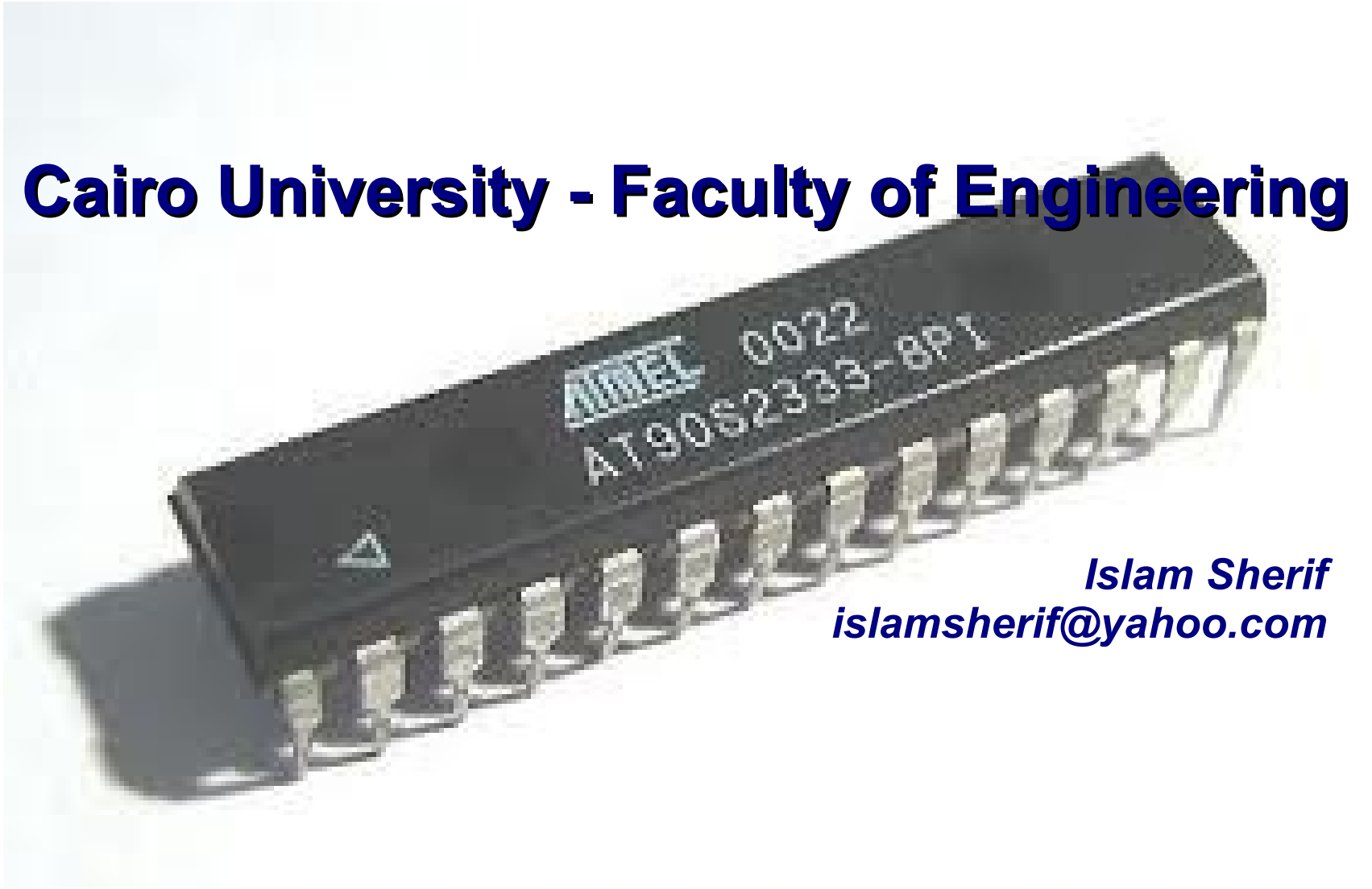
Introduction to Micro-controllers



Cairo University - Faculty of Engineering

Islam Sherif
islamsherif@yahoo.com

Agenda

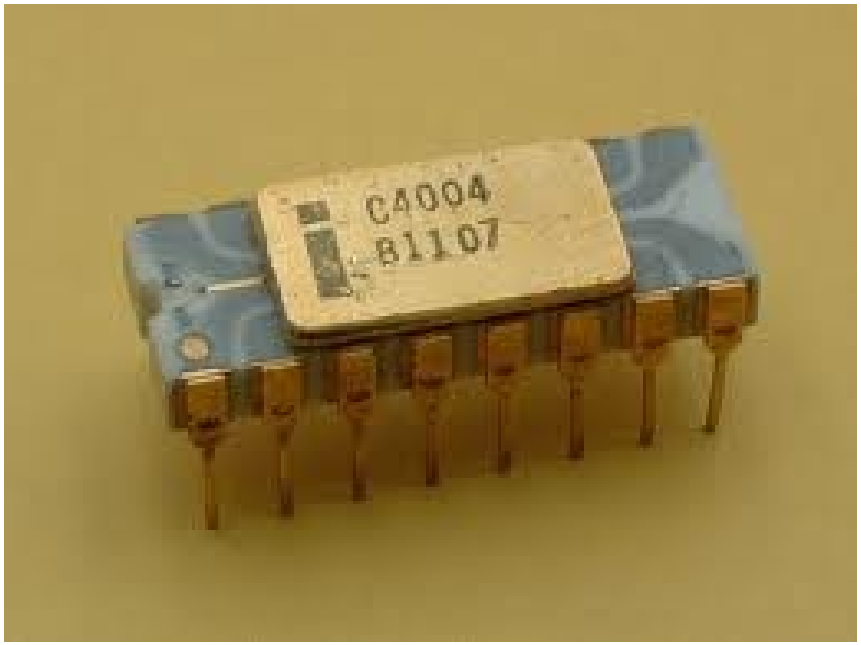
- Introduction
- History Of Microcontrollers
- Different Microcontroller Features
- Suppliers Of Microcontrollers
- Programming Microcontrollers
- Tools To Use With Microcontrollers
- What's Next ?

Introduction

- Microcontrollers are special types of processors
- They are built to integrate a lot of functionality inside one programmable chip !
