

## EE345L Fall 2009

**Lecture 1 objectives**

Course description  
 Definitions  
 Architecture of the 9S12  
 Data flow graph  
 Device Driver  
 Fixed-point

**1.1. Embedded Computer Systems****1.1.1. Applications**

An *embedded computer system* is an electronic system  
 it includes a microcomputer like the Freescale 9S12  
 it is configured to perform a specific dedicated application  
 software is programmed into ROM  
 software is not accessible to the user of the device  
 software solves only a limited range of problems  
 microcomputer is embedded or hidden inside

A typical automobile now contains an average of ten microcontrollers. In fact, upscale homes may contain as many as 150 microcontrollers and the average consumer now interacts with microcontrollers up to 300 times a day. General areas that employ embedded microcomputers encompass every field of engineering:

- communications,
- automotive,
- military,
- medical,
- consumer,
- machine control.

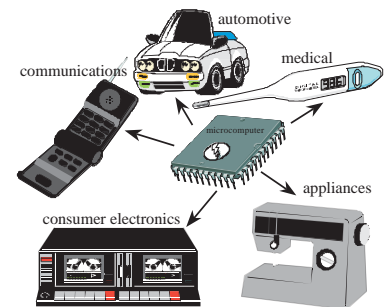


Figure 1.1. Example embedded computer systems.

Each **embedded microcomputer** system  
 accepts inputs,  
 performs calculations, and  
 generates outputs  
 runs in “real time.”

In a **real time system**, upper bound on the time required to  
 perform the input/calculation/output  
 respond to external events

Because of the real time nature of these systems, we will study the rich set of features built into these microcontrollers to handle all aspects of time.

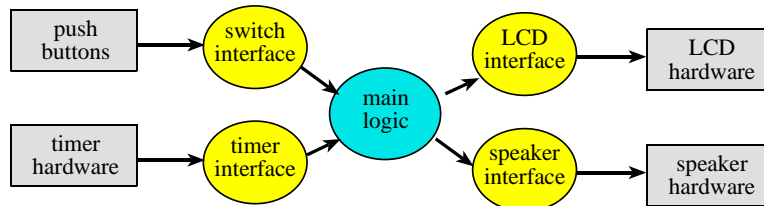
**What really matters?**

Speed (bytes/sec)  
 Power (watts)  
 Size (cm<sup>3</sup>) and weight (g)  
 Accuracy (% error)

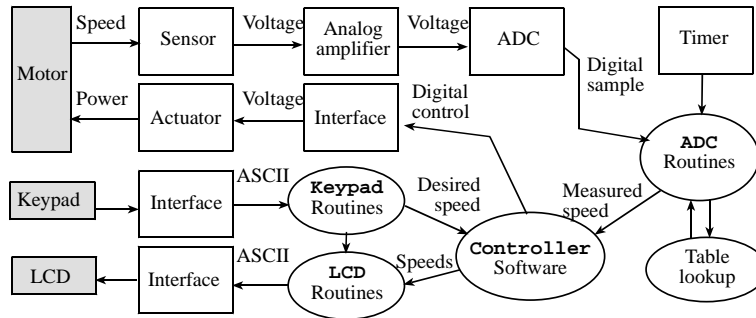
**How do you choose a microcontroller?**

Ability to process data (speed, precision, data type)  
 I/O devices  
 RAM and ROM size

An **interface** is defined as the hardware and software that combine to allow the computer to communicate the external hardware. We must also learn how to interface a wide range of inputs and outputs that can exist in either digital or analog form.



Data flow graph of an alarm clock



A data flow graph showing how signals pass through a motor controller.

A general-purpose computer system

- keyboard,
- disk
- graphics display
- software solves a wide variety of purposes
- software can be changed by user

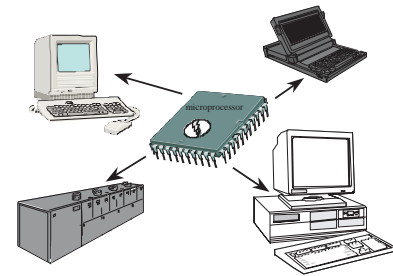


Figure 1.2. Example general-purpose computer systems.

**1.1.2. Software Issues**

Software maintenance is extremely important.

- verification of proper operation,
- updates,
- fixing bugs,
- adding features,
- extending to new applications,
- change user configurations)

Good comment

- how it works**
- why it is done this way**
- how to change**

Bad comment

- what it is doing**

**1.1.3. Memory-mapped Architecture**

In a system with *memory mapped I/O*,

- I/O devices are connected like memory
- I/O devices are assigned addresses, and software accesses I/O using these addresses
- software inputs from an input device
  - same instructions as a memory read
- software outputs from an output device
  - same instructions as a memory write.

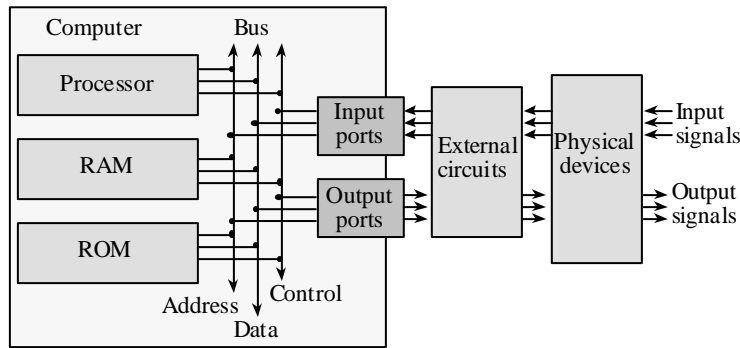


Figure 1.3. A memory-mapped computer system.

