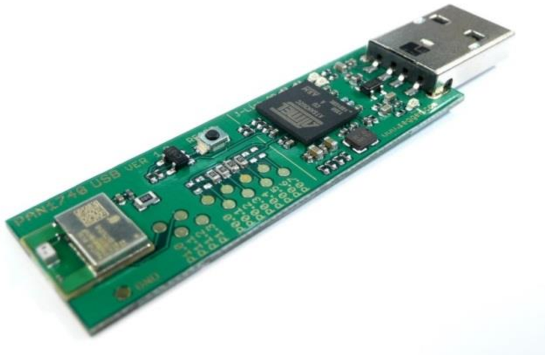
Panasonic

PAN1740/1740A

Bluetooth® Low Energy Module

Design 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. Please consult the most recently issued Design Guide before initiating or completing a design.

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

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 Design 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 Design Guide applies to the Bluetooth development modules PAN1740 and PAN1740A Experimenter Kit (PAN1740 EXP and PAN1740A EXP). The intention is to enable our customers to easily and quickly integrate Panasonic's module PAN1740 and PAN1740A in their product. The product is referred to as "the PAN1740/1740A" or "the module" within this document.

This Design Guide describes the hardware and gives useful hints.

1.2 Revision History

Revision	Date	Modifications/Remarks
1.0	2019-11-07	First 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.
✎	**Tip** Indicates useful information designed to facilitate working with the Module.
⇨ [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:** `pan1760.c` contains the actual module initialization. The message `Failed to save your data` is displayed. Enter the value `Product 123`.

1.4 Related Documents

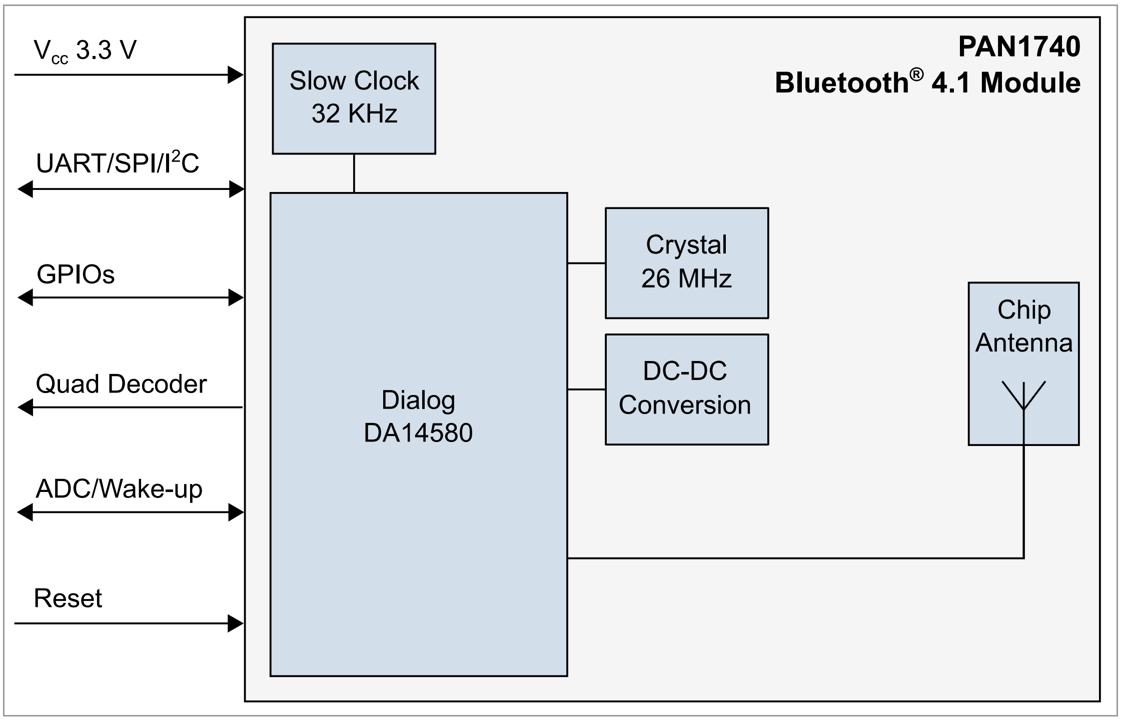
Please refer to the Panasonic website for more information as well as related documents ⇨ 10.2.2 Product Information.

2 Overview

2.1 PAN1740

The PAN1740 is a short-range Bluetooth Low Energy (LE) single mode module used for the implementation of Bluetooth functionality into various electronic devices. The PAN1740 is fully compliant with the Bluetooth 4.1 standard. It includes dedicated hardware for Link Layer Implementation of Bluetooth LE and interface controllers for enhanced connectivity capabilities.

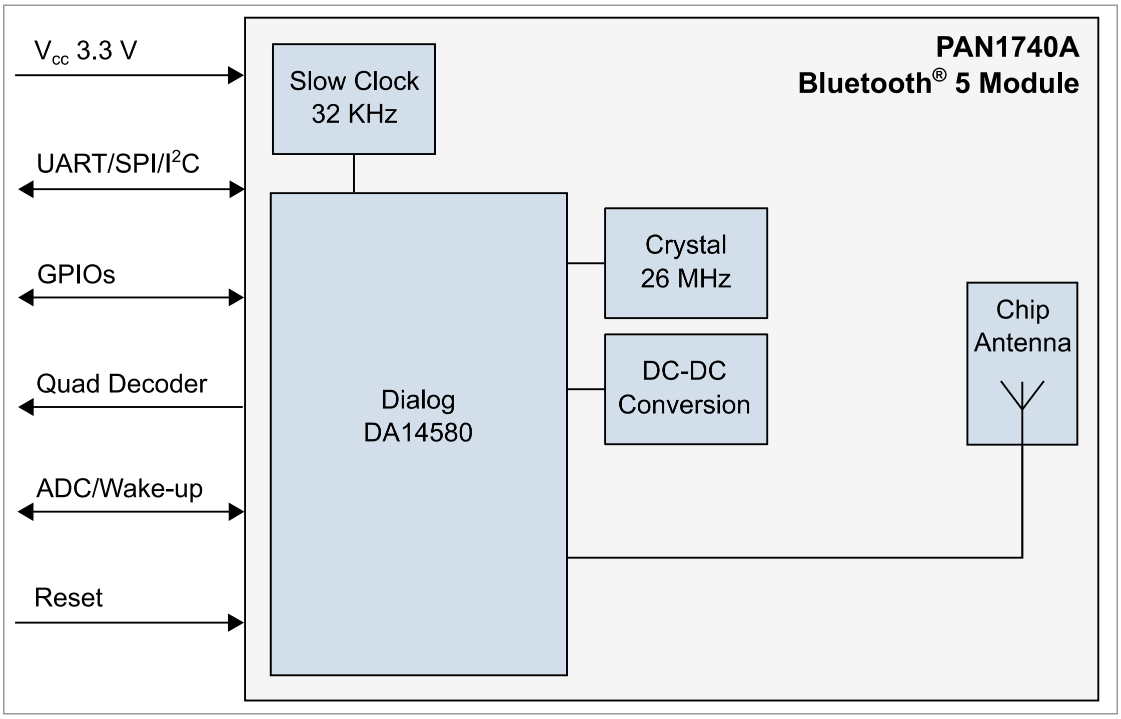
Block Diagram



2.2 PAN1740A

The PAN1740A is an optimized version of the PAN1740, offering a reduced boot time and supporting up to eight connections. It has a fully integrated radio transceiver and baseband processor for Bluetooth 5 LE. It can be used as a stand-alone application processor or as a data pump in hosted systems. The device is optimized for remote control units (RCU) requiring support for voice commands and motion/gesture recognition. Its integrated Audio Unit (AU) offers easy interface for MEMS microphones over PDM, external codecs over PCM/I^{2}S, and a Sample Rate Converter unit. The Bluetooth LE firmware includes the L2CAP service layer protocols, Security Manager (SM), Attribute Protocol (ATT), the Generic Attribute Profile (GATT), and the Generic Access Profile (GAP). All profiles published by the Bluetooth SIG as well as custom profiles are supported. The transceiver interfaces directly to the antenna and is fully compliant with the Bluetooth 5 standard. The PAN1740A has dedicated hardware for the Link Layer Implementation of Bluetooth LE and interface controllers for enhanced connectivity capabilities.