- The core of the chip is a processor
- Many things can be done using a microcontroller

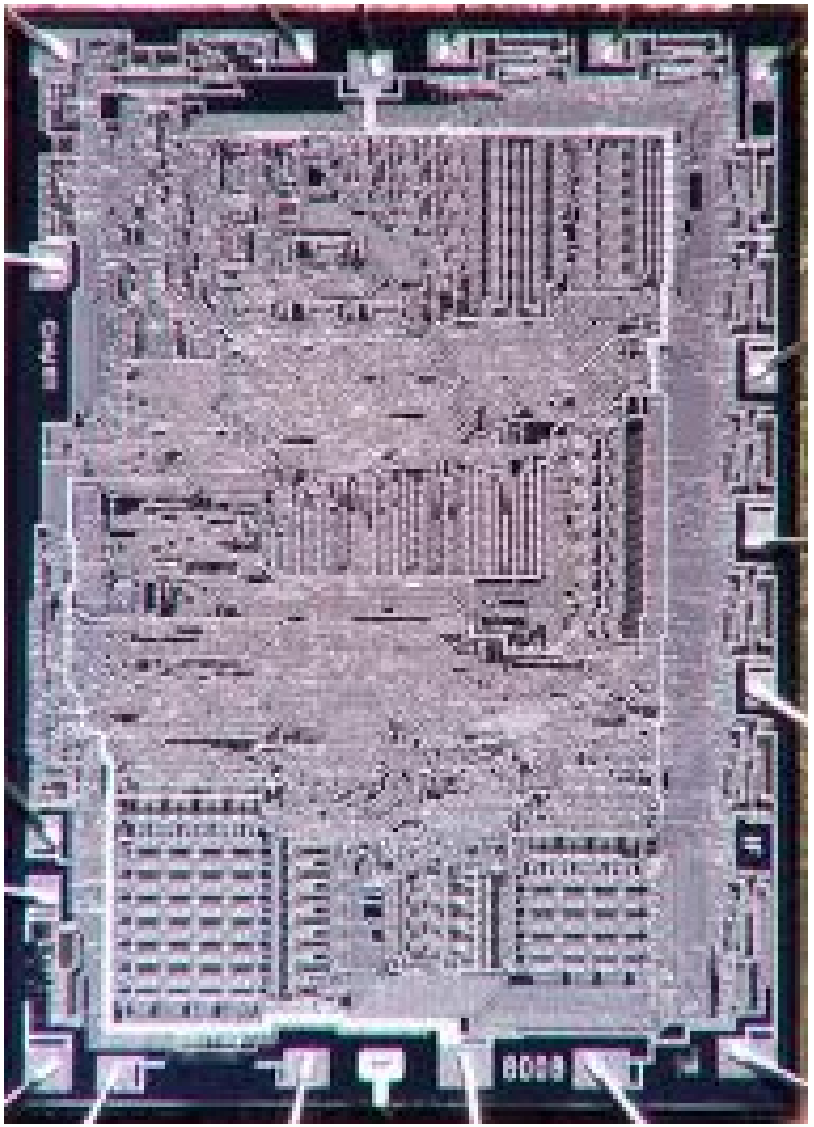
History of Microcontrollers

- Microprocessors are the core for computers we use since their beginning and until now
- Microprocessors were invented in the early 70's by Intel
- The first microprocessor was the Intel 4004 announced in November 1971



History of Microcontrollers

- In April, 1972 Intel announced the 8008, the first 8-bit microprocessor !

