# Introduction to Embedded System (IES)

Team Emertxe

# Pass on the ball

1) Let us get the ball rolling ☺

2) Introduce yourself

3) Expectations from this program

# Course span-out


EMERTXE

# Introduction to Embedded System

# Let us ponder…

- What do you understand as Embedded System?
- Name few examples of Embedded System from our daily usage
- How different they are from a general purpose system (GPS)?
- What difference can be there between multiple Embedded Systems?

**"Any Hardware System which is intended to do a specific task can be called as an Embedded System"**

# ES - Examples

Examples – Automotive, Satellite communication, Consumer electronics, Medical, Imaging, Robotics etc..

# ES - Classifications

- Embedded systems can be classified into four different categories:
  - Stand alone Embedded System - Performs a single and specific functionality
  - Real time Embedded System - Provides real time guarantee in terms of response and predictability
  - Network appliances - High focus on packet processing
  - Mobile devices - Hand held devices
- Embedded systems - “Choice-points”
  - Embedded Systems type and expectations drastically vary
  - In order to meet customer needs specific “Choice-points” to be decided
  - These are popularly known as “Design parameters”

# ES – Choice points

- Let us take an example of Mobile Vs. Automotive Embedded device
- Try to compare various choice points

| Parameter | Automotive | Mobile |
|---|---|---|
| Safety | ↑ | ↔ |
| Cost sensitivity | ↔ | ↑ |
| Performance | ↑ | ↔ |
| Reliability | ↑ | ↔ |
| Time to market | ↑ | ↓ |
| Unit cost | ↑ | ↔ |

- For each choice made corresponding compromise factor to be considered
- Optimizing every other parameter is not a possible option

# ES – Design metrics

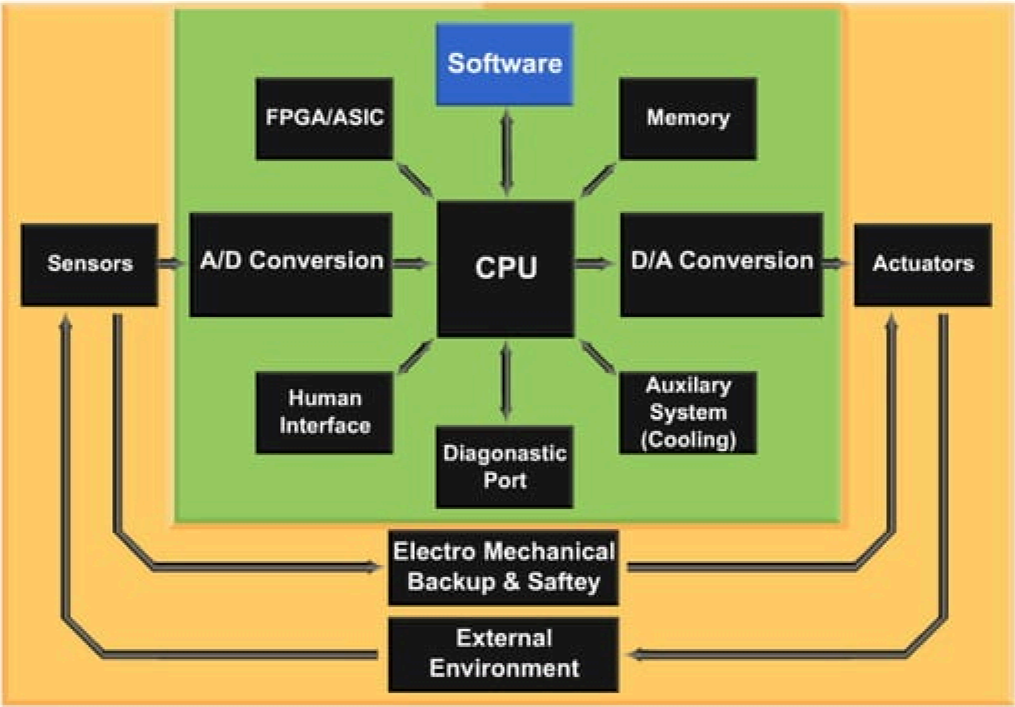
- Time to Prototype
- Power
- Performance & Correctness
- Size
- NRE
- Maintainability & Flexibility
- Safety
- Unit Cost
- Time to Market

# ES - Challenges

- Efficient Inputs/Outputs
- Embedding an OS
- Code optimization
- Testing and debugging

