03. Basic Embedded Systems - Comprehensive Technical Reference

> **Module Focus**: Technical MCQs & Numerical Problems on Microcontrollers, 8051 Architecture, Registers, Interrupts, Timers, Peripherals (ADC/DAC/PWM) & Communication Protocols (UART/SPI/I2C/CAN).

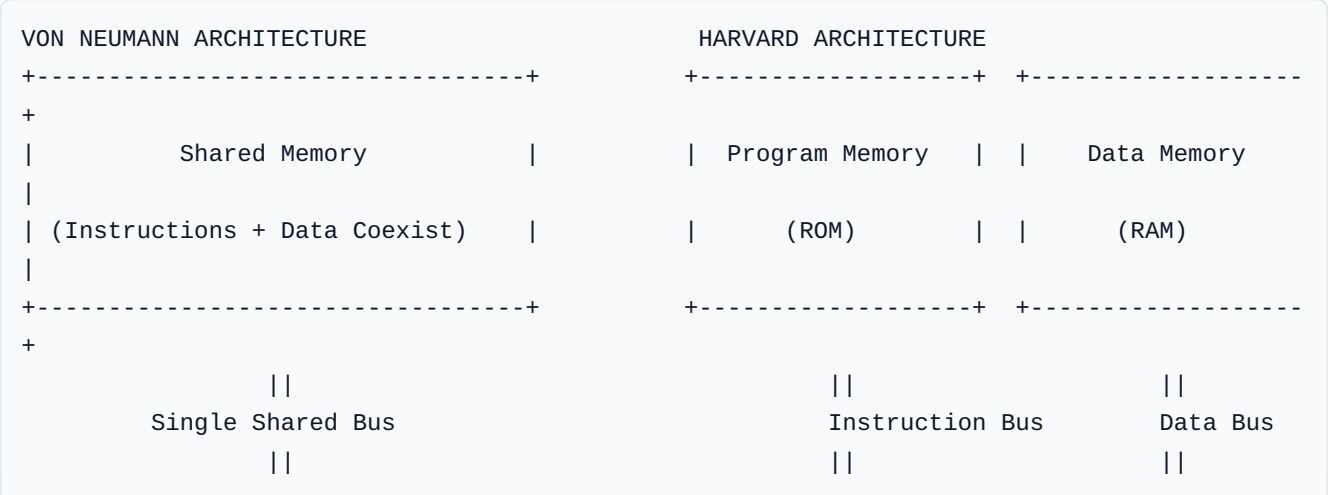
---

1. Microprocessor vs Microcontroller Architecture

1.1 Fundamental Differences

Feature	Microprocessor (e.g., Intel x86, ARM Cortex-A)	Microcontroller (e.g., 8051, Microchip PIC, STM32, ATmega328P)
Silicon Integration	Contains **only the CPU** (ALU, Registers, Control Unit) on-chip.	Contains **CPU, RAM, ROM/Flash, Timers, I/O Ports, ADC, UART** integrated on a single silicon die.
System Cost & Size	High cost, large PCB footprint due to external RAM/ROM/IC chips.	Low cost, compact footprint suitable for embedded products.
Power Consumption	High power consumption (requires dedicated cooling/heat sinks).	Ultra-low power (operates on microamps $ \mu\text{A}$, battery friendly).
Memory Access Speed	Faster clock speeds (GHz range), uses multi-level hardware cache architecture.	Moderate clock speeds (MHz range, e.g., 12 MHz – 168 MHz), no hardware cache.
Application Domain	General-purpose computing (Laptops, Servers, Smartphones).	Dedicated real-time control (Washing machines, Automotive ECUs, Medical devices).

1.2 Von Neumann vs Harvard Memory Architecture



```
      +--------------------+                          +---------------------------------+
      |      CPU Core      |                          |            CPU Core             |
      +--------------------+                          +---------------------------------+
```

Detailed Comparison:

1. **Von Neumann Architecture**:
2. **Structure**: Single unified memory space holds both program instructions and runtime data. Connected to CPU via a single shared Address/Data bus.
3. **Bottleneck**: Known as the **Von Neumann Bottleneck**. CPU cannot read an instruction AND read/write data memory simultaneously in the same clock cycle.
4. **Examples**: Standard x86 PC architectures, early microprocessors (8085).
5. **Harvard Architecture**:
6. **Structure**: Separate physical memory spaces for Program Memory (Flash/ROM) and Data Memory (RAM), connected via separate independent Instruction and Data buses.
7. **Advantage**: Allows simultaneous instruction fetching and data reading/writing in the same clock cycle, enabling **Pipelining**.
8. **Examples**: 8051 (Logical Harvard), ARM Cortex-M series, AVR, PIC microcontrollers.

---

2. The 8051 Microcontroller Architecture & Memory Map

2.1 Internal Architecture Overview

The Intel 8051 is an 8-bit CISC microcontroller operating on Harvard architecture featuring: - **8-bit CPU** with Accumulator (`ACC`) and `B` register. - **4 KB Internal ROM/Flash** (Program Memory address range `0000H` to `0FFFH`). - **128 Bytes Internal RAM** (Data Memory address range `00H` to `7FH`) [Expandable to 256 Bytes in 8052]. - **Special Function Registers (SFRs)** (Address range `80H` to `FFH`). - **Four 8-bit I/O Ports**: Port 0 (`P0`), Port 1 (`P1`), Port 2 (`P2`), Port 3 (`P3`) (Total 32 I/O lines). - **Two 16-bit Timers/Counters**: Timer 0 (`T0`) and Timer 1 (`T1`). - **Full-Duplex Serial UART Port**. - **5 Interrupt Sources** (2 External, 2 Timer, 1 Serial).

