COMP 551:
Advanced Robotics Lab

Lec09: Embedded Systems

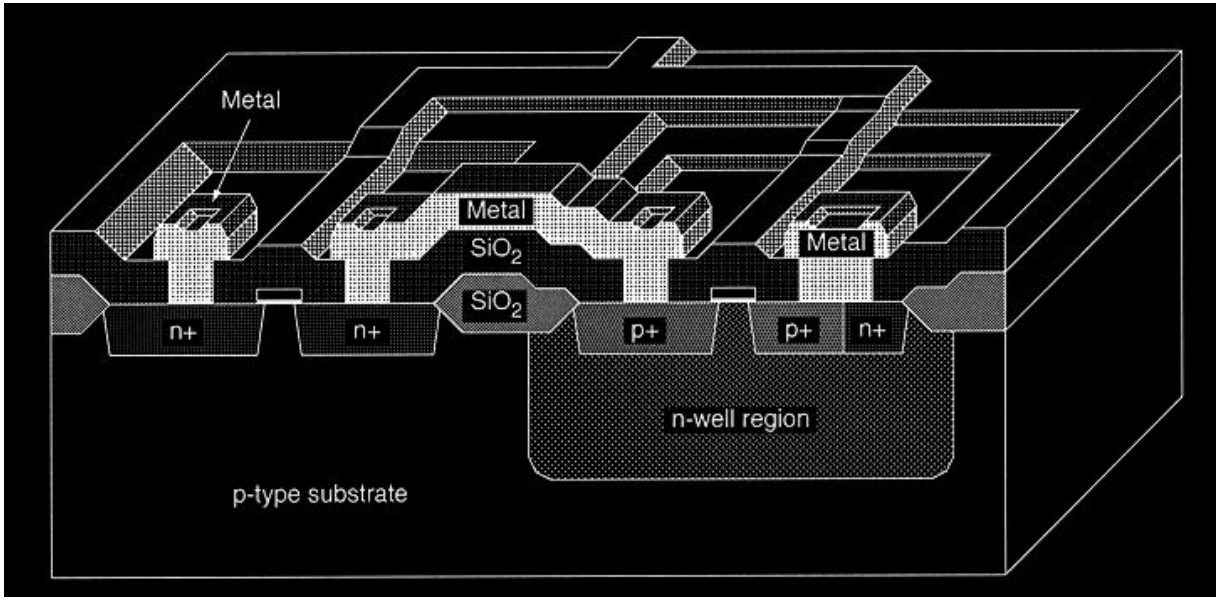
James McLurkin
Rice University
jmclurkin@rice.edu

2. Why Develop on the “Metal”?

What is “Metal”?

Why develop here?

1. Your system is constrained by power, space, or budget
2. You have “Hard Real Time” constraints
3. You have an obsessive-compulsive need to control every aspect of your processor



Why Develop on the “Metal”?

System Constraint: Consumer electronics

- (This slide is from 2005: A lot has changed since then…)

Why Develop on the “Metal”?

