Mikropočítačové Systémy
MIPS
Distribuované vnorené počítačové systémy
Distributed Embedded Computer System
(Microcontrollers)

Prednáška 6.
Meranie frekvencie (prietok, rýchlosť).

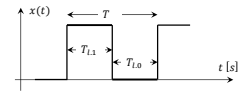
Nemôžeme presnejšie regulovať ako meriame.
Mikro <=> Makro.
Inžinier začína tam, kde končia návody a príručky.

1

1

Definície:

Frekvencia $f\ [Hz, s^{-1}]$ je fyzikálna veličina, ktorá udáva počet opakovaní periodického javu za jednotku času.
Frekvenciu môžeme definovať aj ako prevrátenú hodnotu periódy kmitov $f = \frac{1}{T}$.
Kruhová frekvencia $\omega = 2\pi f$. Ak sa pohybujeme po kružnici reprezentuje uhlovú rýchlosť. $\omega = \frac{d\varphi}{dt}$.

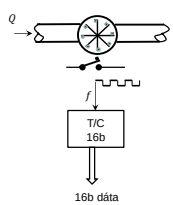


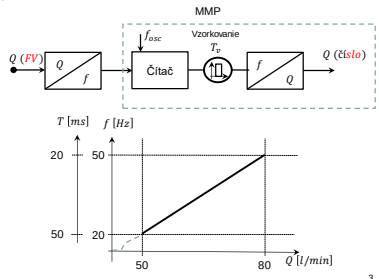
Ak platí $T_{L1} = T_{L0}$ potom je plnenie signálu: $pl = 50\%$.

2

2

Úloha 1.: Meranie a vyhodnocovanie prietoku. Ako snímač prietoku použijeme snímač prietoku s frekvenčným výstupom. Rozsah meranej veličiny je $Q \in (50\ až\ 80\ [l/min])$, čomu odpovedá frekvencia impulzov na výstupe snímača v rozsahu $f \in (20\ až\ 50\ [Hz])$.

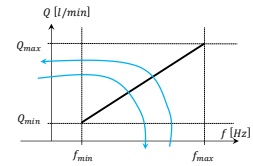




Ako, a na základe čoho nastavíme parametre meracieho kanála?

3

3



$T/C \equiv P$

$Q = f[f]$ $[l/min; Hz]$

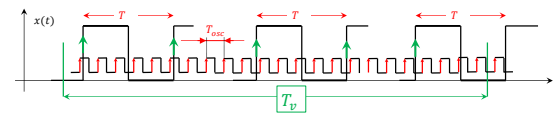
$Q = Q_{min} + \frac{Q_{max} - Q_{min}}{f_{max} - f_{min}}(f - f_{min});$ $[l/sek\ ; l/sek\ , Hz]$

$f = \frac{1}{T} = \frac{P[nT_v] - P[(n-1)T_v]}{T_v};$ $[Hz; -, sek]$

$f = \frac{1}{T} = \frac{1}{[P[nT_{OSC}] - P[(n-1)T_{OSC}]]T_{OSC}};$

Možnosti merania frekvencie:
1. Meranie počtu impulzov za jednotku času T_v.
2. Meranie času trvania jedného impulzu T.

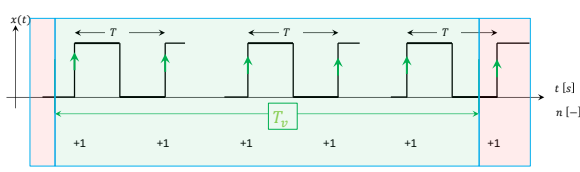
Označme: $\Delta = P[n\ ] - P[(n-1)]$



4

4

1. Meranie počtu impulzov za jednotku času.

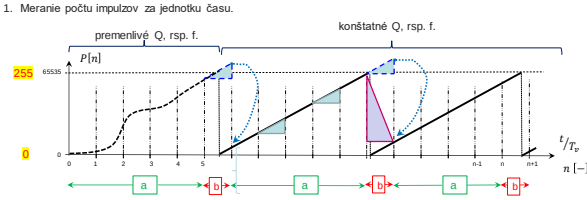


Výhody/nevýhody:
- Treba mať na pamäti: $T(t) = f(Q(t))$.
- T.j. $T \in \langle T_{min}, T_{max} \rangle$
- Frekvencia impulzov, by sa nemala počas merania meniť. ⇒ Požadujeme pomalé zmeny prietoku.
- Synchronizácia merania: $T \ll T_v$ ⇒ Chyba merania na úrovni „ T".

