

PROIECTAREA CU MICROPROCESOARE

Cursul 10 Power Management

Facultatea de Automatică și Calculatoare
Politehnica București



Intro

- Importanța power management crește constant
 - Avem miliarde de dispozitive complexe alimentate cu baterii
 - Cerințe de conectivitate
 - Optimizarea puterii se face adesea cu întârziere în design
 - Trebuie luată în considerare de la început
-

Putere și Energie

- Putere: rata de energie transferată în timp
 - Unitatea folosită de obicei este „Watt”
 - $1 \text{ Watt} = \frac{1 \text{ Joule}}{1 \text{ Second}}$
 - Energie: putere disipată în timp
 - Unitatea folosită de obicei este „Joule”
 - $1 \text{ Joule} = 1 \text{ Watt} \times 1 \text{ Second}$
 - Dispozitivele sunt de obicei evaluate în wați (miliwați)
 - Bateriile sunt de obicei evaluate în Jouli (miliwați oră)
-

Consumul de energie

- În tehnologia CMOS: $P \sim V_{DD}^2 \cdot f$
 - Tranzițiile costă
 - Ce tranziții avem când în cod avem `while(1);` ?
 - Încă se execută instrucțiuni, PC se modifică etc.
 - Toate timerele active numără
 - ADC-ul poate să fie activ
 - Seriala poate transmite/recepționa
 - Circuitele de input GPIO sunt active
 - Power management - procedeul prin care restricționăm ce poate face procesorul
-

Putere dinamică și putere statică

- $P_{total} = P_{dynamic} + P_{static}$
 - Disiparea dinamică a puterii
 - Capacitatea parazită a sarcinii ce comută (majoritatea puterii)
 - Curent de scurtcircuit (restul de putere)
 - Disiparea puterii statice
 - Scurgeri din cauza imperfecțiunilor designului tranzistorului
 - Ne pasă de puterea statică pentru intervale lungi de funcționare
-

Putere statică

- De obicei dispozitivul pentru care să vă faceți griji este SRAM
 - SRAM pierde putere pentru a rămâne în aceeași stare
 - Singurul mod prin care putem limita consumul este să îl oprim
 - Dar apoi pierzi datele din SRAM
 - Dacă dispozitivul rămâne inactiv pentru o perioadă lungă de timp, mutați informațiile în stocarea persistentă (de ex. Flash).
-

Putere dinamică

- În mare parte, putere disipată în schimbarea stărilor
 - Rularea instrucțiunilor
 - Comunicarea pe magistrale
 - **Orice circuit care are nevoie de un semnal de ceas folosește putere dinamică**
 - Legat de frecvența (f) și tensiunea ceasului (V)
 - $P_{switching} = CV^2f$
 - Porniți dispozitivul la o tensiune cât mai scăzută
 - Rulați procesorul sau perifericele la o frecvență de ceas cât mai mică posibil
 - Multe microcontrolere funcționează la tensiuni joase
-

Cum să reduceți puterea dinamică?

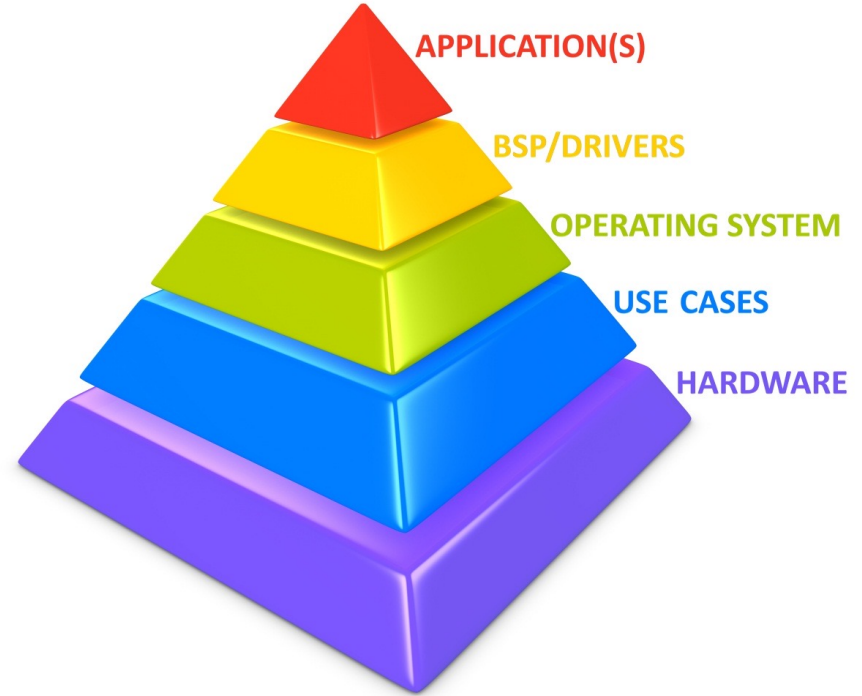
- **Clock gating:** Un circuit care deconectează semnalul de ceas al unui dispozitiv
 - Elimină orice activitate de comutare din acel dispozitiv
 - Fără consum dinamic de energie
 - Timp minim de recuperare (doar reconectați ceasul)
 - Cu toate acestea, menține dispozitivul alimentat (putere statică)
 - Păstrează configurația în registre
-

Software power management - câteva idei generale

- Alegeți hardware cu capacitățile potrivite
 - Permiteți software-ului să gestioneze puterea
 - Alegeți sistemul de operare și drivererele
 - Definiți profiluri de utilizare a energiei
 - Alegeți obiective măsurabile - țintă de consum
 - Utilizați aceste ținte pe tot parcursul procesului de dezvoltare
-

Alte idei

- Alegeți hardware adecvat
- Luați în considerare utilizarea
- Selectați sistemul de operare
- Rezolvați problemele cu driverul/BSP
- Codul aplicației are cea mai mică influență asupra puterii consumate



