Introduction to BLE System Design

BLE = Bluetooth® Low Energy

Workshop Objectives

By the end of this workshop, you will

Understand BLE one-chip solutions

Understand BLE architecture

Learn how to use BLE solutions and development environment to implement:

- BLE connections with PSoC 4 BLE and PRoC BLE
- One-chip, sensor-based system designs with BLE connectivity for the Internet of Things (IoT[2])
- One-chip, CapSense® touch-sensing user interface designs with BLE connectivity

[1] PSoC = Programmable System-on-Chip, PRoC = Programmable Radio-on-Chip
[2] An expansion of the Internet to include everyday physical objects such as thermostats

Set Up and Install Software

Required software and initial steps

Copy the contents of the provided USB drive onto your laptop and install the software in the table below

Follow the on-screen instructions to complete the installation in approximately 15 minutes

Software	Version	File Name
PSoC Creator[1] Installer	3.2 (or newer)	*"PSoCCreatorSetup_3.2.exe"*
CySmart[2] Installer	1.0 SP1 (or newer)	*"CySmartSetup_1.0_sp1.exe"*
BLE Pioneer Kit Installer	Revision *C (or newer)	*"CY8CKIT042BLEKITSetupOnlyPackage_RevSC.exe"*
BLE Lab Exercise Files	2.0	"*BLEWorkshop_2.0.zip*"

Required hardware:

BLE Pioneer Kit (CY8CKIT-042-BLE), shown at right

[1] PSoC 3, PSoC 4 and PSoC 5 Integrated Design Environment (IDE) software that installs on your PC

[2] A GUI-based software tool that installs on your PC to test and debug BLE functionality

Bluetooth Terms

Bluetooth Classic

A legacy standard for personal area networks made popular by audio streaming to cell phone headsets

Operates in the 2.4-GHz ISM[1] Band with GFSK[2] modulation and supports up to a 3-Mbps data rate

Bluetooth Low Energy (BLE)

A standard for short-range, low-power wireless applications that communicates state or control information

Operates in the 2.4-GHz ISM[1] Band with GFSK[2] modulation and supports a 1-Mbps data rate

Not backward-compatible with Bluetooth Classic

Bluetooth 4.0

An upgraded Bluetooth Classic specification that adds BLE

Bluetooth 4.1 (Bluetooth Spec)

An enhanced Bluetooth 4.0 specification, adopted in Dec. 2013

Includes improved security, lower power and higher throughput[3]

Bluetooth Smart

A brand for Bluetooth 4.0/4.1 products that support only BLE

Bluetooth Smart Ready

A brand for Bluetooth 4.0/4.1 products that support both Bluetooth Classic and BLE

Bluetooth Special Interest Group (SIG)

The organization that oversees the development and licensing of Bluetooth standards

[1] An Industrial, Scientific, Medical (ISM) RF frequency band that is license-free worldwide
[2] Gaussian frequency shift keying
[3] 272 Kbps (15% higher vs. 4.0) including protocol overhead

Additional Terms

BLE Protocol Stack (BLE Stack)

Firmware that implements the Bluetooth 4.0/4.1 specification to provide BLE communication

BLE Profile (Profile)

A Bluetooth specification that describes a set of operations and behaviors that devices use to communicate with one another

Ensures interoperability when two or more devices use a common Profile

For example, keyboards use the HID Profile and Heart Rate Monitors (HRMs) use the HRM Profile

Analog Front End (AFE)

An analog signal-conditioning circuit that uses opamps, filters and comparators to interface to an analog-to-digital converter (ADC)

Internet of Things (IoT)

An expansion of the Internet to include everyday physical objects such as thermostats

PSoC Terms

PSoC

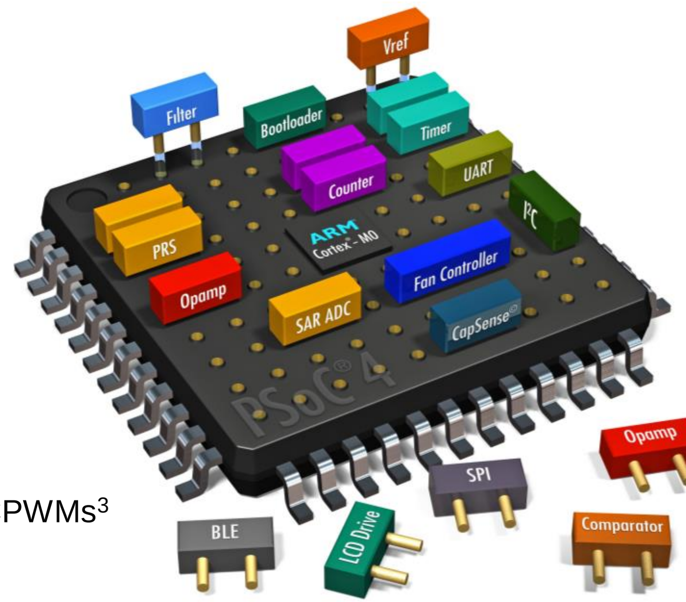
PSoC is the world's only programmable embedded **system-on-chip** integrating an MCU core, **Programmable Analog Blocks**, **Programmable Digital Blocks**, **Programmable Interconnect and Routing**[1] and **CapSense**

Programmable Analog Block

A hardware block that is configured using **PSoC Components**[2] to create Analog Front Ends (AFEs), among other capabilities

Includes **Continuous Time Blocks**, analog-to-digital converters (ADCs) and digital-to-analog converters (DACs)

Illustration of a PSoC Device Being Flexibly Configured by Plugging in PSoC Components



Continuous Time Block (CTB)

A **Programmable Analog Block** that is used to implement continuous time analog circuits such as opamps and programmable gain amplifiers (PGAs)

Programmable Digital Block

A hardware block that is configured using **PSoC Components**[2] to implement custom digital peripherals and glue logic

Includes **Universal Digital Blocks,** Serial Communication Blocks (SCBs) and TCPWMs[3]

Universal Digital Block (UDB)

A PSoC **Programmable Digital Block** that contains: two programmable logic devices (PLDs), one programmable data path with an arithmetic logic unit (ALU), one status register and one control register

Configured in **PSoC Creator**[4] using **PSoC Components**[2], or the graphical state machine editor or Verilog code

Serial Communication Block (SCB)

A PSoC **Programmable Digital Block** that is configurable as a UART, SPI or I^{2}C interface

[1] Connects the Programmable Analog Blocks, Programmable Digital Blocks and I/Os
[2] Free embedded ICs represented by an icon in PSoC Creator software
[3] Timer, counter, pulse-width modulator (PWM)
[4] PSoC 3, PSoC 4 and PSoC 5 Integrated Design Environment (IDE) software that installs on your PC

PSoC Terms

Timer, Counter, PWM (TCPWM) Block

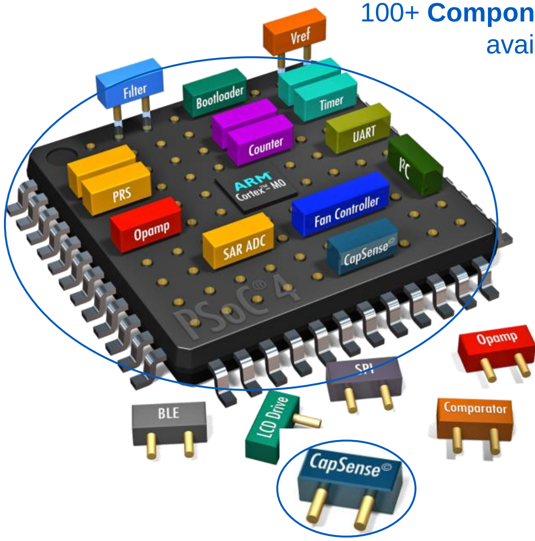
A PSoC **Programmable Digital Block** that is configurable as a 16-bit timer, counter, PWM or quadrature decoder

CapSense®

Cypress's third-generation touch-sensing user interface solution that "just works" in noisy environments and in the presence of water

The industry's No. 1 solution in sales by 4x over No. 2

100+ **Components** available



Programmable Interconnect and Routing

Connects the Programmable Analog Blocks, Programmable Digital Blocks and I/Os

Enables flexible connections of internal analog and digital signals to internal buses and external I/Os

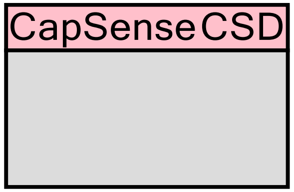
PSoC Creator™

PSoC 3, PSoC 4 and PSoC 5 Integrated Design Environment (IDE)

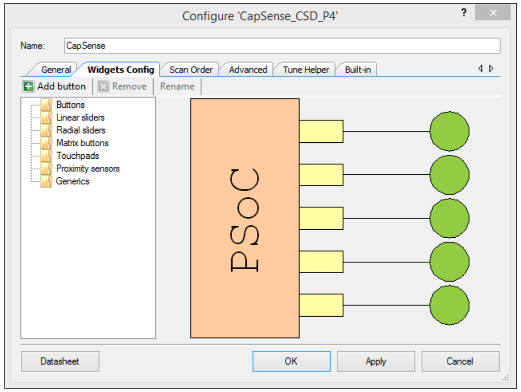
Software that installs on your PC that allows:
- Concurrent hardware and firmware design of PSoC systems, or
- PSoC hardware design followed by export to popular IDEs

CapSense is used to create **touch buttons and sliders**

Component Icon



Component Configuration Tool



Components

Free embedded ICs represented by an icon in **PSoC Creator** software

Used to **integrate multiple ICs** and system interfaces into one **PSoC**

Dragged and dropped as icons to design systems in PSoC Creator

Component Configuration Tools

Simple graphical user interfaces in PSoC Creator embedded in each Component

Used to customize Component parameters

PSoC Terms

PSoC 4

A PSoC with an ARM® Cortex®-M0 MCU

PSoC 4 BLE

A PSoC 4 with up to 256KB flash, 36 I/Os,
10 **Programmable Analog Blocks**, 10 **Programmable Digital Blocks**
and an integrated BLE radio with a royalty-free BLE Protocol Stack

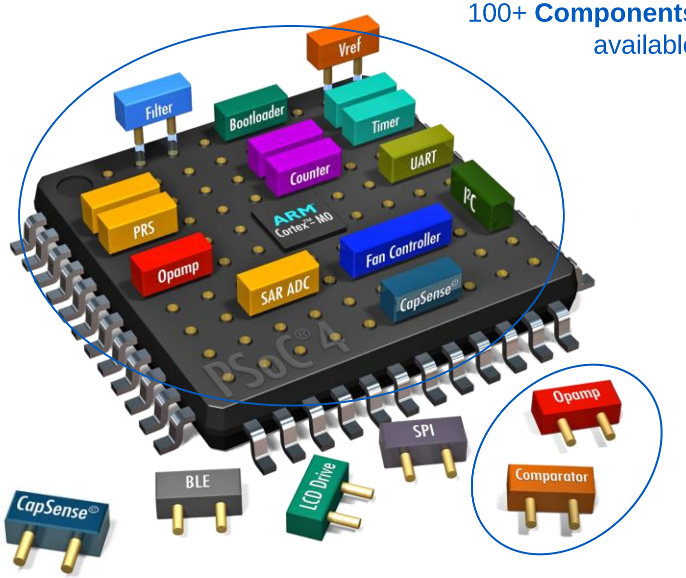
PRoC BLE (Programmable Radio-on-Chip)

An ARM Cortex-M0 MCU with up to 256KB flash, 36 I/Os,
2 **Programmable Analog Blocks**, 6 **Programmable Digital Blocks**,
an integrated BLE radio and a royalty-free BLE Protocol Stack

CySmart™

A GUI-based software tool that installs on your PC to test and debug BLE functionality

100+ **Components** available



Programmable Analog Components are used to create custom AFEs

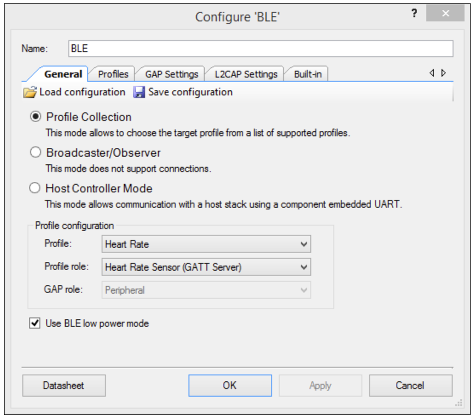
BLE Component

A Component that <u>creates **Bluetooth Smart** products in minutes</u>

Includes a Component Configuration Tool that makes the complex <u>BLE Protocol Stack and Profiles simple to implement with a GUI</u>

<u>Component Icon</u>

<u>Component Configuration Tool</u>



Introduction to BLE System Design

DEMO #1: PSoC CREATOR AND BLE PIONEER KIT OVERVIEW

Demo #1: PSoC Creator Overview

Objectives:

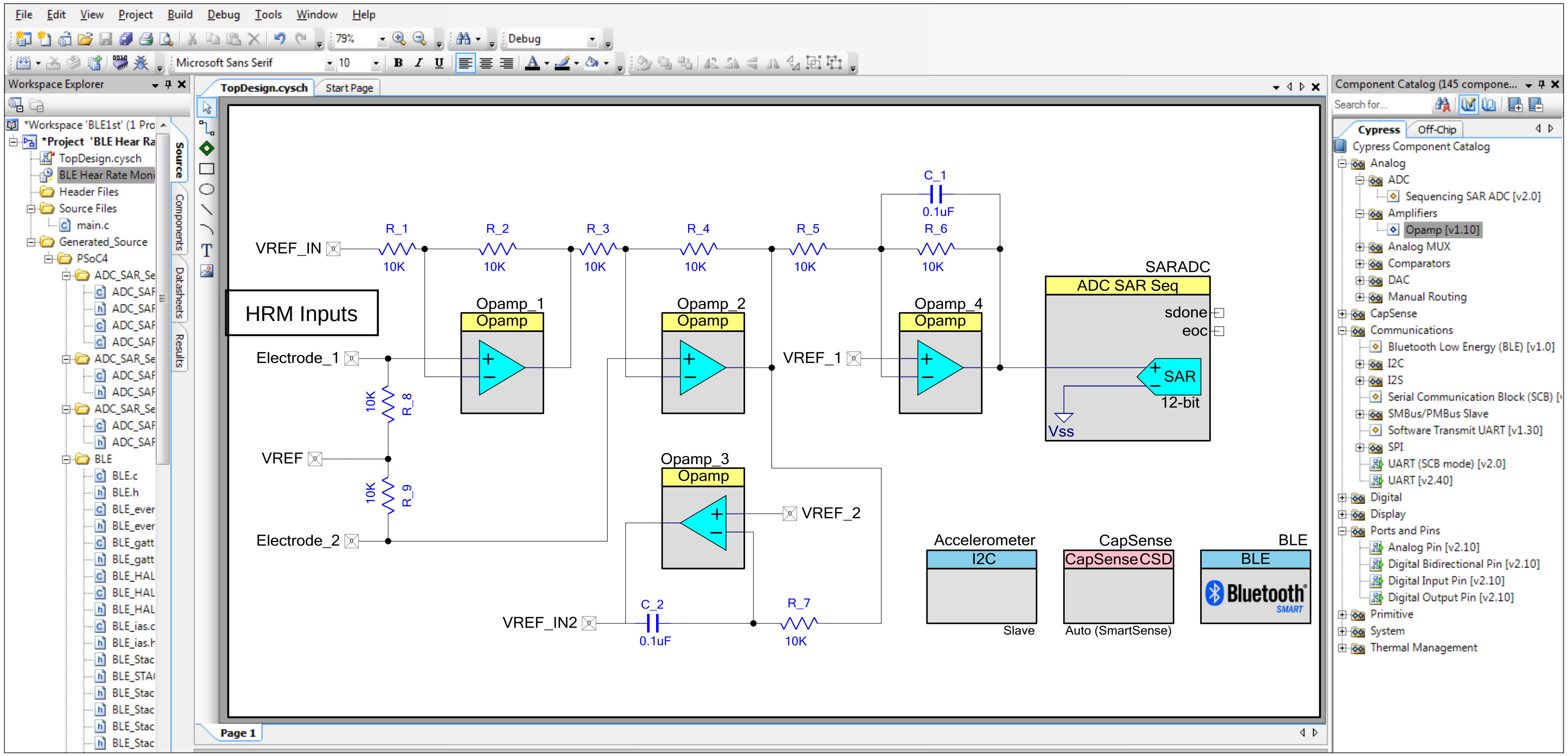
Learn about the PSoC Creator workflow:

- Create a new project
- Find 100s of example projects
- Place and configure a Component
- Open a datasheet
- Assign signals to pins
- Build and debug a design

Software tool:

PSoC Creator IDE

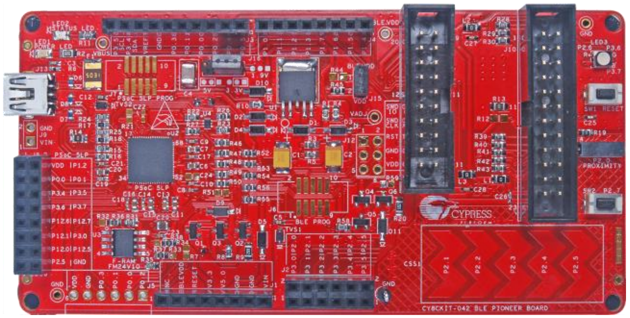
A BLE Heart Rate Monitor Example Project in the PSoC Creator IDE



BLE Pioneer Kit Overview

The BLE Pioneer Kit (CY8CKIT-042-BLE) contains:

BLE Pioneer Kit Baseboard



BLE Pioneer Kit baseboard

Is compatible with Arduino™ and Digilent® Pmod™ hardware ecosystems

Features onboard CapSense slider, RGB LED, push buttons and Cypress F-RAM

Includes PSoC 5 for program and debug

Supports 1.9 V, 3.3 V, 5 V and coin cell battery operation

Modules[1]

Two FCC-certified[2] BLE modules that plug into the BLE Pioneer Kit Baseboard

Feature an onboard antenna and provide access to all GPIOs

Support BLE-UART bridge via an onboard four-pin header

PSoC 4 BLE Module

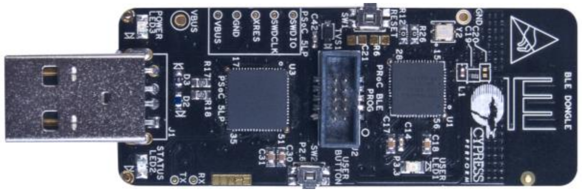
PRoC BLE Module

BLE-USB bridge with PRoC BLE

Enables the use of a PC to develop and debug BLE peripherals

Features an onboard LED, push button and PSoC 5 for program and debug

BLE-USB Bridge with PRoC BLE



Example projects

Demonstrate how to use PSoC Creator to implement common BLE Profiles, such as Heart Rate Monitor (HRM), Glucose Meter and Human Interface Device (HID)

Mobile apps[3]

Include CySmart mobile apps[3] for both iOS and Android mobile operating systems to test and debug BLE systems

[1] Additional BLE modules are available, refer to the wrap-up section for more details

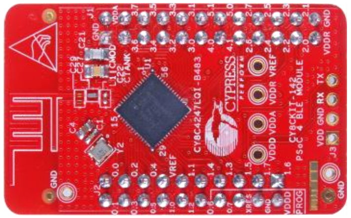
[2] Designation for products manufactured or sold in the U.S. that meet the electromagnetic interference standards of the Federal Communications Commission

[3] Mobile apps are software programs that run on a mobile device

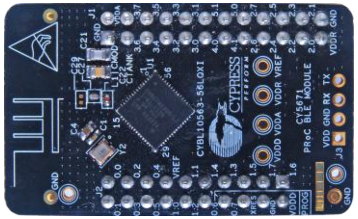
BLE Pioneer Kit Supports PSoC 4 BLE and PRoC BLE

Feature	PSoC 4 BLE	PRoC BLE
Applications	IoT sensor nodes, wearables, small home appliances, home automation and portable medical devices	Mice, keyboards, trackpads, game controllers, remote controls, toys and BLE bridges
CPU Core	ARM Cortex-M0	ARM Cortex-M0
CPU Speed (MHz)	48	48
Flash/SRAM Sizes (KB)	128/16-256/32	128/16-256/32
ADC	1-Msps 12-bit SAR[1]	1-Msps 12-bit SAR[1]
Opamps	4	-
Comparators	2	-
IDACs	2	-
UDBs	4	-
Timers, Counters, PWMs	4/4/8	4/4/8
CapSense (I/Os)	Yes (36)	Yes (36)
I/Os	36	36
Serial Interfaces	4 SPI, 2 I^2C, 4 UART, I^2S	2 SPI/I^2C/UART, I^2S
Packages	56-QFN, 68-CSP	56-QFN, 68-CSP

PSoC 4 BLE Module



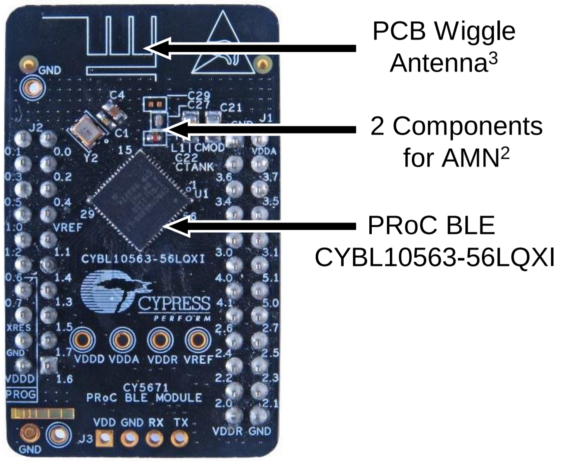
PRoC BLE Module



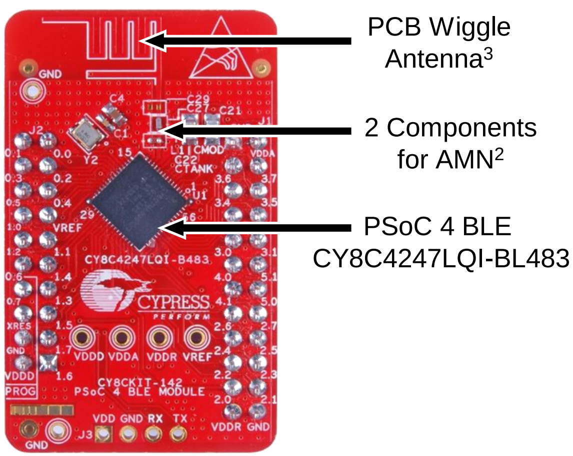
[1] Successive approximation register

PSoC 4 BLE and PRoC BLE Modules

PRoC BLE Module



PSoC 4 BLE Module

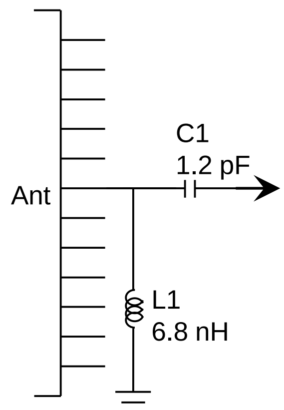


Cypress's BLE solutions integrate the Balun[1], simplifying AMN[2] design

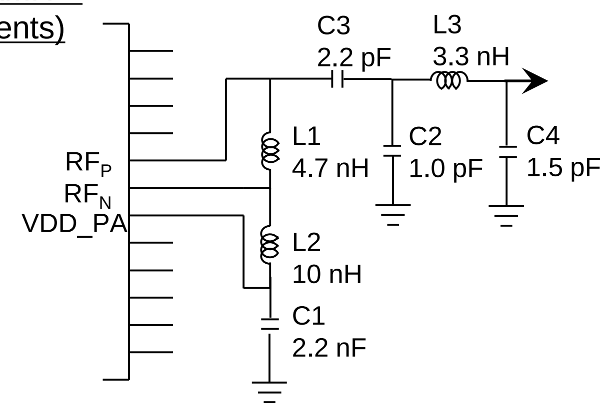
Traditional AMNs[2] are sensitive to PCB layout and parasitics and require tuning

Typical AMNs[2] have 7-9 external components vs. **only two** for the Cypress solution

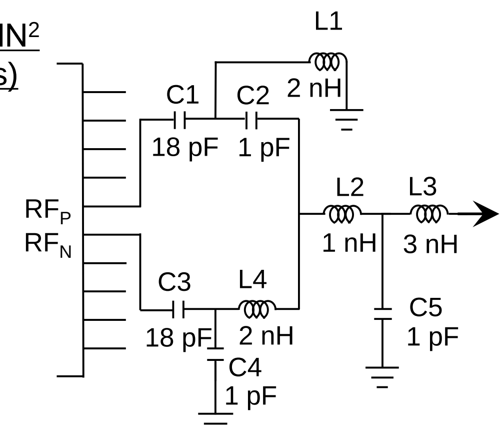
Cypress's AMN[2] (2 components)



Traditional AMN[2] (7 components)



Vendor B's AMN[2] (9 components)



[1] An electrical device that converts a differential RF signal to a single-ended signal or vice-versa
[2] Antenna matching network: An RLC circuit network that provides Balun functionality, antenna impedance matching and low-pass filtering
[3] A low-cost antenna made from a trace on the surface of a PCB

BLE Pioneer Kit Baseboard and BLE-USB Bridge

BLE Pioneer Kit Baseboard to Develop a "GAP Peripheral[1]"

Power-Selection Jumper (J16) (1.9 V, 3.3 V, 5 V)

Arduino-Compatible I/O Header (J3 and J4)

Current Measurement Jumper (J15)

BLE Module Header (J10 and J11)

RGB LED

BLE Reset Button

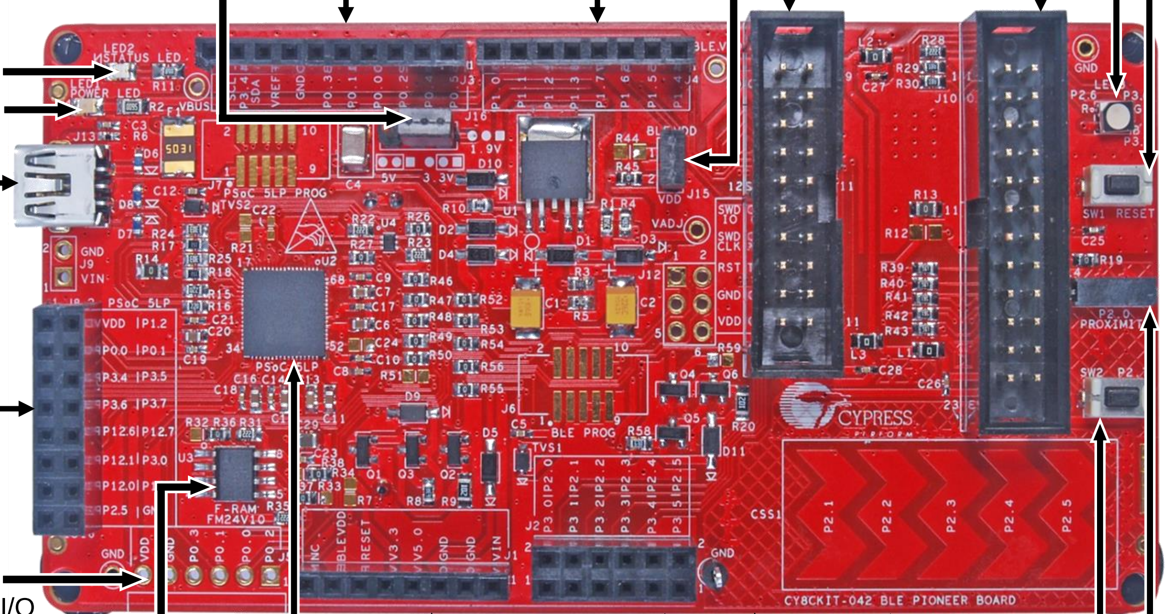
Status LED

Power LED

USB Connector (J13)

