MICROPROCESSOR ENGINEERING

Mastering 8086 Microprocessor Serial Communication: A Comprehensive Technical Guide to Interfacing, Protocols, and Implementation

Published: January 7, 2026 | Index ID: #997

The **8086 microprocessor**, introduced by Intel in 1978, remains a cornerstone of computer architecture education and legacy industrial control systems. As the first 16-bit processor in the x86 lineage, it revolutionized computing by offering a significant leap in data processing capability. However, the internal 16-bit parallel architecture of the 8086 poses a unique challenge when communicating with external devices over long distances or with standardized peripherals like Personal Computers (PCs). This necessitated the development and implementation of **serial communication** protocols and specialized interface hardware.

The Fundamental Paradigm: Serial vs. Parallel Communication

In the context of the 8086, data is processed internally in parallel format across a 16-bit data bus. While **parallel communication** allows for high-speed data transfer by sending multiple bits simultaneously, it is limited by physical constraints. As the distance between the sender and receiver increases, parallel cables become bulky, expensive, and highly susceptible to **crosstalk** and **clock skew**.

Serial communicationaddresses these limitations by transmitting data one bit at a time over a single signal path. This method reduces the number of required conductors, minimizes electromagnetic interference (EMI), and allows for long-distance communication. For the 8086 to interact with a PC or a serial terminal, the parallel data from the 8086's internal bus must be converted into a serial stream and vice versa.

Technical Comparison of Communication Schemes

Feature	Parallel Communication	Serial Communication
Transmission Path	Multiple wires (e.g., 8, 16, or 32 bits)	Single wire (or differential pair)
Distance	Short range (typically < 5 meters)	Long range (up to 1200 meters with RS-485)
Complexity	Higher (requires synchronization across all lines)	Lower (requires timing synchronization only)
Cost	High (due to cable thickness and connectors)	Low (simple twisted pair or single wire)
Common 8086 Interface	8255 PPI (Programmable Peripheral Interface)	8251 USART (Universal Synchronous/ Asynchronous Receiver/Transmitter)

Architectural Requirements for 8086 Serial Interfacing

The 8086 microprocessor does not possess an integrated on-chip UART (Universal Asynchronous Receiver/ Transmitter). To achieve serial communication, engineers typically employ an external peripheral chip, most notably the **Intel 8251A USART**. This programmable chip acts as the bridge between the 8086's parallel bus and the serial communication line.

The Role of the 8251 USART

The 8251 USART is specifically designed to handle both **synchronous** and **asynchronous**data transmission. In the asynchronous mode commonly used for PC interfacing, the chip performs several critical functions:

- Parallel-to-Serial Conversion: It accepts 8-bit data from the 8086 and shifts it out bit-by-bit.
- Serial-to-Parallel Conversion: It receives incoming serial bits and assembles them into a byte for the 8086 to read.
- Framing: It automatically adds Start bits, Stop bits, and Parity bits to the data stream.
- Baud Rate Control: It synchronizes data transfer according to a predefined clock frequency.

Interfacing the 8251 with the 8086

Connecting the 8251 to the 8086 requires careful mapping of the address and data buses. The 8251 uses a **C/D (Control/Data)**pin to distinguish between command words and actual data. Typically, this pin is tied to one of the 8086's address lines (often A0 or A1). When A1 is low, the CPU accesses the data buffer; when A1 is high, it accesses the control/status register.

Physical Layer Standards: RS-232, RS-422, and RS-485

The logic levels generated by the 8086 and 8251 are standard **TTL (Transistor-Transistor Logic)**, where 0V represents logic '0' and 5V represents logic '1'. However, these levels are not robust enough for long-distance transmission or communication with a standard PC serial port. Therefore, a physical layer standard must be employed.

1. RS-232C Standard

RS-232 is the most common standard for serial communication between a microprocessor and a PC. It uses bipolar signaling: logic '1' is represented by a voltage between -3V and -15V, while logic '0' is represented by a

voltage between +3V and +15V. To interface the 8251 (TTL) with a PC (RS-232), a **level shifter** such as the **MAX232**IC is mandatory. The MAX232 uses a charge pump to generate the required positive and negative voltages from a single 5V supply.

2. RS-422 and RS-485 Standards

For industrial environments where noise immunity is paramount, RS-422 or RS-485 are preferred. Unlike RS-232, which is single-ended (referenced to ground), these standards use **differential signaling**. This means the receiver looks at the voltage difference between two wires, effectively cancelling out common-mode noise.

The Asynchronous Data Frame Mechanism

In asynchronous serial communication, there is no shared clock signal between the 8086 system and the PC. Instead, both devices must agree on a **Baud Rate** (bits per second) and a **Frame Format**. A typical data frame consists of:

1. Start Bit: A single logic '0' bit that alerts the receiver that a data byte is arriving.
2. Data Bits: Usually 7 or 8 bits representing the character (ASCII).
3. Parity Bit: An optional bit used for basic error detection (Even, Odd, or None).
4. Stop Bits: One or two logic '1' bits that return the line to an idle state.

Mathematical Calculation of Baud Rate

The baud rate is determined by the frequency of the clock supplied to the 8251 chip (TxC and RxC pins). The 8251 allows for clock prescaling (Baud Rate Factors) of 1x, 16x, or 64x. The formula is:

Baud Rate = Clock Frequency / (Baud Rate Factor × Divisor)

For example, to achieve a baud rate of 9600 with a 16x factor, the input clock must be 153.6 kHz.

