Generování hodinového signálu, modul MCG,

spotřeba MCU

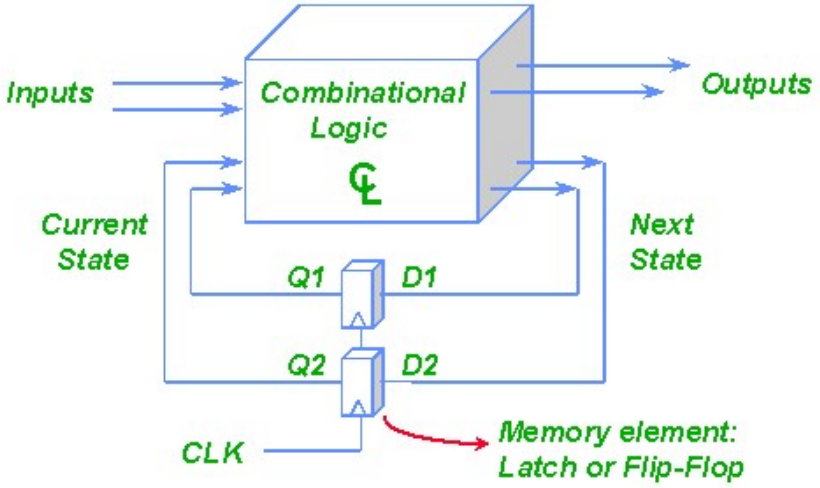
Mikroprocesorové a vestavěné systémy (IMP)

Richard Růžička

Fakulta informačních technologií VUT v Brně

Hodinový signál?

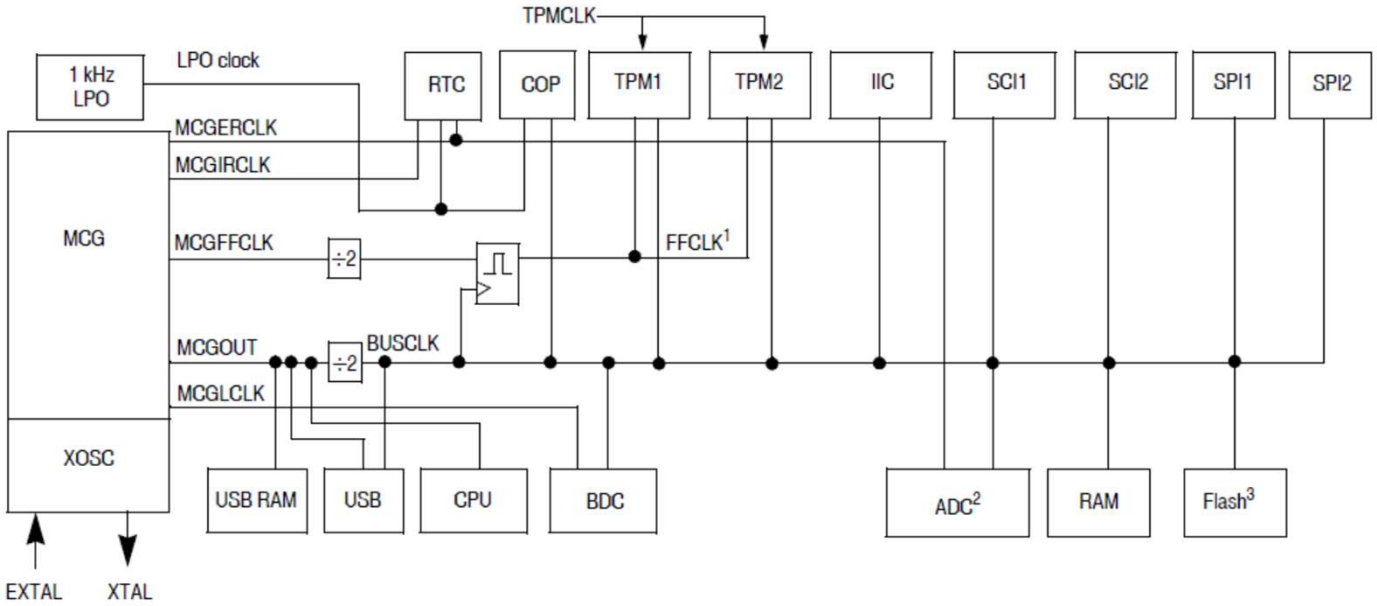
- Elektrický signál, který střídavě nabývá hodnot log. 0 a log. 1.
- Potřebuje jej každý sekvenční číslicový obvod (také MCU), jeho změny „způsobují“ změnu stavu.



Na co všechno je v MCU hodinový signál?

- Určitě jej potřebuje jádro MCU (procesor),
- ale taky:
  - čítače/časovače,
  - sériová rozhraní,
  - hodiny reálného času,
  - A/D převodník

    a další moduly.



Je vidět, že v MCU je více různých hodinových signálů, některé různě upravené z jednoho zdroje kmitů, další z jiných (alternativních) zdrojů.

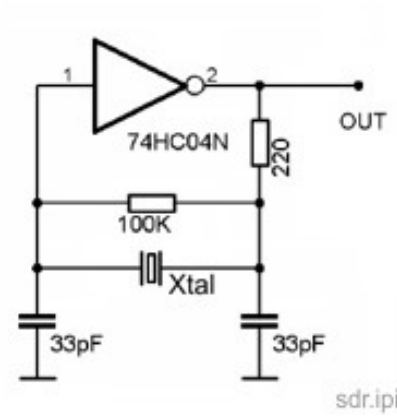
Zdroj kmitů - oscilátor

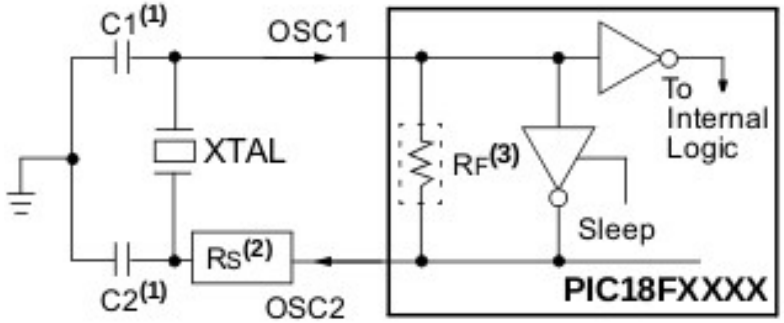
- Mít jeden zdroj kmitů pro celé MCU může být nepraktické:
  - Když se hlavní zdroj „porouchá“ (ulomený krystal, přepsání konfigurace), MCU by to ani nepoznal (zamrzne),
  - když hlavní zdroj hodin je „příliš žravý“ (cena za kvalitu) třeba pro pouhé čekání,

➡ Proto je vstupem generátoru hodin v MCU často více zdrojů než jeden.

Zdroj kmitů

- Oscilátor





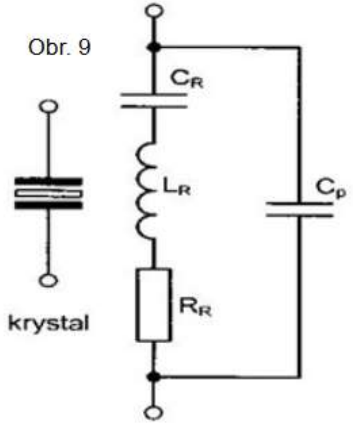
Oscilátory s krystalem

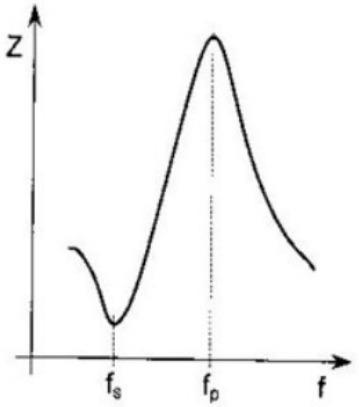
Pro větší stabilitu kmitočtu, v IT velmi používané.

Krystal je pasivní součástka, oscilátor je obvod, v němž se periodicky mění napětí. Vynalezen 1929.

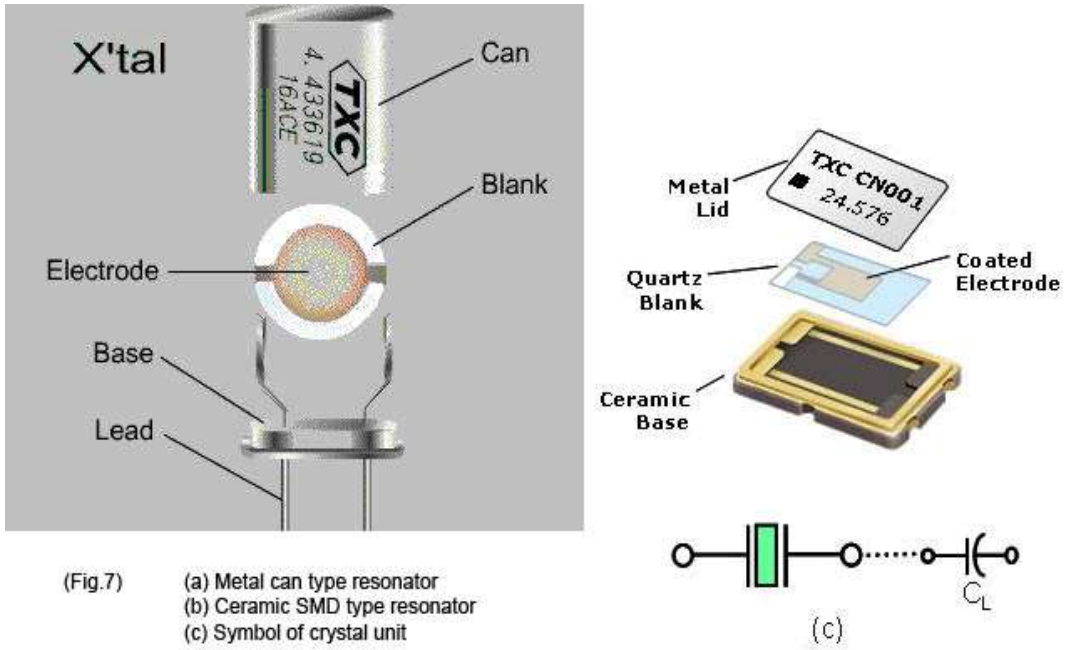
Krystal se do obvodu oscilátoru vkládá typicky do zpětné vazby.

Krystal se chová jako RLC rezonanční obvod. V sériové rezonanci má nejnižší impedanci, proto obvod se zpětnou vazbou kmitá na jeho sériovém rezonančním kmitočtu.

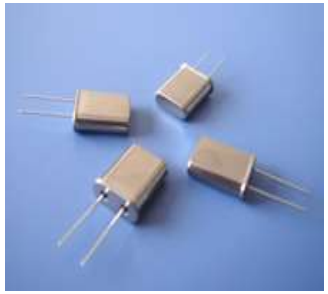


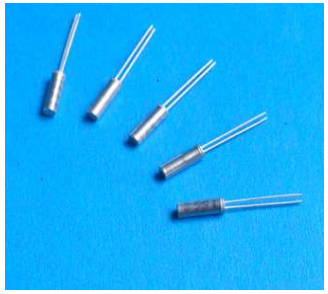


Krystal – obvodová součástka



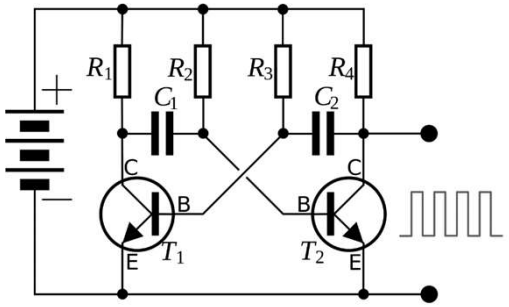
(Fig.7) (a) Metal can type resonator
(b) Ceramic SMD type resonator
(c) Symbol of crystal unit

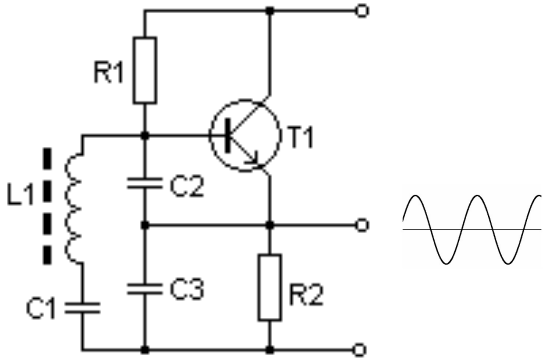




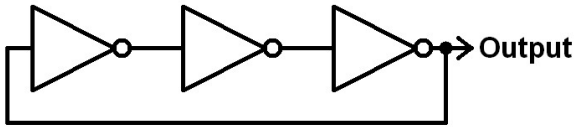
Oscilátor bez krystalu?