PSoC 5 I/O Header (J8)

Digilent Pmod-Compatible I/O Header



1Mb Serial F-RAM[3] FM24V10

PSoC 5 for Program and Debug

Arduino-Compatible I/O Header (J1 and J2)

Coin-Cell battery holder (on back, not shown)

CapSense® Slider

Push-Button Switch

Proximity Header (J14)

BLE-USB Bridge With PRoC BLE to Simulate a "GAP Central[1]"

Wiggle Antenna[2]

PRoC BLE CYBL10162-56LQXI

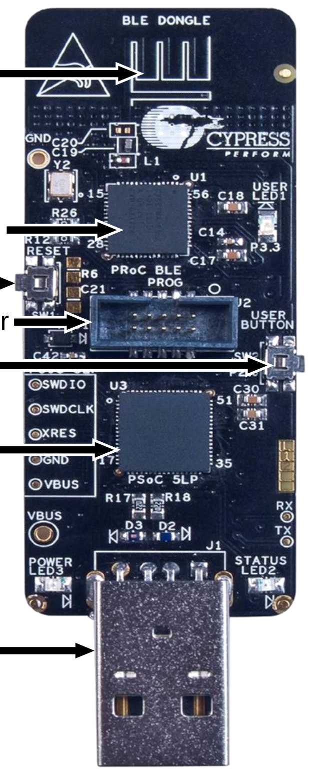
Reset Button

Programming Header

User Button

PSoC 5 for Program and Debug

USB Plug (J1)



[1] BLE terms that will be defined later
[2] A low-cost antenna made from a trace on the surface of a PCB
[3] Ferroelectric RAM with an I^2C serial interface

Introduction to BLE System Design

CYPRESS BLE SOLUTION OVERVIEW

Cypress BLE Solution Overview

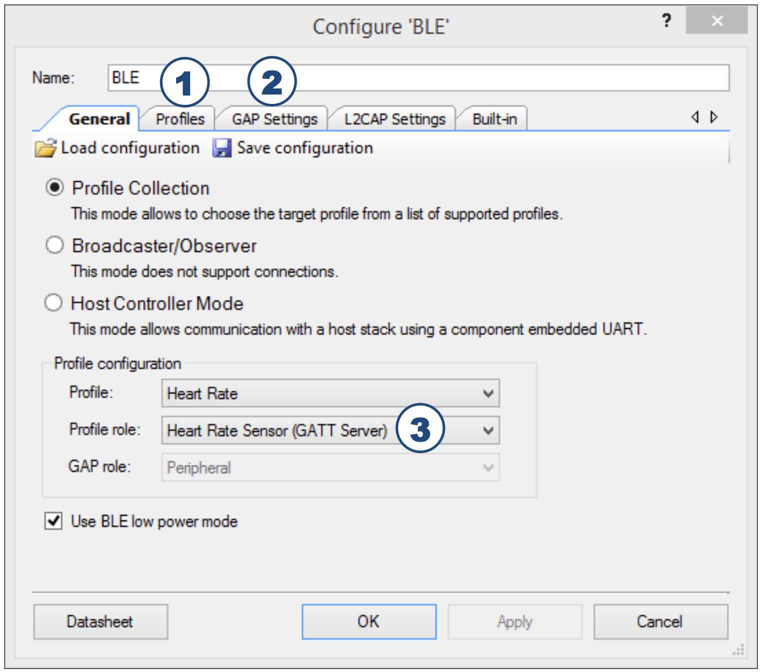
Cypress BLE integrates the entire BLE Architecture–Radio, BLE Stack and Application–on one chip

The royalty-free **BLE Stack** provided by Cypress is a complete implementation of the Bluetooth 4.1 Specification

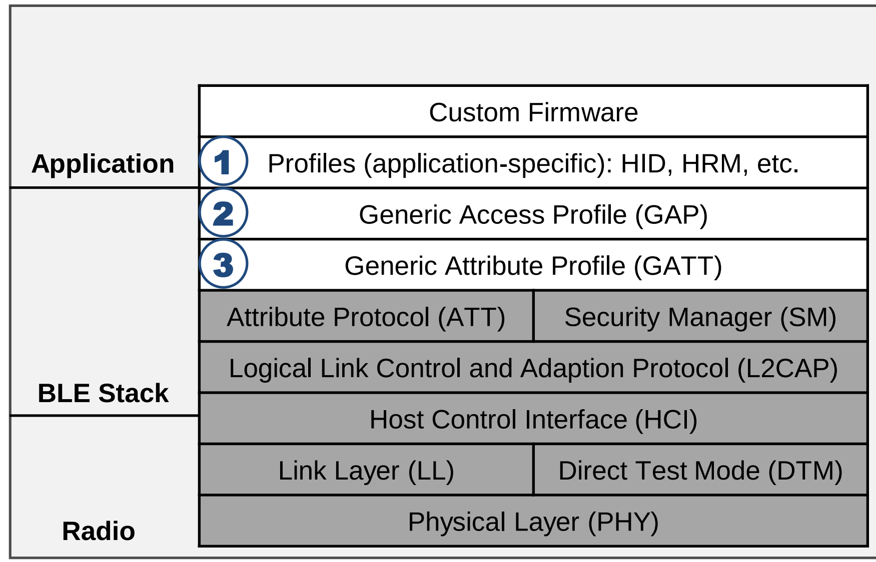
The **Application** is a combination of your firmware and the Cypress provided BLE Profile

The **Generic Access Profile (GAP)** and **Generic Attribute Profile (GATT)** define how BLE devices connect and exchange data[1]

Simple: BLE Component Configuration Tool



Difficult: Typical BLE Architecture Diagram from the Bluetooth Spec[2]



All of the parameters of the BLE Stack and Radio are configured using the BLE Component Configuration tool. **Simply select the Profile, GAP and GATT parameters** in the tool. The **BLE Component automatically configures the remaining parameters** for the BLE Stack and radio.

The BLE Component simplifies the BLE Stack and Profile configuration process into three simple steps

[1] GAP and GATT are further defined on the following slides

[2] Source for terms and definitions: *Bluetooth Low Energy: The Developer's Handbook*

BLE Profiles

BLE Profile

A Bluetooth specification that guarantees application-level interoperability between devices that use the same **Profile**.

For example, keyboards use the HID Profile and heart rate monitors (HRMs) use the HRM Profile

Standard Profiles (or Adopted GATT Profiles)

Guarantee interoperability between two devices using the same **Profile**

Defined by the **SIG** in the Bluetooth Spec

Assigned a 128-bit **Universally Unique Identifier (UUID)**

Natively supported by client[1] operating systems
  e.g., Google Android 4.x, Apple iOS 8.x, Microsoft Windows 8.1

Custom Profiles

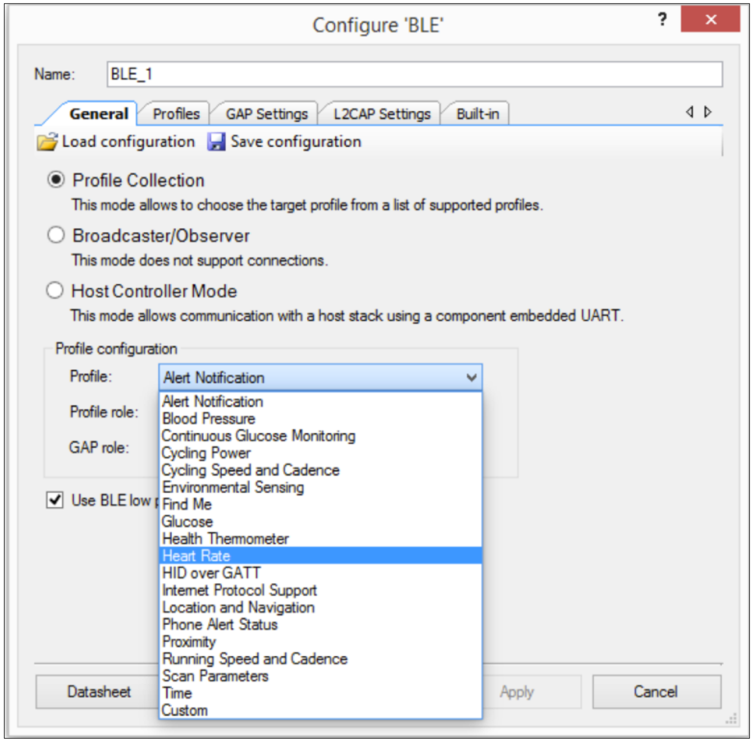
Non-standard **Profiles** for custom applications not defined by the **SIG**

Often provided by solution vendors for proprietary technologies
  e.g., Cypress provides a custom **CapSense Profile**

Require a custom **UUID**

Require custom software on the Client[1]
  e.g., Cypress provides mobile apps for iOS/Android Clients[1] with support for the **Custom CapSense Profile**

Profiles in the BLE Component



The BLE Component supports all Standard Profiles and enables quick creation of Custom Profiles that meet the Bluetooth Spec

[1] A BLE device that requests and receives data, e.g., a mobile phone

The Anatomy of a Profile[1]

A Profile is a collection of "Services"

1 For example, the Blood Pressure **Profile** contains four **Services**: "Generic Access," "Generic Attribute," "Blood Pressure" and "Device Information"

A Service is a collection of "Characteristics"

2 For example, the Blood Pressure **Service** contains three **Characteristics**: "Blood Pressure Measurement," "Intermediate Cuff Pressure" and "Blood Pressure Feature"

A Characteristic is a collection of "Attributes"

3 For example, the Blood Pressure Measurement **Characteristic** contains one **Attribute** referred to as a set of "Fields" in the Bluetooth Spec as seen on the image on the right

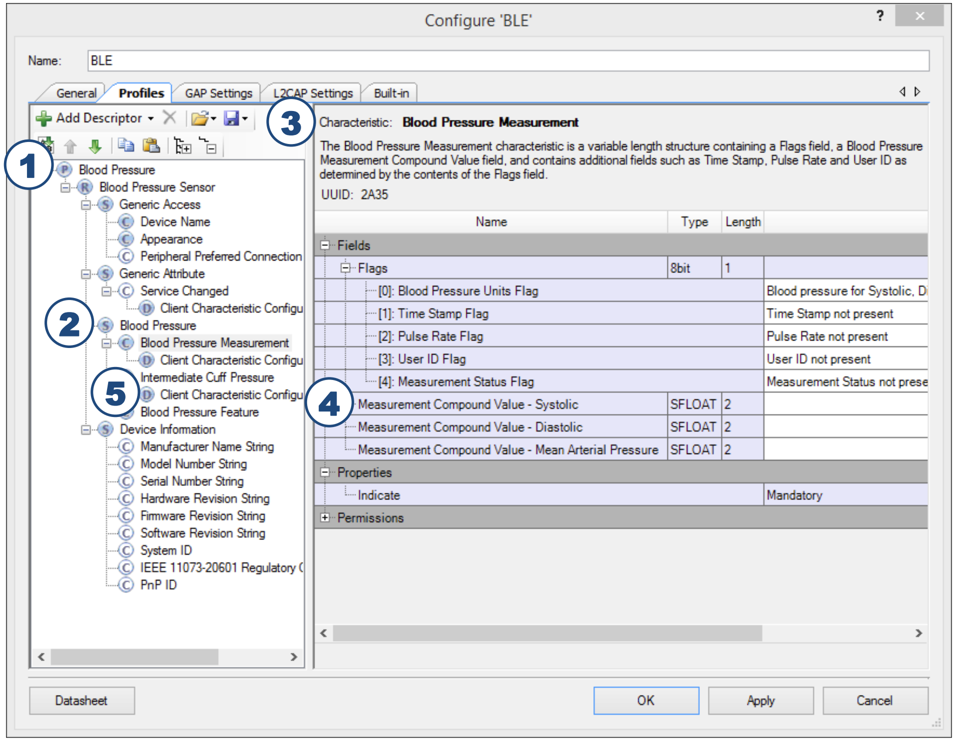
An Attribute is the smallest unit of information

4 For example, the actual Blood Pressure value stored in one of several "Measurement Compound Value" Fields as seen in the image on the right

A Descriptor is a type of Attribute

5 **Descriptors** provide additional information about a given **Characteristic**

Profile Tab in the BLE Component Configuration Tool



Easily configure the parameters for the Profiles, Services, Characteristics and Attributes

The BLE Component enables easy configuration of Profiles in the GUI-based Component Configuration Tool

[1] For more details on the actual Blood Pressure Profile, Services, Characteristics and Attributes refer to the Bluetooth Spec or BLE Component Datasheet

GAP: Establishing a BLE Connection

Generic Access Profile (GAP)

Defines how BLE devices discover each other, establish a connection and interact based on their roles

A BLE device can operate in the following “GAP roles”:

GAP Peripheral: Role in which a device, like a fitness monitor, connects to a **GAP Central** device, like a mobile phone

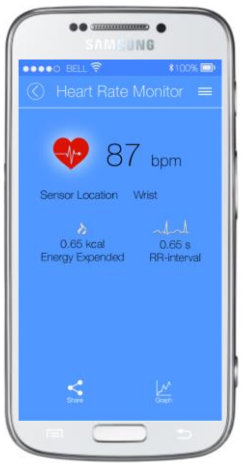
GAP Central: Role in which a device, like a mobile phone, connects to a **GAP Peripheral** device, like a fitness monitor

GAP Broadcaster: Role in which a device only advertises or transmits data[1]

GAP Observer: Role in which a device only listens or scans for devices[1]

Establishing a BLE Connection in Three Easy Steps

Bluetooth Smart-Ready Mobile Phone



Bluetooth Smart Fitness Monitor

GAP Central Scans for Peripherals	① ←	**GAP Peripheral** Transmits Availability and Capabilities
GAP Central Initiates Connection	② →	**GAP Peripheral** Accepts Connection
GAP Central Exchanges Data	③ ↔	**GAP Peripheral** Exchanges Data

[1] GAP Broadcaster and GAP Observer roles are included for completeness but not used in the this introductory workshop. Refer to the Appendix slide for examples of GAP roles.

