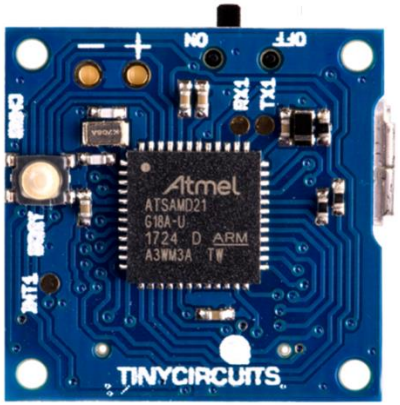
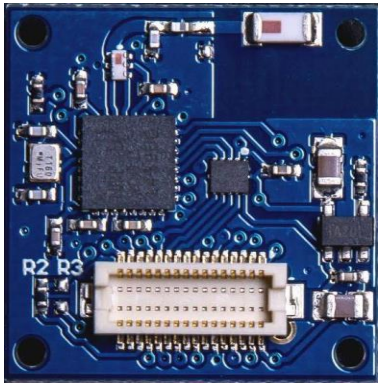
CSE190 Winter 2025



Lecture 23

Power Management



Wireless Embedded Systems

Aaron Schulman

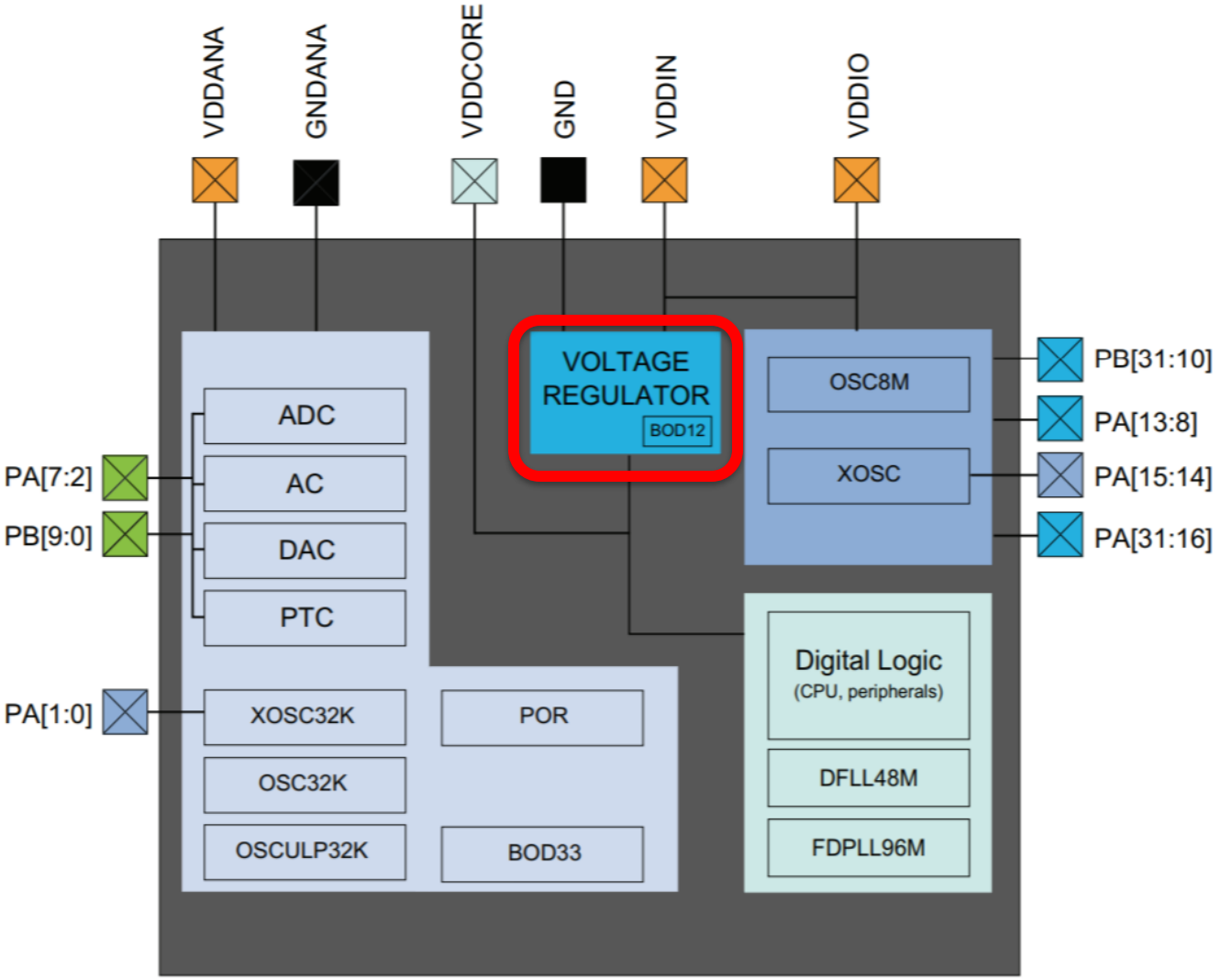
Dynamic power consumption

- Mostly the power dissipated changing states
  - Running instructions
  - Communicating over a bus
  - **Anything that needs a clock uses dynamic power**

- Related to clock frequency (f) and voltage (V)
- $P_{switching} = CV^2 f$
  - Run your device at as low voltage as possible
  - Run your CPU or peripheral at as low clock freq as possible

- Many microcontrollers run logic at low voltages to save energy

Microcontrollers regulate 1.6-3.6V input Voltage down to 1.2V



Voltage can be dynamically scaled up/down: Although there are tradeoffs

The main regulator have two possible programmable voltage range detailed below:

- Range 1: High-performance range.

The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 80 MHz. The Flash access time for read access is minimum, write and erase operations are possible.

- Range 2: Low-power range.

The main regulator provides a typical output voltage at 1.0 V. The system clock frequency can be up to 26 MHz.The Flash access time for a read access is increased as compared to Range 1; write and erase operations are possible.

Static power consumption

- Common device to worry about is SRAM
  - SRAM leaks power to stay in the same state
- The only way we can stop it is to turn it off
  - But then you lose data in SRAM
- If you want to go idle for a long time, move information into persistent storage.

