Chapter 1: INTRODUCTION

Manuel Jiménez

EMBEDDED SYSTEMS:

Theory and Applications Using the MSP430

UPRM - Spring 2010

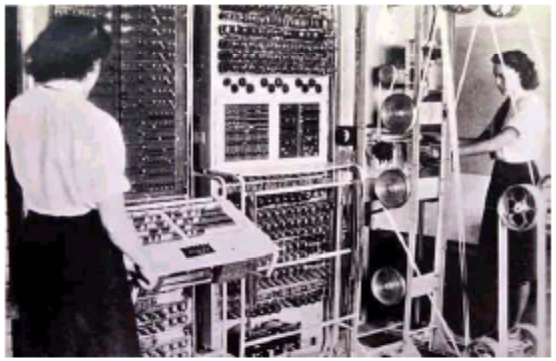
Chapter Outline

- Embedded Systems Overview
  - Early Forms
  - Birth and Evolution of Modern Systems
  - Contemporary Systems
- System Structure
  - Hardware Components
  - Software Components
- System Classification
- Life Cycle of Embedded Designs
- Design Constraints
  - Functionality
  - Cost
  - Performance
  - Power and Energy
  - Time-to-Market
  - Reliability & Maintainability
- Summary

Embedded System Definition

- Electronic systems containing tightly coupled hardware and software components
  - Perform a single function
  - Form part of a larger system
  - Are not intended to be independently programmable by the user
  - Are expected to work with minimal or no human interaction
  - Reactive, Real-time Operation
  - Tightly Constrained

Early Embedded Systems



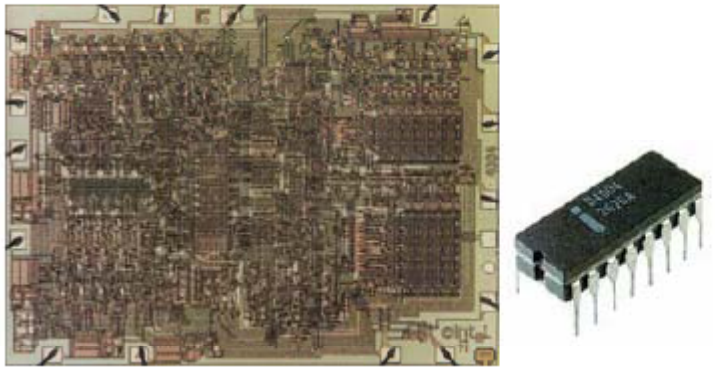
Colossus Mark II

Apollo Guidance Computer

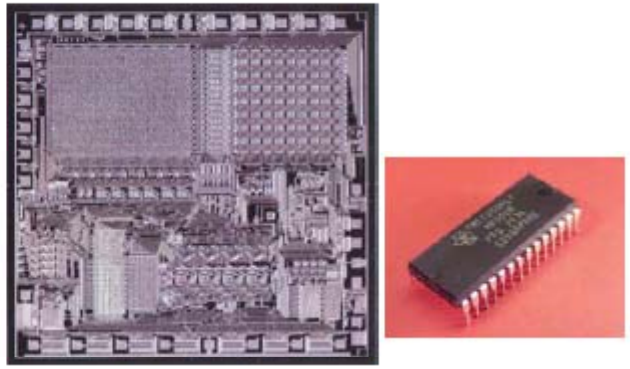
- Early Computers
  - Similar to an ESYS
    - Single functioned
    - Not user programmable
  - Unlike Today's ESYS
    - Large and power thirsty
    - Not integrated
- First Modern ESYS
  - The Apollo Guidance Computer (AGC)
    - Guidance & Navigation
    - CPU+MEM+I/O
    - Assembly Programmed

Modern Embedded Systems

- Born with the Microprocessor
  - TMS1000: The first microcontroller
  - Intel 4004: The first commercial microprocessor
  - US Navy CADC: High-performance embedded system