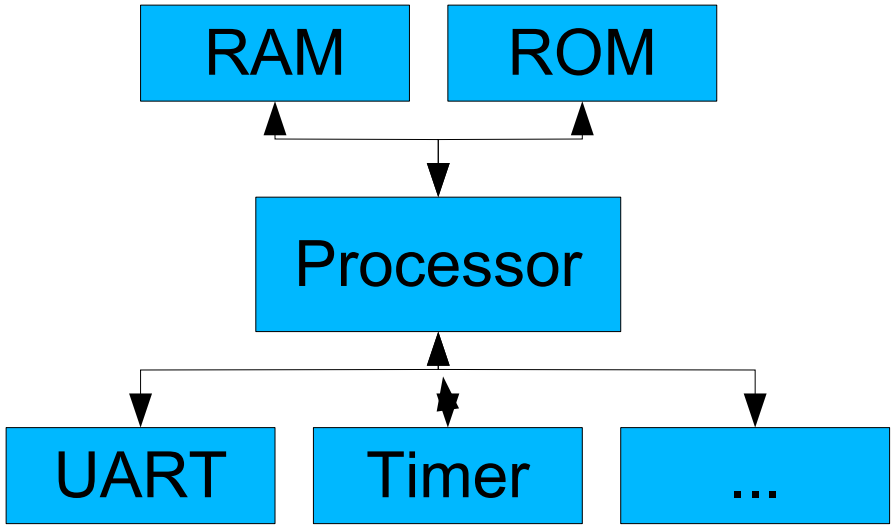


History of Microcontrollers

- In 1978, the 8086 processor was released. It is the father of the most famous line of processors, x86, commonly used in all Intel-based computers until now !

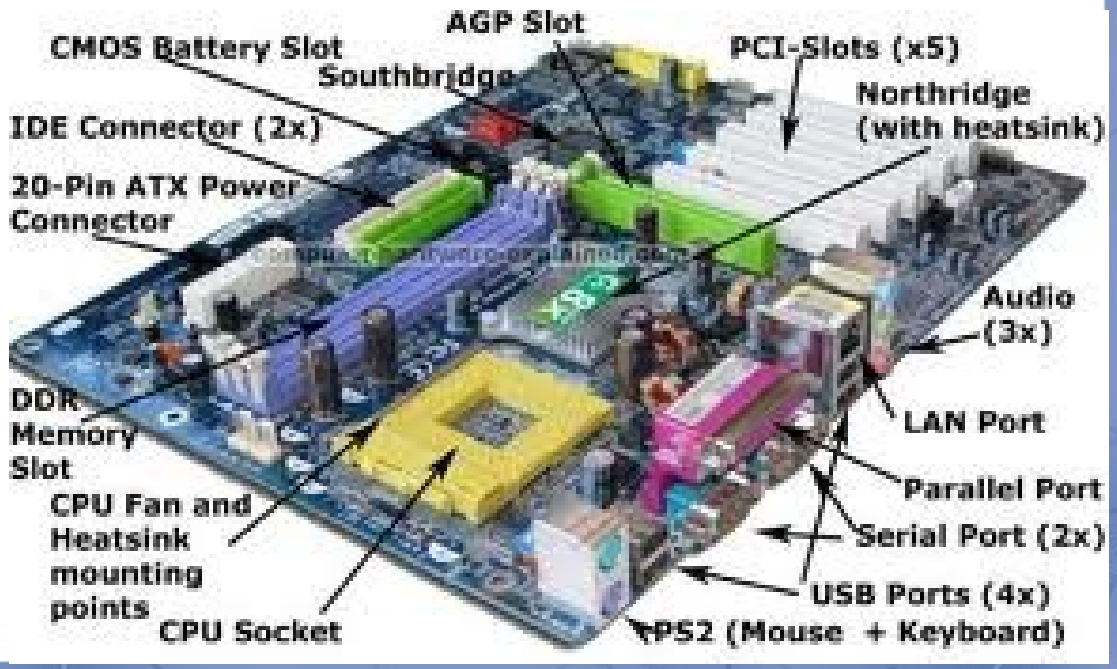
History of Microcontrollers

- Any computer system needs the following:
  - Processor
  - RAM & ROM
  - Peripherals



History of Microcontrollers

- In traditional computer systems, the mother board integrates together all system components



History of Microcontrollers

- Microcontrollers try to integrate as much stuff as possible in a single chip
- It serves for applications where portability is needed
- It also helps when power consumption is needed to be lowered

