MICROS 32 BITS
STM - GPIO

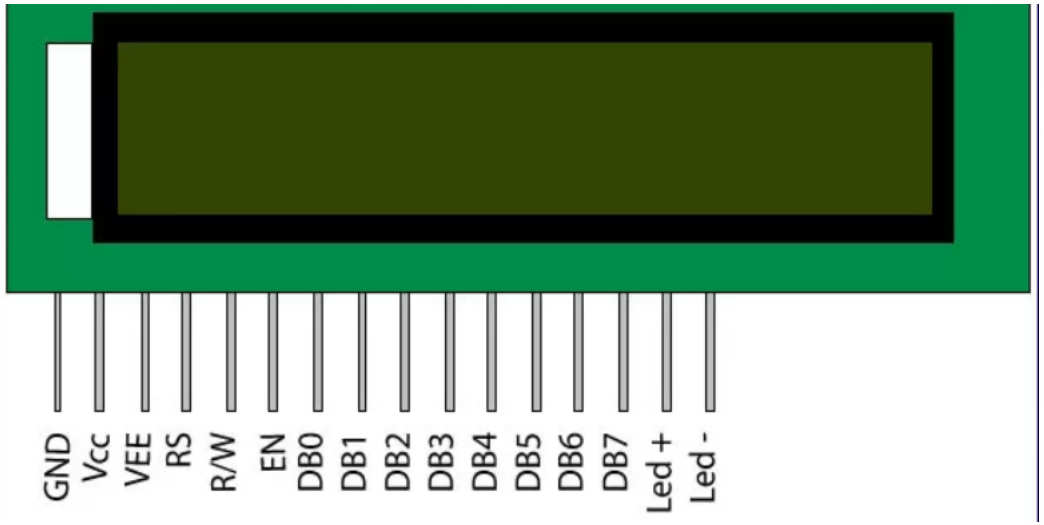
ROBINSON JIMENEZ MORENO – LUISA FERNANDA GARCIA

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

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:



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-

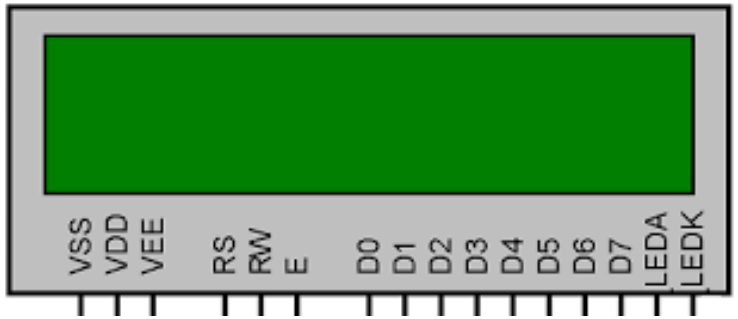
Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clock-Cycles
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clear display & set address counter to zero	165
Cursor Home	0	0	0	0	0	0	0	0	1	x	Set adress counter to zero, return shifted display to original position. DD RAM contents 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			

```
char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char cgram=0x40;
char ddram=0x80;
```

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).



Set DDRAM Address – comando

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

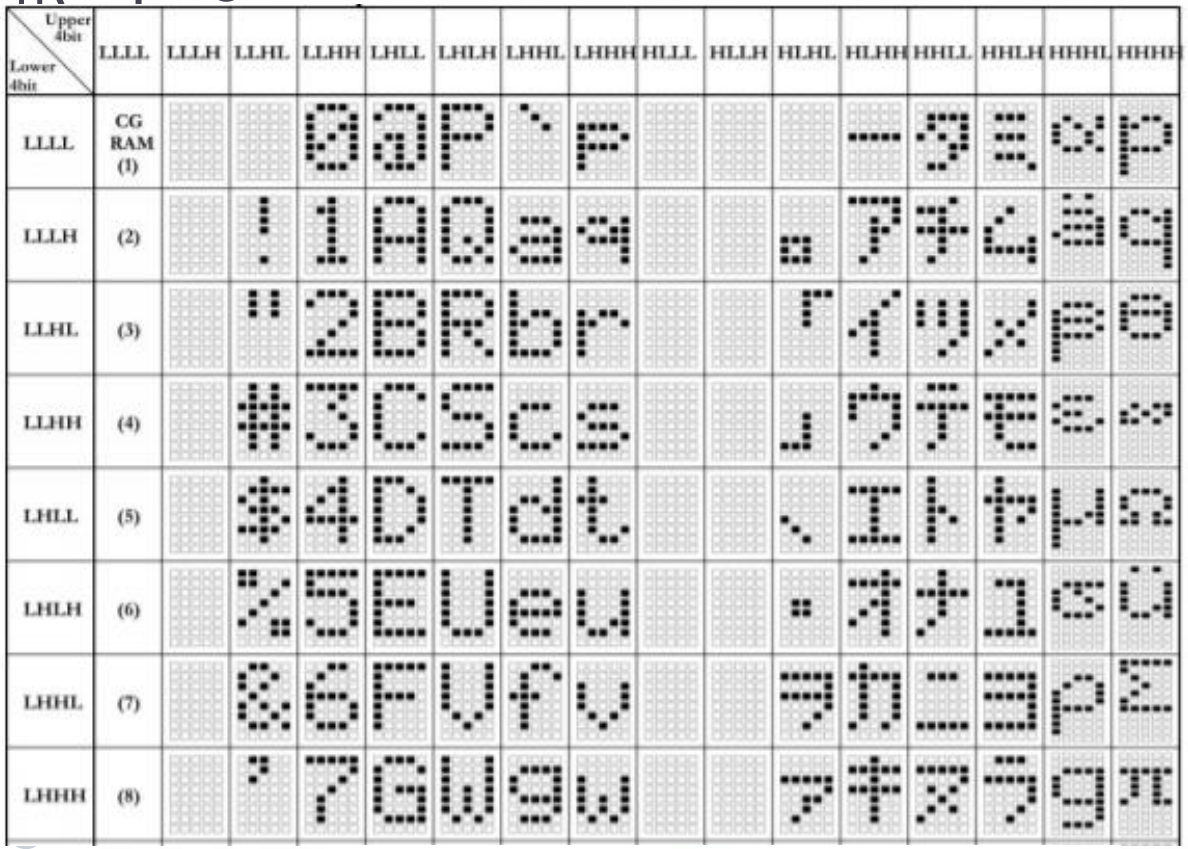
char ddram=0x80;
char ddram=0xC0;

