INGENIERÍA MECATRÓNICA

Forma

Descripción generada automáticamente

Diego Cervantes Rodríguez

Lenguaje C, Programación Microcontroladores: ATmega328P (Arduino)

Microchip Studio

LCD (Liquid Crystal Display) Alfanumérico

Contenido

[LCD Alfanumérico 2](#_Toc137448318)

[Descripción de los Pines del LCD 3](#_Toc137448319)

[Posición de Caracteres ASCII (Datos) en el LCD: RAM-DDRAM 4](#_Toc137448320)

[Generador de Caracteres Personalizados: ROM-CGROM 4](#_Toc137448321)

[Comandos de Control del LCD 7](#_Toc137448322)

[Código C Microchip Studio: Paquetes 4 bits 10](#_Toc137448323)

[Declaración de Caracteres Personalizados en Microchip Studio 13](#_Toc137448324)

[Puertos B, C y D Físicos en el Arduino 15](#_Toc137448325)

[Conexión Física del LCD con la Placa Arduino UNO 16](#_Toc137448326)

[Programar Físicamente la Placa Arduino UNO con el Programa de Microchip Studio 17](#_Toc137448327)

# LCD Alfanumérico

El LCD son las siglas en ingles de *Liquid Crystal Display* o en su traducción al español como visualizador de cristal líquido o comúnmente conocido como simplemente LCD, existen muchos tipos de modelos que venden los diferentes fabricantes, estos incluyen de 1 a 4 líneas de texto y desde 8 a 40 caracteres por línea, además con o sin iluminación de fondo (backlight), todos los LCD alfanuméricos son compatibles con la placa HITACHI HD44780, que es un microcontrolador contenido en una PCB intermedia que ayuda a controlar el LCD con una placa como el Arduino UNO. En la figura 1.1 se muestra el aspecto de LCD comercial de 2 líneas X 16 caracteres. El LCD en general tiene 16 pines, descritos a continuación.

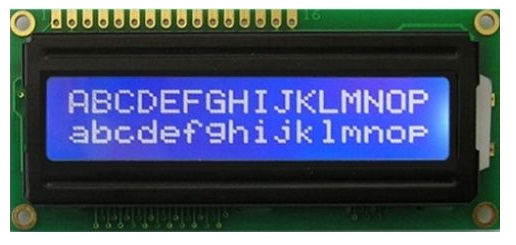


Figura 1.1 LCD comercial.

Las características generales de los LCDs es que el envío de datos puede ser en 8 bits o 4 bits, maneja diferentes caracteres como ASCII, Katakana, Griegos y Matemáticos, puede desplazarse de derecha a izquierda y viceversa, tiene una memoria de 40 caracteres por cada una de sus dos líneas de la pantalla, se pueden agregar 8 caracteres personalizados, son de consumo reducido (menores a 8 mW), al emplear un potenciómetro de 10K en el pin 3 se regula el contraste (en vez de usar un potenciómetro se puede poner un diodo Zener). Los pines 4, 5 y 6 son los de control y todos se muestran en la figura 1.2.

Se tiene al pin 5 de lectura o escritura (R/W) para no enviar varios paquetes a la vez, osea saber si está ocupado el puerto y que no se mande otro hasta que se desocupe, pero esto se usaba anteriormente, en la actualidad no se utiliza, por lo que solo se manda a tierra.

