AVR(ATmega128)로 장난하기

AVR(ATmaga128)로 텍스트LCD 장난하기

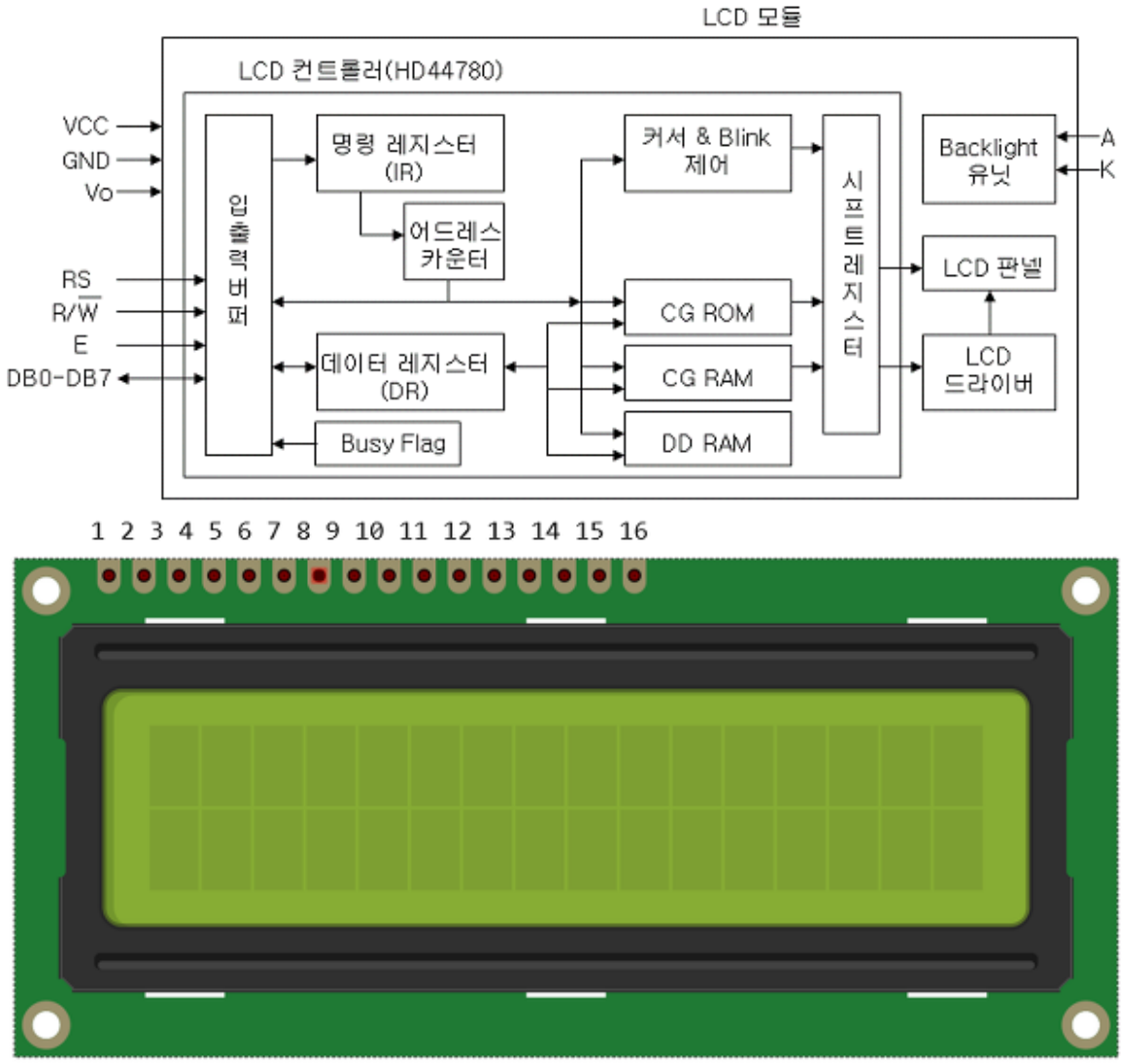
고군~ 2018. 3. 23. 22:40

안녕하세요 고군입니다~

이번 시간에는 텍스트LCD에 대하여 알아보고 AVR을 사용하여 화면을 표시해보도록 하겠습니다.

LCD는 저렴한 가격으로 인해 산업현장에서는 아직도 많이 사용되고 있으며, 표시(Display)장치와 구동(Driver)장치가 하나로 되어 있습니다.

LCD는 8문자 x 2라인, 16 x 2, 16 x 4, 20 x2, 40 x 2라인 등 많은 종류가 있는데 여기서는 16 x 2 LCD에 대해 알아보겠습니다.