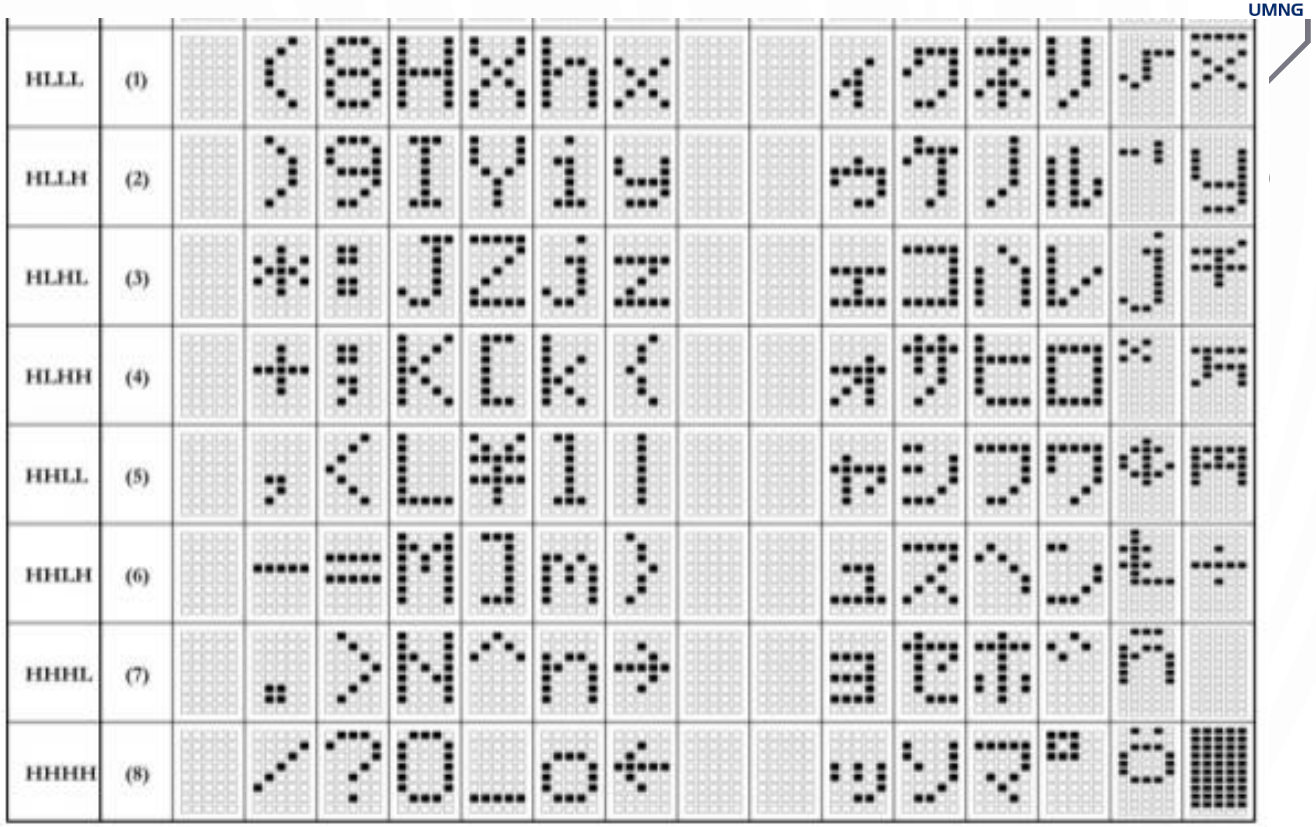
COMANDOS

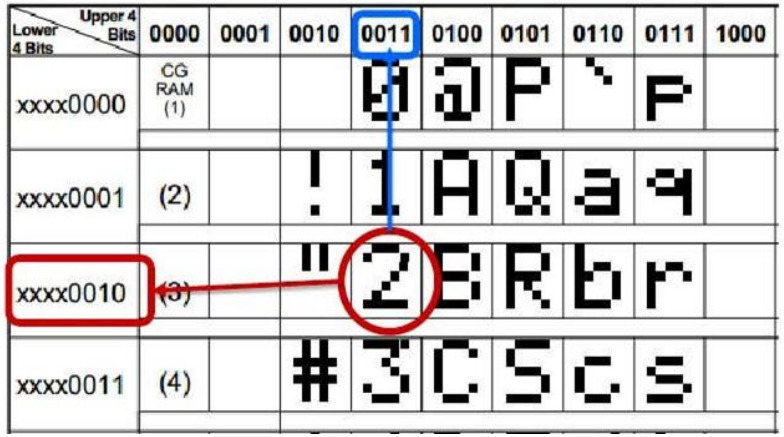
```
//*********Comandos***********\\

char clear = 0x01;//Limpiar pantalla
char home = 0x02;//Return
char mode_set1 = 0x06;//Incremento del cursor y modo normal
char mode_set2 = 0x04;//Incremento del cursor y desplaza la visual cada vez que se le escribe un dato
char disp_on = 0x0E;//Display ON, cursor ON, parpadeo
char disp_off = 0x08;//Display OFF, cursor OFF, no parpadeo
char disp_shift = 0x1C;//Desplaza el display -- a la derecha
char disp_shift1 = 0x18;//Desplaza el display -- a la izquierda
char disp_shift2 = 0x14;//Mueve el cursor -- a la derecha
char disp_shift3 = 0x10;//Mueve el cursor -- a la derecha
char set = 0x3C;//Bus a 8 bits, LCD 2 lineas, caracter 5x10
char pos_LCD=0x0;
char w_linea1=(0x80+pos_LCD);//Posición cero de la primera fila
char w_linea2=(0xC0+pos_LCD);//Posición cero de la segunda fila
```

TABLA ASCII CGRAM-LCD







Números en ASCII 0x30 + #

'2' en ASCII = b'00110010'=0x30 +2

Nota: se debe enviar centenas-decenas-unidades uno a uno.

