Introduction

EE8205: Embedded Computer Systems

http://www.ecb.torontomu.ca/~courses/ee8205/

Dr. Gul N. Khan
http://www.ecb.torontomu.ca/~gnkhan
Electrical Computer and Biomedical Engineering
Toronto Metropolitan University

Overview

- Embedded Software Systems: Course Management
- Real-time and Embedded Systems
- Embedded System Applications
- Characteristics of Embedded Systems

Text by Wolf: part of Chapter 1, Text by Navabi: part of Chapters 8 and 9

Electrical, Computer and Biomedical Engineering

EE8205: Embedded Computer Systems

http://www.ecb.torontomu.ca/~courses/ee8205/

Instructor: **Dr. Gul N. Khan**

Email: gnkhan@torontomu.ca

URL: http://www.ecb.torontomu.ca/~gnkhan/

Telephone: 416 979-5000 ext. 556084, Office**:** ENG448

Office Hours: Monday 11:45AM-1:00PM and by appointment

Department of Electrical, Computer and Biomedical Engineering

Toronto Metropolitan University

Lectures and Projects

Lecture Material

- You will need to take notes from lectures and also require text-reference books and some research articles identified by the instructor.

Labs and Projects

- Aimed at concept reinforcement and practical research experience.

Lecture, Labs, Projects, and other Material is available at the course website:

http://www.ecb.torontomu.ca/~courses/ee8205/

Assessment and Evaluation

Labs:	20%
Project:	40%
Final Exam:	40%

Course Text-Reference Books and other Material

- *M. Wolf,* Computers as Components:
  Principles of Embedded Computing System Design, 4th Edition,
  Morgan Kaufman/Elsevier Publishers 2016, ISBN 978-0-12-805387-4
- *Daniel W. Lewis*, Fundamental of Embedded Software with ARM Cortex M3, 2nd Edition, Pearson 2013, ISBN 978-0-13-291654-7
- *Z. Navabi*, Embedded Core Design with FPGAs, McGraw-Hill, 2007, ISBN-13: 9780071474818 (ISBN-10: 0071474811)
- *D. C. Black, J. Donovan, B. Bunton & A. Keist*, SystemC: From the Ground Up, 2nd Edition, 2010, ISBN 978-0-387-69958-5
- *F. Vahid & T. Givargis*, Embedded System Design, 1st Edition John Wiley 2002, ISBN 0-471-38678-2
- *Alan Burns and Andy Wellings*, Real-time Systems & Programming Languages, Addison-Wesley 2001, ISBN 0 201 72988 1

Embedded Processors and Micro-controllers Data Sheets are available at the Course Website http://www.ecb.torontomu.ca/~courses/ee8205/

In addition to the text/reference books, lectures may contain material from research articles to be identified by the instructor.

Course Content and Topics

- Introduction to Embedded Computer Systems
  Text by Wolf: part of Chapter 1, Text by Navabi: part of Chapters 8, 9
- Hardware Software Codesign of Embedded Systems
  Text by Wolf Chapter 8 and Support Material from the course web page
- SystemC and Embedded System Co-design
  Text by Wolf: part of Chapter 7, Research & SystemC Articles
- Embedded CPU and IP Cores
  Text by Wolf: part of Chapters 2, 3 and 4
- ARM Cortex M3 Microcontroller & its Embedded Applications
  Text by Lewis: part of Chapters 5-8, Wolf: Chapter 2
- Real-time Operating System and Scheduling
  Textbooks by Lewis Chap 9 and 10, Wolf: Chap 6, Burns & Wellings: part of Chap 13
- Hardware Software Co-synthesis of Embedded Systems
  Text by Wolf: Chapter 8 and Support Material from the course web page
- Fault-tolerant Embedded Systems
  Text by Burns and Wellings, part of Chapter 5 and Support Material at the Webpage
- Introduction to Embedded SoC & Embedded System on Programmable Chips (FPGA)
  Text by Navabi: part of Chapters 6, 7. Articles and Support Material at course web page
  (if time permits)
- Digital Camera and other Embedded System Case Studies
  Text by Vahid & Givargis: Chapter 7, Text by Wolf: part of Chapter 8

Introduction

- What are Embedded Systems?
- Challenges in Embedded Computing System Design
- Design Methodologies

Main Aim of the Course

- To introduce embedded computer systems
  - Software and hardware components of an embedded system
- To understand real-time operating systems
- Embedded Computer Architecture
- Hardware Software Codesign

Ideally Student should have the knowledge of:

- Basics of Programming C or C++ and Computer Architectures
- Introduction to Operating Systems

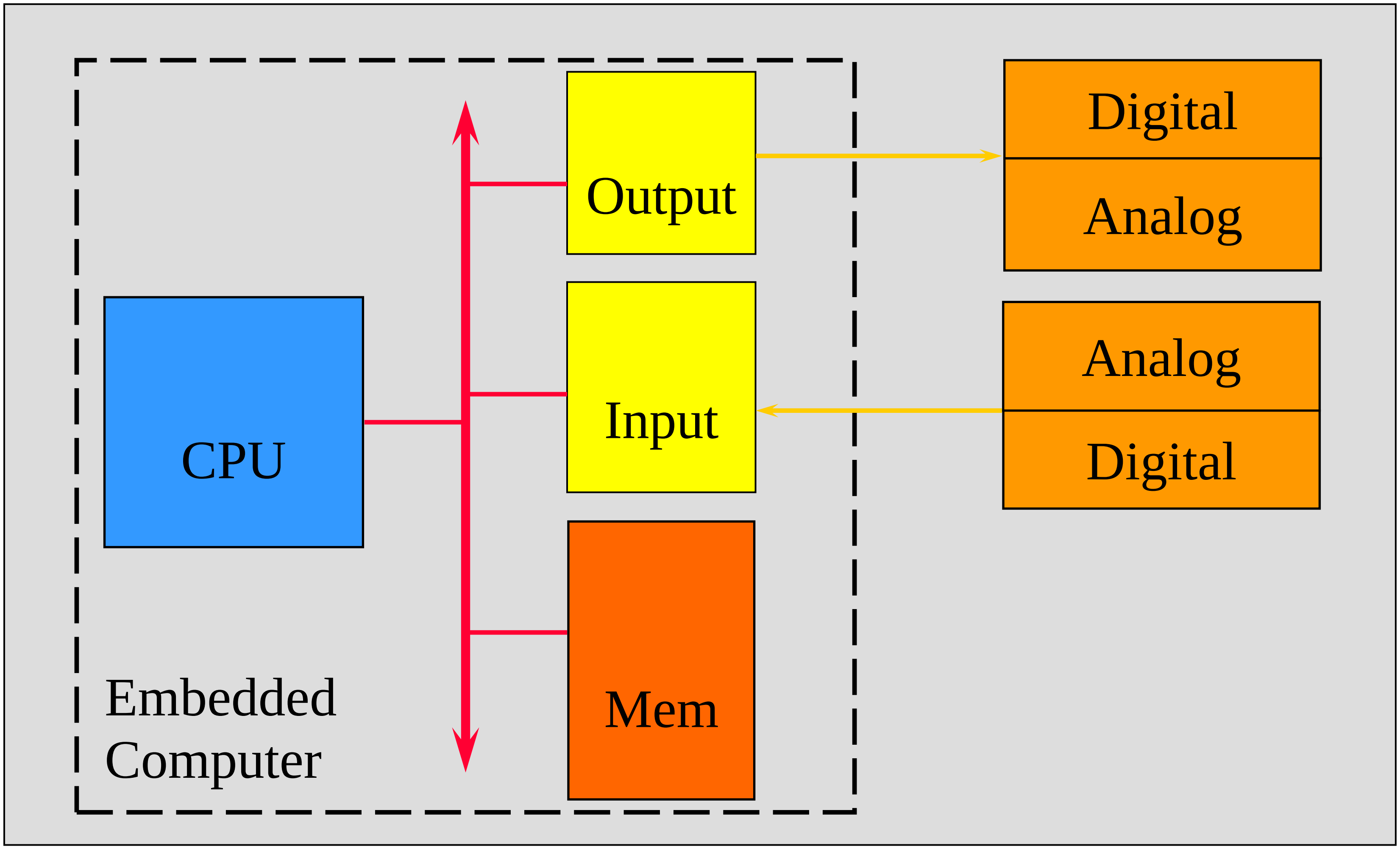
What is an Embedded/Real-time System?

Most real-time systems (RTS) are also embedded systems.

- An embedded system is an information processing system that responds to externally generated input stimuli within a finite and specified period.
  - The correctness depends not only on the logical result but also the time it was delivered
  - Failure to respond is as bad as the wrong response!
- Embedded system: any device that includes a programmable computer but is not itself a general-purpose computer.
- Take advantage of application characteristics to optimize the design:

Don't need all the general-purpose bells and whistles.