Block Diagram

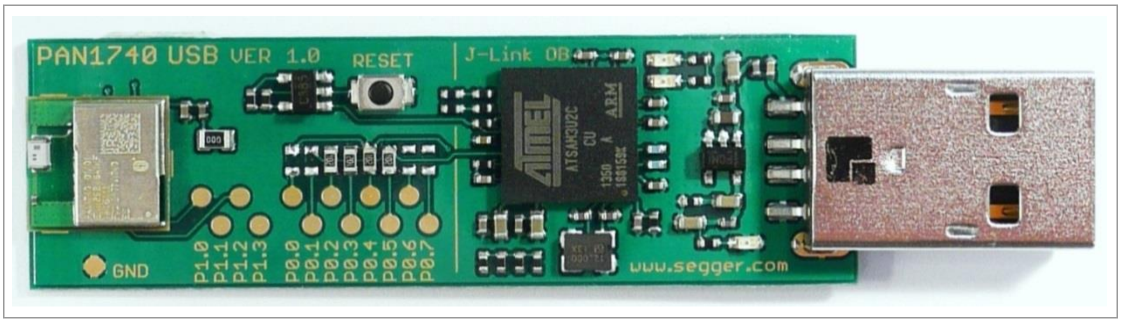


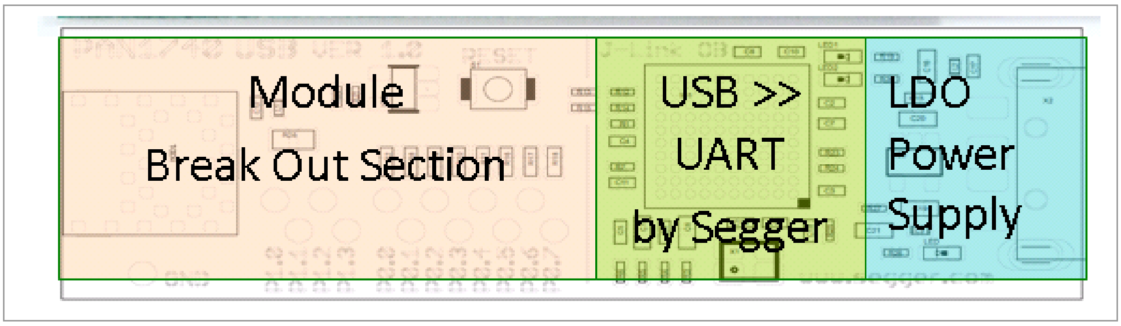
3 Reference Design

3.1 USB-Dongle

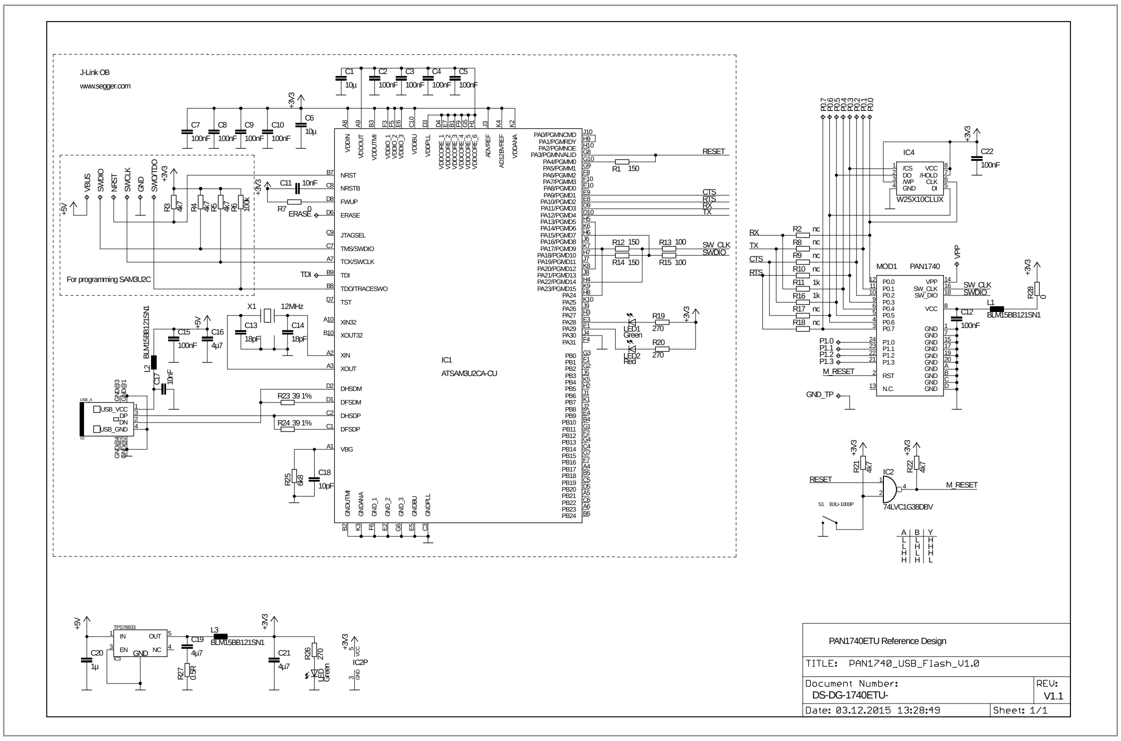
3.1.1 Functionality

- Atmel µC includes Segger USB-to-UART programmer (serial number on the backside)
- OTP cannot be damaged (fail-safe development)
- Runs with Dialog®'s Keil® Compiler projects
- Runs with Connection Manager
- Can be used for software development "on the fly"





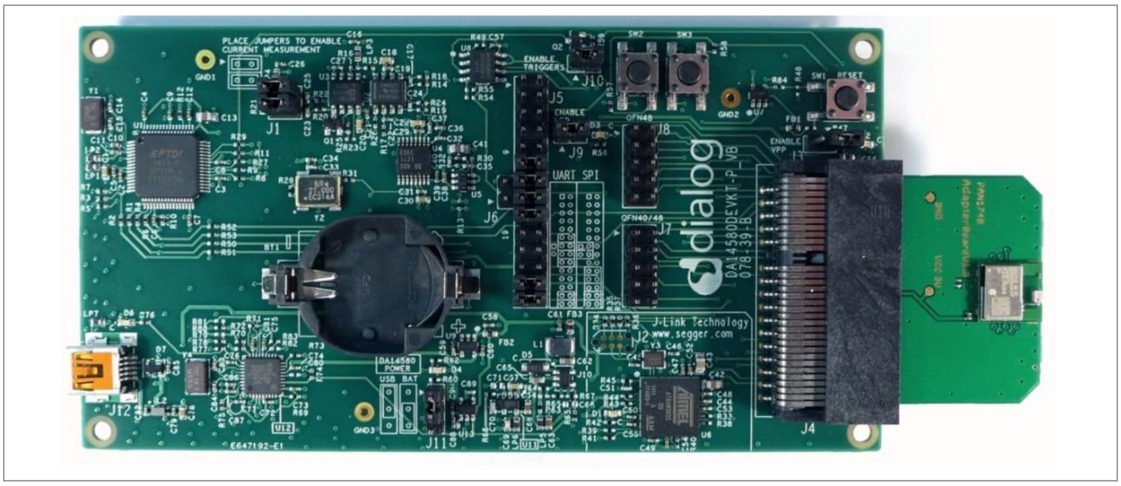
3.1.2 Schematic



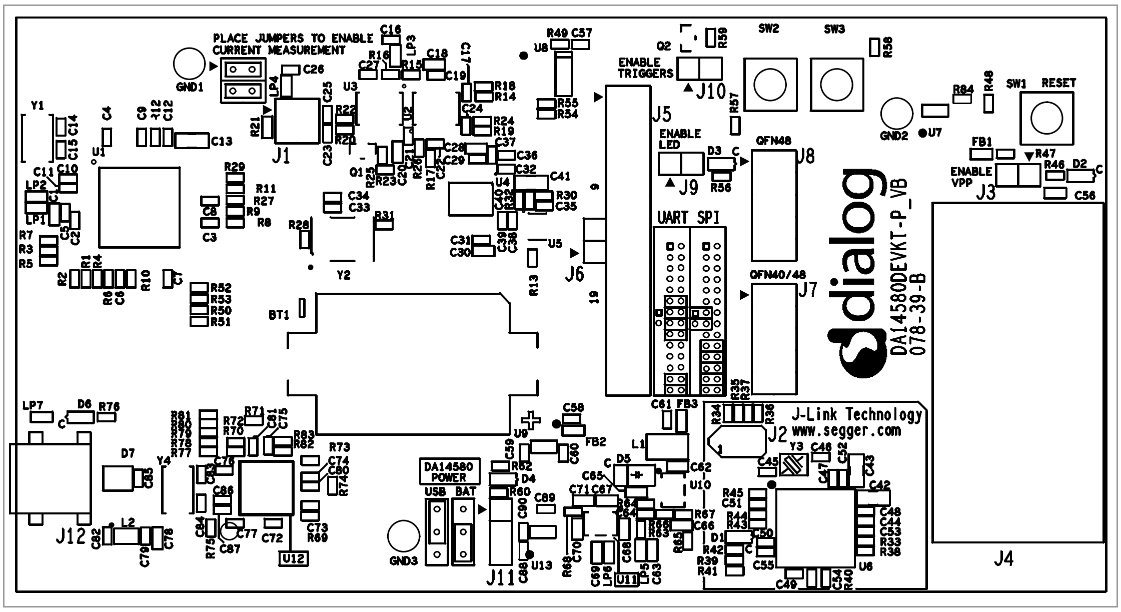
3.2 Mother Board

3.2.1 Functionality

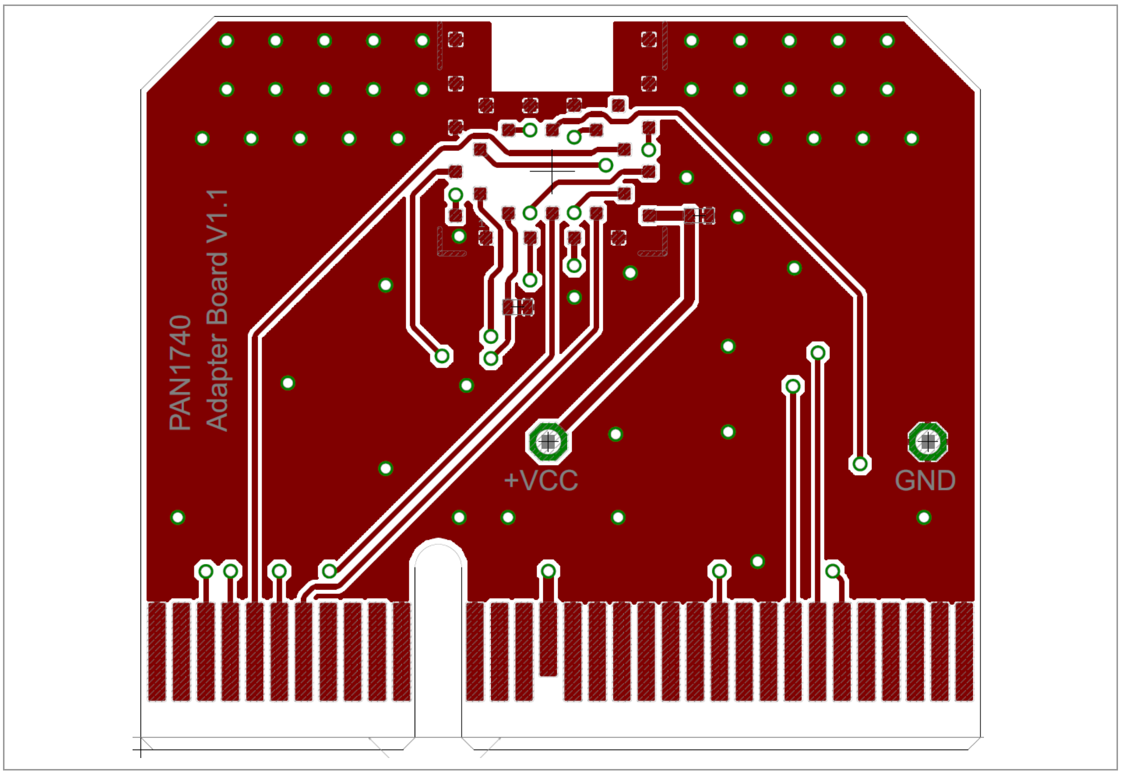
- Can be used with PAN1740/1740A Adapter Board
- OTP can be programmed
- Runs with Dialog’s Keil Compiler projects
- Runs with Connection Manager
- Can be used for software development "on the fly"
- Runs with Smart Snippets™ including Power Profiler



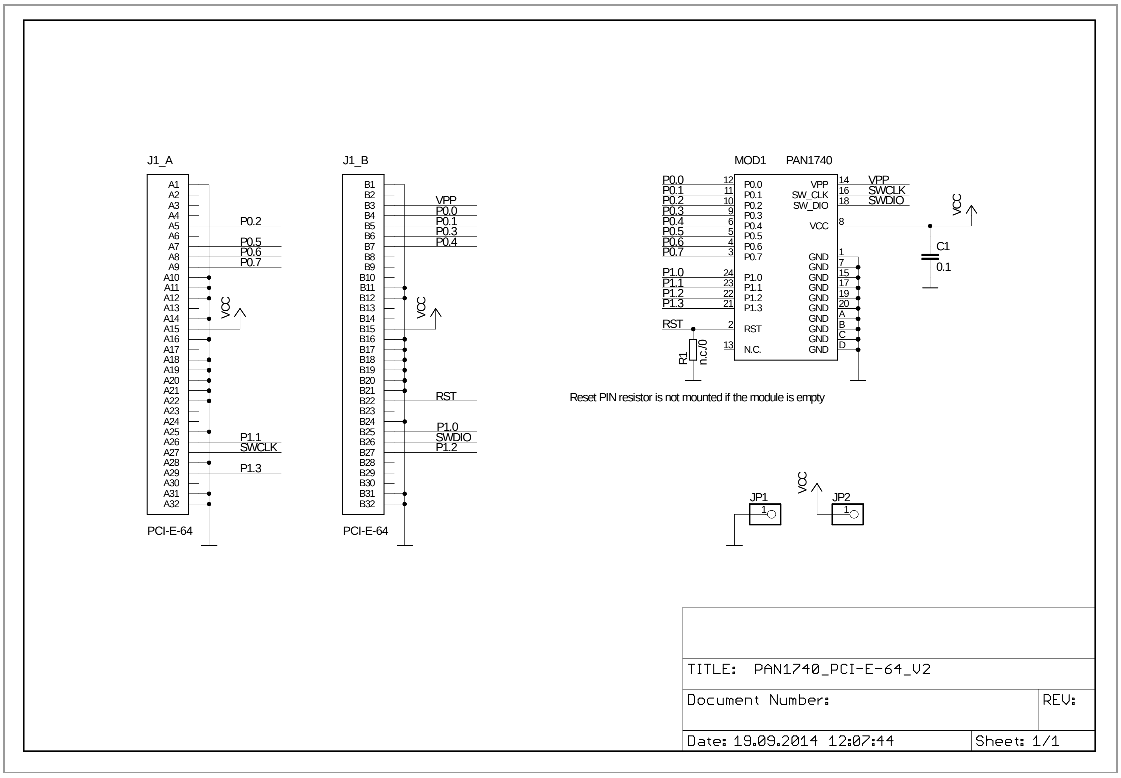
3.2.2 Placement



3.3 Adapter Board



Schematic



Reset PIN resistor is not mounted if the module is empty

TITLE: PAN1740_PCI-E-64_V2
Document Number:
REV:
Date: 19.09.2014 12:07:44
Sheet: 1/1

3.4 Placement Recommendations

Antenna “Keep-Out Area”

Do not place any ground plane under the red marked restricted antenna area in any layer! This would be affecting the performance of the chip antenna in a critical manner.

