AN14757

MCX W23 Health Care IoT Peripheral Software Architecture

Rev. 1.0 — 5 August 2025 Application note

Document information

Information	Content
Keywords	AN14757, MCXW23, MCX W23, Health care, BOD, shelf, privileged, TrustZone, NHS5204, Ultra-low power, Small footprint, Bluetooth Low Energy, Integrated flash, Security, IoT, Coin battery, Small Body-worn device, Apllication sotfware, Registers, Health care, Peripheral
Abstract	This document introduces the software architecture for the MCX W23 Health care IoT Peripheral application.

1 Introduction

This document provides an overview of the software architecture for the MCX W23 Health care IoT Peripheral application. Designed as a model implementation, this application showcases the key features of the MCX W23 platform and serves as a foundation for developing product-quality applications. Its modular architecture enables developers to easily adapt and extend components to meet specific project requirements.

The Health care IoT Peripheral application operates in the Bluetooth Low Energy peripheral role, allowing it to be discovered and connected to by a central device, such as a smartphone.

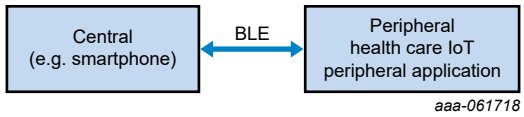

Figure 1. Bluetooth Low Energy Peripheral role

When activated, the application performs the following tasks:

- Measures ambient temperature.
- Processes and stores measurements securely in encrypted flash memory.
- Enables a central Bluetooth Low Energy device to access real-time temperature and battery data via the connectivity service.
- Ensures operational safety through a dedicated safety service that:
  - Configures security features at startup.
  - Monitors battery health and execution flow using watchdogs.
  - Logs critical events to flash memory.

The system runs on FreeRTOS, which manages concurrent tasks while meeting real-time constraints. Trusted Firmware-M (TF-M) ensures secure execution by isolating services in protected environments.

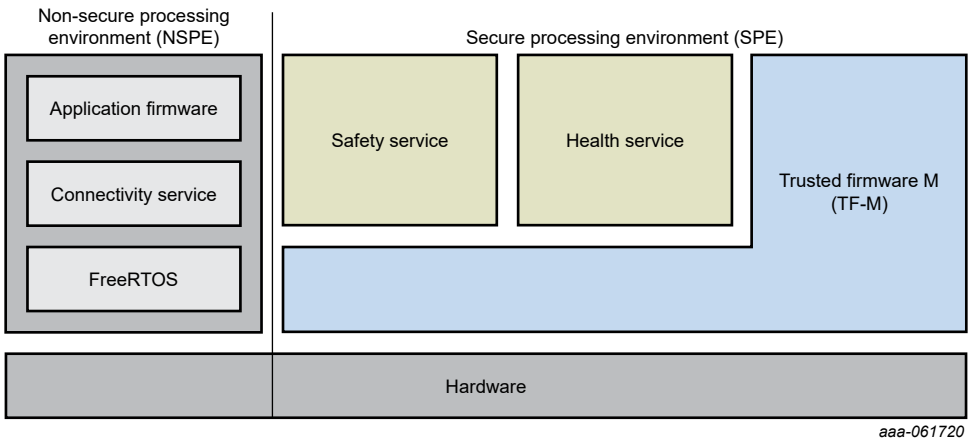

Figure 2. Collaboration diagram

1.1 Features

The main features of the Health care IoT Peripheral application are:

- Device lifecycle:
  - Shelf state: Low-power mode during storage and transport
  - Active state: The device performs measurements and communicates via Bluetooth Low Energy

  - End of life state: Triggered when battery levels are critically low
- Temperature monitoring
  - Configurable measurement interval
  - Encrypted data storage on internal or external flash
- Wireless connectivity using Bluetooth Low Energy
  - Bluetooth battery, temperature, and current time service supported
  - Includes pairing and bonding
- Security
  - Trusted firmware M (TF-M) to separate and isolate the different services
- Safety
  - Independent safety service that detects and mitigates safety issues
  - Detection and recovery from erroneous conditions with the help of BODs, watchdogs, and so on
- Power management to reduce power consumption and avoid current peaks in all states
- Support for both operating system based and bare-metal scheduling.
- An operating abstraction layer (OSAL) allows easy replacement of the used operating system.
- Modular architecture facilitates reuse and customization of components, such as replacing the temperature service with another health-related service.

The above sketches the features of the final application. The version included in the SDK is a work in progress and has the following limitations:

- End of life state not implemented.
- Temperature measurements stored unencrypted in flash.
- Battery service is not yet functional.
- No detection and recovery from erroneous conditions.
- Power management is only partially implemented:
  - Timings are not optimized.
- No security features are yet implemented (no TF-M, no secure partitions). As a consequence, all services are currently implemented as regular FreeRTOS tasks.

2 Logical view

This section provides a detailed explanation of the modular architecture of the application. It begins with an overview of the various services that make up the system. Subsequent sections break down each service into its constituent modules, highlighting their structure and functionality. The final section illustrates how these modules and services interact across the different lifecycle states of the device.

By the end of this chapter, developers have a comprehensive understanding of the system's modular design, enabling them to effectively customize and extend the application to suit their specific needs.

2.1 Services

The software architecture is organized into three semi-independent services, each responsible for a distinct functional domain. These services interact through well-defined interfaces and are coordinated primarily by the safety service.

