LCD interfacing

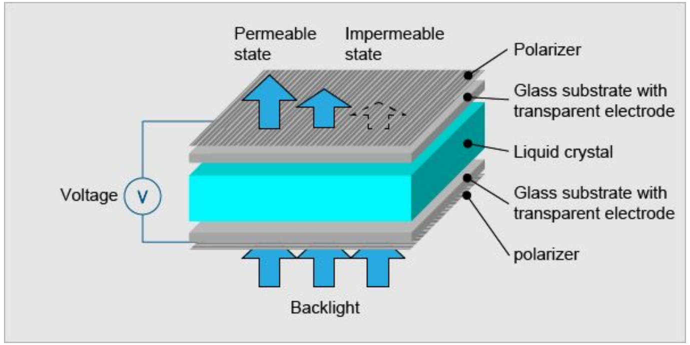
By: Yehia M. Abu Eita

Outlines

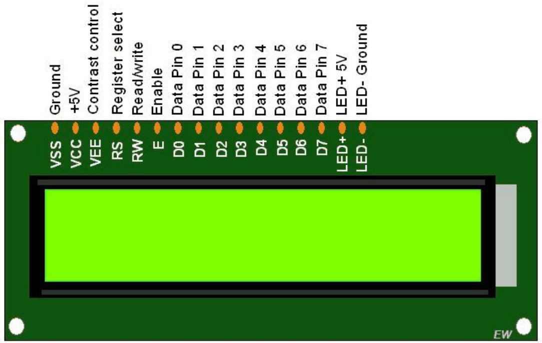
- Introduction
- LCD 16x2 pin description
- LCD instruction set
- LCD modes
- 8-bit mode programming
- 4-bit mode programming
- Creating custom characters

Introduction

- It is acronym of **L**iquid **C**rystal **D**isplay.
- Liquid crystal refers to the intermediate status of a substance between solid (crystal) and liquid.


Permeable state
Impermeable state
Polarizer
Glass substrate with transparent electrode
Liquid crystal
Voltage
V
Glass substrate with transparent electrode
polarizer
Backlight

LCD 16x2 pin description

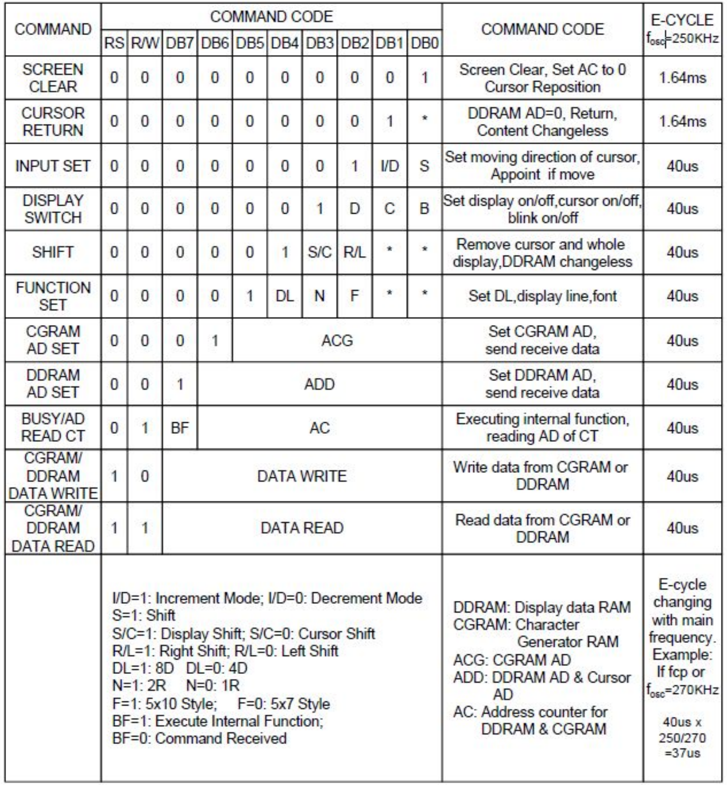


Pin Number	Pin Name	Description
1	VSS	Ground
2	VCC	+5 Volts
3	VEE	Contrast Control 0 volts: High contrast

