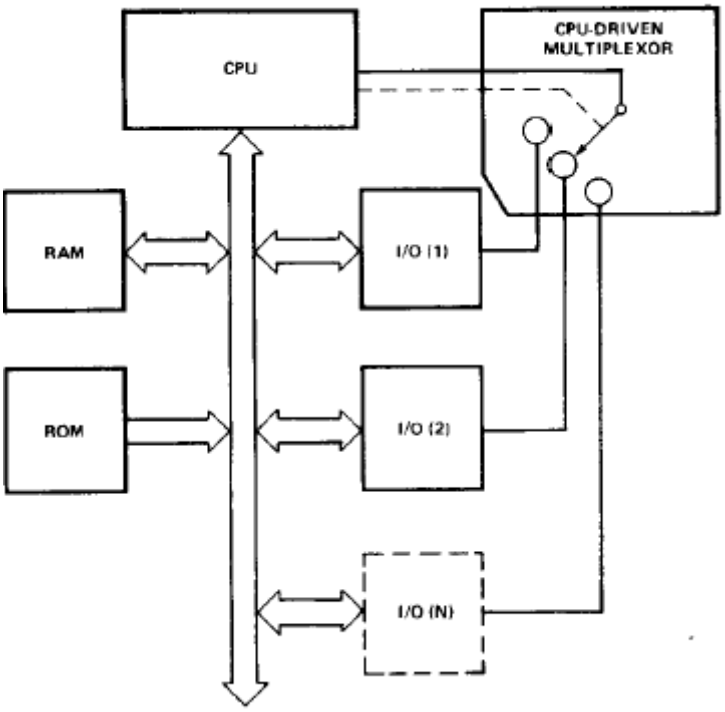
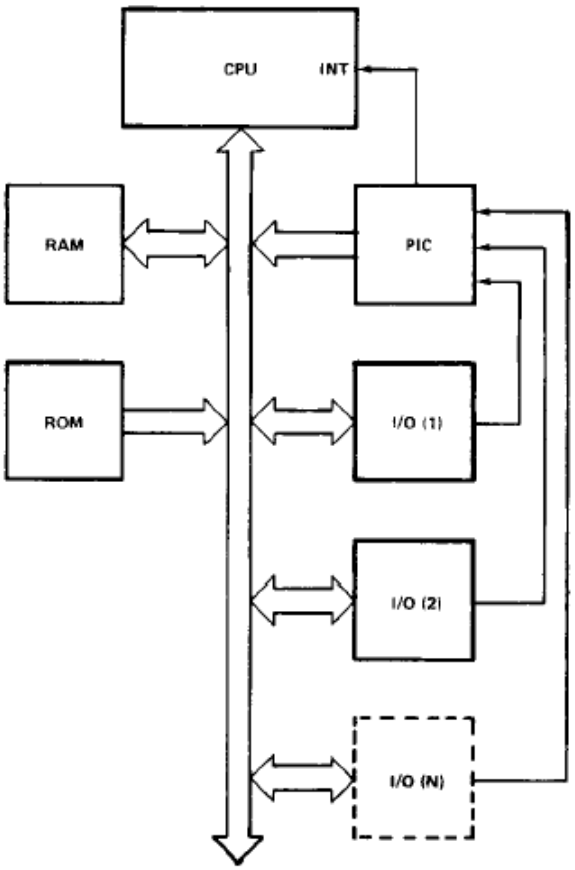
PRERUŠOVACÍ PODSYSTÉM PROCESORA (Všeobecná časť)

Na počiatku bolo všetko jednoduché. Prvé počítače nemali prerušovací podsystém. Postupnosť vykonávania inštrukcií sa riadila pomocou obsahu adresného počítadla. 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, … .



Obr. Spôsob obsluhy periférií: *Polled Method* – opytovacia metóda

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.



Obr. Spôsob obsluhy periférií: Interrupt

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:

- **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.

