Čítače a časovače
v mikrokontrolérech

Mikroprocesorové a vestavěné systémy (IMP)

Richard Růžička

Fakulta informačních technologií VUT v Brně

O co jde?

Příklad: je třeba počkat určitý čas

(než se něco stane, než nastane správná chvíle, …).

```
#define SCALE   15501    // System-dependent time scaling factor

void WaitMS(unsigned long ms)
{ unsigned long counter = ms * SCALE; // Compute how long to wait
  while(counter > 0) { counter--;}    // Waste time
}
```

Docela obvyklá úloha ve vestavných systémech.

► Mikrokontrolér musí něco udělat, až je na to ten správný čas – když se to hodí okolnímu světu, ne když se to hodí jemu.

Čekání, až uplyne určitá doba

```
#define SCALE   15501    // System-dependent time scaling factor

void WaitMS(unsigned long ms)
{ unsigned long counter = ms * SCALE; // Compute how long to wait
  while(counter > 0) { counter--;}    // Waste time
}
```

- **Pěkné, ale:**
  - **různé verze kompilátoru a různé úrovně optimalizace kódu** mohou významně změnit dobu trvání této smyčky,
  - **změny na hw úrovni také mohou změnit dobu vykonání** - např. se také může měnit frekvence hodin kvůli spotřebě, teplotě, kód se může vykonávat v různě rychlé paměti, v cache atd.
  - **přijde-li během vykonávání smyčky přerušení**, celé se to zdrží,
  - **zbytečně se maří čas procesoru a energie**.

Řešení?

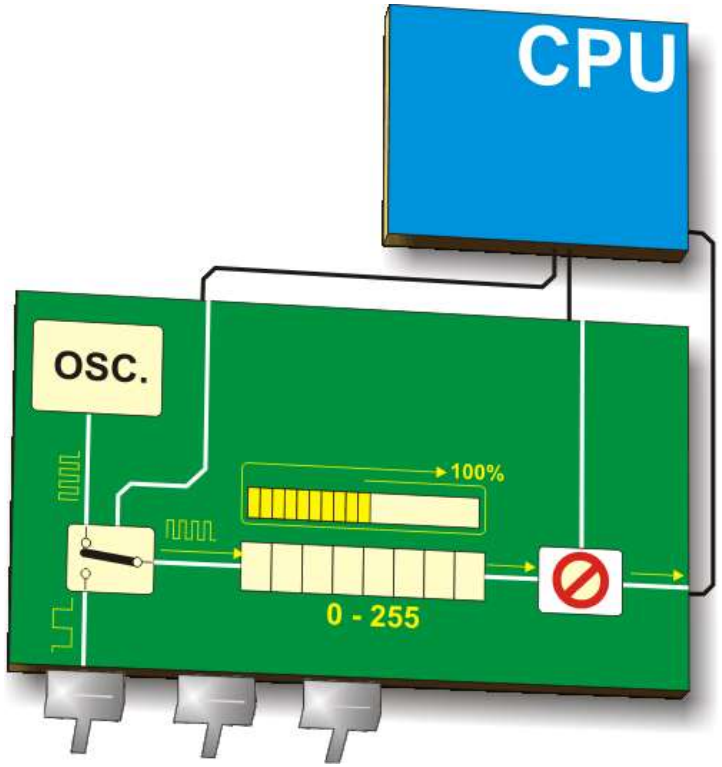
Je-li časování ve vestavných systémech tak důležité, pak se vyplatí mít

- **hardwarový modul na počítání času!**
  - **běží nezávisle na CPU** (vlastně paralelně s procesorem, který se během čekání na událost může věnovat jiné smysluplné činnosti), nikdo jej nepřerušuje,
  - **může mít svůj zdroj hodin**, nezávislý na procesoru,
  - lze jej **zapínat a vypínat podle potřeby**,
  - je **optimalizován na tuto činnost** (lepší spotřeba).

Časovač v MCU

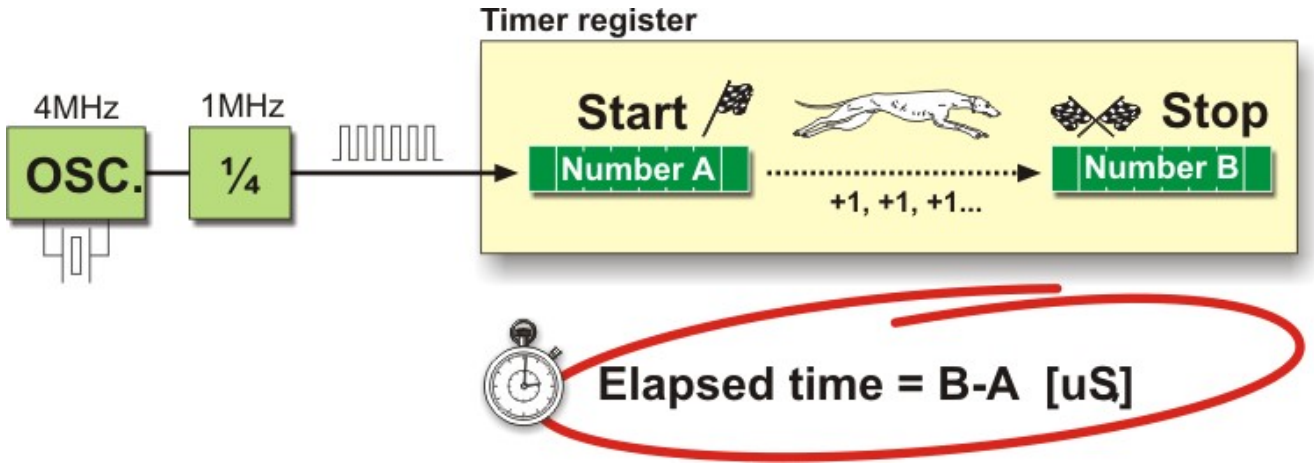
- A proto dnes **v každém mikrokontroléru** najdeme **ne jeden, ale hned několik** hw zařízení k měření času, tzv. **časovače** (timers).

Jádrem je vždy **čítač**, čítající hodinové pulsy.



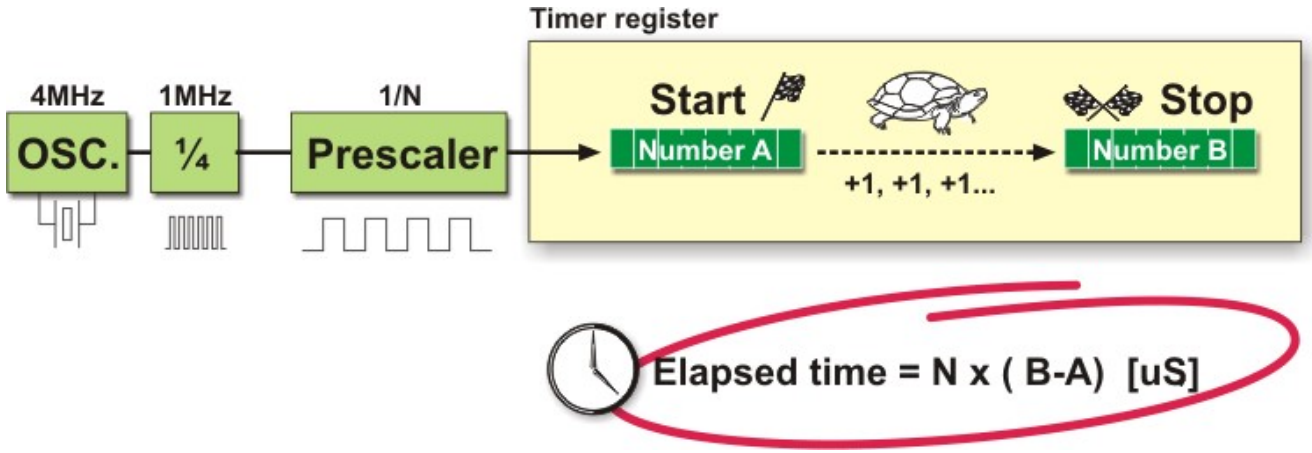
Jak to tedy funguje?

Jádrem je vždy čítač, počítající „nějaké“ pulsy – nejčastěji pulsy hodinového signálu.

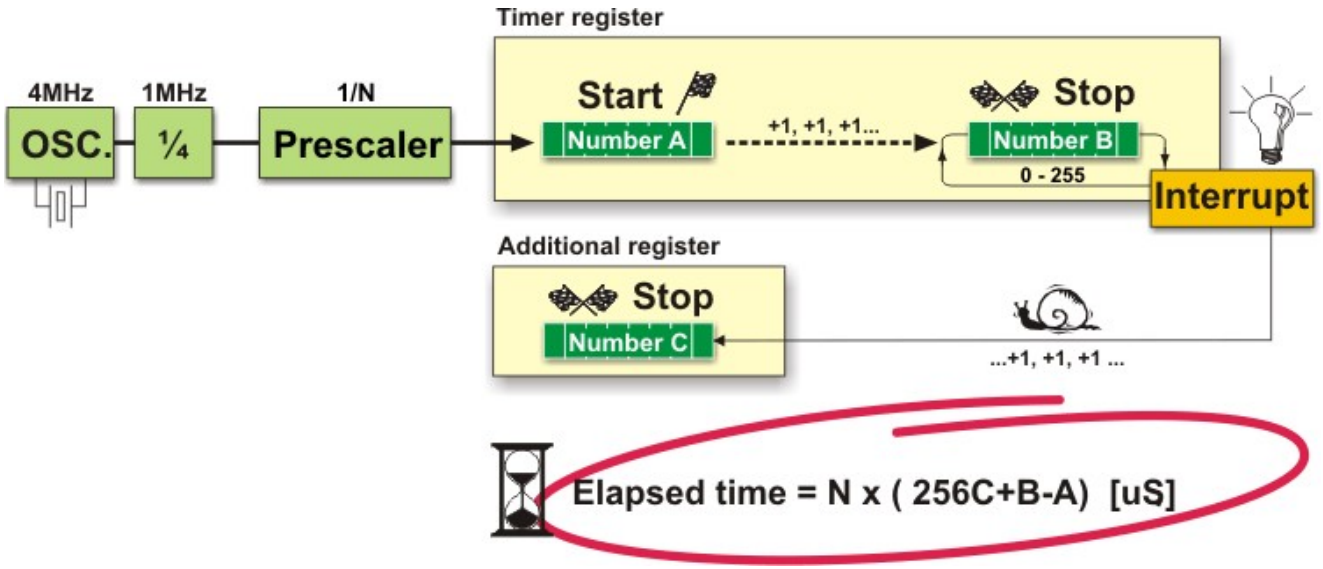


Předdělič

… je-li třeba odměřit delší čas (počítání „normálních“ hodinových pulsů by vedlo k rychlému přetečení registru)



Využití přerušení od čítače



… pro odměření opravdu „dlouhých“ časových intervalů - počítá se několikeré přetečení čítače.