2.2 Complete 8051 Memory Map

1. Program Memory (ROM) Map:

- **Internal ROM**: `0000H` to `0FFFH` (4 KB).
- **External ROM**: `1000H` to `FFFFH` (Up to 64 KB total).
- **$\overline{\text{EA}}$ Pin (External Access)**:
- If $\overline{\text{EA}} = 1$ (High): CPU executes instructions from Internal ROM (`0000H` – `0FFFH`) first, then automatically switches to External ROM (`1000H` – `FFFFH`).
- If $\overline{\text{EA}} = 0$ (Low): CPU bypasses internal ROM completely and fetches ALL instructions from External ROM (`0000H` – `FFFFH`).
- **ROM Access Instructions**: `MOVC A, @A+DPTR` or `MOVC A, @A+PC` .

```
PROGRAM MEMORY (ROM) MAP                DATA MEMORY (RAM) MAP
+-----------------------+ FFFFH         +-----------------------+ FFFFH
|                       |               |                       |
|   External Code ROM   |               |   External Data RAM   |
|     (Up to 64 KB)     |               |    (MOVX, 64 KB)      |
|                       |               |                       |
+-----------------------+ 1000H         +-----------------------+ 0000H
|   Internal Code ROM   | 0FFFH         +-----------------------+ FFH
|        (4 KB)         |               | Special Function Regs | Direct Addressing Only
+-----------------------+ 0000H         |    (SFRs: 80H-FFH)    |
                                        +-----------------------+ 7FH
                                        | Scratchpad RAM (80B)  |
                                        +-----------------------+ 30H
                                        | Bit-Addressable (16B) | 2FH / Bit 7FH-00H
                                        +-----------------------+ 20H
                                        | Bank 3 (R0-R7)        | 1FH - 18H
                                        | Bank 2 (R0-R7)        | 17H - 10H
                                        | Bank 1 (R0-R7)        | 0FH - 08H
                                        | Bank 0 (R0-R7)        | 07H - 00H
                                        +-----------------------+
```

2. Internal Data RAM Structure (`00H` – `7FH` / `FFH`):

```
+-----------------------+ 7FH
|  Scratchpad RAM       | General-purpose variables, buffers & System Stack
|  (80 Bytes: 30H-7FH)  | Addressable via Direct or Indirect (@R0, @R1)
+-----------------------+ 2FH
|  Bit-Addressable RAM  | 16 Bytes (128 bits: Bit addresses 00H to 7FH)
|  (16 Bytes: 20H-2FH)  | Direct bit manipulation (SETB 07H, CLR 20H)
+-----------------------+ 1FH
|  Register Bank 3      | 8 Bytes (R0 - R7) [18H - 1FH]
+-----------------------+ 17H
|  Register Bank 2      | 8 Bytes (R0 - R7) [10H - 17H]
+-----------------------+ 0FH
|  Register Bank 1      | 8 Bytes (R0 - R7) [08H - 0FH]
+-----------------------+ 07H
|  Register Bank 0      | 8 Bytes (R0 - R7) [00H - 07H] Default Bank on Reset
+-----------------------+ 00H
```

Bit-Addressable RAM Breakdown (`20H` – `2FH`):

- RAM byte `20H` holds Bit addresses `00H` to `07H`.
- RAM byte `21H` holds Bit addresses `08H` to `0FH`.
- RAM byte `2FH` holds Bit addresses `78H` to `7FH`.
- *Example*: Bit address `05H` corresponds to Bit 5 of RAM location `20H`.

3. Special Function Registers (SFR) Map (`80H` – `FFH`):

- SFRs control peripherals, I/O ports, timers, interrupts, and CPU execution state.
- **Addressing Mode**: SFRs are accessed using **Direct Addressing ONLY**.
- **Bit-Addressable SFRs**: SFR addresses that end in `0` or `8` (e.g., `80H`, `88H`, `90H`, `98H`, `A0H`, `A8H`, `B0H`, `B8H`, `D0H`, `E0H`, `F0H`) are bit-addressable.

SFR Symbol	Name	Direct Address	Bit Addressable?	Reset Value
`ACC` / **`A`**	Accumulator	`E0H`	Yes (`E0H` – `E7H`)	`00H`
`B`	Multiplication / Division Register	`F0H`	Yes (`F0H` – `F7H`)	`00H`
`PSW`	Program Status Word	`D0H`	Yes (`D0H` – `D7H`)	`00H`
`SP`	Stack Pointer	`81H`	No	`07H`
`DPL`	Data Pointer Low byte	`82H`	No	`00H`
`DPH`	Data Pointer High byte	`83H`	No	`00H`
`P0`	Port 0 Latch	`80H`	Yes (`80H` – `87H`)	`FFH`
`P1`	Port 1 Latch	`90H`	Yes (`90H` – `97H`)	`FFH`
`P2`	Port 2 Latch	`A0H`	Yes (`A0H` – `A7H`)	`FFH`
`P3`	Port 3 Latch	`B0H`	Yes (`B0H` – `B7H`)	`FFH`
`TMOD`	Timer Mode Register	`89H`	No	`00H`
`TCON`	Timer Control Register	`88H`	Yes (`88H` – `8FH`)	`00H`
`TL0` / **`TH0`**	Timer 0 Low / High Byte	`8AH` / `8BH`	No	`00H`
`TL1` / **`TH1`**	Timer 1 Low / High Byte	`8CH` / `8DH`	No	`00H`
`IE`	Interrupt Enable Register	`A8H`	Yes (`A8H` – `AFH`)	`00H`
`IP`	Interrupt Priority Register	`B8H`	Yes (`B8H` – `BFH`)	`00H`
`SCON`	Serial Control Register	`98H`	Yes (`98H` – `9FH`)	`00H`
`SBUF`	Serial Data Buffer	`99H`	No	Indeterminate

2.3 Key 8051 Registers in Detail

1. Accumulator (`ACC` or `A`) [Address: `E0H`]:

- 8-bit primary register used for all arithmetic (ADD, SUBB), logical (ANL, ORL, XRL), and data transfer operations.
- Direct SFR Address: `0E0H`. Bit address range: `E0H` (LSB) to `E7H` (MSB).

