AN4635
Application note

Minimization of power consumption using LPUART for STM32 microcontrollers

Introduction

STM32 microcontrollers listed in *Table 1* feature an alternative UART (universal asynchronous receiver transmitter) interface, enabling them to operate with minimum power requirements.

This document explains how to fully exploit the advantages of the low-power UART (LPUART), thus extending product battery life.

It shows in practical examples the extremely low-power consumption of the device waiting for a communication. The code used to perform the measurements described in *Section 6.1* and *Section 6.2* is supplied in the package X-CUBE-LPUART, and can be downloaded from *www.st.com*.

The following documents (available on *www.st.com*) must be considered as reference:

- *STM32L0xx ultra-low-power features overview* (AN4445)
- *STM32L4xx ultra-low-power features overview* (AN4621)
- *Optimizing power and performances with STM32L4xx (*AN4746)
- *STM32L4x6 advanced Arm®-based 32-bit MCUs* (RM0351)
- *Ultra-low-power STM32L0x2 advanced Arm®-based 32-bit MCUs* (RM0376)
- *STM32H7x3 advanced Arm®-based 32-bit MCUs* (RM0433)
- *Multiprotocol wireless 32-bit MCU Arm®-based Cortex®-M4 with FPU, Bluetooth® Low Energy and 802.15.4 radio solution* (RM0434)
- *STM32G4xx advanced Arm®-based 32-bit MCUs* (RM0440)
- *STM32G0x1 advanced Arm®-based 32-bit MCUs* (RM0444)
- *STM32L4Rxxx and STM32L4Sxxx advanced Arm®-based 32-bit MCUs* (RM0432)
- *STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs* (RM0438)

Table 1. Applicable products and software

Type	Series, line or part number
Microcontrollers	STM32G0 and STM32G4 Series
	STM32H7 Series
	STM32L0, STM32L4, STM32L4+ and STM32L5 Series
	STM32Wx5 line
STM32 Embedded Software	X-CUBE-LPUART

Contents

List of tables

List of figures

1 Definitions

Table 2. List of acronyms

Term	Description
LSE	Low-speed external clock
LSI	Low-speed internal clock
HSE	High-speed external clock
HSI, HSI16	High-speed internal clocks
MSI	Multispeed internal clock source
UART	Universal asynchronous receiver transmitter
LPUART	Low-power UART
MCU	Microcontroller
USART	Universal synchronous and asynchronous receiver transmitter
BLE	Bluetooth® Low Energy
CPU	Central processing unit (part of the MCU)
NVIC	Nested vector interrupt controller
DMA	Direct memory access
TC	Transmission complete
RF	Radio frequency
RM	Reference manual
SWD	Serial wire debug interface

2 Summary of features

While the LPUART peripheral is practically the same, there are significant differences in its integration on the MCUs addressed by this document. They feature different Arm® Cortex® cores[a], and there are other architectural differences impacting the LPUART efficiency.

2.1 Clock subsystem

2.1.1 MSI clock

This is an internal oscillator capable of fast, simple and seamless change in operating frequency of low-power oriented devices. As the maximum frequency is different for various MCU, so are the MSI ranges This gearing up of the MSI reduces the choice of low speeds, the slowest possible frequency on STM32WB, STM32L4 and STM32L4+ is 100 kHz vs. 65 kHz on STM32L0xx MCUs.

For speeds closer to 1 MHz a direct comparison of efficiency is difficult. In the STM32L4, STM32L4+, STM32L5 and STM32WB Series the MSI can use the HW auto calibration with LSE in its PLL-mode. This makes the MSI a lot more precise, no such option is available on STM32L0 products.

The MSI clock is not implemented on STM32G0, STM32G4 and STM32H7 Series, not intended as low-power devices.

2.1.2 HSI clock

The STM32L0 Series features a simple clock factor-4 divider associated with the HSI clock source, making the HSI the effective source of either the 16 or the 4 MHz. MCUs of other low-power Series do not have a divider directly on the HSI16 clock. As a result, the STM32L0 is much more efficient in certain applications requiring UART speeds higher than 9600 Bd. If (for STM32WB, STM32L4, STM32L4+ and STM32L5 Series) 16 MHz is not efficient, the solution is to use a second source for system clock.

STM32G0 and STM32G4 Series MCUs feature an even more flexible divider, allowing to set the two dividers in the 1-512 range using the HPRE bits in the RCC_CFGR register.

STM32H7 Series MCUs have a similar flexible setup consisting of two cascaded dividers, but with oscillator having 64 MHz nominal frequency.

2.1.3 Low speed clock sources

LSI clock frequency also differs, but this is not relevant to our case.

arm

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

2.1.4 CSI clock

This is a low-power oscillator integrated on STM32H7 devices. Its nominal frequency (4 MHz) is sufficient for most of the low-power tasks.

2.2 Power management

The main regulator on STM32L4, STM32L4+, STM32G0, STM32G4 and STM32WB Series has only two ranges, vs. the three available on the STM32L0xx MCUs. Moreover, these two ranges are shifted towards higher frequencies, supported by the more powerful MCU.

This deficiency is compensated by a the Low-power run mode.

STM32L0xx MCUs cannot return to the Low-power run mode directly upon waking from Sleep or Stop Low-power mode, this limitation is not present in the Series featuring a Low-power mode. Low-power run on STM32L4, STM32L4+, STM32G0, STM32G4 and STM32WB Series is not limited by the MSI range 1, but works up to 2 MHz system clock speed. HSI16 can still be used as peripheral clock, even in Low-power run.

A single voltage regulator with programmable ranges is available on STM32H7 devices.

2.3 Comparison with USART peripheral

The LPUART has less features compared to the USART, however it is able to operate using less power and using the LSE clock more efficiently.

The main features for the two peripherals are summarized in *Table 3*.

Table 3. Comparison of features

Configuration	LPUART	USART
LSE clocked 9600 baud option	+	-
Synchronous mode	-	+
Ir SIR compatibility	-	+
Smartcard mode	-	+
Auto baud rate detection	-	+
Modbus communication	-	+
LIN mode	-	+

The USART is also able to operate using LSE clock, the communication speed is limited to 4000 baud in case of oversampling by 8, to 2000 baud when oversampling is by 16.

When using the LPUART, only the 32.768 kHz LSE clock is required for serial communication up to 9600 baud, with minimal energy consumption and very precise speed setting made possible by the external crystal.

For higher speeds the LPUART energy efficiency advantage drops, but still remains close to 5% (see *Section 6.1.9: Higher communication speed*).

This document focuses on communication at 9600 baud, common to many applications: the efficiency advantage of LPUART is obvious when using wakeup from Stop at the 9600 baud speed setting.

3 Operation modes

Real-world scenarios may cover a wide variety of configurations, using different baud rates, ratios of transmit/receive and delays between messages. All these factors influence the choice of the operation mode.

3.1 Polling mode

Polling mode (also known as blocking mode) is the simplest possible operating mode. The CPU is processing a single task, switching to a low-power mode during periods of inactivity. There is almost no processing overhead, making it possible to use very low system clock speeds.

This mode is extremely efficient for very simple scenarios, however it effectively blocks the CPU from executing other tasks, like data processing or concurrent communication.

3.2 IT mode

The second option is to rely completely on interrupts, waking the CPU for every transferred byte. This operation mode uses the advantage of Arm® Cortex® advanced NVIC to keep the processing split to atomic operations, never blocking the CPU and achieving real-time responses.

This mode however strains the CPU a little more, adding processing overhead related to stack and context resuming.

3.3 DMA mode

In DMA mode the CPU is spared a large portion of processing, setting a DMA channel to move data between the peripheral and the SRAM. The CPU can spend part of the processing time in Sleep mode. The user can disable half-buffer interrupt when not needed for circular buffer management, to let the CPU core rest even longer. In most of the cases the DMA cannot be used in combination with the Stop mode, hence all DMA channels have to be disabled before entering the Stop Low-power mode. STM32H7 MCUs are an exception to this rule, as detailed in *Section 6.3: Measurement on the STM32H743 Nucleo144 board*.

The LPUART transfer can be done thanks to the batch acquisition mode (BAM) in which the MCU is in Sleep or in Low-power sleep mode (CPU is clocked off). The consumption is optimized by configuring the Flash memory in Power-down mode and switching off its clock, and by clocking only the DMA, the LPUART and the SRAM.

The DMA has one definitive advantage over the previous two methods: the transfer to memory is handled independently by the CPU load and the interrupt is only triggered once the data is stored in RAM for the firmware to process the following action. With interrupt and polling mode the firmware must read the data from the peripheral register, and with wrong timing some data may be lost in overrun error.

3.4 Combined mode examples

Real-world applications usually are a mix of the described modes, the developers always try to strike the best bargain between conflicting needs.The following examples are not covered in bundled source codes.

3.4.1 Interrupt with polling

Some embedded systems are not real-time critical, in this case there is an option to block the CPU for a limited time to process a message frame. Especially in case of transmission, energy normally used to process all TC interrupts is saved, the message is transmitted in blocking mode, possibly with clock speed lowered and power regulator switched to Low-power mode (LPRUN). Only approximately half clock speed is required if the CPU can treat the message in blocking mode. Normal operation is then restored, preferably waiting for incoming reply interrupt in Stop mode.

3.4.2 Combining DMA with direct access

The DMA channel is convenient for transmission of data, and is power-efficient during reception.

The downside is that in DMA mode the LPUART cannot take advantage of wakeup from Stop mode. The reason is that after wakeup event the DMA has difficulty picking up on the ongoing communication. This is a serious disadvantage for all applications that tend to remain idle for long periods of time.

The DMA can still be used for transmission and then reception can proceed with blocking or interrupt approach. In communication systems where incoming messages are coming in quick succession or predictable timing, the DMA based reception is an efficient option.

4 Other considerations

For a complete overview of low-power advantages refer to the already mentioned AN4445, AN4621 and AN4746. The following recommendations are specific for our case and example.

4.1 Execution from SRAM

If the program can be executed from SRAM, there is an option to turn off the embedded Flash memory (by clock gating), further reducing the power consumption.

4.2 GPIO configuration

Some GPIO settings may have great influence on power consumption while others do not.

The pins used for the UART communication lines should be configured to their alternate function mode. It is not recommended to activate pull-ups if lowest possible power consumption is the most important goal, however, in some applications it may be necessary to improve the communication reliability. The speed setting has no consequences regarding the power consumption on the tested baud rates.

Half-duplex mode, where applicable, may also lead to further power savings, however this configuration is not addressed in this document.

Other pins, not used by the application, must be configured as analog inputs. The developer has to put the debug lines also to analog, once the application is ready for deployment.

4.3 Clock configuration

For LPUART peripheral clock, LSE source is the choice to reach 9600 baud with wakeup from Stop mode.

HSI/CSI is recommended for higher speeds.

The obvious (and default) choice for system clock is usually the MSI oscillator. On STM32L4, STM32L4+ and STM32WB Series the MSI offers even more flexibility and its fluctuations can be corrected using HW auto calibration with the LSE in its PLL-mode.

Prescalers and PLL can be used to derive other speeds, different options are analyzed later in this document. These solutions are not the best ones when power consumption is the first concern. User must refer to the product datasheet for typical power consumptions for different clock configurations.

It makes sense to use a single high speed clock for system and peripherals, as each high speed oscillator contributes to power consumption. On the other hand, it is advantageous to run the low speed clock in parallel to the system clock, and use it for the peripherals able to work at that frequency.

Table 4. Clock options

Source	Advantages	Limitations
HSI/HSI16	– Ready to use high speed clock with precise trimming	– Worst MHz / W ratio option – Fixed speed on most of STM32 MCUs
MSI	– Easy and fast clock throttling – Lowest overall power consumption (on STM32L0 Series when clocked below 1 MHz)	– Only available on low-power MCUs – Clock is relatively unstable and imprecise (unless the auto-calibration using LSE in PLL-mode is used)
HSE	– Potentially most power efficient at 1 MHz and above on STM32L0 Series – Efficiency comparable to MSI on STM32L4 Series	– Needs additional external components(1) – Limited options of speed control (prescalers only)

1. HSE is used by the RF block in STM32WB Series MCUs, so the components are needed anyway.

4.3.1 Clock prescalers

The RCC module offers the possibility to tune down peripheral bus clocks, the AHB and APB frequency (RCC_CFGR register).

Slowing the AHB brings significant power savings if there is no other clock source having a suitable frequency. The memory interface is also clocked with AHB. While power consumption with prescaler of 8 drops by approximately 50%, the processing capability of the system drops even more due to delays in fetching instructions from the program memory and storing the data in the memory. If possible, it is better to slow down the system clock rather than set an AHB prescaler (*Section 6.1.6: Changing AHB divider ratio*).