Na co lze využít časovače ve vestavném systému:

- jednorázově přesné odměření časového intervalu – odezva v sw nebo hw za konkrétní dobu,
- periodické generování události s přesně nastavenou periodou,
  - událost v software,
  - generování hw výstupu na pinu,
  - spouštění modulu v mikrokontroléru,
- na pozadí běžící přesné hodiny (je možné se kdykoli podívat, „jaký je čas“.

Časovače v MCU Kinetis

TPM (Timer/PWM Module)

- univerzální časovač, propojení na piny (input capture, output compare, PWM), generuje přerušení.

LPTMR (Low-power Timer)

- čítač/časovač pro všechny úsporné režimy, slouží k probuzení MCU a generování přerušení, může spouštět ostatní moduly.

PIT (Periodic Interrupt Timer)

- periodicky generuje přerušení

RTC (Real-time Clock)

- produkuje 1 Hz signál, který generuje přerušení, může probudit MCU, běží z nezávislého zdroje hodin.

Tolik různých modulů podobného účelu je nejlepším důkazem, jak je to důležité.

Proč tolik různých časovačů na jednom MCU?

- Jde o hledání kompromisu mezi funkcionalitou a spotřebou.
- HW moduly často pracují na svém úkolu, zatímco zbytek MCU spí – aby se ušetřila energie.
  - pro jednoduché čítání - jen aby se MCU „probudil včas“ není třeba krmit příliš složitou strukturu,
  - na druhou stranu, i během spánku může čítač zařídit některé další věci, aniž by se kvůli tomu musel opakovaně budit a startovat CPU – zde se hodí komplexnější modul časovače, navázaný na další moduly.

Příklad:

spotřeba RTC je jen asi 0,35 µA, spotřeba TPM nejméně 86 µA

spotřeba běžícího MCU 200 – 6000 µA

Modul RTC

Real Time Clock v MCU Kinetis KL05

K čemu modul RTC?

- Některé funkce vestavěných systémů je třeba provádět periodicky (nejsou řízeny událostmi, ale mají daný „svůj čas“*, kdy mají být vykonány):
  - periodické vzorkování nějakého signálu,
  - měření reálného času atd.
- Softwarově to sice lze poměrně snadno realizovat pomocí smyčky s vhodným zdržením nebo s podporou časovače, má to však i svoje nevýhody (zbytečná zaneprázdněnost CPU, závislost na momentálním nastavení hodinového signálu atd.
- Časovač (TPM) by bylo možno také užít, ale je to trochu složitý (a „žravý“) modul.
- Hardwarovou podporou pro to je (jednoduchý a úsporný) modul RTC.

*čas měřený modulem RTC je v řádech, které jsou smysluplné v reálném světě - žádné mikrosekundy, ale sekundy.

Co umí modul RTC

- Počítat sekundy 32bitovým čítačem! 2^{32} sekund je více než 136 let!
  - Může vyvolávat přerušení,
  - může vzbudit MCU z režimu spánku
    - každou sekundu,
    - po uplynutí určitého počtu sekund (nastavení „budíku“).
  - Sekundové pulsy lze vyvést na pin.
  - Může být nezávislý na zdroji hodin pro zbytek MCU.
  - Resetuje se jen při zapnutí napájení.
  - Potřebuje k tomu jen velmi málo energie.

Registry modulu RTC

Registr, v němž si přečteme, kolik sekund uplynulo

32 768 Hz

RTC memory map

...lute ...ess ...hex)	Register name	Width (in bits)	Access	Reset value
4003_D000	RTC Time Seconds Register (RTC_TSR)	32	R/W	0000_0000h
4003_D004	RTC Time Prescaler Register (RTC_TPR)	32	R/W	0000_0000h
4003_D008	RTC Time Alarm Register (RTC_TAR)	32	R/W	0000_0000h
4003_D00C	RTC Time Compensation Register (RTC_TCR)	...	R/W	0000_0000h
4003_D010	RTC Control Register (RTC_CR)	...	W	0000_0000h
4003_D014	RTC Status Register (RTC_SR)	...	W	0000_0001h
4003_D018	RTC Lock Register (RTC_LR)	...	W	0000_00FFh
4003_D01C	RTC Interrupt Enable Register (RTC_IER)	32	R/W	0000_0007h

Registr, kam se nastaví, po kolika sekundách se má něco stát.

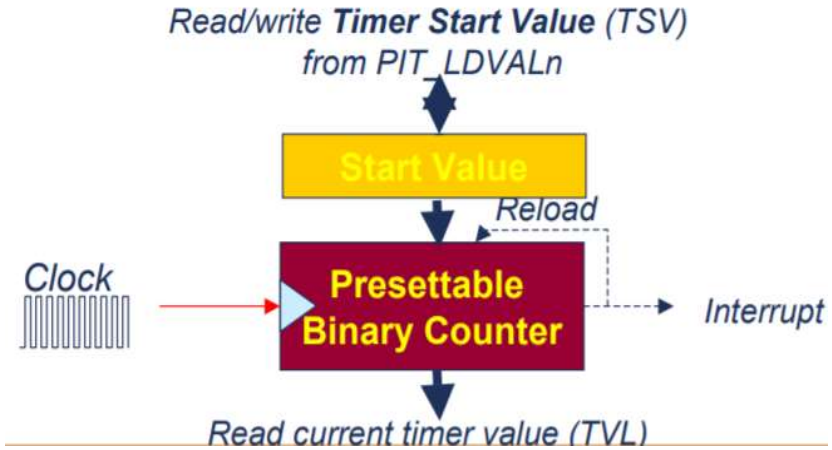
Registry pro konfiguraci a sledování stavu modulu

Modul PIT

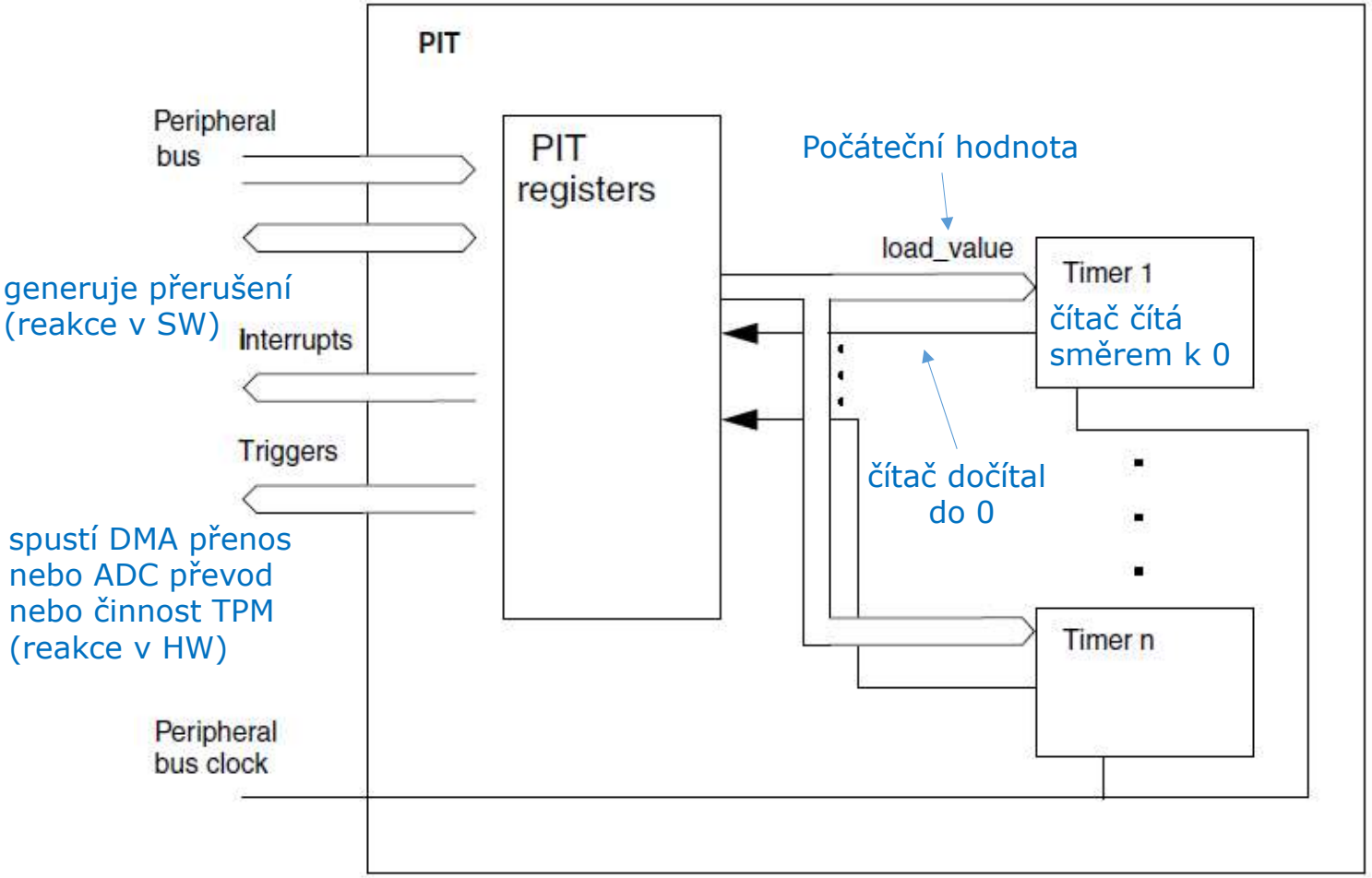
Periodic Interrupt Timer v MCU Kinetis KL05

K čemu modul PIT?

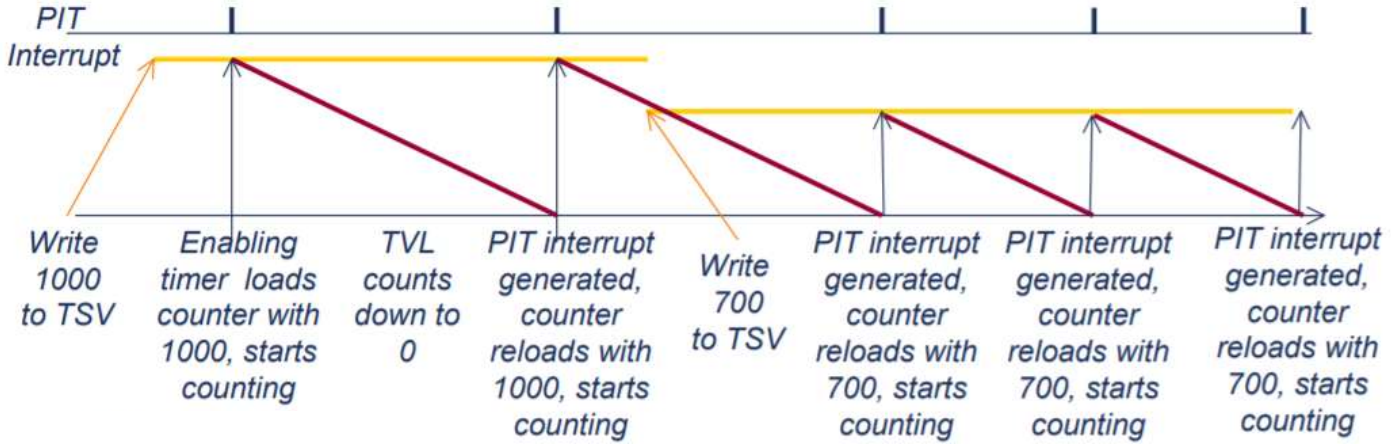
- Generuje periodicky přerušení (spouští akci v SW) nebo spustí „trigger“ pro ADC/TPM/DMA (spouští akci v HW).
  - Periodu lze nastavit počáteční hodnotou - od ní čítač každým tikem hodin odečítá, až dojde k 0 – to je „interrupt“ (nebo „trigger“),
  - znovu se nahraje počáteční hodnota.