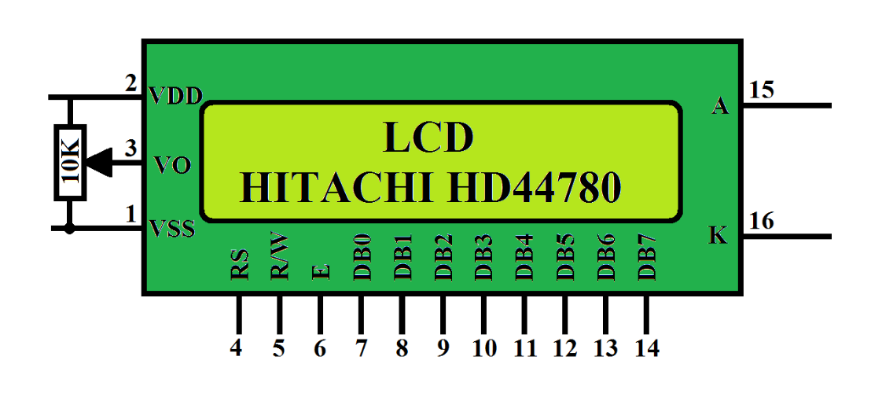


Figura 1.2 Distribución de pines del LCD

## Descripción de los Pines del LCD

Los pines del LCD alfanumérico son los que se muestran en la tabla 1.1

Tabla 1.1 Pines del LCD

|  |  |  |  |
| --- | --- | --- | --- |
| **Patita** | **Nombre** | **Descripción** | **Valores que se emplean** |
| 1 | **VSS** | Tierra | **GND = 0 Volts = 0 Lógico = 0.** |
| 2 | **VDD** | Fuente de Voltaje | **Vcc = 5 Volts = 1 Lógico = 1.** |
| 3 | **VO** | Ajuste de Contraste | Máximo Contraste Cuando:  **VO = GND.** |
| 4 | **RS** | Selección de Registro | RS = 0; **Comandos.**  RS = 1; **Datos.** |
| 5 | **R/W** | Lectura / Escritura | R/W = 0; **Escribir en el LCD.**  R/W = 1; **Leer en el LCD.** |
| 6 | **E** | Habilitación del LCD | E = 0; **Inhabilitado.**  E = 1; **Habilitado.** |
| 7 – 14 | **DB0 - DB7** | Data Bus | Modo 8 Bits = Se emplea usando todo el Bus.  Modo 4 Bits = Usa solo el nibble alto del Bus. |
| 1. - 16 | **A - k** | Ánodo y Cátodo | Iluminación trasera del LCD *(opcional).*  Poner una resistencia en serie de 20 a 100 Ω. |

Se puede manejar un LCD de 4 formas diferentes:

* **11 líneas:** 3 de control y 8 de datos.
* **10 líneas:** 2 de control y 8 de datos – R/W se manda a tierra.
* **7 líneas:** 3 de control y 4 de datos.
* **6 líneas:** 2 de control y 4 de datos – R/W se manda a tierra, *es la que se usa con Arduino UNO porque los Puertos* ***B****,* ***C*** *y* ***D*** *son de 6 bits, solo el puerto* ***D*** *es de 8*.

Cuando se usan 3 líneas de control, se emplean **E (Habilitación)**, **RS** **(Selección)** y **R/W** **(Lectura/Escritura)** pudiendo leer y escribir en el LCD, cuando se usan solo 2 líneas de control se emplean solamente **E** y **RS**, **R/W** no se emplea ya que solo se puede escribir en el LCD y se conecta directamente a GND este pin.

Un dato es un código ASCII para mostrar un carácter en el LCD, mientras que un comando es decirle al LCD que limpie su pantalla, mueva el cursor a la izquierda o derecha, que sea obscuro el cursor, que sea una sola línea, que utilice letras de 5X8 pixeles o 5X10 pixeles, etc. **El comando es una instrucción que debe seguir el LCD, mientras que el dato es una letra que debe mostrar.**

Para enviar un **comando** al LCD se necesita poner los 3 pines de control de la sig. manera:

**•RS = 0, •R/W = 0, • E = 1.**

Para enviar un **dato** al LCD se necesita poner los 3 pines de control de la sig. manera:

**•RS = 1, •R/W = 0, •E = 1.**

## Posición de Caracteres ASCII (Datos) en el LCD: RAM-DDRAM

El LCD posee un área de memoria RAM donde se almacenan los caracteres en código ASCII que se visualizan en el LCD, llamada DDRAM que significa en ingles *Data Display RAM* o Memoria de Acceso Aleatorio para Visualizar Datos, su capacidad es de 80 bytes, 40 por línea, en la línea 1 empieza en la posición 0x00 y termina en la posición 0x27 y en la línea 2 comienza en la posición 0x40 y termina en la posición 0x67, solo se pueden visualizar un máximo de 32 bytes, es decir 16 caracteres por línea como se muestra en la figura 1.3.

**El DDRAM es básicamente una memoria de posiciones en la LCD, donde se le indica a qué posición quiero mandar un carácter dentro de la pantalla del LCD**, ya sea para poner la letra al inicio de la primera línea, al final de la primera línea, a la mitad de la segunda línea, etc.

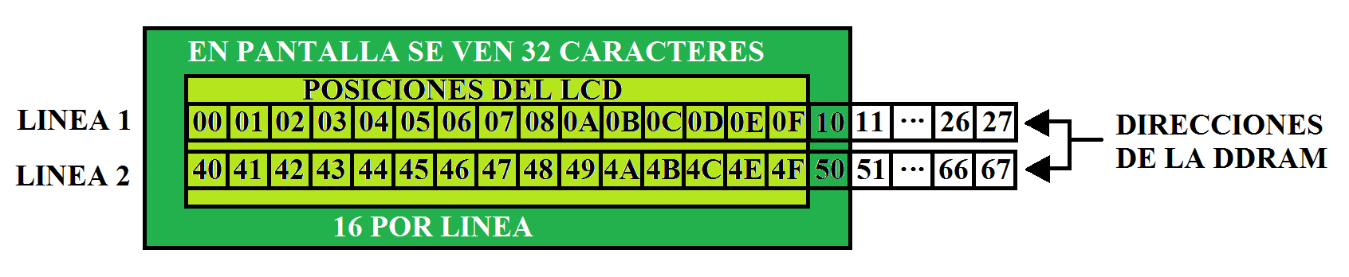


Figura 1.3 DDRAM

## Generador de Caracteres Personalizados: ROM-CGROM

Es el área de memoria no volátil (de largo plazo) llamada CGROM, donde se almacena una tabla con los patrones de todos los caracteres que se puede visualizar en el LCD grabados de fábrica, pueden variar desde 192 a 200 caracteres de 5X7 puntos y 32 de 5X10 puntos (pixeles), ya que se pueden crear 8 letras o símbolos personalizados.