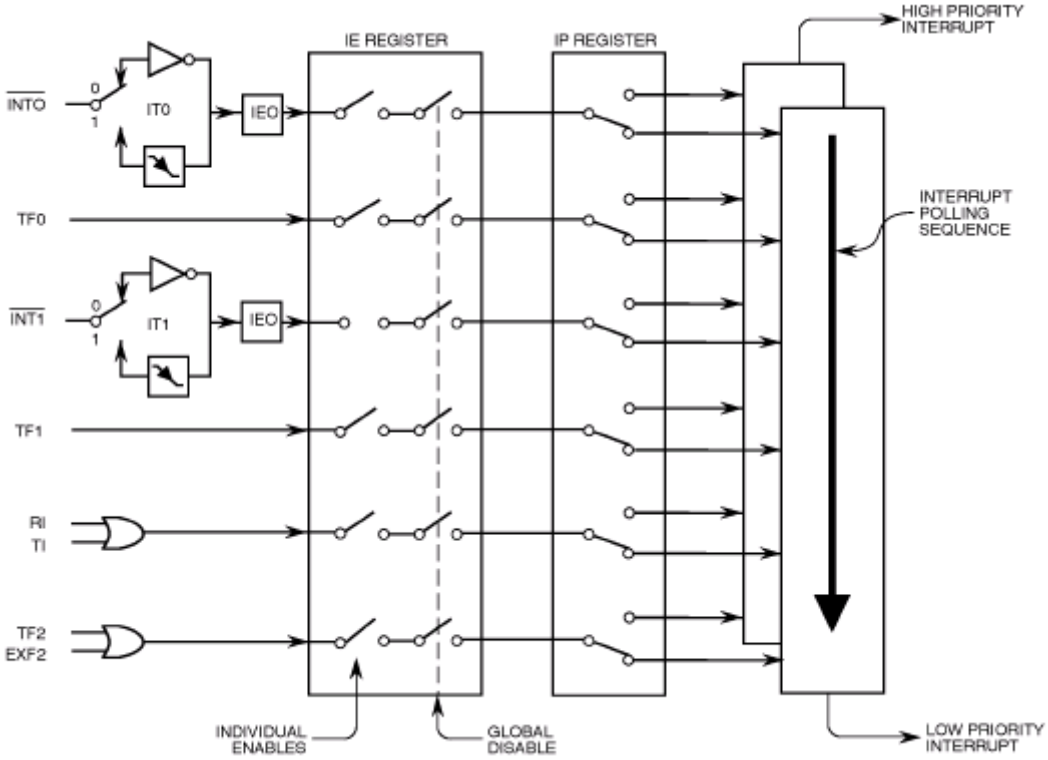
Vnútorné – softwarové. Priamo súvisí s vykonávaným programom a nemôžeme ho zakázať.

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

PIC Programmable Interrupt Controler - Kontrolér prerušenia (8259A/8259A-2)

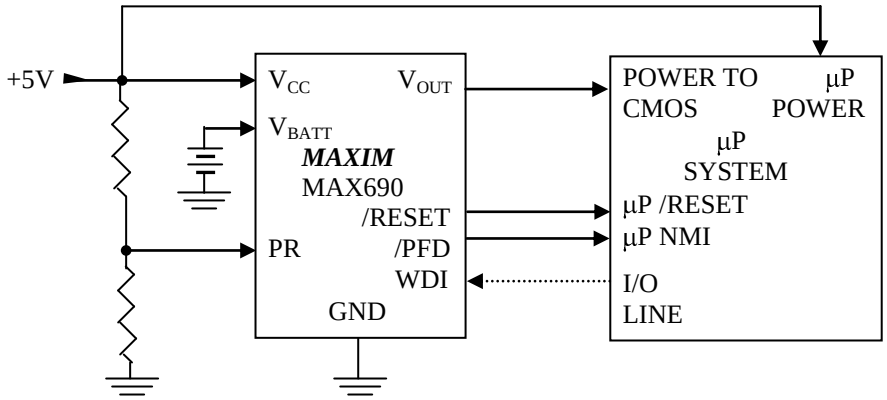
Programovateľný obvod PIC (PROGRAMMABLE INTERRUPT CONTROLLER -Niekedy sa táto skratka číta: Priority Interrupt Controller) tzv. kontrolér prerušenia, obyčajne realizovaný obvodom 8259A, Tento obvod dokáže obsluhovať až 8 zdrojov prerušenia. Možno ho zapájať do kaskády. V maximálnej zostave dokáže obslúžiť až 64 prerušení bez použitia prídavných obvodov. Obvod je programovateľný ako I/O periféria. Programovo sa dá nastaviť priorita. Dá sa meniť počas behu hlavného programu.
V procesoroch je zaradená obdoba tohto obvodu medzi periférie.