Struktura a funkce modulu



Činnost PIT a registry



PIT_LDVALn – hodnota TSV (Timer Start Value)
PIT_TVALn – hodnota TVL (Current Timer Value)

PIT_MCR – Main Control Register – zapnutí PIT
PIT_TCTRLn – zapnutí kanálu, povolení přerušení
PIT_TFLGn – příznak přerušení

Příklad

Hodnota TSV (registr PIT_LDVALn) pro přerušení každých T sekund:

$$\mathbf{TSV = round(T*f_{count} - 1)}$$

Pro přerušení každých 137,41ms:

$$\mathbf{TSV = 137.41\ ms * 24\ MHz - 1 = 3297839}$$

LPTMR

Low-Power Timer v MCU Kinetis KL05

LPTMR

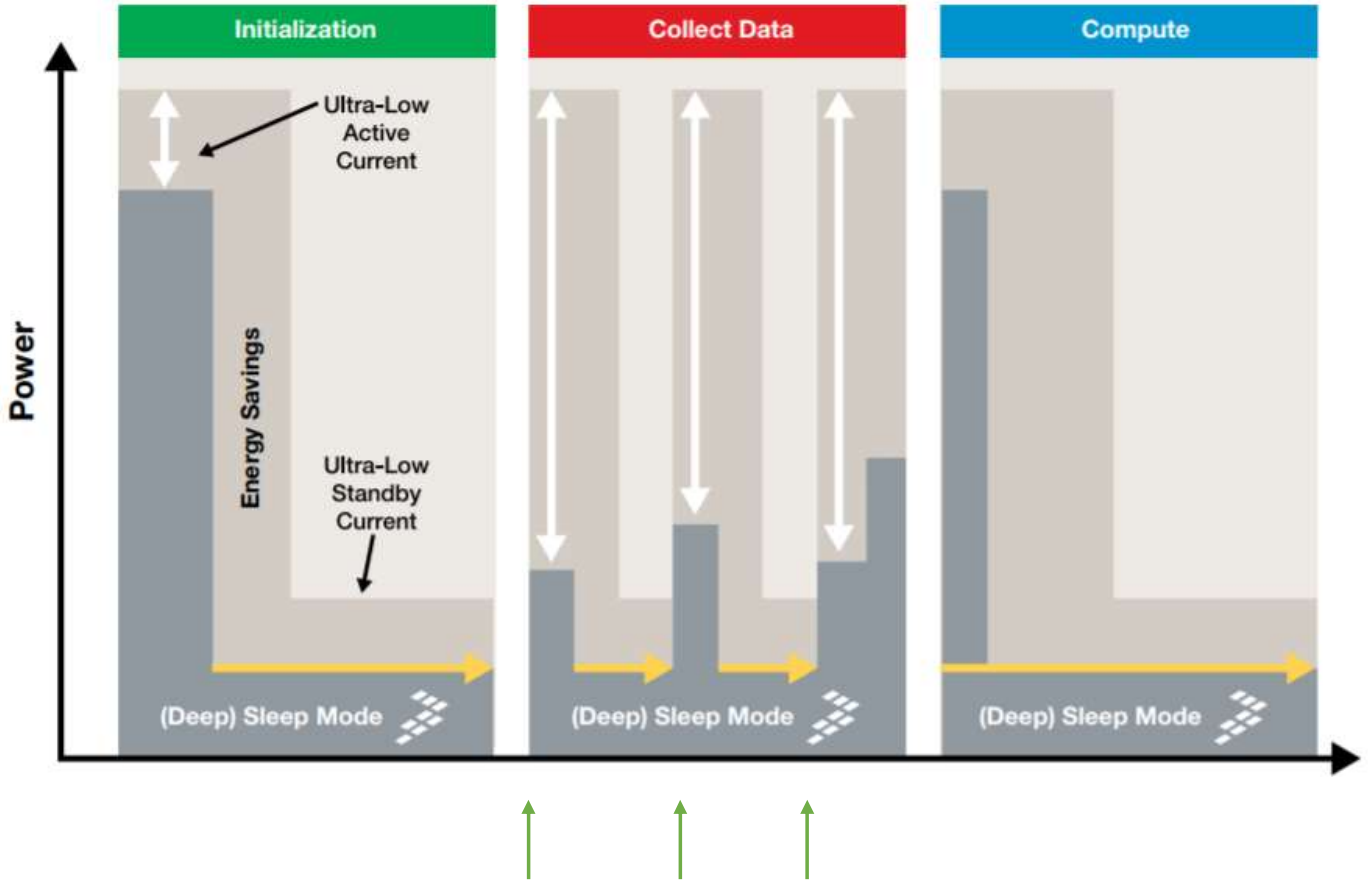
Časovač, který umožňuje

- **generovat přerušení v nastaveném čase** - čítá pulsy hodinového signálu, zpomalené předděličkou, přerušení je generováno v okamžiku dosažení přednastavené hodnoty,
- **čítat pulsy** nějakého **vnějšího signálu** (přivedeného přes vestavěný filtr).

V okamžiku dosažení přednastavené hodnoty dokáže **probudit MCU**.

… proto důraz na low-power: MCU spí (šetří energii) a probudí se, „až je čas“

Typická low-power aplikace MCU



MCU se musí nějak budit ve vhodných chvílích.

LPTMR

Jádro tvoří 16bitový čítač:

LPTMRx_CNR – aktuální obsah vlastního čítače.

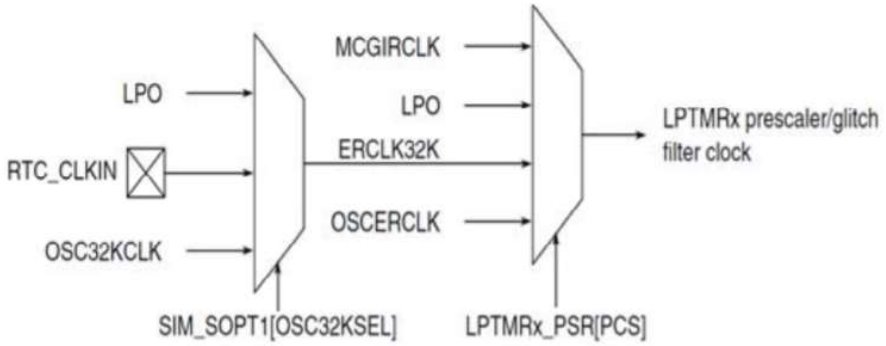
LPTMRx_CMR – hodnota k porovnání,
pokud CNR dosáhne této hodnoty,
generuje se přerušení.

LPTMRx_CSR – zapnutí, nastavení režimu, povolení přerušení,
příznak přerušení.

LPTMRx_PSR – nastavení předděliče/vstupního filtru.

Vstupy a výstupy LPTMR

Vstup v režimu časovače může být z řady zdrojů hodin v MCU:



Vstup v režimu čítače pulsů může být buď z pinu nebo z výstupu CMP.

Výstup:

- přerušení
- AD převodník (spuštění převodu)
- CMP (spouštění porovnání)
- TPM (spouštění čítání)
- TSI (touch sensing – kapacitní senzor dotyku)