Impact of Placement on the Antenna Radiation Pattern

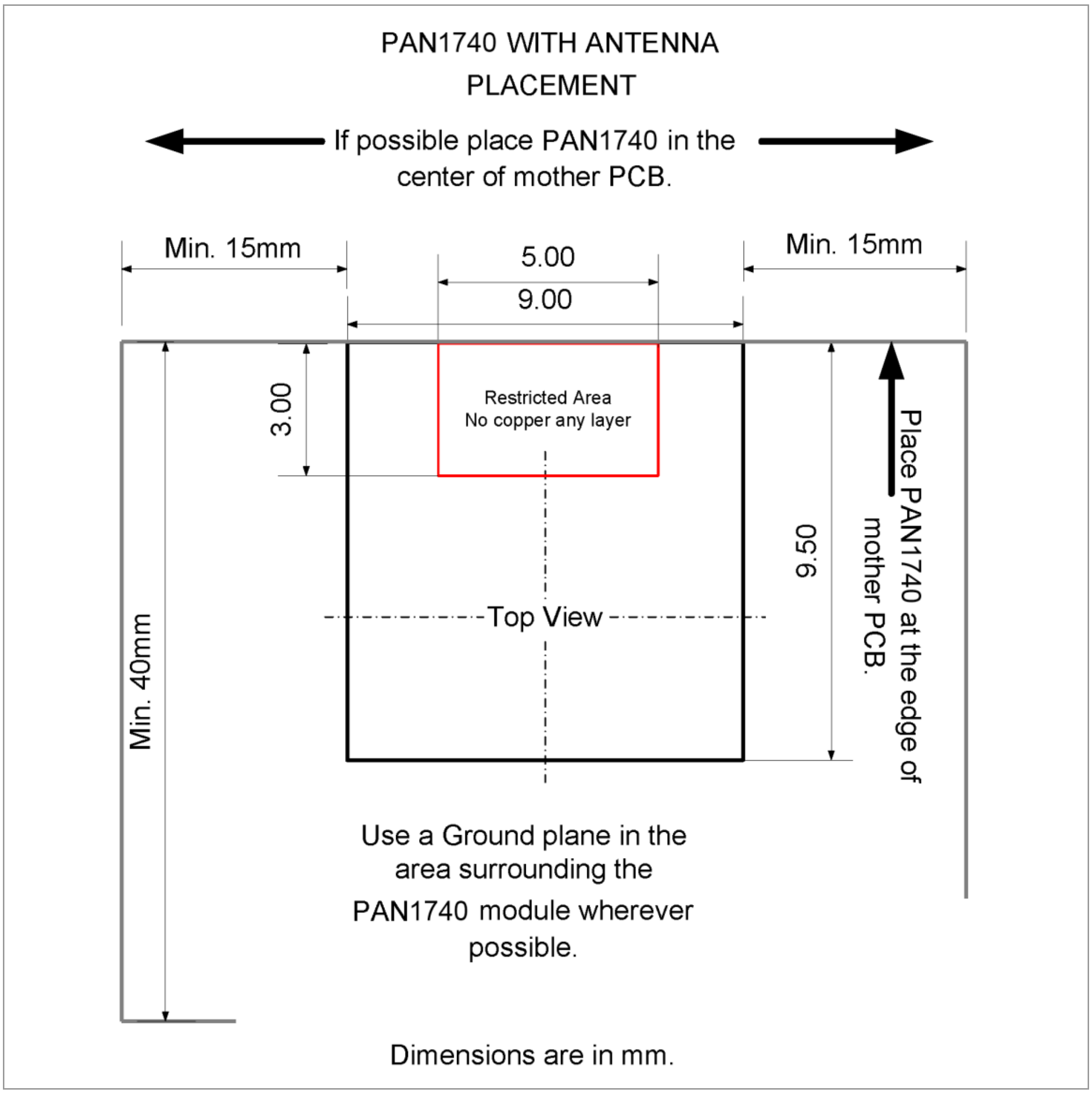
The placement of the module, surrounding material, and customer components has an impact on the radiation pattern of the antenna.

It is recommended to verify the perfect position of the module in the target application before fixing the design.

The following conditions must be met:

- ✓ Keep this product away from heat. Heat is the major cause of decreasing the life of these products.
- ✓ Keep this product away from other high frequency circuits.

To download the design files, please refer to the download area on the product website ⇨ 10.2.2 Product Information.


PAN1740 WITH ANTENNA PLACEMENT
If possible place PAN1740 in the center of mother PCB.
Min. 15mm
5.00
9.00
Min. 15mm
3.00
Restricted Area
No copper any layer
Top View
9.50
Place PAN1740 at the edge of mother PCB.
Min. 40mm
Use a Ground plane in the area surrounding the PAN1740 module wherever possible.
Dimensions are in mm.

4 Run the Keil Project (Example)

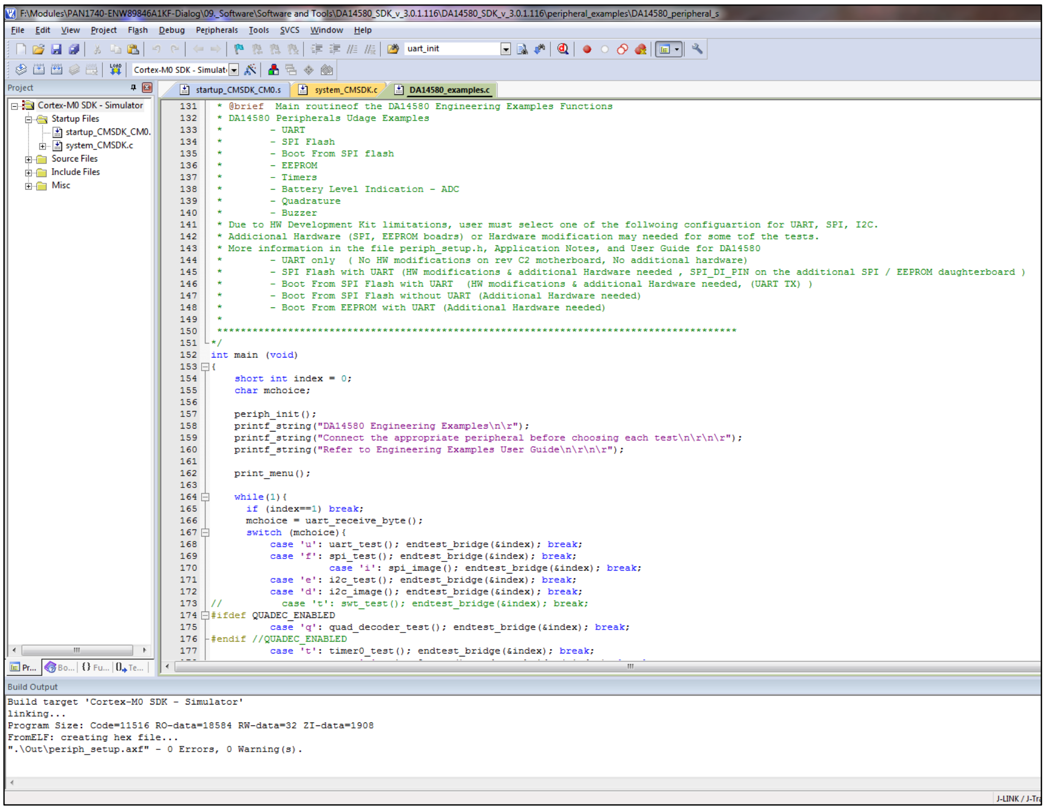
The following requirements must be met:

✓ SDK is installed.

✓ USB-Dongle is installed.

Use always the latest release from Dialog website.

1. Open the Dialog SDK.
2. Open the proximity project example. The project is located in the following SDK folder.



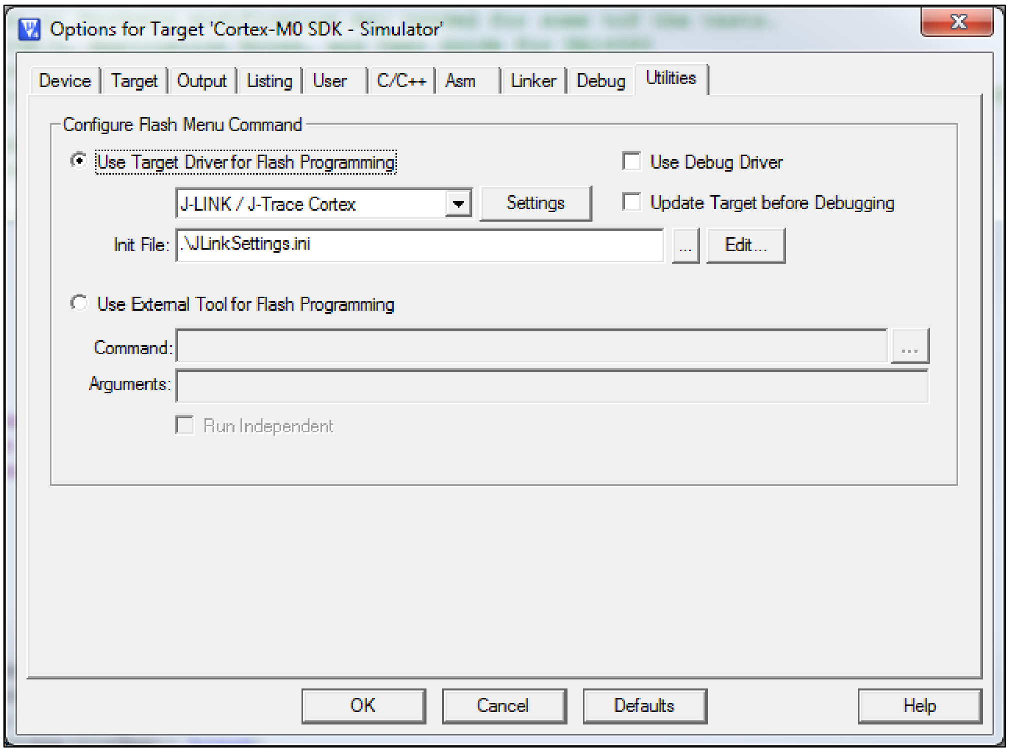
F:\Modules\PAN1740-ENW89846A1KF-Dialog\09._Software\Software and Tools\DA14580_SDK_v_3.0.1.116\DA14580_SDK_v_3.0.1.116\peripheral_examples\DA14580_peripheral_s

```c
 * @brief  Main routineof the DA14580 Engineering Examples Functions
 * DA14580 Peripherals Udage Examples
 *         - UART
 *         - SPI Flash
 *         - Boot From SPI flash
 *         - EEPROM
 *         - Timers
 *         - Battery Level Indication - ADC
 *         - Quadrature
 *         - Buzzer
 * Due to HW Development Kit limitations, user must select one of the follwoing configuartion for UART, SPI, I2C.
 * Addicional Hardware (SPI, EEPROM boadrs) or Hardware modification may needed for some tof the tests.
 * More information in the file periph_setup.h, Application Notes, and User Guide for DA14580
 *         - UART only  ( No HW modifications on rev C2 motherboard, No additional hardware)
 *         - SPI Flash with UART (HW modifications & additional Hardware needed , SPI_DI_PIN on the additional SPI / EEPROM daughterboard )
 *         - Boot From SPI Flash with UART  (HW modifications & additional Hardware needed, (UART TX) )
 *         - Boot From SPI Flash without UART (Additional Hardware needed)
 *         - Boot From EEPROM with UART (Additional Hardware needed)
 *
 ****************************************************************************************
*/
int main (void)
{
    short int index = 0;
    char mchoice;

    periph_init();
    printf_string("DA14580 Engineering Examples\n\r");
    printf_string("Connect the appropriate peripheral before choosing each test\n\r\n\r");
    printf_string("Refer to Engineering Examples User Guide\n\r\n\r");

    print_menu();

    while(1){
      if (index==1) break;
      mchoice = uart_receive_byte();
      switch (mchoice){
          case 'u': uart_test(); endtest_bridge(&index); break;
          case 'f': spi_test(); endtest_bridge(&index); break;
                    case 'i': spi_image(); endtest_bridge(&index); break;
          case 'e': i2c_test(); endtest_bridge(&index); break;
          case 'd': i2c_image(); endtest_bridge(&index); break;
//          case 't': swt_test(); endtest_bridge(&index); break;
#ifdef QUADEC_ENABLED
          case 'q': quad_decoder_test(); endtest_bridge(&index); break;
#endif //QUADEC_ENABLED
          case 't': timer0_test(); endtest_bridge(&index); break;
```

```
Build target 'Cortex-M0 SDK - Simulator'
linking...
Program Size: Code=11516 RO-data=18584 RW-data=32 ZI-data=1908
FromELF: creating hex file...
".\Out\periph_setup.axf" - 0 Errors, 0 Warning(s).
```