[그림1] LCD 모듈구조 및 외형

핀 번호	기호	기 능	
1	Vss	0V	전원
2	VDD	5V	

3	V_L	VR 10k	
4	RS	H : 데이터, L : 인스트력션	
5	R / /W	H : 리드, L : 라이트	
6	E1	H : 인에이블 신호	
7	D0	데이터 버스 4비트 사용시 : D4~D7만 사용 상위 4비트 하위 4비트 8비트 사용시 : D0~D7사용	
8	D1		
9	D2		
10	D3		
11	D4		
12	D5		
13	D6		
14	D7		
15	A	LDE 백라이트 전원	
16	K		

LCD 핀 기능

▶ **V_{SS}**

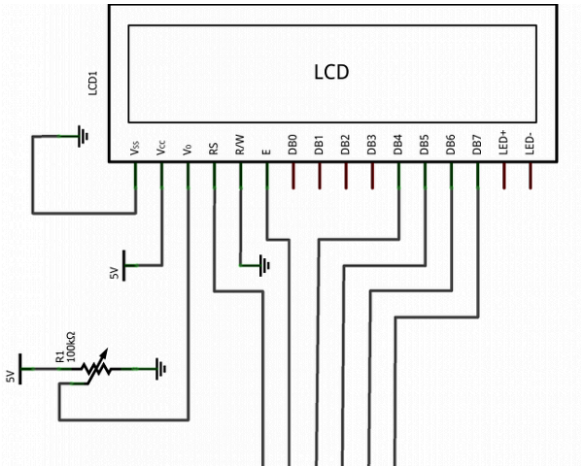
LCD에 전원을 인가하는 단자로, 0V(GND)에 연결합니다.

▶ **V_{DD}**

LCD에 전원을 인가하는 단자로, +5V에 연결합니다.

▶ **V_L**

LCD의 밝기를 조절하는 단자로서, 10kΩ의 가변 저항을 연결하여 밝기를 조정합니다. 밝기 조정을 하지 않으려면 GND에 연결합니다.



▶ RS(Register Select)신호, 핀4

입력 신호를 받는 단자로서 액정 표시 모듈에 데이터 혹은 명령을 읽기/쓰기할 때 사용합니다.

▶ R / /W(Read/Write) 신호, 핀5

입력 신호로서 액정 표시 모듈에 데이터 혹은 명령을 라이트/리드할 때 사용합니다. 리드/라이트 신호 사양은 LCD 제조 회사마다 조금씩 다르다는 것을 유의하여야 합니다.

▶ E1(Enable) 신호, 핀6

이 신호는 Active High입니다. 이것은 GAL 16V8의 입력 어드레스 라인을 조합하여 생성해 낸 출력 어드레스 라인을 이용하여 LCD 인에이블 신호를 선택하도록 합니다.

▶ D0~D7(Data Bus)

데이터 버스로서 CPU와 LCD 사이에 데이터를 주고받기 위한 데이터 버스입니다. 만약 4비트를 사용할 경우에는 D4~D7만 사용합니다.

▶ A

백 라이트의 전원 단자로서 IN4001(다이오드)을 연결하여 전압을 강하시켜서 사용합니다.

▶ K

백 라이트의 전원 단자로서 GND에 연결합니다.

1. LCD 제어기의 내부 구성

명령(Instruction)과 데이터(Data)를 위한 2개의 레지스터, BF(Busy Flag), AC(Address Counter), 문자발생램(CGRAM), 문자발생롬(CGROM), 데이터표시램(DDRAM)이 있습니다.

(1) 레지스터(Register)

▶명령레지스터(IR): DDRAM과 CGRAM에 대한 어드레스와 클리어, 커서시프트 등 제어명령을 보유

▶데이터레지스터(DR): DDRAM과 CGRAM에 쓴 데이터나 읽은 데이터를 일지적으로 저장

레지스터 선택방법: RS(4번핀)과 R/W(5번핀)을 사용하여 선택

RS=0, R/W=0 : IR쓰기(각종 제어 명령 쓰기)

RS=0, R/W=1 : BF읽기, AC읽기

RS=1, R/W=0 : DR쓰기

RS=1, R/W=1 : DR읽기

(2) 비지 플래그(Busy Flag : BF)

1이면 LCD의 콘트롤러가 동작 중으로 명령 수행 불능, 0이면 다음 명령 수행 가능 표시

(3) 어드레스 카운터(Address Counter : AC)

DDRAM과 CGRAM의 주소를 지정할 때 사용하는데 IR에 주소 정보를 쓰면 주소 정보가 AC로 전송되고 DDRAM이나 DDROM에 데이터를 쓰면 AC는 자동으로 +1혹은 -1이 됩니다..

RS	R/W	동작 기능
0	0	IR을 선택하여 제어 명령 쓰기
0	1	D7로부터 비지 플래그를 읽기/어드레스 카운터를 D0~D6으로부터 읽기
1	0	DR 선택하여 데이터 값을 쓰기(DR에서 DD RAM 혹은 CG RAM로)
1	1	DR 선택하여 데이터 값을 읽어 오기(DD RAM 혹은 CG RAM에서 DR로)

(4) 표시 데이터 RAM(Display Data RAM : DD RAM)

80x8비트 용량으로 80개의 8비트 아스키코드를 저장할 수있다. 0x00-0F 주소가 LCD 1행의 1-16번째, 0x40-4F 주소가 LCD 2행의 1-16번째 문자로 표시됩니다.(1행 2행의 번호 불연속임에 주의해야 한다.)