Slowing down the APB limits the bandwidth between the LPUART peripheral and the core. This is usually not a problem since the amount of data transferred is low. Especially in case of DMA transfers it is not an issue. In case of CPU driven transfers the core may be stalled waiting for the bus transfer to complete. This causes the overall energy budget for the operation to increase in extreme cases. Also, if the bus transfer is not complete, the core is prevented from going to a low-power mode such as Stop or Sleep modes. In case of LPUART example coming along with this AN it is usually safe to set a APB prescaler value of 4 but higher values are likely to cause problems and provide no power consumption advantage (*Section 6.1.7: Different peripheral clock settings*).

4.4 Power configuration

This section deals with practical implications when configuring power modes for LPUART communication. For complete information read the dedicated sections in the cited reference manuals.

4.4.1 Use of Stop and Sleep modes

It is a paradigm in embedded software that during the idle time the CPU must not run and actively check flags, but rather switch to Low-power mode, suspend clock, and only resume operation on external interrupts or events.

In case of Arm® MCUs this is achieved by executing WFE or WFI instruction. All STM32 microcontrollers offer highly configurable selection of Idle modes, described in detail in the

reference manuals. The great advantage of LPUART is the ability to take advantage of Stop mode when waiting for message reception (*Section 6.1.1: Stop vs. Sleep mode*).

If the application running on STM32L0xx products does not use VREFINT frequently and spends most of the time in low-power modes, configure Ultra-low-power mode with fast wakeup (ULP and FWU bit in PWR configuration register). Even in applications that use internal voltage reference it may be interesting to switch it off and only check the startup time when a measurement is needed (keep in mind that VREFINT startup requires some additional energy). For short periods of Low-power mode (i.e. Sleep mode between bytes typically) there is no point turning it off and on again, the overall energy budget will increase.

Note: Ultra-low-power mode without fast wakeup is unable to keep the LPUART Tx register fed, or to keep up with incoming data in blocking and interrupt mode, even with 9600 baud communication speed.

There is no option to completely switch off the VREFINT on the other Series addressed by this document.

In most cases it is also desirable to activate the Sleep mode between processing individual bytes. In case or data reception even Stop mode is available (*Section 6.1.4: Going to Stop between received bytes*). This practice offers great advantage at higher clock speeds, but at core clock speeds near the bare minimum needed for communication it is not recommended. The overhead related to putting device to Sleep mode and waking it causes a slight increase of power consumption which is then not balanced by the very short time spent in Low-power mode (*Section 6.1.2: Short periods of Sleep mode and Low-power run*).

4.4.2 Run time configuration

The voltage regulator settings are very important. Use the core voltage scaling as much as possible (see dedicated sections in datasheets and reference manuals). As an example, on STM32L0xx switching from Range 1 to Range 3 is an easy way to lower typical consumed power by more than 25% (*Section 6.1.11: Voltage regulator settings*).

Other low-power Series (except STM32L0) provide only two ranges of the main regulator, however all these Series feature a Low-power run mode.

As an example, on STM32L0xx MCUs in polling mode, which requires the least CPU processing resources, it is possible to take advantage of the Low-power run mode. This mode essentially bypasses the main voltage regulator and powers the core with power consumption approximately another 25% lower than the Range 3 of the main power regulator can achieve (*Section 6.1.2: Short periods of Sleep mode and Low-power run*).

On STM32L0xx MCUs waking from Stop or Sleep mode brings the main regulator back on, and because of the current low clock speed (MSI Range 1 at most), the core must immediately dedicate all processing power to data reception. In practical terms the LPRUN is great for transmission phase, but when waiting for data reception it is usually better to take advantage of the Stop mode instead of keeping the CPU in LPRUN. It is however possible to put the main voltage regulator temporarily to Range 3 (it must be in Range 2 when switch to LPRUN is done). Changing configuration of main regulator is quicker than switching the regulator and less likely to cause dropped bytes in incoming message – wakeup from Stop and power configuration changes must be handled in time shorter than 1 byte duration.

On STM32L4, STM32L4+ and STM32L5 Series the LPRUN mode provides more functionality. It works up to 2 MHz, which provides enough processing power for 57600 Bd

communication speed in interrupt mode. It is even possible to wake from Stop or Sleep directly to Low-power run, without waking the main regulator.

The STM32WB regulator power scaling is identical to that of STM32L4 and STM32L5, but, because of the second core, there are more options, exceeding the scope of this document.

The STM32G4 and STM32G0 also have two base power scaling ranges, with Range 2 available up to 26 and 16 MHz, respectively.

4.4.3 STM32H7 core domain sections

The use of the Cortex®-M7 core needs special care to optimize power consumption.

The core domain is split in three sections, as detailed in already cited RM0433. For LPUART operation it is important to know that basic DMA and low power peripherals are within the D3 domain.

4.4.4 SMPS

Some STM32L4+, STM32L5 and STM32WB products feature an integrated SMPS, a power supply option that leads to increased efficiency, especially if the available V_{DD} branch of the PCB is relatively high voltage (like 3.3 V).

A detailed explanation of the usage of the embedded SMPS is out of the scope of this document.

4.5 Cache memory

Several STM32 microcontrollers feature a cache memory. Having the instructions or data available in the cache instead of waiting for fetch from system memory improves execution efficiency and subsequently also power consumption. Refer to “*STM32 power mode examples*” (AN4777), available on *www.st.com*, to learn more about different cache configurations.

5 Reliability and communication quality

5.1 Noise and frequency shift

Line noise should cause no issues at the low communication speed typical for low-power applications. In some cases, random noise can be mitigated by adding a weak pull-up. This is of course at expense of additional energy. For example the internal pull-up (no extra parts) increases power consumption by approximately 30 µA during transmission in the example described in *Section 7*.

Framing errors occur as result of incorrect base frequency (speed setting) at one of the communication participants. The LSE clock source, which is best suited for low-power applications, is very accurate. Only if LPUART peripheral clock is derived from MSI clock then the framing errors are very likely. The LPUART also cannot auto detect the uncalibrated communication speed of its counterpart, relying only on the precise speed settings with framing reliability.

5.2 Dropped bytes

The main means of achieving lower power consumption is to tune down core frequency and use low-power modes. This imposes a challenge of waking up and responding to peripheral events.

In transmission case this may cause delay between bytes sent, leading to prolonged period of activity and thus impact on system efficiency, but no data loss.

The true problem is data reception. In polling or interrupt driven mode, if the received data in the LPUART data register is not read in time i.e. before a new data is received, an overrun error occurs and any data received during overrun is lost.

To avoid missing data in reception, the following guidelines can be provided:

- ensure a correct clock speed (not below a certain minimum) for the core
- boost the LPUART interrupts priorities to the highest possible level in the application to make them basically uninterruptible
- use the DMA for data transfers.

6 Power consumption comparison

All the pictures in this section show the detailed measurement of differently configured Low-power Nucleo running the following routine, graphically illustrated in *Figure 1*:

1. Run mode in waiting loop – showing the regular power consumption.
2. Transmission of 100 byte message.
3. Idle state 10 ms: Stop or Sleep mode waiting for reply (depending on configuration).
4. Reception of the same 100 bytes.
5. Compare the message to detect communication errors and go back to *1*.

Figure 1. Test loop description

The device starts running full speed in the loop to showcase the regular power consumption on its clock configuration. This is followed by approximately 100 ms of 9600 baud transmission, then 10 ms of idle state, waiting for incoming reply. Only few measurements feature the higher communication speed of 57600, where the communication phases 2 and 4 are significantly shorter. Receiving the reply then takes another 100 ms.

Both the transmission and reception usually consumes less power than the regular run mode, as the device is configured to save power in short periods between single bytes of the message.

Measurement detailed in the following pages are an example of how the X-CUBE-LPUART Expansion Package can be used: it does not cover all the MCU with LPUART and it does not compare all their configurations, but points out interesting features and shows some available solutions.

Note: All measurement have been performed at temperatures around 25°C on Nucleo boards, they are in line with datasheet typical values. It is however not guaranteed that every single MCU will reproduce the very same values.

6.1 Measurements on the STM32L053 Nucleo board

6.1.1 Stop vs. Sleep mode

The first comparison deals with the difference between the Sleep and Stop modes.

While waiting for the incoming communication, the device tries to minimize the power consumption. The black line in *Figure 2* represents debug configuration with device waiting in Sleep mode. Using the LSE clock and LPUART peripheral the Stop mode can be employed up to speed of 9600 baud, this is the green line. Finally, with purple line one can see the absolute minimum current absorbed in Stop mode where the debug lines are configured as analog input and the device is using the ULP + FWU combination.

Figure 2. Stop vs. Sleep mode

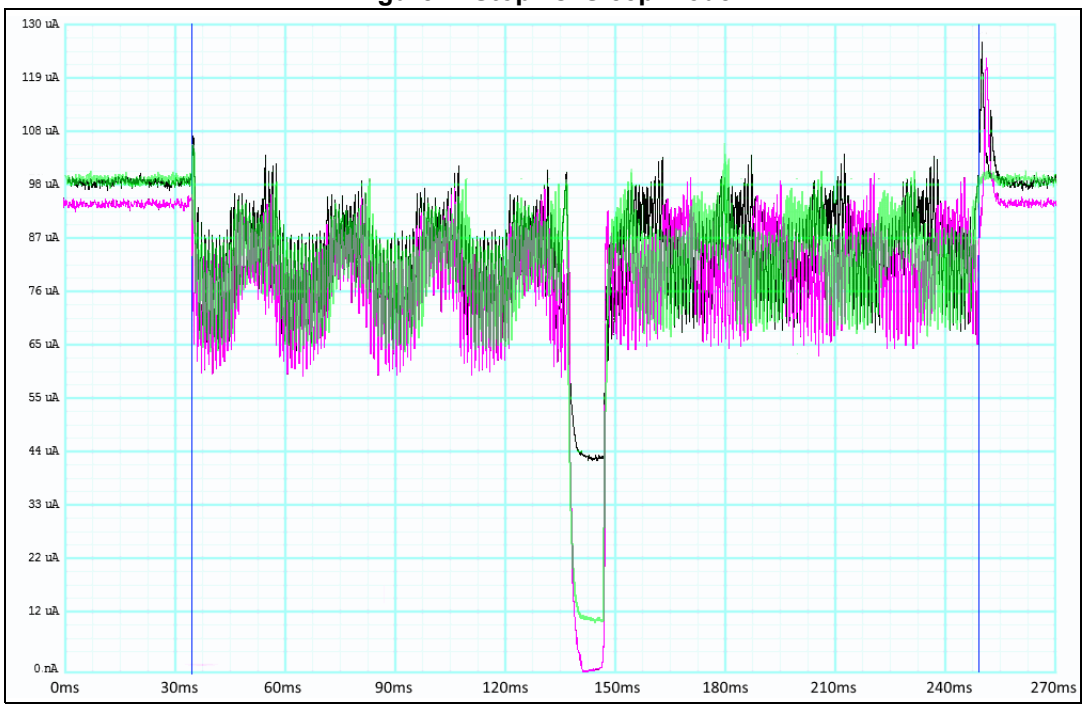


Table 5. Configurations - Stop vs. Sleep mode

Curve in *Figure 2*	Black	Green	Purple
Idle mode between transmissions	Sleep	Stop	
Debug interface	ON		OFF
Current average over phases 2, 3 and 4	83.3 µA	80.3 µA	77.3 µA
Clock	500 kHz MSI		

Table 5. Configurations - Stop vs. Sleep mode (continued)

Curve in *Figure 2*	Black	Green	Purple
Voltage regulator	Range 3		
ULP/FWU	+/+		
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	Interrupt operation (Com IT) with defines RXSLEEP and TXSLEEP		

6.1.2 Short periods of Sleep mode and Low-power run

Under most circumstances it is worthwhile to use WFE to put the core to suspended mode while waiting for a flag or event, but this may work against the effort of optimizing the power consumption in some cases. When the core clock speed is just enough to perform the requested operation, adding WFE instruction may actually increase the power consumption a bit. Consider using the Low-power run regulator mode instead. The Low-power run effect is terminated with each wakeup from either Stop or Sleep mode, but until that wakeup it saves a lot of energy.

In *Figure 3* the red line represents configuration where the execution in Low-power run mode is terminated by the code that goes to Sleep mode after each byte transmission. The green line is a code that runs with voltage regulator set to Range 3 and makes no attempts to use Sleep mode during transmission. Finally the black line is the power consumption of a code that stays in LPRUN until it goes to Stop mode after finishing the transmission. All three codes then use the regulator Range 3 along with Sleep mode after each byte during reception phase. It is worth noting that in case of such a short time in Stop mode it would be more efficient to leave the code in LPRUN all the time.