- Buď RC (LC) oscilátor





- nebo (typicky právě v číslicových obvodech) tzv. „ring oscillator“



Hodiny pro různé moduly MCU

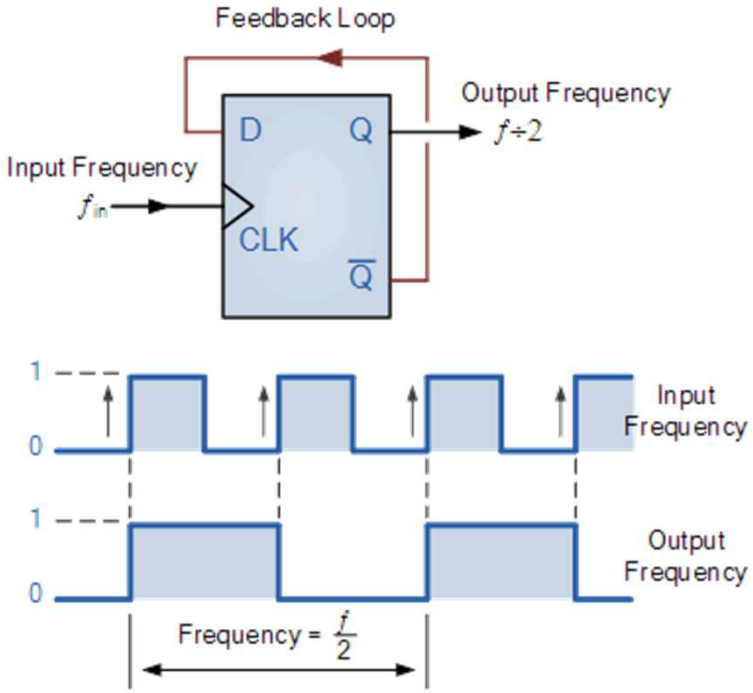
- Mít jeden hodinový signál pro všechny moduly je nepraktické:
  - Některé musí běžet pomaleji, některé rychleji,
  - některé moduly musí běžet stálou rychlostí, u jiných se mění podle okolností.

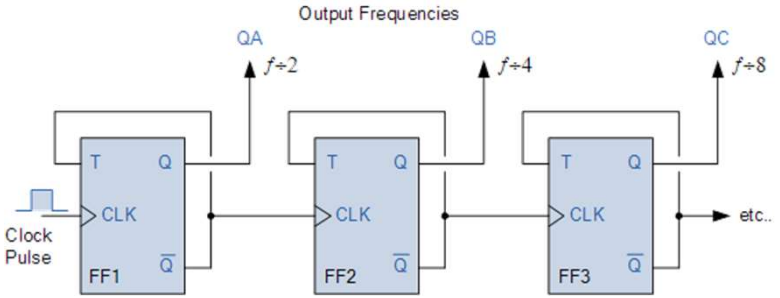
➡ Proto je výstupem generátoru hodin v MCU často více signálů než jeden.

… máme tedy více zdrojů kmitů, z nichž se generují různé hodinové signály. To má v MCU Kinetis na starosti modul MCG. Z jednoho zdroje je generováno více hodinových signálů.

Změna frekvence hodin

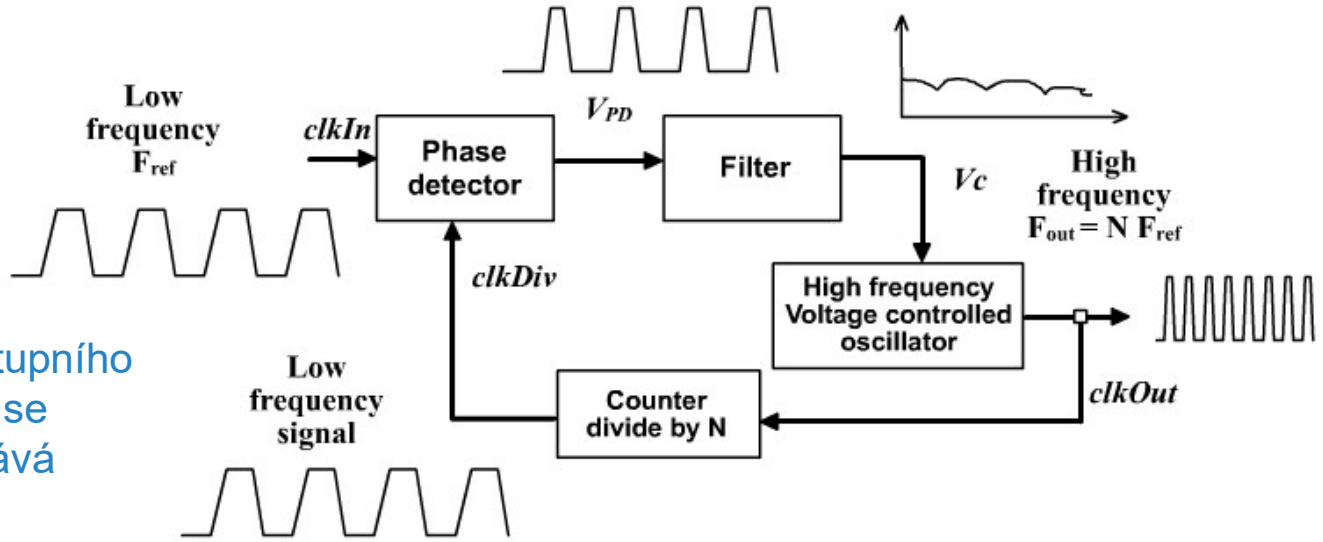
- Dělení kmitočtu





Fázový závěs

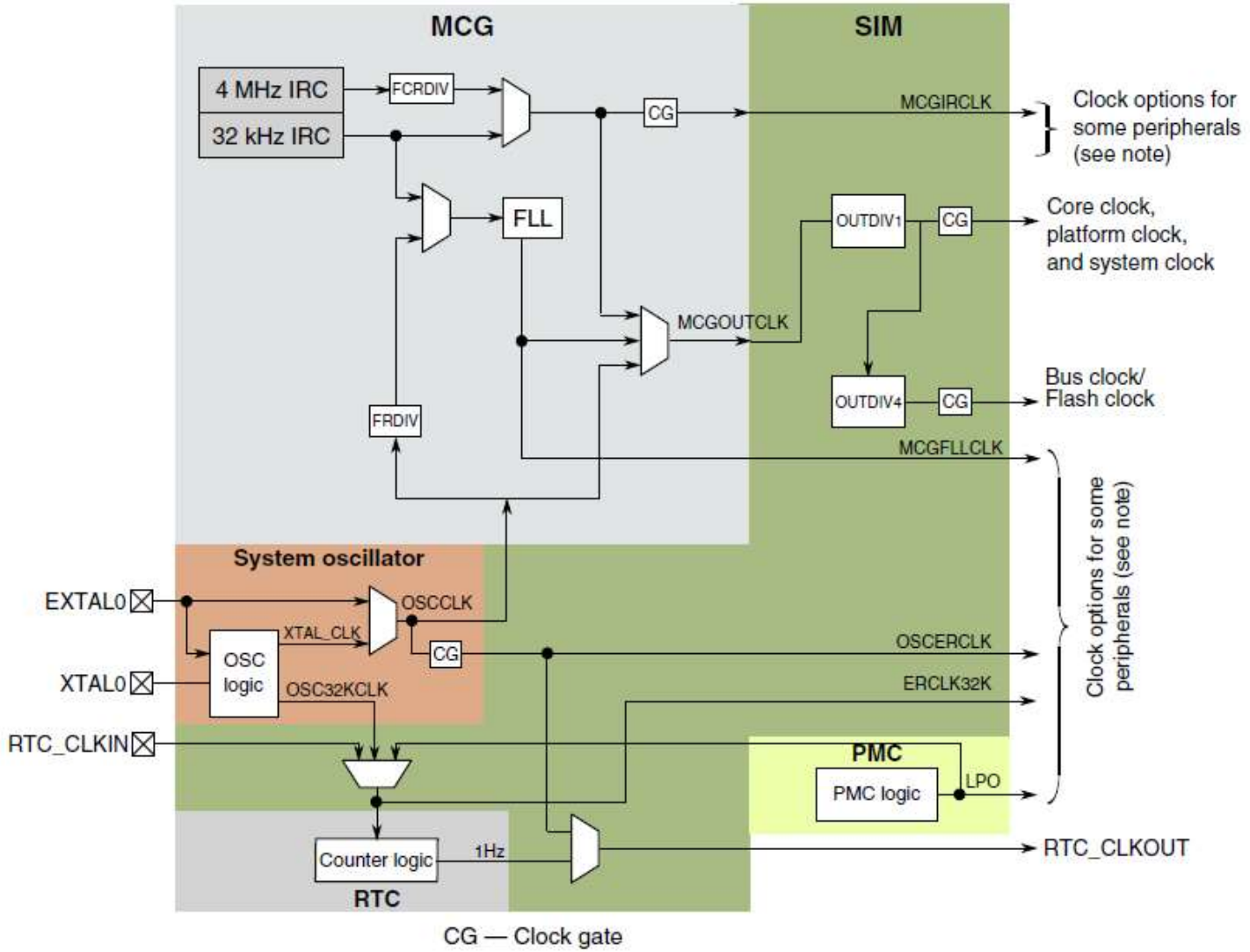
angl. Phase-Locked Loop, PLL



fáze vstupního signálu se porovnává s fází výstupního

... protože fáze je derivací frekvence, bude přesnost frekvence výstupu závislá na přesnosti frekvence vstupu.

Generátor hodin v MCU Kinetis KL05



Kde se bere signál?

- Zdroje kmitů hodinového signálu u MCU Kinetis KL05:

Tyto lze použít jako hlavní (pro jádro, sběrnici,...)

  - Internal Reference Clock (IRC) – vnitřní zdroj 4MHz a 32kHz.
  - System Oscillator – „vnější“ zdroj (kvalitu lze ovlivnit vnějšími součástkami) – buď krystal + oscilátor v MCU nebo celý vnější oscilátor.

  - Low-Power Oscillator (LPO) – 1kHz nezávislý oscilátor v modulu PMC (Power Management Controller)

Low Power Oscillator (LPO)

- 1kHz nezávislý zdroj hodin,
- běží ve všech režimech včetně režimů nízké spotřeby a spánku (s výjimkou VLLS0),
- Používá se:
  - pro watchdog (aby se MCU mohl „vzpamatovat“ i když dojde k zastavení hodin MCU),
  - pro RTC, který bdí i v režimech spánku,
  - stejně tak i pro LPTMR.

Internal Reference Oscillator

- Dva zdroje: 32 kHz a 4 MHz.
- Frekvence není přesná, ale lze vyladit.

Symbol	Description	Min.	Typ.	Max.	Unit	Notes
f_{ints_ft}	Internal reference frequency (slow clock) — factory trimmed at nominal V_{DD} and 25 °C	—	32.768	—	kHz	
f_{ints_t}	Internal reference frequency (slow clock) — user trimmed	31.25	—	39.0625	kHz	
f_{intf_t}	Internal reference frequency (fast clock) — user trimmed at nominal V_{DD} and 25 °C	3	—	5	MHz	

- 4 MHz zdroj lze jednoduše dělit na

  2 MHz, 1 MHz, 500 kHz, 250 kHz,

  125 kHz, 62,5 kHz a 31,25 kHz.

(dělitel je FCRDIV v registru MCG_SC)

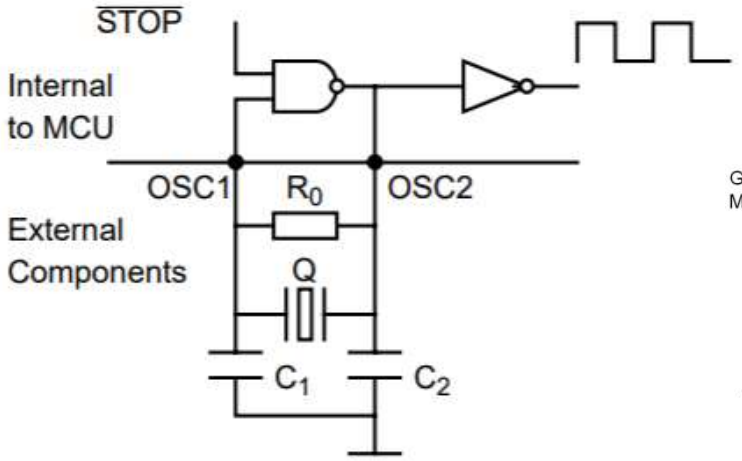
- frekvenci z 32 kHz zdroje lze zvýšit pomocí FLL (nebo PLL).

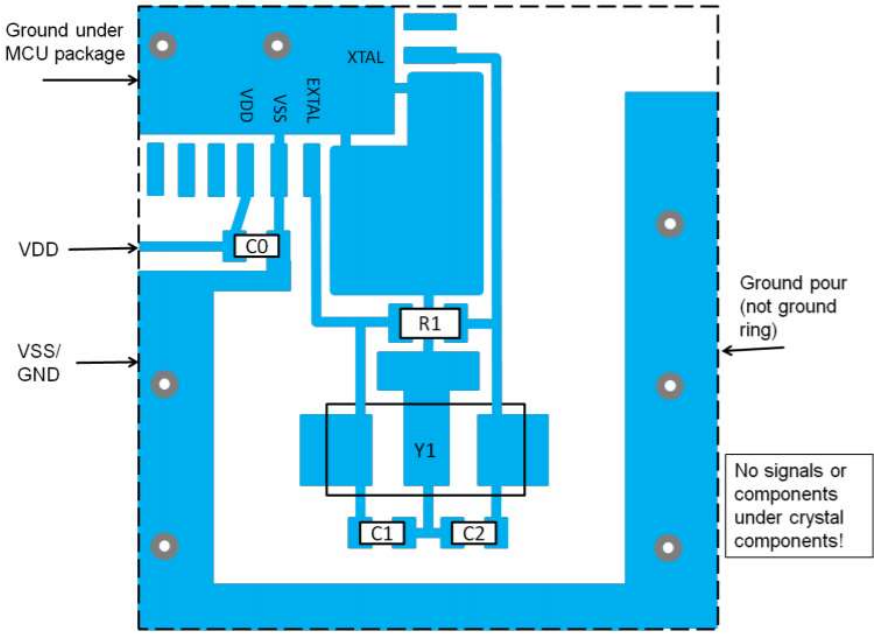
System Oscillator

- Předpokládá vně MCU alespoň krystal nebo rezonátor, který udává frekvenci:
  - 32 – 40 kHz
  - 3 – 32 MHz
- nebo vnější zdroj pravoúhlého signálu (střída 40-60%):
  - 0 – 48 MHz.

