# STM32U5 Series power optimization using LPBAM

## Introduction

The STM32U5 Series microcontrollers are based on the high-performance Arm® 32-bit Cortex®-M33 CPU with Arm® TrustZone® and FPU. They use an innovative architecture to reach best-in-class, ultra-low power figures thanks to their high flexibility and advanced set of peripherals.

In addition to several CPU active mode configuration, in which higher performance or lower power consumption can be favored, the STM32U5 Series support a large number of low-power modes, each of them with several options. This allows the designer to achieve the best compromise between low-power consumption figure, shorter startup time, available set of peripherals or amount of SRAM, and maximum number of wakeup sources.

An embedded SMPS (switched-mode power supply) step-down converter is available on specific part numbers with a "Q" suffix (such as STM32U5xxxxxxQ), increasing energy performance in both active and low-power modes.

The STM32U5 devices support four main low-power modes:

- Sleep mode

  The CPU clock is off, but all peripherals can be kept active. All peripheral interrupts or events can wake up the CPU.
- Stop mode

  High-speed clocks are off by default, and all peripherals and memory retention can be kept active. Four Stop modes are available: Stop 0, Stop 1, Stop 2 and Stop 3, from highest consumption to the lowest. LPBAM (low-power background autonomous mode) is a innovative feature, supported in Stop 0, Stop 1 and Stop 2 modes. Thanks to LPBAM, some peripherals keep working autonomously with the DMA (direct memory access). This reduces drastically the application power consumption when peripherals must be kept active. The number of peripherals that can be kept active thanks to LPBAM is lower in Stop 2 versus Stop 0 and Stop 1. LPBAM is not supported in Stop 3 mode.
- Standby mode

  The internal regulator is switched off. Most peripherals and SRAM retention are then lost. Up to 64 Kbytes of SRAM2, and 2 Kbytes of BKPSRAM can be retained in Standby mode.
- Shutdown mode

  This is similar to Standby mode, but the power supply brownout reset and monitoring are deactivated. The switch to $V_{BAT}$ is then not possible in this mode.

The combination of highly flexible low-power modes with autonomous peripherals thanks to LPBAM, and high energy-efficient processing allows STM32U575xxxxQ/STM32U585xxxxQ devices to achieve an industry leading EEMBC® ULPBench™ score, up to 535 ULPMark™.

# 1 General information

This application note applies to the STM32U5 Series that are Arm® Cortex® core-based devices.

*Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.*

### Reference documents

[1] Reference manual *STM32U575xx and STM32U585xx advanced Arm-based 32-bit MCUs* (RM0456)

[2] Datasheets for STM32U575xx (DS13737) and STM32U585xx (DS13086)

[3] Application note *STM32 microcontroller GPIO configuration for hardware settings and low-power consumption* (AN4899)

[4] Application note *STM32U575/585 power optimization* (AN5652)

[5] EEMBC organization on http://www.eembc.org

# 2 LPBAM presentation

## 2.1 LPBAM overview

The LPBAM (low-power batch autonomous mode) is an operating mode that allows peripherals to be functional and autonomous independently from the device power modes, down to Stop 2 mode, without any software running. The LPBAM subsystem can chain different operations thanks to DMA linked-list transfers. The DMA operations can be related to:

- Peripheral data transfer
- Peripheral reconfiguration

Using LPBAM optimizes automatically the consumption:

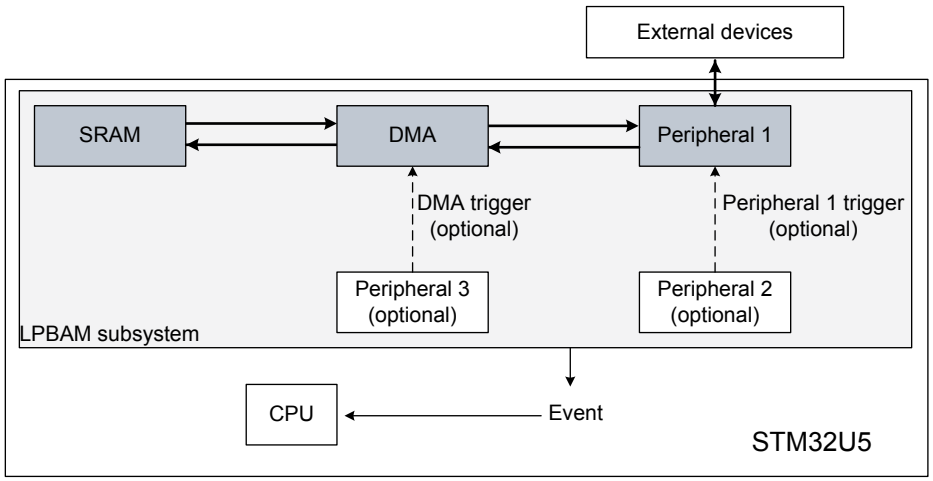
- The bus clock and kernel clocks of peripherals are distributed only when requested by the autonomous peripherals. The bus clock (also named system clock) is distributed over AHB and APB to all peripherals that are enabled, which includes a DMA and a SRAM at minimum.
- Internal RC oscillators are automatically powered on and off depending on peripherals clocks requests. The external oscillators and PLL cannot be used for LPBAM.
- Analog peripherals are automatically powered on and off when needed.
- The device can be in a low power mode down to Stop 2, without any need to wake up for managing peripheral operation, thus saving energy loss during device wakeup time and run operation.

A large choice of hardware triggers allows peripheral activity to start automatically even in Stop mode. The peripheral interrupts, when enabled, wake up the device from Stop mode.

A typical and basic LPBAM use case is a periodic peripheral operation (such as ADC conversion, or sensor acquisition through communication interface such as $I^2C$ or SPI), when the device is in Stop 2 mode. Wakeup source can be any of the peripheral interrupts such as:

- Peripheral end of transfer/conversion
- DMA transfer complete
- Error detection

**Figure 1. LPBAM typical application**




More complex applications can be built using several DMA channels or chaining operations from different peripherals on the same channel.

## 2.2 Peripherals supporting LPBAM

The peripherals supporting LPBAM can be classified in two categories:

- Autonomous peripherals, with clock request capability

  These peripherals support DMA transfers in Stop mode.
- LPBAM passive peripherals

  These peripherals do not support DMA requests nor clock request generation. Some of them can be reconfigured thanks to the DMA when they receive the system clock due to DMA clock request. Some others can provide a hardware trigger to the autonomous peripherals.

Any interrupt generated by autonomous or passive LPBAM peripherals wakes up the STM32U5 device from Stop mode.

*Note: LPBAM is not supported in Stop 3 mode.*

**Table 1. Peripherals supporting LPBAM**

| Peripherals category | LPBAM category | Functional down to Stop 0/1 mode | Functional down to Stop 2 mode |
|---|---|---|---|
| SRAMs | Passive | SRAM1, SRAM2, SRAM3, SRAM4<br>(+ BKPSRAM) | SRAM4 |
| DMA | Autonomous | GPDMA1, LPDMA1 | LPDMA1 |
| Communication peripherals | Autonomous | I2C1/2/3/4<br>SPI1/2/3<br>LPUART1<br>U(S)ART1/2/3/4/5 | I2C3<br>SPI3<br>LPUART1 |
| Timers | Autonomous | LPTIM1/2/3/4 | LPTIM1/2/4 |
| | Passive | RTC | RTC |
| Application peripherals | Autonomous | MDF1, ADF1 | ADF1 |
| Analog peripherals | Autonomous | ADC4<br>DAC1 (two converters) | ADC4<br>DAC1 (two converters) |
| | Passive | Comparators<br>OPAMPs<br>VREFBUF | Comparators<br>OPAMPs<br>VREFBUF |
| I/Os | Passive | GPIO<br>LPGPIO | LPGPIO |

# 3 LPBAM hardware mechanism

## 3.1 Power and clock architecture

The STM32U5 is split into two domains: CPU domain (CD) and SmartRun domain (SRD). The figure and table below show the distribution of AHB and APB peripherals in these two domains.

**Figure 2. Architecture of power and clock domains**

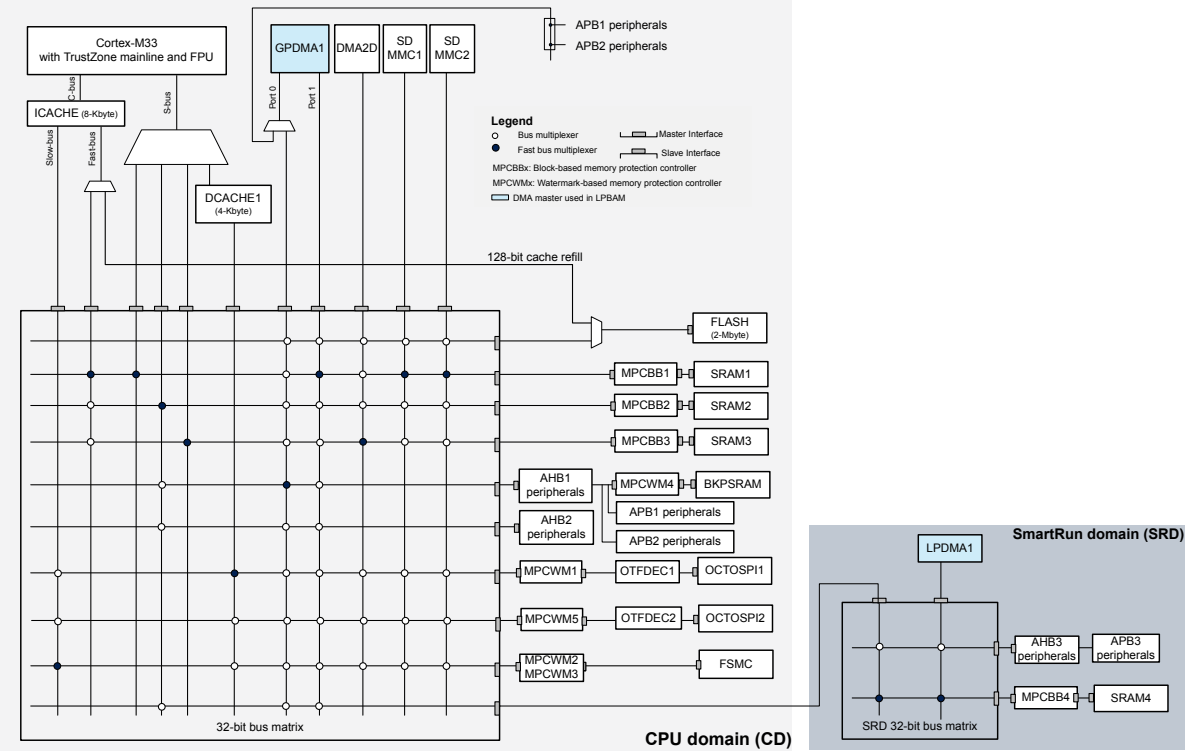



**Table 2. AHB/APB bus distribution over domains**

| AHB/APB bus | Domain | LPBAM versus low-power modes |
|---|---|---|
| AHB1, AHB2(1) | CD | LPBAM down to Stop 1 mode |
| APB1, APB2(1) | | |
| AHB3 | SRD | LPBAM down to Stop 2 mode |
| APB3 | | |

