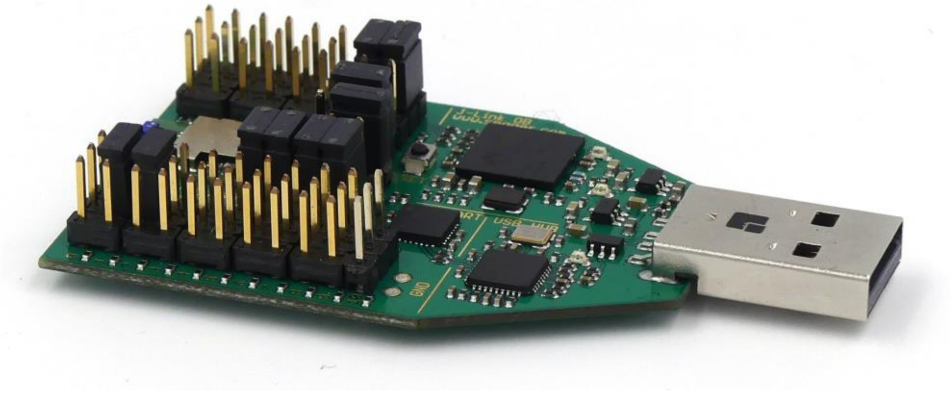
PAN1760A

Bluetooth Low Energy Module

Software Guide

Rev. 1.2



Wireless Modules

Engineering Samples (ES)

If Engineering Samples are delivered to the customer, these samples have the status “Engineering Samples”. This means that the design of this product is not yet concluded. Engineering Samples may be partially or fully functional, and they may differ from the published Product Specification.

Engineering Samples are not qualified and they are not to be used for reliability testing or series production.

Disclaimer

The customer acknowledges that samples may deviate from the Software Guide and may bear defects due to their status of development and the lack of qualification mentioned above.

Panasonic rejects any liability or product warranty for Engineering Samples. In particular, Panasonic disclaims liability for damages caused by:

- The use of the Engineering Sample other than for evaluation purposes, particularly the installation or integration in another product to be sold by the customer,
- Deviation or lapse in function of the Engineering Sample,
- Improper use of the Engineering Sample.

Panasonic Industrial Devices Europe GmbH disclaims any liability for consequential and incidental damages. In case of any queries regarding the Engineering Samples, please contact your local sales partner or the related product manager.

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	2017-10-04	Initial version
1.1	2018-04-05	Update to “Toshiba Bluetooth SDK 3.4.1”
1.2	2018-12-12	Updated to “Toshiba Bluetooth SDK 4.1.1”

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.
✎	**Tip** Indicates useful information designed to facilitate working with the PAN1760A.
⇨ [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.
✓	**Requirement** Indicates a requirement that must be met before the corresponding tasks can be completed.
➔	**Result** Indicates the result of a task or the result of a series of tasks.

Symbol	Description
This font	**GUI text** Indicates fixed terms and text of the graphical user interface. **Example:** Click **Save**.
`This font`	**File names, messages, user input** Indicates file names or messages and information displayed on the screen or to be selected or entered by the user. **Examples:** `pan1760a.c` contains the actual module initialization. The message `Failed to save your data` is displayed. Enter the value `Product 123`.
[Key]	**Key** Indicates a key on the keyboard, e. g. **[F10]**.

1.4 Related Documents

[1] PAN1760A Design Guide

Please refer to the Panasonic website for more information as well as related documents ⇨ 11.2.2 Product Information.

2 Introduction

The PAN1760A USB stick is a development platform for the Bluetooth® Low Energy (LE) 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/.

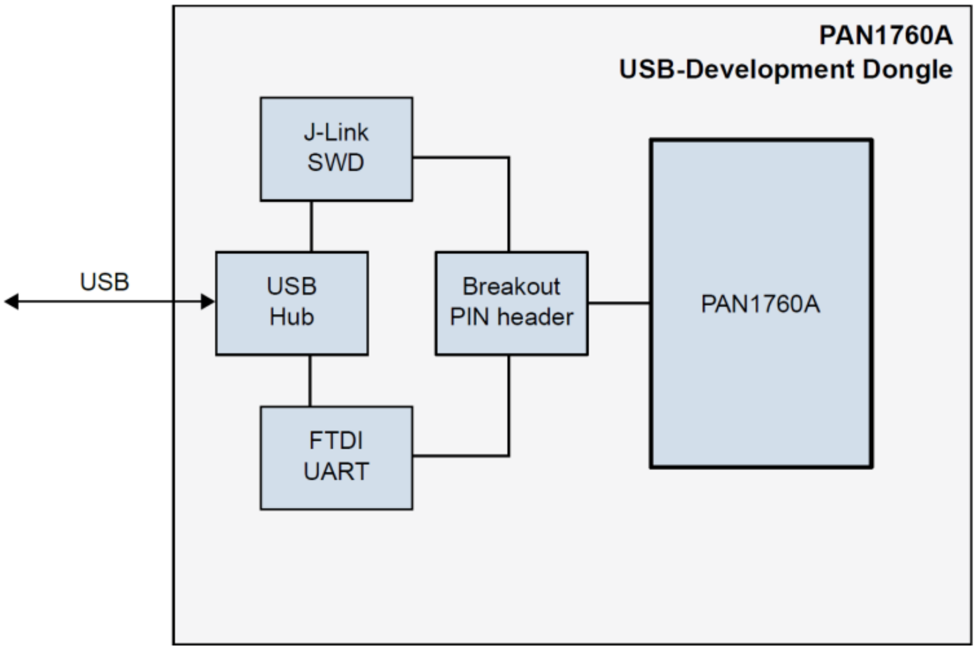
The Toshiba Bluetooth SDK includes a straight-forward serial cable replacement protocol called “SPP over Bluetooth LE profile”.

This Software Guide explains the usage of the SPP over Bluetooth LE profile in the 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 1.2.

To download the latest version please visit https://pideu.panasonic.de/produkte/bluetooth/pan1760a-bluetooth-low-energy.html

2.1 Architecture

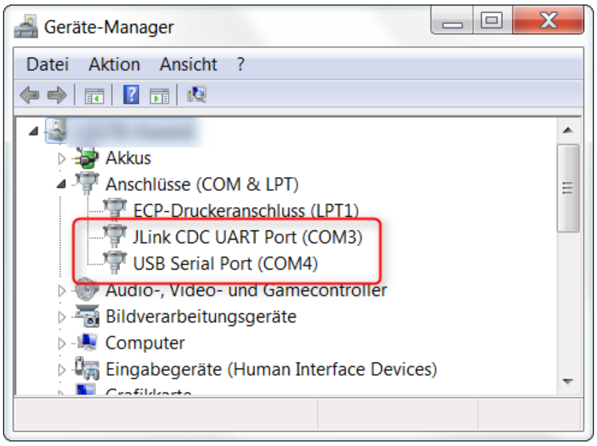


The PAN1760A USB stick consists of the following components:

- 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.

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


Geräte-Manager
Datei
Aktion
Ansicht
?
Akkus
Anschlüsse (COM & LPT)
ECP-Druckeranschluss (LPT1)
JLink CDC UART Port (COM3)
USB Serial Port (COM4)
Audio-, Video- und Gamecontroller
Bildverarbeitungsgeräte
Computer
Eingabegeräte (Human Interface Devices)
Grafikkarte

Only the COM port labeled as “USB Serial Port (COM4)” can be used to interface directly with the PAN1760A module.

2.2 Device Drivers

2.2.1 FTDI USB UART

Depending on the operating system that is used, drivers for the “FTDI USB UART” 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.

2.2.2 “Segger J-Link” SWD Debugger

Depending on the operating system that is used, drivers for “Segger J-Link” SWD debugger 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 Preparations

All examples shown in this document are based on the Bluetooth SDK version 4.1.1. It is mandatory to have that version downloaded before proceeding.

Toshiba provides a Bluetooth SDK that is available after registration.

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

After registration the Toshiba Bluetooth SDK can be downloaded from the “Developer Zone”:

https://apps.toshiba.de/extranet/display/TBTSW/.

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

When the PAN1760A Software Guide ZIP file is unpacked, the following directories are present in the current directory:

- **Binaries**: contains pre-compiled applications for immediate evaluation
- **Configurations**: contains the source code for all applications

Furthermore the following files are present in the current directory:

- `PAN1760A Software Guide.pdf`: this document
- `pan1760a_unzipper.exe`: unzipper tool
- `bluetooth_sdk_differences.zip`: differences to Toshiba Bluetooth SDK version 4.1.1

3.1 Unzipper Tool

Setting up the correct directory layout is tedious and error prone. In order to facilitate this process, the tool **unzipper** 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:\pan`.
>
> If a different path was used, please make sure to adopt the path names in the examples accordingly.

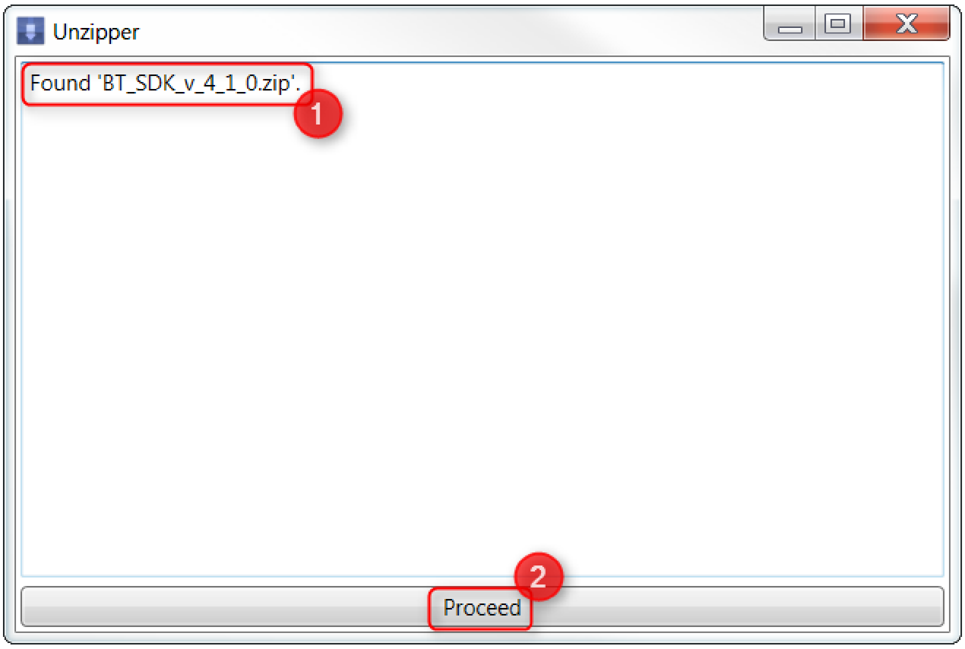
The following requirements must be met to work correctly with **unzipper**:

✓ `BT_SDK_v_4_1_1.zip` (Toshiba Bluetooth SDK version 4.1.1) is present in the same directory additionally

1. Execute `pan1760a_unzipper.exe`.

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

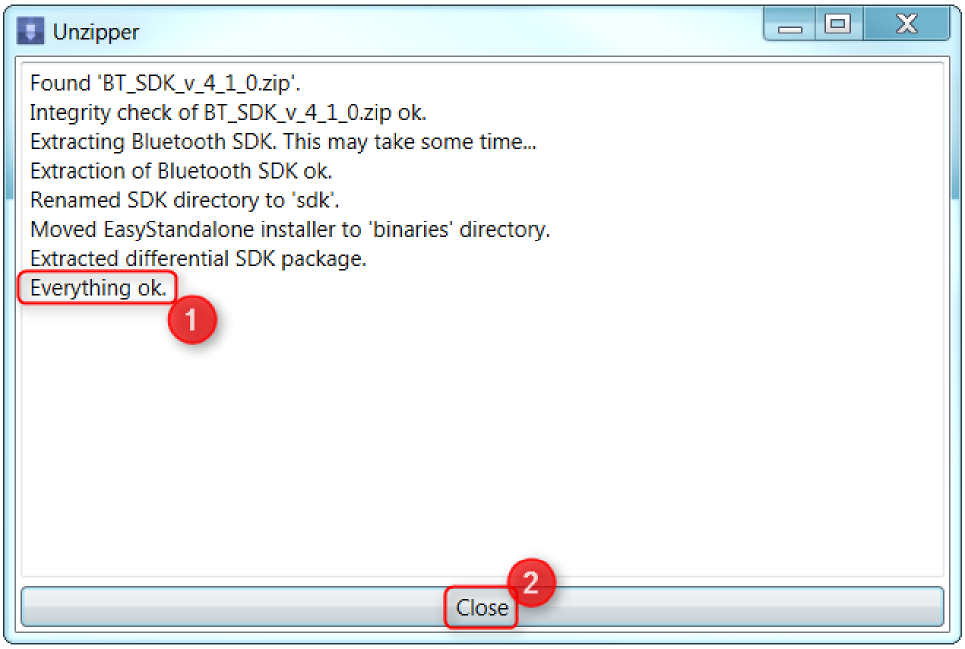
➔ This message(1) will be displayed.



Found 'BT_SDK_v_4_1_0.zip'.

2. Click **Proceed**(2).

➔ This message(1) will be displayed.