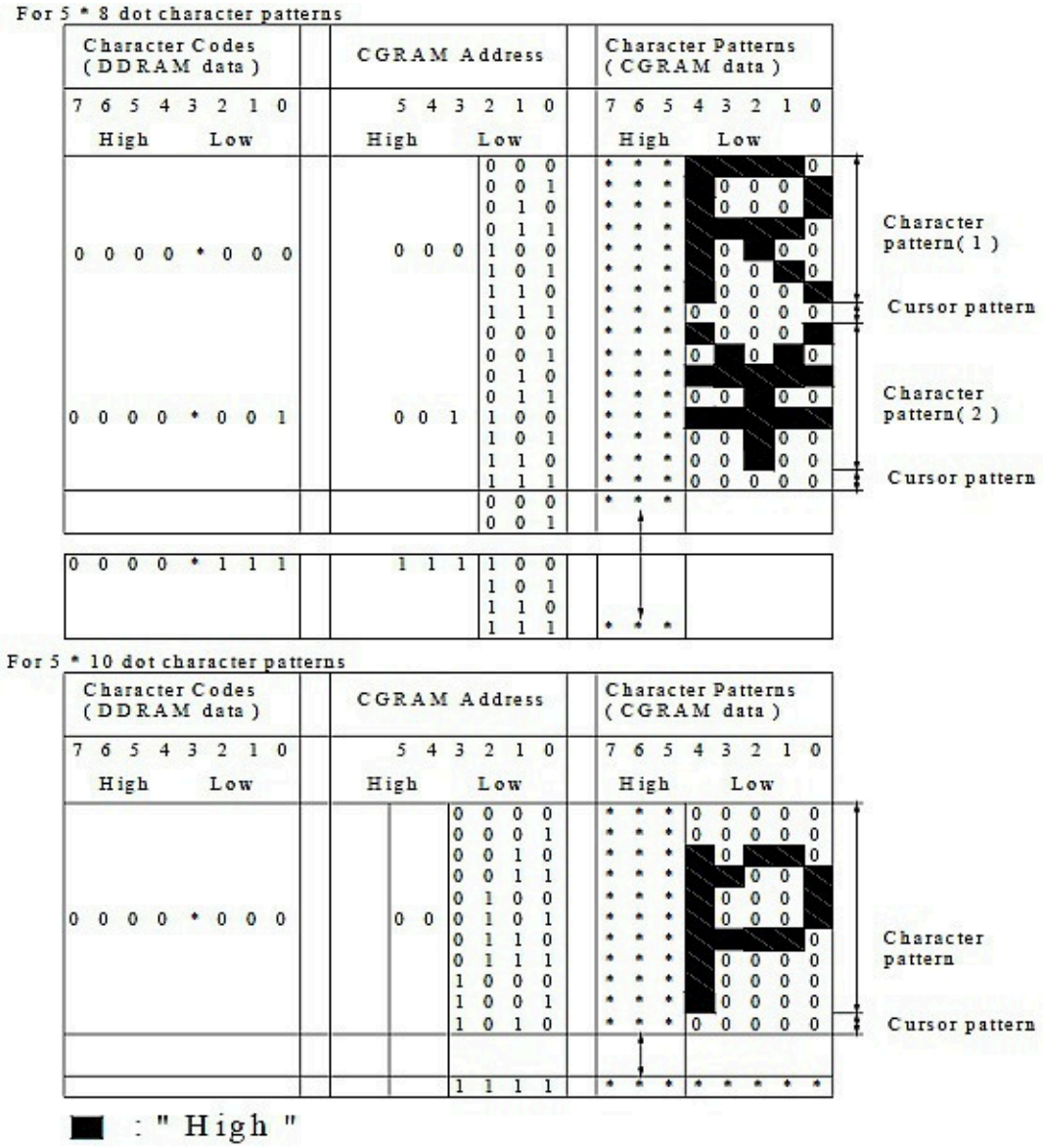
(5) 문자 발생 ROM(Character Generator ROM : CG ROM)

사용자가 자유로이 문자를 만들 때 사용하는 램으로 5x7은 8개, 5x10은 4개 만들어 저장 가능합니다.

(6) 문자 발생 RAM(Character Generator RAM : CG RAM)

5x7, 5x10 도트의 문자를 내장하고 있다. 특수기호, 숫자, 영문자의 문자코드는 아스키코드와 일치합니다.

[그림2] CGROM 문자코드와 문자 패턴사이의 대응관계



[그림3] CGRAM 어드레스, DDRAM와 CGROM 사이의 관계

2. LCD 명령어

실제 텍스트 LCD를 제어 할 때 사용되는 명령어의 상세한 모드 설정과 사용 방법에 대한 내용은 아래 표에 나타나 있습니다.

