MICROS 32 BITS
STM - GPIO

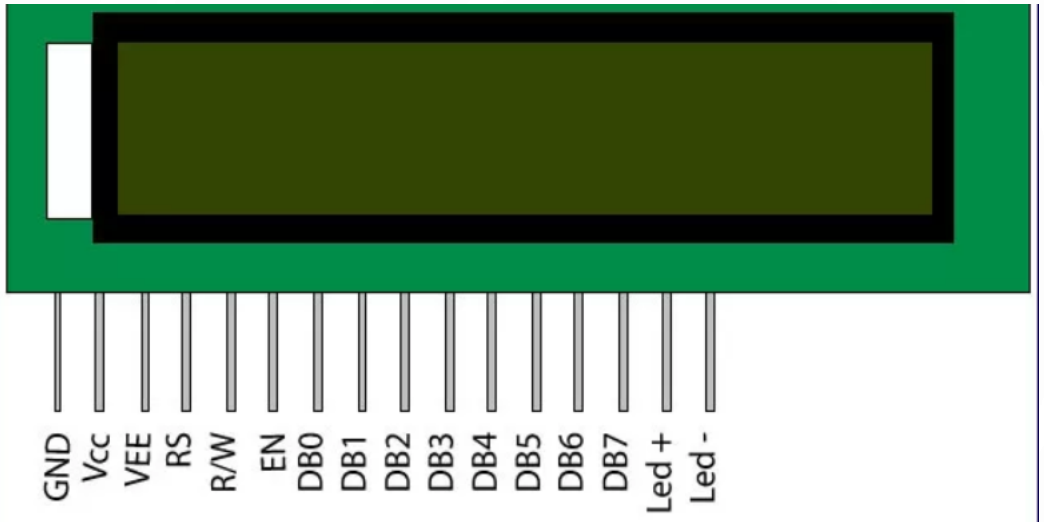
ROBINSON JIMENEZ MORENO

LCD

La utilización de elementos de salida como los display de cristal líquido se usan convencionalmente en muchos circuitos electrónicos los cuales requieren entregar información a un usuario de forma numérica, alfabética, alfanumérica o gráfica, los encontramos en relojes, calculadoras, celulares, termómetros digitales y un gran número de aplicaciones más. Su uso requiere la comprensión básica de su forma de operación, que asemeja mucho a una memoria, en primer lugar requiere tiempos mínimos de habilitación, puesta y lectura de datos, así como manejo de la tabla de configuración la cual suele tener una estructura como la siguiente:

Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clock-Cycles
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clear display & set address counter to zero	165
Cursor Home	0	0	0	0	0	0	0	0	1	x	Set adress counter to zero, return shifted display to original position. DD RAM contents remains unchanged.	3
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direction (I/D) and specify automatic display shift (S).	3
Display Control	0	0	0	0	0	0	1	D	C	B	Turn display (D), cursor on/off (C), and cursor blinking (B).	3
Cursor / Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Shift display or move cursor (S/C) and specify direction (R/L).	3
Function Set	0	0	0	0	1	DL	N	F	x	x	Set interface data width (DL), number of display lines (N) and character font (F).	3
Set CGRAM Address	0	0	0	1	CGRAM Address						Set CGRAM address. CGRAM data is sent afterwards.	3
Set DDRAM Address	0	0	1	DDRAM Address							Set DDRAM address. DDRAM data is sent afterwards.	3
Busy Flag & Address	0	1	BF	Address Counter							Read busy flag (BF) and address counter	0
Write Data	1	0	Data								Write data into DDRAM or CGRAM	3
Read Data	1	1	Data								Read data from DDRAM or CGRAM	3

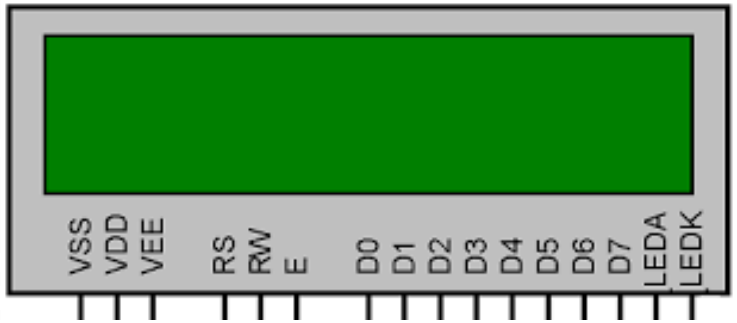
x : Don't care						
	I/D	1 0	Increment Decrement	R/L	1 0	Shift to the right Shift to the left
	S	1 0	Automatic display shift	DL	1 0	8 bit interface 4 bit interface
	D	1 0	Display ON Display OFF	N	1 0	2 lines 1 line
	C	1 0	Cursor ON Cursor OFF	F	1 0	5x10 dots 5x7 dots
	B	1 0	Cursor blinking	DDRAM : Display Data RAM CGRAM : Character Generator RAM		
	S/C	1 0	Display shift Cursor move			



Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

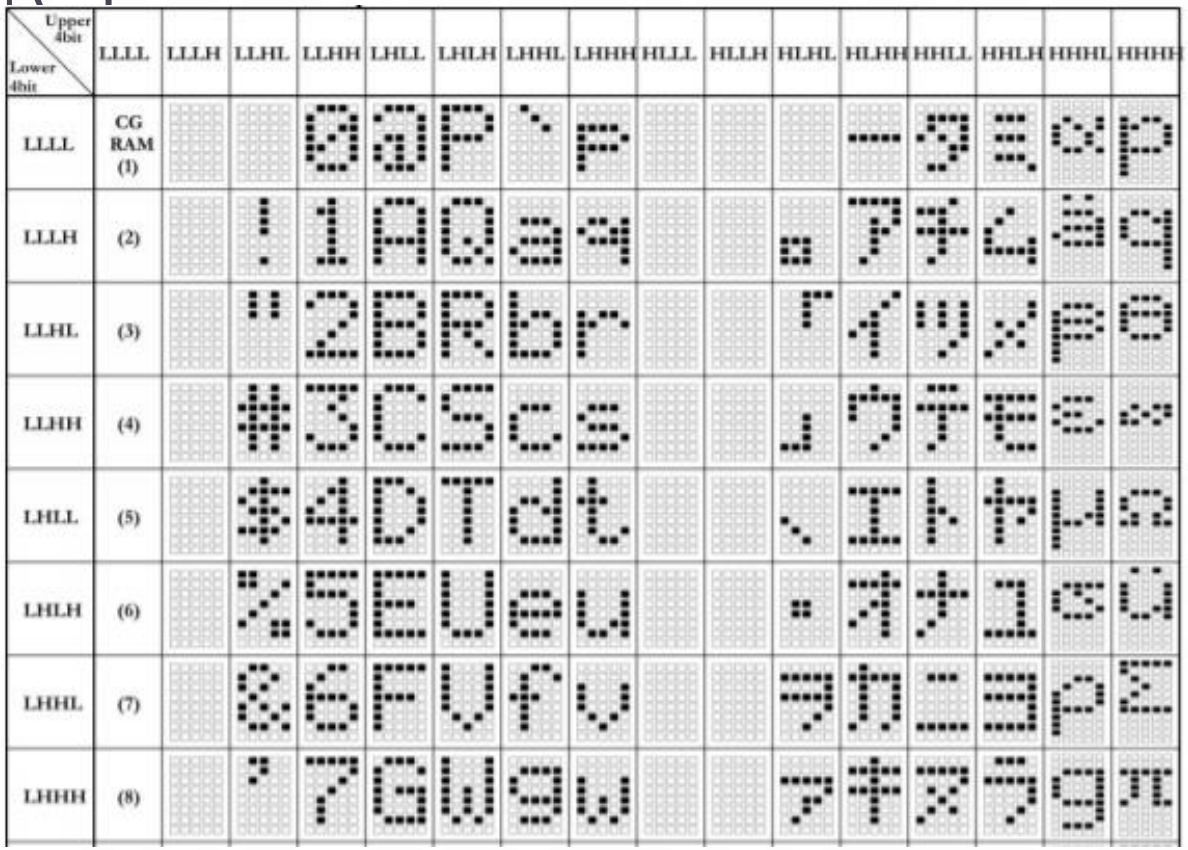
Para el uso adecuado de esta, en el programa a realizar se debe manejar retardos que retengan el dato enviado al LCD para que este pueda ingresar la información adecuadamente, ya que debida la velocidad de operación del micro que suele ser muy superior, es posible que el dato cargado en el puerto no logre ser validado.

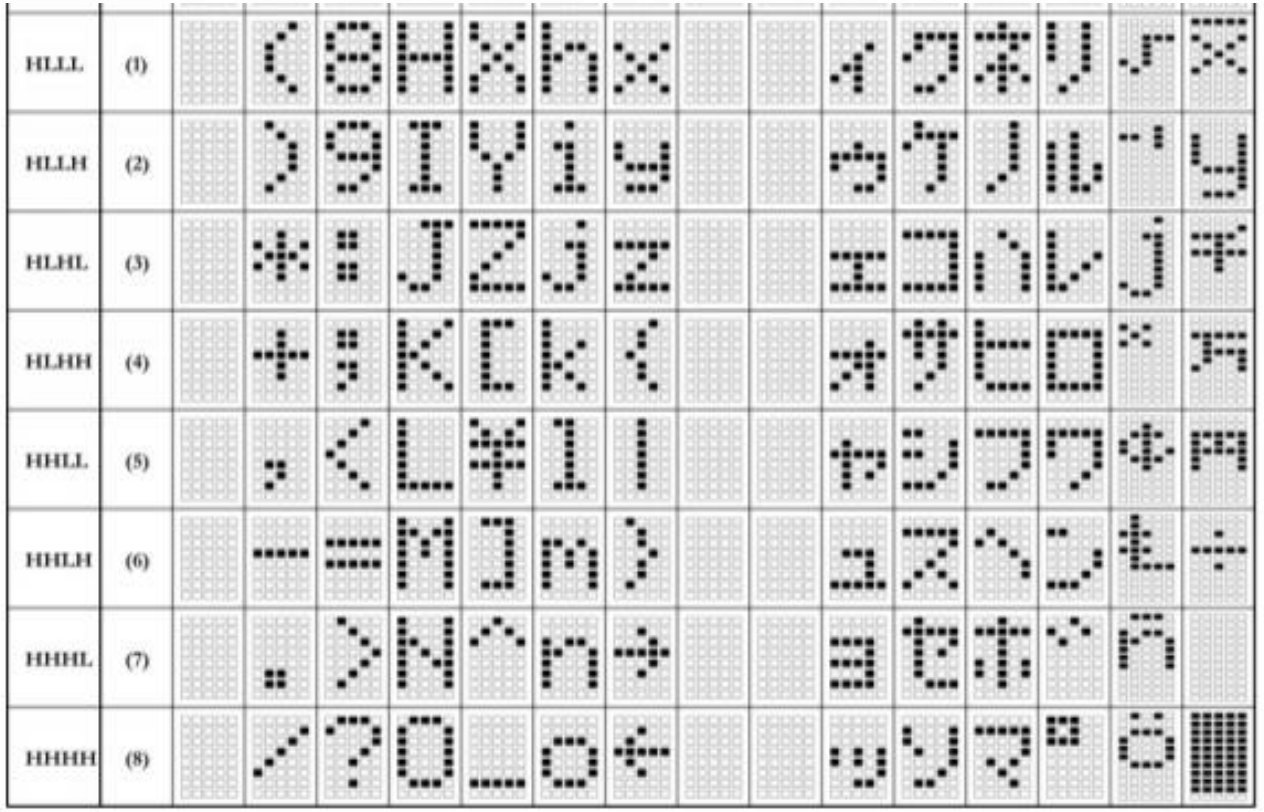
Los tiempos de escritura del LCD son los de habilitación (pin e por enable), tiempo mínimo para selección de requerimiento en el pin de indicación de dato o comando (pin RS) y el de cambio de modo lectura a escritura (pin RW).



