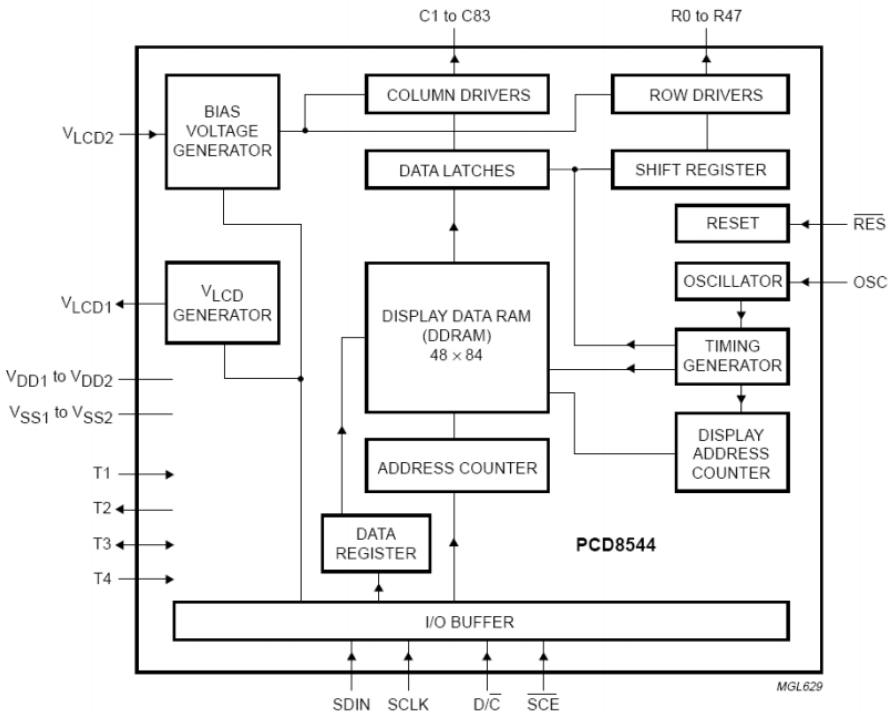
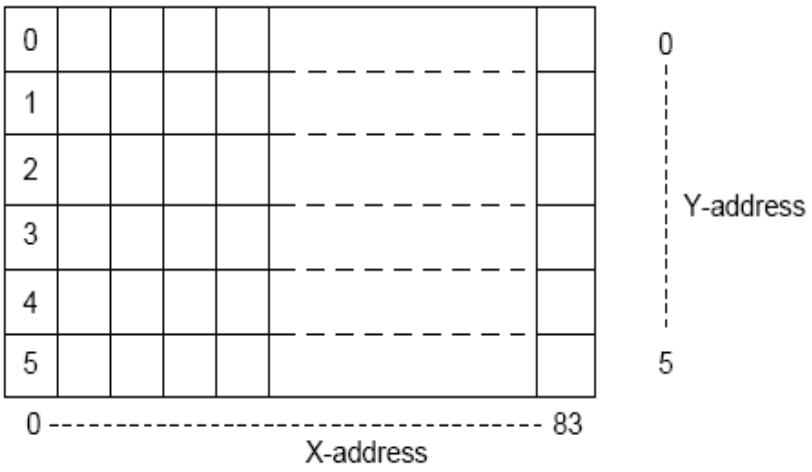
การเขียนโปรแกรมให้ทำงานตามไดอะแกรมเวลา (Timing diagram)