명령	명령	RS	R/W	데이터 DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	설명	실행 시간
명령 쓰기	화면지우기	0	0	0	0	0	0	0	0	0	1	화면지우기, 커서홈	1,52ms
	커서홈	0	0	0	0	0	0	0	0	1	-	커서 처음 위치로 이동	1,52ms
	엔트리 모드 셋	0	0	0	0	0	0	0	1	I/D	S	어드레스자동증가/감소(I/D) 표시 쉬프트(S)	37us
	표시 On/Off 제어	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 깜박임(B) On/Off	37us
	표시, 커서 쉬프트	0	0	0	0	0	1	S/C	R/L	-	-	표시, 커서 이동	37us
	기능 셋	0	0	0	0	1	DL	N	F	-	-	인터페이스라인(DL), 라인수(N), 문자폰트(F)	37us
	CGRAM 어드레스	0	0	0	1	CGRAM 어드레스(ACG)						CGRAM 어드레스 설정	37us
	DDRAM 어드레스	0	0	1	DDRAM 어드레스(ADD)							DDRAM 어드레스 설정	37us
명령 읽기	비지체크, 어드레스	0	1	BF	어드레스 카운터(AC)							비지플래그 읽기 어드레스 카운터 읽기	0us
데이터 쓰기	데이터 쓰기	1	0	write data								CGRAM 또는 DDRAM에 데이터 쓰기	37us
데이터 읽기	데이터 읽기	1	1	read data								CGRAM 또는 DDRAM에서 데이터 읽기	37us

I/D=1 : 어드레스 자동증가	I/D=0 : 어드레스 자동감소	DDRAM : 표시 데이터 RAM
S=1 : 전체 쉬프트	S=0 : 쉬프트 하지 않음	CGRAM : 폰트 제작 RAM
S/C=1 : 표시 쉬프트	S/C=0 : 커서 이동	ACG : CGRAM 어드레스
R/L=1 : 오른쪽으로 쉬프트	R/L=0 : 왼쪽으로 쉬프트	ADD : DDRAM 어드레스
DL=1 : 8비트	DL=0 : 4비트	AC : 어드레스 카운터
N=1 : 2라인	N=0 : 1라인	(DDRAM, CGRAM 어드레스)
F=1 : 5x10 dots	F=0 : 5x8 dot	
BF=1 : 내부 동작중	BF=0 : 명령/데이터 받기 가능	

텍스트 LCD 명령 표

(1) 디스플레이 클리어(Clear Display)

RS=R/W=0, DB=0b00000001, 실행시간 1.64msec, 표시내용을 소거하고 커서는 홈(1행1열, 0번지)로 돌아감, DDRAM에 스페이스코드(0x20)가 쓰여지고 AC=0으로 됩니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1

(2) 홈 리턴(Return Home)

RS=R/W=0, DB=0b0000001x, 실행시간 1.64msec, DDRAM의 AC=0x00로 세트되어 커서가 홈으로 가며 DDRAM의 내용은 변동 없습니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	-

(3) 엔트리 모드 설정(Entry Mode Set)

커서의 이동방향과 표시 내용의 시프트 여부 설정, 실행시간 0.04msec

RS=R/W=0, DB=0b00000101 : DDRAM에 데이터를 써넣은 후 표시 전체를 좌로 이동, 커서는 이동 안합니다.

RS=R/W=0, DB=0b00000111 : DDRAM에 데이터를 써넣은 후 표시 전체를 우로 이동, 커서는 이동 안합니다.

RS=R/W=0, DB=0b00000100 : 표시를 이동하지 않고 AC를 감소시키고 커서가 좌측으로 이동하게 합니다.

표시를 이동하지 RS=R/W=0, DB=0b00000110 : 않고 AC를 증가시키고 커서가 우측으로 이동하게 합니ㅏㄷ

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	I/D	S

(4) 디스플레이 제어(Display On/Off Control)

커서나 전체표시의 온오프제어 및 커서위치 문자의 깜박임 제어, 실행시간 0.04msec RS=R/W=0, DB=0b00001dcb : d=1일때 전체 온, c=1 커서 온, b=1 커서 위치 문자 점멸합니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	C	B

(5) 커서 또는 디스플레이 시프트(Cursor or Display Shift)

DDRAM의 내용을 변경시키지 않은 상태에서 커서를 움직이거나 전체문자를 이동, 실행시간0.04msec

RS=R/W=0, DB=0b000100xx : AC를 1감소시키고 커서를 좌로 이동

RS=R/W=0, DB=0b000101xx : AC를 1증가시키고 커서를 우로 이동

RS=R/W=0, DB=0b000110xx : 모든 표시와 커서를 좌로 이동

RS=R/W=0, DB=0b000111xx : 모든 표시와 커서를 우로 이동

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	S/C	R/L	-	-

(6) 기능 설정(Function Set)

표시행수와 문자 폰트, 인터페이스 길이를 설정, 실행시간 0.04msec

RS=R/W=0, DB=0b0011NFxx : 8비트 인터페이스

RS=R/W=0, DB=0b0010NFxx : 4비트 인터페이스

여기에서 NF=00(1행 5x7 폰트), NF=01(1행 5x10 폰트), NF=1x(2행 5x7 폰트)