3. Open **Keil Compiler**.

4. Configure the Flash Target: Flash > Configure Flash Tools). Choose **Settings**.



Options for Target 'Cortex-M0 SDK - Simulator'

Configure Flash Menu Command

● Use Target Driver for Flash Programming: J-LINK / J-Trace Cortex

Init File: .\JLinkSettings.ini

○ Use External Tool for Flash Programming

Command:

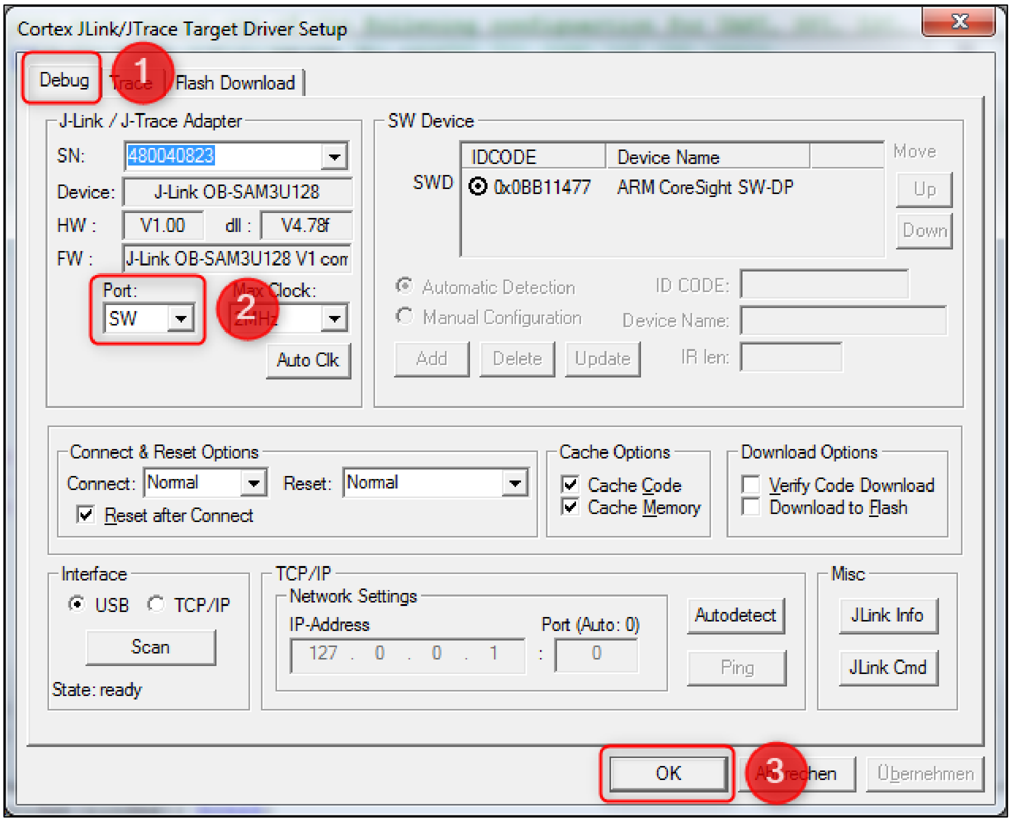
Arguments:

□ Run Independent

□ Use Debug Driver

□ Update Target before Debugging

5. Click on the tab **Debug** (1).



Cortex JLink/JTrace Target Driver Setup

J-Link / J-Trace Adapter

SN: 480040823

Device: J-Link OB-SAM3U128

HW : V1.00 dll : V4.78f

FW : J-Link OB-SAM3U128 V1 con

Port: SW

Max Clock: 2MHz

SW Device

	IDCODE	Device Name
SWD	0x0BB11477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:

○ Manual Configuration Device Name:

IR len:

Connect & Reset Options

Connect: Normal Reset: Normal

☑ Reset after Connect

Cache Options

☑ Cache Code

☑ Cache Memory

Download Options

□ Verify Code Download

□ Download to Flash

Interface

● USB ○ TCP/IP

State: ready

TCP/IP

Network Settings

IP-Address: 127 . 0 . 0 . 1

Port (Auto: 0): 0

6. Setup the Port to **SW** (2).

7. Click **OK** (3).

8. Click on the icon **Build** to build the target files.

9. Click on the icon **Debug** to run the debug session.

   ➔ The proximity project has now been compiled and downloaded into the RAM of the PAN1740/1740A ETU.

For more detailed information on project examples, please refer to Dialog’s download website https://www.dialogs.com/. Recommended is the proximity example as this is the most common profile.

5 Proximity Profile with two PAN1740/1740A USB-Dongles (Example)

This example uses Dialog's SDK version 3.0.2.1.

The following requirement must be met:

✓ Microsoft® Visual C++ 2010 Express – Freeware Compiler is installed.

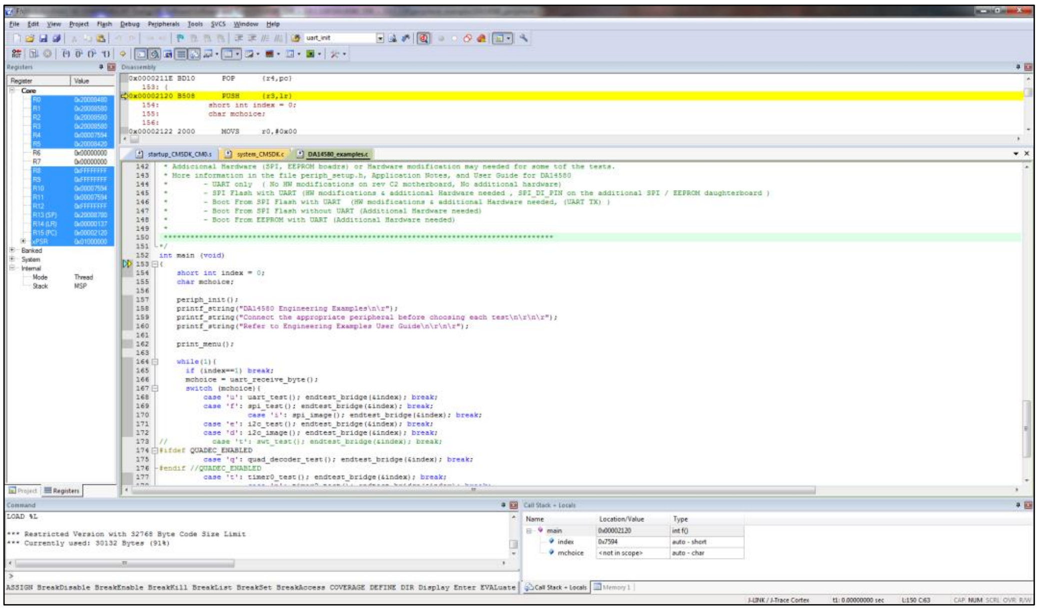
> ⓘ Use always the latest release from Dialog website.

Receiver Configuration

1. Navigate to `DA14580_SDK_3.0.2.1\dk_apps\keil_projects\proximity\monitor_fe_usb`.
2. Download the image `proximity/monitor_fe_usb` into the USB-Dongle.
   For details please refer to ⇨ 4 Run the Keil Project (Example).
3. Open **Keil Compiler** and compile this project.

Download the .hex-file

4. Start and stop the debug mode or load the .hex-file with the **Connection Manager**.
   For debug mode using the **Keil Compiler**, be sure to check the Configure Flash Tool setting described in ⇨ 4 Run the Keil Project (Example).



Register	Value
Core	
R0	0x20008480
R1	0x20008580
R2	0x20008580
R3	0x20008580
R4	0x00007594
R5	0x20008420
R6	0x00000000
R7	0x00000000
R8	0xFFFFFFFF
R9	0xFFFFFFFF
R10	0x00007594
R11	0x00007594
R12	0xFFFFFFFF
R13 (SP)	0x20008780
R14 (LR)	0x00000137
R15 (PC)	0x00002120
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Stack	MSP

```
0x0000211E BD10      POP      {r4,pc}
   153: {
0x00002120 B508      PUSH     {r3,lr}
   154:          short int index = 0;
   155:          char mchoice;
   156:
0x00002122 2000      MOVS     r0,#0x00
```

```
 * Addicional Hardware (SPI, EEPROM boadrs) or Hardware modification may needed for some tof the tests.
 * More information in the file periph_setup.h, Application Notes, and User Guide for DA14580
 *          - UART only  ( No HW modifications on rev C2 motherboard, No additional hardware)
 *          - SPI Flash with UART (HW modifications & additional Hardware needed , SPI_DI_PIN on the additional SPI / EEPROM daughterboard )
 *          - Boot From SPI Flash with UART  (HW modifications & additional Hardware needed, (UART TX) )
 *          - Boot From SPI Flash without UART (Additional Hardware needed)
 *          - Boot From EEPROM with UART (Additional Hardware needed)
 *
 ****************************************************************************************
 */
int main (void)
{
    short int index = 0;
    char mchoice;

    periph_init();
    printf_string("DA14580 Engineering Examples\n\r");
    printf_string("Connect the appropriate peripheral before choosing each test\n\r\n\r");
    printf_string("Refer to Engineering Examples User Guide\n\r\n\r");

    print_menu();

    while(1){
      if (index==1) break;
      mchoice = uart_receive_byte();
      switch (mchoice){
          case 'u': uart_test(); endtest_bridge(&index); break;
          case 'f': spi_test(); endtest_bridge(&index); break;
                    case 'i': spi_image(); endtest_bridge(&index); break;
          case 'e': i2c_test(); endtest_bridge(&index); break;
          case 'd': i2c_image(); endtest_bridge(&index); break;
//          case 't': swt_test(); endtest_bridge(&index); break;
#ifdef QUADEC_ENABLED
          case 'q': quad_decoder_test(); endtest_bridge(&index); break;
#endif //QUADEC_ENABLED
          case 't': timer0_test(); endtest_bridge(&index); break;
```