5

5

1. Meranie počtu impulzov za jednotku času.



??? Oba intervaly odpovedajú kladnému prietoku. ???

$P[nT_v] - P[(n-1)T_v] > 0$
$P[nT_v] - P[(n-1)T_v] < 0$

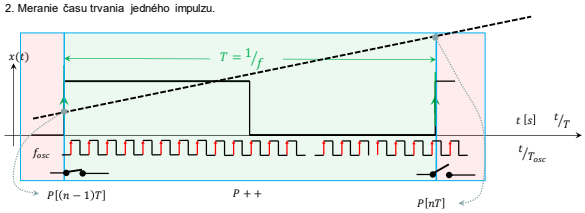
	D	H	B
+50	200	0xC8	1100 1000
+50	250	0xFA	1111 1010
$\Delta = P_n - P_{n-1}$	50	0x32	0011 0010

D	H	B
250	0xFA	1111 1010
44	0x2C	0010 1100
50	0x32	0011 0010

6

6

2. Meranie času trvania jedného impulzu.

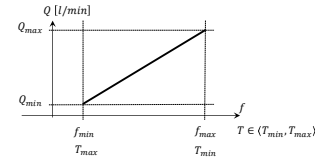


Výhody/nevýhody:
- Treba mať na pamäti: $T(t) = f(Q(t))$.
- T.j. $T \in \langle T_{min}, T_{max} \rangle$
- Frekvencia impulzov sa môže relatívne rýchle meniť.
- Potrebujeme pomocný zdroj vysokej frekvencie f_{osc}. $f_{osc} \gg f$.
- Synchronizácia merania: $T_{osc} \ll T \Rightarrow$ Chyba merania na úrovni „T_{osc}".
- Opäť treba deklarovať premenné tak, aby bol rozdiel $P[nT] - P[(n-1)T]$ stále kladný.

7

7

Zhrnutie:
Požadujeme aj v najhoršom prípade presnosť merania lepšiu ako 1%.



1. Meranie počtu impulzov za jednotku času.

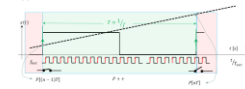
$T = \frac{T_v}{\Delta}$

Keďže T_v je konštanta musí platiť:
$T_v > 100.T_{max}$, t.j. $\Delta_{min} > 100$ a súčasne $\Delta_{max} < 2^{16}$

2. Meranie času trvania jedného impulzu.

$T = \Delta.T_{osc}$

Keďže T_{osc} je konštanta musí platiť:
$T_{osc} < \frac{T_{min}}{100}$, t.j. $\Delta_{min} > 100$ a súčasne $\Delta_{max} < 2^{16}$



8

8

Meranie otáčok: Spracovanie informácie z:
IRC (**I**nkrementálny**R**otačný**C**oder)
EnCoder

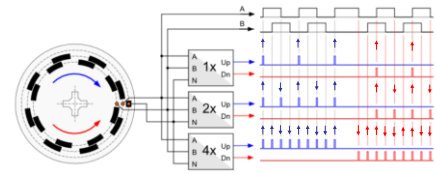
Literatúra:

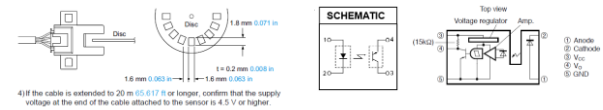
https://www.atpjournal.sk/buxus/docs/atp%20journal%202012%20str%2034-35.pdf
http://plc-automatizace.cz/knihovna/data/kodovani/IRC-code.htm
Balogh:
https://senzor.robotika.sk/senzorwiki/index.php/Inkrement%C3%A1ny_sn%C3%ADma%C4%8D
Kozáková & ... :
https://www.researchgate.net/publication/340059022_Robust_QFT-Based_Control_of_the_DC_Motor_Laboratory_Model
Huba & ... :
1.) https://www.researchgate.net/publication/333729636_Learning_Objects_and_Experiments_for_Active_Disturbance_Rejection_Control
2.) https://www.mdpi.com/2078-2489/11/3/151

9

9

Elektronika **sleduje nábežnú/dobežnú hranu** a **vyhodnocuje smer**





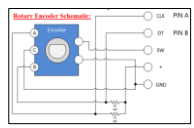
4) If the cable is extended to 20 m 65.617 ft or longer, confirm that the supply voltage at the end of the cable attached to the sensor is 4.5 V or higher.