Frekvenci lze dále měnit naprogramováním FLL nebo PLL, je-li třeba.

Zapojení s krystalem





Proč tolik možností?

- Vnitřní oscilátor je úsporné řešení (netřeba dalších součástek), ale přesnost (a stabilita) nemusí vyhovovat v některých aplikacích.
- Externí oscilátor lze vyrobit daleko přesnější a stabilnější.

- Oscilátor kmitající na desítkách kHz má řádově nižší spotřebu než megahertzový, ale je pro některé aplikace pomalý.
- Typická frekvence 32,768 kHz, tzv. „hodinkový“ krystal – dělením 2^{15} lze snadno získat přesné 1s pulsy.

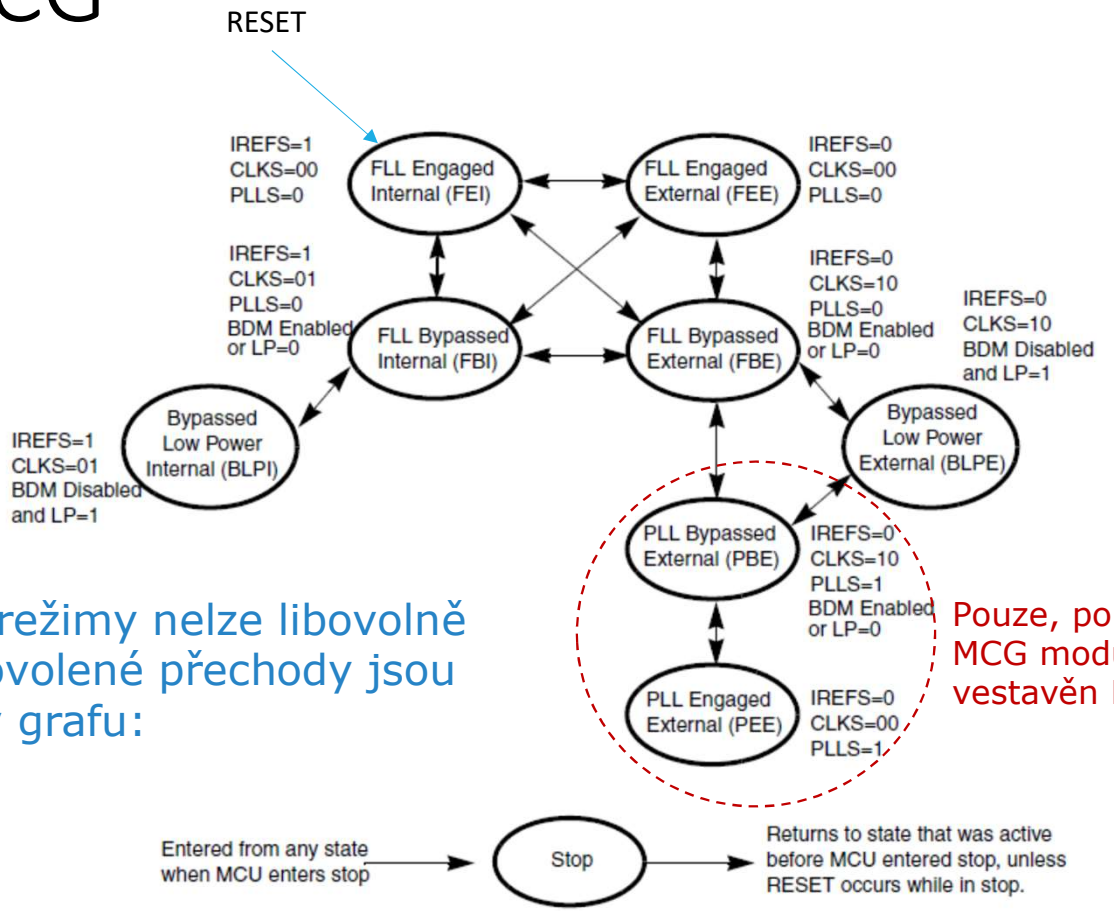
Co všechno umí MCG?

- „Krmit“ MCU hodinovým signálem z interního zdroje,
- „krmit“ MCU hodinovým signálem z externího zdroje,
- změnit frekvenci hodinového signálu z interního nebo externího zdroje s pomocí FLL (nebo PLL na některých typech).

To všechno lze softwarově nastavovat

a plynou z toho (kombinováním uvedených možností) různé režimy činnosti MCG.

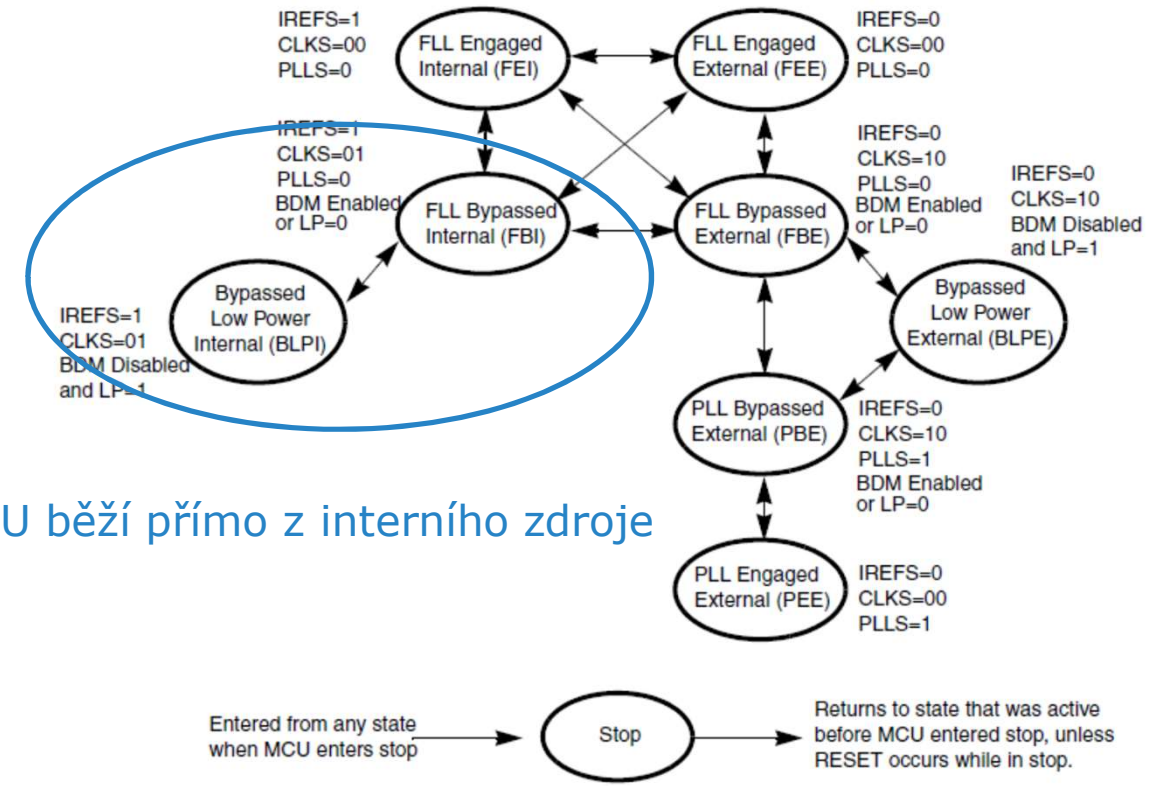
Režimy MCG



Pozor, mezi režimy nelze libovolně přepínat. Dovolené přechody jsou vyznačeny v grafu:

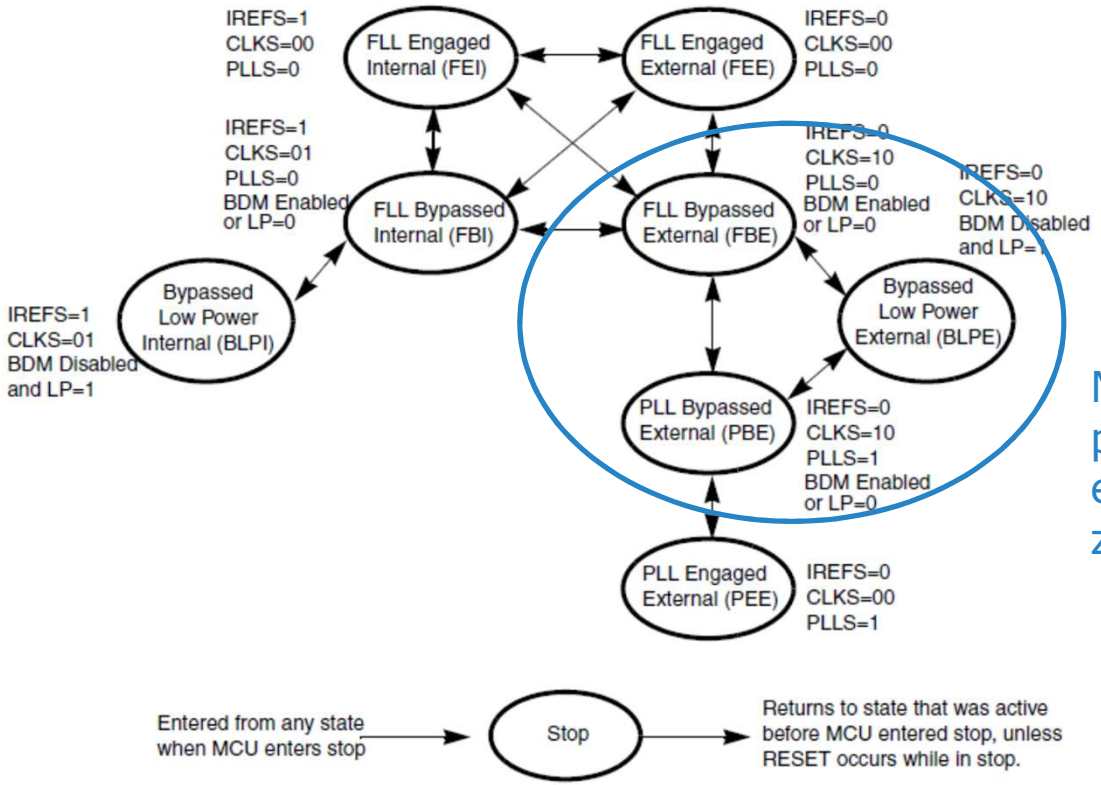
Pouze, pokud má MCG modul MCU vestavěn PLL