Cada carácter se representa por un código binario de 8 bits muy parecido al ASCII empleando el mismo código para los códigos ASCII del 32 al 125, del 126 en adelante difieren a los símbolos ASCII, la tabla 1.2.2 muestra los caracteres que tiene grabados de fábrica. En la tabla 1.2.2 existe una columna denominada CG RAM donde se puede grabar y acceder a ocho caracteres personalizados definidos por el usuario, los caracteres están hechos en una matriz de 5X8 puntos como se ve en la figura 1.4 (a) y en (b) un carácter de llave que se puede hacer y grabar en la CGROM, todas las letras se conforman de 5X7 pixeles mientras los caracteres especiales pueden usar toda la matriz de pixeles que es de 5X8.

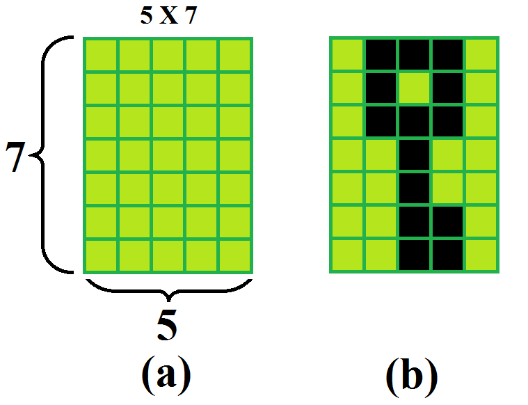


Figura 1.4 matriz 5 X 7 (a) y carácter hecho por el usuario (b).

Los caracteres descritos en la tabla 1.2.2 son obtenidos de la tabla de código ASCII de la tabla 1.2.1, de hecho, si se compara el número de cada carácter ASCII incluido en la matriz de la tabla 1.2.2 con cada carácter de la tabla ASCII 1.2.1, se podrá observar que es el mismo.