DISPLAY CHARACTER ADDRESS CODE:

Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F





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)

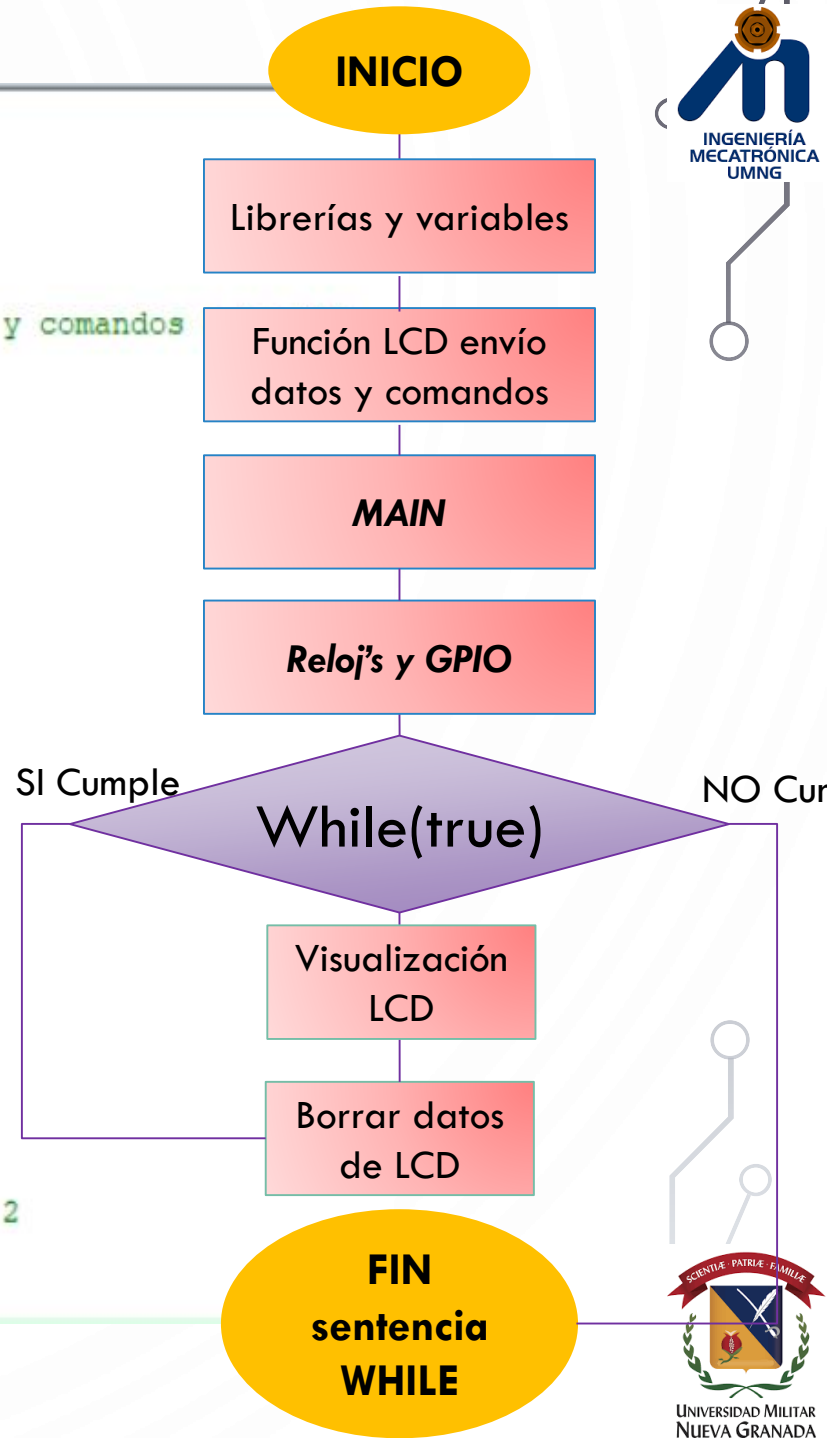
Notes:

1. Character code bits 0 to 2 correspond to CGRAM address bits 3 to 5 (3 bits: 8 types).
2. CGRAM address bits 0 to 2 designate the character pattern line position. The 8th line is the cursor position

Parte inicial de la configuración del LCD se debe establecer si se ha de trabajar a una línea o a dos y si se enviaran datos a 4 bits o a 8 bits, el modo de operación del modulo se inicia activando el pin enable del LCD, posteriormente se debe ajustar el pin RW a cero al igual que el de RS para indicar que se enviaran comandos y no datos de visualización, ahora podemos empezar a enviar la información de configuración (instrucciones), de la tabla validamos que la instrucción a ejecutar.

