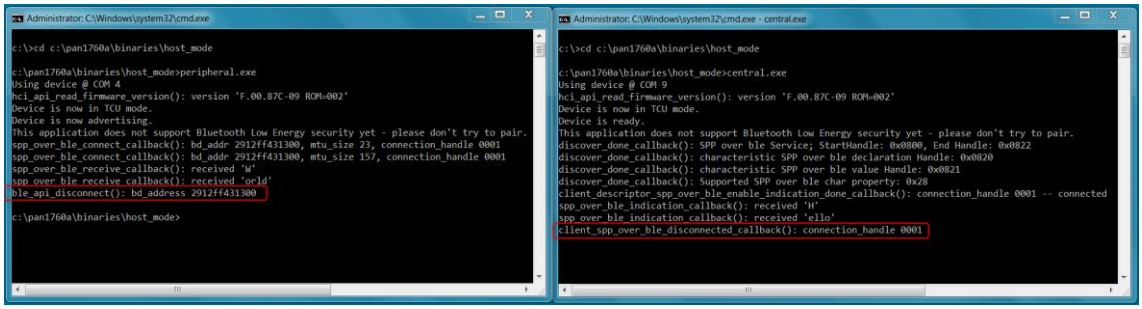


The central device will immediately recognize the peripheral device and establish a connection.



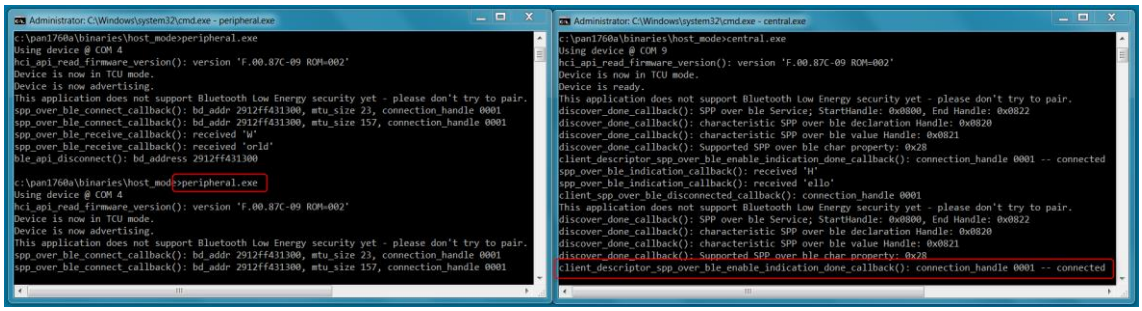
Afterwards, data can be transferred back and forth by simply pressing key in the respective terminal window.

In this example the one side sent *Hello*, while the other side sent *World.*



Any application can be terminated by pressing CTRL-C. The connection between the 2 devices will be shut down accordingly.

If you want to restart the demo after pressing CTRL-C, make sure to press the reset button on the corresponding PAN1760A USB stick.



In this example, *peripheral.exe* was interrupted by pressing CTRL-C and the connection between the two devices was terminated.

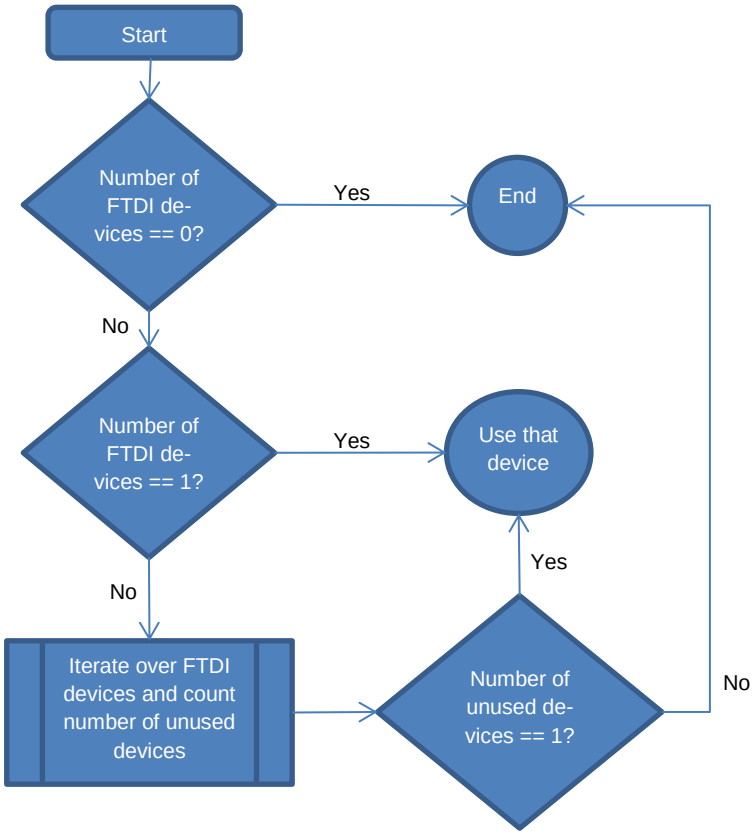
Afterwards the corresponding PAN1760A USB stick was reset by pressing the reset button.

Then *peripheral.exe* was started again and the connection was immediately established again.

5.3 COM Port Handling

The sample applications seem to magically find out which COM port they should use.

In order to make development easier, the applications don't require that a COM port is given and instead use the following heuristic to find the COM port they should use:



If you just use one PAN1760A USB stick at a time, then this scheme will always find the correct COM port to use.

If you experiment with 2 or more PAN1760A USB sticks, then you can reset and restart any of the devices at any time, as long as you keep the application on all the other PAN1760A USB sticks running.

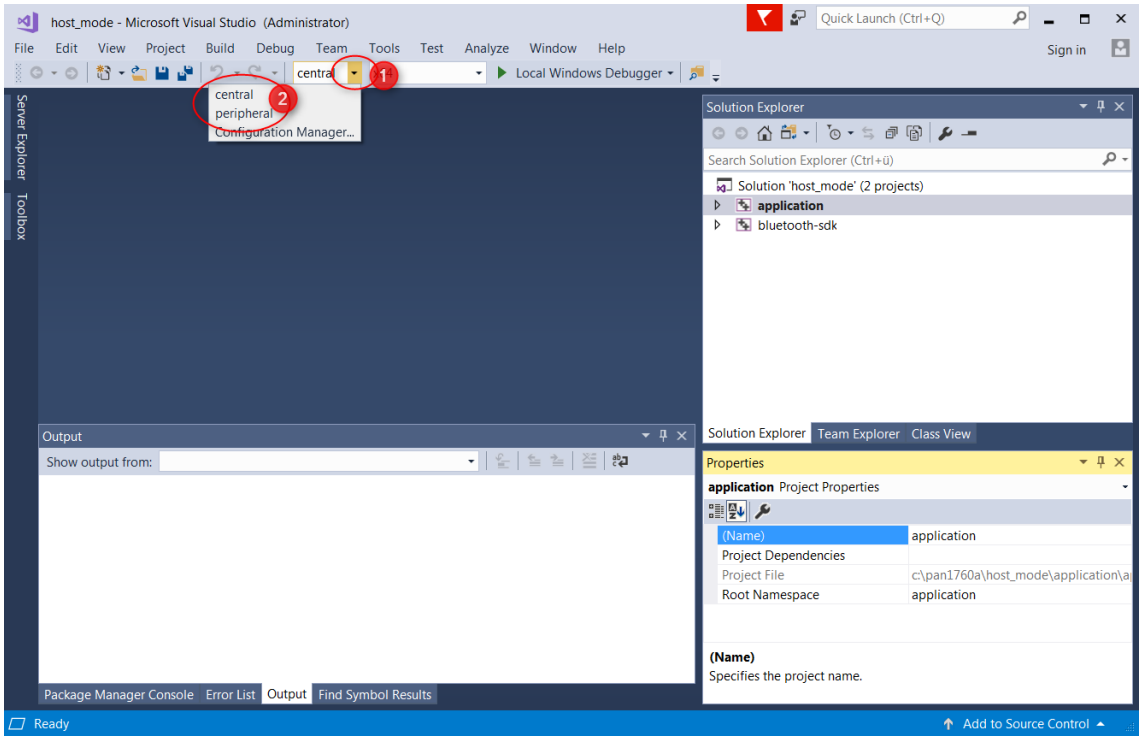
The benefit of this approach is that you don't have to remember any of the COM ports as long as you attach the PAN1760A USB sticks and start the applications one after another.

The downside of this approach is that if you want to start from scratch and restart all the applications, you may need to detach one of the PAN1760A USB sticks, because more than one unused FTDI device will be found by the scheme.

