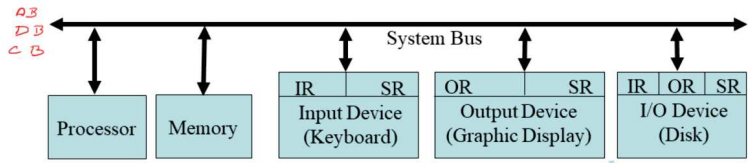
Mikropočítačové Systémy - MIPS

Prednáška 6 – I. časť
Prerušenia

1

Interrupts - Prerušenia

Na počiatku bolo všetko jednoduché. Prvé počítače nemali prerušovací podsystém. Postupnosť vykonávania inštrukcií sa riadila pomocou obsahu adresného čítača. Neskôr sa zistilo, že niektoré periférne obvody sú vzhľadom na CPU a pamäť pomalé. Procesor sa musel o ich pozornosť usilovať. Neustále kontroloval stavový register takéhoto pomalého zariadenia, aby zistil, či je alebo nie je pripravené na spoluprácu s procesorom. Týmto sa rýchlosť procesora znížila. Takáto metóda zisťovania stavu periférie sa nazýva: *Polled Method* - opytovacia, testovacia, … .



Tri základné spôsoby komunikácie s I/O podsystémom:
Polled Method – opytovacia metóda
Interrupt Method – prerušenia
DMA – priamy prenos medzi pamäťou a perifériami

Pomalá / Rýchla per.

2

Prenosová rýchlosť I/O zariadení:

•Klávesnica: „cca“ 10 znakov (B)/sek. Charakteristika prenosu: Komunikácia procesor – klávesnica. Väčšinou sa náhodné prenášajú byty, resp. niekoľko bytov. CPU musí neustále *testovať „zatlačenie*“ klávesnice. Kód zatlačeného znaku sa uloží do *Input register* a radič klávesnice nastaví príznak pripravenosti znaku. Procesor znak prečíta a zruší príznak => znak sa prečíta len raz. Procesor trávi veľa času testovaním: => *Interrupt*.

Opačný proces možno badať smerom k displayu, resp. k tlačiarni.

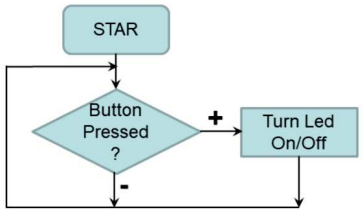
•Laserová tlačiareň: cca 100 000 znakov/sek,

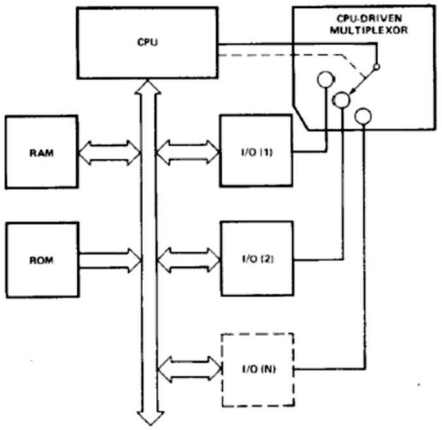
•Grafický display: cca 30 000 000 znakov/sek,

•Pevný disk: cca milióny B/sek. Charakteristika prenosu: Prenos veľkého množstva údajov organizuje procesor po bytoch, resp. slovách. => Strata času. Výhodnejšie je prenos organizovať po blokoch (zabezpečuje hardware) : => Rýchlejší prenos. Procesor „oddychuje“, resp. robí inú prácu. => *DMA*.

Obsluha periférií:
Polled Method – opytovacia metóda

Spôsob komunikácie – protokol je riadený CPU pomocou programu.
Nech má I/O(2) „pripravený“ znak pre CPU.
CPU adresuje I/O(2) a prečíta znak do ACU. Atď.
Ak nie je rýchlosť CPU a I/O(2) rovnaká, CPU preberie znak niekoľko krát.
Riešenie je v použití: *Status registra (bitu)*.

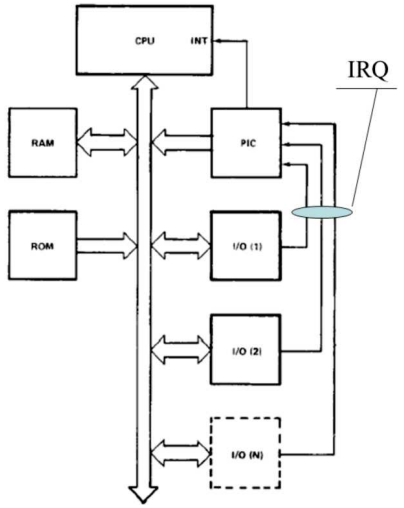




3

Obsluha periférií:
Interrupt– prerušenie

Niekedy je potrebné prerušiť beh programu, napr.: pokles napájania. Prerušením behu programu môžeme tiež informovať CPU o ukončení udalosti: napr. odvysielanie znaku cez USART. Moderné CPU majú inštrukcie, ktoré sa správajú podobne ako hardwarové prerušenie. T.j. Ak je prerušená činnosť CPU, riadenie sa dočasne odovzdá inému „programu“. Po skončení sa vráti k pôvodnej činnosti.
Viacnásobné prerušenia: Priorita.