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.

```
//////////////// dato 2 lineas fijo////////////////

#include "STM32F7xx.h"
char dato[6]={'M','i','c','r','o','s'};
char dato2[4]={'2','0','2','0'};
char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char ddram1L=0x80;
char ddram2L=0xC0;

void send_comando(char a){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= a;
  GPIOG->ODR &=~(1UL<<8);//RS=0
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}

void send_dato(char b){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= b;
  GPIOG->ODR |=(1UL<<8);//RS=1
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}
```

```
int main(void){

  RCC->AHB1ENR = 64;;
  GPIOG->MODER =0x00055555;
  send_comando(clear);
  send_comando(set);
  send_comando(disp_on);
  send_comando(mode);

while(true){
 send_comando(ddram1L+5);
for (int i=0;i<6;i++){
   send_dato(dato[i]); }
 send_comando(ddram2L+7);
for (int i=0;i<4;i++){
   send_dato(dato2[i]);}
}
}
```

EJERCICIO EN CLASE:

Implemente un contador de eventos mediante el pulsador de la tarjeta que visualice un conteo de 000 hasta 999.

Repita el ejercicio anterior cambiando el pulsador por una cuenta automática mediante SysTick.

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

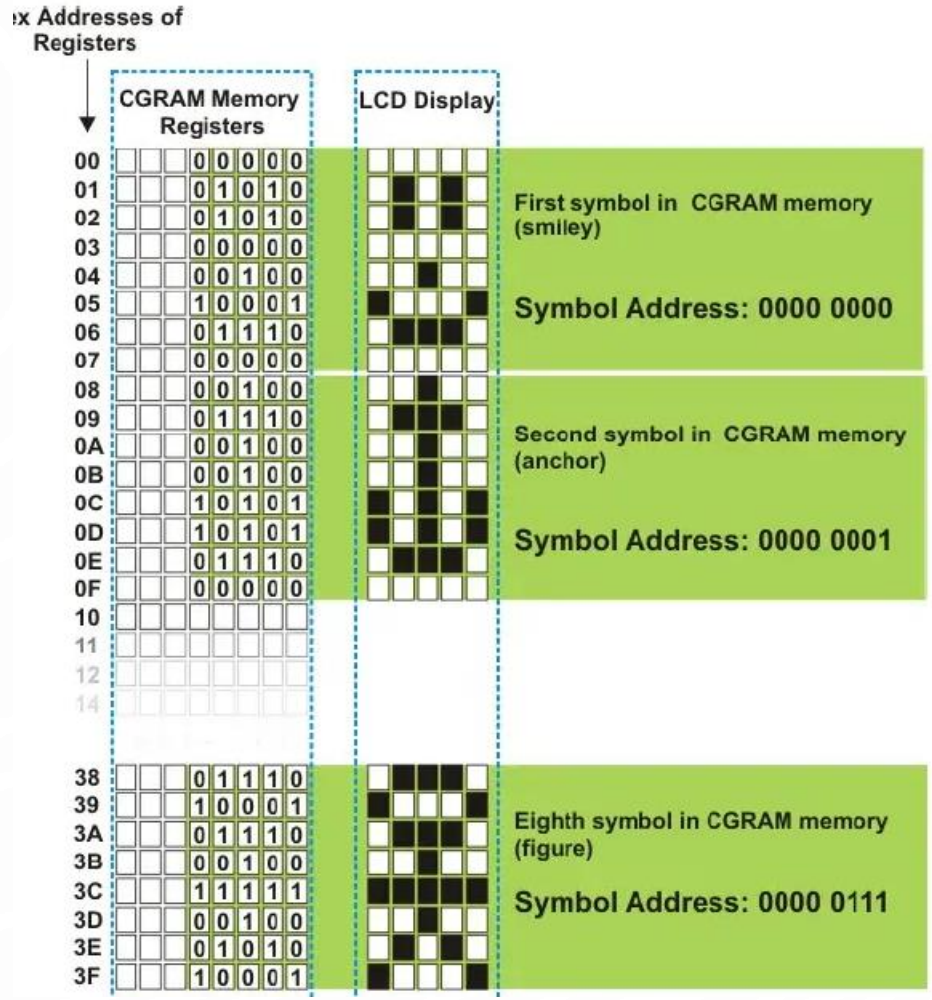
Set CGRAM Address -- comando.

char cgram=0x40;

cada carácter ocupa 8 posiciones, por ejemplo desde la dirección 0 hasta la 7 de la CGRAM

Luego se envía el dato

	CGRAM Data								MATRIZ 5x8					
HEXA	b7	b6	b5	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0	
0x15	0	0	0	1	0	1	0	1	■		■		■	
0x15	0	0	0	1	0	1	0	1	■		■		■	
0x15	0	0	0	1	0	1	0	1	■		■		■	
0x0E	0	0	0	0	1	1	1	0		■	■	■		
0X04	0	0	0	0	0	1	0	0			■			
0X04	0	0	0	0	0	1	0	0			■			
0x0E	0	0	0	0	1	1	1	0		■	■	■		
0X04	0	0	0	0	0	1	0	0			■			



https://www.academia.edu/10937574/LCD_funcionamiento_del_display_de_16x2

```
char TRIDENTE [8]={0X15,0X15,0X15,0X0E,0X04,0X04,0X0E,0X04};
```

```c
#include "STM32F7xx.h"

int Cont=0,time1=10000,fila=0,columna=0;
char TRIDENTE [8]={0X15,0X15,0X15,0X0E,0X04,0X04,0X0E,0X04};
char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char cgram=0x40;
char ddram=0x80;

void send_comando(char a){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= a;
  GPIOG->ODR &=~(1UL<<8);//RS=0
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(Cont=0;Cont<time1;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}

void send_dato(char b){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= b;
  GPIOG->ODR |=(1UL<<8);//RS=1
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(Cont=0;Cont<time1;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}
```

```
int main(void){

   RCC->AHB1ENR = 64;;
   GPIOG->MODER =0x00055555;
   send_comando(cgram);//LLENADO DE CARACTERES
   for(columna=0;columna<8;columna++){
   send_dato(TRIDENTE[columna]);}
   send_comando(clear);
   send_comando(set);
   send_comando(disp_on);
   send_comando(mode);
   send_comando(0XC);// FUNCION PARA APAGAR EL CURSOR
   //
while(true){
   for(int as=0;as<16;as++){send_comando(clear);send_comando(ddram+as);
   send_dato(0X0); for(int tm=0;tm<800000;tm++);}
}
}
```

