Embedded systems power consumption issues

Text: Chapter 3.7

STM32L100RC Technical Reference Manual, Chapter 5

STM32L4x5 and STM32L4x6 Reference Manual, Chap. 5.3

CPU power consumption

- Most modern CPUs are designed with power consumption in mind to some degree.

- Power vs. energy:
  - heat depends on power consumption;
  - battery life depends on energy consumption.

CMOS power consumption

- Voltage drops: power consumption proportional to V^2.
- Toggling: more activity means more power.
- Leakage: basic circuit characteristics; can be eliminated by disconnecting power.

- Dynamic power consumption: occurs during switching of ON/OFF of n and p networks
- Static power consumption: "leakage" current (I_{DDQ}) during quiescent period (no switching), when one of n and p network should be OFF

CPU power-saving strategies

- Reduce power supply voltage (reduce V^2).
- Run at lower clock frequency.
  - Reduce dynamic energy consumption.
- Disable function units when not in use.
  - Reduce dynamic energy consumed.
  - Ex. Disable clocks that control switching in units
- Disconnect parts from power supply when not in use
  - Reduce both static and dynamic power.

Power management styles

- Static power management: does not depend on CPU activity.
  - Example: user-activated power-down mode.
    - Initiate with some instruction/control register
    - Interrupt to exit this mode
- Dynamic power management: based on CPU activity.
  - Example: disabling of function units.

Application: PowerPC 603 energy features

- Provides doze, nap, sleep modes.
- Dynamic power management features:
  - Uses static logic.
  - Can shut down unused execution units.
  - Cache organized into subarrays to minimize amount of active circuitry.

PowerPC 603 activity

- Percentage of time units are idle for SPEC integer/floating-point:

unit	Specint92	Specfp92
D cache	29%	28%
I cache	29%	17%
load/store	35%	17%
fixed-point	38%	76%
floating-point	99%	30%
system register	89%	97%

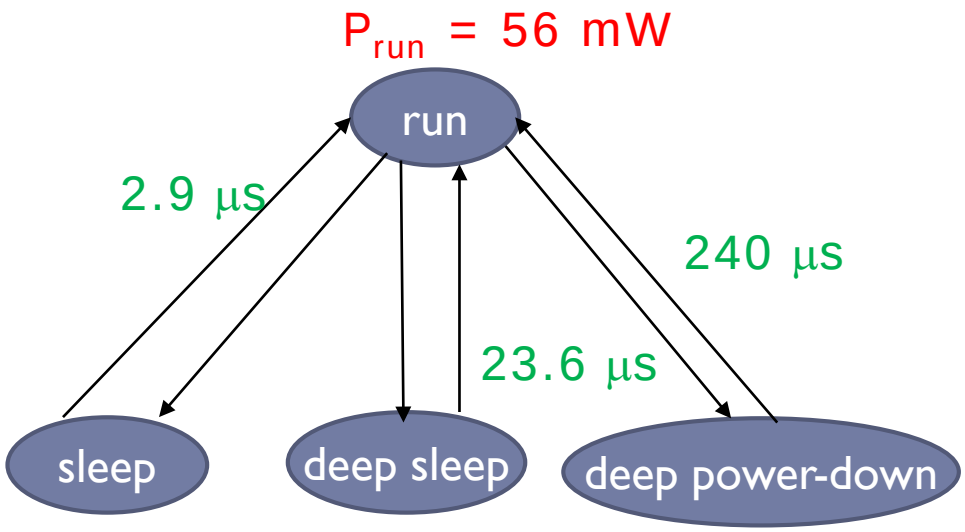
Power-down costs

- Going into a power-down mode costs:
  - time;
  - energy.
- Must determine if going into mode is worthwhile.
- Can model CPU power states with power state machine.

Application: NXP LPC 1311

- LPC 1311 has an ARM Cortex-M3 CPU
- Four power-management modes:
  - Run: normal operation.
  - Sleep: stops CPU clock, with CPU and peripheral logic still powered. Peripherals may initiate interrupts that wake up the CPU.
  - Deep sleep: stops CPU clock, powers down most analog blocks, maintains CPU and peripheral registers and SRAM values.
  - Deep power-down: power and clocks shut down to entire chip, registers and SRAM not maintained.