1. *Only a subset of peripherals support LPBAM (see Table 1).*

In Stop 0 and Stop 1 modes, CPU and SmartRun domains are fully powered, and can both support dynamic activity with LPBAM. Both GPDMA1 and LPDMA1 can then be used. GPDMA1 can access all device SRAMs, whereas LPDMA1 can access only SRAM4.

In Stop 2, the CPU domain is put in a lower leakage mode, which forbids any dynamic activity. Only SmartRun domain is fully powered and can sustain dynamic activity thanks to LPBAM. Only LPDMA1 can be used, which can access only SRAM4. LPDMA1 linked-list items and peripheral data buffers must all be located in SRAM4. This application note focus on LPBAM in Stop 2 as this is the most efficient mode for energy.

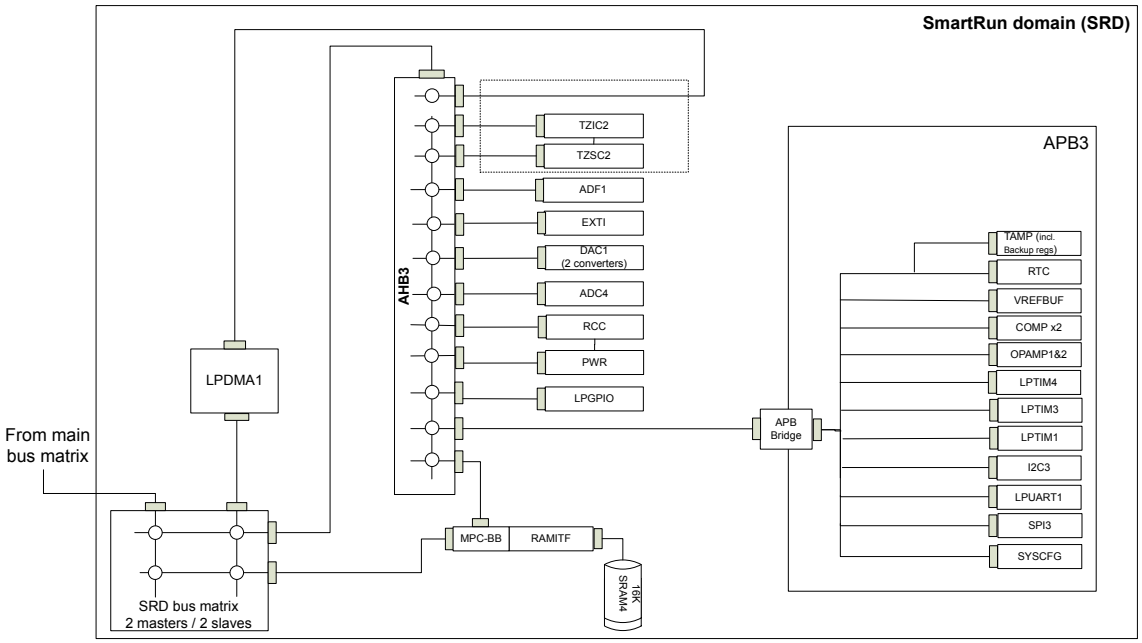
**Caution:** In Stop 0 and Stop 1 modes, autonomous peripherals that are mapped on AHB3 and APB3 (belonging to the SmartRun domain) can be used only with LPDMA1 and SRAM4. The main reason is that when a bus clock is requested by an SRD peripheral, AHB and APB clocks are distributed only in the SmartRun domain, and not in the CPU domain.

### 3.1.1 Clock distribution in SRD during LPBAM activity in Stop 0, Stop 1 or Stop 2 mode

Autonomous peripherals can request their clock (kernel or bus clocks), and can then stay functional in Stop mode where the clocks are disabled by default to reduce consumption (except for LSE and LSI low-power low-speed oscillators).

In Stop 2 mode, only the SmartRun domain content can be functional, with clocks running in this domain. The figure below zooms on the SRD architecture, showing all peripherals that are functional in Stop 2 mode.

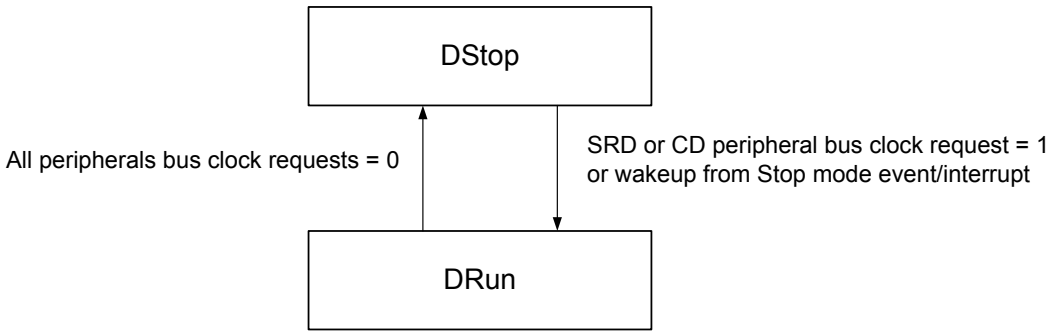
**Figure 3. SRD architecture**




The SmartRun domain can have two states:

- SRD in DStop state: AHB3/APB3 clocks are not present.
- SRD in DRun state: AHB3/APB3 clocks are present and distributed in Stop mode to all SRD peripherals that are enabled in the RCC (see Table 4).

**Figure 4. SRD state transitions**




*Note: The SRD can be forced to be always in DRun state by setting SRDRUN bit in PWR-CR2.*

The table below details domain states and clock distribution.

**Table 3. Clock distribution for SRD peripherals versus SRD states**

| SRD state | SRD peripheral bus clock (PCLK3/ HCLK3) | SRD peripheral kernel clock |
|---|---|---|
| DStop | OFF | ON(1) or OFF |
| DRun | ON(2) | ON(2) |

1. *Only if this peripheral is enabled in the RCC, and if this peripheral clock kernel request is set, or if the kernel clock of this peripheral is always enabled in Stop mode.*
2. *Only if this peripheral is enabled in the RCC (see Table 4).*

**Table 4. Clock enable for SRD peripherals**

| - | RCC enable bits | | | Run mode | Sleep mode | Stop 0/Stop 1 modes | | Stop 2 mode | |
|---|---|---|---|---|---|---|---|---|---|
| | xxEN | xxSMEN | xxAMEN(1) | | | SRD in DStop | SRD in DRun | SRD in DStop | SRD in DRun |
| SRD “xx” peripheral bus clock (AHB/APB) | 0 | x | x | OFF | OFF | OFF | OFF | OFF | OFF |
| | 1 | 0 | x | ON | OFF | OFF | OFF | OFF | OFF |
| | 1 | 1 | 0 | ON | ON | OFF | OFF | OFF | OFF |
| | 1 | 1 | 1 | ON | ON | OFF | ON | OFF | ON |

1. *This bit exists only for peripherals located in SRD. This bit must also be set to generate the wakeup from Stop interrupts.*

**Caution:** Any SRD peripheral that needs to be functional, or to be accessed by LPDMA1 during Stop 0, Stop 1 or Stop 2 mode must be configured with the three enable bits set in RCC_AHB3ENR/RCC_APB3ENR, RCC_AHB3SMENR/RCC_APB3SMENR, and RCC_SRDAMR registers.

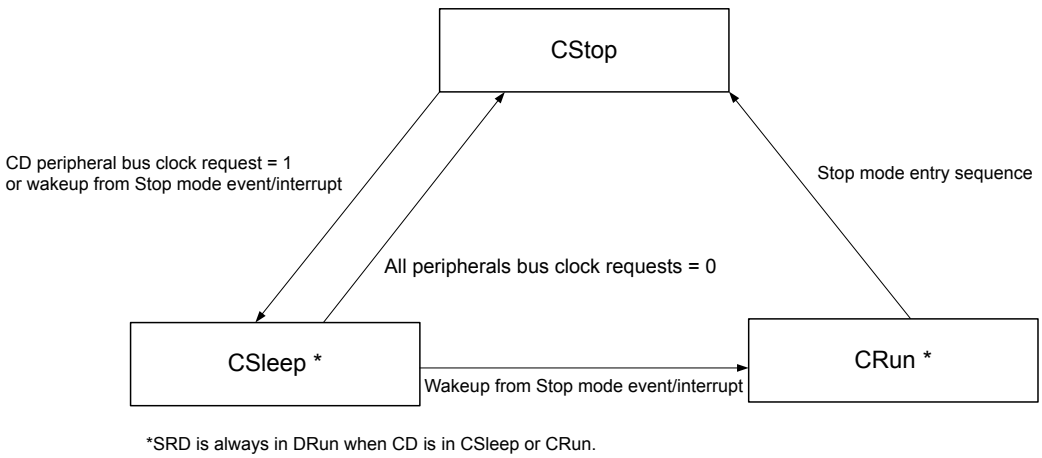
### 3.1.2 Clock distribution in CD during LPBAM activity in Stop 0 or Stop 1 mode

In Stop 0 and Stop 1 modes, both CPU domain and Smart run domain content can be functional, with clocks running in these domain. SRD peripherals clocks follow the rules described in previous section.