# Embedded System Components

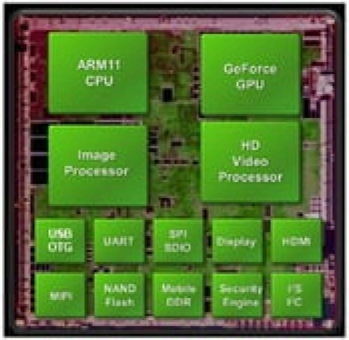
# ES - Components




ΣMERTXE

# CPU – µC | µP

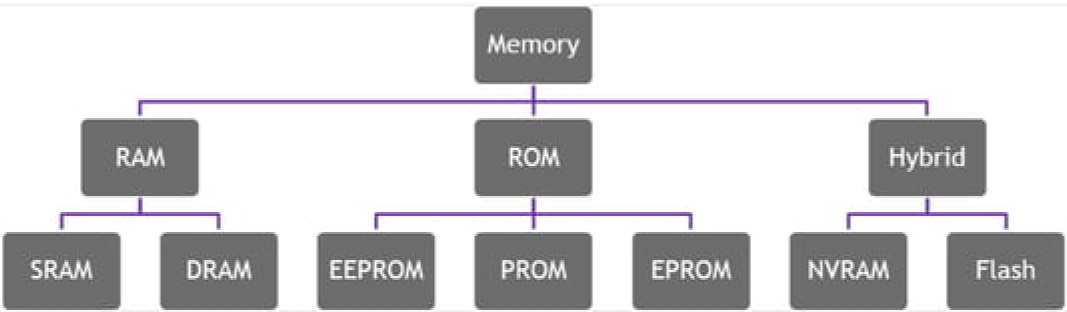
- The “Brain” of the system
- Implementation can be:
  - Universal
  - Digital Signal Processors (DSP)
  - System On a Chip (SoC)




- “On-the board” v/s “On-the-chip”
- ASIC/SoC tape-out process provides lot of advantages than using Universal processors
- However it requires huge up-front investment (in terms of millions!)

# Memory

- Various types of memory exist with specific characteristics
- Based on Volatility, Write-ability, Cost and Speed - Parameters to compare
- Higher level group/category is provided as follows:



# Memory - compare!

| Type | Volatile (Y/N) | Writable (Y/N) | Erase Size | Max Erase Cycle | Cost per byte | Speed |
|---|---|---|---|---|---|---|
| SDRAM | Y | Y | Byte | Unlimited | Expensive | Fast |
| DRAM | Y | Y | Byte | Unlimited | Moderate | Moderate |
| Masked ROM | No | No | N/A | N/A | Inexpensive | Fast |
| PROM | No | Once | N/A | N/A | Moderate | Fast |
| EPROM | No | Yes | Entire Chip | Limited | Moderate | Fast |
| EEPROM | No | Yes | Byte | Limited | Expensive | Fast (R) Slow (W/E) |
| Flash | No | Yes | Sector | Limited | Moderate | Fast (R) Slow (W/E) |
| NVRAM | No | Yes | Byte | Unlimited | Expensive | Fast |

# Memory - Space

| Memory Space | Memory Area | Contents |
|---|---|---|
| Name | - | Name and stack diagram of words |
| Code | CODE | Executable machine code |
| Data | DATA | Application & system variables, stack, buffers |
| Constant Data Space | CONST | Application & system constants, Virtual machine code |
| Local name space | DATA | Local and other related data structure |

EMERTXE

# Components...

### ADC

- Analog to Digital Convertors
- Safety and Security
- Comfort and convenience
- Driver information
- Multimedia
- Engine controls

### DAC

- Multimedia
- Motor controls

### HID

- Displays
- Input devices

### Actuators

- Binary (Relays)
- Continuous (Motors)

