24EC504 – Unit II – Part 3: Communication Protocols, Displays, Low Power and Debugging

1. UART, SPI and I²C Communication Protocols

These are the three classic serial protocols that connect a microcontroller to sensors, displays, memory chips and other peripherals.

1.1 Key Concepts

- **Synchronous vs asynchronous:** synchronous protocols (I²C, SPI) share a clock line; asynchronous (UART) has no shared clock, so both devices must be set to the same speed (baud rate).
- **Half-duplex vs full-duplex:** half-duplex allows both directions but not at the same time; full-duplex allows simultaneous two-way traffic.

1.2 UART (Universal Asynchronous Receiver/Transmitter)

Type: asynchronous, full-duplex. Logic high = 1, low = 0. With no shared clock, both ends must use exactly the same speed (for example 9600 or 115200 baud). TX of one device connects to RX of the other (lines crossed).

Data frame:

1. **Idle:** line held high.
2. **Start bit (1 bit):** breaks the idle state to signal incoming data.
3. **Data bits (5–9):** sent LSB first; the payload.
4. **Optional parity bit (1):** error check; odd, even, mark (1) or none.
5. **Stop bit(s) (1–2):** return the line to idle high and end the frame.

Error	Cause
Framing error	Receiver expects a stop bit but the line is not high; the frame is broken
Parity error	Parity bit does not match the odd/even count of the data bits; corruption
Overrun error	Receiver buffer is full because software did not process the previous character; data lost
Break condition	Not strictly an error: input stays low longer than one character time; sometimes used to pull a reset pin

1.3 I²C (Inter-Integrated Circuit)

Type: synchronous, half-duplex, multi-master/multi-slave. Speeds: standard 100 kbps, fast 400 kbps, high-speed up to 3.4 Mbps. Up to 127 devices on a 7-bit address bus. Needs mandatory 4.7 kΩ pull-up resistors (open-drain). Two wires: SDA (data) and SCL (clock).

Anatomy of a transmission:

1. **Start condition:** SDA pulled low while SCL is high.
2. **Address:** master broadcasts the 7/10-bit target address.
3. **R/W bit:** tells the addressed device whether the master wants to read or write.
4. **ACK/NACK:** the device confirms it responded.
5. **Data payload:** 8 bits, followed by another ACK/NACK.
6. **Stop condition:** SDA goes high while SCL is high, releasing the bus.

Edge cases:

- **Multi-master arbitration:** wired-AND logic on SDA means any node pulling low makes the bus read low. The master whose data goes high while the bus reads low loses arbitration and backs off.
- **Clock stretching:** a slow slave (for example doing a slow ADC reading) holds SCL low; the master waits until it releases the line.
- **Deadlock:** the master holds SCL high waiting for SDA to release, while a stuck slave holds SDA low waiting for an ACK pulse. Fix: the master forces 9 dummy clocks to push the slave through its cycle, or issues a hardware reset.

1.4 SPI (Serial Peripheral Interface)

Type: synchronous, full-duplex, high speed (MHz range), no start/stop bits (continuous streaming), wire-hungry. Ideal for SD cards, displays and real-time sensors.

- **MOSI** (Master Out, Slave In): forward data line.
- **MISO** (Master In, Slave Out): return data line.
- **SCLK:** clock generated by the master.
- **SS/CS:** active-low slave select.

Topologies:

- **Independent slaves:** each slave has its own select line; every new slave costs one more master GPIO pin.
- **Daisy chain:** data flows through the slaves in sequence; to reach the last slave the master streams through the whole chain.

Four SPI modes (master and slave must agree on clock polarity CPOL and phase CPHA):

Mode	CPOL	CPHA	Behaviour
0	0 (idle low)	0 (sample first edge)	Captured on rising edge, output on falling edge

Mode	CPOL	CPHA	Behaviour
1	0 (idle low)	1 (sample second edge)	Captured on falling edge, output on rising edge
2	1 (idle high)	0	Captured on falling edge, output on rising edge
3	1 (idle high)	1	Captured on rising edge, output on falling edge

1.5 Protocol Comparison and Selection

Protocol	Complexity	Speed	Wires	Duplex	Topology limit
UART	Simple	Slowest	2 (TX, RX), no clock	Full	1-to-1, point-to-point
I²C	Moderate (software complex)	Fast (100 k–3.4 Mbps)	2 (SDA, SCL)	Half	Multi-master, up to 127 slaves
SPI	High (wiring heavy at scale)	Fastest (8 Mbit/s+)	4 (MOSI, MISO, SCLK, SS)	Full	1 master, multiple slaves