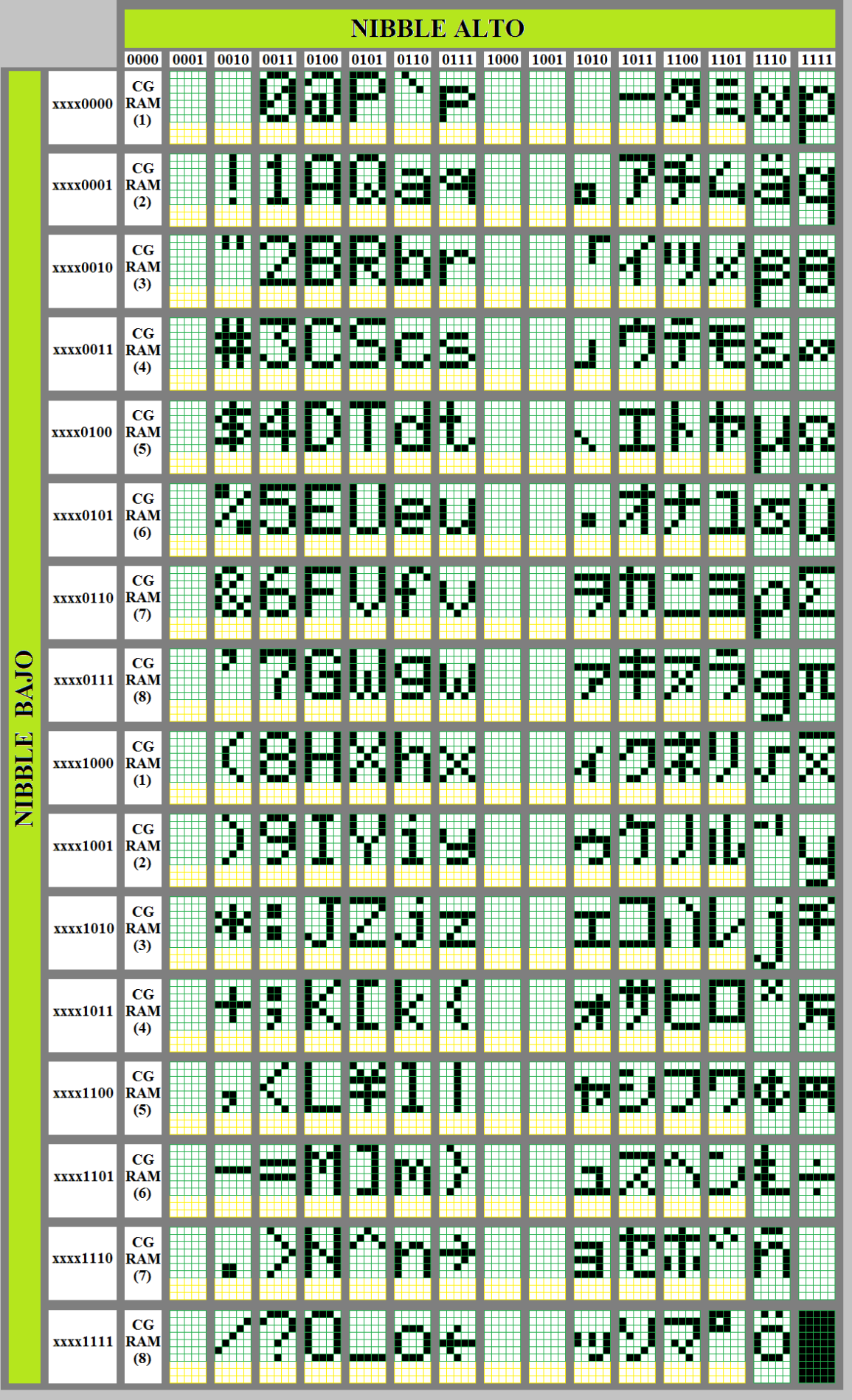
Tabla 1.2.1 Tabla de código ASCII



Cuando se quiera crear caracteres especiales, se debe mandar un comando a las siguientes direcciones de la tabla 1.2.2:

1. 0**1**00 0000 **= 0x40**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 0.
2. 0**1**00 **1**000 **= 0x48**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 1.
3. 0**1**0**1** 0000 = **0x50**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 2.
4. 0**1**0**1** **1**000 = **0x58**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 3.
5. 0**11**0 0000 = **0x60**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 4.
6. 0**11**0 **1**000 = **0x68**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 5.
7. 0**111** 0000 = **0x70**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 6.
8. 0**111** **1**000 = **0x78**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 7.

Tabla 1.2.2 Caracteres ASCII definidos de fábrica en la Memoria CGROM



## Comandos de Control del LCD

Los comandos de control que acepta el microcontrolador intermedio HD44780 se resumen en la tabla 1.3, así como los tiempos de ejecución, donde:

* **X = Don’t care:** Estado donde no importa si hay un 1, un 0 o no está conectado a nada.
* **I/D = Incremento/Decremento:** Sirve para mover la dirección en la memoria DDRAM, que es donde se indica la posición del carácter que se quiere mostrar en la LCD. Cada fila puede mostrar 16 caracteres.
  + Con 1 se mueve la posición del caracter hacia la derecha.
  + Con 0 se mueve la posición del caracter a la izquierda.
* **S = Desplazamiento:** Sirve para desplazar todos los caracteres de la pantalla hacia un lado indicado por I/D o dejar la pantalla del LCD estática
  + Con 1 la pantalla se desplaza.
  + Con 0 la pantalla se queda estática.
* **D = Encender o Apagar:** Sirve para encender o apagar la pantalla y guardar en la memoria de posiciones DDRAM los caracteres para que se muestren en el mismo lugar cuando esta se vuelva a encender, se usa para ahorrar energía.
  + Con 1 la pantalla se enciende.
  + Con 1 la pantalla se apaga.
* **C = Cursor Display:** Mostrar u ocultar el cursor en el LCD.
  + Con 1 se muestrael cursor.
  + Con 0 se oculta el cursor.
* **B = Blink Cursor:** Hace que el cursor parpadee o se quede estático.
  + Con 1 el cursor parpadea.
  + Con 0 se muestra estático.
* **R/L = Cursor Right/Left:** Permite mover el cursor a la derecha o a la izquierda sin que se escriba nada al hacerlo.
  + Con 1 el cursor se mueve hacia la derecha.
  + Con 0 el cursor se mueve hacia la izquierda.
* **S/C = Set Cursor:** Permite mover el cursor a alguna parte de la pantalla del LCD, ya sea a otra fila o a cualquier otra posición de las 16 disponibles en alguna de las 2 filas.
  + Con 1 el cursor se puede mover de forma horizontal de una fila a otra en las 2 existentes.
  + Con 0 el cursor se puede mover a cualquier posición del LCD que sea accesible con la memoria de posiciones DDRAM, osea a cualquiera de las 16 posiciones en las 2 filas.
* **DL = Data Load:** Indica si los datos se le mandarán al LCD de jalón con 8 bits o de 4 en 4, cada conjunto de 4 bits es llamado nibble, el nibble alto son los 4 primeros bits vistos de izquierda a derecha y el nibble bajo son los 4 restantes de un número de 1 byte, osea 8 bits. Usualmente del Arduino y otros microcontroladores los datos se mandan de 4 en 4 ya que los puertos tienen 6 pines disponibles.
  + Con 1 se indica que los datos se manden en paquetes de 8 bits.
  + Con 0 se indica que los datos se manden en paquetes de 4 bits.
* **N = Number of Lines:** Indica si se quieren usar las dos filas del LCD o si solo se quiere usar una.
  + Con 1 se puede utilizar las dos filas del LCD.
  + Con 0 se puede usar solo una fila del LCD.
* **F = Font:** Con este bit se indica si se quieren crear caracteres de 5X10 puntos o de 5X7 puntos (pixeles), osea que se puede cambiar el tamaño de las letras en el LCD.
  + Con **1** los caracteres del LCD se crean de 5X10 pixeles, **letras grandotas**.
  + Con **0** los caracteres del LCD se crean de 5X7 pixeles, **letras chiquitas**.
* **BF = Busy Flag:** La bandera de ocupado sirve para saber si se puede mandar un nuevo dato a la LCD o no. En la tabla se puede ver en la última columna cuánto tiempo le toma al LCD cumplir cada ejecución, pero si no se quiere hacer el manejo de tiempos se puede utilizar esta bandera para saber si el bus de bits está siendo ocupado o no. Máximo se puede tener un tiempo de espera de 2 milisegundos entre cada instrucción mandada al LCD, por lo que hacer uso de esta bandera es medio innecesario.

