Katkestused e. *interruptid*

Interrupt – signaal (riistvaralise) sündmuse kohta, mis põhjustab spetsiaalselt selle sündmuse jaoks mõeldud koodiosa täitmise

Interrupti kood ehk:

- Interrupt handler
- Interrupt service routine (ISR)

Tavaliselt ütleme selle koodi kohta lihtsalt interrupt või katkestus.

Alternatiivid

Kaks võimalust et tuvastada mõne sündmuse toimumist:

- Kontrollida olekut regulaarselt (nagu while tsüklis pinni kontrollimine)
  - Kulutab asjatult ressurssi
- Interruptid
  - Toimub kohe peale sündmust
  - Võib segada ajakriitilise koodi täitmisele vahele (välditav)

AVRi interrupte

- Sisendpinni signaali muutus
- Loenduri täitumine või väärtuse võrdumine etteantud väärtusega (counter compare match)
- Analoog-digitaal-muunduri töötsükli lõpp
- UARTi andmete vastuvõtmise või saatmise lõpp

Ja palju veel ...

```
#include <avr/interrupt.h>
ISR(MINGI_INTERRUPT_vect)
{
    // Kood mis täidetakse ainult sündmuse toimumisel
}
int main()
{
    // Seadista perifeeria
    sei();              // Interruptide lubamine. VÄGA TÄHTIS!
    while(1)
    {
        // Kood mida täidetakse muul ajal
    }
    return 0;
}
```

Näitekood

- Nupuvajutuse tuvastamine
- LEDi vilgutamine loenduriga

Interrupti tüübid

- Blokeeruvad (vaikimisi)
  - ISR(INTERRUPT_vect)
  - ISR(INTERRUPT_vect, ISR_BLOCK)
- Mitte-blokeeruvad
  - ISR(INTERRUPT_vect, ISR_NOBLOCK)
- “Naked” (sellest räägime hiljem)
  - ISR(INTERRUPT_vect, ISR_NAKED)

Keerulisematel protsessoritel kui AVR on interruptidel prioriteedid.

Ohud

- Liiga pikad ja liiga tihti tulevad interruptid
  - Protsessor täidab pidevalt vaid katkestusi ja ei jõua muuga tegeleda.
  - Mitte-blokeeruva katkestuse ajal kutsutakse uus katkestus. Call stack(pinu, magasin) saab täis. (stack?!)
- Interrupt võib käivituda ajakriitilise koodi jooksul
  - Vältimiseks keelata katkestused selle koodi ajaks

```
cli();
// ajakriitiline kood
sei();
```

- Interrupt võib muuta/lugeda sama (>8bit) mäluvälja mida parajast koodis loetakse/muudetakse. Lahendus sama mis eelmiselgi.

Materjale

http://home.roboticlab.eu/et/avr/interrupts

AVR C library dokumentatsioon interruptide kohta

http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

Wikipedia leht call stacki kohta (natuke rohkem informatsiooni kui AVR-i jaoks vaja)

http://en.wikipedia.org/wiki/Call_stack

Assembly või Assembler

Assembler on madalaima taseme programmeerimiskeel, mille käsud vastavad masinkoodile.

Protsessori ühe takti jooksul täidetakse (üldjuhul) üks Assembleri käsk.

Assembleri kasutusalad

- Ajakriitilise või optimeeritud koodi kirjutamine
- Protsessorispetsiifiliste koodi kirjutamine
  - (DSP-de MAC, x86 prosede SIMD)
- Debugimine.
  - Assembleris näeb mis tegelikult toimub.

Registrisse laadimine

- Tehteid ja võrdlusi tehakse ainult registrites r1-r32 olevate väärtustega

- Registrisse saab laadida käskudega:
  - LDS [mäluaadress], [register]
    - LDS 0x0100, r17
  - LDI [register], arv
    - LDI r16, (1<<PD1)
  - IN [register], [sisend-väljund-register]
    - IN r16, PINB

Kodutöö

Kirjutada Assembleris kasutades 16-bitist loendurit Timer/Counter1 LED-i vilgutamise kood tarkvaralist vaheloendurit kasutamata.

Vihje:

Ületäitumise katkestuse asemel kasutada arvuga võrdumise katkestust. (Output compare).

Materjale

http://home.roboticlab.eu/et/avr/architecture

- Sissejuhatuse AVRi Assemblerisse

http://www.avr-asm-download.de/beginner_en.pdf

- AVR Assembleri juhend

http://www.atmel.com/dyn/resources/prod_documents/doc1022.pdf

- AVRi käsustik andmelehe lõpus