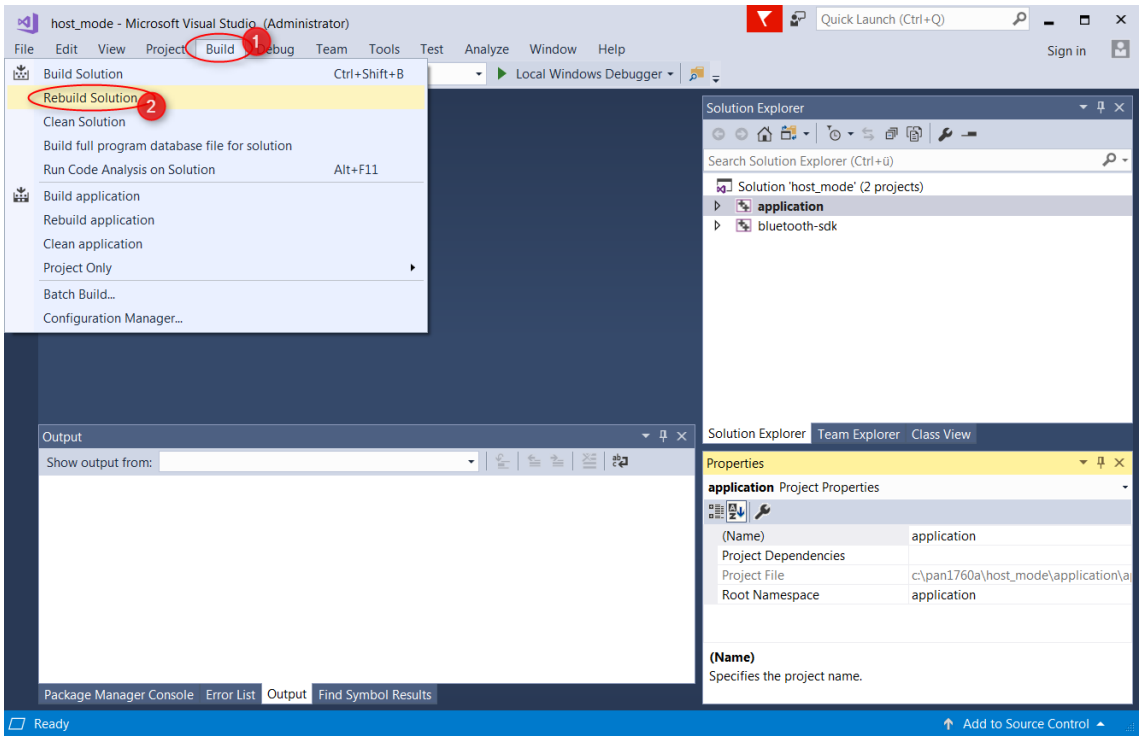
5.4 Recompiling the Sample Project

In order to recompile the host mode sample project it is necessary to have *Visual Studio 2017* installed as explained in ⇨ 3.4 Microsoft Visual Studio 2017.

The sample project solution file is located under *host_mode\host_mode.sln* and needs to be opened in *Visual Studio 2017.*



The configuration can be selected using the *Solution Configurations* drop down list.

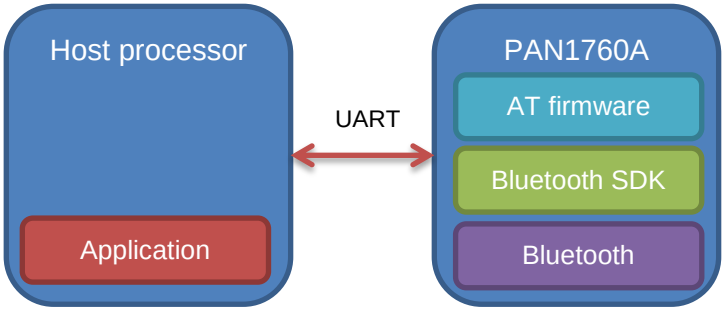


Afterwards *Rebuild Solution* from the *Build* menu can be used to build the currently selected configuration.

The resulting binaries can be found in the *host_mode\x64* directory as *peripheral.exe* and *central.exe.*

6 AT Command Mode

With the PAN1760A USB stick a standard PC can act as the host processor.



The on-board *FTDI USB UART* will transparently convert the UART connection to a USB connection for easier use.

The Toshiba Bluetooth SDK will be run in the context of the PAN1760A module as part of the AT command firmware.

The application on the host processor will use the AT command interface.

6.1 Programming the AT Command Firmware

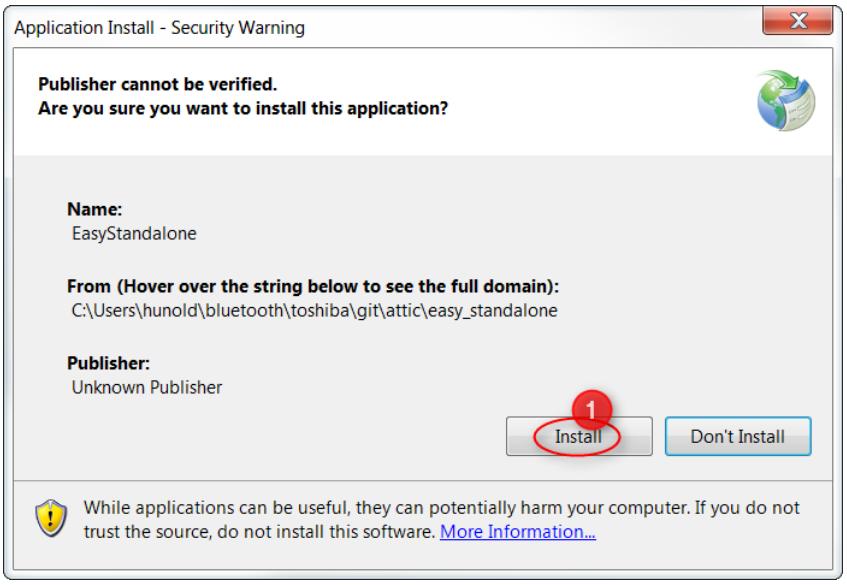
The AT command firmware is a regular PAN1760A module firmware that needs to be programmed.

For programming the Toshiba Bluetooth SDK includes the *EasyStandalone* application which can be used to write a firmware to the in-chip flash memory.

6.1.1 Installing *EasyStandalone*

If the *unzipper* tool has been used as explained in ⇨ 3.3.2 Unzipper tool then the installer for the *EasyStandalone* tool will be present in the *sdk\EasyStandalone* directory.

Please execute *setup.exe* from the *sdk\EasyStandalone* directory.



Continue with the installation by clicking *Install.*

After the installation has finished *EasyStandalone* will start up automatically and is available from the *Start menu* as well.

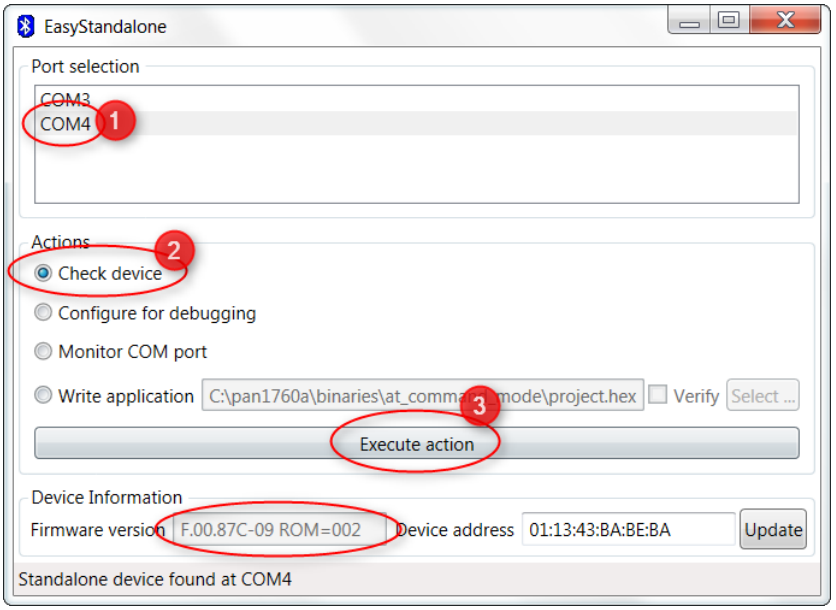
6.1.2 Programming

> *EasyStandalone* in explained in more detail in the chapter *APPENDIX: Working with EasyStandalone Tool* in the *Bluetooth SDK developers guide*.

Please remove all of the attached PAN1760A USB sticks.

Make sure that all the PAN1760A USB sticks are jumpered to *host mode* as explained in ⇨ 0 Host Mode.

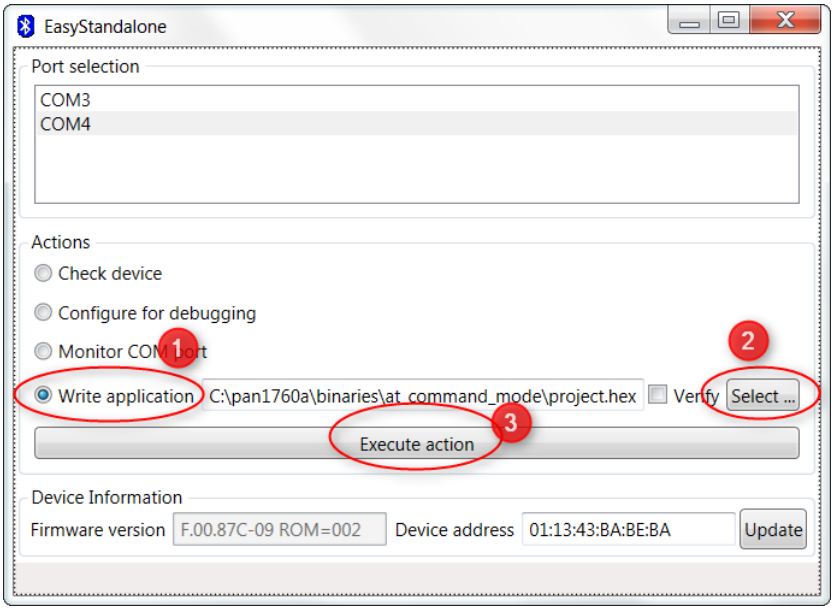
Now attach one of the PAN1760A USB sticks and launch *EasyStandalone* from the *Start menu.*



EasyStandalone does not have an automatic detection, so you need to make sure that the COM port selection is correct.

