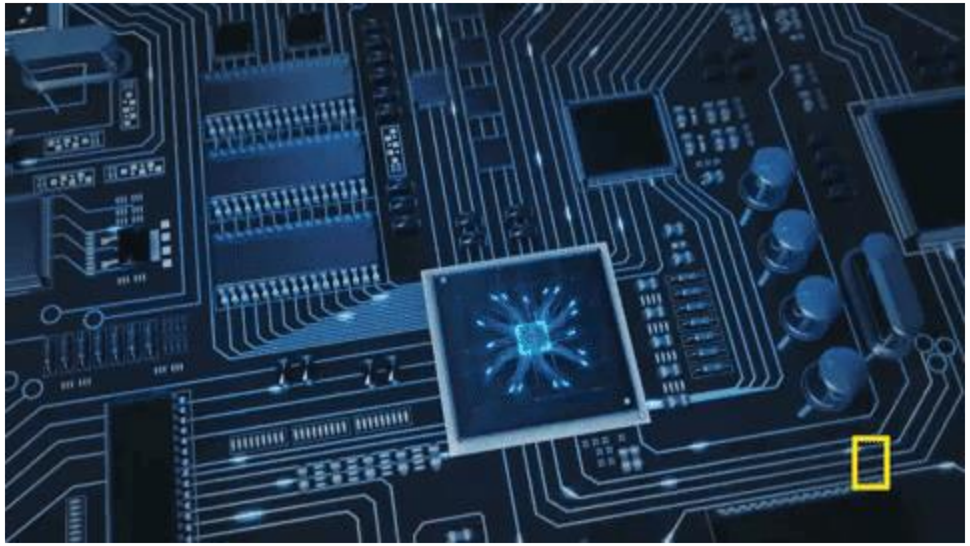
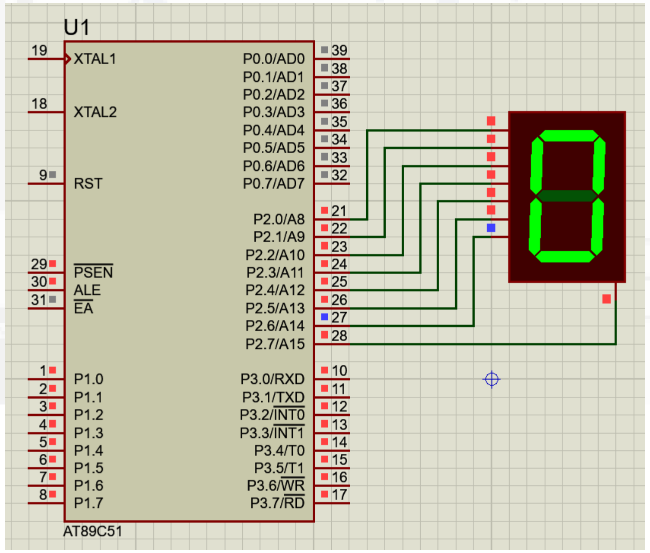
# BEE613B
# Embedded Systems Design
# Module-1: Introduction



**Presented by,**
**Mr.Shreeshayana R**
**Assistant Professor**
**Electrical and Electronics Engineering**
**ATME College of Engineering, Mysuru**

# SESSION-1


U1
AT89C51
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD

# Course Overview

- Course Code: BEE613B
- Course Title: Embedded Systems Design
- Type: Professional Elective
- Prerequisite: C Programming, Microcontrollers Fundamentals
- Contact Hours: 40 Hours

# Course Objectives

1. Introduction to **Embedded System Design**
2. **Characteristics & attributes** of Embedded Systems
3. **Overview** of Embedded System Applications
4. **Software & Hardware development** for RTOS-based Embedded Systems

# Module 1: Introduction to Embedded Systems

- - History, Classifications & Applications
- - Core Components: Microprocessors, Microcontrollers, ASICs, Sensors, Actuators
- - Communication Interfaces, Embedded Firmware, PCB Design

# Module 2: Characteristics & Quality Attributes

- - Operational & Non-Operational Quality Attributes
- - Application-Specific Embedded Systems (e.g., Washing Machine, Automotive)

# Module 3: Hardware-Software Co-design & Development

- - Computational Models in Embedded Systems
- - Embedded Hardware: Analog/Digital Components, VLSI Design, EDA Tools

# Module 4: Embedded Firmware & Development Environments

- - Firmware Design Approaches & Development Languages
- - Cross Compilation, Debugging Tools: Simulators, Emulators, Disassemblers

# Module 5: RTOS-Based Embedded System Design

- - Operating System Basics, Multitasking & Scheduling
- - Tasks, Processes, and Threads in Embedded Systems

# Textbooks & References

1. **Shibu K V, “Introduction to Embedded Systems”, McGraw Hill**
2. NPTEL: IIT Delhi (Prof. Santanu Chaudhary)
3. Arduino & Raspberry Pi Documentation
4. Vijay Madisetti, "Internet of Things: A Hands-on Approach"
5. Raj Kamal, "Internet of Things: Architecture and Design"