History of Microcontrollers

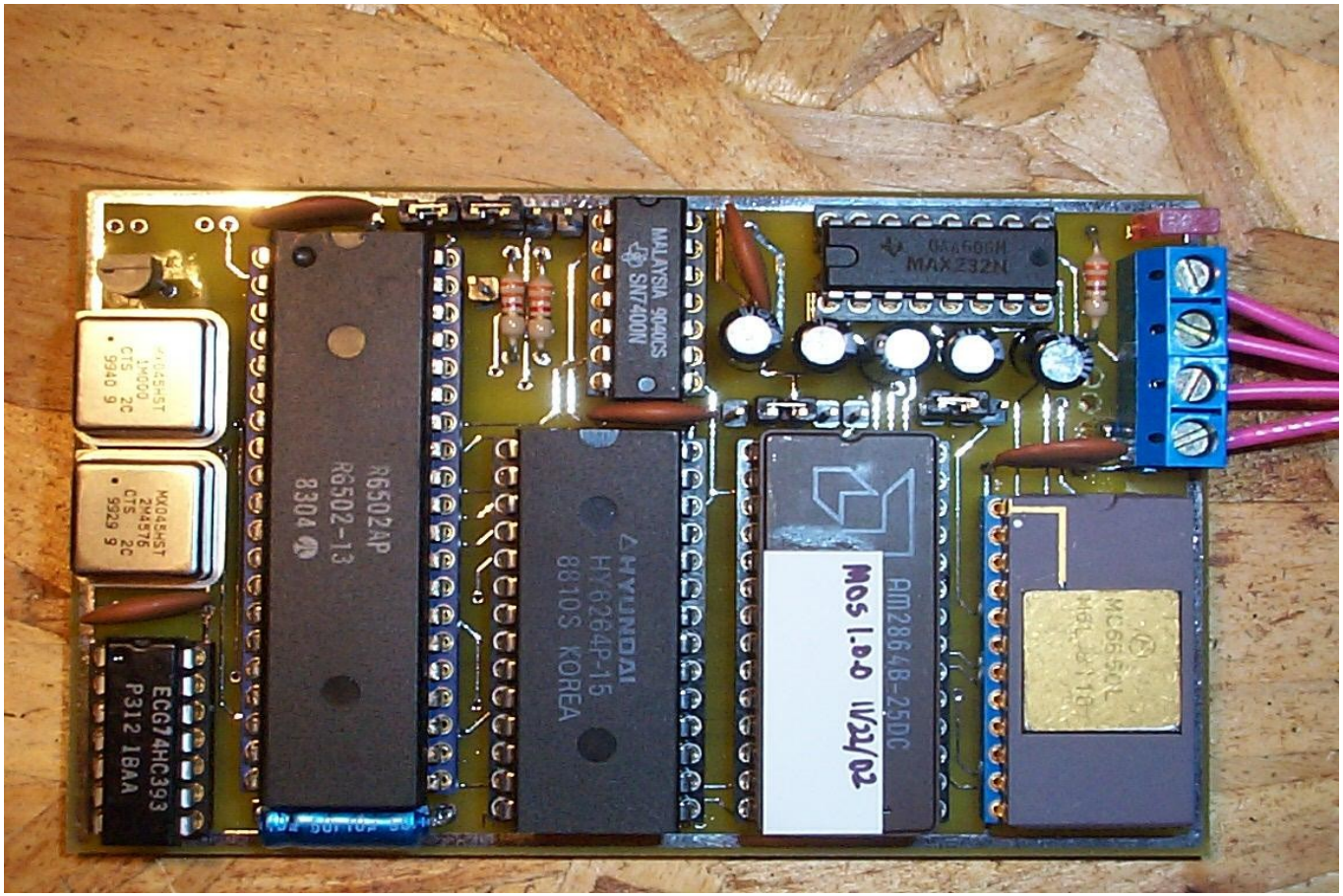
- The first microcontroller was born in 1980, the MCS-51, (*aka* 8051)
- 8-bit CPU
- 16-bit address bus (64 KB of memory)
- Harvard architecture
- Integrated UART
- 2 16-bit Timers
- All in one chip !