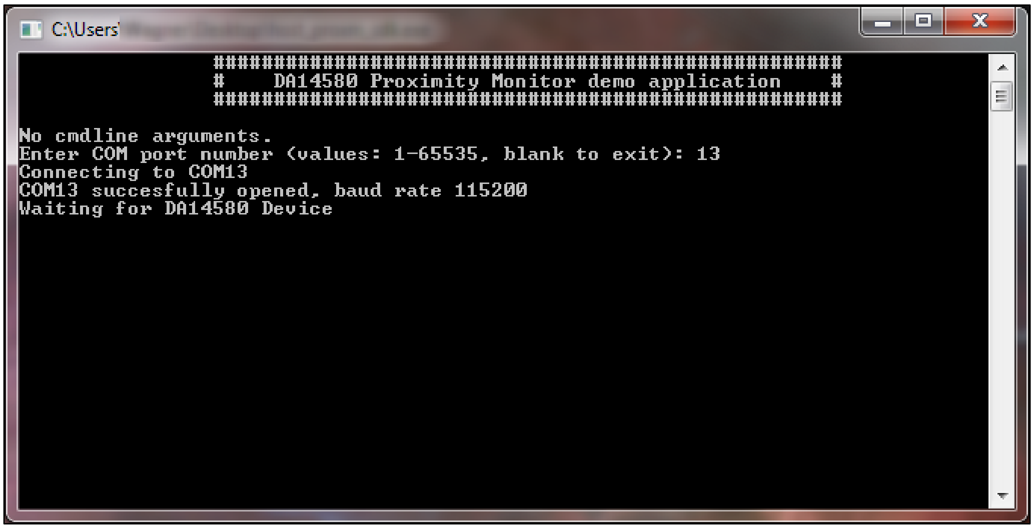
```
LOAD %L

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 30132 Bytes (91%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate
```

Name	Location/Value	Type
main	0x00002120	int f()
index	0x7594	auto - short
mchoice	<not in scope>	auto - char

J-LINK / J-Trace Cortex   t1: 0.00000000 sec   L:150 C:63   CAP NUM SCRL OVR R/W

ⓘ The Debug session must be stopped.

1. Open the folder `Monitor Host Application` in the SDK.
2. Open the project file `host_proxm.sln` with the Microsoft C++ Compiler.
3. Compile and run the software.
4. Determine the correct COM port using Windows® Device Manager and enter this port number in the Proximity Host application (DOS window).



```
                ####################################################
                #     DA14580 Proximity Monitor demo application     #
                ####################################################

No cmdline arguments.
Enter COM port number (values: 1-65535, blank to exit): 13
Connecting to COM13
COM13 succesfully opened, baud rate 115200
Waiting for DA14580 Device
```

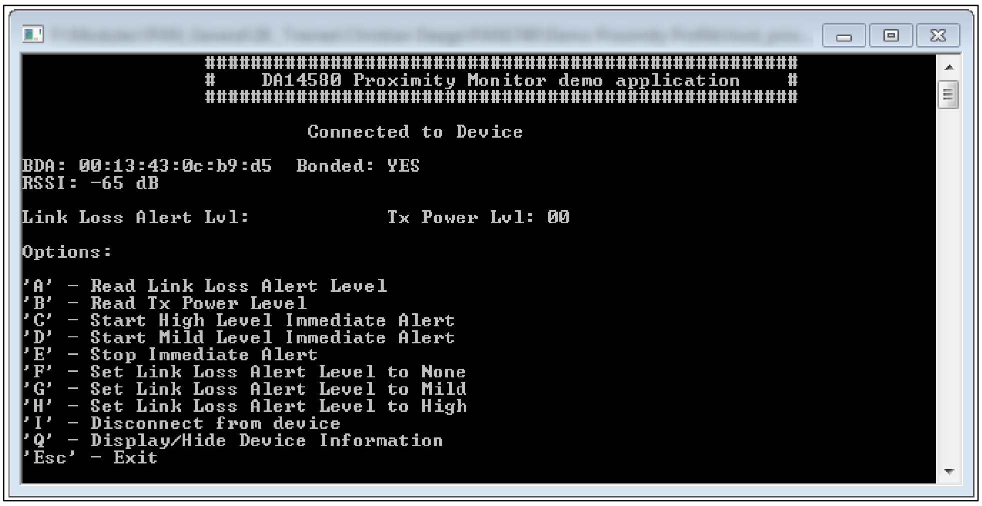
Transmitter Configuration

1. Download the image `proximity/reporter_fe_usb` into the USB-Dongle.
2. Open **Keil Compiler** and compile this project.
3. Start and stop the debug mode or load the .hex-file with the **Connection Manager** to download the .hex-file.
4. Open the reporter host application.
5. Open the project file `host_proxr.sln` with the Microsoft C++ Compiler.
6. Compile and run the software.

7. Determine the correct COM port using Windows Device Manager and enter this port number in the proximity host application (DOS window).

   ➔ The receiver side will show the connection status.

   ➔ The USB-Dongles are connected with the proximity profile.



```
                ####################################################
                #     DA14580 Proximity Monitor demo application     #
                ####################################################

                         Connected to Device

BDA: 00:13:43:0c:b9:d5  Bonded: YES
RSSI: -65 dB

Link Loss Alert Lvl:            Tx Power Lvl: 00

Options:

'A' - Read Link Loss Alert Level
'B' - Read Tx Power Level
'C' - Start High Level Immediate Alert
'D' - Start Mild Level Immediate Alert
'E' - Stop Immediate Alert
'F' - Set Link Loss Alert Level to None
'G' - Set Link Loss Alert Level to Mild
'H' - Set Link Loss Alert Level to High
'I' - Disconnect from device
'Q' - Display/Hide Device Information
'Esc' - Exit
```

For more detailed information on Proximity Profile, please refer to Dialog's download website https://www.dialogs.com/.

6 Smart Snippets

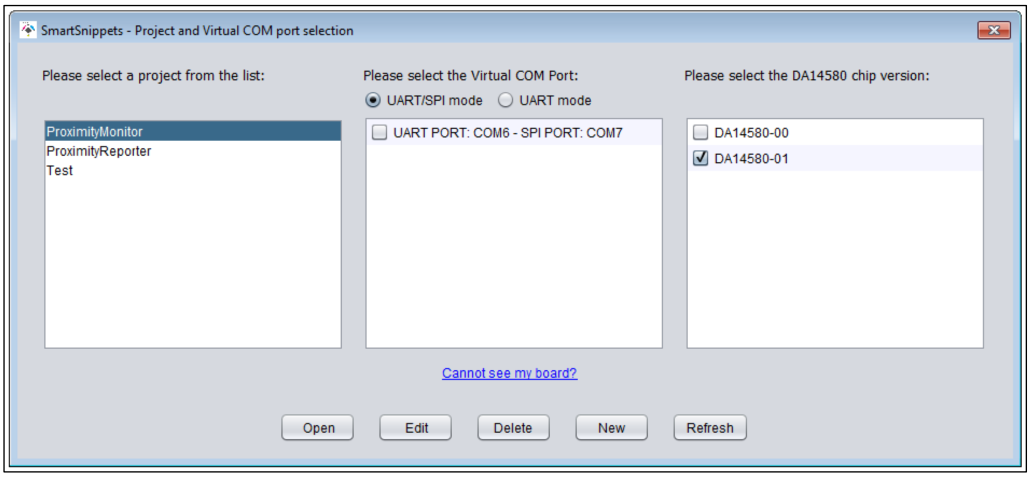
The following description describes the structure and the usage of Smart Snippets in a nutshell. For more information as well as related documents, please refer to the Panasonic website ⇨ 10.2.2 Product Information.

6.1 Program Structure

For details, please refer to the Smart Snippets help (`Help/User Guide/Sleep Mode Advisor`).

6.1.1 Project and Port Selection

1. Open **Smart Snippets**.
2. Select a project, the virtual COM port, and the chip version to be able to control the development kit.



SmartSnippets - Project and Virtual COM port selection

Please select a project from the list:
- ProximityMonitor
- ProximityReporter
- Test

Please select the Virtual COM Port:
● UART/SPI mode ○ UART mode
□ UART PORT: COM6 - SPI PORT: COM7

Please select the DA14580 chip version:
□ DA14580-00
☑ DA14580-01

When the application launches the first time, there will be no projects to select.

Click **New** to create a new project. The name should not contain any spaces or special characters.

3. Select the chip version **DA14580** and a virtual COM port which is assigned to the connected dialog development kit.
4. Click **Open** to establish a connection to the development kit.

   ➔ The software will show its default layout with a toolbar (Board Setup, UART Booter, Power Profiler, Sleep Mode Advisor, OTP Programmer, SPI Flash Programmer, EEPROM Programmer, and SPotA) and a few of these tools in the center of the display.

6.1.2 Board Setup

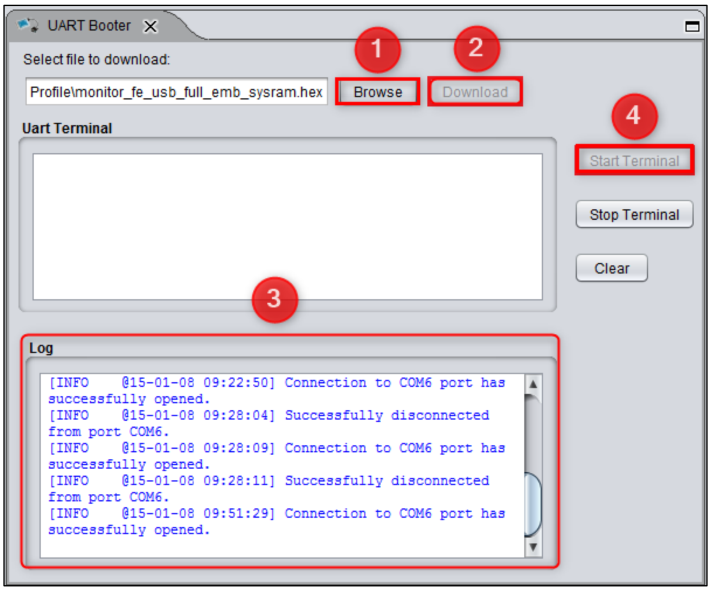
The tab **Board Setup** in the toolbox has to be used before any other tool as it establishes a communication with the development kit during the boot sequence and comes along with two lists.

The upper list contains UART ports with its baud rate, which connect the FTDI chip version “DA14580”. The other list selects the GPIO pin which enables 6.8 V for OTP programming.

6.1.3 UART Booter

The **UART Booter** enables to download application code directly into the RAM of the “DA14580” to test its behavior in terms of power consumption for instance.

1. Click **Browse** (1) to select the desired code (.bin, hex., .ihex).



Select file to download: Profile\monitor_fe_usb_full_emb_sysram.hex

```
[INFO    @15-01-08 09:22:50] Connection to COM6 port has
successfully opened.
[INFO    @15-01-08 09:28:04] Successfully disconnected
from port COM6.
[INFO    @15-01-08 09:28:09] Connection to COM6 port has
successfully opened.
[INFO    @15-01-08 09:28:11] Successfully disconnected
from port COM6.
[INFO    @15-01-08 09:51:29] Connection to COM6 port has
successfully opened.
```

2. Click **Download** (2) and observe the **Log** (3) to handle the requested hardware reset.
3. Optional: Click **Start Terminal** (4) to receive debugging information via UART.

> ⓘ Note that an activated UART connection disables the OTP connection with the result that the UART connection has to be closed to enable the OTP connection and vice versa.

