Peripheral Interfaces in Embedded Systems

Derricko Swink

Comparing Peripheral Interfaces: GPIO, UART, and I2C

Embedded systems rely on various interfaces to communicate with peripherals, each designed for specific types of data exchange and control. Here, we'll explore three interfaces: General-Purpose Input/Output (GPIO), Universal Asynchronous Receiver-Transmitter (UART), and Inter-Integrated Circuit (I2C) while outlining their differences and explaining why one might be used over another in embedded applications.

Differences Between GPIO, UART & I2C

1. **GPIO (General-Purpose Input/Output)**
    a. GPIO is a simple interface used to control digital signals, functioning either as input or output.
    b. Unlike communication protocols, GPIO does not transmit structured data but rather sends high (1) or low (0) signals to turn devices on or off.
    c. It is commonly used for toggling LEDs, reading button states, or controlling relays.
2. **UART (Universal Asynchronous Receiver-Transmitter)**
    a. UART is a serial communication protocol that enables direct device-to-device communication using two wires: TX (transmit) and RX (receive).
    b. It is asynchronous, meaning it does not require a clock signal, but both devices must agree on the baud rate for successful communication.

    c. It is widely used for debugging, communicating with sensors, or linking microcontrollers to other embedded systems.

3. **I2C (Inter-Integrated Circuit)**
    a. I2C is a synchronous serial communication protocol that allows multiple devices to communicate using only two wires: Serial Data Line (SDA) and Serial Clock Line (SCL).
    b. It supports multiple devices on a single bus, distinguishing them using unique addresses.
    c. It is often used for interfacing with sensors, EEPROMs, and display modules where multiple peripherals need to share the same communication lines.

Choosing One Interface Over Another

- **Use GPIO when** simple digital control is needed, such as turning an LED on/off or reading a button state. It is the easiest to implement but lacks the ability to send structured data.
- **Use UART when** direct two-way communication between two devices is necessary, such as sending text-based commands to a microcontroller or debugging via a serial console.
- **Use I2C when** multiple peripherals need to communicate over a shared bus with minimal wiring, such as in sensor networks or memory modules.

Conclusion

Each of these interfaces serves a specific role in embedded systems. GPIO is ideal for basic control tasks, UART is best for direct serial communication, and I2C is efficient for connecting multiple devices on the same bus. The choice of interface depends on the application requirements, data complexity, and the number of peripherals involved.

References

- Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann.
- Peatman, J. B. (1998). *Embedded Design with the PIC18F452 Microcontroller*. Pearson.
- Horowitz, P., & Hill, W. (2015). *The Art of Electronics*. Cambridge University Press.