History of Microcontrollers

- At the beginning, microcontrollers enabled building single board computers with few chips
- Memory integration remained a problem
- It was normal to find a controller and a RAM chip as well as a ROM

History of Microcontrollers

- Example, an SBC (Single Board Computer)



History of Microcontrollers

- RAM used to come in chips separately, to enable application programs to store data e.g. variables, and arrays
- ROM is rather used to store code when system is powered down
- First ROM chips where programmed using 12V, and erased using UV

History of Microcontrollers

- The chip needed to be removed from the board for programming/erasure !
- About 30 minutes were needed per erasure !
- Chip life was limited to 20 ~ 30 programming cycles

History of Microcontrollers

- The next break through for microcontrollers came with the introduction of flash memory
- Flash memory is electrically erased and programmed in fractions of a second
- Flash memory also has a much longer lifetime (endurance) of few thousands of erase/program cycles
- Flash memory requires high voltage (~12V) for erasure and programming

History of Microcontrollers

- As flash memory advanced, it started to get integrated inside microcontrollers as well !
- RAM also started to get integrated (in a limited amount) inside controllers
- Microcontrollers started to be a computer in a single chip

Different Microcontroller Features

- As microcontrollers advanced more and more features were added, specialized microcontrollers were also created for special purposes

Different Microcontroller Features

- The most important part of the microcontroller is the CPU used in the chip
- The famous 8051 used the 8051 CPU architecture
- CPUs are also defined by their “bus-width”
- The wider the bus, the more powerful the CPU is

Different Microcontroller Features

- Many CPUs exist nowadays
- The most common classifications are:
  - Bus width
    - 8-bits
    - 16-bits
    - 32-bits
  - RISC vs. CISC architecture

Different Microcontroller Features

- Based on the CPU used, suppliers create “microcontroller family” creating many variants of the same CPU but with different options
- 8-Bit microcontrollers are suitable for small scale applications, and low power devices
- 16-Bit microcontrollers are mid-range to large applications
- 32-Bit microcontrollers are usually used in mobile phone devices and digital media players

Different Microcontroller Features

- ADC is another very important feature of many modern microcontrollers
- ADC (Analog to Digital Converter) is a part (peripheral) of many controllers that is used to measure analog signals
- It is used to read measurements of many physical sensors like temperature, pressure, humidity, etc.

Different Microcontroller Features

- UART (Universal Asynchronous Receiver Transmitter)
- It is almost a standard component in many microcontrollers
- It is used for serial communications with different electronic devices, like e.g, a personal computer
- It can be used to support the famous RS-232 communication standard

Different Microcontroller Features

- Timer
- Timer is a time-keeping device
- It is used to do time measurements and periodic tasks
- Timers are basically digital counters with a controlled clock source

Different Microcontroller Features

- Counter
- A counter is a simple digital counter that can count an external trigger (e.g. negative edge or a positive edge)
- It is usually configured to select among different triggers
- A counter in combination with a timer can be used to measure frequencies as well

Different Microcontroller Features

- PWM
- Pulse-Width Modulation is a technique commonly used to control electrical loads like heaters, light sources, dc motors, …
- PWM peripheral is usually a set of timer/counter/comparator combined together to generate a controllable square wave with a defined duty cycle and frequency

Different Microcontroller Features

- Many other peripherals now exist, and are almost standard package for many microcontrollers, like:
  - SPI
  - IIC
  - EEPROM
  - Watchdog timer

Suppliers of Microcontrollers

- The first supplier was Intel, with the famous 8051 controller
- Another key player, is Motorola (now Freescale Semiconductors)
- Atmel
- Microchip
- Renesas

Suppliers of Microcontrollers

For example:
Atmel has a rich set of microcontroller products:

Microcontroller / Family	Bus Width	CPU
AT89xxx	8-bit	8051
AVR	8-bit	AVR
XMega	8-bit	AVR
AVR UC3	32-bit	AVR32
ATSAMxx	32-bit	ARM

Suppliers of Microcontrollers

- Example controller, AT89C52
  - CPU based on 8051
  - 24 MHz clock speed
  - 8 KB Flash memory
  - 256 Bytes RAM
  - 3 x Timers
  - 1 x UART
  - 40-pin DIP package

Suppliers of Microcontrollers

- Example controller, AT89C52
  - CPU based on 8051
  - 24 MHz clock speed
  - 8 KB Flash memory
  - 256 Bytes RAM
  - 3 x Timers
  - 1 x UART
  - 40-pin DIP package

Suppliers of Microcontrollers

- Example controller, ATMega16
  - AVR CPU
  - 16 MHz clock speed
  - 16 KB of Flash memory
  - 1 KB of RAM
  - 2 x 16-bit timer, 1 x 8-bit timer
  - 8 channel ADC (10-bit)
  - 4 x PWM channels
  - 40-Pin DIP package