- The Safety Service is the central authority for system integrity and operational control. It performs the following functions:
  - Configures system safety parameters at startup
  - Monitors critical conditions such as battery health and wake-up pin state
  - Detects and mitigates safety issues in real time
  - Controls the overall system state and governs the activation and deactivation of other services
- The Health Service is responsible for data acquisition and secure storage. Its key responsibilities include:
  - Performing sensor measurements (for example, ambient temperature)
  - Encrypting and storing measurement data in persistent memory
  - Operating under the control of the safety service, which manages its activation and deactivation
- The Connectivity Service enables wireless communication with external devices via Bluetooth Low Energy. Its main functions are:
  - Allowing a Bluetooth Low Energy central device (for example, a smartphone) to connect and retrieve sensor and battery data
  - Exchanging the latest measurement data with the health service
  - Adapting Bluetooth Low Energy profiles and services based on the type of measurements performed. In the Health care IoT Peripheral application, the Health Thermometer Service is implemented
  - Like the health service, its operation is managed by the safety service

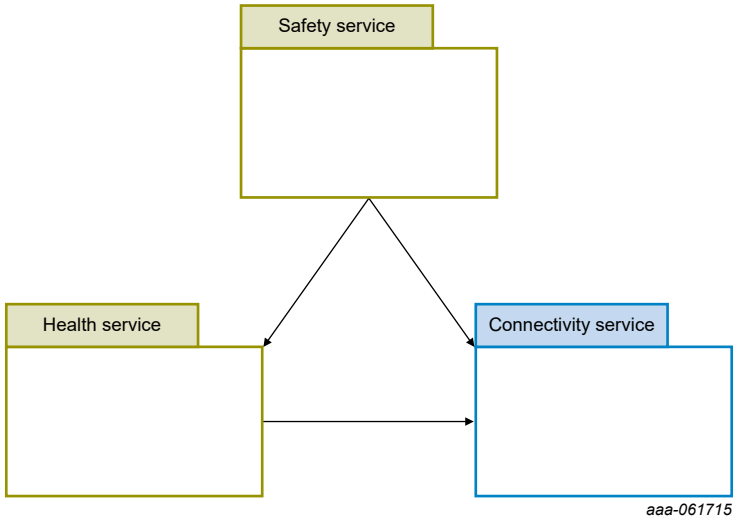

Figure 3. Services

The application applies Trusted Firmware-M (TF-M) to manage and isolate its core services. Each service operates within its own TF-M Secure Partition, ensuring strong separation and security boundaries between components.

Services communicate through a secure Inter-Process Communication (IPC) mechanism provided by TF-M. This mechanism enforces strict access control and data integrity during interservice communication.

Whenever a context switch occurs either due to task scheduling or an interrupt, TF-M automatically reconfigures the system security settings. This process ensures that the newly activated service executes within its designated secure environment. The same security enforcement applies when handling interrupts routed to services within Secure Partitions.

2.2 States

2.2.1 System states

The system can be in four different states based on activation button/magnet and battery condition:

- **Shelf mode** is the initial state entered when a new device is powered on for the first time. In this mode, all services remain inactive, and the system enters a low-power, power-off state with the wakeup pin enabled. If the device is in active mode and the wakeup button is pressed, the system transitions back to shelf mode.
- Pressing the activation button or removing the magnet while in shelf mode transitions the system to the **active** state. In this state, all three services are activated. The system begins collecting measurements, and the Bluetooth subsystem is enabled to allow external central devices to connect and retrieve data. The safety service monitors battery health and starts watchdog timers to detect potential system lockups.
- When the battery can no longer support reliable measurements or data storage, the system enters the **end-of-life with connectivity** state. The Bluetooth subsystem remains active, allowing a central device to connect and retrieve the battery status and the last recorded temperature. However, new measurements are no longer taken, and the Bluetooth sensor service is no longer updated. For demonstration or testing purposes, this state can also be triggered manually using the activation button.
- **End-of-life without connectivity**: If the battery condition deteriorates further, the system can no longer maintain Bluetooth connections. It transitions to its final state—power-off mode without wakeup capability. This state can also be manually triggered for testing or demonstration purposes using the activation button.

Note: *For demonstration and testing, the activation button can be used to manually transition between states.*

The safety service is responsible for managing the overall system state.

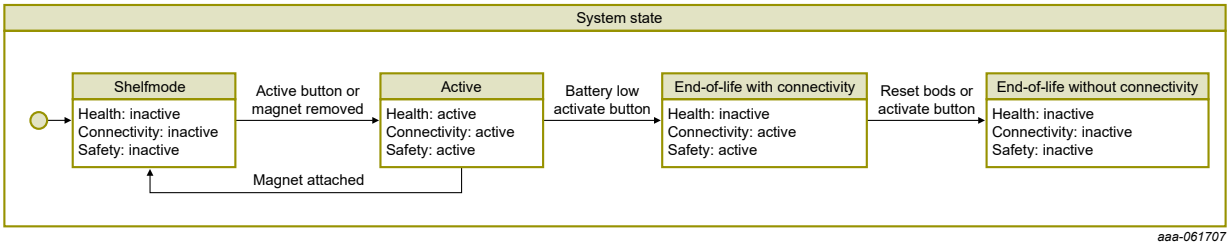

Figure 4. System states

2.2.2 Safety service states

The safety service can be in two states:

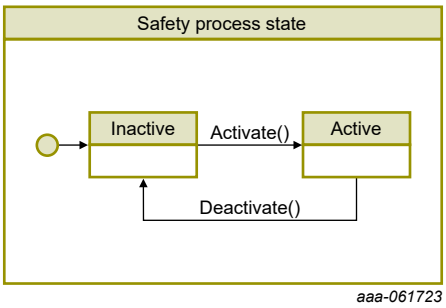

Figure 5. Safety service states

- In an **inactive** state, none of the safety mechanism are active. This state is active when no software is running and there's no risk of security breaches
- In an **active** state, the watchdog, reset BOD, battery monitor BOD are active.

2.2.3 Health service states

The health service operates through the following states:

- **Inactive**: This is the default state where the health service remains idle and performs no operations.
- **Syncing**: Entered immediately after activation or when a new measurement cycle begins. In this state, the system coordinates with the connectivity service to identify idle periods in radio activity. This helps distribute tasks efficiently and minimizes power consumption by avoiding current peaks.
- **Measuring**: Once a suitable gap in radio activity is found, the system transitions to this state to configure and initiate a sensor measurement.
- **Processing**: Triggered by an interrupt signaling the completion of a measurement. The data is processed synchronously, and the result is added to the internal data cache.
- **Storing**: When the cache reaches capacity, the data is encrypted and written to flash memory during available radio idle time.
- **Waiting**: After storing, the system enters a low-activity state where it waits for the next scheduled measurement. A software timer is used to trigger the next cycle.

