SEARAN

dotstack™ Software Development Kit

for NXP LPC17xx microcontrollers

Getting Started

Version 1.8.8

SEARAN LLC

Table of Contents

1 Overview

The dotstack™ Software Development Kit is designed for developing Bluetooth solutions using LPC17xx microcontrollers from STMicroelectronics. The kit consists of the following components:

- Header files and object code of the core dotstack™ library.
- Sample applications demonstrating how to integrate and use dotstack™ APIs.
- API documentation.

The purpose of this Getting Started document is to help start using the SDK quickly.

2 System Requirements

Supported IDE and compiler:

- IAR Embedded Workbench for ARM version 6.40 or later

Supported development board for sample applications:

- Blueboard LPC1768-H board

Supported Bluetooth evaluation boards:

- Panasonic PAN1315ETU
- Panasonic PAN1323ETU
- CSR 8811 - DK-8811-10055-1A

The easiest way of connecting the Panasonic Bluetooth evaluation boards to the LPC17xx development board is to use the battery board from TI (http://www.ti.com/tool/soc-bb).

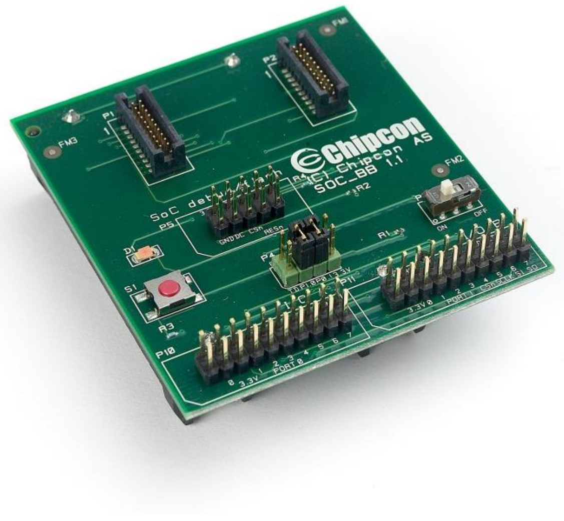


Figure 1. Battery Board.

It has sockets matching those on the Panasonic ETU boards and outputs all relevant signals to a set of standard 0.1" headers.

