



그래픽 LCD 사용하기

1. 점찍기
2. 선그리기
3. 사각형, 원 그리기
4. 글씨쓰기





서론



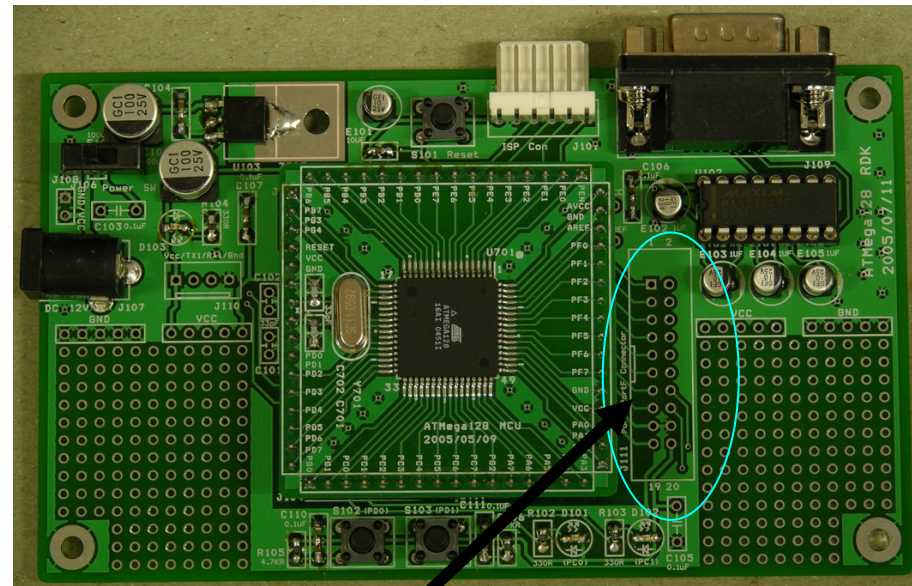
- 제품을 개발할 때, 그래픽 LCD는 기본으로 사용하게 되는데, 이를 처음에 사용하려면 어떻게 해야 할 지 막막하다. 이런 분들을 위해서, 기초부터 탄탄히 하시라는 의미에서 초기에 가장 많이 사용하게 되는 저렴한 흑/백 LCD인 그래픽 LCD를 가지고 사용법에 관련된 매뉴얼을 작성하게 되었다.
- 이 LCD는 SG12864H1로 용산 광장층 입구 쪽에 있는 SY 전자 등에서 판매하는 SUNLIKE LCD Module인 SG12864D와 같은 칩셋을 사용하지만, 약간 전원 등의 핀 번호가 다르며, Vout 전압도 연결하는 법이 약간 다르다.
- 하드웨어적인 차이 외에는 SG12864A~G와 같은 소스로 작성이 가능하므로, 참조하기 바란다.



AVR Mega 128 개발보드쪽 핀 연결

- AVR Mega 128 개발보드의 포트는 별도로 보드에 박스 커넥터로 외부 출력으로 뽑아둔 A와 F포트를 사용하였습니다.

- A6 : Backlight On/Off
- A5 : E
- A4 : D/I
- A3 : R/W
- A2 : /RES
- A1 : CS2
- A0 : CS1
- F7~F0 : Data 7 ~ Data 0

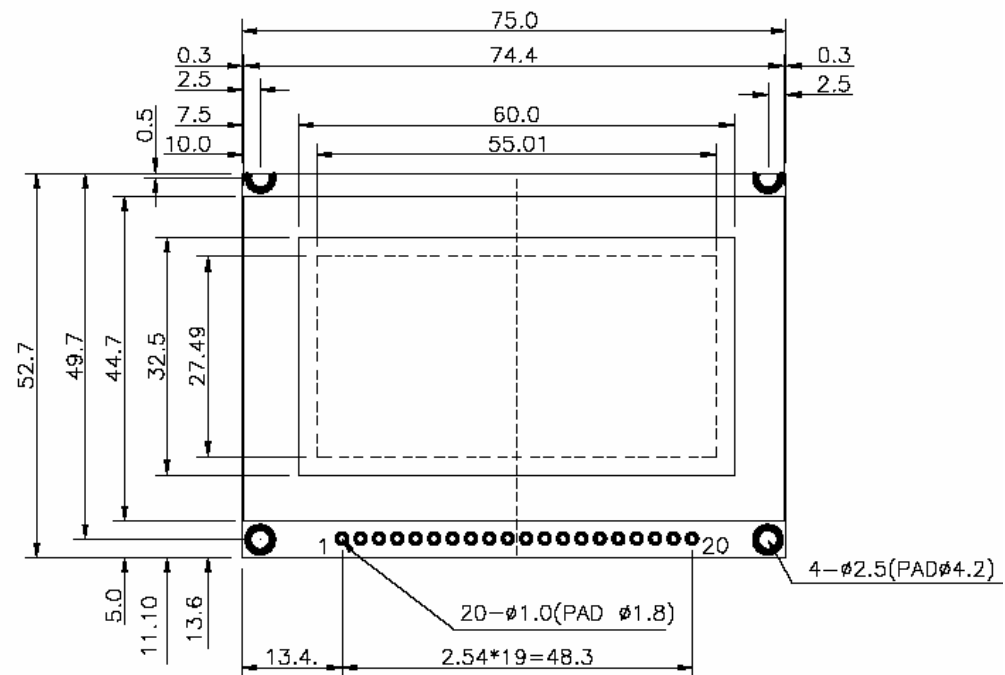




- Features

- Built-In Controller(KS0107 or Equivalent)
- +5V Power Supply
- 1/64 Duty Cycle
- 8-Bit Parallel Interface
- 4.2V LED Forward Voltage

NO.	SYMBOL	LEVEL	FUNCTION
1	V _{DD}	5V	Power Supply Voltage
2	V _{SS}	0V	Power Supply Ground
3	V _O	-	Contrast Adjustment Voltage
4~11	DB0~DB7	H/L	Data Bus Line
12	CS1	H	Chip Select Signal For IC1
13	CS2	H	Chip Select Signal For IC2
14	/RES	L	Reset Signal
15	R/W	H/L	H : Read / L : Write
16	D/I	H/L	H : Data, L : Instruction Code
17	E	H→L	Enable Signal
18	V _{out}	-	Power Supply Voltage For LCD
19	A	4.2V	LED Power (+)
20	K	0V	LED Power (-)

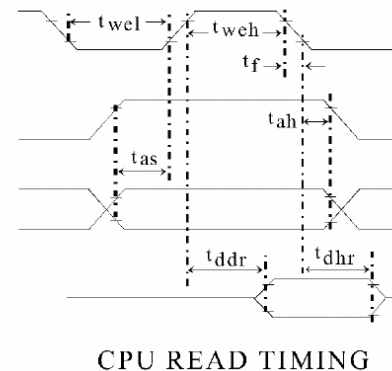
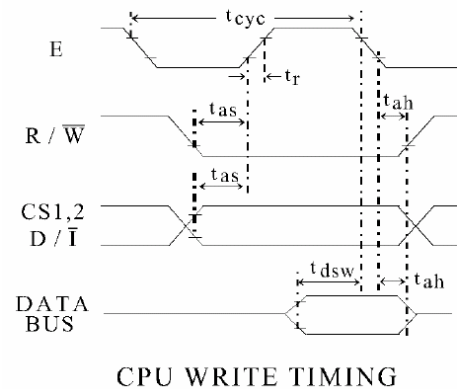




그래픽 LCD 타이밍 차트

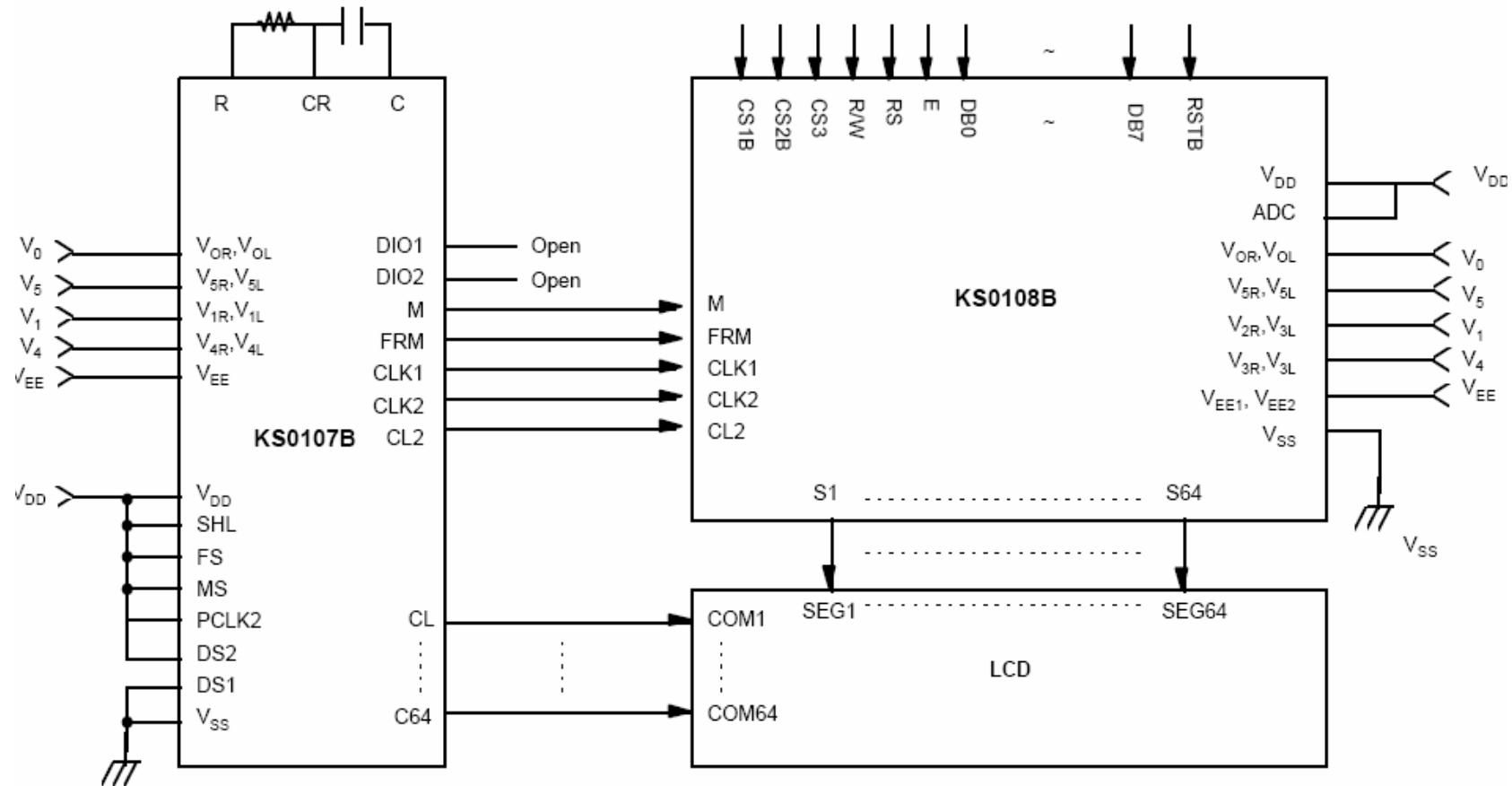


Item	Symbol	Condition	Min.	Max.	Unit
E Cycle time	t_{cyc}	Vdd = 5V $\pm$ 5% Vss = 0V Ta = 25°C	1000	-	nS
E High level width	t_{weh}		450	-	
E Low level width	t_{wel}		450	-	
E Rise time	t_r		-	25	
E Fall time	t_f		-	25	
Address setup time	t_{as}		140	-	
Address hold time	t_{ah}		10	-	
Data setup time	t_{dsw}		200	-	
Data delay time	t_{ddr}		-	320	
Data hold time (write)	t_{dhw}		10	-	
Data hold time (read)	t_{dhr}		20	-	





LCD Controller 구성도





그래픽 LCD 칩셋의 레지스터들



- Output Register

RS	R/W	Function
L	L	Instruction
	H	Status read (busy check)
H	L	Data write (from input register to display data RAM)
	H	Data read (from display data RAM to output register)

- 입력 버퍼
 - 입력 버퍼는 칩을 Enable 또는 Disable에 관계한다. 따라서, CS1B~CS3이 Active mode에 있지 않다면, 데이터의 입출력 및 instruction이 동작하지 않는다. 따라서, 내부 상태는 변화가 없다. 그렇지만, RSTB와 ADC는 관계없이 동작한다.
- 입력 레지스터
 - 입력 레지스터는 MPU에 인터페이스를 제공하며, RAM을 디스플레이 하는 곳에 그것을 쓰기 전에 임시로 데이터가 저장된다. 그런 다음에 디스플레이 램에 그것을 쓴다. E 신호의 falling edge에서 래치되고, 내부 동작에 의해 디스플레이 램으로 자동적으로 써진다.
- 리셋
 - 전원을 켤 때, RSTB를 Low로 떨어뜨리면서 초기화 된다.



그래픽 LCD에 Instruction 주는 법

(칩셋에서 지원하는 Instruction)

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	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display ON/OFF	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is not affected. L: OFF, H: ON
Set Address	L	L	L	H	Y address (0~63)						Sets the Y address in the Y address counter.
Set Page (X address)	L	L	H	L	H	H	H	Page (0~7)			Sets the X address at the X address register.
Display Start Line	L	L	H	H	Display start line (0~63)						Indicates the display data RAM displayed at the top of the screen.
Status Read	L	H	B U S Y	L	O N / O F F	R E S E T	L	L	L	L	Read status. BUSY L: Ready H: In operation ON/OFF L: Display ON H: Display OFF RESET L: Normal H: Reset
Write Display Data	H	L	Write Data								Writes data (DB0:7) into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read Display Data	H	H	Read Data								Reads data (DB0:7) from display data RAM to the data bus.

그래픽 LCD에 Instruction 주는 법

(그래픽 LCD 모듈에서 지원하는 Instruction)

Instruction	C O D E										
	R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
1. Display on/off	0	0	0	0	1	1	1	1	1	1/0	controls the on/off display RAM data and internal status are not affected. 1: on, 0: off
2. Display start line	0	0	1	1	display start line (0 ~ 63)						specifies a RAM line displayed at the of the screen
3. Set page (X address)	0	0	1	0	1	1	1	page (0 ~ 7)			set page (X address) of RAM at the page (Xaddress) regisister.
4. Set address	0	0	0	1	Y address (0 ~ 63)						set Y address at the "Y" address counter
5. Status read	1	0	B U S Y	0	on / off	R E S E T	0	0	0	0	Read the status. RESET 1: reset 0: normal On/ Off 1: display off 0: display on BUSY 1: on the internal operation
6. Write display data	0	1	Write data								Writes data DB0(LSB) to DB7(MSB) on the data bus into display RAM
7. Read display data	1	1	Read data								Reads data DB0(LSB) to DB7(MSB) from display RAM to the data bus

자세히 보면 알 수 있지만, 칩셋에서 지원하는 Instruction Set과 모듈에서 지원하는 Instruction Set가 비슷하지만, 각각 RS와 D/I로 이름이 바뀌어 있음을 알 수 있다.

그래픽 LCD에 Instruction 주는 법

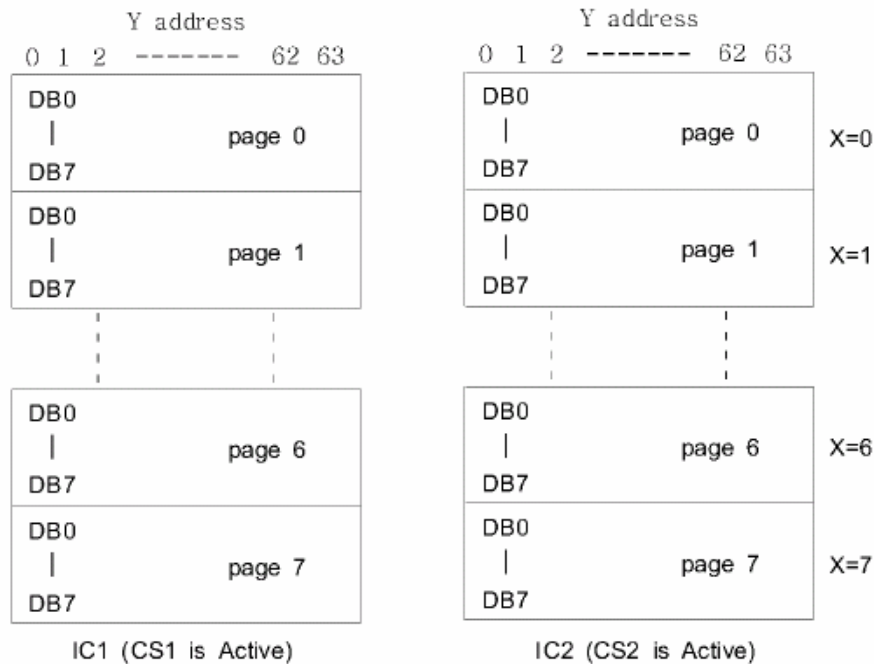
(구체적인 방법)

- 이 그래픽 LCD는 64*128 Pixel로 구성되어 있으며, CS0와 CS1에 의해 칩셀렉트로 64*64를 각각 On/Off 한다.
- 이 때, 각 CS로 선택된 LCD만 데이터 입력에 반응한다. 따라서, 해당하는 64*64 부분의 LCD를 끄고 컴에 의해, 한쪽에만 컨트롤을 할 수 있게 된다.
- 또한, 64*128에서 64쪽이 X이고, 128 쪽이 Y이다.

그래픽 LCD에 Instruction 주는 법

(구체적인 방법)

- X page : 0~7번까지 있음.
- Y address : 0 ~ 63까지 있음.





주요 Instruction 세트



Set page (X address)

	R/ $\overline{W}$	D/ $\overline{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	0	0	1	0	1	1	1	A	A	A

Set Y address

	R/ $\overline{W}$	D/ $\overline{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	0	0	0	1	A	A	A	A	A	A

Write display data

	R/ $\overline{W}$	D/ $\overline{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	1	1	D	D	D	D	D	D	D	D

Writes 8-bit data DDDDDDDD (binary) into the display data RAM.
Then Y address is increased by 1 automatically.

Read display data

	R/ $\overline{W}$	D/ $\overline{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	1	1	D	D	D	D	D	D	D	D

Read out 8-bit data DDDDDDDD (binary) from the display data RAM.
Then Y address is increased by 1 automatically.
One dummy read is necessary soon after the address setting.





부가적인 Instruction 세트



Display on/ off

	R/ $\bar{W}$	D/ $\bar{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	0	0	1	1	A	A	A	A	A	A

Z address AAAAAA (binary) of the display data RAM is set at the display start line register and displayed at the top of the screen.
fig. 13-2 are the examples of display when the start line= 0~3.

Display on/ off

	R/ $\bar{W}$	D/ $\bar{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	0	0	0	0	1	1	1	1	1	D

The display data appears when D is 1 and disappears when D is 0.
Through the data is not on the screen width D="0", it remains in the display data RAM. Therefore, you can make it appear by changing D="0" into D="1".

Status read

	R/ $\bar{W}$	D/ $\bar{I}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CODE	1	0	B	0	F	R	0	0	0	0

BUSY : When "B" is 1, the LSI is in internal operation. no instructions are accepted while busy is 1, so you should make sure that busy is 0 before writing the next instruction.

ON/ OFF : This bit shows the display conditions.
When F is 1, the display is in off condition.
When F is 0, the display is in on condition.

RESET : R=1 shows that the system is being initialized.
In this condition, any instruction except status read Instruction cannot be accepted.
R=0 shows that initializing has finished and the system is in the usual operation.

<http://www.compraza.co.kr>

davidryu1@yahoo.co.kr





함수 초기화



- // Defined Basic Graphic LCD Pin Function
- void RS(int onoff);
- void RW(int onoff);
- void E(int onoff);
- void E_Pulse(void);
- void DI(int onoff);
- void CS2(int onoff);
- void CS1(int onoff);

- // LCD Command
- void BackLight(int onoff);
- void DisplayOnOff(int onoff);
- void DataInput(char data);
- void DisplayStartLine(int displayline);
- void SetPage(int X);
- void SetAddress(int Y);
- char StatusRead(void);
- void WriteDisplayData(char data);
- char ReadDisplayData(void);
- void Reset(void);





- void RS(int onoff){
- if(onoff==ON)
- PORTA = PORTA|0x04;
- else
- PORTA = PORTA&0xFB;
- }

- void RW(int onoff){
- if(onoff==ON)
- PORTA = PORTA|0x08;
- else
- PORTA = PORTA&0xF7;
- }

- void E(int onoff){
- if(onoff==ON)
- PORTA = PORTA|0x20;
- else
- PORTA = PORTA&0xDF;
- }





- void DI(int onoff){
- if(onoff==ON)
- PORTA = PORTA|0x10;
- else
- PORTA = PORTA&0xEF;
- }

- void CS2(int onoff){
- if(onoff==ON)
- PORTA = PORTA|0x02;
- else
- PORTA = PORTA&0xFD;
- }

- void CS1(int onoff){
- if(onoff==ON)
- PORTA= PORTA|0x01;
- else
- PORTA = PORTA&0xFE;
- }





```
• void BackLight(int onoff){  
•   if(onoff==ON)  
•       PORTA = PORTA|0x40;  
•   else  
•       PORTA = PORTA&0xBF;  
•   }  
• void DisplayOnOff(int onoff){  
•     RW(OFF);  
•     DI(OFF);  
•     if(onoff==ON){  
•         DataInput(0x3F);  
•     }else{  
•         DataInput(0x3E);  
•     }  
• }  
• void DataInput(char data){  
•     RS(ON);  
•     PORTF = data;  
•     E_Pulse();  
• }
```





-
- void DisplayStartLine(int displayline){ /* display start line 0~63 */
 - RW(OFF);
 - DI(OFF);
 - DataInput(displayline | 0xc0);
 - }

 - void SetPage(int X){
 - RW(OFF);
 - DI(OFF);
 - DataInput(X | 0xb8);
 - }

 - void SetAddress(int Y){
 - RW(OFF);
 - DI(OFF);
 - DataInput(Y | 0x40);
 - }

 -





```
• char StatusRead(void){
•     char status;
•     RW(ON);
•     DI(OFF);
•     status=(PINA & 0x40);
•     E_Pulse();
•     return(status);
• }
• void WriteDisplayData(char data){
•     RW(OFF);
•     DI(ON);
•     DataInput(data);
• }
• char ReadDisplayData(void){
•     RW(ON);
•     DI(ON);
•     E_Pulse();
•     return(PORTA);
• }
```





- void Reset(void){
- RS(OFF);
- RS(ON);
- E_Pulse();
- }

- void Initgraph(void){
- int i,j;
- CS2(ON);
- CS1(ON);
- DisplayOnOff(ON);
- Reset();
- }





점 찍기



void pixel(char x, char y, char onoff);

- 원하는 점의 X page를 계산한다.
 - $page = x/8$;
- 원하는 점의 그 페이지 내에서의 데이터 위치를 계산한다.
 - $offset = x\%8$; ← Data의 비트 자리
- 현재 자리에 있는 점을 찍는다.
 - 점을 찍을 때 기존에 있던 data와 Or를 해서 찍어야 기존 데이터가 지워지지 않게 된다.
 - 만약 offset이 1이면 $offset_D = 0x01$
 - 2이면 $offset_D = 0x02$
 - 4이면 $offset_D = 0x04$
 - `WriteDisplayData(data | Offset_D);`
- 이 때, CS0과 CS1을 y값(0~127)까지 풀화면으로 쓰기 위해서, 잘 선택해 가면서 y 값도 잘 바뀌 가면서 전체 y 값에서 잘 골라서 쓸 수 있도록 한다. 쓴다.
 - $chipselct = y/64$;
 - `SetAddress(y);`
 - `if(chipselct){`
 - `CS2(ON);`
 - `CS1(OFF);`
 - `}else{`
 - `CS1(ON);`
 - `CS2(OFF);`
 - `}`



선긋기(I)