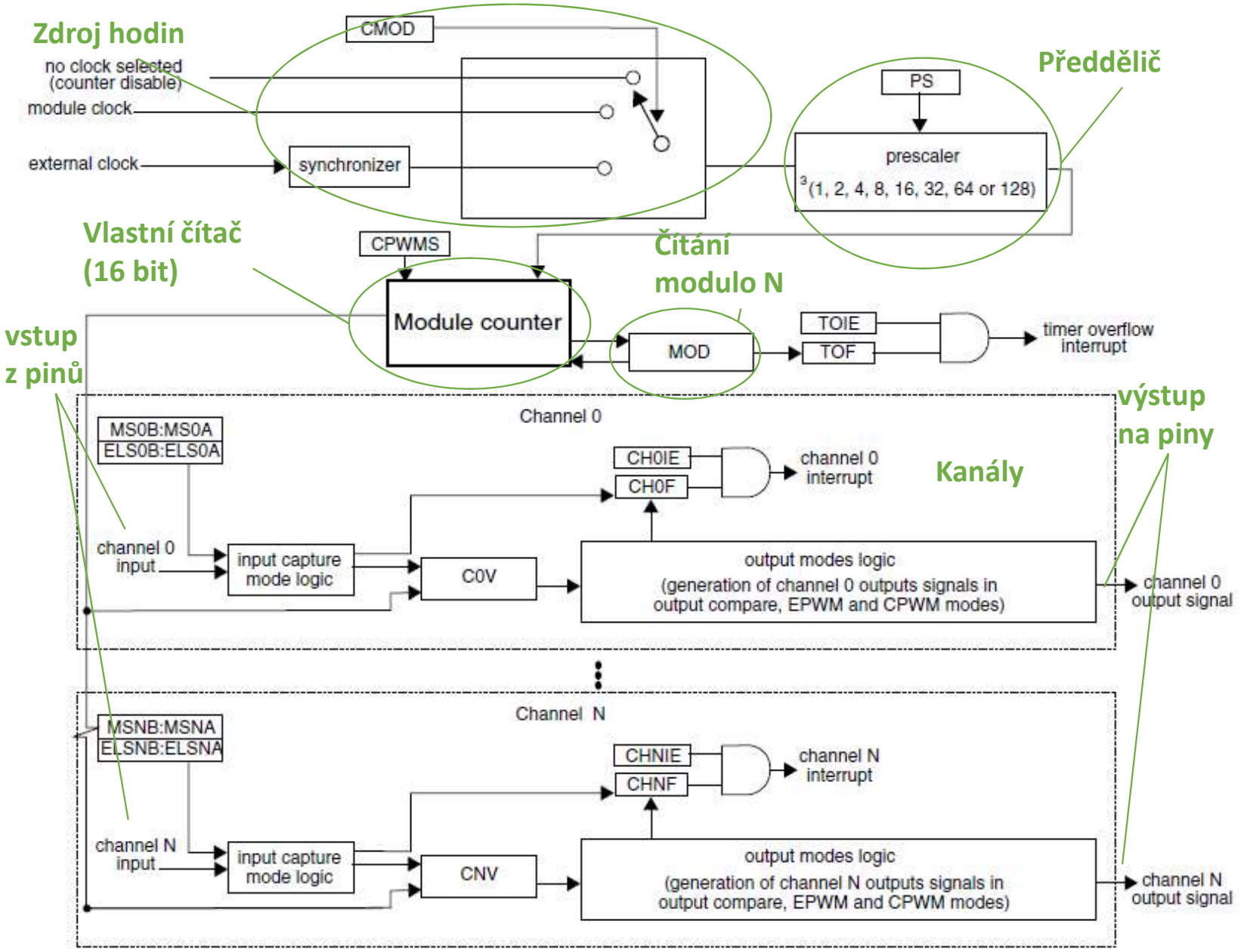
TPM

Timer/PWM Module v MCU Kinetis KL05

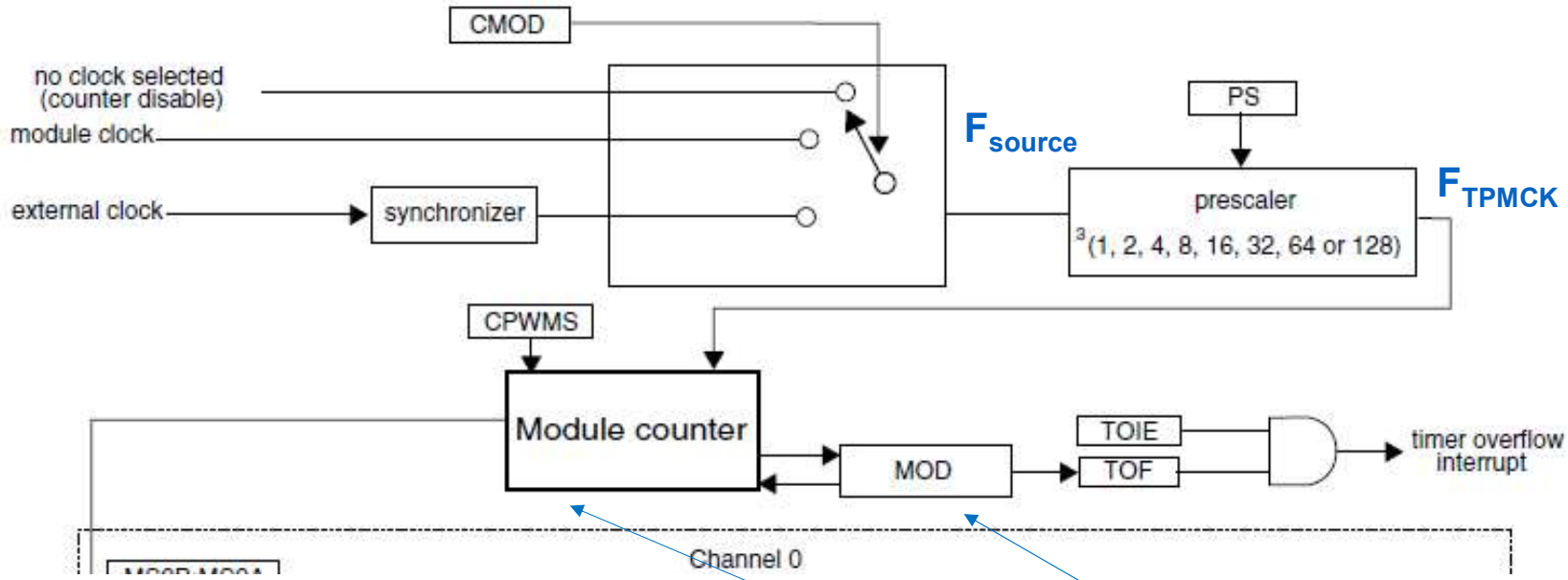
Typický modul čítače/časovače

v moderním MCU sestává z

- **Zdroj hodinového signálu** (výběr možností podle aplikace, požadavku na spotřebu, přesnost atd.)
- **Předdělič frekvence hodin** (aby bylo možno nastavit rychlost čítání)
- **Vlastní čítač** (registr počítající pulsy) s nastavením funkce „modulo.“
- Jeden čítač (registr, počítající pulsy) může sloužit pro **více kanálů** připojení na vnější svět, u nichž lze nastavit různé režimy činnosti, např:
  - **Zachycení hrany** signálu (změny úrovně, značící nějakou událost) na nějakém pinu (angl. input capture),
  - **Generování signálu** na nějakém pinu – změna výstupu v okamžiku, kdy čítač nabude určité hodnoty (angl. Output compare),
  - Speciální případ předchozího – generování pulsně-šířkové modulace (PWM).


Zdroj hodin
CMOD
no clock selected (counter disable)
module clock
external clock
synchronizer
Předdělič
PS
prescaler
3 (1, 2, 4, 8, 16, 32, 64 or 128)
Vlastní čítač (16 bit)
CPWMS
Module counter
Čítání modulo N
MOD
TOIE
TOF
timer overflow interrupt
vstup z pinů
Channel 0
MS0B:MS0A
ELS0B:ELS0A
CH0IE
CH0F
channel 0 interrupt
Kanály
channel 0 input
input capture mode logic
C0V
output modes logic
(generation of channel 0 outputs signals in output compare, EPWM and CPWM modes)
channel 0 output signal
výstup na piny
Channel N
MSNB:MSNA
ELSNB:ELSNA
CHNIE
CHNF
channel N interrupt
channel N input
input capture mode logic
CNV
output modes logic
(generation of channel N outputs signals in output compare, EPWM and CPWM modes)
channel N output signal

Vlastní čítač a co s ním souvisí



Aktuální stav čítače je dostupný v registru TPMx_CNT.

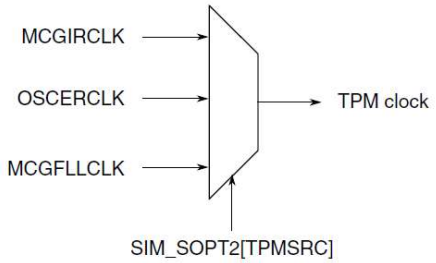
Modulo se nastaví v registru TPMx_MOD.

Bity CMOD, PS, CPWMS, TIOE, TOF jsou v registru TPMx_SC.

F_{TPMCK} je hodinový signál budící volně běžící čítač (Module counter)

Základní vztahy 1

- **$F_{TPMCK} = F_{source}$/ prescaler**
  - Kmitočet hodin pro čítač z několika zdrojů lze upravovat předděličkou
  - Zdrojem hodin je
    - interní hodinový signál



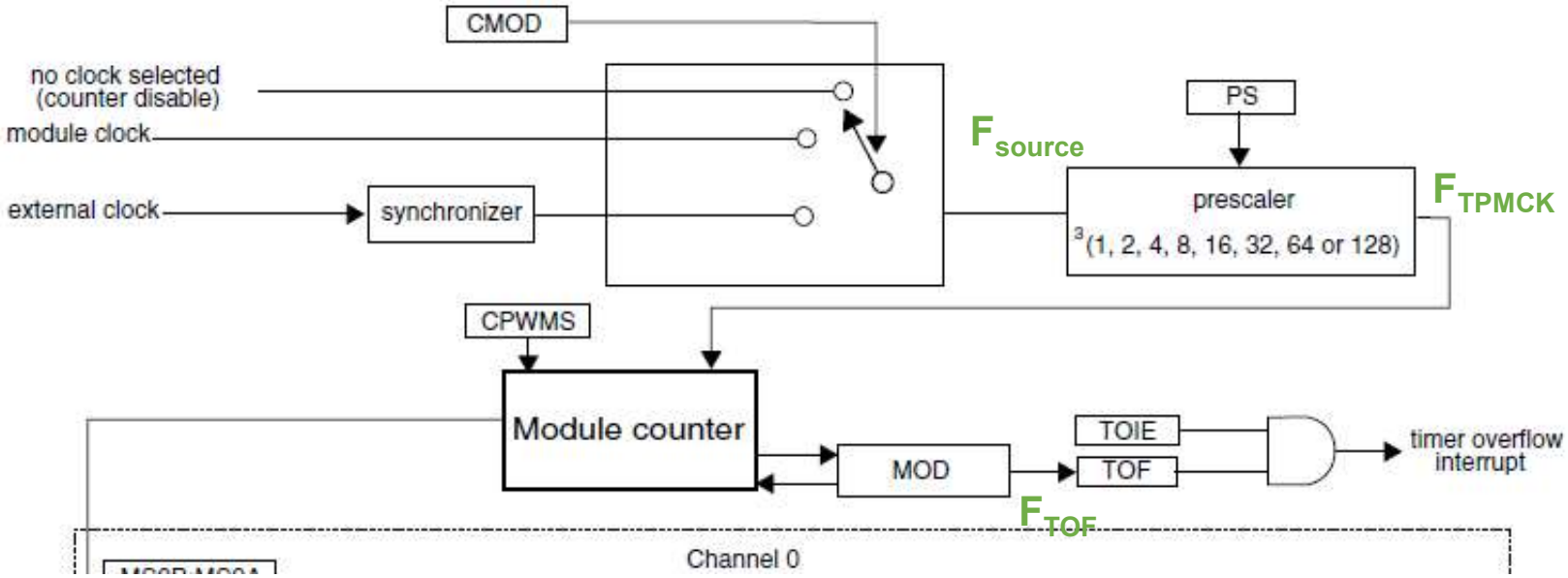
2-0 PS	Prescaler Factor
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

    - externí signál TPMx_CLKIN0, TPMx_CLKIN1
      (vybírá se v registru SOPT4).