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	DL	N	F	-	-

(7) CGRAM 어드레스 설정(Set CGRAM Address)

CGRAM의 6비트 주소를 설정, 설정 후 전송 데이터는 CGRAM의 데이터로 취급됨, 실행시간 0.04msec RS=R/W=0, DB=0b01xxxxxx : xxxxxx 부분의 값이 AC로 설정됩니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	AC(Address Counter)					

(8) DDRAM 어드레스 설정(Set DDRAM Address)

DDRAM의 7비트 주소를 설정, 설정 후 전송 데이터는 DDRAM의 데이터로 취급됨, 실행시간 0.04msec RS=R/W=0, DB=0b1xxxxxxx : xxxxxxx 부분의 값이 AC로 설정됨

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	AC(Address Counter)						

(9) 비지 플래그와 어드레스 읽기(Read Busy Flag & Address)

BF의 값이나 AC의 값을 읽어 들인다. 실행시간 0.04msecRS=0, R/W=1 : DB7이 BF의 값이고 DB6-0가 AC값입니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0

0	1	BF	AC(Address Counter)

(10) CGRAM 또는 DDRAM에 데이터 쓰기(Write Data to CGRAM or DDRAM)

이전에 주소가 설정된 램에 쓰고 엔트리 모드에 따라 주소값이 +1 혹은 -1 RS=1, R/W=0 : 기능설정의 인터페이스 길이에 따라 DB를 통해 쓴다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	Write Data							

(10) CGRAM 또는 DDRAM에 데이터 읽기(Read Data from CGRAM or DDRAM)

읽기 전에 주소설정 명령을 해야 함, 안 그러면 두 번째 데이터부터 유효 RS=1, R/W=1 : 기능설정의 인터페이스 길이에 따라 DB를 통해 읽습니다.