# Activity-Based Learning

- - Design a simple Embedded System (e.g., Remote Control)
- - Microcontroller Experiments: LED, LCD, DAC Interfacing

# Course Outcomes & Bloom’s Taxonomy

C0-1. Explain Embedded System characteristics (L2)

CO-2. Understand circuit emulators & debugging (L2)

CO-3. Analyze hardware & software requirements (L3)

CO-4. Develop programming skills for Embedded Applications (L4)

CO-5. Design Embedded Systems for real-time applications (L4)

# Assessment Structure (CIE & SEE)

- - CIE: 50% (Assignments: 25 marks, Tests: 25 marks)
- - SEE: 50% (University Exam, 3-hour duration)
- - Minimum passing marks: 40% (40 out of 100)

# Module-1 Introduction

# Comparison Between Embedded Systems and General Purpose Computing Systems

# Comparison Table

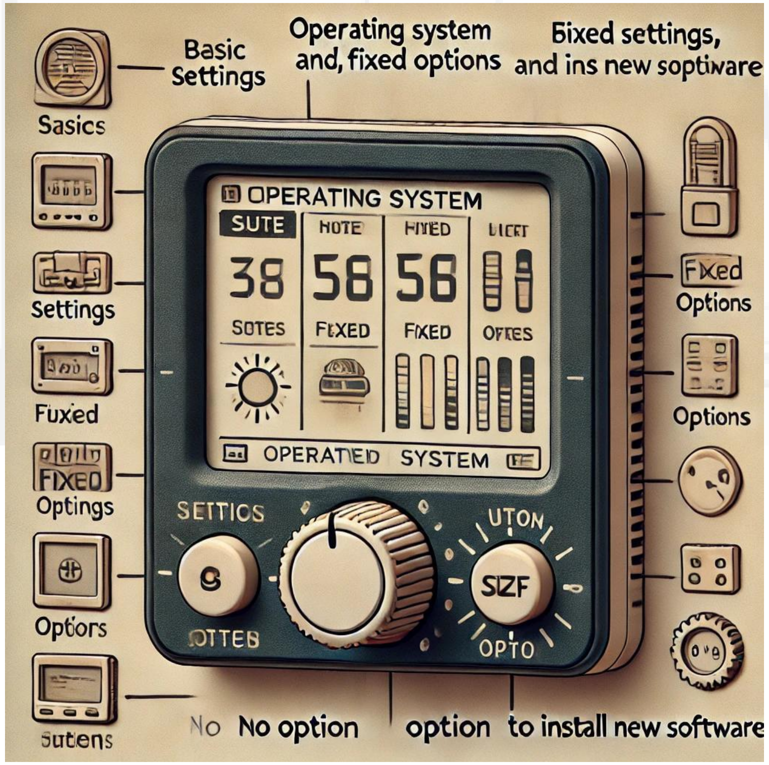
| Feature | Embedded System | General Purpose Computing System |
|---|---|---|
| Purpose | Designed for a specific function or task | Designed for multiple applications and tasks |
| Operating System | Fixed or custom OS, often non-modifiable | User can install or change OS (Windows, Linux, etc.) |
| Hardware Flexibility | Limited hardware customization | Highly flexible, supports multiple peripherals |
| User Applications | Cannot install third-party applications | Users can install and run multiple applications |
| Interfaces | Minimal, task-specific interfaces (e.g., IR remote) | Multiple interfaces (USB, Bluetooth, Wi-Fi, Ethernet, etc.) |

| Feature | Embedded System | General Purpose Computing System |
|---|---|---|
| Computational Requirements | Real-time processing with strict deadlines | General-purpose processing, not always real-time |
| Power Efficiency | Optimized for lower power consumption | Higher power consumption due to multitasking capability |
| Memory Availability | Limited and optimized for function | Large memory (RAM, HDD/SSD) for multiple applications |
| Software Modification | Not user-modifiable | Users can modify, update, and install new software |
| Functionality Expansion | Fixed function, cannot be easily repurposed | Can be repurposed for various tasks |
| Example Devices | DVD Player, Washing Machine, Car ECU | Desktop PC, Laptop, Palmtop, Workstation |