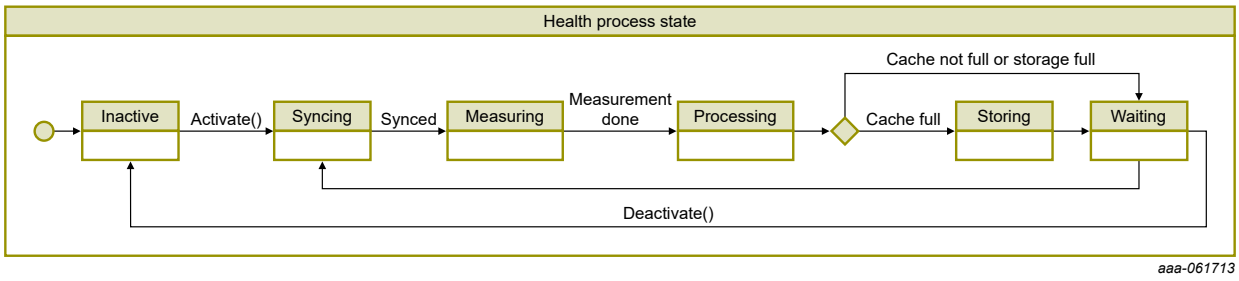

Figure 6. Health service state

2.2.4 Connectivity service states

The connectivity service operates in one of the following four states:

- **Inactive**: This is the initial state where the entire radio subsystem remains idle.
- **Advertising**: Upon instruction, the service transitions to this state to broadcast its presence. Bluetooth central devices can initiate a connection. If the device is already bonded, only the bonded central is allowed to connect.
- **Pair/Bond**: After a connection is established with an unbonded device, the service enters this state to perform pairing and bonding. It remains in this state until all bonding data is securely stored in persistent memory.

- **Connected**: This state is entered either after a successful bonding process or when a bonded device reconnects. In this state, the peripheral exchanges the packets regularly with the central device. If no application data is available, empty packets are exchanged to maintain synchronization.

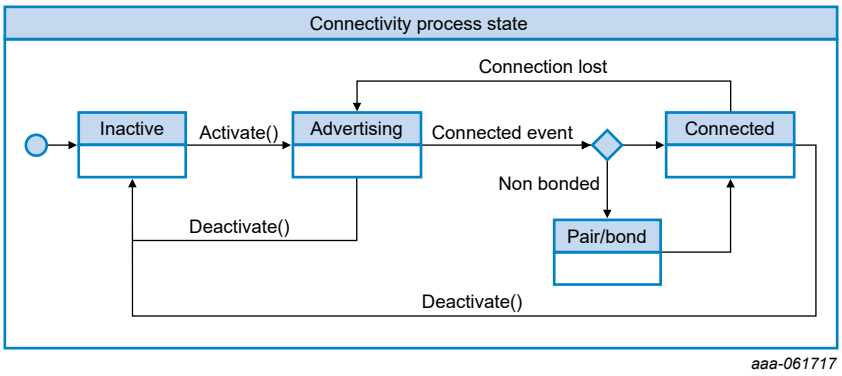

Figure 7. Connectivity service states

2.3 Decomposition

In this section, the software is decomposed into different components, each responsible for a specific task in the system.

2.3.1 Safety service

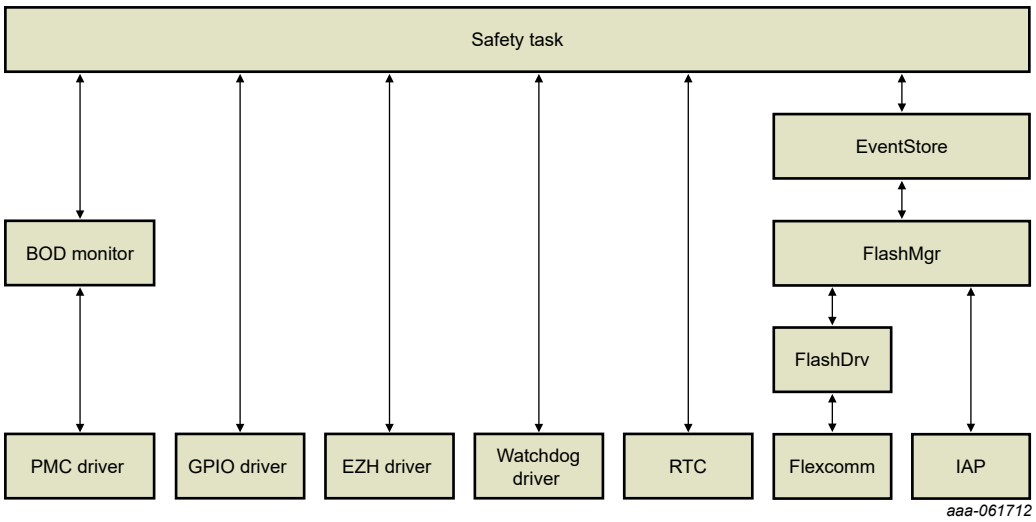

Figure 8. Safety service decomposition

2.3.1.1 Safety task

- Manages the system state and controls activation and deactivation of health service and connectivity service.
- Manages the safety service state.
- Configures the overall security features.
- Changes system state based on buttons.
- Changes system state based on BOD monitor events (low battery event and BOD resets).
- Handles watchdog events.
- Stores important events into persistent flash memory.

- Feeds the battery state to the connectivity service.

2.3.1.2 BOD monitor

The MCX W23 uses two BODs:

- A battery monitoring BOD that is configured to a relatively high voltage level. This BOD is triggered when the battery is near its end of life. The current fluctuates over time and may cause spurious BOD interrupts. The BOD monitor is responsible for determining the battery condition based on the interval and frequency of these events. A notification is sent to the safety task when the battery is end-of-life.
- A reset BOD: This BOD is configured to the minimal operating voltage of the MCX W23. Whenever the voltage drops below this level, the device may no longer operate as expected. This BOD unconditionally resets the device to bring it back to a known state. During initialization, the BOD monitor inspects the reset source and inform the safety task that when the device was reset due to this BOD.

2.3.1.3 Event store

The event store is responsible for:

- Recording important events into non-volatile memory.
- Timestamping of the events. The timestamps are always relative to the RTC start time. The Bluetooth Low Energy central can set (through the current time service) the actual time. This is recorded as an event and used to convert the timestamps to world time.
- Storing the events reliably such that the complete event log is not lost when writing new events to flash fails (for example, due to a BOD reset).
- Helper functions are available to parse the saved log.
- Ensuring other services can add events to the store in a thread safe way.