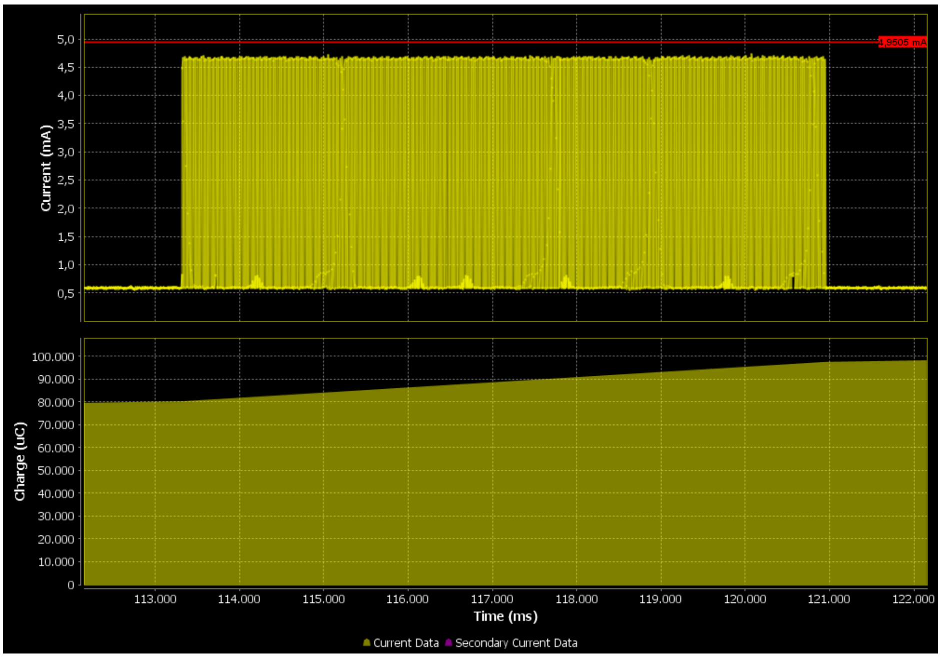
6.1.4 Power Profiler

The tool **Power Profiler** enables the user to measure the power consumption of the desired application with all its functionality.

Start the measurement

1. Click **Initialize**.
   ➔ The program will be initialized and the COM port connection will be opened.
2. Click **Start**.

Measurement example of the Bluetooth scanning process



On the right hand side, the tool provides the measurement information about Peak Current (mA), Average Current (mA), Charge (µC) and the Sleep Mode of the current measurement.

In addition, the tool provides the following control functionalities:

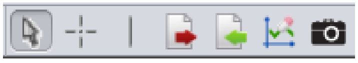
Auto Trigger Mode

In Auto Trigger Mode the measurement process starts as soon as the current (mA) exceeds a user-definable threshold.

Auto Stop Mode

In Auto Stop Mode the measurement process stops automatically when the user-definable time (ms) is elapsed.

Toolbar



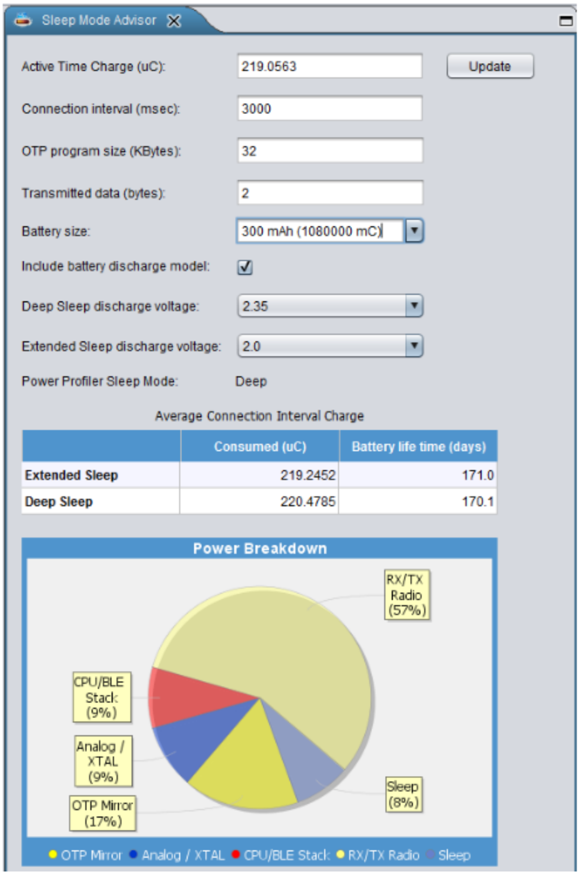
The toolbar can be found in the top of Smart Snippets and enables the user to add:

- Measurements and markers
- Export/import data to/from CSV-files
- Clear secondary current data
- Take snapshots of the Power Profiler chart (.png), which can be found in the path `Dialog/Smart Snippets/Projects/UserProject`.

6.1.5 Sleep Mode Advisor

The tool **Sleep Mode Advisor** uses the gathered consumption data from the Power Profiler and depicts the power consumption in the sleep modes: Deep Sleep and Extended Sleep in a circle diagram. The parameters such as battery size, to obtain reliable calculation results have to be configured.

The following figure from the Smart Snippets help shows an example that prefers the Extended Sleep Mode as the battery lasts 171 days compared to 170 days in deep sleep.



Active Time Charge (uC): 219.0563

Connection interval (msec): 3000

OTP program size (KBytes): 32

Transmitted data (bytes): 2

Battery size: 300 mAh (1080000 mC)

Include battery discharge model: ☑

Deep Sleep discharge voltage: 2.35

Extended Sleep discharge voltage: 2.0

Power Profiler Sleep Mode: Deep

Average Connection Interval Charge

	Consumed (uC)	Battery life time (days)
Extended Sleep	219.2452	171.0
Deep Sleep	220.4785	170.1

6.1.6 OTP Programmer, SPI Flash Programmer, and EEPROM Programmer

To burn the OTP memory and the OTP header on the “DA14580” with a user-definable .hex-, .ihex-, or .bin-file, the tool OTP Programmer is used.

The tab **OTP Image** serves the purpose to read and burn the OTP memory while the tab **OTP header** is used to validate and burn the OTP header. The tab **OTP NVDS** is used to burn the OTP NVDS memory block.

The SPI Flash Programmer enables the user to download an image file to the SPI flash memory of the “DA14580”. The functionality is similar to the OTP Programmer functionality, but the used firmware is different.

The EEPROM Programmer is used for downloading an image file to the “DA14580” EEPROM Memory similar to the OTP Programmer and SPI Flash Programmer functionality.

6.1.7 SPotA (Software Patch over the Air)

The tab **SPotA** can be used to execute software patches from changing a single variable in the code which resides in the SRAM to changing an instruction or data value read from the ROM used for protocol realization.

ⓘ Note that a SPotA is only possible with a SPotA capable counterpart.

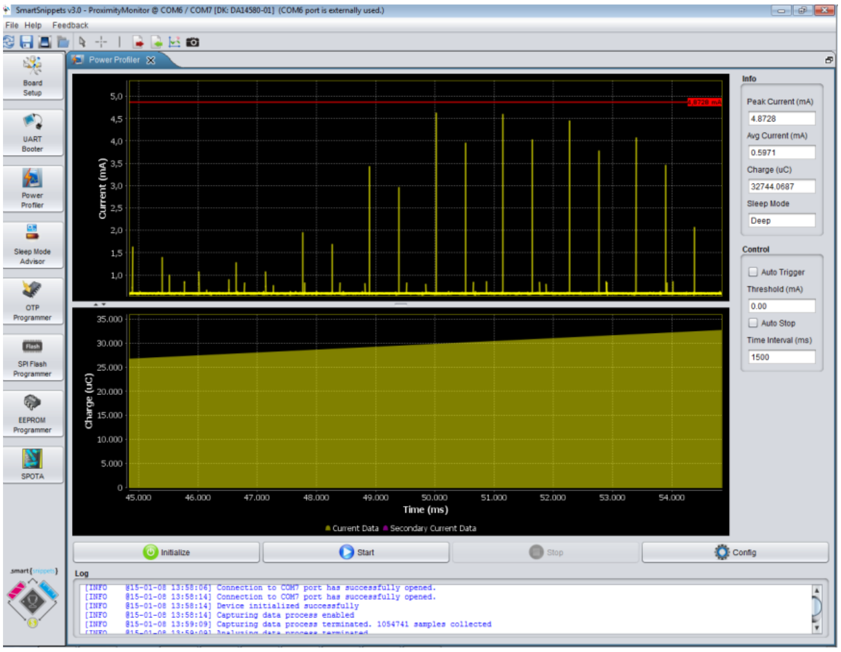
6.2 Application Demo: Proximity Profile

This application example demonstrates the usage of Smart Snippets based on the previously used proximity profile demo. Two evaluation boards “DA14580” with PAN1740/1740A adapter boards are used which are connected to Smart Snippets.

One kit is loaded with the application code for the monitor `monitor_fe_usb_full_emb_sysram.hex` by the UART Booter. The other kit is loaded with the application code for the reporter `reporter_fe_usb_full_emb_sysram.hex`.

After that, the applications **host_proxm_sdk** and **host_proxr_sdk** will be launched and the respective COM ports will be entered and opened.

Once the devices are connected, Smart Snippets enables live evaluation of the application code for instance by measuring the power consumption as depicted below.



SmartSnippets v3.0 - ProximityMonitor @ COM6 / COM7 [DK: DA14580-01] (COM6 port is externally used.)

Info

Peak Current (mA): 4.8728

Avg Current (mA): 0.5971

Charge (uC): 32744.0687

Sleep Mode: Deep

Control

□ Auto Trigger

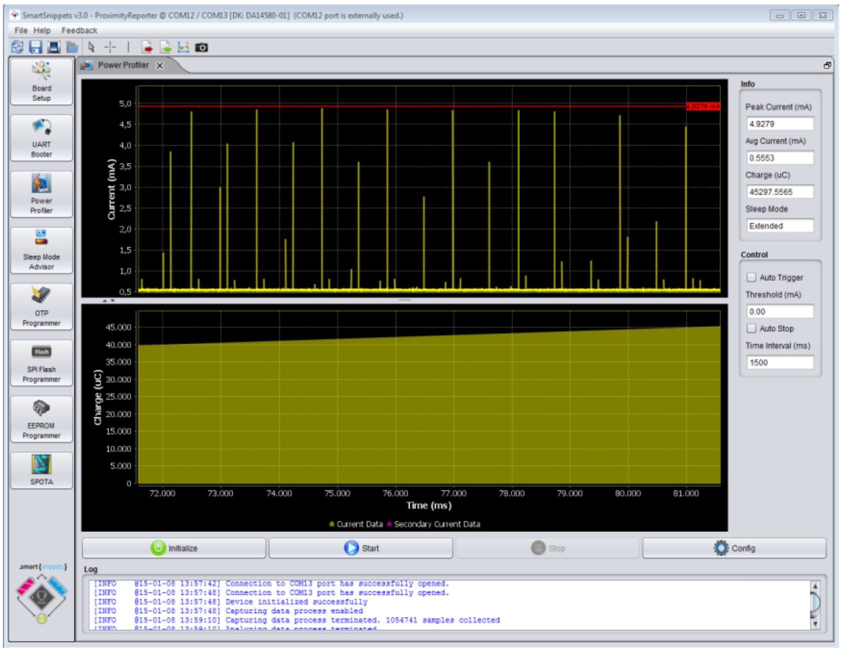
Threshold (mA): 0.00

□ Auto Stop

Time Interval (ms): 1500

Log

```
[INFO    @15-01-08 13:58:06] Connection to COM7 port has successfully opened.
[INFO    @15-01-08 13:58:14] Connection to COM7 port has successfully opened.
[INFO    @15-01-08 13:58:14] Device initialized successfully
[INFO    @15-01-08 13:58:14] Capturing data process enabled
[INFO    @15-01-08 13:59:09] Capturing data process terminated. 1054741 samples collected
[INFO    @15-01-08 13:59:09] Analyzing data process terminated
```



SmartSnippets v3.0 - ProximityReporter @ COM12 / COM13 [DK: DA14580-01] (COM12 port is externally used.)