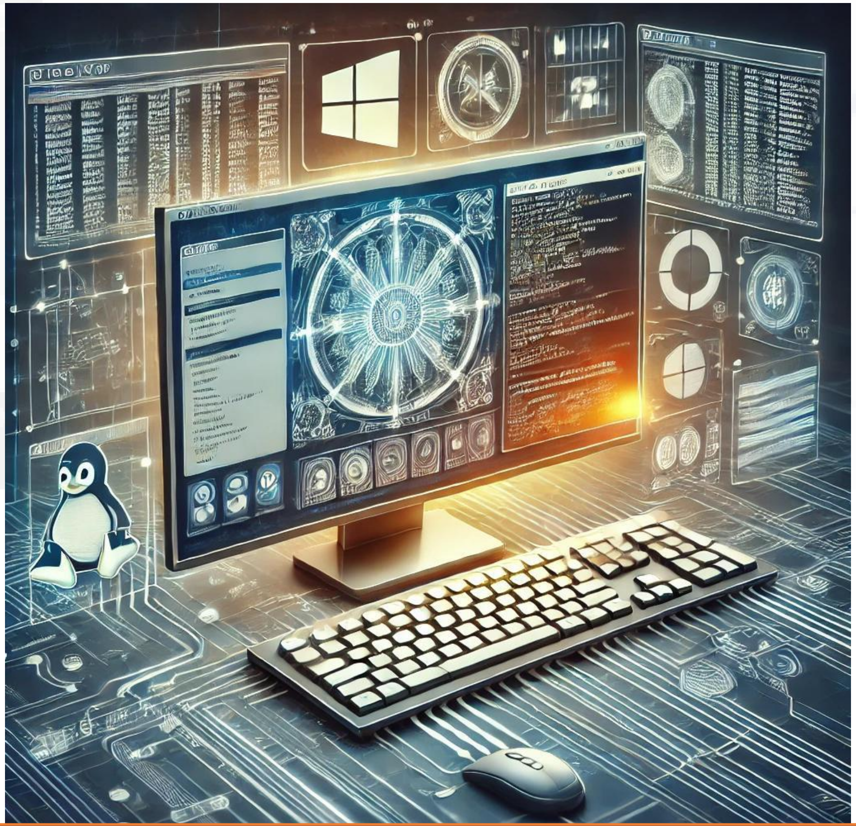
# Purpose of an Embedded System: A washing machine with predefined functions.

