

1. **POWER SUPPLY pins**

* **VSS** = Ground
* **VCC** = +5V
* **VEE**  = control contrast using potentiomter: 0 🡪 5V
* **LEDA (Anode)** = +5V, to power the backlight
* **LEDK (Cathode) =** GND, to power the backlight

1. **DATA pins**

The **16x2 LCD** can receive 8-bit data or ***commands*** through its data pins **(D0 to D7)**. It operates in two modes: **8-bit mode** and **4-bit mode**, which are initialized and selected through specific LCD commands (instructions):

1. **8-bit mode**

* All 8 data pins **(D0 to D7)** are used.
* Each character (or command) is sent as a full 8-bit binary value (or byte) in one go.

1. **4-bit mode**

* Only the upper 4 data pins **(D4 🡪 D7)** are used **( D0** 🡪 **D3** not used**).**
* Data is sent in **two 4-bit chunks** instead of a full 8-bit byte at once.
* This mode is more efficient in terms of GPIO pin usage, as it reduces the number of pins required from the microcontroller.
* First, the higher nibble (4 most significant bits) of the byte is sent, followed by the lower nibble (4 least significant bits).

1. **CONTROL pins**
2. **RW (Read/Write) pin**

* When **RW = 0**, the LCD is in **write mode**, meaning the ***microcontroller can send data*** or commands to the LCD.
* When **RW = 1**, the LCD is in **read mode**, meaning the microcontroller can read data from LCD.

+ **RS = 0**: check the the **busy flag** on **pin D7** after pin **Enale pulsed**.

+ **RS = 1**: reading display data (from DDRAM or CGRAM), which includes the **actual character** data at the **current cursor position** after **pin Enable pulsed**.

1. **RS (Register Select) pin**

* When **RS = 0**, the input to the LCD is treated as a **command** (for example, clearing the display or moving the cursor).
* When **RS = 1**, the input is treated as **data**, which is the **actual character** (ASCII) to be displayed on the screen.

1. **E (Enable) pin**

* A high-to-low transition **(falling edge)** on this pin tells the LCD to **process the data** or **send command** presented on the data bus **(D0–D7)**.
* Tell the LCD that the data or command is ready for reading/writing.

1. **COMMAND**

* An LCD command is an instruction sent to the LCD to perform specific tasks such as clearing the screen, moving the cursor, turning the display on or off, ...
* These commands are sent when RS (Register Select) is set to 0 (command mode).
* When the power is initially turned on, the LCD needs to wait for a period of approximately 15 ms to stabilize its operation. After that, you can start the initialization process to put the LCD.
* When powered up, the LCD starts in 8-bit mode, meaning it expects 8-bit commands sent on the D0–D7 pins.
* After each command to LCD need some time to process (few milliseconds), so we need to check if the LCD is busy

**Sending 4-bit Commands after Power-Up**

* Immediately after power-up, the LCD is designed to accept a few specific 4-bit "half commands" (like 0x33 and 0x32) to initialize itself. This is a temporary mechanism to allow you to transition to 4-bit mode without needing to use all 8 data lines.
* However, this is a temporary state, and if you do not properly complete the initialization sequence (i.e., you don't send the commands to switch to 4-bit mode), the LCD stays in 8-bit mode and expects 8-bit commands. If you keep sending only 4 bits, the LCD won’t understand them, because it’s still waiting for the full 8-bit commands.
* This means that after the initialization window, if you don't properly configure it to switch to 4-bit mode, the LCD will become unreliable and may not respond correctly and will misinterpret further 4-bit commands as incomplete 8-bit commands.

|  |  |  |
| --- | --- | --- |
| **Command (HEX)** | **Commane Name** | **Description** |
| **0x01** | Clear Display | Clears the display and sets cursor to the home position (top-left). |
| **0x02** | Return Home | Moves the cursor to the home position without clearing the display. |
| **0x04** | Decrement Cursor | Sets the cursor to move left after each character is written. |
| **0x06** | Increment Cursor | Sets the cursor to move right after each character is written (default). |
| **0x08** | Display Off | Turns off the display (cursor and display both off). |
| **0x0C** | Display On, Cursor Off | Turns on the display, but hides the cursor. |
| **0x0E** | Display On, Cursor On | Turns on the display and shows the cursor. |
| **0x0F** | Display On, Cursor Blinking | Turns on the display, shows the cursor, and makes it blink. |
| **0x10** | Shift Cursor Left | Moves the cursor left by one position. |
| **0x14** | Shift Cursor Right | Moves the cursor right by one position. |
| **0x18** | Shift Entire Display Left | Shifts the entire display to the left. |
| **0x1C** | Shift Entire Display Right | Shifts the entire display to the right. |
| **0x28** | 4-bit, 2-line mode | Initializes the LCD in 4-bit mode, with two lines (for 16x2 LCDs). |
| **0x38** | 8-bit, 2-line mode | Initializes the LCD in 8-bit mode, with two lines (for 16x2 LCDs). |
| **0x80** | Set Cursor Position | Sets the cursor position (combine with address for specific position). |
| **0xC0** | Set Cursor to Line 2 | Moves the cursor to the beginning of the second line. |

1. **DISPLAY CHRACTER**

* A 16x2 LCD has 32 positions for characters: 16 positions in the first row && 16 positions in the second row.
* **Each space (or position)** on the 16x2 LCD ***has a specific address in the DDRAM*** (Display Data RAM), and each DDRAM address corresponds to a specific position on the display
* **First row addresses:**  Start from **0x80 🡪 0x8F.**
* **Second row addresses:** Start from **0xC0 🡪 0xCF.**
* When you send data (a character) to a specific DDRAM address, the character appears on the display at that position.
* When ***no visible character is displayed*** , the DDRAM typically holds the **value 0x20,** which corresponds to a space (blank character).

*\*\* you can also store custom characters in DDRAM by defining them in the CGRAM (Character Generator RAM), but this is more advanced. For standard operations, DDRAM holds the ASCII values of the characters to be displayed.*