2. B Register [Address: `F0H`]:

- 8-bit register used exclusively alongside Accumulator for multiplication (`MUL AB`) and division (`DIV AB`).
- `MUL AB`: Multiplies unsigned 8-bit value in `A` by unsigned 8-bit value in `B`.
- Product is 16-bit: High byte stored in `B`, Low byte stored in `A`.
- Flag state: `CY` is cleared. `OV` is set to `1` if product > 255 (`B` $\neq 0$), otherwise `OV = 0`.
- `DIV AB`: Divides unsigned 8-bit value in `A` by unsigned 8-bit value in `B` (A / B).
- Quotient stored in `A`, Remainder stored in `B`.
- Flag state: `CY` is cleared. If `B = 0` (division by zero), quotient/remainder are undefined and Overflow flag `OV` is set to `1`.

3. Program Status Word (`PSW`) [Address: `D0H`]:

8-bit flag register reflecting current status of CPU operations:

Bit	Symbol	Bit Address	Description
`PSW.7`	**`CY`**	`D7H`	**Carry Flag**: Set if an arithmetic operation produces a carry out of bit 7 (or borrow in subtraction).
`PSW.6`	**`AC`**	`D6H`	**Auxiliary Carry Flag**: Set if carry occurs from bit 3 to bit 4 (used for BCD math).
`PSW.5`	**`F0`**	`D5H`	**User Flag 0**: General-purpose software flag.
`PSW.4`	**`RS1`**	`D4H`	**Register Bank Select 1**
`PSW.3`	**`RS0`**	`D3H`	**Register Bank Select 0**
`PSW.2`	**`OV`**	`D2H`	**Overflow Flag**: Set during signed arithmetic overflow (carry into bit 7 $\oplus$ carry out of bit 7).
`PSW.1`	**`--`**	`D1H`	User-definable flag / Reserved.
`PSW.0`	**`P`**	`D0H`	**Parity Flag**: Set to `1` if Accumulator contains an **odd number of 1s** (Odd parity check). Clear if even.

Register Bank Selection Table (`RS1`, `RS0`):

$$\begin{array}{|c|c|c|c|} \hline \mathbf{RS1} & \mathbf{RS0} & \mathbf{Selected\ Register\ Bank} & \mathbf{RAM\ Address\ Range} \\ \hline 0 & 0 & \text{Bank 0 (Default)} & \text{00H – 07H} \\ \hline 0 & 1 & \text{Bank 1} & \text{08H – 0FH} \\ \hline 1 & 0 & \text{Bank 2} & \text{10H – 17H} \\ \hline 1 & 1 & \text{Bank 3} & \text{18H – 1FH} \\ \hline \end{array}$$

Worked Example: Flag States after `ADD A, R0`

Suppose $A = \text{0x85}\ (1000\ 0101_2)$ and $R0 = \text{0x9B}\ (1001\ 1011_2)$: $$\begin{array}{r@{\quad}l} 1000\ 0101 & (\text{0x85}) \\ + 1001\ 1011 & (\text{0x9B}) \\ \hline 1\ 0010\ 0000 & (\text{Result } A = \text{0x20}, \text{Carry out} = 1) \end{array}$$ - **Carry Out from Bit 7**: Yes $\implies \mathbf{CY = 1}$. - **Carry Out from Bit 3 to Bit 4**: Bit 3 sum $(0+1) + \text{carry } 1 = 2 \implies$ Carry to Bit 4 $\implies \mathbf{AC = 1}$. - **Overflow**

(`OV`): Carry into Bit 7 (1) $\oplus$ Carry out of Bit 7 (1) = $1 \oplus 1 = 0 \implies \mathbf{OV = 0}$. - **Parity (`P`)**: Result $A = \text{0x20} = 0010\ 0000_2$ (contains one `1`, odd count) $\implies \mathbf{P = 1}$.

4. Data Pointer Register (`DPTR`):

- 16-bit register composed of two 8-bit SFRs: `DPH` (High byte at `83H`) and `DPL` (Low byte at `82H`).
- Used to hold 16-bit memory addresses for external Data RAM (`MOVX`) and Program ROM (`MOVC`).
- *Instructions using DPTR*:
- `MOV DPTR, #1234H` : Loads 16-bit immediate value (`DPH=12H` , `DPL=34H`).
- `MOVX A, @DPTR` : Reads byte from external RAM at 16-bit address stored in `DPTR` .
- `MOVC A, @A+DPTR` : Reads lookup table byte from ROM at address `(A + DPTR)` .
- `INC DPTR` : Increments 16-bit DPTR (Note: There is no `DEC DPTR` instruction in standard 8051!).