LPC1311 power state machine

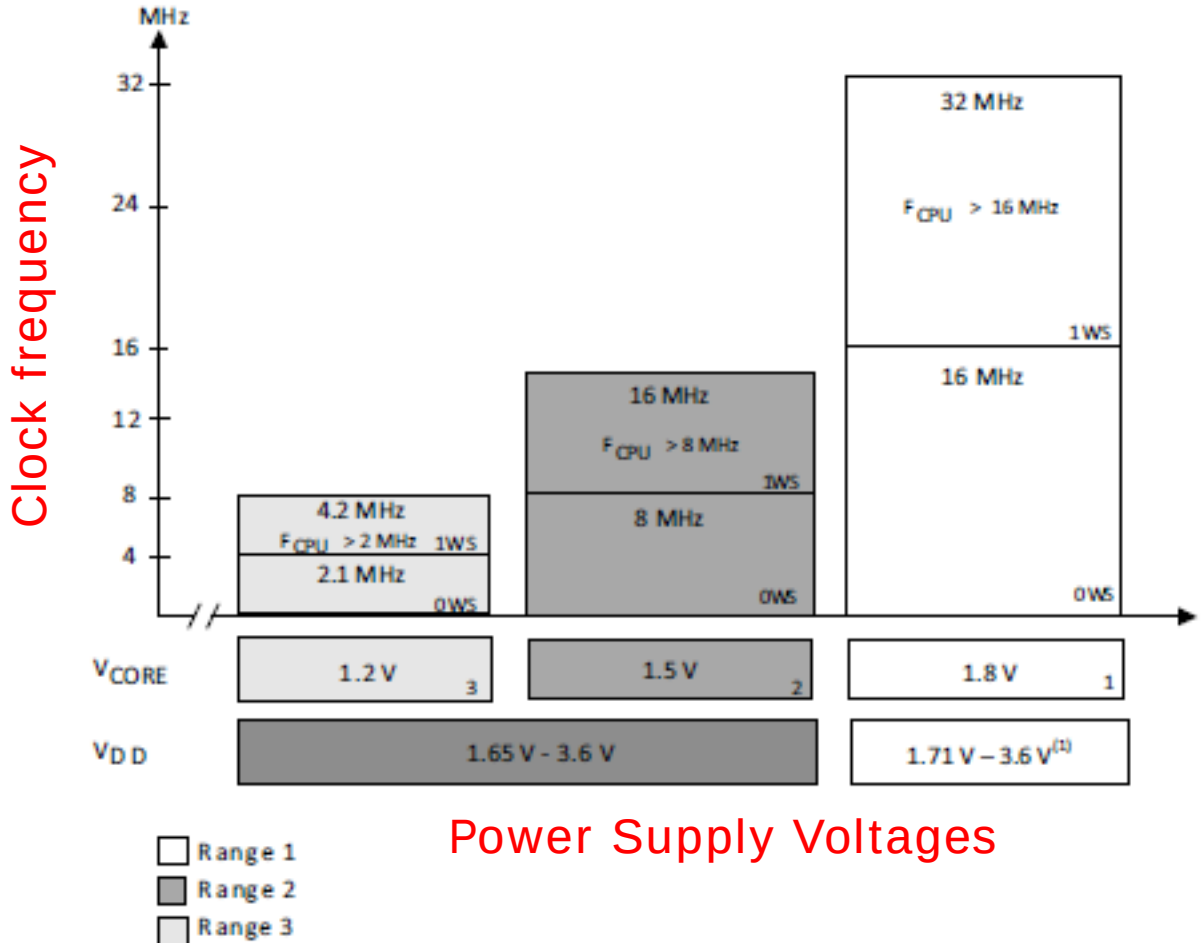


P_{sleep} = 6.6 mW $P_{deep\ sleep}$ = 89 μA $P_{deep\ power\text{-}down}$ = 667 nA