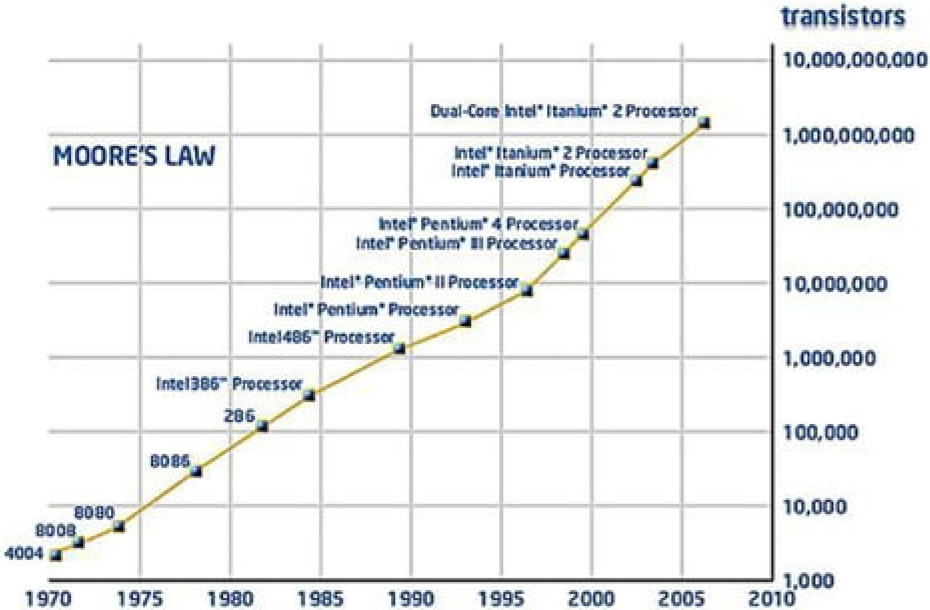
### Sensors

- Light sensors
- Water sensors

**FPGA and ASIC - Taping out custom chips for optimization purpose**

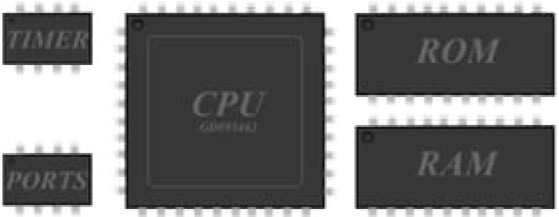
# HW – Processors and Controllers

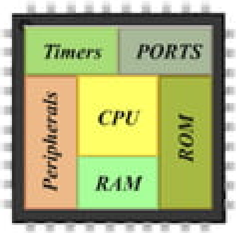
# Evolution of processors




EMERTXE

# Micro processor v/s Micro controller




- All separate components
- More flexible
- More design complexity

- All components in a single chip
- Less flexible
- Less design complexity

# Choosing a Micro-processor:

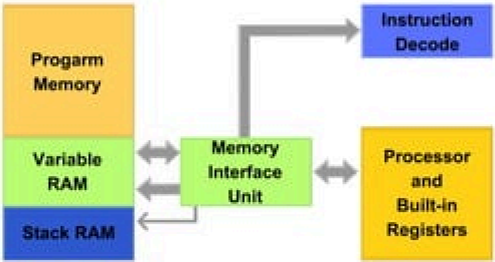
**Choice points:**

- Application
- Performance
- Price
- Availability
- Availability of Tools
- Special Capabilities

**Classifications:**

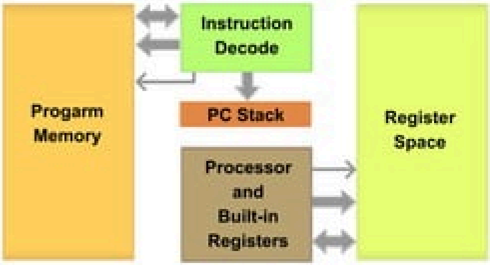
- Bit-depth
- Architecture
- Use based:
  - GPP - Proper & Micro controllers
  - ASP - DSP & ASIC

# Von Neumann & Harvard Architecture




Von Neuman Architecture

Shared signals and memory for code and data




Harvard Architecture

Physically separate signals and storage for code and data

# RISC v/s CISC

## RISC:

**Advantages:**

- Moved complexity from HW to SW
- Provided a single-chip solution
- Better usage of chip area
- Better speed
- Feasibility of pipe-lining
  - Single cycle execution stages
  - Uniform Instruction format

**Disadvantages:**

- Greater burden on SW

## CISC:

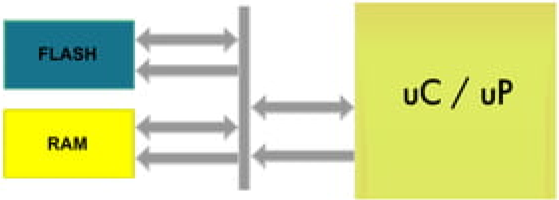
**Advantages:**

- Moved complexity from SW to HW
- Compact code
- Ease of compiler design
- Easier to debug