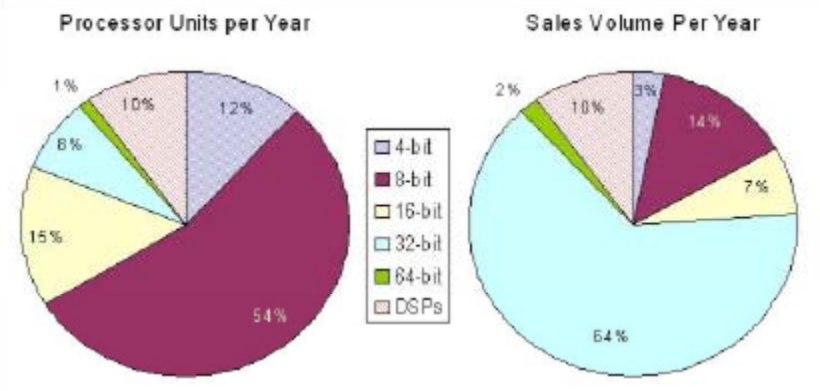
Intel 4004



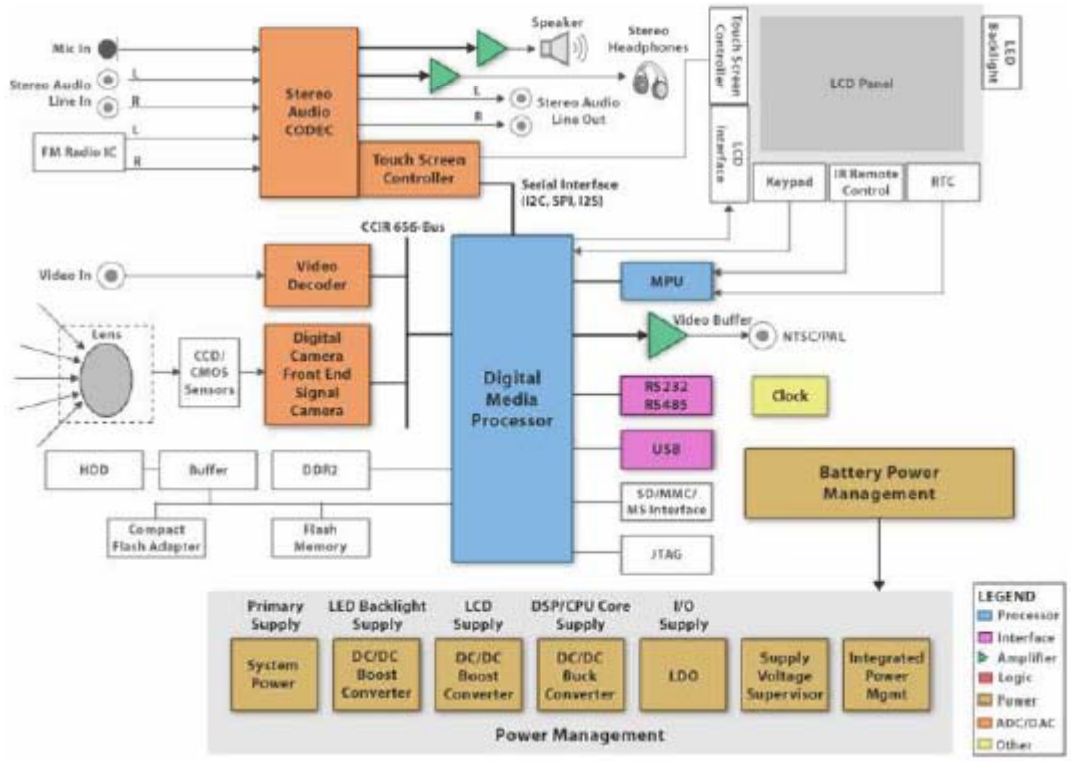
TI's TMS1000

Today's MCU Market

- Plenty of Vendors
  - TI (MSP430)
  - Microchip (PIC)
  - Intel (8051, 80x86)
  - Freescale (HC11, HC08)
  - Zilog (Z80, Z8000)
  - ARM Ltd. (ARM7)
  - Natl. Inst (COP)
- Plenty of Sizes
  - 4-bit, 8-bit, 16-bit, 32-bit, 64-bit
  - CISC Vs. RISC
  - Harvard Vs. vonNeumann
- Wide Market
  - Over 6 Billion chips per year
  - Nearly $50 billion sales