Todas las instrucciones de comandos que se le puede mandar a los pines del LCD son mostradas en la Tabla 1.3.

Tabla 1.3 Comandos de Control

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| COMANDO | CODIGO DEL COMANDO | | | | | | | | | |  | TIEMPO |
| RS | R/W | DB7 | DB6 | DB5 | | DB4 | DB3 | DB2 | DB1 | DB0 |
| Limpiar Pantalla | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | **1** | 1.64ms |
| Retorno a Casa | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | **1** | **X** | 1.64ms |
| Ajuste modo de entrada | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | **1** | **I/D** | **S** | 40µs |
| Control de Pantalla | 0 | 0 | 0 | 0 | 0 | 0 | | **1** | **D** | **C** | **B** | 40µs |
| Desplazamiento de Cursor & Pantalla | 0 | 0 | 0 | 0 | 0 | **1** | | **S/C** | **R/L** | **X** | **X** | 40µs |
| Ajuste de Función | 0 | 0 | 0 | 0 | **1** | **DL** | | **N** | **F** | **X** | **X** | 40µs |
| Ajuste de dirección de la CGROM | 0 | 0 | 0 | **1** | [**Dirección del CGROM**](#_Generador_de_Caracteres) | | | | | | | 40µs |
| Ajuste de la dirección de la DDRAM | 0 | 0 | **1** | [**Dirección del DDRAM**](#_Posición_de_Caracteres) | | | | | | | | 40µs |
| Lectura de la bandera de ocupado | 0 | **1** | **BF** | [**Dirección del DDRAM**](#_Posición_de_Caracteres) | | | | | | | | 1µs |
| Escribir en la RAM | **1** | 0 | **Escribir Dato** | | | | | | | | | 46µs |
| Leer de la RAM | **1** | **1** | **Leer Dato** | | | | | | | | | 46µs |

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 1.5 Pines de Comandos del LCD

**Comando Limpiar Pantalla** ***(Clear Display):***Este comando borra toda la pantalla del LCD y manda el cursor a la posición 00 de la DDRAM. Se debe enviar un 1 en modo comando con RS = 0 y E = 1.

**Comando Retorno a Casa *(Return Home):*** Este comando regresa a la posición de inicio es decir la 00 de la DDRAM, sin alterar el contenido de la pantalla del LCD se envía en modo comando un 2 o un 3 ya que el primer bit es un estado de no importa.

**Comando de Ajuste modo de entrada *(Entry Mode Set):*** Este comando controla el cambio e incremento y decremento de los datos en la LCD.

|  |  |
| --- | --- |
| **I/D = 1** | Incrementa uno la dirección de la DDRAM cuando un carácter es escrito o leído desde DDRAM, osea que se mueve la posición a la derecha. |
| **I/D =** 0 | Decrementa uno la dirección de la DDRAM, osea que se mueve la posición a la izquierda. |
| **S = 1** | La pantalla se desplaza en el sentido que se encuentre el Bit I/D a la derecha cuando es cero y a la izquierda cuando I/D es uno, también parecerá como si el cursor no se mueve, pero la pantalla hace. |
| **S =** 0 | La pantalla no se desplaza. |

**Comando de Control de Pantalla *(Display Control):*** Este comando controla los efectos del LCD.

|  |  |
| --- | --- |
| **D = 1** | La pantalla se enciende. |
| **D =** 0 | La pantalla se apaga, los datos permanecen en la DDRAM y se muestran inmediatamente que se encienda la pantalla. |
| **C = 1** | El Cursor se visualiza en la octava línea y la décima línea dependiendo la selección de la fuente a emplear. |
| **C =** 0 | El cursor no se muestra. |
| **B = 1** | El cursor parpadea. |
| **B = 0** | Parpadeo con un cursor rectangular. |

**Comando de Desplazamiento de Cursor & *Pantalla (Cursor and Display Shift):*** Este comando controla los desplazamientos del cursor y de la pantalla.

|  |  |
| --- | --- |
| **R/L = 1** | El cursor se desplaza a la derecha sin escribir. |
| **R/L =** 0 | El cursor se desplaza a la izquierda sin escribir. |
| **S/C = 1** | El efecto de desplazamiento es aplicado sobre toda la pantalla, el efecto es horizontal no se aplica verticalmente. |
| **S/C = 0** | El efecto de desplazamiento se aplica sobre el cursor sin alterar el contenido de la DDRAM, este efecto se emplea para buscar o corregir en la pantalla. |