The *bus* contains

- address (input, output, RAM or ROM),
- data, and
- control information

The address specifies

- which slave module will communicate with the processor
- one address per memory access cycle

The data contains

- information that is being transferred.

Control signals specify

- the direction of the transfer.

We call a complete data transfer a *bus cycle*. The processor always controls

- the address (where to access),
- the direction (read or write), and
- the control (when to access.)

Type	Address Driven	Data Driven by	Transfer to
Read Cycle	processor	RAM, ROM, Input	processor
Write Cycle	processor	processor	Output or RAM

Table 1.7. Simple computers generate two types of cycles.

Asynchronous *Serial Communications Interface* (SCI)

**Interaction between PC/9S12**  
9S12 network

The synchronous *Serial Peripheral Interface* (SPI)

- Shift registers
- Liquid Crystal Display (LCD) drivers
- Analog to Digital Converters (ADC)
- Digital to Analog Converters (DAC)**
- Other microprocessors

The *timer* features on the 9S12:

**Fixed periodic rate interrupts**

- Computer Operating Properly (COP) protection against software failures
- Pulse accumulator for external event counting or gated time accumulation

**Pulse Width Modulated outputs (PWM)**

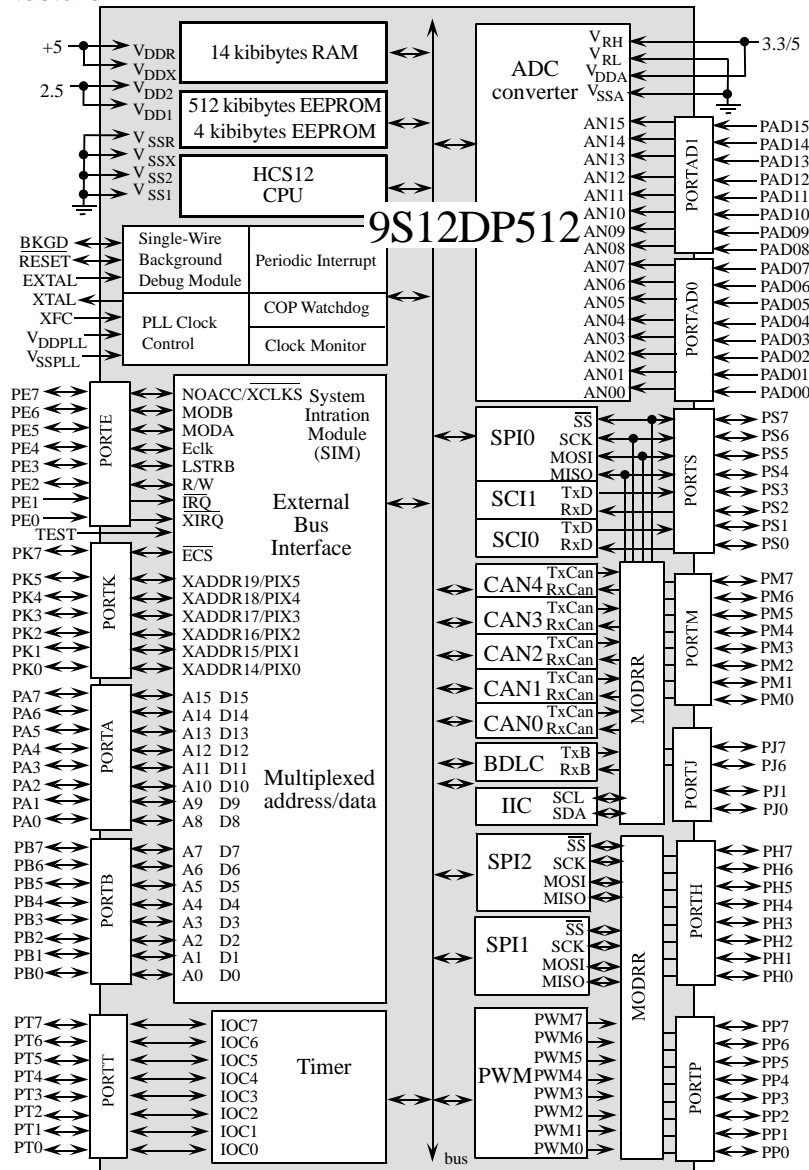
- Event counter system for advanced timing operations

**Input capture used for period and pulse width measurement**

- Output compare used for generating signals and frequency measurement

**ADC systems are available with a 10-bit resolution.**

9S12DP512 Architecture



Block diagram of a Freescale 9S12C32.

The standard address map of a **single chip 9S12DP512** is as follows:

Address	Size	Device	Device	Contents
\$0000 to \$03FF	1 Kib	I/O ports	Input/output devices	Access external devices
\$0400 to \$07FF	1 Kib	EEPROM	Electrically erasable PROM	Fixed constants
\$0800 to \$3FFF	14 Kib	RAM	Random Access Memory	Variables and stack
\$4000 to \$FFFF	48 Kib	EEPROM	Electrically erasable PROM	Programs and fixed constants

Table 2.3. The 9S12DP512 has 512 Kibibytes of EEPROM and 14 Kibibytes of RAM.

The one serial port will be used for both the on-chip debugger and your communication software

A **device driver** is a collection of software functions that allow higher level software to utilize an I/O device.

Collection of public methods (functions)

`SCI0_Init`

`SCI0_InChar`

`SCI0_OutChar`

Collection of private objects (functions, globals, I/O ports)

`SCI0CR2`

`SCI0BD`

`SCI0SR1`

`SCI0DRL`

**complexity abstraction**

**divide a complex problem into simple subcomponents**

**functional abstraction**

**divide a problem into modules**

**grouped by function**

**\*\*\*\*\*Show Fixed point example in Lab 1d\*\*\*\*\***

**Define fixed-point**

**Why we use fixed-point**

**How to use decimal fixed-point**