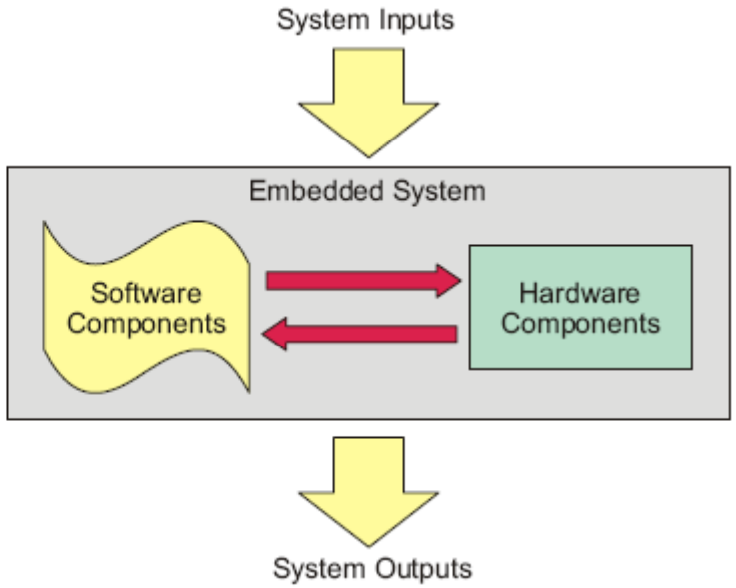
Contemporary Systems



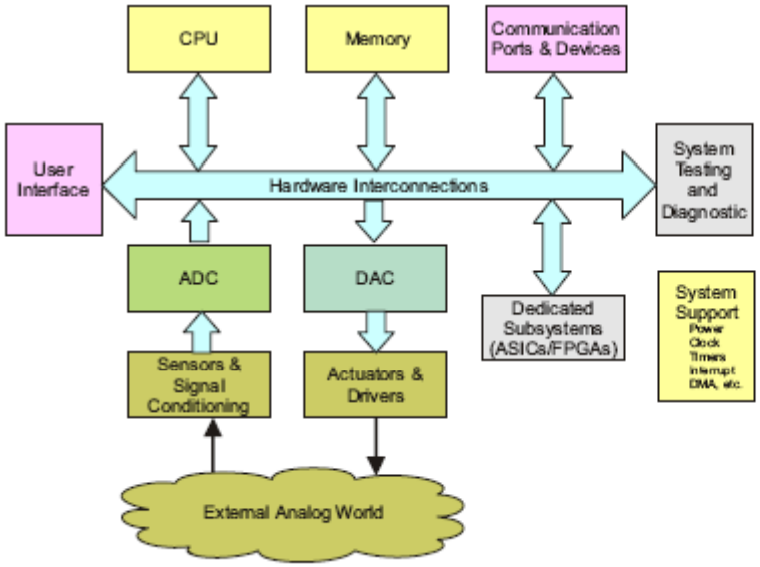
Media Player Example

Embedded System Structure

- Hardware Components: Electronics Infrastructure
  - CPU
  - Memory
  - I/O Subsystem
- Software Components: System Functionality
  - Firmware
  - Operating System
  - Application Programs

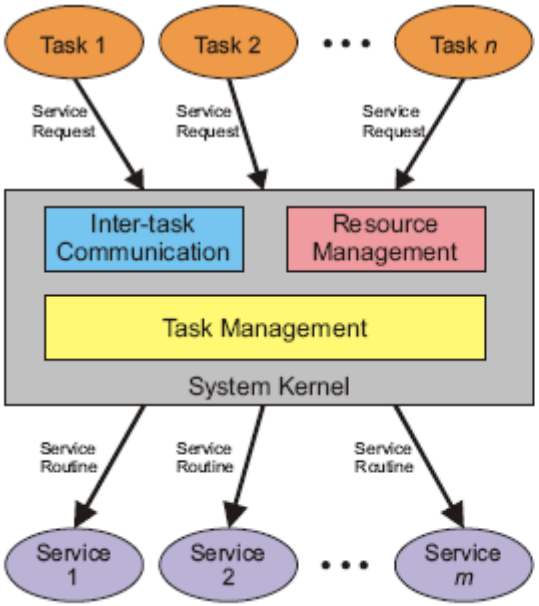


Hardware Components



- Central Processing Unit
  - Registers, ALU, CU
- Memory
  - Program Memory
  - Data Memory
- I/O Devices
  - Communication Ports
  - User Interfaces
  - Sensors & Actuators
  - Diagnostics Support
  - System controllers
  - Dedicated Hardware

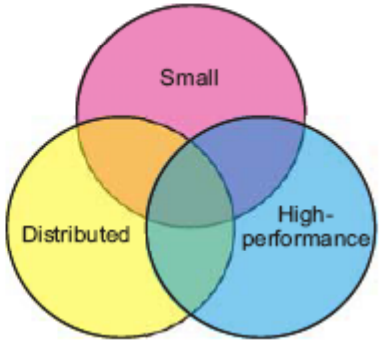
Software Components



- System Tasks
  - Actions making use of system resources
- System Kernel
  - Component managing system resources
  - Coordinates task services
- Services
  - Routines performing specific tasks

Types of Embedded Systems

- Small
  - MCU-based, low component count
  - Large volume
  - Single tasked
  - Low-cost, maintenance free
- Distributed
  - Multi-chip, board-level
  - Multi-tasked
  - Medium volume & cost
  - Maintainable, upgradeable
- High-performance
  - Dedicated board-level hardware
  - Task intensive, RTOS-based
  - Low-volume, high cost
  - High maintenance