Režimy MCG



MCU běží přímo z interního zdroje

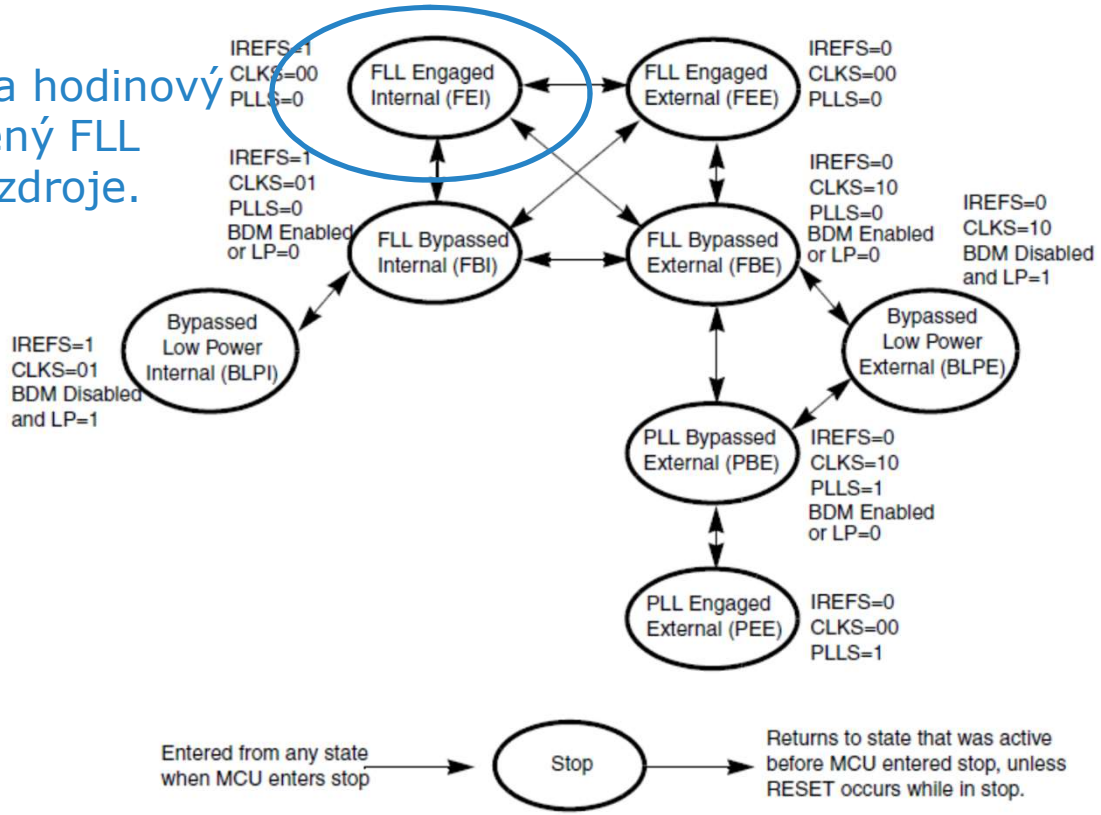
Režimy MCG



MCU běží přímo z externího zdroje

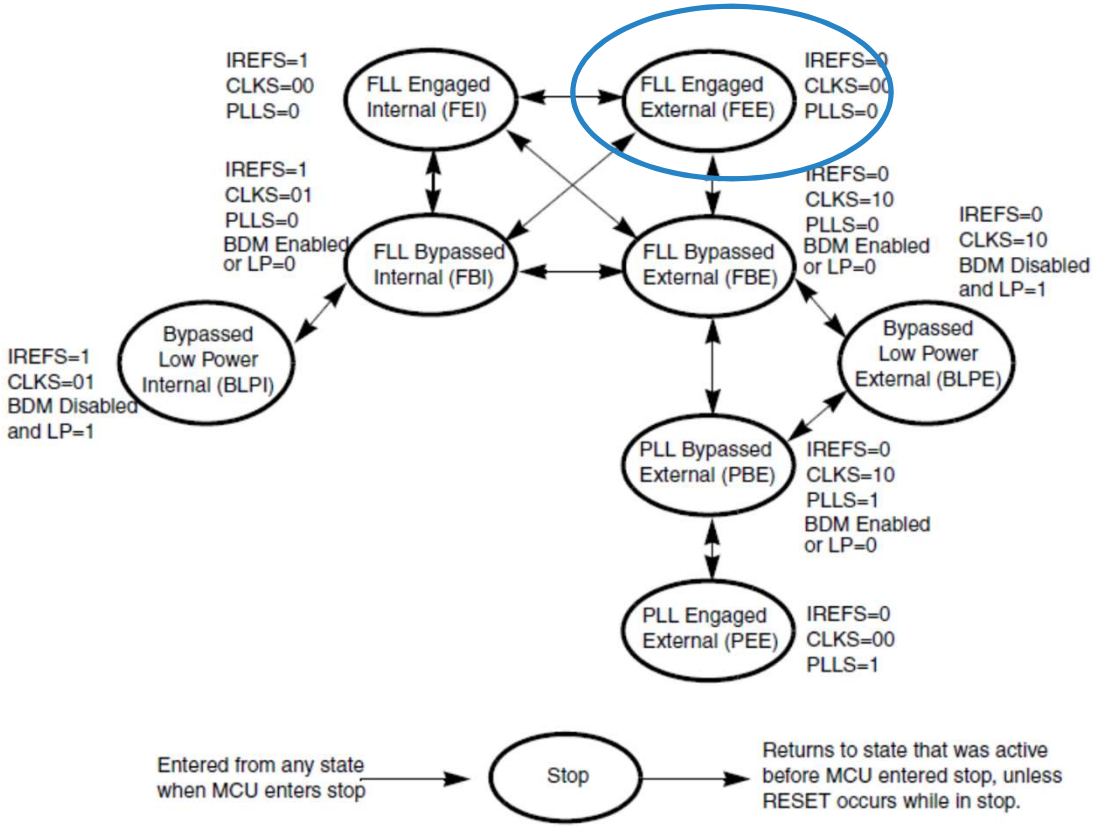
Režimy MCG

MCU běží na hodinový signál tvořený FLL z interního zdroje.

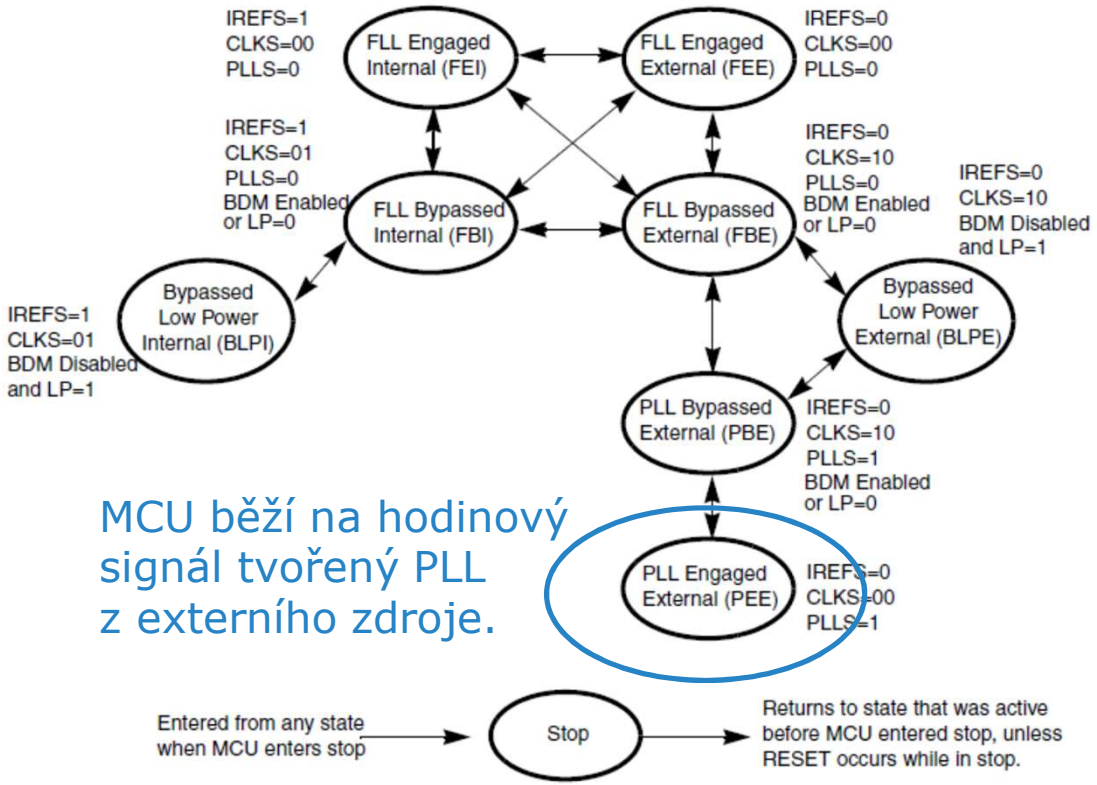


Režimy MCG

MCU běží na hodinový signál tvořený FLL z externího zdroje.

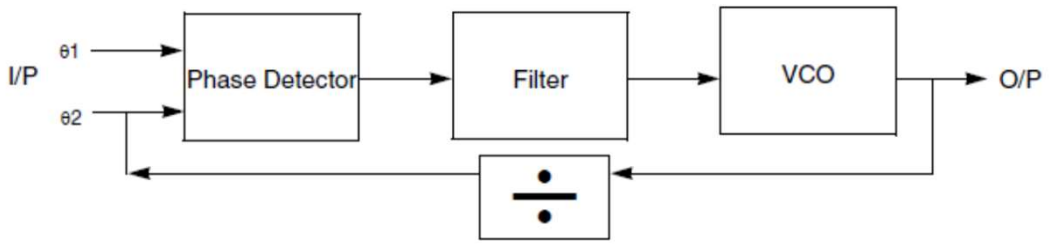


Režimy MCG



MCU běží na hodinový signál tvořený PLL z externího zdroje.

Jak funguje PLL (Phase-Locked Loop)?



- VCO = Voltage Controlled Oscillator
  - generuje hodinový signál, ale:
    - fáze tohoto signálu se porovnává s fází vstupního signálu (který může být zlomkem signálu výstupního).
    - Protože fáze je derivací frekvence, **bude přesnost výstupního signálu odpovídat přesnosti vstupního.**

Povšimněte si, že

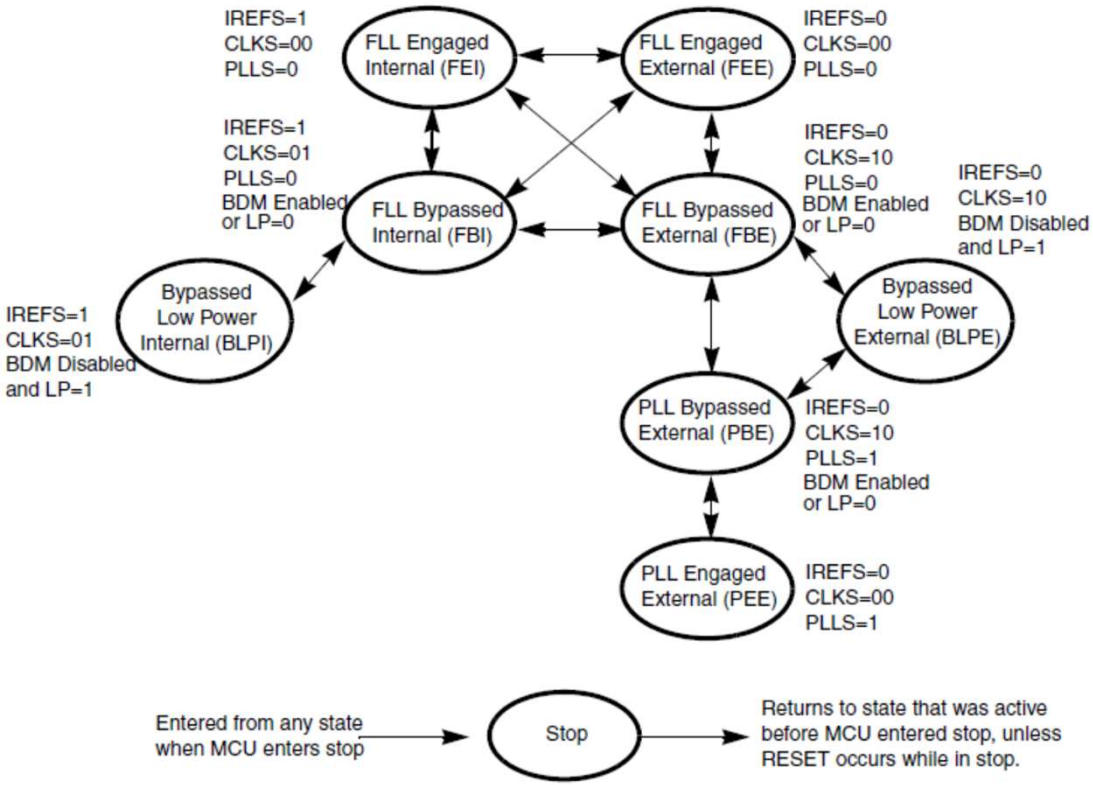
kombinace PLL a vnitřní zdroj hodin neexistuje!

Proč?

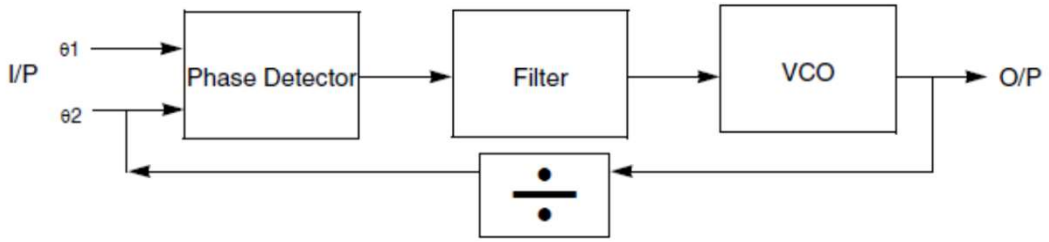
Protože PLL je přesný způsob generování hodin (daleko přesnější než FLL), zato energeticky náročnější.

Interní zdroj je ale nepřesný.

Nemá to tedy smysl.
Utratí se energie navíc, ale nic lepšího (oproti FLL) bychom za ni nedostali.

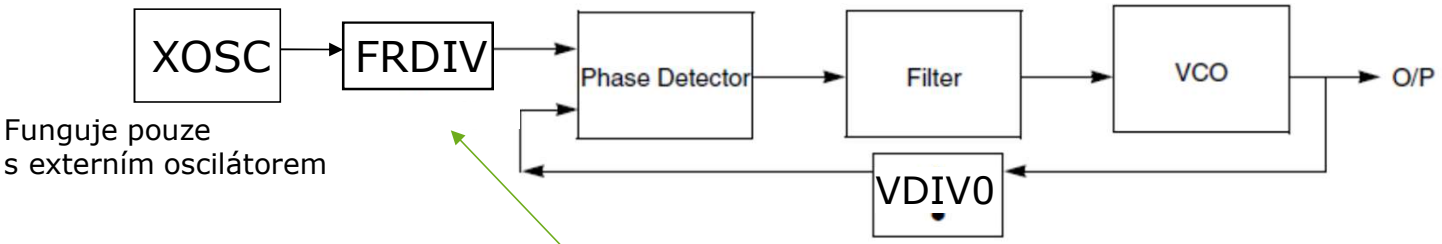


Jak funguje PLL?



