Hardware Engineer Guide

For

SERIAL COMMUNICATION

By Shimi Cohen

Table of Contents

Author	Date	Version	Changes
Shimi Cohen	13/4/2025	1.0	First Draft

INTRODUCTION

Fundamentals

Digital communication transfers data as discrete voltage levels between electronic systems or devices, revolutionizing hardware engineering by providing reliable, noise-resistant data transfer mechanisms. Unlike analog transmission, digital methods use discrete voltage levels.

KEY CONCEPTS

- **A.** Logic levels represent binary states (high/low, 1/0)
- **B.** Signal transitions encode data
- **C.** Synchronization aligns transmitter and receiver

REPRESENTATION TECHNIQUES

- Non-Return to Zero (**NRZ**): Maintains voltage level for bit duration
- Return to Zero (**RZ**): Returns to zero between bits
- Manchester Encoding Combines clock and data
- Differential Manchester: Transition at start, mid-bit transition for 0

SERIAL COMMUNICATION

Serial communication is the dominant method for transferring data between electronic devices. Unlike parallel communication, which sends multiple bits at once across multiple lines, serial communication transmits data **one bit at a time over a single line or pair of lines**, making it more efficient for long distances, simpler board designs, and noise reduction.
It's the backbone of everything from debugging tools to sensor networks and high-speed data.

At its core, serial communication protocols define **how data is structured, synchronized, and verified during transmission**. They differ in speed, complexity, error handling, and physical layers. From simple byte-streaming like UART to more complex protocols like USB or Ethernet, each protocol balances tradeoffs in bandwidth, power, topology, and control. Understanding serial protocols is essential for designing, debugging, and optimizing embedded systems.

Design Consideration

POWER EFFICIENCY

Strategy	Implementation	Savings Potential
Clock Gating	Disable clocks when idle	30-50%
Voltage Scaling	Reduce operating voltage	Quadratic reduction
Sleep Modes	Power down when inactive	90%+ during idle
Low-Power Signaling	Reduced voltage swing	Linear with voltage
Data Compression	Fewer bits transmitted	Proportional to ratio

EMI/EMC CONSIDERATIONS

Electromagnetic compatibility ensures systems operate properly in their intended environment.

EMI Reduction Techniques

Technique	Implementation	Effect
Differential Signaling	Balanced transmission	Common mode rejection
Shielding	Conductive barriers	Direct EM isolation
Filtering	Capacitors, ferrites	Frequency-selective attenuation
Controlled Impedance	Matched traces	Reduced reflections/emissions
Spread Spectrum Clocking	Frequency modulation	Peak emission reduction
Edge Rate Control	Slew rate limiting	High-frequency content reduction

TRANSMISSIONS MODES

Simplex mode: communication occurs in only one direction — like a one-way street — where one device is always the transmitter and the other is always the receiver, with no ability to respond.

Half-Duplex mode: allows two-way communication but only one direction at a time; devices must take turns transmitting and receiving - only one person can speak while the other listens.

Full-Duplex mode: enables simultaneous two-way communication, allowing both devices to send and receive data at the same time, much like a phone call. Full-duplex systems typically require more complex hardware to manage the simultaneous flow of data but offer the fastest and most efficient data exchange.

SIGNAL CHARACTERISTICS

The following terms describe the signal behavior and characteristics:

Parameter	Description	Impact
Rise/Fall Time	Time to transition between states	Affects maximum frequency
Jitter	Timing deviation from ideal	Reduces timing margins
Crosstalk	Unwanted coupling between signals	Creates false transitions
Ringing	Oscillations after transitions	Can cause false triggers
Reflection	Signal bouncing at impedance mismatches	Creates standing waves

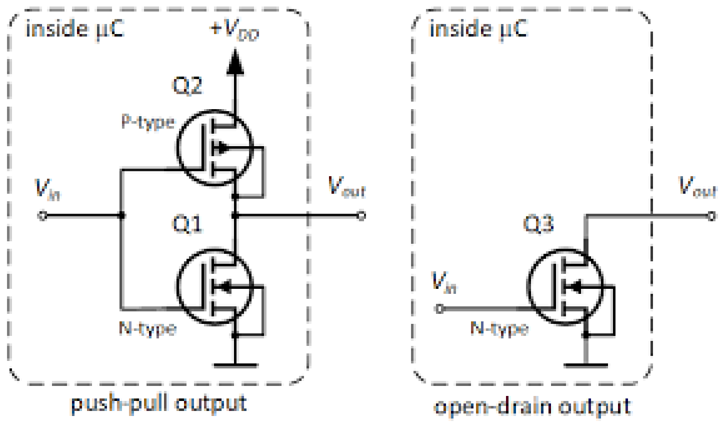
OPEN-DRAIN VS. PUSH-PULL

Push-Pull Configuration

Push-pull outputs use two transistors (NMOS and PMOS) to actively drive the signal high or low.
High-speed transitions between logic levels and low power consumption during idle states.
Ideal for protocols with dedicated lines for transmission (e.g., TX/RX in UART).
Unidirectional communication lines (e.g., SPI, UART), Common in SPI for clock and data lines.