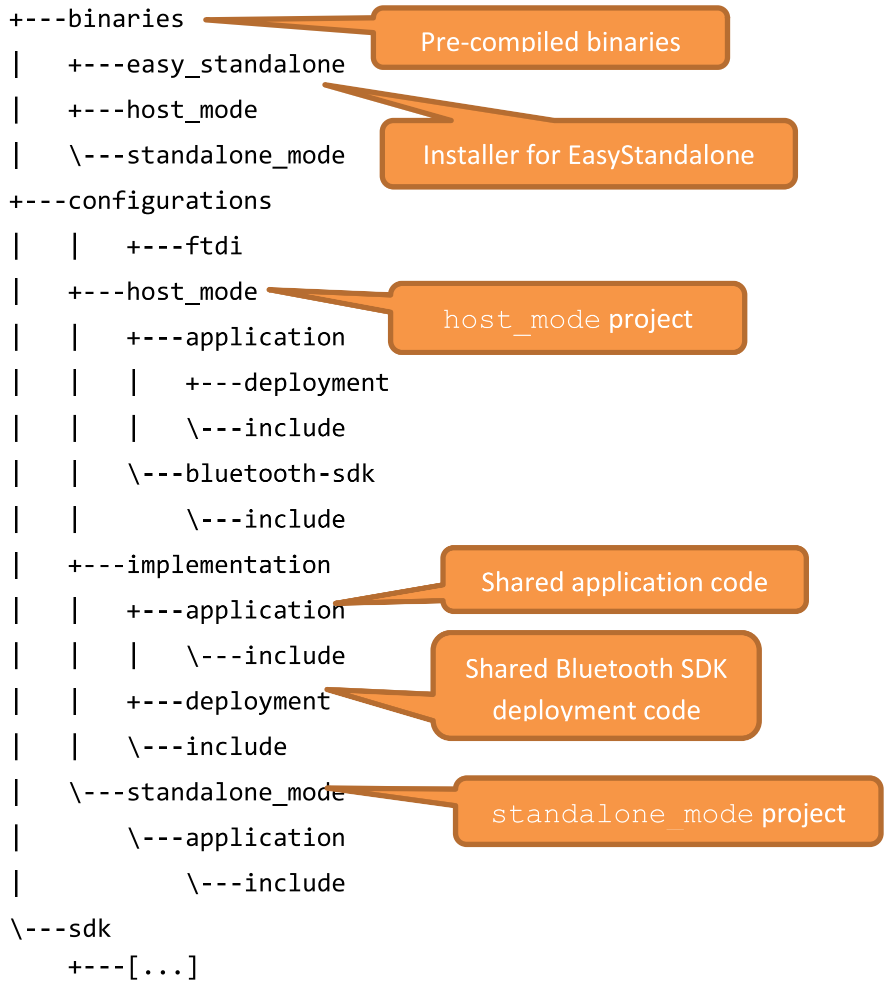


Found 'BT_SDK_v_4_1_0.zip'.
Integrity check of BT_SDK_v_4_1_0.zip ok.
Extracting Bluetooth SDK. This may take some time...
Extraction of Bluetooth SDK ok.
Renamed SDK directory to 'sdk'.
Moved EasyStandalone installer to 'binaries' directory.
Extracted differential SDK package.
Everything ok.

3. Click **Close**(2).

3.2 Directory Layout

After **unzipper** has finished the following directory layout is present:



```
+---binaries
|   +---easy_standalone
|   +---host_mode
|   \---standalone_mode
+---configurations
|   |   +---ftdi
|   +---host_mode
|   |   +---application
|   |   |   +---deployment
|   |   |   \---include
|   |   \---bluetooth-sdk
|   |       \---include
|   +---implementation
|   |   +---application
|   |   |   \---include
|   |   +---deployment
|   |   \---include
|   \---standalone_mode
|       \---application
|           \---include
\---sdk
    +---[...]
```

In the installation process the Bluetooth SDK is slightly modified for some of the use cases. The changes are explained in ⇨ 11.1 Changes to the Toshiba Bluetooth SDK.

3.3 Manual Installation

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

1. Create a destination directory and change into it.
2. Unzip the ZIP file **PAN1760A Software Guide** into that directory.
3. Unzip the Toshiba Bluetooth `SDK BT_SDK_v_4_1_1.zip` in that directory.
4. Rename the **BT_SDK_v_4_1_1** directory to **sdk**.

5. Copy **sdk/at_command/easy_standalone** directory to **binaries/EasyStandalone**.
6. Change into the **sdk** subdirectory.
7. Unzip `bluetooth_sdk_differences.zip` from parent directory over existing files.

3.4 Further Readings

This document mostly explains the specific features of the PAN1760A module.

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, for example the “Bluetooth SDK developers guide”.

All documentation from Toshiba can be found in the Bluetooth SDK directory `BT_SDK_v_4_1_1\documentation`.

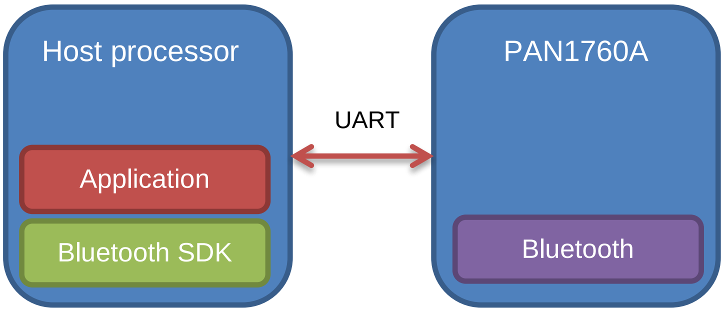
4 Operating Modes

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

The main operating modes are:

- Host mode
- Standalone mode

4.1 Host Mode

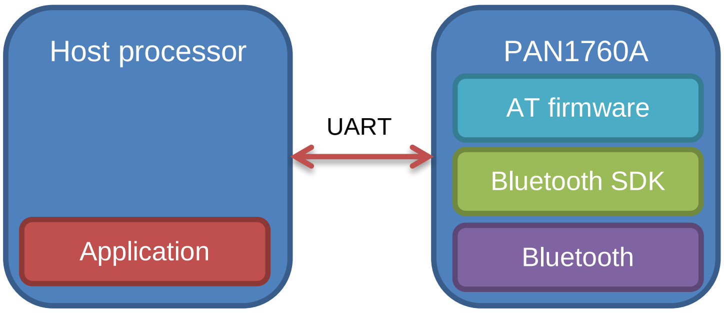


The host mode operating mode can be used in a dual chip product configuration, where the chips are connected via UART.

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.

4.2 AT Command Mode



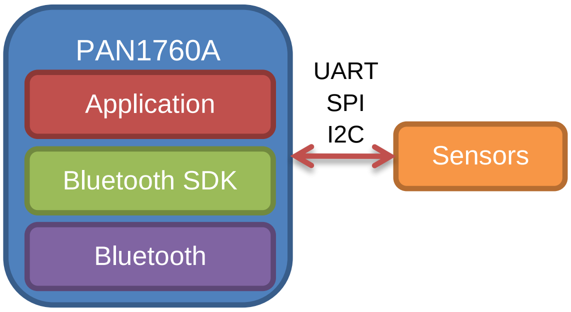
The AT command mode operating mode can be used in a dual chip product configuration as well and again the chips are connected via UART.

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.

4.3 Standalone Mode



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

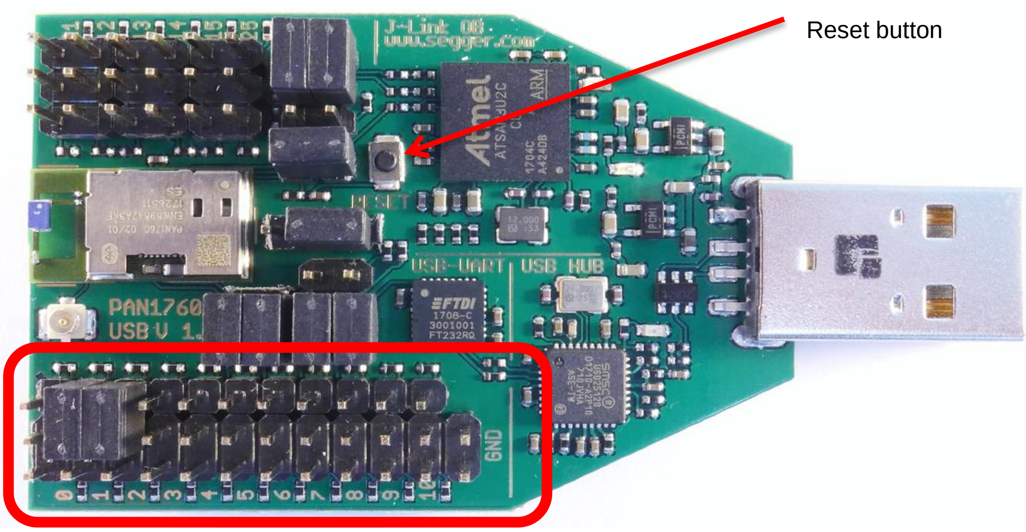
The Toshiba Bluetooth SDK as well as the customer application is run in the context of the Bluetooth processing.

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.

4.4 Mode Selection

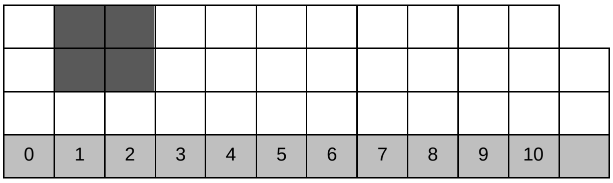
The operating mode of the PAN1760A module is determined by two GPIO jumpers in the PIN header breakout section.



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

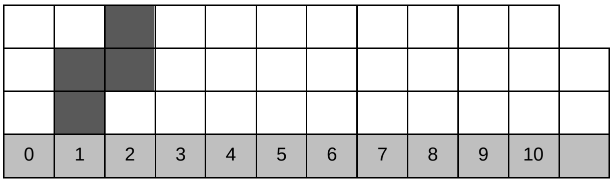
When the mode has been changed the PAN1760A module needs to be reset: Press the “reset” button.

4.4.1 Host Mode



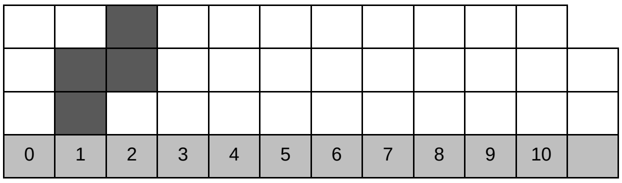
- Both GPIO jumpers must be set to the upward position, connecting the GPIOs to GND

4.4.2 Standalone Mode



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

4.4.3 AT Command Mode



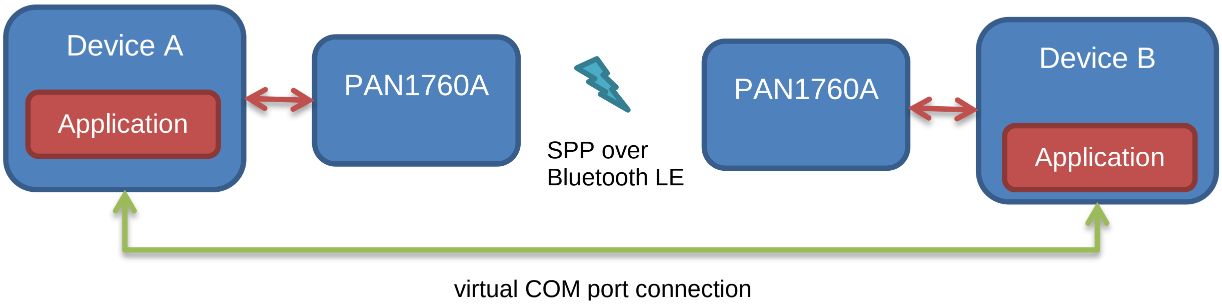
- GPIO jumper 1 must be set to the downward position, connecting GPIO jumper 1 to VCC
- GPIO jumper 2 must be set to the upward position, connecting GPIO jumper 2 to GND

Please note that the AT command firmware must be written to the in-device flash as explained in chapter ⇨ 8.2 Programming the Firmware.

5 SPP over Bluetooth LE 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.



The Bluetooth SDK already contains such a wireless serial cable replacement profile which is called “SPP over Bluetooth LE” profile.

If used on two devices, device A and B, the SPP over Bluetooth LE profile basically provides a virtual COM port connection between these two devices.

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

- `BT_SDK_v_4_1_1\documentation\client_profiles\spp_over_ble_profile.pdf`
- `BT_SDK_v_4_1_1\documentation\server_profiles\spp_over_ble_profile.pdf`

6 Bluetooth Low Energy Basics

The following Bluetooth Low Energy basics will be briefly explained:

- Peripheral device and central
- Master and slave
- Server and client

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.

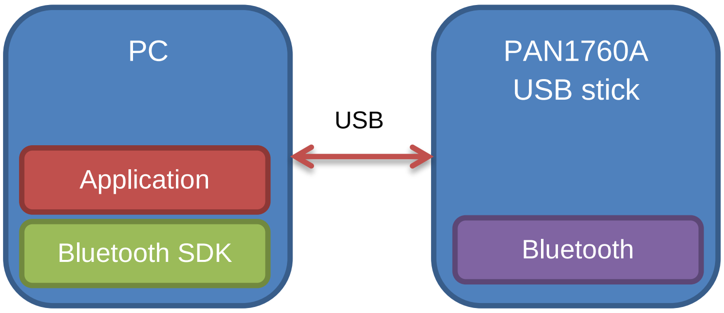
Then there might be a sensor device, which acts as a peripheral device and waits for an incoming connection while advertising. When a connection comes in, it acts as a slave and accepts the connection, then works as a server to provide its sensor information.

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.

7 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, which is a PC for the PAN1760A USB stick.