Embedding a Computer



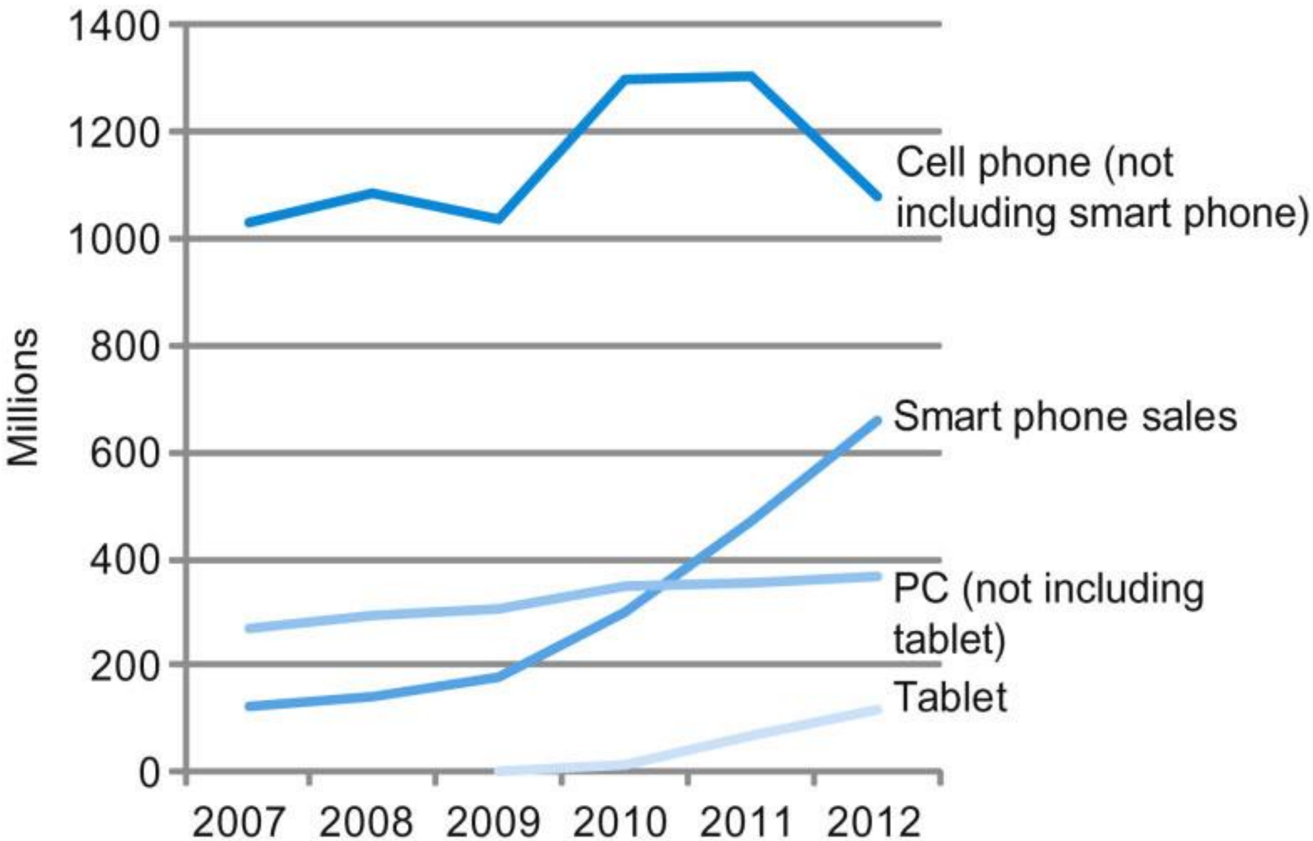
Embedded Systems

- Electronic devices that incorporate a computer (usually a microprocessor) within their implementation.
- A computer is used in such devices primarily as a means to simplify the system design and to provide flexibility.
- Often the user of the device is not even aware that a computer is present.
- Embedded Systems Rule the World
  - Embedded processors account for 100% of worldwide microprocessor production.
  - Embedded:desktop = 100:1
  - 99% of all processors are for the embedded systems market.
  - Number of embedded processors in a typical home is estimated at 50-60.

(A recent ACURA Model has more than 50 processors)

Embedded CPU Applications

Some Embedded and PC Systems



- Cell phones, PCs and TVs manufactured and shipped per year.
- More than 1.3 billion cell phones sold in 2015-2023. 1.56 billion in 2018 as compared to 240 million PCs in 2023.
- A 2024 survey of U.S. families found that they owned 17 E-GADGETS.

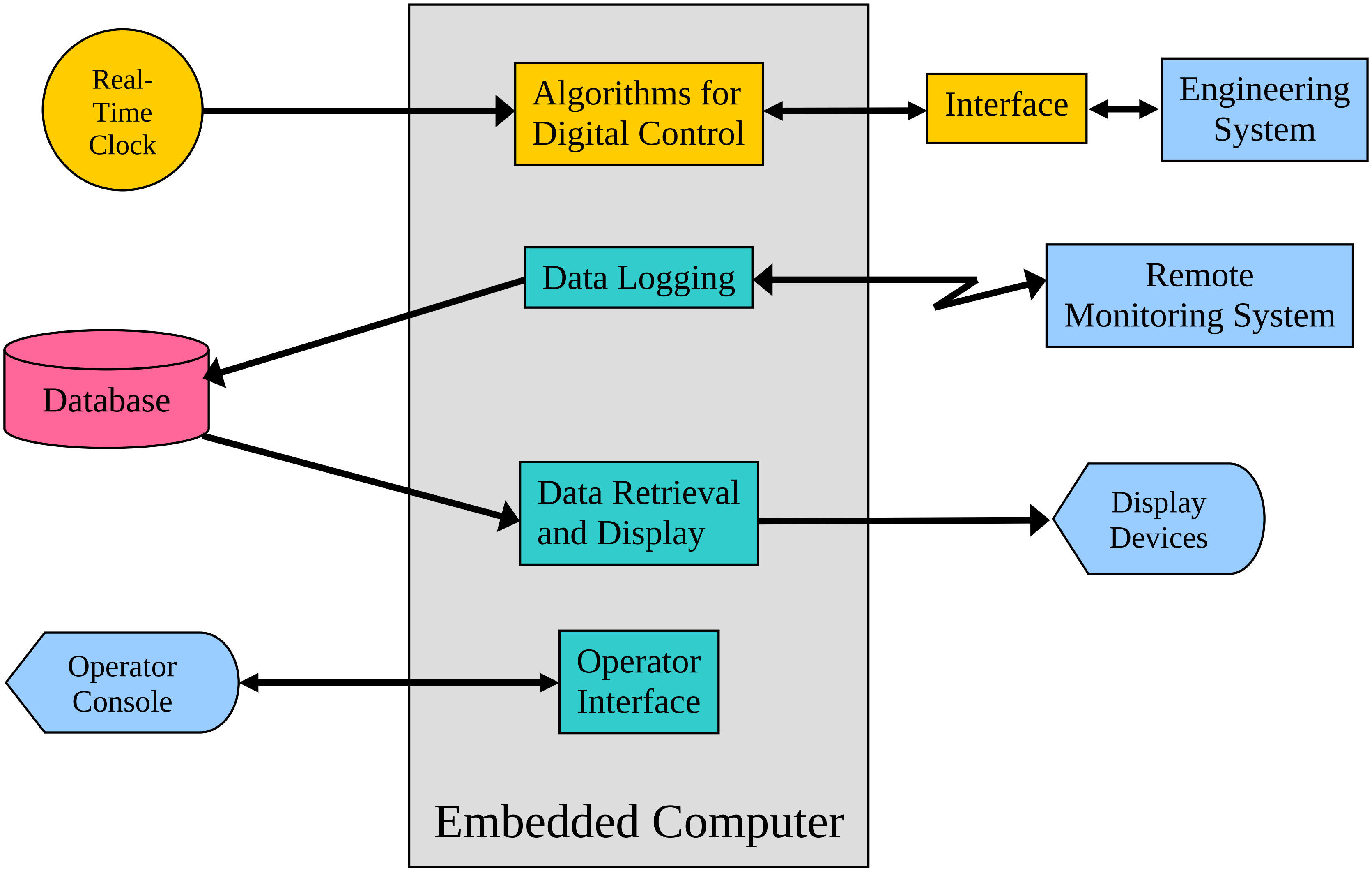
Early History of Embedded Systems

- First microprocessor was Intel 4004 in early 1970's.
- HP-35 calculator used several chips to implement a microprocessor in 1972.
- Automobiles used microprocessor-based engine controllers starting in 1970's.
  - Control fuel/air mixture, engine timing, etc.
  - Multiple modes of operation: warm-up, cruise, hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.

- Microcontroller: includes I/O devices, on-board memory.
- Digital signal processor (DSP): microprocessor optimized for digital signal processing.

  Typical embedded word sizes: 8-bit, 16-bit, and 32-bit.

A Typical Embedded System



Real Time Systems

- Real-time systems (RTS) process the events.
- Events occurring on external inputs cause other events to occur as outputs.
- Minimizing response time is usually a primary objective, or otherwise the entire system may fail to operate properly.

Types of Real Time System

- Hard real-time — e.g. Flight control systems.
- Soft real-time — e.g. Data acquisition system.
- Real real-time — e.g. Missile guidance system.
- Firm real-time

Types of Real Time System

- **Hard real-time —** systems where it is absolutely imperative that responses occur within the required deadline.
  For example: Flight control systems.
- **Soft real-time —** systems where deadlines are important, but which will still function correctly if deadlines are occasionally missed. For example: Data acquisition system.
- **Real real-time —** systems which are hard real-time and which the response times are very short.
  For example: Missile guidance system.
- **Firm real-time —** systems which are soft real-time but in which there is no benefit from late delivery of service.

A single system may have all hard, soft, and real real-time subsystems.

In reality many systems will have a cost function associated with missing each deadline.

Multi-Tasking and Concurrency

- Most real-time systems are also embedded systems with several inputs and outputs and multiple events occurring independently.
- Separating tasks simplifies programming but requires somehow switching back and forth among the three tasks (*multi-tasking*).
- *Concurrency* is the appearance of simultaneous execution of multiple tasks.

Concurrent Tasks for a Thermostat

/* Monitor Temperature */	**/* Monitor Time of Day */**	**/* Monitor Keypad */**

do forever {
    measure temp ;
    if (temp < setting)
        start furnace ;
    else if (temp >
          setting + delta)
        stop furnace ;
}
	do forever {
	    measure time ;
	    if (6:00am)
	        setting = 72^{o}F ;
	    else if (11:00pm)
	        setting = 60^{o}F ;
	}
		do forever {
		    check keypad ;
		    if (raise temp)
		        setting++ ;
		    else if (lower temp)
		        setting-- ;
		}

Embedded System Applications

Aerospace	Navigation systems, automatic landing systems, flight attitude controls, engine controls, space exploration (e.g., the Mars Pathfinder).
Automotive	Fuel injection control, passenger environmental controls, anti-lock braking, air bag controls, GPS mapping.
Children's Toys	Nintendo's "Game Boy", Mattel's "My Interactive Pooh", Tiger Electronics' "Furby".
Communi-cations	Satellites; network routers, switches, hubs.