Takto bol zapojený přerušovací podsystém procesora 8051



Obr. Prerušovací podsystém 8051

RESET: Vstupný pin



Obr. Resetovacie obvody

Nízka úroveň (dlhšia ako minimálna ... 1,5 us) na tomto vstupe vygeneruje signál **reset**, a to aj vtedy, keď oscilátor nebeží.
Pri vstupe do obsluhy prerušenia a volania podprogramu sa automaticky odloží obsah **Program Counter** (PC) ako návratová adresa do zásobníka - **Stack**. Pre **stack** je alokovaná časť **SRAM**. Počas podprogramu obsluhy **reset**u, sa musí inicializovať **SP** (predtým ako sa objaví volanie podprogramu alebo prerušenie). Stack Pointer SP je read/write a je súčasťou I/O priestoru.

Obsluha Reset-u a Interrupt-u

AVR poskytuje niekoľko zdrojov prerušenia. Prerušenia a reset majú samostatné programové vektory prerušenia v pamäti programu. Všetky zdroje prerušenia v procesoroch AVR sú maskovateľné. Ak chceme povoliť činnosť niektorého zdroja prerušenia, musíme najskôr povoliť **Global Interrupt Enable** v **Status Register**. 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**.

Resetovanie AVR

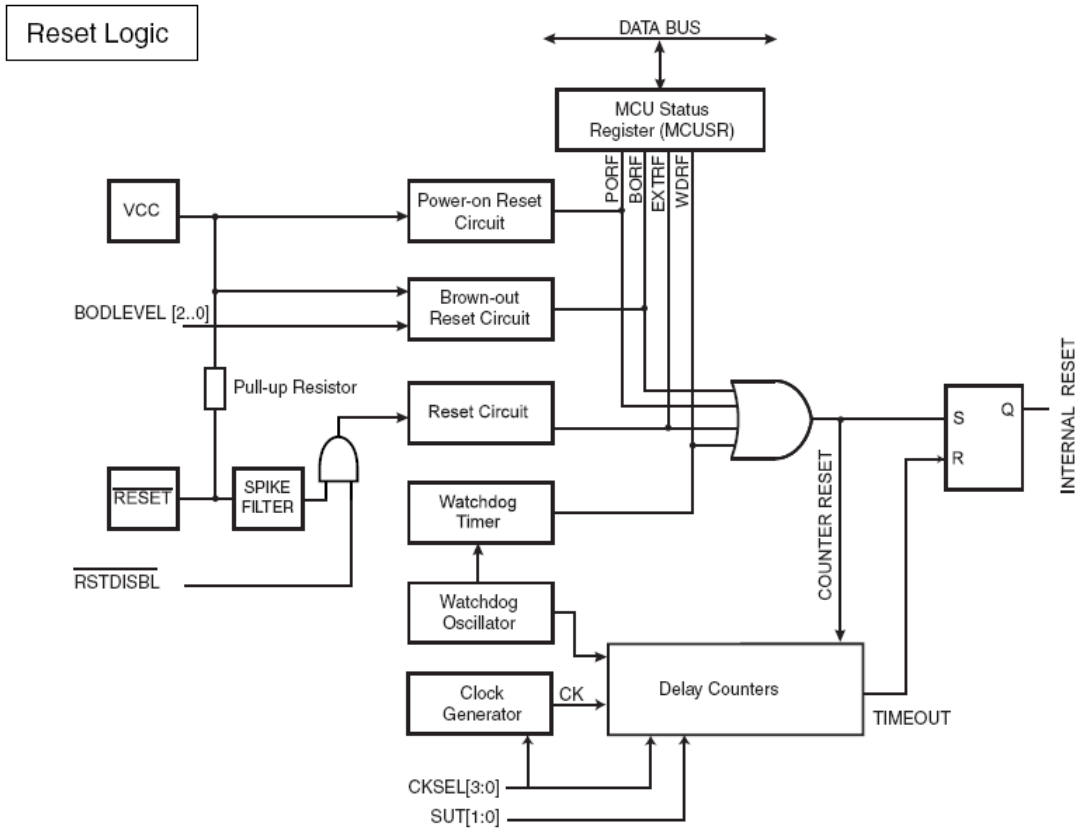
Počas Reset-u, všetky I/O Register-e sa nastavia na inicializačné hodnoty (okamžite, ako sa aktivuje signál **Reset**. Netreba k tomu žiaden hodinový signál) a program sa začne vykonávať od **Reset** vektora. Prvá inštrukcia vektora **Reset** musí byť **JMP** – absolútny skok na inštrukcie obsluhy rutiny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