STM32L100RC Ultra-low-power MCU

- ARM Cortex-M3 CPU
- Designed for low-power applications
- Low-power modes:
  - 0.35µa standby mode (3 wakeup pins)
  - 1.3µa standby mode + real-time clock
  - 0.65µa stop mode (16 wakeup lines)
  - 1.5µa stop mode + real-time clock
  - 11µa low-power run mode
  - 238 mA/MHz run mode
  - 8 µs wakeup time

STM32L1xxx performance vs. V_{DD} and V_{CORE}



Source: STM32L100RC Reference Manual

STM32L1xx low-power modes

- Low power run mode: regulator in low power mode, limited clock frequency, limited #peripherals running
- Sleep mode: Cortex-M3 core stopped, peripherals still running
- Low power sleep mode: Cortex-M3 core stopped, limited clock frequency, limited # peripherals running, regulator in low power mode, RAM in power down, flash stopped
- Stop mode: all clocks stopped, regulator in low power mode
- Standby mode: VCORE domain powered off

Source: STM32L100RC Reference Manual

STM32L1xx low power modes

Mode name	Entry	Wakeup	Effect on V_{CORE} domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Low power run	LPSDSR and LPRUN bits + Clock setting	The regulator is forced in Main regulator (1.8 V)	None	None	In low power mode
Sleep (Sleep now or Sleep-on-exit)	WFI	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Low power sleep (Sleep now or Sleep-on-exit)	LPSDSR bits + WFI	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources, Flash CLK OFF	None	In low power mode
	LPSDSR bits + WFE	Wakeup event			
Stop	PDDS, LPSDSR bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers, internal and external lines)	All V_{CORE} domain clocks OFF	HSI and HSE and MSI oscillators OFF	ON, in low power mode (depending on PWR_CR)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper event, RTC timestamp event, external reset in NRST pin, IWDG reset			OFF

Source: STM32L100RC Reference Manual

Conserve power in **run** mode

- Reduce system clock frequencies
- Turn off clocks to unused peripherals
- Configure voltage regulator for low-power mode

Sleep mode

- Enter by executing WFI (wait for interrupt) or WFE (wait for event) instruction
  - Can select immediate entry to sleep mode, or entry upon completion of the lowest-priority interrupt handler
  - *"Low power sleep mode"* if voltage regulator configured for low power mode
- All I/O pins retain their functionality
- Exit sleep mode by any NVIC interrupt request or event
- Wakeup time shortest of the low-power modes

Stop mode

- Based on Cortex-M3 *"deepsleep"* mode, combined with peripheral clock gating
- All core clocks stopped and oscillators disabled
- Flash memory enters low power mode
- Can also configure other power-saving options
- Enter via WFI/WFE, but with SLEEPDEEP bit and other options set in system control register
- Exit by triggering any EXTI
- Longer recovery time than sleep mode