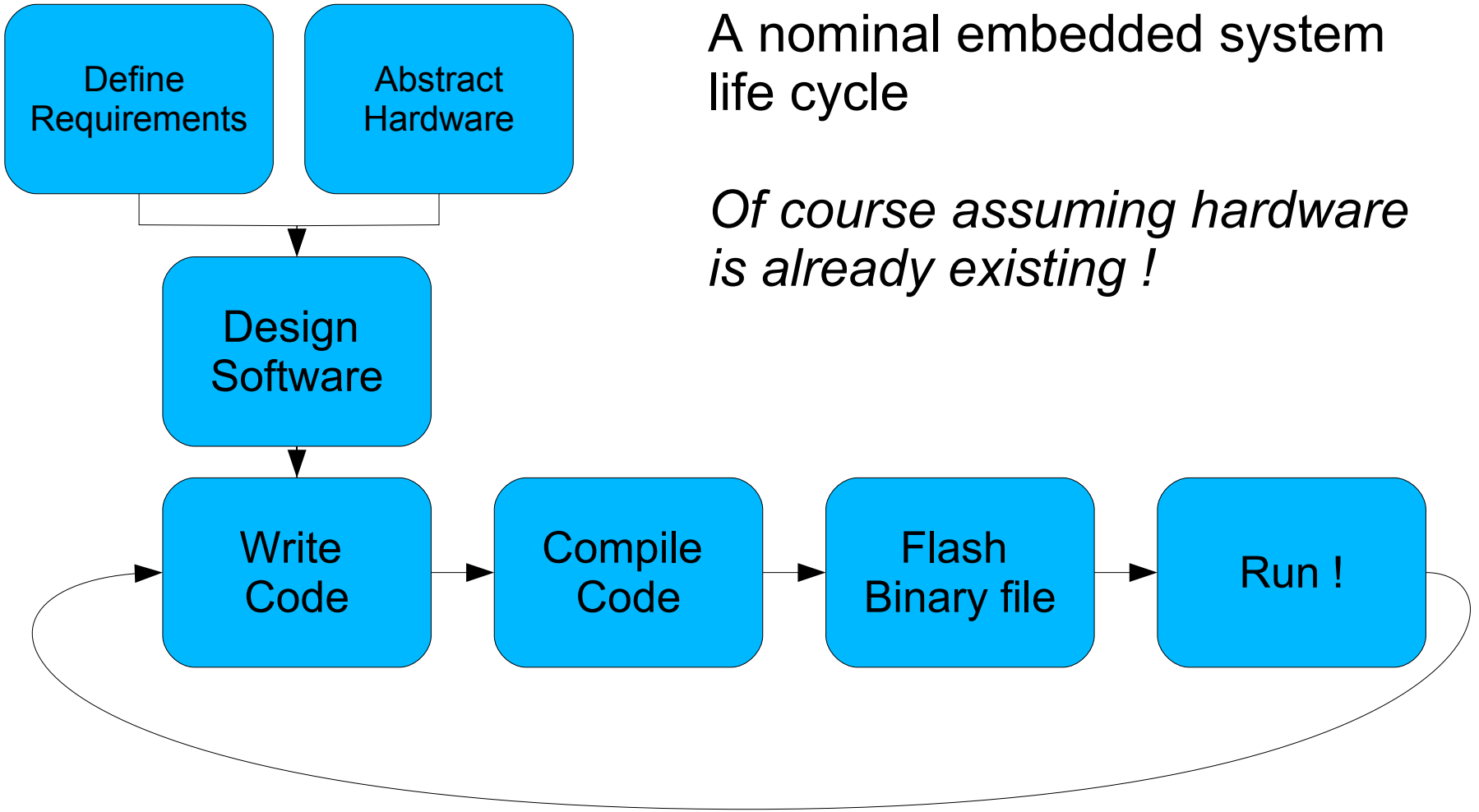
Suppliers of Microcontrollers

- Example controller, MC9S12XEP100
  - CPU12X 16-bit CPU
  - 50 MHz clock speed
  - 1 MB of Flash memory
  - 64 KB of RAM
  - Huge count of peripherals !
  - 144-Pin LQFP package

Programming Microcontrollers

- Programming controllers is all the steps needed to make something useful out of a controller
- Many steps and tools are involved in order to create a useful program “application” on a controller
- The term sometimes also refers to downloading the code to the controller