Embedded System Applications

Computer Peripherals	Printers, scanners, keyboards, displays, modems, hard disk drives, CD/DVD-ROM drives.
Home	Dishwashers, microwave ovens, VCRs, televisions, stereos, fire/security alarm systems, lawn sprinkler controls, thermostats, cameras, clock radios, answering machines.
Industrial	Elevator controls, surveillance systems, robots.
Instrumen-tation	Data collection, oscilloscopes, signal generators, signal analyzers, power supplies.

Embedded System Applications

Medical	Imaging systems (e.g., XRAY, MRI, and ultrasound), patient monitors, and heart pacers.
Office Automation	FAX machines, copiers, telephones, and cash registers.
Personal	Personal Digital Assistants (PDAs), pagers, cell phones, wristwatches, video games, portable MP3 players, GPS.

Embedded Real-Time Software Examples

Property	***FAX Machine***	***CD/DVD Player***
Microprocessor:	16-bit	16-bit
Number of Threads:	6	9-12
Read-Write Memory (RAM):	2048 Bytes	512 Bytes
Total RAM Actually Used:	1346 Bytes (66%)	384 Bytes (75%)
Amount Used by Kernel:	250 Bytes (19%)	146 Bytes (38%)
Read-Only Memory (ROM):	32.0 KB	32.0 KB
Total ROM Actually Used:	28.8 KB (90%)	17.8 KB (56%)
Amount Used by Kernel:	2.5 KB (8.7%)	2.3 KB (13%)

Embedded System Examples

- Aircraft and jet engine control
- Satellites, Space crafts
- Nuclear reactor and power system control
- Networking devices like routers, switches etc.
- Personal digital assistant (PDA).
- Printer, Plotters etc.
- Cell phone
- Television and other Consumer Electronics
- Household appliances
- Automobile: engine, brakes, dash, etc.

Automotive Embedded Systems

Today's high-end automobile may have 100 microprocessors:

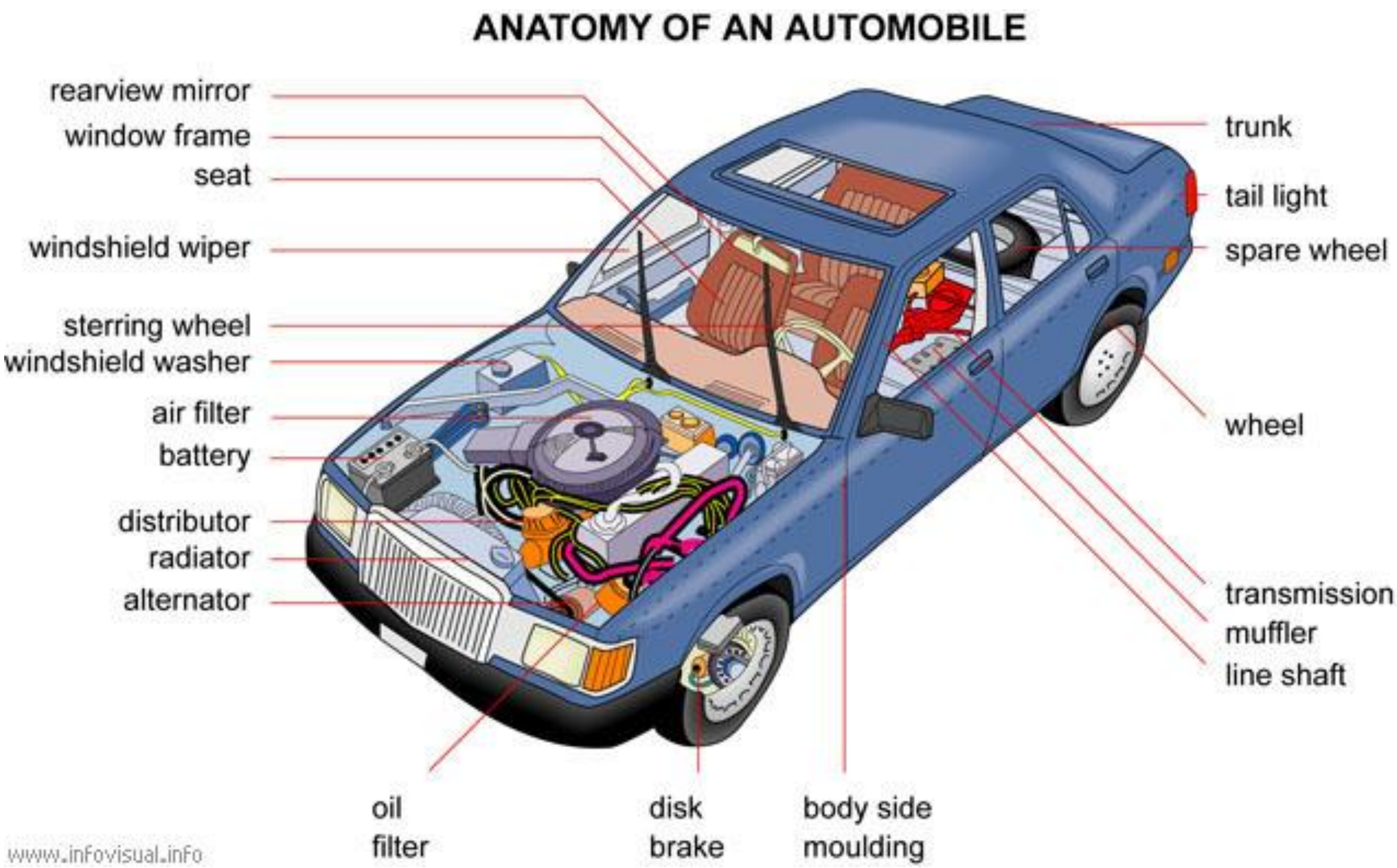
- 4-bit microcontroller checks seat belt
- Microcontrollers run dashboard devices
- 16/32-bit microprocessor controls engine

BMW 850i brake and stability control system

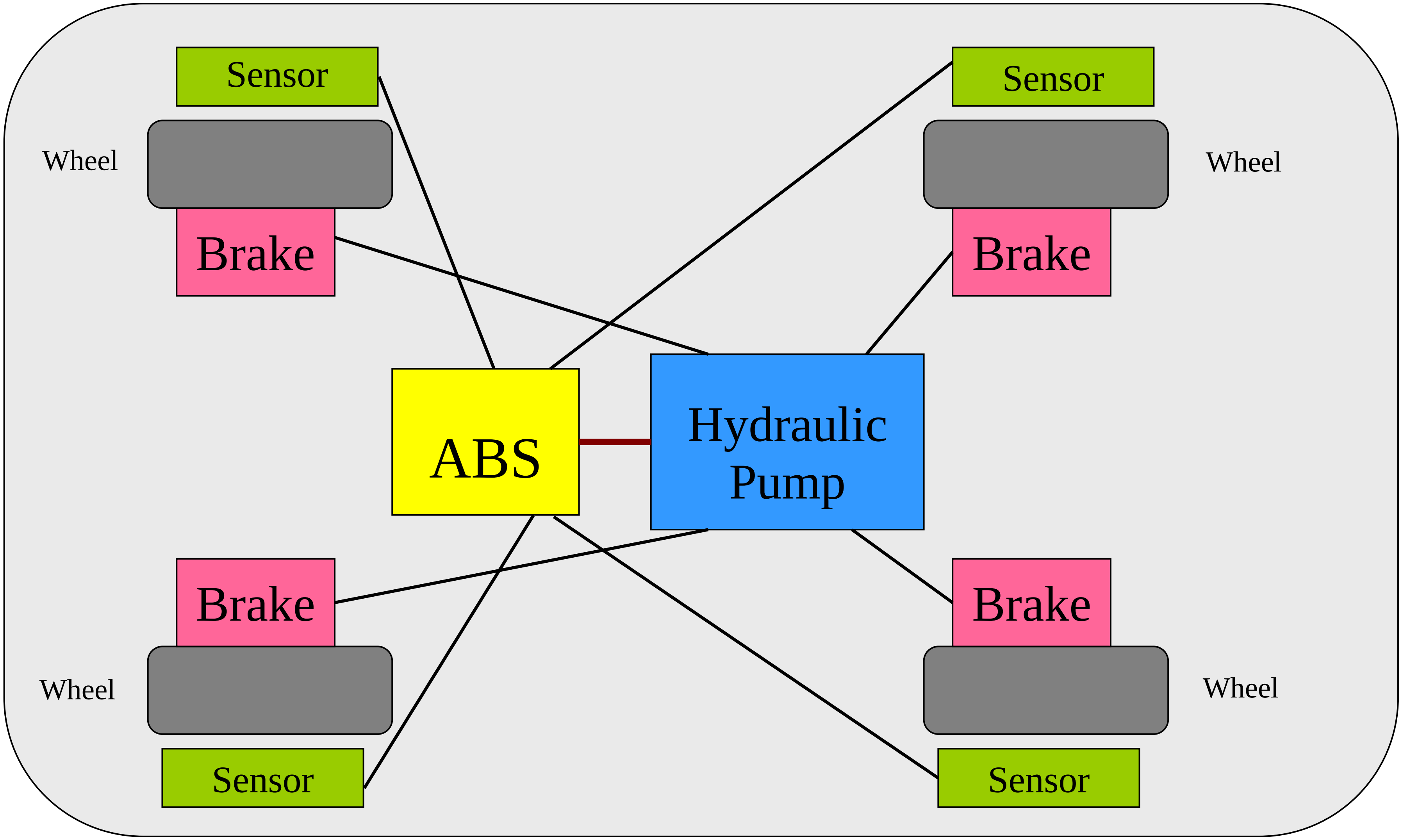
- Anti-lock brake system (ABS): Pumps brakes to reduce skidding.
- Automatic Stability Control (ASC+T): Controls engine to improve stability.
- ABS and ASC+T communicate.