**Disadvantages:**

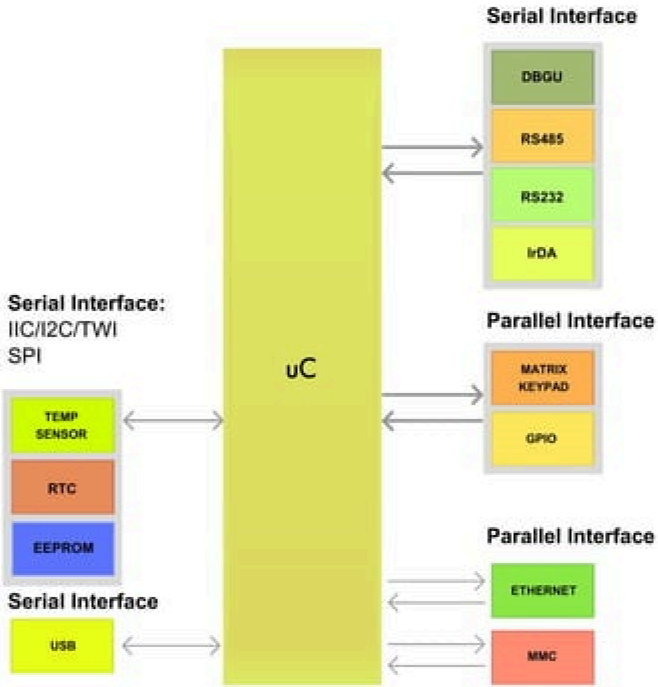
- Increased design errors
- Longer design time
- Performance tuning unsuccessful
- High complexity
- Time to market increases

# Interfacing



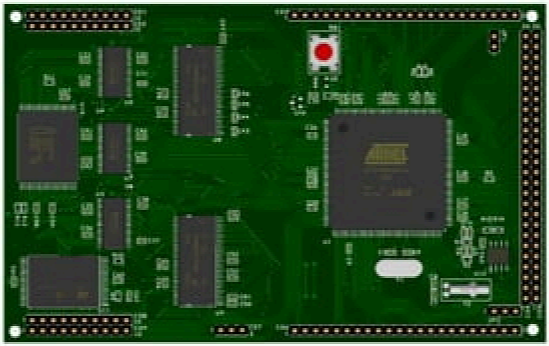

- Processor has to interface with memory and various I/O devices
- Bus protocols are used for communication
- Consisting of:
  - Address bus
  - Data bus
  - Control lines:
    - CS
    - RD
    - WR

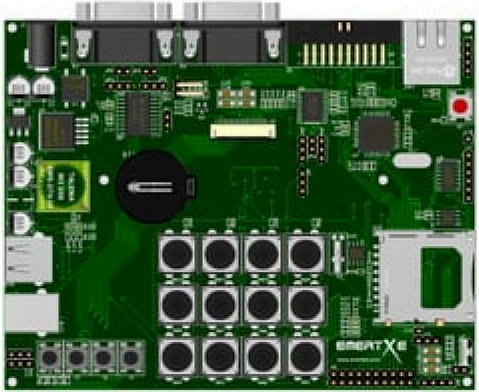
# Peripheral Interfacing




- Processor has to interface with memory and various I/O devices
- Bus protocols are used for communication
- Consisting of:
  - Address bus
  - Data bus
  - Control lines:
    - CS
    - RD
    - WR