Select a COM port from the list of devices in the *Port selection* list, then choose *Check device* and press *Execute action.*

If all goes well, then the firmware version and the Bluetooth device address will be displayed.



Next please choose *Write application* and use the *Select…* button to choose the *project.hex* AT command firmware.

The AT command firmware has been copied from the Toshiba Bluetooth SDK to the *binaries\at_command_mode* folder.

Then choose *Execute action* to start the programming. A progress bar will be shown and after a couple of seconds the programming will be finished.

Now *EasyStandalone* can be closed.

Then jumper the PAN1760A USB stick to *AT command mode* as explained in ⇨ 4.2.3 AT Command Mode.

Repeat the same steps with the other PAN1760A USB stick.

6.2 PC Application

The software package includes a sample project called *at_command_mode* that shows how to use the AT command firmware in a plain C application.

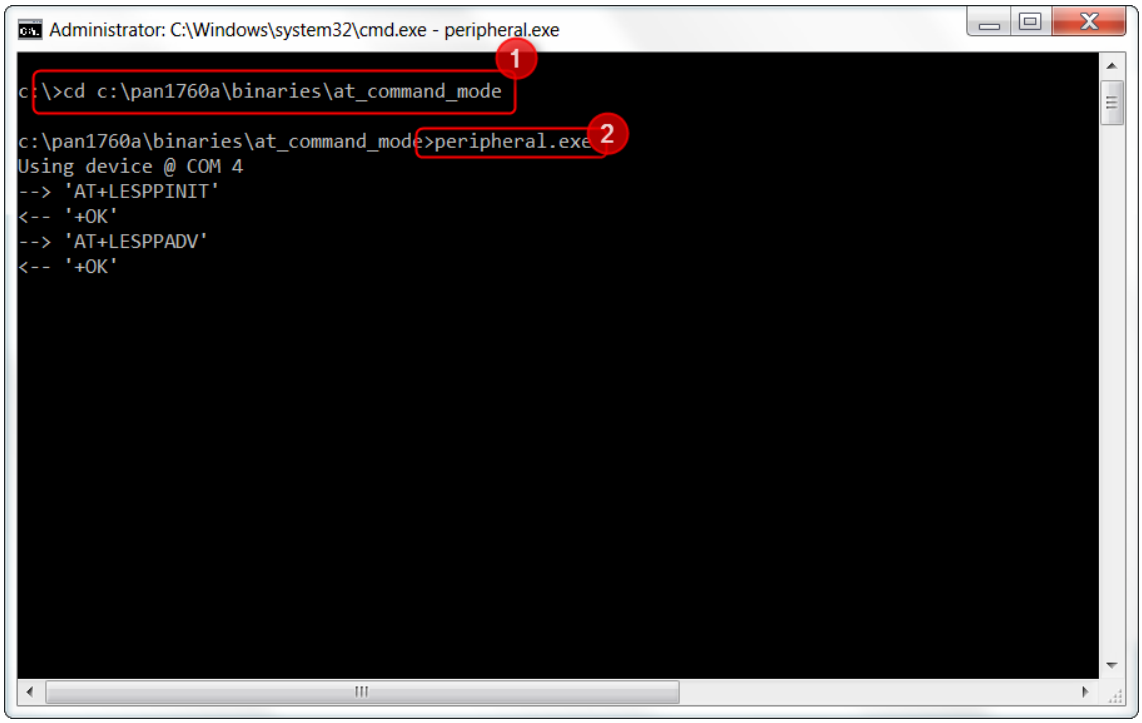
It showcases the use of the SPPoverBLE profile and comes in 2 project configurations: one will compile an application acting as a central device the other will compile an application acting as a peripheral device.

The software package includes pre-compiled binaries that can be used for testing and are available under the *binaries/at_command_mode* subdirectory in the installation directory.

Please remove all of the attached PAN1760A USB sticks.

Make sure that all the PAN1760A USB sticks are jumpered to *AT command mode* as explained in ⇨ 4.2.3 AT Command Mode

Now attach one of the PAN1760A USB sticks and launch a Windows command line prompt, for example by executing *cmd.exe* from the *Start menu.*



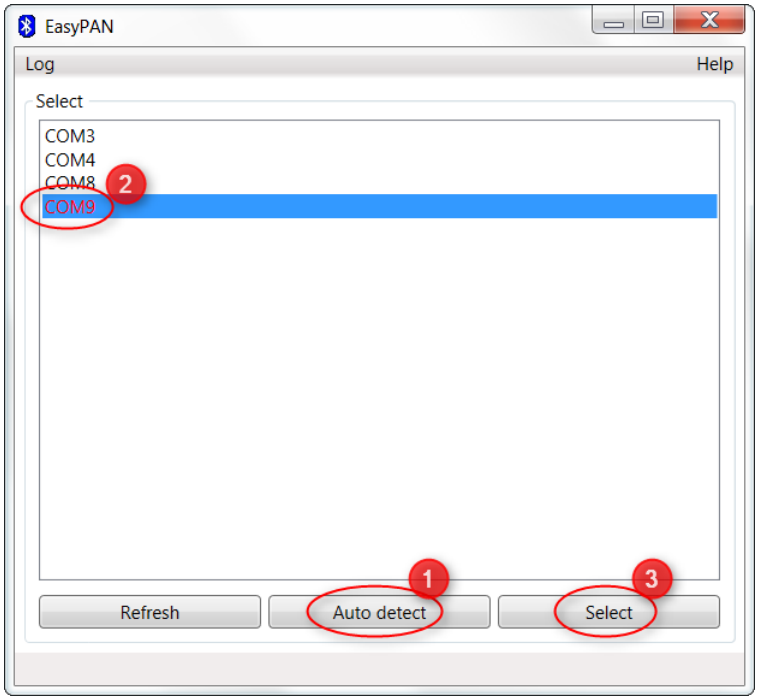
Navigate to the *binaries\at_command_mode* folder with the *cd* command and execute *peripheral.exe.*

This PAN1760A USB stick will now act as a peripheral device, but using the AT command interface.

The AT command firmware in the Toshiba Bluetooth SDK does not support central mode yet. For the time being, we use *EasyPAN* to act as the central device instead.

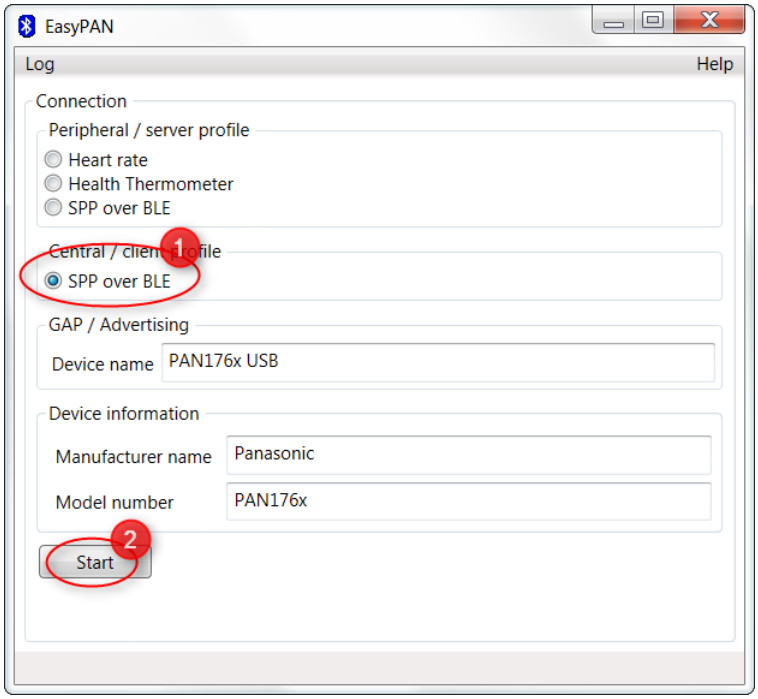
Now take the other PAN1760A USB stick and make sure it is jumpered to *host mode* as explained in ⇨ 0

Host Mode. Then launch *EasyPAN*.



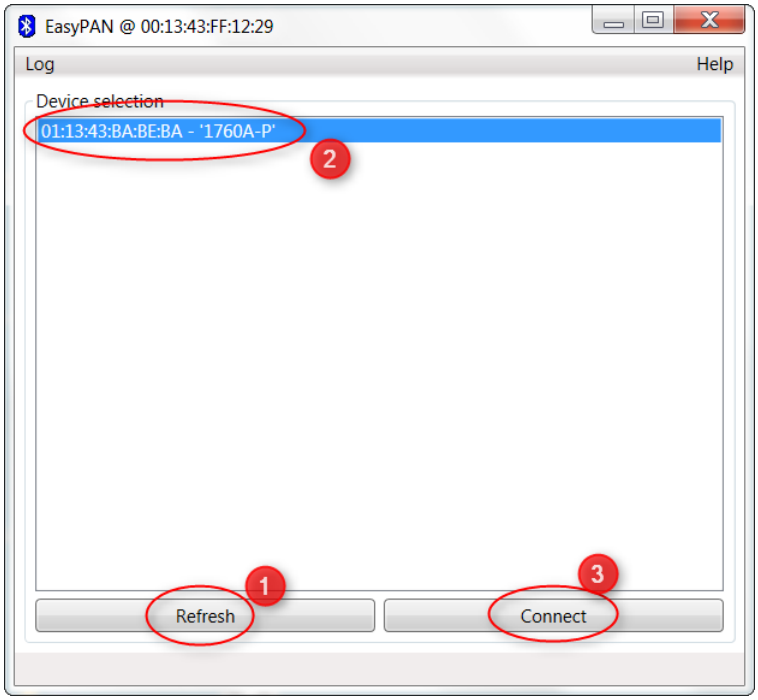
Select the correct COM port for the new device as explained in ⇨ 3.1 Device drivers or use the *Auto detect* button to locate it.