7.1 PC Application

The software package includes a sample project called “host_mode” that shows how to use the Bluetooth SDK in a plain C application.

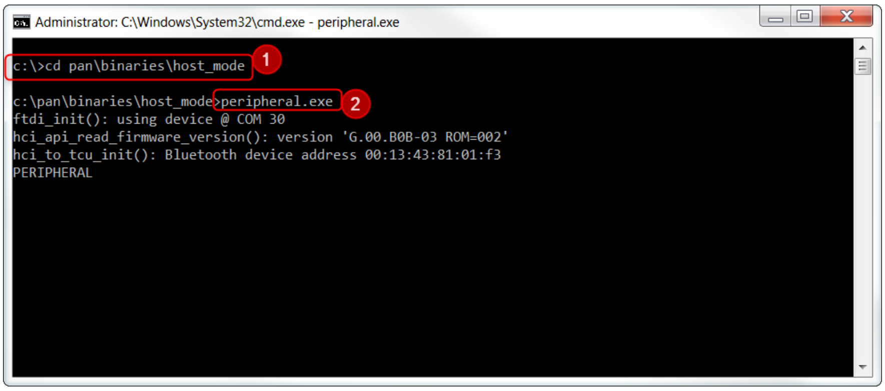
It showcases the use of the SPP over Bluetooth LE profile and comes in two project configurations:

- 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 precompiled binaries that can be used for testing. The binaries are available under the subdirectory `binaries/host_mode` in the installation directory.

1. Remove all of the attached PAN1760A USB sticks.
2. Make sure that all the PAN1760A USB sticks are jumpered to **mode** as explained in ⇨ 4.4.1 Host Mode.
3. Attach one of the PAN1760A USB sticks and launch a Windows command line prompt, for example by executing `cmd.exe` from the **Start menu**.

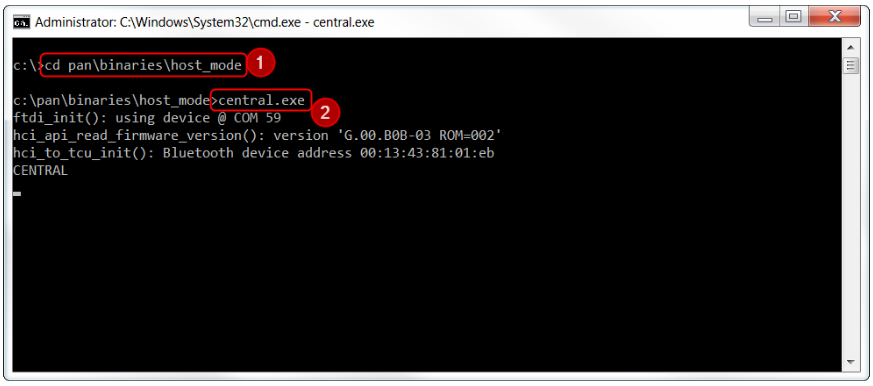
4. Navigate to the folder `binaries\host_mode`(1) with the cd command and execute `peripheral.exe`(2).



```
c:\>cd pan\binaries\host_mode

c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 30
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:f3
PERIPHERAL
```

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

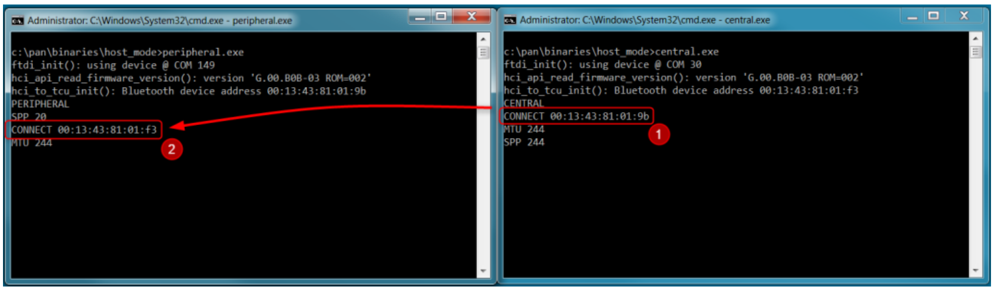
5. Attach the other PAN1760A USB stick and launch another Windows command line prompt.
6. Navigate to the folder `binaries\host_mode`(1) with the cd command and execute `central.exe`(2).



```
c:\>cd pan\binaries\host_mode

c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 59
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:eb
CENTRAL
```

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

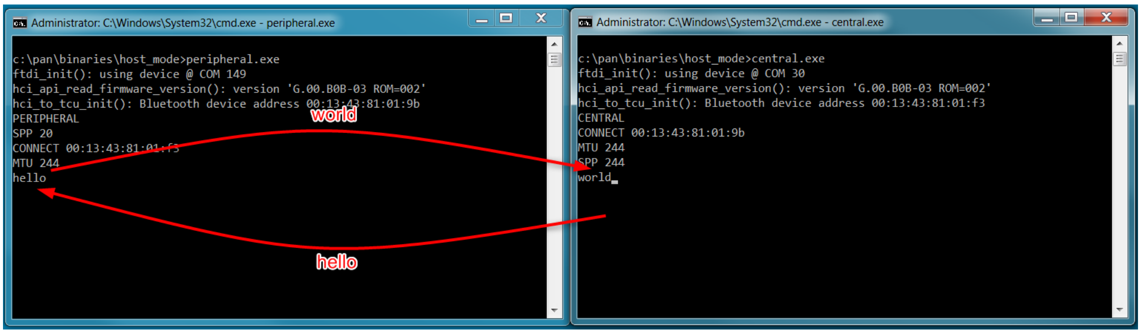
➔ The central device will immediately recognize the peripheral device and establish a connection.


Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
2
Administrator: C:\Windows\System32\cmd.exe - central.exe
1

```
c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244
```

```
c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 30
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:f3
CENTRAL
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244
```

Now data can be transferred back and forth by pressing key in the respective terminal window. In this example the one side sent **hello**, while the other side sent **world**.


Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
Administrator: C:\Windows\System32\cmd.exe - central.exe
world
hello

```
c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244
hello
```

```
c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 30
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:f3
CENTRAL
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244
world_
```

Terminate application

Any application can be terminated.

1. Press **CTRL**+**C**

   ➔ The connection between the two devices will be shut down accordingly.

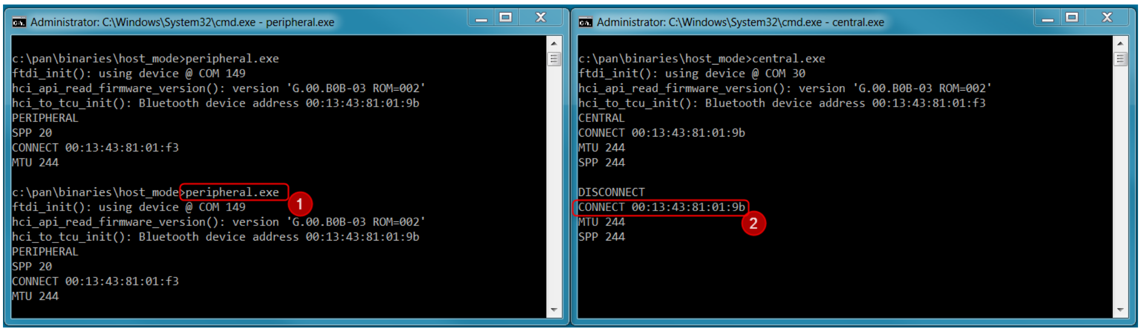
Restart after pressing CTRL+C

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 the following example, `peripheral.exe`(1) 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`(1) was started again and the connection was immediately established again(2).


Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
Administrator: C:\Windows\System32\cmd.exe - central.exe
1
2

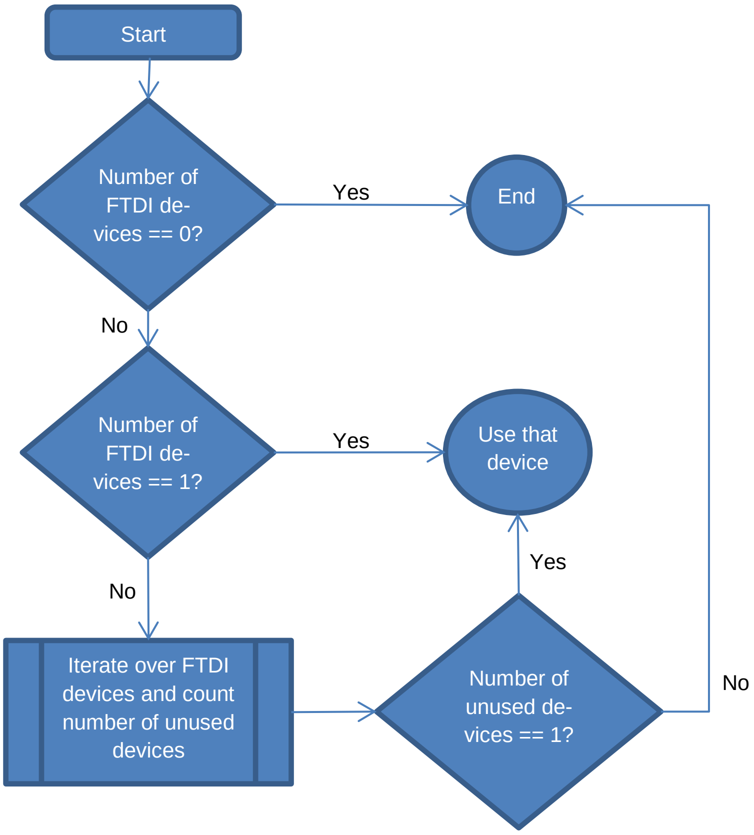
```
c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244

c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244
```

```
c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 30
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:f3
CENTRAL
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244

DISCONNECT
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244
```

7.2 COM Port Handling

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

In order to make development easier, the applications do not require that a COM port is given. Instead use the following heuristic to find the COM port they should use:


Start
Number of FTDI devices == 0?
Yes
End
No
Number of FTDI devices == 1?
Yes
Use that device
No
Yes
Iterate over FTDI devices and count number of unused devices
Number of unused devices == 1?
No

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 two 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.

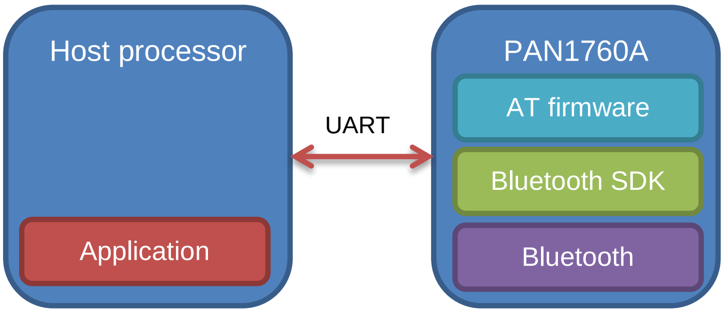
The benefit of this approach is that you don’t have to remember any of the COM ports as long as you attach the PAN1760A USB sticks and start the applications one after another.

But 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.

8 AT Command Mode

The AT command firmware is a special PAN1760A module firmware that comes with the Toshiba Bluetooth SDK delivery.

The Toshiba Bluetooth SDK will be run in the context of the PAN1760A module as part of the AT command firmware and does most of the necessary Bluetooth processing that usually the host processor would need to do.



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

In this scenario a standard PC can act as the host processor and the application on the host processor will use the AT command interface of the AT command firmware.

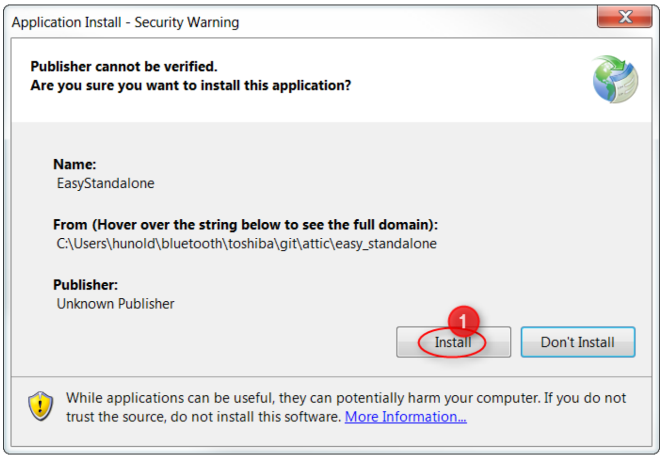
Please note that the AT command firmware in the Toshiba Bluetooth SDK does not support the central role yet.

8.1 Installing EasyStandalone

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

If the tool **unzipper** has been used as explained in ⇨ 3.1 Unzipper Tool then the installer for the tool **EasyStandalone** will be present in the directory `binaries\EasyStandalone`.

1. Execute `setup.exe` from the directory **binaries\EasyStandalone**.
2. Click **Install**(1).



Application Install - Security Warning

Publisher cannot be verified.
Are you sure you want to install this application?

Name:
EasyStandalone

From (Hover over the string below to see the full domain):
C:\Users\hunold\bluetooth\toshiba\git\attic\easy_standalone

Publisher:
Unknown Publisher

While applications can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information...