2.3.1.4 Flash manager

The flash manager offers functions to erase flash pages and write data to flash. The flash manager synchronizes its activities with the radio subsystem to avoid current peaks when those activities would coincide. When functions are called from other services/tasks, the flash manager offloads the execution to the safety task.

2.3.2 Health service

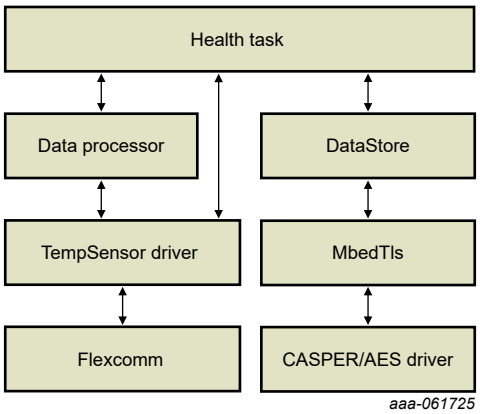

Figure 9. Health service decomposition

2.3.2.1 Health task

- Manages the state of the health service.
- Plans the sensor data acquisition.
- Stores the reported processed sensor data using the data store.
- Synchronizes with the connectivity service (to avoid current peaks) during measurement.
- Feeds the latest measurement data to the connectivity service.
- Processes activation and deactivation commands from safety task.

2.3.2.2 Data Processor

- Offers functions to acquire and process new sensor data.
- Interacts with the sensor driver to start the measurement and fetch the data.
- Informs the health task whenever newly processed data is available.

2.3.2.3 DataStore

- Manages the flash region that is used to store the sensor data. Tracks usage and handles exception cases gracefully:
  - When the flash is full, new samples are dropped.
  - Old data that was already written to flash is retained. New data is appended. This situation might occur after an unexpected system reset.
- Timestamps all data.
- The data store is responsible for encrypting the sensor data.
- To reduce power consumption, the encrypted sensor data is cached until a complete page can be written.
- Offers the functions to fetch and decode stored data.

2.3.3 Connectivity service

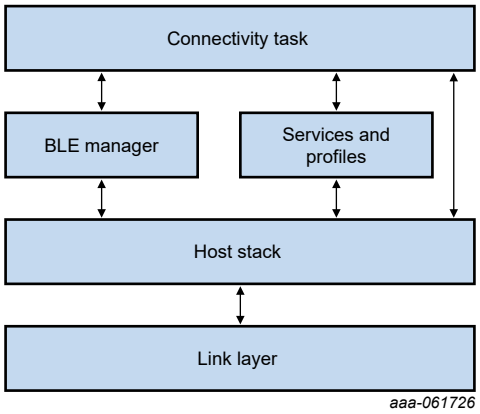

Figure 10. Connectivity service decomposition

The connectivity service consists of the following components:

2.3.3.1 Connectivity task

- Manages the state of the connectivity service.
- Initializes the Bluetooth Low Energy middleware components.
- Processes the commands from the safety service to activate and deactivate again.

- Handles synchronization information requests. Other components might request idle information from the radio subsystem to plan their activities.

2.3.3.2 Bluetooth Low Energy manager

This component is responsible for initializing the GAP host stack layer and handling of connection-related events from the host stack.

It offers functions to:

- Start advertising.
- Stop advertising.
- Disconnect.
- Unbond a paired device.

In addition to the above functions, it also informs the connectivity task each time when:

- A connection is established.
- A new device is bonded.
- The connection is lost.

The Bluetooth Low Energy manager autonomously:

- Handles the different phases of advertising (fast advertising, slow advertising, …).
- Configures of the advertisement parameters and data (for example, direct vs undirected advertising based on the bonding information).
- Handles the pairing and bonding process.
- Manages the bonding information.
- Handles connection lost.

2.3.3.3 Services and profiles

This component consists of the different services supported by the Health care IoT Peripheral application, like Device Information Service (DIS), temperature, current time, and battery service.