Figure 3. Sleep mode vs. LPRUN

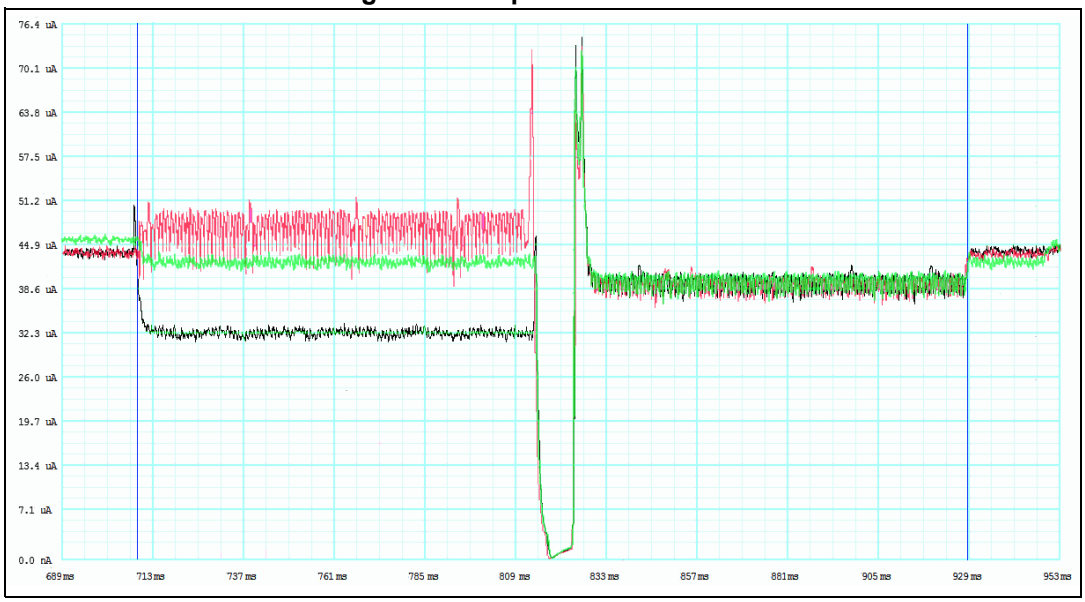


Table 6. Configurations - Sleep mode vs. LPRUN

Curve in *Figure 3*	Black	Green	Red
Idle mode between transmissions	Stop		
Debug interface	OFF		
Current average over phases 2, 3 and 4	34.9 µA	40.0 µA	42.0 µA
Clock	130 kHz MSI		
Voltage regulator	LPRUN, then Range 3	Range 3	
AHB/APB ratio	1/1		
ULP/FWU	+/+		
Baud rate	9600 Bd		
SW used	Blocking operation (Com Polling) with define RXSLEEP		Also TXSLEEP defined

6.1.3 Interrupt operation overhead

The interrupt mechanism on the Arm® Cortex® core is very efficient, but still represents an additional effort. This fact is best illustrated by comparison on the very minimum core clock suitable for interrupt driven operation of our example code, 250 kHz.

As shown in *Figure 4*, the IT code (black line) actually spends almost no time in Sleep mode (waiting for interrupt) and the power consumption is comparatively high. Blocking operation represented by the red line saves some energy in waiting for event (Tx buffer empty or Rx buffer full).

This comparison cannot be absolutely fair, as a different code is executed, nevertheless it holds significance of code optimization too.

Figure 4. Interrupt operation overhead

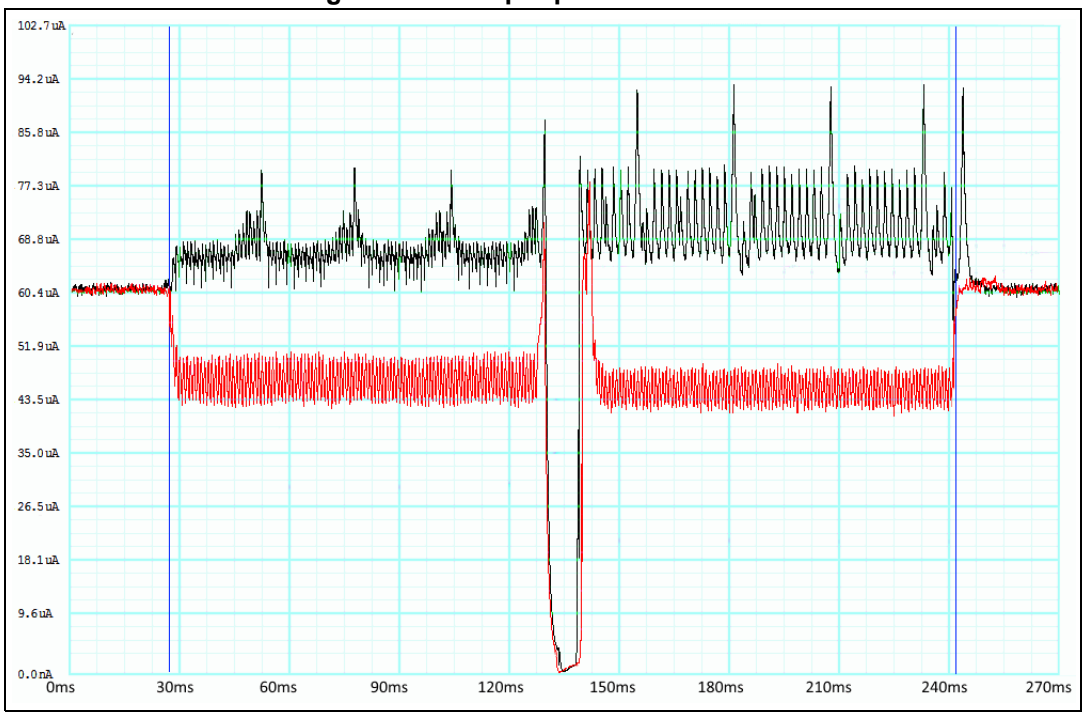


Table 7. Configurations - Interrupt operation

Curve in *Figure 4*	Black	Red
Idle mode between transmissions	Stop	
Debug interface	OFF	
Current average over phases 2, 3 and 4	66.6 µA	44.3 µA
Clock	250 kHz MSI	
Voltage regulator	Range 3	
AHB/APB ratio	1/1	
ULP/FWU	+/+	
Baud rate	9600 Bd	
SW used	Interrupt operation (Com IT) with defines RXSLEEP and TXSLEEP	Blocking operation (Com Polling) with defines RXSLEEP and TXSLEEP

6.1.4 Going to Stop between received bytes

The LPUARTs ability to wake from Stop on external event leads to possibility to enter the Stop mode even during pause between bytes in reception phase.

The best results comes with operational speeds slightly below the limit of voltage regulator Range 3, specifically 4 MHz MSI clock. In this way the interrupt overhead time is very short and the MCU spends most of the time in Stop mode.

Lower speeds actually lead to slightly higher power consumption during reception. The difference is not big, just around 10%. Yet still using polling and very slow clock setting the power consumption can be even lower than this, at the expenses of the flexibility of serving any other tasks.

Figure 5. Using the Stop during data reception

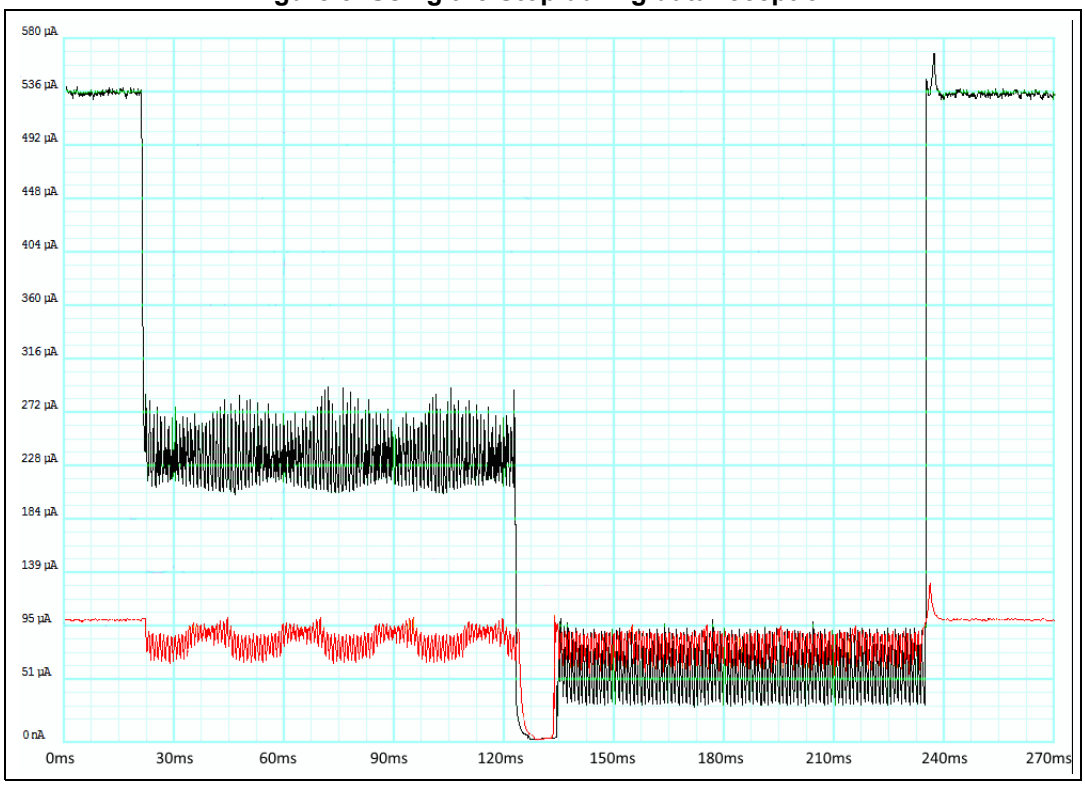


Table 8. Configurations - Stop during data reception

Curve in *Figure 5*	Black	Red
Idle mode between transmissions	Stop	
Debug interface	OFF	
Current average	55.5 µA (Rx only)	61.8 µA (Rx only)
Clock	4 MHz MSI	500 kHz MSI
Voltage regulator	Range 3	
AHB/APB ratio	1/1	
ULP/FWU	+/+	
Baud rate	9600 Bd	
SW used	Interrupt operation (Com IT) with defines RXSTOP and TXSLEEP	

6.1.5 Different oscillator clock speeds

Using the MSI clock the application computing power and power consumption can be easily tuned to imminent needs of the system. It is of course possible to change the clock settings on the fly, during execution.

Figure 6 shows the test execution at 2 MHz (black line), 500 kHz (red line) and 130 kHz (green line). The difference is large, but using the Sleep mode between both transmitted and received bytes the duty cycle makes it less prominent, at higher speeds the core spends more time in Low-power state.

There is, of course, no observable difference during the Stop mode.

Figure 6. Core clock speed comparison

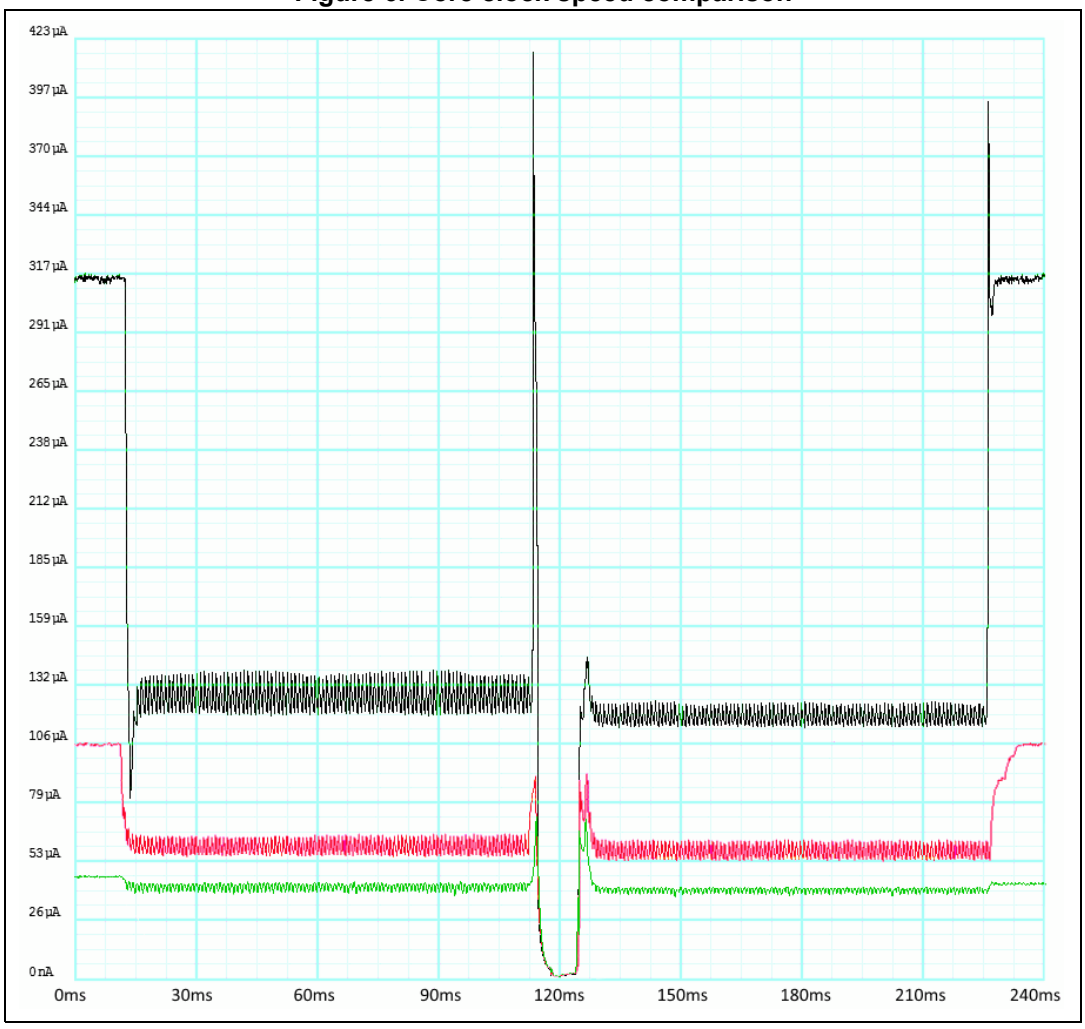


Table 9. Configurations - Core clock speed

Curve in *Figure 6*	Black	Green	Red
Idle mode between transmissions	Stop		
Debug interface	OFF		