Alegerea hardware-ului potrivit

- Cea mai mare influență asupra consumului de energie
 - Definește cel mai bun caz de economisire a energiei
 - Caracteristici CPU
 - Opriți module, de ex. periferice
 - Scalare dinamică a tensiunii și a frecvenței (DVFS)
 - Definește punctele de operare
 - Moduri de putere redusă (low power)
 - Trebuie să vă uitați la un design mai larg pentru a asigura compatibilitatea cu cele de mai sus
-

Cazuri de utilizare

- Funcția pe care o îndeplinește un dispozitiv
 - Cu sau fără interacțiunea utilizatorului
 - Exemplu ipotetic:
 - Aparat medical
 - Pe baterii
 - Ecran LCD
 - Monitorizează semnele vitale
 - Încarcă date prin Wi-Fi
-

Exemplu



1. Dispozitivul efectuează o măsurătoare completă
 2. Dispozitivul încarcă un set de date măsurate
 3. Utilizatorul își verifică propriile valori vitale folosind un afișaj încorporat
 4. Dispozitivul este inactiv în așteptarea următoarei măsurători
-

Exemplu

- Câtă funcționalitate este necesară pentru fiecare caz de utilizare?
 - Prin urmare, ce drivere [blocuri hardware] trebuie să fie activate pentru fiecare caz de utilizare?
 - Energia estimată pentru fiecare caz de utilizare:
 - Consumul estimat de energie
 - Timp de funcționare estimat în cazul utilizării
-

Exemplu

Utilizare caz	Curent mediu (mA)	Durata (s)	Frecvența pe zi	Timp total (s/zi)	ENERGIE UTILIZATĂ (mAh/zi)
Măsurare parametri vitali	158	1	288	288	13
Încărcarea datelor	250	3	288	864	60
Verificare parametri vitali (user check)	320	30	15	450	40
Inactiv (Hibernare)	1			84798	24
TOTAL					136

Până și un model simplu de consum vă ajută să determinați din timp dacă durata de viață estimată a bateriei este sau nu fezabilă pentru produsul vostru!

Exemplu

- Încărcarea datelor e responsabilă de cea mai mare utilizare a energiei
 - Poate că măsurarea și încărcarea la fiecare 5 minute este prea costisitoare
 - Poate măsurați la fiecare 5 minute, dar încărcați la fiecare 30 de minute
 - Încărcați și dacă există o schimbare majoră a elementelor vitale
 - User Vitals Check este al doilea ca mărime
 - Presupune un timeout de afișare de 30 de secunde
 - Poate fi mai scurt?
 - Majoritatea celorlalte module hardware sunt oprite în acest caz de utilizare
-

Sistemul de operare

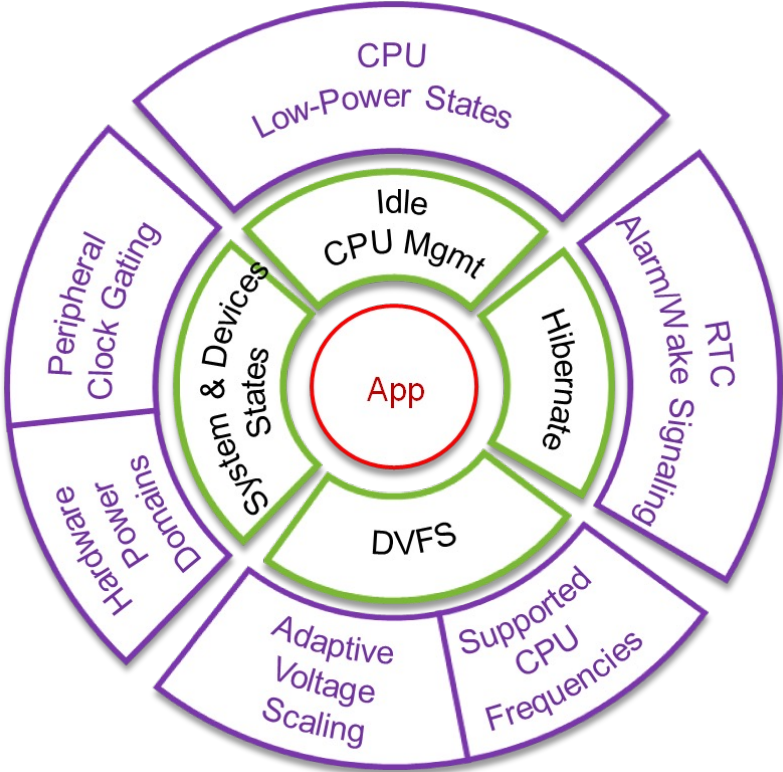
- Impact semnificativ asupra economisirii energiei
 - Trebuie să accepte funcții de low power
 - DVFS
 - Moduri inactiv/sleep
 - Power framework nativ este cel mai eficient
 - BSP trebuie scris pentru a rezolva problemele de alimentare
 - Fiecare driver are stări de putere bine definite
-

Power Framework

■ = Hardware power management

■ = Software de aplicație

■ = RTOS Power Mgmt Framework



BSP & Drivers

- Definiți cerințele de putere pentru fiecare driver.
Specifică:
 - Care stări de putere sunt implementate
 - ON, STANDBY, SLEEP, OFF
 - În ce moduri de operare va fi folosit driverul. De exemplu:
 - Când este pornit, trebuie să funcționeze la 200MHz și 100MHz
 - SLEEP poate avea ca rezultat un ceas de 1 MHz
 - Participarea la DVFS
 - Notificare de transfer DMA
-