# HW Architecture

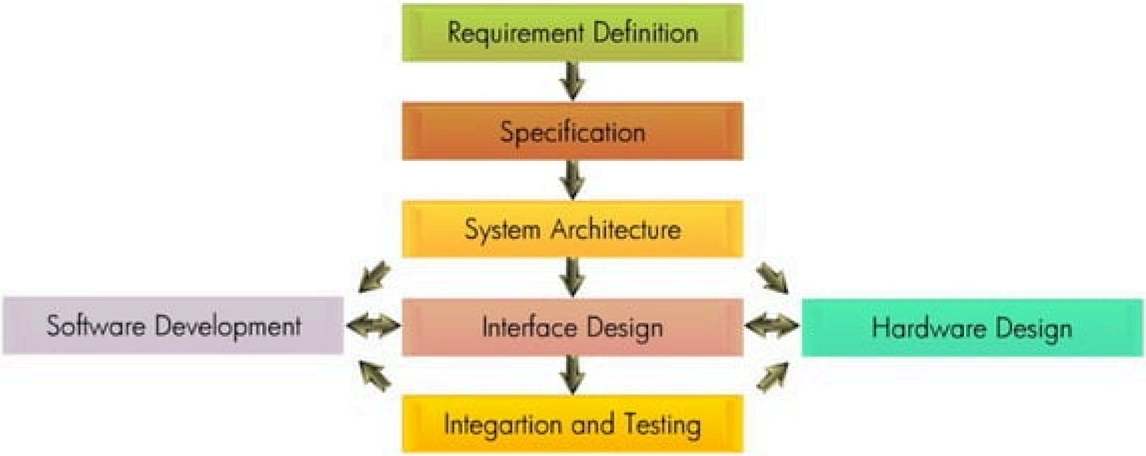





HW board with interfacing

# SW – Development, Architecture, Environment

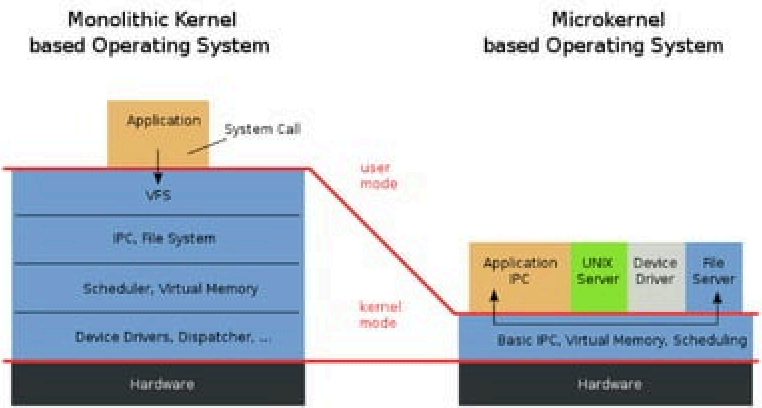
# SW development




In Embedded Systems, SW and HW development happens in a combined manner. SW development details and life cycle are covered in SDLC topic in detail.

# Architectures

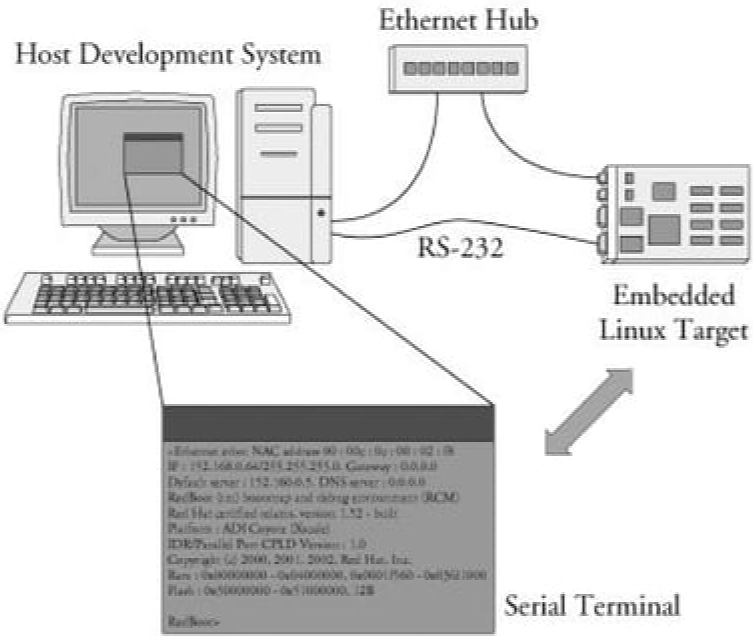
- Super loop
- Interrupt controlled
- Co-operative multi tasking
- Pre-emptive multi-tasking
- Micro and Monolithic Kernel



# Dev Environment

- Embedded development environment is quite different and relatively complex than application development
- The simple reason the embedded software developed in a 'host' machine and executed on a 'target' which makes the compilation and debugging process quite challenging
- Popularly known as Integrated Development Environment (IDE), this environment consist of the following key candidates:
  - Editor - For scripting Embedded program
  - Configuration
  - Tool chain - Cross compiler, linker and associated tools
  - Target download and debug environment - Installation and test