Software Implementation: Assembly Language Programming

Programming serial communication on the 8086 involves initializing the 8251 USART by sending a sequence of control words to its internal registers. This process is crucial because the 8251 starts in a non-deterministic state after reset.

Initialization Sequence

The initialization usually follows this pattern:

- Mode Instruction: Defines the baud rate factor, character length, parity, and stop bits.
- Command Instruction: Enables the transmitter (TxEN) and receiver (RxEN), and resets error flags.

Sample Logic Flow (Assembly)

The following logic outlines how to transmit a character from the 8086 to a PC:

```
MOV AL, 0B6H ; Control word to initialize 8251
OUT CTRL_PORT, AL

CHECK_READY:
IN AL, STATUS_PORT ; Read 8251 status
AND AL, 01H ; Check if TxRDY (Transmitter Ready) bit is set
JZ CHECK_READY ; Loop if not ready
```

```
MOV AL, 'A' ; Load character to be sent
OUT DATA_PORT, AL ; Output character to 8251Case Study: 8086 Interfacing via MTS-86 Trainer Kit
```

Many educational environments use the **MTS-86 Training System**. This kit provides an onboard 8251 USART and an RS-232 port. In a typical lab scenario, the 8086 is programmed to send a string (e.g., "HELLO WORLD") to a PC running a terminal emulator like PuTTY or HyperTerminal.

Common Failure Modes and Solutions

Symptom	Probable Cause	Technical Solution
Garbage Characters	Baud rate mismatch or incorrect parity.	Verify the clock frequency and 8251 mode register settings match the PC terminal.
No Data Received	TX/RX lines swapped (Null Modem issue).	Ensure Pin 2 (RD) and Pin 3 (TD) are correctly crossed if not using a straight-through cable.
Framing Error	Stop bit length mismatch.	Ensure both systems are set to the same number of stop bits (usually 1).
Overrun Error	8086 is not reading data fast enough.	Implement Interrupt-Driven I/O instead of polling to ensure timely data retrieval.

Advanced Interfacing: The 8255 PPI in Serial Context

While the 8255 Programmable Peripheral Interface is primarily a parallel device, it can be used for "bit-banging" serial data. This involves using software loops to manually toggle a single pin of a 8255 port (e.g., Port A, Bit 0) to mimic a serial protocol. While this consumes significant CPU cycles, it demonstrates the versatility of the 8086 I/O architecture in environments where a dedicated USART is unavailable.

Hardware Handshaking (Flow Control)

To prevent data loss when the PC or the 8086 is too busy to process incoming bytes, **Hardware Handshaking**is utilized. This involves additional signals:

- RTS (Request to Send): The 8086 signals it wants to send data.
- CTS (Clear to Send): The PC signals it is ready to receive.
- DTR (Data Terminal Ready) and DSR (Data Set Ready): Used to indicate that both devices are powered on and connected.

Practical Implementation Field Guide

When implementing 8086-to-PC serial communication in a modern context, follow these steps:

Step 1: Hardware Verification

Verify that the 8086 system clock is stable. If using a 5MHz 8086, ensure the peripheral clock for the 8251 is derived correctly to provide standard baud rates. Check the MAX232 voltages; you should see roughly -9V on the RS-232 TX pin when idle.

Step 2: Command Word Configuration

The first byte sent to the 8251 after a hardware reset must be the Mode Instruction. If you send a Command Instruction first, the chip will misinterpret it. A common trick is to send three 0x00 bytes followed by a 0x40 (Internal Reset) to ensure the 8251 is in a known state before configuration.

Step 3: PC-Side Setup

Use a USB-to-RS232 adapter if your PC lacks a native COM port. In your terminal software, set the parameters to **9600 Baud, 8 Data Bits, No Parity, 1 Stop Bit**. This is the industry-standard "9600-8-N-1" configuration.

Broad Implications and Technological Synthesis

Understanding serial communication in the 8086 environment provides a deep look into the evolution of I/O processing. The transition from the **8086** (16-bit) to the **8051**(8-bit microcontroller) saw the integration of UARTs directly onto the silicon, a trend that continues in modern ARM and RISC-V architectures. However, the fundamental concepts—start/stop bits, baud rates, and level shifting—remain unchanged.

The study of the 8086 and its interaction with the 8251 USART highlights the necessity of **abstraction layers**in engineering. The 8086 handles high-level logic, while the 8251 manages the timing-sensitive serial bitstream. This separation of concerns is a principle that governs modern networking and peripheral interfacing today. Whether using a 40-year-old microprocessor or a modern SoC, the principles of reliable, asynchronous data transfer established during the 8086 era continue to underpin the global infrastructure of digital communication.

Baca Juga (Artikel Terkait)

- **Mastering the 8085 Microprocessor: A Comprehensive Technical Guide to Architecture, Instruction Sets, and Interrupt Systems**

  URL: http://123caramba.com/8085-microprocessor-architecture-instruction-set-technical-guide.pdf

Tags: #8086 Microprocessor #Serial Communication #8251 USART #RS 232 #Embedded Systems