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	1	Read Data							

3. LCD 초기화

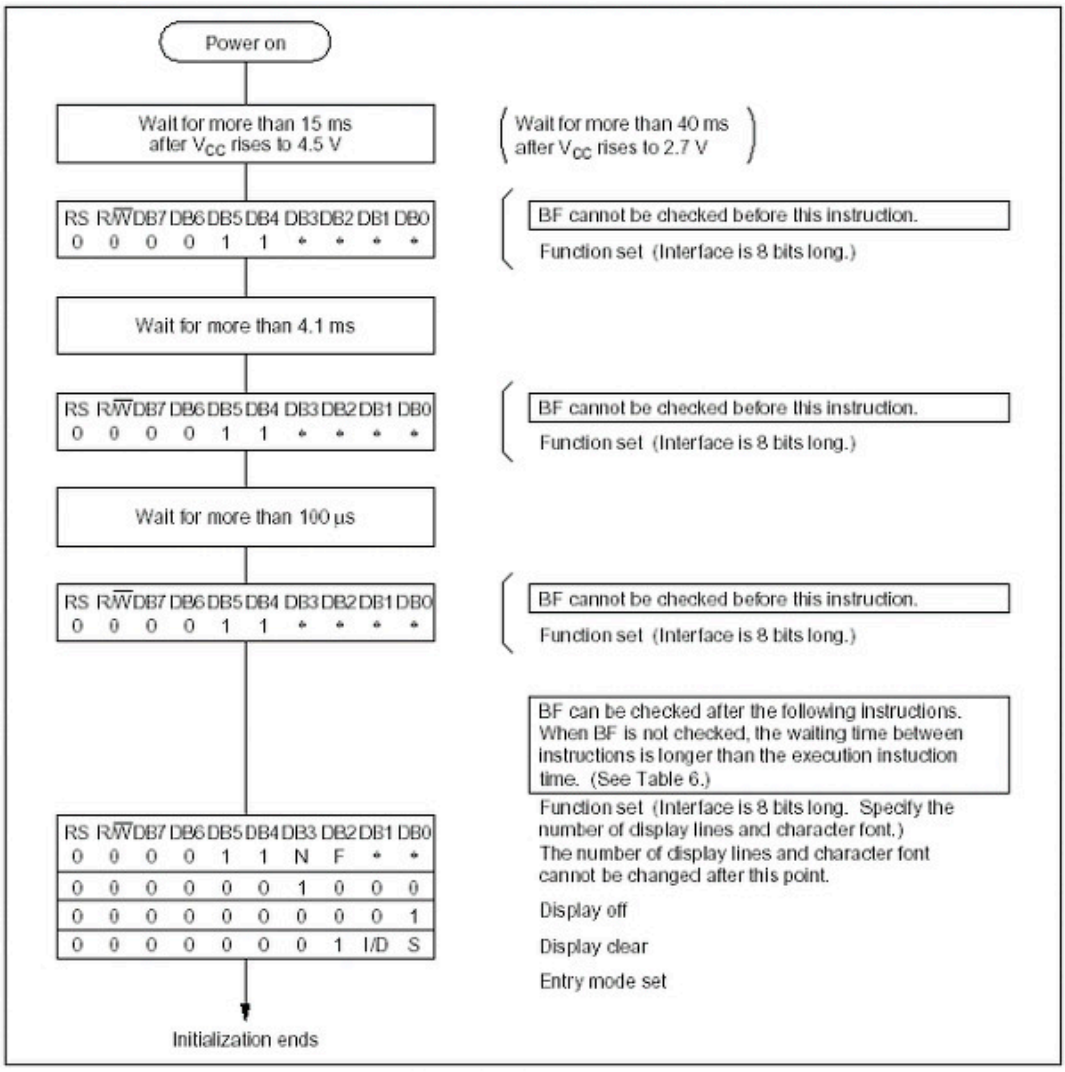


Figure 23 8-Bit Interface

텍스트 LCD를 작동시키기 위해서는 위와 같은 초기화 과정이 필요합니다. 위 그림은 데이타시트에서 발취해 놓은 것으로 제조사마다 차이는 있을 수 있으나 거의가 비슷하다고 생각하시면 됩니다. 프로그램 상에는 아래와 같은 순서로 코드를 넣어 초기화 시켜주면 되는데 중간에 지연시간에 주의해서 작성해야 합니다.

```cs
// 텍스트 LCD를 초기화하는 함수
void LCD_Init(void){
    _delay_ms(100);
        // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_ms(10);
        // 비지 플래그를 체크하지 않는 Function Set
```

```
    LCD_wCommand(0x38);
    _delay_us(200);
        // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_us(200);
        // 비지 플래그를 체크하는 Function Set
    LCD_wBCommand(0x38);
        // 비지 플래그를 체크하는 Display On/Off Control
    LCD_wBCommand(0x0c);
        // 비지 플래그를 체크하는 Clear Display
    LCD_wBCommand(0x01);
}

                                        Colored by Color Scripter
```

4. 텍스트LCD 제어에 사용되는 디지털 출력 레지스터

-DDRx : PORTx 데이터 방향 레지스터

bit	7	6	5	4	3	2	1	0
	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0
Red/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

-PORTx : PORTx 데이터 방향 레지스터

bit	7	6	5	4	3	2	1	0
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0
Red/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

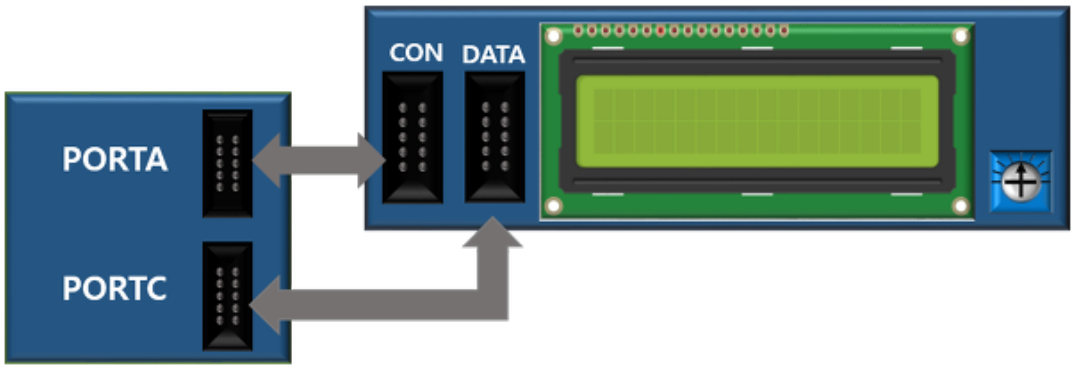
-PINx : PORTx 입력 핀 어드레스

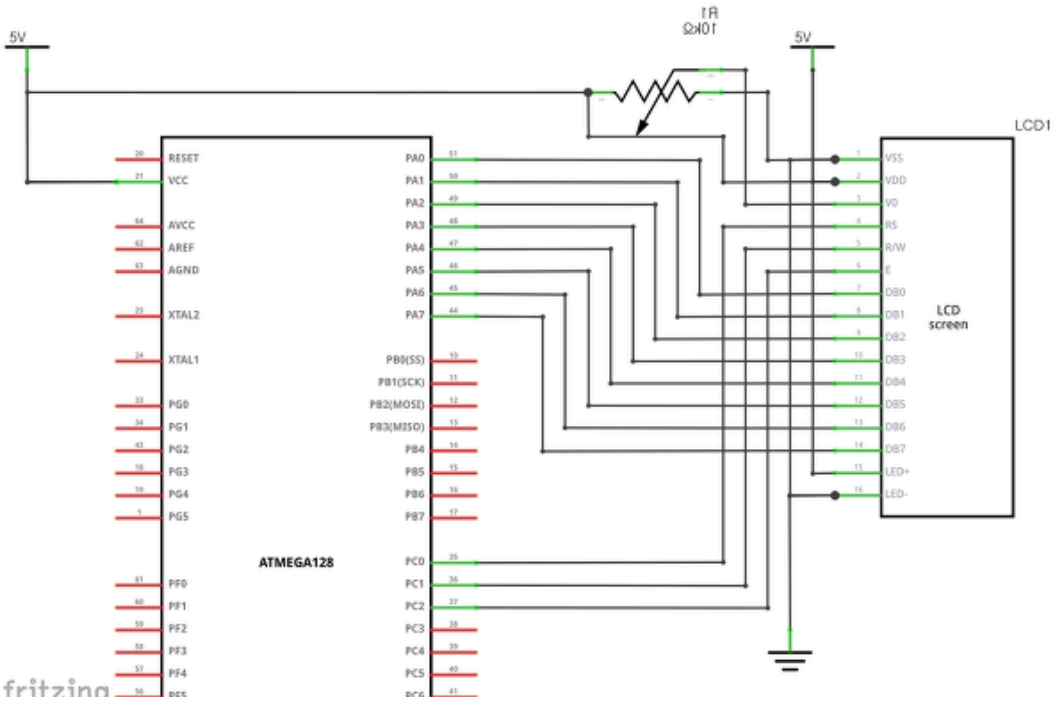
bit	7	6	5	4	3	2	1	0
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0
Red/Write	R	R	R	R	R	R	R	R
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