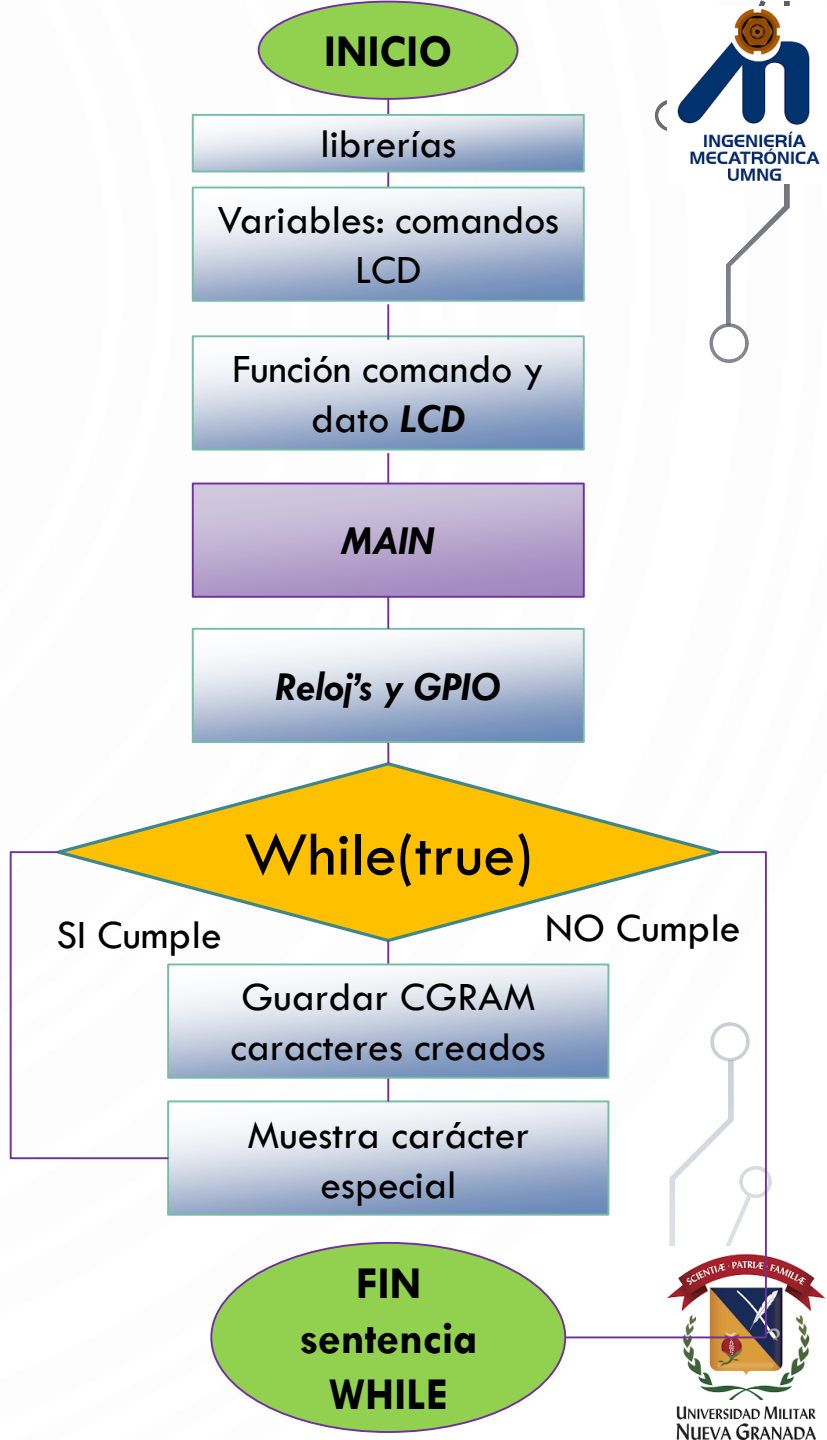
Mediante la opción Fuction Set se valida que el dato a enviar tiene el formato 001 DL N F XX, para operación a 4 bits DL debe valer 0, para dos líneas N debe ser 1 y si lo utilizamos a 5x7 caracteres F debe ser cero, por lo que el dato será: 00101000 en binario o 40 en decimal, podemos proseguir con la instrucción Entry Mode Set para ajustar que la visualización se realice de izquierda a derecha (Incremento) con desplazamiento de cursor indicador, el dato a transmitir será 000001 I/D S, para nuestro ejemplo 00000111 en binario o 7 en decimal y finalmente se activa el display con la instrucción Display control 00001 DCB que para nuestro caso será 00001100 en binario o 12 en decimal.

```c
#include "stm32f4xx.h"                    // Device header
int ix;
int time=50000;
char i[]={'M','I','C','R','O','S'};   //vector de la palabra de visualizacion linea 1
char a[]={'X','C','8'}; //vector de la palabra de visualizacion linea 2
int lcd (int a){  GPIOB->ODR |=(1UL<<9); GPIOB->ODR |=a;  //función para envió da datos y comandos
for(ix=0;ix<time;ix++);GPIOB->ODR |=(0UL<<9);
for(ix=0;ix<time;ix++);}
int main (void)
{
  RCC->AHB1ENR |=(1UL<<1);
  GPIOB->MODER=0X5555;
  GPIOB->OTYPER=0;
  GPIOB->OSPEEDR=0X10004001;
  GPIOB->PUPDR=0X10004001;

int c,j=0;
GPIOB->ODR |=(0UL<<10);
 lcd(0x38);lcd(0x06);lcd(0x0C);  //inicialización de la LCD a 2 líneas y 8 bits
 while(1){  GPIOB->ODR |=(1UL<<9);
GPIOB->ODR |=(0UL<<10); lcd(0x85);//posición de escritura primera fila columna 5
 for (j=0;j<6;j++){GPIOB->ODR |=(1UL<<10);
   lcd(i[j]);}     //visualización de la línea 1
 for(ix=0;ix<time;ix++);GPIOB->ODR |=(1UL<<9);
GPIOB->ODR |=(0UL<<10); lcd(0xc5);//posición de escritura segunda fila columna 5
 for (j=0;j<=3;j++){
GPIOB->ODR |=(1UL<<10);lcd(a[j]); for(ix=0;ix<time;ix++);}  //visualización de la línea 2
GPIOB->ODR |=(0UL<<10);lcd(0x01);//borrado de la LCD
}}
```