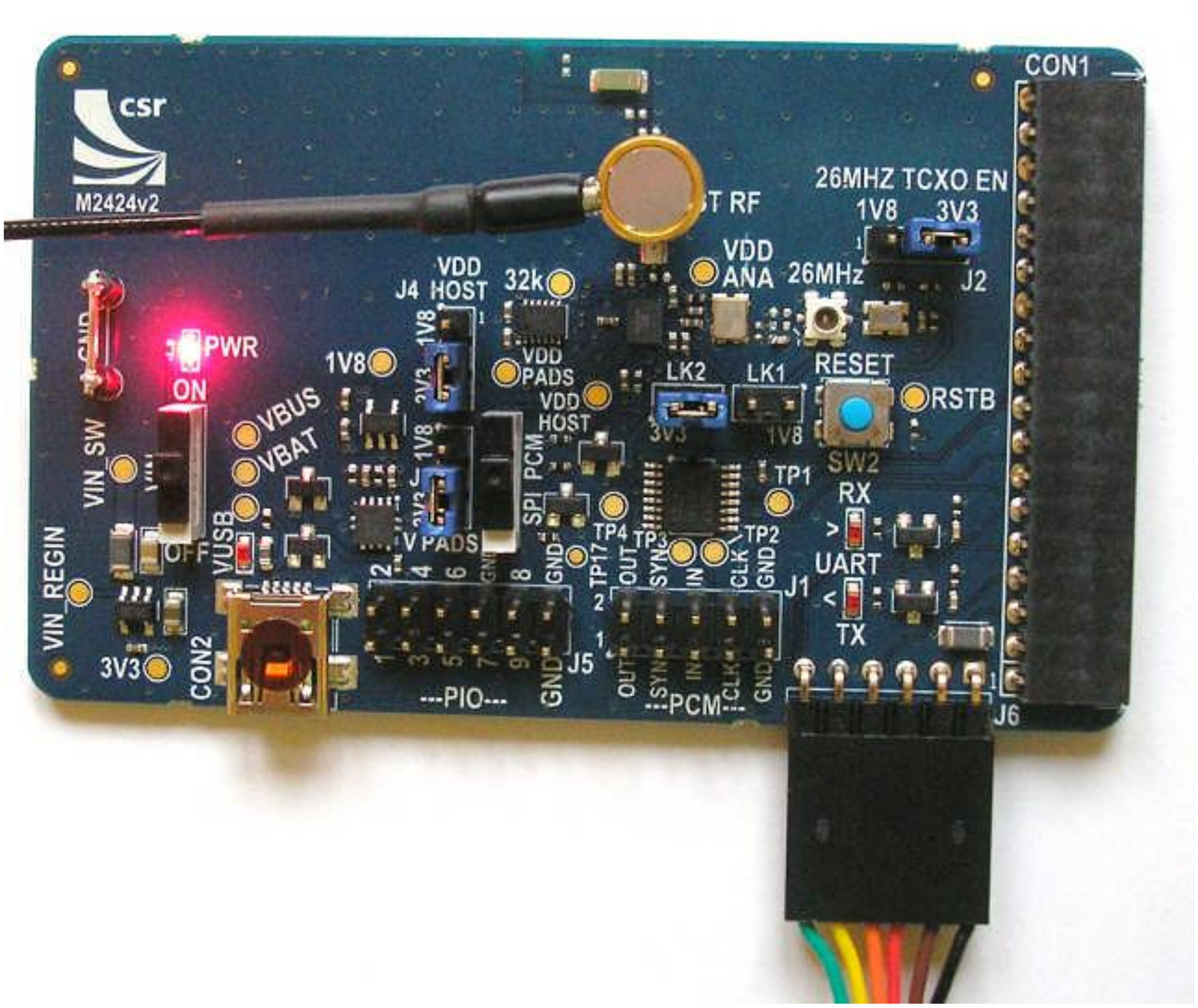


Figure2. CSR 8811 – DK-8811-10055-1A board.

3 Software Setup

The SDK is distributed as a ZIP archive. To install, just extract it to an arbitrary location in your system. The root directory of the SDK installation is denoted in this document as <SDK>. The following table describes the directory structure of the SDK.

Directory	Description

<SDK>/dotstack	dotstack™ libraries and header files.
<SDK>/samples	Sample applications.
<SDK>/os/freertos	FreeRTOS source code.
<SDK>/docs	Documentation.
<SDK>/tools/sdp_compiler	SDP database compiler.

Table 1. SDK directory structure.

The dotstack™ library and header files for the Cortex-M3 architecture built with the IAR toolchain are located in *<SDK>/dotstack/cortex-m3-iar*.

The dotstack™ library is supplied in four variants:

Variant	**Description**
libdotstack_rel.a	Release build, logging disabled.
libdotstack_rel_log.a	Release build, logging enabled
libdotstack_dbg.a	Debug build, logging disabled.
libdotstack_dbg_log.a	Debug build, logging enabled

Table 2. dotstack™ library variants.

The libraries were built with the following compilation options:

Common options:	--cpu=Cortex-M3---silent --fpu=None --endian=little --dlib_config DLib_Config_Normal.h
Release options:	-Ohz -DNDEBUG
Debug options:	--debug -Ol --no_cse --no_unroll --no_inline --no_code_motion --no_tbaa --no_clustering --no_scheduling

4 Hardware Setup

The SDK sample programs are designed to run on the Blueboard LPC1768-H evaluation board.

4.1 Connecting Panasonic ETU boards

Use the following table to connect a PAN13xxETU Bluetooth evaluation board to the Blueboard LPC1768-H evaluation board:

LPC1768-H signal	LPC1768-H pin	Battery Board pin	PAN13xxETU signal	PAN13xxETU pin
LPC_UART1 TX	SCK0 (J6)	P10.9	BT_HCI_RX	J1.7
LPC_UART1 RX	SSEL0 (J6)	P10.11	BT_HCI_TX	J1.9
LPC_UART1 CTS	MISO0 (J6)	P10.15	BT_HCI_RTS	J2.18
LPC_UART1 RTS	P0.22 (J13)	P10.13	BT_HCI_CTS	J1.3
MAT2.0	SSEL1 (J6)	P10.7	SLOW_CLK	J1.5
P0.21	P0.21 (J13)	P11.4	BT_NSHUTD	J2.19
3V3	3V3	P11.3	3V3	J2.7
GND	GND	P11.20	GND	J1.1, J1.19

Table 3. Blueboard LPC1768-H to PAN13xxETU connections.