Hibernate, Suspend, Standby

- Unele hardware facilitează moduri de consum foarte scăzut de energie
 - **Suspendare** : tot hardware-ul este oprit, cu excepția memoriei RAM, al cărei conținut este protejat
 - **Hibernare** : conținutul RAM este stocat în memorie nevolatilă și totul este oprit
-

Hibernate, Suspend, Standby

- Costul de intrare/ieșire din aceste moduri
 - Putere
 - Timp – latența dispozitivului
 - Depinde de cât de mult din sistem este PORNIT când se intră în modulul de funcționare
 - Trebuie să salvați starea și să reinițializați
 - Hibernare are, de asemenea, costuri de stocare a memoriei RAM, care variază în funcție de cantitatea de memorie RAM utilizată
 - De asemenea, poate reduce durata de viață a sistemului
-

Power Management la nivelul aplicației

- Cel mai de sus nivel software
 - Codul scris prost poate afecta negativ consumul de energie
 - Un sistem de operare cu funcții de power management încorporate simplifică lucrurile
 - Dezvoltatorul aplicației este mai puțin preocupat de detalii
-

Power Management la nivelul aplicației

- Aplicația constă dintr-un număr de task-uri independente
 - Fiecare task (sau grup de task-uri) își înregistrează nevoile de putere:
 - Ce periferice sunt folosite
 - Mod minim de operare
 - Sistemul de operare se ocupă de gestionarea energiei împreună cu comutarea contextului
-

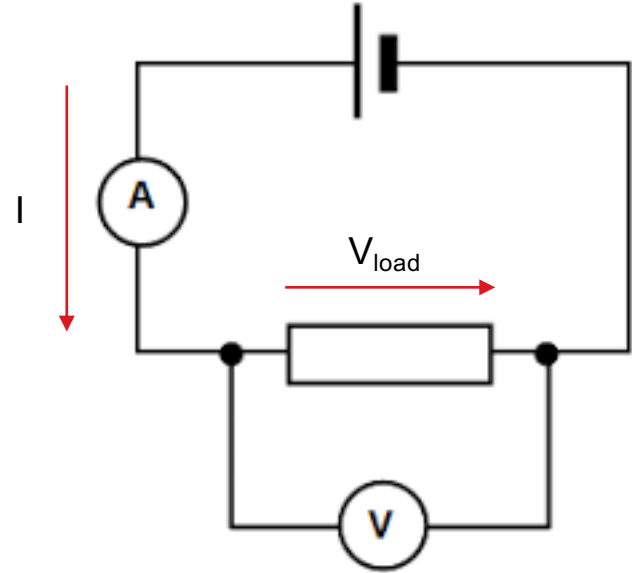
Măsurare și testare

- Puterea ar trebui măsurată de la prima linie de cod
 - Posibil prin planificare:
 - Setarea cerințelor de putere pentru drivere
 - Definirea cazurilor de utilizare
 - Maparea cazurilor de utilizare la aplicații
 - **Toți inginerii de software ar trebui să poată măsura consumul de energie!**
-

Cum măsurăm consumul?

Reminder:

- Ampermetrul măsoară curentul în serie cu sarcina
- Voltmetrul măsoară tensiunea, în paralel cu sarcina
- $P = I * V_{load}$
- Pentru sarcini alimentate din surse stabilizate (cum este cazul unui microcontroler), puterea variază doar prin variația curentului consumat
 - De obicei, curent mai mare = workload mai mare

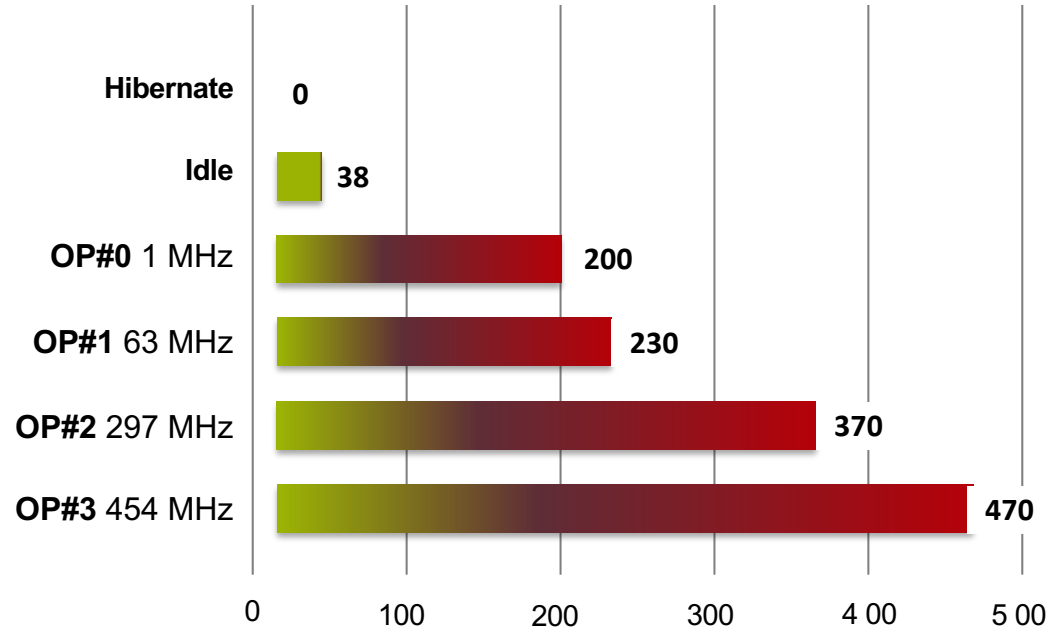


Măsurare și testare

- Îndeplinirea cerințelor de consum de putere ar trebui privită ca parte integrală a funcționalității codului
 - De exemplu:
 - Un driver Wi-Fi poate funcționa bine cu toate rețelele wireless
 - Trebuie să poată fi oprit și să reducă puterea la [aproape] zero
 - Trebuie să pornească din nou și să fie complet funcțional
 - Funcționalitatea trebuie să fie repetabilă [să zicem, de 100.000 de ori]
-