Open-Drain Configuration

Open-drain outputs use a single transistor (NMOS) to pull the signal low, relying on external pull-up resistors to achieve a high state. Slower transitions due to pull-up resistor size and bus capacitance, higher power consumption during active transfers, bidirectional communication lines (e.g., I^{2}C, 1-Wire).
Suitable for multi-device bus systems where multiple devices share the same line.
Common in I^{2}C for SDA/SCL lines and in 1-Wire for data/power sharing.



Clock & Timing

CLOCK RECOVERY METHODS

Technique	Method	Applications
Edge Detection	Timing from signal transitions	NRZ with sufficient transitions
PLL/DLL	Phase/Delay Locked Loop	High-speed serial links
Oversampling	Multiple samples per bit	Microcontroller UART receivers
Embedded Clock	Clock encoded with data	Manchester coding

SYNCHRONIZATION CHALLENGES

Challenge	Description	Mitigation
Frequency Offset	Transmitter/receiver clock mismatch	PLL tracking, periodic resync
Phase Noise	Random timing variations	Filtering, averaging
Jitter Accumulation	Progressive timing errors	Elastic buffers, re-clocking
Initial Acquisition	Finding correct bit boundaries	Training sequences, preambles

FRAMING AND PACKET STRUCTURE

Successful digital communication requires organizing data into recognizable structures.

Component	Purpose	Examples
Preamble	Synchronization	10101010 pattern, SFD
Start Delimiter	Frame boundary identification	UART start bit, USB SYNC
Address Field	Source/destination identification	MAC address, device ID
Control Field	Protocol information	Frame type, sequence numbers
Data Field	Payload information	Application data
Error Check	Integrity verification	CRC, checksum
End Delimiter	Frame termination	UART stop bits, EOF marker

Common Terminology

Baud Rate	Symbols per second
Bit Rate	Data bits transmitted per second
Throughput	Actual data transfer rate
Latency	Delay between Transmit and Receive
Jitter	Variation in periodic timing
Duty Cycle	Ratio of high time to period
SNR	Signal-to-noise ratio
BER	Bit error rate
Clock Polarity (CPOL)	Defines idle state of clock in SPI; affects how data is sampled.
Clock Phase (CPHA)	Determines whether data is sampled on leading or trailing clock edge
Pull-Up Resistor	Essential in I2C and 1-Wire to ensure bus lines return to high state.
Bus Arbitration	Mechanism in I2C for multiple masters to safely share a single bus.
ACK/NACK	Acknowledgement signals in I2C/USB
Start/Stop Condition	Specific signal patterns that mark beginning/end of I2C or 1-Wire
Bit Stuffing	Technique in USB to prevent long runs of identical bits
Handshake	Flow control method to manage data transmission
CRC/Error Checking	Methods used in USB and I2C to detect corrupted data
Addressing Scheme	Unique device identification method (used in I2C and USB).
Open-Drain	Output type that pulls low but leaves line floating high
Line Termination	Resistors/circuitry to match impedance and avoid signal reflections.
Flow Control	Mechanisms to prevent data overrun in UART.
Bus Contention	Error caused when two devices drive the bus simultaneously
Frame Format	Structure of transmitted data packet
Transaction	A complete data exchange
Enumeration	USB process where host detects and assigns address to new device

ARCHITECTURE

Network Configurations

Network topology refers to the physical or logical arrangement of devices within a communication system, significantly impacting performance, reliability, and scalability.

COMMON NETWORK TOPOLOGIES

Topology	Description	Advantages	Disadvantages
Point-to-Point	Direct connection	Simple, fast, private	Limited scalability
Bus	All devices share BUS	Easy to implement	Contention, single POF
Star	Devices connect to hub	Easy fault isolation	Hub failure affects all
Ring	Devices form closed loop	Built-in redundancy	Complex recovery from failures
Mesh	Multiple interconnections	High reliability	Complex, resource-intensive
Tree	Hierarchical arrangement	Scalable	Dependent on root node

PHYSICAL IMPLEMENTATION CONSIDERATIONS

When implementing a specific topology, hardware engineers must consider the following:

* **Transmission Medium**: Copper, fiber, wireless capabilities
* **Maximum Distance**: Cable length limitations based on medium
* **EMI Susceptibility**: Environmental noise considerations
* **Power Requirements**: Particularly for active components
* **Connector Types**: Physical durability and connection reliability
* **Expandability**: Future growth accommodation

IMPEDANCE-MATCHING TECHNIQUES

* Series Termination: Resistor at source end absorbs reflections
* Parallel Termination: Resistor at destination matches line impedance
* AC Termination: RC network provides frequency-dependent termination
* Differential Termination: Resistor between differential pairs

Control Hierarchies

Control hierarchies establish the relationship between communicating devices.