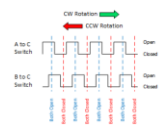
10

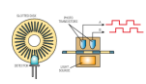
10

Spracovanie informácie z

„ROTARY ENCORED"





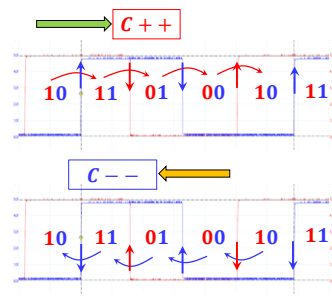


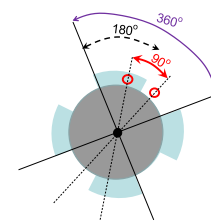
- Operating voltage: 5V.
- Pulses/360° Rotation: 20.
- Output: 2-bit gray code
- Mechanical Angle: 360° continuous.
- With built in push button switch (push to operate)
- Dimensions: (30 x 18 x 30) mm.
- Compatible with Arduino/Raspberry Pi controller board.

https://www.handsontec.com/dataspecs/module/Rotary%20Encoder.pdf

11

11

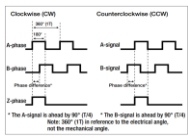




12

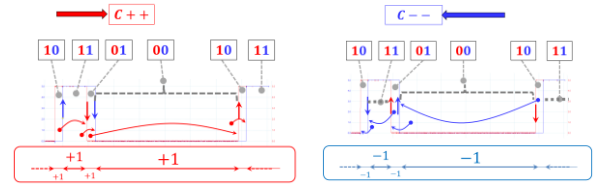
12

25mm DC Optical Encoder Motor
9V/86 rpm (za prevodovkou); Gear Ratio = 1:75
(Motor má cca 6450 rpm)



8*4 = 32

13



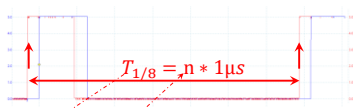
Meranie polohy je nepravidelne:
Inkrement/Dekrement nepredstavuje rovanký uhol pootočenia.
Meranie rýchlosti je vo všeobecnosti zaťažené „dýchaním - jitter" času trvania jedného inkrementu/dekrementu.

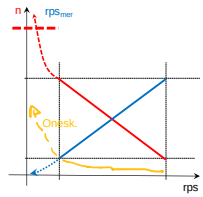
14

Meranie rýchlosti:

$1rps = \frac{1\text{ot}}{\text{s}} = \frac{1\text{ot}}{1000000\ \mu\text{s}} = \frac{\frac{1}{8}\text{ot}}{125000\ \mu\text{s}}$

$rps_{mer} = \frac{125\,000}{n}$ $[ot.s^{-1}; "float"]$ $rps_{mer} = \frac{1\,250\,000}{n}$ $[\{xx.z\}; 10 * ot.s^{-1}; "int"]$ $\{xx.z\} * 10 \equiv XXZ$

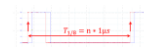




rps	$T_{1/8}$ [µs]	n [−]	UNO/MEGA	Onesk. [s]
0	-	-	-	-
1	125 000	125 000	long (4B)	0.125
10	12 500	12 500	int (2B)	0.012 5 !
20	6 250	6 250	int (2B)	0.006 3
100	1 250	1 250	int (2B)	0.001 25

15

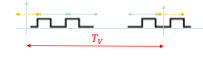
V predchádzajúcej časti sme rýchlosť otáčania vyhodnocovali ako funkciu „času trvania jedného impulzu".



Nevýhoda: Premenlivé oneskorenie. $G_n(s) = \frac{K}{Ts+1} e^{-sD(rps)}$

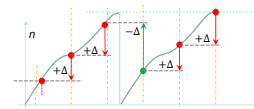
Druhým spôsobom merania rýchlosti otáčania sa motora:

$v = \frac{n(t_n) - n(t_{n-1})}{T_V} * k_v$



Kde $n(t_n)$ a $n(t_{n-1})$ sú dva údaje z počítadla impulzov vzdialené o čas T_V. Počítadlo je, napr. 16-bitové a po napočítaní 65535 impulzov sa pretočí na hodnotu nula.
Úlohu: Návrh algoritmu vyhodnotenia rýchlosti tak, aby sme pri pretočení počítadla nevyhodnotili jeden krát opačnú rýchlosť, viď. obr. sme už riešili.