Pin Number	Pin Name	Description
4	RS	Register Select 0: Command Register 1: Data Register
5	RW	Read / Write 0: Write 1: Read
6	E	Enable High-Low pulse
7-14	D0-D7	Data pins
15	LED+	+5 Volts
16	LED-	Ground

LCD instruction set

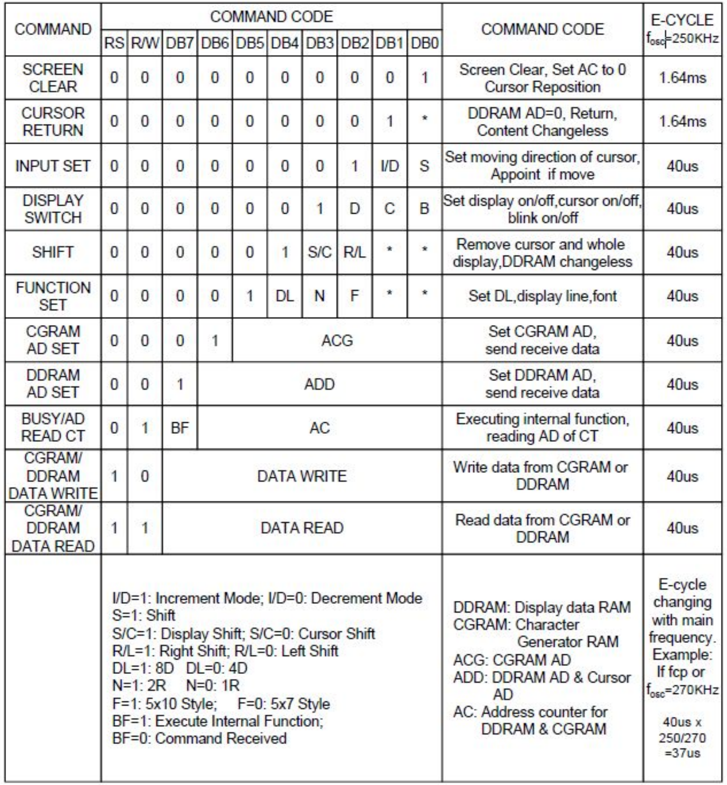
HEX code	Command to LCD	Execution Time
0x01	Clear the display screen	1.64ms
0x06	Shift the cursor right (e.g. data gets written in an incrementing order, left to right)	40 us
0x0C	Display on, cursor off	40 us
0x0E	Display on, cursor blinking	40 us
0x80	Force the cursor to the beginning of the 1st line	40 us
0xC0	Force the cursor to the beginning of the 2nd line	40 us
0x10	Shift cursor position to the left	40 us



COMMAND	COMMAND CODE										COMMAND CODE	E-CYCLE f_{osc}=250KHz
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
SCREEN CLEAR	0	0	0	0	0	0	0	0	0	1	Screen Clear, Set AC to 0 Cursor Reposition	1.64ms
CURSOR RETURN	0	0	0	0	0	0	0	0	1	*	DDRAM AD=0, Return, Content Changeless	1.64ms
INPUT SET	0	0	0	0	0	0	0	1	I/D	S	Set moving direction of cursor, Appoint if move	40us
DISPLAY SWITCH	0	0	0	0	0	0	1	D	C	B	Set display on/off,cursor on/off, blink on/off	40us
SHIFT	0	0	0	0	0	1	S/C	R/L	*	*	Remove cursor and whole display,DDRAM changeless	40us
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*	Set DL,display line,font	40us
CGRAM AD SET	0	0	0	1	ACG						Set CGRAM AD, send receive data	40us
DDRAM AD SET	0	0	1	ADD							Set DDRAM AD, send receive data	40us
BUSY/AD READ CT	0	1	BF	AC							Executing internal function, reading AD of CT	40us
CGRAM/ DDRAM DATA WRITE	1	0	DATA WRITE								Write data from CGRAM or DDRAM	40us
CGRAM/ DDRAM DATA READ	1	1	DATA READ								Read data from CGRAM or DDRAM	40us
	I/D=1: Increment Mode; I/D=0: Decrement Mode S=1: Shift S/C=1: Display Shift; S/C=0: Cursor Shift R/L=1: Right Shift; R/L=0: Left Shift DL=1: 8D DL=0: 4D N=1: 2R N=0: 1R F=1: 5x10 Style; F=0: 5x7 Style BF=1: Execute Internal Function; BF=0: Command Received										DDRAM: Display data RAM CGRAM: Character Generator RAM ACG: CGRAM AD ADD: DDRAM AD & Cursor AD AC: Address counter for DDRAM & CGRAM	E-cycle changing with main frequency. Example: If fcp or f_{osc}=270KHz 40us x 250/270 =37us