GATT: Defining How to Communicate

Generic Attribute Profile (GATT)

Defines the way that two BLE devices exchange data

A BLE device can operate in the following "GATT roles":

GATT Server: A device that receives requests and sends data, typically a **GAP Peripheral**, like a fitness monitor

GATT Client: A device that requests and receives data, typically a **GAP Central**, like a mobile phone

GATT Database (DB)

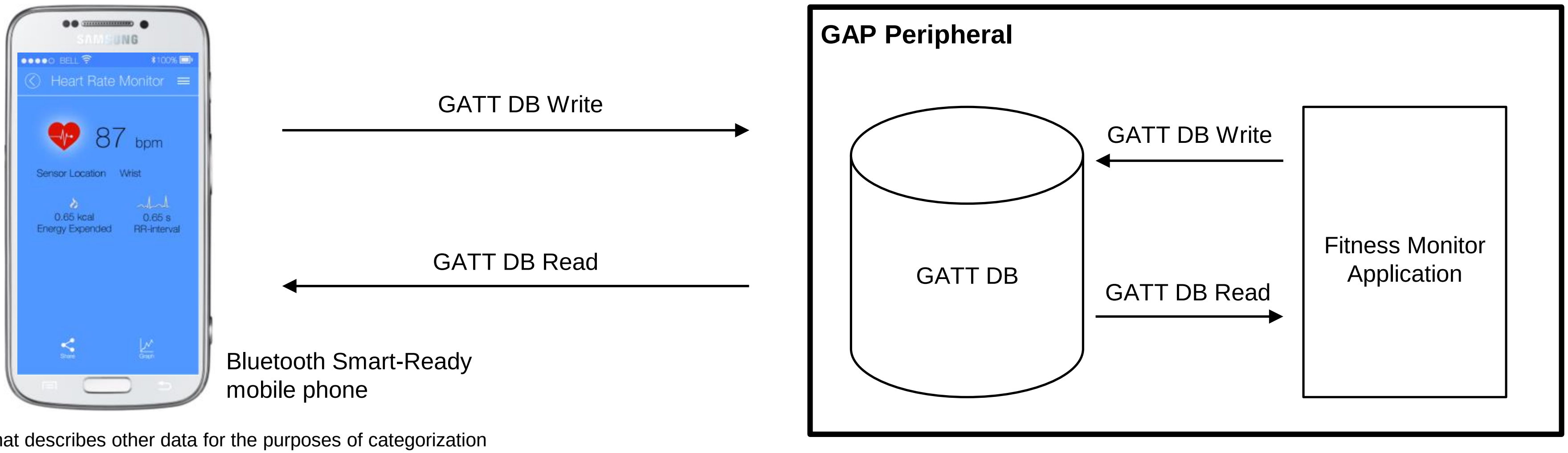
Stores and provides data and metadata[1] in the Bluetooth Spec format

Runs in a GAP Peripheral and responds to read and write requests from both GAP Central and the GAP Peripheral itself

BLE Communicates via GATT DB Reads and Writes

GATT Client in a GAP Central
Like a Mobile App[2] on a Mobile Phone

GATT Server in a GAP Peripheral
Like a Fitness Monitor Application



[1] Data that describes other data for the purposes of categorization
[2] Mobile apps are software programs that run on a mobile device

Attribute Protocol: Communication Example

Attribute Protocol (ATT) defines the rules for BLE communication

Enables **GATT Clients** to find and access **Attributes** on a **GATT Server** using six operations: Requests, Responses, Commands, Notifications, Indications and Confirmations

1. Read example (GATT Client initiated)
Method by which the GATT Client makes a request and the GATT Server responds with data

Example code:
```
CyBle_HrscGetCharacteristicValue(BodyLocation);
```

GATT Client
Sends **request** for data

Read Body Sensor Location →

GATT Server
Receives request:
responds with data

← Body Sensor Location = Wrist

GATT Client
Receives requested data

2. Write example (GATT Client initiated)
Method by which the GATT Client sends a command to the GATT Server

Example code:
```
CyBle_HrscSetCharacteristicValue(Command);
```

GATT Client
Sends a **command**

Write Energy Expended = 0 →

GATT Server
Receives the **command**

3. "Notify" example (GATT Server initiated)
Method by which the GATT Server sends a notification to the GATT Client without a request or confirmation

Example code:
```
CyBle_HrssSendNotification(Notification);
```

GATT Client
Receives the **notification**

← Notify Heart Rate = 87 bpm

GATT Server
Sends a **notification**

4. "Indicate" example (GATT Server initiated)
Method by which the GATT Server sends data to the GATT Client without a request and requires a confirmation

Example code for the GATT Server:
```
CyBle_GattsIndication(Indication);
```

Example code for the GATT Client response:
```
CyBle_GattcConfirmation( );
```

GATT Server
Sends an **indication** like an error message

← Indicate Heart Rate Sensor Error

GATT Client
Receives the **indication**:
responds with a
confirmation

Error Confirmed →

GATT Server
Receives the
confirmation response

Security Manager: Establishing Secure Connections

The Security Manager (SM) defines the following security methods:

Pairing: A process to establish a secure connection using authentication and key distribution

Authentication: A process to verify the identity of a device

Key distribution: A process of exchanging security keys contained in 128-bit data packets for pairing two devices

Bonding: A process of storing keys and authentication data in memory, so two devices can reconnect without the pairing process

Whitelist: An exclusive set of **GAP Central** devices that a **GAP Peripheral** can pair with, maintained in the **LL** hardware of the **GAP Peripheral** to enable low-power, secure and fast connections.

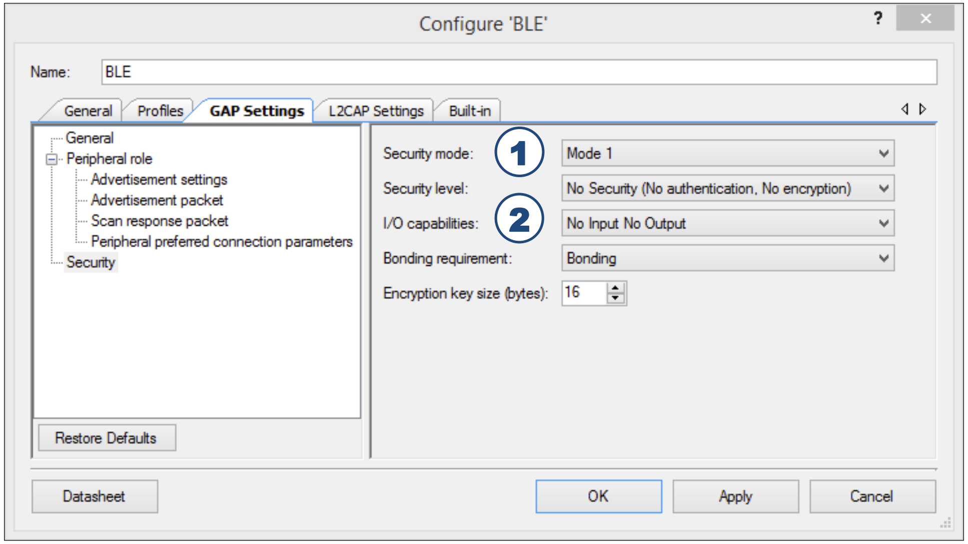
Cypress BLE SM: Supported I/O Capabilities and Security Levels[1]

1

Security	**Level 1**	**Level 2**	**Level 3**
Mode 1	No Security	Unauthenticated + Encrypted	Authenticated + Encrypted
Mode 2	Unauthenticated + Data Signed[2]	Authenticated + Data Signed[2]	N/A

2

I/O Capabilities
Display Only
Display: Yes/No
Keyboard Only
No Input, No Output
Display

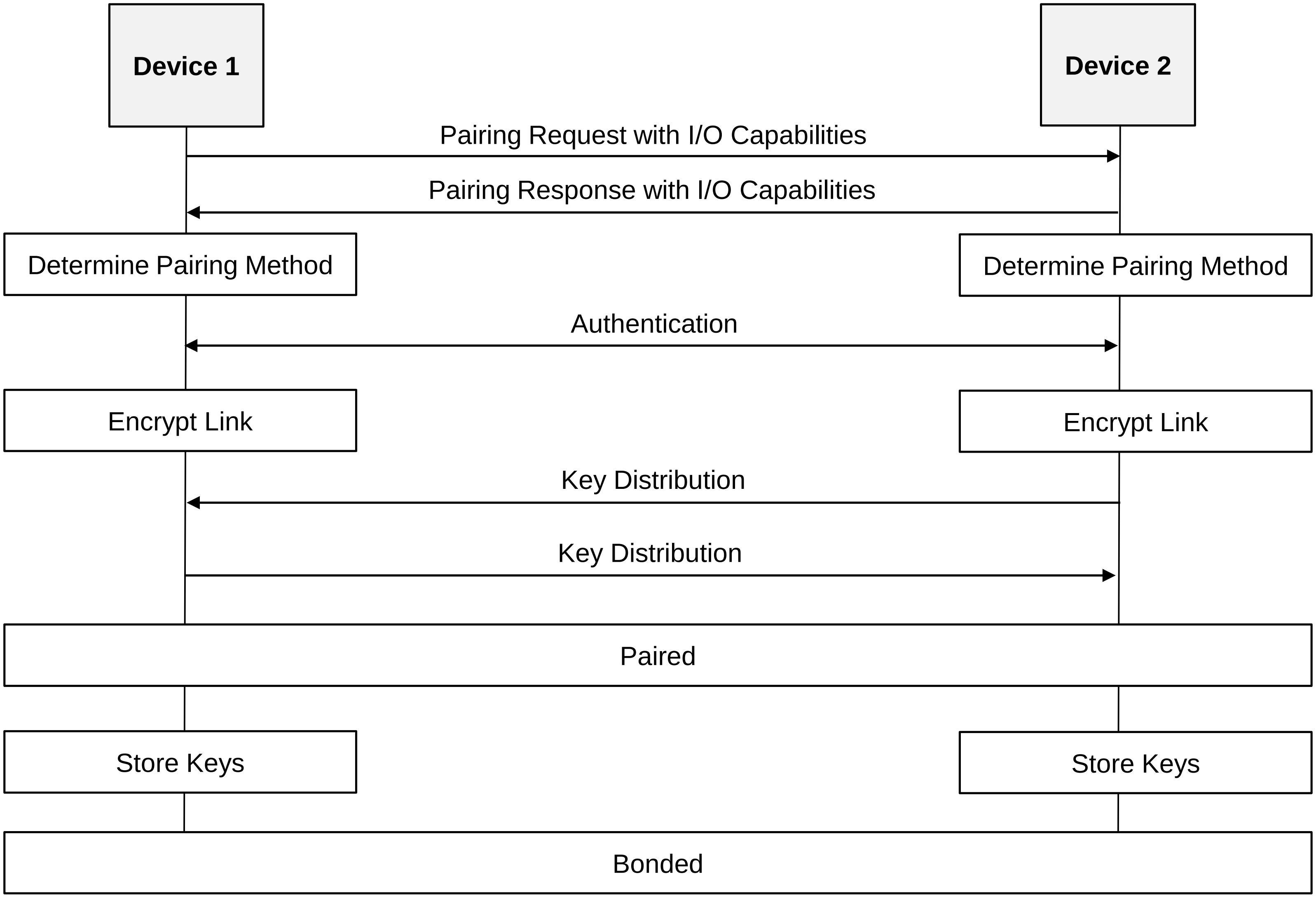


The BLE Component enables easy SM configuration in its GUI-based tool, without writing any firmware

[1] Refer to the BLE Component Datasheet for more information on these terms

[2] Data that is signed with a security key to ensure data integrity

Secure Connection Example



Introduction to BLE System Design

DEMO #2: BLE COMPONENT OVERVIEW

Demo #2: BLE Component Overview

Objectives:

Review the BLE Component and Component Configuration Tool

Learn where GAP, GATT and Profiles are configured

Learn how to use the Component Datasheet and APIs[1]

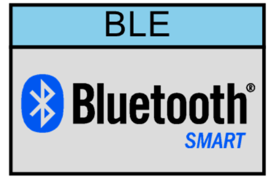
Software tool:

PSoC Creator IDE

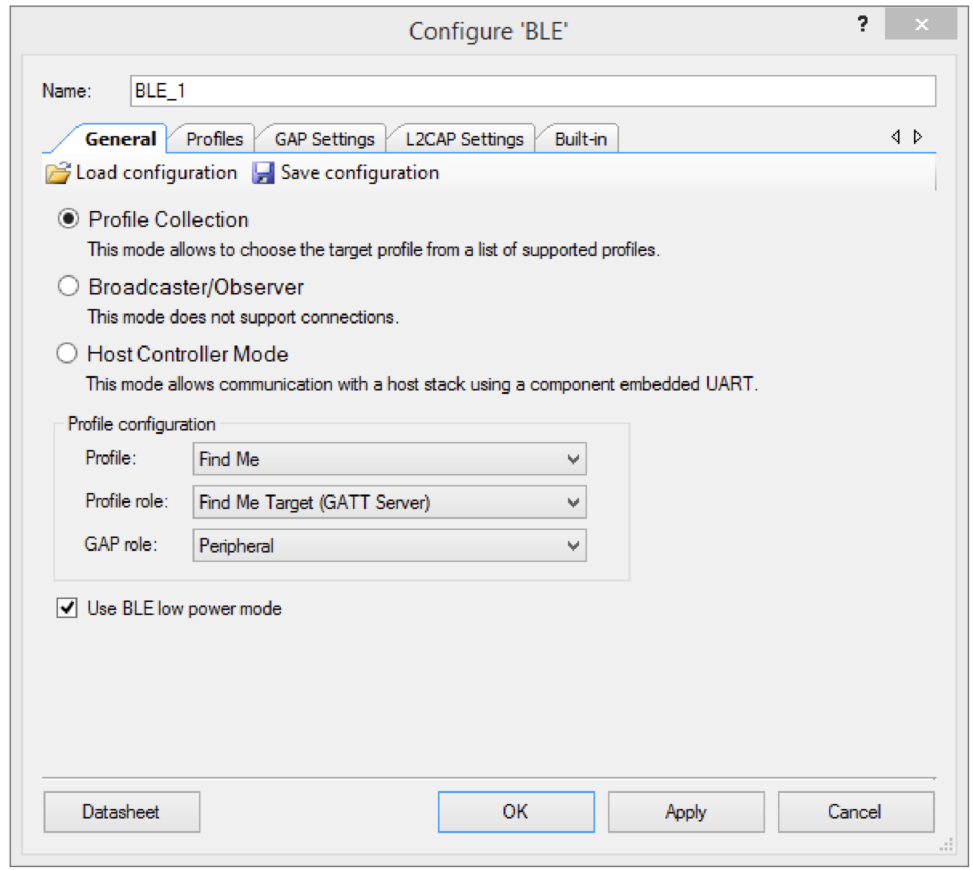
Component:

BLE Component

BLE Component Icon



BLE Component Configuration Tool



[1] Application programming interfaces are simplified sets of instructions used to interact with a Component

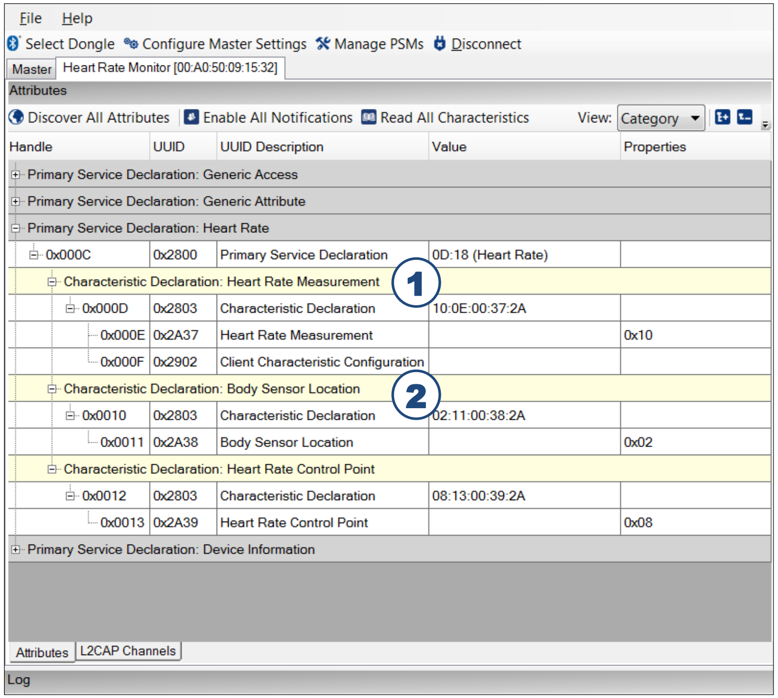
Introduction to BLE System Design

LAB #1: YOUR FIRST PSoC 4 BLE DESIGN

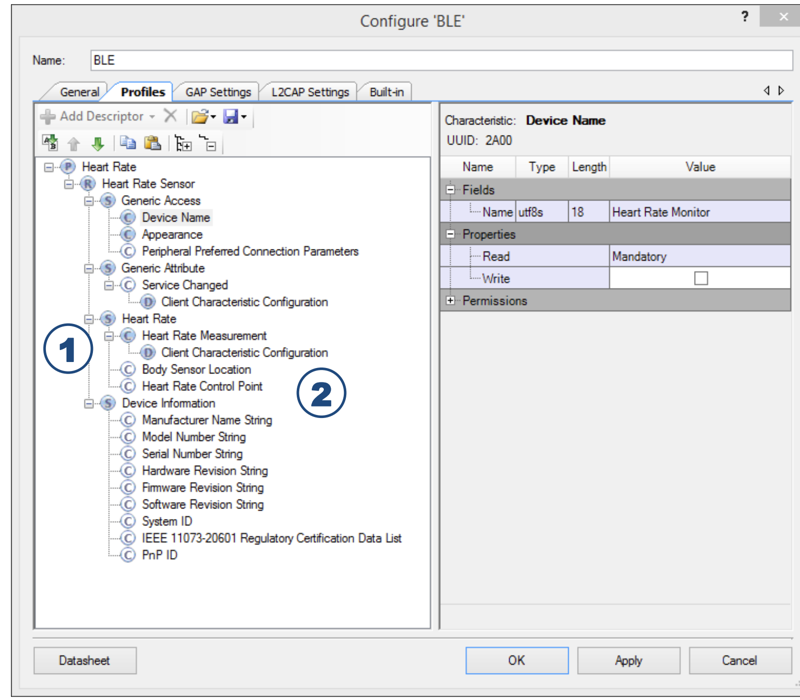
CySmart BLE Test and Debug Tool Overview

CySmart tool sends read and write requests to the GATT DB in BLE GAP Peripherals

CySmart BLE Test and Debug Tool

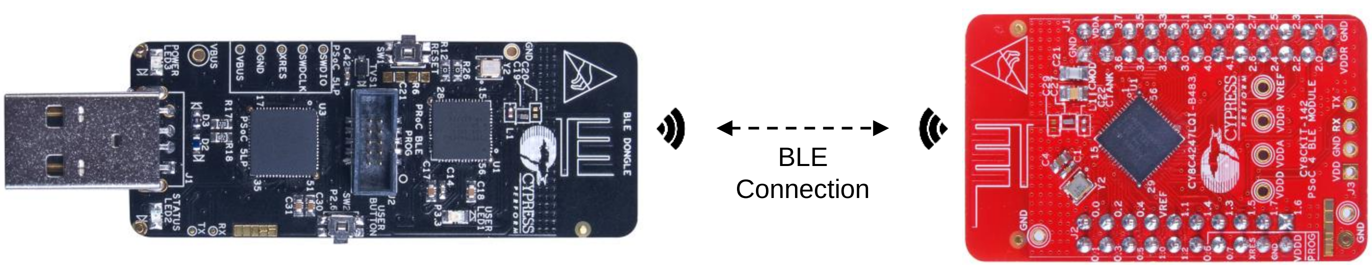


BLE Component Configuration Tool Profile Settings



BLE-USB Bridge with PRoC BLE Connected to a PC

PSoC 4 BLE Module Connected to the BLE Pioneer Baseboard

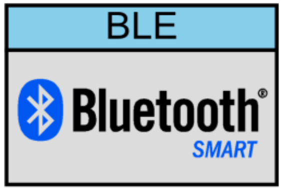


Lab #1: Your First PSoC 4 BLE Design

Objectives:

Learn how to use PSoC Creator to implement and debug PSoC designs
Implement a simple blinking LED design
Learn how to use the BLE Component
Implement a standard "Find Me" Profile with the Immediate Alert Service (IAS)[1]
Learn how to use the CySmart tool to debug BLE designs

BLE Component Icon



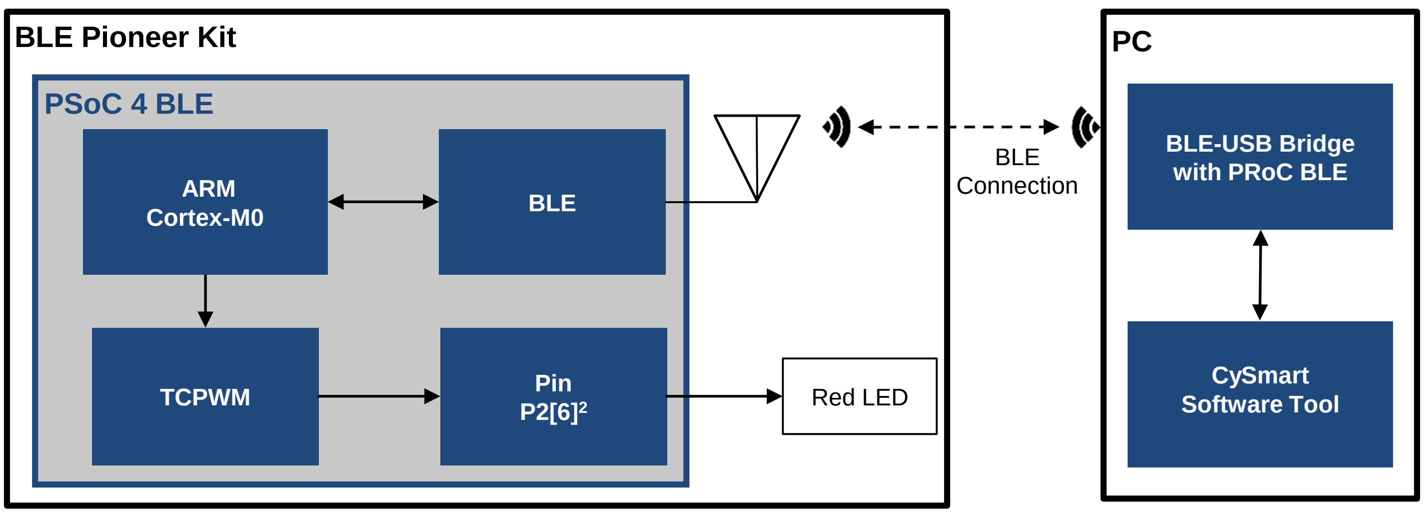
Software tools:

PSoC Creator IDE
CySmart

Components:

Pin Component
BLE Component
TCPWM Component

Lab 1: Block Diagram



[1] The "Find Me" Profile with IAS is a standard Profile in the Bluetooth Spec; refer to the Bluetooth Spec or Lab Manuals for more information
[2] Represents the logical pin placement at Port 3, Pin 7 in PSoC Creator

Introduction to BLE System Design

BLE ARCHITECTURE OVERVIEW

BLE Architecture

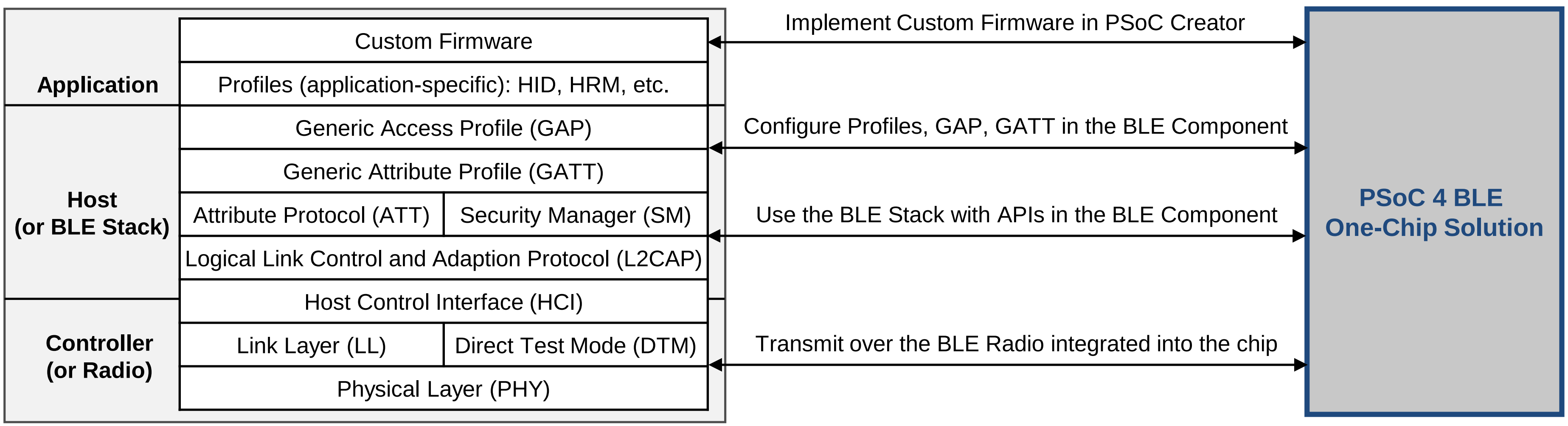
The BLE architecture consists of three parts: Application, Host and Controller

The **Application** implements specific functionality using the **Host** and **Controller**

The **Host** (or the **BLE Stack**) is a software stack with communication protocols that manage how two or more BLE devices communicate with each other