Výhoda: Konštantné oneskorenie.

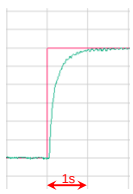


Max 65535

16

Už sme niekoľko krát naznačili, že rýchlosť by sa nemala počas vyhodnocovania meniť.
Mala by byť konštantná.
V „MFCH tabuľky pre 7 až 9 ročník ZDŠ" sa na strane 101 píše:
Rýchlosť rovnomerného priamočiareho pohybu $v = \frac{s}{t}$ $[ms^{-1}]$; $s = vt[m]$

Podstatné je pre nás slovo „rovnomerný". V opačnom prípade by sme museli použiť integrálny počet.
Našťastie $1 \ll 100$. Viď. pravítko. Na obrázku je nakreslená prechodová charakteristika – skoková zmena rýchlosti. Zrejme bude vyhovovať čas vyhodnocovania kratší ako 10ms.

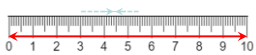


Na akomkoľvek úseku $\Delta t = 0.01\ sek$ môžeme považovať rýchlosť (tu rýchlosť otáčania) za konštantu.

Pre DC motor platí: počet impulzov na otáčku je
440*4 = 1760 = kv (440 je delenie kruhu)
Prejdená dráha (pootočenie) je dané vzorcom

$\Delta n = n_{nové} - n_{staré} = rps * kv * \Delta t$

$\Rightarrow \quad rps = \frac{\Delta n}{kv * \Delta t}$



17

Nasledovná tabuľka nám naznačuje, že musíme urobiť kompromis.
Ak chceme zvýšiť presnosť, musíme zväčšiť Δt. To ale spôsobí zväčšenie chyby v prechodných procesoch.

Cele sme si to ale pokazili tým, že na začiatku sme predpokladali, že prietok je len kladný. A obdobný algoritmus sme použili aj na meranie rýchlosti.
Motor sa môže ale točiť v oboch smeroch (auto môže aj cúvať).
Zmení sa niečo ak budeme predpokladať rýchlosť oboch znamienok?

$rps = \frac{\Delta n}{kv * \Delta t} \Rightarrow (int)((\Delta n * \underbrace{\Delta t^{-1} * 10}_{n = rps * 1760 * 1s})/kv)$

$G_N(s) = \frac{K}{Ts+1} e^{-sD}$

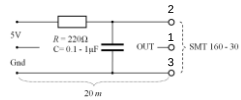
rps merané osciloskopom	n [−] za 1sek	Δn za 0,01sek	rps výpočet	Onesk. [s] Δt
1	1 760	17	0.9	0.01
10	17 600	176	10.0	0.01
20	35 200	352	20.0	0.01
21	36 960	369	20 ?0? (20.9)	0.01
60	105 600	1 056	60.0	0.01
61	107 360	1 073	60 ?0? (60.9)	0.01

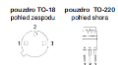
18

SMT 160-30 (172) snímač teploty:

Snímač teploty s PWM výstupom. PWM vo funkcii D/A prevodníka.
Je to prevodník teplota na širkovo modulovaný signál PWM.
Merací rozsah je (–45 °C až 150 °C). Vyrába sa v púzdrach, napr.:
T018, T092, T0220.

Spôsob pripojenia:

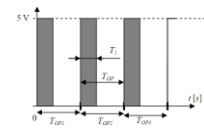




Základné vlastnosti, parametre :
- Rozsah meranej teploty je -45 až 130°C
- Absolútna presnosť ±0.7 °C
- Odchýlka prevodovej charakteristiky od lineárnej je < 0.2 °C
- Výstupný signál je kompatibilný s TTL a CMOS logikou
- Spotreba obvodu je menšia ako 1 mW
- Snímač je kalibrovaný vo výrobe
- Výstup PWM signál s frekvenciou opakovania: $f_{OP} = 1 \div 4\,[kHz]$, $T_{OP}(pre\ \ 4kHz) = 250\,[\mu s]$

19

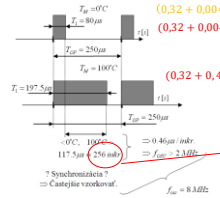
SMT 160-30 snímač teploty:



Plnenie ako funkcia meranej teploty:
$pl. = \frac{T_1}{T_{OP}} = 0.32 + 0.0047 \cdot T_M\ [-;\ ^\circ C]$ $\quad T_{M\,max} = (1 - 0{,}32)/0{,}0047 \Rightarrow 144{,}68℃$

Spracovanie informácie:
Treba zmerať aj T_1 aj T_{OP} počas jednej periódy opakovania.