If you use the battery board, remove all jumpers on it. For easy identification of battery board pins, use the following diagram.

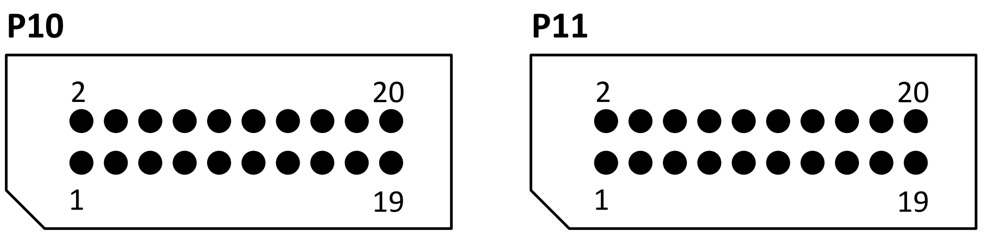


Figure 3. Battery board pin-out.

4.2 Connecting CSR 8811 - DK-8811-10055-1A

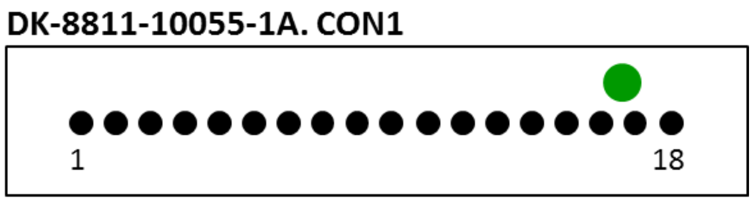
Use the following table to connect CSR 8810 - DK-8811-10055-1A board to the Blueboard LPC1768-H evaluation board:

LPC1768-H signal	LPC1768-H pin	CSR8811 female connector (CON1) pin	CSR8811 signal
LPC_UART1 TX	SCK0 (J6)	4	UART_RX
LPC_UART1 RX	SSEL0 (J6)	6	UART_TX
LPC_UART1 CTS	MISO0 (J6)	3	UART_RTS
LPC_UART1 RTS	P0.22 (J13)	5	UART_CTS
P0.21	P0.21 (J13)	15	RSTB
3V3	3V3	1,2,14	VDD
GND	GND	7,12,16,18	GND

Table 4. Blueboard LPC1768-H to CSR 8811 - DK-8811-10055-1A connections.

Also two contacts on the CSR 8810 board have to be interconnected with a wire – RSTB (next to the RESET button) and PIN 15 on the CON1.

For easy identification of CSR 8811 - DK-8811-10055-1A pins, use the following diagram.



5 Software Configuration

All sample applications use a common configuration file:

<SDK>/samples/ lpc17xxl/common/config.h

5.1 BT_CONTROLLER option

The BT_CONTROLLER configuration option selects the type of Bluetooth controller. It must be set to match the controller on the Bluetooth evaluation board. The following table shows the value of the setting for supported evaluation boards:

Evaluation board	BT_CONTROLLER value
PAN1315ETU	BT_CONTROLLER_CC2560
PAN1323ETU	BT_CONTROLLER_CC2564
CSR 8811 – DK-8811-10055-1A	BT_CONTROLLER_CSR8811A08

Table 5. BT_CONTROLLER setting values.

The default value for this option is BT_CONTROLLER_CC2564.

5.2 BT_UART_INITIAL_BAUD_RATE and BT_UART_WORKING_BAUD_RATE options

The BT_UART_INITIAL_BAUD_RATE option specifies the baud rate to use to communicate with the Bluetooth controller after it has been reset.