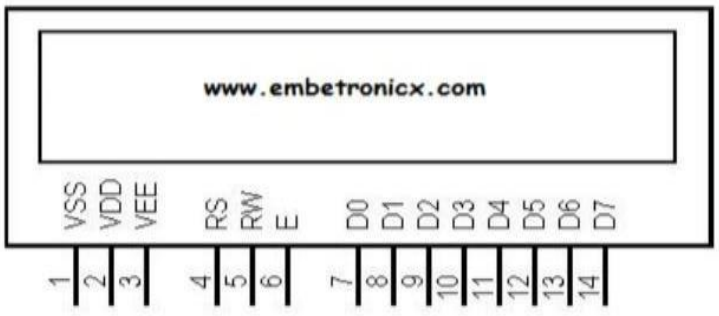
Selection guide:

- Highest continuous data rate (SD cards, displays): **SPI**.
- Many sensors on one board with only two pins spare: **I²C**.
- Simple link between two devices, or longer distance, no need for speed: **UART**.

2. LCD, OLED and Touch Display Interfacing

2.1 LCD Display

LCDs (used in computers, digital watches, DVD and CD players) replaced bulky, power-hungry CRTs. A **16×2 character LCD** shows 16 characters on each of 2 lines, each character in a 5×7 pixel matrix. It has two registers: the **Command register** (instructions such as initialize, clear screen, set cursor) and the **Data register** (ASCII value of the character to show).



Pin	Name	Function
1	Ground	0 V
2	Vcc	5 V (4.7–5.3 V)
3	VEE	Contrast, through a variable resistor
4	RS	Command register when low; data register when high
5	R/W	Low to write, high to read
6	EN	Data latched on a high-to-low pulse
7	DB0–DB7	8-bit data pins
8	LED+	Backlight supply (5 V)
9	LED−	Backlight ground

- A **4-bit** data bus needs 7 lines in total (3 control + 4 data); an **8-bit** bus needs 11 (3 control + 8 data). The control lines are EN, RS and RW.
- **Send data:** R/W = 0 (write), RS = 1 (data register), then a high-to-low pulse on EN. Strings go one character at a time, since the data lines are 8 bits wide.
- **Send command:** R/W = 0, RS = 0 (command register), then a high-to-low pulse on EN. The LCD needs a set of standard initialization commands before use.

2.2 OLED Display

A 128×64 OLED is compact, high-contrast, self-illuminating and graphical. Modules are driven by the **SSD1306** IC, which has its own controller and supports SPI and I²C (modules may support only SPI, only I²C or both). A common module is the 4-pin I²C-only version.

Pin	Function
VCC	Power, 3.3 V or 5 V
GND	Ground
SCL	I²C clock
SDA	I²C data

2.3 Touch Screen Interfacing

A touch screen detects the presence and location of a touch in the display area, an alternative to mouse and keyboard for navigating a GUI. Components: a **touch sensor** (structure depends on type), a **controller** (reports touch location to the device software) and **software drivers** (usually written in C).

Type	Working principle	Advantages	Disadvantages
Resistive	Two conductive layers with spacers; pressure makes contact and changes resistance	Low cost, durable, works with finger, stylus or gloves	Lower clarity, single-touch only, needs pressure
Capacitive	Transparent conductive layer (ITO or copper) senses electrostatic field changes; surface capacitive or projected capacitive (P-CAP, multi-touch)	High clarity, fast, multi-touch gestures	Needs conductive objects, more expensive
Infrared (IR)	IR LEDs and sensors form a beam grid; a touch interrupts the beams	Excellent display quality, no pressure needed, any object	Sensitive to dust and sunlight, higher cost
Surface acoustic wave (SAW)	Ultrasonic waves cross the surface; a touch absorbs part of the wave	High clarity, accurate, durable glass	Affected by dust, water and contaminants; costlier than resistive

LCD vs OLED vs Touch Display

Feature	LCD	OLED	Touch display
Light source	LED backlight	Self-emissive pixels	Depends on the panel beneath
Power	Low to moderate	Very low for dark content	Slightly higher (touch controller)
Contrast	Moderate	Excellent (true black)	Depends on technology
Viewing angle	Moderate	Very wide	Depends on panel
Response time	Moderate	Very fast	Fast
Input	Display only	Display only	Display + user input
Common interfaces	Parallel, I²C backpack, SPI	I²C or SPI	I²C (touch) + SPI/RGB/MIPI (display)
Typical use	Calculators, industrial panels, meters	Smartwatches, medical devices, phones	Smartphones, tablets, HMIs, kiosks, medical equipment

3. Low Power Design Techniques

3.1 Definition and Need

Low power design is a set of techniques that reduce the dynamic and static power of an IC. Power depends on activity, frequency, transition time, capacitive load, voltage, leakage current and peak current; higher voltage raises the power of every component. For battery-powered and handheld devices, battery life and OFF-to-ON transition time matter most. For plugged-in products, lower power cuts cooling needs and electricity cost; in server farms even a small per-chip saving adds up.