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

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

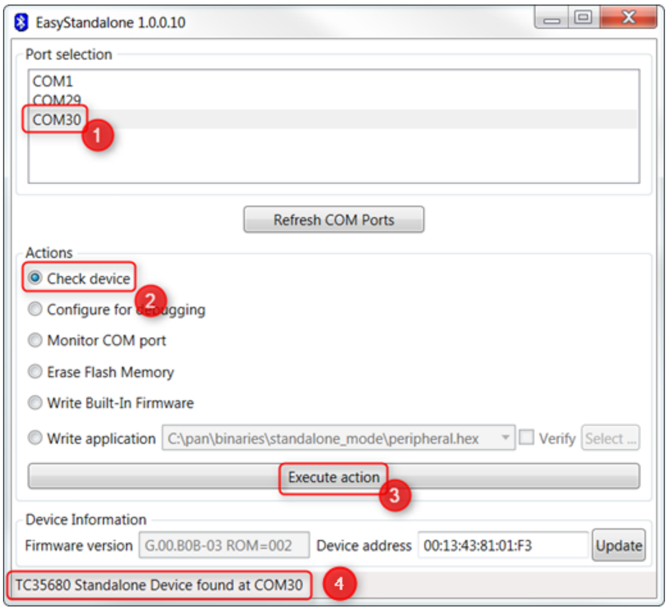
8.2 Programming the Firmware

The following requirements must be met:

- ✓ All of the attached PAN1760A USB sticks are removed
- ✓ All the PAN1760A USB sticks are jumpered to host mode as explained in ⇨ 4.4.1 Host Mode

1. Attach one of the PAN1760A USB sticks.
2. Go to the **Start menu** and launch **EasyStandalone**.
3. Make sure that the COM port selection is correct because **EasyStandalone** does not have an automatic detection.

4. Select a COM port(1) from the list of devices in the **Port selection**.



Port selection: COM30 (selected)

Actions: ● Check device

Write application: C:\pan\binaries\standalone_mode\peripheral.hex

Firmware version: G.00.B0B-03 ROM=002

Device address: 00:13:43:81:01:F3

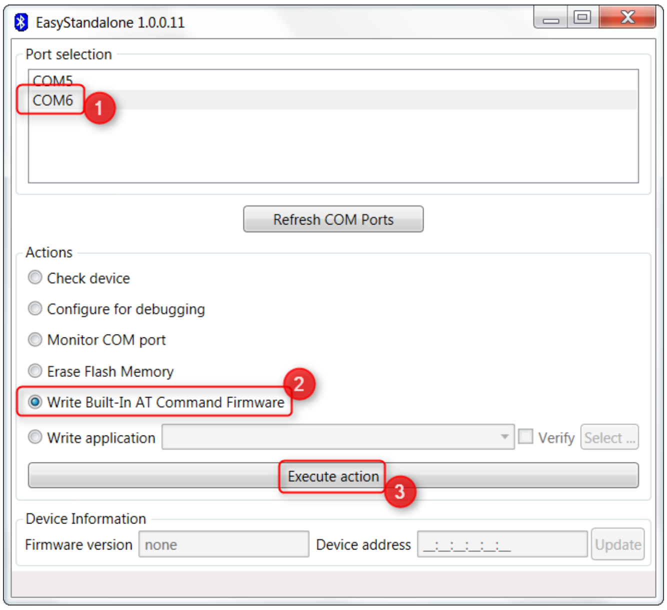
TC35680 Standalone Device found at COM30

5. Choose **Check device**(2) and click **Execute action**(3).

   ➔ The firmware version and the Bluetooth device address will be displayed(4).

6. Make sure that the COM port selection(1) is correct.

7. Choose **Write Built-In Firmware**(2).



Port selection: COM6 (selected)

Actions: ● Write Built-In AT Command Firmware

Firmware version: none

Device address: __:__:__:__:__:__

8. Click **Execute action**(3) to start the programming.

   ➔ A progress bar will be shown and after a couple of seconds the programming will be finished.

9. Now **EasyStandalone** can be closed.

10. Jumper the PAN1760A USB stick to “AT command mode” as explainend in ⇨ 4.4.3 AT Command Mode.

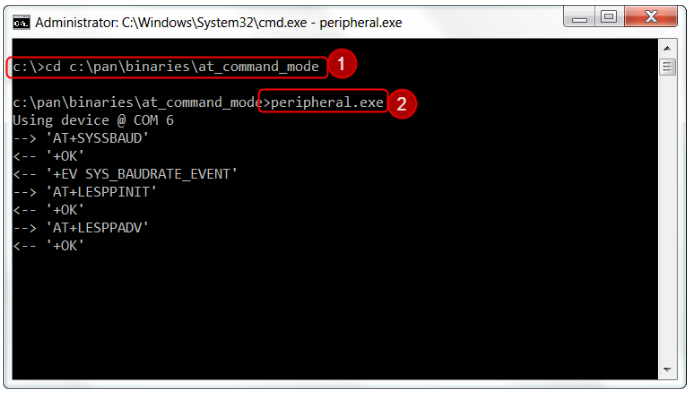
8.3 PC Application

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 SPP over Bluetooth LE profile and comes in one project configuration that will compile an application acting as a peripheral device.

> The software package includes precompiled binaries that can be used for testing. The binaries are available under the subdirectory `binaries/at_command_mode` in the installation directory.

1. Remove all of the attached PAN1760A USB sticks.
2. Make sure that all the PAN1760A USB sticks are jumpered to **AT command mode** as explained in ⇨ 4.4.3 AT Command Mode.
3. Attach one of the PAN1760A USB sticks and launch a Windows command line prompt, for example by executing `cmd.exe` from the **Start menu**.
4. Navigate to the folder `binaries\at_command_mode`(1) with the cd command and execute `peripheral.exe`(2).



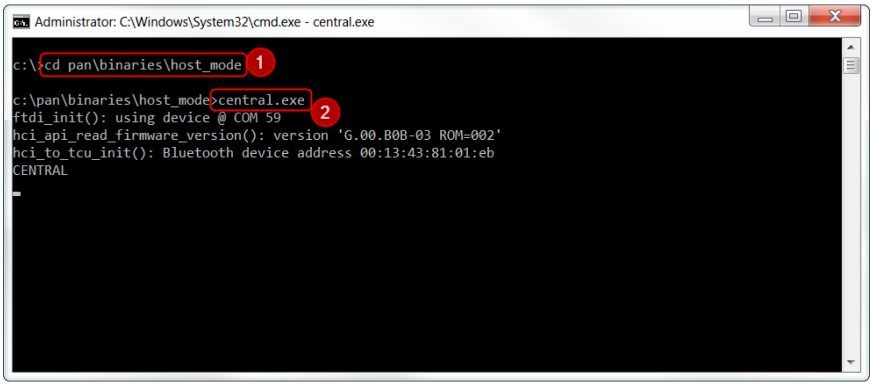
```
c:\>cd c:\pan\binaries\at_command_mode

c:\pan\binaries\at_command_mode>peripheral.exe
Using device @ COM 6
--> 'AT+SYSSBAUD'
<-- '+OK'
<-- '+EV SYS_BAUDRATE_EVENT'
--> 'AT+LESPPINIT'
<-- '+OK'
--> 'AT+LESPPADV'
<-- '+OK'
```

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

> Because the AT command firmware in the Toshiba Bluetooth SDK does not support central role yet, we use EasyPAN to act as the central device instead.

5. Attach the other PAN1760A USB stick and launch another Windows command line prompt. Make sure that this PAN1760A USB sticks is jumpered to **host mode** as explained in ⇨ 4.4.1 Host Mode.

6. Navigate to the folder `binaries\host_mode`(1) with the cd command and execute `central.exe`(2).

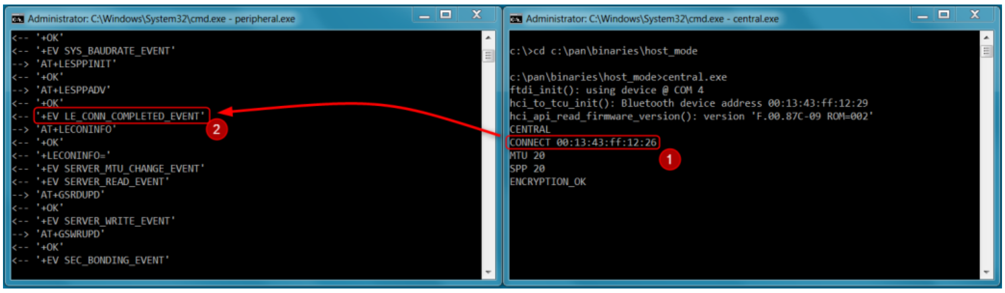


```
c:\>cd pan\binaries\host_mode

c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 59
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:eb
CENTRAL
```

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

The AT command mode sample application uses the sample COM port handling as explained in ⇨ 7.2 COM Port Handling.

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

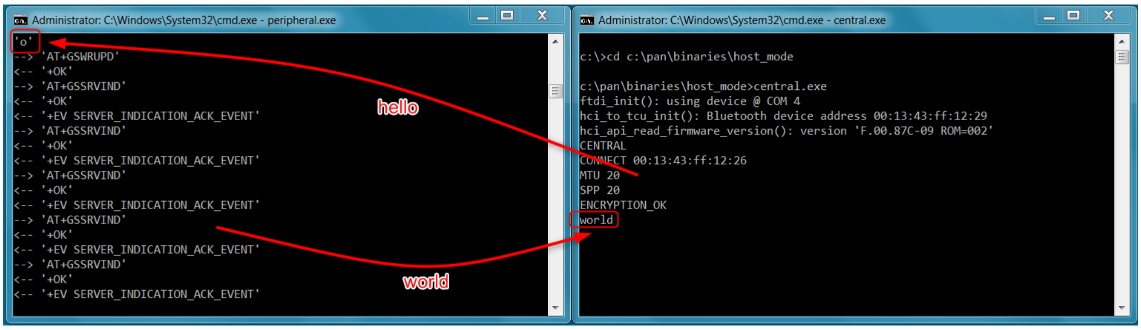

Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
Administrator: C:\Windows\System32\cmd.exe - central.exe
2
1

```
<-- '+OK'
<-- '+EV SYS_BAUDRATE_EVENT'
--> 'AT+LESPPINIT'
<-- '+OK'
--> 'AT+LESPPADV'
<-- '+OK'
<-- '+EV LE_CONN_COMPLETED_EVENT'
--> 'AT+LECONINFO'
<-- '+OK'
<-- '+LECONINFO='
<-- '+EV SERVER_MTU_CHANGE_EVENT'
<-- '+EV SERVER_READ_EVENT'
--> 'AT+GSRDUPD'
<-- '+OK'
<-- '+EV SERVER_WRITE_EVENT'
--> 'AT+GSWRUPD'
<-- '+OK'
<-- '+EV SEC_BONDING_EVENT'
```

```
c:\>cd c:\pan\binaries\host_mode

c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 4
hci_to_tcu_init(): Bluetooth device address 00:13:43:ff:12:29
hci_api_read_firmware_version(): version 'F.00.87C-09 ROM=002'
CENTRAL
CONNECT 00:13:43:ff:12:26
MTU 20
SPP 20
ENCRYPTION_OK
```

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

In this example the one side sent **hello**, while the other side sent **world**.


Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
Administrator: C:\Windows\System32\cmd.exe - central.exe
hello
world

```
'o'
--> 'AT+GSWRUPD'
<-- '+OK'
--> 'AT+GSSRVIND'
<-- '+OK'
<-- '+EV SERVER_INDICATION_ACK_EVENT'
--> 'AT+GSSRVIND'
<-- '+OK'
<-- '+EV SERVER_INDICATION_ACK_EVENT'
--> 'AT+GSSRVIND'
<-- '+OK'
<-- '+EV SERVER_INDICATION_ACK_EVENT'
--> 'AT+GSSRVIND'
<-- '+OK'
<-- '+EV SERVER_INDICATION_ACK_EVENT'
--> 'AT+GSSRVIND'
<-- '+OK'
<-- '+EV SERVER_INDICATION_ACK_EVENT'
```

```
c:\>cd c:\pan\binaries\host_mode

c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 4
hci_to_tcu_init(): Bluetooth device address 00:13:43:ff:12:29
hci_api_read_firmware_version(): version 'F.00.87C-09 ROM=002'
CENTRAL
CONNECT 00:13:43:ff:12:26
MTU 20
SPP 20
ENCRYPTION_OK
world
```

Terminate application

Any application can be terminated.

1. Press CTRL+C
   ➔ The connection between the two devices will be shut down accordingly.

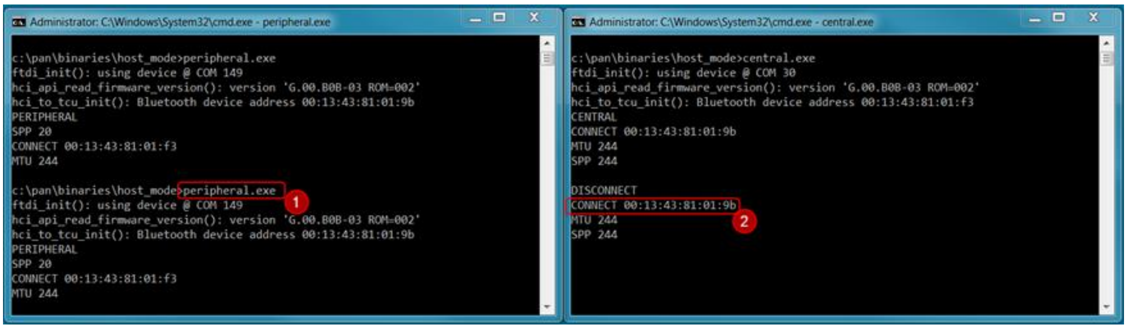
Restart after pressing CTRL+C

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 the following example, `peripheral.exe`(1) was interrupted by pressing CTRL+C and the connection between the two devices was terminated.

Afterwards the corresponding PAN1762 USB stick was reset by pressing the reset button.