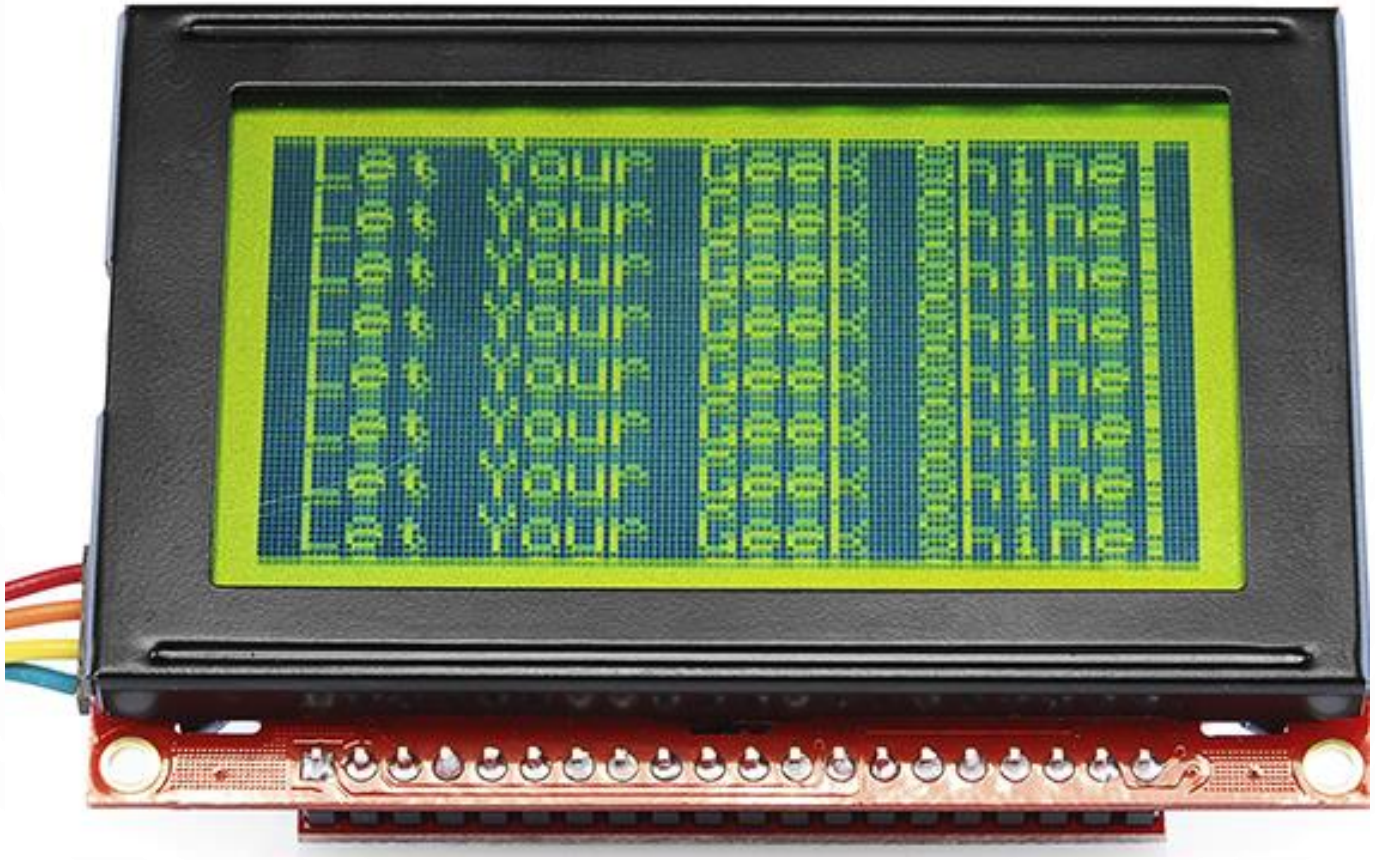
INGENIERÍA MECATRÓNICA UMNG

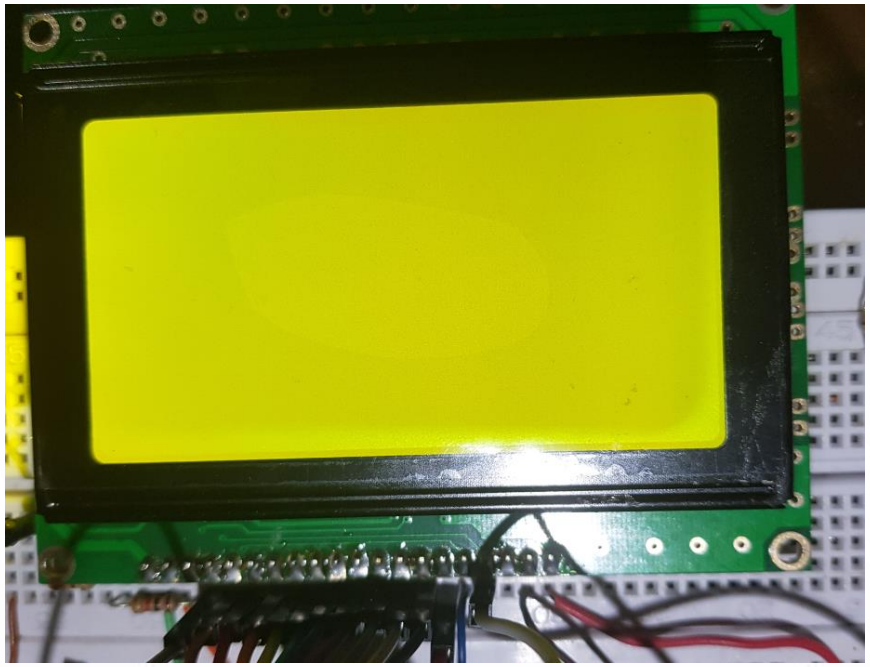
EJERCICIO EN CLASE:

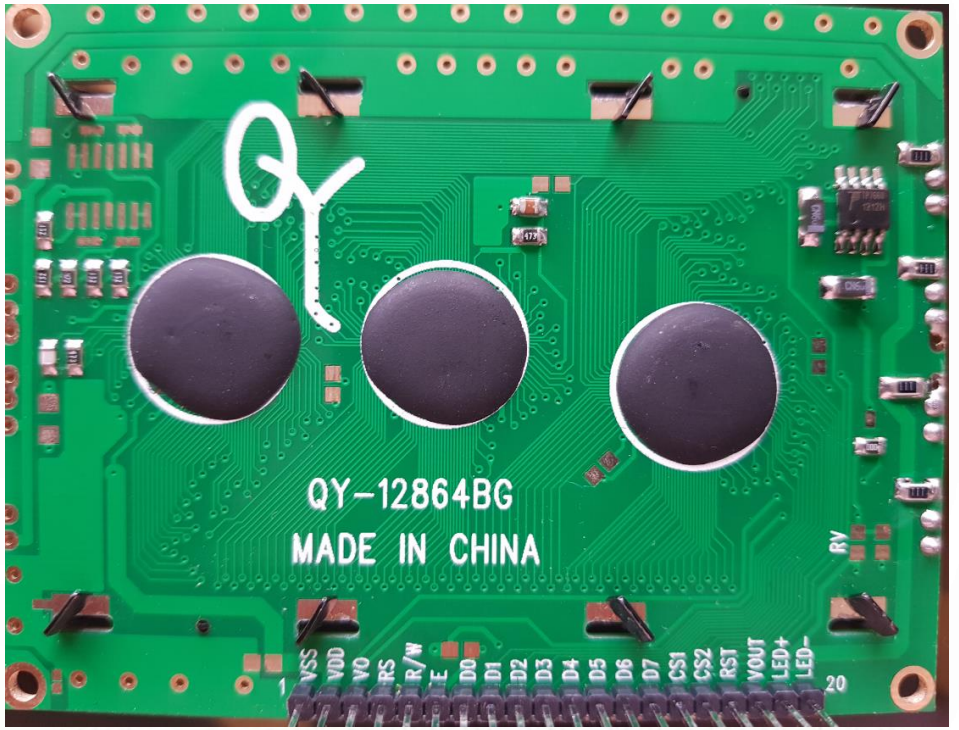
Implementen una imagen puede ser un animal, una flor, un logo de superhéroe, lo que prefieran, que se desplace de un lado al otro en la pantalla de la LCD, centrado en una fila.

UNIVERSIDAD MILITAR NUEVA GRANADA

LCD GRAFICA







Pin Configuration		
Pin	Simbol	Description
1	Vss	Power Supply (Vss =0)
2	VDD	Power Supply (VDD=5V±10%), 5.5V maximum(25°C)
3	Vo	Operating voltage for LCD(variable)
4	D/I	H:Data L:Instruction Code
5	R/W	Read / Write
6	E	Enable trigger
7..14	DB0..DB7	Data bus
15	CS1	Chip Select 1
16	CS2	Chip Select 2
17	$\overline{\text{RESET}}$	Controller reset
18	VEE	Supply voltage for LCD Drive (VDD-VEE=15V Maximum)
19	LED_A	Backlight anode
20	LED_K	Backlight cathode

