Chapitre VI : Input/Output

V Input/Output

In an embedded system, input/output (I/O) interfaces are essential components that enable the microcontroller to interact with its environment. They provide the link between the physical world (sensors, actuators) and the digital world (processing, computation, control). Each I/O port transmits, converts, or controls information depending on the nature of the signal — analog, digital, serial, or parallel.

The proper functioning of an embedded system heavily depends on the quality, speed, and reliability of its I/O interfaces, especially in critical applications such as automotive, robotics, aerospace, and medical systems.

The figure below schematically illustrates the functional principle of input/output in a microcontroller-based system. Two types of ports can be distinguished:

- **Input ports**, connected to input peripherals (such as a push button or a sensor), allow the microcontroller to receive information from the external world.
- **Output ports**, connected to output peripherals (such as a display or a motor), allow the microcontroller to transmit control signals to the physical environment.

Thus, the microcontroller acts as a central decision unit that receives data through its inputs, processes it according to the program stored in memory, and then sends the corresponding results or control actions through its outputs. This architecture, typical of embedded systems, forms the basis of all human–machine or machine–machine interactions.

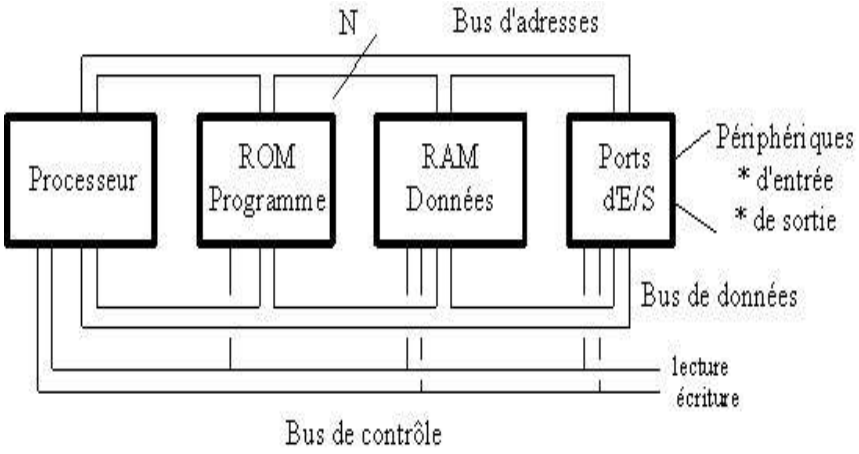


Figure VI. 1 - Functional diagram of input/output in a microcontroller-based system

V.1Serial and Parallel Transmission

In embedded electronics, data transmission between a microcontroller and its peripherals can be performed in two main ways: parallel or serial.

In parallel transmission, multiple bits of data (commonly 8, 16, or 32) are sent simultaneously, each on its own dedicated line. For example, transmitting one byte (8 bits) requires eight data lines, usually accompanied by several control lines for read/write, clock, or enable signals. This method offers a very high transfer rate, as several bits are transmitted per clock cycle. It is therefore preferred for fast, short-distance exchanges, such as those between the processor and internal memory within a microcontroller.

However, it also presents limitations: it requires a large number of physical connections, making wiring more complex and increasing the size of the printed circuit board (PCB). Moreover, maintaining synchronization between lines becomes challenging over longer distances due to electromagnetic noise. Parallel transmission is mainly found in internal buses of microcontrollers (e.g., AHB, APB, or AXI in ARM architectures), in processor-memory communication (RAM, ROM, EEPROM), or in high-speed interfaces for peripherals such as analog-to-digital converters (ADC) and LCD displays.

In contrast, serial transmission sends data bit by bit over one or two lines, significantly reducing the number of pins required and simplifying hardware design. Although generally slower than parallel transmission, it offers better noise immunity and allows reliable communication over longer distances. This approach is now widely used in modern embedded systems for communication between boards, sensors, or external modules.

Protocols such as SPI, I²C, UART, or CAN are based on this principle, each with specific characteristics depending on speed, distance, and device type.

Serial transmission can be:

- **Synchronous**, when a shared clock (as in SPI or I²C) provides synchronization; or
- **Asynchronous**, when it relies on start and stop bits in each frame (as in UART).

