**🧮 What is a 7-Segment Display?**

A **7-segment display** is an electronic display device used to display **decimal numbers** (0–9) and some **hexadecimal characters (A–F)**. It consists of **7 LEDs** (segments) arranged in a specific pattern to form numbers or letters when lit.

**🔹 Structure:**

* Segments are labeled as **a, b, c, d, e, f, g**
* Some displays also include an 8th segment (dot) called **dp** (decimal point)

**🔹 Types:**

* **Common Cathode (CC):** All cathodes are connected to GND; segments light up when HIGH is applied.
* **Common Anode (CA):** All anodes are connected to VCC; segments light up when LOW is applied.

**🔹 How It Works:**

To display a digit like "2", you turn ON specific segments (a, b, g, e, d). This is done by applying signals (HIGH/LOW) to the segment pins through a **microcontroller or driver circuit**.

**⚙️ What is an 8-bit Shift Register?**

A **shift register** is a sequential logic circuit used to **store and shift binary data**. An **8-bit shift register** stores **8 bits** of data and can shift them left or right.

**🔹 Types:**

* **Serial-In Serial-Out (SISO)**
* **Serial-In Parallel-Out (SIPO)**
* **Parallel-In Serial-Out (PISO)**
* **Parallel-In Parallel-Out (PIPO)**

The most commonly used in projects is **SIPO**, like the **74HC595** IC.

**🔹 How It Works:**

* **Clock Pulse** is used to shift bits.
* **Serial Input (DS or SER)**: You send 1 bit at a time.
* **Clock (SH\_CP)**: On each rising edge, data shifts into the register.
* **Latch (ST\_CP)**: Transfers data to the output pins.
* After 8 clock pulses, all 8 bits are loaded.

**🔹 Use Cases:**

* Controlling LEDs
* Expanding I/O pins of microcontrollers
* Display driving (e.g., 7-segment displays!)

**🧮 7-Segment Display – How It Works**

**📌 Imagine: You want to display the number “3” on a 7-segment display.**

**🔹 Step-by-Step:**

1. **Segments Needed for "3"**:  
   You need to light up these segments: **a, b, c, d, g**
2. **Powering the Segments**:
   * Each segment is an LED.
   * You turn ON a segment by applying voltage (current flows through it).
   * If it's **Common Cathode**: Connect the **common pin to GND** and apply **HIGH** (1) to segments a-g that you want to light.
   * If it's **Common Anode**: Connect the **common pin to VCC** and apply **LOW** (0) to segments you want to light.
3. **Driving It**:
   * You can connect each segment to a **microcontroller pin** (like Arduino, 8051, etc.).
   * Or you can use a **shift register** to reduce pin usage (more on that next).

**⚙️ 8-bit Shift Register (like 74HC595) – How It Works**

Imagine your microcontroller only has a few I/O pins, but you need to control 8 LEDs (or 7-segment display segments). A shift register helps with that!

**🔹 Key Pins of 74HC595:**

* **SER (Serial In)** – Send bits one by one.
* **SRCLK (Shift Clock)** – Moves each bit into internal memory.
* **RCLK (Latch Clock)** – Pushes the 8-bit data to the output pins.
* **Q0–Q7** – The 8 output pins.

**🔹 Step-by-Step Working:**

1. **Initialize All Pins LOW**
2. **Send Bits** (e.g., 10110011) – one bit at a time to SER:
   * On each bit, pulse the **SRCLK** (rising edge).
   * Internally, the bits get "shifted in" like a line of people.
3. **Latch the Output**:
   * Once 8 bits are sent, pulse the **RCLK**.
   * Now the 8 output pins (Q0–Q7) reflect the data.
4. **Connect Outputs to Something**:
   * You connect these Q0–Q7 pins to the segments of a **7-segment display**.

**🧠 What is ATtiny85?**

The **ATtiny85** is an 8-bit **microcontroller** made by **Atmel (now Microchip)**. It's part of the AVR family (same as Arduino Uno's ATmega328P), but smaller and with fewer pins.

**🔹 Key Features:**

| **Feature** | **Value** |
| --- | --- |
| CPU | 8-bit AVR |
| Flash Memory | 8 KB (for your program) |
| SRAM | 512 Bytes (temporary variables) |
| EEPROM | 512 Bytes (permanent storage) |
| I/O Pins | 6 (shared with analog, PWM, etc.) |
| Timers | 2 (one 8-bit, one 16-bit) |
| ADC | 10-bit, 4 channels |
| Operating Voltage | 2.7V–5.5V |
| Clock Speed | Up to 20 MHz (usually runs at 8 MHz by default) |

**⚙️ How Does ATtiny85 Work?**

Think of it like a tiny brain:

* You **write a program** (e.g., blink an LED, read a sensor, etc.)
* You **upload it** to the ATtiny85
* It **runs that program** continuously until it’s powered off

**🔹 Programming It:**

1. **Write Code** in Arduino IDE or PlatformIO  
   (Yes! You can use Arduino-style code with ATtiny85)
2. **Upload It** using:
   * An **Arduino Uno** as ISP programmer
   * Or a **USB programmer** like USBasp or TinyUSB
3. **Power It** with 3V or 5V (battery, USB, or regulator)
4. **Use I/O Pins** to connect LEDs, sensors, buttons, etc.

🧠 Pins PB0–PB5 can be used as:

* Digital I/O
* Analog inputs
* PWM output
* I2C/SPI (limited support)
* Timer/Counter inputs

**🧪 Example Use Cases:**

* Mini temperature sensor
* Tiny blinking LED light
* Digital thermometer
* Infrared remote receiver
* Small automation devices
* Driving a 7-segment display using fewer components

**🔁 Comparison to Arduino Uno:**

| **Feature** | **ATtiny85** | **Arduino Uno (ATmega328P)** |
| --- | --- | --- |
| Size | Very Small | Larger |
| Pins | 6 | 20 |
| Flash | 8 KB | 32 KB |
| Cost | ~$1 | ~$5–10 |
| USB | ❌ (need external programmer) | ✅ |

Let me know if you want a **wiring diagram**, **code example**, or **how to burn the bootloader** — I’d be happy to help you build something with it!

4o

**Connection :**

ATTINY 85

* Power to Power of 8 bit
* Pb2 to shift register clock
* Pb1 to 330 ohm to common of 7 segment display
* Pb0 to terminal 2a of push button
* Ground to ground
* Pb4 input of 8 bit shift register
* Pb3 to output register clock of shift register

8bit Shift Register

* Output1 to A of 7segment display
* Output enable to ground
* Shift reg clear to positive
* Ground to ground
* Output7 to G of 7 seg display
* Output 6 to F of 7 seg display
* Output 5 to E of 7 seg display
* Output 4 to D of 7 seg display
* Output 3 to C of 7 seg display
* Output 2 to B of 7 seg display

Push button

* 1B to positive
* 2B to 10kiloohm to negative

Breadboard

* Pos to Pos
* Neg to Neg

What is happening?