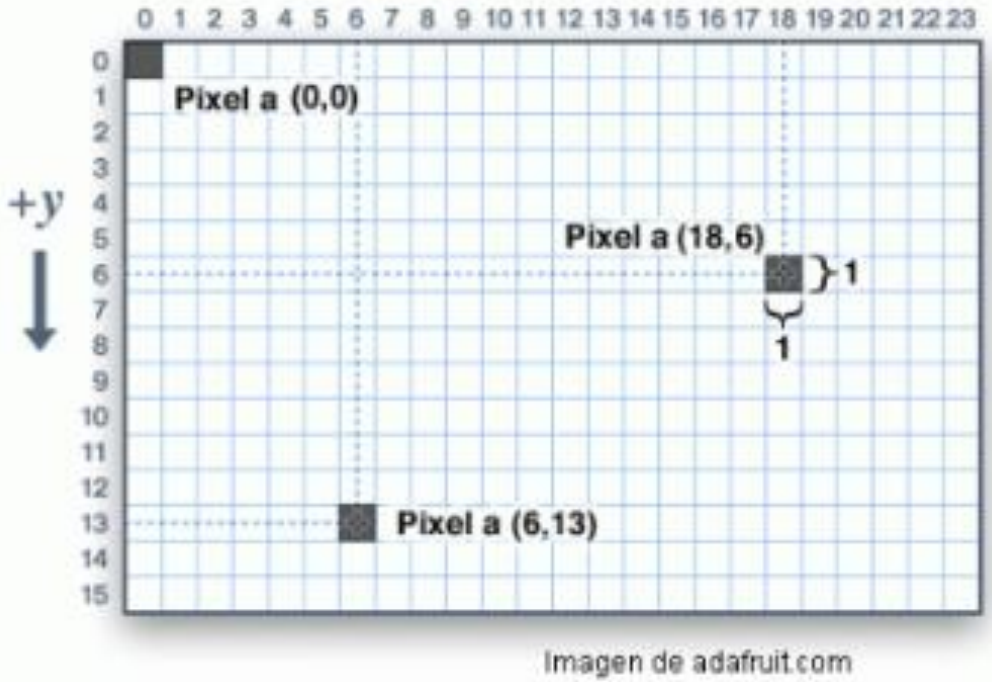
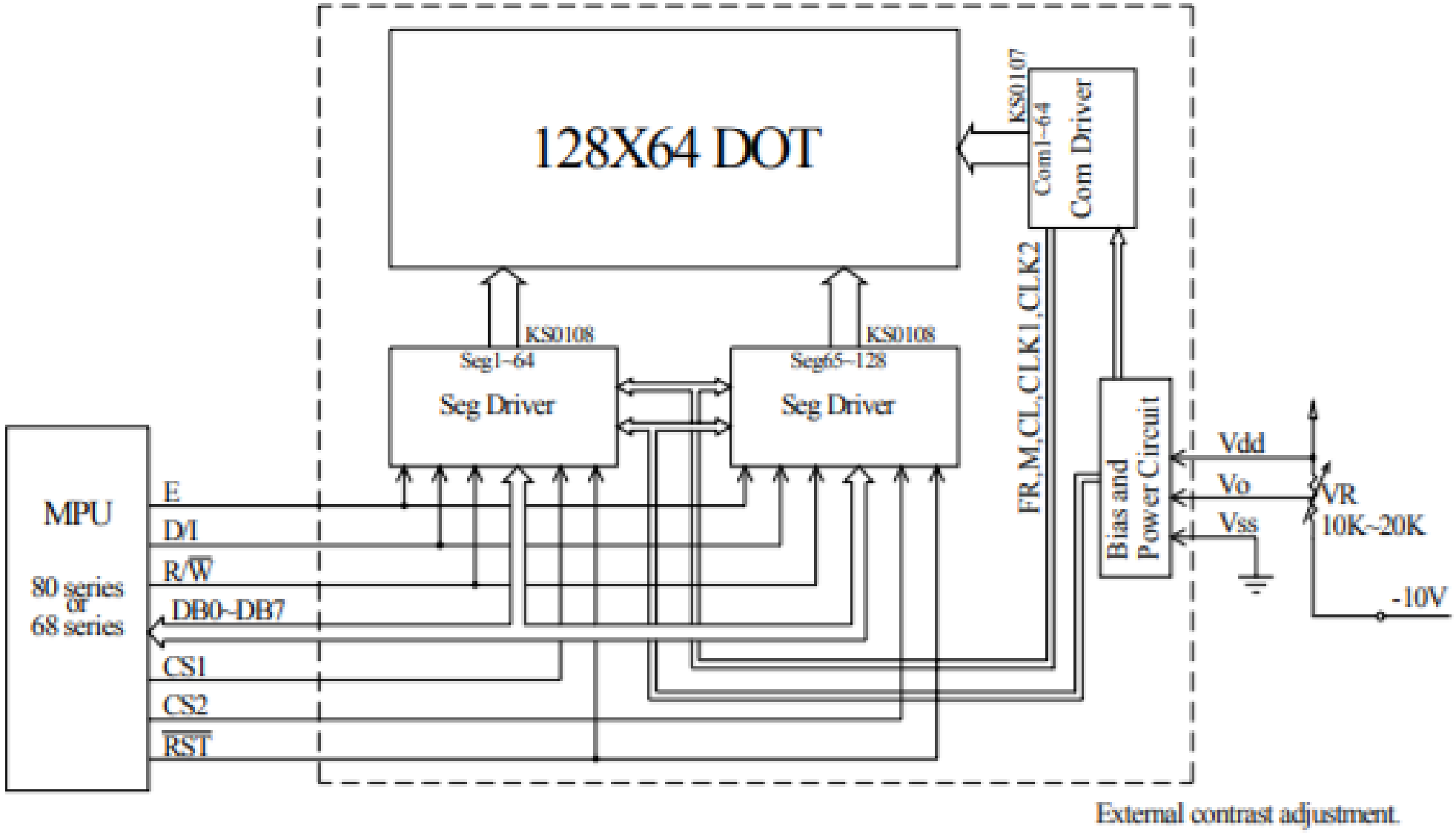
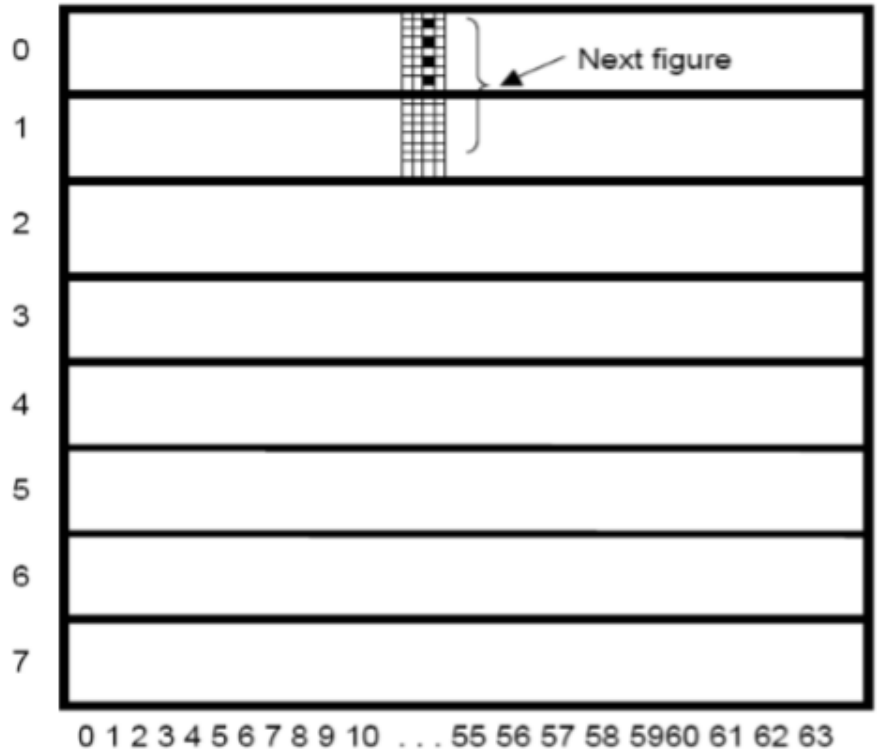