EJEMPLO CGRAM

```
int j=0;
int i=0;
int l=0;
int cur=0x80;
int graf=0;
int time=1000;      //50 ciclos de maquina
char conteo=0;
char bandera=0;
char dato;
char envio = 0x31;
char envio1[] = {"DATOS SENSORES ="};

//*****************************************************************************************************************
//COMANDOS LCD
char clear = 0x01;  //0b00000001;
char home = 0x02;   //0b00000010;
char set = 0x3C;    //0b001111XX;  //Bus a 8 bits, LCD 2 lineas, caracter 5x10
char set1 = 0x3C;    //0b001110XX;  //Bus a 8 bits, LCD 2 lineas, caracter 5x10

char disp_on  = 0x0C;  //0b00001100;  //Display ON, cursor OFF, NO parpadeo
char disp_off = 0x08;  //0b00001000;  //Display OFF, cursor OFF, NO parpadeo

char mode_set1 = 0x06;  //0b00000110;  //Incremento del cursor y modo normal
char mode_set2 = 0x04;  //0b00000100;  //Incremento del cursor y desplaza la visual. cada vez que se escribe un dato

char disp_shift  = 0x1C;  //0b00011100;   //desplaza el display -- a la derecha
char disp_shift1 = 0x18;  //0b00011000;   //desplaza el display -- a la izquierda
char disp_shift2 = 0x14;  //0b00010100;   //mueve el cursor -- a la derecha
char disp_shift3 = 0x10;  //0b00010000;   //mueve el cursos -- a la izquierda

char pos_LCD =0;
char w_linea1 = (0x80+ pos_LCD);   //0b1000000 posicion cero primera fila
char w_linea2 = (0xC0 + pos_LCD);  //0b1100000 posicion cero segunda fila
```