4

Tvorcovia hardwaru prišli na to, že testovanie príznaku sa ľahko realizuje obvodovo. Do procesora dorobili vstup, asynchrónne pracujúci s názvom INTerrupt, ktorý procesor „testuje na pozadí“ – počas SC. Cez tento vstup sa hardwarovo procesoru oznamuje, že mimo procesora sa niečo deje, o čom by mal vedieť. Mal by zareagovať. Takýto spôsob obsluhy periférneho obvodu, takéto prerušenie vykonávania programu je nazvané hardwarové prerušenie a počítačová prax to jednoducho nazvala prerušenie – INTERRUPT. Iné ako hardwarové nebolo. V tejto etape vývoja počítačov je prerušenie každému jasné: Požiadavka a vyhodnotenie prerušenia vzniká na hardwarovej úrovni (na základe žiadosti o prerušenie IRQ). CPU vie vykonávať len a len program, z tohto dôvodu obsluha prerušenia je len vykonanie programu, ktorý sa len nepatrne líši od bežného podprogramu.

Túto idilku pokazil Intel, keď do svojich procesorov zaradil inštrukciu int (software interrupt). Aby sme v tom mali jasno, na úvod treba povedať, že prerušenia delíme do skupín:

Vonkajšie – hardwarové. Prerušenie vyvolané technickými prostriedkami. O prerušenie požiada periféria, keď potrebuje reakciu procesora. Niekedy sa tento typ prerušenia nazýva aj **asynchrónné.** Priamo nesúvisí s vykonávaným programom a môže nastať kedykoľvek. Procesor má zvyčajne dva prerušovacie vstupy pre externé prerušenia (CPU má rád poriadok, preto treba deje zosynchronizovať):

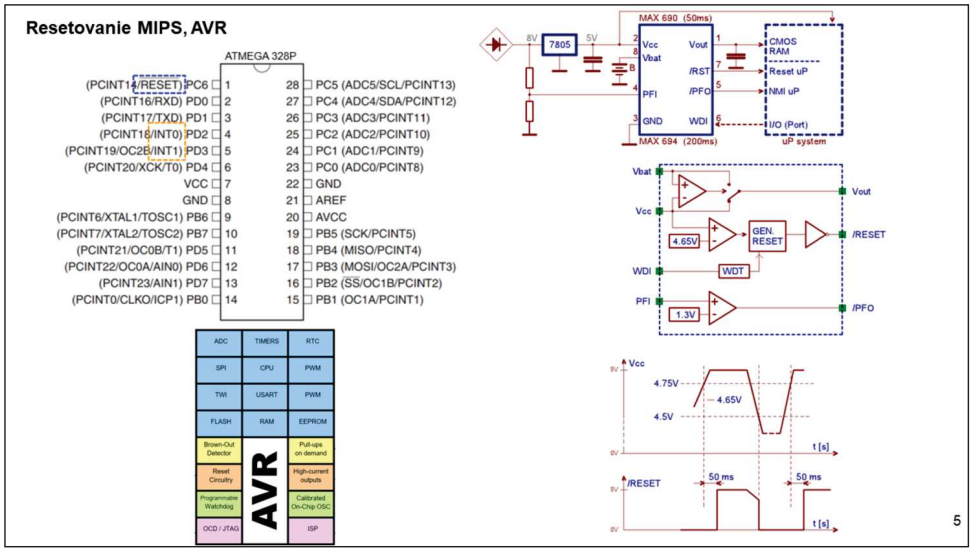
- **Vstup nemaskovateľného prerušenia** (signál procesora: Non Mascable Interrupt - NMI). NMI - sa používa hlavne na signalizovanie

“katastrofických “ udalostí ako je pokles napájacieho napätia alebo chyba parity operačnej pamäte alebo zbernice. Tento vývod procesora je teda určený pre prerušenia, ktoré nie je možné zakázať.

- **Vstup maskovateľného prerušenia** (signál procesora: Mascable Interrupt - INTR). Obsluhu tohto prerušenia možno programovo zakázať, inštrukciou CLI, vynulovaním bitu IF (Interrupt Flag) v príznakovom registri procesora. Pokiaľ je IF = 0 procesor prerušenie ignoruje. Tento príznak sa vzťahuje len na INTR. Nie na vnútorné prerušenia a NMI.

Vnútorné – softwarové. Priamo súvisí s vykonávaným programom a nemôžeme ho zakázať.

Niekedy sa tento typ prerušenia nazýva aj **synchrónnym.** Procesory AVR nemajú softvarové prerušenia. Môžu len niektoré prerušenia softwarovo vyvolať.



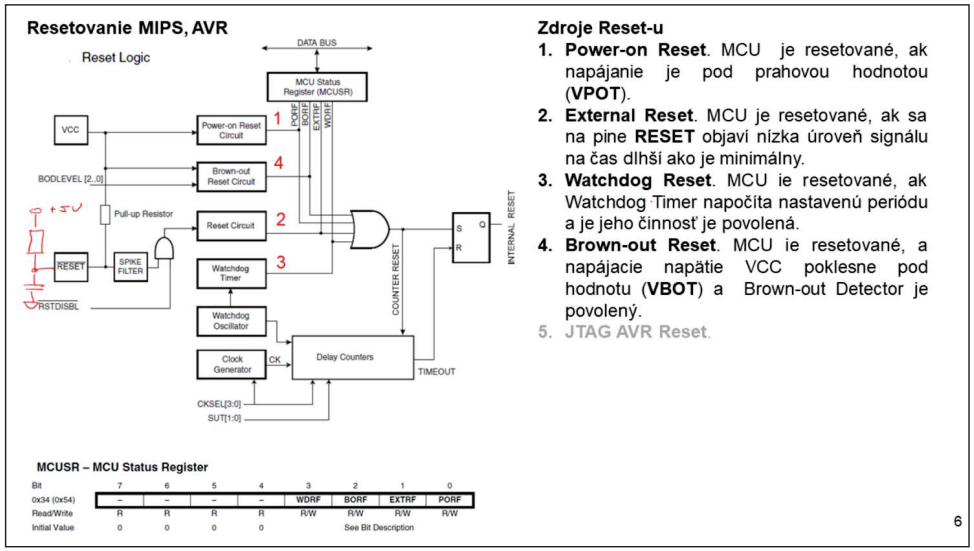
Resetovanie:

Počas Reset-u, všetky I/O Register-e sa nastavia na inicializačné hodnoty (okamžite, ako sa aktivuje signál Reset. Netreba k tomu žiaden hodinový signál) a program sa začne vykonávať od **Reset** vektora. Inštrukcia vektora **Reset** musí byť **JMP** – absolútny skok na inštrukcie obsluhy rutuny Reset.

