# 영어단어 학습기

## 목표

2행 16열 문자LCD를 이용해서, 학습용 단어를 첫 행에 나타내고 그 뜻 풀이를 16문자 이내로 설명해서 둘째 행에 나타내게 하는 학습용 응용 프로젝트를 실행한다.

## 주요 필요 부품

ATmega328P, 2행 16열 문자LCD, 토글버튼

# LCD (2행16열 문자표시기)를 MCU(ATmega328P) 로 구동하기

## LCD사용 디자인 전제

### MCU 디지털포트 자원 절약: 4비트 데이터 신호방식

문자표시용LCD (그래픽 표시용LCD도 있다)의 데이터는 4 또는 8비트로 제어가능한데, MCU의 디지털포트를 최소 사용하면서 구현하기 위해 여기선 4비트제어 방식을 활용

### LCD전용 사용: Busy 플래그 안씀

단일 프로세스(즉, 싱글 태스킹 sigle tasking)로 LCD를 구동할 것이므로, 여러 프로세스가 동시에 접근하는 상황을 염두에 두지 않아도 됨. 따라서, LCD를 사용해도 되는지 확인 하지 않아도 되고 코딩시 늘 LCD에 데이터를 쓸 수 있다고 여김. 따라서, Busy플래그 사용 않음.

### LCD에 데이터 쓰기(출력) 만 사용

LCD에 표시된 내용이 무엇인지 읽는 기능, 즉 입력기능은 사용하지 않음. 따라서, LCD RW 핀(pin 5)은 항상 GND에 묶여 있어야 함. 왜냐하면, Low신호 일 때 쓰기(Write) 기능으로 동작하기 때문.

All time delays are longer than those specified in most datasheets

in order to accommodate slower than normal LCD modules. This

requirement is well documented but almost always ignored. The

information is in a note at the bottom of the right hand

(Execution Time) column of the instruction set.

## ATmega328P 사용 디자인 전제

### HW개발환경

만능기판(5\*7cm)에 DIP(Dual In-line Package)타입의 ATmega328P를 28핀 소켓에 올려 사용.

USB tiny programmer를 이용해서 컴파일된 hex코드를 호스트PC에서 부터 MCU로 업로드, 이 때 부트로더(boot loader)는 필요없음.

### SW개발환경

# 프로그램 모듈 구성

## main.c

### main()

### config()

//목적: LCD의 제어모듈 초기화

/\* 아래와 같이 MCU와 LCD제어핀을 연결함

----------- ----------

| ATmega328 | | LCD |

| PB0| |K |

| PD7|---------------->|D7 |

| PD6|---------------->|D6 |

| PD5|---------------->|D5 |

| PD4|---------------->|D4 |

| | |D3 |

| | |D2 |

| | |D1 |

| | |D0 |

| | | |

| PD3|---------------->|E |

| | GND --->|RW |

| PD2|---------------->|RS |

----------- ----------

\*/

void config()

{

/\*

LCD의 4비트 데이터 신호용, MCU(Atmega328P) 디지털핀 배정하고, 출력모드로 선언

defines.h참조

lcd\_D7\_ddr - lcd\_D4\_ddr 은 DDRD로 정의

lcd\_D7\_bit - lcd\_D4\_bit 은 PORTD7 – PORTD4로 정의

DDRD(Data Direction Register D) 즉, D포트그룹의 데이터방향 레지스터의 각 비트를 1로 하면 출력으로 선언됨. 비트를 1로 설정하는 매크로 \_BV()와 비트연산자 OR(|)를 활용함에 유의.

\*/

lcd\_D7\_ddr |= \_BV(lcd\_D7\_bit);

lcd\_D6\_ddr |= \_BV(lcd\_D6\_bit);

lcd\_D5\_ddr |= \_BV(lcd\_D5\_bit);

lcd\_D4\_ddr |= \_BV(lcd\_D4\_bit);

// LCD 백라이트 캐소드 핀(cathode pin (K))을 출력용으로 선언

lcd\_Backlight\_ddr |= \_BV(lcd\_Backlight\_bit);

// LCD 백라이트 끄기. 0이되어야 그라운드가 되어 전류가 흘러 백라이트 켜짐

// 따라서, 1을 쓰면 백라이트 꺼짐

lcd\_Backlight\_port |= \_BV(lcd\_Backlight\_bit);

/\*

토글스위치는 입력용으로 선언

defines.h참조

tactile\_Switch\_ddr 은 DDRB로 정의

tactile\_Switch\_bit 은 PORTB4로 정의

DDRB(Data Direction Register B) 즉, B포트그룹의 데이터방향 레지스터의 각 비트를 0으로

하면, 입력으로 선언됨. 비트를 1로 설정하는 매크로 \_BV()의 부호를 바꾸는 ~ 비트연산자와

비트단위 AND연산을 하기 위해 비트연산자 &를 활용함에 유의.

\*/

tactile\_Switch\_ddr &= ~\_BV(tactile\_Switch\_bit);

// LCD의 기타 제어핀 설정

// E line - output

lcd\_E\_ddr |= \_BV(lcd\_E\_bit);

// RS line - output

lcd\_RS\_ddr |= \_BV(lcd\_RS\_bit);

// LCD VDD pin - Output

//lcd\_VDD\_ddr |= \_BV(lcd\_VDD\_bit);

//turn off LCD VDD

//lcd\_VDD\_port &= ~\_BV(lcd\_VDD\_bit);

//turn on LCD VDD

//lcd\_VDD\_port |= \_BV(lcd\_VDD\_bit);

// initialize the LCD controller as determined by the defines (LCD instructions)

// initialize the LCD display for a 4-bit interface

lcd\_init\_4d();

}//config

### lcd\_init\_4d()

/\*============================== 4-bit LCD Functions ======================\*/

/\*

함수명: lcd\_init\_4d

목적: LCD모듈을 4비트 데이터 인터페이스로 사용하기 위해 초기화

Entry: equates (LCD instructions) set up for the desired operation

Exit: no parameters

참고: LCD모듈 사용가능 여부를 판단하기 위한 busy플래그 대신, 적정 시간지연을 활용

\*/

void lcd\_init\_4d(void)

{

// Power-up delay

// initial 40 mSec delay

\_delay\_ms(100);

// IMPORTANT - At this point the LCD module is in the 8-bit mode and it is expecting to receive

// 8 bits of data, one bit on each of its 8 data lines, each time the 'E' line is pulsed.