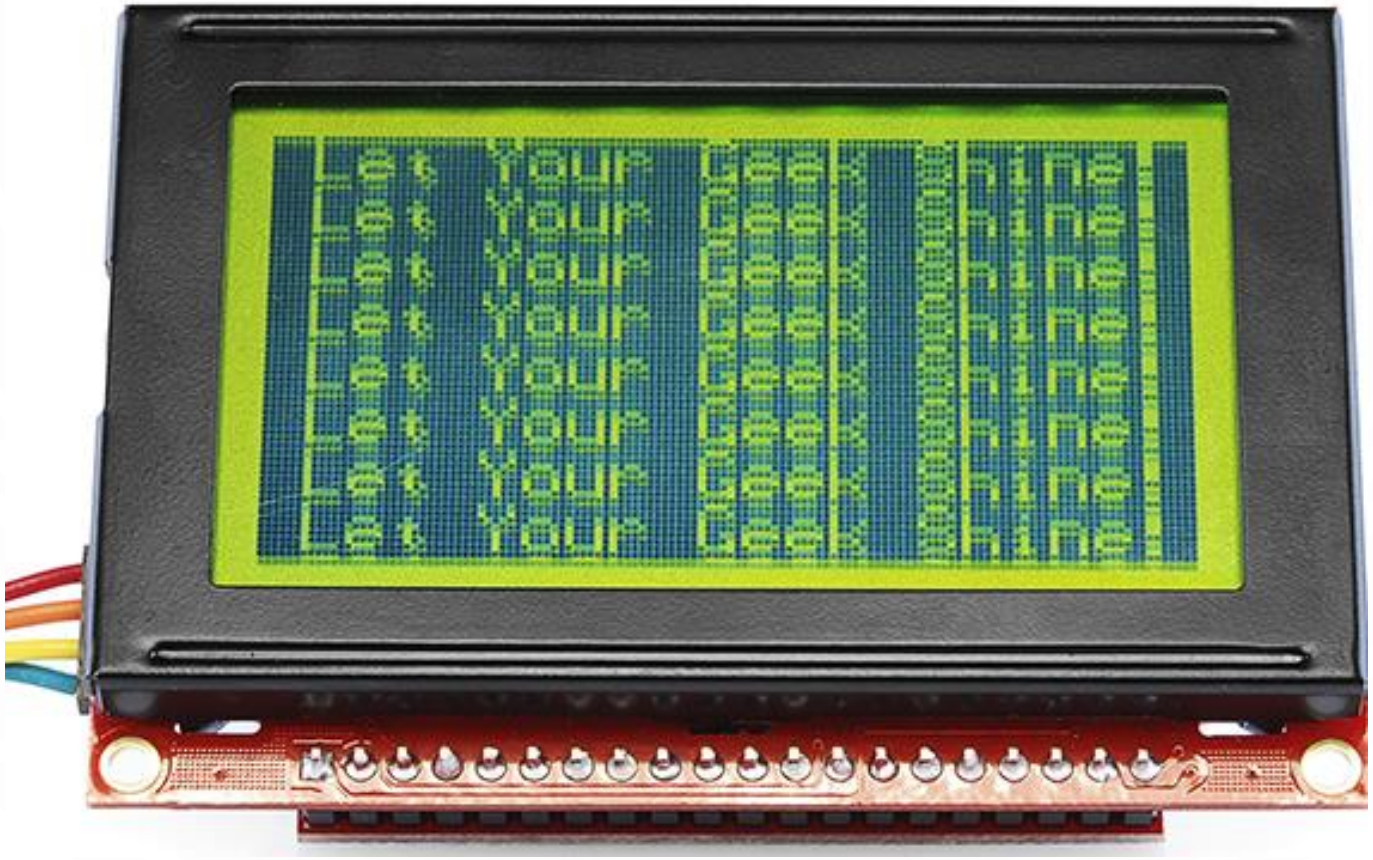
```
char CGRAM [8] = {0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47};  //posiciones para guardar los caracteres personali
char ghost[8] = {0x19,0x1F,0x15,0x1F,0x11,0x1F,0x1D,0xC};


char  matriz_mario [8][8] = { {0x6,0xF,0x1C,0x18,0x18,0x1C,0xF,0x6},
                              {0x0,0x7,0xF,0xC,0xC,0xF,0x7,0x0},
                              {0x1,0x3,0x3,0x3,0x5,0x5,0x5,0x2},
                              {0x1F,0x1F,0x1F,0x19,0x11,0x18,0x1,0x3},
                              {0x10,0x10,0x1F,0x8,0xE,0x11,0x1E,0x1C},
                              {0x1,0x0,0x7,0x9,0x11,0x15,0x15,0x15},
                              {0x0,0x10,0x1F,0x11,0x11,0x1F,0x0,0x0},
                              {0x8,0x10,0x1C,0x12,0x11,0x15,0x15,0x15}
                            };
```

```
void comando_lcd(char b){
  //RS=PB8, Enable=PB9, DATA= PB0(LSB)-PB7(MSB)
  GPIOB->ODR = b;
  GPIOB->ODR &=  ~(1UL <<8);  //RS=0
  GPIOB->ODR |= (1UL << 9);   //Enable = 1
  for(j=0;j<time;j++);        //delay para que el comando sea ejecutado por la LCD  tado por la LCD
  GPIOB->ODR &=  ~(1UL <<9);  //Enable = 0

}

void dato_lcd(char c){
  //RS=PB8, Enable=PB9, DATA= PB0(LSB)-PB7(MSB)
  GPIOB->ODR = c;
  GPIOB->ODR |= (1UL << 8);  //RS=1
  GPIOB->ODR |= (1UL << 9);   //Enable = 1
  for(j=0;j<time;j++);        //delay para que el comando sea ejecutado por la LCD
  GPIOB->ODR &=  ~(1UL <<9);  //Enable = 0

}
```

```c
int main(void)
{
  RCC->AHB1ENR =0xFF; //Puertos A,B,C,D,E,F,G,H

  GPIOC->MODER &=  ~(3UL << 2*13); //pulsador como entrada (PC13)

  //RS=PB8, Enable=PB9, DATA= PB0(LSB)-PB7(MSB)
  GPIOB->MODER =  0x555555;         //Pines del PB0 al PB11 como salida
  GPIOB->OTYPER = 0;
  GPIOB->OSPEEDR = 0x555555;        //medium speed
  GPIOB->PUPDR =  0x555555;        //pull up

  //*****************************************************************************
  //CONFIGURACION SYSTICK
    SystemCoreClockUpdate();
    SysTick_Config(SystemCoreClock);

//*****************************************************************************
    //CONFIGURAR LA LCD
    comando_lcd(clear);
    comando_lcd(home);
    comando_lcd(set1);
    comando_lcd(disp_on);
    comando_lcd(mode_set1);
    comando_lcd(w_linea1);
    dato_lcd('H');
    dato_lcd('O');
    dato_lcd('L');
    dato_lcd('A');
    dato_lcd(':');
```

```
  car_esp();      //Guardar en la CGRAM los 8 caracteres creados

  comando_lcd(0x87);
  dato_lcd(0x2);
  comando_lcd(0x88);
  dato_lcd(0x3);
  comando_lcd(0x89);
  dato_lcd(0x4);
  comando_lcd(0xC7);
  dato_lcd(0x5);
  comando_lcd(0xC8);
  dato_lcd(0x6);
  comando_lcd(0xC9);
  dato_lcd(0x7);

  for(j=0;j<1000000;j++);    //espera para poderlo ver

'*****************************************************************************************
 while(true)
   {

     car_esp();
     for (cur=0x80;cur<0x90;cur++){
       comando_lcd(clear);
       comando_lcd(cur);
       dato_lcd(0x0);              //envia pacman Open
       for(j=0;j<200000;j++);
       comando_lcd(clear);
       comando_lcd(cur);
       dato_lcd(0x1);              //envia pacman close
       for(j=0;j<100000;j++);
     }
     }}
```

LCD GRAFICA



Los bloques que componen una imagen digital, son tratados por sus coordenadas horizontal (X) y verticales (Y), justo lo contrario al sistema de coordenadas cartesianas estándar de las matemáticas, pero es una práctica establecida en muchos sistemas gráficos por ordenador desde los principios de los escáneres gráficos que trabajaban de arriba a abajo.