Embedded Systems Life Cycle

- Birth
  - Need & opportunity
  - Specifications
- Design
  - Proof-of-concept
  - Manufacturing design
- Growth
  - Production & deployment
- Maturity
  - Maintenance & upgrade
- Decline
  - System disposal

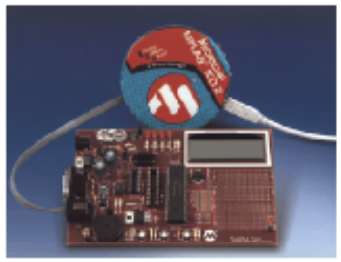
Design Constraints

- **Functionality**
  - System ability to perform the function it was designed for (REQ)
- **Cost**
  - Amount of resources needed to conceive, design, and produce an embedded system
- **Performance**
  - System ability to perform its function in time.
  - Affected by both HW & SW factors
- **Size**
  - Physical space taken by a system solution.
- **Power and Energy**
  - Energy required by a system to perform its function.
- **Time to Market**
  - The time it takes from system conception to deployment.
- **Maintainability**
  - System ability to be kept functional during its mature life.

Functionality

- Functional verification is a difficult task
  - Can consume up to 70% of development time
- Verification Methods
  - Simulation Techniques
    - Behavioral (HDL-based)
    - Logic (Circuit Modeling)
    - Processor (Software)
  - JTAG Debugger
    - Hardware supported through dedicated ports
    - Used also for testing (boundary scan test)
    - Cost effective

MSP430 FET Tool
Courtesy of Texas Instruments Inc.



PIC In-circuit Debugger
Courtesy of Microchip Corporation

Functionality (Cont.)

- In-Circuit Emulators
  - Replace MCU in target system
  - A powerful debugger
  - Expensive
- ROM Monitors
  - Monitor functions in ROM
  - Status sent via serial port

68HC11 In-circuit Emulator
Courtesy of Nohau Systems

8051 In-circuit Emulator
Courtesy of Signum Systems

ICE Test Pod
Courtesy of Signum Systems

8086 In-circuit Emulator
Courtesy of Nohau Systems

Performance: HW Factors

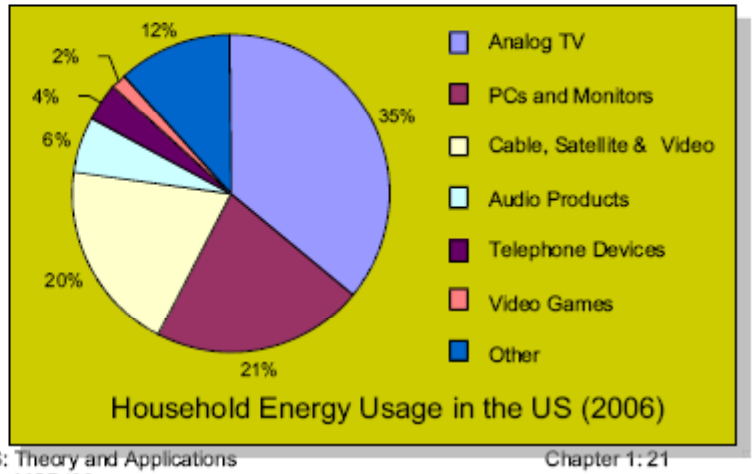
- Clock Frequency
  - System clock speed: not an absolute performance metric
- Architecture
  - Determines how clock cycles are used
- Component Speed
  - Response time and access time
- Handshaking
  - Signalization required to complete a transaction
- Low-power Modes
  - Wake-up times might affect application speed
- High speed is *expensive!!!*
  - Use it wisely

Power & Energy

- Critical parameter for a long chain of design events
  - System reliability
    - Stress, noise, and heat
  - Cooling Costs
    - High power = lot of heat to remove
  - Power Supply Requirements
    - Larger batteries of power supply
  - Size, Weight, and Form
    - Mechanical system parameters affected by heat density

Power & Energy (Cont.)

- Environmental Impact of Embedded Systems
  - Average individual uses 60 microprocessors per day
  - Household electronics accounts for 11% of all energy consumed in the USA
    - 147,000,000,000 KWh (147TWh) per year
  - Excludes digital TVs and large appliances
  - Excludes industry, schools, hospitals, etc
  - Trend continues to grow… Is there a limit?



Household Energy Usage in the US (2006)

Power and Energy (Cont.)

- Save the Planet! Design Power Efficient Embedded Systems
  - Use low-power MCUs and peripherals
    - Activate CPU standby and sleep modes
  - Write power efficient code
    - Every wasted CPU cycle is energy that will never come back
  - Use power management techniques
    - Power and clock gating plus efficient coding techniques