Ak užívateľský program nepoužije prerušenia, môže byť uložený do pamäte za adresu vektora **Reset**. Atď.

Potom čo sa objaví **Reset**, aktivuje sa oneskorovací čítač. Počas tohto oneskorenia môže napájanie dosiahnuť stabilnú hodnotu, potrebnú k normálnej činnosti. Oneskorenie sa dá nastaviť pomocou **Fuses** - poistiek.

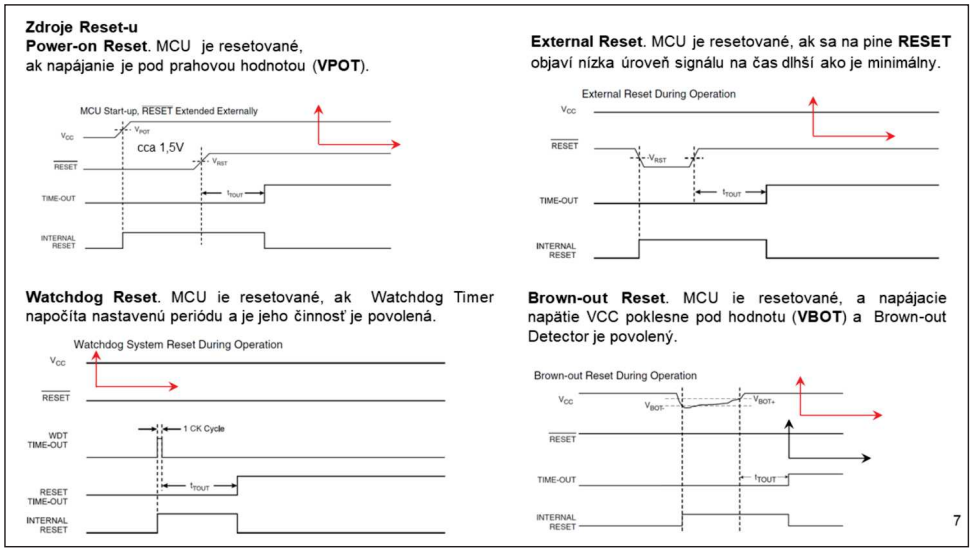
ATmega328 má 4 zdroje Reset-u:



Zdroje Reset-u

1. **Power-on Reset**. MCU je resetované, ak napájanie je pod prahovou hodnotou (**VPOT**).
2. **External Reset**. MCU je resetované, ak sa na pine **RESET** objaví nízka úroveň signálu na čas dlhší ako je minimálny.
3. **Watchdog Reset**. MCU ie resetované, ak Watchdog Timer napočíta nastavenú periódu a je jeho činnosť je povolená.
4. **Brown-out Reset**. MCU ie resetované, a napájacie napätie VCC poklesne pod hodnotu (**VBOT**) a Brown-out Detector je povolený.
5. JTAG AVR Reset.

6



7

Počas Reset-u, sa všetky I/O Register-e nastavia na inicializačné hodnoty (okamžite, ako sa aktivuje signál Reset. Netreba k tomu žiaden hodinový signál) a program sa začne vykonávať od **Reset** vektora. Inštrukcia vektora **Reset** musí byť **JMP** – absolútny skok na inštrukcie obsluhy rutuny Reset.

Ak užívateľský program nepoužije prerušenia, môže byť uložený do pamäte za adresu vektora **Reset**. Atď.

Potom čo sa objaví **Reset**, aktivuje sa oneskorovací čítač. Počas tohto oneskorenia môže napájanie dosiahnuť stabilnú hodnotu, potrebnú k normálnej činnosti. Oneskorenie sa dá nastaviť pomocou **Fuses** - poistiek.

Príznaky RESET sa nulujú zápisom nuly alebo Power-on Reset-om.

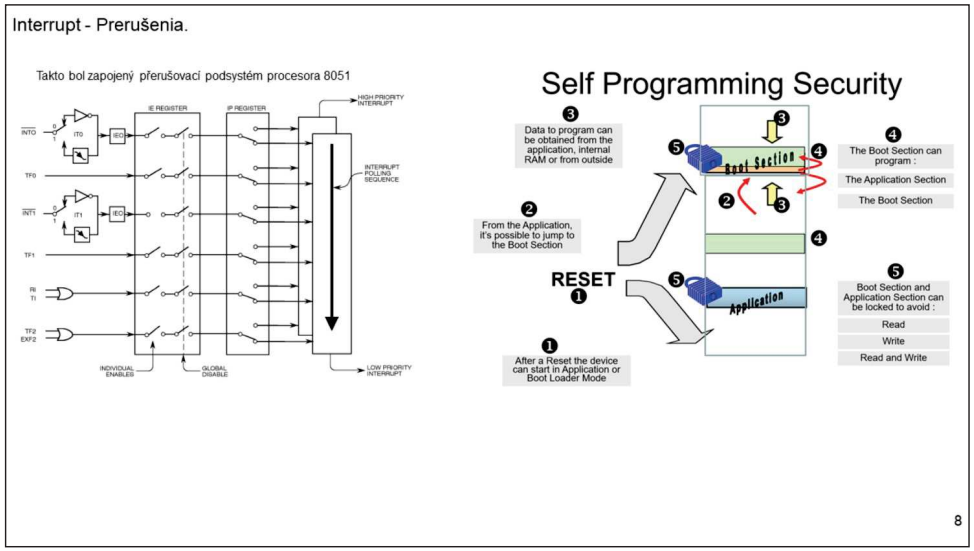
Power-on Reset. MCU je resetované, ak napájanie je pod prahovou hodnotou (**VPOT**).