MASTER-SLAVE ARCHITECTURE

In master-slave configurations, a single master controls all communication.

Characteristic	Description
Initiation	Only the master can initiate COMM
Addressing	Master specifies target slave device
Contention	No collisions (master controlled)
Complexity	Simple slave implementation
Examples	I^2C, SPI, 1-Wire

PEER-TO-PEER ARCHITECTURE

Peer-to-peer systems allow any device to initiate communication with others.

Characteristic	Description
Initiation	Any device can initiate communication
Addressing	Direct addressing between peers
Contention	Requires collision detection/avoidance
Complexity	All devices equally complex
Examples	UART, 1-Wire

MULTI-MASTER ARCHITECTURE

Multi-master systems combine elements of both approaches, allowing multiple devices to act as masters.

Characteristic	Description
Initiation	Multiple devices can initiate
Addressing	Masters address target devices
Contention	Requires arbitration mechanism
Complexity	Moderate to high
Examples	I2C, 1-Wire

Synchronous vs. Asynchronous

SYNCHRONOUS COMMUNICATION

Synchronous protocols use a shared clock signal to coordinate data transmission and reception.

* **Clock Distribution**: Either separate clock line or embedded in data
* **Tight Timing**: Reduces overhead but requires precision
* **Examples**: SPI, I^{2}C, JTAG
* **Synchronous Communication Characteristics**

Aspect	Implementation	Benefits
Timing	Clock line or embedded clock	Precise timing control
Speed	Generally higher throughput	Efficient for bulk data
Overhead	Minimal protocol overhead	More efficient bandwidth use
Distance	Usually, shorter maximum distances	Simpler circuitry
Complexity	Clock skew management required	Predictable timing

ASYNCHRONOUS COMMUNICATION

Asynchronous methods operate without a shared clock, using start/stop bits and agreed-upon baud rates.

* **Start/Stop Framing**: Marks beginning and end of data units
* **Clock Recovery**: The Receiver derives timing from data transitions
* **Examples**: UART, RS-232, RS-485
* **Asynchronous Communication Characteristics**

Aspect	Implementation	Benefits
Timing	Self-clocking with start/stop bits	No clock distribution needed
Speed	Generally lower throughput	More robust over distance
Overhead	Higher due to framing bits	Simpler implementation
Distance	Can achieve longer distances	Greater timing flexibility
Complexity	Baud rate matching required	Easier to implement

1-WIRE PROTOCOL

Single -Wire Interface

Developed by Dallas Semiconductor (now **Maxim Integrated**), 1-Wire provides bidirectional communication over a single data line plus ground reference.

1-WIRE PINOUT

Signal	Function	Electrical Characteristics
DQ	Data/Power Line	Open drain with pull-up resistor
GND	Signal Ground	Common reference

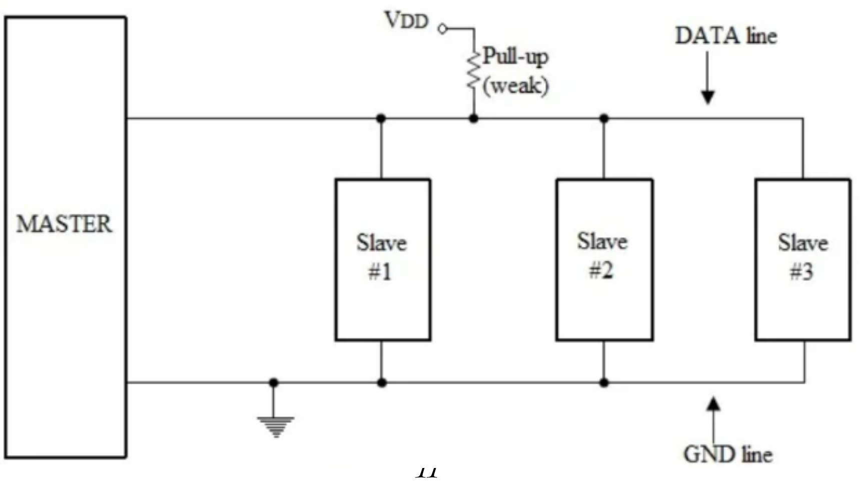
PULL-UP RESISTOR

Bus Length	Standard Speed	Overdrive Speed
<10m	4.7kΩ	2.2kΩ
10-20m	3.3kΩ	Not recommended
20-100m	Active pull-up	Not recommended

***Parasitic Power**: The 1-Wire interface often includes optional weak parasitic power. Master provides strong pull-up (**≤1kΩ**) during specific operations and slaves can draw power from data line during inactive periods.*

LAYOUT TOPOLOGIES

- **Linear Topology**: Devices along single bus with short stubs
- **Star Topology**: Multiple branches from central point (limited)