LCD instruction set

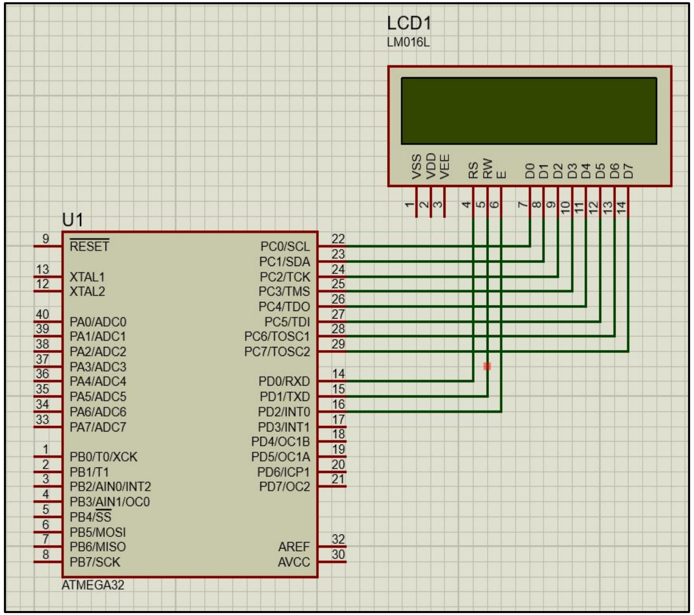
HEX code	Command to LCD	Execution Time
0x14	Shift cursor position to the right	40 us
0x18	Shift entire display to the left	40 us
0x1C	Shift entire display to the right	40 us
0x38	2 lines, 5x8 matrix, 8-bit mode	40 us
0x28	2 lines, 5x8 matrix,4-bit mode	40 us
0x30	1 line, 8-bit mode	40 us
0x20	1 line, 4-bit mode	40 us