External Reset musí mať minimálnu dĺžku trvania. Kratšie môžu byť odfiltrované.

Watchdog Reset. MCU ie resetované, ak Watchdog Timer napočíta nastavenú periódu a je jeho činnosť je povolená.

Úrovne registrované Brown-out Detector-om sa dajú zvoliť pomocou BODLEVEL Fuses. Prahová hodnota má hysteréziu:

VBOT+ = VBOT + VHYST/2 and VBOT- = VBOT - VHYST/2. hysterézia je cca 50mV. A typické hodnoty nastavitelné pomocou Fuses sú: 1,8V, 2,7V a 4.3V.



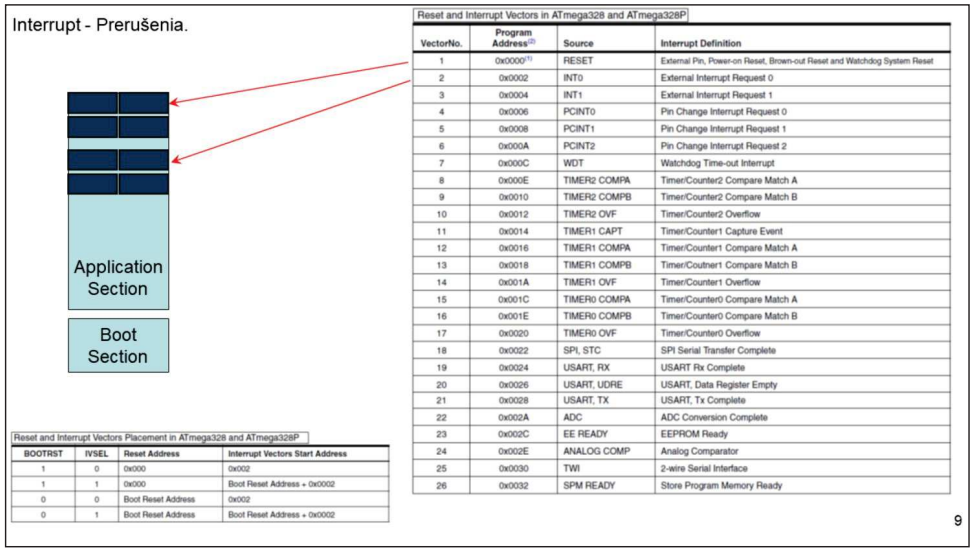
Procesory AVR majú niekoľko prerušení. Každému prerušeniu odpovedá samostatný prerušovací vektor. Každému prerušovaciemu vektoru odpovedá program obsluhy prerušenia. Najnižšie pamäťové miesta v pamäti programu sú definované ako: *Reset* a *Interrupt vektory*. Každé prerušenie má priradený bit pre povolenie prerušenia. Prerušovací podsystém ma pridelený bit (**Global Interrupt Enable** – **GIE** v **Status Register**) pre povolenie, resp. zakázanie prerušení ako celku.

Každý zdroj prerušenia má pridelený samostatný bit, do ktorého musí byť zapísaná jednotka, ak chceme toto prerušenie povoliť.

Všetky zdroje prerušenia môžu byť „automaticky“ zakázané. Závisí to od celkového nastavenia AVR. Najnižšia adresa v pamäti programu odpovedá **RESET**u a potom nasledujú postupne podľa priority (reset má najvyššiu prioritu. Niektoré zdroje ho zaraďujú do kategórie NMI. Nižšia adresa vyššia priorita.) prerušovacie vektory ostatných zdrojov prerušení. Z prerušení má najvyššiu prioritu:

INT0 – External Interrupt Request 0.

Prerušovacie vektory môžu byť presmerované do BOOTFLASH pamäte. Podobne to platí aj pre **RESET**.



Interrupt - Prerušenia.

Application Section

Boot Section

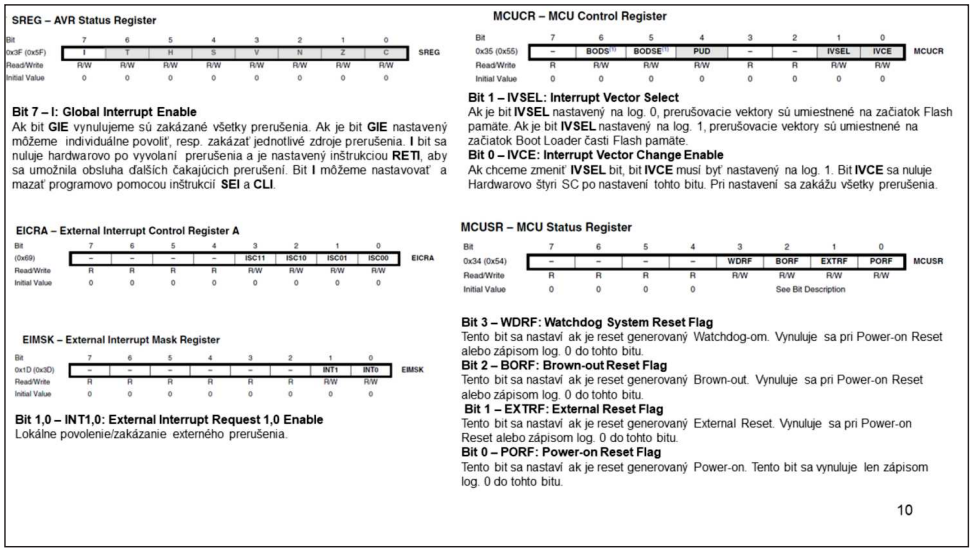
Reset and Interrupt Vectors Placement in ATmega328 and ATmega328P