---

3. Interrupts, Timers & Counters Mechanics

3.1 Interrupt Vector Table (IVT) Address Map

When an interrupt occurs, CPU halts current execution, pushes 16-bit Program Counter (`PC`) onto the stack (`SP` incremented by 2), and branches to the designated vector address in program ROM:

Interrupt Source	Flag / Hardware Event	Vector Address	Vector Pin	Priority (Default)
System Reset	Power-on / RST pin HIGH	`0000H`	`RST`	Highest
External Interrupt 0 (`INT0`)	Low level / Falling edge on `P3.2`	`0003H`	`P3.2`	1
Timer 0 Interrupt (`TF0`)	Timer 0 counter overflow	`000BH`	Internal	2
External Interrupt 1 (`INT1`)	Low level / Falling edge on `P3.3`	`0013H`	`P3.3`	3
Timer 1 Interrupt (`TF1`)	Timer 1 counter overflow	`001BH`	Internal	4
Serial Port Interrupt (`RI` / `TI`)	Rx character ready (`RI`) / Tx complete (`TI`)	`0023H`	`P3.0` / `P3.1`	5
Timer 2 Interrupt (`TF2` / `EXF2`)	Timer 2 overflow (8052 only)	`002BH`	Internal	Lowest

---

3.2 Interrupt Control Registers (`IE` , `IP` , `TCON`)

1. Interrupt Enable Register (`IE`) [Address: `A8H` - Bit Addressable]:

Format: `[ EA | ET2 | ES | ET1 | EX1 | ET0 | EX0 ]`

- `EA` **(Bit 7)**: Global Interrupt Enable (`1` = Enable interrupts, `0` = Disable all interrupts).

- `ET2` **(Bit 5)**: Timer 2 Interrupt Enable (8052).
- `ES` **(Bit 4)**: Serial Port Interrupt Enable.
- `ET1` **(Bit 3)**: Timer 1 Interrupt Enable.
- `EX1` **(Bit 2)**: External Interrupt 1 Enable.
- `ET0` **(Bit 1)**: Timer 0 Interrupt Enable.
- `EX0` **(Bit 0)**: External Interrupt 0 Enable.

2. Interrupt Priority Register (`IP`) [Address: `B8H` - Bit Addressable]:

Format: `[ - | - | PT2 | PS | PT1 | PX1 | PT0 | PX0 ]` - Setting a bit to `1` gives that interrupt high priority. High-priority interrupts can interrupt low-priority interrupt service routines (ISRs).

3. Interrupt Triggering Control (`TCON` bits):

- `IT0` / `IT1`: External Interrupt 0/1 Trigger Type Select:
- `0` $\to$ Low Level-triggered interrupt on pin `P3.2` / `P3.3`.
- `1` $\to$ Falling Edge-triggered interrupt on pin `P3.2` / `P3.3`.
- `IE0` / `IE1`: External Interrupt 0/1 Edge Flag (automatically cleared by hardware when vectoring to ISR).

3.3 Timers and Counters Operation & Modes

The 8051 contains two 16-bit timers/counters: **Timer 0** (`TH0` + `TL0`) and **Timer 1** (`TH1` + `TL1`).

1. `TMOD` Register (Timer Mode Control Register) [Address: `89H` - Not Bit Addressable]:

Format: `[ GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 ]`
(Upper nibble configures Timer 1, Lower nibble configures Timer 0)

- `GATE`:
- `0` $\to$ Timer is enabled when software bit `TRx` is set (`TRx = 1`).
- `1` $\to$ Timer is enabled ONLY when `TRx = 1` AND external pin `INTx` (`P3.2` or `P3.3`) is HIGH. (Used for measuring external pulse width).
- `C/T` **(Counter/Timer Select)**:
- `0` $\to$ **Timer Mode**: Input clock is internal machine clock ($\frac{f_{\text{osc}}}{12}$).
- `1` $\to$ **Counter Mode**: Input clock comes from external pin pulses (`P3.4` for `T0`, `P3.5` for `T1`).
- `M1, M0` **(Mode Bits)**:

$$\begin{array}{|c|c|l|} \hline \mathbf{M1} & \mathbf{M0} & \mathbf{Mode} & \mathbf{Operating\ Description} \\ \hline 0 & 0 & \text{Mode 0} & \text{13-bit Timer (8-bit THx + 5-bit TLx, counts 0000H to 1FFFH / 8192 counts)} \\ \hline 0 & 1 & \text{Mode 1} & \mathbf{16\text{-bit Timer}} \text{ (Full 0000H to FFFFH range, 65536 counts)} \\ \hline 1 & 0 & \text{Mode 2} & \mathbf{8\text{-bit Auto-Reload}} \text{ (TLx counts; overflow reloads THx value automatically)} \\ \hline 1 & 1 & \text{Mode 3} & \text{Split Timer mode (Timer 0 splits into two independent 8-bit timers)} \\ \hline \end{array}$$

3.4 Timer Calculation Worked Examples

Example 1: 16-bit Timer Delay Calculation (Mode 1)

Problem: Calculate the 16-bit load values for `TH0` and `TL0` to generate a **5 ms delay** assuming crystal frequency $f_{\text{osc}} = 12\text{ MHz}$.

Step-by-step Solution: 1. **Calculate Machine Cycle Clock Frequency (f_{mc})**: $$f_{\text{mc}} = \frac{f_{\text{osc}}}{12} = \frac{12\text{ MHz}}{12} = 1\text{ MHz}$$ 2. **Calculate Machine Cycle Time (T_{mc})**: $$T_{\text{mc}} = \frac{1}{f_{\text{mc}}} = \frac{1}{1\text{ MHz}} = 1\ \mu\text{s}$$ 3. **Calculate Required Number of Clock Counts (N)**: $$N = \frac{\text{Target Delay}}{T_{\text{mc}}} = \frac{5\text{ ms}}{1\ \mu\text{s}} = \frac{5000\ \mu\text{s}}{1\ \mu\text{s}} = 5000\text{ counts}$$ 4. **Calculate Timer Initial Load Value (X)**: Since Mode 1 is a 16-bit timer with maximum count capacity $2^{16} = 65536$: $$X = 65536 - N = 65536 - 5000 = 60536_{10}$$ 5. **Convert Initial Value to Hexadecimal**: $$60536_{10} = \text{EC78}_H$$ - `TH0` $= \text{ECH}$ - `TL0` $= \text{78H}$

Example 2: Baud Rate Generation Calculation for UART (Mode 2)

Problem: Calculate the auto-reload value for `TH1` to generate a **9600 Baud Rate** for 8051 UART using Timer 1 in Mode 2 (8-bit auto-reload), given crystal frequency $f_{\text{osc}} = 11.0592\text{ MHz}$ and `SMOD = 0`.