Table 9. Configurations - Core clock speed (continued)

Curve in *Figure 6*	Black	Green	Red
Current average over phases 2, 3 and 4	136.6 µA	42 µA	57.1 µA
Clock	2 MHz MSI	130 kHz MSI	500 kHz MSI
Voltage regulator	Range 3		
AHB/APB ratio	1/1		
ULP/FWU	+/+		
Baud rate	9600 Bd		
SW used	Blocking operation (Com Polling) with RXSLEEP and TXSLEEP defined		

6.1.6 Changing AHB divider ratio

Changing the AHB divider ratio leads to decreased power consumption since most of the circuitry runs at lower frequency.

This is demonstrated in *Figure 7*, where the black line is the current absorption of the example running at 1 MHz MSI clock, and the red line is the same code running with AHB divider set to 4. If the clock can be directly configured to lower frequency, it is always better to choose this option. The green line represents the same example, only the MSI oscillator is set directly to 250 kHz. Proven by the difference between Sleep mode and run phase the actual processing power is the same, but the solution without AHB divider is more efficient.

Use of the AHB divider only makes sense with external clock or if there is necessity to use the HSI.

Figure 7. Lowering oscillator speed compared to AHB divider

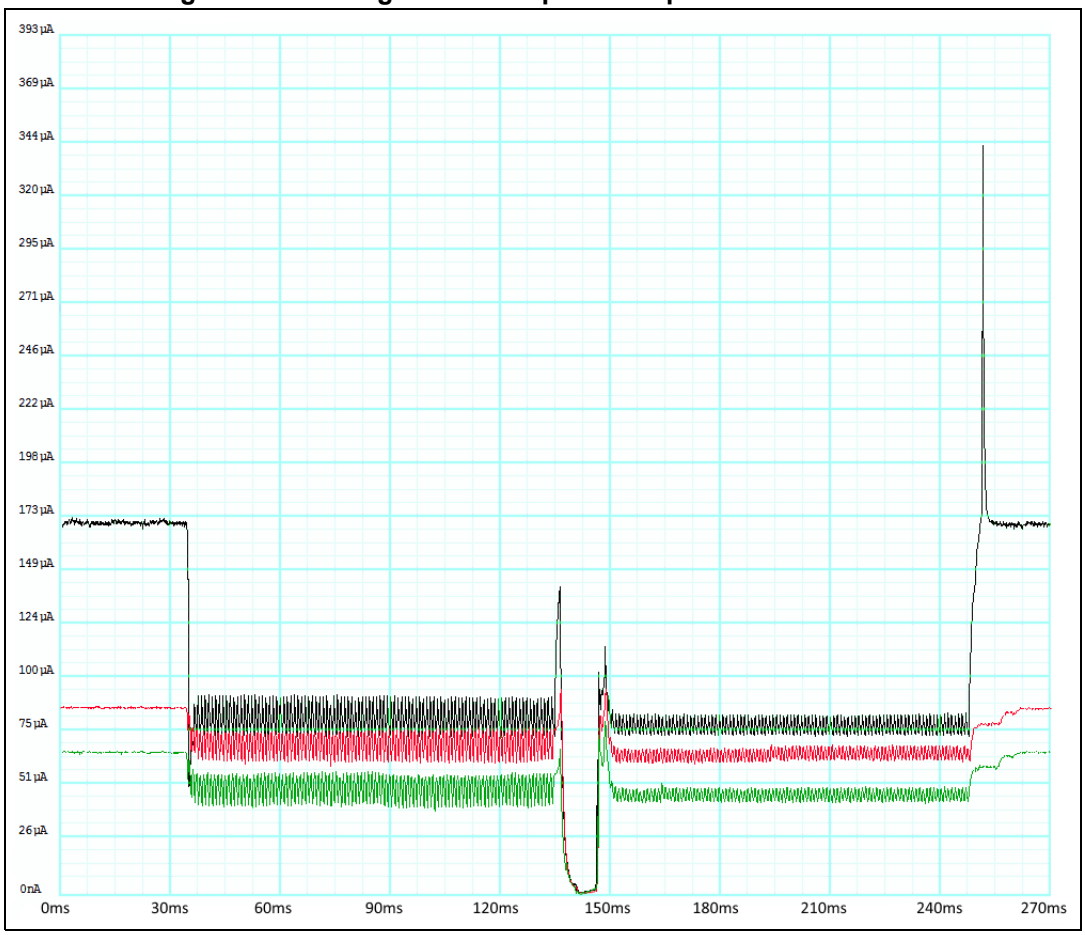


Table 10. Configurations - AHB divider

Curve in *Figure 7*	Black	Green	Red
Idle mode between transmissions	Stop		
Debug interface	OFF		
Current average over phases 2, 3 and 4	75.9 µA	44.3 µA	63.9 µA
Clock	1 MHz MSI	250 kHz MSI	1 MHz MSI
Voltage regulator	Range 3		
AHB/APB ratio	1/1		4/1
ULP/FWU	+/+		
Baud rate	9600 Bd		
SW used	Blocking operation (Com Polling) with RXSLEEP and TXSLEEP defined		

6.1.7 Different peripheral clock settings

DMA based communication was measured on three different APB clock divider settings, as shown in *Figure 8*, where the black line represents current consumption with no divider, the red and green lines with divider set to 4 and to 8, respectively.

Figure 8. Three different settings of peripheral clock divider

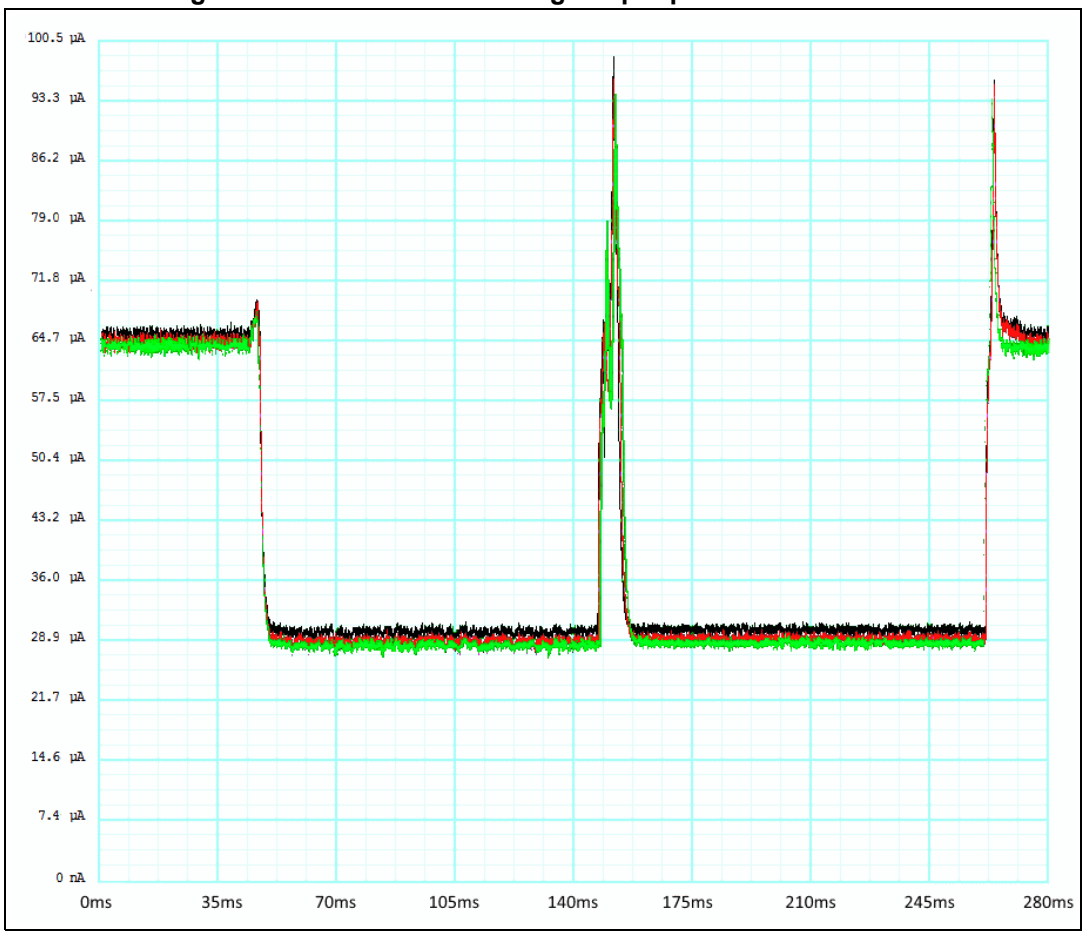


Table 11. Configurations - Clock divider

Curve in *Figure 8*	Black	Red	Green
Idle mode between transmissions	Stop		
Debug interface	OFF		
Current average over phases 2, 3 and 4	31.5 µA	30.5 µA	30.1 µA
Clock	250 kHz MSI		
Voltage regulator	Range 3		
AHB/APB ratio	1/1	1/4	1/8
ULP/FWU	+/+		

Table 11. Configurations - Clock divider (continued)

Curve in *Figure 8*	Black	Red	Green
Baud rate	9600 Bd		
SW used	DMA operation (Com DMA) with half buffer interrupt disabled		

The power trace of the DMA operated example is different from the interrupt or blocking mode code execution. There is a peak in half of the transmission, caused by the half message interrupt. Then for half of the reply data reception the core stays in Sleep mode, it is only active during the second half. Different arrangement is of course possible.

6.1.8 ULP bit setting

The role of the ULP bit is to disconnect the internal voltage reference during Stop mode. The power analysis screen-shot in *Figure 9* illustrates that this setting (black line) achieves lower power consumption in the Stop mode, but produces significant peaks when waking up (compared to green trace without ULP).

The microcontroller waiting for incoming data in the ULP mode consumes approximately 0,8 µA compared to almost 3 µA without ULP mode, but if it wakes from idle mode too frequently, the effect is negated by the voltage reference startup peaks.

