EMBEDDED SYSTEMS
(CMPE 30274)
Module 2.2

Parallel Input and Output

Objectives

- Define parallel Input and Output
- Understand the role of parallel I/O.
- Discussed the different parallel I/O architectures in embedded systems
- Know the serial communication used in embedded system.
- Identify the parallel I/O Interfaces and Protocols.

Parallel Input and Output

refers to the method of exchanging data between the embedded system and external devices using a parallel interface. It involves transferring multiple bits of data simultaneously on separate lines or buses.

Parallel interfaces allow multiple bits of data to be transferred simultaneously using multiple data lines. This can provide high-speed data transfer, but it requires a larger number of wires and can be more complex to implement.

Example - Memory Bus:

The connection between the processor and main memory in a computer is a classic example of parallel I/O.

A data bus with multiple lines transfers data between the processor and memory at high speeds.

Role of Input-Output in embedded systems

1. Data Acquisition
2. Control
3. Communication

Role of Input-Output in embedded systems

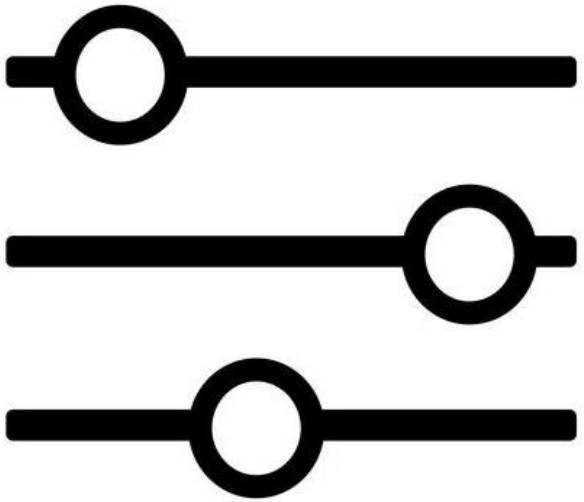
1. Data Acquisition:

- In embedded systems, data acquisition refers to the process of collecting data from various sources such as sensors, analog signals, or external devices.
- I/O interfaces enable the connection and communication between the embedded system and these data sources.
- For example, sensors used in temperature monitoring, motion detection, or environmental sensing provide input to the embedded system through I/O interfaces.
- The embedded system receives and processes this data, which can be used for decision making, monitoring, or control purposes.

Role of Input-Output in embedded systems

2. Control

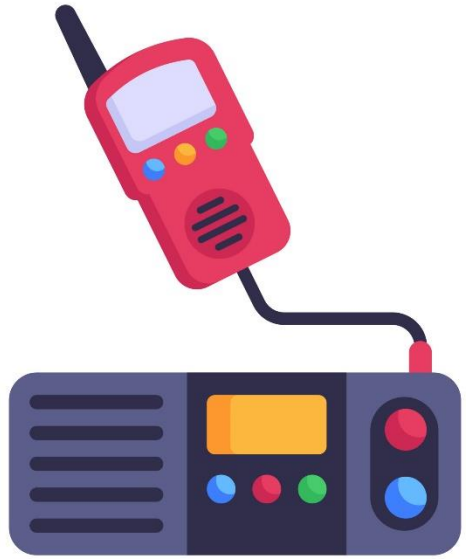
1. Embedded systems often perform control functions, such as regulating temperature, managing motors, or controlling industrial processes.
2. I/O interfaces are used to send control signals from the embedded system to external devices or actuators.
3. By generating appropriate control signals through I/O interfaces, the embedded system can manage and regulate the behavior of connected devices.
4. For example, in an automated home system, the embedded system can use I/O interfaces to control lighting, HVAC systems, or security devices.



Role of Input-Output in embedded systems

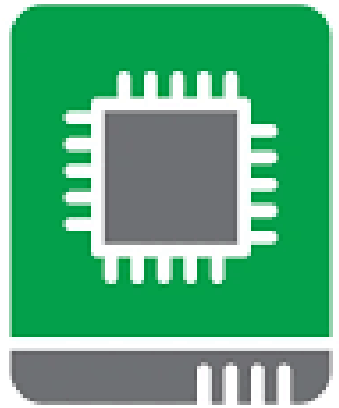
3. Communication:

1. Communication is a vital aspect of embedded systems, allowing them to interact with other systems, devices, or networks.
2. I/O interfaces enable data exchange and communication between the embedded system and external devices, other embedded systems, or larger networks.
3. Communication protocols, such as Ethernet, Wi-Fi, Bluetooth, or serial interfaces like I2C or SPI, are implemented using specific I/O interfaces.
4. These interfaces facilitate data transmission, enabling embedded systems to send or receive information, commands, or status updates.



Different parallel I/O architectures in embedded systems

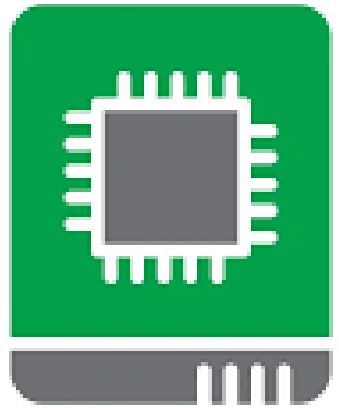
1. General-Purpose Input/Output (GPIO):
2. Parallel Peripheral Interfaces
3. Bus-Based Architectures
4. Memory-Mapped I/O
5. Parallel I/O Expanders:
6. Multichannel Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs)
7. Parallel Communication Interfaces
8. Parallel Display Interfaces



Different parallel I/O architectures in embedded systems

1.General Purposed Input Output:

GPIO pins are the most basic form of parallel I/O in embedded systems. They can be configured individually as either inputs or outputs, allowing the microcontroller or processor to communicate with external devices such as sensors, actuators, or other peripherals.

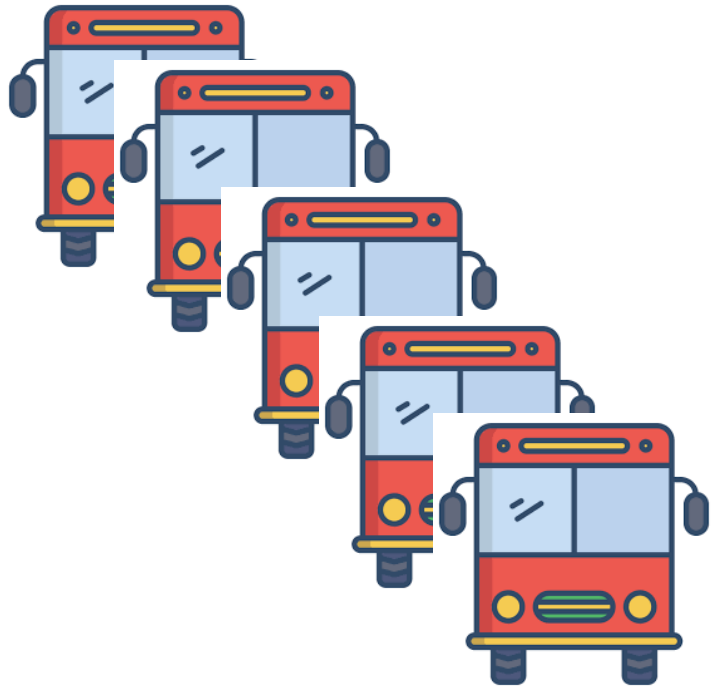


Different parallel I/O architectures in embedded systems

2. Parallel Peripheral Interfaces (e.g., Parallel Port):

These interfaces provide a parallel data transfer mechanism between the microcontroller or processor and external devices.

Examples include parallel ports, which were commonly used in older PCs for connecting printers and other peripherals.



Different parallel I/O architectures in embedded systems

3. Bus-Based Architectures (e.g., ISA, PCI):

In some embedded systems, especially those with higher performance requirements, bus-based architectures like ISA (Industry Standard Architecture) or PCI (Peripheral Component Interconnect) may be used.

These buses allow multiple devices to communicate with the processor in parallel, providing higher throughput compared to serial interfaces.

Different parallel I/O architectures in embedded systems

4. Memory-Mapped I/O:

In memory-mapped I/O, the I/O devices are accessed using the same address space as the system memory.

This allows the processor to read from and write to I/O devices using load and store instructions, making the I/O operations appear similar to memory operations.

Different parallel I/O architectures in embedded systems

5. Parallel I/O Expanders:

Parallel I/O expanders are integrated circuits that provide additional GPIO pins to the microcontroller or processor.

They typically communicate with the microcontroller or processor via a serial interface such as I2C or SPI, but they internally manage multiple parallel I/O pins.

Different parallel I/O architectures in embedded systems

6. Multichannel Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs):

Some embedded systems require parallel conversion of multiple analog signals to digital or digital signals to analog.

Multichannel ADCs and DACs provide parallel conversion of multiple channels simultaneously.

Different parallel I/O architectures in embedded systems

7. Parallel Communication Interfaces (e.g., Parallel ATA):

While not as common in modern embedded systems, interfaces like Parallel ATA (PATA) were used for connecting storage devices such as hard disk drives in older systems.

These interfaces transfer data in parallel across multiple data lines.

Different parallel I/O architectures in embedded systems

8. Parallel Display Interfaces (e.g., RGB):

In systems requiring high-speed display output, parallel interfaces such as RGB (Red, Green, Blue) are used to transfer pixel data from the processor or graphics controller to the display device in parallel.

GPIO (General-Purpose Input/Output)

It refers to a set of pins that can be individually configured as either input or output to communicate with external devices.

1. GPIO Pins:

- GPIO pins are typically found on microcontrollers and provide a means to interface with the external world. Each GPIO pin can be independently configured as an input or output.

2. Input Mode:

- a GPIO pin is used to read the state or value of an external device or sensor. It can detect signals such as button presses, sensor readings, or logic levels from other digital devices.

3. Output Mode:

- a GPIO pin can drive an external device or component. It can generate signals to control LEDs, motors, relays, or other devices by setting the pin to a high (logic level 1) or low (logic level 0) state.

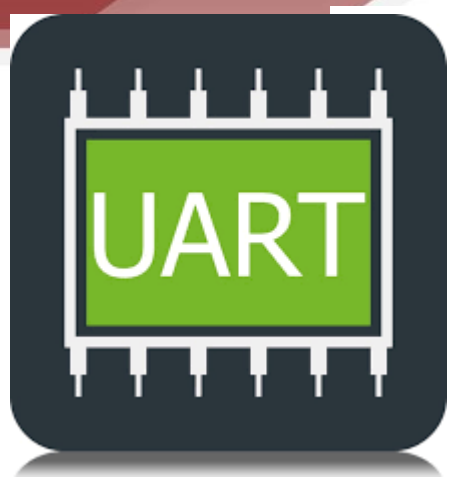
GPIO (General-Purpose Input/Output)

4. Register Access:

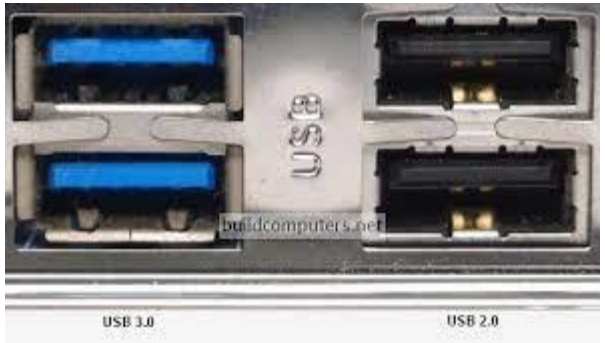
- To perform parallel I/O using GPIO, the microcontroller provides specific registers associated with each GPIO pin. These registers allow the programmer to read or write the state of the pin or configure its behavior as an input or output.

5. Control and Data Transfer:

- The microcontroller's software can manipulate the state of the GPIO pins by setting or clearing their corresponding registers. This allows control signals or data to be exchanged between the microcontroller and external devices in parallel.



Serial Communications

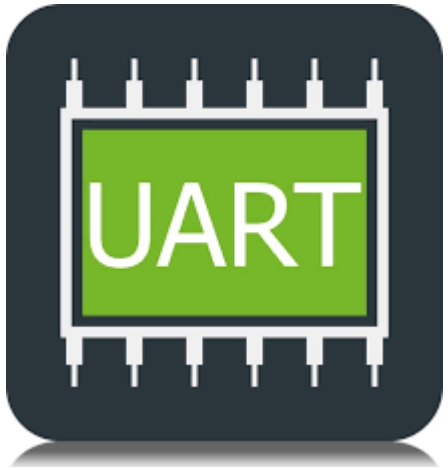


Serial Communications

1. UART (Universal Asynchronous Receiver-Transmitter)
2. SPI (Serial Peripheral Interface)
3. I2C (Inter-Integrated Circuit)
4. CAN (Controller Area Network)
5. RS-232 (Recommended Standard 232)
6. USB (Universal Serial Bus)

Serial Communications

Is widely used in embedded systems for various purposes, including data transfer, control, and configuration.



Commonly used serial communication protocols in embedded systems:

1. **UART (Universal Asynchronous Receiver-Transmitter):** UART is one of the simplest and most commonly used serial communication protocols.

It enables asynchronous serial communication between devices and is often employed for point-to-point communication between a microcontroller and peripheral devices, such as sensors, GPS modules, and Bluetooth modules.

2**. SPI (Serial Peripheral Interface):** SPI is a synchronous serial communication protocol commonly used for short-range communication between a microcontroller and peripheral devices.

It is often employed for high-speed data transfer to devices such as flash memory, display modules, ADCs, DACs, and wireless communication modules.

3. I2C (Inter-Integrated Circuit):

I2C is a multi-master, multi-slave serial communication protocol that facilitates communication between microcontrollers and peripheral devices.

It is widely used for connecting sensors, EEPROMs, real-time clocks, and other peripherals to a microcontroller. I2C allows multiple devices to share the same bus, reducing the number of required pins on the microcontroller.

4. CAN (Controller Area Network):

CAN is a robust serial communication protocol commonly used in automotive and industrial applications.

It enables reliable communication between multiple nodes in a network, making it suitable for applications that require high-speed data transfer and fault tolerance, such as in-vehicle communication, industrial automation, and robotics.

Serial Communications

5. **RS-232 (Recommended Standard 232):** RS-232 is a legacy serial communication standard widely used for long-distance communication between devices.

It uses voltage levels to represent data and provides a simple interface for connecting devices such as modems, terminals, and serial printers to microcontrollers.

6. **USB (Universal Serial Bus):** USB is a versatile serial communication protocol used for connecting a wide range of devices to a host system, including computers and embedded systems.

USB supports higher data rates and offers features such as plug-and-play functionality, power delivery, and device enumeration.

It is commonly used for connecting peripherals like keyboards, mice, storage devices, and communication modules to embedded systems.

Real-life examples

1. UART (Universal Asynchronous Receiver-Transmitter):

- GPS modules: UART is commonly used to communicate with GPS modules for receiving positioning data.

- Bluetooth modules: UART is used to establish a serial communication link between microcontrollers and Bluetooth modules for wireless data transmission.

- RFID readers: UART is often employed to interface with RFID readers for reading data from RFID tags.

2. SPI (Serial Peripheral Interface):

- Flash memory: SPI is frequently used to connect microcontrollers with SPI flash memory chips for storing program code or data.

- Display modules: SPI is utilized to communicate with display modules like LCDs or OLEDs for sending display data.

- Analog-to-Digital Converters (ADCs): SPI is commonly employed to interface microcontrollers with ADCs to convert analog signals into digital data.

Real-life examples

3. I2C (Inter-Integrated Circuit):

- EEPROM chips: I2C is extensively used to communicate with EEPROM chips for reading and writing data.

- Sensors: Many sensors, such as temperature sensors, humidity sensors, and accelerometers, use I2C for providing data to microcontrollers.

- Real-time clocks (RTCs): I2C is often used to interface with RTC modules for timekeeping and date retrieval.

4. CAN (Controller Area Network):

- Automotive systems: CAN is widely used in automotive applications for communication between electronic control units (ECUs) in vehicles, such as engine control units, transmission control units, and ABS control units.

- Industrial automation: CAN is employed in industrial control systems for connecting various devices, such as programmable logic controllers (PLCs), motor controllers, and sensors.

- Medical devices: CAN is utilized in medical devices for interconnecting different components, such as monitoring devices, infusion pumps, and diagnostic equipment.

Real-life examples

5. RS-232 (Recommended Standard 232):

- Legacy computer peripherals: RS-232 was commonly used to connect devices like modems, printers, and serial mice to computers in the past.

- Industrial equipment: RS-232 is still employed in various industrial applications for connecting equipment such as barcode scanners, data loggers, and PLCs.

- Serial communication with microcontrollers: RS-232 is often used to establish communication between microcontrollers and external devices, such as sensors or displays.

6. USB (Universal Serial Bus):

- Computer peripherals: USB is widely used for connecting devices like keyboards, mice, printers, scanners, and external storage devices to computers.

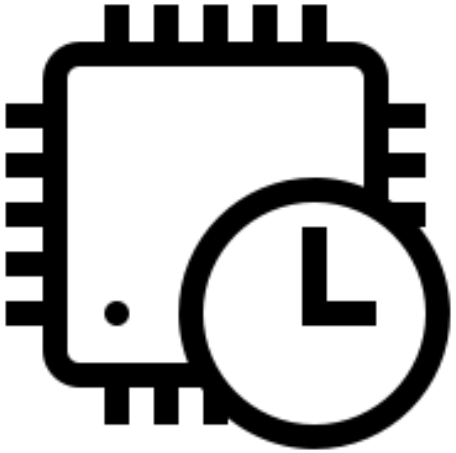
- Mobile devices: USB is used for charging and data transfer between smartphones, tablets, and other mobile devices and computers.

- Embedded systems: USB is employed in many embedded systems for connecting microcontrollers or single-board computers with external devices, such as sensors, displays, or wireless modules.

Real-time systems and parallel I/O

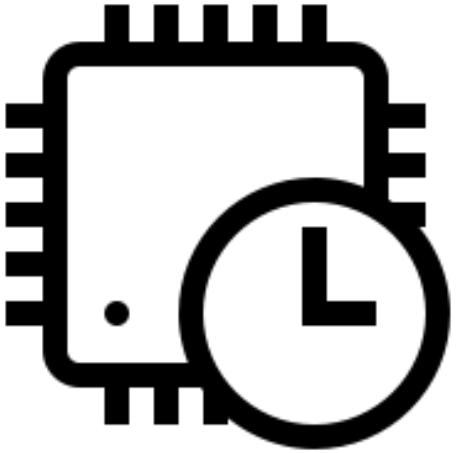
Real-time system is a computer system or software that is designed to respond to events and process data in a time-critical manner.

It is characterized by the need to meet strict timing constraints, where the correctness of the system depends not only on the logical accuracy of the computations but also on the timeliness of their execution.



In some **real-time systems**, parallel I/O can be employed to meet the timing requirements of data transfer.

By using parallel communication interfaces, such as parallel buses or dedicated parallel I/O pins, multiple bits or bytes of data can be transferred simultaneously, enabling faster data transfer rates.



Real-time systems are commonly found in various embedded applications, such as automotive systems, industrial automation, medical devices, avionics, and more.

Real life application

1. Implantable cardiac pacemakers are electronic devices that help regulate an individual's heart rhythm by delivering electrical pulses to the heart.
2. Aerospace and Avionics
3. Industrial Automation
4. Automotive Systems
5. Power Grid Control
6. Telecommunications
7. Medical Imaging
8. Robotics Systems

Next meeting (March 18, 2024)

Short Quiz on Module1 and 2

End of presentation