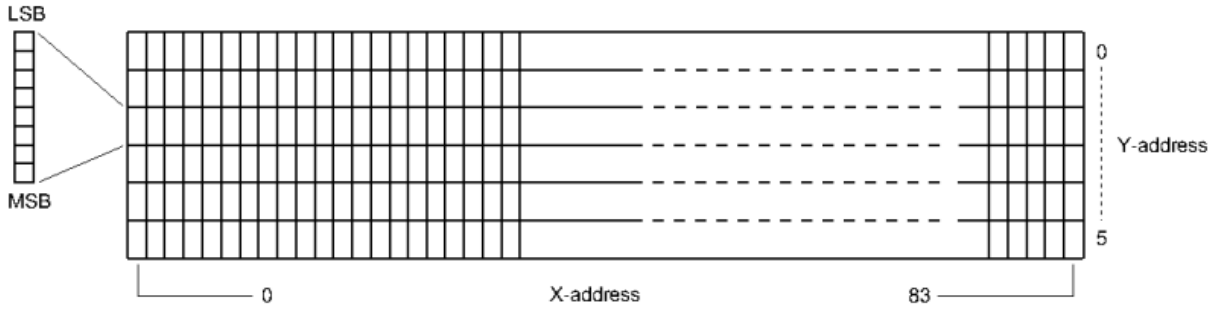
ตัวอย่างการทำงานของ LCD Nokia 5110



รูปที่ 1 แสดงโครงสร้างภายใน ตัวควบคุมจอ LCD PCD8544.



รูปที่ 2 แสดงค่าของ x และ บนจอภาพเทียบกับตำแหน่งหน่วยความจำ



รูปที่ 3 แสดงตำแหน่งจุดภาพเมื่อเทียบกับ ตำแหน่งบิตของหน่วยความจำ

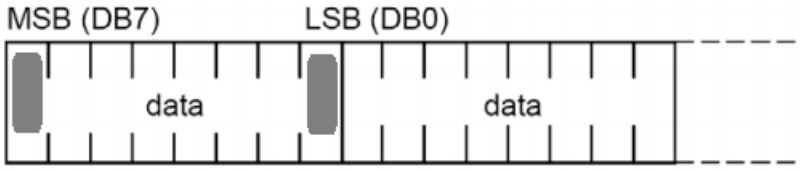
ตำแหน่งขาสัญญาณ

Red			Blue
1	VCC	1	RST
2	GND	2	CE
3	SCE	3	DC
4	RES	4	DIN
5	DC	5	CLK
6	SDIN	6	VCC
7	SCLK	7	BL
8	LED	8	GND

Pin's Name	Functions
1. VCC	Pin +VCC; using Power Supply from 2.7 - 5 VCD
2. GND	Pin Ground
3. SCE	Pin CHIP ENABLE to control operation of Pin Controllers
4. RESET	Signal RESET for operation of LCD
5. D/C	Pin to configure the data formats between Data and Command.
6. SDIN	Pin DATA (SERIAL DATA LINE)
7. SCLK	Pin CLOCK (SERIAL CLOCK LINE)
8. LED	Pin to control operation of LED (Back Light)