- Zpětnovazební regulační smyčka, dělička ve zpětné vazbě určuje poměr mezi vstupní a výstupní frekvencí.
- Fázový detektor porovná fáze vstupního a výstupního signálu, rozdíl fází vyjádří napětím (kladným nebo záporným).
- Tímto napětím se dolaďuje frekvence oscilátoru.
- Dolnopropustný filtr stabilizuje chod zpětné vazby (odstraňuje šum a oscilace výstupu detektoru).

PLL v MCG



Funguje pouze
s externím oscilátorem

Kombinací dělicího poměru předděličky RDIV
a děličky ve zpětné vazbě VDIV lze dosáhnout
řady různých frekvencí výstupního signálu.

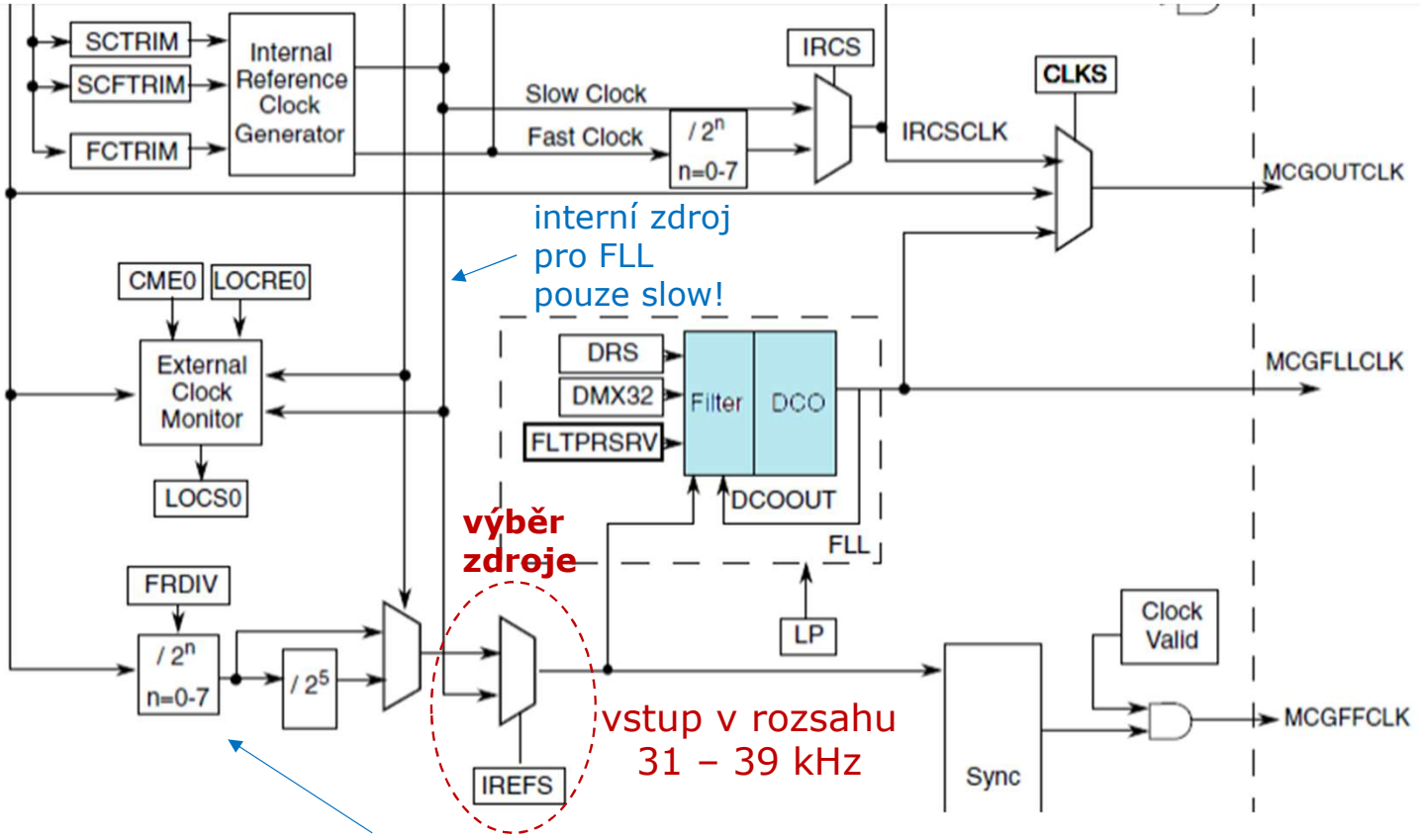
MCG_C1

Bit	7	6	5	4	3	2	1	0
Read Write	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

MCG_C6

Bit	7	6	5	4	3	2	1	0
Read Write	LOLIE0	PLLS	CME0	VDIV0				
Reset	0	0	0	0	0	0	0	0

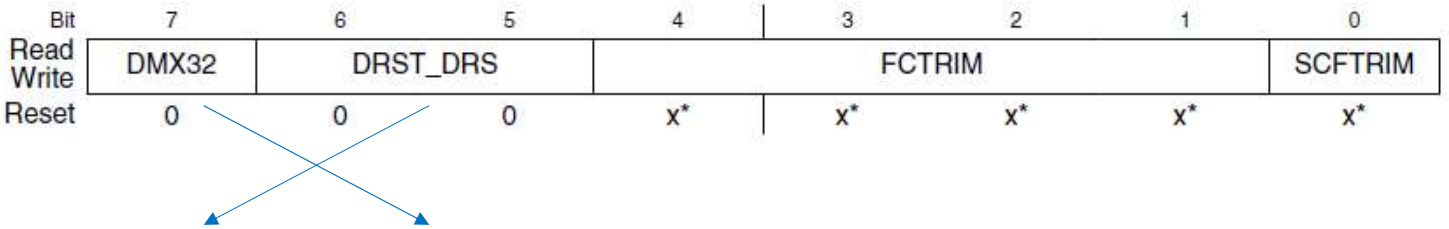
Zdroj pro FLL



Je-li vstupem signál z externího zdroje, je třeba pomocí FRDIV dostat frekvenci vstupního signálu pro FLL do rozsahu 31,25 – 39,065 kHz!

Výstup FLL

Frekvence generovaná FLL se nastavuje v MCG_C4:



Bit	7	6	5	4	3	2	1	0
Read Write	DMX32	DRST_DRS		FCTRIM				SCFTRIM
Reset	0	0	0	x*	x*	x*	x*	x*

DRST_DRS	DMX32	Reference Range	FLL Factor	DCO Range
00	0	31.25–39.0625 kHz	640	20–25 MHz
	1	32.768 kHz	732	24 MHz
01	0	31.25–39.0625 kHz	1280	40–50 MHz
	1	32.768 kHz	1464	48 MHz
10	0	31.25–39.0625 kHz	1920	60–75 MHz
	1	32.768 kHz	2197	72 MHz
11	0	31.25–39.0625 kHz	2560	80–100 MHz
	1	32.768 kHz	2929	96 MHz

Jaké hodiny může mít KL05?

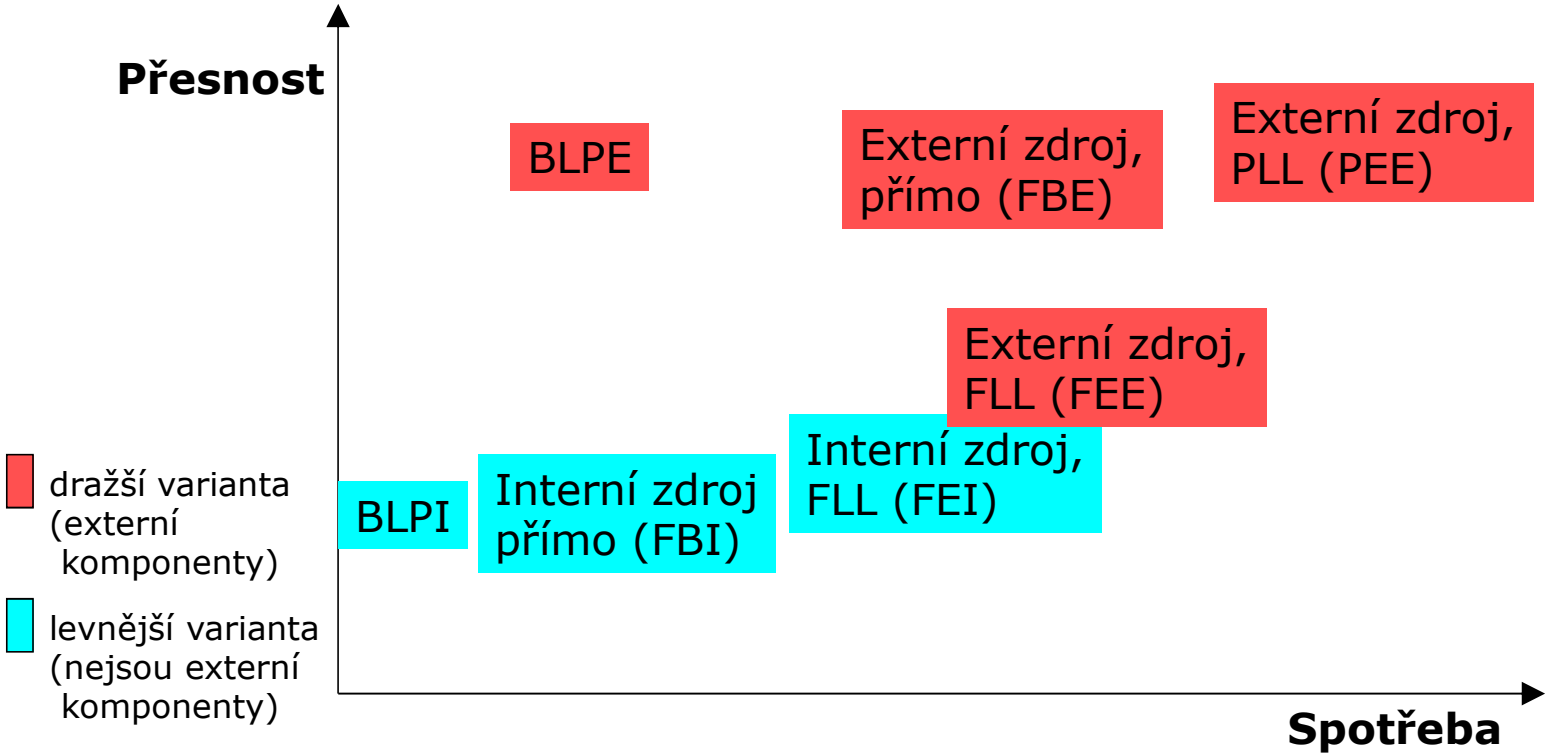
Table 24-12. MCGOUTCLK Frequency Calculation Options

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
FEI (FLL engaged internal)	(f_{int} * F)	Typical $f_{MCGOUTCLK}$ = 21 MHz immediately after reset.
FEE (FLL engaged external)	(f_{ext} / FLL_R) *F	f_{ext} / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBE (FLL bypassed external)	OSCCLK	OSCCLK / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBI (FLL bypassed internal)	MCGIRCLK	Selectable between slow and fast IRC
BLPI (Bypassed low power internal)	MCGIRCLK	Selectable between slow and fast IRC
BLPE (Bypassed low power external)	OSCCLK	

1. FLL_R is the reference divider selected by the C1[FRDIV] bits, F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits.

MCGOUTCLK se dále dělí dělitelem 1-16, lze nastavit v registru SIM_CLKDIV1.

K čemu jsou dobré všechny ty režimy MCG?



Sledování přítomnosti hodinového signálu

MCG_C6:

Bit	7	6	5	4	3	2	1	0
Read / Write	LOLIE0	PLLS	CME0	VDIV0				
Reset	0	0	0	0	0	0	0	0

Clock Monitor Enable:
je-li zapnut a ztratí-li
se hodinový signál,
indikuje to bit v MCG_SC,
případně se generuje reset.
Funguje jen pro externí
zdroj hodin!

MCG_SC2:

Bit	7	6	5	4
Read / Write	LOCRE0	0	RANGE0	
Reset	1	0	0	0

MCG_SC:

Bit	7	6	5	4	3	2	1	0
Read	ATME	ATMS	ATMF	FLTPRSRV	FCRDIV			LOCS0
Write								
Reset	0	0	0	0	0	0	1	0

Proč sledovat, zda hodiny jdou?

- Pokud zmizí externí hodinový signál, MCU se zastaví. Toto není možné detekovat softwarově, musí to udělat HW (proto je na to obvod v MCG).
- Je-li přitom zapnuto něco „nebezpečného“, není, kdo by to vypnul, může nastat škoda.
- Řešení – RESET:
  - po RESETu začíná MCU vždy s vnitřním zdrojem hodin,
  - po resetu se zpravidla celý systém inicializuje do nějakého bezpečného stavu,
  - lze detekovat, že důvodem k RESETu byl výpadek externích hodin a podle toho se zařídit.