2.3.4 Data flow

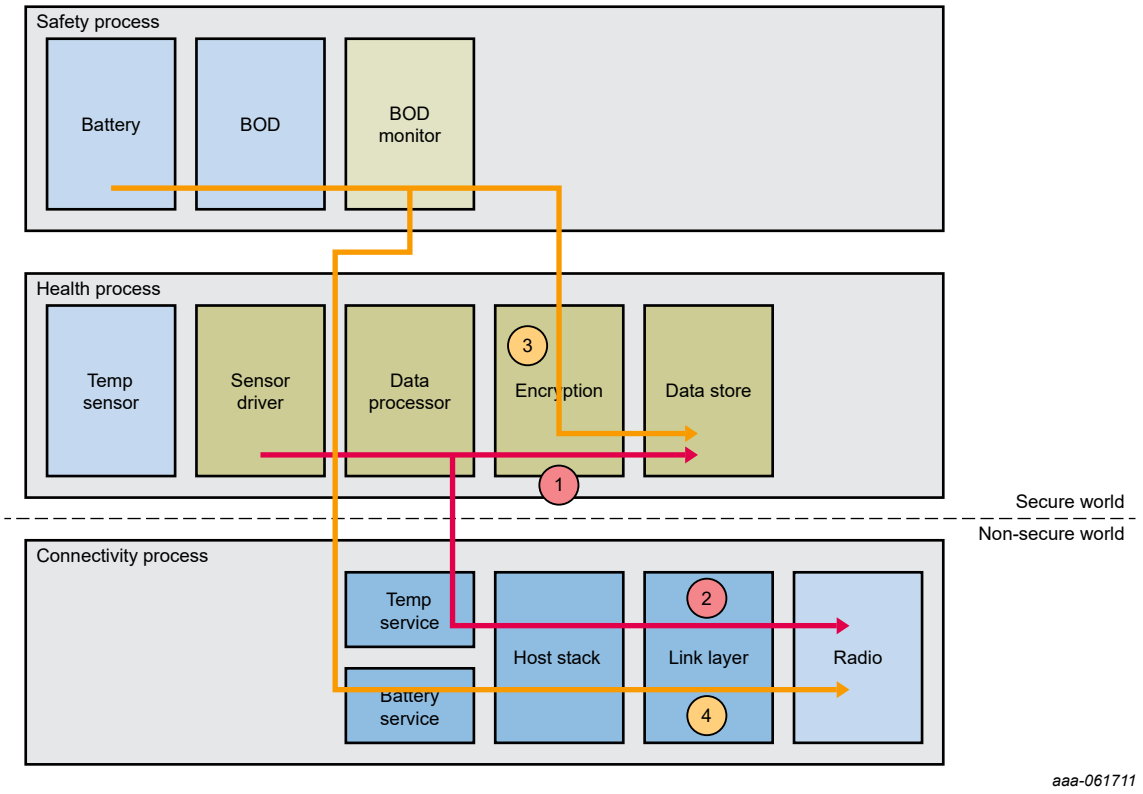

Figure 11. Data flow diagram

1. Temperature sensor data stored in NVM.
2. Temperature sent over Bluetooth Low Energy.
3. Key events stored in NVM.
4. Battery level sent over Bluetooth Low Energy.

2.4 Use cases

The sequence diagrams in this section demonstrates how the different components interact with each other to handle common use cases.

2.4.1 System initialization

The below sequence is always executed at system startup before any of the other use cases is executed. This sequence ensures that the system is in a well-defined state in which all security and safety-related measures are configured and enabled.

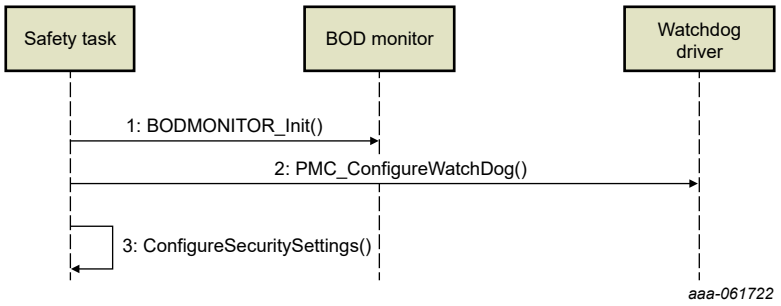

Figure 12. Use case system initialization

1. The BOD monitor is initialized to prevent the system usage outside the operating voltage range. The system resets when this happens.
2. The watchdog is configured to ensure that the system is recovered when the software crashes
3. All other security and safety-related measures are configured and enabled.

2.4.2 Power on of the new device

A new device is a device that is flashed with the Health care IoT Peripheral application while the rest of the flash is blank.

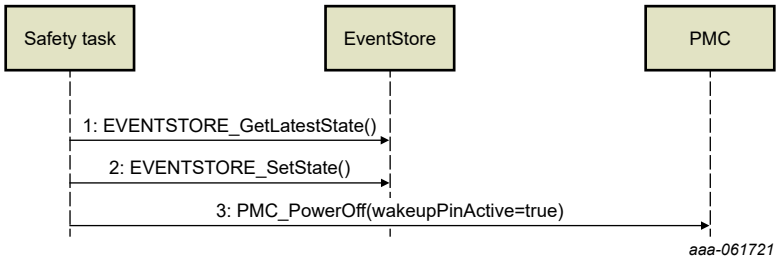

Figure 13. Use case power on of new device

1. The event log is read from the flash and the latest state is retrieved.
2. No state information is found. The system state is initialized to shelf mode and this event is added to the event store.
3. The system is put in power off mode with the wake-up pin active.

2.4.3 Wake up in shelf mode

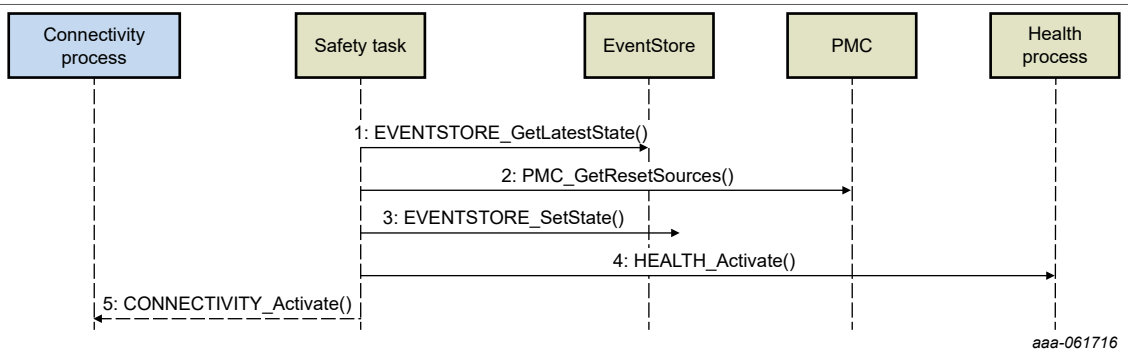

Figure 14. Use case wake-up in shelf mode

1. The event log is read from the flash and the latest state is retrieved.

2. The system woke up due to the assertion of the wake-up pin.
3. The state transition is recorded and stored in the flash.
4. The health process is activated.
5. The connectivity process is activated through the registered callback function.

Note: *When the device wakes up from another state or the reset source is "normal boot", the flash containing data and events is reinitialized and the system is put in shelf mode again. This is to allow easily to do the factory reset by power cycling the board (for demo and testing purposes)[1].*

2.4.4 Battery low

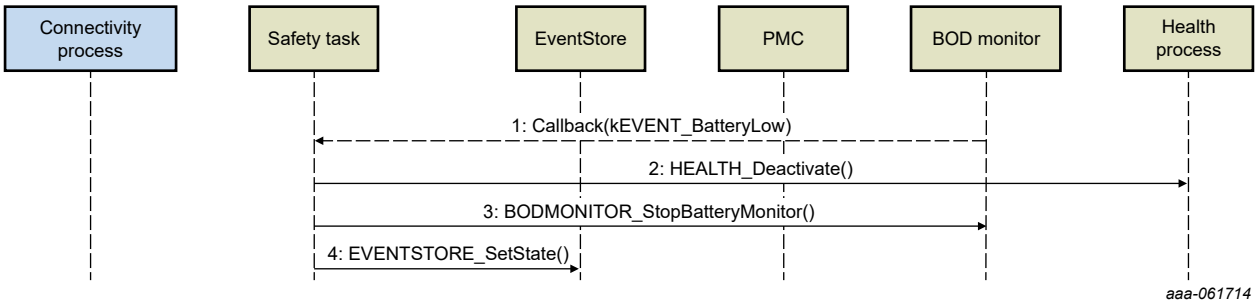

Figure 15. Use case battery low

1. The safety task is informed when the BOD monitor detects the battery is low.
2. The health process is immediately stopped to prevent unreliable measurements.
3. Battery monitoring is no longer needed.
4. The system state transitions to end-of-life with connectivity. The new state is written to flash.