# Purpose of a General-Purpose Computing System: A laptop running multiple applications.

# Operating System in an Embedded System: A digital thermostat with a fixed interface.

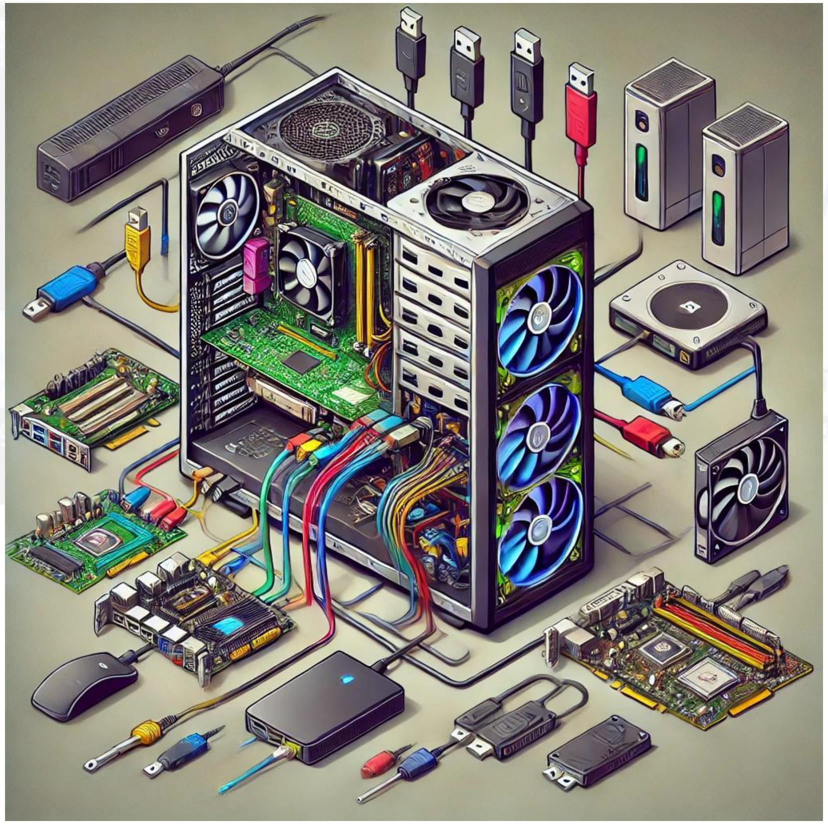


# Operating System in a General-Purpose Computing System: A desktop PC with a flexible OS.



# Hardware Flexibility in an Embedded System: A microwave oven with fixed buttons.

# Hardware Flexibility in a General-Purpose Computing System: A desktop PC with multiple peripherals.



# User Applications in an Embedded System: A smart thermostat with limited options.

# User Applications in a General-Purpose Computing System: A smartphone with multiple third-party apps.

# Classification of Embedded Systems

Based on Evolution and Complexity

# First Generation Embedded Systems

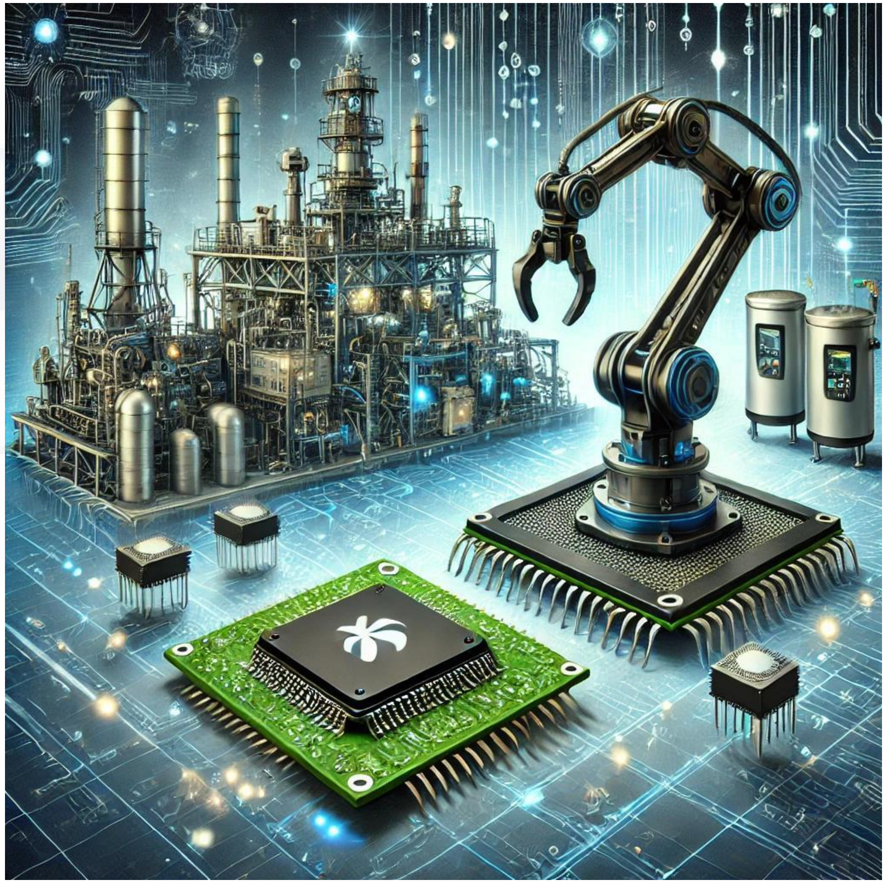
- Built around 8-bit microprocessors (e.g., 8085, Z80) and 4-bit microcontrollers. Simple circuits with Assembly language firmware.
- **Examples**: Digital telephone keypads, stepper motor control units.

# Second Generation Embedded Systems

- Uses 16-bit microprocessors and 8/16-bit microcontrollers. More complex instruction sets, some with embedded OS.
- **Examples**: Data Acquisition Systems, SCADA systems.

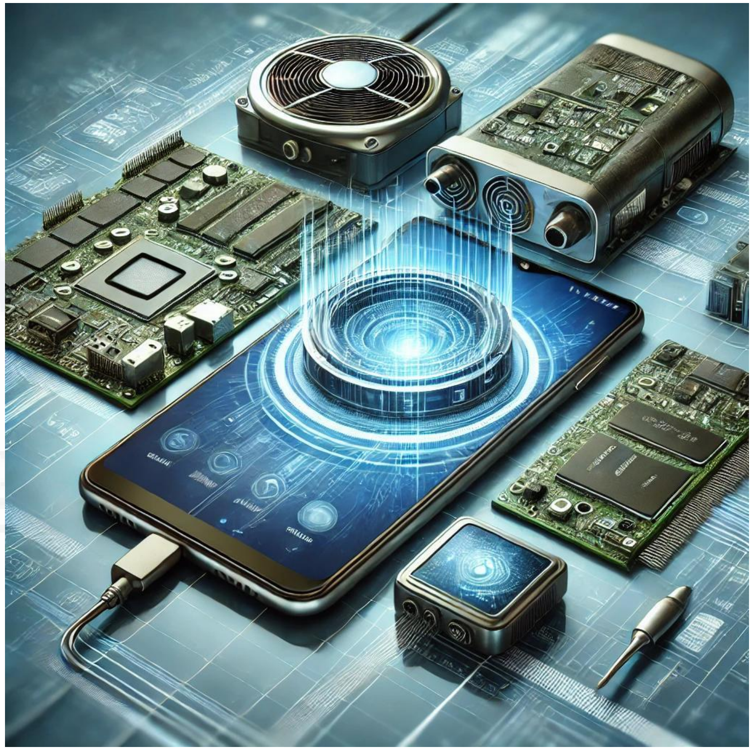
# Third Generation Embedded Systems

- Introduced 32-bit processors, 16-bit microcontrollers, and specialized chips like DSPs and ASICs. Instruction pipelining and real-time OS became common.
- Examples: Robotics, industrial process control, networking.



# Fourth Generation Embedded Systems

- Involves System-on-Chip (SoC), multicore processors, and high-performance embedded OS. Greater miniaturization and integration.
- **Examples:** Smartphones, Mobile Internet Devices (MIDs).