In practice, serial links are used to connect a microcontroller to digital sensors (temperature, pressure, luminosity), to communicate with external memories or displays, or to interface communication modules such as Wi-Fi, Bluetooth, or GPS. In industrial and automotive applications, the CAN bus is a typical example of a differential serial transmission, ensuring robustness and reliability in electrically noisy environments.

Thus, parallel transmission is preferred for internal high-speed communications requiring wide bandwidth, whereas serial transmission is ideal for external interconnections, where simplicity, robustness, and flexibility are more important than raw speed.

To better visualize their fundamental differences, the following table provides a comparative summary between serial and parallel transmission:

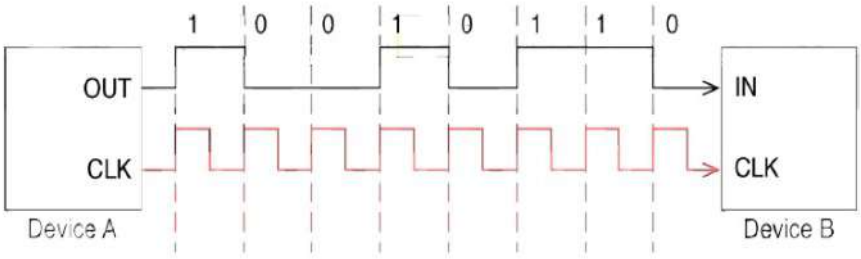
Criterion	Parallel Transmission	Serial Transmission
Principle	Simultaneous transmission of several bits over multiple lines	Sequential transmission of bits one by one on a single line
Number of lines	High (often 8 to 32 wires)	Low (1 to 2 wires)
Speed	Very fast over short distances	Variable, stable over long distances
Maximum distance	Limited (a few cm to dm)	Longer (up to several meters depending on protocol)
Wiring	Complex and costly	Simple and economical
Applications	Internal buses: CPU ↔ memory	External buses: sensors, modules, communication boards
Examples	AHB, AXI, PCI, parallel memory bus	SPI, I²C, UART, CAN, RS-485

V.2Synchronous and Asynchronous Interfaces

I/O interfaces establish a data exchange between two entities: a transmitter (typically the microcontroller) and a receiver (sensor, memory, communication module, etc.). The key difference between the two types of interfaces lies in the presence or absence of a shared clock signal for synchronization.

V.2.1 Synchronous Interfaces

A synchronous interface uses a common clock signal to coordinate data exchange between the transmitter and the receiver. The clock, usually generated by the microcontroller (master), defines the precise moment when each bit must be transmitted or read. As a result, both devices operate in perfect coordination, ensuring fast, stable, and reliable communication. This type of interface is widely used in embedded systems that require continuous and high-speed data exchange.



Main advantages include:

- Precise synchronization between communicating devices.
- High transfer speed, suitable for high-frequency sensors and fast memories.
- Reduced risk of synchronization errors.

However, the presence of an additional clock line and the need for a master device make implementation slightly more complex.

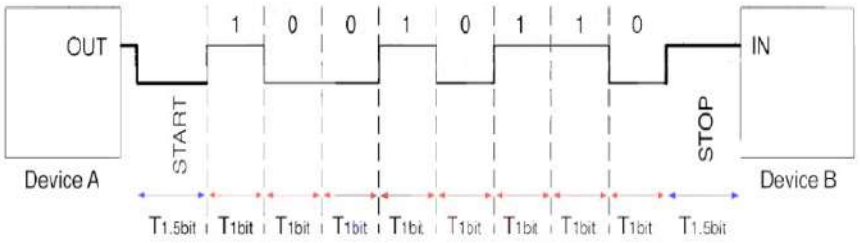
Common synchronous protocols in embedded systems include:

- SPI (Serial Peripheral Interface): a fast, full-duplex protocol commonly used for Flash memories and high-performance sensors.
- I²C (Inter-Integrated Circuit): a half-duplex, multipoint protocol widely used for communication among multiple peripherals on the same bus.

V.2.2 Asynchronous Interfaces

An asynchronous interface, on the other hand, does not rely on a shared clock signal. Each device manages its own transmission and reception timing, which simplifies wiring but requires software-based synchronization mechanisms. Data is transmitted in the form of frames, containing start and stop bits, which allow the receiver to detect the beginning and end of each binary word.