**Comando de Ajuste de Función *(Function Set):*** Este comando para definir tipo de letra, el número de líneas, así como el largo de los datos.

|  |  |
| --- | --- |
| **DL = 1** | Establece una longitud de la interfaz de datos de 8 bits DB7 a DB0. |
| **DL = 0** | Establece una longitud de la interfaz de datos de 4 bits DB7 a DB4 lo que hace que se deba de recibir dos veces, primero el nibble alto y después el nibble bajo. |
| **N = 1** | LCD de 2 líneas. |
| **N = 0** | LCD de 1 línea. |
| **F = 1** | Fuente de 5X10 puntos (pixeles). |
| **F = 0** | Fuentes de 5X8 puntos. |

**Comando de Ajuste de dirección de la CGRAM *(Set CGROM Address):*** Con este comando se envía un 1 en DB6 y la dirección de la CGROM que se escribir en el LCD.

**Comando de Ajuste de la dirección de la DDRAM *(Set DDRAM Address):*** Con este comando se puede modificar la posición a donde apunta la DDRAM, esto se hace enviando un 1 a DB7 más la dirección a donde se quiera apuntar dentro de la DDRAM.

**Comando** **de Lectura de la bandera de ocupado (*Read Busy Flag*):** Enviando un 1 en R/W se lee BF y también se lee l dirección DDRAM donde apunta.

# Código C Microchip Studio: Paquetes 4 bits

Dentro del código C de Microchip Studio solo se pueden manejar 6 pines por cada puerto (**PuertoB**, **PuertoC** y **PuertoD**) excepto por el puerto **D** que, si puede manejar 8 puertos, ya que el Arduino UNO solo cuenta con estos puertos, entonces hay un problema ya que para mostrar un caracter en el LCD la instrucción de datos que se debe mandar es de 8 bits, además para hacer el manejo de la configuración del LCD se deben utilizar los 3 **pines de control** **RS**, **R/W** y **E**. Aunque podemos prescindir de usar R/W ya que solo se quiere escribir en el LCD aun así se deben usar todavía 2 pines todavía, por lo que se debería tener que mandar una instrucción de 10 pines. Debido a esta situación, las **instrucciones de datos** (caracteres), se manda en dos partes de 4 bits llamados **nibbles**: **nibble alto** (conjunto de 4 bits de la izquierda) y **nibble bajo** (conjunto de 4 bits de la derecha).

Por lo tanto, para realizar la conversión de bits de caracteres o comandos de 8 bits que se quiere mandar al LCD (4 bits de datos y 2 de control) se debe hacer uso de máscaras AND y OR dentro del código C para que estas operaciones tengan sentido.

En el siguiente ejemplo se pretende introducir el caracter O (ASCII o mayúscula) que corresponde a la dirección 0100 1111 de la memoria CGROM (**columna 5**: **0100**, **fila 8**: **1111**), 4F en hexadecimal o 79 en decimal (vistos en las tablas 1.2.1 y 1.2.2).

A esta dirección de memoria se aplican las siguientes máscaras AND para obtener el nibble alto y bajo y poder de esta manera mandar esa dirección de 8 bits en 2 paquetes de 4bits:

Gráfico

Descripción generada automáticamente

* **Máscara AND**: Es una máscara que al aplicarse a un número binario cualquiera:
  + **Donde tenga 1**: *Deja pasar el número tal cual como viene originalmente*.
  + **Donde tenga 0**: Convierte el número original a 0.
    - **1111 0000 = F0 hexadecimal**: Con esta máscara **se obtiene el nibble alto** y lo demás se queda como 0. *Esta máscara se aplica primero.*

Tabla

Descripción generada automáticamente

Esta máscara se aplica a los 8 bits de la dirección perteneciente al carácter de la memoria CGROM de caracteres ASCII que se quiere mostrar en la pantalla del LCD, esto para primero mandar el **nibble alto** (los primeros 4 bits vistos de izquierda a derecha) y después el **nibble bajo** (los últimos 4 bits vistos de izquierda a derecha), ya que el Arduino UNO en sus puertos solo cuenta con 6 pines y de esta manera se pueden mandar 4 bits de datos y 2 de control **E** y **RS**, ya que el pin **R/W** del LCD se manda directo a tierra en la conexión del circuito.

|  |  |  |
| --- | --- | --- |
| Nombre Pin | Definición | Valores Lógicos Para Mandar |
| RS | Selección de Registro | RS = 0; **Comandos.**  RS = 1; **Datos.** |
| R/W | Lectura / Escritura | R/W = 0; **Escribir en el LCD.**  R/W = 1; **Leer en el LCD.** |
| E | Habilitación del LCD | E = 0; **Inhabilitado.**  E = 1; **Habilitado.** |

Ahora dentro del código C se tiene que considerar que el número binario obtenido de la máscara AND sigue siendo de 8 bits, por lo que, aunque se anuló el **nibble bajo** (se volvieron todos sus bits 0) y se quedó solo con el **nibble alto** (sus bits se quedaron como estaban) utilizando la **máscara AND**, a los 6 bits del puerto le faltan los 2 bits de control **E** y **RS**, por lo que se debe usar una herramienta del código en lenguaje C para arreglar esta situación:

* **Left shift b << i:** Esta instrucción recorre i lugares los bits pertenecientes al número binario b hacia la **izquierda**.
* **Right shift b>>i:** Esta instrucción recorre i lugares los bits pertenecientes al número binario b hacia la **derecha**.