Timing and Signaling

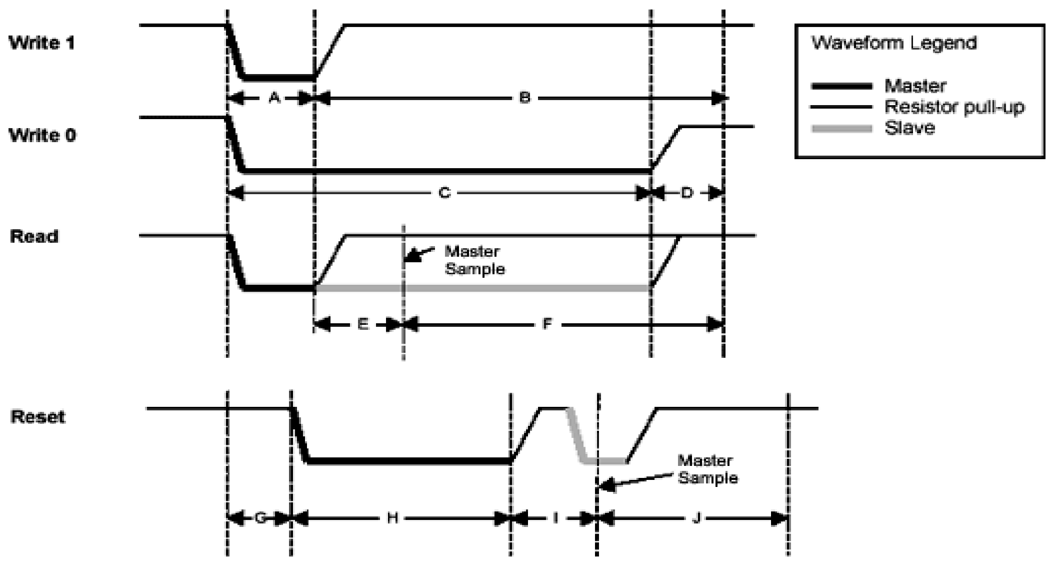
1-Wire uses precisely timed signal transitions to encode data and commands.

SPEED MODES

Parameter	Standard	Overdrive
Bit Rate	15.4 kbps	125 kbps
Reset Recovery	480 µSec	48 µSec
Time Slot	60-120 µSec	6-12 µSec
Sample Window	15 µSec	2 µSec
Max Network Size	100+ devices	10-20 devices

BASIC SIGNALING ELEMENTS

Signal	Master Action	Standard	Overdrive
Reset Pulse	Pull bus low	480-640 µSec	48-80 µSec
Presence Detect	Release bus, sample	60-240 µSec	6-24 µSec
Write 0	Pull low for entire slot	60-120 µSec	6-12 µSec
Write 1	Pull low briefly, release	1-15 µSec low	1-2 µSec low
Read Bit	Pull low briefly, sample	Sample at 15 µSec	Sample at 2 µSec
Recovery Time	Bus idle between slots	1 µSec	1 µSec



Device Addressing

Every 1-Wire device contains a globally unique 64-bit ROM registration number for addressing.

ROM CODE STRUCTURE

Field	Size	Purpose
Family Code	8 bits	Identifies device type
Serial Number	48 bits	Unique device identifier
CRC	8 bits	Error detection for previous 56 bits

ROM COMMAND SET

Command	Code	Function
READ ROM	0x33	Reads 64-bit ROM code (single device only)
MATCH ROM	0x55	Addresses specific device by ROM code
SKIP ROM	0xCC	Addresses all devices simultaneously
SEARCH ROM	0xF0	Algorithm to identify all devices on bus
ALARM SEARCH	0xEC	Identifies devices in alarm state
RESUME	0xA5	Addresses last device selected

SEARCH ALGORITHM IMPLEMENTATION

The Search ROM (0xF0) command uses a binary tree approach:

1. Master issues Search ROM command
2. For each bit position (0-63):
   a. Read bit value
   b. Read complement of bit value
   c. Based on responses, write direction bit
3. Process builds complete ROM codes iteratively
4. CRC verifies correct reading
5. Algorithm continues until all devices are discovered

