Using Interrupts (ATmega16)

Dipl.- Ing. Falk Salewski

Lehrstuhl Informatik XI
RWTH Aachen
salewski@informatik.rwth-aachen.de

WS 05/06

Interrupts (IR)

- Interrupts can be used to interrupt the sequential execution of the program flow.
- Interrupt sources can be
  - external events (e.g. change of signal at PORTB2)
  - internal events (e.g. timer overflow)
- In the case of an interrupt event the execution of the main routine is interrupted and the interrupt service routine (ISR) of the according event is executed.
- After executing the ISR the execution proceeds where it has been interrupted.

Example: External IR 2

```
//LEDs at PORTA to 5V & Switch at PINB2(external IR2) to GND
#include <avr/io.h>             //I/O register access
#include <avr/interrupt.h>      //IR-support
#include <avr/signal.h>         //for "SIGNAL" (ISR)

int main (void)
{
 outp(0xFF,DDRA);               //PORTD: output
 outp(0x00,DDRB);               //PORTB: input
 outp(0xFF,PORTB);              //PORTB: Pull-ups aktivated
 cbi(MCUCSR,6);                 //IR2 with falling edge (default)
 sbi(GICR,5);                   //ext. IR2 aktivated          (individual IR activation)
 sbi(SREG,7);                   //Global IR enable            (global IR activation)

 while(1)
 {
       //some main program
 }
}

SIGNAL (SIG_INTERRUPT2) //ISR Interrupt 2; is executed as soon as a falling edge is detected at the according PIN (PORTB2)
{                               //alternative: SIGNAL(_VECTOR(18))
 if(PORTA==0xFF)
 {
   outp(0x00,PORTA);            //Pin=low  => LEDs off
 }
 else
 {
   outp(0xFF,PORTA);            //Pin=high => LEDs on
 }
}
```

Example: Timer0 IR

```
//LEDs at PORTA to 5V blink with 11.444Hz
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

int main (void)
{
 outp(0xFF,DDRA);              //PortA is output
 outp(0xFF,PORTA);             //PortA set to high
 outp(0x00,TCNT0);             //Counter Register = 0
 outp(0x00,OCR0);              //Compare Register = 0
 outp(0x00,TCCR0);             //stop Timer0
 sbi(TIMSK,0);                 //timer0 IR activated
 outp(0x05,TCCR0);             //start Timer0 with prescaler = 1024
 sbi(SREG,7);                  //Global IR enable              (global IR activation)
 while(1)
 {
        //some main program
 }
 return(1);
}

SIGNAL (SIG_OVERFLOW0)   //ISR for Timer0 Overflow
{                        //alternativ: SIGNAL(_VECTOR(9))
        if(PORTA==0xFF)
        {
          PORTA=0x00;
        }
        else
        {
          PORTA=0xFF;
        }
}
```

Interrupt sources: iom16.h

- /* Interrupt vectors, path: C:\Programme\WinAVR\avr\include\avr */

- #define SIG_INTERRUPT0 _VECTOR(1)
- #define SIG_INTERRUPT1 _VECTOR(2)
- #define SIG_OUTPUT_COMPARE2 _VECTOR(3)
- #define SIG_OVERFLOW2 _VECTOR(4)
- #define SIG_INPUT_CAPTURE1 _VECTOR(5)
- #define SIG_OUTPUT_COMPARE1A _VECTOR(6)
- #define SIG_OUTPUT_COMPARE1B _VECTOR(7)
- #define SIG_OVERFLOW1 _VECTOR(8)
- #define SIG_OVERFLOW0 _VECTOR(9)
- #define SIG_SPI _VECTOR(10)
- #define SIG_UART_RECV _VECTOR(11)
- #define SIG_UART_DATA _VECTOR(12)
- #define SIG_UART_TRANS _VECTOR(13)
- #define SIG_ADC _VECTOR(14)
- #define SIG_EEPROM_READY _VECTOR(15)
- #define SIG_COMPARATOR _VECTOR(16)
- #define SIG_2WIRE_SERIAL _VECTOR(17)
- #define SIG_INTERRUPT2 _VECTOR(18)
- #define SIG_OUTPUT_COMPARE0 _VECTOR(19)
- #define SIG_SPM_READY _VECTOR(20)

using Interrupts…

- Keep the ISR short!
- If you change variables: Keep in mind that you could jump to the ISR from everywhere in your program!
- If you want to create a section were no IRs are allowed clear the Interrupt flag in the SREG register (careful!)
- More information about Interrupts in the ATmega16 data sheet.
- The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.