The CPU domain can have three states:

- CD in CStop state: AHB/APB clocks are not present.
- CD in CSleep state: CPU clock is stopped. AHB/APB clocks are present, and distributed in Stop mode to all CD and SRD peripherals that are enabled in the RCC (see Table 4 and Table 6).
- CD in CRun state: CPU runs. AHB/APB clocks are present, and distributed in Stop mode to all CD and SRD peripherals that are enabled in the RCC (see Table 4 and Table 6).

**Figure 5. CD state transitions**




*SRD is always in DRun when CD is in CSleep or CRun.

The table below sums up domain states and clock distribution.

**Table 5. Clock distribution for CD peripherals versus CD states**

| CD state | CPU clock | CD peripheral bus clock | CD peripheral kernel clock |
|---|---|---|---|
| CStop | OFF | OFF | ON(1) or OFF |
| CSleep | OFF | ON(2) | ON(2) |
| CRun | ON | ON(2) | ON(2) |

1. *Only if this peripheral is enabled in the RCC and if this peripheral clock kernel request is set, or if the kernel clock of this peripheral is always enabled in Stop mode.*
2. *Only if this peripheral is enabled in the RCC. Refer to Table 6.*

**Table 6. Clock enable for CD peripherals**

| - | RCC enable bits | | Run mode | Sleep mode | Stop 0/Stop 1 modes | |
|---|---|---|---|---|---|---|
| | xxEN | xxSMEN | | | CD in CStop | CD in CSleep |
| CD “xx” peripheral bus clock (AHB/APB) | 0 | x | OFF | OFF | OFF | OFF |
| | 1 | 0 | ON | OFF | OFF | OFF |
| | 1 | 1 | ON | ON | OFF | ON |

### 3.1.3 Clock requests for autonomous peripherals

The table below describes clock requests events for each autonomous peripheral.

**Table 7. Peripheral clock requests events**

| Autonomous peripheral | Bus clock request | Kernel clock request | Possible kernel clock sources |
|---|---|---|---|
| LPTIM | Set for DMA transfer (released when completed)<br>Set for interrupt generation (released when interrupt flag is cleared) | No request: the oscillator must remain enabled in Stop mode (not needed in external clock mode). | MSIK(1)(2)<br>HSI16(3)(4)<br>LSI(5)<br>LSE(5) |
| SPI | Set for DMA transfer (released when completed)<br>Set for interrupt generation (released when interrupt flag is cleared) | Master mode: generated by hardware trigger or software start, until end of transfer<br>Slave mode: no request | HSI16<br>MSIK(1) |
| I2C | Set for DMA transfer (released when completed)<br>Set for interrupt generation (released when interrupt flag is cleared) | Master mode: generated by hardware trigger or software start, until end of transfer<br>Slave mode: generated upon START condition detection on $I^2C$ bus until end of transfer | HSI16<br>MSIK(1) |
| LPUART | Set for DMA transfer (released when completed)<br>Set for interrupt generation (released when interrupt flag is cleared) | Transmitter mode: generated by hardware trigger or software start, until end of transfer<br>Receiver mode: generated upon START bit detection until end of transfer | HSI16(4)<br>MSIK(1)(6)<br>LSE |
| ADC | Set for DMA transfer (released when completed)<br>Set for interrupt generation (released when interrupt flag is cleared) | Generated by hardware trigger, until end of conversion | HSI16<br>MSIK(1) |

| Autonomous peripheral | Bus clock request | Kernel clock request | Possible kernel clock sources |
|---|---|---|---|
| DAC | Set for DMA transfer (released when completed) | Generated by hardware trigger until end of conversion | HSI16<br>MSIK(1) |
| MDF/ADF | Set for DMA transfer (released when completed) | Always set | MSIK(1) |
| GPDMA/LPDMA | Set by hardware trigger or peripheral DMA request, until end of DMA transfer<br>Set by channel complete event to load next LLI, until end of LLI transfer<br>Set for interrupt generation (released when interrupt flag is cleared) | N/A | N/A |

1. *Limited to 24 MHz. This oscillator offers a large choice of frequencies, with low power consumption.*
2. *MSIK must be forced enabled in Stop mode by setting MSIKERON in RCC_CR.*
3. *HSI16 must be forced enabled in Stop mode by setting HSIKERON in RCC_CR.*
4. *HSI16 consumption is higher than MSIK at same frequency, but this oscillator can be used when accuracy is needed, and MSI in PLL-mode cannot be used.*
5. *LSI is recommended for low power, when 32 kHz resolution is enough and high accuracy not required. If accuracy is required, LSE must be used instead.*
6. *To get better accuracy, MSIK must be used in PLL-mode (requiring external 32.768 oscillator on LSE). In order to use PLL-mode in Stop mode, MSIPLLEN and MSIPLLFAST bits must both be set in RCC_CR, with MSIPLLSEL = 0.*

## 3.2 Speed limitations

The LPBAM operates in Stop mode, in which internal oscillators are disabled by default (except for low-speed clocks such as LSI and LSE), and SRAM4 is in a low-power state. The speed of peripherals must take into account the fact that the maximum clock speed in Stop 1 and Stop 2 modes is 24 MHz (similar to Stop 0 with the regulator in Range 4), and the fact that the clock has to be woken up each time it is needed in Stop mode, which introduces a delay.

**Caution:** The speed of peripherals is limited compared to operation in Run or Sleep mode Range 4, as the clock may be disabled when the peripheral requests it.

### 3.2.1 System clock latency after clock request in Stop 2 mode

The peripheral requests its bus clock (AHB or APB clock) when a data is available and must be read by DMA, or when its buffer is empty and must be filled by DMA. More generally, the system clock is requested each time a DMA transfer must be done. The system clock is also requested in order to generate a wakeup interrupt.

As soon as a clock request is set, this delay must be considered before the clock reaches the peripheral which includes:

$$\left[t_{SU}(HSI16)\ or\ t_{SU}(MSI)\right] + t_{SU}(SRAM4) + t_{dig1}$$

Where:

- $t_{SU}$(HSI16) if STOPWUCK = 1 in RCC_CFGR1: HSI16 oscillator selected as wakeup from stop clock (see document [2] for the startup time value).
- $t_{SU}$(MSI) if STOPWUCK = 0 in RCC_CFGR1: MSIS oscillator selected as wakeup from stop clock (see document [2] for the startup time value).
- $t_{SU}$(SRAM4): this delay is based on a digital counter on AHB3 clock covering at least the SRAM4 startup time, around 3 µs (see the table below for $t_{SU}$(SRAM4) values depending on startup clock).
- $t_{dig1}$ corresponds to around 10 AHB clock cycles that are added in digital system.

**Table 8. Delay from clock request to peripheral being clocked**

| System clock | $t_{SU}$(SRAM4) | | $t_{dig1}$ | Unit |
|---|---|---|---|---|
| | SRAM4FWU = 0 | SRAM4FWU = 1(1) | | |
| MSIS 24 MHz | 60 | 0 | 10 | AHB3 clock cycles |
| HSI16 or MSIS 16 MHz | 37 | | | |
| MSIS 12 MHz | 26 | | | |
| MSIS 4 MHz | 3 | | | |
| MSIS 2MHz | | | | |
| MSIS 1.33MHz | | | | |
| MSIS 1 MHz | | | | |

1. *Setting SRAM4FWU (SRAM4 fast wakeup mode) in PWR_CR2 reduced the trigger latency and improves the wakeup time from Stop. The additional consumption in this mode is less than 200 nA (see Section 5 ).*

A system clock with lower frequency than 1 MHz can be used but increases the wakeup time and energy loss during this wakeup. This configuration is not recommended. In case MSIS frequency is lower than 1 MHz, $t_{SU}$(SRAM4) delay in cycles is the same than up to 4 MHz.

**Caution:** The latency after which the peripheral gets the clock after a clock request is not fixed. The clock may already be present if another peripheral clock request is already set. In this case, the oscillator startup time is removed from the previous formula. The synchronization delay can also vary and depends on AHB3 to APB3 prescaler.

*Note: AHB3 and APB3 clocks can be forced to be always present in SRD in Stop modes, by setting SRDRUN bit in PWR_CR2. These configurations increase the consumption, but allow an almost fixed latency following clock request. Only a jitter due to synchronization uncertainties remain in this case. Refer to Section 5 .*

### LPDMA transfer

The peripheral generates the DMA request after receiving the bus clock. The table below gives the number of AHB cycles needed for a LPDMA transfer (hypothesis: only one DMA request at a time, and APB frequency equals AHB frequency, no prescaler). The latency of the DMA request service is increased if it is served after another one, or if an APB peripheral is accessed, with prescaler versus AHB.

In addition to LPDMA transfer, $t_{dig2}$ delay is needed to go back to Stop 2 mode without any system clock running.

**Table 9. LPDMA transfer delay, in peripheral hardware request transfer mode**

| LPDMA transfer type | $t_{LPDMA}$ LPDMA transfer duration(1) | $t_{dig2}$ | Unit |
|---|---|---|---|
| AHB3 to AHB3 (SRAM4 or peripheral) | 6 | 10 | AHB3 clock cycles |
| APB3 peripheral to SRAM4 | 8 | | |
| SRAM4 to APB3 peripheral | 9 | | |

1. *One cycle must be added when AHB3 frequency is above 16 MHz, due to one cycle SRAM wait state.*

### Total data transfer duration from peripheral clock request to end of DMA transfer, in Stop 2 mode

The total duration from peripheral clock request to end of DMA transfer and no more system clock running, is summarized in the formula below (for a lonely DMA request, with APB clock frequency same as AHB clock frequency):

$$Delay = \left[t_{SU}(HSI16) \text{ or } t_{SU}(MSI)\right] + t_{SU}(SRAM4) + t_{dig1} + t_{LPDMA} + t_{dig2}$$

When SRDRUN = 1 in PWR_CR2, only LPDMA transfer duration ($t_{LPDMA}$) remains, which increases a lot the data rate capability, but with much higher consumption.