Formula: $$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{f_{\text{osc}}}{12 \times (256 - \text{TH1})}$$

Step-by-step Solution: 1. With $\text{SMOD} = 0$, $2^{\text{SMOD}} = 2^0 = 1$: $$\text{UART Clock Frequency} = \frac{f_{\text{osc}}}{32 \times 12} = \frac{11.0592\text{ MHz}}{384} = 28800\text{ Hz}$$ 2. Set up equation for baud rate = 9600: $$9600 = \frac{28800}{256 - \text{TH1}}$$ 3. Solve for $(256 - \text{TH1})$: $$256 - \text{TH1} = \frac{28800}{9600} = 3$$ 4. Calculate `TH1`: $$\text{TH1} = 256 - 3 = 253_{10} = \text{FD}_H\quad (\text{or } -3 \text{ in 2's complement})$$

---

4. Embedded Communication Protocols

4.1 Comparative Protocol Matrix

Parameter	UART	SPI	I^2C	CAN
Full Name	Universal Asynchronous Receiver-Transmitter	Serial Peripheral Interface	Inter-Integrated Circuit	Controller Area Network
Clocking Type	**Asynchronous** (No shared clock)	**Synchronous** (Shared SCLK)	**Synchronous** (Shared SCL)	**Asynchronous** (Bit stuffing & synchronization)
Data Lines	2 (`TxD` , `RxD`)	4 (`MOSI` , `MISO` , `SCLK` , `SS` / `CS`)	2 (`SDA` , `SCL`)	2 (`CAN_H` , `CAN_L` differential pair)

Parameter	UART	SPI	I^2C	CAN
Duplex Mode	Full Duplex	Full Duplex	Half Duplex	Half Duplex
Bus Topology	Point-to-Point (2 devices)	Master - Multi-Slave	Multi-Master, Multi-Slave	Multi-Master Bus
Typical Speed	9600 bps – 115.2 kbps	10 Mbps – 50 Mbps (Very Fast)	100 kbps (Standard), 400 kbps (Fast), 3.4 Mbps	125 kbps – 1 Mbps
Device Addressing	None (Direct wiring)	Hardware Chip Select (`CS` pin per slave)	Software 7-bit or 10-bit address	Message ID priority filtering
Max Distance	Short (< 15 m RS232)	Short (< 10 cm PCB)	Short (< 2 m PCB)	Long (Up to 40 m @ 1 Mbps, 1 km @ 40 kbps)

4.2 Detailed Protocol Mechanics

1. UART (Universal Asynchronous Receiver-Transmitter):

- **Frame Format**:

```
Idle (1) ---> [ START (0) ] + [ 5-8 Data Bits (LSB First) ] + [ Optional Parity Bit ] +
[ STOP Bit(s) (1) ] ---> Idle (1)
```

- **Level Shifting**: Microcontroller TTL levels ($0\text{V} / 5\text{V}$) require a **MAX232 driver IC** to interface with RS-232 levels (Logic `0` = $+3\text{V}$ to $+15\text{V}$, Logic `1` = -3V to -15V).

---

2. SPI (Serial Peripheral Interface):

- **Signals**:
- `MOSI`: Master Output Slave Input.
- `MISO`: Master Input Slave Output.
- `SCLK`: Serial Clock generated by Master.
- `SS` / `CS`: Slave Select / Chip Select (Active Low).
- **SPI Clock Modes (`CPOL` and `CPHA`)**:

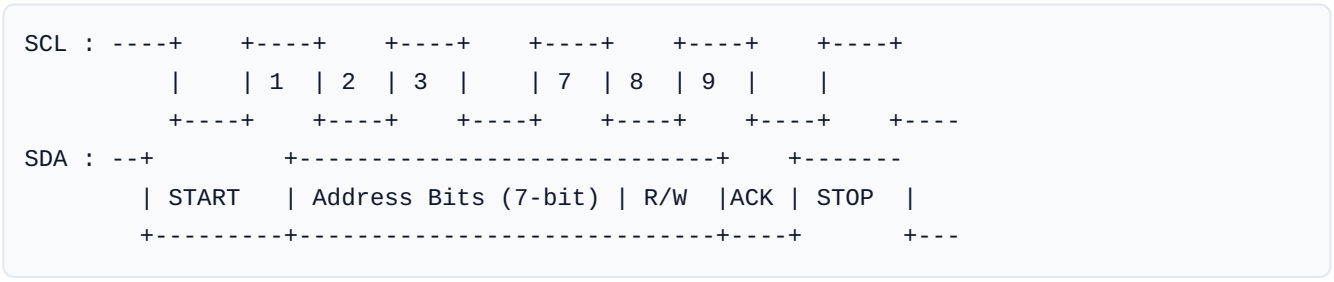
```
CPOL = 0 : Idle Clock state is LOW
CPOL = 1 : Idle Clock state is HIGH
CPHA = 0 : Data sampled on 1st Leading Edge
CPHA = 1 : Data sampled on 2nd Trailing Edge
```

$$\begin{array}{|c|c|c|ll} \hline \mathbf{SPI\ Mode} & \mathbf{CPOL} & \mathbf{CPHA} & \mathbf{Sampling\ Clock\ Edge} \\ \hline \text{Mode 0} & 0 & 0 & \text{Rising edge (Idle Low)} \\ \hline \text{Mode 1} & 0 & 1 & \text{Falling edge (Idle Low)} \\ \hline \text{Mode 2} & 1 & 0 & \text{Falling edge (Idle High)} \\ \hline \text{Mode 3} & 1 & 1 & \text{Rising edge (Idle High)} \\ \hline \end{array}$$