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

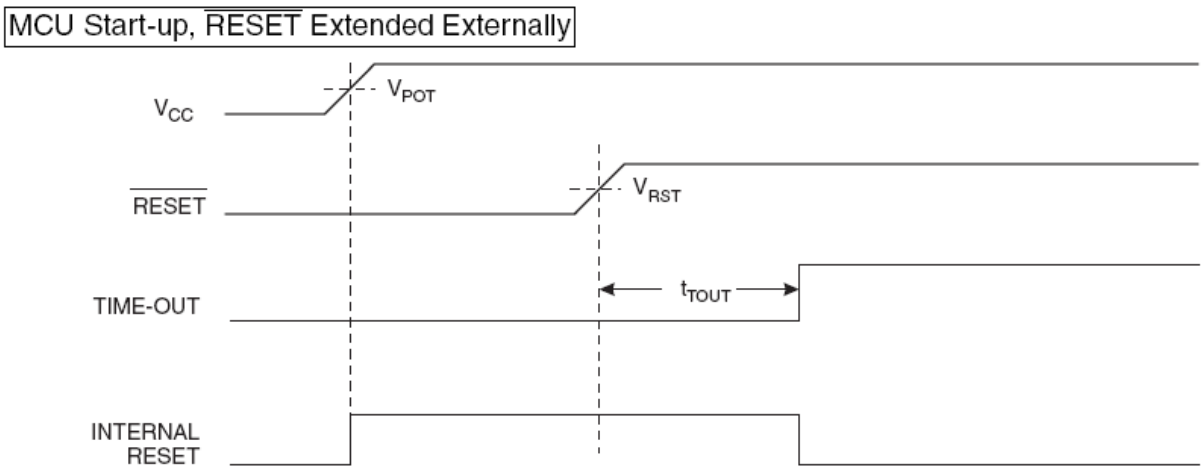
Potom čo sa objaví **Reset**, aktivuje sa oneskorovacie počítadlo. Počas tohto oneskorenia môže napájanie dosiahnuť stabilnú hodnotu, potrebnú k normálnej činnosti. Oneskorenie sa dá nastaviť pomocou **Fuses** - poistiek.