Programing Microcontrollers

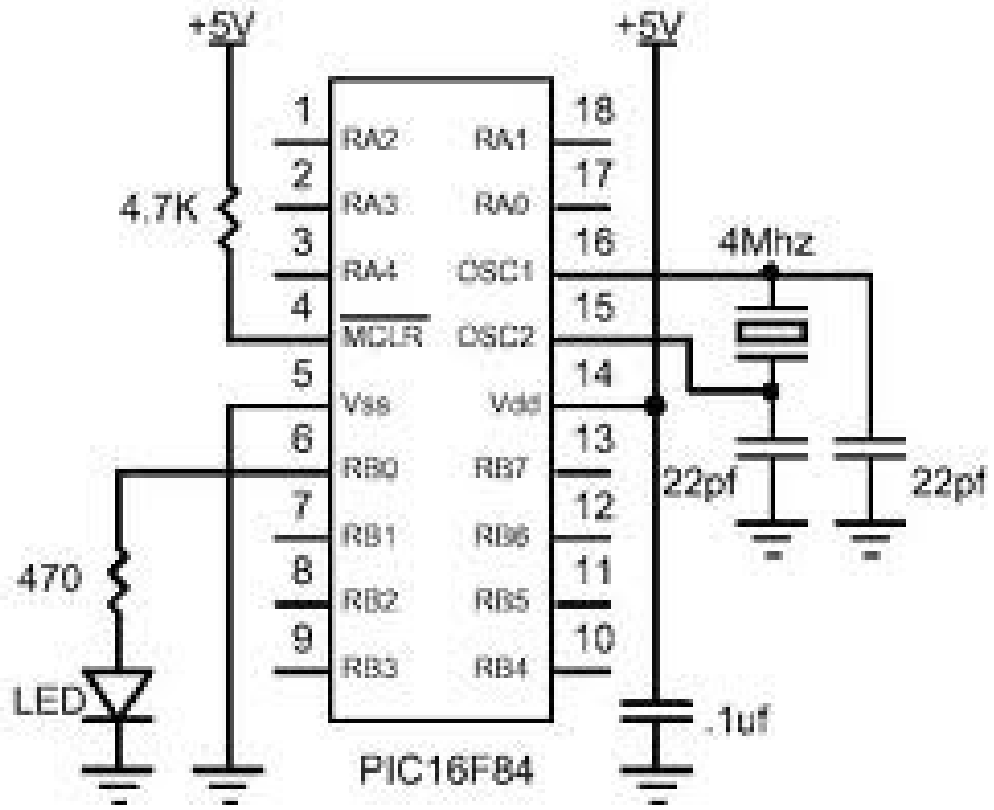


A nominal embedded system life cycle

Of course assuming hardware is already existing !

Programing Microcontrollers

- Example:
- Let's try to create a simple system that flashes a LED every 1 second using the following hardware



Programming Microcontrollers

Q : What is required ?

A : Turn on LED for 1 second then turn off LED for 1 second

Programming Microcontrollers

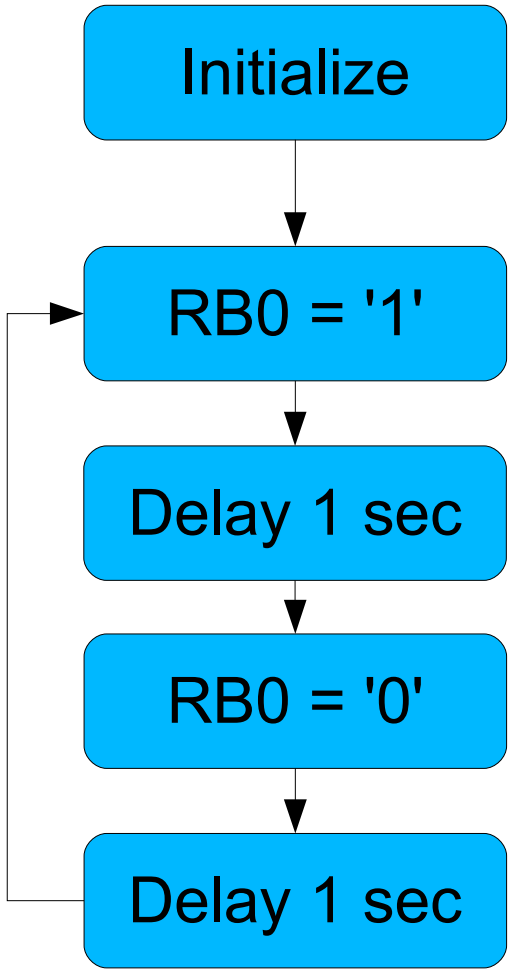
Q : How to control LED ?

A : LED turns ON when RB0 has a logical '1', and turns OFF when RB0 has a logical '0'

Programming Microcontrollers

Q : How to create the software ?

A :



Programming Microcontrollers

- Writing code is done after selecting a programming language.
- The two options that are most common are:
  - ANSI C (over 90% of the cases nowadays)
  - Assembly

Programming Microcontrollers

- ANSI C is the most common and most convenient language for embedded systems nowadays
- ANSI C enjoys a very light footprint, and is portable across different platforms allowing code reuse and independence on a certain CPU
- ANSI C compilers exist for almost all microcontroller targets

Programming Microcontrollers

- Assembly language is most efficient in code size and performance
- It needs a great deal of effort to accomplish something useful
- Assembly language also requires a great deal of experience to deal with
- It also suffers from lack of portability, with small chances of reuse in case a new/better target is available

Programming Microcontrollers

- Downloading the resulting application to the microcontroller requires a special interface/device
- Many cases existed
- In the old ages, of UV EPROM, this was done using a special device, that programmed the EPROM
- The EPROM IC was usually removed from its socket, erased, programmed, then fixed again !