Imagen de adafruit.com

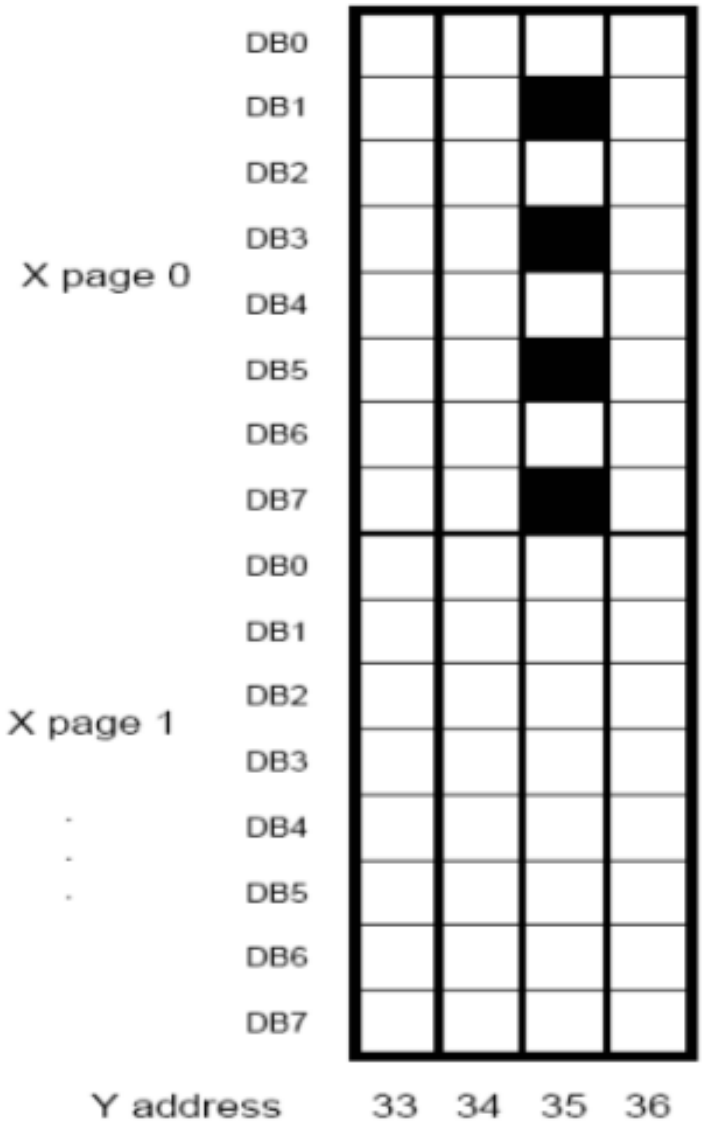

128X64 DOT
KS0107
Com1~64
Com Driver
KS0108
Seg1~64
Seg Driver
KS0108
Seg65~128
Seg Driver
FR,M,CL,CLK1,CLK2
Bias and
Power Circuit
Vdd
Vo
Vss
VR
10K~20K
-10V
MPU
80 series
or
68 series
E
D/I
R/W
DB0~DB7
CS1
CS2
RST
External contrast adjustment.

Tiene dos controladores, uno controla la parte izquierda y el otro la parte derecha. Agregan tres señales nuevas, las señales CS1 y CS2 para la activación de los chips de pantalla, una señal I/D que es el equivalente a RS en los alfanuméricos y una señal de RESET muy útil a la hora de inicializar el GLCD.



Cada display tiene dos páginas de 64 columnas que van desde la dirección 0x40 a 0x7F esto conforma la dirección “Y”, que básicamente es como un contador que se auto incrementa de manera automática al escribir en la pantalla de tal forma que la coordenada “Y” se auto ajusta.

En el caso de la coordenada “X” no hay auto ajuste en esta coordenada y son 8 segmentos de 8 bits verticales.



La página 0 de la coordenada “X” ocupa la dirección 0xB8 luego 0xB9..... hasta 0xBF.

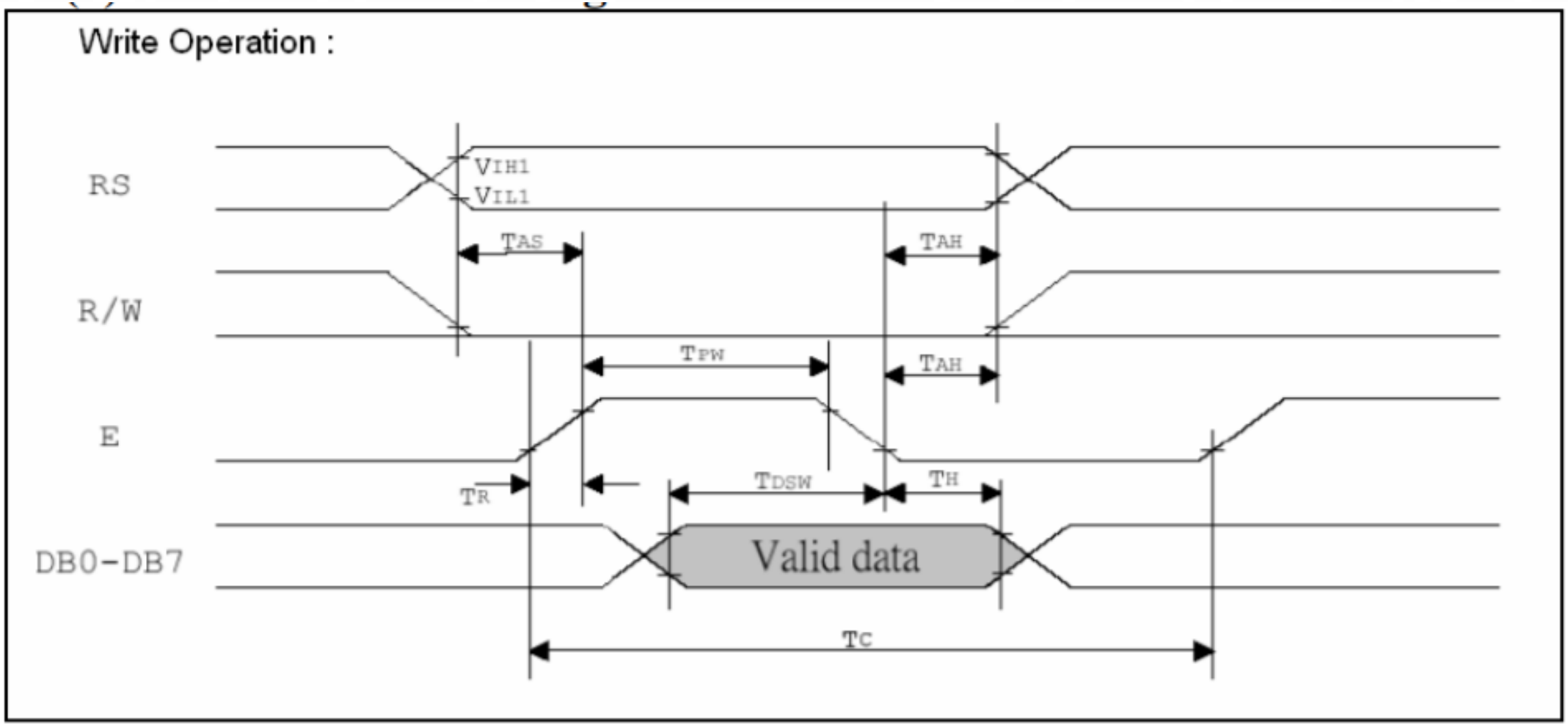
En el ejemplo hay un byte con el binario 10101010