Note: *The connectivity process continues in this state.*

2.4.5 BOD reset

In this use case, the system is reset due to a voltage dip.

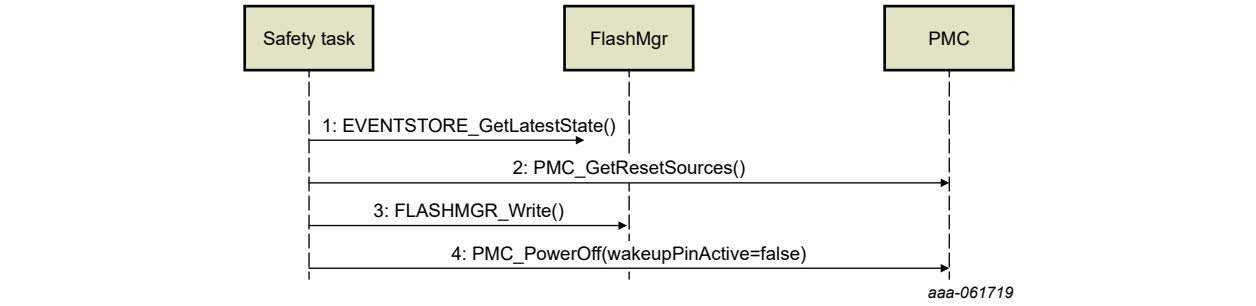

Figure 16. Use case BOD reset

1. The event log is read from flash.
2. The last registered state transition is end-of-life-with-connectivity.
3. The safety task fetches the reset source and sees that the system has been reset due to a BOD reset. A Bluetooth connection can no longer be maintained and the system transitions to end-of-life-without-connectivity.
4. The new state is written to flash.

1 Not yet implemented

5. The system is put in the power-off mode without the ability to wake up.

A BOD reset is considered spurious, if it happens in any other state than end-of-life-with-connectivity. The safety task restores the system to its previous state.

2.4.6 Measurement cycle

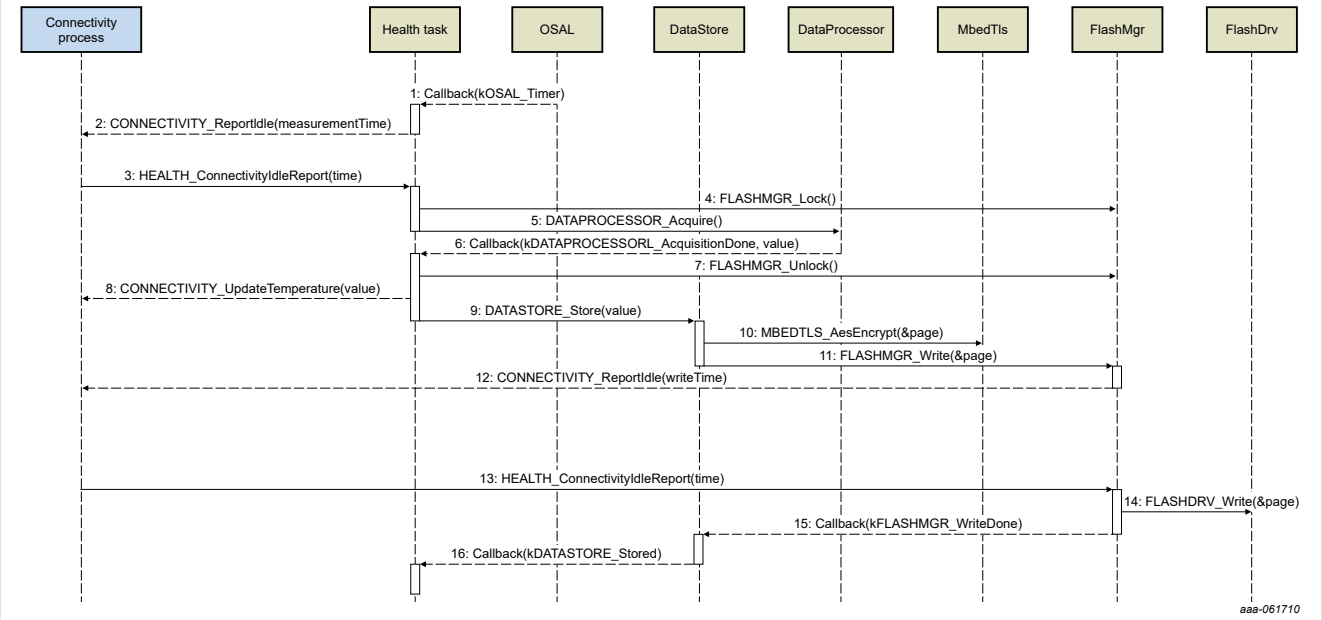

Figure 17. Measurement cycle with full cache

1. The recurring OS timer signals that it is time for a new measurement.
2. The radio subsystem is active and may be busy in transmitting or receiving the data. To avoid concurrent activities and high peak currents, the connectivity process is asked to report when the radio subsystem is idle for at least *measurementTime* time.
3. The radio reports it is idle until *time*.
4. The flash manager is locked to avoid any flash writes to avoid high peak currents.
5. The data processor is asked to start a new acquisition.
6. The data processor informs the application that the acquisition is done. Optionally, new sensor data is passed alongside.
7. The flash manager is unlocked.
8. The new measurement is passed to the connectivity process.
9. The data store is asked to store the new sensor data
10. The new data completes the cached flash page. The page is encrypted.
11. The page is passed to the flash manager to be written to flash.
12. The connectivity process is asked to report when it is idle for at least the *writeTime* time.
13. The connectivity process reports it is idle until *time*.
14. The page is written to flash.
15. After the page is written, the data store is informed.
16. The data store informs the health task that the new data is stored.