How to save dynamic power
Clock gating

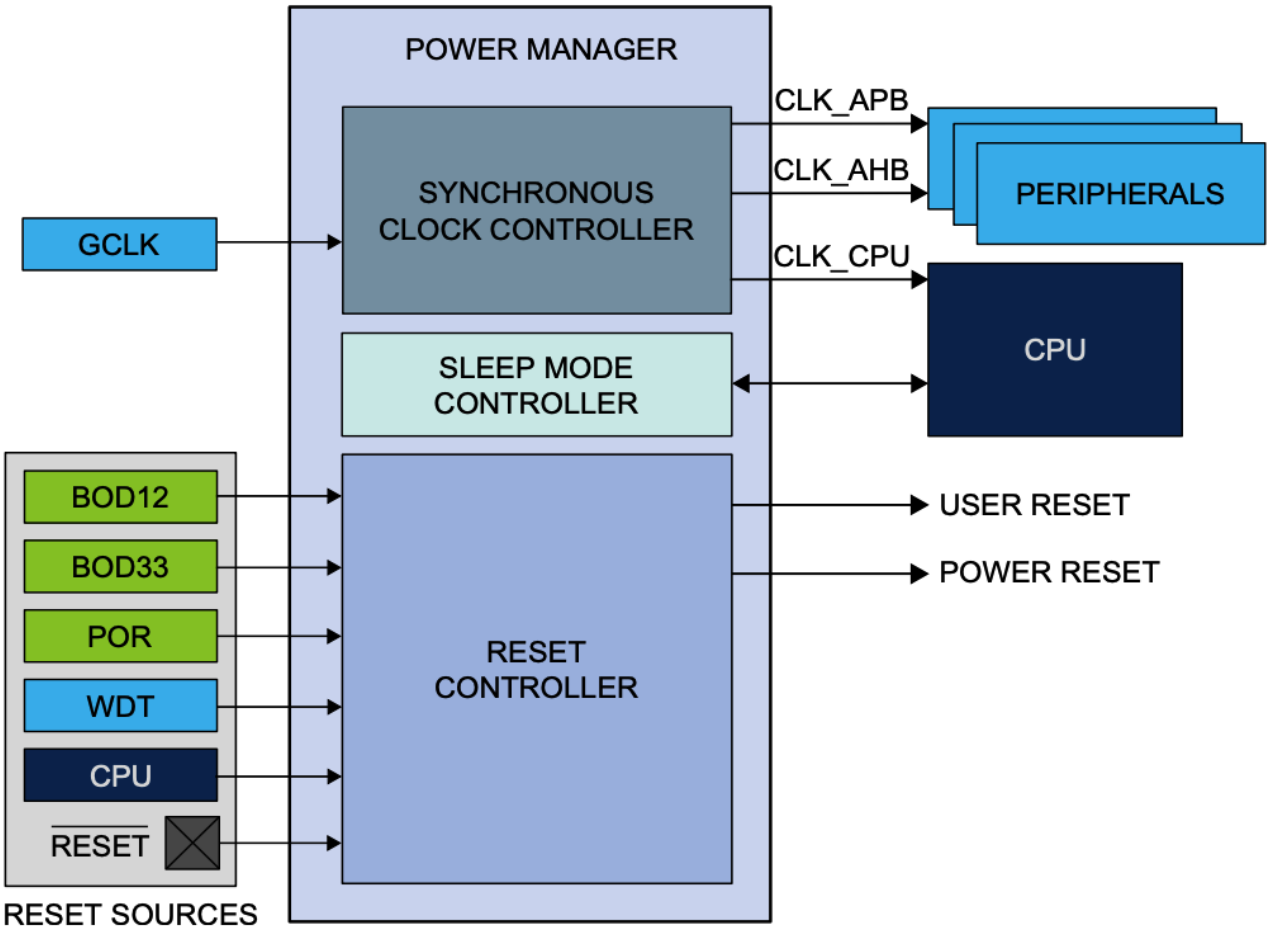
Clock gate: A circuit that disconnects a clock from a device

- Eliminates all switching activity in that device
  - No dynamic power consumption
  - Minimal recovery time (just ungate the clock)
  - Also effectively makes it useless (as you learned already)