# Classification Based on Complexity and Performance

## 1. Small-Scale Embedded Systems

- Simple applications, not time-critical.
- Built on low-cost 8/16-bit microprocessors/microcontrollers.
- May or may not have an OS.
- Example: Electronic toys.

## 2. Medium-Scale Embedded Systems

- Moderately complex hardware and software.
- Uses 16/32-bit microprocessors/microcontrollers or DSPs.
- Usually includes an embedded OS.

## 3.Large-Scale/Complex Embedded Systems

- High-performance, mission-critical applications.
- Uses 32/64-bit RISC processors, RSoC, multi-core processors.
- Includes multiple processors, co-processors, and accelerators.
- Uses high-performance Real-Time Operating Systems (RTOS).
- Examples: Media encoding/decoding, cryptographic processing.

# Applications of embedded systems

1. **Consumer Electronics** – Camcorders, cameras.
2. **Household Appliances** – TVs, DVD players, washing machines, fridges, microwaves.
3. **Home Automation & Security** – Air conditioners, sprinklers, alarms, CCTV cameras.
4. **Automotive Industry** – ABS, engine control, ignition, automatic navigation.
5. **Telecom** – Cell phones, telephone switches, multimedia applications.
6. **Computer Peripherals** – Printers, scanners, fax machines.
7. **Networking** – Routers, switches, hubs, firewalls.
8. **Healthcare** – Scanners, EEG, ECG machines.
9. **Measurement & Instrumentation** – Digital multimeters, CROs, logic analyzers, PLC systems.
10. **Banking & Retail** – ATMs, currency counters, POS systems.
11. **Card Readers** – Barcode scanners, smart card readers, handheld devices.

**Purpose of Embedded Systems**

1. **Data Collection & Processing:** Converts, stores, or displays data (**e.g., ECG monitors, cameras**).
2. **Data Communication:** Enables wired/wireless data transfer (**e.g., routers, home automation**).
3. **Signal Processing:** Enhances audio/video signals (**e.g., digital hearing aids**).
4. **Monitoring:** Observes real-time data without control (**e.g., ECG machines, oscilloscopes**).
5. **Control Systems:** Regulates processes using sensors & actuators (**e.g., air conditioners**).

| Feature | Microprocessor | Microcontroller |
| --- | --- | --- |
| **Definition** | A silicon chip representing a CPU that performs arithmetic and logical operations. | A highly integrated chip with CPU, RAM, ROM, timers, and I/O ports for embedded systems. |
| **Dependency** | Dependent on external hardware (memory, timers, interrupt controllers, etc.). | Self-contained and does not require external hardware for functioning. |
| **Purpose** | General-purpose usage in industrial and high-performance computing. | Designed for specific tasks in embedded systems. |
| **I/O Ports** | Does not include built-in I/O ports; requires external components like programmable peripheral interface chips. | Includes multiple built-in I/O ports (8, 16, or 32-bit) for direct use. |
| **Applications** | Used in computers, servers, and high-performance devices. | Used in IoT, robotics, automotive systems, and consumer electronics. |
| **Architecture** | Based on Harvard or Von-Neumann architecture with RISC or CISC instruction sets. | Similar architectures, often domain-specific instruction sets (e.g., AVR for automotive). |
| **Power Efficiency** | Less efficient in terms of power consumption. | Optimized for power efficiency and compactness. |
| **Target Market** | High-end markets where performance is critical. | Embedded markets, where cost and size are important. |
| **Cost** | Expensive due to dependency on external hardware. | Cost-effective, with all required components integrated. |

| Feature | RISC (Reduced Instruction Set Computing) | CISC (Complex Instruction Set Computing) |
|---|---|---|
| **Instruction Set** | Lesser number of instructions. | Greater number of instructions. |
| **Instruction Pipelining** | Supports instruction pipelining for increased execution speed. | Generally, no instruction pipelining is present. |
| **Instruction Set Orthogonality** | Orthogonal instruction set (any instruction can operate on any register or use any addressing mode). | Non-orthogonal instruction set (instructions are instruction-specific regarding registers and addressing modes). |
| **Operations** | Operations are performed only on registers; memory operations are limited to load and store instructions. | Operations can be performed directly on registers or memory, depending on the instruction. |
| **Number of Registers** | Large number of general-purpose registers are available. | Limited number of general-purpose registers. |
| **Programming** | Requires more lines of code as the instructions are simpler and perform smaller operations. | Instructions are like macros in C language, allowing complex operations with a single instruction, reducing the code length. |
| **Instruction Length** | Single, fixed-length instructions. | Variable-length instructions. |
| **Hardware Complexity** | Uses less silicon and has a lower pin count due to simpler instruction decoding. | Requires more silicon and additional decoder logic to handle complex instruction decoding. |
| **System Architecture** | Uses Harvard architecture (separate buses for program and data memory). | Can use either Harvard or Von-Neumann architecture (shared bus for program and data memory). |

## Endianness in Memory Storage

Endianness defines how multi-byte data is stored in memory by a processor. It determines the order in which bytes are arranged in a system where the word size is greater than one byte.