---

3. I^2C (Inter-Integrated Circuit):

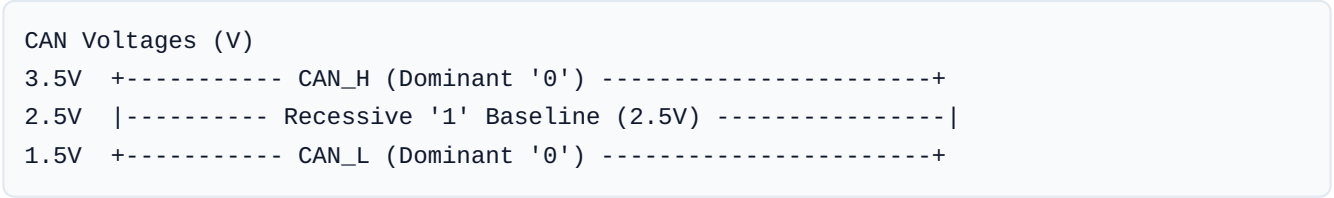
- **Lines**: `SDA` (Serial Data) and `SCL` (Serial Clock).
- **Driver Architecture**: Both lines use **Open-Drain / Open-Collector** drivers with external **Pull-Up Resistors** ($R_P \approx 2.2\text{ k}\Omega - 10\text{ k}\Omega$).
- **Pull-Up Resistor Calculation**: $$R_{P(\text{min})} = \frac{V_{DD} - V_{OL(\text{max})}}{I_{OL}},\quad R_{P(\text{max})} = \frac{t_r}{0.8473 \times C_{\text{bus}}}$$
- **Bus Framing Conditions**:
- **START Condition**: `SDA` transitions from **HIGH to LOW** while `SCL` remains **HIGH**.
- **STOP Condition**: `SDA` transitions from **LOW to HIGH** while `SCL` remains **HIGH**.
- **ACK/NACK Bit**: On the 9th clock cycle, transmitter releases `SDA`; receiver pulls `SDA` LOW (`ACK = 0`) or leaves it HIGH (`NACK = 1`).



---

4. CAN Bus (Controller Area Network):

- **Physical Layer**: Uses a twisted-pair line (`CAN_H` and `CAN_L`) terminated with $120\ \Omega$ resistors at each physical end.
- **Differential Signal Voltage Levels**: $$V_{\text{diff}} = V_{\text{CAN\_H}} - V_{\text{CAN\_L}}$$
- **Dominant Bit (`0`)**: $V_{\text{CAN\_H}} \approx 3.5\text{V}$, $V_{\text{CAN\_L}} \approx 1.5\text{V} \implies V_{\text{diff}} \approx 2.0\text{V}$. Overrides Recessive.
- **Recessive Bit (`1`)**: $V_{\text{CAN\_H}} \approx 2.5\text{V}$, $V_{\text{CAN\_L}} \approx 2.5\text{V} \implies V_{\text{diff}} \approx 0.0\text{V}$.



- **Arbitration Mechanism**: Uses **Carrier Sense Multiple Access with Collision Resolution (CSMA/CR)** via non-destructive bitwise arbitration based on message Identifiers (Lower ID numerical value = Higher Priority).

- **Bit Stuffing**: After 5 consecutive identical bits in a frame, the transceiver automatically inserts 1 inverted bit to ensure clock synchronization.

---

5. Analog Peripherals: PWM, ADC & DAC

5.1 Pulse Width Modulation (PWM)

PWM generates an analog-equivalent output voltage by varying the pulse width (T_{ON}) of a periodic digital square wave at constant frequency f:

```
+------+      +------+
|      |      |      |
| TON  | TOFF |      |
+------+------+------+
|<--- T total ------>|
```

Key Equations:

$$\text{Period } T = T_{\text{ON}} + T_{\text{OFF}} = \frac{1}{f_{\text{PWM}}}$$ $$\text{Duty Cycle } D = \left(\frac{T_{\text{ON}}}{T_{\text{ON}} + T_{\text{OFF}}} \right) \times 100\% = \frac{T_{\text{ON}}}{T} \times 100\%$$ $$\text{Average Output Voltage } V_{\text{avg}} = D \times V_{\text{max}} + (1 - D) \times V_{\text{min}}$$ $$\text{RMS Output Voltage } V_{\text{rms}} = \sqrt{D \cdot V_{\text{max}}^2 + (1 - D) \cdot V_{\text{min}}^2}$$ $$\text{PWM Bit Resolution } N = \log_2\left(\frac{f_{\text{timer}}}{f_{\text{PWM}}} \right)$$

Worked Numerical Example: Motor Speed Control

Problem: A PWM system operating at a carrier frequency of $f = 2\text{ kHz}$ controls a DC motor operating from $V_{\text{max}} = 12\text{ V}$ ($V_{\text{min}} = 0\text{V}$). Calculate: 1. Total period T. 2. Required T_{ON} and T_{OFF} for an average motor voltage of $V_{\text{avg}} = 7.2\text{ V}$. 3. Duty cycle percentage $D\%$.