Figure 9. ULP bit effect

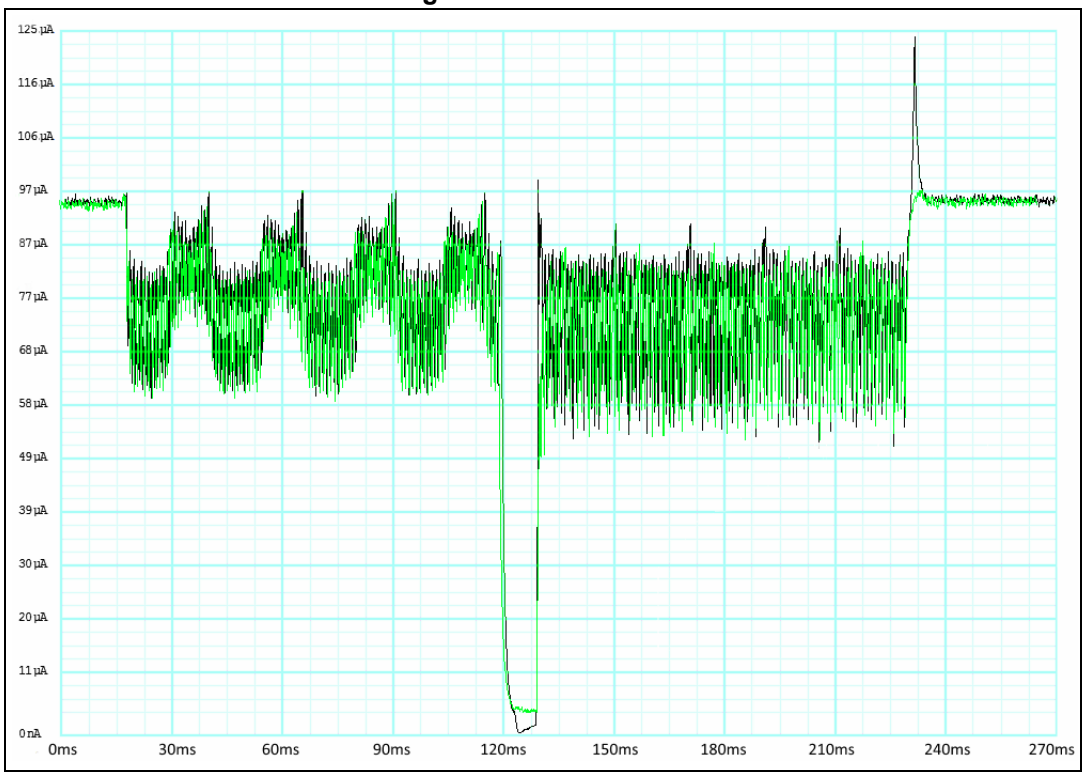


Table 12. Configurations - ULP bit effect

Curve in *Figure 9*	Black	Green
Idle mode between transmissions	Stop	
Debug interface	OFF	
Current average over phases 2, 3 and 4	74.5 µA	72.6 µA
Clock	500 kHz MSI	
Voltage regulator	Range 3	
AHB/APB ratio	1/1	
ULP/FWU	+/+	-/-
Baud rate	9600 Bd	
SW used	Interrupt operation (Com IT) with defines RXSTOP and TXSLEEP	

6.1.9 Higher communication speed

While no advanced power saving features of the LPUART can be used on higher communication speeds (as compared with regular USART peripheral), still the LPUART simpler circuitry will result in lower power consumption (red line) compared to a full featured USART (black line), see *Figure 10*.

Figure 10. Operating at 57600 baud

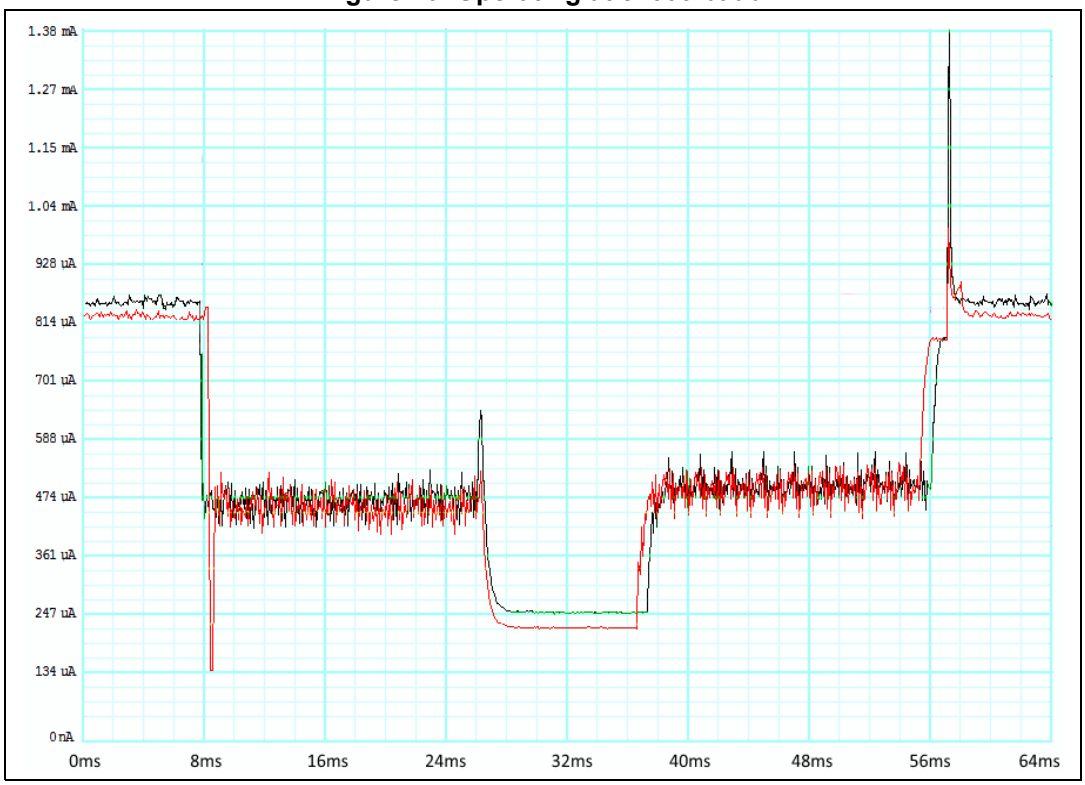


Table 13. Configurations - Higher communication speed

Curve in *Figure 10*	Black	Red
Idle mode between transmissions	Stop	
Debug interface	ON	
Current average over phases 2, 3 and 4	434.5 µA	415.2 µA
Clock	4 MHz MSI	
Voltage regulator	Range 1	
AHB/APB ratio	1/1	
ULP/FWU	+/+	
Baud rate	57600 Bd	
SW used	Interrupt operation (Com IT) modified with USART on SYSCLK	Interrupt operation (Com IT) modified with LPUART on SYSCLK

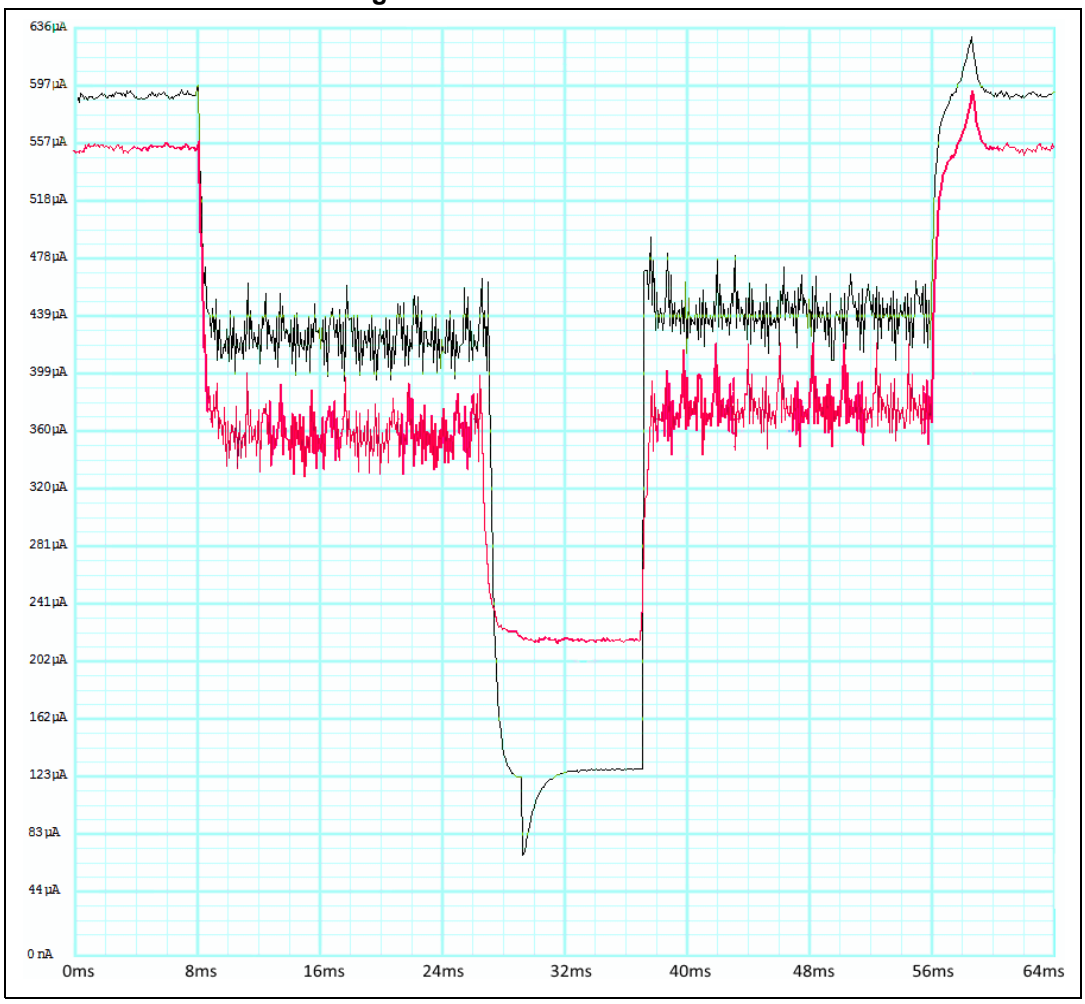
Note: The timing is slightly different for the 2 cases. This is a common problem when MSI is used to clock the LPUART peripheral. MSI clock is generally less precise than LSE or even HSI.

6.1.10 Wakeup from Stop mode on HSI

Another option to use with higher communication speed is to exploit the feature of wakeup from Stop mode with peripheral being clocked from the HSI clock (see *Figure 11*).

The HSI oscillator draws approximately as much current in combination with the Stop mode as the MSI at 2 MHz requires in Sleep mode. It is however more precise and with provides the core with more processing power in run mode.

Figure 11. HSI vs. MSI at 4 MHz



Note that the timing with MSI is slightly off in the transmission phase. This is a common problem when MSI is used to clock the LPUART peripheral.

Table 14. Configurations - HSI vs. MSI

Curve in *Figure 11*	Black	Red
Idle mode between transmissions	Stop	Sleep
Debug interface	OFF	
Current average over phases 2, 3 and 4	369.1 µA	329.7 µA
Clock	4 MHz HSI	4 MHz MSI
Voltage regulator	Range 3	
AHB/APB ratio	1/1	
ULP/FWU	+/+	

Table 14. Configurations - HSI vs. MSI (continued)

Curve in *Figure 11*	Black	Red
Baud rate	57600 Bd	
SW used	Interrupt operation (Com IT)	

6.1.11 Voltage regulator settings

Setting lower core voltage level is an easy and straightforward way to save energy.

In *Figure 12* it is demonstrated that even with HSI clock the run mode can be significantly more current-hungry just by changing the voltage.

Note that the setting has little or no influence on idle mode.

Figure 12. Comparison between voltage regulator settings

Table 15. Configurations - Voltage regulator settings

Curve in *Figure 12*	Red	Green	Black
Idle mode between transmissions	Sleep		
Debug interface	OFF		

Table 15. Configurations - Voltage regulator settings (continued)

Curve in *Figure 12*	Red	Green	Black
Current average over phases 2, 3 and 4	296.9 µA	300.8 µA	301.1 µA
Clock	4 MHz HSI		
Voltage regulator	Range 3	Range 2	Range 1
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	DMA operation (Com DMA)		

6.1.12 GPIO pull-up

Figure 13 demonstrates the energetic costs of enabling the GPIO pull-up, in this case on the transmission line. Black line refers to internal GPIO pull-up enabled, red line to no pull-up.

It is clearly visible that no other phases are affected by extra power consumption.