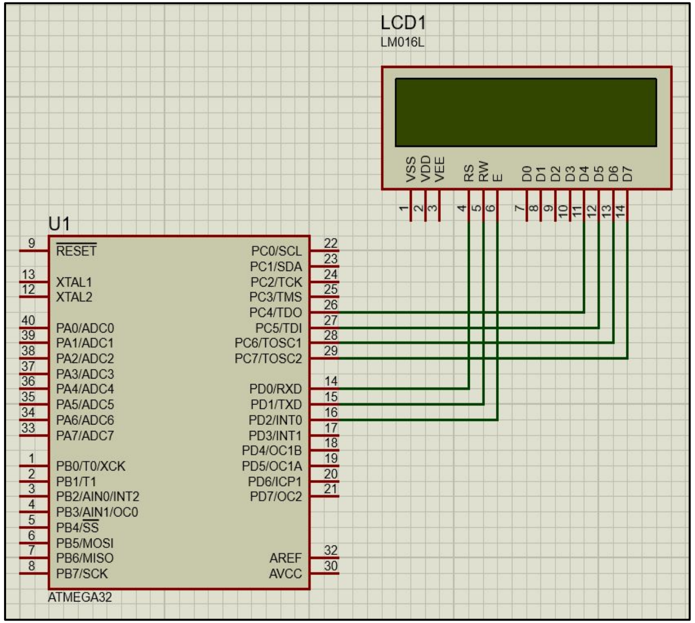
COMMAND	COMMAND CODE										COMMAND CODE	E-CYCLE f_{osc}=250KHz
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
SCREEN CLEAR	0	0	0	0	0	0	0	0	0	1	Screen Clear, Set AC to 0 Cursor Reposition	1.64ms
CURSOR RETURN	0	0	0	0	0	0	0	0	1	*	DDRAM AD=0, Return, Content Changeless	1.64ms
INPUT SET	0	0	0	0	0	0	0	1	I/D	S	Set moving direction of cursor, Appoint if move	40us
DISPLAY SWITCH	0	0	0	0	0	0	1	D	C	B	Set display on/off,cursor on/off, blink on/off	40us
SHIFT	0	0	0	0	0	1	S/C	R/L	*	*	Remove cursor and whole display,DDRAM changeless	40us
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*	Set DL,display line,font	40us
CGRAM AD SET	0	0	0	1	ACG						Set CGRAM AD, send receive data	40us
DDRAM AD SET	0	0	1	ADD							Set DDRAM AD, send receive data	40us
BUSY/AD READ CT	0	1	BF	AC							Executing internal function, reading AD of CT	40us
CGRAM/ DDRAM DATA WRITE	1	0	DATA WRITE								Write data from CGRAM or DDRAM	40us
CGRAM/ DDRAM DATA READ	1	1	DATA READ								Read data from CGRAM or DDRAM	40us
	I/D=1: Increment Mode; I/D=0: Decrement Mode S=1: Shift S/C=1: Display Shift; S/C=0: Cursor Shift R/L=1: Right Shift; R/L=0: Left Shift DL=1: 8D DL=0: 4D N=1: 2R N=0: 1R F=1: 5x10 Style; F=0: 5x7 Style BF=1: Execute Internal Function; BF=0: Command Received										DDRAM: Display data RAM CGRAM: Character Generator RAM ACG: CGRAM AD ADD: DDRAM AD & Cursor AD AC: Address counter for DDRAM & CGRAM	E-cycle changing with main frequency. Example: If fcp or f_{osc}=270KHz 40us x 250/270 =37us

LCD modes

8-bit mode

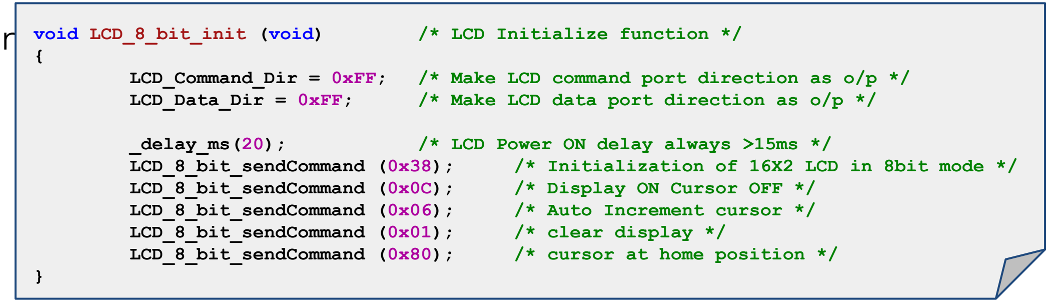


4-bit mode



8-bit mode programming

- **Initialization**:
  - **Power ON** the LCD
  - **Wait for 15 ms**, Power ON initialization time
  - **Send 0x38 command** to initialize **2 lines**, **5x8 matrix**, **8-bit mode**
  - **Send** any **Display ON** command (**0x0E**, **0x0C**)
  - **Send 0x06 command** (increment cursor)



```
void LCD_8_bit_init (void)          /* LCD Initialize function */
{
        LCD_Command_Dir = 0xFF;     /* Make LCD command port direction as o/p */
        LCD_Data_Dir = 0xFF;        /* Make LCD data port direction as o/p */

        _delay_ms(20);              /* LCD Power ON delay always >15ms */
        LCD_8_bit_sendCommand (0x38);       /* Initialization of 16X2 LCD in 8bit mode */
        LCD_8_bit_sendCommand (0x0C);       /* Display ON Cursor OFF */
        LCD_8_bit_sendCommand (0x06);       /* Auto Increment cursor */
        LCD_8_bit_sendCommand (0x01);       /* clear display */
        LCD_8_bit_sendCommand (0x80);       /* cursor at home position */
}
```