This communication mode is particularly suited to simple, low-cost, and medium-speed links, typically used for exchanges between a microcontroller and a computer, or between two independent modules.



Its advantages include:

- Simple wiring, since no clock line is required.
- Greater flexibility for devices operating at different speeds.
- Cost-effective implementation for applications where speed is not critical.

Its limitations include:

- Lower data rate compared to synchronous interfaces.
- Reduced tolerance to timing errors, since synchronization depends on software.

The UART (Universal Asynchronous Receiver-Transmitter) protocol is the most representative example of this communication type. It is widely used in RS-232, RS-485, or for communication between a microcontroller and Bluetooth, GPS, or Wi-Fi modules.

Characteristics	Synchronous Interfaces	Asynchronous Interfaces
Clock signal	Present and shared between devices	Absent; each device manages its own timing

Characteristics	Synchronous Interfaces	Asynchronous Interfaces
Synchronization	Hardware-based (via clock)	Software-based (via start/stop bits)
Transmission speed	High (up to several MHz)	Moderate (from a few kbps to Mbps)
Hardware complexity	Higher (clock line + master/slave)	Simpler (two wires TX/RX are sufficient)
Typical applications	High-speed sensors, memories, integrated circuits	Serial communication between modules or with a PC
Protocol examples	SPI, I²C, CAN (differential synchronous)	UART, RS-232, RS-485

V.3Communication Standards

In embedded electronic systems, communication standards define the rules that allow different components — microcontrollers, sensors, communication modules, and actuators — to exchange information in a reliable, fast, and standardized way. They ensure interoperability between devices, robustness of communication in constrained environments (temperature, vibration, interference), and optimized energy performance.

These standards are divided into:

- Serial communications (SPI, I²C, UART, CAN, USB), and
- Parallel communications (e.g., IEEE 1284).

The choice of standard depends on factors such as required data rate, distance, number of devices, and system constraints in the embedded architecture.

V.3.1 USB (Universal Serial Bus)

The USB (Universal Serial Bus) is today one of the most widely used communication interfaces in embedded electronic systems. It enables a microcontroller to communicate with a computer, external memory, sensor, or diagnostic module.

This synchronous serial bus is based on a master/slave architecture, where the host (typically a PC or a main microcontroller) initiates and controls all communications with the connected peripherals.

Data transfer occurs through differential signal lines (D+ and D–), which help reduce electromagnetic interference, complemented by a power line (VBUS, typically 5 V) and a ground line (GND).

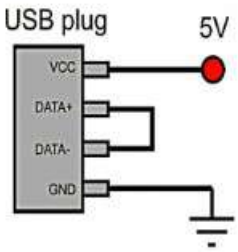


Figure VI. 12 - Pinout diagram of a USB connector

Recent versions (USB 3.x, USB-C) add extra differential pairs to increase throughput and support new features such as video transmission and fast charging.

Since its introduction in 1996, USB has evolved to meet the growing demands for speed and power in embedded systems. The first generations (USB 1.x) provided speeds up to 12 Mbit/s, sufficient for simple control interfaces such as keyboards or low-data-rate devices.

USB 2.0, still widely used today, raised this rate to 480 Mbit/s, enabling fast data transfers between a microcontroller and a host computer.

Subsequent generations — USB 3.x and USB4 — introduced full-duplex communication and speeds up to 40 Gbit/s, allowing integration into high-performance embedded systems such as industrial cameras, video processing systems, or robotic platforms.

The USB-C connector, now standard in modern architectures, is characterized by its reversibility and versatility: it can simultaneously carry data, power (up to 100 W via Power Delivery), and video signals. Its compact design makes it an ideal choice for portable devices and miniaturized embedded boards.

Version / Type	Maximum Data Rate	Transmission Mode	Available Power	Backward Compatibility	Typical Applications
USB 1.x	1.5–12 Mbit/s	Half duplex	5 V / 100	Yes	Keyboards, simple

Version / Type	Maximum Data Rate	Transmission Mode	Available Power	Backward Compatibility	Typical Applications
			mA		sensors
USB 2.0	480 Mbit/s	Synchronous	5 V / 500 mA	Yes	Programming, data transfer
USB 3.x	5–20 Gbit/s	Full duplex	5 V / 900 mA	Yes	Industrial cameras, high-speed acquisition
USB4	Up to 40 Gbit/s	Full duplex	5–20 V / 5 A	Yes	High-performance embedded systems
USB-C	Variable depending on standard	Multifunction (data, power, video)	Up to 100 W	Yes	IoT devices, compact embedded boards

The following figure provides an overview of the main generations of USB connectors and their corresponding maximum data transfer rates.