Príklad: Pomocou snímača SMT 160 – 30 meriame teplotu v rozsahu T_M = (T_0 až 100 °C).
T_0 = (0 až 20 °C).



$(0{,}32 + 0{,}0047 * T_M) * T_{OP}$
$(0{,}32 + 0{,}0047 * 0) * 250us \Rightarrow 80us$
$(0{,}32 + 0{,}47) * 250us \Rightarrow 197{,}5us$

T_M počítame zo vzťahu:
$T_M = \frac{\frac{T_1}{T_{OP}} - 0.32}{0.0047}\ [^\circ C; -; -]$

T_1 aj T_{OP} treba merať tak, aby presnosť merania odpovedala 8-bitovému prevodníku.

20

Informácia je prvotne spracovaná pomocou C/T mikropočítača.
Vlastnosti určíme takto: Počítadlo viacej napočíta ak horší prípad odpovedá :
T_{OP} (1kHz) = 1000 μs. A $f_{osc} = 8MHz \Rightarrow T_{osc} = 0{,}125\mu s$

Za čas T_{OP} počítadlo napočíta $\frac{1000\,[\mu s]}{0.125\,[\mu s]} = 8000\,[SC]$

$\Rightarrow$ Treba použiť 16 bitové počítadlo.

Čas spracovania: $(0.25[ms] \div 1[ms])$

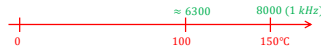
Ak nechceme použiť aritmetiku pohyblivej rádovej čiarky, upravíme vzťah do tvaru:

$T_M = \frac{\frac{T_1}{T_{OP}} - 0{,}32}{0{,}00470}\ [℃; -; -]$

$T_M * 10 = \frac{\frac{T_1 \cdot 10^5}{T_{OP}} - 32000}{47}$

Rozsahy čísel:
$T_1 \in (\text{"0"} \div 8000)$
$T_{OP} \in (2000 \div 8000)$
4kHz 1kHz

Výsledok predpokladáme v tvare napr.: 995 ≡ 99.5. Správny výsledok získame, len ak je vzorec naprogramovaný v „správnom" poradí a ak zvolíme správne dátové typy.



21

Postup výpočtu:

$T_M = \frac{\frac{T_1}{T_{OP}} - 0{,}32}{0{,}00470}\ [℃; -; -]$

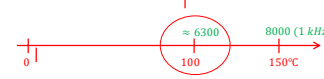
$T_M * 10 = \frac{T_1 * 10^5}{T_{OP}} - 32000$ / 47

$\left[\frac{(T_1 * 10^5)}{T_{OP}} - 32000\right] / 47$

$[(T_1.*)/T_{OP} - 32000] / 47$

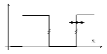
$\frac{6300 * 100000}{8000} - 32000$ / 47

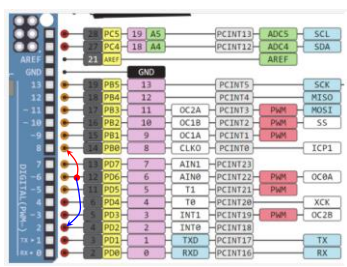
< 470000 / 47 $\doteq 99{,}4$

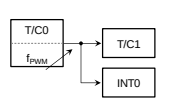


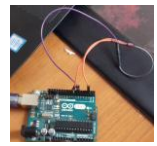
22

Meranie frekvencie ARDUINO UNO.
Zdroj PWM preladiteľného signálu.
Zapojenie:









23

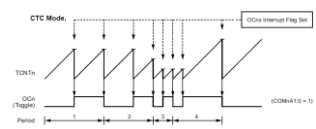
Zdroj PWM preladiteľného signálu.
Nastavenie:

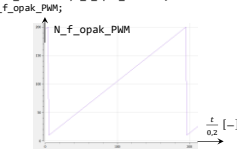
Port D, pin 6 output. T/C0 je nastavený v móde 2 (Clear Timer on Compare Match Mode).
Plnenie je 50%. Delička je nastavená na deleno N = 64.
Frekvencia PWM signálu sa mení v rozsahu $f_{PWM} \in \langle 12{,}5\ kHz\ \ az\ 625Hz \rangle$ t.j. $OCR0A \in \langle 9,\ 199 \rangle$