El segmento o pagina “X 0” en la dirección 0xB8 tiene 64 bytes verticales correspondiendo cada uno de ellos a una columna de la coordenada “Y” por tanto cada segmento “X” puede controlar 512 pixeles los ocho segmentos controlan 4096 pixeles en cada mitad de la pantalla

Cada controlador puede manejar 4096 pixeles y claro está la pantalla tiene un total 8192 pixeles que se manejan con los dos controladores de pantalla.

MANEJO DE PANTALLAS GRAFICAS nota sobre un POWERTIP PG12864 (128x64 pixeles)

Write Operation :



. 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

```
send_comando(1,0x40);
send_comando(2,0x40);
send_comando(1,0xB8);
send_comando(2,0xB8);
send_comando(1,0x3F);
send_comando(2,0x3F);
```

```
void send_comando(int cs, char a){
GPIOD->ODR =0;//RS=0 o DI
if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
else if(cs==2){GPIOD ->ODR |=(1UL<<9); // Enable
CS2
GPIOD ->ODR |= a;
GPIOD->ODR |=(1UL<<11); // Enable 1
for(int Cont=0;Cont<10000;Cont++);
GPIOD->ODR &=~(1UL<<11); // Enable 0
GPIOD->ODR =0; // cs 0
}
}

void send_dato(int cs, char b){
GPIOD->ODR |=(1UL<<10);//RS=1 o DI
if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
else if(cs==2){GPIOD ->ODR |=(1UL<<9); // Enable
CS2
GPIOD ->ODR |= b;
GPIOD->ODR |=(1UL<<11); // Enable 1
for(int Cont=0;Cont<10000;Cont++);
GPIOD->ODR &=~(1UL<<11); // Enable 0
GPIOD->ODR =0; // cs 0
} }
```

```
while(true){

  // Recorre las 8 paginas (vertical)
  for(i = 0; i < 8; ++i){
   //Comienzo, en cada página, desde la dirección 0
    send_comando(1,0x40);
    send_comando(2,0x40);
    //Selecciono la direccion dentro de la pagina
    send_comando(1, i | 0xB8);
    send_comando(2, i | 0xB8);
    // Recorre las dos mitades (horizontales)
    for(j = 0; j < 64; ++j)
    {  send_dato(1, 0xFF);  // Enciende/apaga pixeles
       send_dato(2, 0xFF);  // Enciende/apaga pixeles
    }
  }
}
```

```c
#include "STM32F7xx.h"
char setX= 0xB8;
char disp_on= 0x3F;
char starL=0xC0;
char setY=0x40;
int i, j;

void send_comando(int cs, char a){
  GPIOD->ODR =0;//RS=0
  if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
    else    if(cs==2){GPIOD ->ODR |=(1UL<<9); // Enable CS2
  GPIOD->ODR |=(1UL<<11); // Enable 1
  for(int Cont=0;Cont<20000;Cont++);
  GPIOD ->ODR |= a;
  GPIOD->ODR |=(1UL<<11); // Enable 1
  GPIOD->ODR &=~(1UL<<11); // Enable 0
  GPIOD->ODR &=0X00FF; // cs 0
    }
}

void send_dato(int cs, char b){
  GPIOD->ODR |=(1UL<<10);//RS=1
  if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
    else    if(cs==2){GPIOD ->ODR |=(1UL<<9); // Enable CS2
  GPIOD->ODR |=(1UL<<11); // Enable 1
  for(int Cont=0;Cont<20000;Cont++);
  GPIOD ->ODR |= b;
  GPIOD->ODR |=(1UL<<11); // Enable 1
  GPIOD->ODR &=~(1UL<<11); // Enable 0
  GPIOD->ODR &=0X00FF; // cs 0
  } }
```

```c
int main(void){

  RCC->AHB1ENR = 0x08;;
  GPIOD->MODER =0x05555555;
  GPIOD->ODR &=(1UL<<12);//RST=1
  send_comando(1,disp_on);
  send_comando(2,disp_on);
  send_comando(1,starL);
  send_comando(2,starL);
  send_comando(1,disp_on);
  send_comando(2,disp_on);

while(true){

   // Recorre las 8 paginas (vertical)
   for(i = 0; i < 8; ++i){
       //Comienzo, en cada página, desde la dirección 0
      send_comando(1,setY);
      send_comando(2,setY);
      //Selecciono la direccion dentro de la pagina
      send_comando(1, i | setX);
      send_comando(2, i | setX);
      // Recorre las dos mitades (horizontales)
      for(j = 0; j < 64; ++j)
      {  send_dato(1, 0xFF * 1);  // Enciende/apaga pixeles
         send_dato(2, 0xFF * 0);  // Enciende/apaga pixeles
      }
   }
}
}
```