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x002
1	1	0x000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Reset and Interrupt Vectors in ATmega328 and ATmega328P

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

9



SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – I: Global Interrupt Enable
Ak bit **GIE** vynulujeme sú zakázané všetky prerušenia. Ak je bit **GIE** nastavený môžeme individuálne povoliť, resp. zakázať jednotlivé zdroje prerušenia. **I** bit sa nuluje hardwarovo po vyvolaní prerušenia a je nastavený inštrukciou **RETI**, aby sa umožnila obsluha ďalších čakajúcich prerušení. Bit **I** môžeme nastavovať a mazať programovo pomocou inštrukcií **SEI** a **CLI**.

EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 1,0 – INT1,0: External Interrupt Request 1,0 Enable
Lokálne povolenie/zakázanie externého prerušenia.

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS(1)	BODSE(1)	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 1 – IVSEL: Interrupt Vector Select
Ak je bit **IVSEL** nastavený na log. 0, prerušovacie vektory sú umiestnené na začiatok Flash pamäte. Ak je bit **IVSEL** nastavený na log. 1, prerušovacie vektory sú umiestnené na začiatok Boot Loader časti Flash pamäte.
Bit 0 – IVCE: Interrupt Vector Change Enable
Ak chceme zmeniť **IVSEL** bit, bit **IVCE** musí byť nastavený na log. 1. Bit **IVCE** sa nuluje Hardwarovo štyri SC po nastavení tohto bitu. Pri nastavení sa zakáže všetky prerušenia.

MCUSR – MCU Status Register

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

Bit 3 – WDRF: Watchdog System Reset Flag
Tento bit sa nastaví ak je reset generovaný Watchdog-om. Vynuluje sa pri Power-on Reset alebo zápisom log. 0 do tohto bitu.
Bit 2 – BORF: Brown-out Reset Flag
Tento bit sa nastaví ak je reset generovaný Brown-out. Vynuluje sa pri Power-on Reset alebo zápisom log. 0 do tohto bitu.
Bit 1 – EXTRF: External Reset Flag
Tento bit sa nastaví ak je reset generovaný External Reset. Vynuluje sa pri Power-on Reset alebo zápisom log. 0 do tohto bitu.
Bit 0 – PORF: Power-on Reset Flag
Tento bit sa nastaví ak je reset generovaný Power-on. Tento bit sa vynuluje len zápisom log. 0 do tohto bitu.

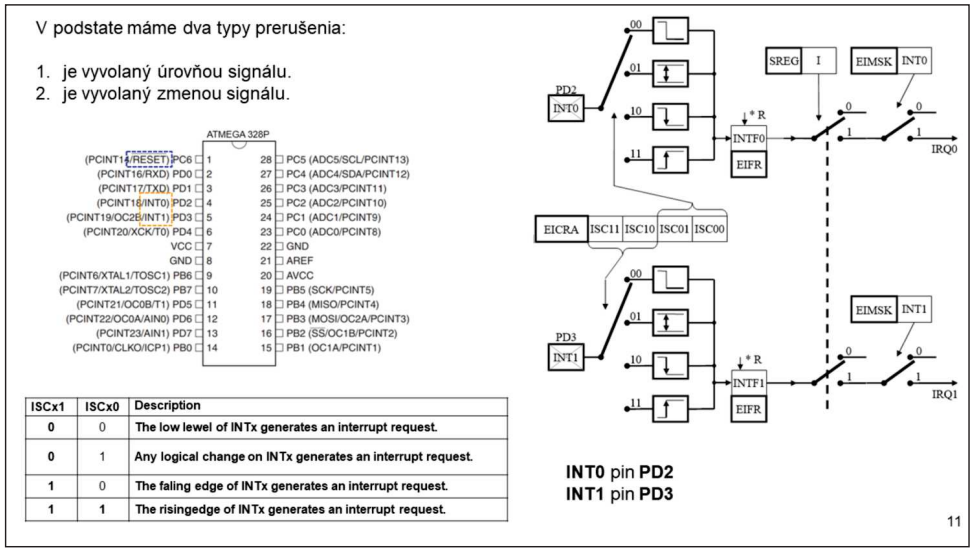
10

Prerušovací vektor 1 má vyššiu prioritu ako prerušovací vektor 2, atď.
Prerušovacie vektory môžu byť posunuté do „Boot Flash section“ nastavením bitu **IVSEL** v registri

MCU Control Register (MCUCR).

Pri obsluhe prerušenia a pri volaní podprogramu sa návratová adresa, obsah registra *-Program Counter* (**PC**) uloží do zásobníka – *Stack*. *Stack* je umiestnený v dátovej pamäti **SRAM**. Veľkosť *Stacku* je limitovaná veľkosť **SRAM** (mínus čosik). Každý užívateľský program musí inicializovať **SP** počas podprogramu obsluhy **Reset**u. *Stack Pointer* - **SP** je umiestnený v **I/O** priestore a možno ho čítať aj doňho zapisovať.