$f_{PWM} = \frac{f_{clk\ I/O}}{2 * N * (1 + OCR0A)}$

```
// OCR0A                f_PWM              T_PWM
// <9+1 az 199+1>==<12,5kHz az 625Hz>==<80us az 1,6ms>

void Zmena_f_opakovania(void){
  N_f_opak_PWM++;
  if (N_f_opak_PWM > 199)N_f_opak_PWM = 9;
  OCR0A = N_f_opak_PWM;
}
```

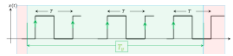




24

Meranie frekvencie: Počet impulzov za jednotku času.
Ako určíme „jednotku času", t.j. T_v?
Aj v najhoršom prípade ($T_{PWM} = T_{PWM\ max} = 1{,}6ms$) treba napočítať aspoň 100 impulzov. ⇒ zvolíme $T_v = 200ms$.
Tu budeme tento interval generovať pomocou oneskorenia:

```
void Generovanie_Tv_02sek(void){
  while(N_ones--) _delay_ms(100);
  N_ones = 2;
}
```



Na cvičení ***použijete*** jeden s T/C tak, aby generoval $T_v = 200ms$. PWM impulzy načítavame v prerušení na pine INT0 (PORTD pin 2.) pri nábežnej hrane.

```
void ini_Interrupts(void){   // INT0
     DDRD &= ~(1<<DDB2);//  PORTD.2  input
     PORTD |= (1<<PORTD2);// Pullup Rezistor
     // 0b00000011;// nabezna hrana na INT0
     EICRA |= (1<<ISC01)|(1<<ISC00);
     EIMSK |= (1 << INT0);     // povolenie INT0
     sei();
}
```

```
ISR(INT0_vect){
 Poc_imp++;
}
```

$25\ 000 \Leftarrow \frac{16\ 000\ 000}{2*64} * 0{,}2$

$$f_{PWM} = \frac{1}{T_{PWM}} = \frac{f_{clk\ I/O}}{2 * N * (1 + OCR0A)}$$

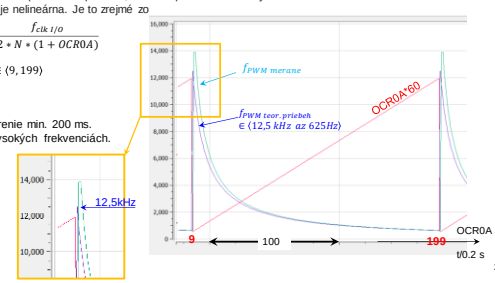
Nastavené — Vypočítané

```
void Vypocet_N_Int0(void){
dif_N_UP_INT0 = Poc_imp - Poc_imp_st;
Poc_imp_st = Poc_imp;
N_UP_INT0 = (25000/dif_N_UP_INT0)-1;
}
```


Výpočet meranej frekvencie: Počet impulzov za jednotku času.
Na úvod treba povedať, že takto postavený príklad (nemáme inú možnosť) má synchronizované meranie parametrov PWM signálu s taktovaním MMP. Ďalej treba pripomenúť, že nedokážeme generovať f_{PWM} „spojito", tak ako by to dokázal napr. snímač prietoku, alebo aspoň s rovnomerným krokom. Prevodová charakteristika $f_{PWM} = f(\langle OCR0A \rangle)$ je nelineárna. Je to zrejmé zo vzorca $f_{PWM} = \frac{f_{clk\ I/O}}{2 * N * (1 + OCR0A)}$

$OCR0A \in \langle 9, 199 \rangle$

Dopravné oneskorenie min. 200 ms.
Veľká chyba pri vysokých frekvenciách.