# IDE – Pictorially…

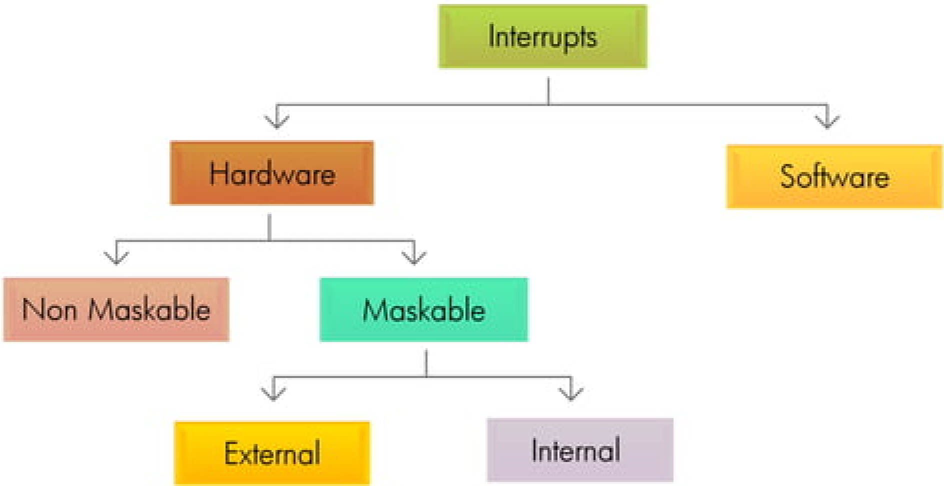


# Interrupts

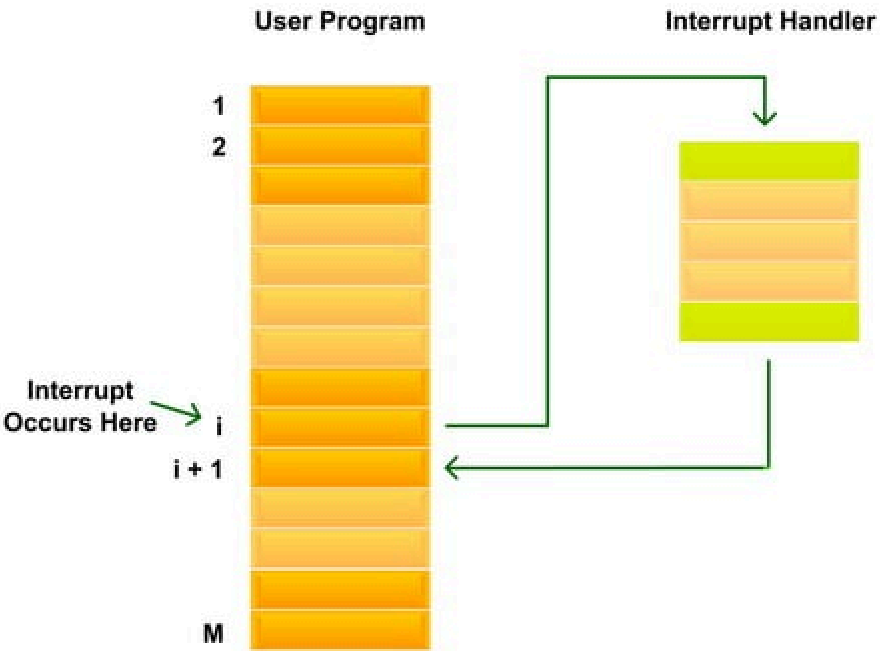
# Interrupt - Basics

- An interrupt is a communication process set up in a microprocessor or microcontroller in which:
  - An internal or external device requests the MPU to stop the processing
  - The MPU acknowledges the request
  - Attends to the request
  - Goes back to processing where it was interrupted
- Interrupt sources
  - External
  - Timers
  - Peripherals
- Interrupt v/s Polling
  - Loss of Events
  - Response
  - Power Management

# Interrupts - Classification



# Interrupts - Handling



# Interrupt Service Routine

- Similar to a subroutine
- Attends to the request of an interrupting source
  - Clears the interrupt flag
  - Should save register contents that may be affected by the code in the ISR
  - Must be terminated with the instruction RETFIE
- When an interrupt occurs, the MPU:
  - Completes the instruction being executed
  - Disables global interrupt enable
  - Places the address from the program counter on the stack
- Return from interrupt

# Interrupt Latency

- Latency is determined by:
  - Instruction time (how long is the longest)
  - How much of the context must be saved
  - How much of the context must be restored
  - The effort to implement priority scheme
  - Time spend executing protected code

Thank You