텍스트LCD **제어를 위한 입/출력 관련 레지스터**

DDRx 레지스터는 포트의 입출력 방향을 결정하는 레지스터이며 DDRxndml 비트값이 0이면 입력, 1이면 출력으로 설정됩니다. Atmega128에서 레지스터는 0~7번까지 8비트가 존재합니다. PORTx 레지스터는 포트에 출력할 논리 값을 저장하는 레지스터이며 x는 포트명입니다. PINx는 DDRx레지스터, PORTx 레지스터 등과는 달리 포트의 논리를 읽을 수 있는 어드레스이면 x는 포트명입니다.

5. AVR 연습용 메인 키트에서 동작시켜 보기





위 그림은 16핀으로 구성된 8비트 제어가 가능한 텍스트 LCD입니다. 텍스트 LCD 제어에서 데이터 핀을 4비트로도 제어가 가능하므로 자신이 구성한 하드웨어에 맞춰 설계을 하면 됩니다. 제어를 위해 CON, DATA 커넥터와 VR이 연결되어 있습니다. CON 커넥터에는 RS, RW, E 신호가 연결되고, DATA 커넥터에는 D0~D7까지의 8비트 데이터 버스가 연결됩니다. 텍스트 LCD 배경과 글자의 대비를 조절하기 위해 가변저항이 연결되었습니다.

PORTA	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Device Pin	D7	D6	D5	D4	D3	D2	D1	D0
In/Out	In/Out	In/Out	In/Out	In/Out	In/Out	In/Out	In/Out	In/Out
PORTC	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Device Pin	-	-	-	-	-	E	RW	RS
In/Out	-	-	-	-	-	Out	Out	Out

텍스트 LCD 출력 장치 실험을 위한 회로 연결

>>> (예제 1.) 텍스트 LCD의 어드레스 카운터를 설정하여 문자열을 텍스트LCD에 출력하기.