UART PROTOCOL

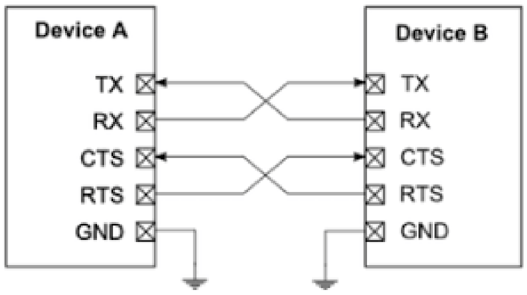
Hardware Implementation

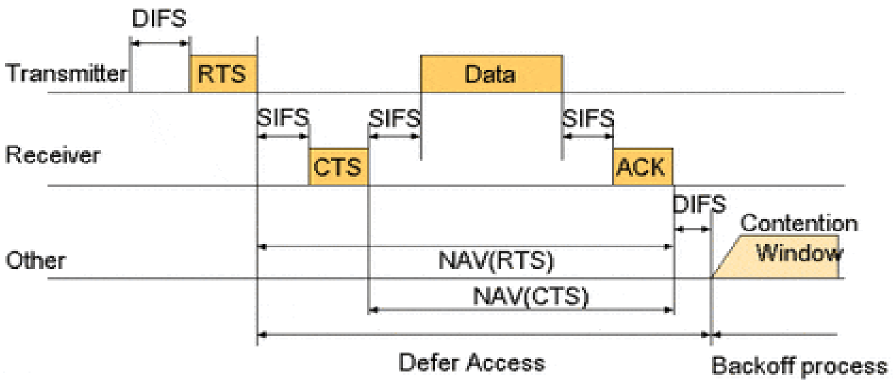
UART (Universal Asynchronous Receiver/Transmitter) is a core serial communication method widely used in embedded systems. While traditionally used for direct device-to-device links, modern systems often embed UART inside other protocols to enhance compatibility or range. A common case is **USB-to-UART bridges** (e.g., FTDI, CP210x), enabling UART over USB. This approach is known as protocol tunneling or encapsulation. UART is also tunneled via **Ethernet** using serial-over-IP.

UART PINOUT

UART uses inverted logic, where the idle line is high (logic 1) and active state is low (logic 0). This setup allows easy break detection and ensures clean synchronization during transmission.
RTS and CTS are optional flow control lines. While TX, RX, and GND are enough for basic UART operation, adding RTS/CTS enhances reliability, especially in high-speed or buffered systems, by managing data flow and preventing overruns.

Signal	Direction	Function
TX	Output	Transmits serial data
RX	Input	Receives serial data
RTS	Output	Request to Send (flow control)
CTS	Input	Clear to Send (flow control)
GND	NA	Signal ground reference





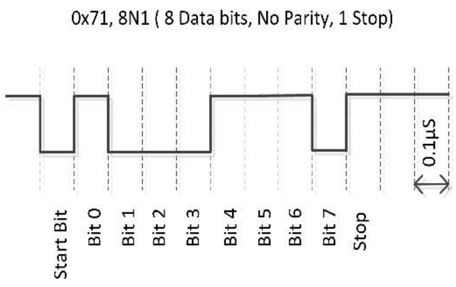
Frame Structure

UART frames encapsulate data with specific control bits to ensure reliable transmission without a clock.

STANDARD UART FRAME

The standard configuration for modern systems is "8N1" (i.e., 8 data bits, no parity, 1 stop bit).

Field	Size	Purpose
Start Bit	1 bit	Always logic 0, marks frame start
Data Bits	8 bits	Payload
Parity Bit	1 bit	Optional error detection
Stop Bits	1 bits	Always logic 1, marks frame end



PARITY BIT (OPTIONAL)

Parity provides basic error detection capabilities.

* **No Parity:** One Symbol is 10Bits (and not 11Bits)
* **Even Parity**: The Total number of 1s must be even
* **Odd Parity**: The Total number of 1s must be odd

TIMING CONSIDERATIONS

Parameter	Calculation	Impact
Bit Time	1 / Baud Rate	Defines timing window
Frame Time	(1 + Data Bits + Parity + Stop Bits) × Bit Time	Total transmission time
Sampling Point	Middle of bit period	Minimizes edge timing effects
Baud Rate Error	Difference between actual and nominal rates	Causes framing errors above ~2%

I²C PROTOCOL

Two -Wire Interface

Inter-Integrated Circuit (I^{2}C) provides a simple, low-speed, two-wire interface.

I²C PINOUT

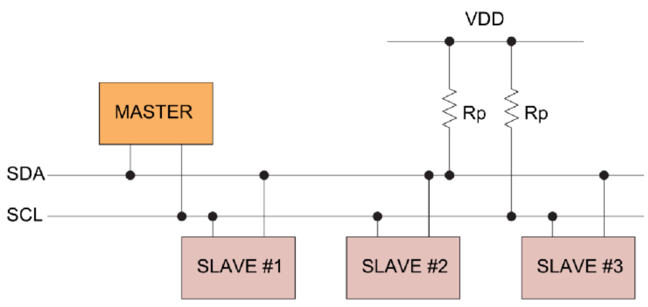
Signal	Function	Electrical Characteristics
SDA	Serial Data Line	Bidirectional, open-drain/open-collector
SCL	Serial Clock Line	Typically controlled by master, open drain
VCC	Supply Voltage	Power reference
GND	Ground	Signal reference

PULL-UP RESISTOR SELECTION

Pull-up resistors must balance between:

- Low enough for fast rise time (RC time constant)
- High enough to limit current when lines are driven low

Bus Capacitance	@100 kHz	@400 kHz	@1 MHz
100 pF	4.7 KΩ	3.3 KΩ	1.0 KΩ
200 pF	3.3 KΩ	2.2 KΩ	620 Ω
300 pF	2.2 KΩ	1.5 KΩ	470 Ω
400 pF	1.8 KΩ	1.2 KΩ	330 Ω