Then `peripheral.exe`(1) was started again and the connection was immediately established again(2).

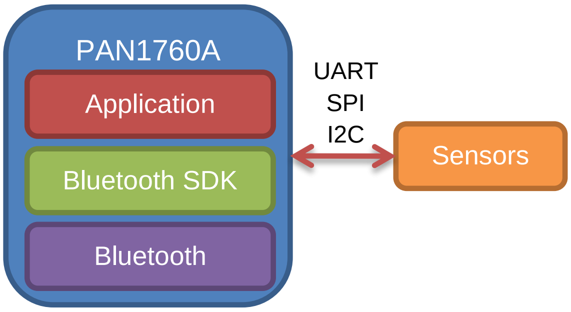

Administrator: C:\Windows\System32\cmd.exe - peripheral.exe
1
Administrator: C:\Windows\System32\cmd.exe - central.exe
2

```
c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244

c:\pan\binaries\host_mode>peripheral.exe
ftdi_init(): using device @ COM 149
hci_api_read_firmware_version(): version 'G.00.B0B-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:9b
PERIPHERAL
SPP 20
CONNECT 00:13:43:81:01:f3
MTU 244
```

```
c:\pan\binaries\host_mode>central.exe
ftdi_init(): using device @ COM 30
hci_api_read_firmware_version(): version 'G.00.B08-03 ROM=002'
hci_to_tcu_init(): Bluetooth device address 00:13:43:81:01:f3
CENTRAL
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244

DISCONNECT
CONNECT 00:13:43:81:01:9b
MTU 244
SPP 244
```

9 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.

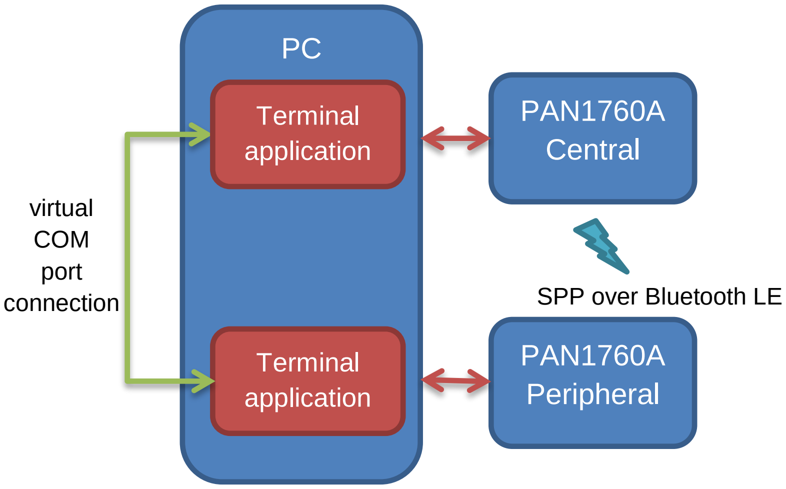


The on-board “FTDI USB UART” is available here as well and can be used by the application running on PAN1760A module to additionally interface with the PC it is connected to.

9.1 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 SPP over Bluetooth LE profile and comes in two project configurations: one will compile as a firmware for a central device and as a firmware for a peripheral device.



These two 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, two terminal applications can be started and used to transfer data back and forth.

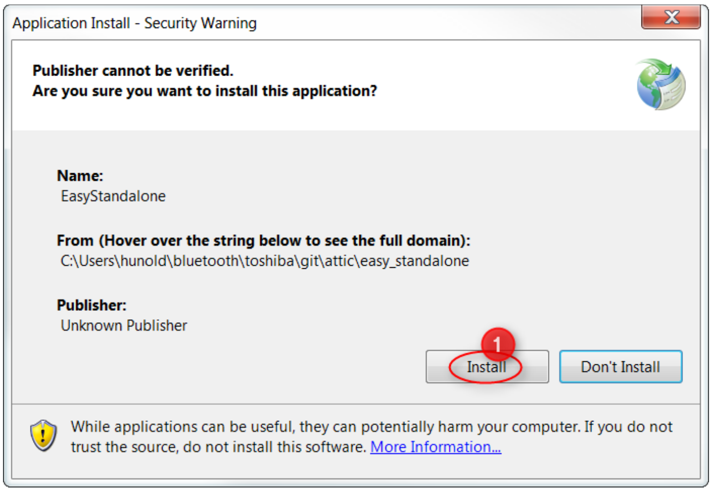
The software package includes pre-compiled binaries that can be used for testing and are available under the installation directory in the subdirectory `binaries/standalone_mode`.

9.2 Installing EasyStandalone

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

If the tool **unzipper** has been used as explained in ⇨ 3.1 Unzipper Tool then the installer for the tool **EasyStandalone** will be present in the directory `binaries\EasyStandalone`.

1. Execute `setup.exe` from the directory **binaries\EasyStandalone**.
2. Click **Install**(1).



Application Install - Security Warning

Publisher cannot be verified.
Are you sure you want to install this application?

Name:
EasyStandalone

From (Hover over the string below to see the full domain):
C:\Users\hunold\bluetooth\toshiba\git\attic\easy_standalone

Publisher:
Unknown Publisher

While applications can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information...

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

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

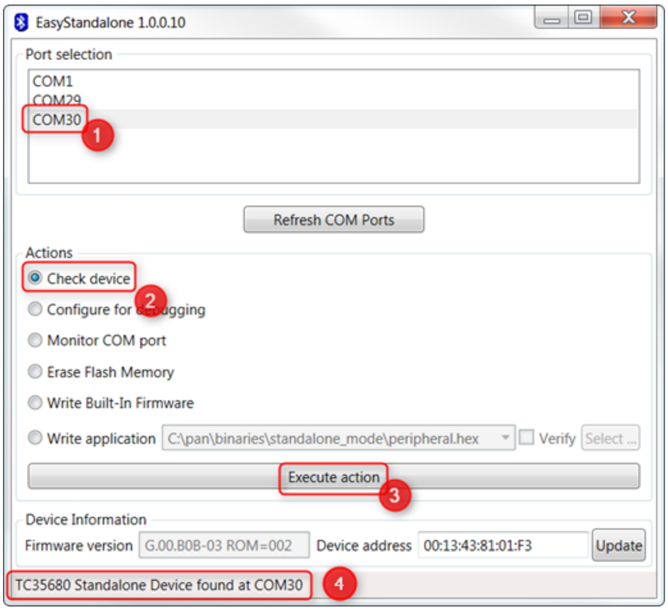
9.3 Programming the Firmware

The following requirements must be met:

✓ All of the attached PAN1760A USB sticks are removed

✓ All the PAN1760A USB sticks are jumpered to host mode as explained in ⇨ 4.4.1 Host Mode

1. Attach one of the PAN1760A USB sticks.
2. Go to the **Start menu** and launch **EasyStandalone**.
3. Make sure that the COM port selection is correct because **EasyStandalone** does not have an automatic detection.
4. Select a COM port(1) from the list of devices in the **Port selection**.



EasyStandalone 1.0.0.10

Port selection: COM1, COM29, COM30 (selected)

Actions:
- ● Check device
- ○ Configure for debugging
- ○ Monitor COM port
- ○ Erase Flash Memory
- ○ Write Built-In Firmware
- ○ Write application C:\pan\binaries\standalone_mode\peripheral.hex ☐ Verify

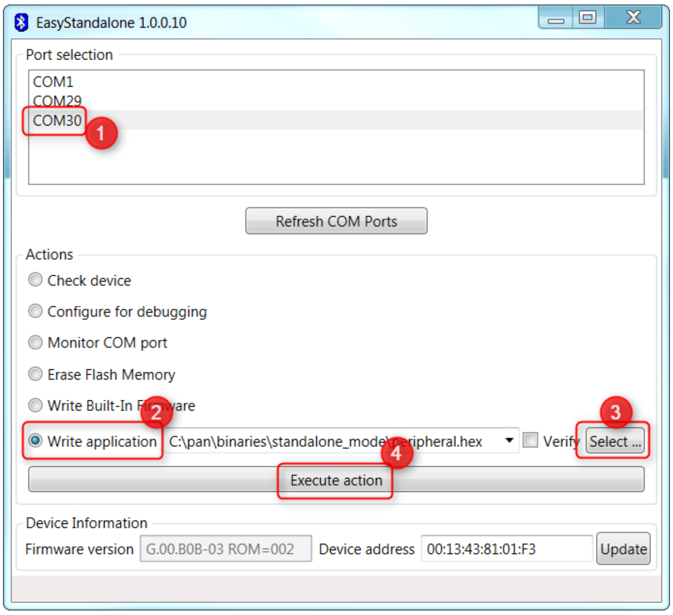
Device Information
Firmware version G.00.B0B-03 ROM=002
Device address 00:13:43:81:01:F3

TC35680 Standalone Device found at COM30

5. Choose **Check device**(2) and click **Execute action**(3).

   ➔ The firmware version and the Bluetooth device address will be displayed(4).

6. Make sure that the COM port selection(1) is correct.
7. Choose **Write application**(2).



EasyStandalone 1.0.0.10

Port selection: COM1, COM29, COM30 (selected)

Actions:
- ○ Check device
- ○ Configure for debugging
- ○ Monitor COM port
- ○ Erase Flash Memory
- ○ Write Built-In Firmware
- ● Write application C:\pan\binaries\standalone_mode\peripheral.hex ☐ Verify

Device Information
Firmware version G.00.B0B-03 ROM=002
Device address 00:13:43:81:01:F3

8. Click **Select…**(3) to choose the `peripheral.hex` firmware from the subdirectory `binaries/standalone_mode`.

9. Click **Execute action**(4) to start the programming.
   - ➔ A progress bar will be shown and after a couple of seconds the programming will be finished.
10. Now **EasyStandalone** can be closed.
11. Jumper the PAN1760A USB stick to “standalone mode” as explained in ⇨ 4.4.2 Standalone Mode.

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

9.4 Testing the Setup

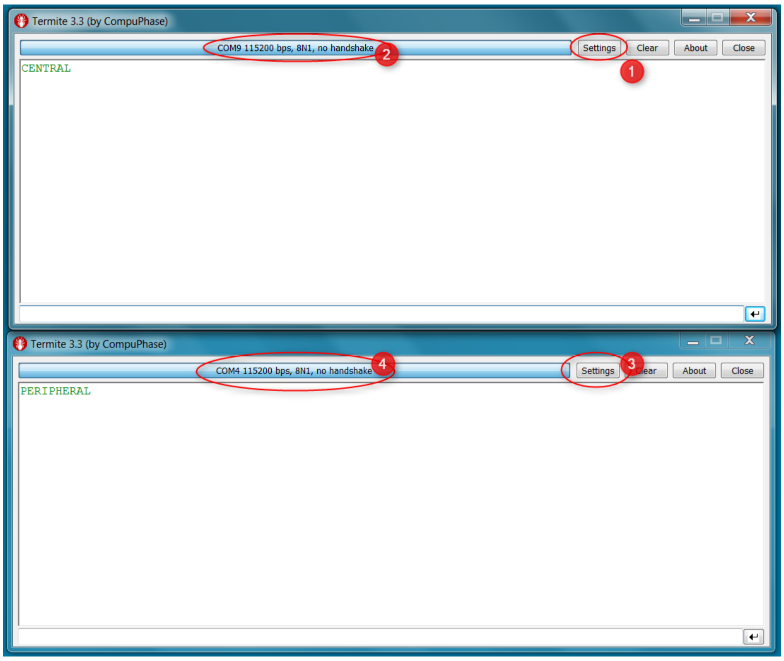
The following requirements must be met:

- ✓ Both PAN1760A USB sticks are jumpered to “standalone mode” as explained in ⇨ 4.4.2 Standalone Mode.

1. Use any terminal application to connect to the well-known COM ports of the t PAN1760A USB sticks.

If you do not have any terminal application yet, you can try **Termite** from https://www.compuphase.com/software_termite.htm. In the following steps **Termite** is used.

2. Click **Settings**(1) to choose the desired COM port.
   - ➔ The chosen COM port will be displayed(2).



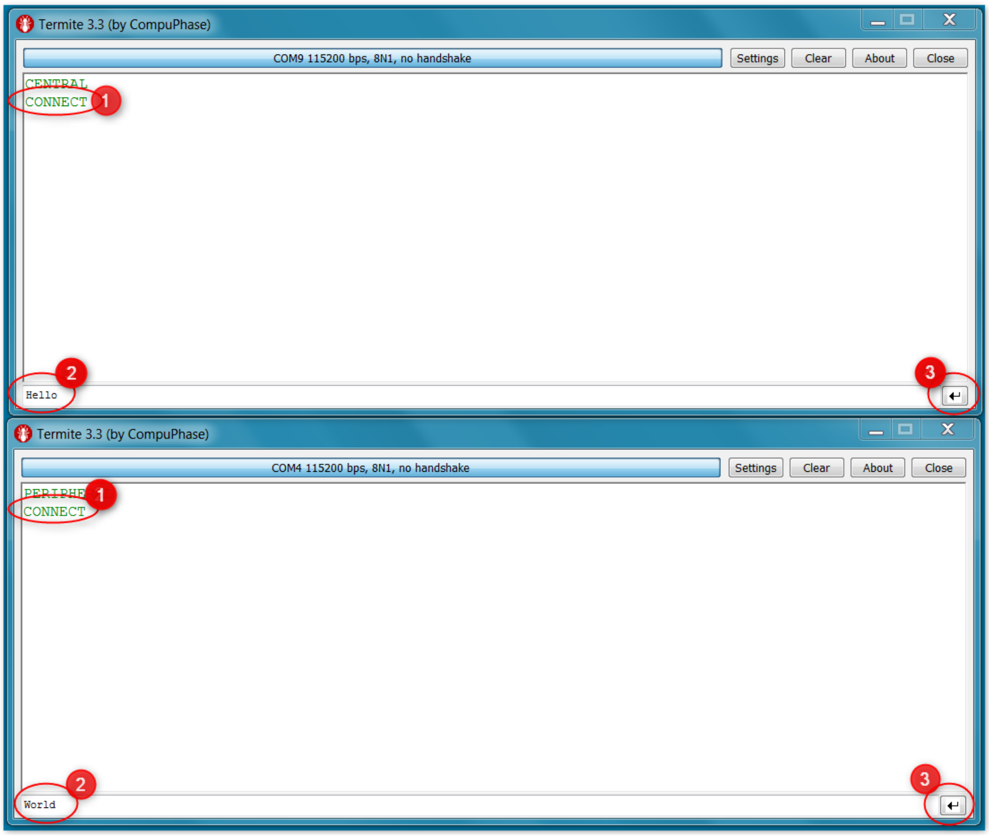
COM9 115200 bps, 8N1, no handshake

```
CENTRAL
```

COM4 115200 bps, 8N1, no handshake

```
PERIPHERAL
```

3. Press the “reset” button on the PAN1760A USB sticks.
   ➔ They will identify themselves by printing either **CENTRAL** or **PERIPHERAL** to the terminal window.