To avoid any underrun/overrun issue, and to reduce consumption, LPBAM developer must ensure that the data transfer rate is slow enough compared to this total data transfer duration.

### 3.2.2 Peripheral clock latency after request

The peripheral requests its kernel clock as soon as it receives a trigger launching an operation, or if an activity is detected on the bus for a communication peripheral (such as Start detection for I2C or UART).

If the peripheral kernel clock is LSI or LSE, this one remains enabled in Stop mode, so the peripheral is always clocked even in Stop mode.

If the peripheral clock is HSI16 or MSIK, this one is switched off by default in Stop mode. As soon as a clock request is set, the clock reaches the peripheral after the oscillator startup time: [$t_{SU}$(HSI16) or $t_{SU}$(MSI)], in case the oscillator was disabled. The clock may have been already enabled if already requested by another peripheral. In this case the peripheral receives immediately the clock upon request.

**Caution:** The LPBAM in Stop mode is not suitable when a fixed latency between trigger and peripheral start is required.

To get rid of oscillator startup time uncertainty, force MSIKERON or HSIKERON bits in RCC_CR, if these oscillators are used as peripheral kernel clocks. The consumption is then increased by the consumption of the oscillator and all circuitry clocked by this oscillator.

## 3.3 How to debug LPBAM scenario

When developing a LPBAM scenario, the three power alternate functions outputs CSLEEP (PA5), CDSTOP (PA6) and SRDSTOP (PA7) are helpful. SRDSTOP must be kept in priority in case LPBAM is done in Stop 2 mode and these three I/Os cannot be reserved for debug.

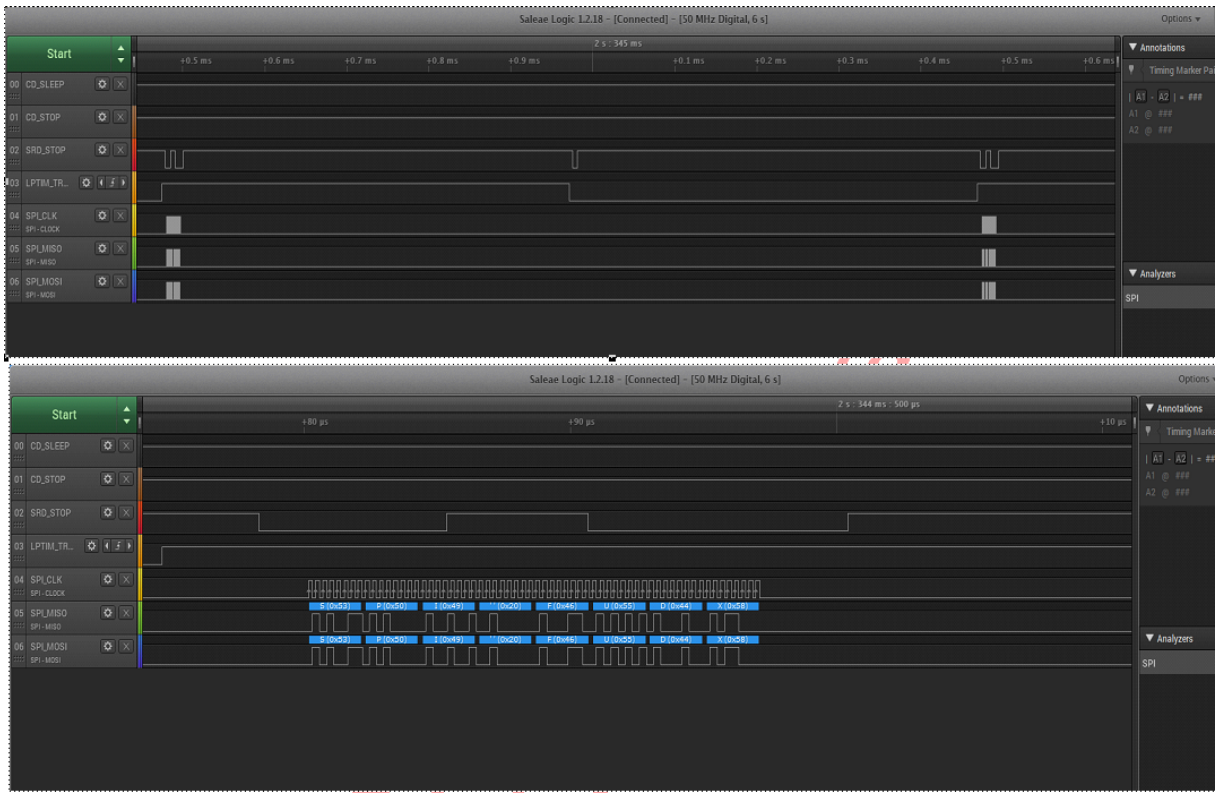
**Table 10. MCU power state versus domain state and debug pin sates**

| STM32U5 power mode | CD state | SRD state | CSLEEP | CDSTOP | SRDSTOP | Comment |
|---|---|---|---|---|---|---|
| Run | CRun | DRun | 0 | 0 | 0 | - |
| Sleep | CSleep | DRun | 1 | 0 | 0 | - |
| Stop 0<br>Stop 1 | CStop | DStop | 1 | 1 | 1 | Stop 0/1 mode with no AHB/APB peripheral activity |
| | CStop | DRun | 1 | 1 | 0 | Stop 0/1 mode with SRD AHB3/APB3 peripheral activity |
| | CSleep | DRun | 1 | 0 | 0 | Stop 0/1 mode with CD and SRD AHB/APB peripheral activity |
| Stop 2 | CStop | DStop | 1 | 1 | 1 | Stop 2 mode with no AHB/APB peripheral activity |
| | CStop | DRun | 1 | 1 | 0 | Stop 2 mode with SRD AHB3/APB3 peripheral activity |
| Stop 3 | CStop | DStop | N/A | N/A | N/A | The I/Os are not driven in Stop 3 mode |
| Standby<br>Shutdown | N/A | N/A | N/A | N/A | N/A | CD and SRD not supplied in Standby and Shutdown modes |

Thanks to debug signals showed in the figure below, the user can see when the device is in full Stop 2 mode (three debug signals driven high), and when AHB3/APB3 clocks are present due to DMA transfer (SRSTOP driven low). The power consumption obviously increases by the ratio SRDSTOP low versus SRDSTOP high.

The figure below is an example of SPI master transfer, with 8 bytes transmitted at each LPTIM PWM rising edge. The SPI is in 32-bit packing mode, so DMA transfers are done every four bytes.

**Figure 6. SPI master 8-byte periodic transfer with LPBAM in Stop 2 mode**



# 4 LPBAM peripheral features

The peripherals listed in Table 1 are functional down to Stop 2 mode, thanks to LPBAM. Table 11 summarizes the main peripherals functions that are supported by the LPBAM utility (LPBAM drivers, located under the Utilities folder of the STM32CubeU5 MCU Package available on www.st.com). Other features than the ones listed in the Table 11 can be supported by creating dedicated DMA linked-lists.

**Table 11. Peripheral functions supported by LPBAM drivers down to Stop 2 mode**

| Peripherals | LPBAM features | Comments |
|---|---|---|
| LPTIM | LPTIM PWM<br>LPTIM input capture<br>LPTIM update event<br>LPTIM start<br>LPTIM stop<br>*All interrupts with wakeup from Stop capability* | - |
| I2C | I2C master transmit (with optional configuration)<br>I2C master receive (with optional configuration)<br>I2C slave transmit (with optional configuration)<br>I2C slave receive (with optional configuration)<br>*All interrupts with wakeup from Stop capability* | Digital filter, timeout and idle bus detection are not supported in Stop mode. |
| SPI | SPI transmit (with optional configuration)<br>SPI receive (with optional configuration)<br>SPI transmit/receive (with optional configuration)<br>*All interrupts with wakeup from Stop capability except transmission transfer filled, TI frame format error, Mode fault, TxFIFO transmission complete (TxFIFO empty).* | Data packing mode is recommended to reach higher SPI frequency (only one DMA transfer every 4 bytes). |
| LPUART | UART receive (with optional configuration)<br>UART transmit without optional configuration)<br>*All interrupts with wakeup from Stop capability, except CTS, Idle line detected.* | LPUART maximum baudrate is limited by the kernel clock wakeup time (see documents [1] and [2]). |
| ADC | ADC conversion (without optional configuration)<br>ADC analog watchdog configuration<br>*All interrupts with wakeup from Stop capability, except end of sampling phase.* | Only single sequence conversions are supported. AUTOFF mode must be used, and DPD mode is recommended to reduce even more consumption. Conversion trigger period must be higher than the sum of ADCLDO startup time (if DPD is set), ADC power-up time (if AUTOFF is set), ADC conversion time, and total delay described in Total data transfer duration from peripheral clock request to end of DMA transfer, in Stop 2 mode. See document [2] for ADC parameters values. |
| DAC | DAC conversion (with optional configuration)<br>*No interrupt with wakeup from Stop capability* | Only sample and hold mode is supported. The DAC output change rate must then be adapted to this mode (underrun detection is not supported in Stop mode). |
| ADF | DMA transfers in Stop mode<br>*All interrupts with wakeup from Stop capability* | - |
| LPDMA | DMA start<br>*All interrupts with wakeup from Stop capability* | - |
| COMP | COMP configuration and start<br>COMP output level configuration | Comparator supports four speed/power modes (see document [2]). |
| GPIO | GPIO write pin (single and multiple write)<br>GPIO read pin (single and multiple read) | A maximum of 16 I/Os can be read or written in Stop 2 mode: LPGPIO1_P[15:0]. Refer to pin definition table in document [2]. |

| Peripherals | LPBAM features | Comments |
|---|---|---|
| OPAMP | OPAMP start | OPAMP supports two speed/power modes (see document [2]). |
| VREFBUF | VREFBUF mode configuration | VREFBUF supports a hold mode to reduce consumption (see document [2]) |

Functions listed in the above table manage peripheral data transfer, peripheral configuration, or both (configuration followed by data transfer). Each function builds the linked-list that is executed by the DMA.