Figure 13. GPIO internal pull-up

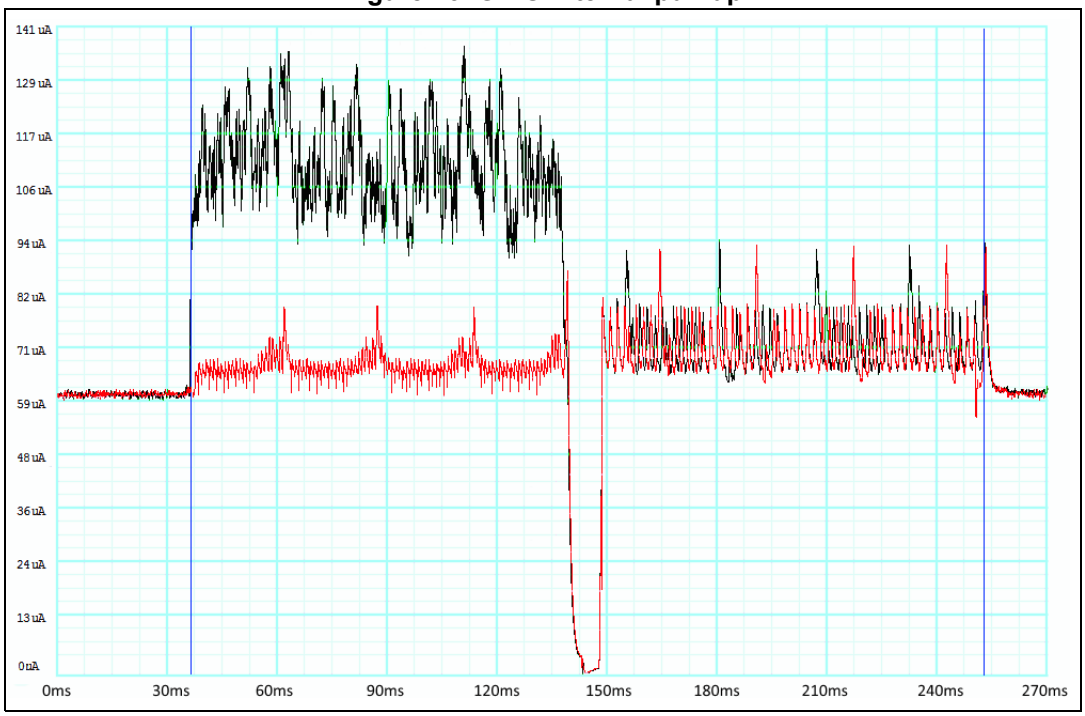


Table 16. Configurations - GPIO pull-up

Curve in *Figure 13*	Black	Red
Idle mode between transmissions	Stop	
Debug interface	OFF	

Table 16. Configurations - GPIO pull-up (continued)

Curve in *Figure 13*	Black	Red
Current average over phases 2, 3 and 4	87.8 µA	66.6 µA
Clock	250 kHz MSI	
Voltage regulator	Range 3	
AHB/APB ratio	1/1	
ULP/FWU	+/+	
Baud rate	9600 Bd	
SW used	Interrupt operation (Com IT) modified with Tx line internal PULL UP	Interrupt operation (Com IT)

6.2 Measurements on STM32L476 Nucleo board

6.2.1 Three approaches at a glance

In this section the advantages and disadvantages of the three basic approaches (DMA, Polling and Interrupt) to govern the communication will be compared.

Looking at *Figure 14*, the DMA one wins this comparison, but, if the pause before reply had been longer, it would have lost by a wide margin. Polling would then appear as the most efficient, but it assumes that a single task be processed. Interrupt seems to solve this problem, but at 200 kHz system clock it has no margin to do so (judging by the negligible time spent in Sleep between bytes).

While DMA and Polling driven communications are still operational at 100 kHz minimal MSI setting, Interrupt operation is not.

Figure 14. Comparison of three different approaches to manage communication

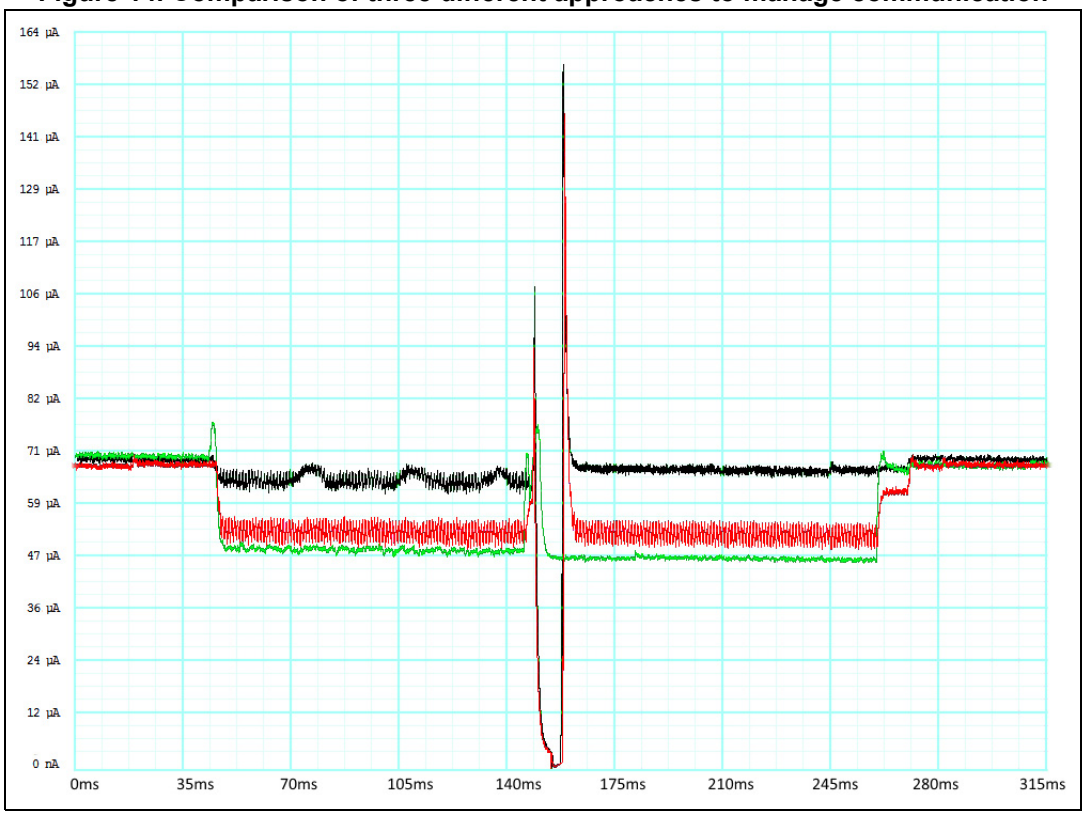


Table 17. Configurations - Managing communication

Curve in *Figure 14*	Black	Red	Green
Idle mode	Stop2		Sleep
Debug interface	OFF		
Current average over phases 2, 3 and 4	64.1 µA	51.6 µA	49.4 µA
Clock	200 MHz MSI		

Table 17. Configurations - Managing communication (continued)

Curve in *Figure 14*	Black	Red	Green
Voltage regulator	Low-power run		
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	Interrupt driven (Com IT) with Sleep between bytes	POLL operation with TXSLEEP and RXSLEEP defines	DMA stream (Com DMA) with half buffer interruptions

6.2.2 Simple polling mode on low core frequency

After reset the MCU initializes with MSI oscillator running at 4 MHz, a compromise between low-power consumption and acceptable communication speed. A lower frequency is needed to establish a communication link using LPUART. Note that at this pace an interrupt from different peripheral may disrupt the ongoing communication.

Figure 15 shows the consumption profile using different low speeds of the MSI oscillator.

Figure 15. Polling communication at minimum settings

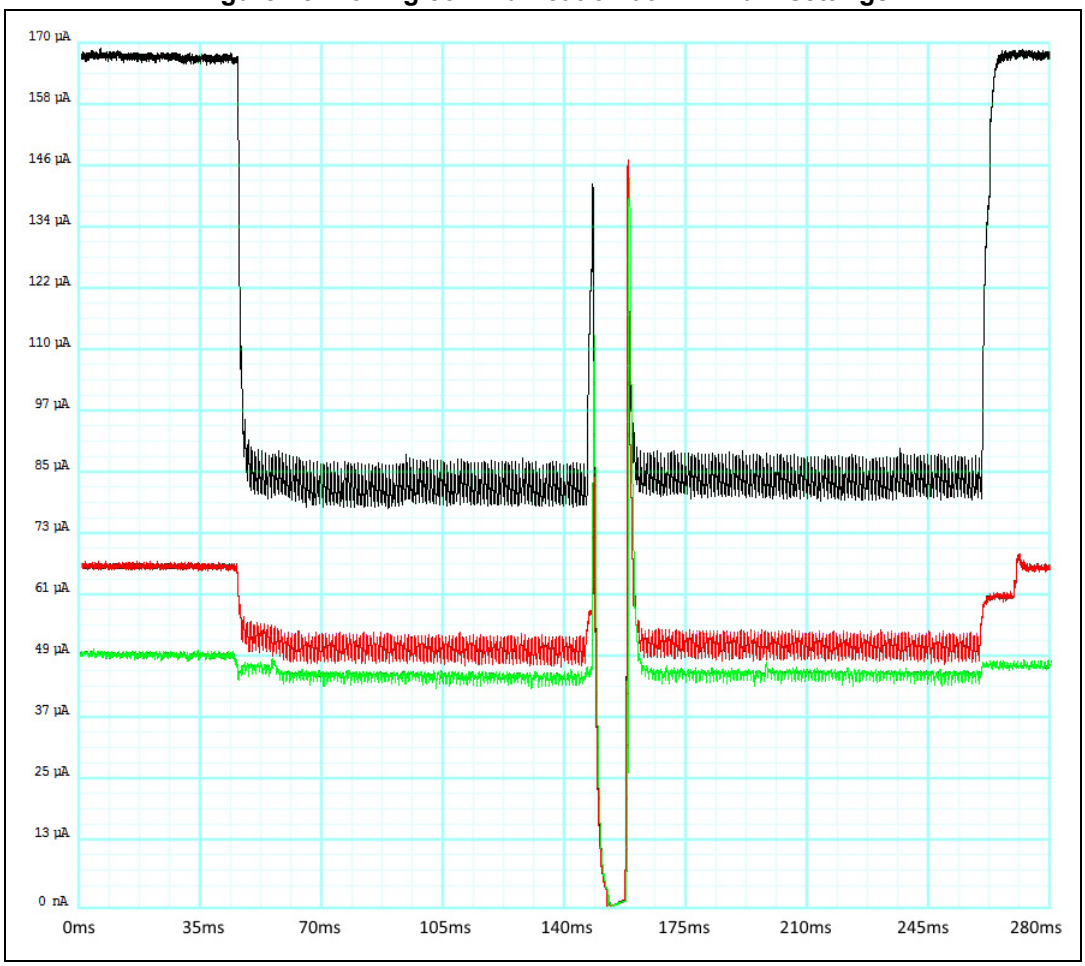


Table 18. Configurations - Polling

Curve in *Figure 15*	Black	Red	Green
Idle mode	Stop2		
Debug interface	OFF		
Current average over phases 2, 3 and 4	81.2 µA	49.7 µA	44.9 µA
Clock	800 kHz MSI	200 kHz MSI	100 kHz MSI
Voltage regulator	Low-power run		
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	POLL operation with TXSLEEP and RXSLEEP		

6.2.3 The role of voltage regulator settings

An important step towards lower power consumption is the use of regulator settings appropriate to the actual core frequency. Speed of 2 MHz (MSI Range 5) is the top setting supported by the Low-power run mode. Remaining connected to the main regulator (in *Figure 16* this is emphasized by using it even during Sleep) a lot of energy is wasted.

Figure 16. Voltage regulator settings

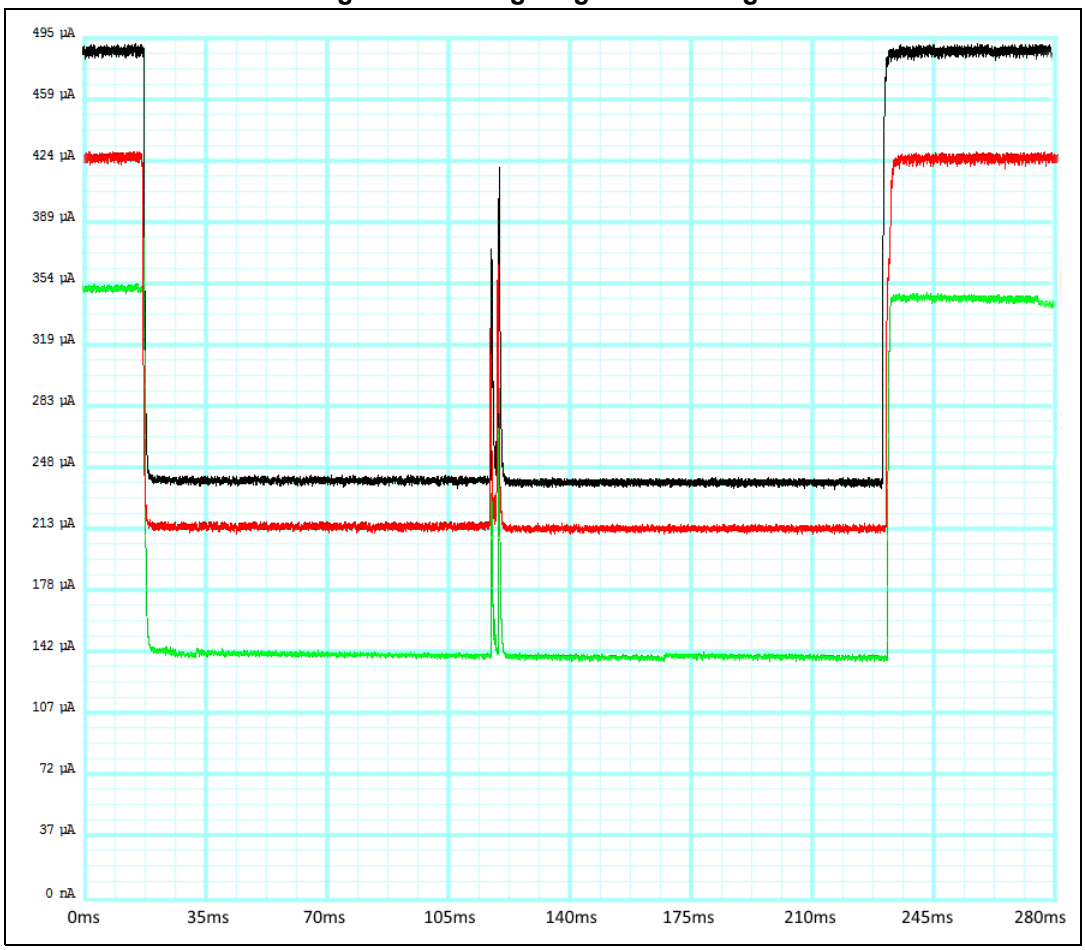


Table 19. Configurations - Voltage regulator settings

Curve in *Figure 16*	Black	Red	Green
Idle mode	Sleep		
Debug interface	OFF		
Current average over phases 2, 3 and 4	244.4 µA	215.5 µA	139.1 µA
Clock	2 MHz MSI		
Voltage regulator	Range 1	Range 2	Low-power run
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	DMA operation (Com DMA) with half buffer interrupt disabled		

6.2.4 Idle modes compared

Microcontrollers of the L4 Series feature three basic Idle modes, namely Sleep, Stop1 and Stop2.

