Character Code (DDRAM Data)									CGRAM Address						Character Patterns (CGRAM Data)							
b8	b7	b6	b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	-	-	-	1	1	1	1	1
						0	0	0				0	0	1				0	0	1	0	0
						0	0	0				0	1	0				0	0	1	0	0
						0	0	0				0	1	1				0	0	1	0	0
						0	0	0				1	0	0				0	0	1	0	0
						0	0	0				1	0	1				0	0	1	0	0
						0	0	0				1	1	0				0	0	1	0	0
						0	0	0				1	1	1				0	0	0	0	0
0	0	0	0	0	-	0	0	1	0	0	1	0	0	0	-	-	-	1	1	1	1	0
						0	0	1				0	0	1				1	0	0	0	1
						0	0	1				0	1	0				1	0	0	0	1
						0	0	1				0	1	1				1	1	1	1	0
						0	0	1				1	0	0				1	0	1	0	0
						0	0	1				1	0	1				1	0	0	1	0
						0	0	1				1	1	0				1	0	0	0	1
						0	0	1				1	1	1				0	0	0	0	0

Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character patterns (CGRAM Data)

CGRAM Address								
b7	b6	b5	b4	b3	b2	b1	b0	
0	1	0	0	0	0	0	0	0x40
					0	0	1	
					0	1	0	
					0	1	1	
					1	0	0	
					1	0	1	
					1	1	0	
					1	1	1	
0	1	0	0	1	0	0	0	0x48
					0	0	1	
					0	1	0	
					0	1	1	
					1	0	0	
					1	0	1	
					1	1	0	
					1	1	1	

CGRAM address starts from 0x40 value

CGRAM allows you to create custom characters by specifying the pixel data for each row.

0	1	0	0	0	0	0	0	0x40
0	1	0	0	1	0	0	0	0x48
0	1	0	1	0	0	0	0	0x50
0	1	0	1	1	0	0	0	0x58
0	1	1	0	0	0	0	0	0x60
0	1	1	0	1	0	0	0	0x68
0	1	1	1	0	0	0	0	0x70
0	1	1	1	1	0	0	0	0x78

```
// Function to store custom characters in CGRAM
void store() {
  // Store custom character at location 0
  cmd(64);  // Set CGRAM address to 0x40
  dat(0);  // First row of custom character
  dat(10);  // Second row of custom character
  dat(21);  // Third row of custom character
  dat(17);  // Fourth row of custom character
  dat(10);  // Fifth row of custom character
  dat(4);  // Sixth row of custom character
  dat(0);  // Seventh row of custom character
  dat(0);  // Eighth row of custom character
  cmd(0xc0); // Move to DDRAM address 0xC0 (second line)
  dat(0);  // Display custom character stored at location 0
  lcd_delay();
```

Set the CGRAM address accordingly

This code is used for sending the custom character to the LCD (specifying each pixel in each Matrix) as the output is shown in the pic

DDRAM is where the display data is stored, determining what characters appear on the screen and where.

Understanding the Code Block

The code block you provided is used to create and display a custom character on an LCD screen connected to a microcontroller. Let's break down each part:

CGRAM (Character Generator RAM)

1.CGRAM stands for **Character Generator RAM**. This is a special area of memory inside the LCD where you can create custom characters.
2.The LCD has a set of predefined characters (like letters and numbers), but sometimes you want to show a character that isn't built-in (like a smiley face). CGRAM allows you to design and store these custom characters.

DDRAM (Display Data RAM)

1.DDRAM stands for **Display Data RAM**. This is where the characters that are actually displayed on the LCD screen are stored.
2.When you write data to DDRAM, you're telling the LCD what characters to show and where to show them on the screen.

The Code Block Explained

Let's go through the code line by line:
1.cmd(64);
- This sets the **CGRAM address** to 0x40 (which is the starting address for storing the first custom character). It's like telling the LCD, "I'm going to give you the design for a new character, and I want you to start storing it at this address."

2.dat(0);
- This sends the first row of pixel data for the custom character to the LCD. Here, 0 means that the first row of the character will be all blank (no pixels turned on).

3.dat(10);
- This sends the second row of pixel data. The value 10 (in binary: 00001010) turns on some pixels in the second row, creating part of the custom character.

