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.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

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 *General Interrupt Control Register* (**GICR**).
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 **resetu**. *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*. 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.
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**.

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) */
```

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**.

Dôležité registre:

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**.

External Interrupts

INT0 pin **PD2**
INT1 pin **PD3**
INT2 pin **PB2**
Externé prerušenia sú vyvolané cez piny **INT0**, **INT1** a **INT2**. 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.
Externé prerušenie **INT2** môže byť vyvolané len nábežnou alebo dobežnou.

Poznamenajme, že prerušenia **INT0** a **INT1** vyvolané zmenou, požadujú prítomnosť **I/O** hodinových signálov. Prerušenia **INT0** 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.

MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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 3, 2 – ISC11, ISC10:** *Interrupt Sense Control 1 Bit 1 and Bit 0*
Externé prerušenie **INT1** sa aktivuje cez pin **INT1**, ak je nastavený **I**-bit v registri **SREG** a bit **INT1** v registri **GICR**.
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 level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

To isté platí aj pre INT0.

MCU Control and Status Register – MCUCSR

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

• **Bit 6 – ISC2: Interrupt Sense Control 2**
Externé prerušenie **INT2** sa aktivuje cez pin **INT2**, ak je nastavený **I**-bit v registri **SREG** a bit **INT2** v registri **GICR**.
Ak je bit **ISC2** vynulovaný prerušenie generuje dobežná hrana. Ak je **ISC2** nastavený na jednotku, nábežná hrana generuje prerušenie. Hrany sú vyhodnocované asynchronne. Pulz na pine **INT2** širší ako 50ns, bude generovať prerušenie. Keď zmeníme **ISC2**, objaví sa prerušenie. Z tohto dôvodu sa doporučuje najskôr vynulovať bit **INT2** v registri **GCIR**. Potom môžeme zmeniť **ISC2** bit. Potom treba softwarovo zrušiť požiadavku o prerušenie zapísaním jednotky do bitu **INTF2** v registri **GIFR**. A nakoniec môžeme opäť povoliť prerušenie.

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	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**.

• Bit 5 – INT2: External Interrupt Request 2 Enable
Ak je bit **INT2** nastavený na log. 1, a **I** bit v **SREG** je tiež nastavený na log. 1, potom je povolené externé prerušenie cez pin **PB2**.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – INTF1: External Interrupt Flag 1
Hrana alebo zmena úrovne na pine **INT1** nastaví požiadavku o prerušenie, **INTF1** 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 6 – INTF0: External Interrupt Flag 0
Hrana alebo zmena úrovne na pine **INT0** nastaví požiadavku o prerušenie, **INTF0** 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 5 – INTF2: External Interrupt Flag 2
Hrana na pine **INT2** nastaví požiadavku o prerušenie, **INTF2** sa nastaví do jednotky. Bit sa nuluje pri vstupe do prerušenia. Programovo možno tento bit vynulovať zapísaním log. 1.

Interrupts

- avr-gcc also provides high level access to the controller's interrupts in <interrupt.h>
- `cli()` to ***disable*** and `sei()` to ***enable*** interrupts globally (contrary to e.g. Motorola HCS12)
- enable interrupts by simply accessing the appropriate registers via `outp()`
- `timer_enable_int(uint8_t interrupts)` may be used to enable timer interrupts specifically

Interrupts (2)

- to declare interrupt handler functions use the following macros:
  `INTERRUPT(interrupt_type);`
  (function is executed with interrupts enabled)
  `SIGNAL(interrupt_type);`
  (executed with interrupts disabled)
- interrupt types are predefined in `<sig-avr.h>`:
  `SIG_OVERFLOW1, SIG_ADC,` etc.