Základní vztahy 2

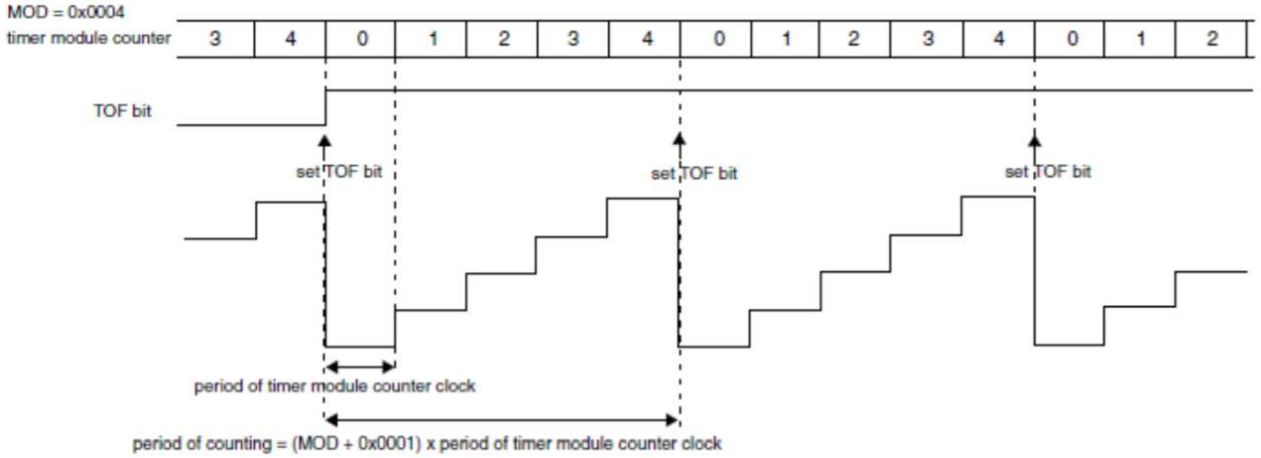
- **$F_{TOF} = F_{source}/ [prescaler*(MOD+1)]= F_{TPMCK}/ (MOD+1)$**

  Kmitočet přetečení čítače je dán nastavením předděličky a obsahem modulo registru MOD



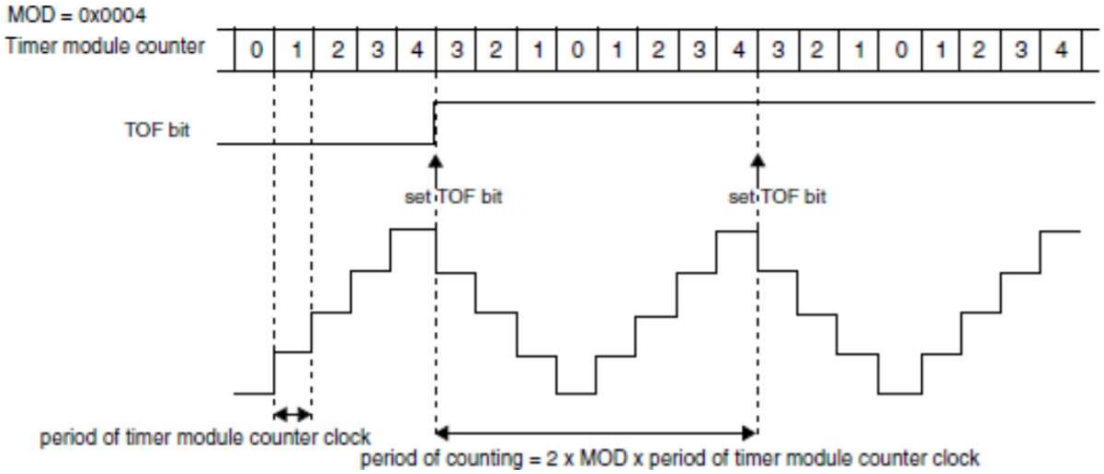
Čítač čítá pulsy

nahoru:



nebo

nahoru
a dolů:



Trigger

Čítač může být spuštěn nebo resetován „vnější“ událostí.

TPMx_CONF[TRGSEL]	Selected source
0000	External trigger pin input (EXTRG_IN)
0001	CMP0 output
0010	Reserved
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	Reserved
0111	Reserved
1000	TPM0 overflow
1001	TPM1 overflow
1010	Reserved
1011	Reserved
1100	RTC alarm
1101	RTC seconds
1110	LPTMR trigger
1111	Reserved

Ovládání čítače

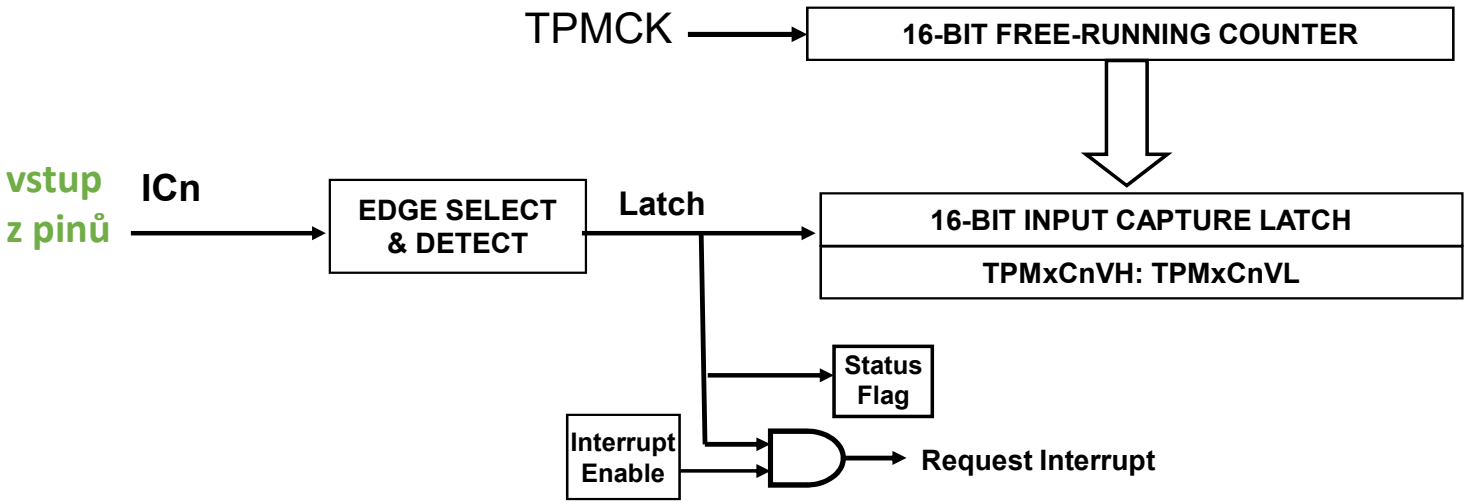
18 CROT	Counter Reload On Trigger When set, the LPTPM counter will reload with zero (and initialize PWM outputs to their default value) when a rising edge is detected on the selected trigger input. The trigger input is ignored if the LPTPM counter is paused during debug mode or doze mode. This field should only be changed when the LPTPM counter is disabled. 0 Counter is not reloaded due to a rising edge on the selected input trigger 1 Counter is reloaded when a rising edge is detected on the selected input trigger
17 CSOO	Counter Stop On Overflow When set, the LPTPM counter will stop incrementing once the counter equals the MOD value and incremented (this also sets the TOF). Reloading the counter with zero due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT set. This field should only be changed when the LPTPM counter is disabled. 0 LPTPM counter continues incrementing or decrementing after overflow 1 LPTPM counter stops incrementing or decrementing after overflow.
16 CSOT	Counter Start on Trigger When set, the LPTPM counter will not start incrementing after it is enabled until a rising edge on the selected trigger input is detected. If the LPTPM counter is stopped due to an overflow, a rising edge on the selected trigger input will also cause the LPTPM counter to start incrementing again. The trigger input is ignored if the LPTPM counter is paused during debug mode or doze mode. This field should only be changed when the LPTPM counter is disabled. 0 LPTPM counter starts to increment immediately, once it is enabled. 1 LPTPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow.

Čítač běží, ale co dál?

- Samo o sobě je to k ničemu, musí to jít nějak použít v aplikaci – provázat se systémem, do nějž vestavujeme MCU:
  - V okamžiku přetečení se generuje přerušení
  - ☞ TPM způsobí přesně načasovanou událost v software
  - Pomocí kanálů se TPM napojí na signály vně MCU
  - ☞ Měří se čas nějaké události v hardware (input capture)
  - Pomocí kanálů TPM generuje nějaký časově přesně vymezený signál
  - ☞ V přesně určených okamžicích se mění napětí výstupního pinu (output compare).

Mějme stále na paměti, že to celé běží paralelně s programem – stačí to správně nakonfigurovat a pak už to funguje samo!

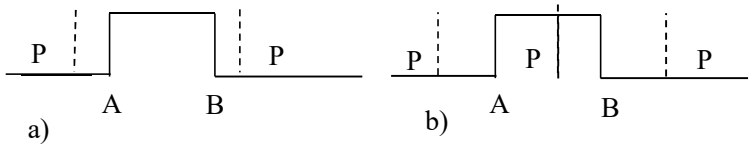
Input Capture



- Je-li detekována očekávaná změna úrovně na pinu (sestupná/vzestupná hrana), zapamatuje se momentální stav čítače (do registru kanálu).

Měření délky pulsu

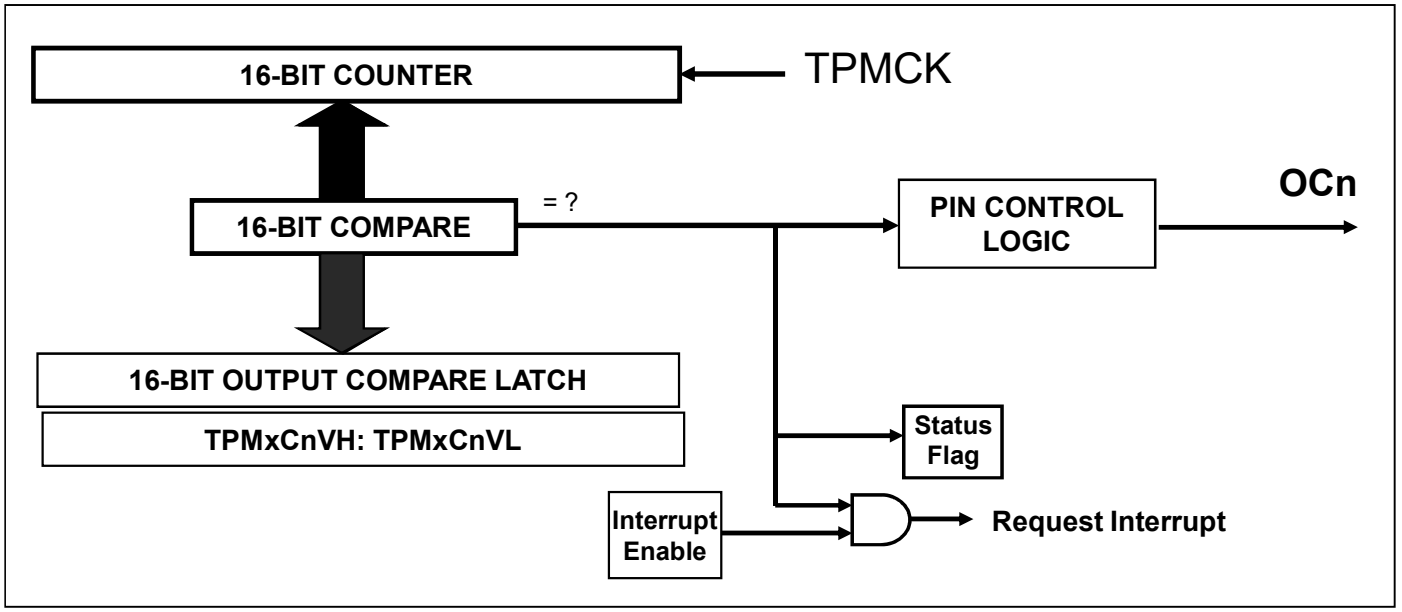
- Jednotku záchytu inicializujeme do režimu detekce obou hran a vždy po zjištění hrany uložíme zachycenou hodnotu z čítače do pomocné proměnné. Pokud máme již zjištěné dvě po sobě následující hodnoty čítače stanovíme jejich rozdíl, který udává délku impulsu v periodách budícího signálu čítače.