Výpočet meranej frekvencie: Trvanie jedného impulzu.
Na úvod treba povedať, že takto postavený príklad (nemáme inú možnosť) má synchronizované meranie parametrov PWM signálu s taktovaním MMP. Tentokrát privedieme PWM signál na PORTB pin0 – vstupný. Ten pin využijeme ako vstup do C/T1. C/T1 načítava f_{osc}. Ak sa objaví nábežná hrana PWM signálu, odpamätáme stav C/T1. A vypočítame, buď frekvenciu alebo tomu odpovedajúcu hodnotu, odmeranú hodnotu $OCR0A$.
Vzorce zostanú tie isté. Tento krát priamo v ISR vypočítame požadovanú meranú veličinu. Len výpis robíme pomalšie (raz 200 ms).

```
ISR(TIMER1_CAPT_vect){
// globalne prerusenie je zakazane
T1_st_vz = T1_no_vz;
T1_no_vz = ICR1;
N_Capt_T1 = ((T1_no_vz - T1_st_vz)>>7)-1;
}
```

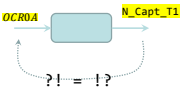
TU

Nastavené $f_{PWM} = \frac{f_{clk\ I/O}}{2*N*(1+OCR0A)} = \frac{16000000}{2*64*(1+OCR0A)} = \frac{125000}{(1+OCR0A)}$

Odmerané $f_{PWM} = \frac{f_{clk\ I/O}}{\Delta}$ ... $= \frac{16000000}{\Delta}$

$\Delta = ((T1\_no\_vz - T1\_st\_vz)$

Tento krát nemusíme riešiť vyčítavanie 16-b registra cez 8-b dátovú zbernicu.
?? Prečo ??




```
void ini_Interrupts(void){// Timer1 Capture
DDRB &= ~(1<<DDB0);//  PORTB.0  input
PORTB |= (1<<PORTB0);// Pullup Rezistor
TIMSK1 |= (1<<ICIE1); // povolenie prerusenia od capture T1
TCCR1B |= (1<<ICES1); // odchytenie na nabeznu hranu
sei();
}
```

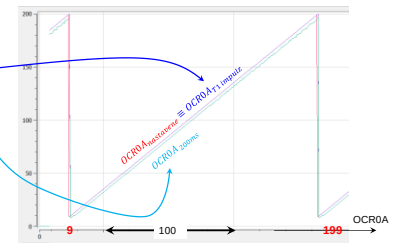

Aj v tomto príklade aj v predchádzajúcom príklade sme menili f_{PWM} až po uplynutí 200ms.
Ak by sme f_{PWM} menili aj počas tohto intervalu, chyba by pri meraní frekvencie, ako priemer za čas 200ms, narastala. Výpočet frekvencie z času trvania jedného impulzu má síce premenlivé dopravné oneskorenie, ale v tu použitej časovej mierke nie je zobraziteľné.

Zopakujeme pri výpočtoch sme vlastne použili rovnosti:

$T_{PWM} = T_{osc} * "počet\ imp."$

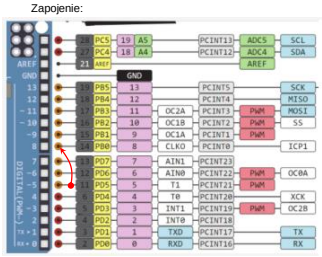
$200ms = T_{PWM} * "počet\ imp."$

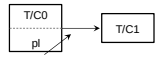
$$f_{PWM} = \frac{f_{clk\ I/O}}{2 * N * (1 + OCR0A)}$$



Meranie plnenia PWM signálu:
Tento krát trocha modifikujeme zapojenie a aj nastavenie parametrov PWM impulzov. Frekvenciu opakovania PWM signálu necháme konštantnú a budeme meniť plnenie. Veľa krát sa meraná veličina zakóduje do plnenia impulzu. A našou úlohou spätne vypočítať veľkosť meranej veličiny.

Zapojenie:



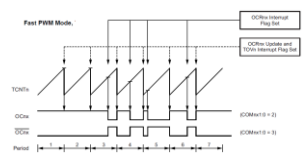


Nastavenie T/C0:

Port D, pin 5 output. T/C0 je nastavený v móde 7 (Fast PWM Mode).
Plnenie je premenlivé a nastavuje sa pomocou registra OCR0B. Delička je nastavená na deleno N = 64.
Frekvencia PWM signálu je daná hodnotou registra OCR0A.

$$f_{PWM} = \frac{f_{clk\ I/O}}{N * (1 + OCR0A)}$$

$f_{PWM} = 2500Hz$
t.j. $OCR0A = 99$.




```
void ini_TC0(void){//  Nastavenie TC0
 set_bit(DDRD,PIND5); //OC0B PWM pin
 // 7 6          5 4          3      2      1 0
 // COM0A[1:0]  COM0B[1:0]WGM0[1:0]
 TCCR0A = 0b00100011;// OC0B  PWM mod = 7
 // 7 6 5 4  3      2   1   0
 //          WGM02  CS0[2:0]
 TCCR0B = 0b00001011;// fosc/64
 OCR0B = pl_8b;
 OCR0A = N_f_opak_PWM;
}
```

