AN4365
Application note

Using STM32F4 MCU power modes with best dynamic efficiency

Introduction

Power consumption presents a major challenge for recent embedded applications.

This application note is split into two main parts:

- The first part describes how to best configure the STM32F4 power modes and measure the corresponding current consumptions specified in the datasheet.
- The second part describes a use case that highlights the power efficiency of STM32F4 for the competitive applications of the embedded system market.

This application note is provided with a firmware package that contains two projects:

- *STM32F4_Current_Consumption_Measuring:* gives a measuring application that can be customized to measure the current consumption of different STM32F4 power modes, and shows how to best configure the system for each mode.
- *STM32F4_Low_Power_Application_Case:* this is an example of an application based on the ADC, I2C and DMA peripherals using STM32F4 low-power modes and features.

For more details about these examples, please refer to the "readme" files inside this firmware package.

Table 1. Applicable products and tools

Type	Part numbers
Microcontrollers	STM32F401 line STM32F405/415 line STM32F407/417 line STM32F427/437 line STM32F429/439 line
Tools	32F401CDISCOVERY 32F429I-DISCOVERY STM32F4DISCOVERY

Contents

List of tables

List of figures

1 STM32F4 power consumption

Embedded system designers are currently facing an urgent need to make a trade-off between power efficiency and high performance, especially with recent high-performance microcontrollers such as the STM32F4 series. Low-power modes are implemented with other features to reduce the average current consumption over the life of the application.

This part aims to describe the STM32F4 power modes and the power consumption of each mode, and highlights the features that contribute in power consumption reduction.

1.1 Power modes

In the high-performance STM32F4 series based on the ARM® Cortex™-M4 32-bit/DSP processor cores, designers at STMicroelectronics have highly optimized low-power modes to define the best compromise between low-power consumption, short startup time and available wake-up sources.

In this part, we give an overview of the STM32F4 power mode consumptions.

1.1.1 Low-power mode overview

By default, and after power-on or a system reset, the STM32F4 is in Run mode, which is a fully active mode that consumes much power even when performing minor tasks. Low-power modes are available to save power when the CPU does not need to be kept running.

It is up to the user to switch between power modes during application processing in order to reduce the overall average consumption by keeping the device as much as possible in low-power modes, as described in *Figure 1*. The user should just select each time the low-power mode with features that fit his design requirements.

Figure 1. Power mode switch

MSv34503V1
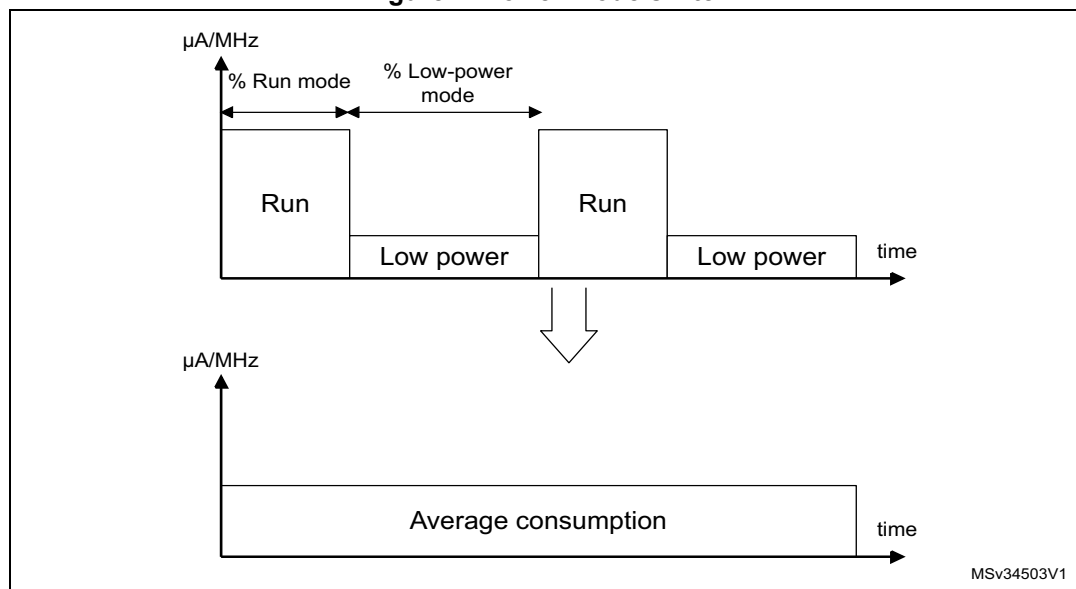
STM32F4 devices feature four main low-power modes:

- **Sleep mode:**
    - Only the CPU clock is stopped.

    The Cortex-M4 clock is stopped and the peripherals are kept running. The current consumption increases with the clock frequency. As in Run mode, the user should be aware of system configuration rules that concern the system clock and voltage regulator scales.

- **Stop mode:**
    - Lowest power consumption while all the SRAM and registers are kept.
    - PLL, HSI, HSE are disabled.
    - All clocks in 1.2 V domain are switched-off.
    - Voltage regulator is working in Normal mode or Low-power mode.
    - Flash memory is working in Stop mode or Deep-power down mode.

    The Cortex-M4 core is stopped and the clocks are switched off (the PLL, the HSI and the HSE are disabled). SRAM and register content are kept. All I/O pins keep the same state as the Run mode. The voltage regulator is working in Normal or Low-power mode and the Flash memory can be configured in Stop mode or Deep-power down mode to save more static power.

- **Standby mode:**
    - The lowest power consumption.
    - The 1.2 domain is powered off (regulator is disabled).
    - SRAM and register contents are lost except in the backup domain.

    The Cortex-M4 core is stopped and the clocks are switched off. The voltage regulator is disabled and the 1.2 V domain is powered-off. SRAM and register contents are lost except for registers in the Backup domain (RTC registers, RTC backup register and backup SRAM), and Standby circuitry. This mode has the lowest current consumption, which depends on the configuration of both backup SRAM (not available for STM32F401x) and RTC.

- **V_{BAT} mode**:
    - The main digital supply (V_{DD}) is turned off.
    - The circuit is supplied through V_{BAT} pin which should be connected to an external supply voltage (a battery or any other source).

    This mode is used only when the main digital supply (V_{DD}) is turned off and the V_{BAT} pin is connected to an external supply voltage (a battery or any other source). The V_{BAT} pin powers the Backup domain (RTC registers, RTC backup register and backup SRAM).

For a further description of these low-power mode features and low-power technology of STM32F4, you can refer to application note "How to achieve the lowest current consumption with STM32F2xx" (AN3430) and to reference manuals RM0090 and RM0368, available from the ST Website.

1.1.2 Power mode power consumptions

Figure 2 summarizes the power consumption of the various power modes of each STM32F4 device.

Figure 2. Current consumption of STM32F4 MCU power modes
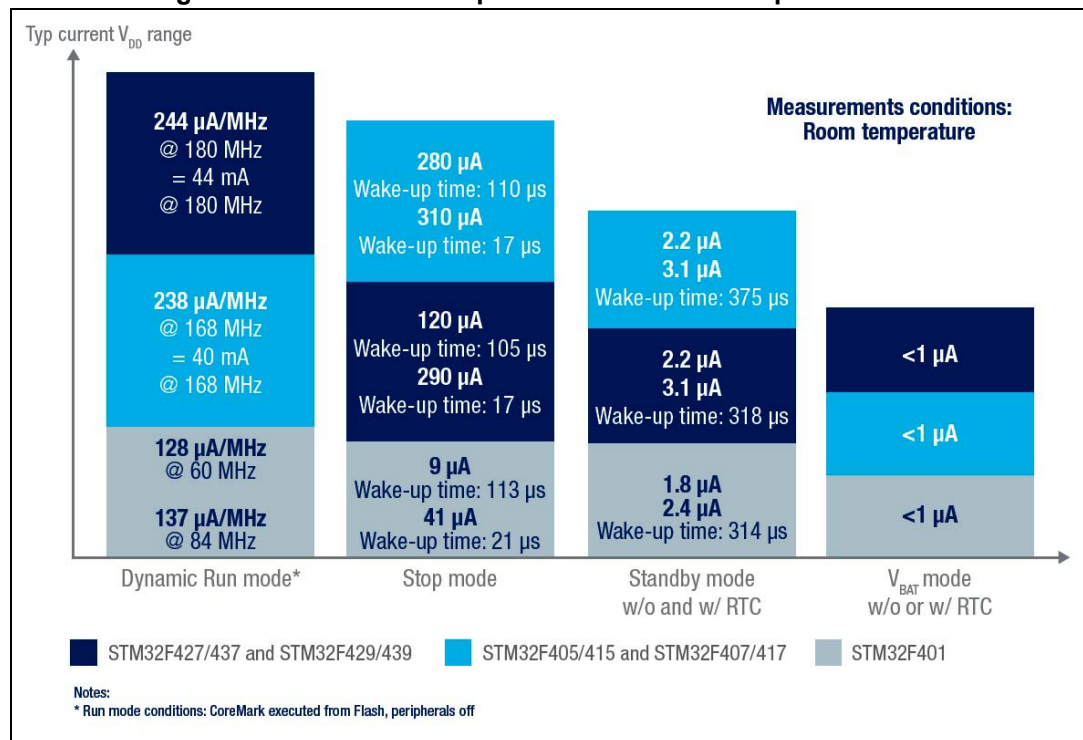
Consumptions of both the Run and the Sleep modes depend on the operating frequency. That is why we find that the dynamic Run mode consumption is presented in *Figure 2* by the ratio µA/MHz at the maximum frequency of each device and when all peripherals are disabled.

For Run mode, the values are measured when the CoreMark benchmark is executed from Flash.

The lowest power drain is achieved when both Standby mode and V_{BAT} mode are used. The Standby mode current is by 2 or 3 µA for all STM32F4 MCUs. The V_{BAT} mode current is less than 1 µA, dedicated to battery use.