Măsurare și testare: exemplu

Tensiunea de
operare
(1,5 V)



Consumul de curent microcontroler (mA)

Impactul asupra duratei de viață a bateriei...

	mAh	Procent Utilizare pe oră	mAh	Baterie (ore)
OP #3	470	10%	47	
OP #2	370	5%	19	
OP #1	230	10%	23	
OP #0	200	15%	30	
Așteptare	38	20%	8	
Hibernează	0	40%	0	
Total			126	19

Cu energy management pentru nucleul procesorului

Presupunem că în ambele cazuri alimentăm dintr-o baterie de 2400mAh

	mAh	Baterie (ore)
OP #3	470	
Total		5

Fără management al energiei

ATMega328P Power Management

- Mai multe stări de power management
 - Mai puține periferice
 - Mai puține ceasuri active
 - Mai puține metode de revenire
- Periferice dezactivabile manual
 - PRR - Power Reduction Register
 - Un bit pt fiecare periferic: ADC, USART0, SPI, Timer/Counter1, Timer/Counter0, Timer/Counter2, TWI(I^2C)
 - One reserved bit (bit 4)

Stări de lucru

Table 9-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators		Wake-up Sources							
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O	Software BOD Disable
Idle			X	X	X	X	X ⁽²⁾	X	X	X	X	X	X	X	
ADC noise Reduction				X	X	X	X ⁽²⁾	X ⁽³⁾	X	X ⁽²⁾	X	X	X		
Power-down								X ⁽³⁾	X				X		X
Power-save					X		X ⁽²⁾	X ⁽³⁾	X	X			X		X
Standby ⁽¹⁾						X		X ⁽³⁾	X				X		X
Extended Standby					X ⁽²⁾	X	X ⁽²⁾	X ⁽³⁾	X	X			X		X

- Notes:
1. Only recommended with external crystal or resonator selected as clock source.
 2. If Timer/Counter2 is running in asynchronous mode.
 3. For INT1 and INT0, only level interrupt.

Stări de lucru - consum

28.3 DC Characteristics

$T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$, $V_{CC} = 2.7\text{V}$ to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.	Typ. ⁽²⁾	Max.	Units
Power supply current ⁽¹⁾	Active 4MHz, $V_{CC} = 3\text{V}$	I_{CC}		1.5	2.4	mA
	Active 8MHz, $V_{CC} = 5\text{V}$			5.2	10	mA
	Active 16MHz, $V_{CC} = 5\text{V}$			9.2	14	mA
	Idle 4MHz, $V_{CC} = 3\text{V}$			0.25	0.6	mA
	Idle 8MHz, $V_{CC} = 5\text{V}$			1.0	1.6	mA
	Idle 16MHz, $V_{CC} = 5\text{V}$			1.9	2.8	mA
Power-down mode ⁽³⁾	WDT enabled, $V_{CC} = 3\text{V}$				44	μA
	WDT enabled, $V_{CC} = 5\text{V}$				66	μA
	WDT disabled, $V_{CC} = 3\text{V}$				40	μA
	WDT disabled, $V_{CC} = 5\text{V}$				60	μA

- Notes:
1. Values with [Section 9.10 “Minimizing Power Consumption”](#) on page 36 enabled (0xFF).
 2. Typical values at 25°C .
 3. The current consumption values include input leakage current.

Registre

- SMCR - registru control sleep
 - biți SM - modul în care să intre
 - SE - sleep enable (se poate intra in sleep)
- *sleep* – instrucțiunea care intră în sleep efectiv
- Cu ajutor / Fără ajutor

```
void main ()
{
    while(1)
    {
        do_stuff();
        // ...

        set_sleep_mode(SLEEP_MODE_IDLE);
        sleep_mode();
    }
}
```

```
void main ()
{
    while(1)
    {
        do_stuff();
        // ...

        SMCR = (1 << SE); // pentru bitii SM2..0 — 0 este IDLE
        __asm__ __volatile__ ( "sleep" "\n\t" :: );
        SMCR = 0;
    }
}
```

Exemplu

- Un termometru cu afișaj digital
 - O dată la 1s măsoară și schimbă afișajul
 - Afișaj conectat prin SPI foarte eficient
- Cum minimizăm consumul microcontroller-ului?

Datele problemei

- Presupunem un consum mediu de $100\mu\text{A}$ pentru senzor și display
- Avem la dispoziție o baterie de ceas CR2032 de 200mAh și 3.3V
- Presupunem că putem folosi toată capacitatea
- Avem un cristal de cuarț, de 8MHz
- ADC configurat inițial cu prescaler maxim 128

$$t_{\text{conversie}} = 13 \text{ cicliADC} \cdot 128 \cdot t_{\text{cicluCPU}}$$

$$t_{\text{conversie}} = 208\mu\text{s} \approx 200\mu\text{s}$$

- Consumul pe SPI presupune schimbul a 100 bytes la 4MHz .

$$t_{\text{comunicare}} = \frac{100 * 8}{4000000} \text{s}$$

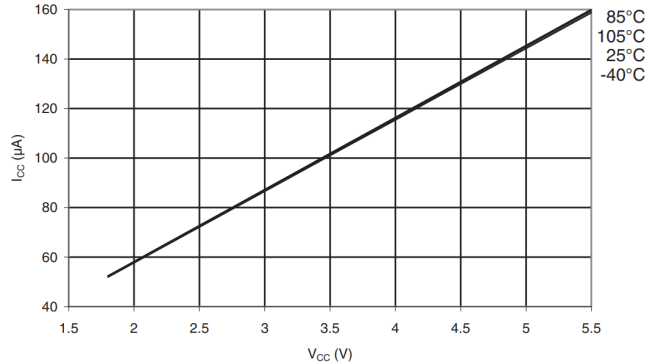
$$t_{\text{comunicare}} = 200\mu\text{s}$$