Afterwards select the COM port and proceed by clicking *Select.*



This device will now be configured as a *central device*.

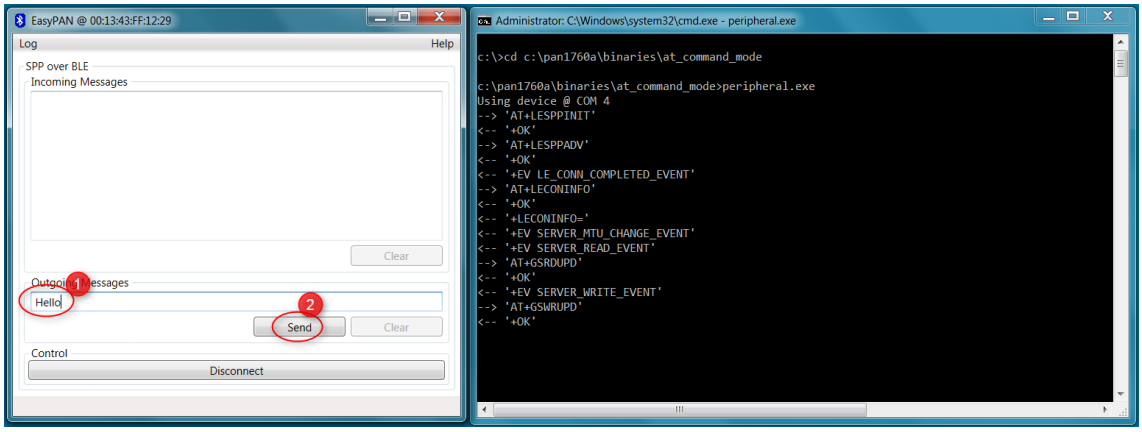
By clicking the *Start* button the device will be initialized accordingly.



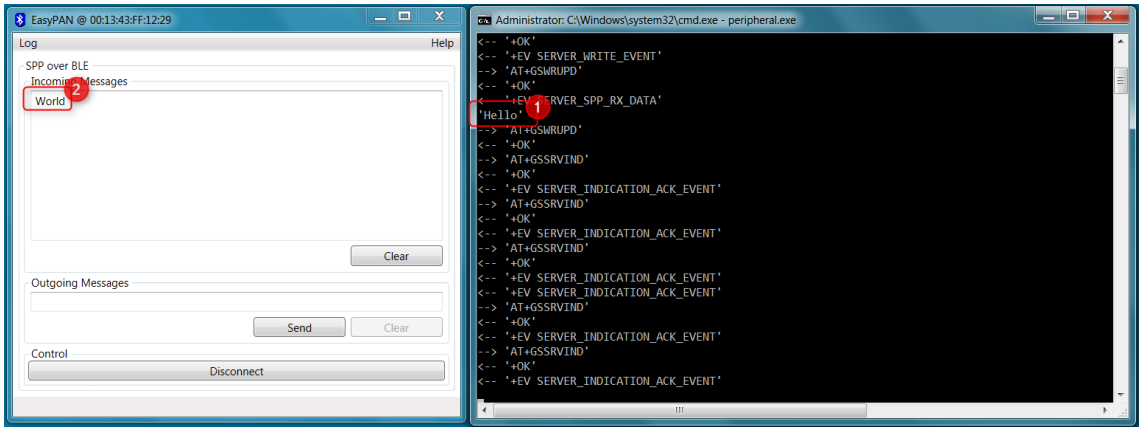
The PAN1760A USB stick will immediately scan the environment for peripheral devices and show them in the *Device selection* list.

If in doubt, please use the *Refresh* button to clear the list of found devices and restart the scanning.

If the other device has been found, please select it in the *Device selection* list and click the *Connect* button.



After the connection has been established, data can be sent back and forth.



From the central to the peripheral side using *EasyPAN* just enter some data into the *Outgoing messages* textbox and press *Send* to send the data.

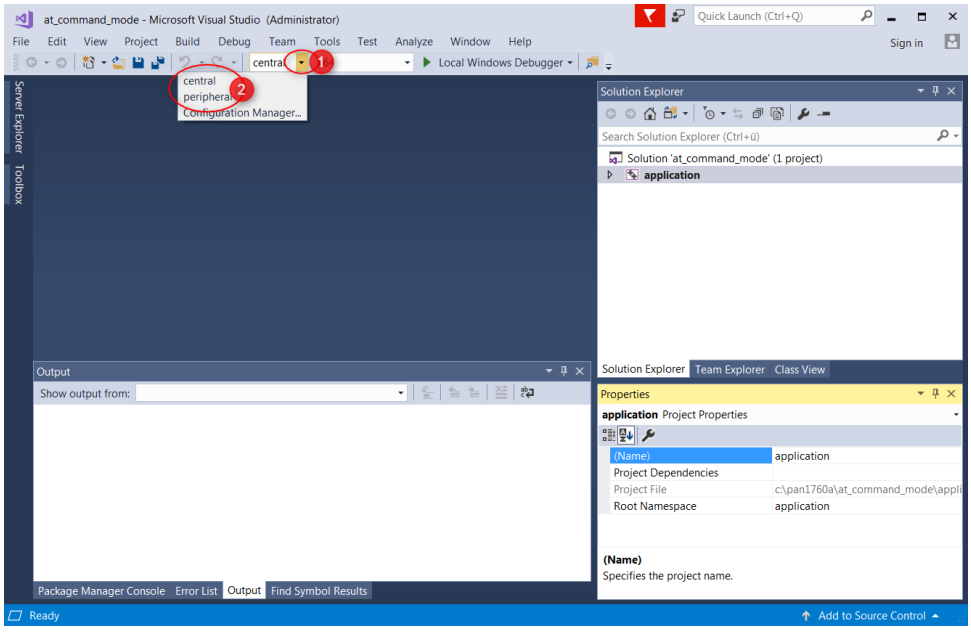
From the peripheral side, just press any keys inside the terminal window.

The AT command mode sample application uses the sample COM port handling as explained in ⇨ 5.3 COM Port Handling

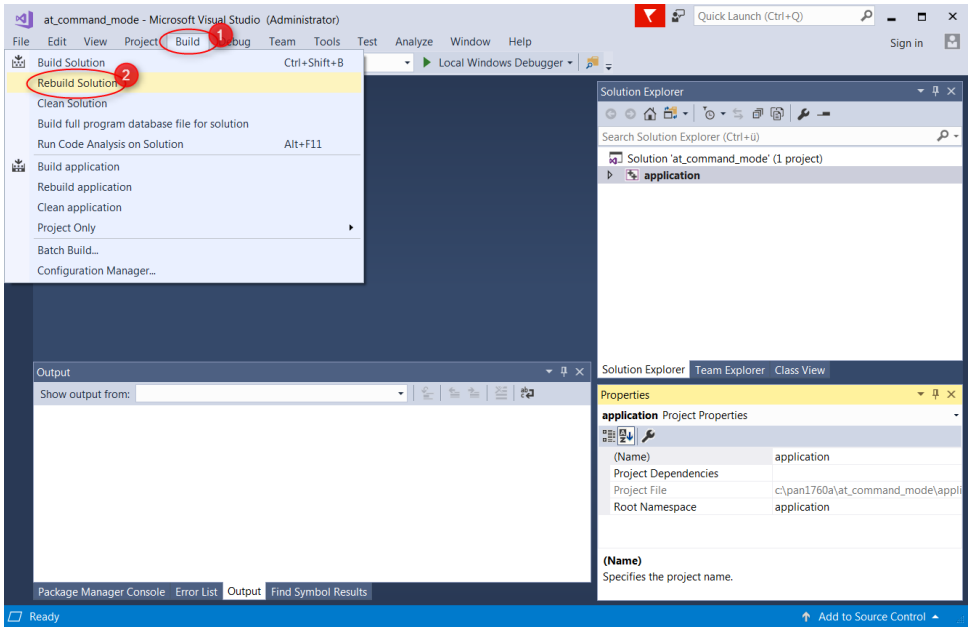
6.3 Recompiling the Sample Project

In order to recompile the AT command mode sample project it is necessary to have *Visual Studio 2017* installed as explained in ⇨ 3.4 Microsoft Visual Studio 2017..

The sample project solution file is located under *at_command_mode\at_command_mode.sln* and needs to be opened in *Visual Studio 2017.*



The configuration can be selected using the *Solution Configurations* drop down list.



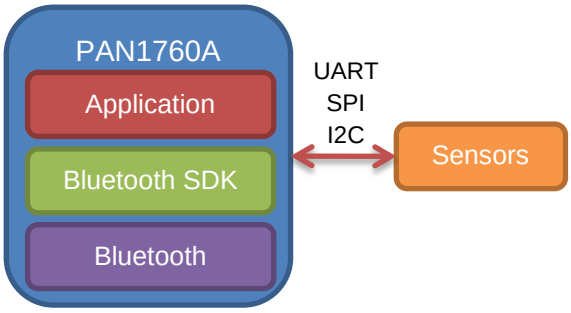
Afterwards *Rebuild Solution* from the *Build* menu can be used to build the currently selected configuration.