`void line(int x1, int y1, int x2, int y2);`

- `void line(int x1, int y1, int x2, int y2){`
- `int deltax, deltay, x, y, xinc1,xinc2, yinc1, yinc2,den, num, numadd,numpixels,curpixel;`
- `deltax = fabs(x2 - x1);        // The difference between the x's`
- `deltay = fabs(y2 - y1);        // The difference between the y's`
- `x = x1;                        // Start x off at the first pixel`
- `y = y1;                        // Start y off at the first pixel`
- `if (x2 >= x1){                    // The x-values are increasing`
- `xinc1 = 1;`
- `xinc2 = 1;`
- `}else{                            // The x-values are decreasing`
- `xinc1 = -1;`
- `xinc2 = -1;`
- `}`
- `if (y2 >= y1){                    // The y-values are increasing`
- `yinc1 = 1;`
- `yinc2 = 1;`
- `}else{                            // The y-values are decreasing`
- `yinc1 = -1;`
- `yinc2 = -1;`
- `}`



선긋기(II)



```
void line(int x1, int y1, int x2, int y2);
```

```
.   if (deltax >= deltay){           // There is at least one x-value for every y-value
.   xinc1 = 0;                       // Don't change the x when numerator >= denominator
.   yinc2 = 0;                       // Don't change the y for every iteration
.   den = deltax;
.   num = deltax / 2;
.   numadd = deltay;
.   numpixels = deltax;             // There are more x-values than y-values
.   }else{                           // There is at least one y-value for every x-value
.   xinc2 = 0;                       // Don't change the x for every iteration
.   yinc1 = 0;                       // Don't change the y when numerator >= denominator
.   den = deltay;
.   num = deltay / 2;
.   numadd = deltax;
.   numpixels = deltay;             // There are more y-values than x-values
.   }
.   for (curpixel = 0; curpixel <= numpixels; curpixel++){
.   pixelL(x, y, ON);               // Draw the current pixel
.   num += numadd;                  // Increase the numerator by the top of the fraction
.   if (num >= den){                 // Check if numerator >= denominator
.   num -= den;                     // Calculate the new numerator value
.   x += xinc1;                     // Change the x as appropriate
.   y += yinc1;                     // Change the y as appropriate
.   }
.   x += xinc2;                     // Change the x as appropriate
.   y += yinc2;                     // Change the y as appropriate
.   }
. }
```



원 그리기



`void circle(int xCenter, int yCenter, int radius);`



- 원 그리기 알고리즘 함수는 인터넷 상에 많이 나와 있으므로, 참고하기 바란다.
여기서 보여주는 이 함수는 예제일 뿐이다.
- `void circle(int xCenter, int yCenter, int radius){`
- `int x, y, r2;`
- `r2 = radius * radius;`
- `pixelL(xCenter, yCenter + radius, ON);`
- `pixelL(xCenter, yCenter - radius, ON);`
- `for (x = 1; x <= radius; x++) {`
- `y = (int) (sqrt(r2 - x*x) + 0.5);`
- `pixelL(xCenter + x, yCenter + y, ON);`
- `pixelL(xCenter + x, yCenter - y, ON);`
- `pixelL(xCenter - x, yCenter + y, ON);`
- `pixelL(xCenter - x, yCenter - y, ON);`
- `}`
- `}`



사각형 그리기



void rectangle(int startx, int starty, int endx, int endy);



- 방법 : 주어진 점으로 네 좌표를 만들고,
- 그 점들을 선으로 연결한다.
- void rectangle(int startx, int starty, int endx, int endy){
- line(startx,starty,endx,starty);
- line(endx,starty,endx,endy);
- line(endx,endy,startx,endy);
- line(startx,starty,startx,endy);
- }



Cleardevice



`void cleardevice(void);`

- 방법 : 모든 칩을 선택하고 모든 페이지 모든 좌표에 0x00을 찍는다.
- `void cleardevice(void){`
- `int x=0,y=0;`
- `CS1(ON);`
- `CS2(ON);`
- `for(x=0;x<8;x++){`
- `SetPage(x);`
- `for(y=0;y<128;y++){`
- `WriteDisplayData(0x00);`
- `}`
- `}`
- `}`