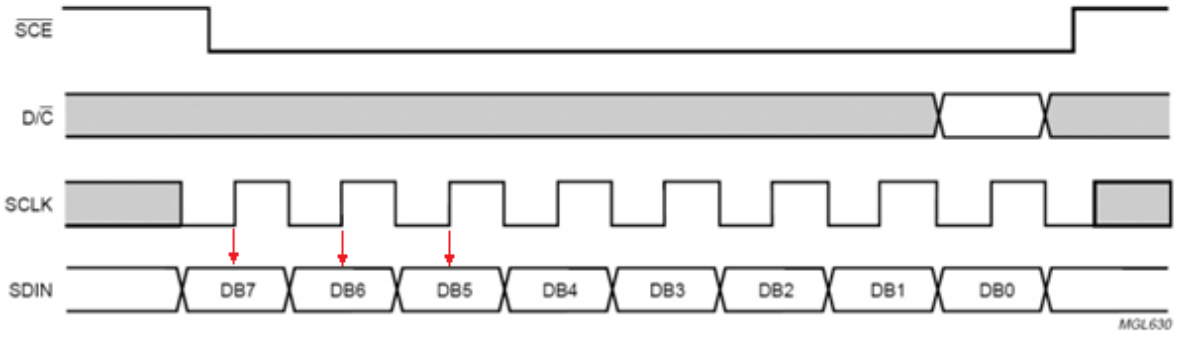
D/C = 1 Data D/C = 0 Command

Communication Format



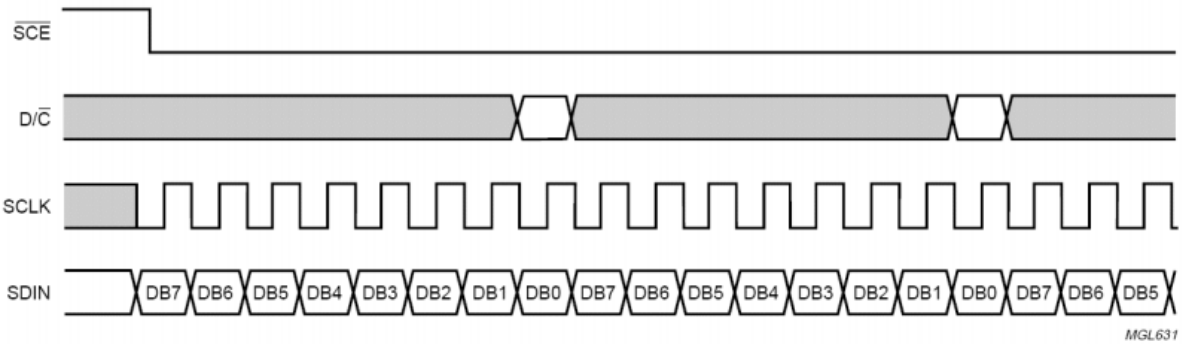
รูปที่ 4 แสดงรูปแบบการสื่อสารข้อมูล

มีไดอะแกรมเวลาดังนี้



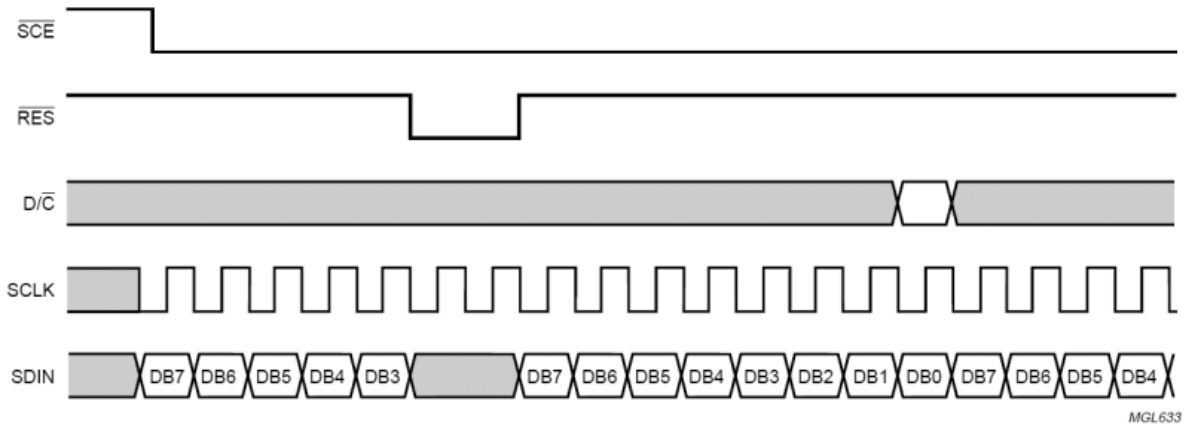
Serial bus protocol - transmission of one byte.

รูปที่ 5 แสดงการส่งข้อมูล 1 ไบต์



Serial bus protocol - transmission of several bytes.