The **Controller** (or the **Radio**) is a physical device that transmits and receives encoded radio signals and decodes these signals

BLE Architecture Diagram from the Bluetooth Spec[1]



PSoC 4 BLE integrates the entire BLE architecture into an easy-to-use, one-chip solution

[1] Refer to the Appendix for definitions of all BLE architecture terms. Source for terms and definitions: *Bluetooth Low Energy: The Developer's Handbook*

BLE Radio: Physical Layer (PHY)

The PHY transmits or receives bits of data using a 2.4-GHz Radio

Uses GFSK[1] modulation in the 2.4-GHz ISM[2] band

Operates at 1-Mbps data rate

Consists of 40 RF channels with 2-MHz of spacing between channels

- 37 channels for data, 3 channels for Advertising[3]

RF Output Power: -18 dBm to +3 dBm (0.01 mW to 2 mW)

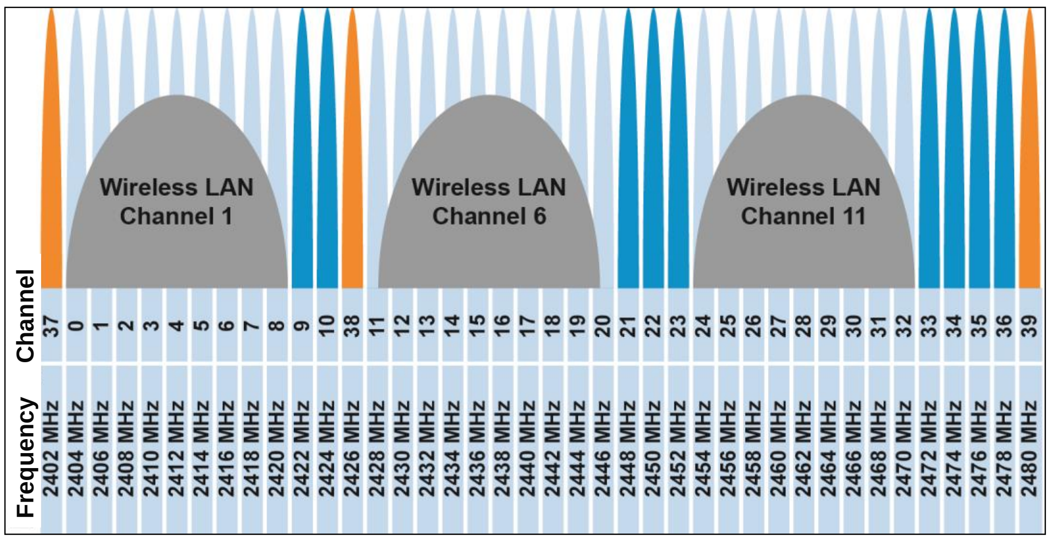
BLE Receiver Sensitivity: -89 dBm

BLE Architecture: PHY

Application	Custom Firmware	
	Profiles	
BLE Stack	GAP	
	GATT	
	ATT	SM
	L2CAP	
	HCI	
Radio	LL	DTM
	PHY	

Refer to slide 31 for abbreviation descriptions

Example of BLE and WiFi Channels Coexisting in the 2.4-GHz ISM[2] Band



- Available BLE channels (0-39)
- BLE Channels used to exchange data
- BLE Channels used to establish a connection
- 802.11 Wireless Local Area Network (LAN) channels

Spacing of channels in the BLE PHY allows Advertising[3] in the crowded 2.4-GHz ISM[2] band

[1] Gaussian frequency shift keying
[2] An Industrial, Scientific, Medical (ISM) RF frequency band that is license-free worldwide
[3] A state in which BLE devices broadcast data to advertise that they are connectable and discoverable to nearby peer BLE devices

BLE Radio: Link Layer (LL)

Implements procedures to establish a reliable physical link, including:

Advertising: A state in which a BLE device broadcasts data to advertise that it is connectable and discoverable by nearby BLE devices

Scanning: A state in which a BLE device scans for nearby advertising BLE devices

Initiating and Creating Connections: The device that initiates the connection is called the **LL Master**; the device that accepts the connection is the **LL Slave**

Data Encryption: A hardware block that implements AES-128 encryption

Error Detection: A hardware block that implements a 24-bit cyclic redundancy check (CRC)

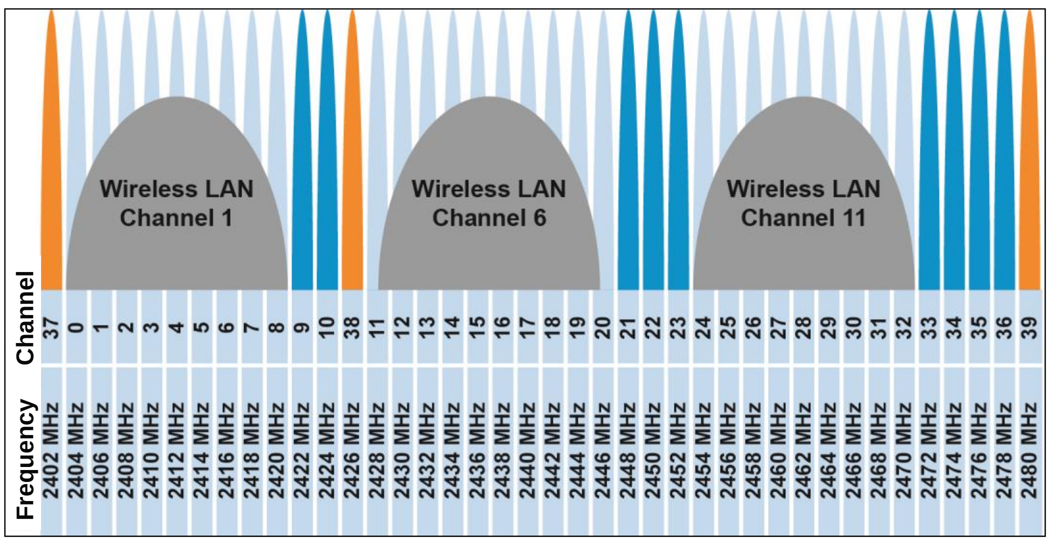
Adaptive frequency hopping (AFH): A process that enables BLE to adapt to the environment by avoiding channels that have poor signal strength or high error rates

BLE Architecture: LL

Application	Custom Firmware	
	Profiles	
BLE Stack	GAP	
	GATT	
	ATT	SM
	L2CAP	
	HCI	
Radio	LL	DTM
	PHY	

Refer to slide 31 for abbreviation descriptions

Example of BLE Adapting to the Environment to Avoid Bad Channels



AFH identifies bad channels 0-8, 11-20 and 24-32 as those with interference and does not use those channels for BLE communication

- Available BLE channels (0-39)
- BLE Channels used to exchange data
- BLE Channels used to establish a connection
- 802.11 Wireless Local Area Network (LAN) channels

The LL on PSoC 4 BLE is implemented in an integrated on-chip Radio

BLE Radio: DTM and HCI

Direct Test Mode (DTM)

A mode to test the **PHY** by transmitting or receiving a sequence of test packets

Typically used for radio compliance testing and production-line calibration

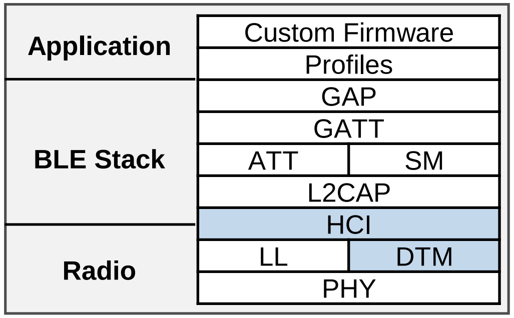
PSoC 4 BLE enables **DTM** via the **Host Control Interface**

Host Control Interface (HCI)

An interface to exchange data between the **BLE Stack** and the **Radio**

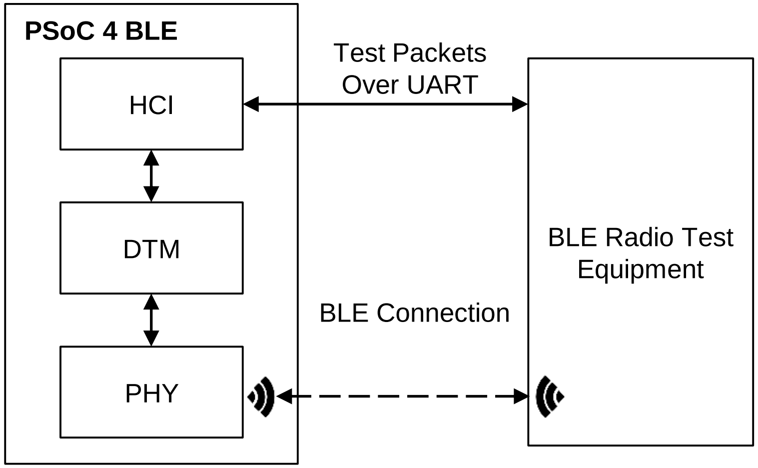
PSoC 4 BLE implements **HCI** over a UART interface to enter the **DTM**

BLE Architecture: HCI and DTM



Refer to slide 31 for abbreviation descriptions

PSoC 4 BLE in DTM for RF Compliance Testing and Calibration



PSoC 4 BLE simplifies RF compliance testing and calibration by providing a DTM over a UART interface

BLE Stack: L2CAP

Logical Link Control and Adaptation Protocol (L2CAP)

Segments large data packets into smaller packets

Reassembles segmented data into larger packets

Determines packet size by the **Maximum Transmission Unit**

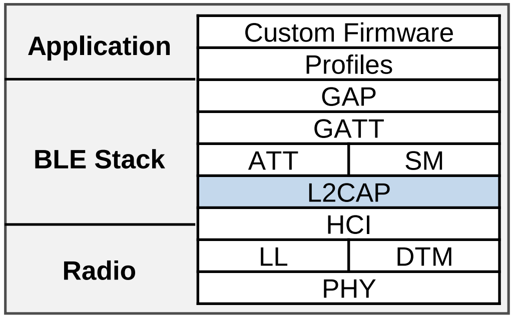
Maximum Transmission Unit (MTU)

The largest possible size for data packets

Segmentation and reassembly improve transmission efficiency by allowing larger **MTU**s

PSoC 4 BLE supports a **MTU** size of 23 to 512 Bytes

BLE Architecture: L2CAP



Refer to slide 31 for abbreviation descriptions

The PSoC 4 BLE L2CAP layer is integrated in the BLE Stack

Introduction to BLE System Design

LAB #3: IoT SENSOR-BASED SYSTEM DESIGN

IoT Sensor-Based Systems

The Internet of Things (IoT) is now a commercial reality

The IoT is how everyday physical objects are connected to the Internet

Fitness monitors are examples of new IoT devices

To learn more about fitness monitors download our *Wearables Solutions Catalog*

Fitness monitors require:

A heart rate monitor (HRM)

Activity monitoring and a step counter

BLE connectivity to a mobile device

A touch-based user interface

Maximum battery life

Designing a fitness monitor requires:

AFEs with opamps and an ADC to amplify, buffer and capture heart rate signals

Accelerometer to capture changes in motion

MCU with a BLE radio to connect to a mobile device

Touch-sensing IC to detect touches and gestures

ICs with low-power modes to minimize system power consumption

IoT products commonly require sensor-based BLE systems

Up3 Fitness Monitor by Jawbone

The newest Jawbone Up3 Fitness Monitor features a heart rate monitor, a touch-sensing interface and connectivity to mobile devices

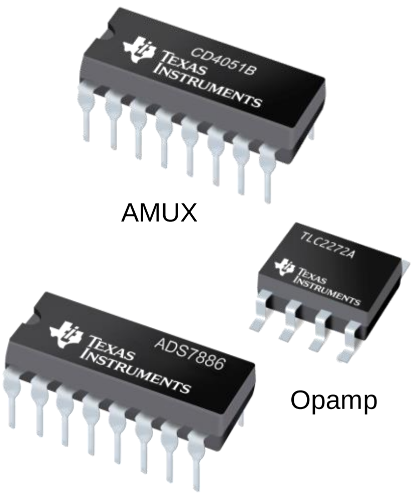
Microsoft Band Fitness Monitor

The new Microsoft Band includes a state-of-the-art heart rate monitor, a touch-sensing interface and connectivity to mobile devices

IoT Sensor-Based Systems Require Custom Analog Front Ends

Implementation of a Heart Rate Monitor AFE in PSoC 4 BLE

Integrate opamps, an **ADC**[1] and **AMUX** …

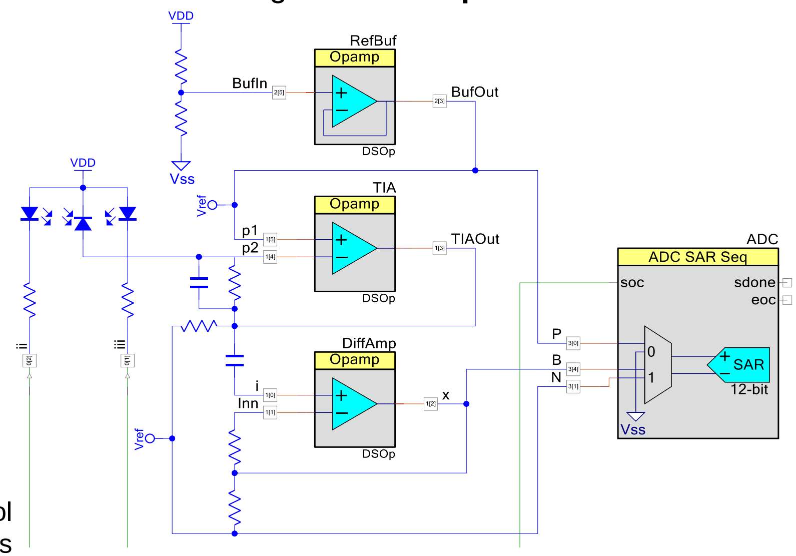


AMUX

Opamp

ADC[1]