[단계 1] AVR Studio4로 Project를 생성하고, C언어 소스 작성하기 링크

AVR Studio4 편집 화면에 아래 C언어 소스 프로그램을 작성하여 화일명.c로 저장합니다.


```cs
// ATmega128의 레지스터 등이 정의되어 있음
#include <avr/io.h>

// _delay_ms() 함수 등이 정의되어 있음
#include <util/delay.h>

#define sbi(x, y) (x |= (1 << y))  // x의 y 비트를 설정(1)
#define cbi(x, y) (x &= ~(1 << y)) // x의 y 비트를 클리어(0)

// CON 포트는 포트 C와 연결됨을 정의
#define LCD_CON      PORTC
// DATA 포트는 포트 A와 연결됨을 정의
#define LCD_DATA     PORTA
// DATA 포트의 출력 방향 설정 매크로를 정의
```

```
#define LCD_DATA_DIR DDRA
// DATA 포트의 입력 방향 설정 매크로를 정의
#define LCD_DATA_IN  PINA
// RS 신호의 비트 번호 정의
#define LCD_RS   0
// RW 신호의 비트 번호 정의
#define LCD_RW   1
// E 신호의 비트 번호 정의
#define LCD_E    2

// 텍스트 LCD로 부터 상태(명령)를 읽는 함수
unsigned char LCD_rCommand(void){
    unsigned char temp=1;

    LCD_DATA_DIR = 0X00;

    cbi(LCD_CON, LCD_RS); // 0번 비트 클리어, RS = 0, 명령
    sbi(LCD_CON, LCD_RW); // 1번 비트 설정, RW = 1, 읽기
    sbi(LCD_CON, LCD_E);  // 2번 비트 설정, E = 1
    _delay_us(1);

    temp = LCD_DATA_IN;      // 명령 읽기
    _delay_us(1);

    cbi(LCD_CON, LCD_E);  // 명령 읽기 동작 끝

    LCD_DATA_DIR = 0XFF;
    _delay_us(1);

     return temp;
}

// 텍스트 LCD의 비지 플래그 상태를 확인하는 함수
char LCD_BusyCheck(unsigned char temp){
    if(temp & 0x80)          return 1;
    else           return 0;
}

// 텍스트 LCD에 명령을 출력하는 함수 - 단, 비지플래그 체크하지 않음
void LCD_wCommand(char cmd){
    cbi(LCD_CON, LCD_RS); // 0번 비트 클리어, RS = 0, 명령
    cbi(LCD_CON, LCD_RW); // 1번 비트 클리어, RW = 0, 쓰기
    sbi(LCD_CON, LCD_E);  // 2번 비트 설정, E = 1

    LCD_DATA = cmd;          // 명령 출력
    _delay_us(1);
    cbi(LCD_CON, LCD_E);  // 명령 쓰기 동작 끝

    _delay_us(1);
}

// 텍스트 LCD에 명령을 출력하는 함수 - 단, 비지플래그 체크함
```

```
67  void LCD_wBCommand(char cmd){
68      while(LCD_BusyCheck(LCD_rCommand()))
69      _delay_us(1);
70      cbi(LCD_CON, LCD_RS); // 0번 비트 클리어, RS = 0, 명령
71      cbi(LCD_CON, LCD_RW); // 1번 비트 클리어, RW = 0, 쓰기
72      sbi(LCD_CON, LCD_E);  // 2번 비트 설정, E = 1
73
74      LCD_DATA = cmd;          // 명령 출력
75      _delay_us(1);
76      cbi(LCD_CON, LCD_E);  // 명령 쓰기 동작 끝
77
78      _delay_us(1);
79  }
80
81  // 텍스트 LCD를 초기화하는 함수
82  void LCD_Init(void){
83      _delay_ms(100);
84          // 비지 플래그를 체크하지 않는 Function Set
85      LCD_wCommand(0x38);
86      _delay_ms(10);
87          // 비지 플래그를 체크하지 않는 Function Set
88      LCD_wCommand(0x38);
89      _delay_us(200);
90          // 비지 플래그를 체크하지 않는 Function Set
91      LCD_wCommand(0x38);
92      _delay_us(200);
93
94          // 비지 플래그를 체크하는 Function Set
95      LCD_wBCommand(0x38);
96          // 비지 플래그를 체크하는 Display On/Off Control
97      LCD_wBCommand(0x0c);
98          // 비지 플래그를 체크하는 Clear Display
99      LCD_wBCommand(0x01);
100 }
101
102 // 텍스트 LCD에 1바이트 데이터를 출력하는 함수
103 void LCD_wData(char dat){
104     while(LCD_BusyCheck(LCD_rCommand()))
105         _delay_us(1);
106
107     sbi(LCD_CON, LCD_RS); // 0번 비트 설정, RS = 1, 데이터
108     cbi(LCD_CON, LCD_RW); // 1번 비트 클리어, RW = 0, 쓰기
109      sbi(LCD_CON, LCD_E); // 2번 비트 설정, E = 1
110
111     LCD_DATA = dat;        // 데이터 출력
112     _delay_us(1);
113     cbi(LCD_CON, LCD_E);  // 데이터 쓰기 동작 끝
114
115     _delay_us(1);
116 }
117
118 // 텍스트 LCD에 문자열을 출력하는 함수
```

```
119 void LCD_wString(char *str){
120     while(*str)
121         LCD_wData(*str++);
122 }
123 
124 // C 언어의 주 실행 함수
125 int main(void){
126     // 포트 A의 방향 설정, 0 : 입력, 1 : 출력
127     DDRA = 0B11111111;
128 
129     // 포트 C의 방향 설정, 0 : 입력, 1 : 출력
130     DDRC = 0B11111111;
131 
132     LCD_Init();          // 텍스트 LCD 초기화 - 함수 호출
133 
134 
135     LCD_wBCommand(0x80 | 0x00);  // DDRAM Address = 0 설정
136     LCD_wString("TEXT LCD");     // 텍스트 LCD 문자열 출력
137 
138 
139     LCD_wBCommand(0x80 | 0x40);  // DDRAM Address = 0x40 설정
140     LCD_wString("HELLO WORLD!"); // WESNET 문자열 출력
141 
142     // 함수의 형태와 같이 정수형(int)의 값을 반환함
143     return 1;
144 }
```

Colored by Color Scripter

[단계 2]AVR Studio 4로 C언어 소스프로그램을 빌드한다. 링크

[단계 3]AVR Studio 4로 타겟보드에 라이팅을 한다. 링크

'AVR(ATmega128)로 장난하기' 카테고리의 다른 글

AVR(ATmaga128)로 FND 4Digt 7세그먼트 전자회로 장난하기 (3)	2018.03.23
AVR(ATmaga128)로 FND 1Digt 7세그먼트 전자회로 장난하기 (0)	2018.03.23
AVR(ATmaga128)로 LED 매트릭스 장난하기 (0)	2018.03.23
AVR(ATmaga128)로 8스위치 장난하기 (0)	2018.03.23