The resulting binaries can be found in the *host_mode\x64* directory as *peripheral.exe* and *central.exe.*

The AT command firmware in the Toshiba Bluetooth SDK does not fully support central mode yet, so the *central.exe* application is non-functional at the moment.

7 Standalone Mode

The PAN1760A USB stick can be used in standalone mode, where the built-in Cortex-M0 microcontroller works as the host processor as well.

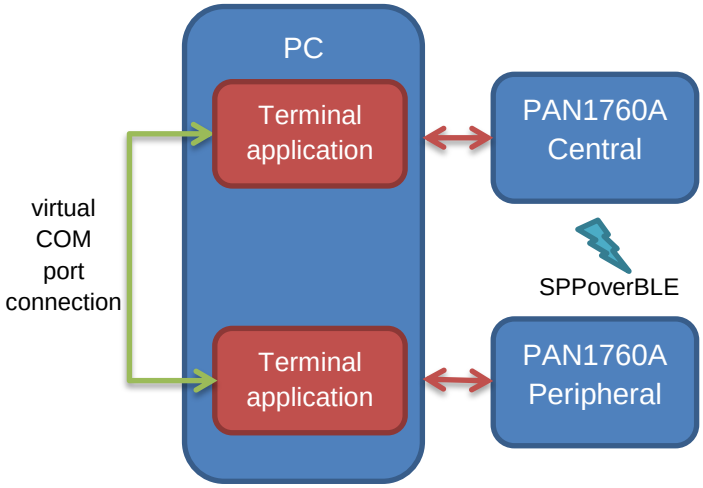


The on-board *FTDI USB UART* is available here as well and can be used to interface with the PC it is connected to.

7.1 Application

The software package includes a sample project called *standalone_mode* that shows how to use the PAN1760A module in a standalone application.

It showcases the use of the SPPoverBLE profile and comes in 2 project configurations: one will compile as a firmware for a central device and as a firmware for a peripheral device.



These 2 different firmwares have to be programmed to indivdual PAN1760A USB sticks.

When powered up, the PAN1760A USB sticks will immediately connect to each other and form a virtual COM port connection.

From the PC that hosts the PAN1760A USB sticks, 2 terminal applications can be started and used to transfer data back and forth.

The software package includes pre-compiled binaries that can be used for testing and are available under the *binaries/standalone_mode* subdirectory in the installation directory.

7.2 Programming the Firmware Files

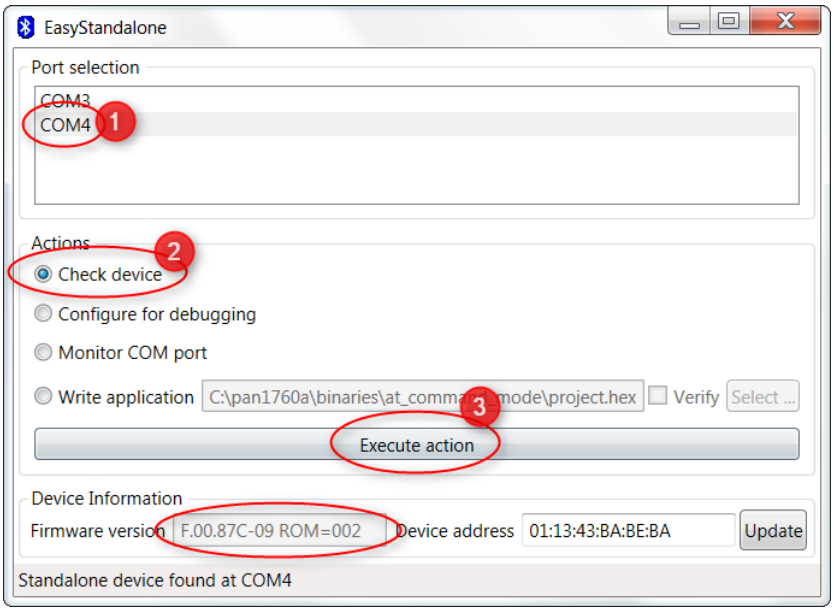
The Toshiba Bluetooth SDK includes the *EasyStandalone* application which can be used to write a firmware to the in-chip flash memory.

> *EasyStandalone* in explained in the chapter *APPENDIX: Working with EasyStandalone Tool* in the *Bluetooth SDK developers guide*.

Please remove all of the attached PAN1760A USB sticks.

Make sure that all the PAN1760A USB sticks are jumpered to *host mode* as explained in ⇨ 0 Host Mode.

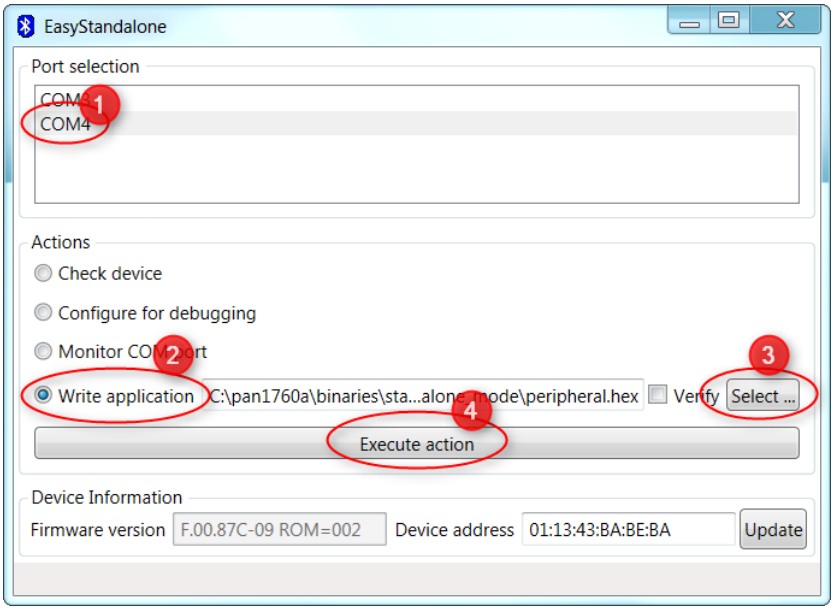
Now attach one of the PAN1760A USB sticks and launch *EasyStandalone* from the *Start menu.*



EasyStandalone does not have an automatic detection, so you need to make sure that the COM port selection is correct.

Select a COM port from the list of devices in the *Port selection* list, then choose *Check device* and press *Execute action.*

If all goes well, then the firmware version and the Bluetooth device address will be displayed.



Next please choose *Write application* and use the *Select…* button to choose the *peripheral.hex* firmware from *binaries/standalone_mode* subdirectory

Then choose *Execute action* to start the programming. A progress bar will be shown and after a couple of seconds the programming will be finished.

Now *EasyStandalone* can be closed.

Then jumper the PAN1760A USB stick to *standalone mode* as explained in ⇨ 4.2.2 Standalone Mode.

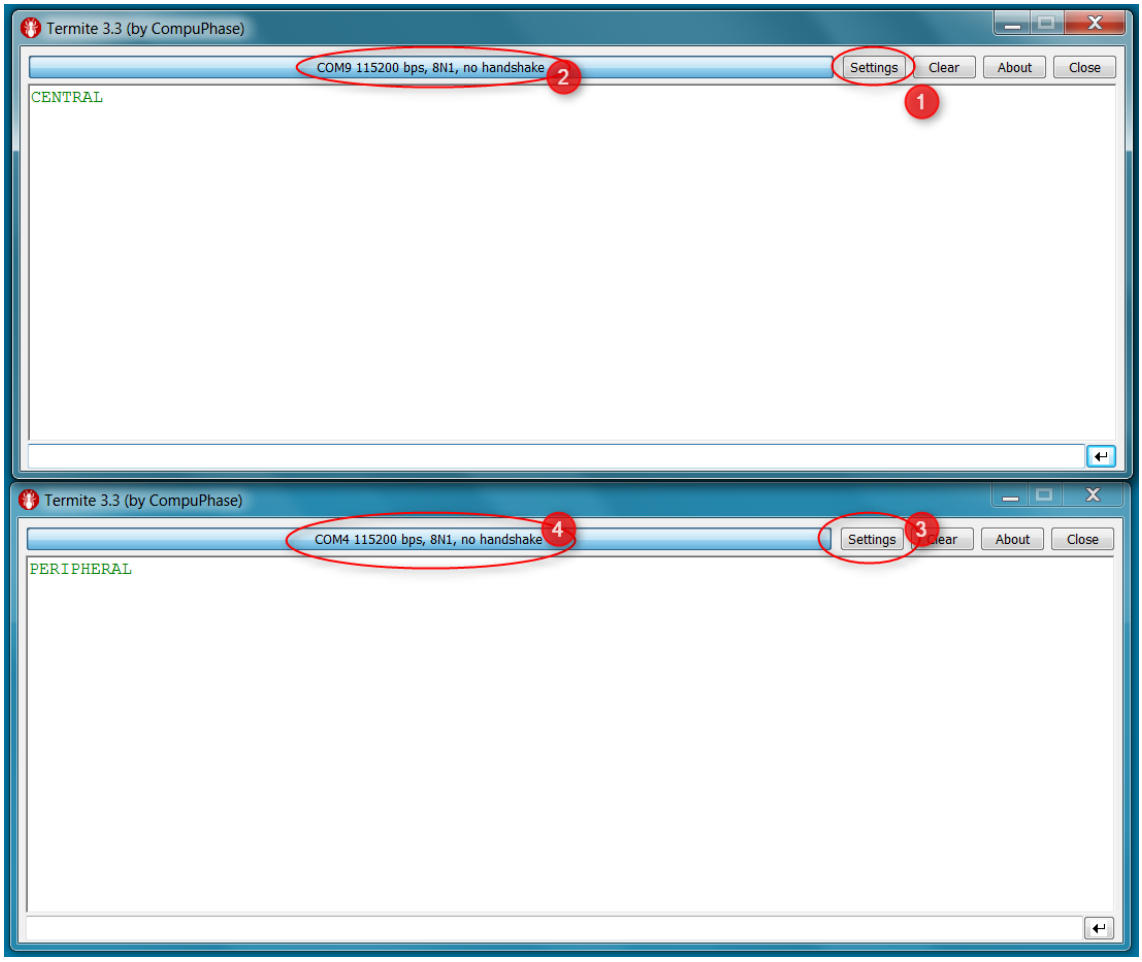
Repeat the same steps with the other PAN1760A USB stick, but using the *central.hex* firmware file.