SPEED GRADES AND TIMING

Mode	Max Clock Frequency	Rise Time (max)	Fall Time (max)
Standard Mode	100 kHz	1000 ns	300 ns
Fast Mode	400 kHz	300 ns	300 ns
Fast Mode Plus	1 MHz	120 ns	120 ns
High Speed Mode	3.4 MHz	40 ns	40 ns

Addressing and Arbitration

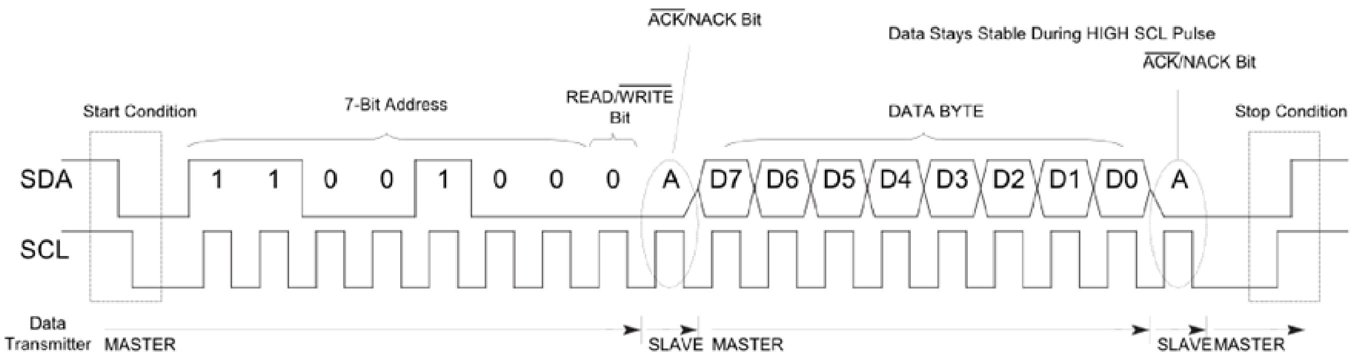
I²C uses a 7-bit addressing scheme (expandable to 10 bits) to support multiple devices on the same bus.

I²C SEQUENCE

Operation	Description	Direction
START Condition	SDA transitions high-to-low while SCL high	Master → All
STOP Condition	SDA transitions low-to-high while SCL high	Master → All
ACK	Receiver pulls SDA low during 9th clock pulse	Receiver → Transmitter
NACK	Receiver leaves SDA high during 9th clock pulse	Receiver → Transmitter
Address Frame	7-bit address + 1 R/W bit + 1 ACK/NACK	Master → Slave
Data Frame	8 data bits + 1 ACK/NACK	Either direction

I²C PACKET

The packet of standard I2C communication:



MULTI-MASTER ARBITRATION

When multiple masters attempt simultaneous transmission:
(This collision detection occurs without corruption of data on the bus)

* All masters monitor SDA while transmitting
* If the expected level differs from the actual level, arbitration lost
* SDA acts as wired-AND (low wins)
* Losing master switches to slave mode
* Winning master continues undisturbed

Advanced Protocol Features

I^{2}C includes several advanced features for complex system requirements.

CLOCK STRETCHING

Allows slower slaves to control communication pace:

* Slave holds SCL low after receiving or sending byte
* Master detects SCL held low
* Master waits for SCL to release
* Communication resumes when slave releases SCL

10-BIT ADDRESSING

Extends address space to support more devices:

* First byte: 11110XX + R/W bit (XX = top 2 bits of address)
* Second byte: Remaining 8 bits of address
* For reads, requires repeated START condition

GENERAL CALL

Broadcast commands to all devices:

* Address byte is 0x00 followed by R/W=0
* Second byte contains command code
* Devices that support general call will ACK

DEVICE ID

Allows enumeration of devices on bus:

* Special address 0x7C followed by R/W=1
* Device responds with 3-byte ID sequence
* Useful for dynamic configuration

SPI PROTOCL

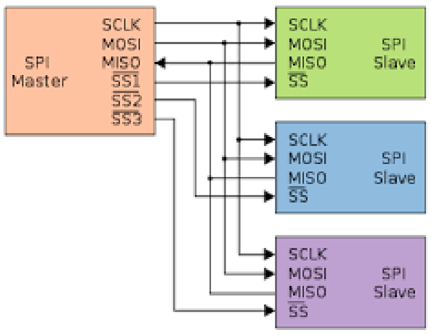
Four -Wire Interface

Serial Peripheral Interface (SPI) provides high-speed, full-duplex communication between a master and multiple slave devices.