The peripheral configuration is also done thanks to a DMA linked-list which writes the peripheral control registers. Several functions can be chained remaining in Stop mode: for instance I2C master transmit, I2C master receive, then again I2C master transmit, I2C master receive. This can be used to perform data acquisitions from different sensors, without any wakeup need to reconfigure the peripheral. Complex use cases involving several peripherals can also be supported remaining in Stop mode, thanks to LPBAM.

*Note: For any update on the drivers, refer to the latest STM32CubeU5 MCU Package available on www.st.com. Several LPBAM examples can be found in the Applications folder, located under the Projects folders of the STM32CubeU5 MCU Package.*

# 5 LPBAM power optimization

Consumption figures provided in this section correspond to a STM32U575xxxxQ device (SMPS selected) with die revision X.

## 5.1 LPBAM in Stop 2 mode: power consumption estimation

A standard LPBAM use case includes usually the following phases:

1. Phase 1 (lowest consumption phase): the device is in Stop 2 mode with SRD in DStop state, waiting for an external or internal trigger event. In case of internal trigger, the peripheral providing the trigger, and its kernel clock source (if any) consumptions must be added.

   Example: Stop 2 with LPTIM clocked by LSI (trigger generated by LPTIM).

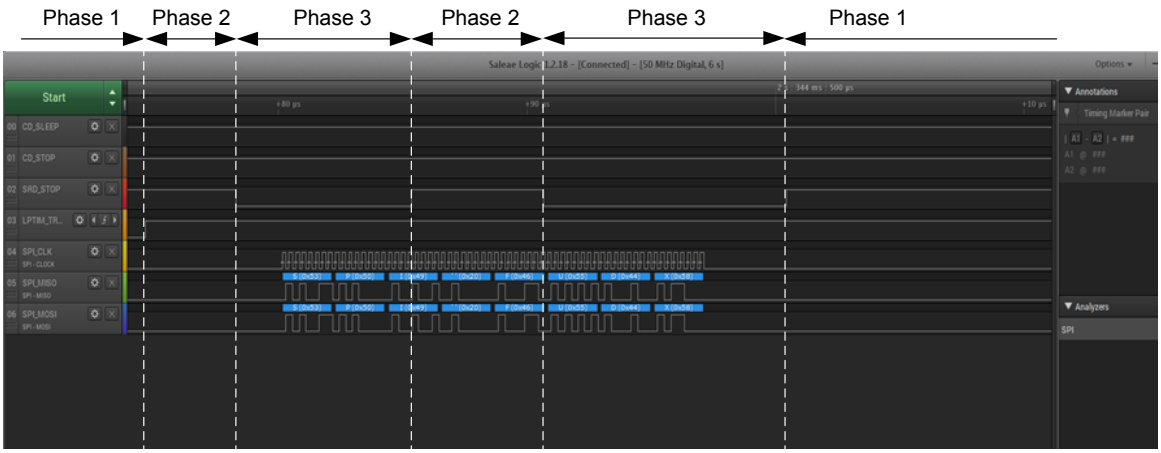
2. Phase 2 (medium consumption phase): the device is in Stop 2 mode with SRD in DStop state, and the trigged peripheral is functional using its kernel clock. This phase is not always present as it depends if the peripheral requests a kernel clock to operate or not.

   Example: Stop 2 with LPTIM clocked by LSI (trigger generated by LPTIM) and SPI master transfer ongoing, using MSIK at 16 MHz.

3. Phase 3 (high consumption phase): the device is in Stop 2 mode with SRD in DRun state (DMA transfer is ongoing).

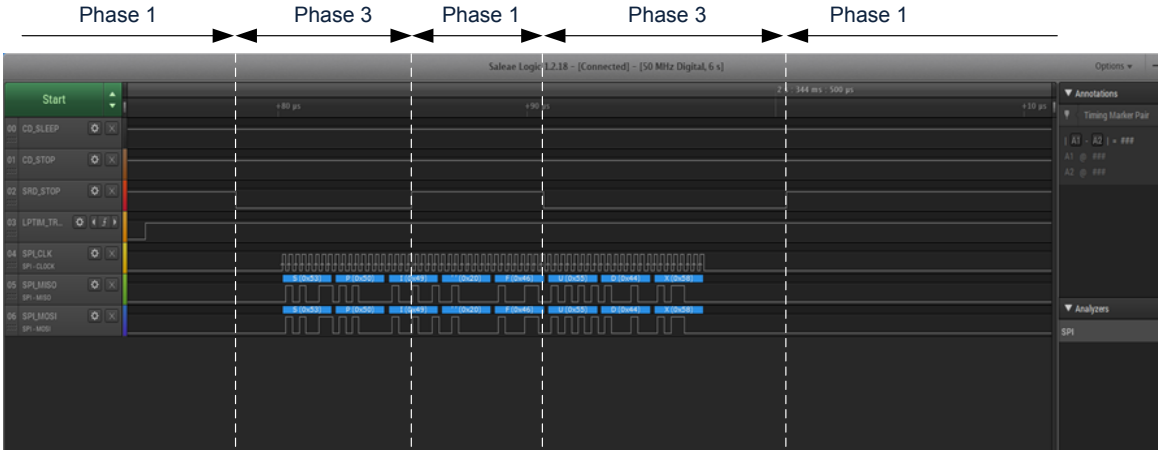
   Example: Stop 2 with LPTIM clocked by LSI (trigger generated by LPTIM), SPI master transfer on going, using MSIK at 16 MHz, and DMA transfer on going, using MSIS also at 16 MHz.

**Figure 7. Consumption phases in LPBAM SPI master use case**



Depending on use cases, only two phases can be present, as shown in the figure below.

**Figure 8. Consumption phases in LPBAM SPI slave use case**




The total consumption depends on the ratio of each phase, versus the total use case period duration.

$$I_{DD} = \frac{I_{DD(phase1)} \times T_{phase1} + I_{DD(phase2)} \times T_{phase2} + I_{DD(phase3)} \times T_{phase3}}{T_S}$$

With

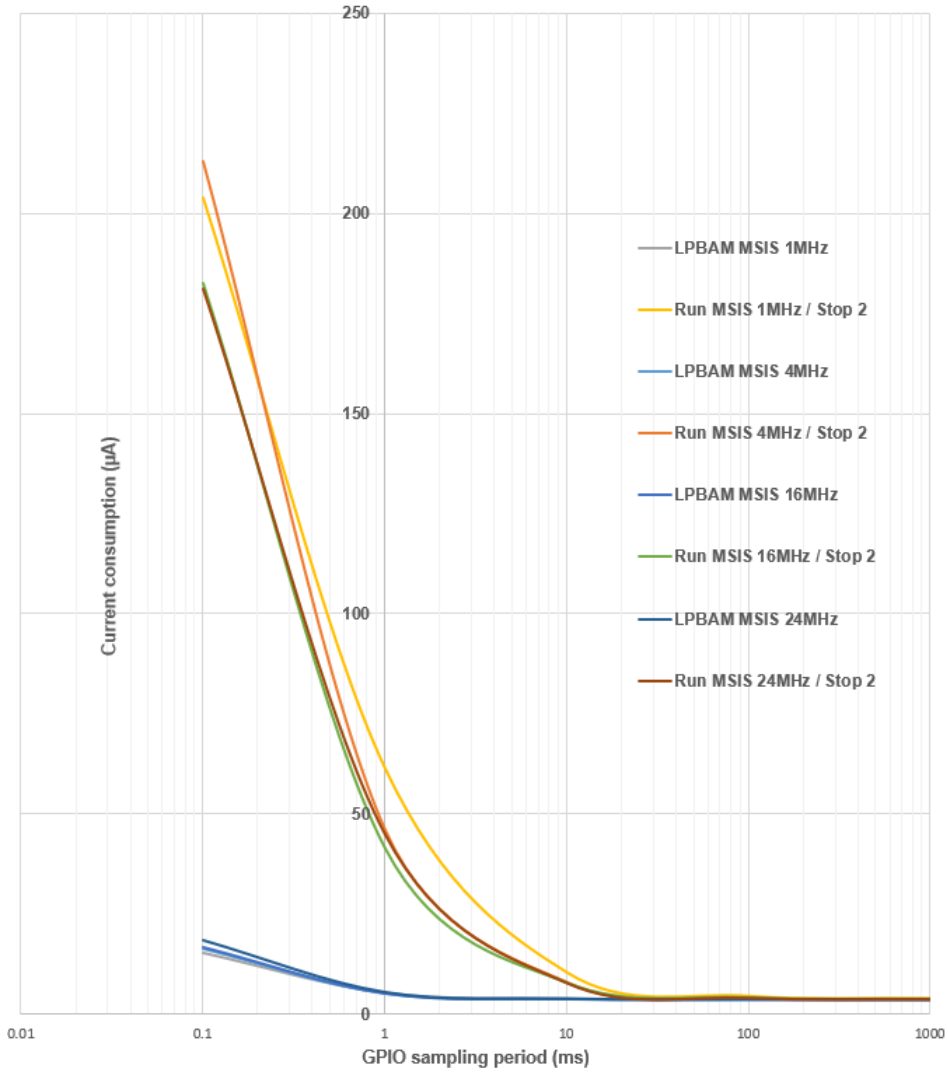
$$T_S = T_{phase1} + T_{phase2} + T_{phase3}$$

### 5.1.1 Impact of DMA transfer rate on average consumption

The curves in the figure below show the impact of the ratio of the Phase 3 (SRD in DRun state for DMA transfer), versus Phase 1 (SRD in DStop state), with the basic example of GPIO input sampling with DMA. A DMA transfer is triggered by LPTIM PWM signal at various frequencies (from 1 Hz to 10 kHz).