COM9 115200 bps, 8N1, no handshake

```
CENTRAL
CONNECT
```

Hello

COM4 115200 bps, 8N1, no handshake

```
PERIPHE
CONNECT
```

World

The central device will immediately try to connect to the peripheral device. If a connection has been established, **CONNECT**(1) will be printed in each of the terminal windows.

Now you can use the input textbox(2) and the “send” button(3) to send data back and forth between the two devices.

10 Development

The source code for all the sample applications is available under the directory `configurations`.

How to recompile the application from the source code for the different usage scenarios is explained in the following chapters.

10.1 Host Mode

10.1.1 Visual Studio® 2017

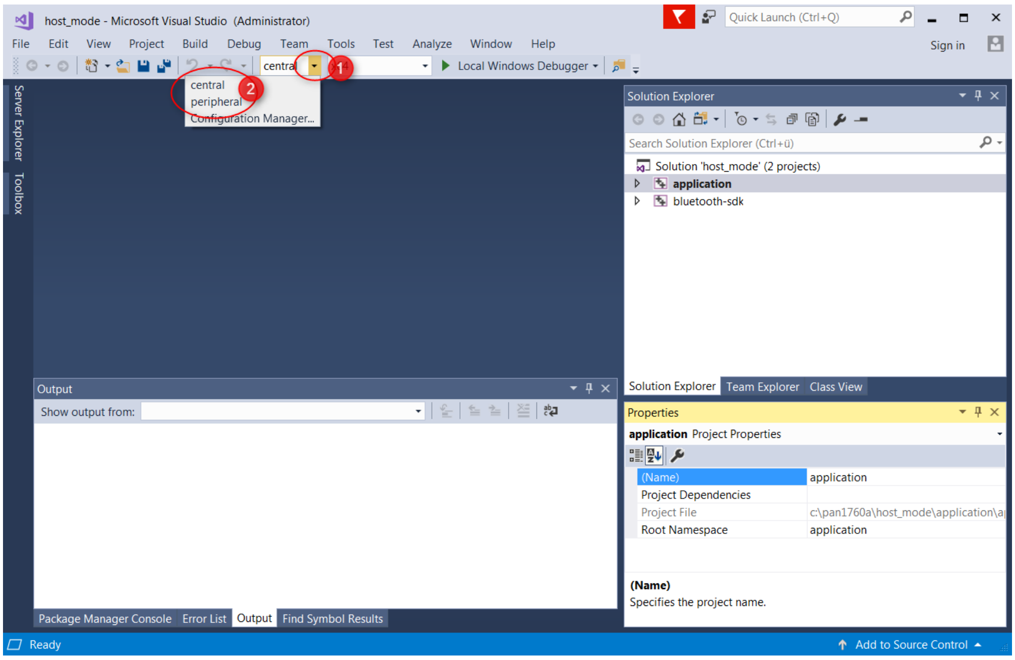
All sample application projects for the host mode that run on the PC are based on **Visual Studio 2017**. It is sufficient to install the **Visual Studio 2017 Community** edition.

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

10.1.2 Working with the Sample Project

The sample project solution file is located under `host_mode\host_mode.sln`.

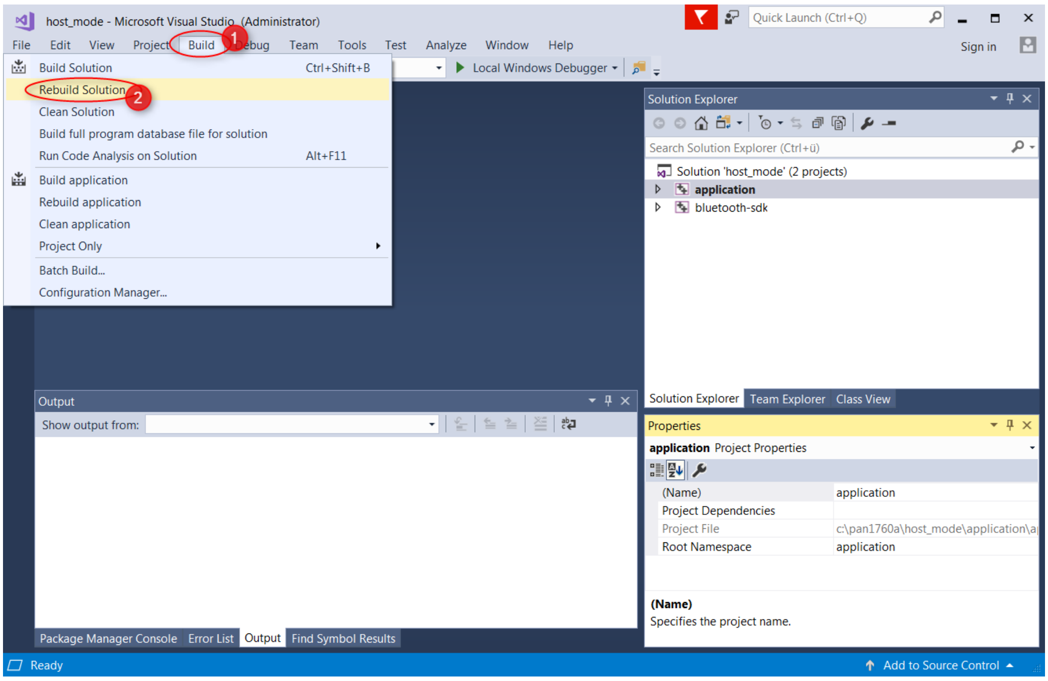
1. Open the sample project in **Visual Studio 2017**.
2. Click on the arrow(1) “Solution Configurations” to select the configuration from the drop down list(2).



(Name)	application
Project Dependencies	
Project File	c:\pan1760a\host_mode\application\a
Root Namespace	application

(Name)
Specifies the project name.

3. Navigate to **Build**(1) and select **Rebuild Solution**(2) to build the currently selected configuration.



(Name) application
Project Dependencies
Project File c:\pan1760a\host_mode\application\a
Root Namespace application

(Name)
Specifies the project name.

➔ The resulting binaries can be found in the directory `host_mode\x64\` as `periperal.exe` and `central.exe`.

10.2 AT Command Mode

10.2.1 Visual Studio® 2017

All sample application projects for the AT command mode that run on the PC are based on **Visual Studio 2017**. It is sufficient to install the **Visual Studio 2017 Community** edition.

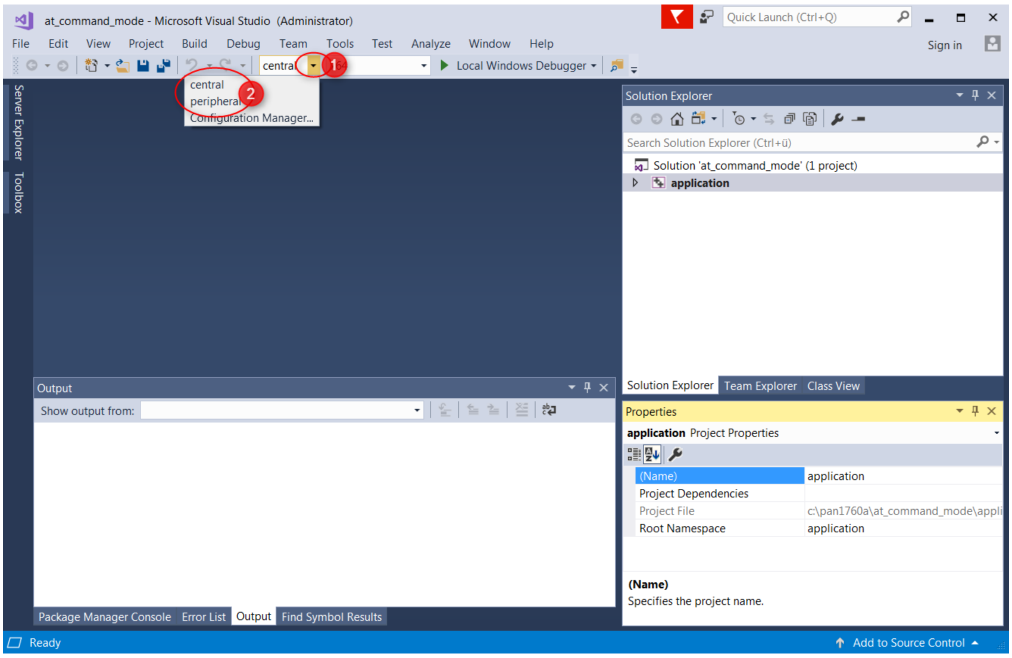
For further information please visit https://www.visualstudio.com/downloads/.

10.2.2 Working with the Sample Project

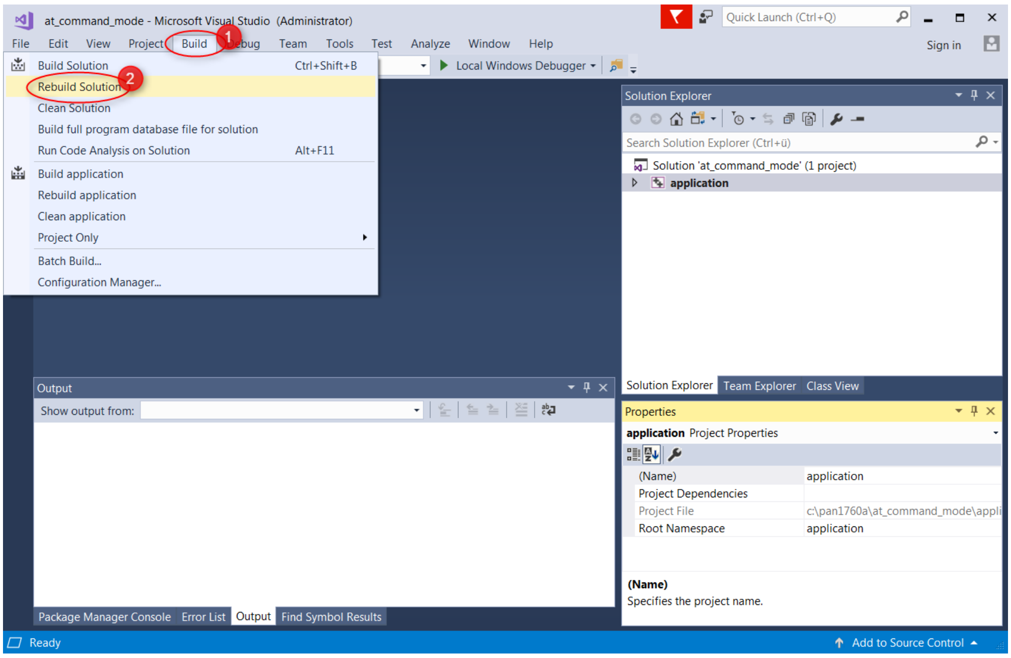
The sample project solution file is located under `at_command_mode\at_command_mode.sln`.

1. Open the sample project in **Visual Studio 2017**.

2. Click on the arrow(1) “Solution Configurations” to select the configuration from the drop down list(2).


at_command_mode - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
central
1
Local Windows Debugger
central
peripheral
2
Configuration Manager...
Server Explorer
Toolbox
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'at_command_mode' (1 project)
application
Output
Show output from:
Solution Explorer Team Explorer Class View
Properties
application Project Properties
(Name) application
Project Dependencies
Project File c:\pan1760a\at_command_mode\appli
Root Namespace application
(Name)
Specifies the project name.
Package Manager Console Error List Output Find Symbol Results
Ready
Add to Source Control

3. Navigate to **Build**(1) and select **Rebuild Solution**(2) to build the currently selected configuration.


at_command_mode - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
1
Sign in
Build Solution Ctrl+Shift+B
Rebuild Solution
2
Clean Solution
Build full program database file for solution
Run Code Analysis on Solution Alt+F11
Build application
Rebuild application
Clean application
Project Only
Batch Build...
Configuration Manager...
Local Windows Debugger
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'at_command_mode' (1 project)
application
Output
Show output from:
Solution Explorer Team Explorer Class View
Properties
application Project Properties
(Name) application
Project Dependencies
Project File c:\pan1760a\at_command_mode\appli
Root Namespace application
(Name)
Specifies the project name.
Package Manager Console Error List Output Find Symbol Results
Ready
Add to Source Control

➔ The resulting binaries can be found in the directory `at_command_mode\x64\` as `periperal.exe`.