Hard Real Time:
Airbag
Deployment

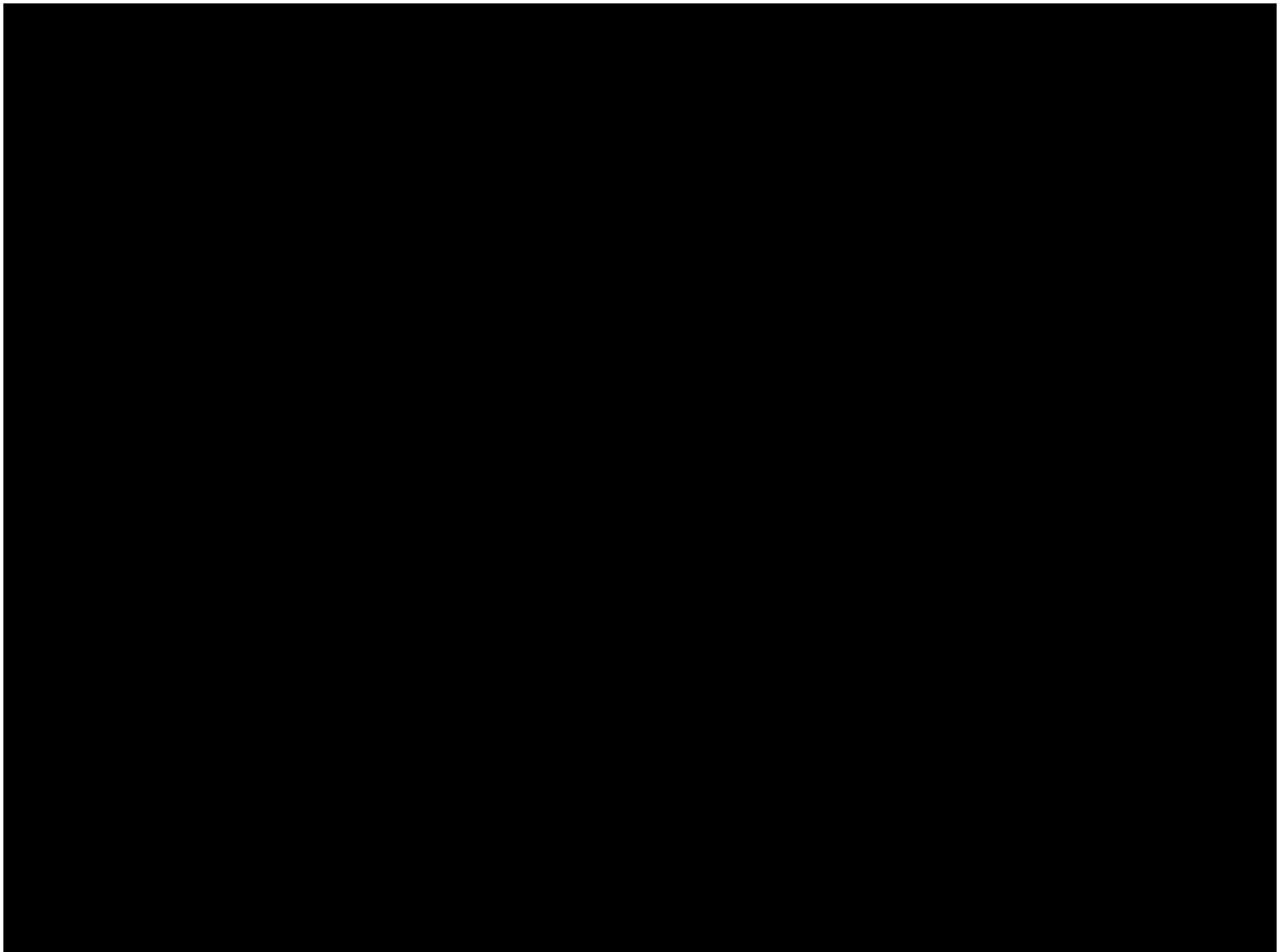
http://www.motionengineering.com/slow_motion_video_archive2.cfm

Why Develop on the “Metal”?

Hard Real Time: Supersonic Fighter Jet Control Systems

Why Develop on the “Metal”?