Solution: 1. **Total Period (T)**: $$T = \frac{1}{f} = \frac{1}{2000\text{ Hz}} = 0.0005\text{ s} = 500\ \mu\text{s}$$ 2. **Duty Cycle Percentage ($D\%$)**: $$V_{\text{avg}} = D \times V_{\text{max}} \implies 7.2\text{ V} = D \times 12\text{ V} \implies D = \frac{7.2}{12} = 0.60 = 60\%$$ 3. **Pulse Widths (T_{ON} and T_{OFF})**: $$T_{\text{ON}} = D \times T = 0.60 \times 500\ \mu\text{s} = 300\ \mu\text{s}$$ $$T_{\text{OFF}} = T - T_{\text{ON}} = 500\ \mu\text{s} - 300\ \mu\text{s} = 200\ \mu\text{s}$$

5.2 Analog-to-Digital Converters (ADC)

ADC Architecture Comparison:

Feature	SAR (Successive Approx.)	Flash ADC (Parallel)	Dual-Slope Integration
Conversion Speed	Moderate (N clock cycles)	Extremely Fast (1 clock cycle)	Slow (2^N clock cycles)
Hardware Complexity	1 Comparator + DAC + Logic	$2^N - 1$ Comparators	Integrator + Comparator + Counter
Resolution	High (8 – 18 bits)	Low-Medium (6 – 10 bits)	Very High (16 – 24 bits)
Noise Immunity	Moderate	Low	High (Rejects line noise)
Common Application	Embedded Microcontrollers	Digital Oscilloscopes, Video	Digital Multimeters (DMM)

5.3 ADC Step Size & Quantization Mathematics

Key Formulas:

1. **Resolution / Step Size (1 LSB Voltage)**: $$\text{Step Size (LSB)} = \frac{V_{\text{ref+}} - V_{\text{ref-}}}{2^N - 1} \approx \frac{V_{\text{ref}}}{2^N}$$
2. **Digital Code Output (D)**: $$D = \left\lfloor \frac{V_{\text{analog}} - V_{\text{ref-}}}{\text{Step Size}} \right\rfloor = \left\lfloor \frac{V_{\text{analog}} - V_{\text{ref-}}}{V_{\text{ref+}} - V_{\text{ref-}}} \times (2^N - 1) \right\rfloor$$
3. **Reconstructed Analog Voltage (V_{recon})**: $$V_{\text{recon}} = D \times \text{Step Size}$$
4. **Quantization Error (e_q)**: $$e_q = V_{\text{analog}} - V_{\text{recon}} \quad \left(-\frac{1}{2}\text{LSB} \le e_q \le +\frac{1}{2}\text{LSB} \right)$$
5. **Effective Number of Bits (ENOB)**: $$\text{ENOB} = \frac{\text{SNR}_{\text{dB}} - 1.76}{6.02}$$

5.4 Complete ADC Worked Numerical Examples

Worked Example 1: 10-bit SAR ADC Math

Problem: A 10-bit ADC has a reference voltage range of $V_{\text{ref-}} = 0\text{V}$ and $V_{\text{ref+}} = 5.0\text{V}$. 1. Calculate the step size (LSB voltage). 2. Find the output digital code (in decimal and hex) for an input voltage $V_{\text{in}} = 3.2\text{ V}$. 3. Calculate the reconstructed analog voltage and quantization error.

Solution: 1. **Step Size (LSB)**: $$\text{LSB} = \frac{V_{\text{ref+}} - V_{\text{ref-}}}{2^{10} - 1} = \frac{5.0\text{ V}}{1023} = 4.8876\text{ mV} \ (0.0048876\text{ V})$$ 2. **Digital Code Output (D)**: $$D = \left\lfloor \frac{V_{\text{in}}}{\text{LSB}} \right\rfloor = \left\lfloor \frac{3.2\text{ V}}{0.0048876\text{ V}} \right\rfloor = \lfloor 654.71 \rfloor = 654_{10}$$ Converting to Hexadecimal: $$654_{10} = 29\text{E}_H \quad (0010\ 1001\ 1110_2)$$ 3. **Reconstructed Analog Voltage & Quantization Error**: $$V_{\text{recon}} = 654 \times

0.0048876\text{ V} = 3.1965\text{ V}$$ $$e_q = V_{\text{in}} - V_{\text{recon}} = 3.2000\text{ V} - 3.1965\text{ V} = +0.0035\text{ V} = +3.5\text{ mV}$$ *(Notice that $e_q = 3.5\text{ mV} < 4.8876\text{ mV} = 1\text{ LSB}$)*

Worked Example 2: 12-bit ADC with Non-Zero Reference

Problem: A 12-bit ADC operates with $V_{\text{ref-}} = 0.5\text{ V}$ and $V_{\text{ref+}} = 3.3\text{ V}$. 1. Compute the step size. 2. Determine the analog input voltage corresponding to a digital code of `0x800` (2048_{10}).

Solution: 1. **Step Size**: $$\text{Span} = V_{\text{ref+}} - V_{\text{ref-}} = 3.3\text{ V} - 0.5\text{ V} = 2.8\text{ V}$$ $$\text{LSB} = \frac{2.8\text{ V}}{2^{12} - 1} = \frac{2.8\text{ V}}{4095} = 0.68376\text{ mV} \ (0.00068376\text{ V})$$ 2. **Analog Input Voltage for Code $D = 2048$**: $$V_{\text{in}} = V_{\text{ref-}} + (D \times \text{LSB}) = 0.5\text{ V} + (2048 \times 0.00068376\text{ V}) = 0.5\text{ V} + 1.4003\text{ V} = 1.9003\text{ V}$$

Worked Example 3: Sensor Interfacing Math (LM35 + 10-bit ADC)

Problem: An LM35 temperature sensor produces an analog output voltage of $V_{\text{sensor}} = 10\text{ mV}/^\circ\text{C}$ (e.g., 250 mV at 25°C). The sensor is connected directly to a 10-bit ADC with an internal reference $V_{\text{ref}} = 2.56\text{ V}$ ($V_{\text{ref-}} = 0\text{V}$). 1. Calculate the ADC resolution in terms of temperature ($^\circ\text{C}$ per LSB). 2. Calculate the temperature reading if the ADC outputs digital code $D = 184_{10}$.