The use of these low-power modes is reinforced by the implementation of various power-saving features in order to reach a highly optimized consumption.

1.2 Power-saving methods and features

In this part, we give a brief description of power-saving features that help a lot to reduce current consumption and reach an optimal trade-off between performance processing and power efficiency.

Note: *The majority of these features are common in all STM32F2 and STM32F4 MCUs, so the user can get more details by referring to application note "How to achieve the lowest current consumption with STM32F2xx" (AN3430) and to reference manuals RM0090 and RM0344.*

1.2.1 System clock configuration and management

The clock controller in STM32F4 provides a high degree of flexibility with various clock sources (external crystal HSE, internal oscillator HSI, phase-locked loop PLL, internal oscillator LSI, external oscillator LSE), which are used to run the core and peripherals.

Dynamic power dissipation of CMOS logic is proportional to the operating frequency when the operating voltage is fixed. The system over-clocking should be avoided by slowing the system clock when the maximum rate is not needed.

The user should pay attention to the minimum system clock required by some peripherals that need a specific clock like Ethernet, USB high speed and full speed, I2S and SDIO. Several prescalers are used to configure the AHB frequency, the high-speed APB (APB2) and the low-speed APB (APB1) domains. To optimize power consumption, the user should use the highest prescalers in order to provide just the needed clocks to peripherals and avoid over-clocking that causes a consumption penalty. Power consumption can be further lowered by gating clocks to the APBx and AHBx peripherals when they are not in use.

1.2.2 Dynamic voltage and frequency scaling

Dynamic current: $I = P/V = C * V * f$

So, reducing the operating voltage of the device is a useful step to reduce the overall power consumption. Furthermore, many embedded systems do not require the system's full processing capabilities at all times because not all subsystems are always active. When this is the case, the system can remain in the active mode without the processor running at its maximum operating frequency. The voltage supplied to the processor can be lowered when a lower frequency is sufficient. With such intelligent power management, we reduce the power drawn from the battery by monitoring the processor input voltage in response to the system's performance requirements.

That consists in scaling the STM32F4 regulator output voltage that supplies the 1.2 V domain (core, memories and digital peripherals) when we lower the clock frequency based on processing needs.

STM32F4 MCUs offer the following scales:

- STM32F427x/437x/429x/439x: three voltage scales
    - scale3 (up to 120 MHz)
    - scale2 (from 120 to 144 MHz)
    - scale1 (up to 180 MHz)
- STM32F405x/415x/407x/417x: two voltage scales
    - scale2 (up to 144 MHz)
    - scale1 (up to 168 MHz)
- STM32F401x: two voltage scales
    - scale3 (up to 60 MHz)
    - scale2 (up to 84 MHZ)

1.2.3 Voltage regulator bypass

This feature consists in using an external source for the CPU core voltage instead of an internal LDO regulator voltage output. When bypassing the voltage regulator, we can save up to 50% of power when the supply voltage is 3.3 V and up to 35% in case when the supply voltage is 1.8 V.

1.2.4 Voltage regulator power tricks

There are new tricks implemented in STM32F4 MCUs to reduce current consumption by monitoring the regulator voltage output.

- **Over-drive mode**: this mode allows the CPU and the core logic to operate at a higher frequency than the normal mode for a given voltage scaling (scale1, scale2 or scale3). This mode is available only with STM32F429x/439x devices.
- **Low-voltage mode:** in Stop mode, we can further reduce the voltage output of the low-power regulator by using the Low-voltage mode. This mode is only available in Stop mode with STM32F429x/439x and STM32F401x devices.
- **Under-drive mode**: the 1.2 V domain is preserved in Reduced-leakage mode. This mode is only available in Stop mode when the main regulator or the low-power regulator is in Low-voltage mode. Under-drive mode is available only with STM32F429x/439x devices.

1.2.5 ART configuration

The ART accelerator implements an instruction prefetch queue and instruction/data caches, which increase the overall system performance speed and efficiency by increasing the Flash memory access.

When the user does not need the system's full processing performance and wants to further reduce the consumption, it is recommended to disable the prefetch buffer by software. Disabling the prefetch buffer avoids extra Flash access that consumes almost up to 10% of the overall consumption without affecting much the overall performance, depending on the code alignment on the Flash memory.

1.2.6 I/O configuration

By default, STM32F4 pins are configured as inputs, except some JTAG pins which can impact the power consumption of the device in different power modes because pins are very sensitive to external noise in input mode I/O.

To avoid extra I/O current, all pins should be configured as analog input (AIN); in this mode the Schmitt trigger input is disabled, providing zero consumption for each I/O pin.

We recommend that the I/O speed frequency (driving level) be configured at the lowest possible speed or as an output push-pull configuration, outputting 0 to the ODR. The user should also disable the MCO pin of the clock output if not used.

1.2.7 Using direct memory access (DMA)

STM32F4 devices have peripherals with DMA. This feature is not just useful to improve performance, it can also be used to reduce power. Peripherals with just one buffer register require the CPU to remain in operation in order to read from the buffer so it does not overflow. However, with DMA, the CPU can go to sleep until the DMA transfer completes. This allows the device to consume less average current over the life of the application.

1.2.8 Power mode switching

Power mode switching in application processing reduces the overall average consumption by keeping the device as much as possible in low-power modes. *Figure 1* illustrates this power management method.

For example:

Total power = $I_{average}$ * V, with:

$$I_{average} = (I_{Run} * Time_{Run} + I_{Standby} * Time_{Standby} + I_{Sleep} * Time_{Sleep} + I_{Stop} * Time_{Stop}) / Total\ time$$

1.2.9 Code optimization

The user should apply the best optimization techniques to minimize the number of cycles required for active processing tasks and the active current required.

Using compiler optimization minimizes the time spent executing instructions (speed) and minimizes branches in code (size), and so minimize the current draw.

Making direct read/write to/from registers instead of using firmware which contains many loops and instructions avoids an extra current consumption.

To achieve the lowest current consumption, the user can do extra optimizations which can not be performed by a compiler optimizer. Simplifying the program flow for common cases will avoid significant processing time and reduce the required memory.

2 STM32F4 current consumption measuring

Many users need to know the best configuration that will let them reproduce the datasheet's current consumption values. In this part, we give a firmware that provides the ideal configurations for the power modes and let measure the power modes' current consumption for different STM32F4 devices.

2.1 Current consumption measuring firmware

This section gives details on the firmware, hardware, software/tools environment used to measure the current consumption of different power modes.

2.1.1 Current consumption application description

The STM32F4's current consumption firmware supports all STM32F4 devices: STM32F429x/439x,STM32F40x/41x and STM32F401x.

This firmware aims to let the user measure the dynamic and static current consumption of different power modes using his own discovery board of each STM32F4's device. The user can perform measurements while changing some parameters.

The approach consists in building a kind of power mode switching menu which would let the user switch from one power mode to another using the button and following dedicated LED signals available on the STM32F4 discovery boards.

This measuring application can be used directly to perform and measure all power modes available in each device. On the other hand, the user has the ability to customize the firmware by selecting only the power modes and sub-modes that require measuring. Some parameters can be changed to perform measurements with all the possible configurations of the power modes.

Using the switching menu consists in following the steps below, as described in *Figure 3*:

- After power-on, the green and red LEDs light up, and remain lit until the user presses the button to enter the first selected mode/sub-mode.

- Once the button is pressed, the LEDs blinking routine starts to indicate which power mode/sub-mode will be entered: first, the red LED blinks a number of times that indicate the power mode that will be entered. After that, the green LED begins blinking a number of times that indicates the sub-mode that will be entered.

- Once the LEDs are switched off, the device enters the selected power mode to let the user measure the current. For Run mode, the device executes the CoreMark benchmark, and when it finishes after at least 10 s, both LEDs are turned on. For Sleep, Stop and Standby modes, the device just remains at these low-power modes, waiting to wake-up when the button is pushed. After wake-up, the LEDs light up, waiting again for the user to press the button to switch to the next mode/sub-mode.

- After exiting the power mode/sub-mode, the green and red LEDs light up again, waiting for the user to press the button to switch to the next mode/sub-mode and begin a new power mode measurement. The last mode is Standby mode, and once the user pushes the wake-up button, a reset is generated to start from the beginning.

Note: *For more details about the LED blinking number of each mode and sub-mode, you should refer to tables in the "readme.txt" file.*