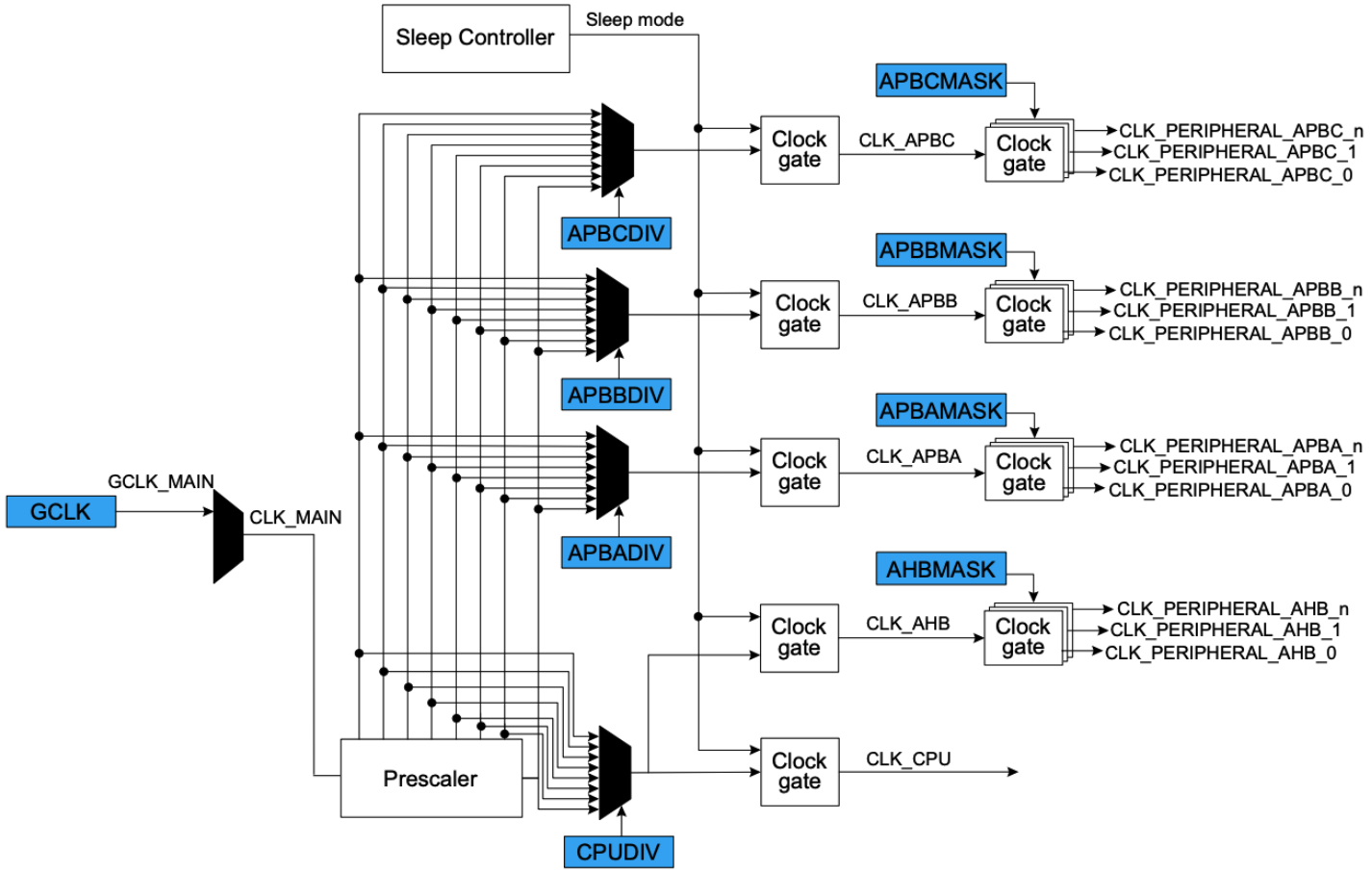
- However, keeps device powered (static power)
  - Retains configuration in registers

Power management peripheral:
Controlling static and dynamic power from software

Figure 15-1. PM Block Diagram



Peripherals: PM lets you disable and enable clocks for peripherals



When should you disable and enable a clock for peripherals?

RCC Peripheral provides clock gating

6.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low Power Sleep modes stops the CPU clock. The memory interface clocks (Flash and SRAM1 and SRAM2 interfaces) can be stopped by software during sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop modes (Stop 0, Stop 1 and Stop 2) stops all the clocks in the V_{CORE} domain and disables the three PLL, the HSI16, the MSI and the HSE oscillators.

  All U(S)ARTs, LPUARTs and I^2Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as the clock source for that peripheral).

  All U(S)ARTs and LPUARTs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON). In that case the LSE remains always ON in Stop mode (they do not have the capability to turn on the LSE oscillator).
- Standby and Shutdown modes stops all the clocks in the V_{CORE} domain and disables the PLL, the HSI16, the MSI and the HSE oscillators.

The CPU's deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register.

When leaving the Stop modes (Stop 0, Stop 1 or Stop 2), the system clock is either MSI or HSI16, depending on the software configuration of the STOPWUCK bit in the RCC_CFGR register. The frequency (range and user trim) of the MSI oscillator is the one configured before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after wakeup even if the LSE was kept ON during the Stop mode.