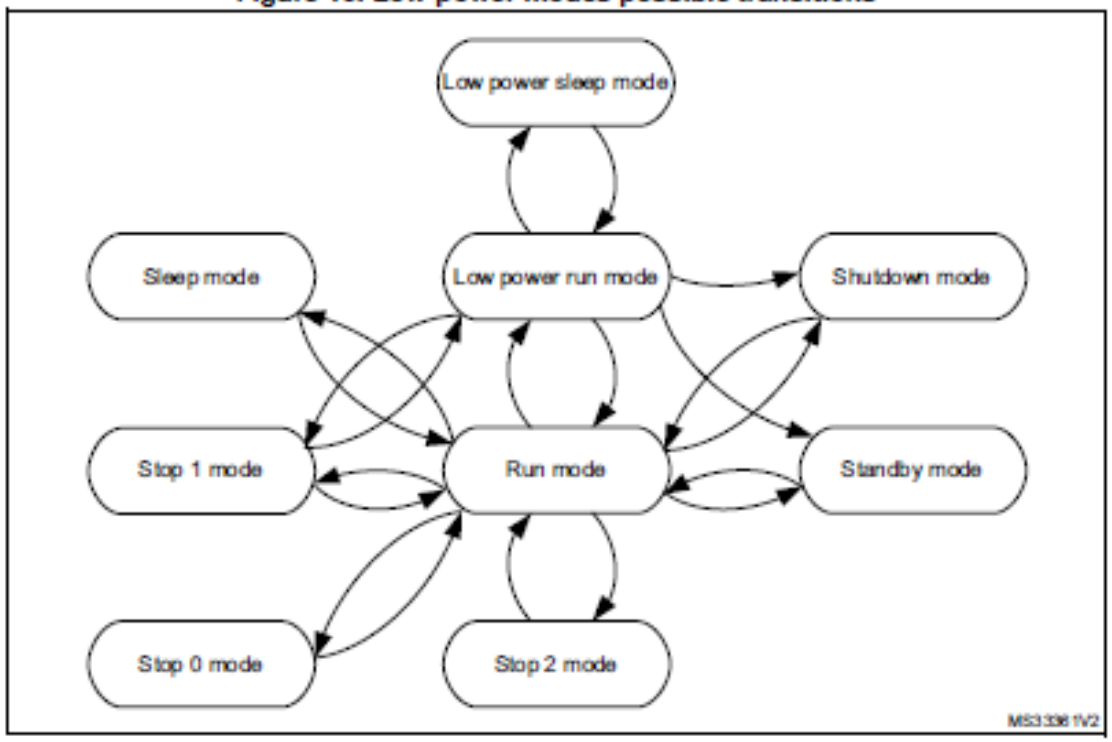
Standby mode

- Lowest power consumption
- Uses Cortex-M3 *"deepsleep"* mode with voltage regulator disabled
- Core domain powered off, oscillators switched off, SRAM and register contents are lost (except RTC)
- Enter via WFI, with configuration options set in system control register
- Exit via WKUP pin or RTC alarm
- Longer recovery time

STM32L4x6 Low-Power Modes

- Seven low-power modes

Figure 13. Low-power modes possible transitions



Defined on next slide

STM32L4x6 Low-Power Modes

- Sleep mode: CPU clock off, all peripherals can run and wake up the CPU when an interrupt or an event occurs.
- Low-power run mode: When system clock < 2 MHz. Regulator is in low-power mode to minimize the regulator's operating current.
- Low-power sleep mode: Entered from the Low-power run mode: Cortex®-M4 is off.
- Stop 0, Stop 1, Stop 2 modes: SRAMs and register contents retained. All Vcore clocks stopped and clock generators disabled (except LSI and LSE). Subsets of peripherals and RTC can run.
- Standby mode: Vcore domain powered off. SRAM2 can be preserved. RTC can remain active.
- Shutdown mode: Vcore domain powered off. All clocks stopped. LSE can run.

Cortex-M0: Sleep Mode Related Instructions

Syntax	Description
WFI	Wait for interrupt. Stops program execution until an interrupt arrives or until the processor enters a debug state
WFE	Wait for event. Stops the program execution until an event arrives (internal event register is set) or until the processor enters a debug state
SEV	Send event to all processors in multiprocessing environment (including itself).

Low-Power Requirements

The low-power requirements of systems vary but can be summarized as follows:

- Low operating power
- Low standby power
- High energy efficiency
- Wakeup latency

Cortex-M0 Low Power Features

Two architectural sleep modes:

- Normal sleep and deep sleep

Two instructions for entering sleep modes:

- WFE and WFI

Sleep-on-exit feature

- Allow processor to stay in sleep mode as much as possible.

Wakeup interrupt controller (WIC)

- An optional feature allows the clock of the processor to be completely removed during deep sleep.

Low-power design implementation

- Since the gate count is very low, the static leakage power is tiny compared to other 32-bit processors.

Cortex-M0 Sleep Mode

The Cortex-M0 processor supports two sleep modes:

- Normal sleep: switch off some of the clock signals.
- Deep sleep: reduce voltage supplies to the memory blocks and switch off additional components.