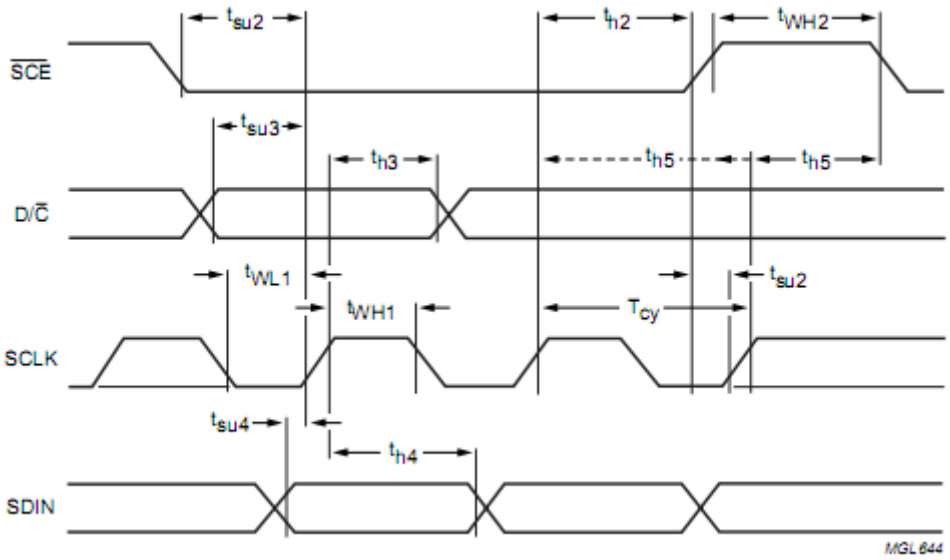
รูปที่ 6 แสดงการส่งข้อมูลหลายไบต์



Serial bus reset function ($\overline{RES}$).

รูปที่ 7 แสดงการส่งสัญญาณเพื่อรีเซ็ต

ค่าเวลาในการส่งข้อมูลแบบอนุกรม (Serial interface timing)



รูปที่ 8 แสดงค่าเวลาต่างๆในการส่งข้อมูล

12 AC CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
f_{OSC}	oscillator frequency		20	34	65	kHz
$f_{clk(ext)}$	external clock frequency		10	32	100	kHz
f_{frame}	frame frequency	f_{OSC} or $f_{clk(ext)}$ = 32 kHz; note 1	–	67	–	Hz
t_{VHRL}	V_{DD} to $\overline{RES}$ LOW	Fig.16	0[2]	–	30	ms
$t_{WL(RES)}$	$\overline{RES}$ LOW pulse width	Fig.16	100	–	–	ns
Serial bus timing characteristics						
f_{SCLK}	clock frequency	V_{DD} = 3.0 V ±10%	0	–	4.00	MHz
T_{cy}	clock cycle SCLK	All signal timing is based on 20% to 80% of V_{DD} and maximum rise and fall times of 10 ns	250	–	–	ns
t_{WH1}	SCLK pulse width HIGH		100	–	–	ns
t_{WL1}	SCLK pulse width LOW		100	–	–	ns
t_{su2}	$\overline{SCE}$ set-up time		60	–	–	ns
t_{h2}	$\overline{SCE}$ hold time		100	–	–	ns
t_{WH2}	$\overline{SCE}$ min. HIGH time		100	–	–	ns
t_{h5}	$\overline{SCE}$ start hold time; note 3		100	–	–	ns
t_{su3}	D/$\overline{C}$ set-up time		100	–	–	ns
t_{h3}	D/$\overline{C}$ hold time		100	–	–	ns
t_{su4}	SDIN set-up time		100	–	–	ns
t_{h4}	SDIN hold time		100	–	–	ns

Notes

1. $T_{frame} = \frac{f_{clk(ext)}}{480}$
2. $\overline{RES}$ may be LOW before V_{DD} goes HIGH.
3. t_{h5} is the time from the previous SCLK positive edge (irrespective of the state of $\overline{SCE}$) to the negative edge of $\overline{SCE}$ (see Fig.15).