Délku impulsu můžeme vyjádřit vztahem:

- DP = K (B-A) za předpokladu, že v době AB nedošlo k přetečení čítače tedy A<B,viz případ a). K je rovno periodě signálu budícího čítač.
- Pokud časový okamžik P, reprezentující přetečení čítače, leží uvnitř intervalu AB, viz případ b), je třeba použít alternativní vztah:
- DP = K (A^D+B) kde A^D je dvojkový doplněk hodnoty A.
- Alternativním postupem pro měření délky impulsu je použití dvou samostatných jednotek pro záchyt hrany. Jedna jednotka se nastaví na záchyt nástupné hrany a druhá na záchyt sestupné hrany.

Output Compare



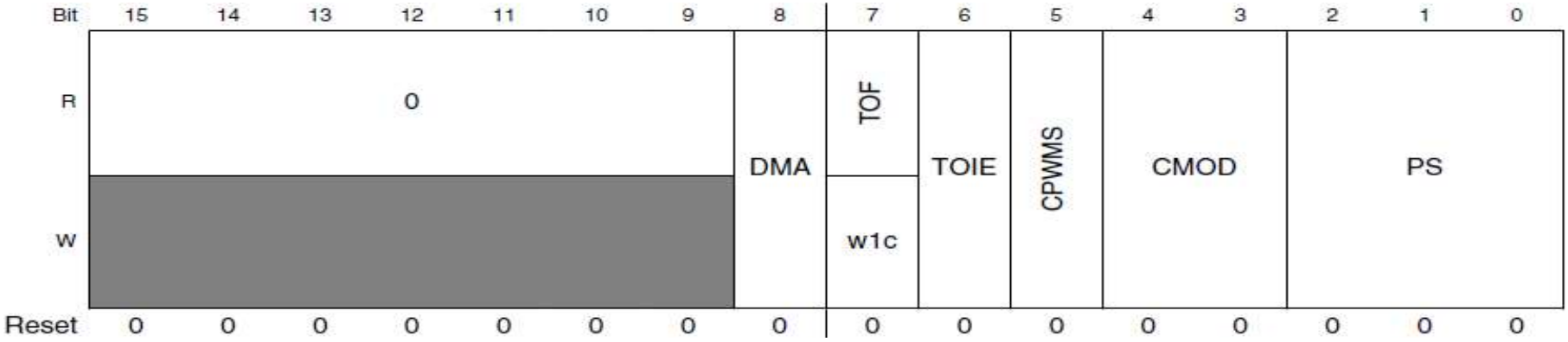
- Jakmile čítač při čítání dojde k určité hodnotě, je generována změna úrovně na výstupním pinu.

Registry TPM … v nich se vše nastavuje

- **TPMxSC** – stavový a řídící registr celého TPM – nastavení přerušení od přetečení čítače, výběr hodin, nastavení předděliče
- **TPMxCONF** – konfigurace spouštění/zastavení/resetu čítače
- **TPMxMOD** - nastavení cyklu čítače (čítání modulo N)
- **TPMxCNT** – vlastní čítač (lze jen číst, zápis čehokoliv čítač nuluje)
- **TPMxCnSC** - kanálový stavový a řídící registr – nastavení režimu kanálu (input capture, output compare, …, nastavení hrany/úrovně na pinu, nast. přerušení)
- **TPMxCnV** - kanálové komparační registry - registry kanálů čítače (čtení hodnoty IC, zápis pro OC)

Řídicí a stavový registr celého TPM

TPMxSC



TOF – příznak přetečení čítače

TOIE – maska příznaku přetečení - povolení přerušení

DMA – maska příznaku přetečení - povolení DMA přenosu startovaného přetečením

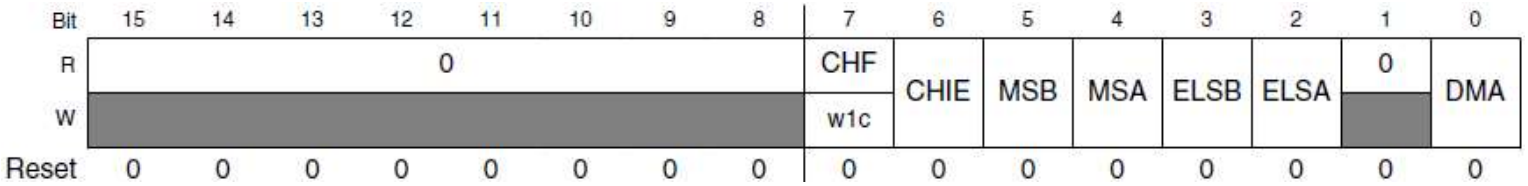
CPWMS –čítání jen nahoru nebo nahoru/dolů

CMOD – výběr zdroje hodin

PS[2:0] – nastavení dělícího poměru předděličky

Řídicí a stavový registr každého kanálu

TPMxCnSC



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHF – příznak shody čítače a obsahu registru kanálu nebo detekce vstupní hrany

CHIE – povolení přerušení při události, která nastaví CHF

DMA – povolení DMA přenosu při události, která nastaví CHF

MSB/A – nastavení režimu časovače: IC, OC, PWM v obou módech

ELSB/A – nastavení polarity vstupní hrany pro IC nebo výstupní úrovně v případě OC nebo PWM

Řídicí a stavový registr každého kanálu

TPMxCnSC

7	6	5	4	3	2	1	0
CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
w1c							
0	0	0	0	0	0	0	0

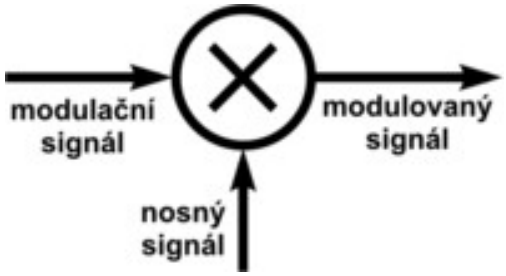
Toto je přepínač z registru TPMxSC celého TPM!

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01	00	Software compare	Pin not used for TPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		01		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		01		Low-true pulses (set Output on match-up, clear Output on match-down)

Pulsně šířková modulace (PWM)

- Co je **modulace**?

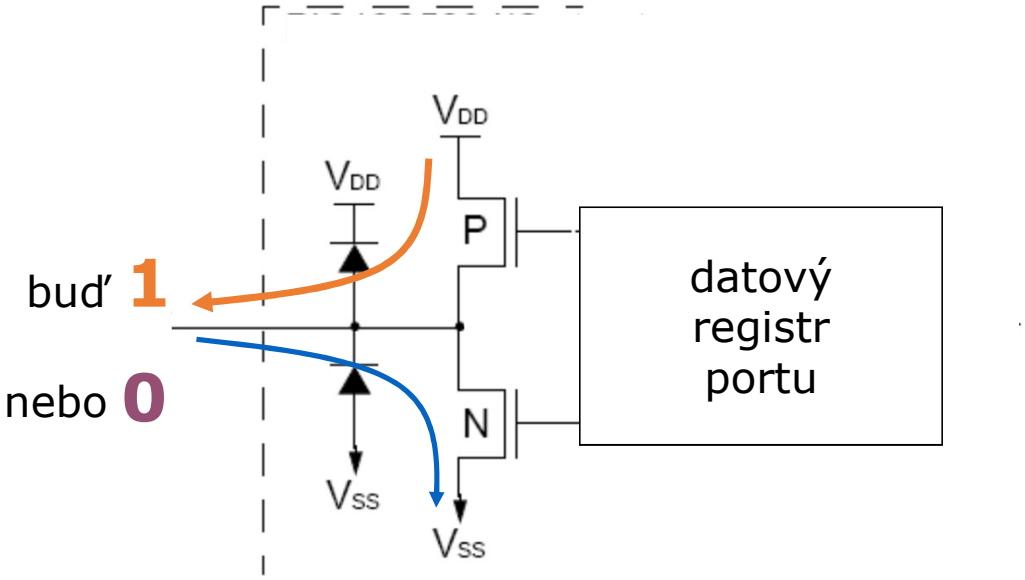
Modulace je nelineární proces, kterým se mění charakter vhodného nosného signálu pomocí modulujícího signálu.



… k čemu je nám to u mikrokontrolérů?

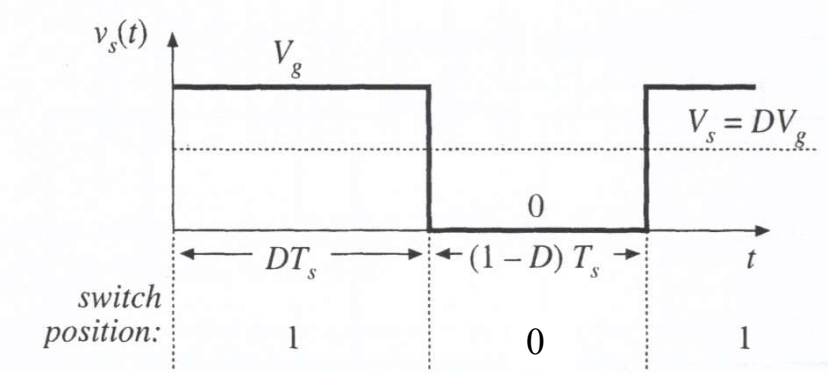
K čemu PWM?

- Vzpomeňte si, **výstup** mikrokontroléru je **číslicový**, umí jen dva stavy: 0 a 1.



… jak tímto řídit množství energie, které dostane spotřebič?

Jak dávkovat energii dvoustavovým spínačem?



