Slide 1

ECE/CE 3720: Embedded System Design

Chris J. Myers

Lecture 6: Interrupt Synchronization

Slide 2

Introduction

- Interrupts provide guarantee on response time.
- Interrupts allow response to rare but important events.
- Periodic interrupts used for data aquisition and control.
- Interrupts can provide a way to buffer I/O data.

Slide 3

What are Interrrupts?

- An automatic transfer of software execution in response to hardware that is asynchronous with current software.
- Hardware can be external I/O device or internal event.
- When hardware needs service, it requests an interrupt.
- Calls *interrupt service routine* as a *background thread*.
- Thread is terminated with `rti` instruction.
- Threads may communicate using *FIFO queues* and synchronize using *semaphores*.
- Threads share global variables while *processes* do not.
- Each potential interrupt has separate `arm` bit.
- *Interrupt enable bit*, I, found in condition code.

Slide 4

Shared versus Dedicated

(See Figures 4.1 and 4.2)

Slide 5

Shared versus Dedicated

- *Wire-* or *negative-logic* interrupt requests:
  1. Can add additional I/O devices w/o redesigning H/W.
  2. No limit to number of interrupting I/O devices.
  3. Microcomputer hardware is simple.
- *Dedicated edge-triggered* interrupt requests:
  1. Software is simpler, easier to debug, and faster.
  2. Less coupling between software modules.
  3. Easier to implement priority.

Slide 6

Interrupt Service Routines (ISR)

- Software executed when hardware requests an interrupt.
- *Polled interrupts* - one large ISR handles all requests.
- *Vectored interrupts* - many small, specific ISRs.
- When the device is armed, the *I* bit is zero, and an interrupt is requested, it is serviced as follows:
  1. Execution of main program is suspended.
  2. All registers are pushed onto the stack.
  3. The ISR, or background thread, is executed.
  4. The ISR executes `rti` instruction.
  5. All registers are restored from the stack.
  6. The main program is resumed.

Slide 7

Interrupt Execution

(See Figure 4.3)

Slide 8

When to Use Interrupts

Cadfly	Interrupts	DMA
Predictable	Variable arrival times	Low latency
Simple I/O	Complex I/O	High bandwidth
Fixed load	Variable load	
Single thread	Multithread	
Nothing else to do	Infrequent alarms	
	Program errors	
	Overflow, illegal op	
	Illegal memory access	
	Machine/memory errors	
	Power failure	
	Real-time clocks	
	Data acquisition/control	

Slide 9

Interthread Communication

- Interrupt threads have logically separate registers/stack, so communication must occur through global memory.

(See Figure 4.4)

Slide 10

Input Device Interrupts

(See Figures 4.5, 4.6, and 4.7)

Slide 11

Output Device Interrupts

(See Figures 4.8, 4.9, and 4.10)

Slide 12

Other Interrupt Issues

1. Periodic interrupts are neither input or output.
2. Essential for data acquisition and control systems.
3. ISR should only occur when needed, come in clean, perform function, and return right away.
4. Gadfly loops and iterations should be avoided in ISRs.
5. Percent of time in ISRs should be minimized.
6. *Interface latency* is time between new input available and when software reads the input data.
7. *device latency* is response time of external I/O device.
8. A *real-time system* guarantees bound on interface latency.