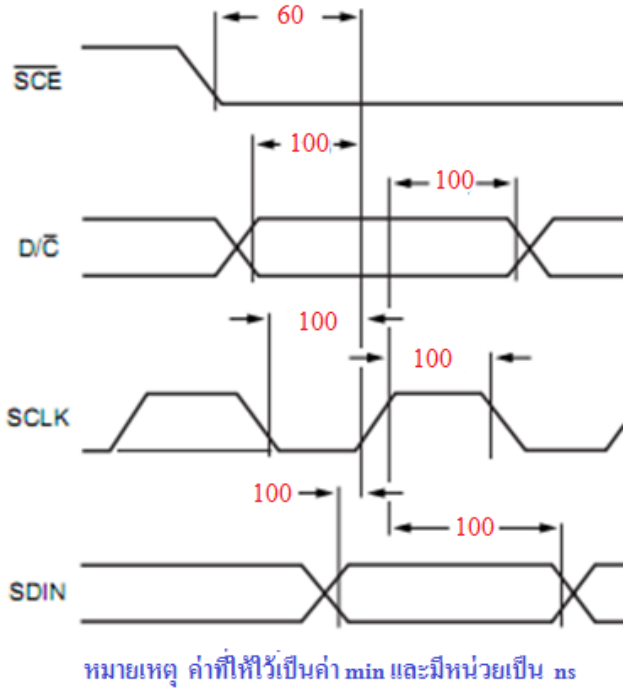
การเขียนโปรแกรมแบบ Top-down and bottom-up design

เขียนจากส่วนย่อยๆ ไปหาส่วนใหญ่ (bottom-up)

เขียนจากโครงใหญ่ ไปหารายละเอียดย่อยๆ (Top-down)

ในที่นี้ใช้วิธี เขียนลายลัเอียดส่วนย่อยๆก่อน แล้วจึงเขียนส่วนใหญ่ โดยเน้นที่การเขียนส่วนที่ใช้งานร่วมกัน คือการส่งข้อมูลแบบอนุกรมแต่ละบิต ที่มีสัญญาณนาฬิกา (Clock) กำกับ

พิจารณาการทำงานตามไดอะแกรมเวลาในรูปที่ 8 มาเขียนเป็นรูปที่ 9 ดังนี้



รูปที่ 9 แสดงค่าเวลาต่างๆในการส่งข้อมูลแต่ละบิต

โปรแกรมส่งข้อมูลหรือคำสั่ง 8 บิต ตามรูปที่ 5 และรูปที่ 9

```
//Sends Data or Command in serial mode
void nokia_write_dorc(uint8_t n_dato)
{
  uint8_t caa;
  for (caa=8;caa>0;caa--){      //MSB first
    digitalWrite(nok_sclk,LOW);
    delayMicroseconds(2);
    if ((n_dato&0x80)==0) digitalWrite(nok_sda,LOW);
    else digitalWrite(nok_sda,HIGH);
    digitalWrite(nok_sclk,HIGH);
    delayMicroseconds(2);
    n_dato = n_dato << 1;
  }
  digitalWrite(nok_sclk,LOW);
}
```

โปรแกรมส่งข้อมูลที่เป็นคำสั่ง (Command) 8 บิต

```
//Command write
void write_command(uint8_t comando)
{
  digitalWrite(nok_dc,LOW);          //Command
  digitalWrite(nok_cs,LOW);          //Chip enable
  nokia_write_dorc(comando);
  digitalWrite(nok_dc,HIGH);
  digitalWrite(nok_cs,HIGH);         //Chip disable
}
```

โปรแกรมส่งข้อมูล (Data) 8 บิต

```
//Data write
void write_data(uint8_t dato)
{
  digitalWrite(nok_dc,HIGH);         //Data
  digitalWrite(nok_cs,LOW);          //Chip enable
  nokia_write_dorc(dato);
  digitalWrite(nok_cs,HIGH);         //Chip disable
}
```