The LPTIM is clocked continuously by LSI around 32 kHz.

**Figure 9. LPBAM in Stop 2: GPIO sampling current consumption**



These curves illustrate also the benefit of using LPBAM in Stop 2 mode versus waking up from Stop 2 mode at each PWM interrupt signal to read the GPIO input register. The benefit is much larger when increasing the sampling frequency. The table below gives the consumption saving between LPBAM and firmware with wakeup at different PWM sampling frequencies.

**Table 12. Consumption saving with LPBAM versus Run/Stop 2 (at 4MHz)**

| Sampling frequency | LPBAM consumption saving |
|---|---|
| 10 kHz | 90% |
| 1 kHz | 85% |
| 100 Hz | 50% |
| 10 Hz | 10% |
| 1 Hz | 2% |

Several system clocks are used from 1 MHz to 24 MHz. In LPBAM mode, the system clock frequency has only a weak impact on average consumption, due to the fact the dynamic consumption is quite linear with frequency, and the DRun duration linearly decreases with system clock frequency.

On the contrary, a higher system clock frequency is recommended when wakeup from Stop is needed.

If the sampling must be with faster or higher resolution, the LPTIM can be clocked continuously by MSIK or HSI16 with a large choice of frequencies. The consumption for a same sampling period is higher than when the LSI is used as oscillator frequency is higher. The table below shows consumptions with LPTIM clocked by MSIK 1 MHz instead of LSI.

**Table 13. GPIO sampling current consumption comparison**

| Sampling period | LPTIM clock | Current consumption (µA) | |
|---|---|---|---|
| | | LPBAM MSIS 1 MHz | Run MSIS 1 MHz / Stop 2 |
| No GPIO sampling | LSI | 3.6 | |
| | MSIK 1 MHz | 19.5 | |
| 50 µs(1) | LSI | Not possible | |
| | MSIK 1 MHz | 25 | Not possible |
| 1 ms | LSI | 5.3 | 61 |
| | MSIK 1 MHz | 20 | 74 |

1. *This is the minimum period for which LPBAM is functional with 1 MHz system clock, due to the delay explained in Total data transfer duration from peripheral clock request to end of DMA transfer, in Stop 2 mode.*

### Speed parameters consumption impact

As described in Section 3.2.1 , SRAM4FWU and SRDRUN configuration bits can be used to increase the LPBAM use case speed.

Setting SRAM4FWU increases the average consumption of around 150 nA at 3 V.

Setting SRDRUN obviously increases a lot the average consumption, as the system clock is always distributed in the SRD, to all enabled peripherals. As an example, the table below shows the impact for previous GPIO sampling use case. The impact increases proportionally to system clock frequency.

**Table 14. GPIO sampling current consumption comparison**

| Sampling period | LPTIM clock | Current consumption (µA) LPBAM MSIS 1 MHz | |
|---|---|---|---|
| | | SRDRUN = 0 | SRDRUN = 1 |
| 1 ms | LSI | 5.3 | 23 |

### 5.1.2 Consumption of ADC conversions versus sampling rate

The consumption comparison for ADC conversions use case is detailed below. A single ADC conversion is triggered by LPTIM PWM, and data is transferred by the DMA. The LPTIM is clocked by LSI.

The three phases are present in this use case:

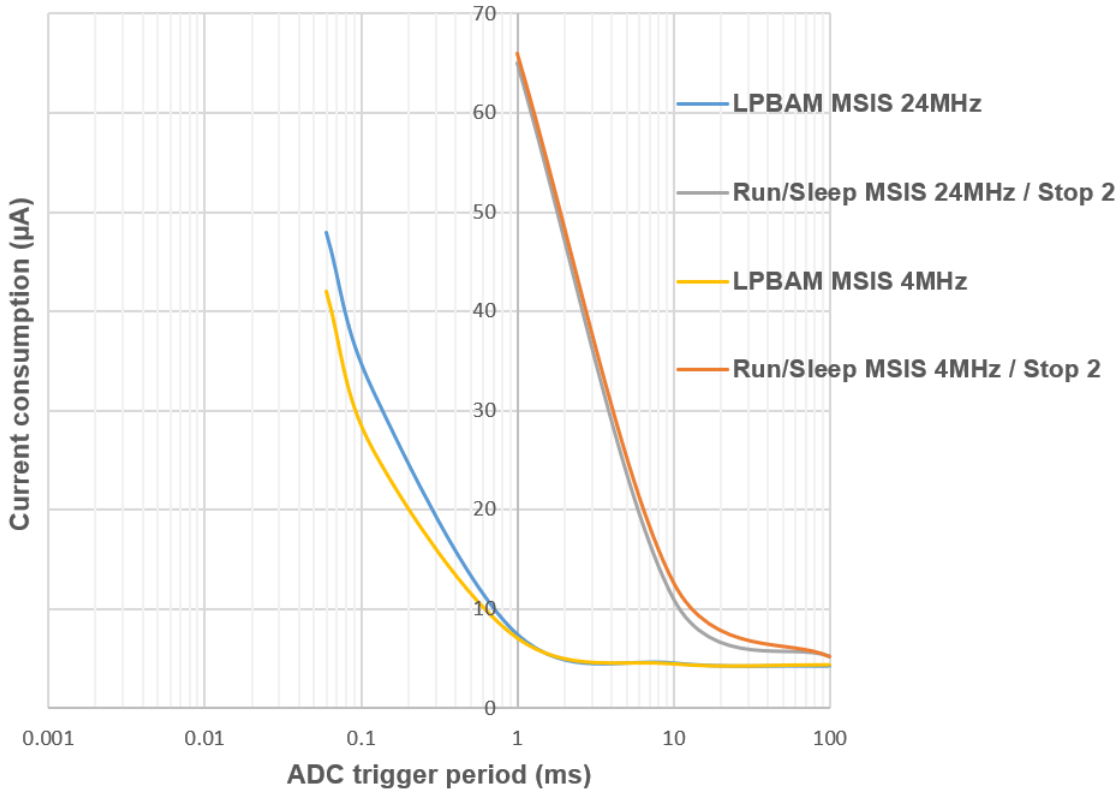
1. Phase 1: The ADC is disabled, and only LPTIM is running in Stop 2 (SRD in DStop).
2. Phase 2: The ADC is running and converting, clocked by MSIK (SRD in DStop).
3. Phase 3: The DMA transfer occurs (SRD in DRun).

In this use case, the Phase 2 consumption has a strong impact due to ADC analog consumption which must be added to the ADC digital interface dynamic consumption, clocked by MSIK. The ADC sampling time is 1.5 cycle, so 14 MSIK cycles are needed for ADC conversion.

The curves in the figure below compare the LPBAM firmware consumption, with other firmware not using LPBAM feature: the ADC conversion is done in Sleep mode. Stop 2 mode is entered between each ADC conversion. LPTIM PWM interrupt is used to wake up from Stop 2 mode. The maximum sampling rate is limited by the wakeup from Stop 2 duration, and cannot be as high as when using LPBAM.

In all tests, the system clock (MSIS) is the same oscillator and frequency than kernel clock (MSIK) to avoid cumulating two oscillators consumption. As demonstrated in Figure 9, the system clock frequency does not impact so much the LPBAM average consumption.

**Figure 10. LPBAM in Stop 2: ADC sampling current consumption**




These curves show that the MSIK frequency has a limited impact on average consumption, due to the fact the ADC conversion duration is always 14 MSIK cycles. The Phase 2 duration varies linearly with kernel clock frequency. MSI and ADC consumptions are not strictly linear with frequency.

The table below gives the consumption saving between LPBAM and firmware, with wakeup at different PWM sampling frequencies.

**Table 15. Consumption saving with LPBAM versus Run/Sleep/Stop 2 (at 24 MHz)**

| Sampling frequency | LPBAM consumption saving |
|---|---|
| 1 kHz | 90% |
| 100 Hz | 60% |
| 10 Hz | 20% |

### 5.1.3 Consumption of SPI master transfers versus sampling rate(number of data and SPI speed)

This section focuses on consumption comparison for SPI master transfers. The SPI transfer is triggered by LPTIM PWM, and data are transferred by the DMA using 32-bit packing (one DMA transfer every 4 bytes). Using 32-bit packing mode is recommended to reach highest SPI frequency for a given system clock frequency, and to reduce the consumption as the system clock is requested four times less often.

The LPTIM is clocked by LSI. SPI kernel clock is MSIK and system clock is MSIS. MSIK and MSIS have the same frequency, equals to twice the SPI clock frequency:

- MSIK = MSIS =16 MHz for SPI 8 MHz
- MSIK = MSIS = 4 MHz for SPI 2 MHz

The following phases are present in this use case:

1. Phase 1: SPI is not transferring, and only LPTIM is running in Stop 2 (SRD in DStop).
2. Phase 2: the SPI master is transferring, clocked by MSIK (SRD in DStop).
3. Phase 3: the DMA transfer occurs (SRD in DRun).

The curves in Figure 11 and Figure 12 compare LPBAM firmware consumption with other firmware not using LPBAM feature: SPI transfer is done in Sleep mode. Stop 2 is entered after SPI transfer. LPTIM PWM interrupt is used to wake up from Stop 2 mode. Another comparison is made with staying in Sleep mode instead of entering Stop 2 mode after SPI transfer: this is interesting only when the system clock frequency is low, and when the Stop 2 duration is too short, so that energy loss during Stop 2 wakeup penalizes the average consumption.

In all tests, the system clock (MSIS) is the same oscillator and frequency than kernel clock (MSIK) to avoid cumulating two oscillators consumption. As demonstrated in Figure 9, the system clock frequency does not impact so much the LPBAM average consumption.