7.3 Testing the Setup

Make sure that both PAN1760A USB sticks are jumpered to *standalone mode* as explained in ⇨ 4.2.2 Standalone Mode.

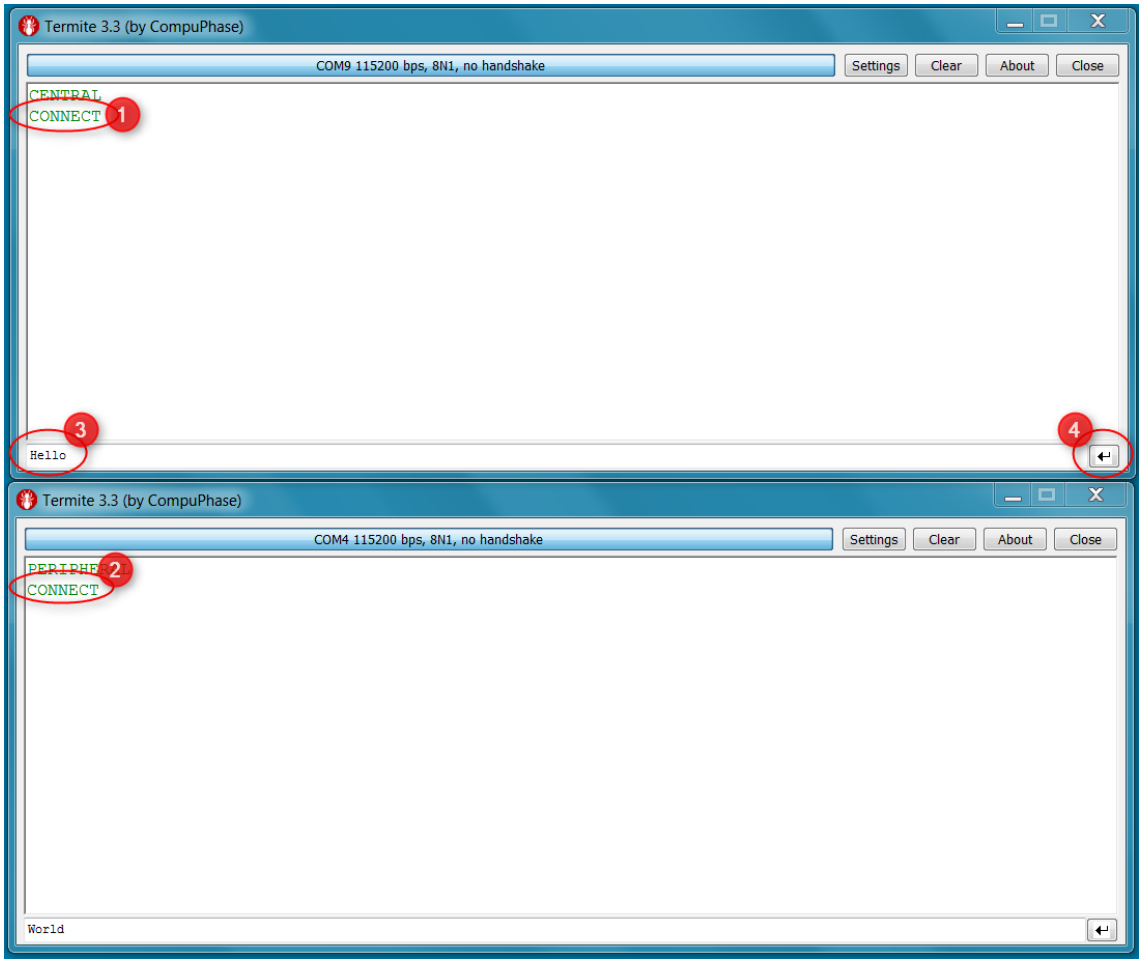
Next use any terminal application to connect to the well-known COM ports of the 2 PAN1760A USB sticks.

If you don’t have any terminal application yet, you can try *Termite* from ⇨ https://www.compuphase.com/software_termite.htm



Use the *Settings* button to select the desired COM port. *Termite* will connect to the COM port automatically.

If you press the reset button on the PAN1760A USB sticks they will identify themselves by printing either *CENTRAL* or *PERIPHERAL* to the terminal window.



The central device will immediately try to connect to the peripheral device. If a connection has been established *CONNECT* will be printed in each of the terminal windows.

Now you can use the *input textbox* and the *send* button to send data back and forth between the 2 devices.

7.4 Working with the Sample Project

In order to work with the standalone mode sample project it is necessary to have *IAR Embedded Workbench* installed as explained in ⇨ 0

IAR Embedded Workbench

7.4.1 Erasing the In-Device Flash Memory

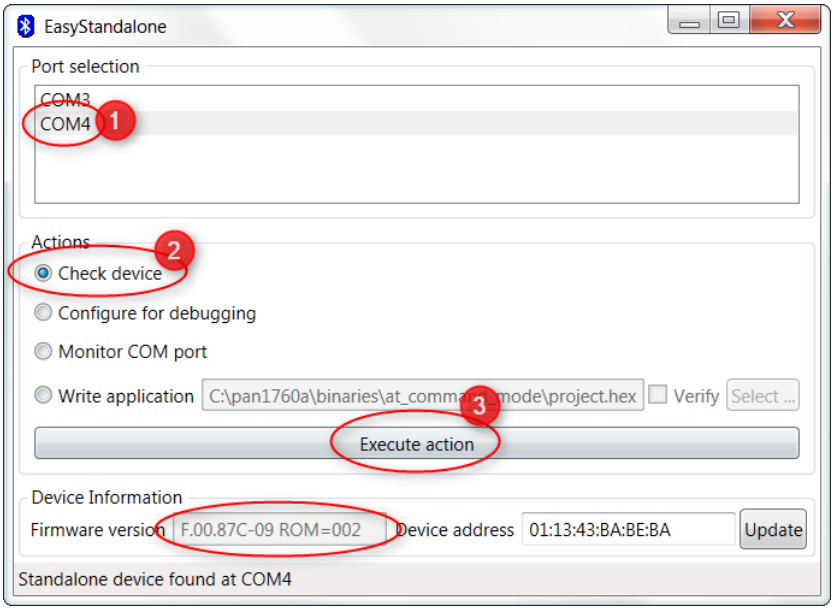
In order to be able to debug applications reliably the in-device flash memory of the PAN1760A module must be erased using *EasyStandalone.*

Please remove all of the attached PAN1760A USB sticks.

Make sure that all the PAN1760A USB sticks are jumpered to *host mode* as explained in ⇨ 0

Host Mode.

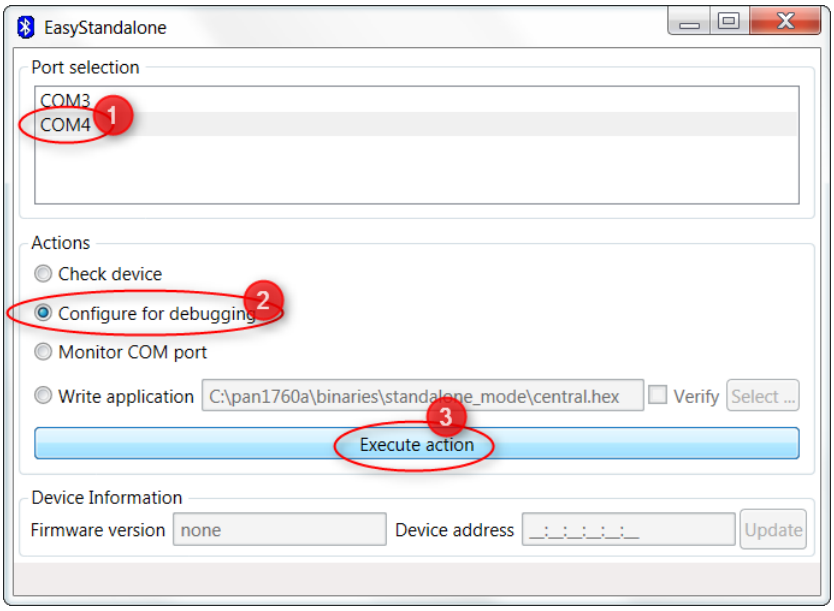
Now attach one of the PAN1760A USB sticks and launch *EasyStandalone* from the *Start menu.*



EasyStandalone does not have an automatic detection, so you need to make sure that the COM port selection is correct.