Info

Peak Current (mA): 4.9279

Avg Current (mA): 0.5553

Charge (uC): 45297.5565

Sleep Mode: Extended

Control

□ Auto Trigger

Threshold (mA): 0.00

□ Auto Stop

Time Interval (ms): 1500

Log

```
[INFO    @15-01-08 13:57:42] Connection to COM13 port has successfully opened.
[INFO    @15-01-08 13:57:48] Connection to COM13 port has successfully opened.
[INFO    @15-01-08 13:57:48] Device initialized successfully
[INFO    @15-01-08 13:57:48] Capturing data process enabled
[INFO    @15-01-08 13:59:10] Capturing data process terminated. 1054741 samples collected
[INFO    @15-01-08 13:59:10] Analyzing data process terminated
```

7 Dialog Serial Port Service (DSPS)

Dialog provides with DSPS its own Bluetooth LE communication profile. Software has been developed for the Development kit and tablets/phones allowing a serial port to be emulated between development kits, handheld devices, and PAN1740/1740A USB Sticks. This software can be downloaded from dialogs support website.

In the following are a few different examples.

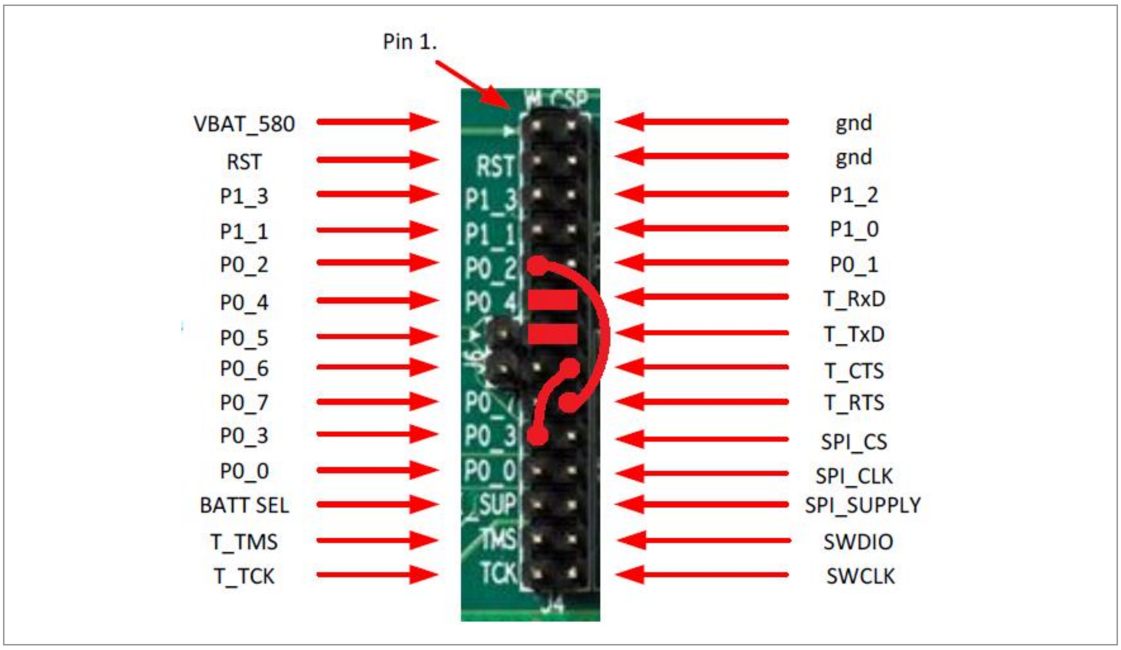
7.1 Pro Kit with PAN1740/1740A Adapter Board (iPod)

The following example shows an easy application of the SPS with Dialog’s Pro Kit with a PAN1740/1740A Adapter Board as device and an iPod running the DSPS application as host to exchange data.

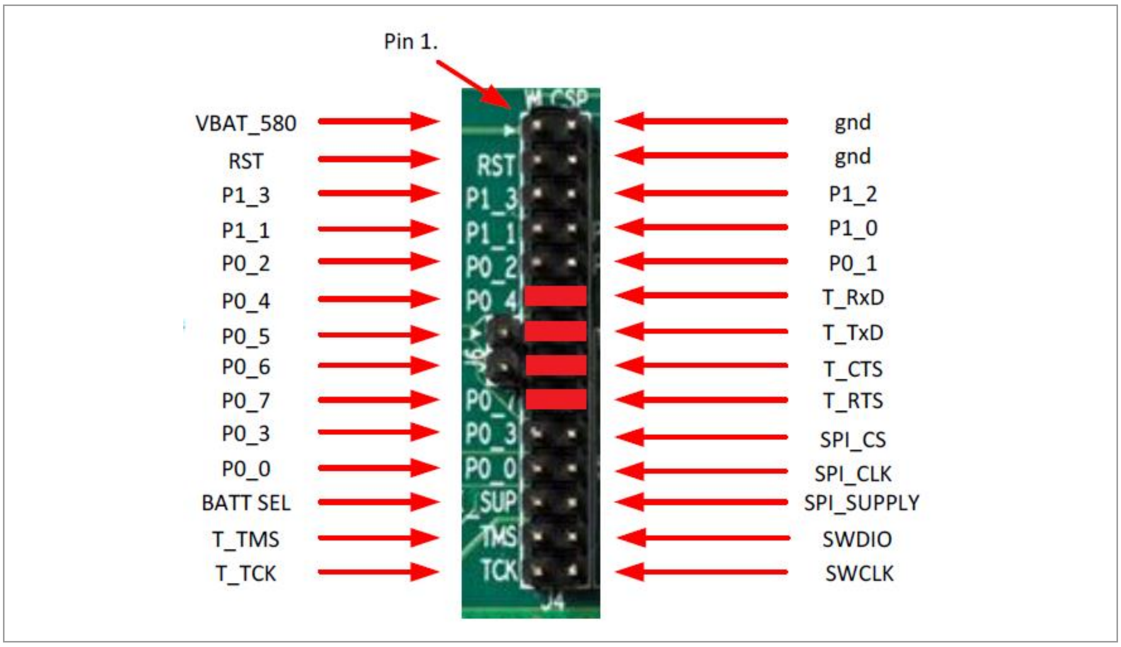
7.1.1 Set up the Pro Kit

The Pro Kit needs to be set up as device using Dialogs SPS application example **sps_device**. The **Connection Manager** or **Smart Snippets** can be used.

The default pin assignment for the DSPS application software with hardware flow control on the Pro Kit:



This pin assignment can be changed to any desired pin assignment in the application source code file `periph_setup.h`. To simplify the connection by using jumpers, the following pin assignment is suitable.



This assignment does only require a simple amendment in the code.

```
#define UART1_RTS_PORT   GPIO_PORT_0
//#define UART1_RTS_PIN    GPIO_PIN_3   // Previous configuration
#define UART1_RTS_PIN    GPIO_PIN_6    // New configuration

#define UART1_CTS_PORT   GPIO_PORT_0
//#define UART1_CTS_PIN    GPIO_PIN_2   // Previous configuration
#define UART1_CTS_PIN    GPIO_PIN_7    // New configuration
```

7.1.2 Start Dialog Serial Port Service (DSPS)

The following requirements must be met:

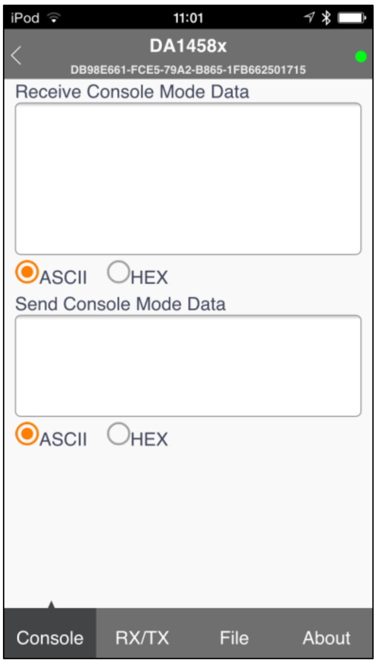
✓ The Pro Kit is set up.

1. Open the iOS application **DSPS**.

   ➔ DSPS starts scanning automatically for discoverable devices supporting the Serial Port Service.

2. Select the desired device to connect it.

➔ The following window appears.



iPod 11:01

DA1458x

DB98E661-FCE5-79A2-B865-1FB662501715

Receive Console Mode Data

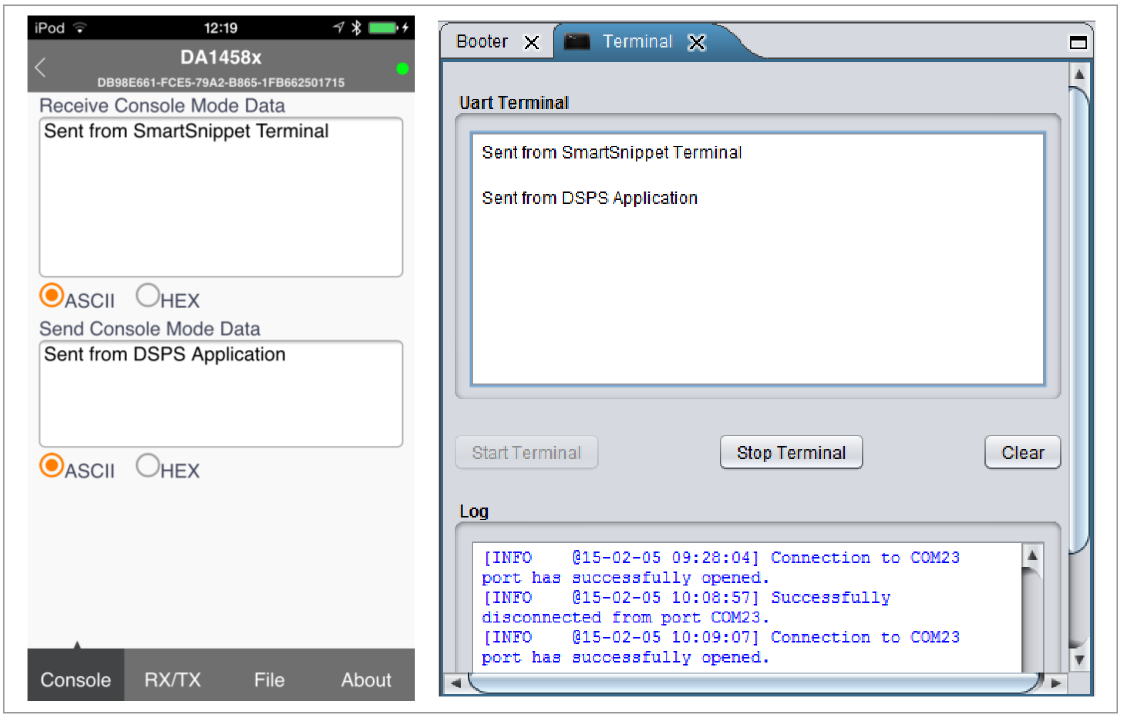
● ASCII ○ HEX

Send Console Mode Data

● ASCII ○ HEX

Tab Description

The tab **Console** enables to send data (ASCII or HEX) immediately after typing the data into the field **Send Console Mode Data** and receives data from the device, e.g. Smart Snippet UART Terminal, as shown below.



iPod 12:19

DA1458x

DB98E661-FCE5-79A2-B865-1FB662501715

Receive Console Mode Data

Sent from SmartSnippet Terminal

● ASCII ○ HEX

Send Console Mode Data

Sent from DSPS Application

● ASCII ○ HEX

Uart Terminal

Sent from SmartSnippet Terminal

Sent from DSPS Application

Log

```
[INFO    @15-02-05 09:28:04] Connection to COM23
port has successfully opened.
[INFO    @15-02-05 10:08:57] Successfully
disconnected from port COM23.
[INFO    @15-02-05 10:09:07] Connection to COM23
port has successfully opened.
```

The tab **RX/TX** allows similar functionality as the tab **Console**, but enables to send data character by character manually or based on a cyclic sending interval which can be determined by the user.