**Figure 11. LPBAM in Stop 2: SPI master 8-byte transfer**

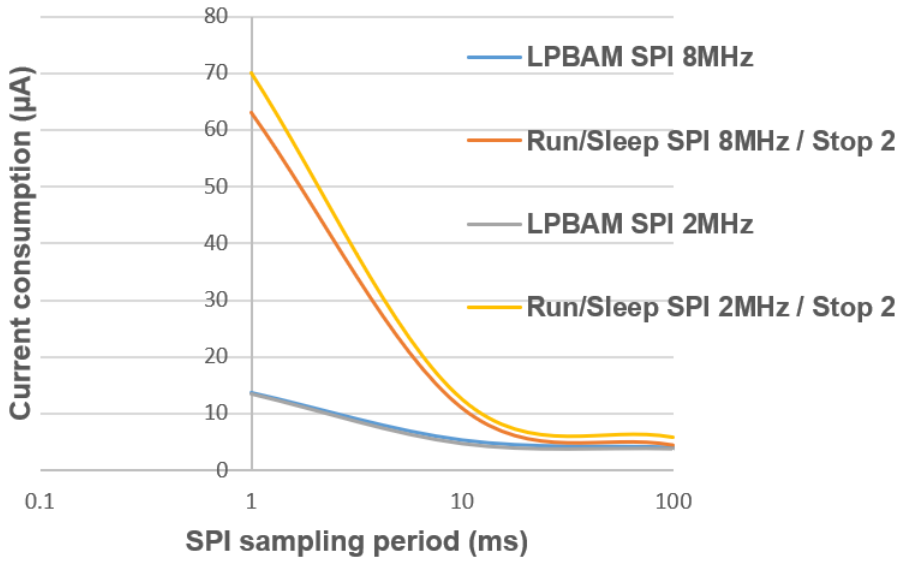
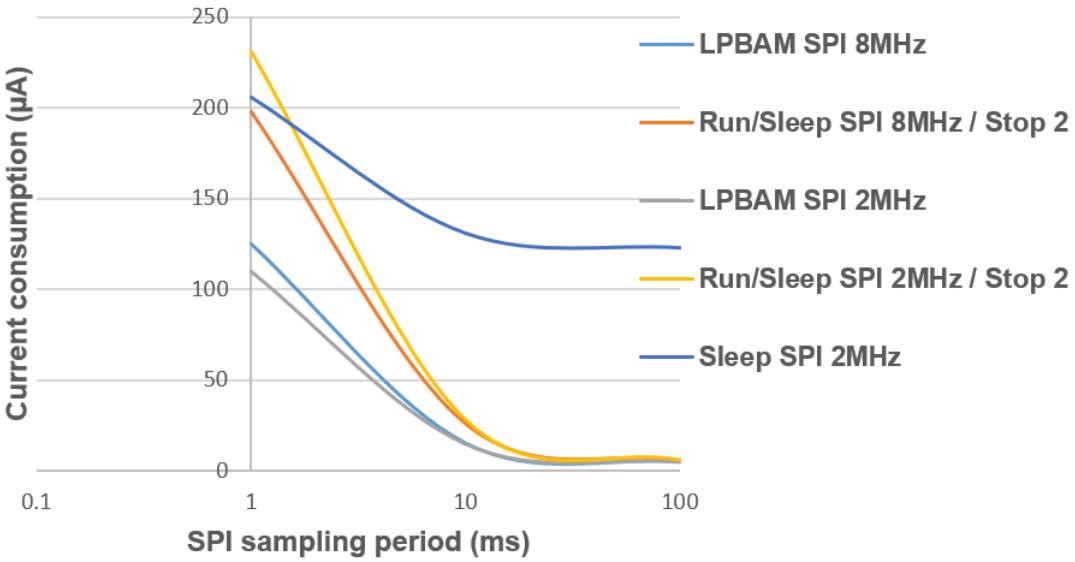


Figure 12. **LPBAM in Stop 2: SPI master 128-byte transfer**




The table below shows the consumption saving between LPBAM and firmware with wakeup at different PWM sampling frequencies.

Table 16. **Consumption saving with LPBAM versus Run/Sleep/Stop 2 modes**

| Sampling frequency | LPBAM consumption saving SPI master 8 MHz | | LPBAM consumption saving SPI master 2 MHz | |
|---|---|---|---|---|
| | 8 bytes | 128 bytes | 8 bytes | 128 bytes |
| 1 kHz | 80% | 35% | 80% | 70% |
| 100 Hz | 50% | 40% | 60% | 45% |
| 10 Hz | 10% | 10% | 30% | 15% |

The curves in Figure 11 and Figure 12 show also the benefit of reducing the clock speed to reduce consumption, when the number of data increases. This is due to the Phase 2 ratio which increases. The Phase 2 consumption is linear with kernel clock frequency. The table below shows the benefit of reducing kernel clock frequency when increasing the number of data, and when the SPI transfer is done continuously (no trigger).

Table 17. **LPBAM consumption saving SPI 2 MHz versus SPI 16 MHz**

| Sampling frequency | 8 bytes | 128 bytes | Continuous |
|---|---|---|---|
| continuous | - | - | 70%(1) |
| 1 kHz | Equivalent | 40% | - |
| 100 Hz | | Equivalent | |
| 10 Hz | | | |

1. *Using LPBAM for Continuous SPI transfer rather than Sleep mode, saves around 15 to 30% consumption*

### 5.1.4 Consumption model of other peripherals

The consumption comparison made for GPIO sampling use case is valid also for peripherals that are clocked only by LSE/LSI, or which do not need any kernel clock, such as:

- SPI slave transfer (no need of any kernel clock)
- LPUART transfer when clocked by LSE
- LPTIM DMA transfers
- ADF DMA transfers

For all these use cases, there are only two phases (see Section 5.1 ):

- Phase 1 and Phase 3 for SPI slave (no kernel clock)
- Phase 2 and Phase 3 for the other peripherals listed above

The consumption is in the range of the one evaluated in Section 5.1.1 when the peripheral kernel clock is LSE or LSI, or when there is no kernel clock.

For the following use cases, the three phases are present:

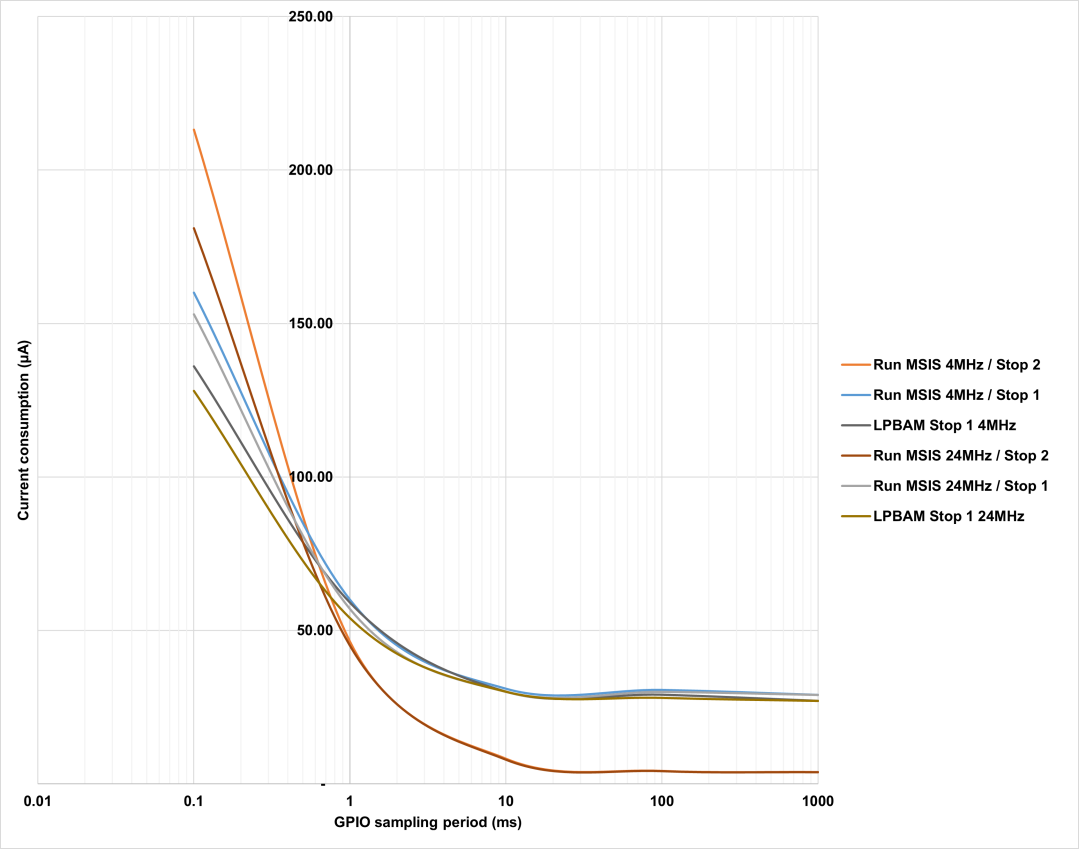
- SPI master transfer
- LPUART transfer when clocked by MSIK or HSI16
- I2C transfer: the consumption through pull-ups on SDA and SCL lines must also be added.
- ADC conversion: the analog converter consumption must be added during conversion.
- DAC conversion: the analog converter consumption must be added during conversion, and the sample and hold mode consumption between each transfer.

## 5.2 LPBAM in Stop 1 mode

The LPBAM in Stop 1 mode allows the use of more peripherals, but the consumption benefits is limited as Stop 1 consumption is much more higher than Stop 2 consumption.