Note, in *Figure 17*, how Stop 2 minimizes the power consumption drastically, and how waking up produces a notable peak. This causes the Stop1 to outperform Stop2 in measured case of approximately 10 ms of Idle. Main reason of this peak is the fact that the Stop2 mode cannot be entered directly from the Low-power mode. Main regulator must be ON when entering and leaving the Stop2.

Figure 17. Comparison of Idle modes

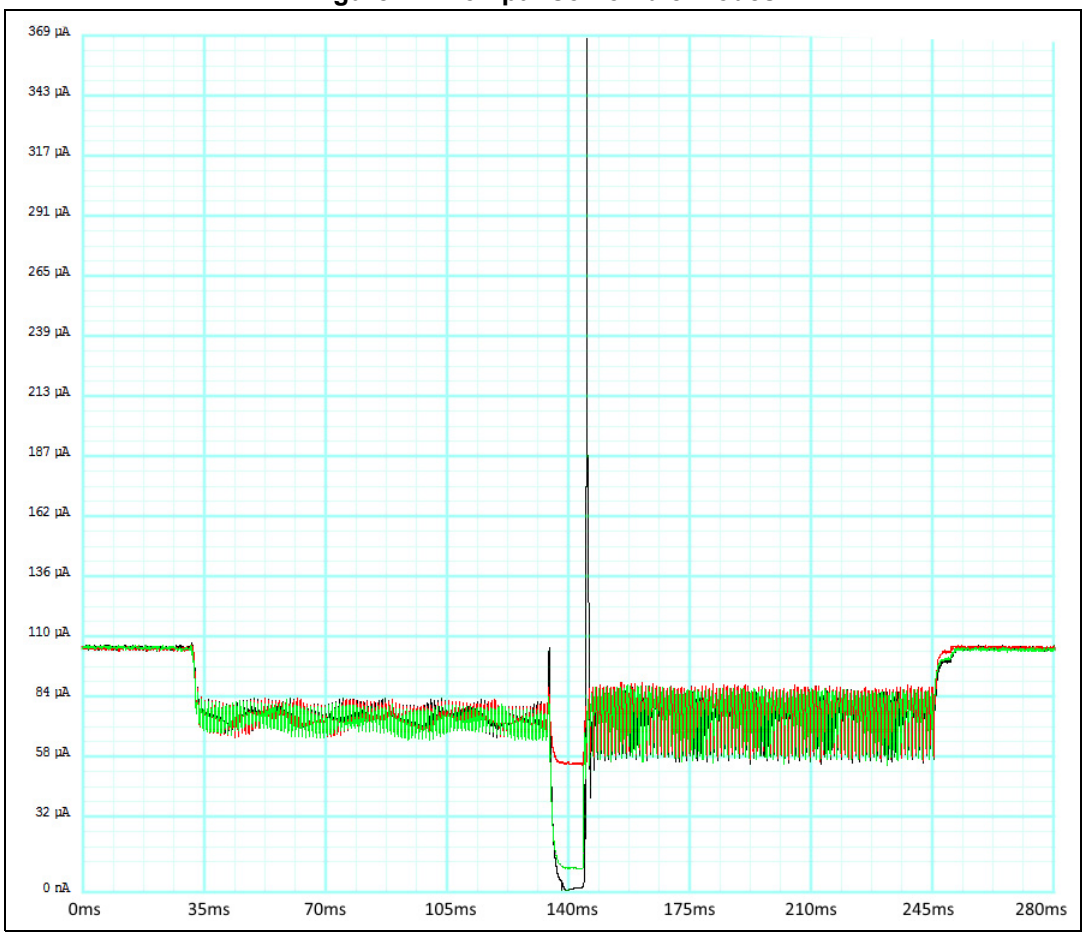


Table 20. Configurations - Idle modes

Curve in *Figure 17*	Black	Red	Green
Idle mode	Stop2	Sleep	Stop1
Debug interface	OFF		
Current average over phases 2, 3 and 4	73.2 µA	75.6 µA	72.8 µA
Clock	400 kHz MSI		
Voltage regulator	Low-power run		
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	IT operation (Com IT) with Rx Stop		

6.2.5 Use of MSI PLL-mode for higher speeds

Use of higher communication speeds require a LPUART clock source different from LSE.

System clock driven by MSI up to 48 MHz can be used, configured in PLL-mode in which it is auto-calibrated using the LSE. Obvious but not efficient solution is to use the HSI clock with dividers (such as the AHB).

HSE external source is another option. The measurement shown in *Figure 18* compares the HSI configurations with use of the MSI PLL-mode to get lower clock frequency and lower consumption along with it.

Figure 18. HSI and PLL current consumption

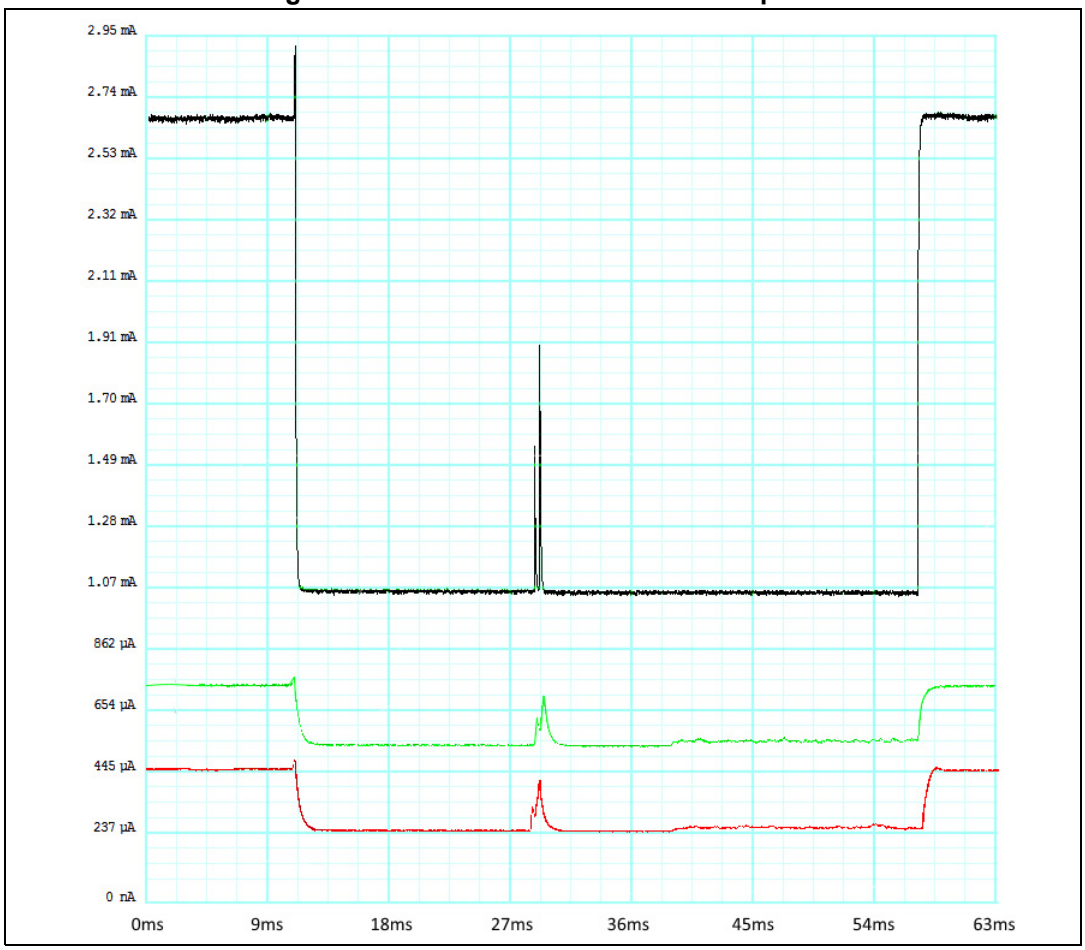


Table 21. Configurations - HSI and PLL

Curve in *Figure 18*	Black	Red	Green
Idle mode	Sleep		
Debug interface	OFF		
Current average over phases 2, 3 and 4	1.05 mA	0.536 mA	0.250 mA

Table 21. Configurations - HSI and PLL (continued)

Curve in *Figure 18*	Black	Red	Green
Clock	HSI16		MSI auto-callibrated using LSE; Sysclk 2 MHz
Voltage regulator	Range2		Low-power run
AHB/APB ratio	1/1	8/1	1/1
Baud rate	57600 Bd		
SW used	DMA operation (Com DMA) with half buffer interrupt disabled		

6.2.6 Using two oscillators

To use Stop modes with higher communication speed, the LPUART peripheral must be clocked from the HSI16 source. It is not necessary to reuse the same 16 MHz clock for the Sysclk as well, especially if this speed is considered too high. For the 57600 Bd speed it is possible to use 2 MHz frequency from MSI, saving a considerable amount of energy.

Figure 19. Using Stop with 57600 Bd speed

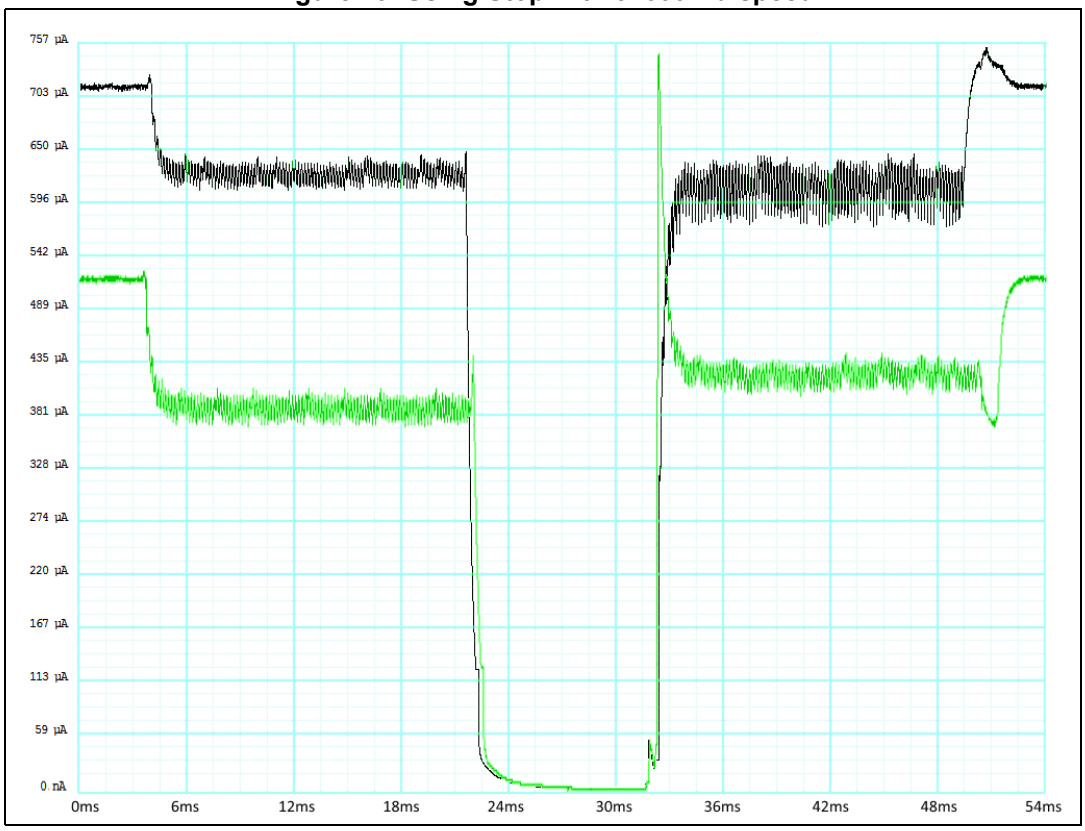


Table 22. Configurations - Stop with 56700 Bd speed

Curve in *Figure 19*	Black	Green
Idle mode	Stop2	
Debug interface	OFF	
Current average over phases 2, 3 and 4	480 µA	321 µA
Clock	HSI16	MSI auto-callibrated using LSE; Sysclk 2 MHz
Voltage regulator	Range2	Low-power run
AHB/APB ratio	8/1	1/1
Baud rate	57600 Bd	
SW used	IT operation (Com IT) with Rx Stop	IT operation (Com IT)

Note: For Stop2 mode the LPRUN must be exited.