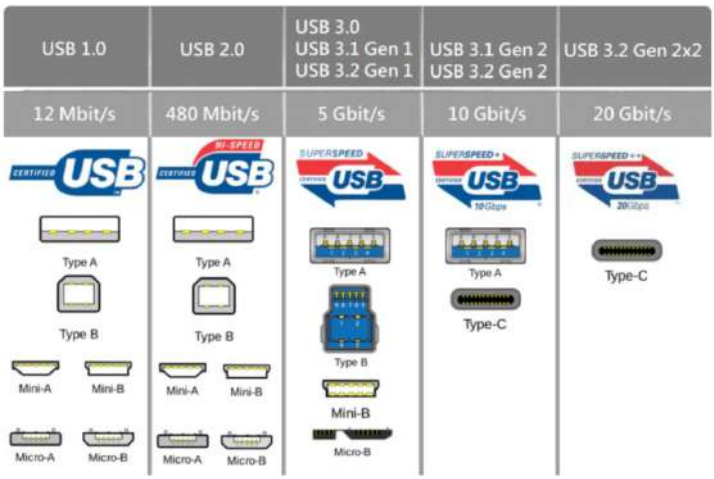


Figure – Evolution of USB connectors and associated data rates

This evolution reflects the USB standard's goal of unifying all types of wired communication — data transfer, power delivery, and video transmission — within a single interface.

V.3.2 Modern Connectors for Serial Communication (e.g., RJ45)

In embedded electronic systems, communication connectors play a crucial role: they provide the physical and electrical interface between modules while ensuring reliable data transmission, even under mechanical, thermal, or electromagnetic stress. Modern connectors are designed to meet the increasing demands for miniaturization, robustness, and versatility in today's embedded architectures.

Among the most widely used connectors, the RJ45 (Registered Jack 45) holds a key position due to its compatibility with the Ethernet standard (IEEE 802.3). It enables high-speed differential serial communication between embedded electronic devices.

This connector uses twisted-pair cabling, which significantly improves immunity to electromagnetic interference (EMI) and reduces signal loss over long distances. RJ45 connectors support several Ethernet standards, including:

- Fast Ethernet (100 Mbit/s),
- Gigabit Ethernet (1 Gbit/s), and
- 10 Gigabit Ethernet (10 Gbit/s), used in some industrial and automotive embedded systems.

In modern embedded applications, the RJ45 connector is integrated not only for network communication, but also for supervision, diagnostics, and firmware updates. For instance, a microcontroller or System-on-Chip (SoC) can be connected to an industrial server, programmable logic controller (PLC), or central control unit through an embedded Ethernet interface. This approach offers high transfer reliability, global standardization, and interoperability with most existing infrastructures.

In addition to the RJ45, several other modern serial connectors are widely used in embedded electronics:

- **USB-C**: A compact, reversible connector capable of carrying **data, power, and video** signals. It is increasingly adopted in portable devices, medical embedded systems, and development boards.
- Molex / JST / MicroFit: Miniature connectors commonly used in robots, drones, wireless sensors, and compact boards. They provide reliable connections in environments subject to vibration and temperature variation.
- Circular connectors (M12 or LEMO): Used in industrial and automotive applications, they offer excellent sealing and high mechanical strength.
- RJ11 and RJ12: Smaller than the RJ45, these connectors are sometimes used for RS-232/RS-485 serial interfaces or in low-speed embedded systems.

