STM32WB Workshop

Americas Marketing and Applications Team
Factoid of the Day

Harald Blatand Gormsson, Danish King

Reign: 958-986
Factoid of the Day

a.k.a. Harald “Bluetooth” Gormsson

Bluetooth: A wireless technology named after Harald Bluetooth, with the analogy that Bluetooth would unite devices like Harald Bluetooth united the tribes of Denmark into a single kingdom.
Objectives

STM32WB Introduction
BLE Basics
Tools & Firmware
Lot’s of Hands-On coding!
<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00</td>
<td>Welcome and Tools install (if needed)</td>
</tr>
<tr>
<td>9:00</td>
<td>A few words..</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-On:</strong> Out-of-the-Box</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-On:</strong> CubeMX</td>
</tr>
<tr>
<td></td>
<td>BLE Basics</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-On:</strong> HRM</td>
</tr>
<tr>
<td>10:30</td>
<td>Break</td>
</tr>
<tr>
<td></td>
<td><strong>Architecture</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Hands-On:</strong> CubeMonitorRF</td>
</tr>
<tr>
<td>12:00</td>
<td>Lunch (1h)</td>
</tr>
<tr>
<td>1:00</td>
<td>More WB Detail</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-on:</strong> Cable replacement</td>
</tr>
<tr>
<td></td>
<td>Hardware considerations</td>
</tr>
<tr>
<td>2:15</td>
<td>Break</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-on:</strong> OTA Firmware Updates</td>
</tr>
<tr>
<td>2:45</td>
<td>Wrap-Up, Q&amp;A, Survey</td>
</tr>
<tr>
<td>5:00</td>
<td></td>
</tr>
</tbody>
</table>
Prerequisites

- Windows 7/10
  - Java JRE v8 (v1.80.0_191 or newer)
- CubeMX, CubeWB, CubeMonitorRF, CubeProgrammer
- ST BLE Sensor App
- LightBlue Explorer App
- IAR EWARM, v8.40.1 + License
- TeraTerm, or equiv.
LightBlue Explorer

PunchThrough

The industry-leading BLE test app for iOS and Android. Used by over a half million people, LightBlue Explorer lets you scan, connect to and browse any nearby Bluetooth Smart device. Includes full support for logging data and simulating peripherals.

Download on the App Store

GET IT ON Google Play
• Activate the time-limited evaluation license with Activation Response file included in the provided zip file

• From IAR Embedded Workbench, go to Help->License Manager

• From the License manager, go to License->Offline Activation
Click “Use an activation response file from IAR Systems” option

Open the ActivationResponse_7900-043-869-0401.txt file
Click “Next”  

Click “Done”
Communication Technologies

<table>
<thead>
<tr>
<th>Datarate</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mbps</td>
<td>850/1900 MHz</td>
</tr>
<tr>
<td>Kbps</td>
<td>900/1800 MHz</td>
</tr>
<tr>
<td>bps</td>
<td>2.4 GHz</td>
</tr>
</tbody>
</table>

WiFi / BT

Short Range

Cellular

- 4G LTE-M
- 4G LTE-NB-IOT
- 5G

LPWAN

- LoRa
- SIGFOX
- M2M
- Sub-GHz

850/1900 MHz
900/1800 MHz
2.4 GHz
13.56MHz
Sub-GHz
Low-data-rate 2.4GHz connectivity

Bluetooth Smart
Point-to-point communication with smartphones and other wireless devices

BLE Mesh / 802.15.4
Home automation with Mesh network
“SMART READY” and “SMART” are abandoned markings
Ultra-low-power and RF
STM32WB Key Takeaways

- Multi-protocol
- Secure
- Dual-core Ultra-Low Power
- Comprehensive Ecosystem
Multiprotocol

- Bluetooth 5
- 2.4 GHz Open
- Thread
- ZigBee
Multiprotocol

- Bluetooth 5
  - Fully certified
  - 2Mbps
  - BLE Mesh

- 2.4 GHz
  - Open
Multiprotocol

- Zigbee 3.0
- OpenThread
- Concurrent BLE + OpenThread
Multiprotocol

- Bluetooth 5
- 2.4 GHz Open
  - OpenMAC
• Fully certified
• Legacy cluster support
• Revision R21 to R23
• Coming in late 2019
**What it delivers**

A secure wireless mesh network for your home and its connected products

- Built on well-proven, existing technologies
- Uses 6LoWPAN and carries IPv6 natively
- Runs on existing 802.15.4 silicon
- New security architecture to make it simple and secure to add / remove products
- 250+ products per network
- Designed for very low power operation
- Reliable for critical infrastructure

**Can support many popular application layer protocols and platforms**

- Application
- UDP
- IP Routing
- 6LoWPAN
- IEEE 802.15.4 MAC
- IEEE 802.15.4 PHY

**A software upgrade can add Thread to currently shipping 802.15.4 products**
Bluetooth Mesh

- Based on Bluetooth 4.0 and later
- Broadcast type, flood the network with messages, no routing
- Shorter range, 3kbps application data rate, 1Mbps on-air bit-rate
- High power consumption

Thread

- IPv6-based using 802.15.4 MAC
- Routing table approach with network self healing
- Medium range, 40Kbps application data rate, 250Kbps on-air bit-rate
- Low power consumption
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

STM32CubeWB
STM32CubeMX
STM32CubeProgrammer
STM32CubeMonitorRF
CubeWB HAL Firmware
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware
Cube Tools

STM32CubeMX
STM32CubeProgrammer
STM32CubeMonitorRF
CubeWB HAL Firmware
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware
Cube Tools

STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware
STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWb HAL Firmware

Cube Tools
Core folder contains application-related source code
Different stacks required for different application types

- BLE projects
- BLE + Thread Static Concurrent mode project
- Open MAC project
- Thread projects

Zigbee 3.0 coming soon!
CubeWB firmware

Encrypted radio stack binaries here

HTML file details update procedure

Nucleo & Dongle boards come preloaded with the BLE stack
Bonus Cube Tool!

STM32CubeMonitor-Power

$70

X-NUCLEO-LPM01A
CubeIDE & Atollic TrueSTUDIO

STM32CubeIDE is an all-in-one multi-OSE development tool, which is part of the STM32Cube software ecosystem.

STM32CubeIDE is an advanced C/C++ development platform with IP configuration, code generation, code compilation, and debug features for STM32 microcontrollers. It is based on the ECU Rapid™ CDT framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the ECU Rapid™ IDE.

STM32CubeIDE integrates all STM32CubeµX functionalities to offer all-in-one tool experience and save installation and development time. After the selection of an empty STM32 MCU or preconfigured microcontroller from the selection of a board, the project is created and initialization code generated. At any time during the development, the user can return to the initialization and configuration of the IPs or middleware and regenerate the initialization code with no impact on the user code.

STM32CubeIDE includes build and stack analyzers that provide the user with useful information about project status and memory requirements.

STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyzer.

Free feature-rich IDE For STM32 developers only
Iterative Design Process

Configure

Code & Debug

Measure

Test
STM32WB55RGV6 (VQFPN68)

Arduino & Morpho Headers

2.4GHz PCB antenna

Buttons & LED's

ST-Link/V2-1
We will use this one!
Hands-On
Out-of-the-Box
Hands-On: Out-of-the-Box

GATT Server

GATT Client
- Power Nucleo board
- Launch ST BLE Sensor app
- What happens?
• Launch LightBlue Explorer app

• We can filter by RSSI to find our device (on iOS)

Note that either the “Device Name” or the “Local Name” may show up on the iOS LightBlue App.
• Click on the “0x0000FE41-” characteristic

• Write new value – 101 hex

• Did the LED come on?

• 100 hex to turn it off
• Click on the “0x0000FE41” characteristic
• Write new value – 101 hex
• Did the LED come on?
• 100 hex to turn it off

Hands-On: Out-of-the-Box
Hands-On
CubeMX
Launch CubeMX
Launch CubeMX & Start project from Board Selector
Filter by STM32WB and double-click on the Nucleo board!
Yes
• Name your project

• Recommended Project location: C:\STM32WB_Workshop\n
• Use EWARM V8 toolchain
**P-NUCLEO-WB55 Board Project**

**GENERATE CODE**

![Image of the STM32CubeMX software interface showing the GENERATE CODE button highlighted]

### Project Settings
- **Project Name**: WB_Tester
- **Project Location**: C:\Users\alice\Documents\STM32\STM32WB55CubeMX\WB
- **Application Structure**: Basic
- **Toolchain / IDE**: IAR EWARM V8
- **Linker Settings**: Minimum Heap Size 64kB, Minimum Stack Size 64kB

### Code Generator

### Advanced Settings
- **Mcu and Firmware Package**: STM32Cube_FW_WB_V1.0.0
- **Firmware Package Name and Version**: STM32Cube_FW_WB_V1.0.0
- **Use Default Firmware Location**: C:\Users\alice\Documents\STM32\STM32Cube\Repository\STM32Cube_FW_WB_V1.0.0
• Open Project
• Expand the **User** file tree and Open **main.c**

• Enable line numbers in IAR (Tools > Options > Editor > Show line numbers)
Add some code to while(1) loop:

You can copy/paste the code bits from LED_Blinky_Lab.txt file from your Labs folder

```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
  HAL_Delay(100);
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
  HAL_Delay(100);
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1);
  HAL_Delay(100);

  /* USER CODE END WHILE */
}
Build the project

Check for errors

Total number of errors: 0
Total number of warnings: 0
Download & Debug (attach your board) 😊
GO!
Enjoy the dancing LED’s! 😊
Stop Debugging at the end of each lab and close IAR Embedded Workbench
BLE Fundamentals
BLE Protocol Stack layers

- **M4 domain**
- **Mailbox system**
- **M0+ domain**
- **Radio domain**

**Application**
- Profiles
- Services
- ACI Interface

**BLE peripheral**
- GAP
- GATT
- ATT
- SM
- L2CAP
- Host Control Interface
- Link Layer
- Radio PHY

**ARM CM4**
- SRAM2
- IPCC
- HSEM

**ARM CM0+**
- BLE STACK

**Radio PHY**
<table>
<thead>
<tr>
<th>Comparison of Classic and Low Energy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classic (BR/EDR)</strong></td>
<td><strong>Low Energy (LE)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Application</strong></td>
<td>Cell phones, headsets, stereo/audio streaming, automotive (handsfree), PCs, etc.</td>
<td>Smartwatches, sport &amp; fitness, home electronics, automation, industry, healthcare, smartphones, etc.</td>
</tr>
<tr>
<td><strong>Voice</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>RF band ISM</strong></td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td><strong>Energy consumption</strong></td>
<td>Reference</td>
<td>0.5…0.01 times Classic as reference</td>
</tr>
<tr>
<td><strong>Coverage</strong></td>
<td>≥ 10 m</td>
<td>≥ 10 m</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>3 classes (max.): +20 dBm, +4 dBm, 0 dBm</td>
<td>max. + 20 dBm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>four informative classes</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Inquiry</td>
<td>Advertising</td>
</tr>
<tr>
<td></td>
<td>Yes, always hopping</td>
<td>Connection only if necessary, then hopping</td>
</tr>
<tr>
<td><strong>Connection setup</strong></td>
<td>100 ms</td>
<td>6 ms</td>
</tr>
<tr>
<td><strong>RF channels</strong></td>
<td>79 with 1 MHz spacing</td>
<td>40 with 2 MHz spacing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 advertising</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 data (+ secondary advertising)</td>
</tr>
<tr>
<td><strong>Modulation</strong></td>
<td>GFSK</td>
<td>GFSK</td>
</tr>
<tr>
<td></td>
<td>• BT = 0.5</td>
<td>• BT = 0.5</td>
</tr>
<tr>
<td></td>
<td>• Deviation = 160 kHz</td>
<td>• Deviation = 250 kHz or 500 kHz</td>
</tr>
<tr>
<td></td>
<td>• Mod index = 0.28...0.35</td>
<td>• Mod index = 0.45...0.55</td>
</tr>
<tr>
<td></td>
<td>8DPSK</td>
<td>Slab Mod index = 0.495...0.505</td>
</tr>
<tr>
<td><strong>Gross data rate</strong></td>
<td>1...3 Mbit/s</td>
<td>1...2 Mbit/s</td>
</tr>
<tr>
<td><strong>Application data rate</strong></td>
<td>0.7...2.1 Mbit/s</td>
<td>0.2...0.6 Mbit/s</td>
</tr>
</tbody>
</table>

- **100X lower**
- **Longer range**
- **Fast connection (only 3 advertising channels to scan)**
- **Relaxed RF requirements (lower cost silicon / passives)**
Strategically placed advertising channels

Remaining 37 channels are data channels

<table>
<thead>
<tr>
<th></th>
<th>BLE</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulation</td>
<td>GFSK 0.45 to 0.55</td>
<td>GFSK 0.28 to 0.35</td>
</tr>
<tr>
<td>Data Rate</td>
<td>1 Mbit/s</td>
<td>1 Mbit/s</td>
</tr>
<tr>
<td>Channels</td>
<td>40</td>
<td>79</td>
</tr>
</tbody>
</table>
| Spacing          | 2 MHz             | 1 MHz              | -</p> </p>
• Link Layer (LL)
  • Radio control
  • Defines packet structure
  • One or more state machines
  • Link-layer-level encryption (via Security Manager)

• Host Control Interface (HCI)
  • Bridge between Radio Domain and M0+ Domain

• L2CAP (Logical Link Control and Adaptation Protocol)
  • Multiplex packets from higher-level protocols (ATT / SMP)
  • Handles segmentation and reassembly of packets
  • Quality of Service (QoS)
• **Link Layer (LL)**
  - Radio control
  - Defines packet structure
  - One or more state machines
  - Link-layer-level encryption (via Security Manager)

• **Host Control Interface (HCI)**
  - Bridge between Radio Domain and M0+ Domain

• **L2CAP (Logical Link Control and Adaptation Protocol)**
  - Multiplex packets from higher-level protocols (ATT / SMP)
  - Handles segmentation and reassembly of packets
  - Quality of Service (QoS)
• Link Layer (LL)
  • Radio control
  • Defines packet structure
  • One or more state machines
  • Link-layer-level encryption (via Security Manager)

• Host Control Interface (HCI)
  • Bridge between Radio Domain and M0+ Domain

• L2CAP (Logical Link Control and Adaptation Protocol)
  • Multiplex packets from higher-level protocols (ATT / SMP)
  • Handles segmentation and reassembly of packets
  • Quality of Service (QoS)
• **Standby** state: Sleep, Stop, Standby

• **Advertising** is the key to initiating all BLE communications!

• An **Initiator** and **Advertiser** negotiate a **Connection**

• In a Connection
  • The Link-Layer **Master** is also the GAP Central
  • The Link-Layer **Slave** is also the GAP Peripheral

*Figure 1.1: State diagram of the Link Layer state machine*
• **Standby** state: Sleep, Stop, Standby

• **Advertising** is the key to initiating all BLE communications!

• An **Initiator** and **Advertiser** negotiate a **Connection**

• In a Connection
  • The Link-Layer **Master** is also the GAP Central
  • The Link-Layer **Slave** is also the GAP Peripheral

![Link Layer State Machine](image)
• **Standby state:** Sleep, Stop, Standby

• **Advertising** is the key to initiating all BLE communications!

• **An Initiator** and **Advertiser** negotiate a **Connection**

• In a **Connection**
  • The Link-Layer **Master** is also the GAP **Central**
  • The Link-Layer **Slave** is also the GAP **Peripheral**

![Figure 1.1: State diagram of the Link Layer state machine](image)
• **Standby** state: Sleep, Stop, Standby

• **Advertising** is the key to initiating all BLE communications!

• As an **Initiator** and **Advertiser** negotiate a **Connection**

• In a **Connection**
  • The Link-Layer **Master** is also the GAP **Central**
  • The Link-Layer **Slave** is also the GAP **Peripheral**
Discovery: Advertising & Scanning

Advertiser Settings:
- Advertising Interval: 20mS

Scanner Settings:
- Scan Interval: 50mS
- Scan Window: 25mS

Advertise on Ch 37:
- 
Advertise on Ch 38:
- 
Advertise on Ch 39:
- 
Discovery Events:
- Scanning: Ch 37
- Scanning: Ch 38
- Scanning: Ch 39
GAP (Generic Access Profile)

Roles and Modes

• Advertising Mode
• Connected Mode
Roles and Modes

- Advertising Mode
- Connected Mode
GAP Central is also a “GATT Client”
GAP Peripheral” is also a “GATT Server”
GAP Central is also a “GATT Client”
GAP Peripheral” is also a “GATT Server”
GATT Central is also a “GATT Client”
GAP Peripheral” is also a “GATT Server”

- Central queries the Services available
  - Peripheral Services and Characteristics are exposed via its’ GATT database

What is my heartrate?

147 bpm

What is your Mfr ID?

Polar
SIG-defined profiles

- GAP Central
  - GATT Client
    - Collector
  - GAP Peripheral
    - GATT Server
      - Heart Rate Sensor
      - Heart Rate Service
      - Device Information Service
All signs point to blue

Introducing Bluetooth direction finding

Read More
GLP Profile defines two roles: Collector & Glucose Sensor
Once the GATT Server’s database information is known to the GATT Client, it can reference data via **Handles**

- **“What is the temperature reported by the Thermometer Service?”**  ATT read command of Handle 0x0102
- **“What are the units of temperature used?”**  ATT read command of Handle 0x0104

<table>
<thead>
<tr>
<th>Handle</th>
<th>UUID</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>0x2800</td>
<td>Thermometer service definition</td>
<td>UUID 0x1816</td>
</tr>
<tr>
<td>0x0101</td>
<td>0x2803</td>
<td>Characteristic: temperature</td>
<td>UUID 0x2A2B, Value handle: 0x0102</td>
</tr>
<tr>
<td>0x0102</td>
<td>0x2A2B</td>
<td>Temperature value</td>
<td>20 degrees</td>
</tr>
<tr>
<td>0x0104</td>
<td>0x2A1F</td>
<td>Descriptor: unit</td>
<td>Celsius</td>
</tr>
<tr>
<td>0x0105</td>
<td>0x2902</td>
<td>Client characteristic configuration descriptor</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0110</td>
<td>0x2803</td>
<td>Characteristic: date/time</td>
<td>UUID 0x2A08, Value handle: 0x0111</td>
</tr>
<tr>
<td>0x0111</td>
<td>0x2A08</td>
<td>Date/Time</td>
<td>1/1/1980 12:00</td>
</tr>
</tbody>
</table>
Attribute protocol details (ATT)

- **Access** GATT database information on the **Server**

- **Operations**
  - Read
  - Write / Write without response
  - Indicate / Notify

- **Four elements**
  - 16-bit **Handle**
  - **Type** of attribute (UUID)
  - **Value**
  - Attribute **Permissions** (Read-only, etc)

---

<table>
<thead>
<tr>
<th>Handle</th>
<th>UUID</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>0x2800</td>
<td>Thermometer service definition</td>
<td>UUID 0x1816</td>
</tr>
<tr>
<td>0x0101</td>
<td>0x2803</td>
<td>Characteristic: temperature</td>
<td>UUID 0x2A2B, Value handle: 0x0102</td>
</tr>
<tr>
<td>0x0102</td>
<td>0x2A2B</td>
<td>Temperature value</td>
<td>20 degrees</td>
</tr>
<tr>
<td>0x0104</td>
<td>0x2A1F</td>
<td>Descriptor: unit</td>
<td>Celsius</td>
</tr>
<tr>
<td>0x0105</td>
<td>0x2A62</td>
<td>Client characteristic configuration</td>
<td>Descriptor</td>
</tr>
<tr>
<td>0x0110</td>
<td>0x2803</td>
<td>Characteristic: date/time</td>
<td>UUID 0x2A0E, Value handle: 0x0111</td>
</tr>
<tr>
<td>0x0111</td>
<td>0x2A08</td>
<td>Date/Time</td>
<td>1/1/1980 12:00</td>
</tr>
</tbody>
</table>
Custom GATT – BLE Drill

- Battery Monitoring Service (BAS)
- Alert Notification Service (ANS)
- Elapsed motor use in minutes
- Unlock the drill via smartphone password
  - Add standard Services & Characteristics (16-bit UUID’s from Bluetooth SIG)
  - Create custom services (128-bit UUID’s)

Greg Hume [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], from Wikimedia Commons
• Universally Unique Identifiers (UUID’s) are simply 128-bit (16-byte) numbers:
  • 10c17863-9471-4427-8d66-82579bf9161a
  • Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  • To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
    • 0000XXXX-0000-1000-8000-00805F9B34FB
  • With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form
    • For example, the Glucose Service in our CGM profile is:
      • 00001808-0000-1000-8000-00805F9B34FB
  • Custom services / characteristics / descriptors need a fully defined 128-bit UUID
    • Our Custom Drill needs an Unlock service.
    • We can generate a random UUID for it at https://www.uuidgenerator.net/