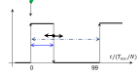
Plnenie impulzu meníme ako v predchádzajúcich príkladoch raz za 200ms v rozsahu od 5 do 70%.

```
void Zmena_plnenia(void){
 pl_8b++;
 if (pl_8b > 70)pl_8b = 5;
 OCR0B = pl_8b;
}
```

Príklad sme zostavili tak, aby čítač pretiekol po 100 impulzov z predde liačky. To, znamená, nastavujeme plnenie v percentách. Z dôvodu „synchronizácie" a celočíselnej aritmetiky je chyba merania plnenia nulová. Vyhodnotenie robíme priamo v ISR a zobrazujeme raz za 200ms. Viď. ďalej.

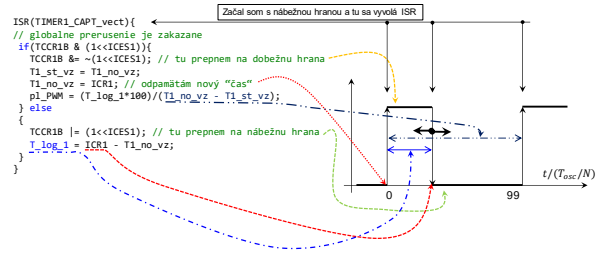
```
void ini_Interrupts(void) {   // Timer1 Capture
 DDRB &= ~(1<<DDB0);//  PORTB.0  input
 PORTB |= (1<<PORTB0);// Pullup Rezistor
 TIMSK1 |= (1<<ICIE1); // povolenie prerusenia od capture T1
 TCCR1B |= (1<<ICES1); // odchytenie na nabeznu hranu
 sei();
}

void ini_TC1(void){
DDRB &= ~(1<<DDB0); //ICP1 = PORTB.0, input
PORTB |= (1<<PORTB0);  // Pullup Rezistor
//  Nastavenie TC1
// 7 6 5 4  3      2   1   0
//         WGM02  CS0[2:0]
TCCR1B |= (1<<CS10); // 0b00000001;// fosc/1
}
```



31

Meranie času trvania log.1, času trvania celého impulzu, ako aj výpočet plnenia v percentách (*100) sa realizuje v obsluhe prerušenia.




```
ISR(TIMER1_CAPT_vect){
// globalne prerusenie je zakazane
 if(TCCR1B & (1<<ICES1)){
   TCCR1B &= ~(1<<ICES1); // tu prepnem na dobežnu hrana
   T1_st_vz = T1_no_vz;
   T1_no_vz = ICR1; // odpamätám nový "čas"
   pl_PWM = (T_log_1*100)/(T1_no_vz - T1_st_vz);
 } else
 {
   TCCR1B |= (1<<ICES1); // tu prepnem na nábežnu hrana
   T_log_1 = ICR1 - T1_no_vz;
 }
}
```


32

Na výpis nepoužívame LCD display, ale sériový monitor alebo serialplot. Využívame len kanál na vysielanie znakov. Nastavenie je realizované nasledovne: (115200bps, 8 dátových bitov, bez parity,1 stoP bit)

```
#define BAUDRATE 115200
#define mybr ((F_CPU + BAUDRATE * 4UL) / (BAUDRATE * 8UL) - 1UL)

void ini_USART0(unsigned int mybr){
 UBRR0 = mybr;
 set_bit(UCSR0B,TXEN0);
 set_bit(UCSR0C,USBS0);
 set_bit(UCSR0C,UCSZ01); // Set format: 8data, 1stop bit
 set_bit(UCSR0C,UCSZ00);
}
void USART_Transmit( unsigned char data ){
 /* Počkaj, až sa vyprázdni vysielací buffer */
while ( !( UCSR0A & (1<<UDRE0)) );
/* Vlož data do buffer,a pošli data */
 UDR0 = data;
}

void zob_text_UART(char *s){
register unsigned char c;
while((c = *s++))USART_Transmit(c); // retazec konci "nulou"
}

void zob_text(char *s){
register unsigned char c;
while((c = *s++))lcd_data(c);
}
```


33