Select a COM port from the list of devices in the *Port selection* list, then choose *Check device* and press *Execute action.*

If all goes well, then the firmware version and the Bluetooth device address will be displayed.

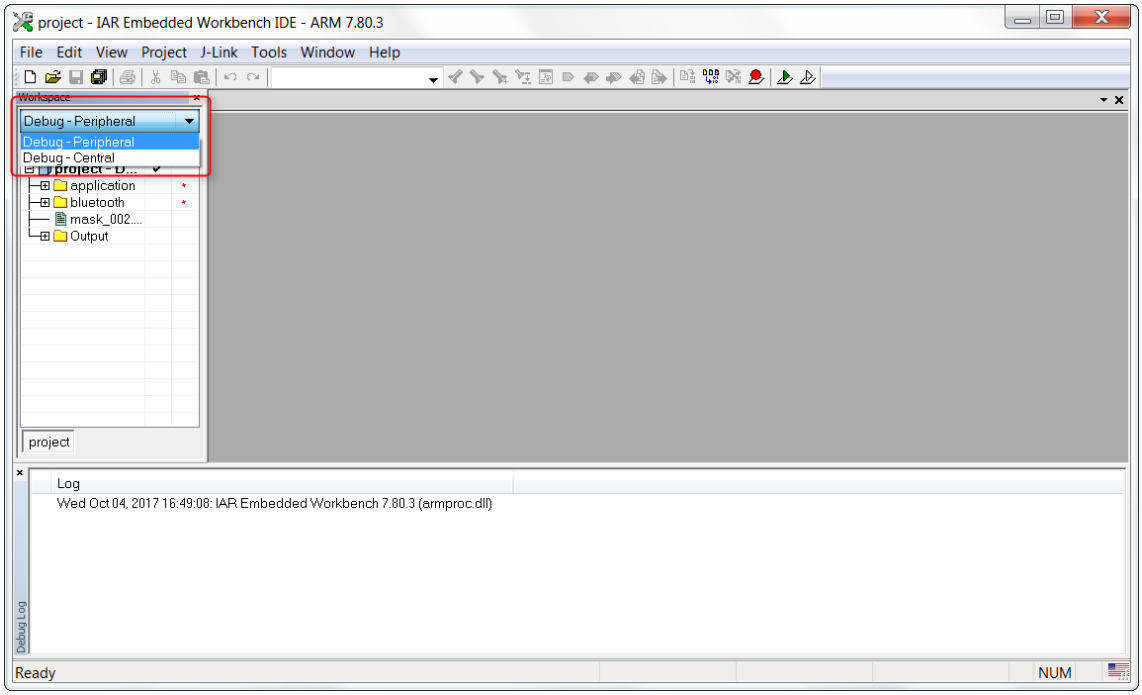


Now select *Configure for debugging* and press *Execute action* again.

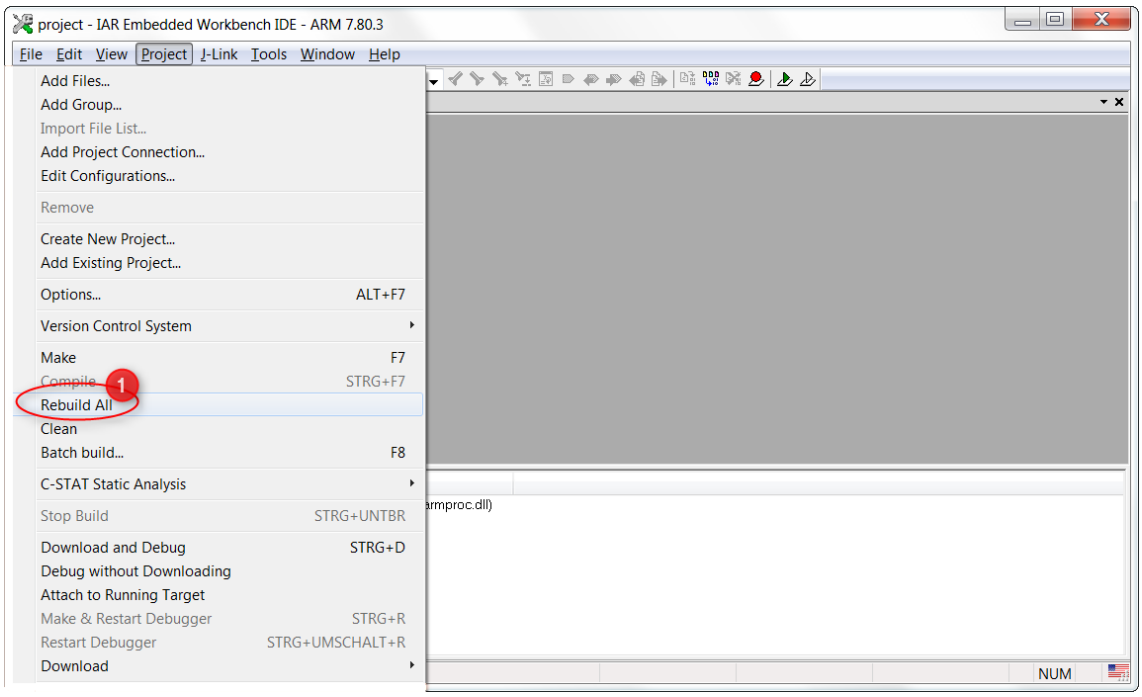
The in-device flash memory will now be erased.

7.4.2 Compilation

The sample project solution file is located under *standalone_mode\project.eww* and needs to be opened in *IAR Embedded Workbench.*



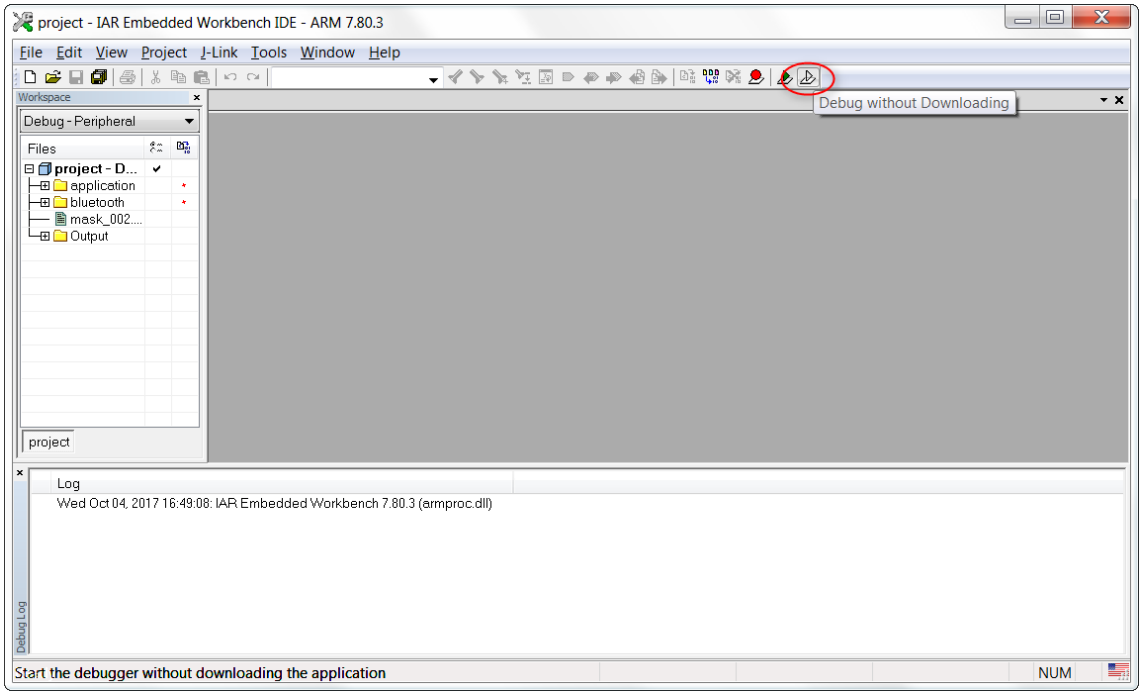
The configuration can be selected using the *Workspace Configurations* drop down list.



Afterwards *Rebuild all* from the *Project* menu can be used to build the currently selected configuration.

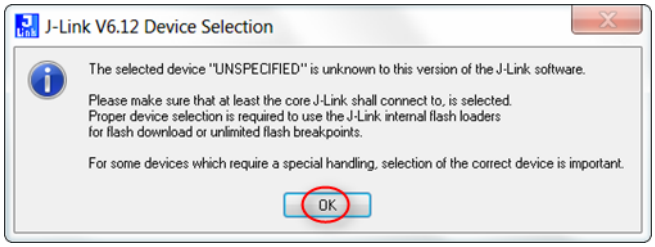
7.4.3 Debugging

Make sure that the PAN1760A USB stick is jumpered to *standalone mode* as explained in ⇨ 4.2.2 Standalone Mode and the in-device flash memory is erased as explained in ⇨ 7.4.1 Erasing the In-Device Flash Memory.