To enter a sleep mode:

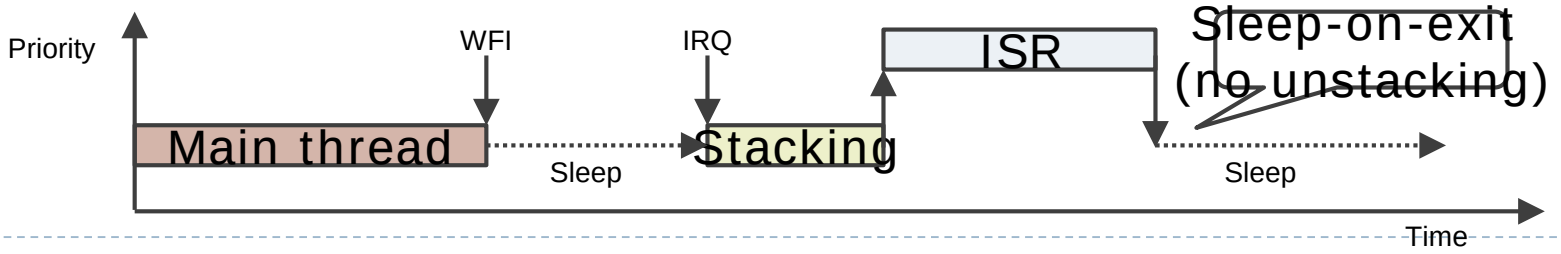
- Use WFE instruction.
- Use WFI instruction.
- Use the sleep-on-exit feature.

Sleep-on-Exit Feature

The sleep-on-exit feature allows the processor to enter a sleep mode as soon as all the exception handlers are completed.

The power consumption is reduced since:

- The execution of unnecessary programs in the main thread is avoided.
- The unnecessary stacking and unstacking operation is avoided.



How to Enable Sleep Features

The sleep features can be programmed by accessing the system control register (SCR) in the in the system control block (at address 0xE000ED10).

SCR

Bits	Field	Description
0	Reserved	-
1	SleepOnExit	Sleep-on-exit enable bit
2	SleepDeep	Sleep mode type bit, 0: normal sleep; 1: deep sleep
3	Reserved	-
4	SeVOnPend	Send event on pend bit, enable generation of event by a new interrupt pending status
[31:5]	Reserved	-

SCR can be accessed by register symbol "SCB->SCR," for example:

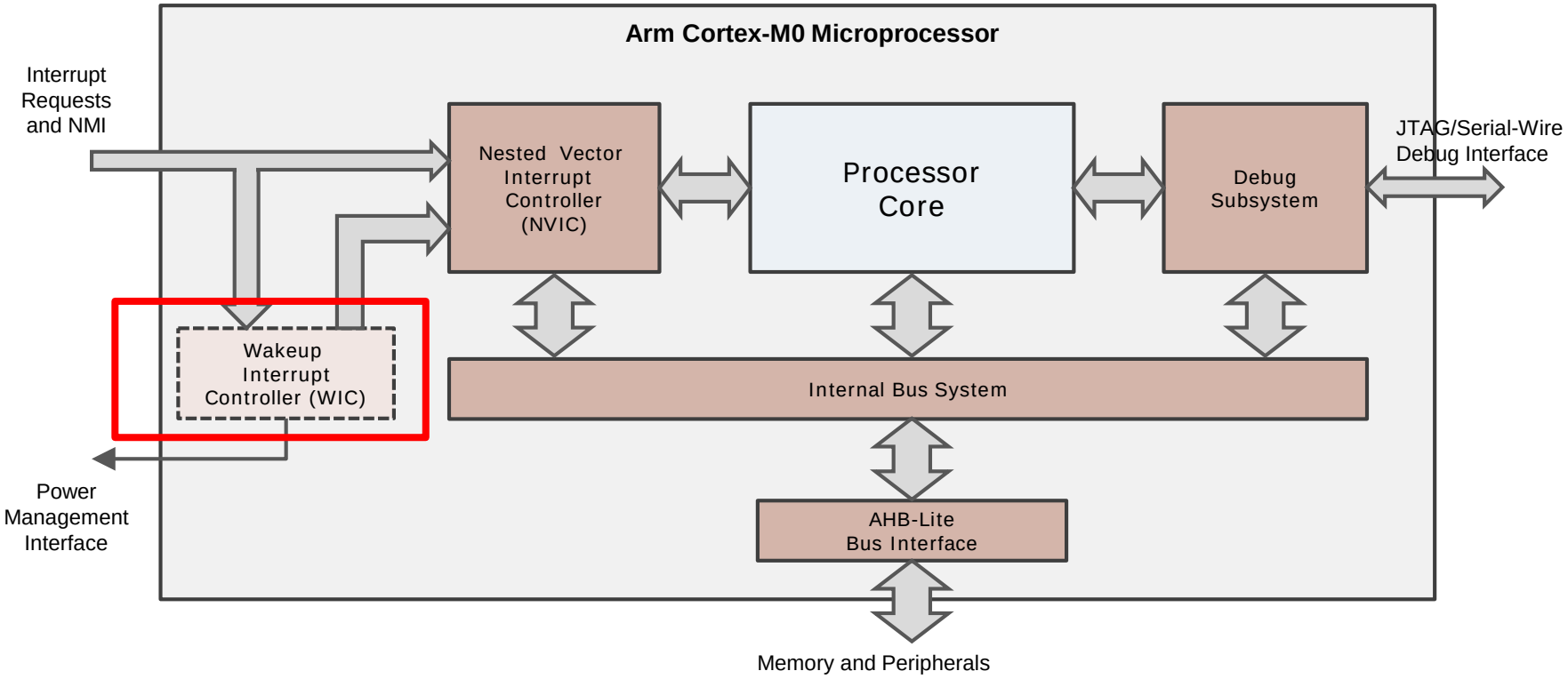
- SCB -> SCR = 1<1; //Enable sleep-on-exit bit