I: Global Interrupt Enable
Ak bit **GIE** vynulujeme sú zakázané všetky prerušenia. Ak je bit **GIE** nastavený môžeme individuálne povoliť, resp. zakázať jednotlivé zdroje prerušenia. **I** bit sa nuluje hardwarovo po vyvolaní prerušenia a je nastavený inštrukciou **RETI**, aby sa umožnila obsluha ďalších čakajúcich prerušení. Bit **I** môžeme nastavovať a mazať programovo pomocou inštrukcií **SEI** a **CLI**.
Externé prerušenia sú vyvolané cez piny **INT0** a **INT1**. *Poznamenajme*, že ak sú tieto prerušenia povolené, vyvolajú sa aj keď odpovedajúce piny budú konfigurované ako výstupné. Táto vlastnosť sa dá využiť pri generovaní *softwarového prerušenia*.
Pri vyvolaní obsluhy prerušenia sa bit *Global Interrupt Enable*, označený ako **I**-bit, vynuluje a všetky ďalšie prerušenia sa zakážu. Používateľ môže vo svojom programe tento bit nastaviť a tým povolí vnorené prerušenia. Všetky povolené prerušenia **môžu** potom prerušiť práve vykonávané prerušenie.



V podstate máme dva typy prerušenia.

1. typ je spúšťaný – zapínaný nejakou udalosťou, ktorá nastaví príznak prerušenia - *Interrupt Flag*.
Tieto prerušenia sú obsluhované vykonananím odpovedajúceho podprogramu obsluhy prerušenia. V okamžiku vstupu do obsluhy prerušenia sa vynuluje požiadavka o obsluhu prerušenia.
Požiadavku o prerušenie - *Interrupt Flag*, môžeme zrušiť zapísaním log. 1 do tohto bitu. Ak vznikne požiadavka o prerušenie v čase, keď je odpovedajúci bit pre povolenie prerušenia vypnutý, bude toto prerušenie zapamätané až do okamžiku povolenia prerušenia. Ak takéto prerušenie chceme zrušiť, musíme ho vynulovať softwarovo.
Obdobne to platí aj pre prípad, keď bit **GIE** je vynulovaný. Keď ho zapneme, nastavené prerušenia sa vykonajú v poradí priority.

2. typ prerušenia bude nastavený tak dlho, ako bude prítomná požiadavka o prerušenie. Takéto prerušenie nemusí mať *Interrupt Flag* (záchytný register). Ak požiadavka o prerušenie zmizne skorej ako sa začne obsluhovať, prerušovací podsystém sa bude správať tak, ako keby táto požiadavka nikdy nenastala.
Pri návrate z prerušenia sa prerušovací podsystém správa tak, ako keby v nasledujúcom kroku neexistovala požiadavka na akékoľvek prerušenie. T.j. vykoná sa jedna inštrukcia „hlavného programu“ a potom sa bude pýtať: Je nejaká ďalšia požiadavka o prerušenie? Ak áno, obslúži prerušenie.
!!!Poznámka!!! *Status Register* nie je automaticky zálohovaný pri vstupe do obsluhy prerušenia a obnovovaný pri návrate z prerušenia. Toto musí zabezpečiť software – teda programátor.
Ak použijeme inštrukciu **CLI** na vypnutie prerušenia, udeje sa to okamžite. Žiadne prerušenie sa nevykoná po inštrukcii **CLI**. Dokonca ani keď sa objaví počas vykonávania tejto inštrukcie.
Existuje niekoľko postupností inštrukcií pri vykonávaní ktorých musí byť prerušenie zakázané: Napr.: počas zápisu do **EEPROM**. Zápis do **EEPROM**, vnútornej, patri medzi tzv. atomické operácie.
Hodnota na pine **INTx** je vzorkovaná pred detekovaním zmeny. Ak hrana alebo prepnutie trvá dlhšie ako 1 **SC** bude generovať prerušenie. Kratšie pulzy nemusia byť zachytené. Ak navolíme prerušenie vyvolané úrovňou, táto musí trvať až do vstupu do obsluhy prerušenia.
Externé prerušenia sú spúšťané pinmi INT0 a INT1 alebo niektorým z pinov PCINT23..0. Upozorňujeme, že ak je povolené, prerušenia sa spustia, aj keď sú piny INT0 a INT1 alebo PCINT23..0 nakonfigurované ako výstupy.

Odozva na prerušenie, AVR

Odozva na požiadavku o prerušenie ktoréhokoľvek prerušenia trvá minimálne 4 **SC**. Po tomto čase sa začne vykonávať obsluha prerušenia.
Počas štyroch **SC** sa uloží obsah **PC** do *Stack*u.
Do **PC** sa zapíše adresa vektoru obsluhy prerušenia a vykoná sa inštrukcia **jump**. Toto trvá 3 **SC**.
Ak sa **interrupt** objaví počas viac cyklovej inštrukcie, táto sa najskôr dokončí a potom sa začne obsluhovať prerušenie.
Ak sa prerušenie objaví počas **SLEEP** módu, odozva sa predĺži o 4 **SC**. Počas tejto doby „*start-up*“ **MCU** zistí z čoho sa to vlastne prebúdza.
Návrat z podprogramu prerušenia trvá 4 **SC**. Počas tejto doby sa vyberú zo zásobníka dva byty – obnoví sa **PC**. **SP** sa inkrementuje o 2 a znovu sa nastaví bit **I** v registri **SREG**. Ak existuje počas inštrukcie **RETI** nejaká neobslúžená požiadavka o prerušenie, najskôr sa vykoná **RETI** potom jedna inštrukcia s hlavného programu a potom sa môže začať realizovať obsluha prerušenia.