Solution: 1. **ADC Step Size Voltage**: $$\text{LSB} = \frac{2.56\text{ V}}{1023} = 2.50244\text{ mV/LSB}$$ 2. **Temperature Resolution per LSB**: $$\text{Temp Resolution} = \frac{\text{LSB Voltage}}{\text{Sensor Sensitivity}} = \frac{2.50244\text{ mV/LSB}}{10\text{ mV}/^\circ\text{C}} = 0.25024^\circ\text{C/LSB}$$ 3. **Temperature for Code $D = 184$**: $$\text{Measured Voltage } V_{\text{in}} = 184 \times 2.50244\text{ mV} = 460.45\text{ mV}$$ $$\text{Temperature } T = \frac{V_{\text{in}}}{10\text{ mV}/^\circ\text{C}} = \frac{460.45\text{ mV}}{10\text{ mV}/^\circ\text{C}} = 46.045^\circ\text{C}$$ *(Alternatively: $T = 184 \times 0.25024^\circ\text{C} = 46.045^\circ\text{C}$)*

---

6. 🎯 CoCubes Embedded Systems Technical MCQ Master Patterns & Quick-Fire Reference

6.1 8051 Architectural Reset States & Register Secrets

CoCubes frequently tests default values and hardware specifics upon system reset:

Register / Pin	Value Post-Reset	CoCubes MCQ Exam Note
Stack Pointer (`SP`)	`07H`	Pushes increment `SP` first; first pushed item goes to RAM `08H` (Bank 1 `R0`).
Program Counter (`PC`)	`0000H`	CPU begins fetching code execution at ROM `0000H`.

Register / Pin	Value Post-Reset	CoCubes MCQ Exam Note
Ports (`P0`, `P1`, `P2`, `P3`)	`FFH`	All port pins default to HIGH logic state (Input mode).
`PSW` / `ACC` / `B`	`00H`	Flags and accumulators cleared to zero.
Port 0 Architecture	Open-Drain	Requires external $10\text{ k}\Omega$ pull-up resistors for output logic HIGH in I/O mode.

6.2 IVT (Interrupt Vector Table) High-Yield Vector Map

CoCubes assessments feature direct memory address matching for IVT vectors:

$$\begin{array}{|c|c|c|c|} \hline \mathbf{Interrupt\ Source} & \mathbf{Hardware\ Flag} & \mathbf{IVT\ Vector\ Address} & \mathbf{Priority\ (Default)} \\ \hline \text{System Reset} & \text{RST Pin} & \mathbf{0000H} & \text{Highest} \\ \hline \text{External Interrupt 0 (\text{INT0})} & \text{IE0 (P3.2)} & \mathbf{0003H} & 1 \\ \hline \text{Timer 0 Overflow (\text{TF0})} & \text{TF0} & \mathbf{000BH} & 2 \\ \hline \text{External Interrupt 1 (\text{INT1})} & \text{IE1 (P3.3)} & \mathbf{0013H} & 3 \\ \hline \text{Timer 1 Overflow (\text{TF1})} & \text{TF1} & \mathbf{001BH} & 4 \\ \hline \text{Serial UART (\text{RI}/\text{TI})} & \text{RI or TI} & \mathbf{0023H} & 5 \text{ (Lowest)} \\ \hline \end{array}$$

> **CoCubes Trick Question**: *"If both INT0 and Timer 0 interrupt occur simultaneously, which vector address does the CPU jump to first?"*
> **Answer**: `0003H` (`INT0` has higher natural hardware priority than Timer 0).

6.3 UART & Baud Rate Quick-Solving Rules

- **Crystal Selection**: 11.0592 MHz crystal is specifically chosen because it divides evenly into standard UART baud rates ($9600, 4800, 2400$) with **0% baud rate error**.
- **Timer Mode for UART**: **Timer 1 in Mode 2** (8-bit Auto-Reload).
- **Reload Values for 11.0592 MHz ($\text{SMOD}=0$)**:
- **9600 Baud**: $\text{TH1} = -3 = 253_{10} = \text{0xFD}$
- **4800 Baud**: $\text{TH1} = -6 = 250_{10} = \text{0xFA}$
- **2400 Baud**: $\text{TH1} = -12 = 244_{10} = \text{0xF4}$
- `SMOD` **Bit (PCON.7)**: Setting $\text{SMOD} = 1$ doubles the baud rate (e.g., 9600 $\to$ 19200 bps with $\text{TH1} = \text{0xFD}$).

6.4 ADC Interfacing & Handshaking Signals (ADC0808)

CoCubes tests hardware handshaking pins during analog sensor acquisition:

1. `SOC` **(Start of Conversion)**: Microcontroller sends a HIGH pulse to initiate ADC conversion.

2. `EOC` **(End of Conversion)**: Output pin from ADC. Goes LOW during conversion and transitions HIGH when conversion is complete (connected to MCU interrupt line or polled pin).
3. `OE` **(Output Enable)**: Microcontroller asserts `OE` HIGH to drive digital converted data onto the 8-bit bus.
4. **Resolution Formula**: $\text{LSB} = \frac{V_{\text{ref}}}{2^N}$. For 8-bit ADC0808 with $V_{\text{ref}} = 5\text{V}$, $\text{LSB} = \frac{5}{256} = 19.53\text{ mV}$.