- Neglijăm restul operațiilor

Calcul consum activ

- $I_{CC_{total}} = I_{CC_{Activ,8MHz,3.3V}} (1 + I_{ADC} + I_{SPI} + I_{TIM1})$
- $I_{CC_{total}} = 3mA \cdot (1 + 0.048 + 0.028 + 0.026)$
- $I_{CC_{total}} = 3mA \cdot (1 + 0.048 + 0.028 + 0.026) = 3.306mA$

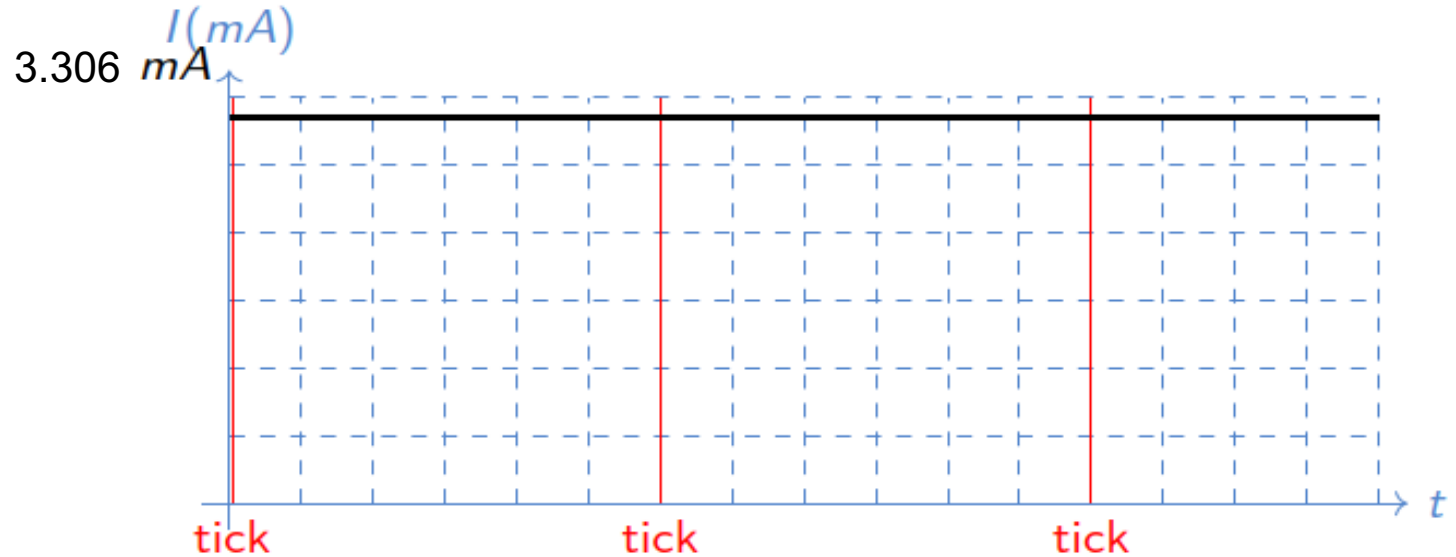
Figure 35-40. ATmega328P: ADC Current vs. V_{CC} (AREF = AV_{CC})



PRR bit	Additional Current consumption compared to Active with external clock (See Figure 30-1 and Figure 30-2)
PRTIM2	3.8%
PRTIM1	2.6%
PRTIM0	1.6%
PRADC	4.8%
PRSPI	2.8%

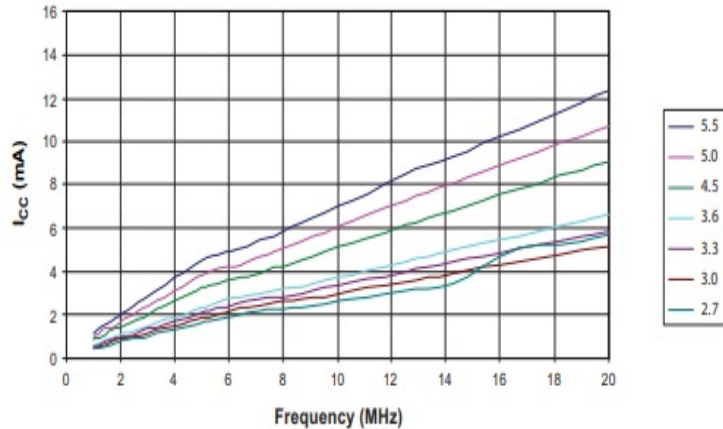
Grafic de consum

- $$t_{\text{baterie}} = \frac{\text{Capacitate}}{\text{Consum}} = \frac{200}{3.306 + 0.1} = 58.8\text{h}$$



Calcul consum activ optimizat

- *Dacă folosim PRR să închidem ADC și SPI când nu avem nevoie?*
- $I_{CC_{adc}} = I_{CC_{Activ,8MHz,3.3V}} (1 + I_{ADC} + I_{TIM1}) = 3.174mA$
- $I_{CC_{comunicatie}} = I_{CC_{Activ,8MHz,3.3V}} (1 + I_{SPI} + I_{TIM1}) = 3.129mA$
- $I_{CC_{rest}} = I_{CC_{Activ,8MHz,3.3V}} (1 + I_{TIM1}) = 3.048mA$



PRR bit	Additional Current consumption compared to Active with external clock (See Figure 30-1 and Figure 30-2)
PRTIM2	3.8%
PRTIM1	2.6%
PRTIM0	1.6%
PRADC	4.8%
PRSPI	2.8%