  ABS was introduced first---needed to interface to existing ABS module.

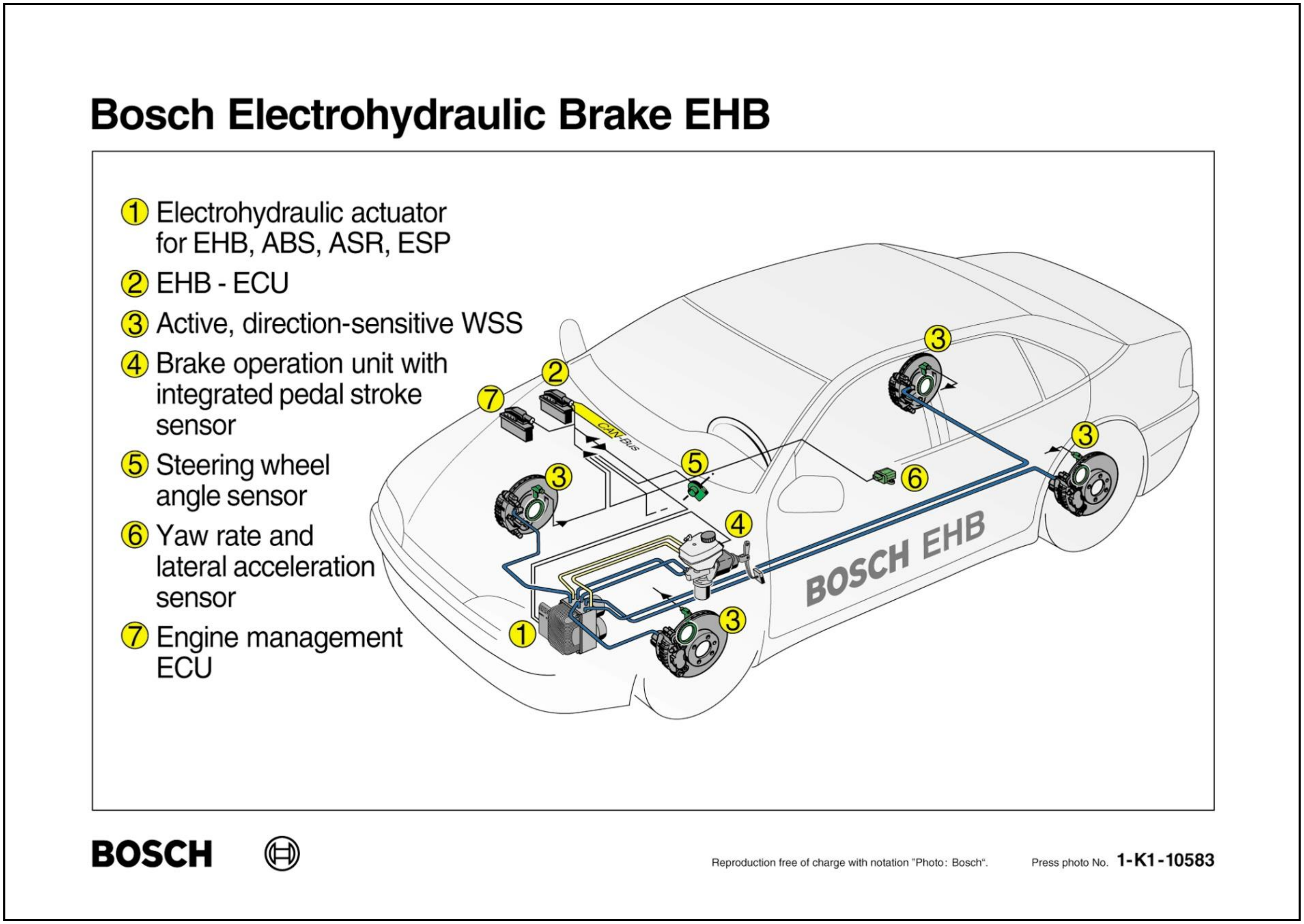
Embedded Systems and Automobile



Anti-lock Brake System (ABS)