CSR 8810/11 controllers use automatic host transport and baud rate selection scheme. After reset the sample applications configure the controller to use H4 transport at 115200 (default value for the BT_UART_INITIAL_BAUD_RATE option) bauds. The value of the BT_UART_INITIAL_BAUD_RATE should be kept low to make the auto configuration process as reliable as possible. Once the transport has been selected the controller is configured to work at the speed specified by the BT_UART_WORKING_BAUD_RATE option by using BCCMD protocol

For TI CC256x controllers, it is always 115200 and must not be changed. Applications use this baud rate to send the HCI vendor specific command that configures the controller for the baud rate specified by the BT_UART_WORKING_BAUD_RATE option.

When changing the value of the BT_UART_WORKING_BAUD_RATE option, make sure that the system clock is fast enough for the USART peripheral to work at that baud rate.

6 Loopback Test

To verify hardware setup, use the loopback test programs located in the *<SDK>/samples/stm32l/tools/loopback_test* directory. The program performs basic initialization of the hardware, configures the Bluetooth controller for loopback mode, and then sends and receives test data in a loop. If data received from the Bluetooth controller matches the data sent to it, the blue LED flashes periodically. In case of a serious communication problem that prevents sending or receiving data completely and thus prevents Bluetooth controller initialization, both green and blue LEDs stay on. In case of other error (during initialization or data exchange) both LEDs start flashing rapidly.

7 Sample Applications

The following table contains a list of supplied sample applications.

Table 6. Sample applications.

Name	Subdirectory	Device Name	Used Profiles
Serial Port	Spp	dotstack SPP Demo	SPP
Heart Rate Sensor	heartrate-sensor	dotstack Heart Rate Sensor Demo	GATTP

The Device Name column in the above table shows the name that is returned by the device in its inquiry scan response. It is how the device is seen when it is discovered by other devices.

All sample applications supplied with the SDK are using FreeRTOS real time OS. Its source code is located in *<SDK>/os/freertos.*

Project and workspace files for the IAR toolchain are located in the *<SDK>/samples/stm32l/freertos/iar* directory. Each subdirectory in this directory contains a project file and a corresponding workspace file. Also, there is a workspace file that includes all sample projects. This file is *<SDK>/samples/ stm32l /freertos/iar/samples-iar.eww.*

All sample applications (except HID Keyboard) are using the same pairing code: 0000. Some of the applications are configured to use Secure Simple Pairing. The default SSP configuration uses the "Just Works" association model. In this mode, the user does not have to enter any pairing code or confirmation.

7.1 Serial Port

This application implements a simple echo service that sends back any data it receives. In addition to that:

- When the User button is pressed, the application replies with "Button1 pressed".
- When the '?' character is detected in the input data, the application replies with a universal answer.

The pairing code for this application is 0000.

To test this functionality on Windows, you will need a terminal program that can communicate using a COM port. Windows XP comes with the Hyper Terminal program. Unfortunately, later Windows versions do not include this software. However, there are multiple alternatives. The easiest ones are:

- Putty. This is an open source software that can be used to communicate over a COM port (in addition to its other features). It can be downloaded from http://www.chiark.greenend.org.uk/~sgtatham/putty/
- If you own a copy of Windows XP, you can reuse its Hyper Terminal software. Just copy the following files from your Windows XP to an arbitrary location on your newer Windows system:

Windows XP file
C:\Program Files\Windows NT\hypertrm.exe
C:\Program Files\Windows NT\hticons.dll
C:\Program Files\Windows NT\htrn_jis.dll
C:\Windows\System32\hypertrm.dll

7.2 Heart Rate Sensor

This sample application simulates a heart rate sensor. It uses Bluetooth Low Energy and the GATT profile.

Initially, the reported heart rate is set to 65 bits per minute. Using the up and down joystick actions the heart rate can be increased or decreased.

8 Evaluation Version

The evaluation version of the SDK contains all the documentation and sample code of the full version. However, the HCI layer in the evaluation version imposes a limit on the number of HCI packets that can be transferred. The limit is set to 1 million packets. This is a very high number of packets that will allow applications to run from several days to several weeks. When the limit is reached, the HCI layer will stop functioning. It will appear as if the application has frozen. In this case, the application needs to be reset.