Calendario

Descripción generada automáticamente con confianza media

Por esta situación es que al **nibble alto** se le debe aplicar la instrucción **Right shift >>** y mover sus 8 bits 2 lugares a la derecha después de haber aplicado la **máscara AND** para que el **nibble de datos** pueda entrar en los 4 bits más significativos del puerto del Arduino que indican los datos, ya que los 2 bits menos significativos son utilizados para manejar los **bits de control** del LCD.

Imagen que contiene Diagrama

Descripción generada automáticamente

Para el **nibble bajo** lo que se hace es repetir el mismo proceso, pero primero se mueve el número binario original dos lugares hacia la izquierda con el comando de **Left shift** y luego se aplica una **máscara AND** de la siguiente manera:

* + **0011 1100 = 3C hexadecimal**: Con esta máscara se obtiene el **nibble bajo** después de haber usado el comando **Left shift <<** para mover dos lugares los bits del número binario original, esto se hace de esta manera para que se pueda manejar de la misma forma como se manejó el **nibble alto** anteriormente.

Diagrama

Descripción generada automáticamente con confianza baja

## Declaración de Caracteres Personalizados en Microchip Studio

Cuando se quiera crear caracteres especiales se debe mandar un comando a las siguientes direcciones de la tabla 1.2.2:

1. 0**1**00 0000 **= 0x40**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 0.
2. 0**1**00 **1**000 **= 0x48**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 1.
3. 0**1**0**1** 0000 = **0x50**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 2.
4. 0**1**0**1** **1**000 = **0x58**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 3.
5. 0**11**0 0000 = **0x60**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 4.
6. 0**11**0 **1**000 = **0x68**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 5.
7. 0**111** 0000 = **0x70**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 6.
8. 0**111** **1**000 = **0x78**, dirección de 8 bits en la matriz CGROM para el caracter personalizado 7.

Debo mandar 8 filas de 5 pixeles a las direcciones antes descritas de la memoria CGROM del LCD, los pixeles donde quiera que se muestre de color negro se ponen con un 1 lógico y los que no con 0 lógico y todo debe estar incluido dentro de una librería personalizada si es que se quiere reutilizar en códigos posteriores:

Imagen que contiene biombo, edificio

Descripción generada automáticamente

Cuando se quiera utilizar una librería propia para usarse al ejecutar el código de control de un LCD se deben seguir los siguientes pasos para crear un archivo #include con extensión .h y que pueda ser importado: #include "..forma/de/llegar/al/archivo/nombre librería personalizada.h"

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

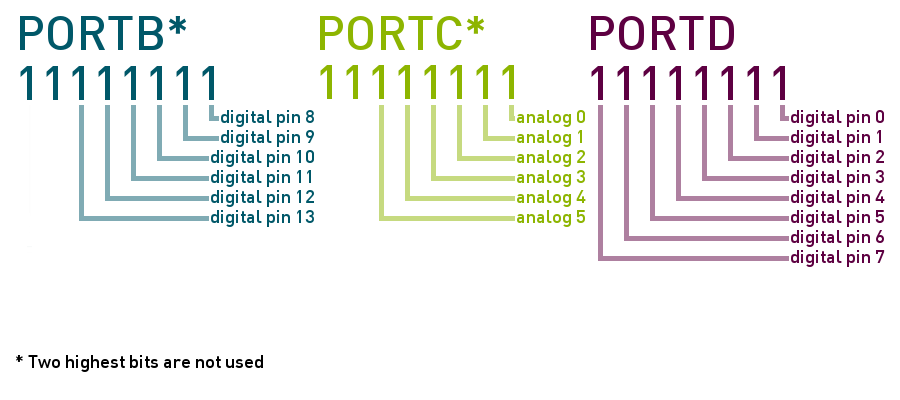
Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

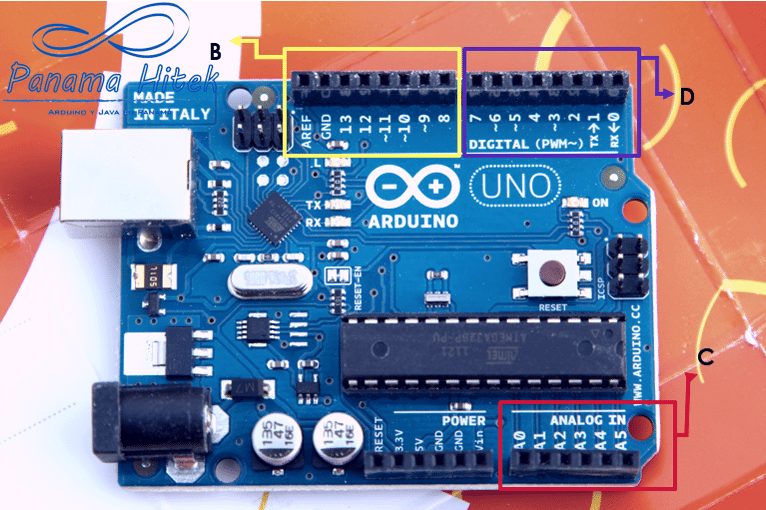
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# Puertos B, C y D Físicos en el Arduino