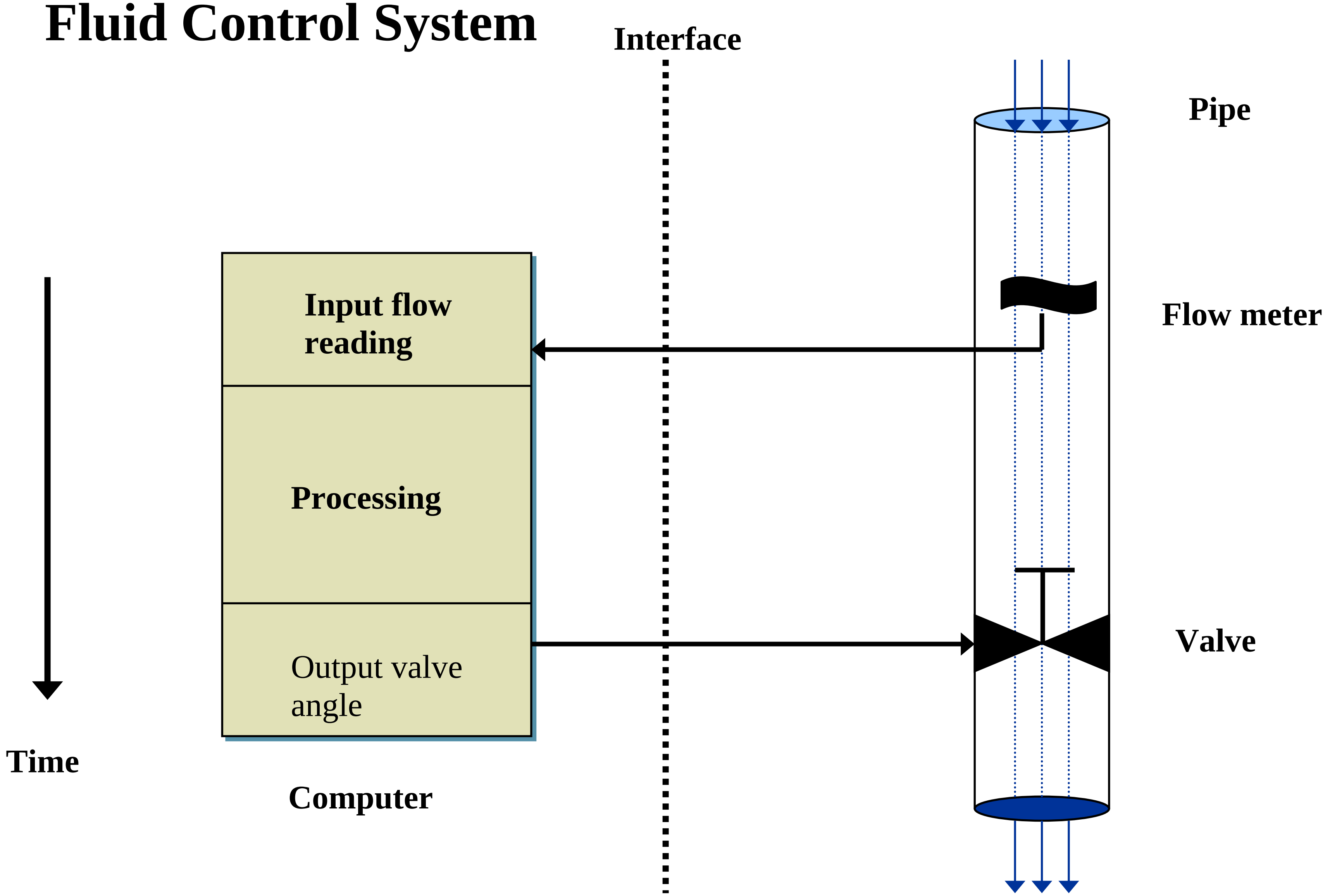


Electrohydraulic Brake



Embedded System Application

Fluid Control System



Embedded System Applications

Programmable Digital Thermostat

Uses: 4-bit Microprocessor

Embedded System Applications

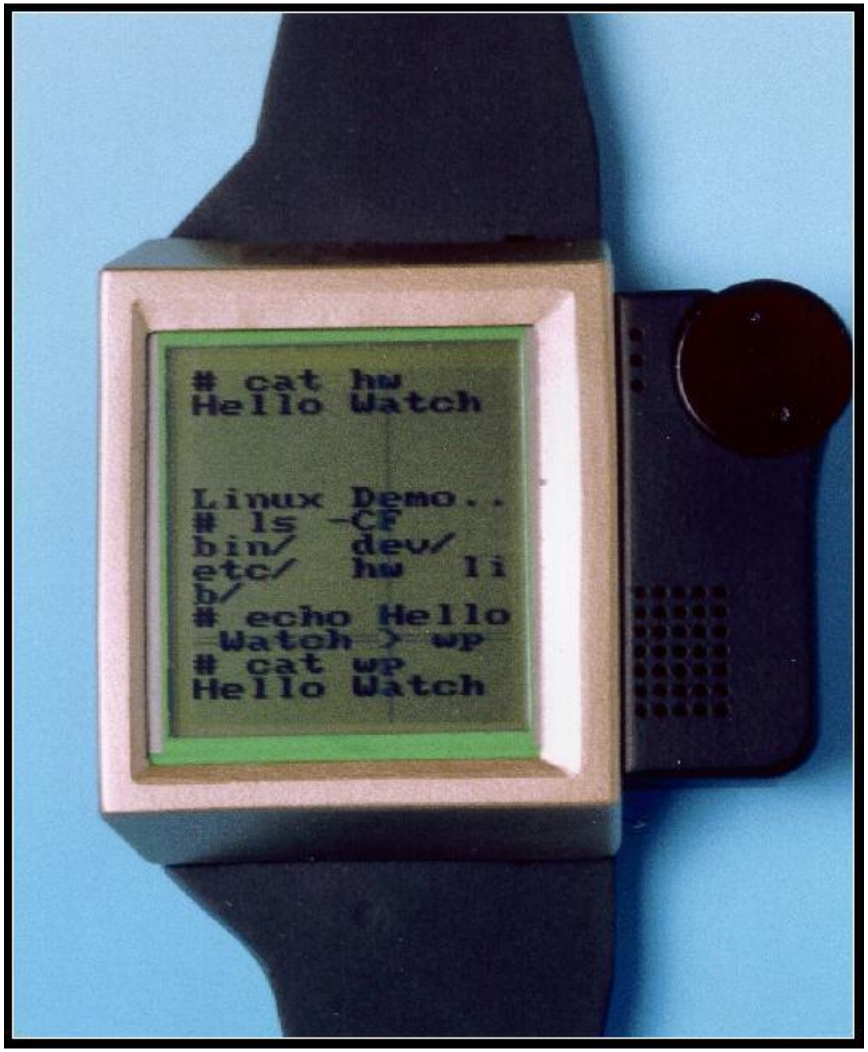
Miele Dishwashers

Microprocessor: 8-bit Motorola 68HC05

DVD Player

Microprocessor:
32-bit RISC

IBM Research's Linux Wristwatch Prototype



Microprocessor
32-bit ARM RISC

Vitality's GlowCap

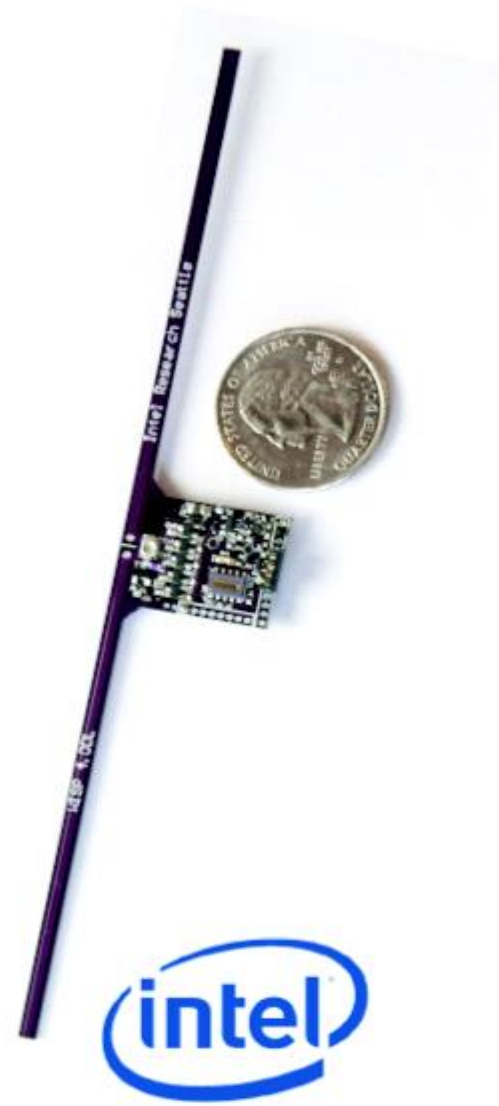
Vitality’s GlowCap

- GlowCap has a tiny Amtel 8-bit picoPower AVR Processor
- Help People to take their medication on-time.
- Sense when the bottle is opened.
- Connect to Vitality server and transmit information wirelessly.

Intel WISP RFID

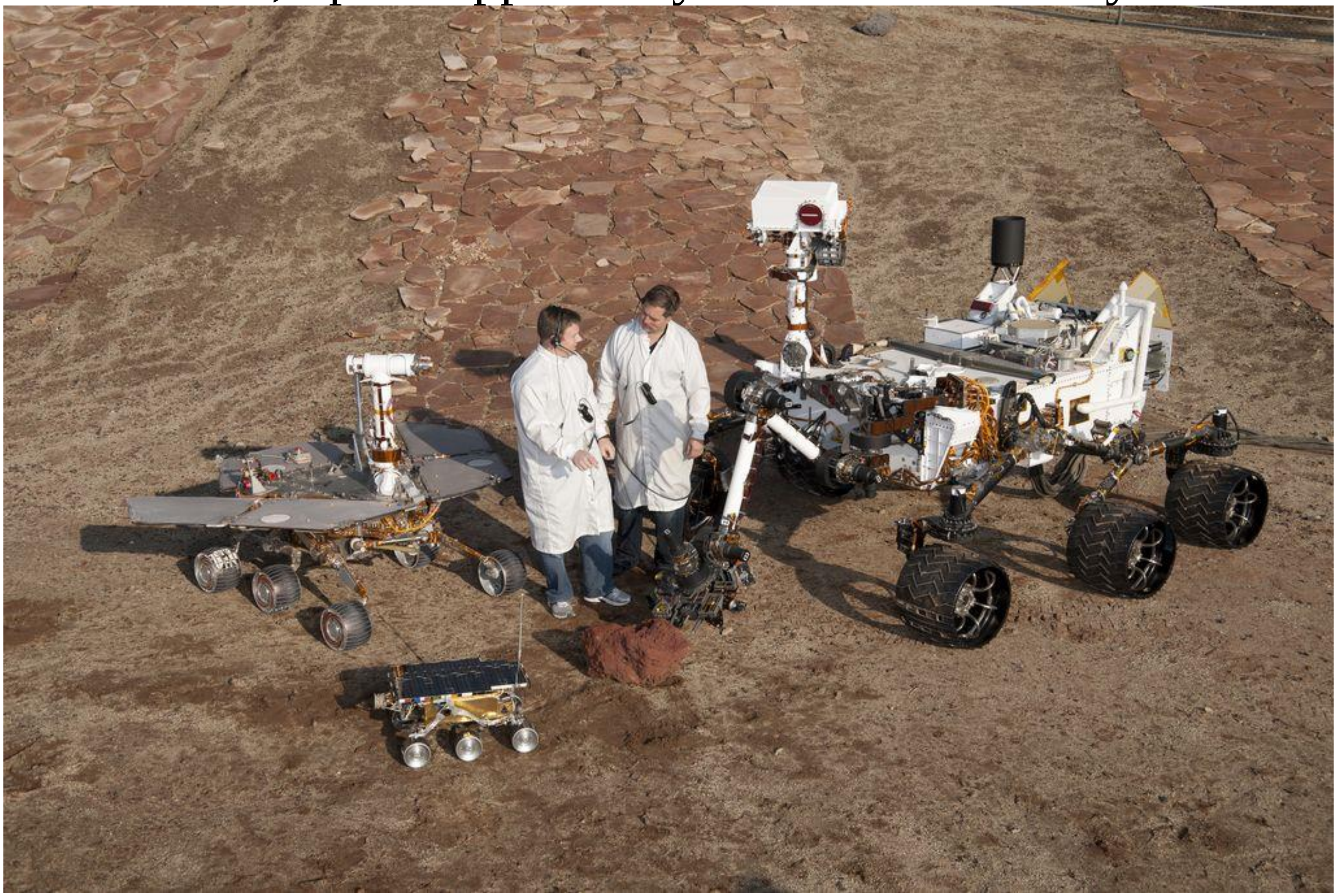
TIMSP430F1232: Low Power Micro-controller

- 16-bit CPU
- 8 Kbytes of flash memory
- 256 bytes of RAM
- 10-bit –ADC with 200 kilo-samples/second
- CPU can run at 8MHz with 3.3V supply voltage