Proč se detekují jen externí hodiny? Co řeší Watchdog?

Sledování kvality hodinového signálu v případě využití PLL (je-li v MCG)

MCG_C6

Bit	7	6	5	4	3	2	1	0
Read / Write	LOLIE0	PLLS	CME0	VDIV0				
Reset	0	0	0	0	0	0	0	0

LOLIE: Loss of Lock Interrupt Enable

Loss of Lock:
pokud PLL ztratí
vazbu, generuje se přerušení.
Přerušovací rutina může vyčkat
na obnovení nebo změnit
režim MCG a pod.

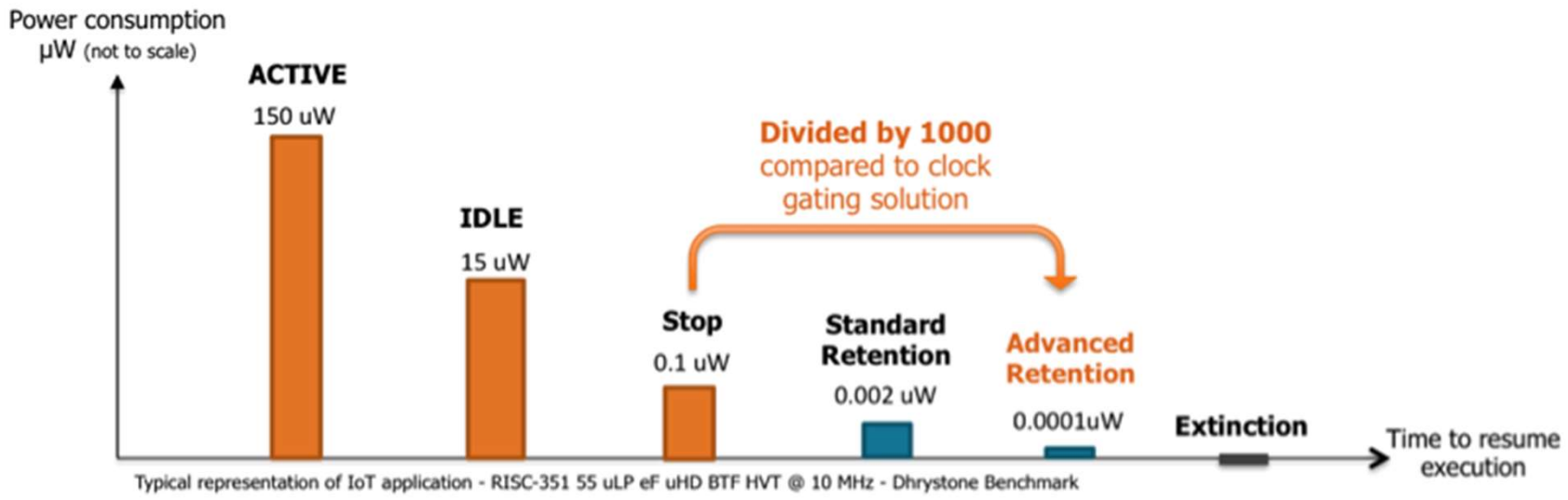
LOLS: Loss of Lock Status – 1 = došlo ke ztrátě
LOCK: Lock Status – 1 = PLL zpětná vazba běží

MCG_S

Bit	7	6	5	4	3	2	1	0
Read	LOLS0	LOCK0	PLLST	IREFST	CLKST		OSCINIT0	IRCST
Write								
Reset	0	0	0	1	0	0	0	0

Proč sledovat, zda PLL „je zavěšeno“?

- Využití PLL předpokládá, že záleží na přesnosti hodinového signálu.
- Pokud se z nějakého důvodu nedaří udržet „závěs“ na vstupním signálu, zřejmě hodiny nejsou takové, jak se předpokládá.
- Softwarově to je velmi obtížné detekovat.



Spotřeba mikrokontroléru

a jak ji ovlivnit při návrhu, jak ji řídit a také jak důraz na spotřebu ovlivní návrh vestavěného systému.

Spotřeba mikrokontroléru

- Většina vyrobena CMOS technologií, proto spotřeba silně závisí na rychlosti přepínání úrovní, tj. frekvenci hodin a použitém napájecím napětí.

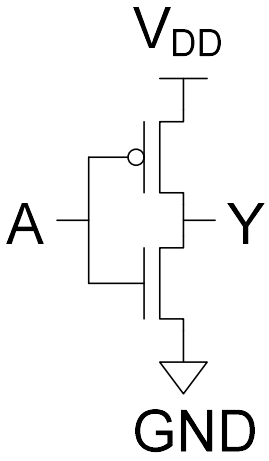
- Napájecí napětí mikrokontrolérů se pohybuje od 1,8V přes typicky 3V až do 5V. Při bateriovém provozu to předpokládá zpravidla více než jeden článek (nebo použití měniče napětí).
- Určitě ale platí, že nižší napětí = nižší spotřeba.

Spotřeba CMOS obvodu

- $P_{total} = P_{dynamic} + P_{static}$
- Dynamic power: $P_{dynamic} = P_{switching} + P_{shortcircuit}$
  - Switching load capacitances (sestava hradlo – izolace – substrát je kondenzátor!)
  - Short-circuit current (co projde shora dolů, když se mění stav)

… asi nejpodstatnější složka: $P_{\text{switching}} = \alpha C V_{DD}^{\ 2} f$

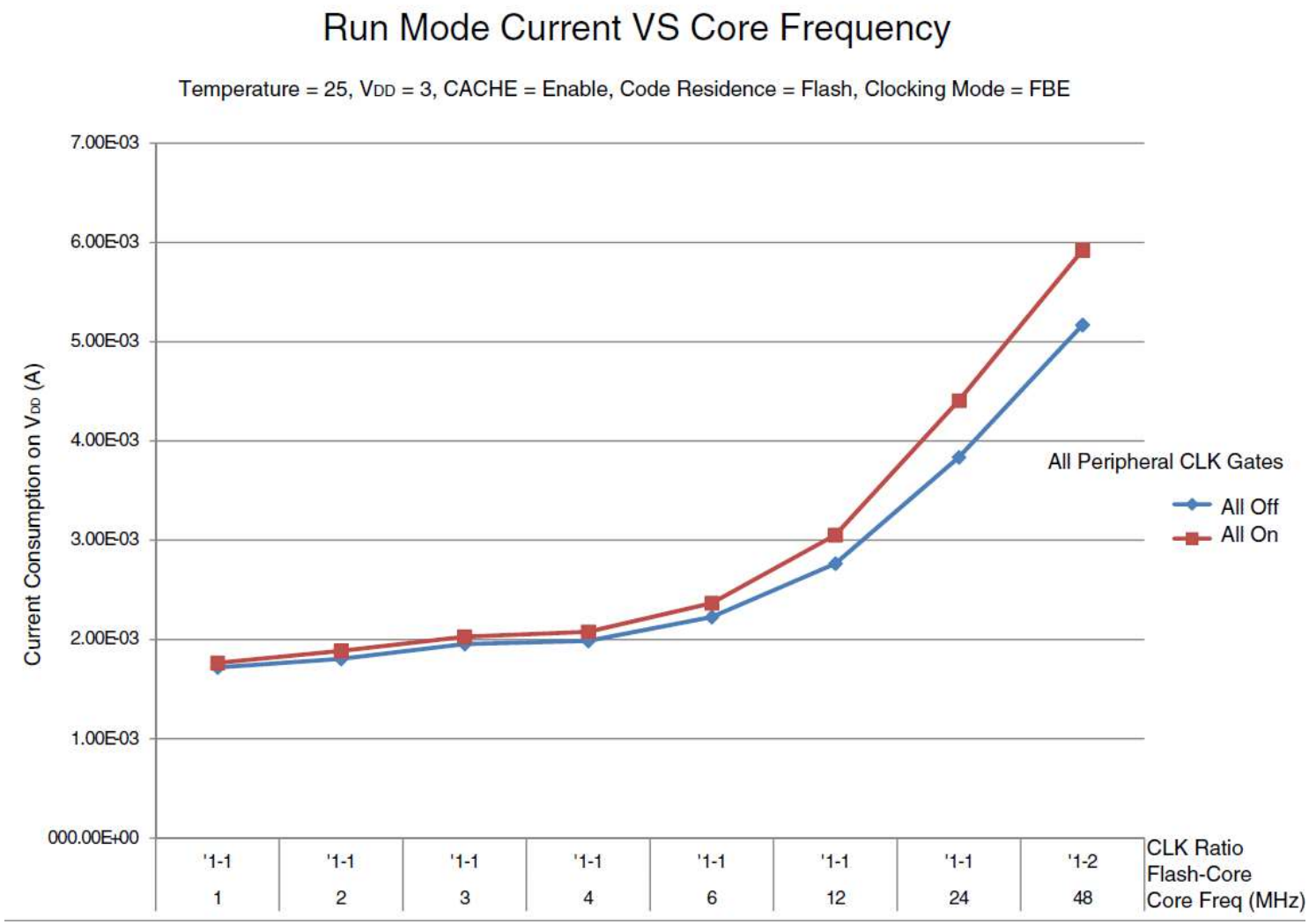
- Static power: $P_{static} = (I_{sub} + I_{gate} + I_{junct} + I_{contention})V_{DD}$
  - Subthreshold leakage (co propouští tranzistory zavřeným kanálem - když je $V_{GS} < V_{th}$)
  - Gate leakage (co proteče izolací mezi hradlem a kanálem)
  - Junction leakage (co projde závěrně polarizovanými přechody)
  - Contention current (statické otevření horní i dolní poloviny)



Spotřeba mikrokontroléru

- Nejvíce změn úrovní se jistě odehrává v jádru při vykonávání programu.
- Je plýtváním energií nechat jádro MCU běžet zbytečně rychle.
- U moderních MCU lze frekvenci hodin nastavit v širokém rozsahu a to softwarově.
- Dále lze konfigurovat či vypínat periferie na čipu tak, aby se vzhledem ke konkrétní aplikaci neplýtvalo.

Spotřeba Kinetis KL05



Příklad z dokumentace KL05

Symbol	Description	Min.	Typ.	Max.[1]	Unit
I_{DDA}	Analog supply current	—	—	See note	mA
I_{DD_RUNCO}	Run mode current in compute operation - 48 MHz core / 24 MHz flash / bus clock disabled, code of while(1) loop executing from flash • at 3.0 V	—	4.0	4.3	mA
I_{DD_RUN}	Run mode current - 48 MHz core / 24 MHz bus and flash, all peripheral clocks disabled, code executing from flash • at 3.0 V	—	4.9	5.3	mA
I_{DD_RUN}	Run mode current - 48 MHz core / 24 MHz bus and flash, all peripheral clocks enabled, code executing from flash • at 3.0 V • at 25 °C • at 125 °C	— —	5.7 6.0	5.8 6.2	mA
I_{DD_WAIT}	Wait mode current - core disabled / 48 MHz system / 24 MHz bus / flash disabled (flash doze enabled), all peripheral clocks disabled • at 3.0 V	—	2.7	2.9	mA

Příklad z dokumentace KL05

Symbol	Description	Min.	Typ.	Max.[1]	Unit
I_{DD_WAIT}	Wait mode current - core disabled / 24 MHz system / 24 MHz bus / flash disabled (flash doze enabled), all peripheral clocks disabled • at 3.0 V	—	2.2	2.3	mA
I_{DD_PSTOP2}	Stop mode current with partial stop 2 clocking option - core and system disabled / 10.5 MHz bus / flash disabled (flash doze enabled) • at 3.0 V	—	1.5	1.7	mA
I_{DD_VLPRCO}	Very-low-power run mode current in compute operation - 4 MHz core / 0.8 MHz flash / bus clock disabled, code executing from flash • at 3.0 V	—	182	253	µA
I_{DD_VLPR}	Very low power run mode current - 4 MHz core / 0.8 MHz bus and flash, all peripheral clocks disabled, code executing from flash • at 3.0 V	—	213	284	µA
I_{DD_VLPR}	Very low power run mode current - 4 MHz core / 0.8 MHz bus and flash, all peripheral clocks enabled, code executing from flash • at 3.0 V	—	243	313	µA

Režimy běhu s nízkou spotřebou

Příklad z dokumentace KL05

Režimy WAIT a STOP

I_{DD_VLPW}	Very low power wait mode current - core disabled / 4 MHz system / 0.8 MHz bus / flash disabled (flash doze enabled), all peripheral clocks disabled • at 3.0 V	—	111	170	µA
I_{DD_STOP}	Stop mode current • at 3.0 V • at 25 °C • at 50 °C • at 70 °C • at 85 °C • at 105 °C	 — — — — —	 257 265 278 295 353	 277 285 303 326 412	µA
I_{DD_VLPS}	Very-low-power stop mode current • at 3.0 V • at 25 °C • at 50 °C • at 70 °C • at 85 °C • at 105 °C	 — — — — —	 2.25 4.08 8.10 14.18 37.07	 5.76 8.27 14.52 23.78 58.58	µA

Příklad z dokumentace KL05

Režimy STOP s velmi nízkou spotřebou

I_{DD_VLLS0}	Very-low-leakage stop mode 0 current (SMC_STOPCTRL[PORPO] = 0) • at 3.0 V • at 25 °C • at 50 °C • at 70 °C • at 85 °C • at 105 °C	— — — — —	0.38 0.88 2.10 4.14 12.00	0.54 1.23 2.95 5.59 15.73	µA
I_{DD_VLLS0}	Very-low-leakage stop mode 0 current (SMC_STOPCTRL[PORPO] = 1) • at 3.0 V • at 25 °C • at 50 °C • at 70 °C • at 85 °C • at 105 °C	— — — — —	0.30 0.79 2.01 4.05 11.96	0.45 1.12 2.82 5.45 15.63	µA

Jak dlouho vydrží KL05 na baterii CR2032?