4.dat(21);
- This sends the third row of pixel data. The value 21 (in binary: 00010101) continues to shape the custom character.

5.dat(17);
- This sends the fourth row of pixel data. The value 17 (in binary: 00010001) adds more detail to the custom character.

6.dat(10);
- This sends the fifth row of pixel data. Again, the value 10 turns on specific pixels.

7.dat(4);
- This sends the sixth row of pixel data. The value 4 (in binary: 00000100) turns on a single pixel in the sixth row.

8.dat(0);
- This sends the seventh row of pixel data. The value 0 means that the seventh row will be blank.

9.dat(0);
- This sends the eighth (and final) row of pixel data. The value 0 means that the eighth row will also be blank.

Moving to DDRAM

10.cmd(0xc0);
- This sets the DDRAM address to 0xC0, which is the starting address for the second line of the display. It's like saying, "I want to start showing characters on the second line of the screen."

11.dat(0);
- This sends the code for the custom character stored at location 0 in CGRAM to DDRAM. It tells the LCD to display this custom character at the current position on the screen.

12.lcd_delay();
- This introduces a delay to give the LCD time to process the commands and data. It's like saying, "Take a moment to get everything set up."

Summary

- **CGRAM** allows you to create custom characters by specifying the pixel data for each row.
- **DDRAM** is where the display data is stored, determining what characters appear on the screen and where.
- This code block creates a custom character by writing pixel data to CGRAM and then displays this character by writing its code to DDRAM.

Purpose

The purpose of this code is to:
1.Create a custom character that isn't available in the LCD's built-in character set.
2.Display this custom character on the LCD screen, specifically on the second line.
By using CGRAM and DDRAM effectively, you can make your LCD show any custom-designed characters you need for your project.

Comprehensive Breakdown of the Code

1.Setting CGRAM Address to Store Custom Character:

```
cmd(0x40); // Set CGRAM address to 0x40
```

- `0x40` is the starting address in CGRAM for the first custom character.

2.Defining Custom Character Pixel Rows:

```
dat(0); // First row of custom character
dat(10); // Second row of custom character
dat(21); // Third row of custom character
dat(17); // Fourth row of custom character
dat(10); // Fifth row of custom character
dat(4); // Sixth row of custom character
dat(0); // Seventh row of custom character
dat(0); // Eighth row of custom character
```

- Each `dat(value)` writes a row of the custom character to CGRAM.
- Values are binary representations of the pixel pattern for each row.

3.Setting DDRAM Address to Display Custom Character:

```
cmd(0xC0); // Move to DDRAM address 0xC0 (second line)
```

- `0xC0` is the starting address for the second line on a 1602 LCD as per the command table you provided.

4.Displaying the Custom Character:

```
dat(0); // Display custom character stored at location 0
```

- `dat(0)` instructs the LCD to display the custom character stored at CGRAM address `0x40` (location 0).

5.Delay for LCD to Process Commands:

```
lcd_delay();
```

- `lcd_delay()` introduces a necessary pause for the LCD to process the commands and data sent to it.

Understanding CGRAM and DDRAM

CGRAM (Character Generator RAM)

- **Purpose**: CGRAM is used to store custom characters.
- **Addressing**:
  - `0x40` is the starting address for the first custom character.
  - Each custom character occupies 8 bytes in CGRAM.
  - The addresses for subsequent custom characters follow this pattern: `0x48`, `0x50`, `0x58`, etc.

DDRAM (Display Data RAM)

- **Purpose**: DDRAM stores the characters to be displayed on the screen.
- **Addressing**:
  - For the 1602 LCD:
    - First line addresses range from `0x00` to `0x0F`.
    - Second line addresses start at `0xC0` as per your command table.
  - `0xC0` is used to move the cursor to the beginning of the second line.

Summary

- **CGRAM** addresses like `0x40`, `0x48`, etc., are used to store custom characters.
- **DDRAM** addresses like `0x00` (first line) and `0xC0` (second line) specify positions on the LCD where characters will be displayed.
- `dat(0)` refers to the custom character stored at the first CGRAM location (`0x40`).

This setup allows you to create custom characters and display them on the 1602 LCD at specific locations, providing flexibility in how you present information on the screen.