System-on-chip Architecture

Lab 1: STM32F0-Discovery low power modes

Robert Margelli, 224854

Contents

1. Purpose of the lab session

The purpose of this lab sessions was to get familiar with the STM32F0-Discovery evaluation board and the low-power states of its MCU.

2. Low power states

The STM32F051x8 MCU provides 4 low power modes, which can be exploited to save energy when the CPU is idle. Normally the CPU is in the run mode. The following is a brief description of the LP modes and their characteristics:

1. Sleep: the least power efficient mode after run mode; wake up by using EXTI line (user button, PA.00); program execution continues from where it was left; the CPU is off but peripherals keep running and can be used to wake up the CPU when receiving an interrupt signal.
2. Stop: automatic wake up using RTC (all clocks are stopped); program execution continues from where it was left; FLASH in deep power down mode; Regulator in LP mode; HSI, HSE OFF and LSI OFF if not used as RTC Clock source.
3. Standby: the less power-hungry state; as in the sleep state wake up using the user button (PA.00); 1.8V domain powered-off; RTC and LSI OFF; when exiting the standby state the MCU is reset. If wanting to continue program execution from a certain point, the user can resort to five backup registers to save data, which will be preserved even when supply voltage is not present or the system receives a reset like in this case.
4. Standby with RTC: similar to the above but RTC is kept on (and clocked by the LSI) and is used to for an automatic wake up.

3. Program details and measurements

To test the low power states, a simple program that walks through them was implemented.
From the RUN state the STOP is reached by pressing the user button; then after approximately 5 seconds the RUN state is reached again and with a user button press the system goes into SLEEP from where it can be woken up by pressing again the same button; finally an additional button press leads to STANDBY with RTC mode which is automatically exited after 8 seconds, a reset signal is sent to the MCU and program execution restarts from the beginning.
Further details on the program can be found in the source code itself. Only files "main.c" and "stm32f0xx_lp_modes.c" were modified.

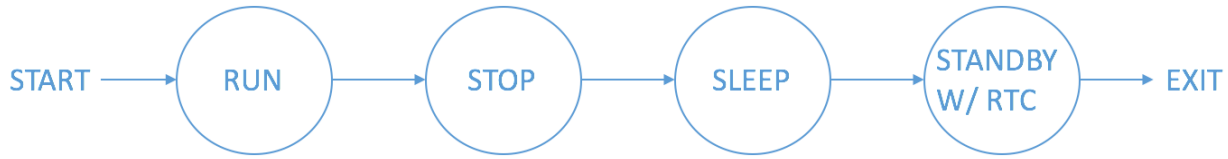

Figure 1 – Concept of the implemented program

A digital multi-meter was used to measure current consumption in the various CPU states.
Table 1 reports the current measurements and compares them with the expected values provided by the MCU manufacturer (refer to pages 50-52 in the MCU's datasheet).
Measurements will rarely match the expected ones due to several reasons. In first place, code differences lead to different power consumptions; secondly it was not possible to have the same operating conditions (mainly temperature) as those used by the manufacturer; finally, different measurement equipment lead to non-matching values.
Overall the measured values are somewhat similar to the expected ones (the biggest difference being found in the run state).
Note: by placing the stop mode invocation before the sleep mode, the PLL is turned off after exiting the stop state and the resulting clock frequency is 8MHz. Under this condition the sleep mode's current consumption is expected to be 0.6mA and not 14.0mA as in operation at 48MHz.

State	Expected Current Consumption [mA]	Measured Current Consumption [mA]	ULP Measured Current Consumption [mA]
Run	22	14.634	14.656
Sleep	0.6	0.807	0.810
Stop	0.004	0.0084	0.0084
Standby with RTC	0.0013	0.0058	0.0058

Table 1 – Current measurements in the various states

To furtherly reduce current consumption the original program was extended by selectively turning off unnecessary peripherals (the list of which can be found in comments to the code itself) in the `StopMode_Measure()` function.
Results for this ultra-low power (ULP) mode yielded no particular improvements with respect to the previous version. This is due to the fact that to turn on a peripheral for its use the program should specify it and in our case none of the peripherals on the above mentioned list were used before entering a low power state. The small differences in the measurements are due to the previously mentioned non-idealities.

4. Conclusions

By measuring currents with the multi-meter, I was able to effectively test the low power modes and asses their purpose of energy saving. In energy-aware applications (such as small IoT devices running on a battery) the CPU can be placed in one of these states when idle and be woken up when an external signal is received (eq. interrupt), guaranteeing big energy savings.