3.2 Techniques

Technique	What it does	Requirements / notes
Clock gating	Turns off the clock to inactive parts, reducing switching activity and dynamic power; done at logic synthesis	Latch-based gating preferred over AND/NOR gating
Multi-voltage design	High-performance blocks at higher voltage, others at lower voltage; cuts static and dynamic power	Separate voltage islands and level shifter (LS) cells at domain crossings
Power gating	Cuts power completely to inactive blocks with power switches, removing static and dynamic power	Isolation gates, defined power states, Power Management Unit (PMU)
Retention with power gating	Saves flip-flop state before power-off and restores it on power-up, avoiding full re-initialization	Retention flops; SAVE/RESTORE signals in the PMU

Advanced methods: combinations of the above, well biasing, zero-pin retention flops, special low-power library cells, Dynamic Voltage and Frequency Scaling (DVFS), Adaptive Voltage and Frequency Scaling (AVFS) and custom design.

3.3 Methodology

1. **Static power verification and exploration:** check RTL, UPF and SDC inputs (lint, CDC, UPF checks) and estimate power early.
2. **Dynamic power verification and analysis:** verify the PMU sequence (shutdown, isolation clamping, save, restore, power-up) and analyze toggle activity.
3. **Software-driven power analysis:** emulation captures peak-power windows over a wider dataset.
4. **Power implementation:** RTL power estimation, logic synthesis, DFT insertion, and physical implementation (floorplan, placement, clock-tree synthesis, routing).
5. **Signoff:** re-check UPF consistency across Netlist, UPF′, PGNetlist and UPF″, plus equivalence checking, static timing and detailed power analysis.

3.4 Challenges

- Very tight voltage-drop margins at ultra-low voltage make IR-drop signoff essential.
- Low supply voltage makes timing sensitive to dynamic voltage drop, hard to cover fully.
- Multi-voltage regions need careful handling of signal transitions; UPF is the standard for this.
- Power gating causes hard electrical and logical transitions when a block turns back on.
- Realistic activity vectors are hard to create; functional verification vectors mislead, so hand-written, vectorless or emulator-based vectors work better.

3.5 Tools

Category	Tools / purpose
Static verification and exploration	Lint and CDC tools, UPF checker, RTL power estimator
Dynamic verification and analysis	Power-aware simulator and debug tool, RTL-to-gate-level waveform translation
Software-driven power analysis	Emulation for cycle-based power data
Power implementation	RTL power estimation, logic synthesis, DFT insertion, physical implementation
Signoff	UPF checking, power-aware equivalence checking, static timing and power analysis

4. Embedded Debugging and Testing

Embedded systems range from IoT devices to industrial machinery, so debugging and testing must handle real-time constraints, limited resources and tight hardware/software integration.

4.1 Debugging Techniques

- **Emulation and simulation:** replicate the embedded environment without hardware; catches issues early.
- **Debugging probes and tools:** JTAG probes, oscilloscopes and logic analyzers trace hardware signals and monitor behaviour in real time.
- **Remote debugging:** diagnose deployed devices without physical access, cutting downtime and maintenance cost.
- **RTOS analysis:** OS-provided tools find task-scheduling problems, memory leaks and resource conflicts.
- **Logging and tracing:** capture events, errors and data flow to study runtime behaviour.

4.2 Testing Techniques

- **Unit testing:** isolated tests of modules or functions (frameworks: Unity, CppUTest).

- **Integration testing:** validates interfaces, data communication and functional integration of hardware and software.
- **Static code analysis:** Coverity, SonarQube or Lint find bugs, memory leaks and coding-standard violations.
- **Hardware-in-the-loop (HIL) testing:** real hardware inside a simulated environment, to validate software against actual hardware.
- **Performance and stress testing:** behaviour under extreme conditions; memory use, response time and resource use.

4.3 Best Practices

- **Early testing:** start as early as possible in development.
- **Continuous integration (CI):** automate testing so every change is validated.
- **Documentation:** record tests, issues and debugging steps for future reference.