6.3 Measurement on the STM32H743 Nucleo144 board

The reference manual describes how to use the D3 domain to minimize power consumption when communicating using the LPUART. This is the only case interesting in the context of this application note.

The larger MCU has non-negligible static power consumption in Stop mode, for the details see the tables in the datasheet.

Since there is only one configuration presented in example, there is no measurement comparison here.

6.4 STM32WB55 Nucleo board testing

This product has several similarities with STM32L476, but it is very different in purpose and application, thanks to its RF part, which is practically unusable with low power regulator. For STM32WBxx devices the embedded SMPS is more useful.

The example covers some settings different from the other products. The STM32WB55 based project also shows how the DMA and DMAMUX must be configured (this setup is very different vs. the STM32L476).

It is also important to keep in mind the device embeds two CPU cores. In the example project the second core (Cortex® M0, responsible for the RF stack) is shut down, but in real applications this will probably not be the case.

Figure 20. Comparison of three different approaches to manage communication

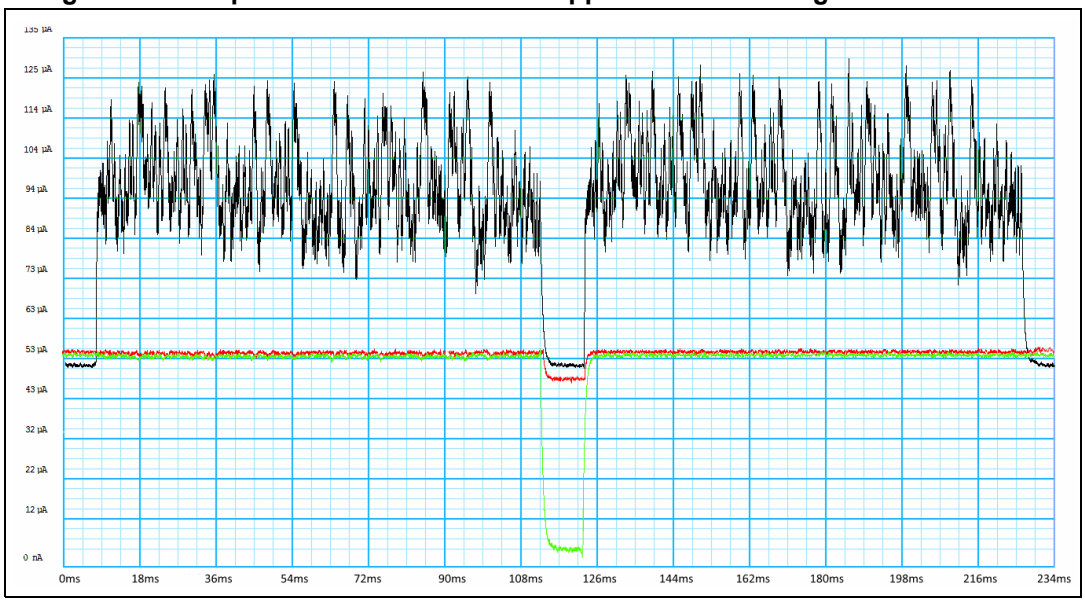


Table 23. Configurations - Managing communication

Curve in *Figure 20*			
Idle mode between transmissions	Sleep	Stop0	Stop2
Debug interface	OFF		
Current average over phases 2, 3 and 4	96 µA	52 µA	50 µA
Clock	800 kHz MSI		
Voltage regulator	Range2		
AHB/APB ratio	1/1		
Baud rate	9600 Bd		
SW used	DMA	Polling	Interrupt

6.5 STM32G474 Nucleo board testing

This device is not low-power oriented, yet embeds the LPUART peripheral. While actually having a low-power regulator, it does not have the MSI clock and the best way to make use of the Low-power run without HSE external clock is to use the AHB clock prescaler on the HSI.

Figure 21. Interrupt operation with three different core clocks

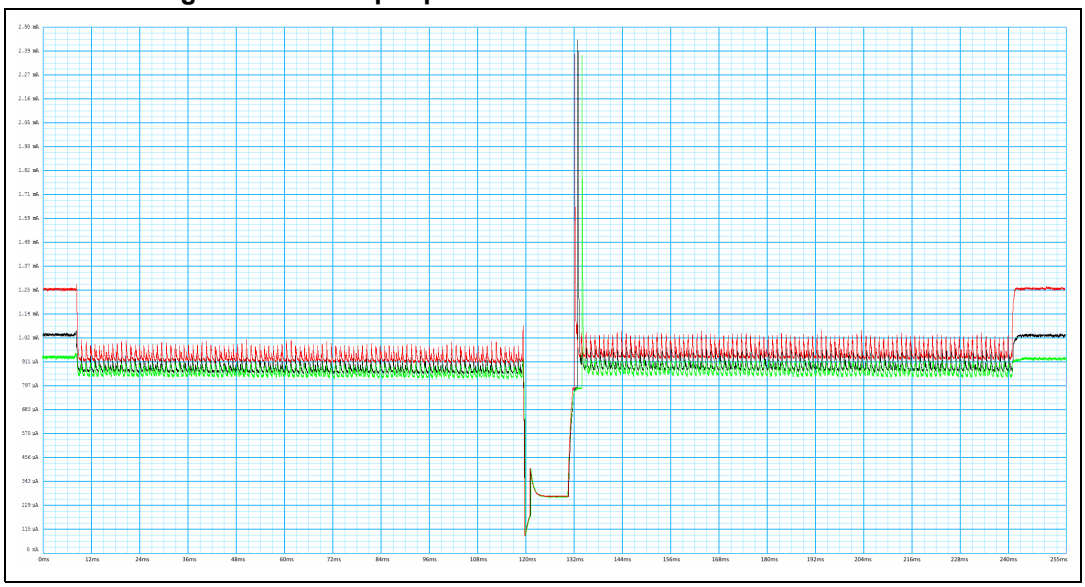


Table 24. Influence of the AHB prescaler

Curve in *Figure 21*			
Idle mode between transmissions	Stop1		
SWD interface	OFF		
Current average over phases 2, 3 and 4	914 µA	858 µA	838 µA
Clock	16 MHz HSI		
Voltage regulator	Low-power		
AHB/APB ratio	1/4	1/8	1/16
Baud rate	9600 Bd		
SW used	Interrupt		

There is no prescaler divider by 32 available. On the divide by 64 setting the example code is barely able to receive the data without overrun. A more optimized code could be able to keep up with this data rate, but some margin is needed for reliability and robustness.

7 Example project

An example code X-CUBE-LPUART is supplied with this document, demonstrating the usage of the different configurations (most of them are switchable using the defines section in the main.c source file).

Two boards are needed to replicate the measurements, namely a primary board and a repeater, loaded with DMA FW built with BOARD2 option.

Measurements of consumption are carried out on the primary board, not on the repeater.

The example has been developed and tested with Nucleo boards.

7.1 HW setup

Just cross connect Tx pins with Rx pins on the opposite board using two wires. JP6 pins of Nucleo-64 or JP5 pins of Nucleo-144 can be used to monitor the power consumption. It is possible to combine different Nucleo boards, keeping in mind that the Rx and Tx pins have different positions.

Most boards require no physical modification, only for STM32H7 Nucleo144 solder bridges SB2 and SB125 have been removed before measurement.

7.2 Configuring the example

Various features and working modes of the example FW can be configures using following preprocessor defines:

- DEBUG_OFF: disables the SWD debug capability on port A and error checking
- BOARD2: configures the SW as the repeater board (only present in DMA version)
- UI: enables user interface (it is impossible to take correct power consumption readings with buttons and LEDs enabled)
- BD_SPEED: sets the communication baud speed
- STOP: configures Stop mode when waiting for reception (where applicable)
- STOP2: configures Stop2 Low-power mode when waiting for reception (where applicable)
- LPRUN: configures the Low-power run mode (where applicable)
- PWR_CR_VOS_CONF: configures the main power regulator (lowest available is the default)
- TXSLEEP: Sleep mode configured between transmitted bytes
- RXSLEEP: Sleep mode configured between received bytes
- RXSTOP: Stop mode configured between received bytes (available in interrupt mode)
- HSI: sets the system clock to HSI mode. MSI is default where available
- RCC_MSIRANGE_SET: if HSI is disabled, sets the MSI speed range.

7.3 Example operation

It is recommended to first load and execute firmware in board 1 (the measured one) then prepare the repeater. The measurement cycle may be then started by pressing the user button on the board 1.

The boards start exchanging messages in an endless loop. In case of communication error and debugging enabled, the loop is halted. With UI option the board will flash the LED to indicate activity. This has significant influence on the current consumption and for all power measurements disables the UI (LED blinking) on board 1.

8 Conclusion

There is no single optimal way of configuring the LP UART with respect to lowest possible power consumption.

Depending on the application and the operational constraints, a lot of parameters have to be taken into account, the general guidelines can be summarized as follows:

- use lower frequencies ;
- use lower voltages;
- keep the MCU as much as possible in low-power modes.

Optimized software is also crucial part of the whole low-power solution. It is advisable to go through the same test with firmware compiled at different optimization levels.

9 Revision history

Table 25. Document revision history

Date	Revision	Changes
22-Apr-2015	1	Initial release.
26-May-2015	2	Updated *Introduction* with addition of software X-CUBE-LPUART. Updated *Section 7: Example project*.
12-Oct-2015	3	Introduced STM32L4 Series. Updated *Introduction*, *Section 2.1: Clock subsystem*, *Section 2.3: Comparison with USART peripheral*, *Section 4.2: GPIO configuration*, *Section 4.4.1: Use of Stop and Sleep modes*, *Section 4.4.2: Run time configuration*, *Section 5.2: Dropped bytes*, *Section 7: Example project* and its subsections. Updated tables *2* to *16*. Updated *Figure 8: Three different settings of peripheral clock divider* and *Figure 12: Comparison between voltage regulator settings*. Added *Section 2.1: Comparison between L0 and L4 series* and its subsections, *Section 4.1: Execution from SRAM*, *Section 6.2: Measurements on STM32L476 Nucleo board* and its subsections.
04-Jun-2019	4	Introduced STM32G0, STM32G4, STM32H7 and STM32WB Series. Updated document title, *Introduction*, *Section 2: Summary of features* and its subsections, *Section 3.3: DMA mode*, *Section 4.3: Clock configuration*, *Section 4.3.1: Clock prescalers*, *Section 4.4.1: Use of Stop and Sleep modes*, *Section 4.4.2: Run time configuration*, *Section 7: Example project* and *Section 8: Conclusion*. Updated *Table 2: List of acronyms* and *Table 4: Clock options*. Added *Table 1: Applicable products and software*, *Section 4.4.3: STM32H7 core domain sections*, *Section 6.3: Measurement on the STM32H743 Nucleo144 board*, *Section 6.4: STM32WB55 Nucleo board testing* and *Section 6.5: STM32G474 Nucleo board testing*. Minor text edits across the whole document.
28-Nov-2019	5	Introduced STM32L4+ and STM32L5 Series. Updated *Introduction*, *Section 3.3: DMA mode* and *Section 6: Power consumption comparison*. Updated *Table 1: Applicable products and software* and *Table 4: Clock options*. Added *Section 4.4.4: SMPS* and *Section 4.5: Cache memory*. Minor text edits across the whole document.