• Universally Unique Identifiers (UUID’s) are simply 128-bit (16-byte) numbers:
  • 10c17863-9471-4427-8d66-82579bf9161a
  • Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed

• To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
  • 0000XXXX-0000-1000-8000-00805F9B34FB

• With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form
  • For example, the Glucose Service in our CGM profile is:
    • 00001808-0000-1000-8000-00805F9B34FB

• Custom services / characteristics / descriptors need a fully defined 128-bit UUID
  • Our Custom Drill needs an Unlock service.
  • We can generate a random UUID for it at https://www.uuidgenerator.net/

• There is also a 32-bit UUID specifier option
UUID’s: 16-bit or 128-bit

• Universally Unique Identifiers (UUID’s) are simply 128-bit (16-byte) numbers:
  
  • 10c17863-9471-4427-8d66-82579bf9161a
  
  • Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  
  • To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
  
  • 0000XXXX-0000-1000-8000-00805F9B34FB

• With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form

  • For example, the Glucose Service in our CGM profile is:

  • 00001808-0000-1000-8000-00805F9B34FB

• Custom services / characteristics / descriptors need a fully defined 128-bit UUID

  • Our Custom Drill needs an Unlock service.

  • We can generate a random UUID for it at https://www.uuidgenerator.net/

• There is also a 32-bit UUID specifier option
UUID’s: 16-bit or 128-bit

- Universally Unique Identifiers (UUID’s) are simply 128-bit (16-byte) numbers:
  - 10c17863-9471-4427-8d66-82579bf9161a
  - Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  - To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
    - 0000XXXX-0000-1000-8000-00805F9B34FB
    - With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form
      - For example, the Glucose Service in our CGM profile is:
        - 0001808-0000-1000-8000-00805F9B34FB

- Custom services / characteristics / descriptors need a fully defined 128-bit UUID
  - Our Custom Drill needs an Unlock service.
  - We can generate a random UUID for it at [https://www.uuidgenerator.net/](https://www.uuidgenerator.net/)

- There is also a 32-bit UUID specifier option
Custom Drill Profile

- **Standard Services & Characteristics (16-bit UUID’s)**
  - Battery service (BAS) UUID: 0x180F
    - **Battery Level Characteristic**: 0x2A19
  - Alert Notification Service UUID: 0x1811
    - **Alert Notification Control Point Characteristic**: 0x2A44
    - **Unread Alert Status Characteristic**: 0x2A45
    - **New Alert Characteristic**: 0x2A46
    - **Supported New Alert Category Characteristic**: 0x2A47
    - **Supported Unread Alert Category Characteristic**: 0x2A48

- **Create custom services (128-bit UUID’s)**
  - 10c17863-9471-4427-8d66-82579bf9161a (Motor run time service)
    - 55671a77-721f-4e1a-9875-7ae95ead642d xxx Characteristic
    - 3d78d6f3-7d34-4f89-a14d-ed3cac297438 xxx Characteristic
  - 0226b0db-d9a6-49c8-bce1-fcc3a40e6e2 (Unlock service)
    - 997e28a5-f05e-4027-89c7-e84ce4ce67ec xxx Characteristic
    - b3b7d2a1-4ebe-4a39-85ef-7dd7b1e4abf xxx Characteristic
• **Connection**: GAP Central connected to a GAP Peripheral  (Connection interval = 7.5ms to 4 secs)

• **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.

• **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.

• **Whitelisting**: Restrict connections from any other than known devices.
• **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)

• **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.

• **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.

• **Whitelisting**: Restrict connections from any other than known devices.
• **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)

• **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.

• **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.

• **Whitelisting**: Restrict connections from any other than known devices.
Security Topics

- **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)

- **Pairing**: Connected devices exchange encryption keys to encrypt the link. There are now paired.

- **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.

- **Whitelisting**: Restrict connections from any other than known devices.

- Security modes are deployed after a BLE connection is established

- BLE Link Layer uses AES-128 CCM mode for authenticated encryption
Typical attacks

- **Passive eavesdropping:**
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption

- **MITM**
  - A malicious device impersonates the other two legitimate devices

- **Identity tracking**
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this by periodically changing the device address.
Typical attacks

- Passive eavesdropping:
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption

- MITM
  - A malicious device impersonates the other two legitimate devices

- Identity tracking
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this by periodically changing the device address.

![Diagram of BLE stack and components](image)
Typical attacks

- **Passive eavesdropping:**
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption

- **MITM**
  - A malicious device impersonates the other two legitimate devices

- **Identity tracking**
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this is by periodically changing the device address.
Pairing Methods

• **Just Works™:**
  - Still vulnerable to MITM attack

• **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC

• **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability

• **Numeric Comparison:**
  - Similar to Just Works™, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks
Pairing Methods

• Just Works™:
  • Still vulnerable to MITM attack

• Out of Band (OOB) Pairing:
  • Keys exchanged over a different wireless technology such as NFC

• Passkey:
  • 6-digit number entered on each device
  • Assumes keypad capability

• Numeric Comparison:
  • Similar to Just Works™, but adds a 6-digit confirmation value
  • Additional protection from MITM attacks

Security Manager (SM)

BLE STACK

ARM CM4+ BLE STACK

GAP

ATT

L2CAP

Host Control Interface

Link Layer

Radio PHY

Application

ATT

SRAM2

IPCC

GATT

SM

ARM CM0+

Profiles

Services

Imp

ACI Interface

Application

SRAM2

IPCC

GATT

SM

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY

ARM CM0+

Radio PHY
**Pairing Methods**

- **Just Works™:**
  - Still vulnerable to MITM attack

- **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC

- **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability

- **Numeric Comparison:**
  - Similar to Just Works™, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks
Pairing Methods

• **Just Works™:**
  • Still vulnerable to MITM attack

• **Out of Band (OOB) Pairing:**
  • Keys exchanged over a different wireless technology such as NFC

• **Passkey:**
  • 6-digit number entered on each device
  • Assumes keypad capability

• **Numeric Comparison:**
  • Similar to Just Works™, but adds a 6-digit confirmation value
  • Additional protection from MITM attacks
A “White List” can optionally be used to filter device addresses

- **Advertising State** - An advertiser shall process connection requests only from devices in the White List
- **Scanning State** – A scanner shall process advertising packets only from White-Listed devices
- **Initiating State** - An initiator shall process connectable advertising packets only from White-Listed devices
BLE Sniffer

Filter Tabs isolate frames by profile or protocol for quick and convenient viewing of specific kinds of data.

Decode Pane shows comprehensive layered decoders of each frame/message with clear, concise descriptions.

Summary Pane displays a one line overview of each data frame/message. Click on any line to reveal detail in multiple panes below.
**Connection request**

**CONNECT_IND PDU**

- **Initiator Addr**: d6 be 89 8e
- **Access Addr**: 45 22 00 be 66 77 d7 59 8a 00 e1 fa 80 00
- **Length**: 0x22 = 36 bytes

**CRC Init**

- **Tx WindowSize**: 0x03 * 1.25 = 3.75mS
- **Conn Interval**: 0x0018 * 1.25 = 30mS
- **Conn Timeout**: 0x0048 * 10 = 900mS
- **Slave Latency**: 0x0000 = No ignored events

**Channel Map**

- All 37 data channels enabled

**CRC**

- **Hop / Sleep Clk Accuracy**: 8f 22 31
BLE 4.0 only but highly detailed

Up to 4.1, 5.0 not covered

Includes 5.0 & Mesh
# Core Specifications

The Bluetooth® Core Specification defines the technology building blocks that developers use to create the interoperable devices that make up the thriving Bluetooth ecosystem. The Bluetooth specification is overseen by the Bluetooth Special Interest Group (SIG) and is regularly updated and enhanced by Bluetooth SIG Working Groups to meet evolving technology and market needs.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Version</th>
<th>Status</th>
<th>Adoption Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>5.0</td>
<td>Active</td>
<td>06 Dec 2016</td>
</tr>
<tr>
<td>CSS</td>
<td>7</td>
<td>Active</td>
<td>06 Dec 2016</td>
</tr>
<tr>
<td>CSA</td>
<td>6</td>
<td>Active</td>
<td>12 Jul 2017</td>
</tr>
</tbody>
</table>
GATT Specifications

Generic Attributes (GATT) services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.

A GATT profile describes a use case, roles, and general behaviors based on the GATT functionality, enabling extensive innovation while maintaining full interoperability with other Bluetooth® devices.

The documents in the "Informative document showing changes" column are provided as a courtesy to help readers identify changes between two versions of a Bluetooth specification. When implementing specifications, use the adopted versions in the "Adopted Version" column.

More about GATT

<table>
<thead>
<tr>
<th>Profile Specification</th>
<th>Version</th>
<th>Status</th>
<th>Adoption Date</th>
<th>Informative document showing changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANP Alert Notification Profile</td>
<td>1.0</td>
<td>Active</td>
<td>13 Sep 2011</td>
<td>N/A</td>
</tr>
<tr>
<td>ANS Alert Notification Service</td>
<td>1.0</td>
<td>Active</td>
<td>13 Sep 2011</td>
<td>N/A</td>
</tr>
<tr>
<td>AIOP Automation IO Profile</td>
<td>1.0</td>
<td>Active</td>
<td>14 Jul 2016</td>
<td>N/A</td>
</tr>
<tr>
<td>AIOS Automation IO Service</td>
<td>1.0</td>
<td>Active</td>
<td>14 Jul 2016</td>
<td>N/A</td>
</tr>
<tr>
<td>BAS Battery Service</td>
<td>1.0</td>
<td>Active</td>
<td>27 Dec 2011</td>
<td>N/A</td>
</tr>
<tr>
<td>BCS Body Composition Service</td>
<td>1.0</td>
<td>Active</td>
<td>21 Oct 2016</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Hands-On
Heart-Rate Monitor
All CubeWB Projects referenced today can be found in the CubeMX Repository folder:
Open the BLE_HeartRate workspace
Open **app_ble.c**

Change the advertised **local name**, using your **Magic Number**!

*(You can change it as you wish, however keep the # of ASCII chars to 5)*