10.3 Standalone Mode

10.3.1 IAR Embedded Workbench

All the sample application projects that run on the embedded Cortex-M0 microcontroller of PAN1760A are based on **IAR Embedded Workbench**.

Because the code sizes of any of the projects exceed 32 kB 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/.

10.3.2 Working with the Sample Project – Windows

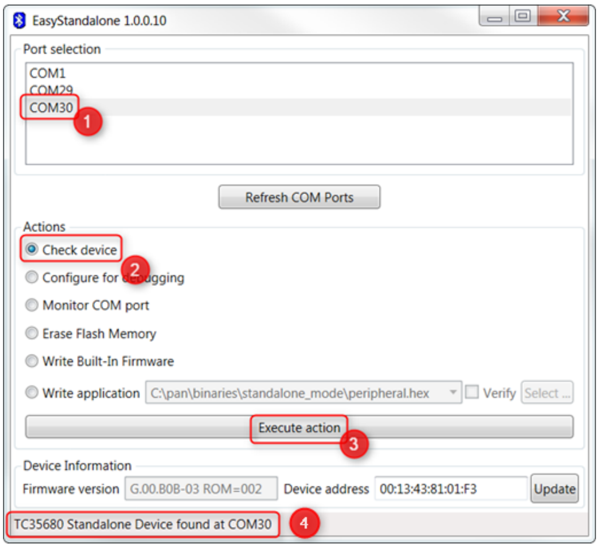
10.3.2.1 Device Preparation

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

The following requirements must be met:

- ✓ All of the attached PAN1760A USB sticks are removed.
- ✓ All the PAN1760A USB sticks are jumpered to "host mode" as explained in ⇨ 4.4.1 Host Mode

1. Attach one of the PAN1760A USB sticks.
2. Go to the **Start menu** and launch **EasyStandalone**.
3. Make sure that the COM port selection is correct because **EasyStandalone** does not have an automatic detection.
4. Select a COM port(1) from the list of devices in the **Port selection**.



EasyStandalone 1.0.0.10

Port selection: COM30

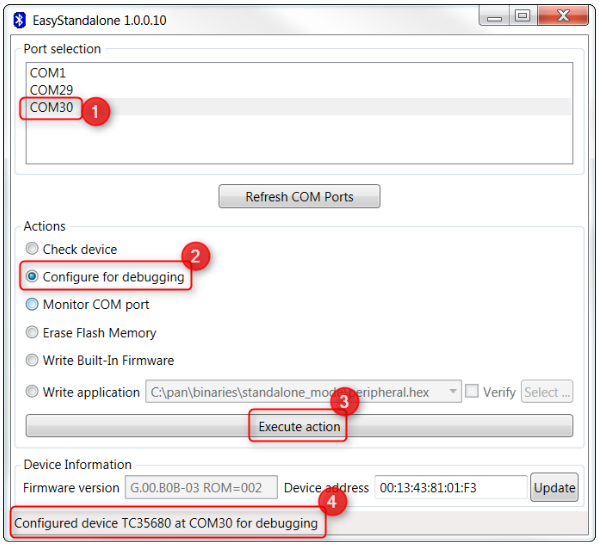
Actions:
● Check device
○ Configure for debugging
○ Monitor COM port
○ Erase Flash Memory
○ Write Built-In Firmware
○ Write application C:\pan\binaries\standalone_mode\peripheral.hex ☐ Verify

Device Information
Firmware version G.00.B0B-03 ROM=002
Device address 00:13:43:81:01:F3

TC35680 Standalone Device found at COM30

5. Choose **Check device**(2) and click **Execute action**(3).
   ➔ The firmware version and the Bluetooth device address will be displayed(4).
6. Choose **Configure for debugging**(2).

7. Click **Execute action**(3) again.



Write application: C:\pan\binaries\standalone_mod eripheral.hex

Firmware version: G.00.B0B-03 ROM=002

Device address: 00:13:43:81:01:F3

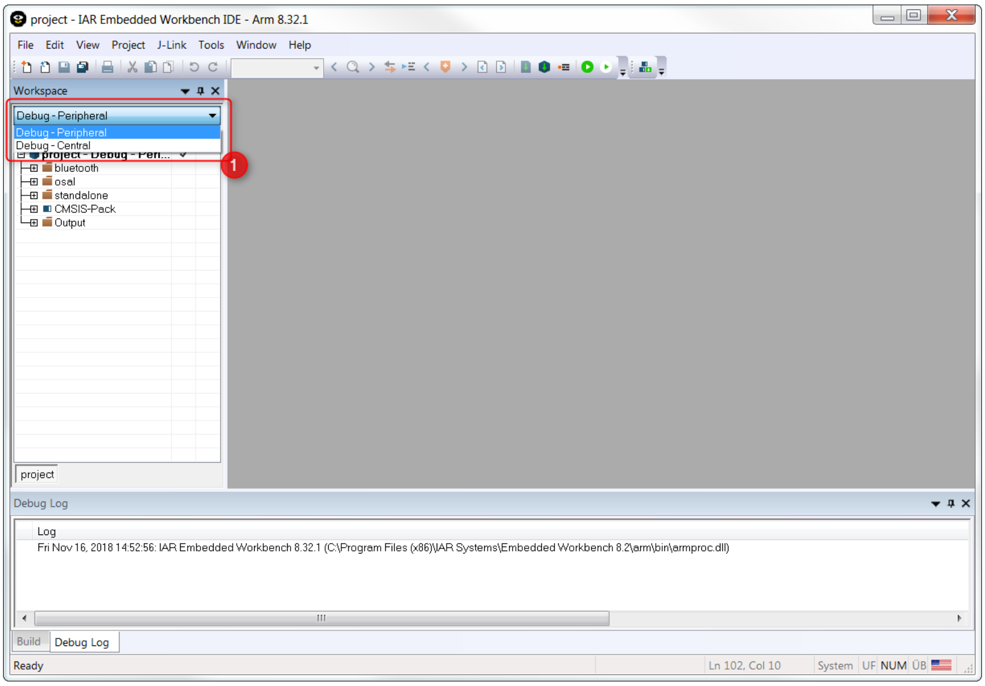
Configured device TC35680 at COM30 for debugging

➔ The in-device flash memory will now be erased(4).

10.3.2.2 Compilation

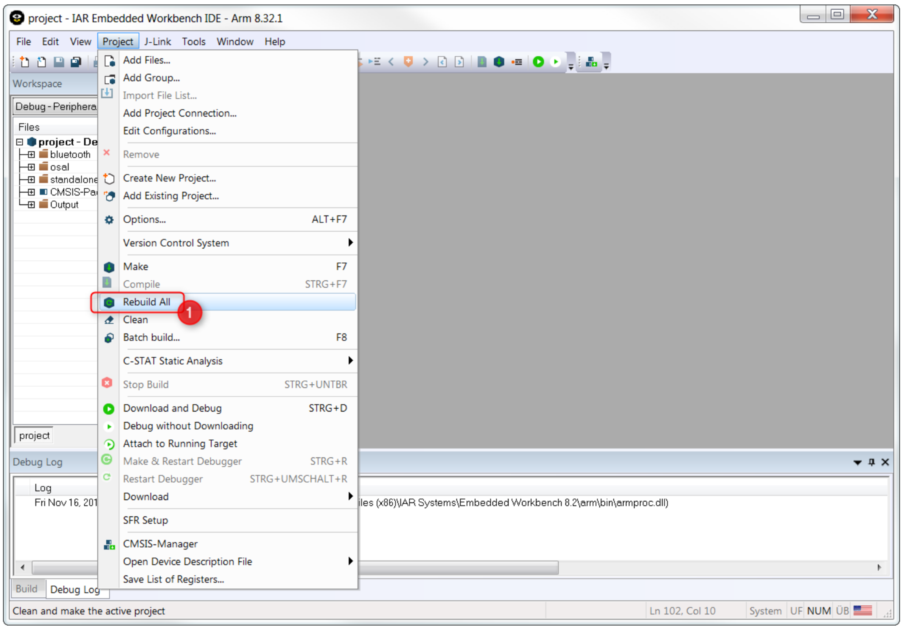
The sample project solution file is located under `standalone_mode\project.eww`.

1. Open the sample project in **IAR Embedded Workbench**.
2. Click on the Workspace “Configuration” drop down list(1) to select the configuration.



Fri Nov 16, 2018 14:52:56: IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)

3. Navigate to **Project** and select **Rebuild All**(1) to build the currently selected configuration.



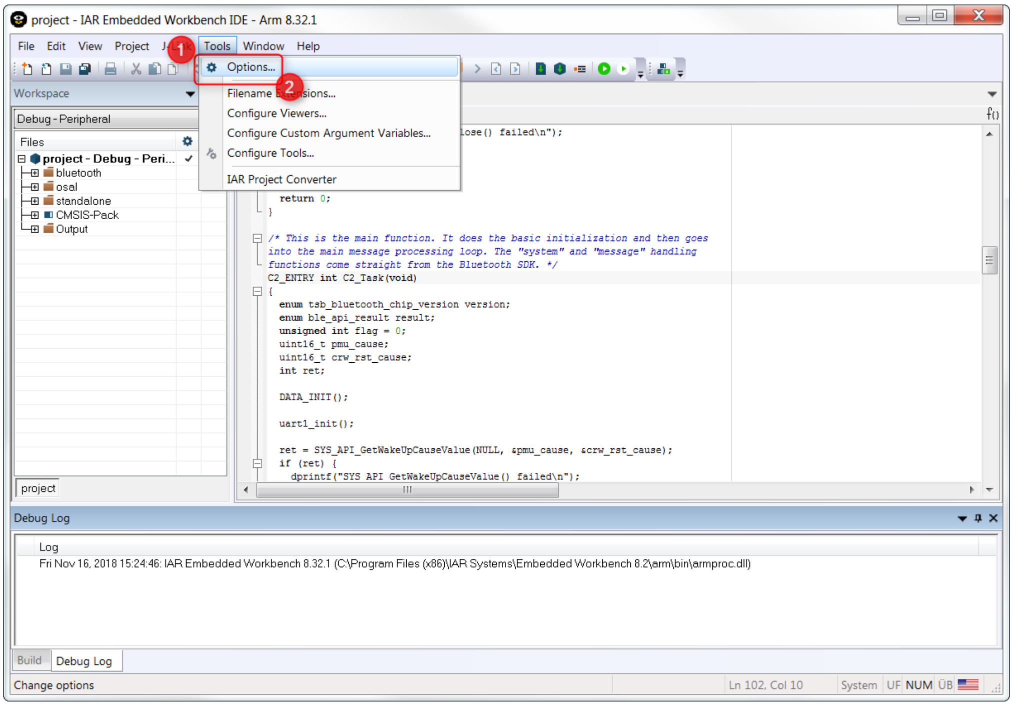
```
Fri Nov 16, 201
iles (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)
```

10.3.2.3 IDE Configuration

Avoid an annoying warning message

In order to avoid an annoying warning message it is necessary to change the configuration of the **IAR Embedded Workbench**.

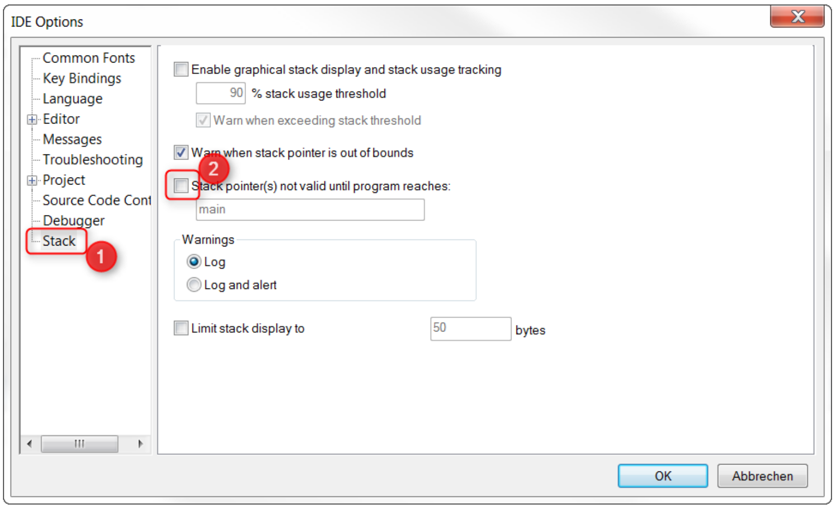
1. Navigate to **Tools**(1) and choose **Options…**(2).



```
lose() failed\n");

  return 0;
}

/* This is the main function. It does the basic initialization and then goes
into the main message processing loop. The "system" and "message" handling
functions come straight from the Bluetooth SDK. */
C2_ENTRY int C2_Task(void)
{
  enum tsb_bluetooth_chip_version version;
  enum ble_api_result result;
  unsigned int flag = 0;
  uint16_t pmu_cause;
  uint16_t crw_rst_cause;
  int ret;

  DATA_INIT();

  uart1_init();

  ret = SYS_API_GetWakeUpCauseValue(NULL, &pmu_cause, &crw_rst_cause);
  if (ret) {
    dprintf("SYS API GetWakeUpCauseValue() failed\n");
```

```
Fri Nov 16, 2018 15:24:46: IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)
```

2. Navigate to Stack(1) to make sure that the option **Stack pointer(s) not valid until program reaches**(2) is not selected.



