INTRO - Interrupts

Daniel Studhalter

März 2013

Outline

Interrupt Service Routine (ISR)

- Procedure
- Concepts

Priorities (interrupt Rules)

Design Rules

Reentrancy

- Critical Sections

Questions

- Answers

Procedure

1. CPU waits until current command terminates
2. CPU determines the address of the executed ISR
3. return address is pushed onto the stack[1]
4. CPU interrupt mask bit is set
5. CPU jumps to ISR
6. ISR is executed
7. return address is popped from the stack
8. CPU registers are popped from the stack
9. ready for any other interrupt

[1] In many processors are all CPU registers pushed on the stack

Concepts

Interrupt mask

Control bit which stops corresponding interrupts which no longer be handled by the CPU

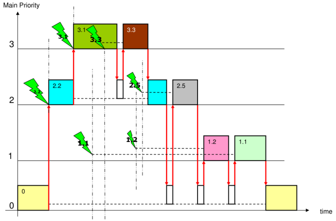
Interrupt latency

Time between interrupt acknowledgment and start the execution of the ISR.

- last command has to terminate
- push registers and return address on the stack
- other ISR is already running

Priorities (interrupt Rules)

Interrupt with higher priority stops a lower one[2]



Remember

- 0 not always lowest priority (Base-Priority)
- main function has always Base Priority (here 0)!

[2] sub priorities may be possible

Design Rules

- less is more!
- Only use for
  - things that can not wait
  - reduction of hardware / software complexity
  - reduction of costs
- BUT
  - debugging is a problem
  - consider number of **required** interrupt sources[3]
  - careful timing analysis

[3] not used → disable

Reentrancy

Reentrancy

A function, subroutine or interrupt can be interrupted at any time (by interrupts) without causing inconsistent conditions or data.

- ISR has the same subroutine as the main program (code-sharing)
- ISR has the same data as the main program (data-sharing)
- Consequence: Each subroutine called by an ISR must be reentrant!

Critical Section

Critical Section

Critical sections can not not be interrupted by another part of the program (also interrupts, as they are turned off when EnterCrititcal ()).

- code in critical section runs in guaranteed order
- increases interrupt latency

Questions

1. What is an interrupt mask?
2. What are the main factors for interrupt latency?
3. What do we understand under code-sharing and data-sharing and in wich context do they appear?
4. Describe how an ISR works?
5. What is a critical section?

Answers

1. What is an interrupt mask?
   Control bit which stops corresponding interrupts which no longer be handled by the CPU.
2. What are the main factors for interrupt latency?
   - last command has to terminate
   - push registers and return address on the stack
   - other ISR is already running
3. What do we understand under code-sharing and data-sharing and in wich context do they appear?
   - ISR has the same subroutine as the main program (code-sharing)
   - ISR has the same data as the main program (data-sharing)
   - Consequence: Each subroutine called by an ISR must be reentrant!

4. Describe how an ISR works?
    4.1 CPU waits until current command terminates
    4.2 CPU determines the address of the executed ISR
    4.3 return address is pushed onto the stack
    4.4 CPU interrupt mask bit is set
    4.5 CPU jumps to ISR
    4.6 ISR is executed
    4.7 return address is popped from the stack
    4.8 CPU registers are popped from the stack
    4.9 ready for any other interrupt
5. What is a critical section?
   Critical sections can not not be interrupted by another part of the program (also interrupts, as they are turned off when EnterCrititcal ()).