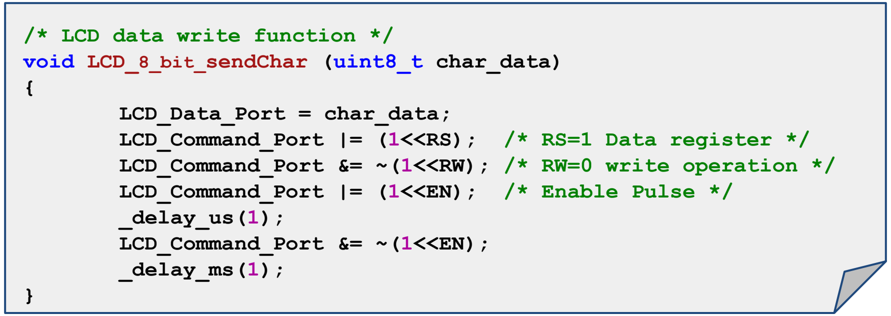
8-bit mode programming

- **Send command function**:
  - **Send the command** value to the data port
  - Make RS pin low, **RS = 0** (command register)
  - Make RW pin low, **RW = 0** (write operation)
  - Generate **high to low pulse** from the **Enable (E) pin** of **minimum delay of 450 ns**

```
void LCD_8_bit_sendCommand (uint8_t cmnd)
{
        LCD_Data_Port= cmnd;
        LCD_Command_Port &= ~(1<<RS); /* RS=0 command register */
        LCD_Command_Port &= ~(1<<RW); /* RW=0 Write operation */
        LCD_Command_Port |= (1<<EN);  /* Enable pulse */
        _delay_us(1);
        LCD_Command_Port &= ~(1<<EN);
        _delay_ms(3);
}
```

8-bit mode programming

- **Display character function**:
  - **Send character ASCII** value to the **data port**
  - Make the RS pin High, **RS = 1** (Data register)
  - Make the RW pin Low, **RW = 0** (Write operation)
  - Generate a **high to low pulse** from **the Enable (E) pin**



```
/* LCD data write function */
void LCD_8_bit_sendChar (uint8_t char_data)
{
        LCD_Data_Port = char_data;
        LCD_Command_Port |= (1<<RS);  /* RS=1 Data register */
        LCD_Command_Port &= ~(1<<RW); /* RW=0 write operation */
        LCD_Command_Port |= (1<<EN);  /* Enable Pulse */
        _delay_us(1);
        LCD_Command_Port &= ~(1<<EN);
        _delay_ms(1);
}
```

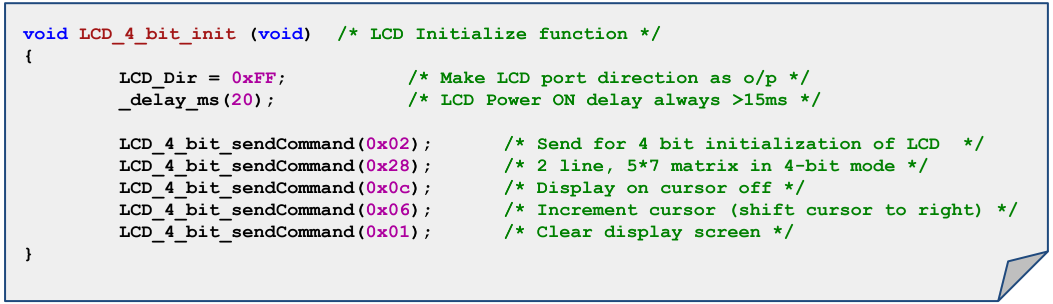
8-bit mode programming

- **Display string function**:
  - This function takes a string (an array of characters) and sends one character at a time to the LCD data function till the end of the string.
  - A ‘**for loop**’ is used for **sending a character** in each iteration.
  - A **NULL** character indicates **end of the string**.

```c
void LCD_8_bit_sendString (uint8_t *str)
{
        uint16_t i;
        for(i=0;str[i]!=0;i++)  /* send each char of string till the NULL */
        {
                LCD_8_bit_sendChar (str[i]);  /* call LCD data write */
        }
}
```