When leaving the Standby and Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Standby mode is configured with the MSISRANGE is the RCC_CSR register, from 1 to 8 MHz. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

CPU: PM lets you control CPU “sleep states”

Two primary sleep modes:

- **Idle:** Memory stable, CPU stopped and can start in one clock cycle

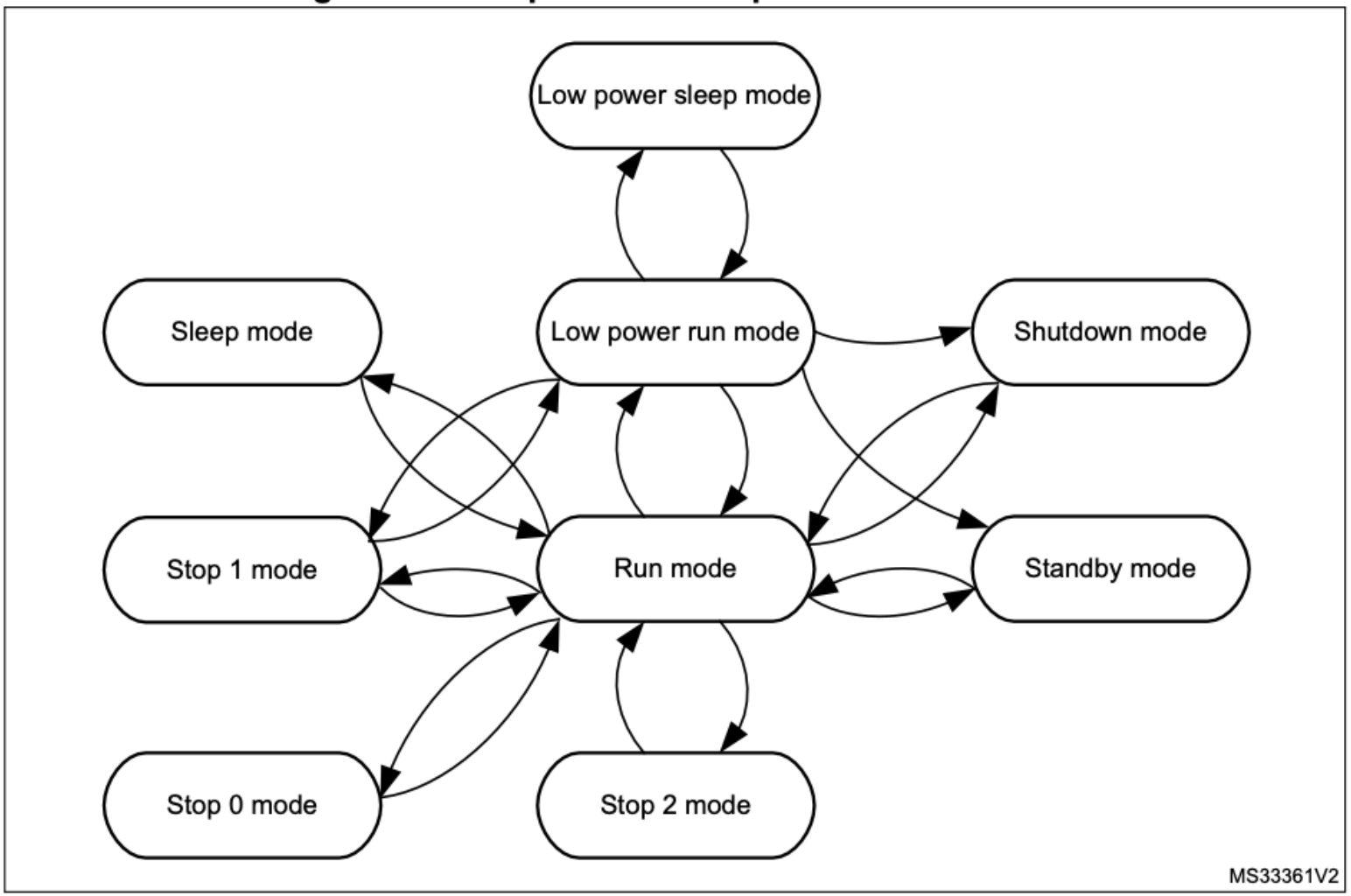
- **Standby:** Memory stable, CPU stopped and takes many clock cycles to start