Zdroje Reset-u

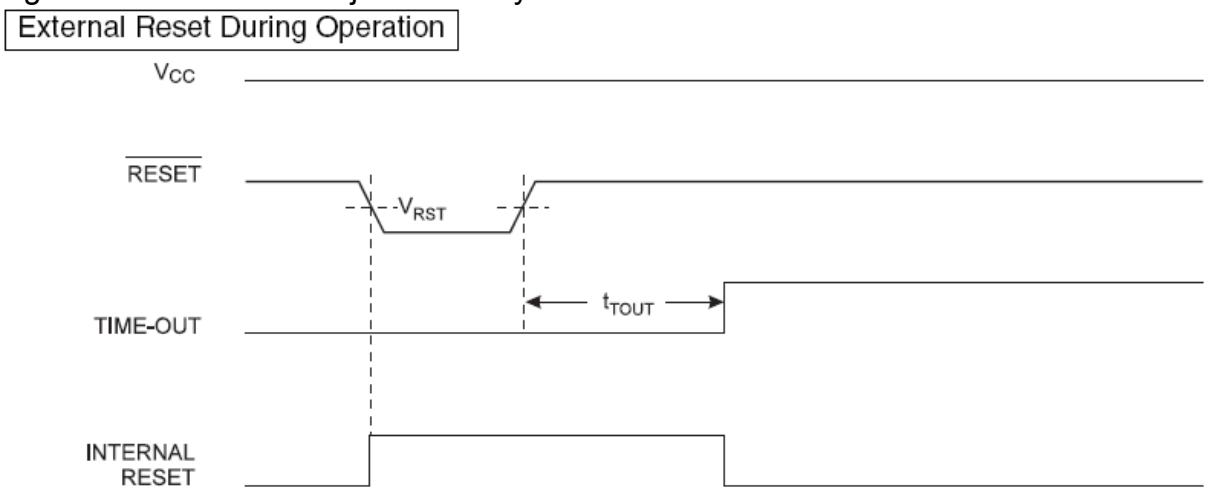
ATmega328 má štyri zdroje Reset-u:



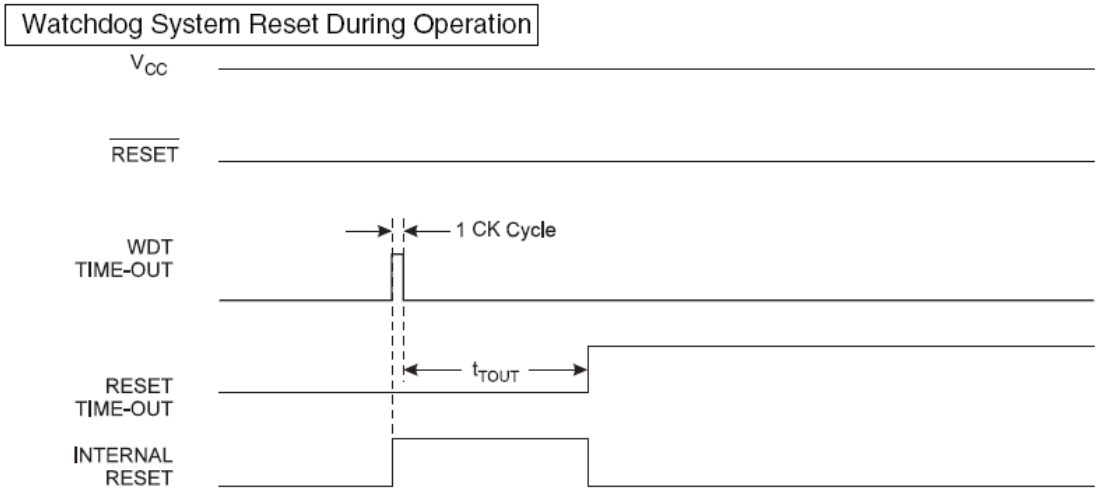
• **Power-on Reset**. MCU je resetované, ak napájanie je pod prahovou hodnotou (**VPOT**).



• **External Reset**. MCU je resetované, ak sa na pine **RESET** objaví nízka úroveň signálu na čas dlhší ako je minimálny.