CR 2032: 3V, 210 mAh

Vše běží, jádro 48 MHz (max. 5,8 mA) ⇨ minimálně 36 hodin

Vše běží ve VLPR režimu, jádro 4 MHz (max. 313 µA)⇨
minimálně 28 dní

ale: ve VLLS0 režimu max. 0,45 µA ⇨ **53 let!**

necháme-li běžet RTC (vezme si navíc 0,357 µA) to bude „jen“ 30 let.

Pozor, probuzení z VLLS0 režimu do běhu (RUN) trvá až 115 µs!

Předpokládá se teplota čipu 25°C, napájení přímo 3V z baterie.

Řízení spotřeby jádra MCU

Realizuje se vhodným nastavením hodin, aby se stihlo, co je potřeba, ale hodiny neběžely zbytečně rychle.

V případě, že není třeba *něco dělat*, jádro se uspí.

Nejčastější situace – čeká se, až nastane nějaká událost, ať už asynchronní (jádro se budí přerušením/resetem) nebo synchronní (uplyne určený čas odpočítávaný např. časovačem nebo RTC).

Příklad

- Panel s displejem, každou sekundu se snímají hodnoty, ty je třeba přepočítat a zobrazit na displeji.
- Snímání, výpočet a zobrazení trvá 16000 taktů.
- Co je výhodnější z hlediska spotřeby:
  - nechat běžet MCU na 16KHz (spotřeba 22µA) nebo
  - provést „akci“ na 25MHz (spotřeba 11mA) a pak usnout (spotřeba 370nA)?

Příklad

Jádro na 16 kHz

16000 taktů na 16kHz zabere celou sekundu

Spotřeba:

22µA x 3V x 1s = 66µWs

Jádro na 25 MHz

16000 taktů se provede za 0,64ms

11mA x 3V x 0,64ms = 21,12µWs

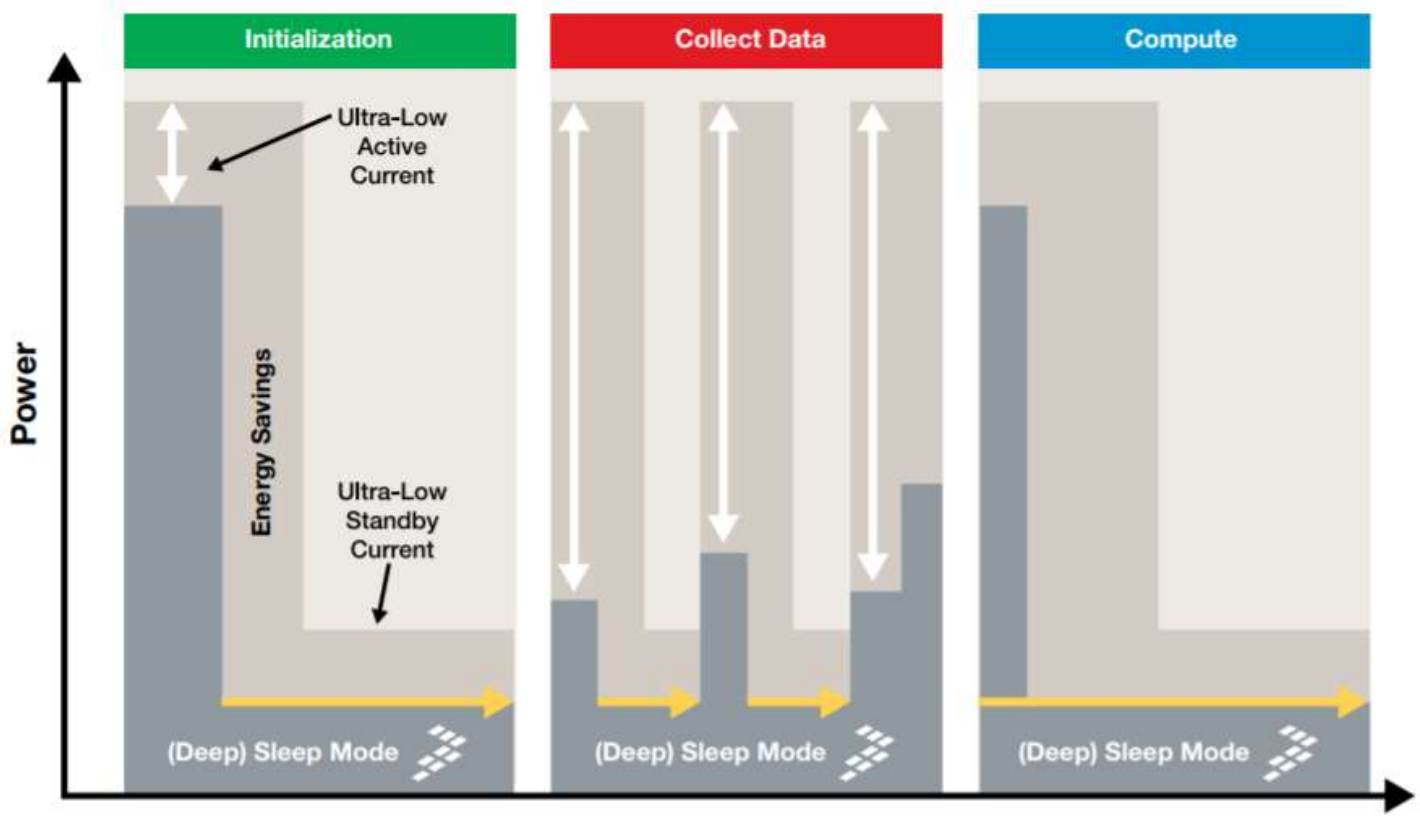
zbytek sekundy (0,99936s) se spí

370nA x 3V x 0,99936 = 1,11µWs

Spotřeba celkem 22,23µWs

Povšimněte si, že jsou situace, kdy paradoxně rychlejší běh jádra může vést k úsporám!

Typická low-power aplikace



Energie = spotřeba x čas ⇨ vše je třeba udělat úsporně a rychle, pak co nejvíce uspat a spát co nejdéle to jde.

Spotřeba přístupu do paměti

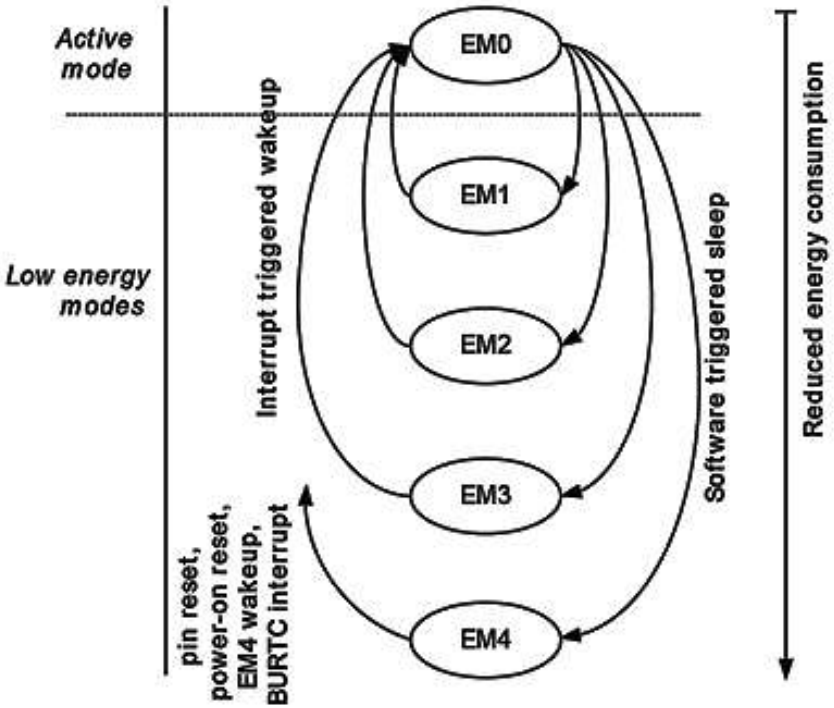
- Různé druhy pamětí mají různou spotřebu, je-li do nich přistupováno.
- Za normálních okolností jsou rozdíly zanedbatelné, ale při velmi malém příkonu jádra začíná i spotřeba paměti hrát roli:

Operating Mode	MCF51QE128	MC9S08QE128
Run Mode @ 2 MHz CPU/1 MHz bus	2 mA	1 mA
Run Mode @ 50 MHz CPU/25 MHz bus	27 mA	11 mA
Lower Power Run Mode @ 32 kHz CPU/16 kHz bus	50 µA (Flash) 19 µA (RAM)	22 µA (Flash) 9 µA (RAM)
Stop 2 — Lowest power mode; partial power down of circuits	370 nA	370 nA
Stop 3 — Internal circuits loosely regulated; external oscillator active	1070 nA	1000 nA
Stop 3 — Wake up time	6 µs	6 µs

Režimy spánku

Bývá jich typicky více.
Čím „hlubší“ spánek,
tím menší spotřeba,
ale také delší a
složitější probuzení!

Např. pokud se vypne i generátor hodin, trvá pak chvíli, než opět začne produkovat stabilní hodinový signál (regulační smyčka PLL).
Mohou to být třeba jednotky µs.



Režimy spánku

Z nejhlubšího spánku se MCU probouzí velmi dlouho (třeba i stovky µs) a zpravidla je jedinou možností probuzení RESET (vypíná se třeba i RAM a tak je třeba vše stejně znovu nastavit/obnovit).

Některé MCU nabízí tzv. „Backup Mode“ jako kompromis mezi nízkou spotřebou hlubokého spánku a možností rychlejšího probuzení – malá část RAM a důležité registry jsou drženy pod napětím.

Režimy činnosti KL05

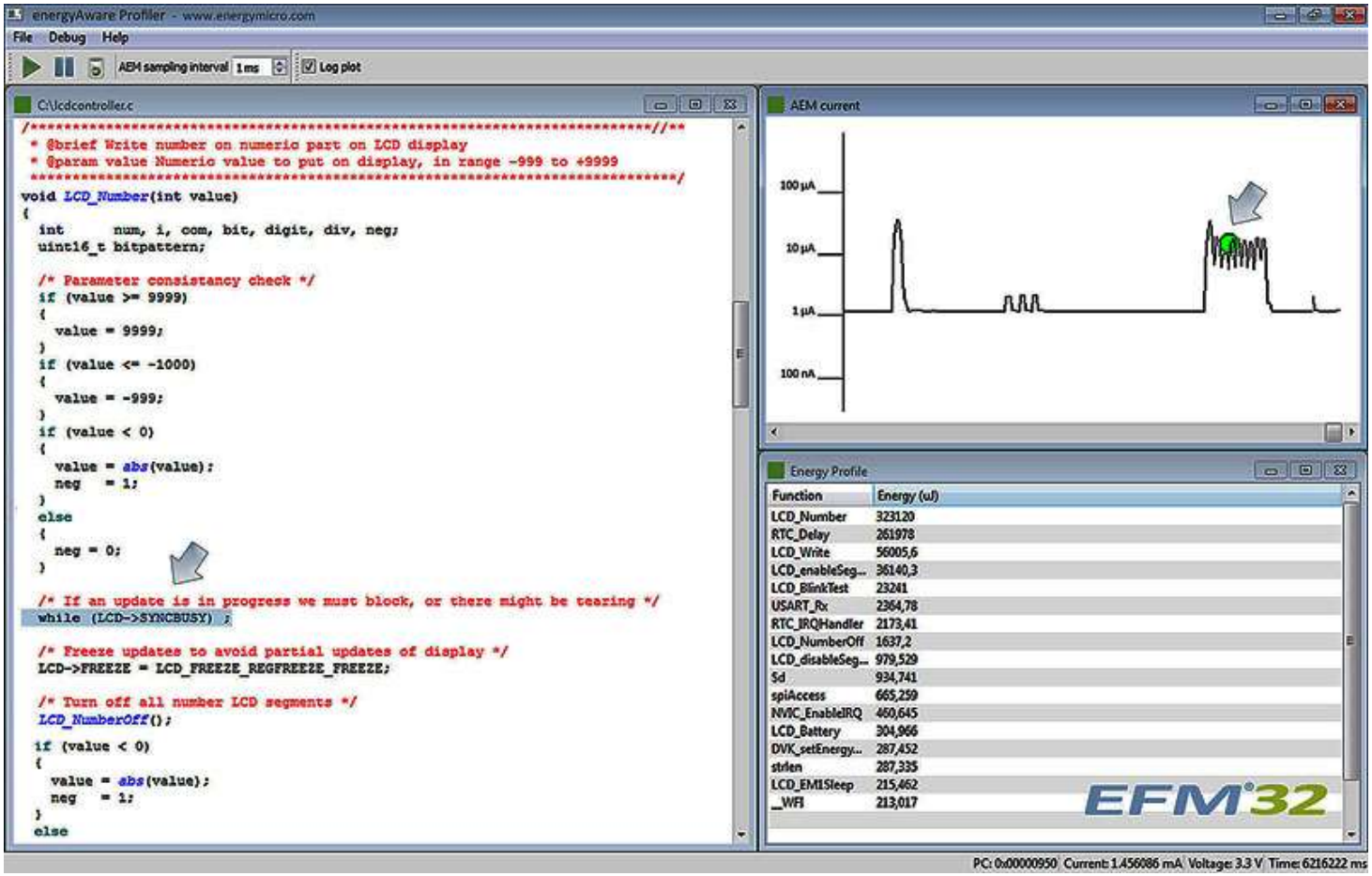
- RUN – vše plně běží
- VLPR (Very Low Power Run) – snížená frekvence hodin
- Wait – jádro ve Sleep režimu, reaguje na přerušení, periferie běží,
- Stop – jádro v DeepSleep režimu, periferie zastaveny,
- VLPW (Very Low Power Wait) – jako Wait, ale s nižší frekvencí hodin,
- WLPS (Very Low Power Stop) – jako Stop, ale snížena frekvence hodin, RAM drží,
- LLS (Low Leakage Stop) – jako WLPS, většina periferií v „retention“ režimu,
- VLLS3 (Very Low Leakage Stop) – řada modulů vypnuta, RAM „drží“, GPIO zachovávají hodnoty.
- VLLS1 – RAM vypnuta, GPIO „drží“ hodnoty,
- VLLS0 – nejhlubší režim spánku, pouze GPIO „drží“.