Curent mediu

- Curentul mediu este media ponderată pe timpi

- $$I_{CC_{\text{mediu}}} = \frac{\sum I_{CC_x} * t_x}{\sum t_x}$$

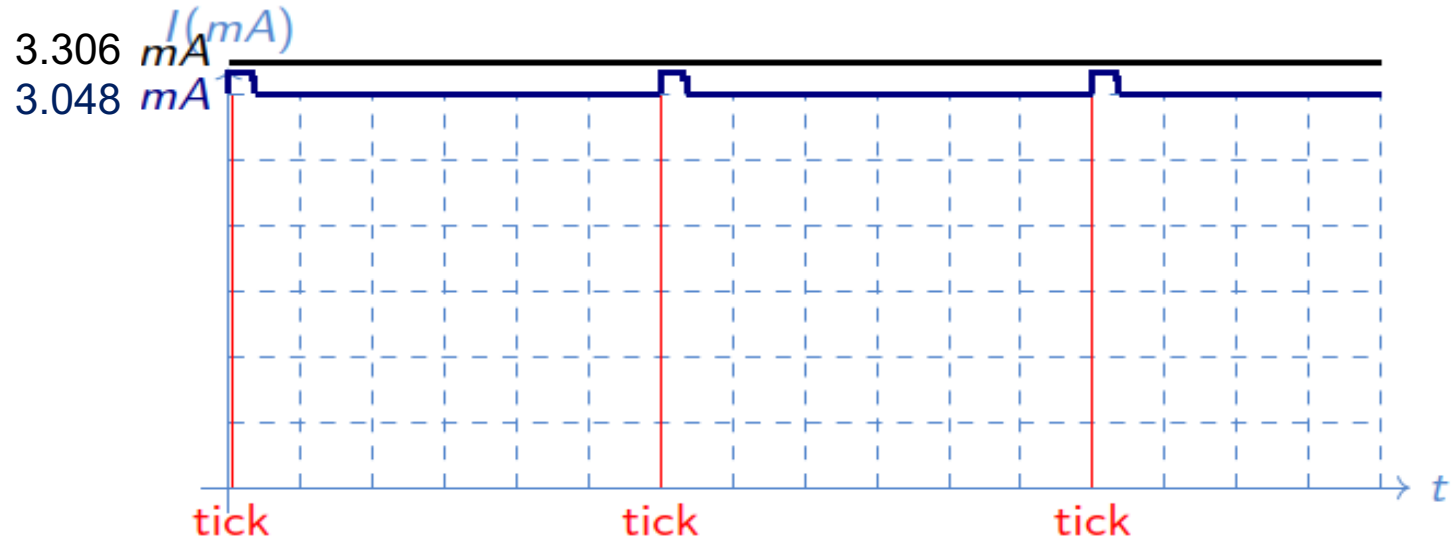
- $$I_{CC_{\text{mediu}}} = \frac{I_{CC_{\text{conversie}}} * t_{\text{conversie}} + I_{CC_{\text{comunicatie}}} * t_{\text{comunicatie}} + I_{CC_{\text{rest}}} * t_{\text{rest}}}{t_{\text{total}}}$$

- $$I_{CC_{\text{mediu}}} = \frac{3.174 * 200 + 3.129 * 200 + 3.048 * 999600}{1000000}$$

- $$I_{CC_{\text{mediu}}} = 3.048 \text{ mA}$$

Grafic de consum optimizat

- $$t_{\text{baterie}} = \frac{\text{Capacitate}}{\text{Consum}} = \frac{200\text{mAh}}{3.05\text{mA} + 0.1\text{mA}} = 63.5\text{h}$$



Idle Mode

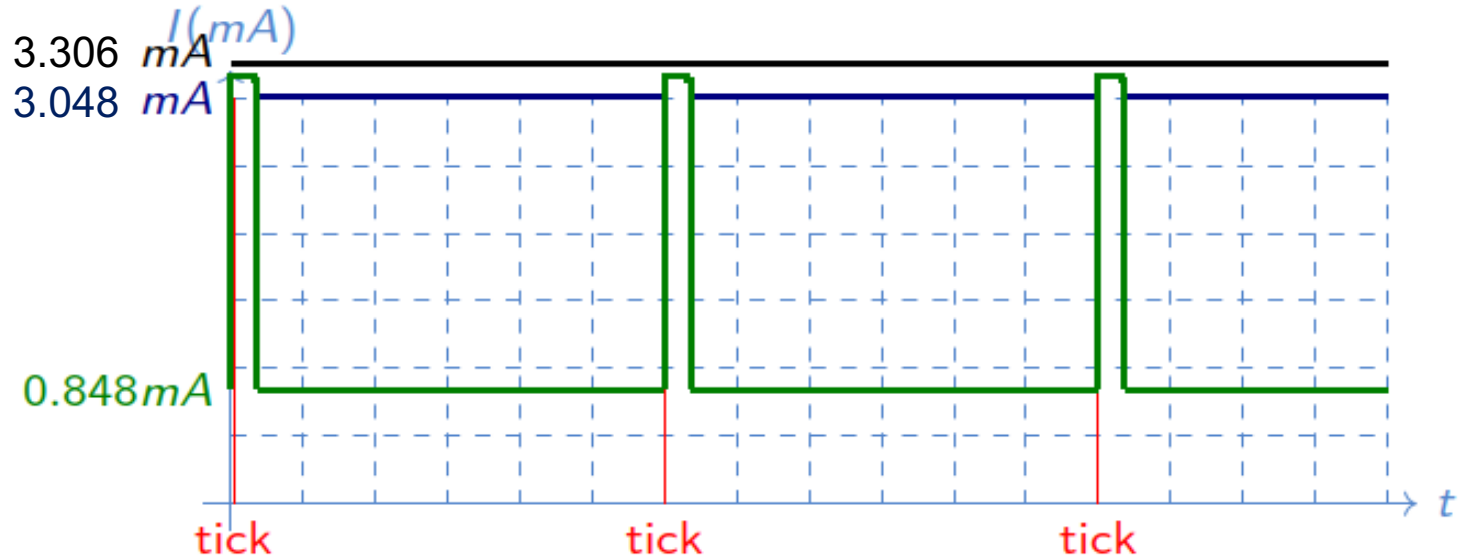
- Putem folosi Idle Mode, cu wake-up pe întreruperea de Timer1
- Putem număra maxim 4s
- Alegem întrerupere de Timer1 la 1s
- Neglijăm timpii de intrare/ieșire din Idle
- $I_{CC_{Idle}} = I_{CC_{Idle,8MHz,3.3V}} (1 + I_{TIM1,Idle}) = 0.8 \cdot 1.06 = 0.848\text{mA}$