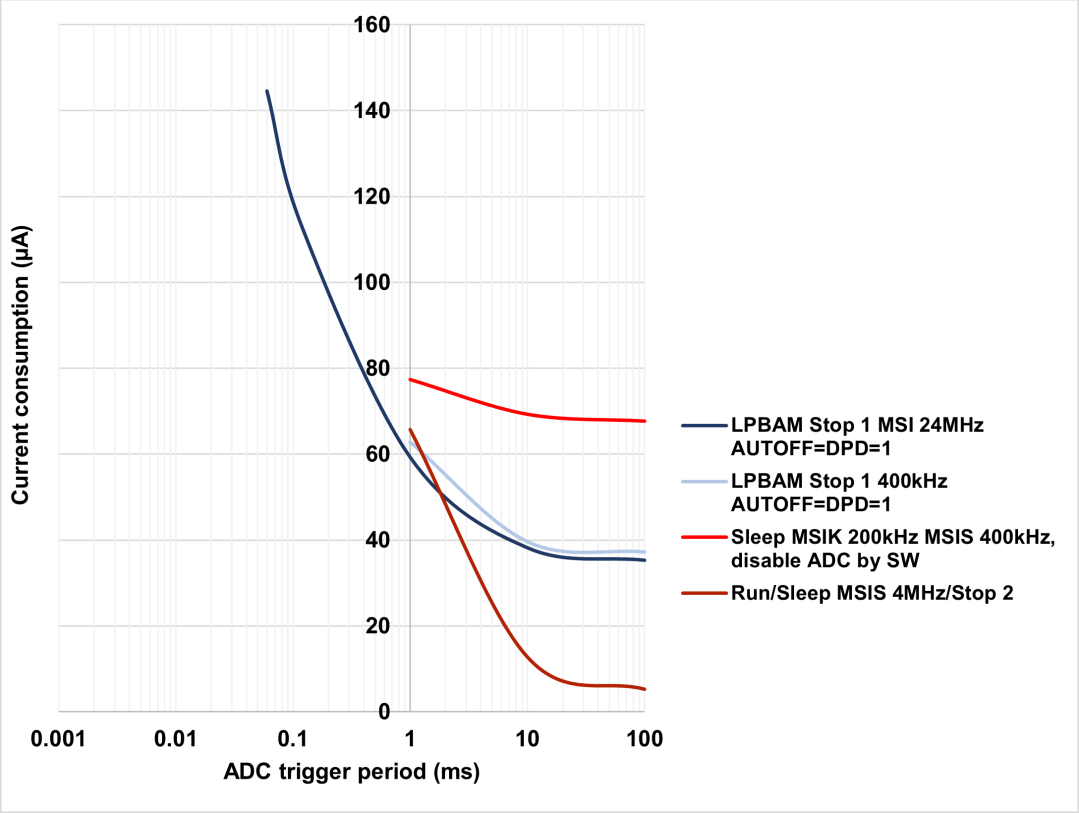
The figure below shows that for a GPIO sampling slower than 2 kHz, it is recommended to enter Stop 2 mode and wakeup to Run mode, rather than using LPBAM in Stop 1 mode.

**Figure 13. LPBAM in Stop 1: GPIO sampling current**



Similarly, the ADC sampling example shows that the LPBAM in Stop 1 mode saves consumption compared to Sleep mode or Stop 2 mode with wakeup, only when the sampling frequency is faster than 500 Hz.

**Figure 14. LPBAM in Stop 1: ADC sampling**



Next figures show the SPI master use cases. In these use cases, it is always recommended to perform the SPI master transfer in Sleep mode and then enter Stop 2 mode, rather than using LPBAM in Stop 1 mode. The consumption measures are done using SPI in CPU domain with GPDMA1. At higher sampling frequency with 128 bytes transfer or more, it is also recommended to stay in Sleep mode rather than entering Stop 2 mode, due to the energy lost with Stop 2 wakeup.

**Figure 15. LPBAM in Stop 1: SPI master 8-byte transfer**

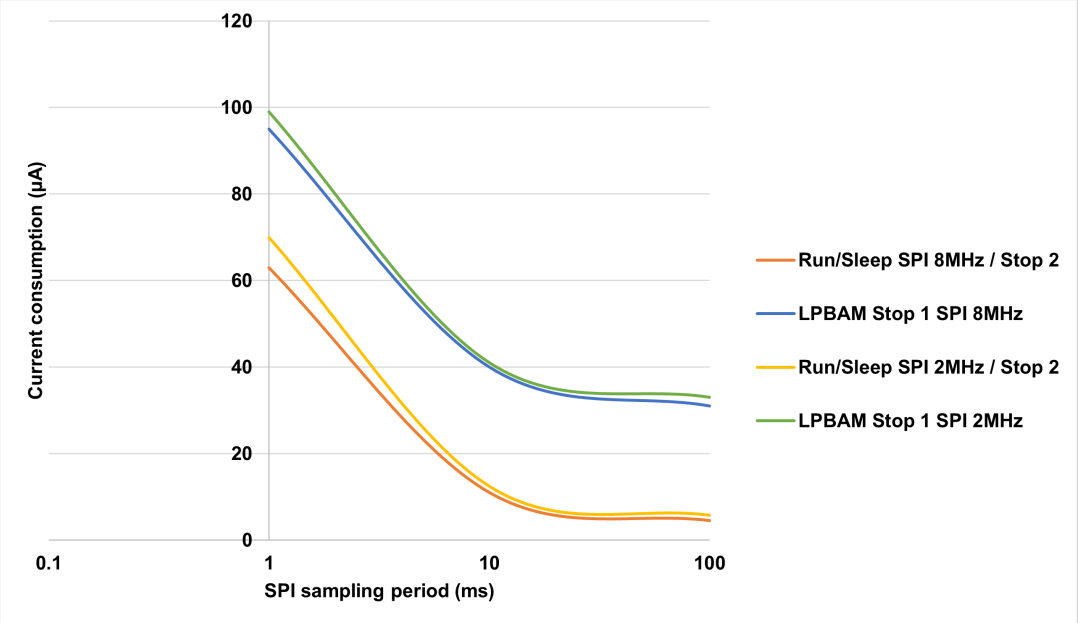
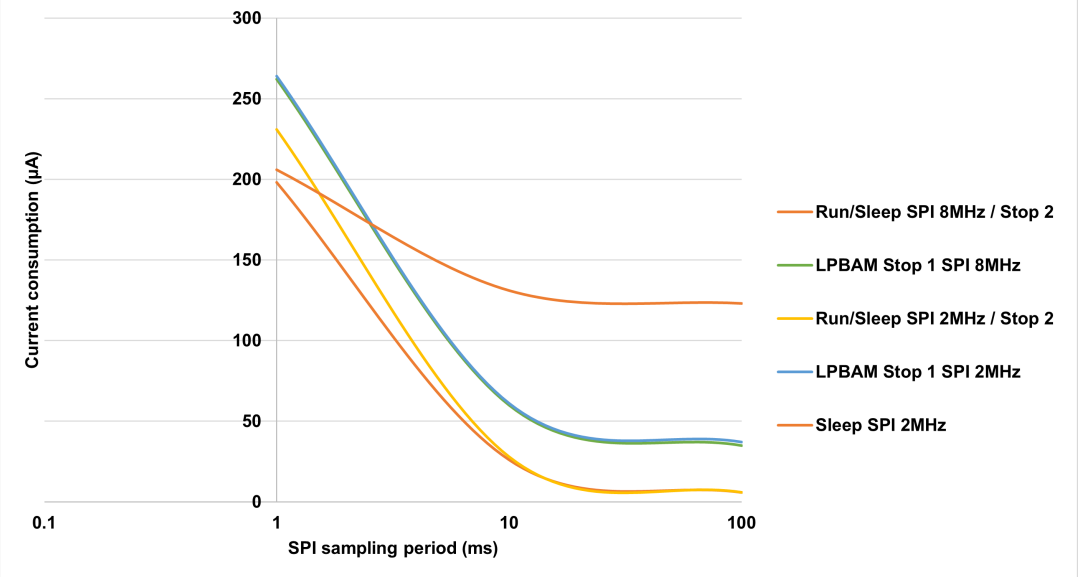



**Figure 16. LPBAM in Stop 1: SPI master 128-byte transfer**




All these examples show that LPBAM in Stop 1 can save power consumption in a more limited range of sampling frequencies, compared to traditional firmware using Run, Sleep and Stop modes Each use case must be studied to evaluate the potential benefits of using LPBAM in Stop 1 mode.

However this feature can be helpful to reach higher sampling frequencies by saving the need to serve an interrupt, as all is done automatically by the hardware.

## 5.3 General rules to optimize consumption using LPBAM

Based on consumption analysis of use cases in previous sections, some general rules can be extracted:

- LPBAM consumption saving increases with sampling frequency (this one is limited by the functional speed limits explained in Total data transfer duration from peripheral clock request to end of DMA transfer, in Stop 2 mode). LPBAM in Stop 2 proves large consumption benefits in the 50 Hz to 10 kHz sampling range.
- To evaluate the consumption, the use case can be split into 2 or 3 phases as described in Section 5.1 .
  - Phase 1 with lowest consumption.
  - Phase 2 consumption impact is directly linked to the used kernel clock, and to this phase ratio compared to the use case period. For consumption point of view, prefer LSE or LSI when supported as peripheral kernel clock, then prefer MSIK with lowest possible frequency.
  - Phase 3 consumption impact is directly linked to the DMA transfer number compared to the use case period. For consumption optimization, the wakeup from Stop clock must select the same oscillator than the peripheral kernel clock, to avoid adding consumption of a second oscillator during this phase.

In addition to these rules, any Stop mode power optimization described in document [4] can be used:

- Configure unused I/Os in analog mode.
- Select oscillators and analog peripherals in the lowest power mode allowed by the application.
- Power down the SRAM for which retention is not needed.
- Enable only peripherals that are used in LPBAM scenario (including RCC).
- Select the wakeup from Stop oscillators depending on the application consumption and wakeup time tradeoff.

Refer to document [4] for more details.

# 6 Conclusion

This application note describes use cases supported thanks to LPBAM, demonstrates the benefits of using LPBAM in Stop 2 mode and describes the different options to optimize the power consumption when using peripherals.

The LPBAM can be used in Stop 0 or Stop 1 mode with a larger number of peripherals, but the consumption saving is much less significant than when using the LPBAM in Stop 2 mode.

More complex use cases can be built without any wakeup need, thanks to the large choice of peripheral hardware triggers, the four LPDMA1 channels, and the linked-list capability. The DMA linked-list can be used to modify a peripheral configuration, removing the need to wake up the CPU just for a configuration change.

# Revision history

Table 18. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 13-Dec-2021 | 1 | Initial release. |

# Contents

# List of tables

# List of figures