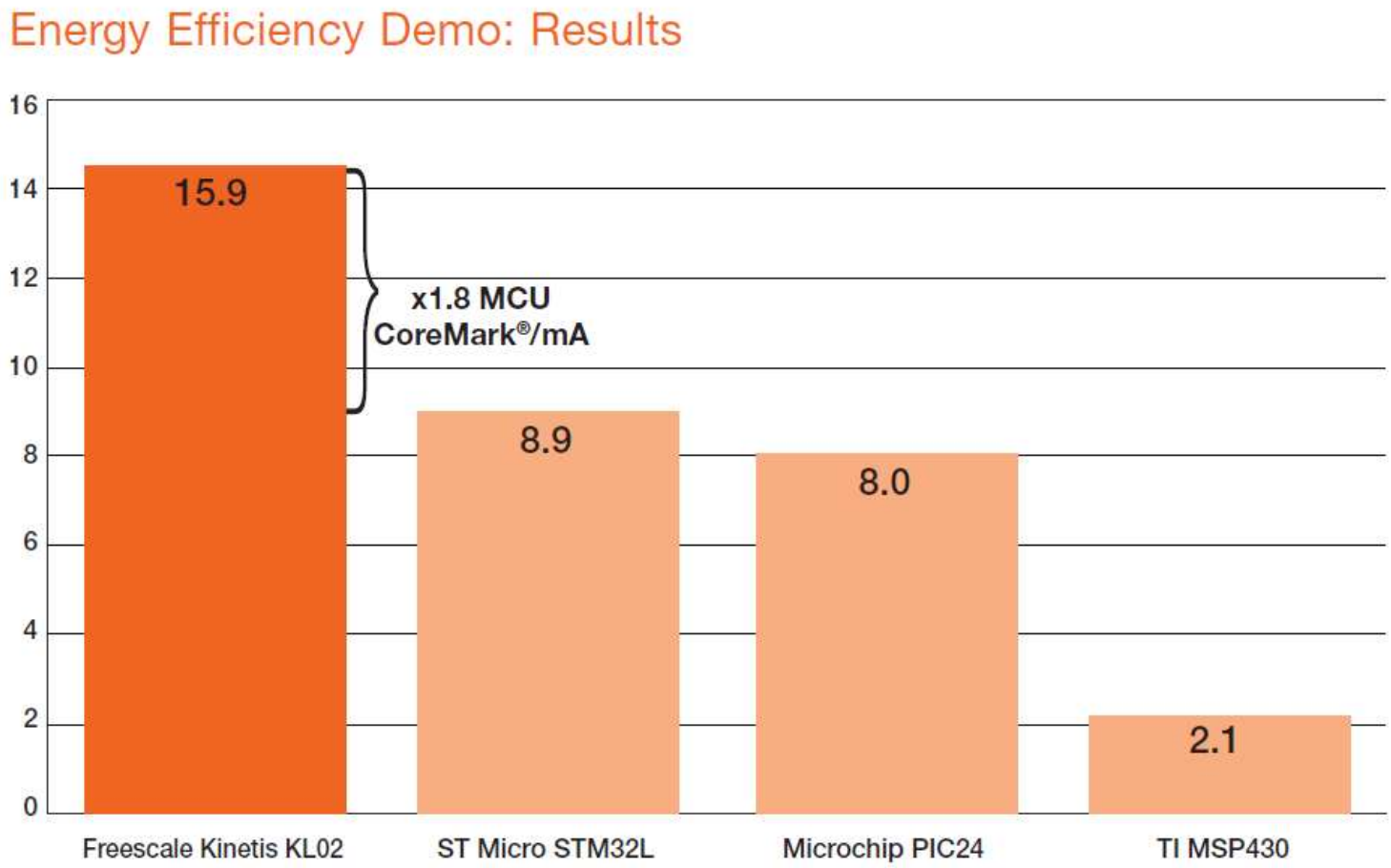
Kinetis L Series MCUs: 10 Flexible Power Modes

Kinetis Power Modes	Recovery Time	KL02 Measured Idd @ 3 V and 25 C
RUN	–	75 uA/MHz*
VLPR	–	36 uA/MHz**
WAIT	–	3.2 mA @ 48 MHz
VLPW	–	93.4 uA @ 4 MHz
STOP	4 us	255 uA
VLPS	4 us	1.9 uA
LLS[a]	N/A	N/A
VLLS3	42 us	1.2 uA
VLLS1	93 us	600 nA
VLLS0	95 us	161 nA/346 nA

* Compute Operation enabled: 3.6 mA @ 48 MHz core/24 MHz bus
** Compute Operation enabled: 144 uA @ 4 MHz core/1 MHz bus
[a] Available on select Kinetis L series devices

Podpora v IDE - příklad





CoreMark je syntetický benchmark pro měření výkonosti CPU pro vestavné systémy. Graf ukazuje, kolik iterací benchmarku zvládne MCU vztaženo na spotřebu 1mA.

Za 1s zvládne KL02 106 iterací, STM32L 93 iterací, PIC 29 iterací a MSP430 16 iterací.

Periferie na čipu

- Lze-li nějakou (zpravidla jednoduchou a rutinní) činnost svěřit některé periferii na čipu, bývá to zpravidla energeticky úspornější řešení nežli to nechat dělat software (jádro).
- Zvláště, lze-li přitom jádro uspat – nechat spát a nebudit.
- Například:
  - DMA pro přenos (odvysílání/přijetí) souboru dat,
  - modul pro záchyt asynchronní události („přerušení“, čítač v režimu „input capture“),
  - generování periodických výstupů čítačem, RTC modulem,
- Ideální je propojit periferie pomocí hw „triggerů“ a nechat je udělat co nejvíce práce samotné.

Energy Harvesting

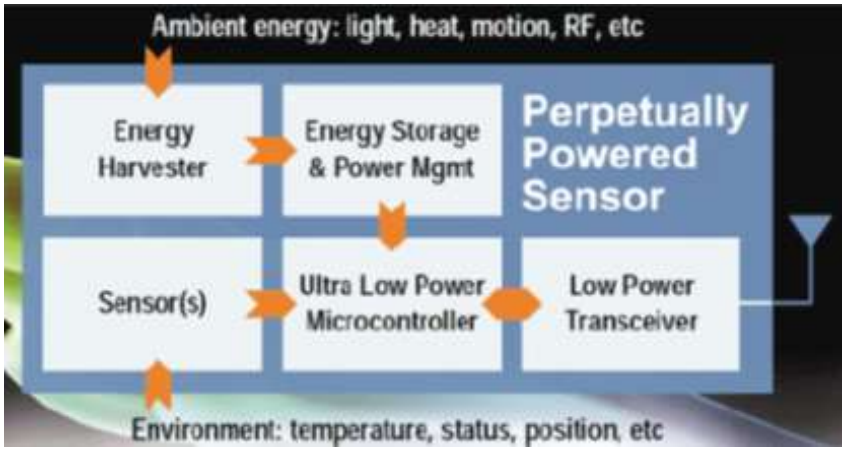
- Možnost prodloužení životnosti baterie až k ideálu energetické soběstačnosti systému.
- Další důvod, proč hledat úspory ve spotřebě MCU.

Energy Harvesting

= získávání energie z okolí (ze zdrojů, které energii v nějaké formě vyzařují):

- světlo,
- teplo,
- vibrace,
- elektromagnetické vlny z vysílačů, …

Typická architektura vestavěného systému, využívajícího Energy Harvesting:

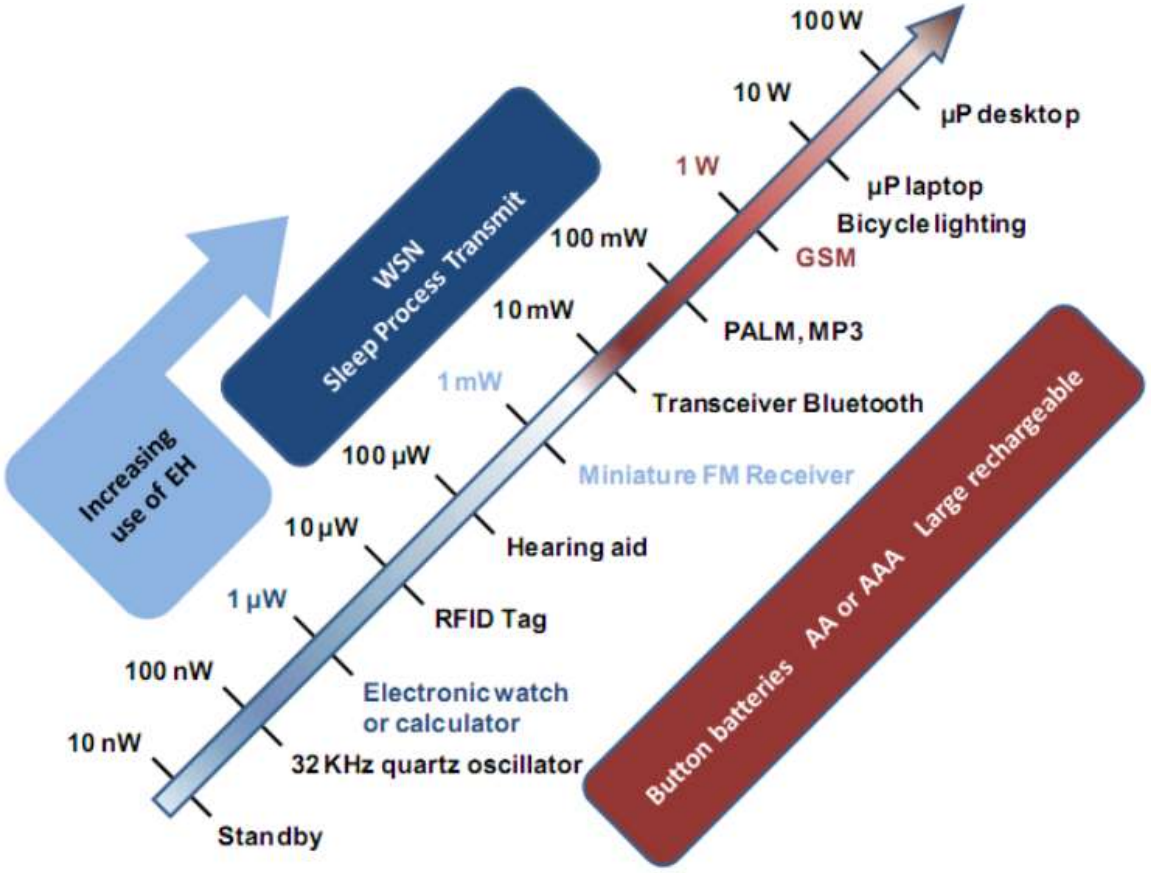


Jakou energii lze získat?

Hustota energie v našem okolí nebývá velká. Zpravidla se tedy sbírá a akumuluje, aby se jednou za čas spustil výpočet/odeslání dat.

Energy Source	Harvested Power
Vibration/Motion	
Human	4 μW/cm²
Industry	100 μW/cm²
Temperature Difference	
Human	25 μW/cm²
Industry	1–10 mW/cm²
Light	
Indoor	10 μW/cm²
Outdoor	10 mW/cm²
RF	
GSM	0.1 μW/cm²
WiFi	0.001 μW/cm²

Kolik energie je třeba?



Source IDTechEx report "Energy Harvesting and Storage for Electronic Devices 2009-2019".