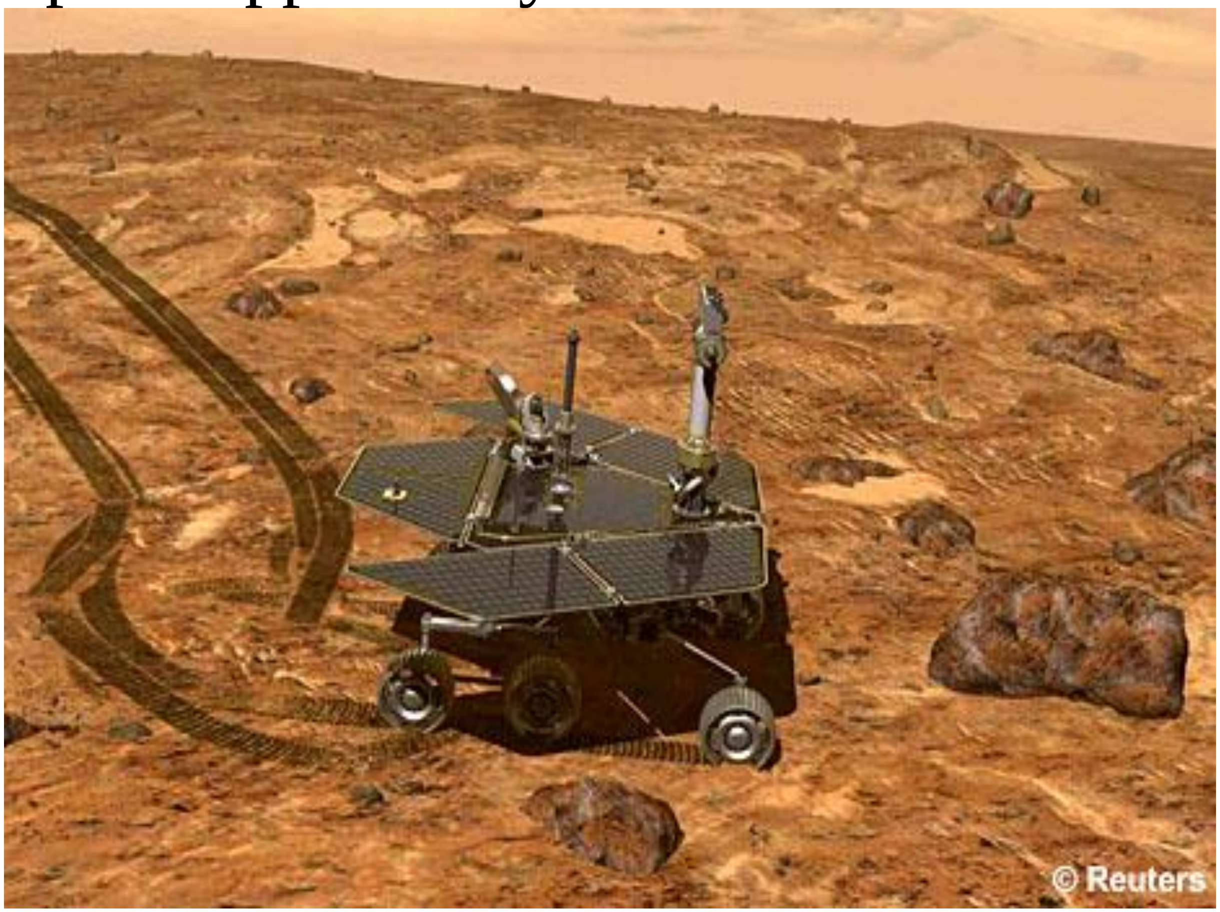
MAR's Rovers

Pathfinder-1997, Spirit/Opportunity-2003 and Curiosity-2012



2003 MAR'S Rover

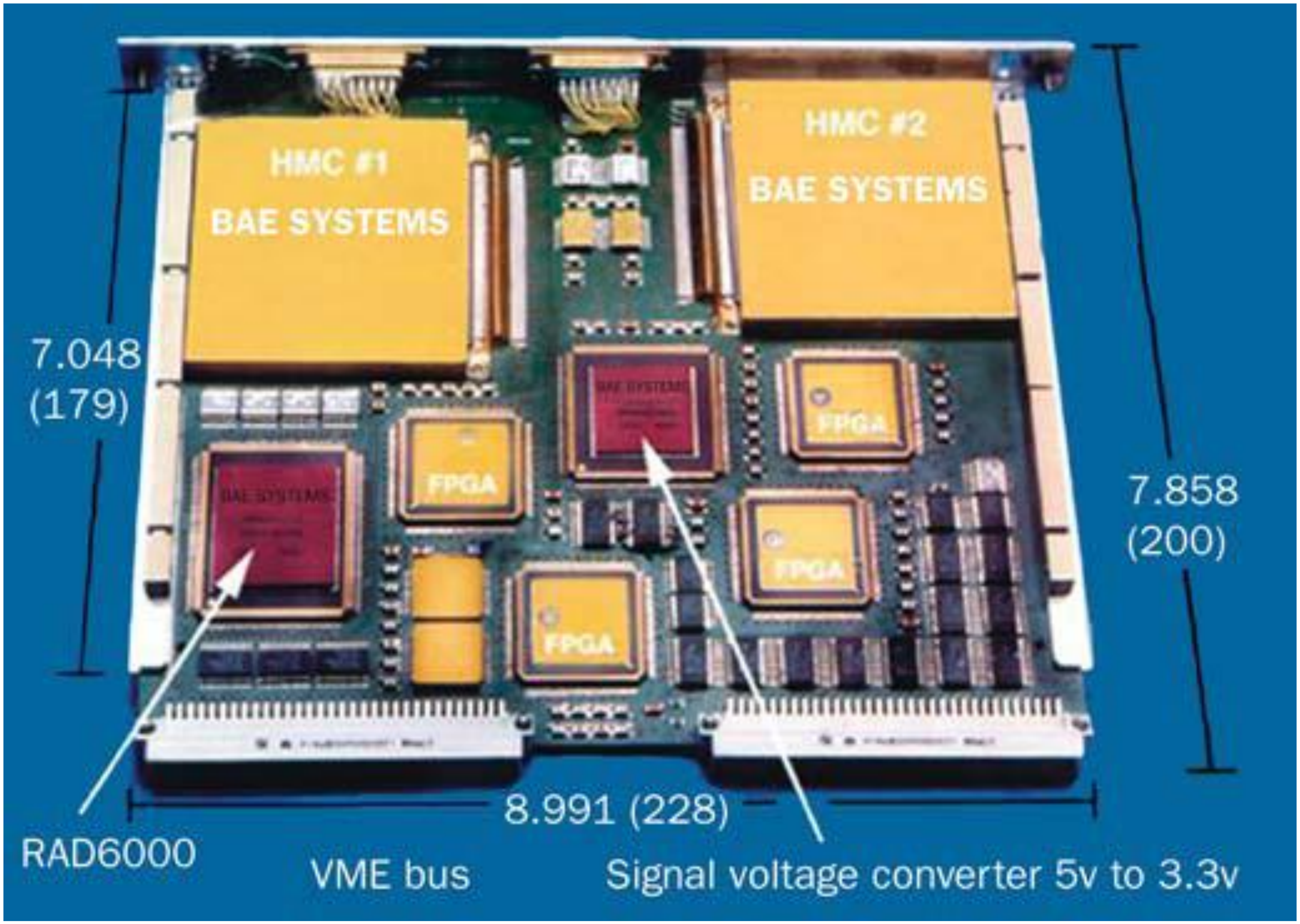
Spirit/Opportunity



- Use BAE Systems RAD6000 32-bit RISC Processor
- Radiation hardened IBM POWER series 6000 CPU
- Employ VxWorks Embedded Real-time Operating System from Wind River.

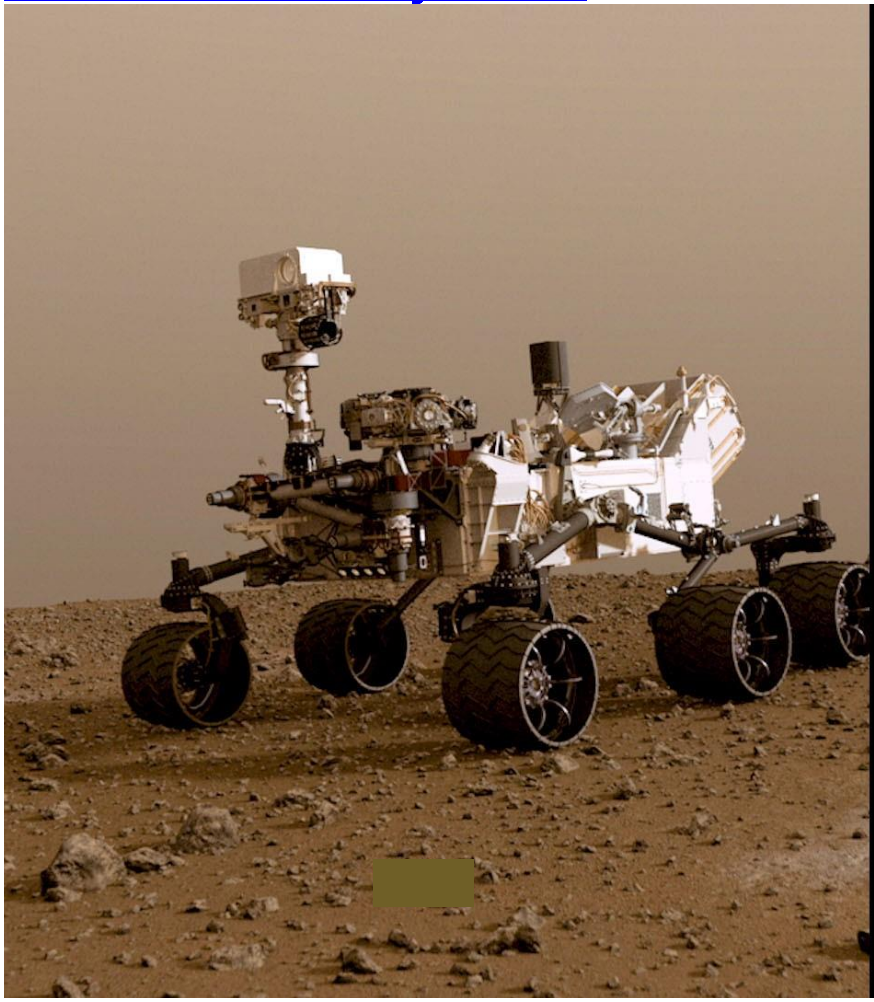
Mars Rover RAD6000 Flight Computer

FPGA-based



MARS Rover 2020 - Perseverance Rover

Landed February 2021



Comparison of embedded Computer Systems for Mars Rovers

Rover (mission,year)	**CPU**	**RAM**	**Storage**	**Operating system**
Sojourner Rover (Pathfinder 1997)	2MHz Intel 80C85	512KB	176 KB	Custom cyclic executive
Pathfinder Lander (1997) (Base station for *Sojourner* rover)	20MHz IBM RAD6000	128 MB	6 MB (EEPROM)	VxWorks (multitasking)
Spirit and ***Opportunity*** (Mars Exploration Rover, 2004)	20 MHz IBM RAD6000	128 MB	256 MB	VxWorks (multitasking)
Curiosity (Mars Science Laboratory, 2011)	200 MHz IBM RAD750	256 MB	2GB	VxWorks (multitasking)
Perseverance 2 Compute Elements (Mars Rover, 2020) Landed 2021	200 MHz IBM RAD750 PowerPC 750	256 MB	2GB Flash Memory 256KB EEPROM	VxWorks (multitasking)

Characteristics of an RTS

- Large and complex — vary from a few hundred lines of assembler or C to 20 million lines of Ada code estimated for the Space Station Freedom
- Concurrent control of separate system components — devices operate in parallel in the real world; better to model this parallelism by concurrent entities in the program.
- Facilities to interact with special purpose hardware — need to be able to program devices in a reliable and abstract way
- Extreme reliability and safe — embedded systems typically control the environment in which they operate; failure to control can result in loss of life, damage to environment or economic loss.
- Guaranteed response times — we need to be able to predict with confidence the worst-case response times for systems; efficiency is important, but predictability is essential

Characteristics of Embedded Systems

- Sophisticated functionality.
- Real-time operation.
- Low manufacturing cost.
- Low power.
- Designed to tight deadlines by small teams.