### Types of Endianness:

1. **Little-Endian:**
   a. The **lower-order byte** is stored at the **lowest memory address**.
   b. The **higher-order byte** is stored at the **next higher memory address**.
   c. Example (for a 4-byte integer Byte3 Byte2 Byte1 Byte0):
      i. Memory Order: **Byte0** → **Byte1** → **Byte2** → **Byte3** (Lowest to Highest Address).
   d. Commonly used in **Intel x86** processors.

1. **Big-Endian:**

    a. The **higher-order byte** is stored at the **lowest memory address**.

    b. The **lower-order byte** is stored at the **next higher memory address**.

    c. Example (for a 4-byte integer Byte3 Byte2 Byte1 Byte0):

        i. Memory Order: **Byte3** → **Byte2** → **Byte1** → **Byte0** (Lowest to Highest Address).

    d. Used in **Motorola, PowerPC, and some network protocols**.

# ASIC – Key Points

1. **Definition:** Custom microchip for specific applications, replacing general-purpose logic.
2. **Advantages:** Reduces cost, compact, high-performance, and task-optimized.
3. **Types: Pre-fabricated** (specialized) & **Custom** (tailored for users).
4. **NRE Cost:** One-time investment, viable for **large-scale production**.
5. **ASSP:** Publicly available ASICs for **specific applications (e.g., ADE7760 Energy Meter**).
6. **Proprietary:** Confidential designs, limited public disclosure.

# Programmable Logic Devices (PLDs) – Key Points

1. **Definition & Types:** Perform system functions; categorized as **Fixed Logic Devices** (non-changeable) and **PLDs** (reconfigurable).
2. **Advantages:** Fast development, cost-effective, flexible inventory, **field reprogrammability**.
3. **Types:**
   - **FPGAs:** High density, high performance (**e.g., Xilinx Virtex**).
   - **CPLDs:** Lower density, predictable timing (**e.g., Xilinx CoolRunner**).
4. **PLDs vs Fixed Logic:** PLDs allow **modification & testing**, while fixed logic devices are **permanent**.
5. **FPGA Evolution:** From **40 MHz, costly** to **300 MHz, affordable, with integrated features**.

# COTS & Sensors/Actuators – Key Points

## Commercial Off-the-Shelf (COTS):

- **Definition:** Ready-made components for easy system integration.
- **Examples:** Remote circuits, ADCs, IR detectors, TCP/IP modules (**e.g., WIZnet**).
- **Advantages:** Fast development, cost-effective, pre-built firmware.
- **Disadvantages: Vendor lock-in, compatibility issues, risk of discontinuation**.

## Sensors & Actuators:

- **Sensor:** Converts energy for measurement/control (**e.g., Hall Effect Sensor in smart shoes**).
- **Actuator:** Converts signals into motion (**e.g., Micro Stepper Motor in smart shoes**).

**Communication Interfaces – Key Points**

**1.Types:**

- •**Onboard:** Internal connections (**I2C, SPI, UART, Parallel**).
- •**External:** Device communication (**USB, Ethernet, Wi-Fi, Bluetooth, RF, GPRS**).

**2.I2C Bus:**

- •**Synchronous, bi-directional, half-duplex (SCL & SDA)**.
- •**Master-Slave Model:** Master controls, slaves respond, supports multi-master.

**3.I2C Communication Sequence:**

- •**Start → Address → Read/Write Bit → Acknowledge → Data Transfer → Stop**.

**4.I2C Data Rates:**

- •**Standard:** 100 kbps, **Fast:** 400 kbps, **High-Speed:** 3.4 Mbps.

# I2C Communication Sequence

1. **Start Condition:** Master pulls **SDA LOW** while **SCL is HIGH**.
2. **Addressing:** Master sends **7-bit/10-bit slave address**.
3. **Read/Write Bit: 0 = Write**, **1 = Read**.
4. **ACK from Slave:** Slave pulls **SDA LOW** if address matches.
5. **Data Transfer:**
   - **Write:** Master sends **8-bit data**.
   - **Read:** Slave sends **8-bit data**.
6. **ACK After Each Byte:** Receiver sends acknowledgment.
7. **Stop Condition:** Master **releases SDA HIGH** while **SCL is HIGH**.

## Serial Communication Interfaces – Key Points

**SPI (Serial Peripheral Interface):**

•Synchronous, full-duplex, four-wire interface (Motorola).
•Master-Slave System: One master, multiple slaves (selected via SS).
•Signal Lines: MOSI, MISO, SCLK, SS.
•Fast, ideal for continuous data transfer but lacks acknowledgment mechanism.

**UART (Universal Asynchronous Receiver Transmitter):**

•Asynchronous serial communication, no clock signal needed.
•Uses Start/Stop bits, optional Parity for error detection.
•TX (Transmit) → RX (Receive) connection.
•Supports hardware handshaking for flow control.