Current medium Idle Mode

- $$I_{CC_{medium}} = \frac{\sum I_{CC_x} * t_x}{\sum t_x}$$
- $$I_{CC_{medium}} = \frac{I_{CC_{conversion}} * t_{conversion} + I_{CC_{communication}} * t_{communication} + I_{CC_{idle}} * t_{rest}}{t_{total}}$$
- $$I_{CC_{medium}} = \frac{3.174 * 200 + 3.129 * 200 + 0.848 * 999600}{1000000}$$
- $$I_{CC_{medium}} = 0.848 \text{ mA}$$

Grafic de consum Idle Mode

- $$t_{\text{baterie}} = \frac{\text{Capacitate}}{\text{Consum}} = \frac{200\text{mAh}}{0.85\text{mA} + 0.1\text{mA}} = 210.5\text{h}$$



Alte stări de power saving

- Power-down - cel mai eficient mod de (ne)lucru
 - Poate fi trezit cu: întrerupere externă, I^2C , watchdog
 - Consum la 25°C de 0.2 μ A fără watchdog
 - Noi am avea nevoie de watchdog să-l trezim \rightarrow 4 μ A
- Power-save - Adaugă timer-ul 2 dacă funcționează cu cristal propriu
 - Consum la 25°C, 3.3V de 0.75 μ A fără watchdog

Curent mediu Power-save Mode

- Curentul mediu este media ponderată pe timpi

- $$I_{CC_{\text{mediu}}} = \frac{\sum I_{CC_x} * t_x}{\sum t_x}$$

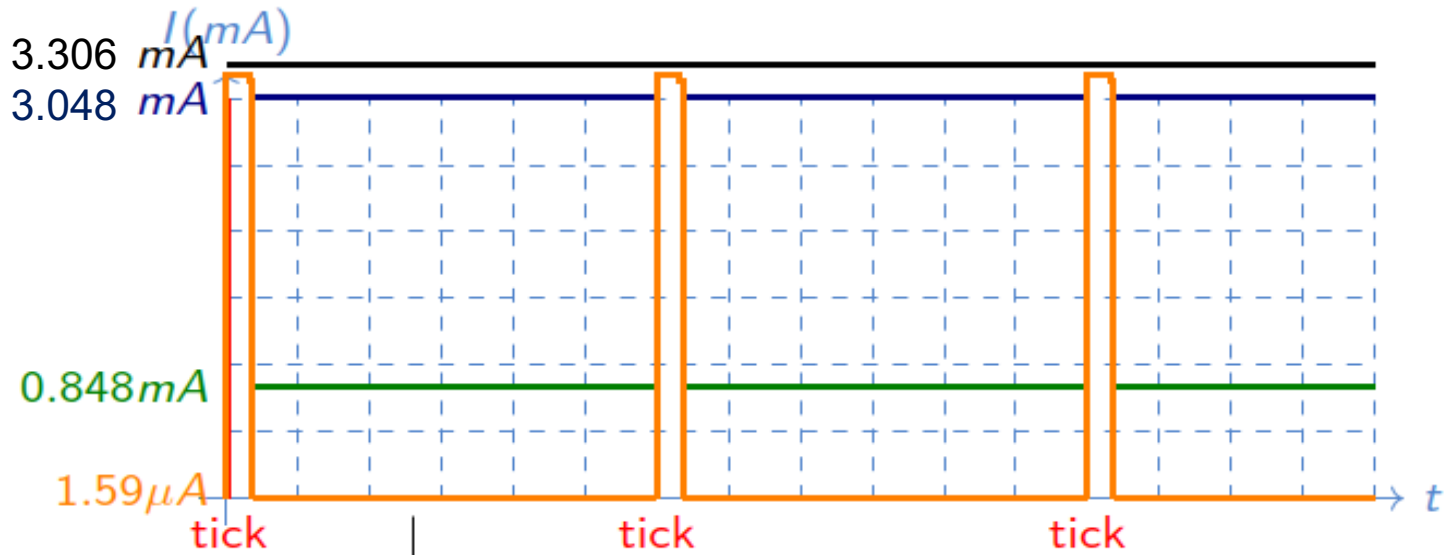
- $$I_{CC_{\text{mediu}}} = \frac{I_{CC_{\text{conversie}}} * t_{\text{conversie}} + I_{CC_{\text{comunicatie}}} * t_{\text{comunicatie}} + I_{CC_{\text{Power-save}}} * t_{\text{rest}}}{t_{\text{total}}}$$

- $$I_{CC_{\text{mediu}}} = \frac{3.174 * 200 + 3.129 * 200 + 0.00075 * 999600}{1000000}$$

- $$I_{CC_{\text{mediu}}} = 1.59 \mu\text{A}$$

Grafic de consum Power-save Mode

- $t_{\text{baterie}} = \frac{\text{Capacitate}}{\text{Consum}} = \frac{200\text{mAh}}{0.00159\text{mA} + 0.1\text{mA}} = 1968.7\text{h} \approx 82\text{zile}$
- Consumul MCU-ului este de 2096 ori mai mic!