The tab **File** enables to send data files instead of character strings. After the definition of a connection interval, the file directory of the device can be browsed to select the desired file.

7.2 Two Pro Kits with PAN1740/1740A Adapter Board

Similar to the previous example it is possible to use two of Dialog's Pro Kits with PAN1740/1740A Adapter Boards to emulate a serial port.

> Note the hardware flow control changes (jumper or source code) required as explained in ⇨ 7.1 Pro Kit with PAN1740/1740A Adapter Board (iPod).

Set up one Pro Kit with the application software **sps_device** and the other Pro Kit with the application software **sps_host** by downloading the .hex-files onto the Pro Kits.

➔ The host will be discovered and connected to the device.

➔ A Terminal can be launched (for instance the Smart Snippet Terminal) and exchange data.

7.3 Two PAN1740/1740A USB Sticks

Establish a serial port connection with two PAN1740/1740A USB sticks

The `da14580_config.h` of the host and the device source code has to be amended.

1. Change the flow control mode from the default hardware flow control to software flow control.

```
/* Previous configuration */
/*
#define CFG_UART_HW_FLOW_CTRL
#undef CFG_UART_SW_FLOW_CTRL

#ifdef CFG_UART_HW_FLOW_CTRL
    #undef CFG_UART_SW_FLOW_CTRL
#endif
*/

/* New configuration */
#define CFG_UART_SW_FLOW_CTRL
#undef CFG_UART_HW_FLOW_CTRL
```

2. Disable the sleep mode, as this is not fully supported for software flow control.

```
/*Sleep modes*/
/* Previous configuration
//#define CFG_EXT_SLEEP
*/

// New configuration
#undef CFG_EXT_SLEEP
```

3. Compile the amended source codes.
4. Download the output .hex-files onto the sticks by using Dialogs **Connection Manager** or **Keil µVision**®.

> ⓘ Note that it is not possible to run the sticks with Dialogs Smart Snippets.

5. Launch two terminals (e.g. Tera Term) with the corresponding COM port and the following settings:

– Baud Rate: 115 200 bps

– Data Bits: 8

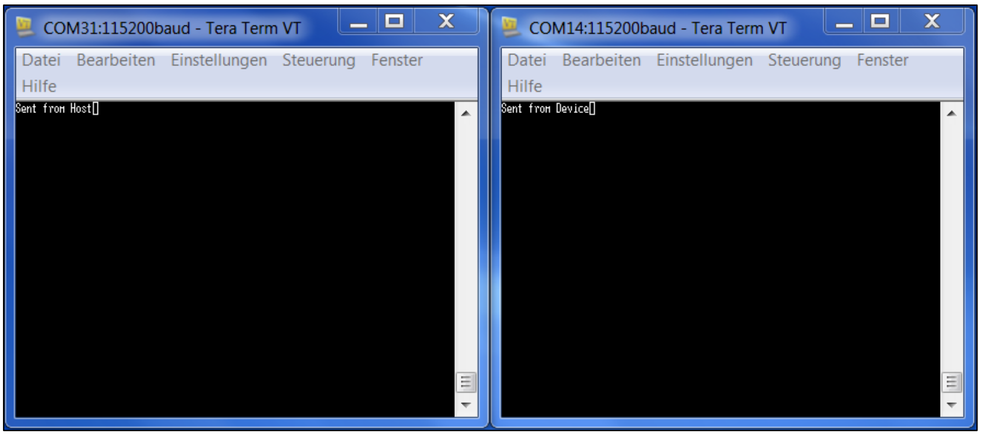
– Stop Bits: 1

– Parity: None

– Flow Control: Xon/Xoff

6. Start data exchange.

➔ The data transmission happens immediately. One terminal window will stay empty for unidirectional data transfer as depicted below.


COM31:115200baud - Tera Term VT
Datei Bearbeiten Einstellungen Steuerung Fenster Hilfe
COM14:115200baud - Tera Term VT
Datei Bearbeiten Einstellungen Steuerung Fenster Hilfe

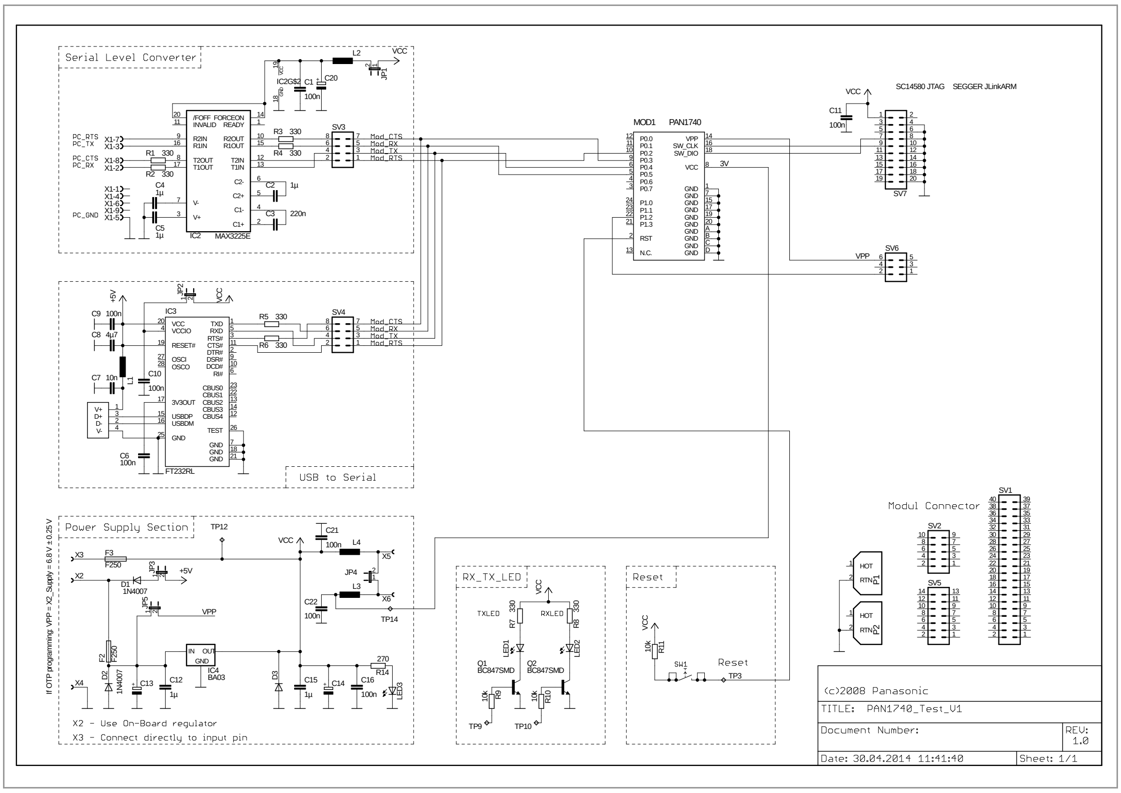
```
Sent from Host
```

```
Sent from Device
```

8 Production Tools

To program the PAN1740/1740A in production a J-Link programmer and the 6.8 V programming voltage on the input pin VPP is required.

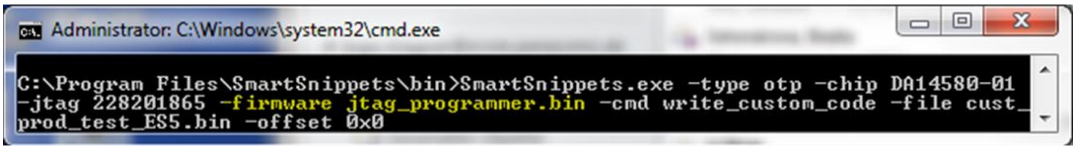
Example schematic for a programming jig



For more details on programming the OTP, please refer to the Smart Snippets website and Dialog's website.

The crystal frequency register and flag as well as the Bluetooth MAC address are already burned.

Example of the programming into OTP:



```
C:\Program Files\SmartSnippets\bin>SmartSnippets.exe -type otp -chip DA14580-01
-jtag 228201865 -firmware jtag_programmer.bin -cmd write_custom_code -file cust_
prod_test_ES5.bin -offset 0x0
```

9 Life Support Policy

This Panasonic Industrial Devices Europe GmbH product is not designed for use in life support appliances, devices, or systems where malfunction can reasonably be expected to result in a significant personal injury to the user, or as a critical component in any life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Panasonic customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Panasonic Industrial Devices Europe GmbH for any damages resulting.

10 Appendix

10.1 Ordering Information

10.1.1 PAN1740

Variants and Versions

Order Number	Brand Name	Description	MOQ[1]
ENW89846A1KF[2]	PAN1740	PAN1740 Bluetooth Low Energy Module	1 500

Evaluation Tools[3]

Order Number	Tool Name	Content	MOQ[3]
ENW89846AYKF	PAN1740 Kit	2x USB stick	1
ENW89846AZKF	PAN1740 ETU	1x USB stick	1
ENW89849A1KF	PAN1740 Beacon w/o connector	1x Beacon (without connector)	T&R: 150 loose: 1
ENW89849AZKF	PAN1740 Beacon w connector	1x Beacon (with connector)	T&R: 150 loose: 1
ENW89849AYKF	PAN1740 Experimenter Kit	5x Beacon (with connector), 1x Dialog Pro Kit, 1x Adapter board for flat cable, 1x flat cable	1
ENW89846AWKF	PAN1740 Daughterboard	1x PAN1740 Daughterboard	1
ENW89846AVKF	PAN1740 Dialog Pro Kit	1x Dialog Pro Kit (Motherboard), 3x PAN1740 Daughterboard, 1x USB stick, 1x USB cable, Battery	1

10.1.2 PAN1740A

Variants and Versions

Order Number	Brand Name	Description	MOQ
ENW89852A1KF[4]	PAN1740A	PAN1740A Bluetooth Low Energy Module	1 500

[1] Abbreviation for Minimum Order Quantity (MOQ). The default MOQ for mass production is 1 500 pieces, fewer only on customer demand. Samples for evaluation can be delivered at any quantity via the distribution channels.

[2] Samples are available on customer demand.

[3] For further information please visit our website ⇨ 10.2.2 Product Information.

[4] Samples are available on customer demand.

Evaluation Tools[5]

Order Number	Tool Name	Content	MOQ[3]
ENW89852AWKF	PAN1740A Kit	2x USB-Stick	1
ENW89852AXKF	PAN1740A ETU	USB-Stick	1
ENW89852AUKF	PAN1740A Daughter Board	PAN1740A Daughter board	1
ENW89852AVKF	PAN1740A Dialog Pro Kit	• Dialog Pro Kit (Motherboard) • 3x PAN1740A Daughter Board • USB-Stick • USB-Cable • Battery	1

[5] Please refer to the Panasonic website for further information ⇨ 10.2.2 Product Information.

10.2 Contact Details

10.2.1 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

10.2.2 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