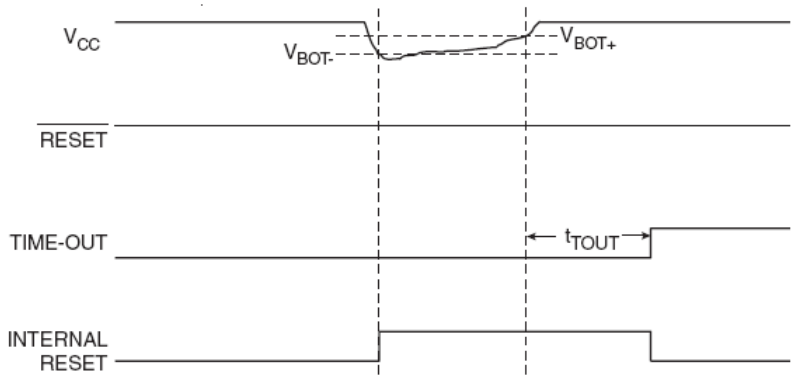


• **Watchdog Reset**. MCU ie resetované, ak Watchdog Timer napočíta nastavenú periódu a je jeho činnosť je povolená.



• **Brown-out Reset**. MCU ie resetované, a napájacie napätie VCC poklesne pod hodnotu (**VBOT**) a Brown-out Detector je povolený.

Brown-out Reset During Operation



V_{BOT} nastavujeme pomocou BODLEVEL fuses

$V_{BOT+} = V_{BOT} + V_{HYST}/2$

$V_{BOT-} = V_{BOT} - V_{HYST}/2$

BODLEVEL Fuse Coding

BODLEVEL 2:0 Fuses	Min. V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
011	Reserved			
010				
001				
000				

V_{HYST} = 50 mV

Prerušenia (Interrupt)

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**) pre povolenie, resp. zakázanie prerušení ako celku.

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

Prerušovací vektor 1 má vyššiu prioritu ako prerušovací vektor 2, atď.
Ak je naprogramovaný bit **BOOTRST** vo Fuse program skáče do „Boot Flash section“
Prerušovacie vektory môžu byť posunuté do „Boot Flash section“ nastavením bitu **IVSEL** v registri **MCU Control Register** (**MCUCR**).

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

MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	-	BODS	BODSE	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 zakážu všetky prerušenia.

MCU Status Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	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	0	0	0	0	

• 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.

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.

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.

I-bit sa automaticky nastaví pri návrate z prerušenia, v okamžiku vykonania inštrukcie **RETI**.

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.

C Code Example

```
char cSREG;
cSREG = SREG;  /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

Ak použijeme inštrukciu **SEI** na povolenie prerušenia, inštrukcia nasledujúca za **SEI** bude vykonaná, a to aj v tom prípade, že bude existovať požiadavka o prerušenie.

C Code Example

```
_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Voľne povedané, inštrukcia nasledujúca po inštrukcii „prerušovacej“ sa k požiadavke o prerušenie správa „atomicky“. Najskôr sa vykoná inštrukcia **SEI**, **RETI**, potom sa vykoná nasledujúca inštrukcia a potom sa začne obsluhovať prerušenie ak je povolené.

Odozva na prerušenie

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 tri **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.

Status Register – SREG :

Bit	7	6	5	4	3	2	1	0	
	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**.

Externé prerušenia

INT0 pin **PD2**
INT1 pin **PD3**

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*.

Externé prerušenia **INT0** a **INT1** môžu byť vyvolané nábežnou alebo dobežnou hranou a nízkou úrovňou.

Poznamenajme, že prerušenia **INT0** a **INT1** vyvolané zmenou, požadujú prítomnosť **I/O** hodinových signálov. Prerušenia I**NT0** a **INT1** vyvolané úrovňou a **INT2** vyvolané zmenou sú vyhodnocované *asynchronne*. Z toho vyplýva, že tieto prerušenia môžu zobúdzať obvod zo *sleep módu* iného ako *IDLE módu*. Vo všetkých *sleep* módoch okrem *IDLE* sa hodiny zastavia.