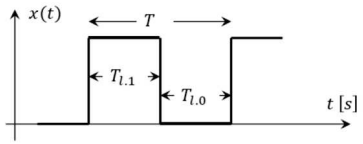
Mikropočítačové Systémy - MIPS

Prednáška 6 – II. Časť
Meranie frekvencie (prietok, rýchlosť)

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

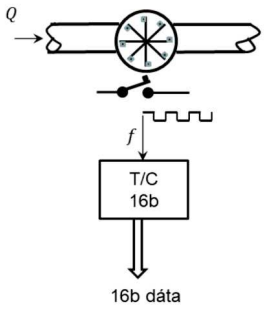
Definície:

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. $\omega = \frac{d\varphi}{dt}$.

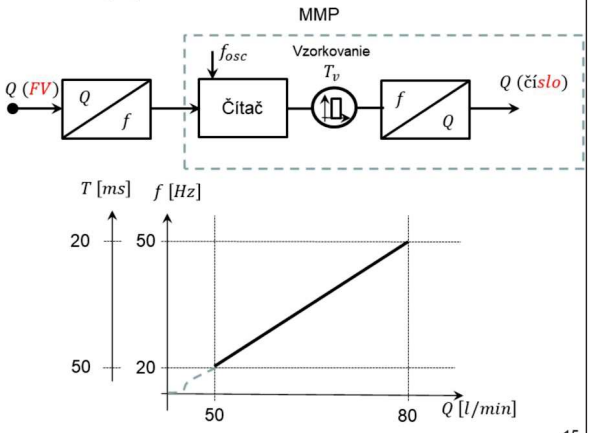


Ak platí $T_{l.1} = T_{l.0}$ potom je plnenie signálu: $pl = 50\%$.

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?

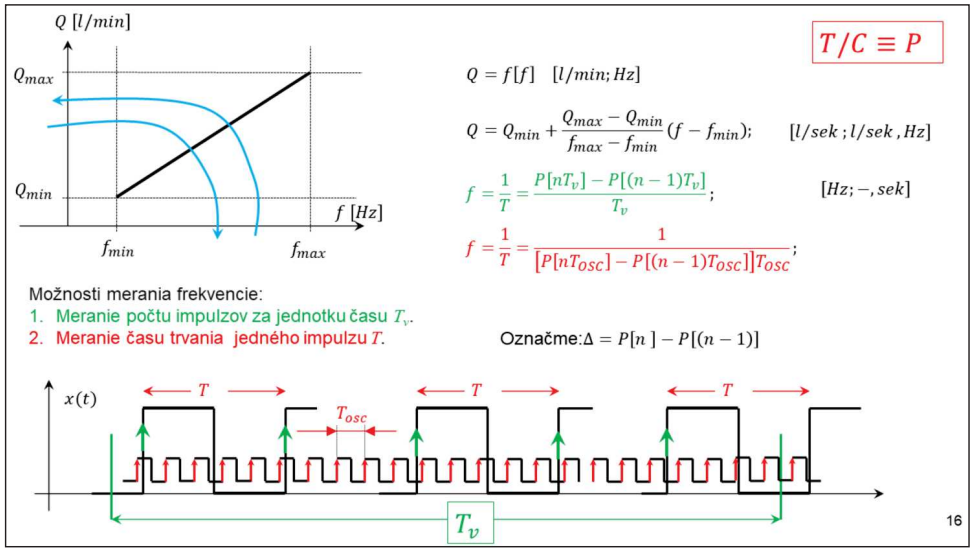


Na výstupe snímača prietoku (objemový, hmotnostný) je frekvencia: f(t) =f(Q(t)).

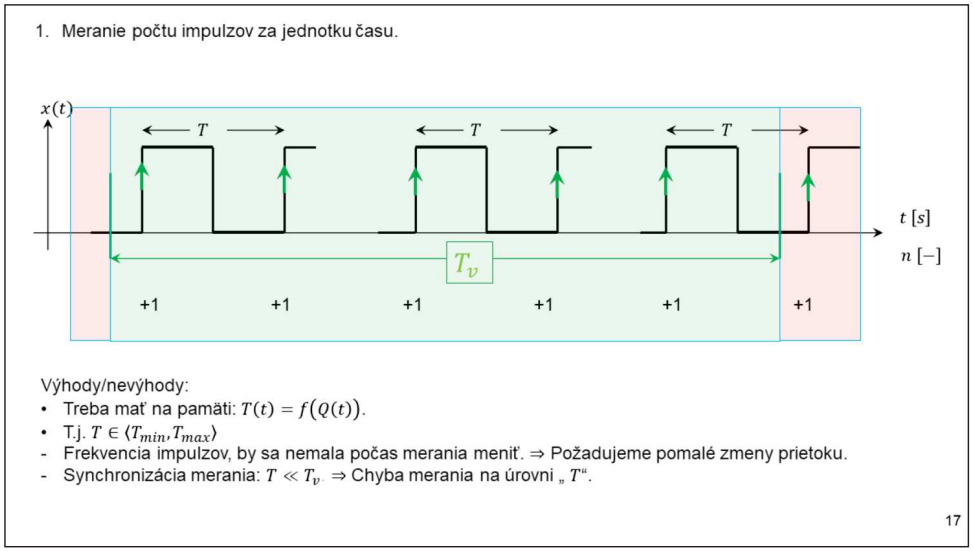
Plnenie je, napr. 50%.

Je zrejmé, že presnosť merania prietoku závisí od presnosti merania frekvencie.

Ak budeme merať frekvenciu s krokom 1Hz, potom budeme merať prietok s krokom 1 l/s.