ส่วนการกำหนดการทำงานของ LCD ดูได้จากตารางคำสั่ง ในตารางที่ 1

เช่น **กำหนดกลุ่มคำสั่ง Function set**

	D/C̄	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)

0 0 1 0 0 0 0 1 = 0x21

BIT	0	1
PD	chip is active	chip is in Power-down mode
V	horizontal addressing	vertical addressing
H	use basic instruction set	use extended instruction set

การกำหนดค่าความเข้ม กำหนดที่ค่า Vop

D/C = 0

Set V_{OP}	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	write V_{OP} to register

1	0	0	1	0	0	0	0	0

= 0xA0

การกำหนดตำแหน่ง Cursor

Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$

ตารางที่ 1 ตารางคำสั่ง

Table 1 Instruction set

INSTRUCTION	D/$\overline{C}$	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	writes data to display RAM
(H = 0)										
Reserved	0	0	0	0	0	0	1	X	X	do not use
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Reserved	0	0	0	0	1	X	X	X	X	do not use
Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	do not use
	0	0	0	0	0	0	0	1	X	do not use
Temperature control	0	0	0	0	0	0	1	TC_1	TC_0	set Temperature Coefficient (TC_x)
Reserved	0	0	0	0	0	1	X	X	X	do not use
Bias system	0	0	0	0	1	0	BS_2	BS_1	BS_0	set Bias System (BS_x)
Reserved	0	0	1	X	X	X	X	X	X	do not use
Set V_{OP}	0	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	write V_{OP} to register

Table 2 Explanations of symbols in Table 1

BIT	0	1
PD	chip is active	chip is in Power-down mode
V	horizontal addressing	vertical addressing
H	use basic instruction set	use extended instruction set
D and E		
00	display blank	
10	normal mode	
01	all display segments on	
11	inverse video mode	
TC_1 and TC_0		
00	V_{LCD} temperature coefficient 0	
01	V_{LCD} temperature coefficient 1	
10	V_{LCD} temperature coefficient 2	
11	V_{LCD} temperature coefficient 3	

โปรแกรมในส่วนของฟังก์ชั่นควบคุมการทำงานของ LCD

```
#define nok_sclk  9      // nokia lcd sclk PIN 7
#define nok_sda   8      // nokia lcd sda  PIN 6
#define nok_dc    7      // nokia lcd d/c  PIN 5
#define nok_res   5      // nokia lcd res  PIN 4
#define nok_cs    6      // nokia lcd cs   PIN 3

//-----------------------------------------------------------------------------
//Sends Data or Command in serial mode
void nokia_write_dorc(uint8_t n_dato)
{
  uint8_t caa;
  for (caa=8;caa>0;caa--){      //MSB first
    digitalWrite(nok_sclk,LOW);
    delayMicroseconds(2);
    if ((n_dato&0x80)==0) digitalWrite(nok_sda,LOW);
    else digitalWrite(nok_sda,HIGH);
    digitalWrite(nok_sclk,HIGH);
    delayMicroseconds(2);
    n_dato = n_dato << 1;
  }
  digitalWrite(nok_sclk,LOW);
}
//-----------------------------------------------------------------------------
//Command write
void write_command(uint8_t comando)
{
  digitalWrite(nok_dc,LOW);          //Command
  digitalWrite(nok_cs,LOW);          //Chip enable
  nokia_write_dorc(comando);
  digitalWrite(nok_dc,HIGH);
  digitalWrite(nok_cs,HIGH);         //Chip disable
}
//-----------------------------------------------------------------------------
//Data write
void write_data(uint8_t dato)
{
  digitalWrite(nok_dc,HIGH);         //Data
  digitalWrite(nok_cs,LOW);          //Chip enable
  nokia_write_dorc(dato);
  digitalWrite(nok_cs,HIGH);         //Chip disable
}
//-----------------------------------------------------------------------------
void cursorxy(unsigned char x, unsigned char y)
{
  write_command(0x40|(y&0x07));   // Y axis
  write_command(0x80|(x&0x7f));   // X axis
}
//-----------------------------------------------------------------------------
//Blank  screen
void lcd_blank_screen(void){
  uint8_t i,j;
  write_command(0x40);    //Set Y address of RAM  0 1 0 0 0 Y2 Y1 Y0     0<= Y <= 5
  write_command(0x80);    //Set X address of RAM  1 X6 X5 X4 X3 X2 X1 X0  0 <= X <=83
  for(j=0;j<6;j++){             //Sends memory map to LCD
    for(i=0;i<84;i++){
      write_data(0x00);
    }
  }
}
//-----------------------------------------------------------------------------
```

```
//Initialization
//Bytes are stored in the display data ram, address counter, incremented automatically
void initlcd(void) {
  digitalWrite(nok_dc,HIGH);
  digitalWrite(nok_cs,HIGH);    //Chip disabled
  delayMicroseconds(200);
  digitalWrite(nok_res,LOW);
  delay(1);
  digitalWrite(nok_res,HIGH);
  write_command(0x21);    //Extended instructions set
  write_command(0xa0);    //Vop
  write_command(0x13);    //Bias
  write_command(0x20);    //Horizontal mode from Left to Right, X coordinate increments automatically,
                          //0x22 for vertical addressing, also normal instructions set again
  write_command(0x09);    //Everything ON
  delay(1);
  lcd_blank_screen();     //Memory map clean
  write_command(0x08);    //Everything OFF
  delay(1);
  write_command(0x0c);    //Normal mode
  cursorxy(0,0);          //Cursor Home.
}
```

ตัวอย่างให้แสดงรูปกล่องที่บรรทัดที่ 4 ขนาดยาว 24 จุด

หมายเหตุ อย่าลืมใส่โปรแกรมส่วนควบคุม LCD ไว้ด้วย

```
void setup() {
  pinMode(nok_sclk, OUTPUT);
  pinMode(nok_sda, OUTPUT);
  pinMode(nok_dc, OUTPUT);
  pinMode(nok_cs, OUTPUT);
  pinMode(nok_res, OUTPUT);
  initlcd();
  cursorxy(0,3);
  for (uint8_t i=0; i < 24; i++){
     write_data(0xFF);
   }

}

void loop() {

}
```

ตัวอย่างแสดงภาพ โลโก้ batman ที่ตำแหน่ง x = 0 y = 0

อธิบายคำสั่งที่เกี่ยวข้อง

pgm_read_byte_near()

เป็นคำสั่ง macro สำหรับการอ่านข้อมูลที่เก็บอยู่ในหน่วยความจำของโปรแกรม โดยอ่านแบบไบต์

pgm_read_word_near() เหมือนกับ pgm_read_byte_near() แต่อ่านแบบ 16 บิต

คำสั่ง macro คืออะไร

คำสั่ง macro เหมือนกับฟังก์ชั่น เพียงแต่ตอนแปลจะยกเอาคำสั่งทั้งหมดของตัวคำสั่งมาแทนที่ตรงคำสั่ง macro ปรากฏอยู่ ส่วนฟังก์ชั่นไม่ได้ยกเอาคำสั่งในฟังก์ชั่นมา

```
extern uint8_t batman[];              //array of batman logo  ,size 80x40

void setup() {
  int ptr = 0;
  pinMode(nok_sclk, OUTPUT);
  pinMode(nok_sda, OUTPUT);
  pinMode(nok_dc, OUTPUT);
  pinMode(nok_cs, OUTPUT);
  pinMode(nok_res, OUTPUT);
  initlcd();
  ptr = 0;
  for (uint8_t j=0; j < 5; j++){
    cursorxy(0,j);
    for (uint8_t i=0; i < 80; i++){
      write_data(pgm_read_byte_near(batman+ptr));
      ptr++;
   }
  }
}
void loop() {

}
```