4-bit mode programming

- **Initialization**:
  - **Power ON** the LCD
  - **Wait for 15ms**, Power-on initialization time
  - **Send 0x02 command** which initializes the LCD in **4-bit mode**
  - **Send 0x28 command** which configures the LCD in **2-line**, **4-bit mode**, and **5x8 dots**
  - Send any **Display ON command** (**0x0E**, **0x0C**)
  - **Send 0x06** command (increment cursor)

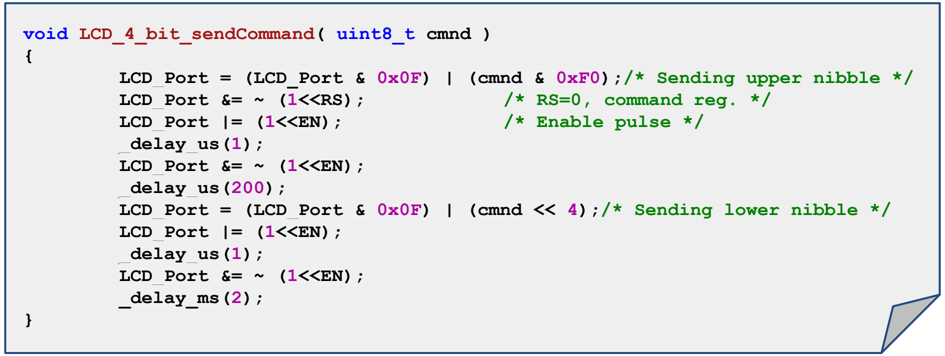


```c
void LCD_4_bit_init (void)  /* LCD Initialize function */
{
        LCD_Dir = 0xFF;             /* Make LCD port direction as o/p */
        _delay_ms(20);              /* LCD Power ON delay always >15ms */

        LCD_4_bit_sendCommand(0x02);        /* Send for 4 bit initialization of LCD  */
        LCD_4_bit_sendCommand(0x28);        /* 2 line, 5*7 matrix in 4-bit mode */
        LCD_4_bit_sendCommand(0x0c);        /* Display on cursor off */
        LCD_4_bit_sendCommand(0x06);        /* Increment cursor (shift cursor to right) */
        LCD_4_bit_sendCommand(0x01);        /* Clear display screen */
}
```

4-bit mode programming

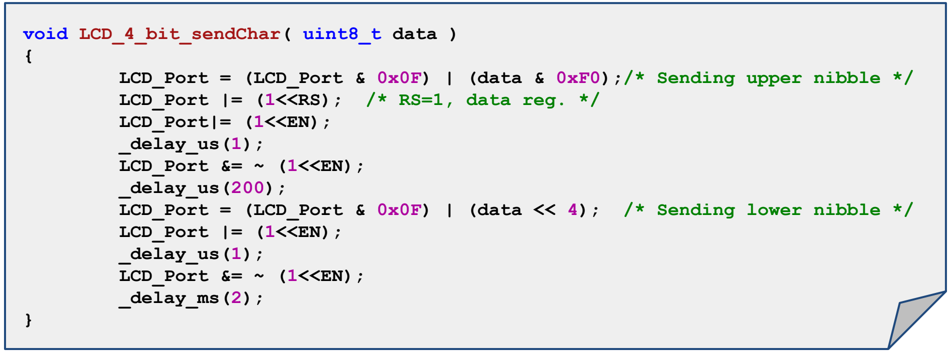
- **Send command function**:
  - **Send** the **Higher nibble** of command
  - Make RS pin low, **RS=0** (command register)
  - Make RW pin low, **RW=0** (write operation) or connect it to ground
  - Generate a **high to low pulse** from **the Enable (E) pin**
  - **Send** the **lower nibble** of command
  - Generate a **high to low pulse** from **the Enable (E) pin**



```c
void LCD_4_bit_sendCommand( uint8_t cmnd )
{
        LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0);/* Sending upper nibble */
        LCD_Port &= ~ (1<<RS);                    /* RS=0, command reg. */
        LCD_Port |= (1<<EN);                      /* Enable pulse */
        _delay_us(1);
        LCD_Port &= ~ (1<<EN);
        _delay_us(200);
        LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);/* Sending lower nibble */
        LCD_Port |= (1<<EN);
        _delay_us(1);
        LCD_Port &= ~ (1<<EN);
        _delay_ms(2);
}
```