SPI PINOUT

Unlike I^{2}C, SPI uses push-pull drivers for all signals, enabling faster speeds without pull-up resistors

Signal	Alternative Names	Function
MOSI	SDI, DI, SI	Master Out Slave In
MISO	SDO, DO, SO	Master In Slave Out
SCLK	SCK, CLK	Serial Clock
CS/SS	NSS, CE	Chip Select/Slave Select
GND	NA	Signal ground reference



ELECTRICAL CHARACTERISTICS

Supply Voltage	Logic Low	Logic High	Rise/Fall Times
3.3V	0-0.8V	2.0-3.3V	<10ns typical
5.0V	0-1.0V	3.5-5.0V	<10ns typical

ADVANTAGES OVER OTHER PROTOCOLS

Amongst the Serial protocols, SPI is usually preferable, due to the following attributes:

* **Full-Duplex**: Simultaneous bidirectional data transfer
* **No Addressing Overhead**: Dedicated CS lines select devices
* **Higher Speed**: Multi-MHz to >50MHz operation
* **Flexible Packet Size**: No fixed word length
* **Simple Hardware**: No complex state machines are required
* **Low Latency**: Minimal protocol overhead

SPI Configurations

SPI supports multiple slave devices using different wiring configurations.

INDEPENDENT SLAVE SELECT

Most common configuration with dedicated CS line per slave.

On one hand: Simple implementation, independent device operation, No software overhead.

But on the other: More complex PCB routing, Limited scalability and uses more I/O pins

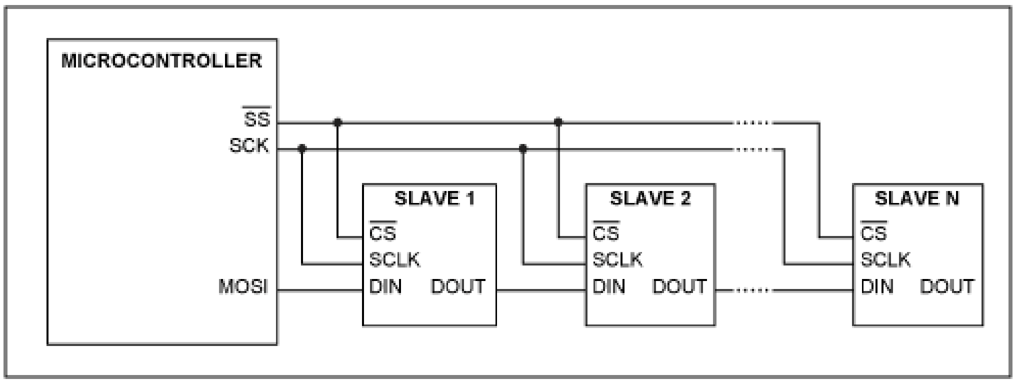
DAISY-CHAIN CONFIGURATION

Advantages: Uses only one CS line, Simplified routing, Good for distribution systems

Disadvantages: Increased latency, Complex data handling, all devices must use same protocol

Devices connected in series with data propagating through chain as followed:

- MOSI connects to first device's data input
- Each device's data output connects to next device's input
- Last device's output connects to master's MISO
- Single CS line activates all devices simultaneously



QUAD SPI

Quad SPI (QSPI) is an enhanced version of SPI that boosts bandwidth by using four data lines (IO0–IO3) instead of just one (MOSI). While standard SPI transmits 1 bit per clock, QSPI can transmit 4 bits per clock cycle, effectively quadrupling the data rate without increasing the clock speed. This makes it ideal for high-speed memory access, like reading from NOR flash in embedded systems. QSPI maintains the same clock and chip select lines as SPI but shifts into higher bit modes after an initial command phase, offering backward compatibility with standard SPI devices.

CLOCK POLARITY AND PHASE

SPI offers four operating modes based on clock polarity (CPOL) and phase (CPHA) settings.

Mode	CPOL	CPHA	Clock Idle State	Data Capture Edge
0	0	0	Low	Rising
1	0	1	Low	Falling
2	1	0	High	Falling
3	1	1	High	Rising

Mode 0 (CPOL=0, CPHA=0):

* Clock idles low
* Data sampled on rising edge
* Data changes on falling edge

Mode 1 (CPOL=0, CPHA=1):

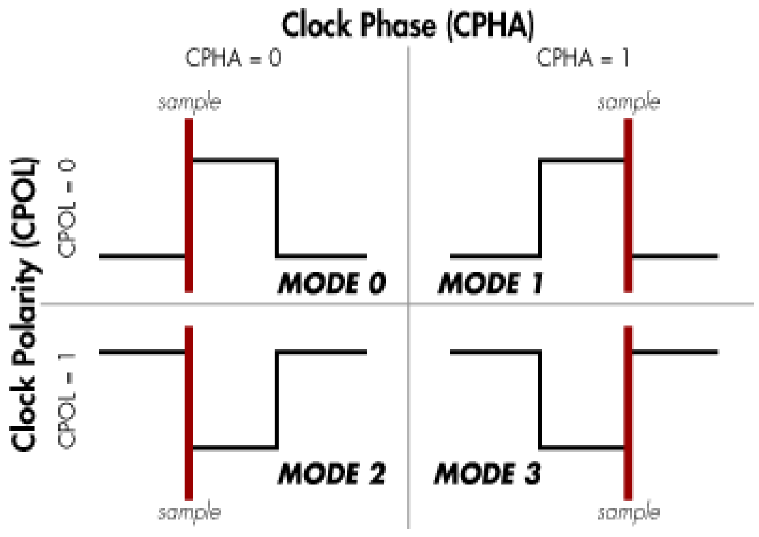
* Clock idles low
* Data sampled on falling edge
* Data changes on rising edge