2.4.7 Bluetooth connection setup and teardown

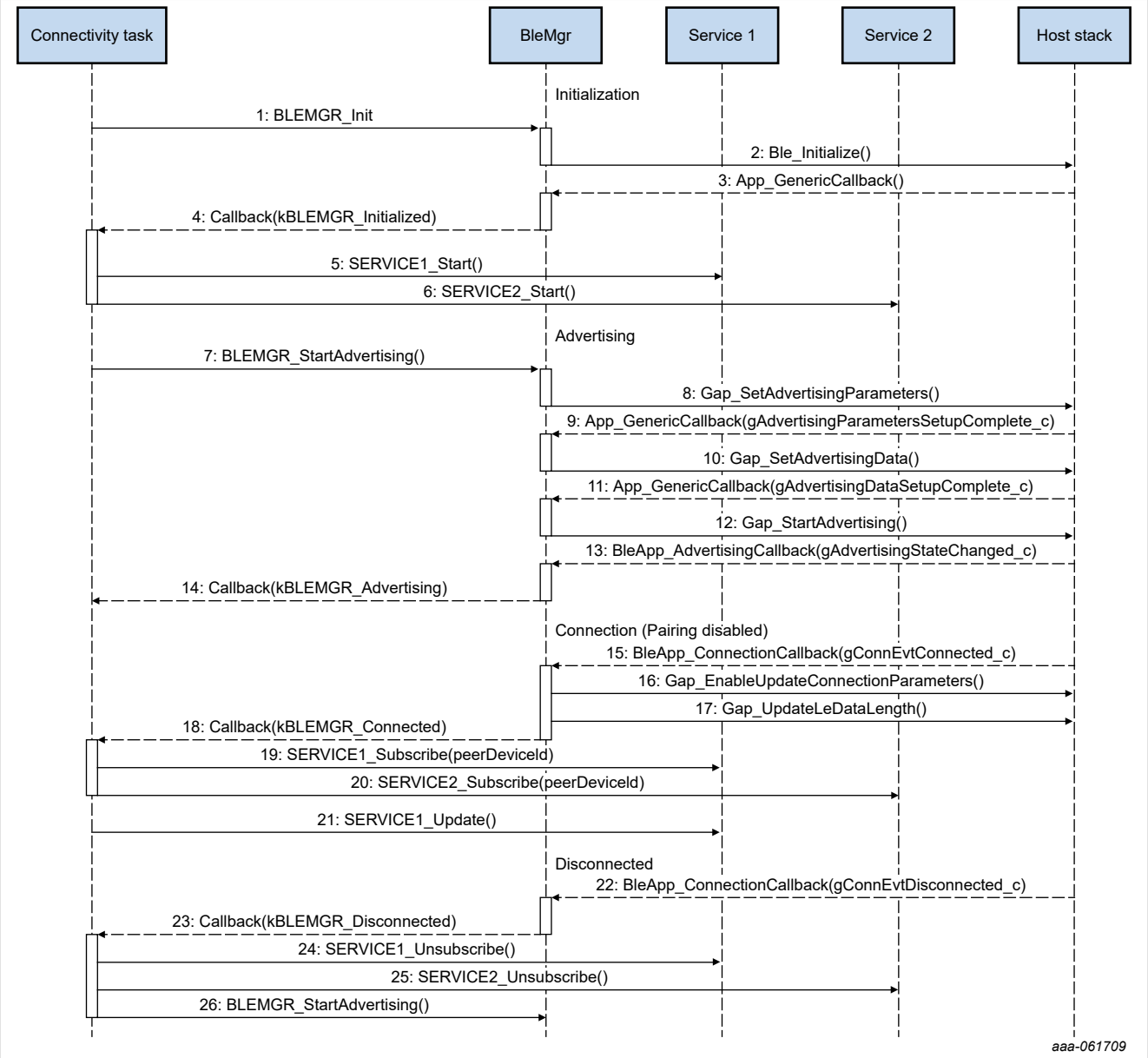

Figure 18. Use case Bluetooth connection setup and teardown

1. The Bluetooth Low Energy manager is initialized.
2. The Bluetooth Low Energy manager initializes the host stack. This is an asynchronous process.
3. After successful initialization, the Bluetooth Low Energy manager is informed.
4. The Bluetooth Low Energy manager on its own informs the connectivity task.
5. The connectivity task can now start service 1 and service 2 and possibly other services.
6. The Bluetooth Low Energy manager is asked to start advertising. This is an asynchronous process.
7. The advertising parameters are configured and confirmed.
8. The advertising data is set and confirmed.
9. The host stack is instructed to start advertising and confirmed.
10. The connectivity task is informed.

11. A connection can be established at any moment. The Bluetooth Low Energy manager is informed when the connection is established.
12. The connection parameters are updated and the packet length extension is configured.
13. The connection task is informed that a connection has been established.
14. The new peer is registered to service 1 and service 2.
15. The data of the service can now be updated (for example, the actual temperature can be updated in the temperature service).
16. When the connection is lost, the Bluetooth Low Energy manager is informed.
17. The Bluetooth Low Energy manager informs the connectivity task.
18. The connectivity task unsubscribes the peer from service 1 and service 2.
19. Advertising is again started to allow new connections.

3 Process view

3.1 Execution architecture

All processes in the system run independently. To achieve this, each process runs in its own task. Within a process, all activities are serialized. The system is event driven. A queue is used to store the events and possible metadata in order of arrival. The task dequeues the (oldest) event and processes it. The task has sufficient knowledge to process the event itself or delegate the event to the correct actor. For example, a battery BOD event is delegated by the task to the BOD monitor.

The event queue is used for two purposes:

- To offload processing of an interrupt to the process task context.
- To handover requests between different processes. For example, to update the sensor service running in the connectivity process with newly captured sensor data in the health process.

Asynchronous actions are broken down in smaller chained synchronous actions. A state machine tracks the actual progress of the asynchronous action. For example: writing to flash is a two-staged action: first the flash controller is instructed to write the page. The actual writing happens asynchronously. A state machine transitions to the “writing” state. The flash controller generates an interrupt when the page has been written. An event is created and handled by the task. The task checks whether this event is expected by checking the current state (writing) and transitions to the next state.