4-bit mode programming

- **Display character function**:
  - **Send** the **Higher nibble** of data
  - Make RS pin high, **RS=1** (data register)
  - Make RW pin low, **RW=0** (write operation) or connect it to ground
  - Generate a **high to low pulse** from **the Enable (E) pin**
  - **Send** the **lower nibble** of data
  - Generate a **high to low pulse** from **the Enable (E) pin**



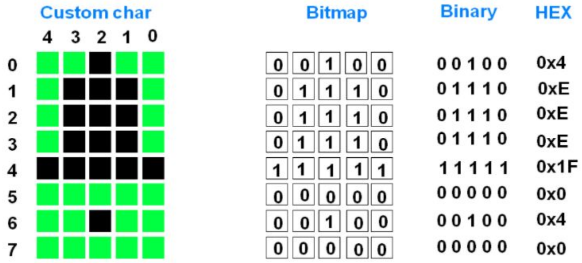
```
void LCD_4_bit_sendChar( uint8_t data )
{
        LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0);/* Sending upper nibble */
        LCD_Port |= (1<<RS);  /* RS=1, data reg. */
        LCD_Port|= (1<<EN);
        _delay_us(1);
        LCD_Port &= ~ (1<<EN);
        _delay_us(200);
        LCD_Port = (LCD_Port & 0x0F) | (data << 4);  /* Sending lower nibble */
        LCD_Port |= (1<<EN);
        _delay_us(1);
        LCD_Port &= ~ (1<<EN);
        _delay_ms(2);
}
```

Creating custom characters

- LCD 16x2 can display user defined **custom characters**
- To print character symbol on LCD, the **character's ASCII** code must be sent to the LCD
- You can print your own custom character, which is **not included** in **ASCII**
- LCD 16x2 has memory space of **64 bytes** called as **CGRAM** (character generator RAM)
- According to the **pixels pattern / bitmap**, a customized character can be created
- **Each character line will be stored in one byte**
- The CGRAM **starting address is 0x40**
- The CGRAM **ending address is 0x7F**
- The 16x2 LCD provides **only 8** custom characters

Creating custom characters

- **Set the CGRAM address**:
  - Set the character starting address
  - Write the bitmap values in that addresses