☐ Enable graphical stack display and stack usage tracking
90 % stack usage threshold
☑ Warn when exceeding stack threshold
☑ Warn when stack pointer is out of bounds
☐ Stack pointer(s) not valid until program reaches:
main

Warnings
● Log
○ Log and alert

☐ Limit stack display to 50 bytes

The reason is that the application does not contain a traditional function `main()` but uses a different name instead.

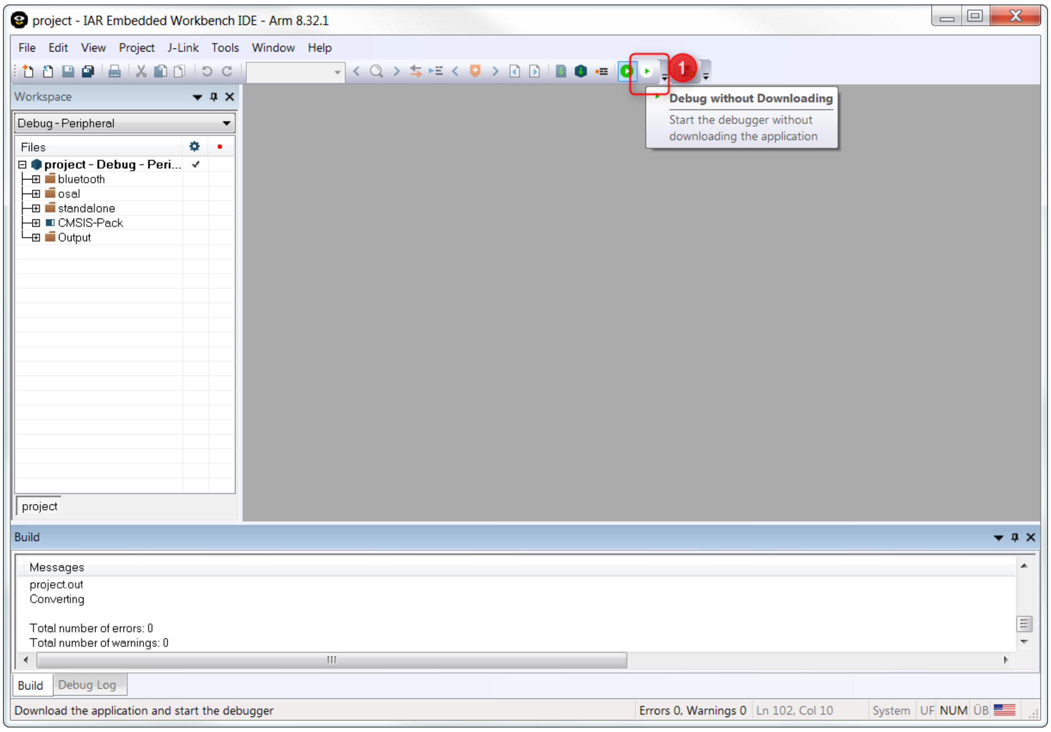
10.3.2.4 Debugging

The following requirements must be met:

✓ The PAN1760A USB stick is jumpered to “standalone mode” as explained in ⇨ 4.4.2 Standalone Mode

✓ The in-device flash memory is erased as explained in ⇨ 10.3.2.1 Device Preparation

1. Click the option **Debug without Downloading**(1) to start the debug process.

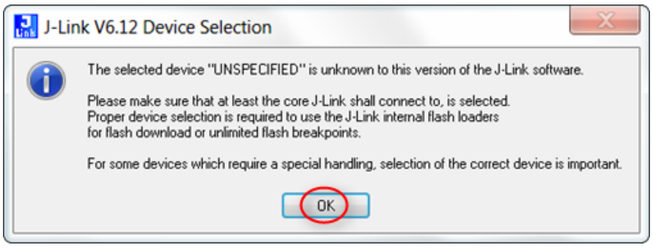


Debug without Downloading
Start the debugger without downloading the application

```
Messages
project.out
Converting

Total number of errors: 0
Total number of warnings: 0
```

Download the application and start the debugger   Errors 0, Warnings 0   Ln 102, Col 10   System   UF NUM ÜB

➔ At the very first start the following message box will appear.



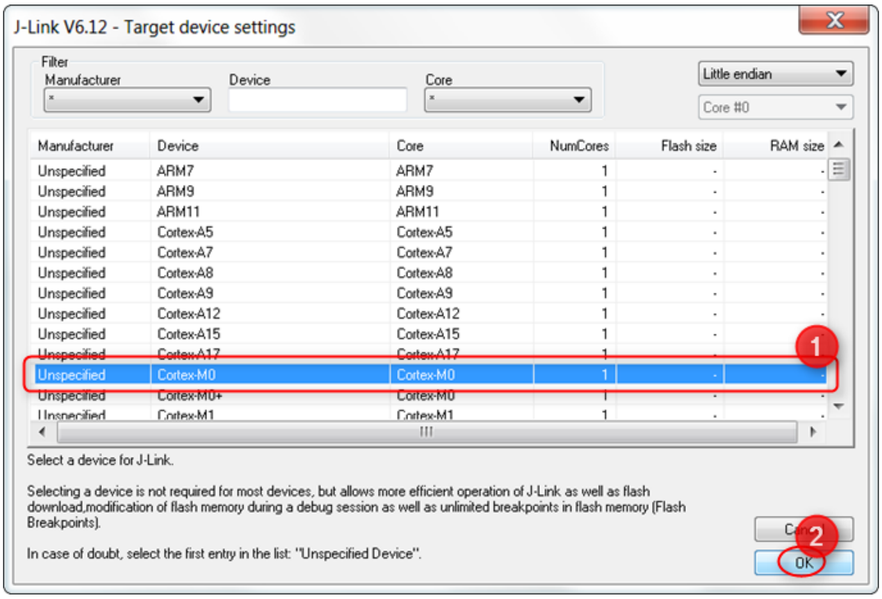
J-Link V6.12 Device Selection

The selected device "UNSPECIFIED" is unknown to this version of the J-Link software.

Please make sure that at least the core J-Link shall connect to, is selected.
Proper device selection is required to use the J-Link internal flash loaders
for flash download or unlimited flash breakpoints.

For some devices which require a special handling, selection of the correct device is important.

2. Click **OK**.
3. Choose from the list of devices **Unspecified/Cortex-M0**(1) and click **OK**(2).



J-Link V6.12 - Target device settings

Filter
Manufacturer: *
Device:
Core: *
Little endian
Core #0

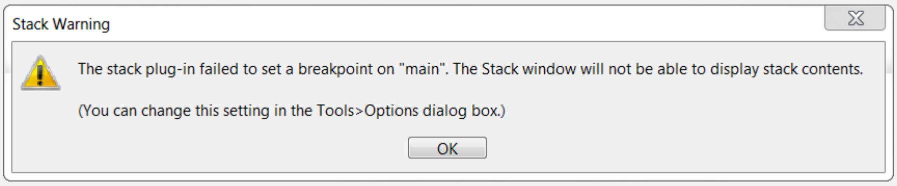
Manufacturer	Device	Core	NumCores	Flash size	RAM size
Unspecified	ARM7	ARM7	1	-	-
Unspecified	ARM9	ARM9	1	-	-
Unspecified	ARM11	ARM11	1	-	-
Unspecified	Cortex-A5	Cortex-A5	1	-	-
Unspecified	Cortex-A7	Cortex-A7	1	-	-
Unspecified	Cortex-A8	Cortex-A8	1	-	-
Unspecified	Cortex-A9	Cortex-A9	1	-	-
Unspecified	Cortex-A12	Cortex-A12	1	-	-
Unspecified	Cortex-A15	Cortex-A15	1	-	
Unspecified	Cortex-A17	Cortex-A17	1	-	
Unspecified	Cortex-M0	Cortex-M0	1	-	-
Unspecified	Cortex-M0+	Cortex-M0	1	-	-
Unspecified	Cortex-M1	Cortex-M1	1	-	-

Select a device for J-Link.

Selecting a device is not required for most devices, but allows more efficient operation of J-Link as well as flash download,modification of flash memory during a debug session as well as unlimited breakpoints in flash memory (Flash Breakpoints).

In case of doubt, select the first entry in the list: "Unspecified Device".

If the following warning message appears, please make sure that you followed the project configuration advice in ⇨ 10.3.2.3 IDE Configuration.

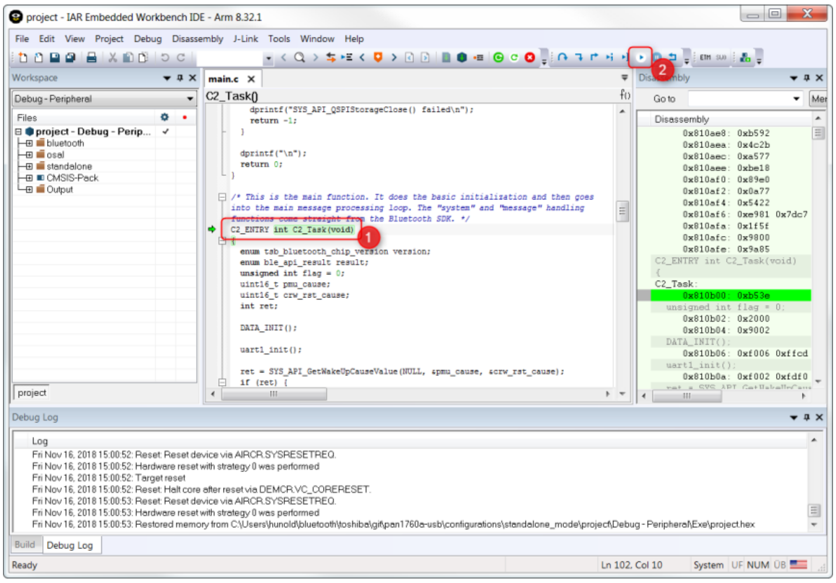


Stack Warning

The stack plug-in failed to set a breakpoint on "main". The Stack window will not be able to display stack contents.

(You can change this setting in the Tools>Options dialog box.)

➔ The application will now be started up but when it reaches the funtion `C2_Task()` (1) the application will be immediately interrupted.



```
    dprintf("SYS_API_QSPIStorageClose() failed\n");
    return -1;
  }

  dprintf("\n");
  return 0;
}

/* This is the main function. It does the basic initialization and then goes
into the main message processing loop. The "system" and "message" handling
functions come straight from the Bluetooth SDK. */
C2_ENTRY int C2_Task(void)
{
  enum tsb_bluetooth_chip_version version;
  enum ble_api_result result;
  unsigned int flag = 0;
  uint16_t pmu_cause;
  uint16_t crw_rst_cause;
  int ret;

  DATA_INIT();

  uart1_init();

  ret = SYS_API_GetWakeUpCauseValue(NULL, &pmu_cause, &crw_rst_cause);
  if (ret) {
```

```
Disassembly
        0x810ae8: 0xb592
        0x810aea: 0x4c2b
        0x810aec: 0xa577
        0x810aee: 0xbe18
        0x810af0: 0x89e0
        0x810af2: 0x0a77
        0x810af4: 0x5422
        0x810af6: 0xe981 0x7dc7
        0x810afa: 0x1f5f
        0x810afc: 0x9800
        0x810afe: 0x9a85
  C2_ENTRY int C2_Task(void)
  {
  C2_Task:
        0x810b00: 0xb53e
    unsigned int flag = 0;
        0x810b02: 0x2000
        0x810b04: 0x9002
    DATA_INIT();
        0x810b06: 0xf006 0xffcd
    uart1_init();
        0x810b0a: 0xf002 0xfdf0
```

```
Log
Fri Nov 16, 2018 15:00:52: Reset: Reset device via AIRCR.SYSRESETREQ.
Fri Nov 16, 2018 15:00:52: Hardware reset with strategy 0 was performed
Fri Nov 16, 2018 15:00:52: Target reset
Fri Nov 16, 2018 15:00:52: Reset: Halt core after reset via DEMCR.VC_CORERESET.
Fri Nov 16, 2018 15:00:53: Reset: Reset device via AIRCR.SYSRESETREQ.
Fri Nov 16, 2018 15:00:53: Hardware reset with strategy 0 was performed
Fri Nov 16, 2018 15:00:53: Restored memory from C:\Users\hunold\bluetooth\toshiba\git\pan1760a-usb\configurations\standalone_mode\project\Debug - Peripheral\Exe\project.hex
```

4. Click the icon “Go”(2) to continue the application.

11 Appendix

11.1 Changes to the Toshiba Bluetooth SDK

During the installation using the tool **unzipper** extracts the Toshiba Bluetooth SDK to the subdirectory `sdk`.

As explained all the sample projects in the software package internally use the Toshiba Bluetooth SDK.

Some changes have been done to the Toshiba Bluetooth SDK to add missing features. Other changes have been done to circumvent problems. All these changes are regularly submitted to Toshiba for further inspection. The might not be adopted in upcoming versions of the SDK.

All changed files are contained in the file `bluetooth_sdk_differences.zip`. The individual changes are described in “unified diff” format in the file `bluetooth_sdk.diff` included in the ZIP file as well.

These changes are briefly explained in the following chapters.

11.1.1 Bluetooth Subsystem

HCI API

The function `hci_api_set_baudrate()` has been implemented to provide a unique API for switching the baud rate for all Panasonic modules.

11.1.2 Host Mode Deployment

SPP over Bluetooth LE Handling

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 SPP over Bluetooth LE profile must be configurable.

11.1.3 Standalone Deployment

Please note that the code for the SPP over Bluetooth LE profile from the host mode deployment is used.

11.2 Contact Details

11.2.1 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 website “Sales & Support” 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

11.2.2 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
http://pideu.panasonic.de/products/wireless-modules.html

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