Processor Wakeup Conditions

The processor can exit the sleep mode on different conditions.

Type	Priority	SeVOnPend	PRIMASK	Wake up	Execute ISR
WFE	IRQ priority > current level	--	0	Yes	Yes
	IRQ priority > current level	0	I	No	No
	IRQ priority ≤ current level	0	--	No	No
	IRQ priority ≤ current level	I	--	Yes	No
WFI	IRQ priority > current level	--	0	Yes	Yes
	IRQ priority > current level	--	I	Yes	No
	IRQ priority ≤ current level	--	--	No	No

Wakeup Interrupt Controller

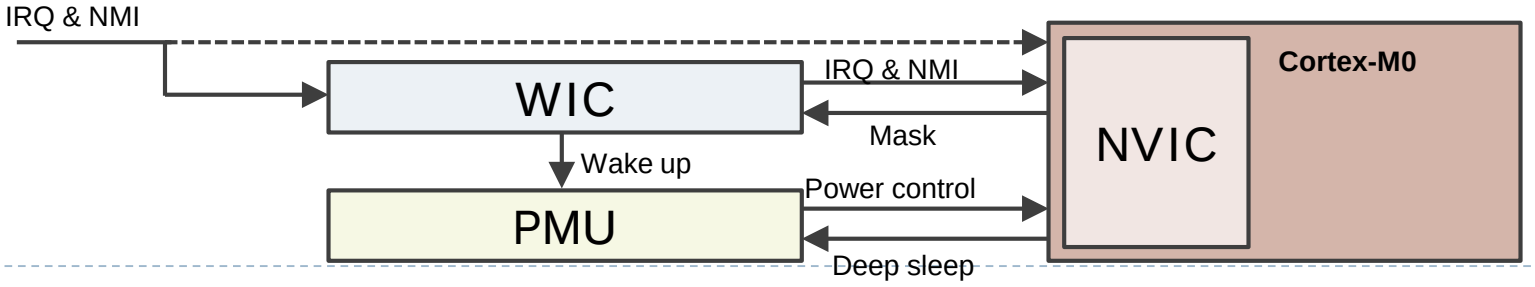


Wakeup Interrupt Controller

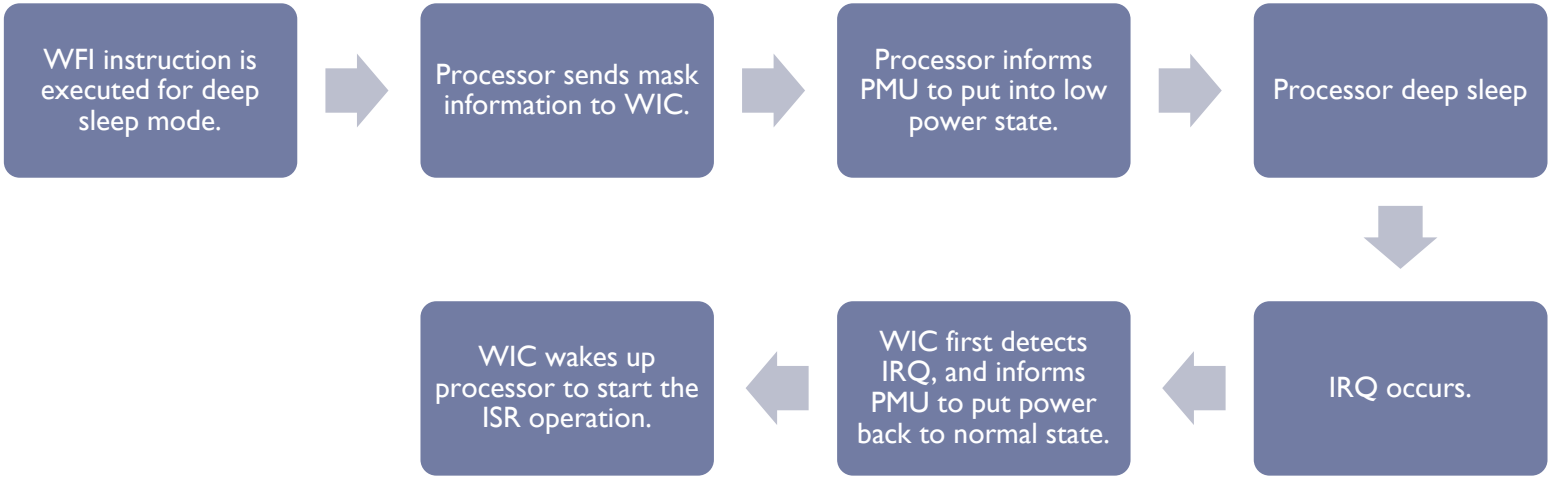
Usually requires a system-level power management unit (PMU)

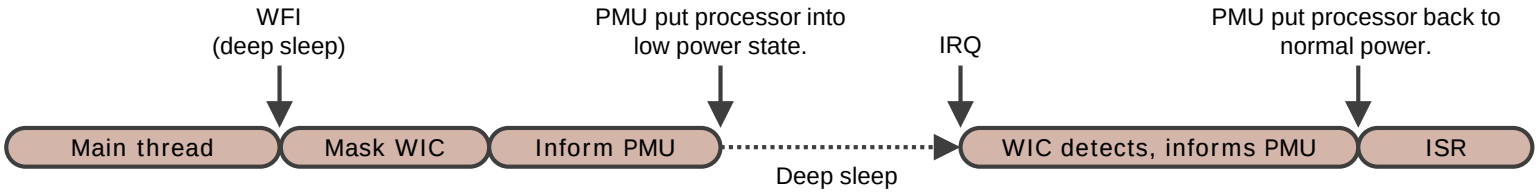
When an interrupt is detected, the WIC sends a request to a PMU in the microcontroller to restore power and clock signals to the processor, and then the processor can wake up and process the interrupt request.

The WIC does not require extra programmable registers.



Enter and Exit Deep Sleep Mode





Developing Low-Power Applications

Despite the low-power features of the processor, various things can reduce power consumption of an application:

- Run the processor at a suitable clock frequency.
- Disable a peripheral when not used.
- For interrupt-driven applications, the processor should stay in sleep mode as much as possible; e.g., use sleep-on-exit.
- Optimize the application code for speed to reduce active cycles (this may be at the cost of a larger code size).
- Turn off some of the clock signals or power supply when that piece of circuit is not used.