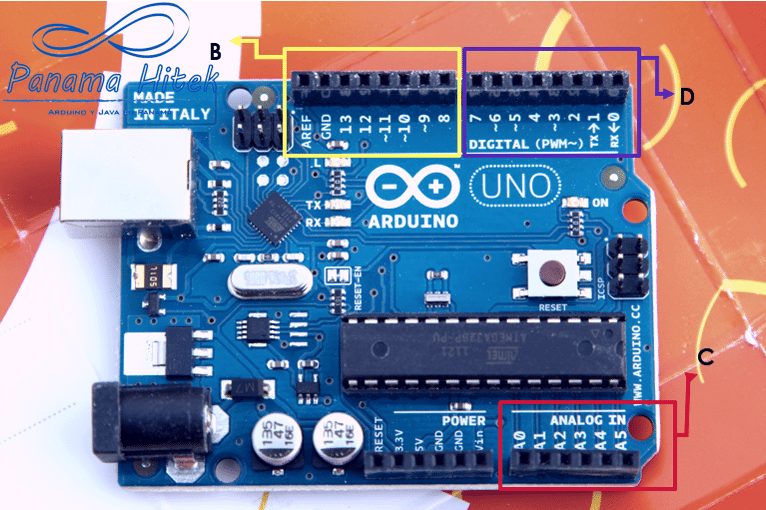


Los pines de mayor peso de los puertos **B** y **C** no se usan, por lo cual en los puertos **B** y **C** existen **6 pines** y en el puerto **D** existen **8 pines**, los pines de cada una de las 3 familias de puertos de la placa Arduino UNO se muestran en la siguiente imagen.



## Conexión Física del LCD con la Placa Arduino UNO

Los **6 pines** de las 3 familias de puertos de la placa Arduino UNO: **B**, **C** y **D** se usarán para mandar los 6 bits de datos y control hacia los pines del LCD.

 Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Debemos tener en cuenta que cuando se use el modo de 4 bits, mandando los 8 bits de comandos y/o datos en dos paquetes de 2 nibbles (4 bits), usando así 6 pines de los puertos de la placa Arduino, solo se deben conectar los **4 pines DB4, DB5, DB6 y DB7 del Data Bus del LCD** hacia el **Puerto del Arduino** que se esté utilizando para programarlo, los demás pines del LCD no se conectan a nada, se dejan al aire tal cual como están o si se quiere, se pueden conectar a tierra.

Tabla 1.1 Pines del LCD

|  |  |  |  |
| --- | --- | --- | --- |
| **Patita** | **Nombre** | **Descripción** | **Valores que se emplean** |
| 1 | **VSS** | Tierra | **GND = 0 Volts = 0 Lógico = 0.** |
| 2 | **VDD** | Fuente de Voltaje | **Vcc = 5 Volts = 1 Lógico = 1.** |
| 3 | **VO** | Ajuste de Contraste | Máximo Contraste Cuando:  **VO = GND.** |
| 4 | **RS** | Selección de Registro | RS = 0; **Comandos.**  RS = 1; **Datos.** |
| 5 | **R/W** | Lectura / Escritura | R/W = 0; **Escribir en el LCD.**  R/W = 1; **Leer en el LCD.** |
| 6 | **E** | Habilitación del LCD | E = 0; **Inhabilitado.**  E = 1; **Habilitado.** |
| 7 – 14 | **DB0 - DB7** | Data Bus | Modo 8 Bits = Se emplea usando todo el Bus.  Modo 4 Bits = Usa solo el nibble alto del Bus. |
| 1. - 16 | **A - k** | Ánodo y Cátodo | Iluminación trasera del LCD *(opcional).*  Poner una resistencia en serie de 20 a 100 Ω. |

**Ánodo =** Terminal positiva (+).

**Cátodo =** Terminal negativa (-).

El potenciómetro conectado al pin **VO** se incluye para que se pueda ajustar el contraste de la pantalla del LCD y de esta manera se vea de la mejor manera el texto que aparece.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

## Programar Físicamente la Placa Arduino UNO con el Programa de Microchip Studio

Recordemos que la forma de programar la placa del Arduino UNO, ya habiendo realizado la configuración de su programador en el programa de Microchip Studio es seleccionando la opción de:

**Tools → Arduino programmer (opción creada y nombrada cuando se configuró el Arduino).**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Es muy importante mencionar que cuando se carga un código a una placa Arduino, debemos tener cuidado que no haya nada conectado a los pines **TX** y **RX** del **Puerto** **D** porque si no Microchip Studio indicará un error cuando tratemos de subir el programa, ya habiéndolo subido podemos utilizar esos pines de nuevo y no habrá problema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente