IOT- Alternate Function

Prepared by Eng. Marwa Al-Tahaineh

Overview

- The core of many embedded systems is a microcontroller, which uses standardised communication protocols to interact with peripheral devices including sensors, displays, memory modules, and more.
- Even in complicated systems, these protocols provide reliable and efficient communication by controlling the data transmission between devices. SPI, I2C, and UART are three popular protocols.

GPIO Alternate Functions

- Many computer microcontrollers particularly popular ones like the Raspberry Pi, allow you to customise GPIO pins for numerous functions.
- This indicates that a single pin can be used as a generic digital input/output or for alternate role (for example, as part of a UART, SPI, or I2C interface).
- This adaptability enables practical hardware design and efficient use of limited physical pins.

On the Raspberry Pi:

- UART uses GPIO14 and GPIO15.
- SPI uses GPIO10, GPIO9, GPIO11, and other pins to choose slaves.
- I2C uses GPIO2 and GPIO3.

Universal Asynchronous Receiver-Transmitter(UART)

Asynchronous serial communication involves exchanging data between devices without a shared clock signal. Rather, it uses precise timing and control bits to assure data integrity.

Data Framing :

Each data packet is framed with certain bits to define its boundaries by utilizing Start and Stop Bits:

- Start bit:- When no data is transmitted, the communication channel stays idle and in a high state. As the announcement of the start of a data packet, the transmitting device pulls the line from high (1) to low(0). This transition directs the receiving device to prepare for incoming data.

- Data Bits:- After the start bit, the data itself is transferred, which can range from 5 to 9 bits based on the communication protocol.

UART(Cont.)

- Parity Bit(optional): A parity bit may be provided to detect errors and maintain data integrity during transmission.

- Stop Bit(s): One or more stop bits are communicated by returning the line to high (1). These bits imply the end of the data packet. On the other hand, a buffer can be used before the next start bit, allowing the receiver to analyze the data that comes in.

Important notes:

- Since there is no shared clock, the transmitter and receiver must operate at a predetermined data transmission rate, known as the baud rate. This agreement guarantees that the receiver will sample the incoming data at the appropriate intervals to maintain synchronization during the communication process.

UART(Cont.)

- The baud rate determines the speed of sending and receiving data. Typical baud rates are 1200, 9600, 19200, and 38400 bits per second. For appropriate data interpretation, the devices need to be configured to use the same baud rate. A mismatch might cause data corruption or communication failure.

- UART's application: Commonly used for serial communication between computers and peripherals, debugging, and applications demanding simplicity and reliability across long distances.

- UART enables direct, asynchronous serial communication between the Raspberry Pi and external devices. The Raspberry Pi can use TX (transmit) and RX (receive) lines for sending and receiving data respectively, making it ideal for interacting with serial peripherals, debugging, and transferring information between two microcontrollers.

Serial Peripheral Interface (SPI)

- A synchronous serial communication protocol is broadly utilized for short-distance communication, especially in embedded systems. It enables fast data transfer between a master device and one or more slave devices.
- SPI is a synchronous protocol that synchronizes data transmission using a clock signal produced by the master device.
- This protocol includes:
  - The Master Device begins and manages all communication on the SPI bus, including generating the clock signal (SCK) and controlling data flow.
  - Slave Devices: Follow the master's commands. Multiple slaves can be attached to a single master, each of which has its own Slave Select (SS) line to be selected using it.

SPI(Cont.)

- Besides, SPI uses four primary lines:
  - Serial Clock (SCK): A clock signal produced by the master to synchronize data delivery.
  - Master Out Slave In (MOSI): A line that transmits data from the master to the slave.
  - Master In Slave Out (MISO): The line used to send data from the slave back to the master.
  - The Slave Select (SS) line allows the master to pick and enable a particular slave device.

SPI(Cont.)

Data Transmission Steps:

- Clock Generation: The master creates the SCK signal to synchronize data flow.

- Slave Selection: The master lowers the SS line to activate the appropriate slave device (low voltage level (0V)).

- Data Exchange: Data is transmitted and received bit by bit (each clock pulse):
  - The master sends a bit to the MOSI line, which the slave reads.
  - The slave transmits a bit via a MISO line, as the master reads.
  - Full-duplex communication enables continuous data transmission in both directions.

SPI(Cont.)

Important notes:

- SPI’s application :- Ideal for high-speed communication with devices such as SD cards and sensors that need fast data rates, also it's perfect for connecting multiple devices in a chain topology.

- In SPI communication, the Raspberry Pi usually behaves as the master device, which means initiating communication and controlling the clock signal (SCLK). Rarely, the Raspberry Pi can be set up as an SPI slave, allowing another master device to govern communication.

Inter-Integrated Circuit(I2C)

- The Inter-Integrated Circuit (I2C) protocol is a synchronous, multi-master, multi-slave serial communication protocol that provides efficient data exchange between several devices with only two wires.
- Only two wires are needed for the synchronous I2C protocol:
  - Signal Serial Clock Line (SCL)conveys the clock signal generated by the master to synchronize data transmission
  - Serial Data Line (SDA): for transferring data between devices.
- Each device on the I²C bus has a unique address (either 7-bit or 10-bit):
  - The most dominant type of addressing is 7-bit, which provides up to 128 distinct addresses (0 to 127).
  - Since some addresses are reserved for particular purposes, the number of addresses open to general devices is limited.
  - 10-Bit Addressing: Developed to increase the address space to cover additional devices since the number of available addresses in the 7-bit addressing scheme is insufficient.

I2C(Cont.)

Data Transmission Steps:

- The master begins communication and manages the clock (SCL).
- The master pulls SDA low and keeps SCL high to inform all devices that a communication process has started (Start Condition).
- The master sends the target slave's address paired with a read/write bit to indicate the desired operation (Address Frame).
- The addressed slave pulls SDA low, indicating that it is ready to communicate (Acknowledgement (ACK)).
- Data is sent in 8-bit bytes, each followed by an acknowledgment bit. The read/write bit delivered in the address frame determines the direction of data flow(Data Transfer).
- After SCL becomes high, the master sets SDA to high, announcing the end of the communication sequence (Stop Condition).

I2C(Cont.)

Important notes:

- It's important to note that I²C allows several masters on the same bus. If two masters begin communication at the same time, arbitration decides which master takes control based on priority while preventing data corruption.
- I²C's application :- is ideal for connecting several low-speed peripherals like EEPROMs, and sensors, keeping wiring demands to a minimum.
- The Raspberry Pi serves as the master in I²C communication, where it has the privilege of handling the clock (SCL) and data (SDA) lines to interact with many slave devices. This configuration is popular because it is simple and expandable.
- However, the Raspberry Pi can also function as a peripheral by responding to requests from an I²C master device, but this is not commonly done.

Overview of GPIO Modes

Mode Type	Purpose	Example Pins
INPUT	Reading button/sensor signals	GPIO4, GPIO17
OUTPUT	Controlling LEDs/motors	GPIO18, GPIO27
PWM	Adjusting brightness/speed	GPIO12, GPIO13
I2C	Communication with sensors	GPIO2 (SDA), GPIO3 (SCL)
SPI	High-speed data transfer	GPIO10 (MOSI), GPIO9 (MISO)
UART	Serial communication	GPIO14 (TX), GPIO15 (RX)
PCM (I2S)	Digital audio transmission	GPIO18, GPIO19, GPIO20, GPIO21

Any Questions???

Important note: The content in this presentation is sourced from trusted sites, books, and published videos/slides.