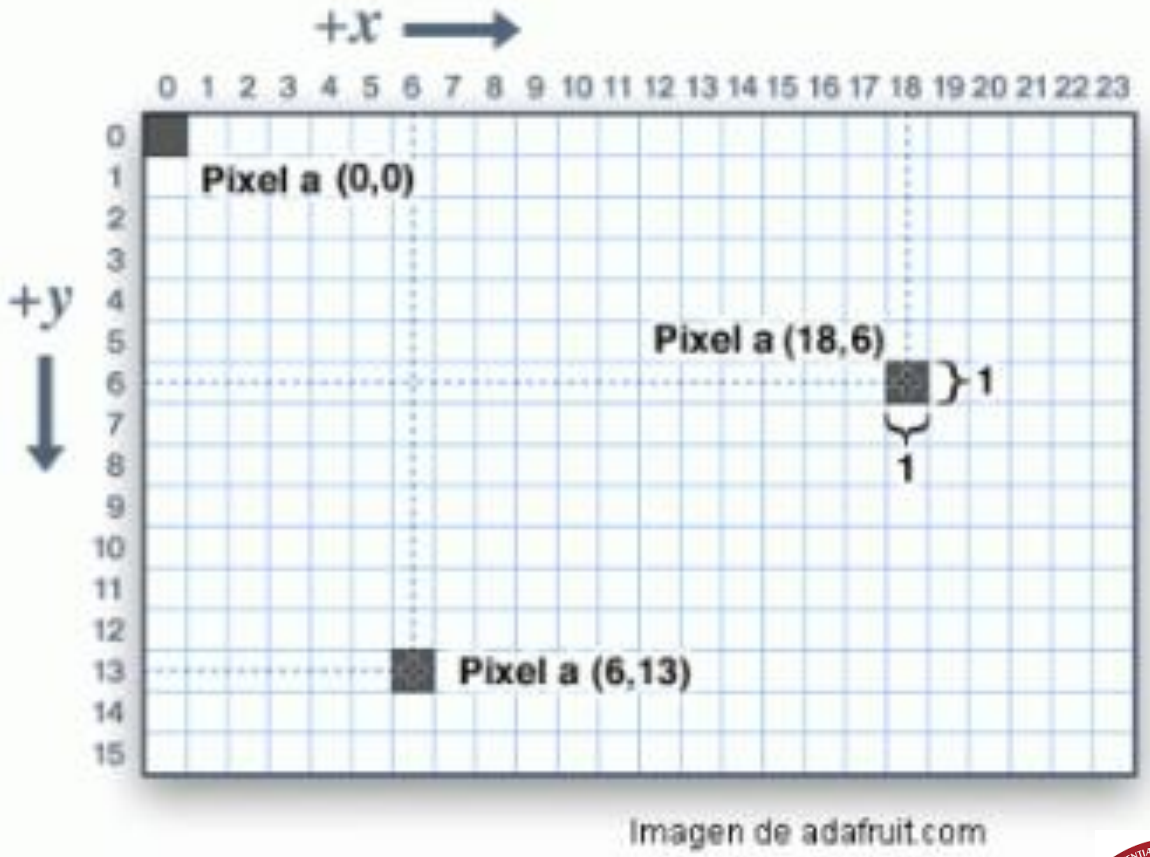
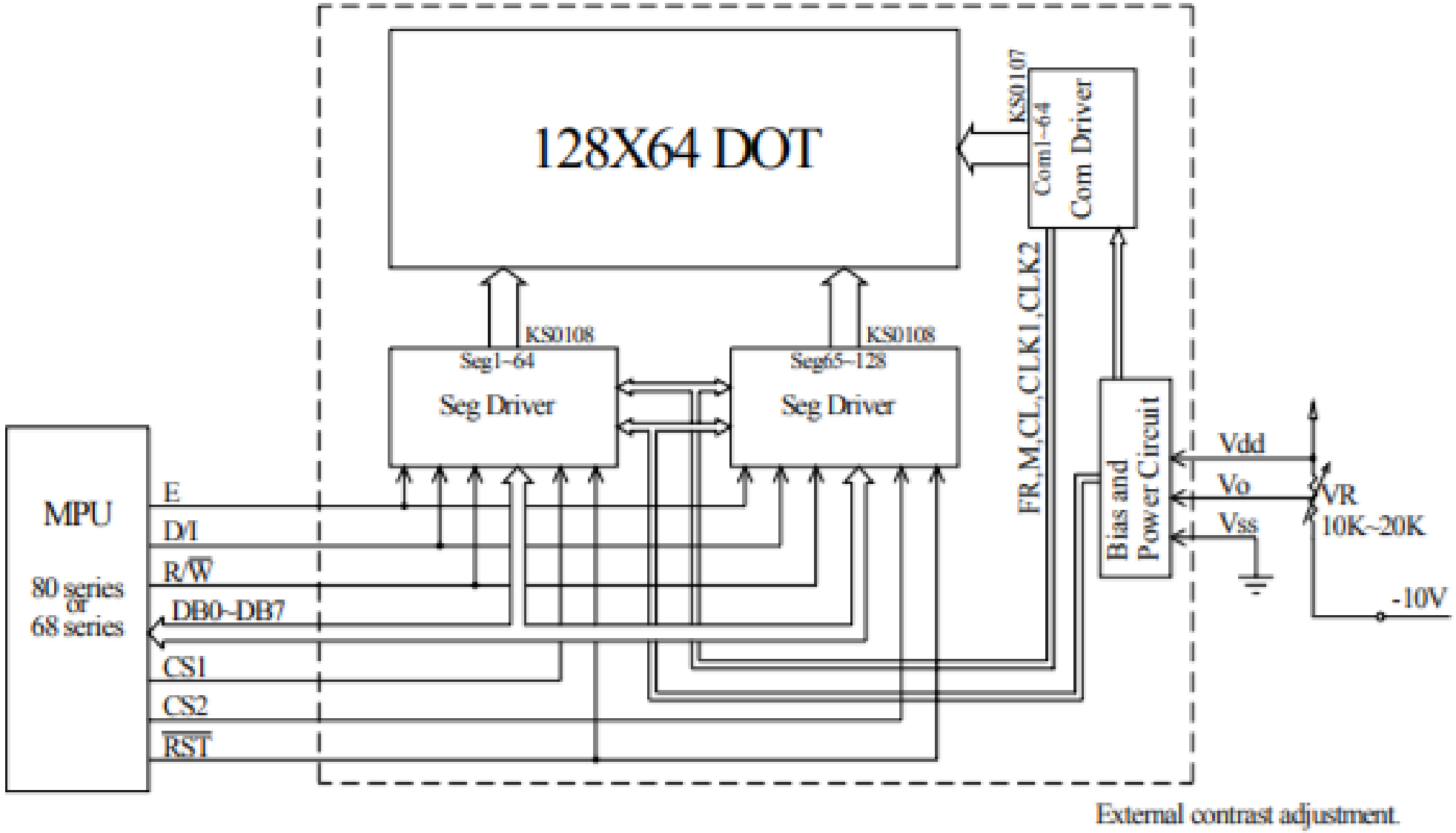