System Constraint: Tightly integrated robots

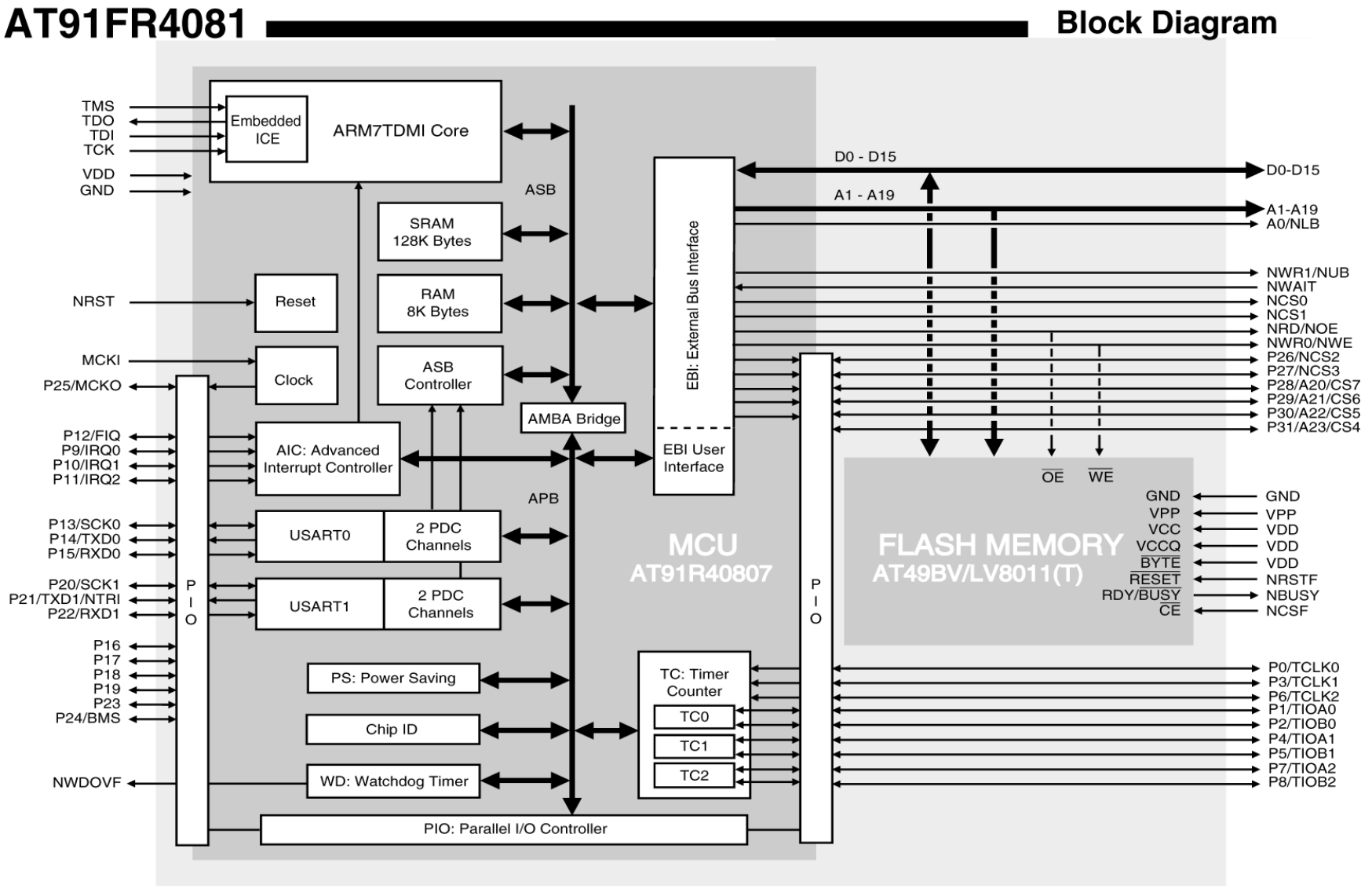


3. System Architecture

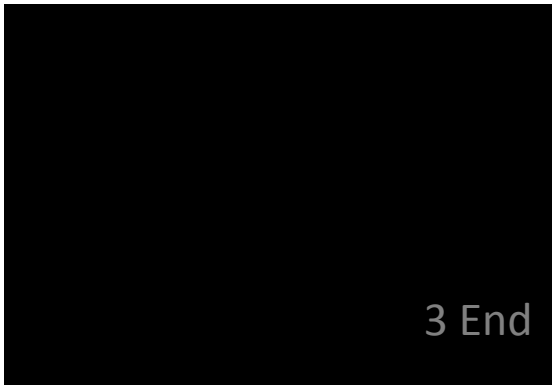
Life is different at the bottom of the food chain…

- Microcontrollers
- User Interfaces

The Microcontroller



The User Interface


3 End

4. Hard Real Time Software

What is “Hard Real Time” Software?

- But my system is I/O bound, and my processor is not working very hard. Do I still need to worry about worst-case performance?

What is “Soft Real Time” Software?

In order to make my software run, I need a “kernel” of code to decide what my processor will do at any given time. Ideally, we can design this program to place a bound on our max latency.

My First Scheduler

```
main() {
  while(true) {
    readSingleCharFromSerialPort(); (3usecs/char)
    readAndProcessSensors(); (10ms ±5ms)
    controlMotor(); (10us)
  }
}
```

The r-one serial port runs at 230 kbps = 43usecs/char
The serial port hardware does not buffer chars

Motor control wants to run at 500hz

Does this code guarantee these constraints?

My Second Scheduler

```
main() {
  while(true) {
    readSerialPortSoftwareBuffer(); (1us)
    readAndProcessSensors(); (10ms ±5ms)
    controlMotor(); (10us)
  }
}

interrupt serialPortHandler() {
  readSerialPortHardwareBuffer()
  writeSerialPortSoftwareBuffer();
}
```

The r-one serial port runs at 230 kbps = 43usecs/char
The serial port hardware does not buffer chars

Motor control wants to run at 500hz

Does this code guarantee these constraints?

My Third Scheduler

```
main() {
  while(true) {
    readSerialPortSoftwareBuffer(); (1us)
    writeMotorCommand(); (1us)
    readAndProcessSensors(); (10ms ±5ms)
  }
}

interrupt serialPortHandler() {
  readSerialPortHardwareBuffer();
  writeSerialPortSoftwareBuffer();
}

interrupt timerHandler() {
  readMotorCommand();
  controlMotor();
}
```

The r-one serial port runs at 230 kbps = 43usecs/char
The serial port hardware does not buffer chars

Motor control wants to run at 500hz

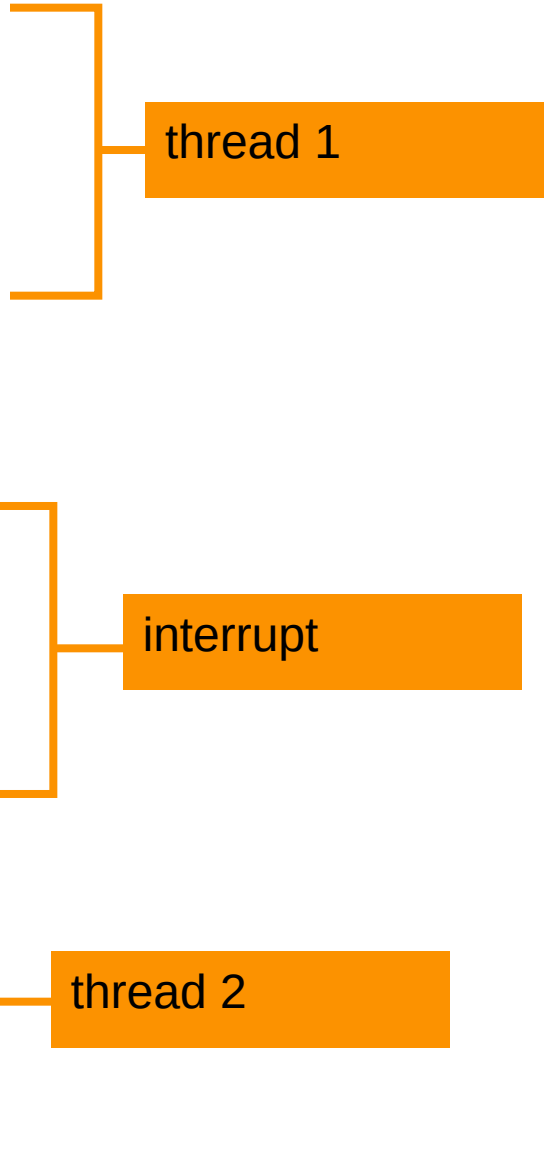
Does this code guarantee these constraints?

My Third Scheduler

```
main() {
   while(true) {
      readSerialPortSoftwareBuffer();
      writeMotorCommand();
      readAndProcessSensors();
   }
}

interrupt serialPortHandler() {
   readSerialPortHardwareBuffer();
   writeSerialPortSoftwareBuffer();
}

interrupt timerHandler() {
   readMotorCommand();
   controlMotor();
}
```



Glossary Summary

Kernel

Scheduler

Interrupt

Thread

Context Switch

Shared Memory

Message Queue

Process

Don’t Write a Scheduler!

Buy one!

THREADX®

Nucleus RTOS

VxWorks

realtime operating system

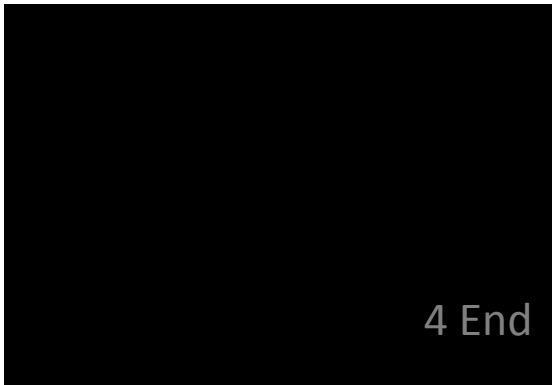
QNX Neutrino Realtime Operating System

Since 1980, manufacturers have relied on QNX OS technology to power their mission-critical applications — everything from medical instruments and Internet routers to telematics devices, 9-1-1 call centers, process control applications, and air traffic control systems. Small or large, simple or distributed, these systems share an unmatched reputation for operating 24 hours a day, 365 days a year, nonstop. Time-tested and field-proven, the QNX® Neutrino® realtime operating system (RTOS) sets the industry standard for reliability, fault tolerance, and scalability.

What makes QNX Neutrino so remarkable? It's a true microkernel operating system. Under QNX Neutrino, every driver, application, protocol stack, and file

More from this section

- QNX Neutrino - at a Glance
- Microkernel Architecture
- Realtime Performance
- POSIX Support
- Power Management Framework
- Symmetric Multiprocessing
- Instrumented Kernel
- Critical Process Monitoring
- Transparent Distributed Processing
- Networking Technologies
- File Systems
- Resource Manager Framework


4 End

Programming Multi-Threaded Systems

Why Threads? Why not Processes?

IPC = InterProcess(or) Communications

Simple Languages

- Assembly, C, C++
- No Java, yet.

Processor Limitations

- No FPU (Floating Point Unit)
- No MMU (Memory Management Unit)

Static Memory Allocation