Programming Microcontrollers

- Later on, with Flash memory, either the ROM chip or the microcontroller itself was removed, inserted into the programmer, programmed then inserted in its socket once again
- Then as chips grew bigger and bigger, SMT technology was introduced where it is not easy to insert/remove the chip (it is soldered and fixed in its place)

Programming Microcontrollers

- As SMT chips appeared, In-System Programming (ISP) also appeared where programming the chip required minimal connections to the chip, and it is usually done in the hardware designed already so that rather the “board” is programmed not the controller !

Tools To Use With Microcontrollers

- Many tools are in the loop when considering development for microcontrollers
- Software tools include:
  - Compilers
  - IDEs
  - Simulators
  - Debuggers

Tools To Use With Microcontrollers

- Hardware tools include:
  - Programmer / In-System programmer
  - Debugger
  - Emulator
  - Oscilloscope / Logic analyzer

Tools To Use With Microcontrollers

- A compiler is that program that is used to translate a high level language (e.g C) to machine language equivalent
- Many free and open source compilers exist for embedded targets like:
  - GCC (GNU C Compiler)
  - SDCC (Small Device C Compiler)

Tools To Use With Microcontrollers

- Simulators usually try to simulate code running on a microcontroller in relation to the outside circuit
- Simulators are usually expensive programs, and not easily available for each target
- Example is Proteus Simulation SW
  - http://www.labcenter.com/products/basicsim.cfm

Tools To Use With Microcontrollers

- When running code on a microcontroller, it is usually running on a different processor than the one that compiled the code !
- It is usually very hard to know what is going on inside the remote CPU !
- A debugger is usually a hardware / software combination of tools that allow you to inspect what is going on inside the remote target

Tools To Use With Microcontrollers

- Debuggers usually allow limited functionalities like:
  - Limited number of breakpoints
  - Watching limited amount of memory (simultaneously)
- Emulators are rather more complex and expensive devices
- They can be used for advanced debugging of remote targets

Tools To Use With Microcontrollers

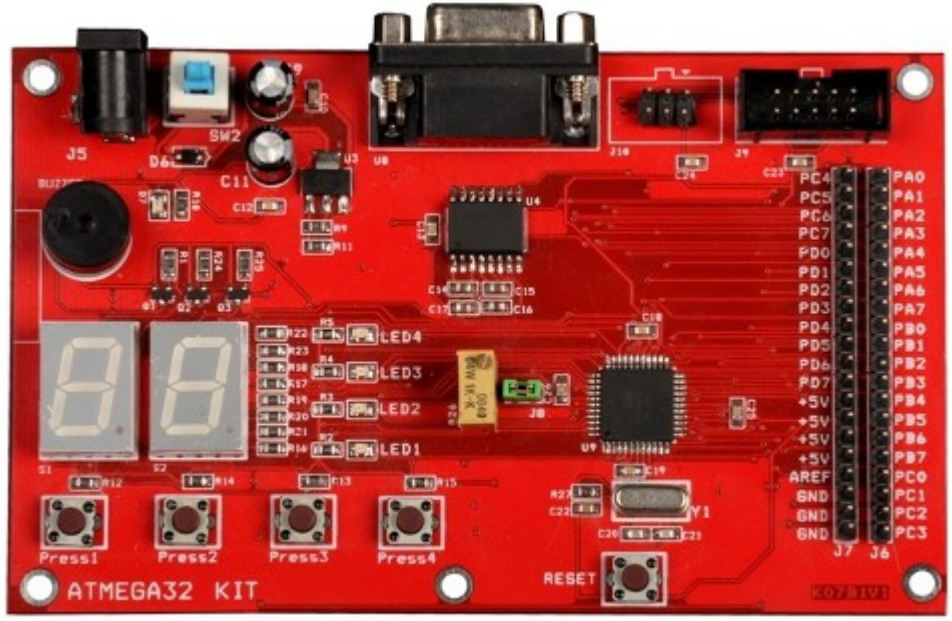
- An emulator can provide unlimited number of breakpoints, watch all the memory at once
- It also provides advanced features like integration with operating systems and measuring CPU performance metrics and so on
- However, they are very expensive and target specific !

What's Next ?

- Learning embedded systems needs trials and trials
- It is like any other field of engineering an experience gained !
- Start by learning something about a simple 8-bit microcontroller and how to program it using C language
- Many pages on-line contain tutorials and useful tips

What's Next ?

- You can build your own hardware sets or buy ready made
- You can also use ready kits which can help you a lot in the beginning of learning



What's Next ?

- The best way is to set a target for yourself and try to reach it !
- Good Luck !