Table 22. Low-power mode summary

Mode name	Entry	Wakeup source(1)	Wakeup system clock	Effect on clocks	Voltage regulators	
					MR	LPR
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF no effect on other clocks or analog clock sources	ON	ON
	WFE	Wakeup event				
Low-power run	Set LPR bit	Clear LPR bit	Same as Low-power run clock	None	OFF	ON
Low-power sleep	Set LPR bit + WFI or Return from ISR	Any interrupt	Same as before entering Low-power sleep mode	CPU clock OFF no effect on other clocks or analog clock sources	OFF	ON
	Set LPR bit + WFE	Wakeup event			OFF	ON

Transitions between CPU modes

Figure 13. Low-power modes possible transitions



When do you wake up a CPU from sleep?

- When an event occurs that you need to handle
  - User input
  - External device input
  - Alarm fires

- **We wake up the CPU with Interrupts**

How does your code let the CPU know it’s ok to let it go to sleep?

- Special CPU instruction called ”Wait for interrupt” or WFI
  - `__WFI()`
- Normally executed at the end of the main() loop
  - You are done handling all of the events that happened
- Need to set what CPU suspend mode you want before WFI

Stop 0 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M4 System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “000” in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex®-M4 System Control register – SLEEPONEXIT = 1 – No interrupt is pending

How much CPU power can we save with sleep and idle states?

I_{DD_ALL} (Run)	Supply current in Run mode	$f_{HCLK} = f_{HSE}$ up to 48MHz included, bypass mode PLL ON above 48 MHz all peripherals disable	Range 2	26 MHz	2.88	2.94	3.05	3.23	3.58	3.18	3.26	3.40	4.02	4.65	mA
				16 MHz	1.83	1.87	1.98	2.15	2.50	2.01	2.16	2.30	2.72	3.34	
				8 MHz	0.97	1.00	1.11	1.27	1.62	1.07	1.16	1.32	1.73	2.36	
				4 MHz	0.54	0.57	0.67	0.84	1.18	0.59	0.69	0.88	1.23	1.96	
				2 MHz	0.33	0.36	0.46	0.62	0.96	0.37	0.45	0.63	0.98	1.70	
				1 MHz	0.22	0.25	0.35	0.51	0.85	0.25	0.33	0.50	0.86	1.57	
				100 kHz	0.12	0.15	0.25	0.41	0.75	0.15	0.21	0.39	0.74	1.45	
				80 MHz	10.2	10.3	10.5	10.7	11.1	11.22	11.57	11.86	12.07	13.11	
				72 MHz	9.25	9.31	9.46	9.68	10.1	10.18	10.41	10.55	10.76	11.80	
				64 MHz	8.25	8.31	8.46	8.67	9.08	9.08	9.37	9.66	9.87	10.91	

I_{DD_ALL} (Standby)	Supply current in Standby mode (backup registers retained), RTC disabled	no independent watchdog	1.8 V	114	355	1540	4146	10735	176	888	3850	10365	26838	nA
			2.4 V	138	407	1795	4828	12451	223	1018	4488	12070	31128	
			3 V	150	486	2074	5589	14291	263	1215	5185	13973	35728	
			3.6 V	198	618	2608	6928	17499	383	1545	6520	17320 (2)	43748	
		with independent watchdog	1.8 V	317	-	-	-	-	-	-	-	-	-	
			2.4 V	391	-	-	-	-	-	-	-	-	-	
			3 V	438	-	-	-	-	-	-	-	-	-	
			3.6 V	566	-	-	-	-	-	-	-	-	-	