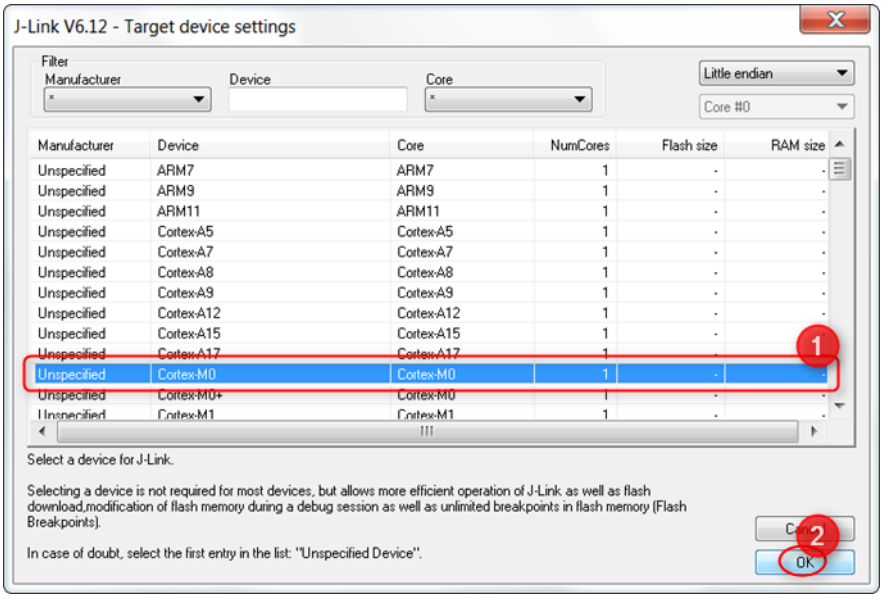


The debug process can be started with the *Debug without Downloading* option.

On the very first start the following message box will appear:

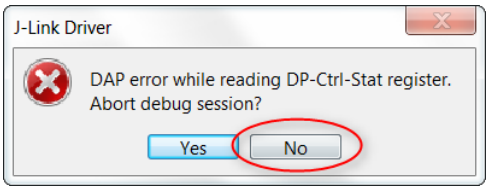


Please select *Ok.*

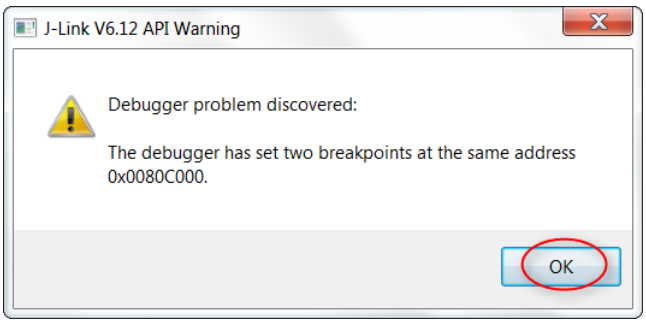


Choose *Unspecified / Cortex-M0* from the list of devices and continue with *Ok.*

Depending on the *J-Link* version that is installed the following warning message may appear as well:



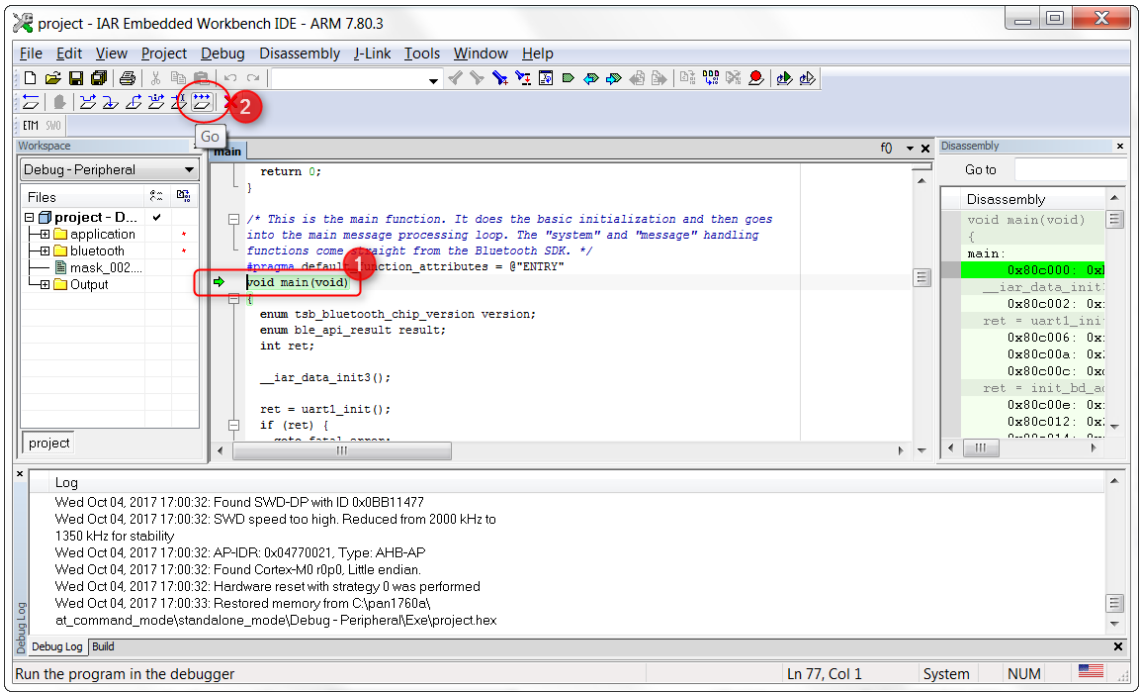
Another warning message might be:



Please select *No* and continue with *Ok.*

These problems are fixed in newer versions of the J-Link driver and/or the next version of the Toshiba Bluetooth SDK.

The application will now be started up, but immediately interrupted when the application reaches the *main()* function.



The application can then be continued by using the *Go* button.

8 Appendix

8.1 Changes to the Toshiba Bluetooth SDK

As explained in ⇨ 3.3 Software Package all the sample projects in the software package internally use the Toshiba Bluetooth SDK.

During the installation using the *unzipper* tool parts of it are copied to the *sdk* subdirectory in the installation directory.

However some changes have been done to the Toshiba Bluetooth SDK. Either to add missing features for the demo or to change certain things to make the implementation of the sample projects easier.

Please note that these changes are not an official part of the Toshiba Bluetooth SDK. These changes might or might not be adopted in upcoming versions of the SDK. If in doubt, please use the files from the original Toshiba Bluetooth SDK instead the ones that are used in the sample projects.

All changes are briefly explained in the following chapters.

8.1.1 Bluetooth subsystem

All changed files are contained in the file *bluetooth_differences.zip*. The individual changes are described in *unified diff format* in the file *bluetooth.diff* as well.

8.1.1.1 HCI API

The function *hci_api_read_data_from_storage_device()* has been added which allows to read data from the in-chip flash memory.

This function is currently missing in the Bluetooth SDK, but is necessary for the host mode application.

8.1.1.2 UART API

The function *uart_application_data_receive_debug_callback()* has been added which adds a debug callback that is triggered after a *complete message* has been received.

This callback is used for debugging purposes, especially in *EasyPAN.*

8.1.1.3 TCU API

The function *tcu_receive_any_message()* has been modified to catch a corner case when receiving a TCU messag. An explanation is available in the file.

8.1.1.4 Low Energy API

The function *spp_over_ble_profile_init()* has been modified and now accepts a *device name* as a parameter.

This is necessary for some of the sample projects where the device name of the device running the SPPoverBLE profile must be configurable.

8.1.1.5 Stub functions

All stub functions have been modified in two ways.

The prefix *WEAK_LINKAGE* has been added to all function definitions. It must be defined during compilation in such a way that when compiled the function is available for *weak linking.*

This is used by the sample application so that the original stub files can be used as is, but certain functions may be redefined and overridden with a different implementation.

The macro *BREAK_INTO_DEBUGGER* has been added to all function implementations. It must be defined during compilation in such a way that when compiled it is replaced by a function that makes sure that the execution of the application is stopped in the debugger.

This is used in the sample applications to be able to quickly find out which callback functions are actually called by the Bluetooth SDK in a certain usage scenario and must be implemented by the application.

8.1.2 Standalone subsystem

All changed files are contained in the file *standalone_differences.zip*. The individual changes are described in *unified diff format* in the file *standalone.diff* as well.

8.1.2.1 Message handling

The function *handle_uart1_api_mod()* in *message.c* has been modified to include a weak linking specifier for IAR.

This is necessary because the standalone application must override *handle_uart1_api_mod()* to provide a different implementation in order to be able to implement the desired use case in the sample application.

8.1.2.2 UART handling

The files *uart1_function.c* and *uart1_function.h* have been added which add a different UART handling for UART1.

In the current version of the SDK UART1 can only be used to send and receive TCU-like messages, but not send arbitrary data.

The new functions allow sending arbitrary data which is necessary to be able to implement the desired use case in the sample application.

8.2 Contact Details

8.2.1 Contact Us

Please contact your local Panasonic Sales office for details on additional product options and services:

- For Panasonic Sales assistance in the **EU**, visit
  ⇨ https://eu.industrial.panasonic.com/about-us/contact-us
  Email: wireless@eu.panasonic.com

- For Panasonic Sales assistance in **North America**, visit the Panasonic Sales & Support Tool to find assistance near you at
  ⇨ https://na.industrial.panasonic.com/distributors

Please visit the **Panasonic Wireless Technical Forum** to submit a question at
⇨ https://forum.na.industrial.panasonic.com

8.2.2 Product Information

Please refer to the Panasonic Wireless Connectivity website for further information on our products and related documents:

- For complete Panasonic product details in the **EU**, visit
  ⇨ https://pideu.panasonic.de/products/wireless-modules.html

- For complete Panasonic product details in **North America**, visit
  ⇨ http://www.panasonic.com/rfmodules