With **Programmable Analog Blocks in PSoC Creator** using **PSoC Components**…



Digital Control Signals

To implement a **HRM AFE in PSoC 4 BLE**

PSoC 4 BLE Module

PSoC 4 BLE features Programmable Analog Blocks:

One 12-bit 1-Msps SAR[2] ADC

Four high-performance opamps (operational in Deep-Sleep mode)

Two low-power comparators (operational in Deep-Sleep mode)

Two current-output Digital Analog Converters (IDACs)

Two analog multiplexers (AMUX) that can be flexibly configured to create custom AFE designs

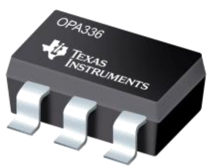
Programmable Analog Blocks can be flexibly configured to create custom AFEs for sensors

[1] MCUs with a BLE radio typically provide an ADC
[2] Successive approximation register

PSoC 4 BLE Integrates AFEs, CapSense and MCUs with a BLE Radio

Multiple AFE ICs, a touch-sensing IC and an MCU with a BLE radio…

Instrumentation Amplifier

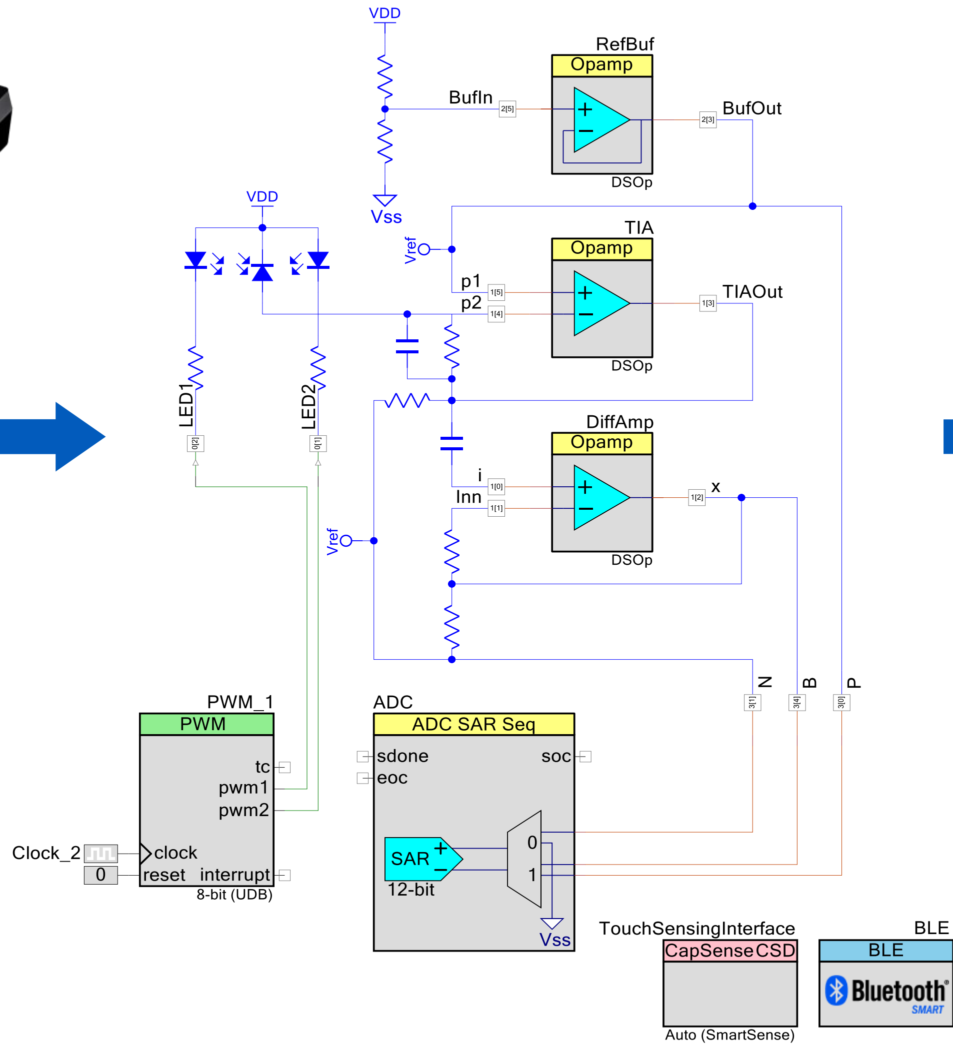


Dual-Channel Opamp

Touch-Sensing IC

MCU with a BLE Radio

Are integrated using **PSoC Components** in the **PSoC Creator** IDE…



To create a **PSoC 4 BLE one-chip solution** for the IoT.

PSoC 4 BLE Module

PSoC 4 BLE Provides Five Low-Power Modes to Minimize Power Consumption

Power Mode	Current Consumption	Code Execution	Digital Peripherals Available	Analog Peripherals Available	Clock Sources Available	Wake-Up Sources	Wake-Up Time
Active	1.7 mA @ 3 MHz	Yes	All	All	All	-	-
Sleep	1.3 mA	No	All	All	All	Any interrupt source	0
Deep-Sleep	**1.3 µA**	No	WDT[1], LCD[2], I^2C/SPI, Link-Layer[3]	Comparator, Opamps, POR[4], BOD[5]	WCO[6], 32-kHz ILO[7]	Comparator, GPIO[8], Opamp, Link-Layer[3], WDT[1], SCB[9]	25 µs
Hibernate	**150 nA**	No	No	Comparator, POR, BOD	No	Comparator, GPIO	2 ms
Stop	**60 nA**	No	No	No	No	Wake-Up pin, XRES[10]	2 ms

Power mode summary:

Average current 18.9-µA for a 1-sec connection interval

Stop mode consumes only 60 nA while retaining I/O state

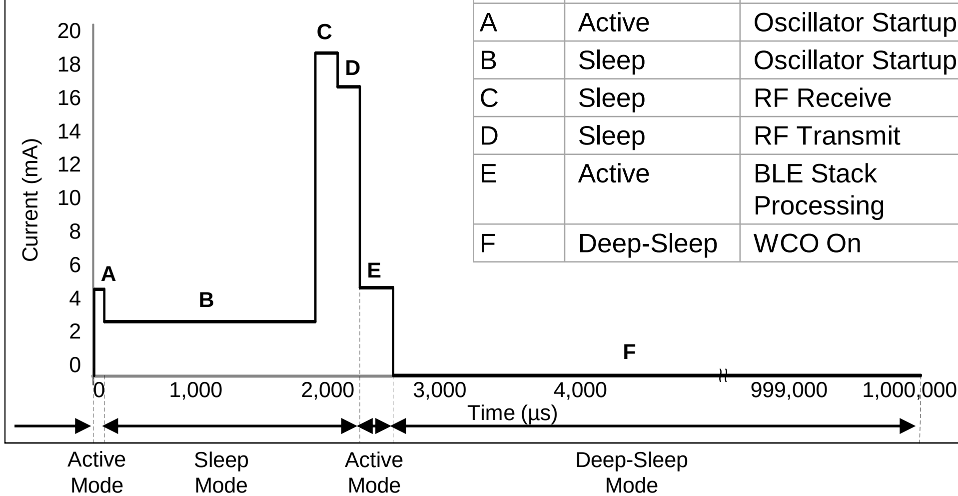
Hibernate mode consumes only 150 nA while retaining SRAM

APIs to switch easily between power modes

This lab uses four of the five low-power modes to create a low-power sensor-based system

Average current consumption for connection interval of 1 sec = **18.9 µA**

PSoC 4 BLE Current Consumption



Stage	Power Mode	Activity
A	Active	Oscillator Startup
B	Sleep	Oscillator Startup
C	Sleep	RF Receive
D	Sleep	RF Transmit
E	Active	BLE Stack Processing
F	Deep-Sleep	WCO On

[1] Watchdog timer
[2] Liquid crystal display
[3] Digital logic managing BLE Protocol
[4] Power-on-reset
[5] Brownout-detect
[6] 32-kHz watch crystal oscillator
[7] 32-kHz internal low-speed oscillator
[8] General-purpose input/output
[9] Serial communication block
[10] External reset

Lab #3: IoT Sensor-Based System Design

Objectives:

Measure simulated heart rate using the Programmable Analog Blocks

Implement a Heart Rate Monitor Profile and send the data over BLE

Optimize the design for low power consumption using Sleep, Deep-Sleep and Hibernate modes

Software tools:

PSoC Creator IDE

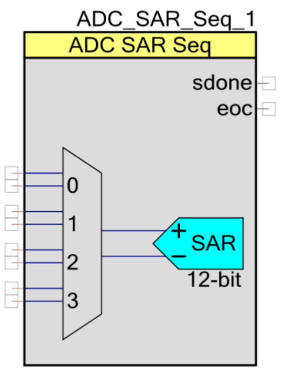
CySmart

Components:

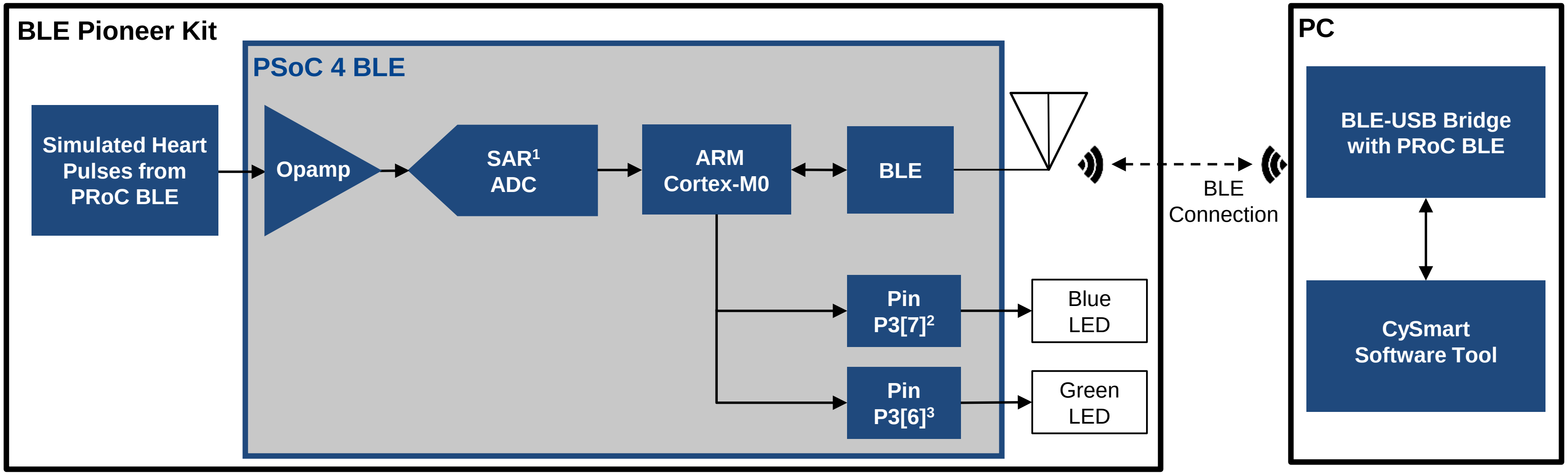
SAR[1] ADC Component

Opamp Component

SAR ADC Component



Lab 3: Block Diagram



[1] Successive approximation register [2] Represents the logical pin placement at Port 3, Pin 7 in PSoC Creator [3] Represents the logical pin placement at Port 3, Pin 6 in PSoC Creator

Introduction to BLE System Design

LAB #4: CapSense DESIGN WITH BLE CONNECTIVITY

CapSense Touch Sensing

CapSense replaces mechanical buttons

A capacitive sensor is used to measure the change in capacitance between a pin and ground

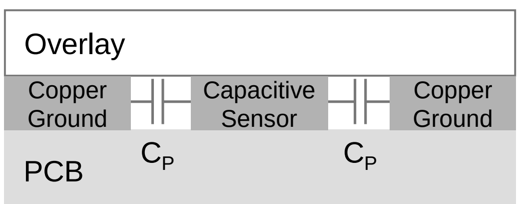
CapSense algorithms and analog circuitry convert the measured capacitance to a raw count

A finger touch increases the capacitance of the system, which in turn increases the raw count

An increase in the raw count above a user-defined threshold registers a touch

Refer to the *Getting Started With CapSense* Guide for details on CapSense algorithms

Capacitive Sensor Without a Finger Touch

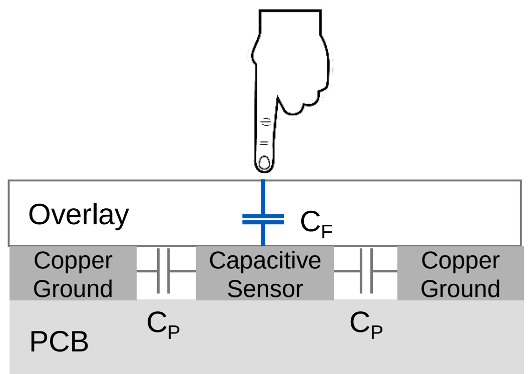


$C_X = 2C_P$

C_X = Total Capacitance on the capacitive sensor node

C_P = Parasitic capacitance

Capacitive Sensor With a Finger Touch

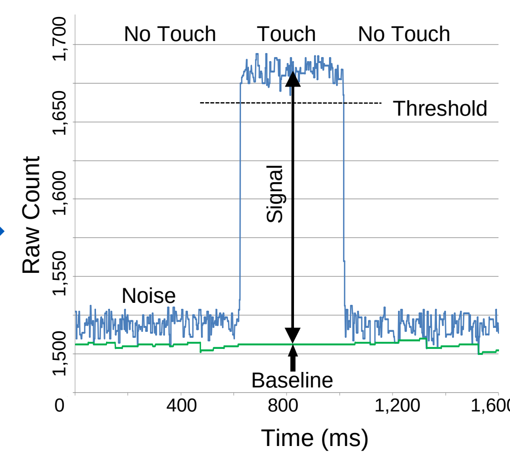


$C_X = 2C_P + C_F$

C_F = Capacitance added by a finger touch

C_F is dependent on the overlay material, overlay thickness and the dimensions of the finger (typical = 9mm) and sensor capacitances