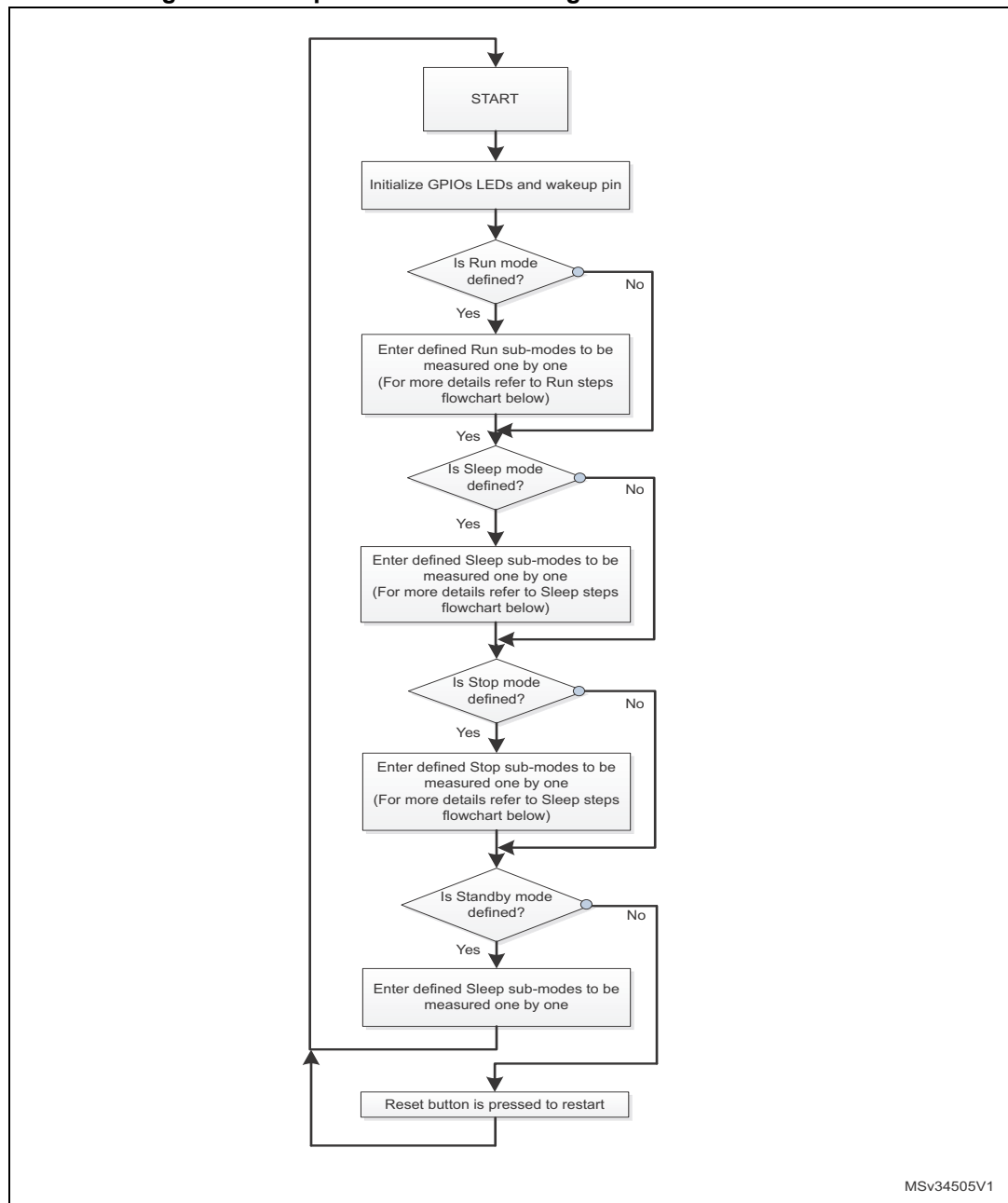**Figure 3. Main power mode switching menu firmware flowchart**

MSv34505V1

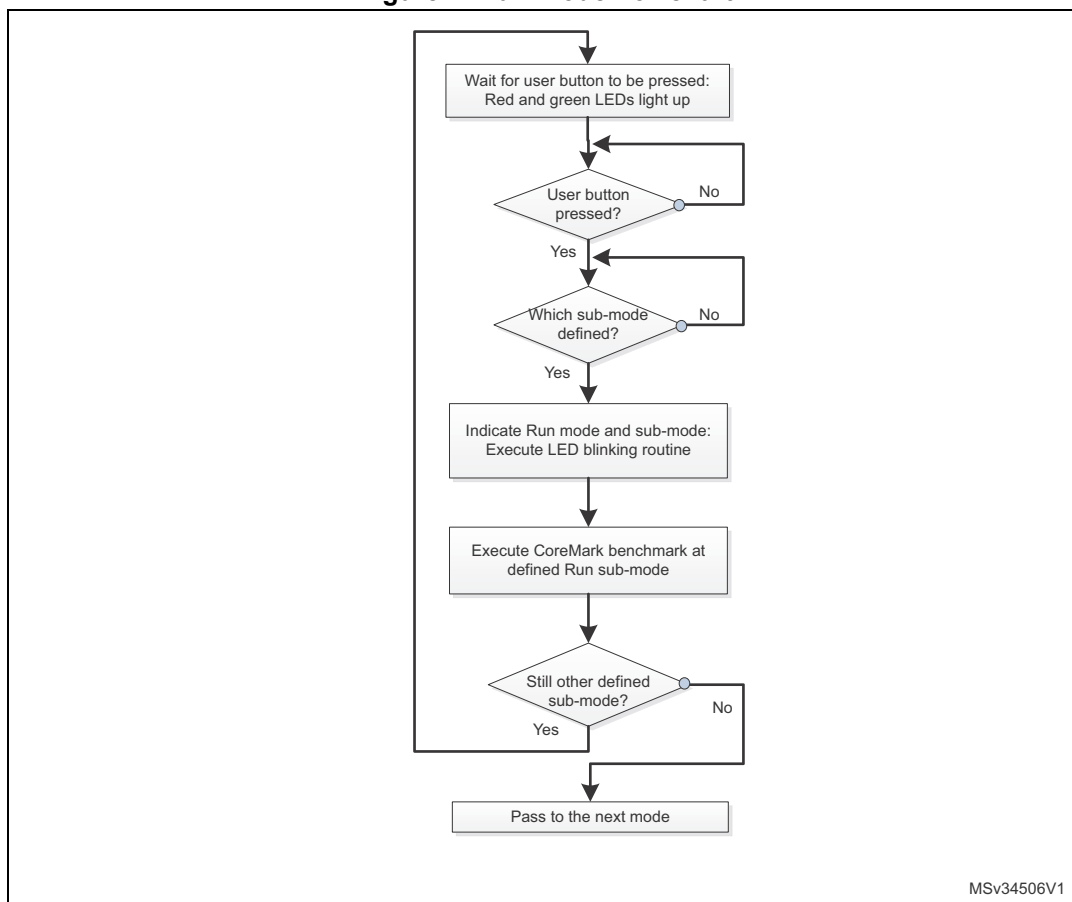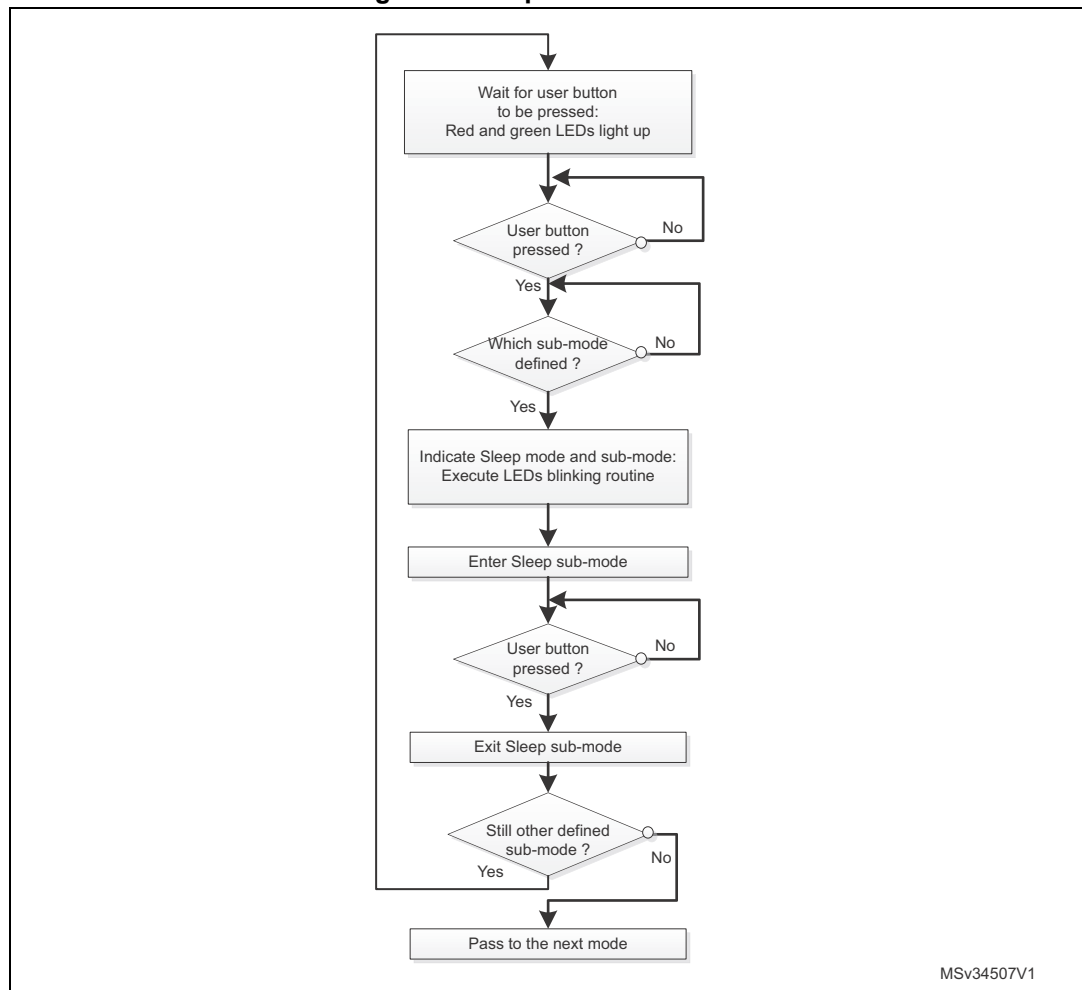**Figure 4. Run mode flowchart**

Figure 5. Sleep mode flowchart

MSv34507V1

Figure 6. Stop mode flowchart
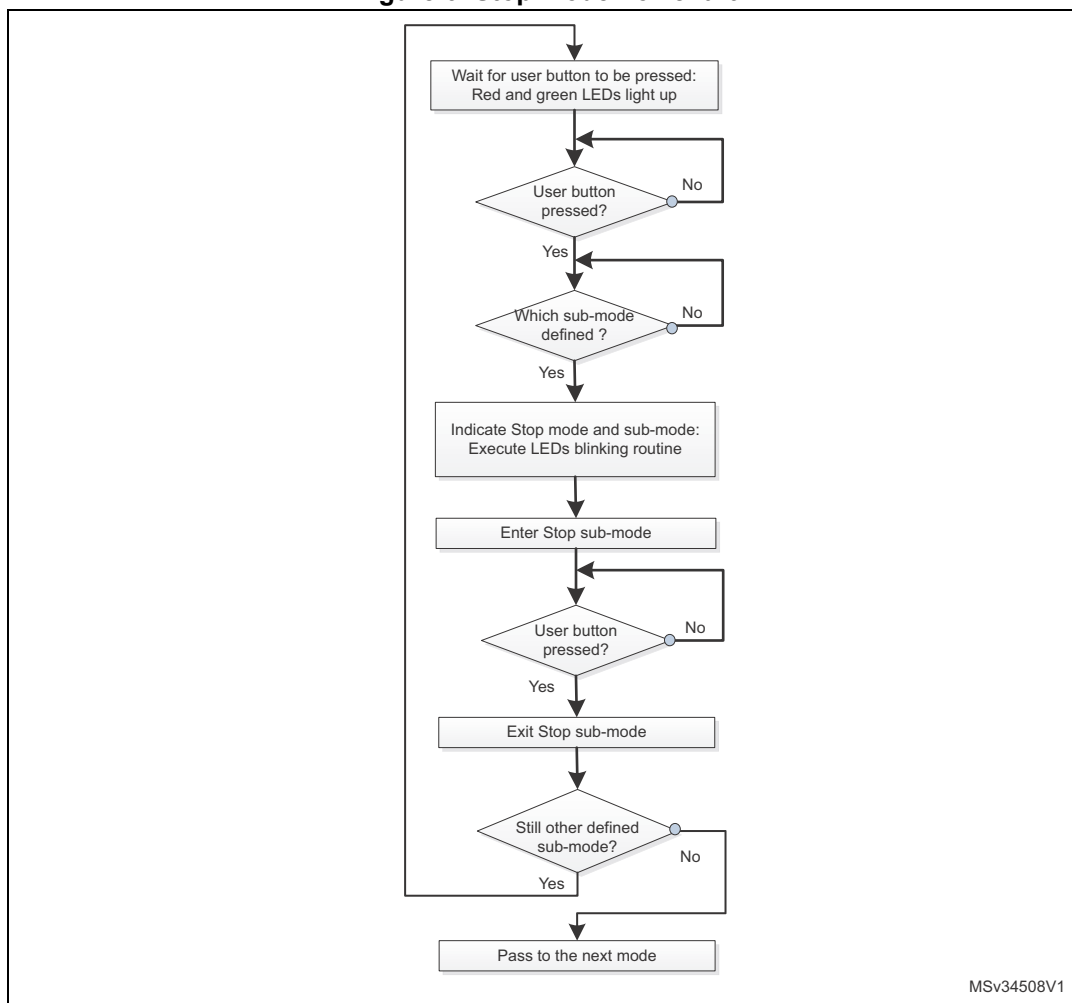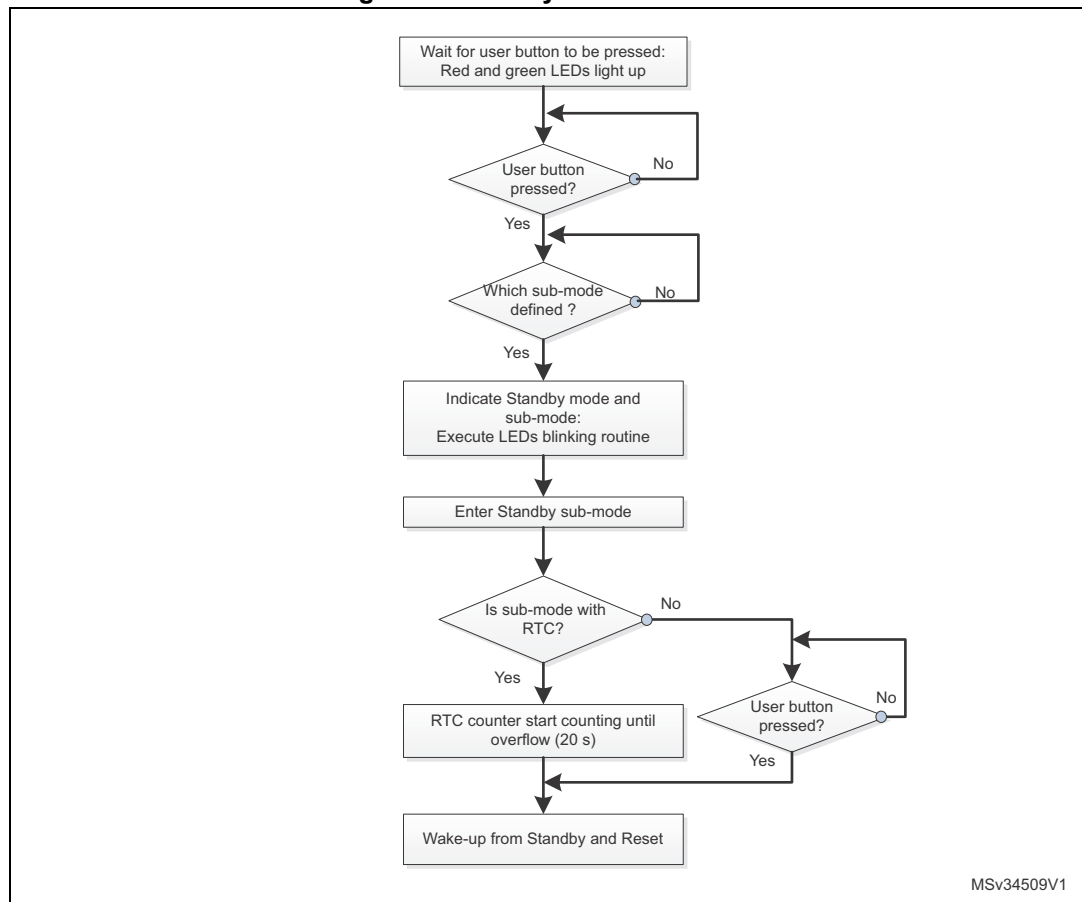

MSv34508V1

Figure 7. Standby mode flowchart

2.1.2 Firmware architecture description

For Run mode, the CoreMark benchmark is executed at the defined system configuration. CoreMark is a standard code developed by the Embedded Processor Benchmark Consortium (EEBMC) to provide a standard set of benchmarks that IDMs subscribe to, in order to measure the performance of their MCUs under "real world" conditions.

Figure 8 shows the different parts of the STM32F4 current consumption measuring project.

Figure 8. Current consumption measuring project overview
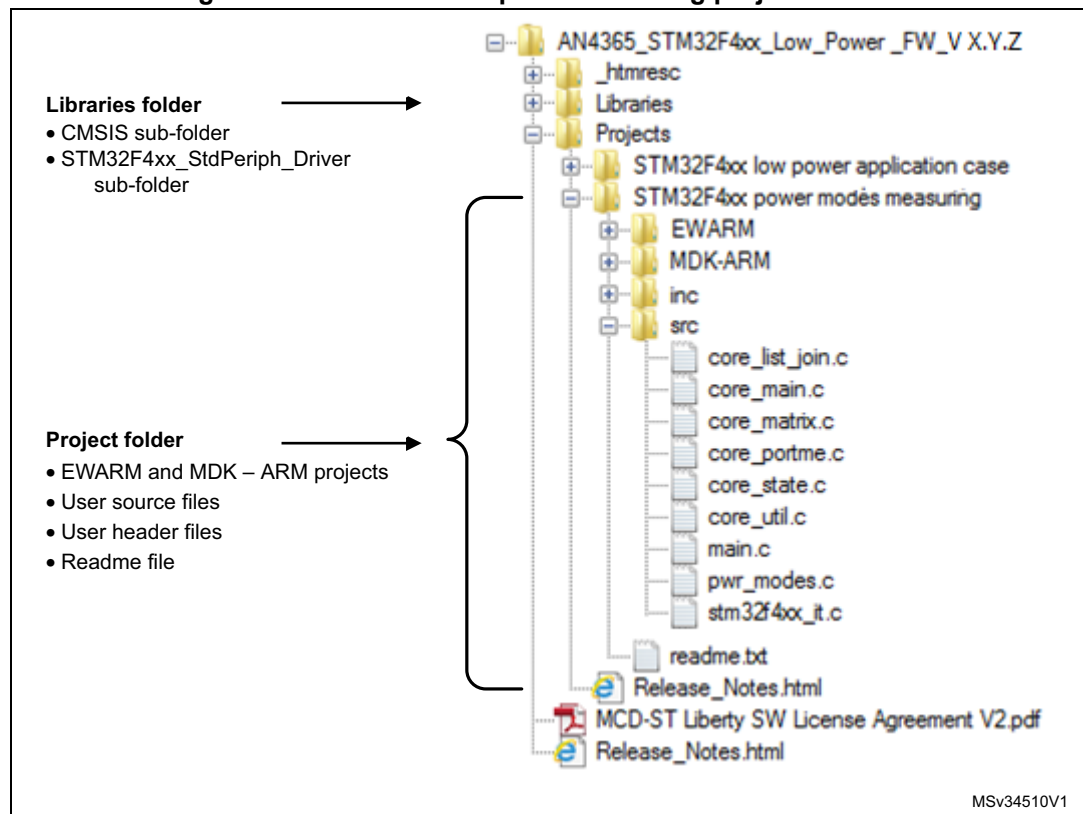
The drivers of the firmware are:

- **main.c:** contains the initialization code and starts the application as per the selected power modes and sub-modes.

- **pwr_modes.c:** contains all the selected sub-mode configurations of all STM32F4 devices.

- **core_portme.c:** controls the CoreMark benchmark.

- **STM32F4_it.c**: contains the interrupt handlers for the application.

- **core_list_join.c & core_main.c &core_matrix.c & core_state.c & core_util.c**: these drivers implement the CoreMark benchmark functions.

- **readme.txt**: this file describes how to use the firmware.

2.1.3 How to use the current consumption firmware

This part provides guidelines on how to use the firmware to achieve the required measurements.

By default, all modes and sub-modes of each device are defined to be measured at power switching menu runtime. The user should just load the required code into the device's Flash memory and start measuring mode by mode. To customize the firmware, proceed as follows:

• **Define Power mode**

As a first step, the user checks the defined power mode in file "main.h" and excludes any undesirable power mode from the measurement process by commenting the corresponding macros.

In this example, we comment the RUN and SLEEP mode macros in order to measure only the Stop and Standby modes.

```
/* Uncomment the corresponding line to select power mode(s) that
will be measured */

//#define RUN_MODE

//#define SLEEP_MODE

#define STOP_MODE

#define STANDBY_MODE
```

To debug code, the user should uncomment the DEBUG macro in "main.h".

• **Define Power sub-modes**

"pwr.modes.h" contains all the power sub-mode macros for all STM32F4 devices. By default, all available sub-modes are defined to be measured. If the user wants to customize his switching menu and measure specific sub-modes, he needs to comment macros of those he will not measure. This way, the firmware will execute only the defined power sub-modes one by one. Only for Standby mode, we can define and measure one only sub-mode because the device will reset at the moment of waking-up.

The following is an example for Stop mode, performed while measuring current consumption for an STM32F429x/439x device. We want to measure only Stop modes with low-power regulator, so we should keep both selected sub-modes, and comment the other ones:

```
//#define StopMainRegFlashStop_enter

//#define StopMainRegFlashPwrDown_enter

#define StopLowPwrRegFlashStop_enter

#define StopLowPwrRegFlashPwrDown_enter

//#define StopMainRegUnderDriveFlashPwrDown_enter

//#define StopLowPwrRegUnderDriveFlashPwrDown_enter
```

After defining the modes and sub-modes, some parameters can be customized, i.e. the ART accelerator status and the supply voltage value.

- Define ART accelerator configuration

We can enable the full ART with both data/instruction caches and prefetch buffer by uncommenting both corresponding macros. We can enable the ART except prefetch by commenting the prefetch macro. See the example below.

```
/* Enable ART */
 #define ART_Enable
/* Enable prefetch when ART is enabled */
// #define Prefetch_Enable
```

- **Define the voltage supply value**

The voltage supply used in the discovery kit while measuring should be specified by uncommenting either the 3.3 V macro or the 1.8 V macro, like in the example below.

```
/* Define the Power supply value */
/* Vdd = 3,3 V */
#define VDD3_3
* Vdd = 1,8 V */
//#define VDD1_8
```

2.2 Current consumption measuring

This section describes the applicable software and hardware environment set-up.

2.2.1 Hardware/software environment description

To start measuring on your STM32F4's discovery board, the minimum requirements are as follows:

- One of STM32F4's discovery boards: STM32F401C-DISCO MB1115B / STM32F4DISCOVERY MB997C / STM32F429I-DISCO MB1075B
- "USB type A to Mini-B" cable, to power the board (through USB connector CN1) from a host PC and to connect to the embedded ST-LINK/V2 for debugging and programming
- An ammeter
- A power supply

Before you start measurements, you should establish the connection with the STM32F4 discovery board as shown in *Figure 9* by proceeding as follows:

- Connect the supply voltage cables to the dedicated external supply input pins (GND pin for the ground, and 3 V pin for the 3.3 V or 1.8 V). LED LD2 (PWR) will light up.
- Remove the Idd jumper of the discovery board dedicated to current consumption measurement (JP1 for STM32F40x/41x, JP2 for STM32F401x and JP3 for STM32F429/439x) and connect the ammeter cables. Ensure that the ammeter is set to the mA scale.

Figure 9 shows an Idd jumper circuitry of STM32F429I-DISCO MB1075B. (For more details, please refer to user manual UM1670 *Discovery kit for STM32F429/439 lines*).

**Figure 9. Jumper for Idd consumption measuring**
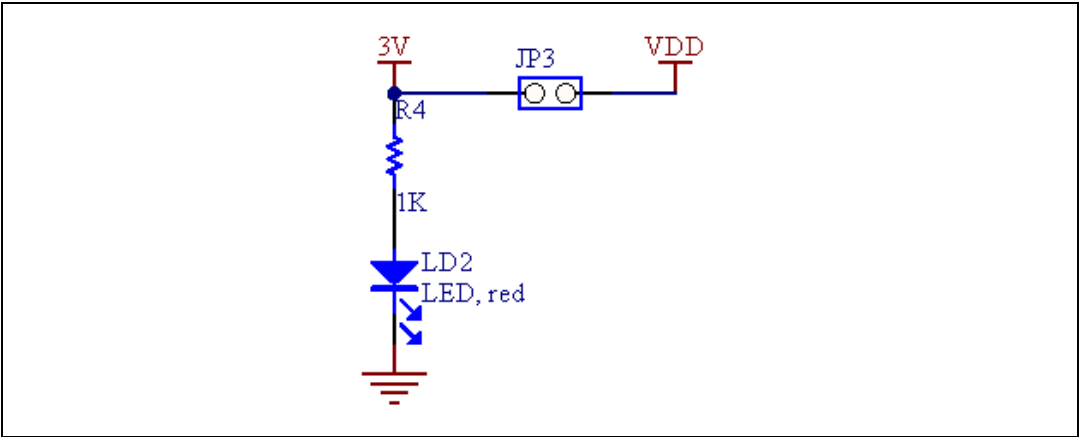
You can refer to *Table 2* to know more about the used pins described in previous parts.

Table 2. Used pins description

Discovery board	Idd jumper	3 V pin of V_{DD} supply	Ground
STM32F4DISCOVERY MB997C	JP1	Pin 5 or Pin 6 of header 25X2 (P1)	Pin 49/50 of header 25X2 (P1)
STM32F401C-DISCO MB1115B	JP2	Pin 5 or Pin 6 of header 25X2 (P1)	Pin 49/50 of header 25X2 (P1)
STM32F429I-DISCO MB1075B	JP3	Pin 1 or Pin 2 of header 32X2 (P1)	Pin 63 or Pin 64 of header 32X2 (P1)

To start measuring, the user should establish the connection of the described parts as shown on *Figure 10*.

Figure 10. Hardware/software environment setting for V_{DD}=1.8 V
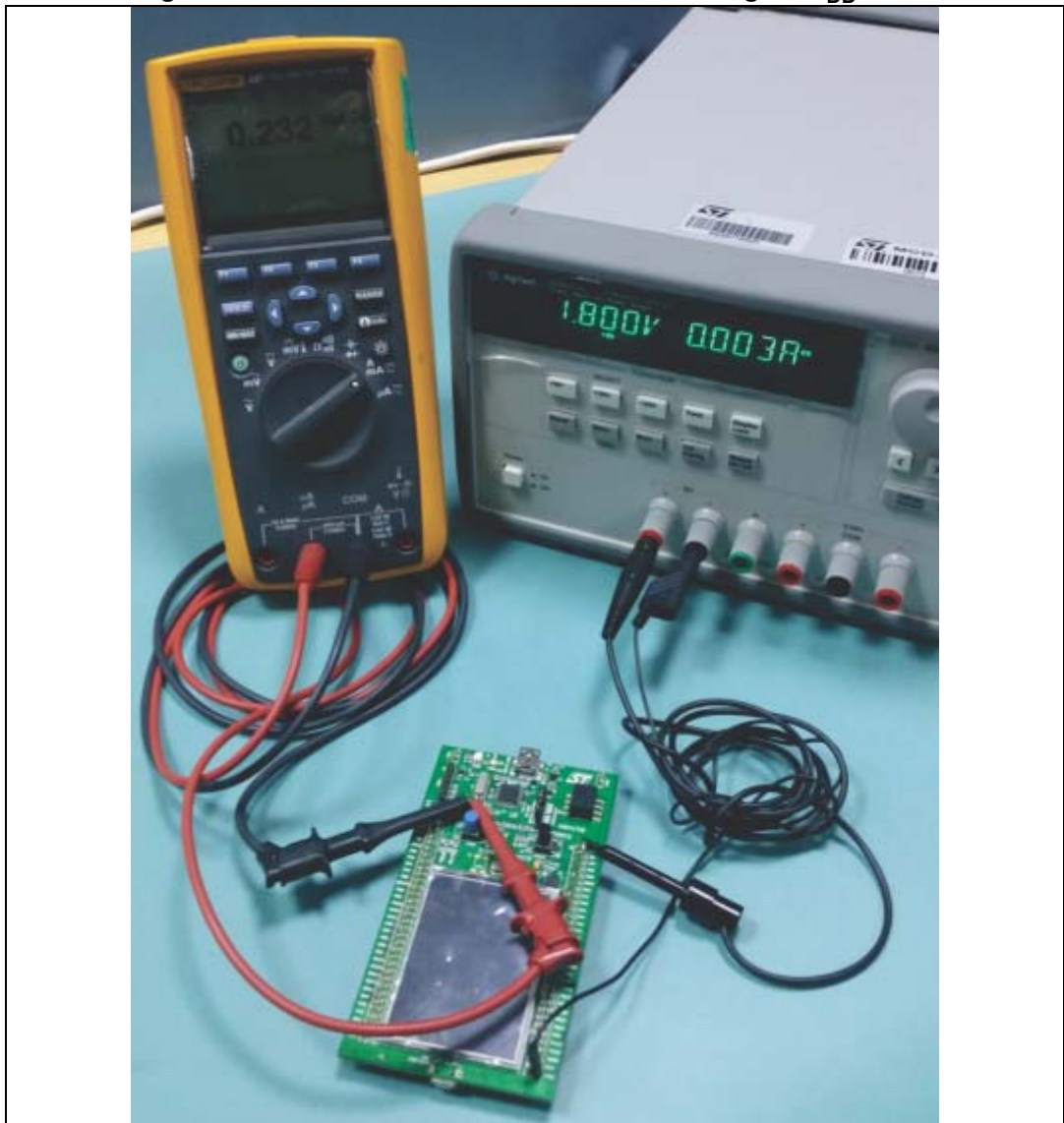
2.2.2 Consumption measuring best practices

After setting the hardware and software environment, the user can run the STM32F4 power consumption firmware and start measuring the current consumption in each mode.

To measure current consumption with the STM32F4 discovery board using the power consumption switching menu, you should proceed as follows:

- **If Run mode** is defined, press the button to enter the first selected Run sub-mode. Once the LEDs stop blinking, the device starts executing the CoreMark benchmark and the user should measure the corresponding current before the LEDs return to lighting status after almost 10 s.
- **If Sleep mode** is defined, press the button to enter the first selected Sleep sub-mode. Once the LEDs stop blinking, the device enters the first selected Sleep sub-mode.

Once the user has taken the measurement, he should press the button to wake-up from sleep and pass to the next sub-mode or mode.

- **If Stop mode** is defined, proceed as described for Sleep mode.

- **If Standby mode** is defined, press the button to enter the selected standby sub-mode. If the sub-mode is with RTC and the wake-up button is not pressed, the device will wake-up and reset after 20 s. The user should make the measurement before wake-up. If the defined sub-mode is without RTC, the user should press the button to wake-up the device after finishing the measurement.

The user should be aware about some constraints while performing the measurement:

- For Run, Sleep and Stop modes, the user should put the ammeter in mA scale. For Standby mode, the user should switch to the µA scale of the ammeter to be able to measure the standby current.

- It is recommended to connect the power supply source to the discovery board only after power-on at the defined voltage supply (3.3 V or 1.8 V).

- The user should ensure that only one of the two power sources is connected: either the USB connector or the power supply source.

- Another concern is the voltage drop out of the ammeter. When set to a low-current range for measuring static power (stop, standby), there is usually a notable drop across the ammeter due to its impedance. This can interfere with voltage-sensitive case (1.8 V measurements), and often a power-down reset will occur. This can be avoided by increasing the power supply voltage output or using regulation techniques relative to the V_{DD} input to the device.

3 STM32F4 low-power application case

Sensors are a fundamental part of the human/machine interface. This part of the application note provides an example of a sensor hub based on STM32F4 devices and focuses on the low-power modes used to reduce the consumption of this application.

The figure below describes a complete sensor hub example, from which we built the simplified application described in the next section.

Figure 11. STM32F4 sensor hub example
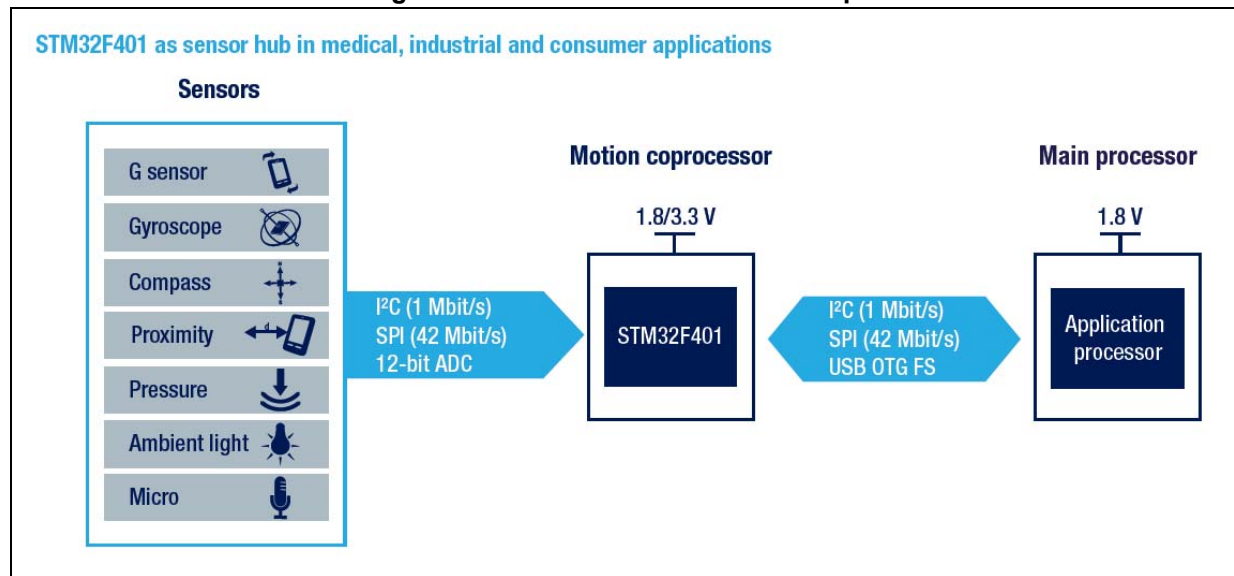
The next parts of this application note describe the following main items:

- Application case overview
- Software and hardware environment
- Current consumption measurement

3.1 Application overview

This application consists of a communication between a sensor hub and a sensor based on two STM32F4 devices. One device is configured as master (sensor hub) and the other as slave (sensor).

The main features of the application are:

- ADC with 1024 V_{BAT} voltage samples
- Low-power techniques described in the first chapter
- Sensor hub FFT samples transformation with maximum device frequency

3.1.1 Application functional description

● **Hardware description**

The main STM32F4 modules used by this application are:

– **ADC peripheral:** used to convert voltages coming from V_{BAT}

– **DMA peripheral:** used to transmit data from the ADC to memory, from memory to I2C and from I2C to memory, and to reduce the CPU load

– **I2C peripheral:** configured as Slave Transmitter and Master Receiver and used to control the transmission between devices

– **RTC peripheral:** used to wake-up CPU from Standby after a defined wake-up time

– **User button:** used to startup the application

Figure 12. Application modules and data flow
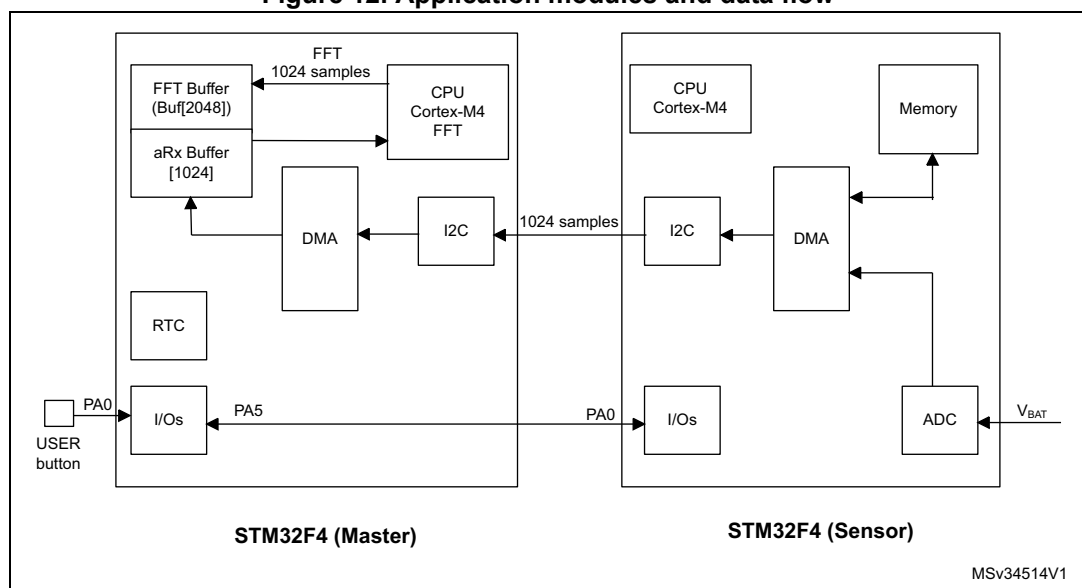
● **Software description**

This software example makes the application operate in master and slave configurations.

Two devices are connected and set up both the transmit and receive data between each other using a common I2C bus.

The state machine in *Figure 13* describes the application and the procedure of the transmission between the two devices.

Initially, the master/slave pair remains in Standby mode.

First, the user presses the user button and the application starts. Then, the master wakes up and wakes the slave by toggling pin PA5 (connected to slave PA0) and then enters in low-power mode (Sleep).

Then, on the Slave side, the ADC starts conversions of V_{BAT} voltages and the DMA stores these converted values in memory. During this operation, the CPU stays in Sleep mode.

Once the DMA transfer is completed, the CPU wakes up just to configure the I2C slave to start transmitting data to master and returns back to sleep mode until the slave (sensor) finishes the transmission.

On the Master side, the I2C receives data from the slave and proceeds the same way as the slave.

When the DMA completes transfer to memory, the CPU takes in charge the FFT transformation of the data in the memory and goes again to standby, but this time for a period of about 200 ms controlled by the RTC peripheral. Once this period is elapsed, the CPU restarts the operations loop.

Figure 13. Application state machine
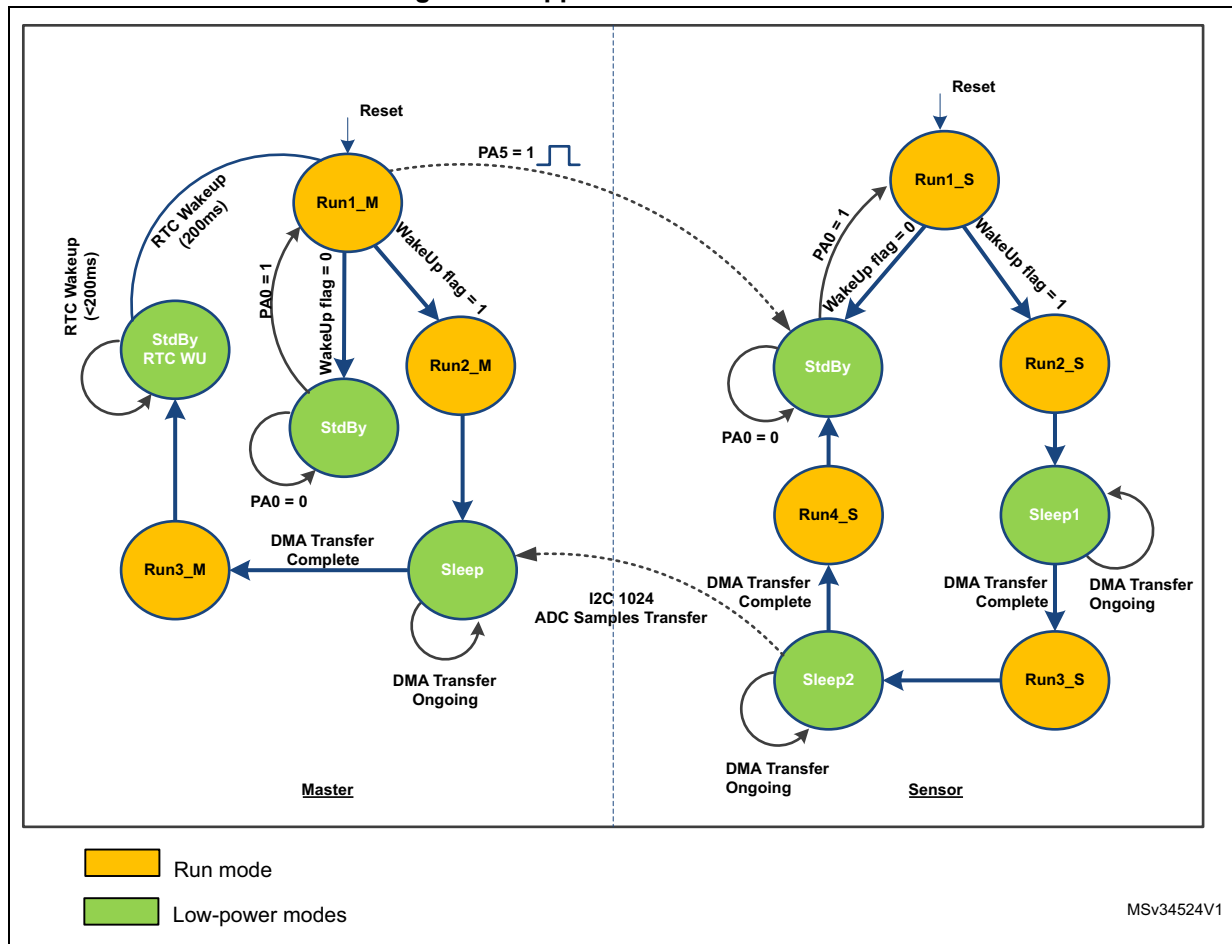
Low-power state machine

Stdby: when the device is powered-on or reset, it enters Standby mode, waiting for a user button push to wake up.

Run1_M: system clock configuration

Run2_M: I2C and DMA configurations

Sleep: DMA samples transfer during CPU Sleep mode

Run3_M: clearing of DMA pending flags on receiver stream and CPU FFT data transformation starts with the maximum device frequency.

StdBy RTC WU: the CPU stays in standby for about 200 ms, waiting for the RTC wakeup

Sensor (Slave) state machine

Run1_S: system clock configuration

Run2_S: ADC and DMA configurations and start of V_{BAT} sampling

Sleep1: DMA samples transfer to memory during CPU Sleep mode

Run3_S: I2C and DMA configurations and start of data transmission to the sensor hub

Sleep2: DMA samples transfer from Sensor to Master via I2C during CPU Sleep mode

Run4_S: clearing of DMA pending flags on transmitter stream

Stdby: once operations are finished, the sensor CPU enters in Standby mode waiting for a request from the Master to wake-up again.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Standby, for example. This is due to the fact that the CPU is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. This is done by uncommenting `#define DEBUG` in main.h

Auto-wakeup

The RTC (clocked by LSI ~32 KHz) provides a programmable time base for waking-up the sensor hub from Standby mode at regular intervals (~200 ms).

Note: *To follow the functional behavior of this application, the user has to visualize signals with an oscilloscope or go in Debug mode.*

3.1.2 Firmware architecture description

This application uses the STM32F4 library and contains the following main source files:
- **main.c:** contains the overall needed functions for the application
- **STM32F4_it.c:** contains the interrupt handlers for the application
- **STM32F4_system.c:** contains the system clock configuration for the STM32F4 devices used in the application

Figure 14. Firmware architecture overview
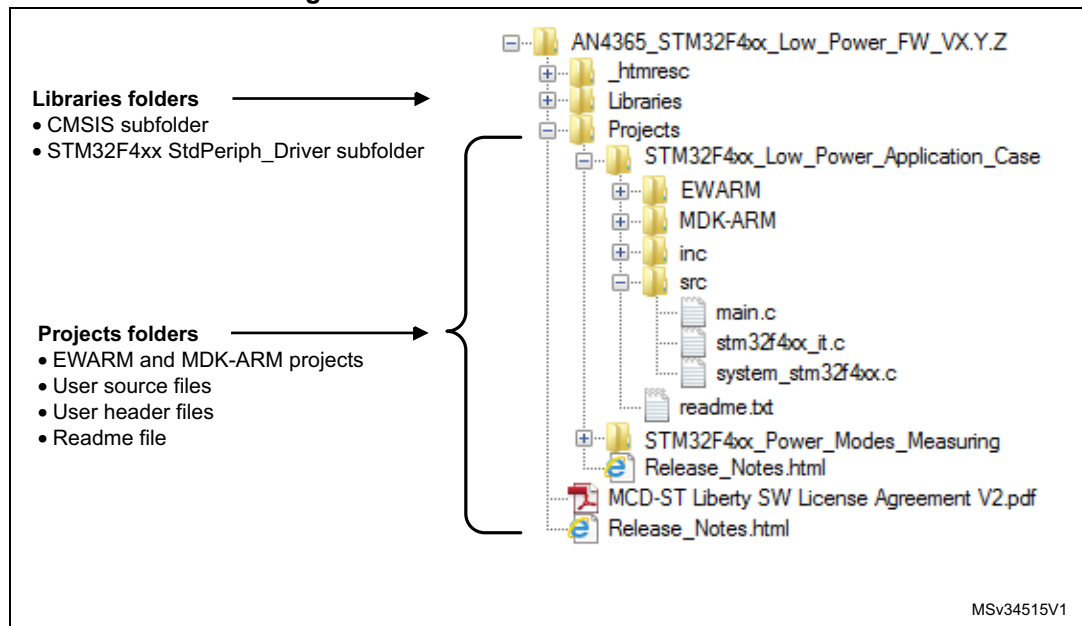
3.2 How to use the application
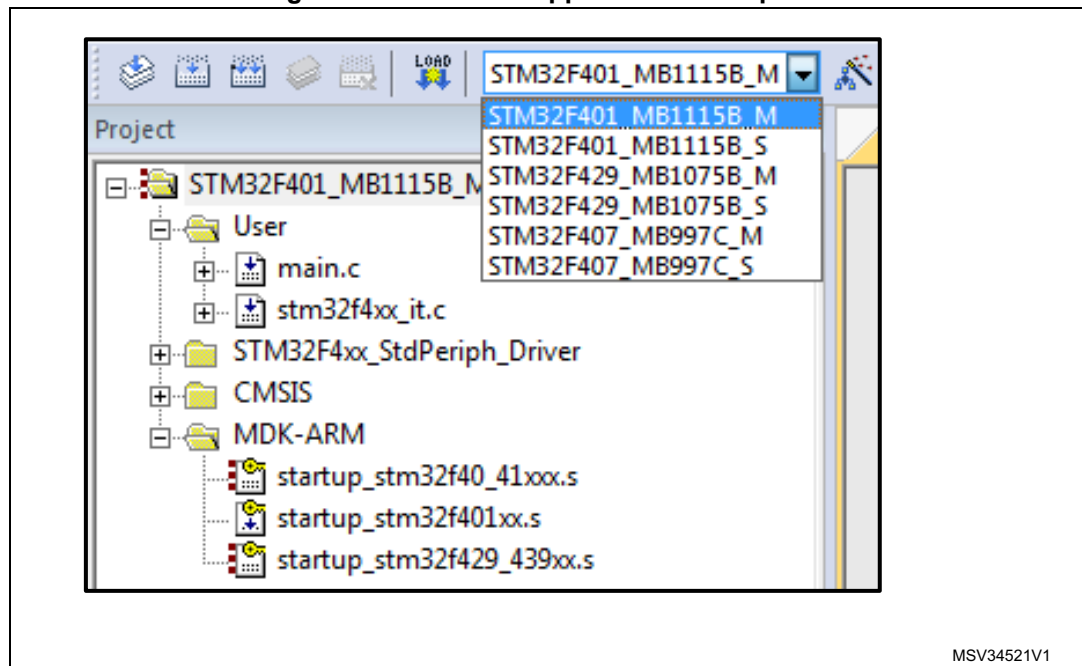
3.2.1 Software requirements

To use this example, you need to load it on two STM32F4 discovery boards. In the toolchain interface, choose the dedicated workspace to configure the I2C peripheral as a Master or Slave device.

This application offers six workspaces:

- STM32F401_MB1115B_M
- STM32F401_MB1115B_S
- STM32F429_MB1075B_M
- STM32F429_MB1075B_S
- STM32F407_MB997C_M
- STM32F407_MB997C_S

The **STM32F401_MB1115B_M**, for example, is a workspace for **STM32F401**x devices with the **MB1115B** Discovery board and configured as **M**aster (**M**). To use a configuration as sensor, the user should choose the workspace with suffix "**_S**", such as STM32F401_MB1115B_**S.**
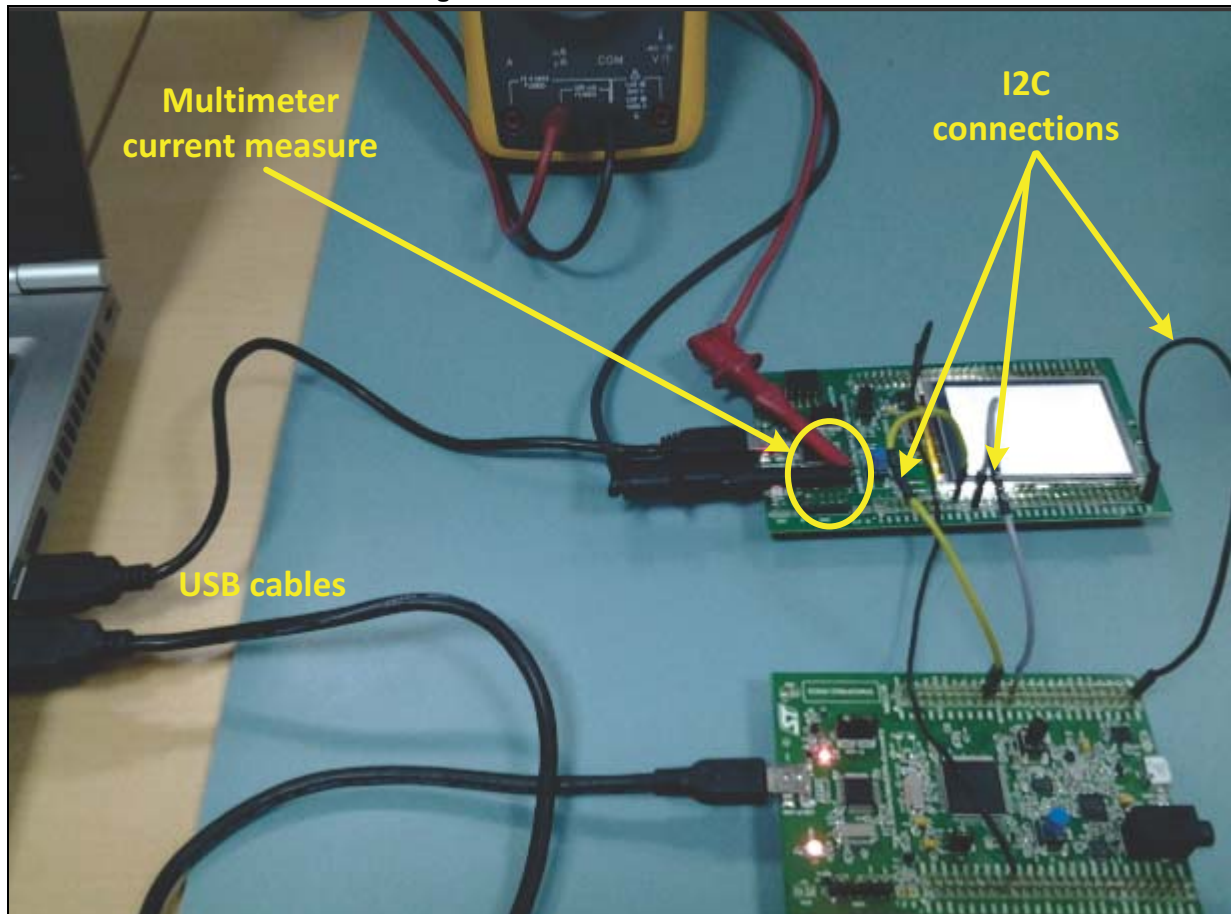
Figure 15. MDK-ARM application workspaces

MSV34521V1

3.2.2 Hardware requirements

To setup the application, the user should connect the two discoveries through I2C lines and GND, as shown in *Figure 16*.

Figure 16. Hardware connections

To run the application on STM32F4 devices, the following steps are required:

1. Connect the two I2Cs of the two devices.
2. Power both discoveries through mini USB connectors.
3. From Windows, launch the adequate toolchain and load the two boards with Master and sensor programs.
4. Connect the multimeter to the desired board instead of each discovery Idd jumper (JPx) as described in *Table 2*.
5. Press the master user button to start the application and then the multimeter displays the average power consumption of each device.

Note: *The R30 resistance should be removed from the MB1115B discovery.*

*The R31 and R34 resistances should be removed from the MB997C discovery.*

3.3 Current consumption measurement

The average current consumption is the sum of the total energy consumed by the system in Dynamic and Static power modes, divided by the average system loop time. The average current is important because it provides a single value, which can be used to accurately determine battery life or the total energy use of the system.

This application involves speed (maximum device frequency used for FFT), complexity (sampling and transfer of 1024 samples) and STM32F4 low-power efficiency using the Sleep and Standby modes associated with other low-power techniques (described in the previous chapter).
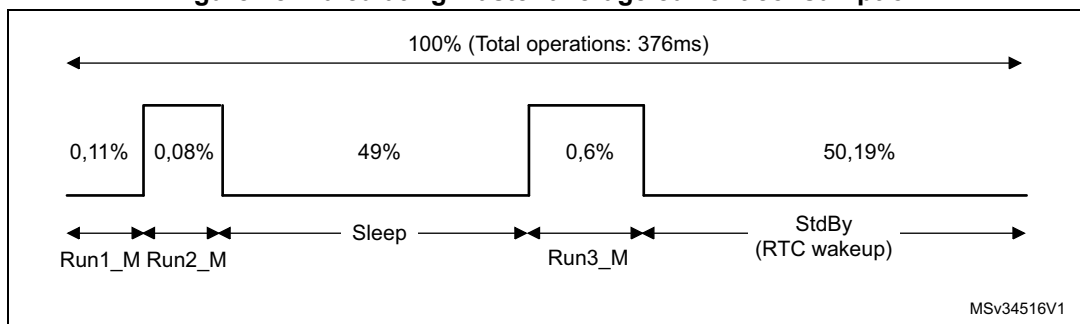
3.3.1 Measuring current consumption

Figure 17. Application waveform overview
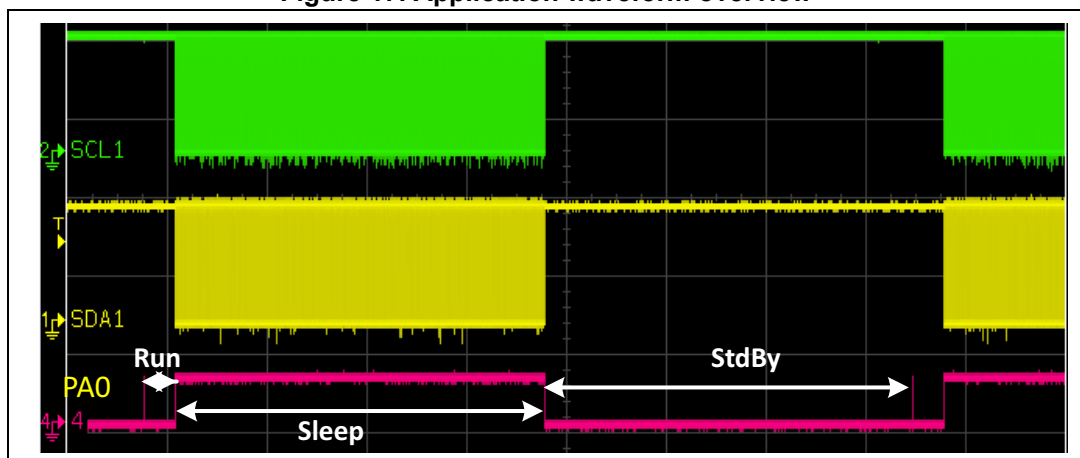
Master current consumption

**Figure 18. Calculating Master average current consumption**

Average Current (mA)~= (**Run1_M***0,11 + **Run2_M***0,08 + **Sleep***49 + **Run3_M***0,6 + **StdBy***50,19)/100

Note: *The standby period on the Master side is controlled by the RTC which is driven by the LSI RC oscillator. The LSI varies from 17 KHz to 47 KHz (refer to the STM32F4 datasheet). This means that the period can be changed from one device to another, and this can slightly impact the consumption.*

Sensor current consumption

Figure 19. Calculating sensor average current consumption
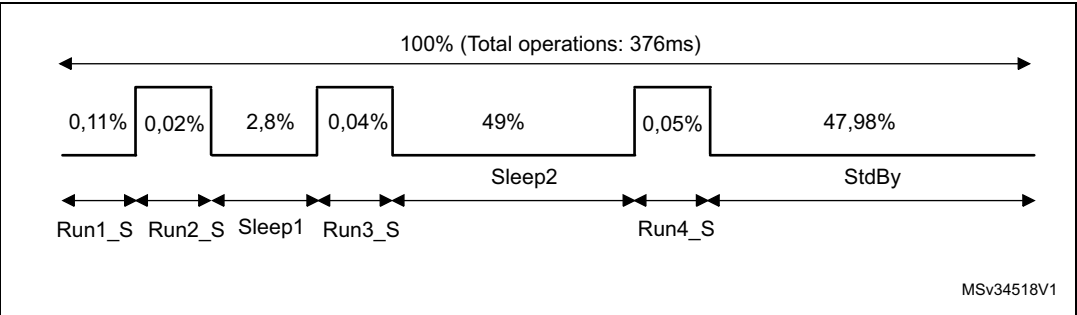
Average Current (mA)~= (**Run1_S***0,11 + **Run2_S***0,02 + **Sleep1***2,8 + **Run3_S***0,04 + **Sleep2***49 + **Run4_S***0,05 + **StdBy***47,98)/100

3.3.2 Current consumption results

Table 3 shows some measures taken with a firmware compiled with the MDK-ARM V4.73 toolchain and device powered with 3 V. For more details about hardware requirements, refer to *Section 3.2.2*.

Table 3. Average measurement results

Device	Consumption (mA)	
	Master	Sensor
STM32F401xC	1.10	1.75
STM32F429xx	1.60	2.00
STM32F407xx	1.60	2.05

4 Conclusion

This application note demonstrates the low-power features integrated into the STM32F4 series that ensure the trade-off between power efficiency and high performance.

The first part of the document provides an overview about these power features and describes how to well configure and measure the different power modes.

The second part is based on an application case of communication between master and sensor that takes the advantage of different STM32F4 low-power features to reach a lower current consumption.

5 Revision history

Table 4. Document revision history

Date	Revision	Changes
28-Mar-2014	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com