Ďalej poznamenajme, že ak sa **CPU** zobúdza z ***Power-down*** *módu* úrovňovým prerušením, treba túto úroveň podržať určitý čas - do konca *start-up*u, aby sa **MCU** zobudilo. Toto robí **MCU** menej citlivou na rušenie. Ak podržíme úroveň kratšie, **MCU** sa zobudí, ale prerušenie sa nevyvolá. Čas pre *start-up* sa dá nastavovať pomocou prepojek.

External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	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	

• Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0

Externé prerušenie **INT1** sa aktivuje cez pin **PD3**, ak je nastavený **I**-bit v registri **SREG** a bit **INT1** v registri **EIMSK**.

Hodnota na pine **INT1** 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.
Nastavený režim viď tab.:

ISC11	ISC10	Description
0	0	The low lewel of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The faling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

To isté platí aj pre **INT0**.

External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	INTF1	INTF0	**EIFR**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 1 – INTF1: External Interrupt Flag 1

Hrana alebo zmena úrovne na pine **INT1** nastaví požiadavku o prerušenie. **INT1** sa nastaví do jednotky. Bit sa nuluje pri vstupe do prerušenia. Programovo možno tento bit vynulovať zapísaním log. 1. Ak je prerušenie vyvolané úrovňou, tento bit je stále vynulovaný.

• Bit 0 – INT0: External Interrupt Flag 0

Hrana alebo zmena úrovne na pine **INT0** nastaví požiadavku o prerušenie. **INT0** sa nastaví do jednotky. Bit sa nuluje pri vstupe do prerušenia. Programovo možno tento bit vynulovať zapísaním log. 1. Ak je prerušenie vyvolané úrovňou, tento bit je stále vynulovaný.

External Interrupt Mask Register

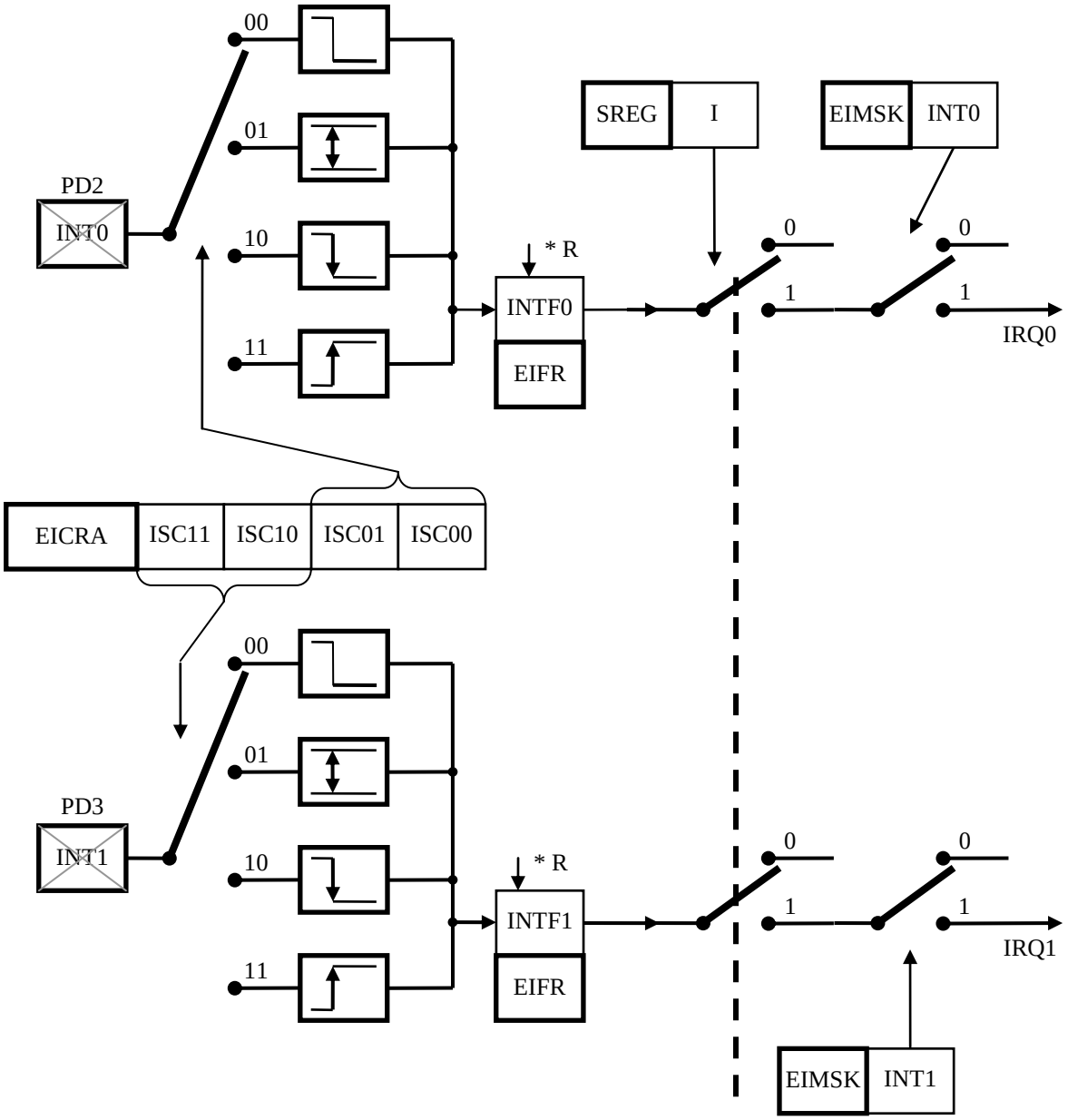
Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	INT1	INT0	**EIMSK**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – INT1: External Interrupt Request 1 Enable

