AN4311
Application note

Assessing STM32L1 Series current consumption

Introduction

The STMicroelectronics ARM® Cortex™-M3 based STM32L1 series uses ST's proprietary ultra-low-leakage process technology with an innovative autonomous dynamic voltage scaling and 5 low-power modes offering unprecedented platform flexibility to fit any application. The STM32L1 series extends the ultra-low-power concept without compromising performance.

This complex architecture means a wider choice of configuration settings and operating modes. This application note describes how to configure your STM32L1 device targeting the key low power features or run mode of this family. It provides verified and ready-to-use code examples to quickly evaluate current consumption on your Discovery board or any other platform.

This document does not provide any set of configurations for device characterization. The STSW-STM32146 firmware attached is for guidance only. Please refer to the related datasheet for guaranteed values and up-to-date characterization data.

Table 1. Applicable products, tool and firmware

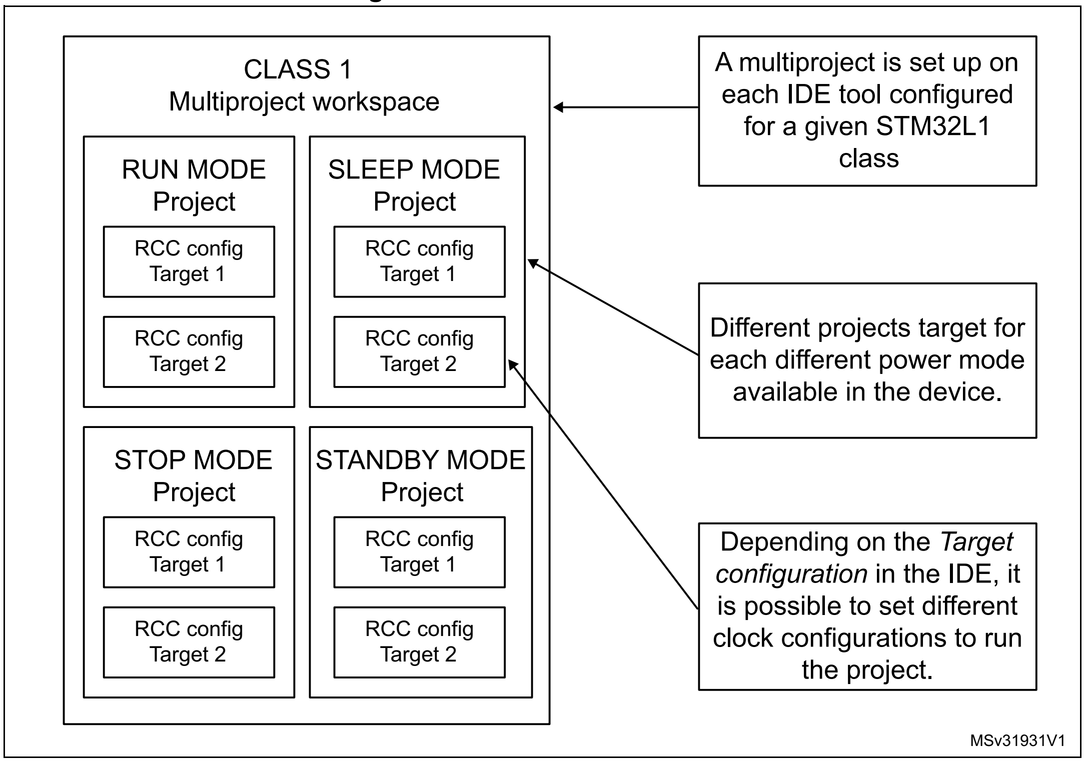
Type	Part numbers and product categories
Evaluation tool	STM32L-DISCOVERY
Firmware	STSW-STM32146

Contents

1 Firmware architecture

This application note and the associated firmware define the basic configuration of the device, necessary to target optimum current consumption. It provides an explicit and didactic approach allowing you benefit from the various optimized low power modes. The architecture of the firmware is described in *Figure 1: Firmware architecture*.

Figure 1. Firmware architecture


CLASS 1
Multiproject workspace
RUN MODE Project
RCC config Target 1
RCC config Target 2
SLEEP MODE Project
RCC config Target 1
RCC config Target 2
STOP MODE Project
RCC config Target 1
RCC config Target 2
STANDBY MODE Project
RCC config Target 1
RCC config Target 2
A multiproject is set up on each IDE tool configured for a given STM32L1 class
Different projects target for each different power mode available in the device.
Depending on the *Target configuration* in the IDE, it is possible to set different clock configurations to run the project.
MSv31931V1

The approach elected for this firmware is based on a multiproject workspace. Each project refers to the active or one of the low power modes available on the device. They are configured for the simplest use case. The `main.h` file contains a set of '*#define*'s allowing you to fine-tune the configuration for a more customized test (you will find more details in *Table 2: Compilation options* or in the code itself as comments). Target settings for each project allow different clock configurations, making it possible to reproduce an exact application case during the evaluation phase.

The clock frequency or oscillator range must be defined according to your requirements in the configuration wizard of the IDE tool.

1.1 Run mode

The Active consumption of the device is evaluated by running different kinds of code in the device. Refer to *Table 2* for more details on compilation options.

Table 2. Compilation options

Define	Description
ENABLE_DEBUG	Allows you to keep the I/O configuration to communicate through the debug tool.
ENABLE_RTC	The RTC is enabled in Stop and Standby modes.
ENABLE_IWDG	IWDG is enabled in Stop and Standby modes.
ENABLE_LCD	The LCD is enabled in Stop mode.
ENABLE_PERIPHERAL_CLOCK	All peripheral clocks are enabled or disabled in Run and Sleep modes.
RUN_MODE	Allows you to choose the code executed in Run mode.
CODE_LOCATION	Allows you to choose in which memory the code is executed.

1.1.1 Emulated dhrystone loop

The Active consumption of the device is evaluated by running an instruction loop in Flash memory. This code is designed to obtain a current consumption approximately equivalent to the Dhrystone benchmark, but written in assembly. The advantage of this approach is to keep a code which is not dependent of the compiler settings.

RAM execution and DMIPS/mA

In order to check the difference in current consumption when the application code is being executed from the internal Flash or internal RAM, you must specify in the option of your compiler, IDE, where to map the file *dhrystone_like.c*. Executing from the RAM results in a lower current consumption as the Flash can be turned off, producing a significantly higher DMIPS/mA rate.

1.1.2 CoreMark code

Consumption of true CoreMark® code can be evaluated. The current consumption may depend on compilers and optimization settings. The necessary configuration allowing measurement of the CoreMark score (Timer, USART...) is disabled to measure only CPU consumption.

1.1.3 Dhrystone code

Take care to disable inlining in the compiler options to respect dhrystone requirements when running this test. For more information on how to do this, refer to your C-compiler documentation.

1.1.4 Fibonacci code

This code executes the calculation of the first 46 numbers of the Fibonacci series. After 46 iterations, the 32-bit result overflows. It is also possible to execute this code from the internal RAM memory with the Flash put in power-down mode in order to achieve lower current consumption and higher performance.

1.1.5 Infinite loop in c: while (1)

To compare the consumption of the core when executing complex calculation versus basic loop, this option is provided to the user. It is also possible to execute it in RAM memory with Flash turned off.

Caution: Please note that the “c while(1); routine” may generate different current consumptions on the device when executed from the Flash memory only with Prefetch and 64-bit access enabled. If the instruction preceding the infinite Branch is a 32-bit instruction, the Branch instruction is accessed from Flash. On the other hand, if the Branch instruction follows a 16-bit instruction, it can be accessed entirely from the prefetch resulting in a smaller current consumption.

This behavior highlights the operation of the memory acceleration versus the different alignment of the code in the memory.

1.2 Low Power Run mode

The difference between RUN mode and Low Power RUN mode resides in the state of the internal Vcore domain regulator. The Vcore domain voltage is switched to Range 2 - 1.5 V and the regulator is put in Low Power mode. Thus the maximum frequency of the system is 121 kHz. The consequence is that the amount of current powering the peripherals and core is limited. High speed system clock configuration is no longer possible. Please note that this limitation has an influence on the Vcore domain only, the other domains are not impacted. The code executed in this project is the dhrystone-like loop.

1.3 Sleep mode

In this mode, the clocks to the ARM® Cortex™-M3 core are disabled as described in section 4.3.5 "Sleep mode" of the Reference Manual. In the code example, when no peripherals are clocked, the Flash memory is configured to be in Low Power mode.

1.4 Stop mode

This mode can be accessed with the regulator ON or in Low Power mode. The latter allows a lower current consumption to be reached, but increases the wakeup time. You can run the RTC with LSE oscillators and the watchdog to visualize their consumption in real application cases.

1.5 Standby mode

In this configuration, you can obtain the lowest possible current consumption available for the STM32L1 devices. The Vcore domain is switched OFF which reduces the leakage, but the content of the registers in this domain is lost including the internal RAM. The backup domain remains powered, supplying the IWDG, RTC and Low Speed clocks. The system is woken-up by a reset generated from the internal or external source, including the WKUP pin's rising edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time-stamp event, external reset in NRST pin, and IWDG reset.

2 Hardware description

Measurement on Discovery board

The recommended hardware to use for the measurements is the STM32L Discovery board, STM32L-DISCOVERY, featuring an STM32L152xxB device, an ultra-low-power ARM® Cortex™-M3 MCU. Using it on another STM32L1 platform is possible. However, the user should perform the following changes:

- The startup file set in the project must be changed according to the chosen device, and the associated preprocessor #define must be updated.
- The user must add or remove the GPIO definitions according to the selected package.
- The user must reconfigure the LCD pins which are not available in the package.

Jumper JP1 must be replaced by a current consumption measurement tool configured in Averaging mode for more accuracy. There is no need to keep the I/Os in a specific state; they will all be internally configured as Analog inputs, resulting in minimal power consumption because the Schmidt triggers are powered off.

For optimal results, follow these steps:

1. Load the code.
2. Disconnect your Discovery board from the debugger tool by removing the jumpers from CN3.
3. Shut down your power supply and restart it (disconnect and reconnect the USB dongle), to generate a Power On Reset and completely disable the internal debug circuitry of the Cortex™-M3.

If you want to use an external clock as system clock source, please configure it as a square waveform, with the same amplitude as the supply voltage (3 V on the Discovery board) in order not to produce extra consumption on the pad.

Caution: If you want to supply the device with an external power supply source on JP1, refer to the following notes.

Caution: The slilicon limitation present on STM32L15xxB PB7 pin may affect current consumption measurement. When PB7 is configured in analog mode and no PVD level is selected, an internal pull-up is activated on PB7 pin. This may induce current flow through external components such as resistors or diode on the STM32L1-DISCOVERY board. To measure only the MCU current consumption, two solutions are possible:

- Disconnect the LD3 diode connected to PB7 internal pull-up
- Configure PB7 as an input and connect PB7 pin to GND to minimize current consumption.

Note: *It is necessary to unsolder SB100. By default, it connects RST pin to the ST-Link device. However, since the ST-Link device is not powered-on, it would either introduce current consumption on internal NRST pull-up, or even keep the device under reset.*

To be able to reprogram your device, just reconnect this line externally, NRST Pin from side header to Pin 5 of CN6.

3 Summary

Using this firmware, together with the STM32L Discovery board, you can quickly and easily evaluate the low power consumption characteristics of the STM32L family, understand how to interpret the datasheet values and apply this knowledge to real applications.

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
02-Oct-2014	1	Initial release.