Real time Constraints

COE 400

Abdulaziz Almulhem

Final Notes!

- Real time operation
- Polled Interrupts
- Vectors
- Interrupt map
- Interrupt design guidelines

Real time means right now!

- Timing correctness is as important as functional correctness
- Hard real-time and soft real-time
  - Hard: an activity must be completed by a specified deadline. Missing the deadline means tasks fail.
  - Soft: Missing a deadline is acceptable! (bad, better, best response)
- Interrupts are the widely used vehicle to build the timeliness in embedded systems.

Interrupts

- To attend devices
- Fear interrupts because they are difficult to understand.
- A bit of experience (hands on) is all what is needed.
- Polled communication is the simplest.
  - Why should I bother myself, let the CPU take care of me? (what are the problems?)

Polled Interrupts

- Consumes CPU time and power
- Unstructured mess
  - Using ISR
- Leads to highly variable latency

- Use ISR anytime a device can asynchronously require service

Vectoring

- Using processor interrupts
  - CPU finishes current instruction
  - Stack CPU state (smartness of programmer)
  - Attend interrupt
  - Return to top of stack
- Memory is intelligently managed
- Many devices can be handled, each with specific vector
- Interrupt vector is pushed

Interrupt Map

- Provide the maximum worst-case time available to service each interrupt
- It is a budget indicating where interrupting time will be spent

	Latency	Max-time	Freq	Desc
INT1		**1000u**	**1000u**	**timer**
INT2		**100u**	**100u**	**Send data**
INT3		**250u**	**250u**	**Serial data in**
INT4		**15u**	**100u**	**Write tape**
NMI	**200u**	**500u**	**once**	**System crash**

Interrupt design guidelines

- Lay out an interrupt map
- Find the maximum worst-case time
  - Don’t exceed this max otherwise system will not function properly
- Make sure that there is enough time to handle every device
- Approximate the complexity of each ISR and assign priorities based on interrupt rate with how long it will take to service each
  - Highest priority to shorter service

Interrupt design guidelines

- Keep ISRs as short as possible considering time-critical events
  - Short in time and not in code
  - HINT: avoid loops or long complex instruction
- Keep ISRs as simple as possible
  - Chances of error are low
- Start prototyping your interrupt handler
  - Don’t worry about what the handler should do rather focus on having it called properly

Hardware consideration

- Be careful of INTR signal generation and hand shaking
- If system is to handle fast streams of data, then add hardware buffer to supplement the code and don’t forget to read it
- Missing interrupts could make system crash so we some intermediate circuitry to house keep activity of device
  - Such as up/down counter in location encoder

References

1. Berger, Arnold, “Embedded Systems Design:An Introduction to Processes, Tools, & Techniques”, 2002,CMP Books, Kensas, US. ISBN 1578200733
2. Frank Vahid, Tony D. Givargis,” **Embedded System Design : A Unified Hardware/Software Introduction**”, Wiley, 2001 , **ISBN:** 0471386782 (Lecture note).
3. Jack Ganssle, “The Art of Designing Embedded Systems”, Butterworth-Heinemann, 2000, ISBN: 0-7506-9869-1