Ak je bit **INT1** nastavený na log. 1, a **I** bit v **SREG** je tiež nastavený na log. 1, potom je povolené externé prerušenie cez pin **PD3**.

• Bit 6 – INT0: External Interrupt Request 0 Enable

Ak je bit **INT0** nastavený na log. 1, a **I** bit v **SREG** je tiež nastavený na log. 1, potom je povolené externé prerušenie cez pin **PD2**.



Samotná obsluha prerušenia, jej zápis, je „funkciou“ samotného prekladača:
Napr.:
AVR – GCC používa dva typy (dve makra) obsluhy prerušenia:

```
INTERRUPT (SIG_ADC)
{
    // tejto zátvorke odpovedá uloženie PC do zásobníka
   // GIE bit je povoleny softwarovo, urobí prekladač
   // sem sa dopíše obsluha prerušenia
   // začneme odložením dôležitých registrov do zásobníka:  PUSH
   // (niektoré prekladače to urobia za nás)
   // samotná obsluha prerušenia
   // ukončíme vybratím, v opačnom poradí, dôležitých registrov zo
   // zásobníka:  POP

} // táto zátvorka odpovedá inštrukcii RETI ,
  //  zo zásobníka sa obnoví PC, a RETI opäť povolí GIE

SIGNAL (SIG_ADC)
{
     // GIE bit je zakázaný vstupom do obsluhy prerušenia, atd...
}
```

pre ImageCraft ICC7AVR je obsluha prerušenia zapísaná nasledovne:

```
#pragma interrupt_handler timer_handler:4
...
void timer_handler()
    {
    ...
    }
```

Poznámka: Zápis podprogramu obsluhy prerušenia nič nerieši: Okrem zápisu podprogramu obsluhy prerušenia treba vykonať:

- Povoliť GIE
- povoliť lokálne prerušenie
- nastaviť „funkčnosť“ periférie, ktorá má prerušenie vyvolať. Napr.:
    - TIMER1:
        - „zhoda“
        - „pretečenie“
    - IRQ: Ak sa jedná o externé prerušenie, napr. INT1, treba nastaviť:
        - prerušenie vyvolané nízkou úrovňou
        - prerušenie vyvolané hranou:
            - nábežnou
            - dobežnou
            - nábežnou/dobežnou