Přepínač „dávkuje“ energii do spotřebiče. Výsledkem je střídavé napětí, jeho střední hodnota (stejnosměrná složka) je

$$V_s = \frac{1}{T_s}\int_0^{T_s} v_s(t)\,dt = DV_g$$

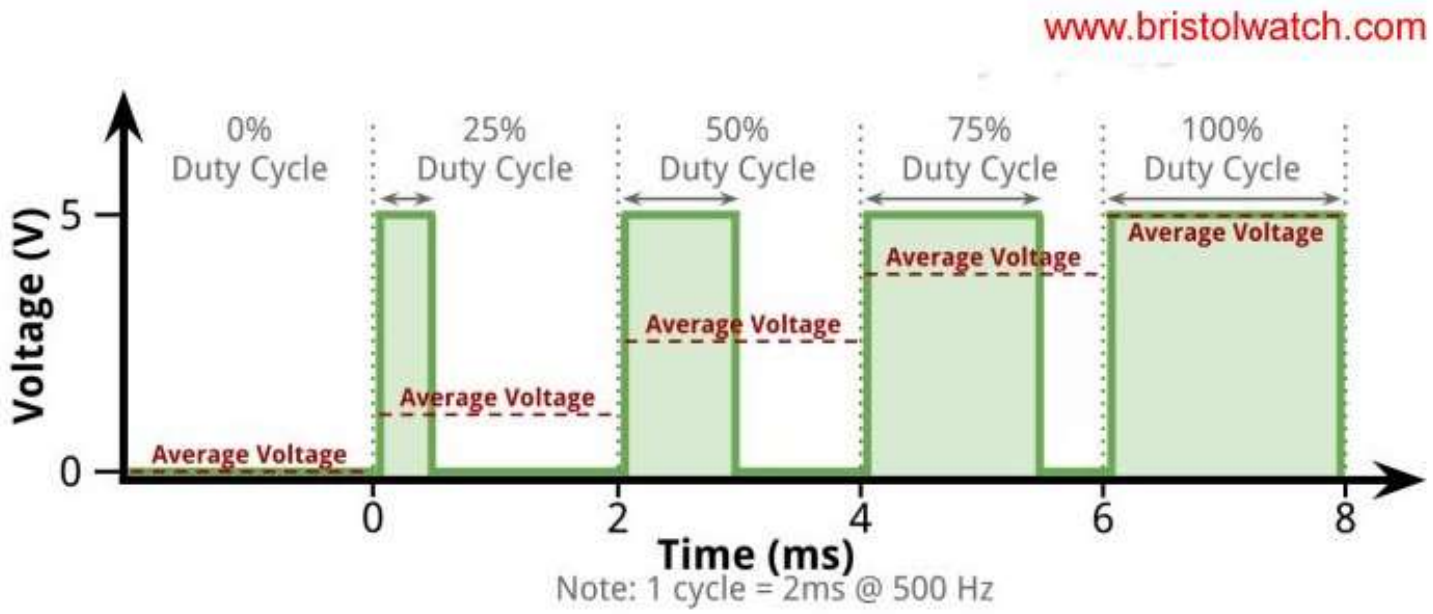
Stejnosměrná složka napětí na výstupu závisí na poměru mezi dobou, kdy je výstup v úrovni 1 a v úrovni 0 – hodnota D.

V rámci periody může D nabývat hodnot z intervalu <0,1> (či v % 0-100).

Hodnotě D říkáme **STŘÍDA** (angl. Duty Cycle).

Řízení stejnosměrné složky pulsního signálu se říká Pulsně-šířková modulace (angl. Pulse-Width Modulation, PWM).

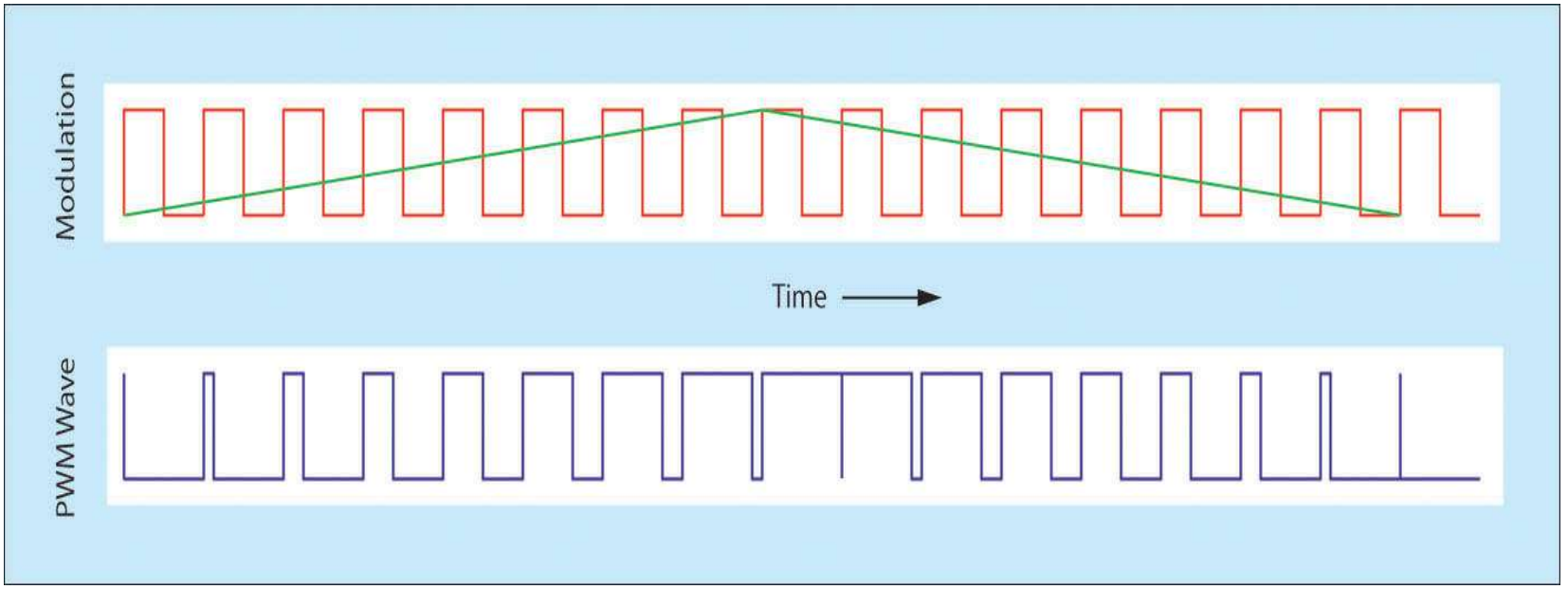
Pulsně šířková modulace



Average output voltage is proportional to duty cycle ON time.

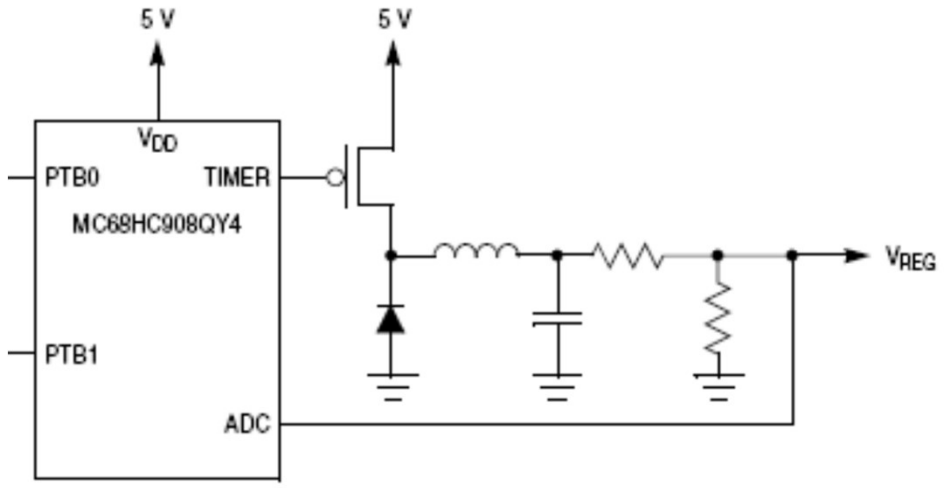
… za předpokladu, že zátěž je zapojena mezi výstup TPM a zem!

Pulsně šířková modulace



Stejnosměrná složka pulsního signálu

- Když získáme stejnosměrnou složku PW-modulovaného signálu, můžeme řídit napětí výstupu 0 – V_{dd}.
  - Připojíme filtr typu dolní propust



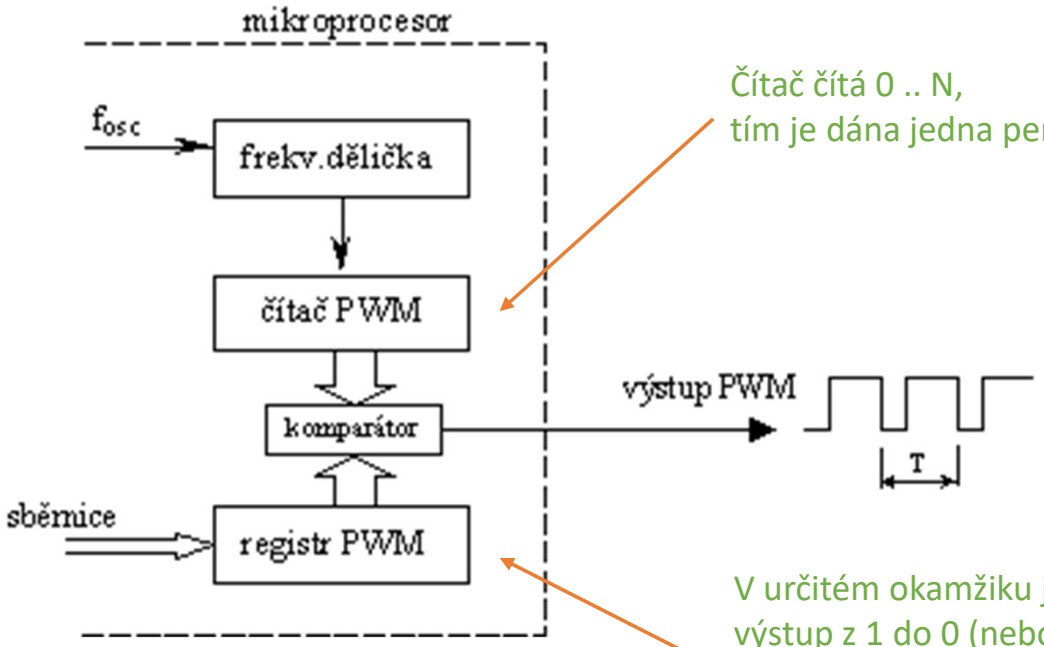
  - využijeme setrvačnosti (LED – setrvačnost oka, motor – mechanická setrvačnost, …).

Ideální PWM v MCU

- Nejlepší by bylo, kdybychom měli v MCU nějaký **registr, do něhož zapíšeme požadovanou střídu**, a od té chvíle by nějaký výstup dával PW-modulovaný signál s touto střídou, dokud bychom hodnotu registru nezměnili.
- Na generování periodických signálů máme v MCU přece Čítače/časovače (Timery, moduly TPM)!
- Šlo by využít TPM na generování PWM?

PWM v MCU

Vlastně něco podobného jako je režim Output Compare:



Čítač čítá 0 .. N,
tím je dána jedna perioda signálu.

V určitém okamžiku je třeba překlopit
výstup z 1 do 0 (nebo naopak).
Číslo v tomto registru (mezi 0 a N)
vlastně určuje střídu signálu!

Jak na PWM u našeho MCU?

- Čítač určuje periodu (frekvenci) signálu

  Délka periody = doba do přetečení čítače

  - Přesné nastavení nebývá nutné, střední hodnota signálu závisí na střídě, ne na délce periody/frekvenci.