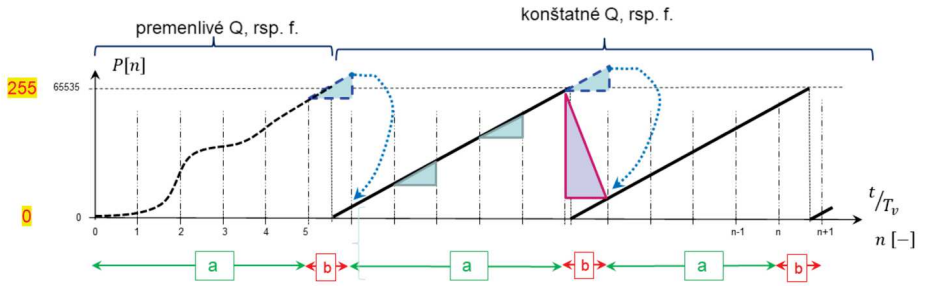


Je zrejmé, že aj v jednom, aj druhom prípade sa dopustíme chyby merania. Impulzy zo snímača („začiatky impulzov“)nie sú synchronné s časovými značkami, resp. so zdrojom frekvencie MMP.



Viac menej je jedno, či za „impulz“ považujeme nábežnú resp. dobežnú hranu. Presnosť sa zvýši, ak budeme načítavať „polovice“ impulzov.

1. Meranie počtu impulzov za jednotku času.



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

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

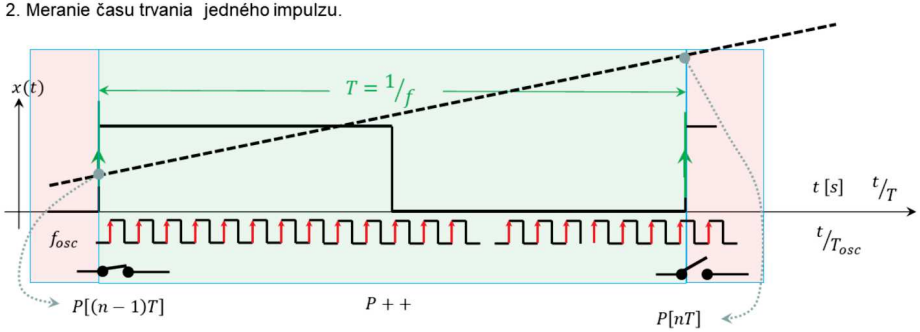
$P[nT_v] - P[(n-1)T_v] < 0$

	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

18

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.

19

Prietok je nulový, resp. kladný. Pre jednoduchosť nech je počítadlo 8b. Budeme stále vyhodnocovať kladný prietok??????

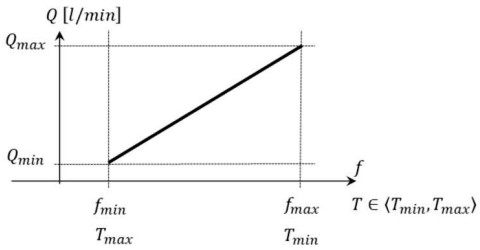
V uvedenom príklade sme deklarovali premenné ako uchar (8b). Rýchlosť je aj v intervale a aj b rovnaká: KLADNÁ.

Opäť použijeme 8b počítadlo, ale deklarujme premenné ako uint.

Výsledok bude rovnaký, aj ked použijeme 16b počítadlo?????

Pri nábežnej hrane odpamätáme stav počítadla.

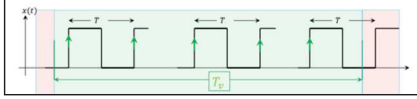
Zhrnutie:
Požadujeme aj v najhoršom prípade presnosť merania lepšiu ako 1%.



1. Meranie počtu impulzov za jednotku času.

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

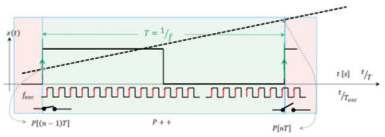
Keďže T_v je konštanta musí platiť:
$T_v > 100.T_{max}$, t.j. $\Delta_{min} > 100$ a súčasne $\Delta_{max} < 2^{16}$



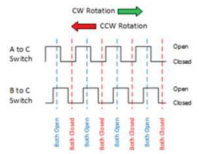
2. Meranie času trvania jedného impulzu.

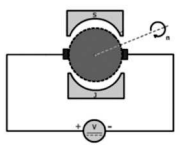
$$T = \Delta.T_{osc}$$

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



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

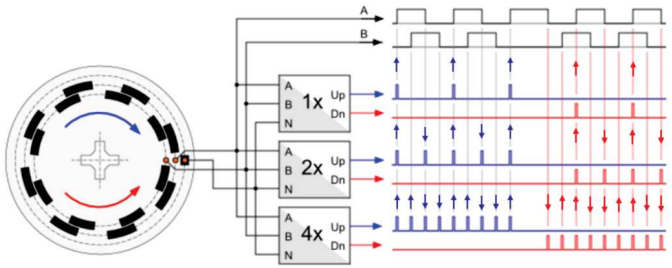


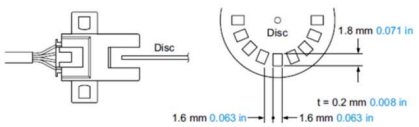


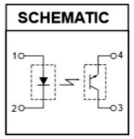
Literatúra:

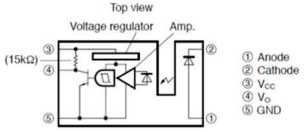
https://www.atpjournal.sk/buxus/docs/atp%20journal%202%202012%20str%2034-35.pdf
http://plc-automatizace.cz/knihovna/data/kodovani/IRC-code.htm
Balogh:
https://senzor.robotika.sk/sensorwiki/index.php/Inkrement%C3%A1lny_sn%C3%ADma%C4%8D
Kozáková & ... :
https://www.researchgate.net/publication/340059022_Robust_QFT-Based_Control_of_the_DC_Motor_Laboratory_Model
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

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









① Anode
② Cathode
③ V_{CC}
④ V_O
⑤ GND

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.