**1-Wire Interface:**

•Asynchronous half-duplex, single-wire (DQ) protocol (Maxim).
•Master-Slave model, each device has a unique 64-bit ID.
•Sequence: Reset → Presence → Address → Function → Data Transfer.
•Bit Writing: 1 = Short LOW pulse, 0 = Long LOW pulse.
•Bit Reading: 1 = Released bus, 0 = LOW bus.

**Parallel Interface – Key Points**

- **Definition:** On-board interface for **memory-mapped peripheral communication** with the host processor.
- **Communication Control:** Managed by **Read/Write & Chip Select signals**.
- **Data Transfer: Host-initiated**, direction controlled by **RD/WR signals**.
- **Interrupt Handling:** Devices notify the processor via **interrupt lines**.
- **Bus Width:** Matches processor's **data bus** (4-bit, 8-bit, 16-bit, etc.).
- **Timing:** Follows **strict protocols** for reliable data transfer.

# External Communication Interfaces – Key Points

## RS-232:

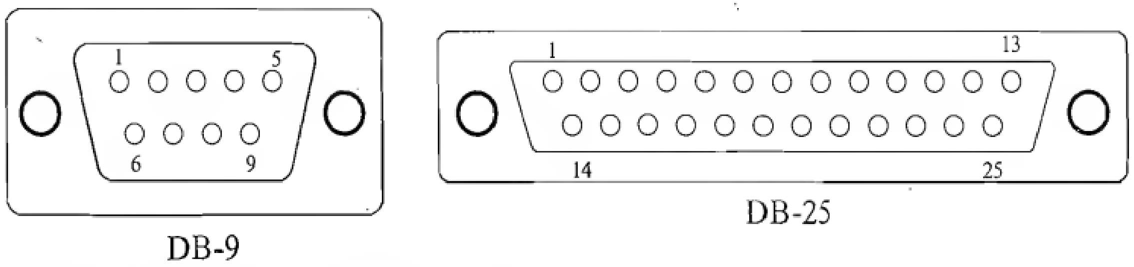
- Point-to-point serial communication (EIA standard).
- Baud Rate: Up to 19.2 Kbps, max 50 ft distance.
- Connectors: DB-9/DB-25, uses voltage signaling (+3V to +25V for 0, -3V to -25V for 1).
- Requires level converters (e.g., MAX232) for TTL compatibility.

## RS-422:

- Differential communication, 100 Kbps, max 400 ft distance.
- Supports multi-drop: 1 transmitter, 10 receivers.

## RS-485:

- Enhanced RS-422, multi-drop support for 32 transmitters & 32 receivers.
- Uses addressing for device identification.
- Ideal for long-distance industrial communication.




**RS-232 Connector Pin Details (DB-9 and DB-25)**

**Key Notes:**

- **DB-9 connectors** are more common in modern systems, while **DB-25 connectors** are mostly obsolete.
- For simple data transmission, only **TXD, RXD, and GND** are essential.
- Here are all the pins for **RS-232 connectors (DB-9 and DB-25)**:

**USB – Key Points**

- **High-speed wired serial bus** (1995, Intel & Microsoft).
- **Star topology**, supports **127 devices**.
- **Connectors: Type A (host), Type B (slave), Mini/Micro USB**.
- **Data Types: Control, Bulk, Isochronous, Interrupt**.
- **Data Rates: USB 1.0 (1.5–12 Mbps), USB 2.0 (480 Mbps), USB 3.0 (4.8 Gbps)**.
- **Power: 5V, 500mA**.

**USB Connector Pins:**

**Wireless Communication Interfaces – Key Points**

**Infrared (IrDA):**
- **Half-duplex, line-of-sight** wireless.
- **Range: 10 cm – 1 m**, **Data Rate: 9.6 kbps – 16 Mbps**.
- **Used in: TV remotes, file transfer, early mobile phones**.

**Bluetooth (BT):**
- **Short-range (2.4 GHz), low-power**.
- **Range: ~30 feet**, **Data Rate: Up to 1 Mbps**.
- **Used in: Phones, headsets, wearables, speakers**.

**Wi-Fi (IEEE 802.11):**
- **IP-based wireless networking**.
- **Range: 100-300 feet**, **Data Rate: 1 Mbps – 150 Mbps**.
- **Used in: Internet access, IoT, smart devices**.

**ZigBee (IEEE 802.15.4):**
- **Low-power, low-data WPAN**.
- **Range: Up to 100m**, **Data Rate: 20 – 250 Kbps**.
- **Used in: Home automation, smart meters, IoT**.

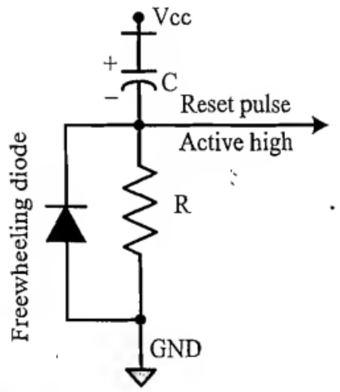
**GPRS (General Packet Radio Service):**
- **Packet-switched mobile data (GSM)**.
- **Max Data Rate: 171.2 kbps**.
- **Used in: Mobile internet, GPS trackers, M2M communication**.

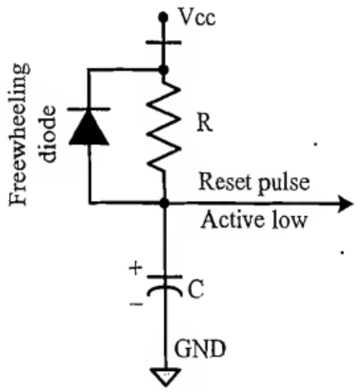
# Embedded Firmware – Key Points

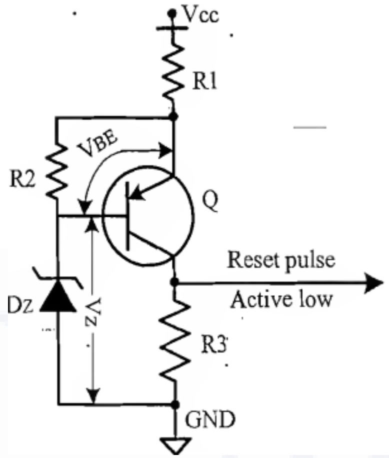
- **Definition:** Program instructions & settings stored in an embedded system.
- **Development:**
  - **High-Level (C/C++)** – Easier, portable, IDE-supported.
  - **Assembly** – Processor-specific, harder to debug.
- **HEX File Creation:** Converts code to machine-readable binary.
- **Control Algorithms:**
  - **Super Loop:** Continuous execution (**while(1){}**).
  - **Task Scheduler:** RTOS-based task management.

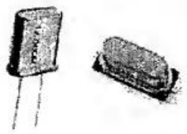
## Other System Components – Key Points

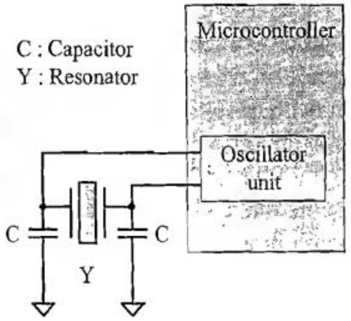
- **Definition:** Essential **circuits/ICs** required for proper **functioning** of an embedded system.
- **Examples:**
  - **Watchdog Timer** – Prevents system hang-ups.
  - **Reset IC/Circuit** – Ensures proper system startup.
  - **Brown-out Protection** – Prevents malfunction during power drops.
- **Integration:** Some **controllers/SoCs** have these built-in; others require **external components**.
- **Additional Components: Level translators, specific function ICs**, and interface circuits as needed.

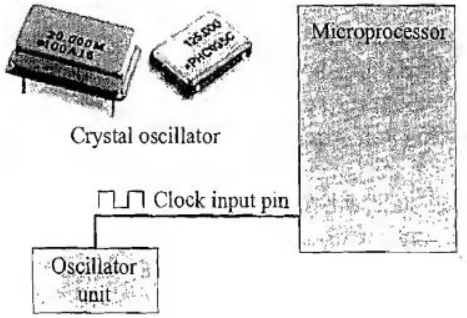


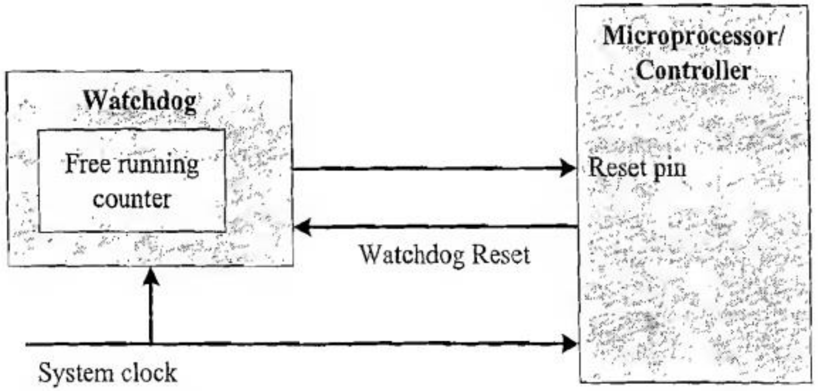





Quartz crystal resonator




Crystal oscillator



## PCB and Passive Components – Key Points

- **Printed Circuit Board (PCB):** The **foundation** of an embedded system, designed based on schematics and used for **component mounting and firmware testing**.
- **Passive Components:** Essential **supporting elements** like **resistors, capacitors, diodes**, etc., ensuring stable operation.
  - **Example:** Regulator IC with **filter capacitors** for a stable, ripple-free power supply.

# *Thank You*