Imagen de adafruit.com


128X64 DOT
KS0107
Com1~64
Com Driver
KS0108
Seg1~64
Seg Driver
KS0108
Seg65~128
Seg Driver
FR,M,CL,CLK1,CLK2
Bias and Power Circuit
Vdd
Vo
Vss
VR
10K~20K
-10V
MPU
80 series
or
68 series
E
D/I
R/W
DB0~DB7
CS1
CS2
RST
External contrast adjustment.

. Display Control Instruction

The display control instructions control the internal state of the KS0108B. Instruction is received from MPU to KS0108B for the display control. The following table shows various instructions

Instruction	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display ON/OFF	0	0	0	0	1	1	1	1	1	0/1	Controls the display on or off. Internal status and display RAM data are not affected. 0:OFF, 1:ON
Set Address	0	0	0	1	Y address (0~63)						Sets the Y address in the Y address counter.
Set Page (X address)	0	0	1	0	1	1	1	Page (0 ~7)			Sets the X address at the X address register.
Display Start Line	0	0	1	1	Display start line(0~63)						Indicates the display data RAM displayed at the top of the screen.
Status Read	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0	Read status. BUSY 0:Ready 1:In operation ON/OFF 0:Display ON 1:Display OFF RESET 0:Normal 1:Reset

arm KEIL

Board Support Version 1.0

Functions available when using the Board Software Componen

General | File System | Graphic | Network | USB | Board Support

Main Page | Usage and Description | Reference

- Board Support
  - Revision History
  - Board Support Examples
  - Reference
    - Interfaces
      - LED
      - Buttons
      - Joystick
      - A/D Converter

Graphic LCD

Interfaces

Graphic LCD Interface. More...

Content

Configuration
Display configuration.

Functions

int32_t **GLCD_Initialize** (void)
Initialize Graphic LCD. More...

int32_t **GLCD_Uninitialize** (void)
De-initialize Graphic LCD. More...

int32_t **GLCD_SetForegroundColor** (uint32_t color)
Set foreground color. More...

int32_t **GLCD_SetBackgroundColor** (uint32_t color)
Set background color. More...

int32_t **GLCD_ClearScreen** (void)
Clear screen (with active background color) More...

int32_t **GLCD_SetFont** (**GLCD_FONT** *font)
Set active font. More...

int32_t **GLCD_DrawPixel** (uint32_t x, uint32_t y)
Draw pixel (in active foreground color) More...

int32_t **GLCD_DrawHLine** (uint32_t x, uint32_t y, uint32_t length)
Draw horizontal line (in active foreground color) More...

int32_t **GLCD_DrawVLine** (uint32_t x, uint32_t y, uint32_t length)
Draw vertical line (in active foreground color) More...

int32_t **GLCD_DrawRectangle** (uint32_t x, uint32_t y, uint32_t width, uint32_t height)
Draw rectangle (in active foreground color) More...

TAREA

Conectar una LCD alfanumérica a la STM, identificando claramente pines de conexionado y mostrando la palabra “buenos” en la primer línea y “días ” en las segunda línea.