```c
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME ,'H','R','S','T','M'};

uint8_t manuf_data[14] = {
  // ...

static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME ,'S','T','M','1','2'};
```
Also change the BLE Device **name** and the **NAME_LENGTH**, using your **Magic Number**!

```c
177 /* Private defines ----------------------------------------
178 #define APPBLE_GAP_DEVICE_NAME_LENGTH 9
```

Build, Program and Launch debugger
• Open your LightBlue Explorer App on **iOS**

• Find your device and tap on it

• **Show** Advertisement Data

• Click on the Heart Rate Measurement and Enable Notifications
• Open your LightBlue Explorer App on **Android**

• Find your device and tap on it

• Tap on the Heart Rate section and select Heart Rate Measurement

• Tap on “SUBSCRIBE” to Enable Notifications
• Disconnect from the LightBlue Explorer App

• Launch the ST BLE Sensor App

• Tap on your device name

• Write down your Nucleo Bluetooth Device Address.
  ➢ *Can you find it in the Mfr-Specific advertised data via LightBlue app?*
WB Architecture
**Balun** – Combine TX and RX signals

**Matching Network** – 50 Ω impedance transformation

**Harmonic Filter** – Reduce out-of-band harmonics
Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics
RF System Front-End

Balun – Combine TX and RX signals

**Matching Network** – 50 Ω impedance transformation

**Harmonic Filter** – Reduce out-of-band harmonics
Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics
Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics
**Balun** – Combine TX and RX signals

**Matching Network** – 50 Ω impedance transformation

**Harmonic Filter** – Reduce out-of-band harmonics
Matching Network + Harmonic Filter

ST IPD Device

Matching Network

Filter

Discrete solution

IPD device from ST
Mass Production NOW

Versions coming for upcoming WLCSP / BGA packages

2.4 GHz low pass filter matched to STM32WB55Cx/Rx

Features
- Integrated impedance matching to STM32WB55Cx and STM32WB55Rx
- LGA footprint compatible
- 50 Ω nominal impedance on antenna side
- Deep rejection harmonics filter
- Low insertion loss
- Small footprint
- Low thickness ≤ 450 μm
- High RF performance
- RF BOM and area reduction
- ECOPACK®2 compliant

Applications
- Bluetooth 5
- OpenThread
- Zigbee®
- IEEE 802.15.4
- Optimized for STM32WB55Cx and STM32WB55Rx
1mm x 1.6mm CSP

PCB recommendations included in datasheet
• 3 autonomous sub-systems
  • Radio sub-system
  • Cortex-M0+ (CPU2)
  • Cortex-M4 (CPU1)

• Common run domain
  • Flash, SRAM2, RCC, PWR, EXTI
### Radio
- **Integrated balun**
- Output power: +6.0 dBm
- BLE RX sensitivity: -96 dBm
- 802.15.4 RX sensitivity: -100 dBm
- RX: 4.5mA
- TX: 5.2mA (0dBm)
- -40°C to +105°C

### Packages
- QFN48 / 68
- WLCSP100
- BGA129

### Control
- **Power supply**: 1.71V to 3.6V w/ DC/DC + LDO PDR/PDR/PV/D/BO
- **Crystal oscillators**: 32MHz (Radio) 32.768kHz (LSE)
- **Internal RC oscillators**: 32 kHz + 4 – 48 MHz + 16 MHz (HSI) + 48MHz +/- 1% acc. over V and T(°C)
- **RTC / AWU / CSS**
- **PLL / FLL**
- **SysTick timer**
- 2 watchdogs (WWDG / IWWDG)
- Up to 72 I/Os
- **Cyclic Redundancy Check**
- **Voltage scaling (2 modes)**

### ARM Cortex-M4 FPU/DPS 64MHz
- Nested Vector Interrupt Controller (NVIC)
- Memory Protected Unit (MPU)
- JTAG / SW debug
- **ART Accelerator™**
- **AHB Bus Matrix**
- 2x DMA 7channels

### Multi-Protocol Radio
- **Bluetooth 5™**
- IEEE 802.15.4
- AES

### ARM Cortex-M0+ MPU 32MHz
- Nested Vector Interrupt Controller (NVIC)
- **SW debug**

### Security
- AES 256-bit / PKA TRNG / PCROP

### Memory
- Up to 1MB Flash
- Up to 256KB SRAM
- **BOOT ROM**
- Secure boot loader

### Connectivity
- 2x SPI, 2x I²C
- 1x USART LIN, smartcard, IrDA, Modem control
- 1x ULP UART
- USB 2.0 FS – Crystal less
- Quad-SPI (XIP)
- SAI (Full duplex)

### Control
- 4x 16-bit 32-bit timers
- 2x ULP 16-bit timers

### Sensing
- 16-keys Capacitive touch

### Display
- 8x40 LCD driver
STM32WB55 Series Portfolio

Flash / RAM

1M / 256K
- STM32WB55CG
- STM32WB55RG
- STM32WB55VG

512K / 256K
- STM32WB55CE
- STM32WB55RE
- STM32WB55VE

256K / 128K
- STM32WB55CC
- STM32WB55RC
- STM32WB55VC

48 pins 7x7mm UQFN (p=0.5)
68 pins 8x8mm VQFN (p=0.4)
100 pins 4.39x4.37mm WLCSP (p=0.4)
129 pins 7x7mm BGA (p=0.5)
STM32WB55 Module

- ST Branded
- Pre-Certified
- Chip Antenna
- 10x10mm
- Large GPIO count
- Pin pitch = 2 layer PCB-ready
- Production in early 2020.
STM32WB35 – Block Diagram

256KB or 512KB Flash
96KB SRAM

- QFN48
- WLCSP47

Late 2019
Positioning

Flash / RAM Size (bytes)

1M / 256K
STM32WB55CG
STM32WB55VG

512K / 256K
STM32WB55CE
STM32WB55RE
STM32WB55VE

512K / 96K
STM32WB35CE
STM32WB35CE

256K / 128K
STM32WB55CC
STM32WB55RC
STM32WB55VC

256K / 96K
STM32WB35CC
STM32WB35CC

Pin count

47 pins WLCSP (p=0.5)
48 pins 7x7mm UQFN (p=0.5) PIN 2 PIN COMPATIBLE
68 pins 8x8mm VQFN (p=0.4)
100 pins 4.39x4.37mm WLCSP
129 pins 7x7mm BGA (p=0.5)*

* Exact part number on BGA 129 To be confirmed
STM32WB & BlueNRG Series

**BLUENRG-MS**
- ARM Cortex-M0 Core
- RX: 7.3mA
- TX: 8.2mA (0dBm)
- Sensitivity: -88dBm

**BLUENRG-1 / 2**
- ARM Cortex-M0 Core
- RX: 7.3mA
- TX: 8.2mA (0dBm)
- Sensitivity: -88dBm

**BLUENRG-LP**
- ARM Cortex-M0+ Core
- RX: 4.5mA
- TX: 5.2mA (0dBm)
- Sensitivity: -96dBm

**BLE4.1**
- RX: 7.3mA
- TX: 8.2mA (0dBm)
- Sensitivity: -88dBm

**BLE4.2**
- RX: 7.3mA
- TX: 8.2mA (0dBm)
- Sensitivity: -88dBm

**BLE5.0**
- RX: 4.5mA
- TX: 5.2mA (0dBm)
- Sensitivity: -96dBm

**BLE5.0 IEEE 802.15.4**
- RX: 4.5mA
- TX: 5.2mA (0dBm)
- Sensitivity: -96dBm

**NETWORK PROCESSOR**
- SINGLE-CORE
- APPLICATIONS PROCESSOR
- DUAL-CORE
• ARMv6-M architecture
• Von Neumann architecture
• 2-stage pipeline
• Single-issue architecture
• Single-cycle MULTIPLY

http://www.arm.com/products/processors/cortex-m/cortex-m0+-processor.php
ARM® Cortex® M4 Core

**FPU**
- Single precision
- Better code efficiency
- Eliminate scaling and saturation
- Support meta-language tools (MATLAB, etc)

**MCU**
- Easy C programming
- Interrupt handling
- Ultra-low power

**DSP**
- Harvard architecture
- Single-cycle MAC
- Barrel shifter

[Link to ARM Cortex M4 Processor](http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php)
Dual core – How does that work?

IPCC: Inter Processor Communication Controller

HSEM: Hardware Semaphore – prevent shared resource access conflicts

IPCC works in both directions
Memory Partitioning

- User Flash (1MB)
  - secure
  - non-secure

- System Flash
  - non-secure

- SRAM2b (32KB)
  - secure
  - non-secure

- SRAM2a (32KB)
  - secure
  - non-secure

- SRAM1 (192KB)
  - non-secure

- AES1
  - Key
- AES2
- PKA
- RNG

Securable by register bit
Memory Partitioning: BLE Stack

User Flash (1MB):
- Updater + Keys (40KB)
- Radio Stack (172KB)
- SFSA = 0x080CB000
- 796KB (199 pages)
- Empty
- BLE App
- HRM App = 16KB (4 pages)
- BLE App

SRAM2a (32KB):
- SNBRSA (0x14) = 0x2003D000
- Empty
- Empty

SRAM2b (32KB):
- SBRSA (0x0a) = 0x20032800
- Secure
- Non-secure
- Secure
- Non-secure

Addresses:
- 0x20030000 to 0x20037FFF
- 0x20038000 to 0x2003FFFF
- 0x20030000 to 0x2003D000
- 0x2003D000 to 0x2003FFFF
Memory Partitioning: BLE+Thread (Concurrent) Stack

User Flash (1MB)
- FUS + Keys + BLE/Thread
- Empty
- BLE_Thread_Static App = 32KB (8 pages)
- Radio Stack (500KB)
- SFSA = 0x08079000
- 484KB (121 pages)

SRAM2a (32KB)
- Secure
- SNBRSA = 0x20038000
- SFSA = 0x08079000

SRAM2b (32KB)
- Secure
- SBRSA = 0x20032800
- SFSA = 0x08079000

Updater + Keys (40KB)

1024KB (256 pages)
## Release Notes for STM32WB Copro Wireless Binaries

**Copyright © 2019 STMicroelectronics**

### License

This software component is licensed by ST under Ultimate Liberty license SLA0044, the “License”.

You may not use this file except in compliance with the License.

You may obtain a copy of the License at: SLA0044

### Purpose

This release covers the delivery of STM32WB Coprocessor binaries.

Here is the list of the supported binaries:

- `stm32wb5x_BLEStack_fw.bin`
  - Full BLE Stack 5.0 certified: Link Layer, HCI, L2CAP, ATT, SM, GAP and GATT database.
  - BT SIG Certification listing: Declaration ID D042164
- `stm32wb5x_BLE_HCI_Layer_fw.bin`
  - HCI Layer only mode 5.0 certified: Link Layer, HCI.
  - BT SIG Certification listing: Declaration ID D042213
- `stm32wb5x_Thread_FTD_fw.bin`
  - Full Thread Device certified v0.1.
  - To be used for Leader / Router / End Device Thread role (full features excepting Border Router).

For complete documentation on STM32WBx, visit: [www.st.com/stm32wb](http://www.st.com/stm32wb)

---

### Update History

**V1.0.0 / 06-February-2019**

**Main Changes**

**First release**

First official release.

**Binary Install Address and version**: Provides Install address for the targeted binary to be used in “STEP 4” of flash procedure.

<table>
<thead>
<tr>
<th>Wireless Processor Binary</th>
<th>Install address</th>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>stm32wb5x_BLEStack_fw.bin</td>
<td>0x000C0000</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_BLE_HCI_Layer_fw.bin</td>
<td>0x00800000</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_Thread_FTD_fw.bin</td>
<td>0x00809000</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_Thread_MTD_fw.bin</td>
<td>0x00809500</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_BLEThread_fw.bin</td>
<td>0x00807000</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_Mac_002_15_4_fw.bin</td>
<td>0x00805000</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
<tr>
<td>stm32wb5x_rmonitor_phy002_15_4_fw.bin</td>
<td>0x00805A00</td>
<td>v1.0.0</td>
<td>02/05/2019</td>
</tr>
</tbody>
</table>

---

### STM32WB_Copro_Wireless_Binaries

**Help**

- Share with
- Print
- New folder

---

![Release Notes.html](image)
# IoT Protection Ready

## Attacks

<table>
<thead>
<tr>
<th>Non Invasive</th>
<th>Attacks description</th>
<th>STM32WB Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>• Temp / Voltage / Clocks</td>
<td></td>
</tr>
<tr>
<td>Fault injection</td>
<td>• Exploit debugger</td>
<td></td>
</tr>
<tr>
<td>Side channel</td>
<td>• Power Analysis</td>
<td></td>
</tr>
<tr>
<td>• Temp sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Power supply monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Clock security system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Tamper pads</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• ECC, Parity check</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• SRAM mass erase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Read out protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Flash-only boot</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Software

| Break the encryption | • Extract keys |
| Exploit debugger / test modes | • Malware |
| • Replay |
| • Customer Key Storage |
| • RNG, Crypto accelerator, CRC |
| • Readout / Write memory protections |
| • Memory Protection Unit |
| • Root Security Service |
| • Secure Firmware Update (SFU) |
| • 96-bit Unique ID |
OTA Update Scenario

Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz

Network Processor
Cortex-M0+
32 MHz

2.4 GHz radio
Modem (BLE, 802.14.5)

Customer Key Storage
AES 128-bit

Radio stack
Empty Flash
FW
Application V 1.0

Closed Sub-system
Radio + Key storage
147

OTA Update Scenario

Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz

New FW package received

Radio + Key storage

Customer Key Storage

AES 128-bit

Closed Sub-system

Radio stack

FW Application V 2.0

Empty Flash

FW Application V 1.0

2.4 GHz radio

Modem (BLE, 802.14.5)

Network Processor
Cortex-M0+
32 MHz

Antenna
OTA Update Scenario

1. New FW package received
2. New FW detected
   Update is launched

Radio stack

Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz

Closed Sub-system
Radio + Key storage

2.4 GHz radio
Modem (BLE, 802.14.5)

Network Processor
Cortex-M0+
32 MHz

Customer Key Storage

AES 128-bit

Antenna

FW Application V 2.0

Empty Flash

FW Application V 1.0
OTA Update Scenario

1. New FW package received
2. New FW detected. Update is launched.
3. App Processor sends new FW package signature and encryption key for authentication.

Closed Sub-system
Radio + Key storage

Radio stack
FW Application V 2.0
Empty Flash
FW Application V 1.0

Antenna

Network Processor
Cortex-M0+
32 MHz

2.4 GHz radio
Modem (BLE, 802.14.5)

Customer Key Storage

AES 128-bit

Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz
OTA Update Scenario

1. New FW package received
2. New FW detected
   Update is launched
3. App Processor send New FW package signature and encryption key for authentication
   Authentication signature matches preprogrammed key
4. if not, the process is aborted and device resets

Radio stack

<table>
<thead>
<tr>
<th>FW Application V 2.0</th>
<th>Empty Flash</th>
<th>FW Application V 1.0</th>
</tr>
</thead>
</table>

Closed Sub-system
Radio + Key storage

Network Processor
Cortex-M0+
32 MHz

Antenna
2.4 GHz radio
Modern (BLE, 802.14.5)

Customer
Key Storage

AES 128-bit

Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz
OTA Update Scenario

1. New FW package received
2. New FW detected
   Update is launched
3. App Processor send New FW package signature and encryption key for authentication
   Authentication signature matches preprogrammed key if not, the process is aborted and device resets
4. New FW package is decrypted with proprietary Key.
OTA Update Scenario

1. New FW package received
2. New FW detected
3. Update is launched
4. App Processor send New FW package signature and encryption key for authentication
   Authentication signature matches preprogrammed key if not, the process is aborted and device resets
5. New FW package is decrypted with proprietary Key.
6. New Firmware replaces older firmware device resets.
AN5156 is a deep-dive into many security topics, some common and some WB-specific.
ART Accelerator™

- **Cortex-M4**
  - Instruction cache = 32 lines of 4x64 bits
  - Data cache = 8 lines of 4x64 bits
  - Pre-fetch buffer

- **Cortex-M0+**
  - Instruction cache = 4 lines of 1x64 bits
  - Data cache = 4 lines of 1x64 bits
  - Pre-fetch buffer
Power schemes

- VDDA
- VREF+
- VFBSMPS
- VDDSMPS
- VDD
- VLCD
- 2 COMP ADC
- VREF buffer
- Voltage Regulator
- SMPS
- CPU1, CPU2
- SRAM1,2
- Digital Peripherals
- Radio sub-system
- Flash
- Reset block
- PLL, HSI, MSI
- Standby circuitry
- USB transceivers
- Backup domain
- LSE, RTC, backup registers
- VDDRF
- VDDUSB
- VBAT
- VDD
- VFBSMPS
SMPS Schematic

OPEN: SMPS=ON
CLOSE: SMPS=OFF

STM32WBxx

VDDSMPS
VLXSMPS
VFBSMPS
VSSSMPS

VDD
4.7uF
C1
Solder
Jumper
SB1
L1 2.2uH
4.7uF
C2
Wurtz 74479774222
Murata : GRM155R61A475MEAA

VSSSMPS

8MHz SMPS configuration

For 4MHz SMPS configuration change $L1 = 4.7\mu H$
**FlexPowerControl**

- **High performance**
  - CoreMark score = 215
- **Outstanding power efficiency**
  - ULPBbench score = 175

**Wake-up time**

- **RUN (Range1) at 64 MHz**: 117 (73) µA / MHz**
- **SLEEP at 64 MHz**: 41 µA / MHz
- **STOP 2 (full retention)**: 1.8 µA / 2.2 µA*  
- **STANDBY + 32 KB RAM**: 320 nA / 600 nA*  
- **STANDBY**: 110 nA / 440 nA*  
- **SHUTDOWN**: 30 nA / 315 nA*  
- **VBAT**: 2 nA / 300 nA*

Typ @ VDD = 1.8 V @ 25 °C

* with RTC  
** from SRAM1

(..) SMPS mode

RF Operation

158
HSE (32MHz) required for radio operation

LSE (32.768KHz) required for most BLE applications
• BLE requires very accurate 32 MHz clock

• Frequency can vary
  • Manufacturing process variations
  • Crystal used
  • PCB design

• Integrated load capacitor bank
  • 64 values for fine tuning
  • MCO clock output pin used for measurement at factory test
  • Stored in OTP

• No need for external capacitance

• AN5042 provides details

AN5042 Application note
HSE trimming for RF applications using the STM32WB Series
Silicon Cost Savings

• Embedded RF balun

• Single IPD from ST

• Simple SMPS circuit

• Integrated HSE crystal tuning caps

• Minimal passives needed

• Simple 2 layer PCB design

Simplified Schematic Diagram
Batch Acquisition Mode (BAM)

Peripheral + DMA + SRAM1

Flash in Power-down mode

CortexM4 in Sleep mode

CortexM0+ in Sleep mode
Hands-On

CubeMonitorRF
• BLE commands
• OpenThread commands
• BLE & 802.15.4 RF tests
• COM-port based
We will run in BLE mode.
Open the **Transparent Mode** workspace
Build, Debug & Run on your Nucleo board
Program via USB debug port

Select device on relevant COM port

Connect to start communication
Program via USB debug port

Select device on relevant COM port

Connect to start communication
**Command Complete** signals successful communications
Click on the **Command Complete** line to get more information on the command sent.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Literal</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCI packet indicator</td>
<td>0x21</td>
<td>HCI M4 Event Packet</td>
</tr>
<tr>
<td>Event_Code</td>
<td>0x0E</td>
<td>Command Complete</td>
</tr>
<tr>
<td>Parameter_Total_Length</td>
<td>0x42</td>
<td></td>
</tr>
<tr>
<td>Num_HCI_Command_Packets</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>CommandOpcode</td>
<td>0xFD62</td>
<td>VS_HCI_CL_DEVICE_INFORMATION</td>
</tr>
<tr>
<td>Status</td>
<td>0x00</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Device Revision</td>
<td>0x2000</td>
<td></td>
</tr>
<tr>
<td>Device Code Id</td>
<td>0x0495</td>
<td></td>
</tr>
<tr>
<td>Device Package</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td>Device Type</td>
<td>0x25</td>
<td></td>
</tr>
<tr>
<td>Device Company</td>
<td>0x000080E1</td>
<td></td>
</tr>
<tr>
<td>UID64</td>
<td>0x00000007A5</td>
<td></td>
</tr>
<tr>
<td>Device ULD96</td>
<td>0x203430...</td>
<td></td>
</tr>
<tr>
<td>Safe Boot Information</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>Res Information</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW version</td>
<td>0x00002000</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW mem</td>
<td>0x16000020</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW Thres</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW, BLE</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>CM4 FW Information</td>
<td>0x00000100</td>
<td></td>
</tr>
</tbody>
</table>
### Command Details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Literal</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCI Packet Indicator</td>
<td>0x21</td>
<td>HCI M4 Event Packet</td>
</tr>
<tr>
<td>Event Code</td>
<td>0x0E</td>
<td>Command Complete</td>
</tr>
<tr>
<td>Parameter_Total_Length</td>
<td>0x42</td>
<td></td>
</tr>
<tr>
<td>Num_HC1_Command_Packets</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>CommandOpcode</td>
<td>0xF062</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>0x00</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Device Revision</td>
<td>0x2009</td>
<td></td>
</tr>
<tr>
<td>Device Code Id</td>
<td>0x0495</td>
<td></td>
</tr>
<tr>
<td>Device Package</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td>Device Type</td>
<td>0x25</td>
<td></td>
</tr>
<tr>
<td>Device Company</td>
<td>0x0000080E1</td>
<td></td>
</tr>
<tr>
<td>UUID4</td>
<td>0x000007A5</td>
<td></td>
</tr>
<tr>
<td>Device UUID6</td>
<td>0x02054308530365066030000048</td>
<td></td>
</tr>
<tr>
<td>Safe Boot Information</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>Rss Information</td>
<td>0x000000000000000</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW version</td>
<td>0x00020002</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW memory size</td>
<td>0x16000002C</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW, Thread</td>
<td>0x000000000000000</td>
<td></td>
</tr>
<tr>
<td>CM0 and Wireless FW, BLE</td>
<td>0x000000000000000</td>
<td></td>
</tr>
<tr>
<td>CM4 FW Information</td>
<td>0x00000100</td>
<td></td>
</tr>
</tbody>
</table>
Lots of categories to choose and filter from

Command categories:
- HCI
- HCI test
- HAL
- GAP
- GATT
- L2CAP
Open and Edit the HR_Init_GAP_GATT.txt Script file

(In your installation zip file, **Scripts** folder)
Set the Bluetooth Address

Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x112233445566)

Modify this value as you wish

Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x112233445566)
• Change the two characters of the Local Name with your Magic number (e.g. change 0x4257 to 0x3130 for magic number “01”).

```
Send(ACI_GAP_SET_DISCOVERABLE;0x00;0x0080;0x00A0;0x00;0x00;0x08;0x425732334D545309;0x03;0x180D02;0x0000;0x0000)
#0x42 57 32 33 4D 54 53 09
# 0x09 - Local name
# 0x53 - "S"
# 0x54 - "T"
# 0x4D - "M"
# 0x33 - "3"
# 0x32 - "2"
# 0x57 - "W"
# 0x42 - "B"
```

* Note Little-Endian byte ordering

### ASCII Character Set for Magic numbers

<table>
<thead>
<tr>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>38</td>
<td>8</td>
</tr>
<tr>
<td>39</td>
<td>9</td>
</tr>
</tbody>
</table>

7 ASCII chars + 0x09 = 0x08. To add characters, also change the LENGTH parameter (x+1)
Save and Load your script

Start Script
Find your device

Show ADV data

Enable Notifications
Now load the `HR_Update_Char.txt` script to send Notification Updates.
Application Note AN5270 describes the ACI/HCI commands available

- via CubeMonitorRF
- via Application API’s

### 2.3.2 ACI_HAL_WRITE_CONFIG_DATA

**Description**

This command writes a value to a low level configure data structure. It is useful to setup directly some low level parameters for the system in the runtime.

**Input parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Size</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>1</td>
<td>Offset of the element in the configuration data structure which has to be written. The valid offsets are:</td>
<td>• 0x00: CONFIG_DATA_PUBADDR_OFFSET • 0x06: CONFIG_DATA_DIV_OFFSET • 0x08: CONFIG_DATA_EER_OFFSET • 0x18: CONFIG_DATA_IP_OFFSET • 0x2C: LL_WITHOUT_HOST • 0x3E: CONFIG_DATA_RANDOM_ADDRESS_WR • 0x2F: CONFIG_DATA_WATCHDOG_DISABLE</td>
</tr>
<tr>
<td>Length</td>
<td>1</td>
<td>Length of data to be written</td>
<td>-</td>
</tr>
<tr>
<td>Value</td>
<td>Length</td>
<td>Data to be written</td>
<td>-</td>
</tr>
</tbody>
</table>
Lunchtime!
The USB Dongle is quite useful as the CubeMonitorRF sniffer.

This project uses the USB CDC class directly (not the STLINK VCOM port) to parse commands.

Although there is no STLINK on board, the USB bootloader can be invoked via **BOOT0 switch** & CubeProgrammer, and the binary can be programmed.
• Move Dongle Switch to Bootloader mode

• Plug in Dongle

• Ensure the driver has enumerated “STM32 Bootloader”
Chapter 1.2.4 details the DFU driver install / update procedure

Old or Native MS drivers must be replaced to properly access the bootloader
Open STM32 CubeProgrammer

Select USB mode and Connect
Open the BLE_TransparentModeVCP_reference.hex file for Dongle
• Disconnect from CubeProgrammer

• Unplug Dongle

• Move Dongle Switch back to normal boot mode

• Plug Dongle back in for normal boot startup

• Now you should be able to use COMxx in CubeMonitorRF
  • (may differ from COM74)

• CONNECT
- Change the Bluetooth Address and Name. (Use your Magic Number!)
- Use Connectable advertising on all channels (37/38/39)
- Use LightBlue Explorer to connect to and interrogate your GAP peripheral.
Hands-On
Custom GATT & Cable Replacement
Open, Compile, Program & Run the **Nucleo** Board **CableReplacement** example

Add a custom GATT Characteristic for LED control

You can copy/paste the code bits from **CableReplacement_Lab.txt** file from your install files **Labs** folder

To add line numbers in IAR:
• Open the workspace
• Build the Project

• Open the following files
  - app_conf.h
  - app_ble.c
  - ble_conf.h
  - crs_stm.h
  - crs_stm.c
  - crs_app.c

To see the header files, expand the C source files.
• STM32WB is the GAP Peripheral / GATT server

• Smartphone is the GAP Central / GATT client.

Compile for GATT Server
  • Modify the #define (line# 100 of app_conf.h)

```
#define GATT_CLIENT 0 /* 1 = Device is GATT Client, 0 = Device is GATT Server */
```
Identify your unique device with your magic number

- Modify your local name (line# 204 of app_ble.c)

  ```
  static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME, 'C', 'R', 'S', '0', '1' };
  ```

- Modify your BLE device name (line# 822 of app_ble.c)

  ```
  const char *name = "BLEcore";
  ```

- Ensure that the BLE device name length in ASCII chars matches (line# 165 of app_ble.c)

  ```
  #define APPBLE_GAP_DEVICE_NAME_LENGTH 7
  ```
• Build and Run the project

• Connect your TeraTerm to the Nucleo’s STLink Virtual COM port
Configure your Serial port for 115,200bps / N / 8 / 1
Cable Replacement Test

- Connect to your device with LightBlue Explorer
- Send and receive ASCII-based messages using the different characteristics

Enable Notifications to receive messages from Nucleo

Write values to Nucleo
Here is the LightBlue Explorer on Android

Enable Notifications to receive messages from Nucleo

Write values to Nucleo
Add a custom characteristic to an existing Service

• Add the UUID definition (line# 74 of ble_conf.h)

```c
#define STM_LED_UUID128 0x00, 0x00, 0xfe, 0x64, 0x8e, 0x22, 0x45, 0x41, 0x9d, 0x4c, 0x21, 0xed, 0xae, 0x82, 0xed, 0x19
```

• Add event element (line# 37 of crs_stm.h)

```c
typedef enum {
    STM_LED_WRITE_EVT,
    CRS_NOTIFY_ENABLED_EVT,
    ...
} CRS_Opcode_evt_t;
```

Note that crs_stm.h is read-only, you will need to change permissions.
• From IAR, right click on the file tab and select “File Properties”

• Uncheck the “Read-only” box

• Click OK
• **Add characteristic handle (line# 32 of crs_stm.c)**

```c
typedef struct {
    ...
    uint16_t CRSRXCharHdle;
    uint16_t LedWriteClientToServerCharHdle;
} CRSContext_t;
```

• **Check for the handle (line# 122 of crs_stm.c)**

```c
case EVT_BLUE_GATT_ATTRIBUTE_MODIFIED:
{
    attribute_modified = (aci_gatt_attribute_modified_event_rp0*)blue_evt->data;
    if(attribute_modified->Attr_Handle == (CRSContext.LedWriteClientToServerCharHdle + 1))
    {
        Notification.CRS_Evt_Opcode = STM_LED_WRITE_EVT;
        Notification.DataTransfered.Length = attribute_modified->Attr_Data_Length;
        Notification.DataTransfered.pPayload = attribute_modified->Attr_Data;
        CRSAPP_Notification(&Notification);
    }
```
• Add uuid array (line# 193 of crs_stm.c)

```c
uint8_t led_uuid[] = { STM_LED_UUID128 };
```

• Change the Max_Attribute_Records parameter (line# 215 of crs_stm.c)

```c
hciCmdResult = aci_gatt_add_service(
    UUID_TYPE_128,
    (Service_UUID_t *) &uuid,
    PRIMARY_SERVICE,
    8,
    &(CRSContext.SvcHdle));
```
• Add LED characteristic (line# 281 of crs_stm.c)

COPY_CRS_UUID(uuid.Char_UUID_128, led_uuid);

hciCmdResult = aci_gatt_add_char(CRSContext.SvcHdle,
    UUID_TYPE_128,
    &uuid,
    2, /* Char_Value_Length */
    CHAR_PROP_WRITE_WITHOUT_RESP,
    ATTR_PERMISSION_NONE,
    GATT_NOTIFY_ATTRIBUTE_WRITE, /* gattEvtMask */
    10, /* encryKeySize */
    1, /* isVariable */
    &(CRSContext.LedWriteClientToServerCharHdle));
Add event action (line# 194 of crs_app.c)

case STM_LED_WRITE_EVT:
    if(pNotification->DataTransfered.pPayload[0] == 0x01)
    {
        BSP_LED_On(LED_BLUE);
    }
    if(pNotification->DataTransfered.pPayload[0] == 0x00)
    {
        BSP_LED_Off(LED_BLUE);
    }
    break;
• Launch the LightBlue app

• Find your device

• Select your device

Note that Android displays the Local Name.
Find your LED characteristic UUID

Write new value

LightBlue App
Write a value

• LED ON = 1
• LED OFF = 0

Concurrently, the CableReplacement characteristics can also be used
Delivery State

FUS + BLE Stack

FUS + BLE Stack

FUS only

Stack must be loaded

48-pin UQFN (0.5 mm pitch) 68-pin VQFN (0.4 mm pitch)
Nucleo Hardware Config for Bootloader access

Only **USB-DFU or USART1** (on PA9/PA10 only) bootloader modes available for secure stack loading!

- **BOOT0 pin HIGH**: CN7-5 to CN7-7
- **USART1 on PA9/10 to ST-LINK**
  - This is not the native USART connection to the STLINK!
  - STL-RX to CN10-19 (PA9)
  - STL-TX to CN10-31 (PA10)
- **USB User connector + Power Jumper JP2 to 5-6**
As we have seen, configuring the Dongle board for USB-DFU is quite easy.

Move the switch and repower the board to the right.
AN5185: Firmware Update Services

AN5185 details the sequence to create your own secure stack loader project, running on the M4

- Command / Response HCI event transactions to the M0+ similar to BLE

Also details on the bootloader sequences used
AN5165 details RF hardware considerations

- PCB stackup recommendations
- RF Front-end (discrete or IPD-based)
- SMPS passives selection
- Clocks

2-layer PCB

With the 2-layer PCB (see Figure 24), the RF signals and routing are on the top layer while the GND layer is used for grounding under the RF traces, and for routing in common planes. The ground plane must be connected under the RF trace, otherwise the return path current can generate and degrade the RF performance.

4-layer PCB

With the 4-layer PCB (see Figure 25), it is recommended to have the following definitions:
- TOP layer: RF signal and routing on the top layer
- VHF/RF layer: routing under the RF traces, routing in the others plane
- Ground layer: power and low frequency routing
- BOTTOM layer: high frequency routing

UFQFN48/VFQFN68 reference boards with IPD

This test STH IPD (integrated passive devices) is to increase the desirable matching network plus the implemented on pass filter keeping equivalent TIAA performance. Figure 41 shows the difference between the two approaches.

Layout recommendations for the 2-layer PCB

Figure 22. PCB layout for the 2-layer PCB (left to right: top and bottom layers)

Figure 23. PCB layout for the 4-layer PCB (left to right: top and bottom layers)

Figure 24. PCB layout with discrete matching network (left) and with IPD (right)
AN5290 details the minimal Bill-of-Materials needed for various scenarios.
AN5129 details a “meander-style” PCB antenna design
AN5246: SMPS

AN5246 details SMPS use cases, component selection, and various typical operating parametrics.

**Introduction**

This document describes how to use the SMPS (Switched Mode Power Supply) integrated in microcontrollers of the STM32WB Series. It is intended to be used by system architects and by HW and board-level SW developers.

The patented implementation detailed in this document differs from the standard ones because it is able to maintain the RF transceiver full performance while, at the same time, providing the best power figure in burst application like those generally used by Bluetooth® Low Energy and IEEE 802.15.4 protocols.

**Figure 2. Load impact on VSTOPSMPS**

**Figure 4. Typical transient current at power ON**
AN5071 details the multitude of low-power options available on the WB

![Diagram of low-power modes](image)

![Graph of current consumption](image)

![Power distribution architecture](image)

**Table 2. STM32WB55 performance with SMPS**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>mA/MHz</th>
<th>CoreMark(^e) per MHz</th>
<th>CoreMark(^e) per mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH ART On</td>
<td>0.077</td>
<td>3.25</td>
<td>42</td>
</tr>
<tr>
<td>SRAM1</td>
<td>0.073</td>
<td>2.40</td>
<td>33</td>
</tr>
</tbody>
</table>
AN5155 is an exhaustive list of all firmware examples and descriptions

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Project Name</th>
<th>Description</th>
<th>ADC</th>
<th>CubeMx</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>ADCInitAppWatchdog_init</td>
<td>How to use an ADC peripheral with an ADC exciting watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADCContinuousConversion_SingleChannel</td>
<td>How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>ADC</td>
<td>ADCContinuousConversion_SingleChannel_multiple_samples</td>
<td>How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_Driver_regular/irregular</td>
<td>How to use an ADC peripheral with both ADC groups (regular and irregular) in their intended use cases.</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_Oversampling</td>
<td>How to use an ADC peripheral with ADC oversampling.</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_SingleConversion_SingleChannel_DMA</td>
<td>How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. Example uses the DMA programming model (for polling or interrupt programming models, refer to other examples).</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_SingleConversion_SingleChannel_DMA_I2C</td>
<td>How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. Example uses the DMA programming model (for polling or interrupt programming models, please refer to other examples).</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_SingleConversion_SingleChannel_DMA_UART</td>
<td>How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. Example uses the DMA programming model (for polling or interrupt programming models, please refer to other examples).</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_SingleConversion_SingleChannel_DMA_USART</td>
<td>How to use an ADC peripheral to perform a single ADC conversion on a channel, at each trigger event from a timer. Converted data are automatically transferred by DMA into a table (DDR3 memory).</td>
<td>-</td>
<td>CubeMx</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC_TemperatureSensor</td>
<td>How to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in Celsius degrees.</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

**CubeMx** denotes that there is an “ioc” CubeMX project file also.
AN5292 shows how to get started using BLE Mesh

Figure 7: Internal project folder

- NUCLEO-WB55Nucleo
- BLE Mesh Lighting Demo
- Core
- EWARM
- STM32_WPAN
- BLE.MeshLightingDemo
- Applications
  - NRF
  - BLE_Thread
  - FreeRTOS
  - Mac_802.15_4
  - Thread
  - USB_Device

STM32WBx5 setup source and include file

IAR project workspace

Application source and include files

Mailbox source file for communication between M3 and M4

Figure 14: Read command from a remote node

Figure 16: VCOM window

Table 1: MAC address management

<table>
<thead>
<tr>
<th>Number</th>
<th>MAC address Management</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Using external MAC address</td>
<td>User can program the nodes with desired unique MAC address. This is stored at specific location in the flash. It is the user’s responsibility to make sure that the programmed MAC address is the device is compliant with the Bluetooth communication requirements.</td>
</tr>
<tr>
<td>2</td>
<td>Using the unique device serial number</td>
<td>It is possible to configure the MAC address of the devices using the unique serial number available in each device. This is in the default setting.</td>
</tr>
<tr>
<td>3</td>
<td>Using static random MAC address</td>
<td>It is possible to configure the MAC address of devices using the static random MAC address.</td>
</tr>
</tbody>
</table>
Hands-On

Over-the-Air Firmware Update
Over-The-Air Firmware Updates

User Flash (1MB)

- Updater + Keys (40KB)
- Radio Stack (172KB)
- User Flash (1MB)
- SFSA = 0x080CB000

- BLE Secure Stack
- Empty
- 768KB (192 pages)

- BLE+OTA App
- OTA Loader App
- 1MB = 1024KB (256 pages)

- OR
- HRM App = ~16KB (4 pages)
- BLE_Ota
- BLE_HeartRate_ota
- BLE_Hot
- BLE_CableReplacement
- BLE_DataThroughput
- BLE_BloodPressure
- BLE_HeartRateFreeRTOS
- BLE_BodyTemp
- BLE_Sensor
- BLE_Bluetooth
- BLE_OtaServer

- OR
- 28KB (7 pages)
- BLE_OtaServer_ota
- BLE_Proximity
- BLE_TransparentMode

- OR
- 212KB (53 pages)
- BLE_Ota
- BLE_HeartRate_ota

- OR
- 768KB (192 pages)
- SFSA = 0x080CB000

- 1MB = 1024KB (256 pages)
- OTA Loader App
- 0x08007000
- 0x08000000
Flash Nucleo board with **BLE_Ota_reference.hex** using CubeProgrammer
Load and personalize your **BLE_p2pServer_ota.eww** project

In `app_ble.c`

```c
240  #if (P2P_SERVER1 != 0)
241  static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME, 'F', '2', 'F', '5', 'R', '1', '2' };
242  uint8_t manuf_data[14] = {

178  #define APPBLE_GAP_DEVICE_NAME_LENGTH 9

772  const char *name = "STM32WB12";
```
Flash your newly created **BLE_p2pServerOTA.bin** to 0x08007000
Verify functionality on the ST BLE Sensor app

You should also see OTA capability

Once seen, disconnect from your device
Connect the Dongle and select **OTA Updater**

Search and Select your Device
(you can see your local name & BLE Address)
Browse for the other OTA binary
• BLE_HeartRate_ota_reference.bin

Update image

Repository → STM32Cube_FW_WB_V1.0.0 → Projects → NUCLEO-WB55_Nucleo → Applications → BLE → BLE_HeartRate_ota → Binary

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE_HeartRate_ota_reference.bin</td>
<td>2/19/2019 9:56 AM</td>
<td>FTE Binary Export...</td>
<td>15 KB</td>
</tr>
</tbody>
</table>
OTA capability detected
Click to start

Erase & Reboot

Select Smartphone file (iCloud, etc)

Flashing begins

Update via ST BLE Sensor app

65 bpm
Energy: 20 kJ
RR Interval: 1.00 s

STM32Cube_FW_WB-OTA
Version: 1.0.0
Mcu type: STM32WBXX
Address: 0x 7000

BLE_HeartRate
2/19/19
10 KB

BLE_p2pServer
2/19/19
32 KB

2 items, 26.05 GB available on iCloud

STM32Cube_FW_WB-OTA
Version: 1.0.0
Mcu type: STM32WBXX
Address: 0x 7000

Flashing the new firmware
1860/31996 bytes
AN5247 details the OTA application in further detail.
BlueST SDK on Github:  [https://github.com/STMicroelectronics/BlueSTSDK_Android](https://github.com/STMicroelectronics/BlueSTSDK_Android)
Bonus Lab! SIG RF Certification Tests

Packet Error Rate (PER) tests with two boards connected

Connect a second board (Dongle for instance)
Packet whitening is disabled in Direct Test Mode (DTM)

**Test mode** → **Packet Error Rate (PER)** → **COM222**

**Configure tester (COM222)**

<table>
<thead>
<tr>
<th>PA Level</th>
<th>31 (+5dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX Frequency</td>
<td>2402 MHz (Channel 37)</td>
</tr>
<tr>
<td>Length of Data</td>
<td>6x25</td>
</tr>
</tbody>
</table>

**Packet Payload**

- 0x00 - Pseudo-Random bit sequence 9
- 0x01 - Pattern of alternating bits '11110000'
- 0x02 - Pattern of alternating bits '10101010'
- 0x03 - Pseudo-Random bit sequence 15
- 0x04 - Pattern of all '1' bits
- 0x05 - Pattern of all '0' bits
- 0x06 - Pattern of alternating bits '00001111'
- 0x07 - Pattern of alternating bits '01010101'

** ACI Commands**

- **Scripts**
- **Beacon**

**ACI Utilities**

- **Power Transmission testing**
- **Frequency Deviation testing**
- **Carrier Frequency Accuracy testing**

**Arm CM4**

- Application
- Profiles
- Services
- ACI Interface
- SRAM2
- IPCC

**Arm CM0+**

- BLE peripheral
  - GAP
  - GATT
  - ATT
  - SM
  - L2CAP
  - Host Control Interface
  - Link Layer
  - Radio PHY (DTM)

**BLE Stack**

- **Peripheral**
- **Host**
- **Application**
  - **Profiles**
    - **Services**
  - **ACI Interface**
  - **SRAM2**
  - **IPCC**

**ARM Cortex-M4**

- **Radio PHY**
- **DTM**

**ARM Cortex-M0+**

- **BLE Stack**
- **Link Layer**
- **Radio PHY**
- **DTM**
We Greatly Value Your Feedback

Use your phone to scan the QR code or type the link into your browser.

https://www.surveymonkey.com/r/PLYWMDC

Thank you!
Releasing Your Creativity

www.st.com/stm32wb