Optimizări ADC

- Senzorul oferă o scară de $10\text{mV}/1^\circ\text{C}$, cu o precizie de 0.5°C
- Rezoluția ADC-ului este de 10 biți, la $3.3\text{V} \rightarrow 3.22\text{mV}$
 - Nu putem reduce rezoluția ADC-ului
 - Frecvența ADC-ului trebuie să fie între 50 și 200kHz
 - Cu PS 128 la 8Mhz este la $62.5\text{kHz} \rightarrow$ putem reduce prescaler-ul de 2 ori
 - Pentru rezoluție mai mică, puteam reduce la PS 16 sau 32

Curent mediu cu ADC rapid

- Am redus prescaler – ul de la 128 la 64

$$t_{conversie} = 13 \text{ cicliADC} \cdot 64 \cdot t_{cicluCPU}$$

$$t_{conversie} = 104\mu\text{s} \approx 100\mu\text{s}$$

- Curentul mediu este media ponderată pe timpi

$$I_{CC_{\text{mediu}}} = \frac{\sum I_{CC_x} \cdot t_x}{\sum t_x}$$

$$I_{CC_{\text{mediu}}} = \frac{I_{CC_{\text{conversie}}} \cdot t_{conversie} + I_{CC_{\text{comunicatie}}} \cdot t_{comunicatie} + I_{CC_{\text{Power-save}}} \cdot t_{rest}}{t_{total}}$$

$$I_{CC_{\text{mediu}}} = \frac{3.174 \cdot 100 + 3.129 \cdot 200 + 0.00075 \cdot 999700}{1000000}$$

$$I_{CC_{\text{mediu}}} = 1.27\mu\text{A}$$

Optimizări frecvență

- Dar dacă am face totul mai încet?
- 1MHz MCU, SPI 500kHz (minim /2 prescaler)
- ADC prescaler 8 → 125000

$$t_{conversie} = 13 \text{ cicliADC} \cdot 8 \cdot t_{cicluCPU}$$

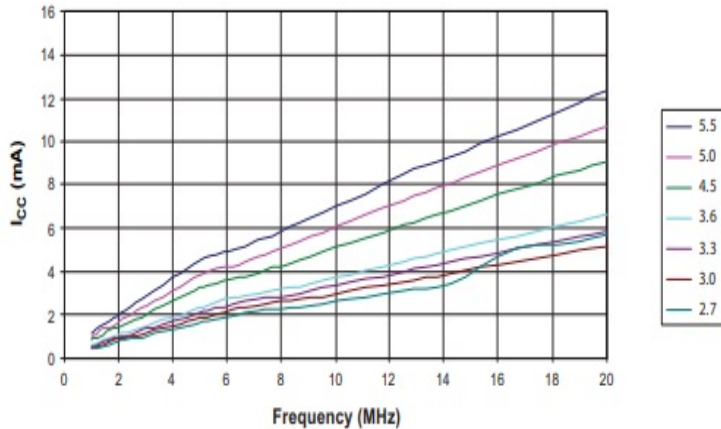
$$t_{conversie} = 104\mu\text{s} \simeq 100\mu\text{s}$$

- SPI durează mai mult

$$t_{comunicare} = \frac{100 * 8}{500000} \text{s} = 1.6\text{ms}$$

Calcul consum activ 1MHz

- $I_{CC_{total}} = I_{CC_{Activ,1MHz,3.3V}} (1 + I_{ADC} + I_{SPI} + I_{TIM1})$
- $I_{CC_{total}} = 0.55\text{mA} \cdot (1 + 0.042 + 0.027 + 0.016) = 0.597\text{mA}$



PRR bit	Additional Current consumption compared to Active with external clock (See Figure 30-1 and Figure 30-2)
PRTIM2	2.5%
PRTIM1	1.6%
PRTIM0	0.7%
PRADC	4.2%
PRSPI	2.7%

Curent mediu cu 1MHz

- Curentul mediu este media ponderată pe timpi

- $$I_{CC_{\text{mediu}}} = \frac{\sum I_{CC_x} * t_x}{\sum t_x}$$

- $$I_{CC_{\text{mediu}}} = \frac{I_{CC_{\text{conversie}}} * t_{\text{conversie}} + I_{CC_{\text{comunicatie}}} * t_{\text{comunicatie}} + I_{CC_{\text{rest}}} * t_{\text{rest}}}{t_{\text{total}}}$$

- $$I_{CC_{\text{mediu}}} = \frac{0.588 * 100 + 0.573 * 1600 + 0.00075 * 998300}{1000000}$$

- $$I_{CC_{\text{mediu}}} = 0.98 \mu\text{A}$$

- Consumul mediu al MCU-ului este de 3300 ori mai mic

- *Q: cât consumă celelalte componente?*

Alte optimizări

- Estimarea întregului cod (am ignorat mult)
- Scăderea tensiunii de alimentare (altă baterie?)
- Modelarea consumului pentru niveluri de baterie diferite
- Schimbarea cerințelor pentru optimizări în plus
 - Trebuie neapărat o citire la 1s pentru temperatură?
 - Trebuie neapărat făcut update la ecran dacă nu s-a schimbat nimic?
- Optimizări pe părțile ignorate (senzor, LCD)

Concluzii

- Cele mai mari economii - Idle sau Power-save
- Premisa - avem deja închise toate perifericele nefolosite
- Premisa2 - Avem un timp de repaus mare între procesări
- ATMEGA328P în general e mai eficient per MIPS la frecvențe mai mari pentru aceeași tensiune
 - Depinde și de ce alte operații mai facem
 - Degeaba reducem frecvența dacă nu reducem și tensiunea
- OBS – limitări practice!