  1. Zvolíme zdroj hodin pro čítač,
  2. zvolíme, dělitel předděličky,
  3. zvolíme „modulo“ čítače.

$$F_{TOF} = F_{source} / [prescaler*(TPMMOD+1)] = F_{TPMCK} / (TPMMOD+1)$$

Modulo čítače u PWM

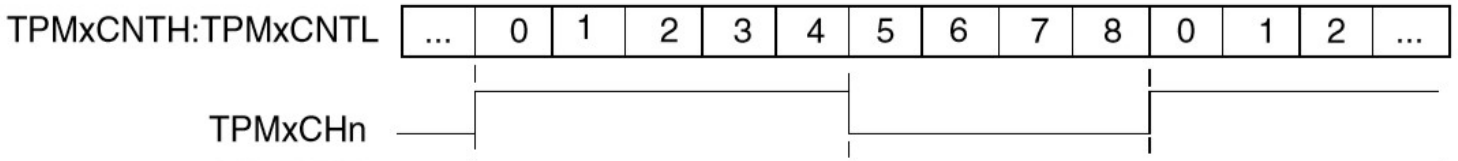
- Jaký má smysl zabývat se nastavením TPMMOD, když na délce periody u PWM až tak nezáleží?

☛**TPMMOD určuje krok nastavení střídy!** (jestliže čítač čítá od 0 do N, pak střídu lze nastavit právě v N+1 krocích).

☛Často nemá smysl mít krok příliš jemný, např. u LED nepoznáme 65536 různých intenzit svitu, stačí třeba 256, navíc je to rychlejší – zapisuje se jen jeden bajt.

Modulo čítače u PWM - příklad

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005



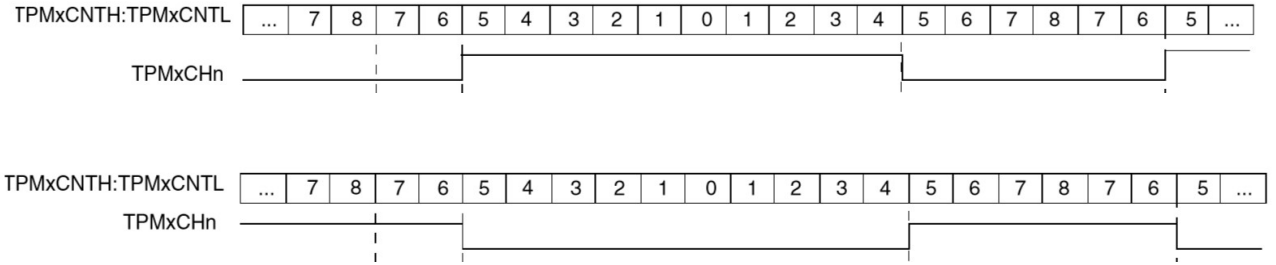
Generování výstupu

- Generování výstupu od TPM zajišťují kanály.
- Komparátor v kanálu TPM zajišťuje okamžik přepnutí – střídu.

☛Každý čítač má několik kanálů ⇨ jedním TPM možné generovat několik signálů s různou střídou (ale stejnou frekvencí).

Generování výstupu

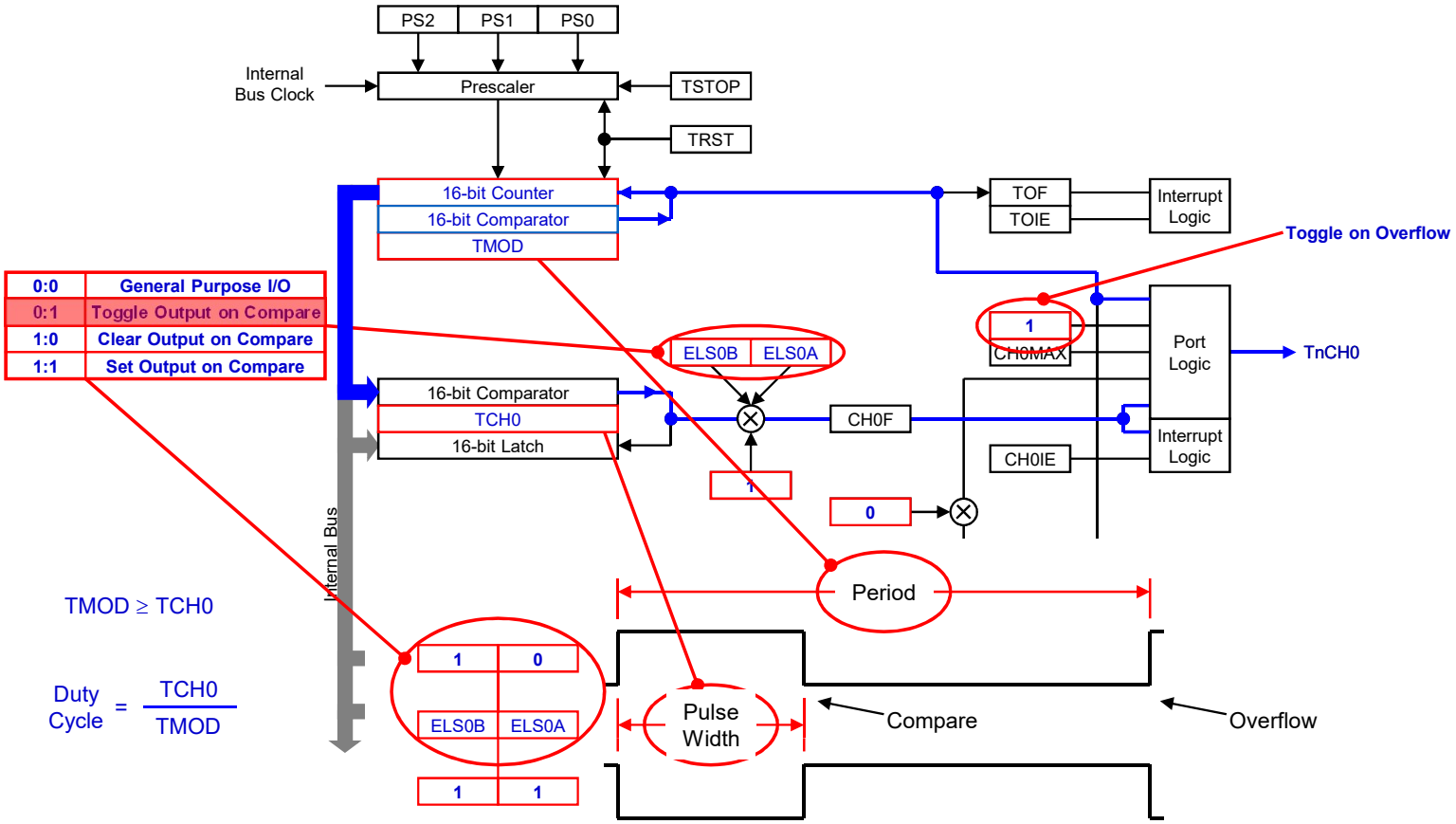
- Máme-li nakonfigurován čítač, je třeba nastavit chování kanálu
  1. Režim činnosti PWM (lze volit tzv. „edge aligned“ nebo „center aligned“)
  2. Polaritu výsupního signálu:



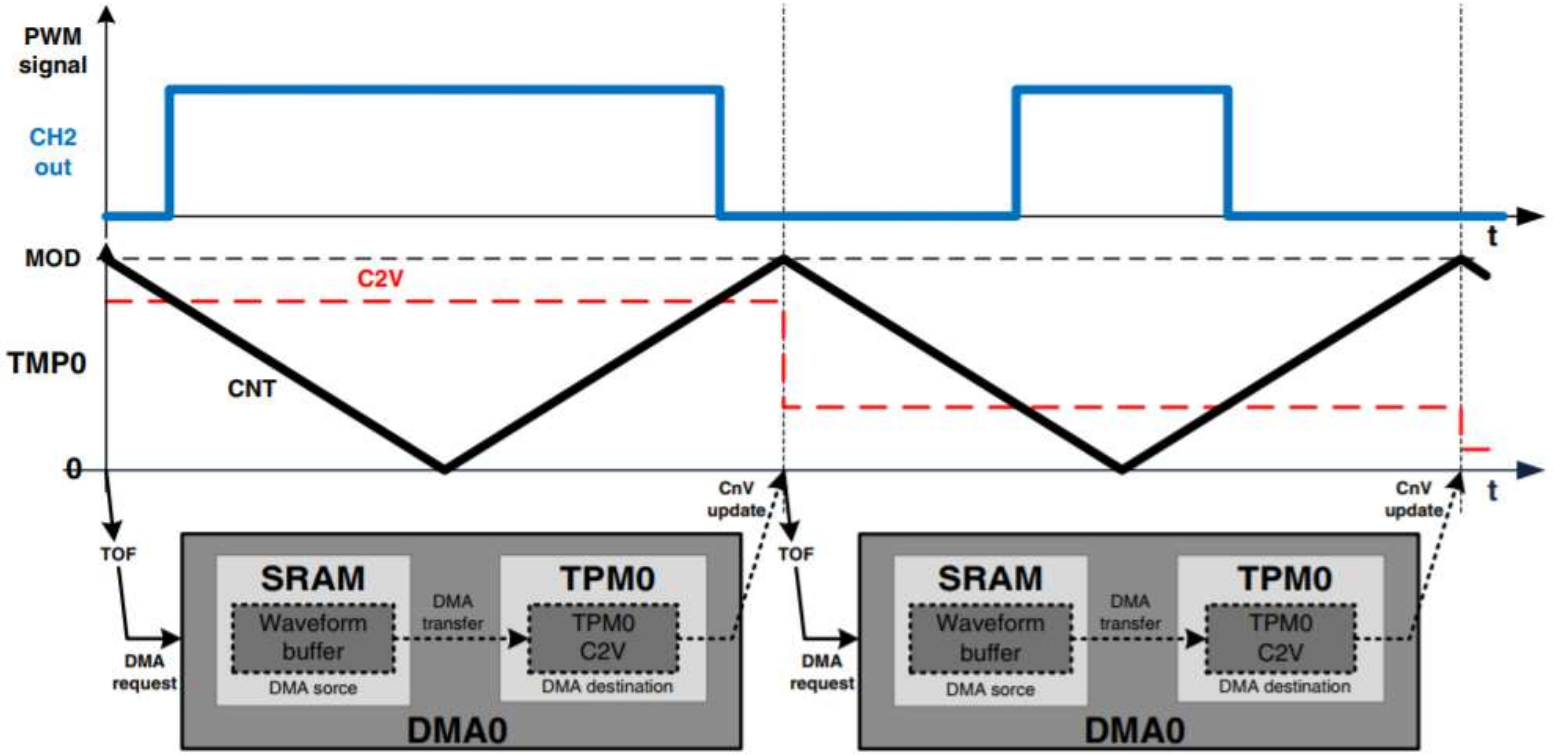
nebo:

  3. Nastavit hodnotu střídy v TPMxCnVH
- A nakonec zapnut čítání čítače – od té chvíle běží na výstupu signál.

Generování PWM kanálem TPM



Příklad – generování průběhu pomocí PWM



Pro každou další periodu přetečení čítače se automaticky nahraje pomocí DMA přenosu nová hodnota střídy z tabulky vzorků signálu umístěné někde v paměti.

Příklad – generování průběhu pomocí PWM