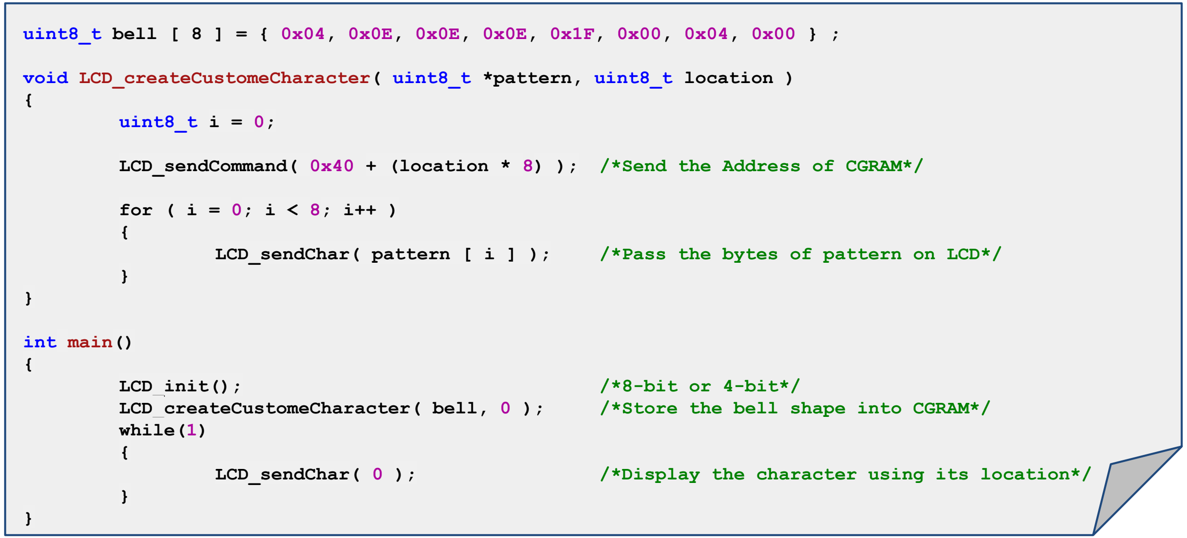


	Custom char 4 3 2 1 0	Bitmap	Binary	HEX
0	🟩🟩⬛🟩🟩	0 0 1 0 0	0 0 1 0 0	0x4
1	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
2	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
3	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
4	⬛⬛⬛⬛⬛	1 1 1 1 1	1 1 1 1 1	0x1F
5	🟩🟩🟩🟩🟩	0 0 0 0 0	0 0 0 0 0	0x0
6	🟩🟩⬛🟩🟩	0 0 1 0 0	0 0 1 0 0	0x4
7	🟩🟩🟩🟩🟩	0 0 0 0 0	0 0 0 0 0	0x0

Command	Code					Description	Execution Time
	RS	R/W	DB7	DB6	DB5-DB0		
Set CGRAM address	0	0	0	1	Address	This is used to set the address of CGRAM	40us

Creating custom characters

- Example:



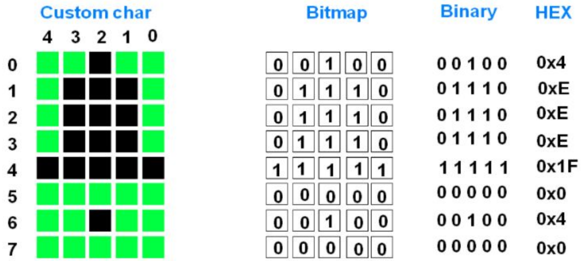
```
uint8_t bell [ 8 ] = { 0x04, 0x0E, 0x0E, 0x0E, 0x1F, 0x00, 0x04, 0x00 } ;

void LCD_createCustomeCharacter( uint8_t *pattern, uint8_t location )
{
        uint8_t i = 0;

        LCD_sendCommand( 0x40 + (location * 8) );  /*Send the Address of CGRAM*/

        for ( i = 0; i < 8; i++ )
        {
                LCD_sendChar( pattern [ i ] );     /*Pass the bytes of pattern on LCD*/
        }
}

int main()
{
        LCD_init();                                /*8-bit or 4-bit*/
        LCD_createCustomeCharacter( bell, 0 );     /*Store the bell shape into CGRAM*/
        while(1)
        {
                LCD_sendChar( 0 );                 /*Display the character using its location*/
        }
}
```



	Custom char	Bitmap	Binary	HEX
	4 3 2 1 0			
0	🟩🟩⬛🟩🟩	0 0 1 0 0	0 0 1 0 0	0x4
1	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
2	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
3	🟩⬛⬛⬛🟩	0 1 1 1 0	0 1 1 1 0	0xE
4	⬛⬛⬛⬛⬛	1 1 1 1 1	1 1 1 1 1	0x1F
5	🟩🟩🟩🟩🟩	0 0 0 0 0	0 0 0 0 0	0x0
6	🟩🟩⬛🟩🟩	0 0 1 0 0	0 0 1 0 0	0x4
7	🟩🟩🟩🟩🟩	0 0 0 0 0	0 0 0 0 0	0x0

Summary

- Now you are familiar with the LCD
- Now you are able to implement your LCD driver
- Remember, 8-bit mode consumes more I/O pins
- Remember, storing a custom character must be in the CGRAM
- Remember, the location of the custom character acts like its ASCII