Mode 2 (CPOL=1, CPHA=0):

* Clock idles high
* Data sampled on falling edge
* Data changes on rising edge

Mode 3 (CPOL=1, CPHA=1):

* Clock idles high
* Data sampled on rising edge
* Data changes on falling edge



DEVICE COMPATIBILITY CONSIDERATIONS

When interfacing different SPI devices:

- Master must reconfigure for each device if modes differ
- Typically handled through software control of CPOL/CPHA bits
- Some controllers support changing mode between CS assertions
- Devices with same mode can share clock and data lines

USB2.0 PROTOCOL

Physical Layer and Speed Grades

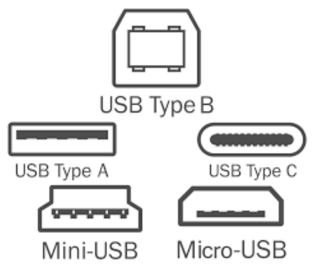
Universal Serial Bus (USB) provides a standardized interface for connecting peripherals to host systems.

USB PINOUT

Signal	Function	Description
VBUS	Power	5V supply from host
D+	Data Plus	Differential data signal
D-	Data Minus	Differential data signal
ID	Mode Detect)OTG)	Single Ended
GND	Ground	Signal reference

USB CONNECTOR TYPES

Connector	Usage	Max Current	Compatible
Type-A	Host ports	500mA-900mA	USB 1.0-3.2
Type-B	Peripheral ports	N/A	USB 1.0-3.2
Mini-B	Smaller peripherals	500mA	USB 1.0-2.0
Micro-B	Mobile devices	500mA-900mA	USB 1.0-3.0
Type-C	Newer devices	Up to 5A	USB 2.0-4.0



USB SPEED GRADES

Speed Grade	Data Rate	Signaling	Cable Requirements
Low Speed	1.5 Mbps	Single-ended	2 twisted pairs
Full Speed	12 Mbps	Single-ended	2 twisted pairs
High Speed	480 Mbps	Differential	2 twisted pairs, shielded

Enumeration Process

USB enumeration allows the host to identify and configure connected devices.

ENUMERATION STEPS

1. **Device Connection**: Host detects voltage change on data lines
2. **Reset**: Host sends reset signal (SE0 for >10ms)
3. **Speed Detection**: Device pulls up D+ (full speed) or D- (low speed)
4. **Default Address**: Device assigned address 0 initially
5. **Descriptor Request**: Host requests device descriptor
6. **Address Assignment**: Host assigns unique address
7. **Configuration**: Host selects device configuration
8. **Driver Loading**: OS loads appropriate driver

USB DESCRIPTORS HIERARCHY

Descriptor	Purpose	Key Information
Device	Basic device info	Class, VID, PID, versions
Configuration	Power requirements	Max power, bus/self-powered
Interface	Functional group	Class, subclass, protocol
Endpoint	Communications channel	Direction, type, max packet size
String	Human-readable info	Manufacturer, product names
HID	Human interface data	Report format
BOS	Platform capabilities	USB versions, device capabilities

DEVICE CLASSES

Class	Code	Typical Devices
HID	0x03	Keyboards, mice, gamepads
CDC	0x02/0x0A	Modems, serial adapters
Mass Storage	0x08	Flash drives, external drives
Audio	0x01	Headsets, speakers
Video	0x0E	Webcams, capture devices
Printer	0x07	Printers
Hub	0x09	USB hubs

Transfer Types

USB supports four transfer types for different communication requirements.

CONTROL TRANSFERS

Used for command and status operations:

- Mandatory for all devices
- Structured as SETUP + optional DATA + STATUS phases
- Always uses endpoint 0
- Used for enumeration and standard requests

BULK TRANSFERS

Designed for large, non-time-critical data:

- Variable packet size (8-1024 bytes depending on speed)
- Error detection and retransmission
- No bandwidth guarantee
- Common for storage devices, printers

INTERRUPT TRANSFERS

Provides guaranteed latency for periodic small transfers:

- Fixed maximum latency (1-255ms)
- Guaranteed bandwidth
- Error detection and retry
- Common for HID devices (keyboards, mice)

ISOCHRONOUS TRANSFERS

For time-critical data streams:

- Guaranteed bandwidth
- No error correction (data loss accepted)
- Fixed packet size per frame
- Common for audio/video streaming