Functional complexity

- Often have to run sophisticated algorithms or multiple algorithms.

  Cell phone, laser printer.
- Often provide sophisticated user interfaces.

Design goals

- Performance.
  Overall speed, deadlines.
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements
  (physical size, etc.)

Non-functional Requirements

- Many embedded systems are mass-market items that must have low manufacturing costs.
- Limited memory, microprocessor power, etc.
- Power consumption is critical in battery-powered devices.
- Excessive power consumption increases system cost even in wall-powered devices.

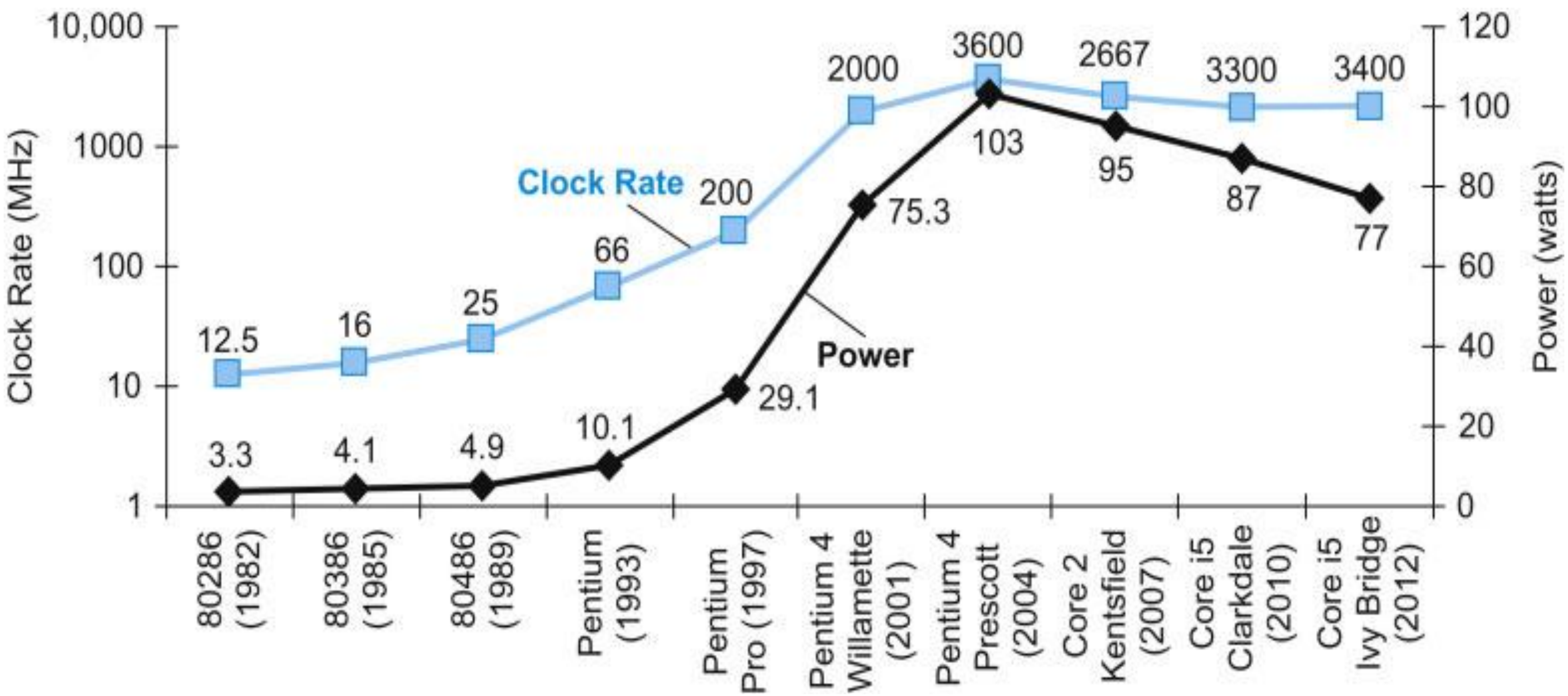
Power

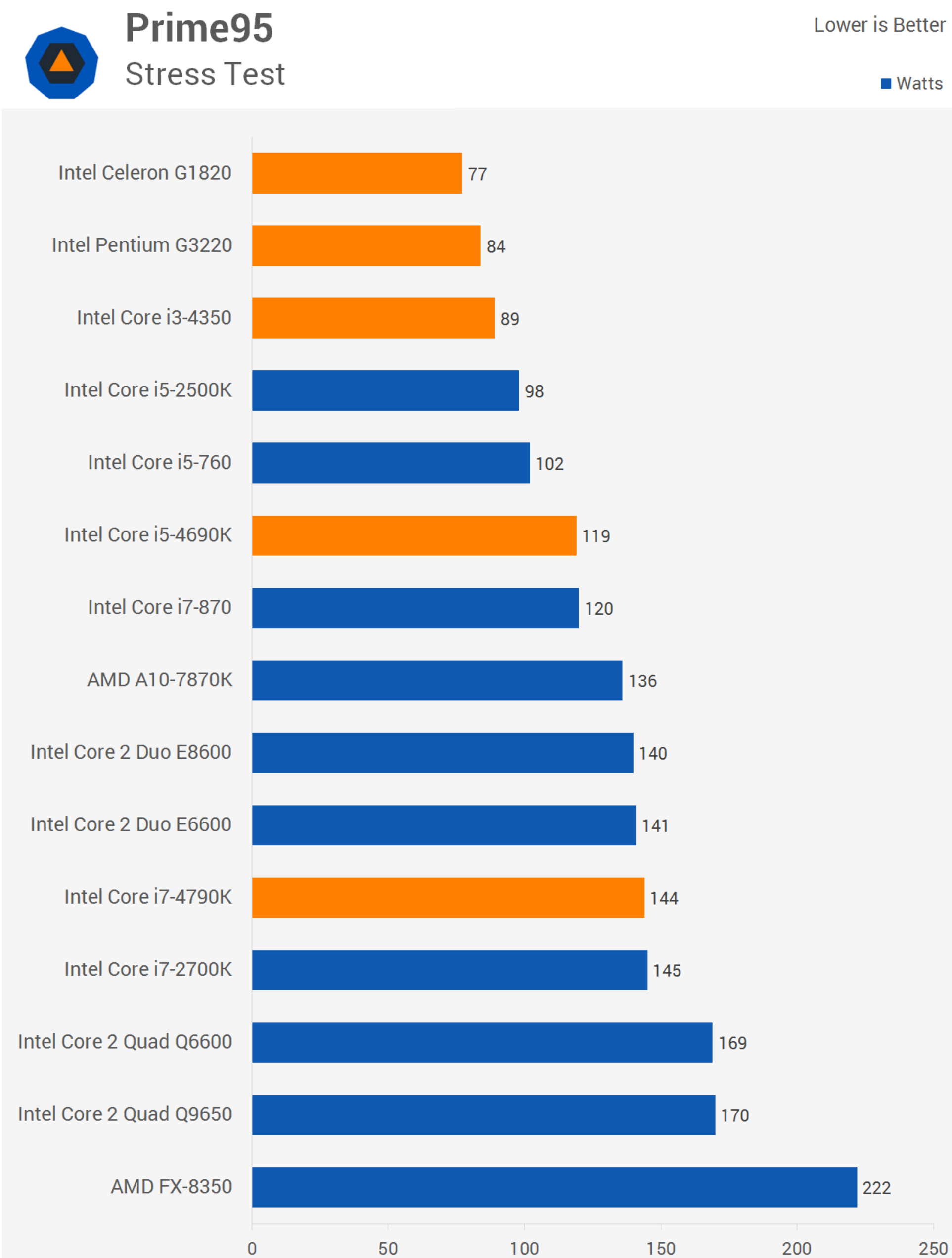
- Custom logic is a clear winner for low power devices.
- Modern microprocessors offer features to help control power consumption.
- Software design techniques can help reduce power consumption.

Power and Clock

Intel x86 Power Requirements

- Pentium-4 made a dramatic jump in power.
- Core-2 reverts to simpler pipeline with lower power.





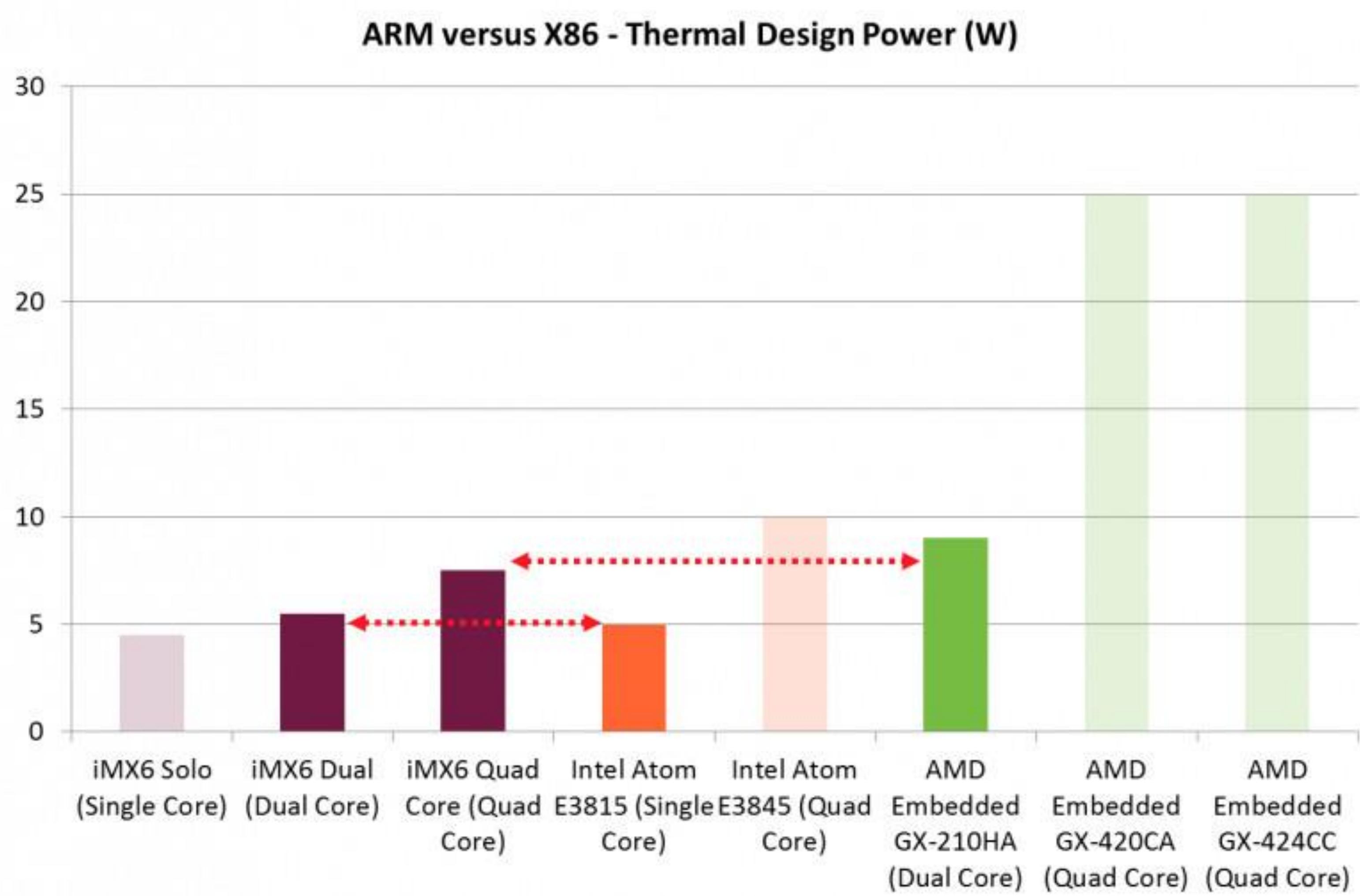
10 Years of Intel CPUs Compared

Prime95 test code calculate prime numbers in rapid succession and will do this until:

(a) It finds a unique prime number and notify.
(b) We stop the test.
(c) CPU hardware fails and the test fails due to a miscalculation (worst-case scenario)

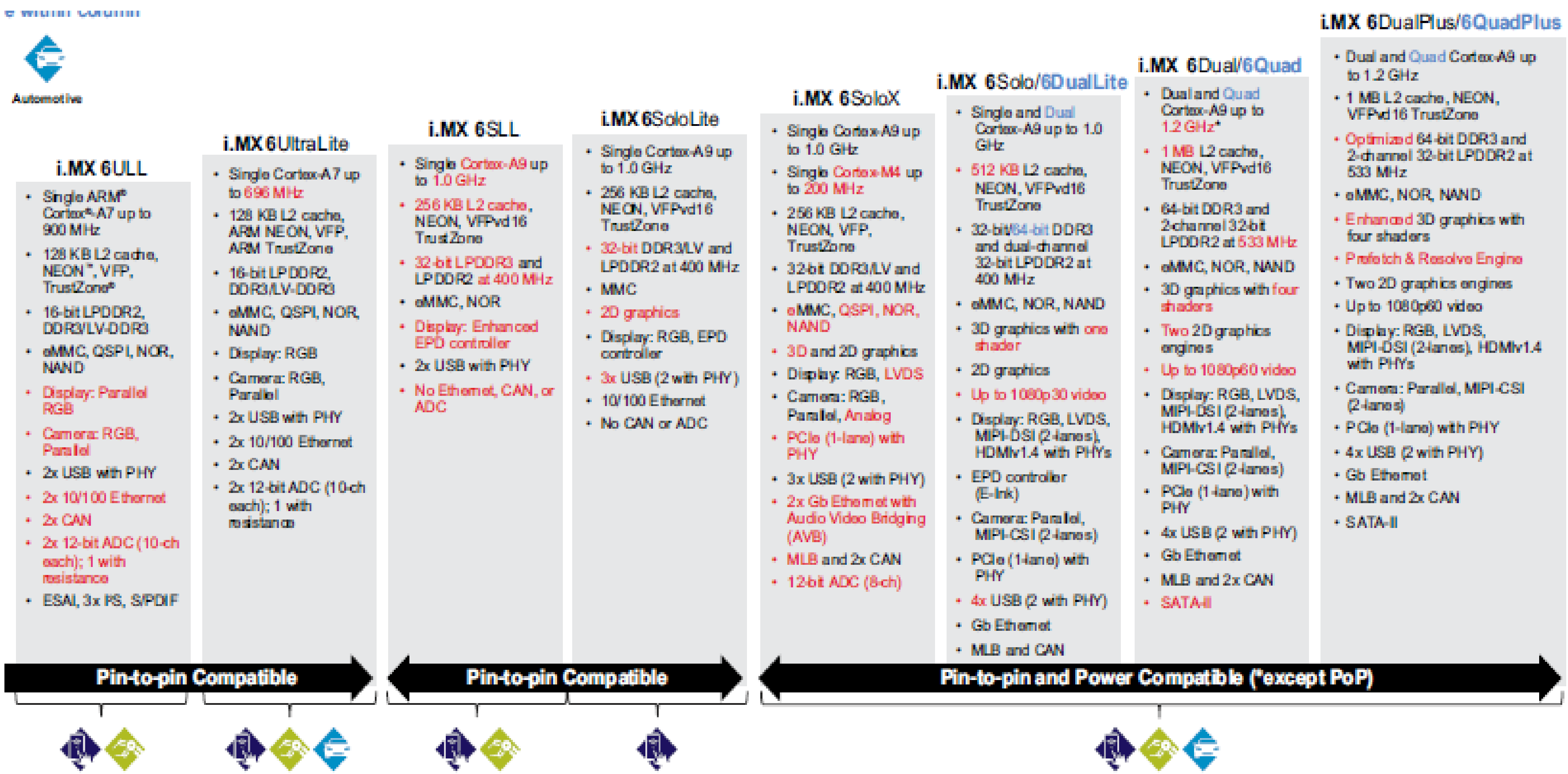
- Inel Celeron G1820 consumed the least amount of power, followed by Pentium G3220, and then the Core i3-4350.
- Between the Core i3-4350 and i5-4690K, there are i5-2500K and i5-760, while the Core i7-870 consumes roughly the same amount of power as the i5-4690K.
- Core 2 Duo E6600 consumed the same amount of power as the i7-4790K and 2700K. The Core 2 Quad processors consumed considerably more, reaching a total system consumption of 170 watts.

ARM vs. x86 Power



iMX6 is Cortex A7, A9 and M4 multicore CPUs NXP SoCs

NXP iMX6 Multiple ARM Processors



Platforms

Embedded computing platform: hardware architecture + associated software.

Many platforms are multiprocessors.

Examples:

- Single-chip multiprocessors for cell phone base band.
- Automotive network + processors.

Heterogeneous systems:

- Some custom logic for well-defined functions
- CPUs+software for everything else

The Performance Paradox

Microprocessors use much more logic to implement a function than does custom logic.

But microprocessors are often at least as fast:

- Heavily pipelined
- Large design teams
- Aggressive VLSI technology

In general-purpose computing, performance often means average-case, may not be well defined.

In real-time systems, performance means meeting deadlines.

- Missing the deadline by even a little is bad.
- Finishing ahead of the deadline may not help.

Characterizing Performance

We need to analyze the system at several levels of abstraction to understand performance:

- CPU
- Platform
- Program
- Task
- Multiprocessor

Design Goals

Reliability

- Mission Critical
- Life-Threatening Application
- 24/7/365 and cannot reboot!

Performance

- Multitasking and Scheduling
- Optimized I/O, Assembly Language
- Limits, Inaccuracies of Fixed Precision

Cost

- Consumer Market: Minimize Manufacturing Cost.
- Fast Time to Market Required
- No chance for future modification

Challenges in Embedded System Design

- How much hardware do we need?
  - How big is the CPU? Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic? Reduce memory accesses?

Does it really work?

- Is the specification correct?
- Does the implementation meet the spec?
- How do we test for real-time characteristics?
- How do we test on real data?

How do we work on the system?

- What is our development platform?

Design Methodologies

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn't skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - Help automate methodology steps;
  - Keep track of the methodology itself.

Top-down design:

- Start from most abstract description;
- Work to most detailed.

Bottom-up design:

- Work from small components to big system.

Real design uses both techniques

Summary

- Embedded computers are all around us.
  Many systems have complex embedded hardware and software.
- Embedded systems pose many design challenges:
  - Design time,
  - Deadlines,
  - Power, etc.
- Design methodologies help us manage the design process.

Where are we heading?

- Embedded Computer Systems
- Hardware Software Co-design of Embedded System
- Embedded CPUs and ARM Cortex M3/M4 Processors
- Cortex M3 Programming for Embedded Applications
- Real-time Operating System (RTX) and Scheduling
- SystemC and Hardware Software Co-design
- Embedded System Co-synthesis
- Embedded System on Programmable Chips
  (if time permits)
- Fault-tolerant Embedded Computer Systems
- Embedded System Case Studies
- A Typical Embedded System Example