

## Handout Interrupts

Was sind Interrupts?:

- Ist eine Unterbrechungsaufforderung
- Tritt ein bestimmtes Ereignis ein wird er ausgelöst
- Das Programm wird unterbrochen und danach fortgesetzt
- kein detektieren eines Status nötig

Wodurch wird ein Interrupt ausgelöst?:

- An einem bestimmten Eingang ändert sich ein Zustand
- Eine Übertragung ist beendet
- Eine gewählte Zeitspanne wurde überschritten

Einstellen eines Interrupts:

- Maskenbit setzen
- Interruptbearbeitung definieren
- Interrupts global freigeben
- GICR und MCUCR einstellen

### 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	

### MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

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	

IVSEL: Interrupt Vector Select

- 0: Anfang des Flash
- 1: Anfang der Flash Boot Loader Section (gesteuert von den BOOTSZ Fuses)

IVCE: Interrupt Vector Chance Enable

- Möchte IVSEL gesetzt werden so muss IVCE 1 sein

MCUCR bits setzen:

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.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

## Interruptverarbeitung mit Interrupt Service Routine:

- nach fest definierter Reihenfolge
- es darf in Assembler bei z.B. `.org 0x000` keine der Definierung ausgelassen werden

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

### In Assembler:

```
.include „m16def.inc“ #Prozessortyp
definieren
.org 0x000
    rjmp main #resethandler
    rjmp int0_handler
    reti
    ....

main:    rjmp main
int0_handler: cbi PORTB, 0
            reti
```

### In C:

```
#include <avr/interrupt.h>

ISR(INT0_vect){
    //Code für den Interrupt
}
```

## Globales einschalten der Interrupts:

### In Assembler:

```
sei (für on)
cli (für off)
```

### In C:

```
sei();
cli();
Bsp: sei();.....
      cli();
      einlesen();
      sei();
```

## Was muss beachtet werden:

- Interruptroutinen immer so kurz wie möglich halten
- Dürfen im max. so lange dauern wie die kürzeste Dauer des Ereignisses
- In die Interruptzeit immer etwas Puffer einrechnen
- Dauer kann z.b. mit einem Oszilloskop gemessen oder simuliert werden
- weitere Laufzeitprobleme sollten beachtet werden