Raw Count Variation on Finger Touch



CapSense algorithms use analog circuits to convert the capacitance to raw count, which is compared to the user-defined threshold to record a touch

SmartSense Auto-tuning

SmartSense Auto-tuning sets, monitors and continuously maintains optimal capacitive sensor performance

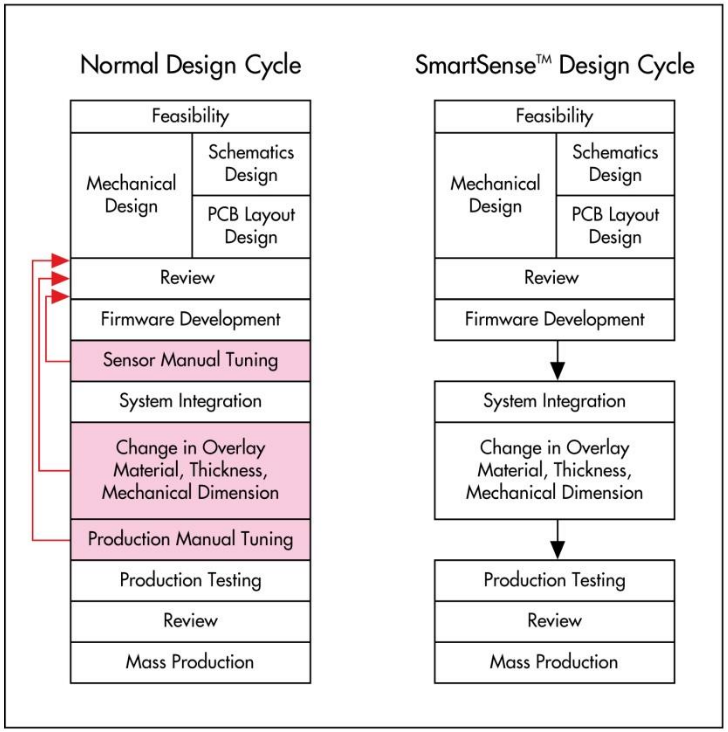
Reduces design effort by eliminating manual tuning (of baseline and threshold values) after the design phase

Adapts to manufacturing variations in PCB, overlay and paint that degrade touch-sensing performance

Adapts to changes in system environment due to RF noise sources

Allows a platform design approach that uses different overlays, button shapes and trace lengths with the same electronics

SmartSense Auto-tuning Cuts Design Cycle Time



Steps eliminated through Auto-tuning functionality

SmartSense Auto-tuning eliminates time-consuming manual tuning and the design iterations caused by it

Lab #4: CapSense Design with BLE Connectivity

Objectives:

Adjust RGB LED color and intensity using the Precision Illumination Signal Modulation (PrISM) Component

Implement a custom BLE Profile with a custom Service to send RGB LED color and intensity data over BLE

Implement a custom Service to send CapSense slider data over BLE

Use the CySmart application to validate the operation

Software tools:

PSoC Creator IDE

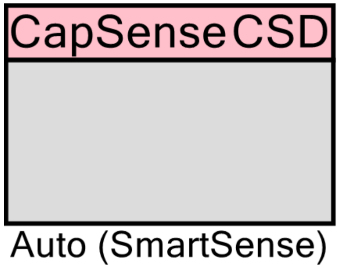
CySmart

Components:

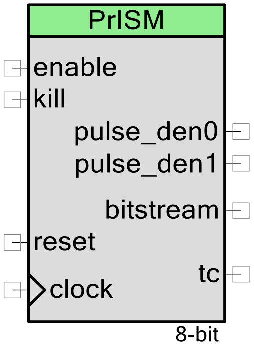
PrISM Component

CapSense CSD Component

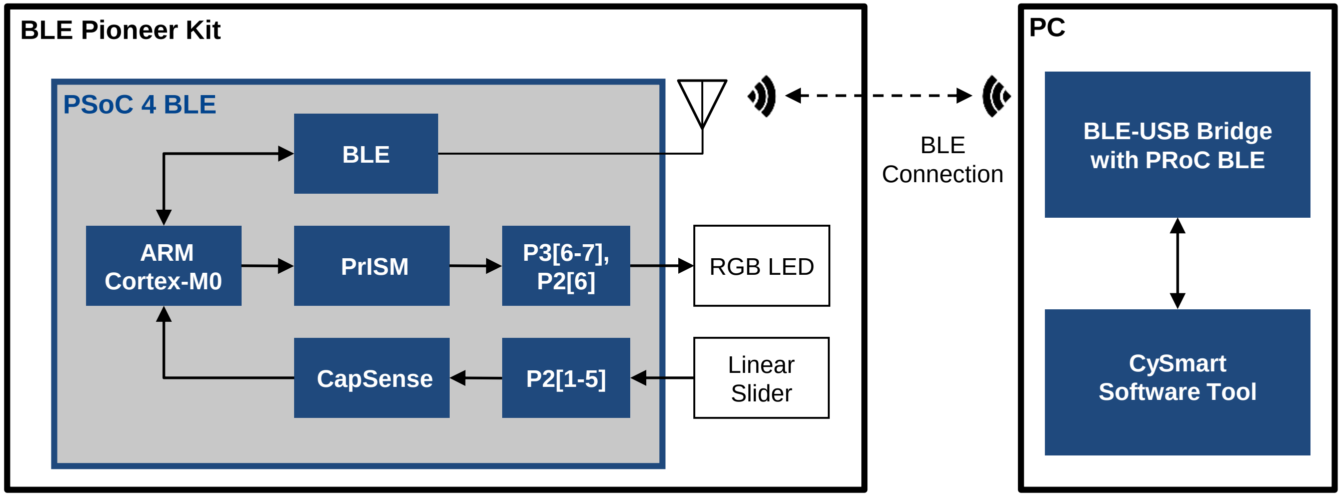
CapSense CSD Component



PrISM Component



Lab 4: Block Diagram



Introduction to BLE System Design

WRAP-UP

References and Links

Product Webpages:

Cypress's BLE solutions webpage: www.cypress.com/BLE

PSoC 4 BLE product webpage: www.cypress.com/PSoC4BLE
PSoC 4 BLE product datasheet: www.cypress.com/PSoC4BLEDatasheet
PRoC BLE product webpage: www.cypress.com/PRoCBLE
PRoC BLE product datasheet: www.cypress.com/PRoCBLEDatasheet
EZ-BLE PRoC Module webpage: http://www.cypress.com/EZ-BLEPRoCModule

PSoC Creator IDE: www.cypress.com/PSoCCreator
BLE Component Datasheet: www.cypress.com/go/comp_BLE
CySmart for Windows® PC: www.cypress.com/CySmart
CySmart for Mobile Apps: www.cypress.com/CySmartMobile
BLE Pioneer Kit: www.cypress.com/CY8CKIT-042-BLE
Remote Control RDK: www.cypress.com/CY5672
Touch Mouse RDK: www.cypress.com/CY5682

BLE Frequently Asked Questions: www.cypress.com/PSoC4BLEKBA
Cypress Wearables Solution Catalog: www.cypress.com/go/WearablesCatalog

Application Notes:

Getting Started with PSoC 4 BLE (AN91267): www.cypress.com/go/AN91267
Getting Started with PRoC BLE (AN94020): www.cypress.com/go/AN94020

Design Guides:

PSoC 4 BLE Antenna Design Guide: www.cypress.com/go/AN91445
CapSense Design Guide: www.cypress.com/go/AN85951

General Online Resources

Cypress Resources

PSoC: www.cypress.com/PSoC
Cypress Roadmap: www.cypress.com/Roadmap
Kits: www.cypress.com/kits
Support: www.cypress.com/support
Training: www.cypress.com/training
Cypress Online Store: www.cypress.com/store
Developer Community & Forums: www.cypress.com/forums
App Notes: www.cypress.com/AppNotes

Cypress BLE Solutions: www.cypress.com/BLE

Cypress's BLE solutions webpage is your *one-stop-shop* for everything BLE, including product datasheets, development kits, App Notes, software downloads, example projects and demo videos

Bluetooth Resources

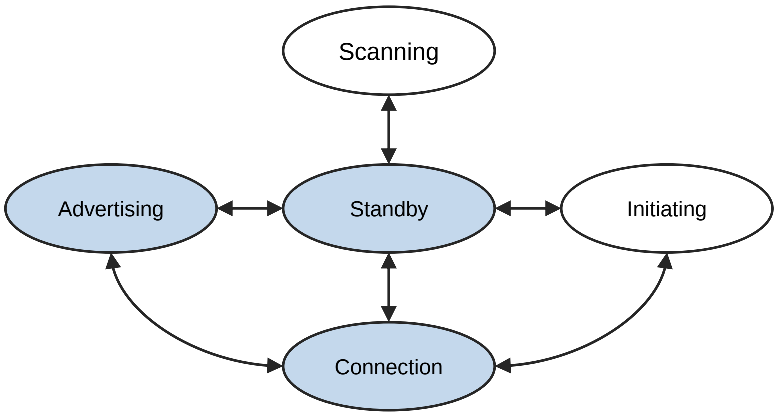
Bluetooth SIG website: www.bluetooth.org
Bluetooth Spec (including Profiles and Services): www.bluetooth.org/en-us/specification/adopted-specifications
Bluetooth Low Energy - The Developer's Handbook by Robin Heydon (ISBN-10:013288836X)

Introduction to BLE System Design

APPENDIX

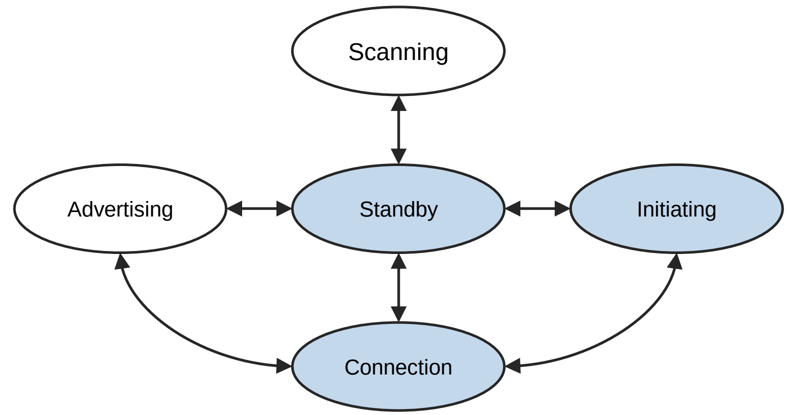
GAP: Example of GAP Roles

GAP Peripheral



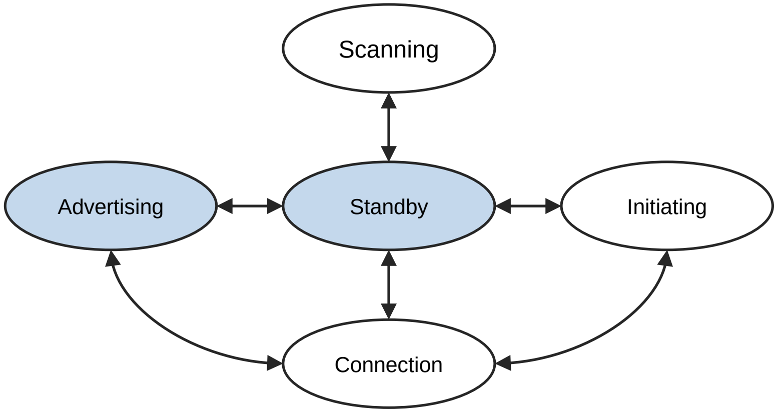
Advertises its capabilities and establishes connections

GAP Central



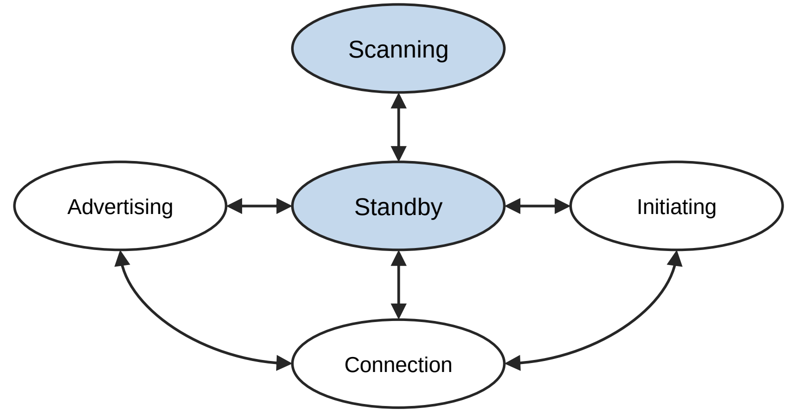
Scans for advertising devices and initiates connections

GAP Broadcaster



Advertises its capabilities only, does not establish connections

GAP Observer



Scans for advertising devices only, does not establish connections