The system must be robust against asynchronous (unexpected) events. When a flash page is being written, the system may expect a flash write done event as the next event. When the battery condition has been degraded, a battery BOD event may happen before the flash write done event.

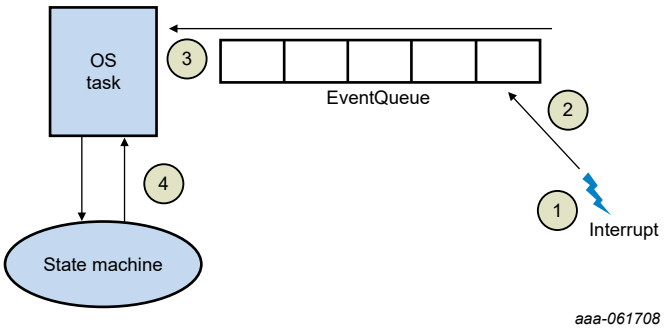

Figure 19. Execution architecture

1. An interrupt happens.
2. The interrupt service routine handles the interrupt. The time critical processing is executed in an interrupt context. The non-time critical processing is offloaded to the task by creating an event with optional metadata.
3. The task is suspended as it is waiting on an empty queue. The operating system schedules the task again as new events have arrived.
4. The task handles the event and possibly updates a state machine.

3.2 Security

3.2.1 Flash memory encryption

The flash memory is split into four regions, each storing different kind of information that requires different encryption.

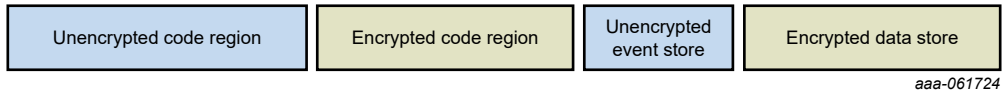

Figure 20. Flash memory layout

1. The unencrypted code region holds most the code. All code in the region is considered “public” [term].
2. The encrypted code region holds the data processing algorithm. This algorithm is considered [term]. PRINCE is used to encrypt and decrypt this region on the fly. Accessing a PRINCE-encrypted memory increases the power consumption.
3. All important events are stored in the unencrypted event store region. This region does not contain sensitive (patient) information and encryption is not necessary.
4. The encrypted data store region contains the measurements. The data in this region is encrypted with AES and not PRINCE. This region can also be placed in external flash.

3.2.2 TrustZone and privileged code

TrustZone allows you to split code into trusted and untrusted code. Trusted code runs in the secure world and has access to sensitive information (encryption keys, privacy related data (for example, sensor data)) while untrusted code runs in the non-secure world. Untrusted code can never steal sensitive information. TrustZone protects against *deliberate* attacks.

Secondly, code can also be split into privileged and non-privileged modes. This mechanism mainly aims to protect against *accidental* data corruption and unintentional code execution due to bugs. The main goal is to increase resilience against bugs to make the system more stable. In non-privileged mode, each thread has only access to the memory region it needs to perform its task.

With this in mind, the data and code memory per process is split as follows:

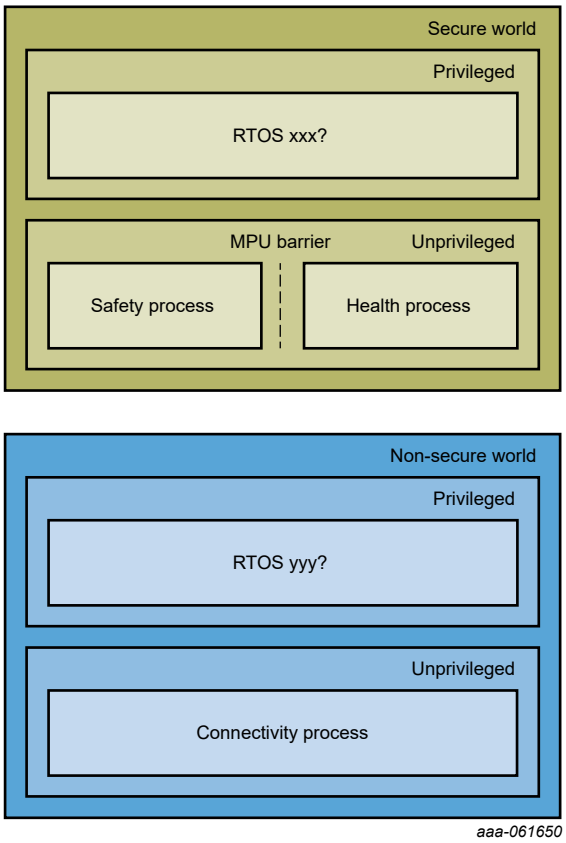

Figure 21. Memory partitioning

4 Performance

4.1 Memory footprint

The below table summarizes the memory consumption of the Health care IoT Peripheral application:

Table 1. Memory usage freertos

Module	Flash usage (in bytes)	Ram usage (in bytes)
Power lib	5090	4
Crypto lib	8608	238
Libc	6271	434
App	8058	1369
Comps	12676	844
Drivers	16592	741
Link layer	153616	18715
Host stack	85553	3562
Profiles	1595	1
Bluetooth Low Energy framework	7610	996
Other	356	320
Total	306025	27220

Table 2. Memory usage bare-metal

Module	Flash usage (in bytes)	Ram usage (in bytes)
Power lib	5090	4
Crypto lib	8608	238
Libc	6303	434
App	8030	1677
Comps	5290	182
Drivers	16122	745
Link layer	153296	18699
Host stack	85553	3586
Profiles	1595	1
Bluetooth Low Energy framework	7610	996
Other	342	322
Total	297839	26884

5 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision history

Table 3 summarizes the revisions done to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14757 v.1.0	05 August 2025	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Suitability for use in industrial applications (functional safety) — This NXP product has been qualified for use in industrial applications. It has been developed in accordance with IEC 61508, and has been SIL-classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Amazon Web Services, AWS, the Powered by AWS logo, and FreeRTOS — are trademarks of Amazon.com, Inc. or its affiliates.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Tables

Figures

Contents