Connector	Type of Transmission	Typical Data Rate	Main Characteristics	Embedded Applications
RJ45 (Ethernet)	Differential serial	100 Mbit/s – 10 Gbit/s	Industrial standard, long range, high noise immunity	Industrial networks, IoT systems, supervision
USB-C	Multifunction serial	Up to 10 Gbit/s	Data, power, and video; reversible connector	IoT devices, embedded boards
JST / Molex / MicroFit	Simple or multi-line serial	< 100 Mbit/s	Compact, durable, low-cost	Drones, sensors, robotics
M12 / LEMO	Shielded serial	Up to 1 Gbit/s	Waterproof, robust, industrial-grade	Vehicles, automation, harsh environments
RJ11 / RJ12	Differential serial	< 10 Mbit/s	Compact, cost-effective	RS-232, RS-485, compact devices

V.4Classification of I/O Interfaces in Embedded Electronics

Input/Output (I/O) interfaces ensure communication between the microcontroller and its hardware environment. They enable the exchange of information, the measurement of physical signals, and the control of actuators. Depending on the nature of the exchanged signal and the communication function they perform, I/O interfaces in embedded systems can be classified into four main categories: digital, analog, industrial/network, and specialized interfaces.

V.4.1 Digital Interfaces

Digital interfaces transmit information in binary form (0 or 1). They are ubiquitous in embedded systems, particularly for communication between the microcontroller and logical peripherals. These interfaces are divided into three main subcategories:

- Simple interfaces: General Purpose Input/Output (GPIO) pins are individually configurable as inputs or outputs and are used to read or drive simple logic signals. PWM (Pulse Width Modulation) is a digital output with modulated pulse width, commonly used to control motor speed or light intensity.
- Serial communication interfaces: These transmit data bit by bit, reducing the number of required lines. The most common ones include SPI, I²C, UART, CAN, and 1-Wire.
- Parallel interfaces: Used to transfer multiple bits simultaneously, they offer very high data throughput at the cost of more complex wiring. Two types can be distinguished: Internal buses (e.g., AHB, AXI, APB) for communication between the processor, memory, and peripherals. External buses such as IEEE 1284, used for high-speed short-distance transfers.

V.4.2 Analog Interfaces

Analog interfaces enable the embedded system to interact with the real world, where physical quantities such as temperature, pressure, voltage, and current vary continuously. They perform conversion between analog and digital signals using dedicated circuits:

- ADC (Analog-to-Digital Converter): Converts an analog signal into a digital value, allowing the microcontroller to perform numerical processing.
- DAC (Digital-to-Analog Converter): Generates an analog signal from a digital value, typically used to drive proportional actuators or generate waveforms.

These interfaces are essential in measurement, control, and analog regulation systems, such as temperature sensors, microphones, pressure sensors, and amplifiers.

V.4.3 Industrial Communication Interfaces and Networks

These interfaces handle communication between multiple embedded systems or between an embedded system and industrial machinery. They ensure reliable transmission, often over long distances, thanks to standardized protocols. The main types include:

- RS-232 / RS-485: Asynchronous or differential serial interfaces, well suited to robust industrial links.
- Modbus / Profibus: Standardized industrial automation and supervision protocols.
- Ethernet / RJ45: High-speed network communication for connected systems (e.g., IoT, SCADA).
- USB (Universal Serial Bus): A universal interface for communication, diagnostics, and peripheral power supply.

These interfaces are commonly used in programmable logic controller (PLC) networks, inter-board communication, or supervisory and maintenance systems.

V.4.4 Specialized and Debugging Interfaces

These interfaces are designed for specific purposes, particularly for programming, testing, and diagnostics of embedded systems. They provide direct access to the microcontroller core for configuration or firmware updating. The most common include:

- JTAG (Joint Test Action Group): A standardized interface for testing, programming, and debugging integrated circuits.
- SWD (Serial Wire Debug): A simplified variant of JTAG, widely used in ARM microcontrollers.
- ISP (In-System Programming): Allows reprogramming of the microcontroller without removing it from the circuit board.

These interfaces are indispensable during development, maintenance, and production phases, ensuring precise diagnostics and easy system updates.

Summary Table – Classification of I/O Interfaces in Embedded Electronics

Category	Interfaces / Examples	Signal Type	Mode	Main Applications
Digital	GPIO, SPI, I²C, UART, CAN, 1-Wire	Binary	Serial / Parallel	Sensors, memories, displays

Category	Interfaces / Examples	Signal Type	Mode	Main Applications
Analog	ADC, DAC	Continuous	—	Physical sensors, analog control
Industrial and Network	RS-485, Modbus, Ethernet, USB	Differential digital	Serial	Automation, supervision, IoT
Specialized	JTAG, SWD, ISP	Digital	Serial	Debugging, programming, maintenance