//

// Since the LCD module is wired for the 4-bit mode, only the upper four data lines are connected to

// the microprocessor and the lower four data lines are typically left open. Therefore, when

// the 'E' line is pulsed, the LCD controller will read whatever data has been set up on the upper

// four data lines and the lower four data lines will be high (due to internal pull-up circuitry).

//

// Fortunately the 'FunctionReset' instruction does not care about what is on the lower four bits so

// this instruction can be sent on just the four available data lines and it will be interpreted

// properly by the LCD controller. The 'lcd\_write\_4' subroutine will accomplish this if the

// control lines have previously been configured properly.

// Set up the RS and E lines for the 'lcd\_write\_4' subroutine.

lcd\_RS\_port &= ~\_BV(lcd\_RS\_bit); // select the Instruction Register (RS low)

lcd\_E\_port &= ~\_BV(lcd\_E\_bit); // make sure E is initially low

// Reset the LCD controller

lcd\_write\_4(lcd\_FunctionReset); // first part of reset sequence

\_delay\_ms(10); // 4.1 mS delay (min)

lcd\_write\_4(lcd\_FunctionReset); // second part of reset sequence

\_delay\_us(200); // 100uS delay (min)

lcd\_write\_4(lcd\_FunctionReset); // third part of reset sequence

\_delay\_us(200); // this delay is omitted in the data sheet

// Preliminary Function Set instruction - used only to set the 4-bit mode.

// The number of lines or the font cannot be set at this time since the controller is still in the

// 8-bit mode, but the data transfer mode can be changed since this parameter is determined by one

// of the upper four bits of the instruction.

lcd\_write\_4(lcd\_FunctionSet4bit); // set 4-bit mode

\_delay\_us(DELAY\_INST); // 40uS delay (min)

// Function Set instruction

lcd\_write\_instruction\_4d(lcd\_FunctionSet4bit); // set mode, lines, and font

\_delay\_us(DELAY\_INST); // 40uS delay (min)

// The next three instructions are specified in the data sheet as part of the initialization routine,

// so it is a good idea (but probably not necessary) to do them just as specified and then redo them

// later if the application requires a different configuration.

// Display On/Off Control instruction

lcd\_write\_instruction\_4d(lcd\_DisplayOff); // turn display OFF

\_delay\_us(DELAY\_INST); // 40uS delay (min)

// Clear Display instruction

lcd\_write\_instruction\_4d(lcd\_Clear); // clear display RAM

\_delay\_ms(4); // 1.64 mS delay (min)

// ; Entry Mode Set instruction

lcd\_write\_instruction\_4d(lcd\_EntryMode); // set desired shift characteristics

\_delay\_us(DELAY\_INST); // 40uS delay (min)

// This is the end of the LCD controller initialization as specified in the data sheet, but the display

// has been left in the OFF condition. This is a good time to turn the display back ON.

// Display On/Off Control instruction

lcd\_write\_instruction\_4d(lcd\_DisplayOn); // turn the display ON

\_delay\_us(DELAY\_INST); // 40uS delay (min)

}

### lcd\_write\_string\_4d()

/\*

함수명: lcd\_write\_string\_4d

목적: LCD에 문자열 표시

Entry: 표시하고 싶은 문자열(바이트의 배열)

Exit: no parameters

참고: LCD모듈 사용가능 여부를 판단하기 위한 busy플래그 대신, 적정 시간지연을 활용

\*/

void lcd\_write\_string\_4d(uint8\_t theString[])

{

volatile int i = 0; // character counter\*/

while (theString[i] != 0)

{

lcd\_write\_character\_4d(theString[i]);

i++;

\_delay\_us(DELAY\_INST); // 40 uS delay (min)

}

}

### lcd\_write\_character\_4d()

/\*

함수명: lcd\_write\_character\_4d

목적: LCD에 문자 표시

Entry: 표시하고 싶은 문자 (바이트)

Exit: no parameters

참고: LCD모듈 사용가능 여부를 판단하기 위한 busy플래그 대신, 적정 시간지연을 활용

\*/

void lcd\_write\_character\_4d(uint8\_t theData)

{

lcd\_RS\_port |= \_BV(lcd\_RS\_bit); // select the Data Register (RS high)

lcd\_E\_port &= ~\_BV(lcd\_E\_bit); // make sure E is initially low

lcd\_write\_4(theData); // write the upper 4-bits of the data

lcd\_write\_4(theData << 4); // write the lower 4-bits of the data

}

### lcd\_write\_instruction\_4d()

/\*

함수명: lcd\_write\_instruction\_4d

목적: LCD에 명령어 전달 (바이트)

Entry: 전달하고 싶은 명령어 (바이트)

Exit: no parameters

참고: LCD모듈 사용가능 여부를 판단하기 위한 busy플래그 대신, 적정 시간지연을 활용

\*/

void lcd\_write\_instruction\_4d(uint8\_t theInstruction)

{

lcd\_RS\_port &= ~\_BV(lcd\_RS\_bit); // select the Instruction Register (RS low)

lcd\_E\_port &= ~\_BV(lcd\_E\_bit); // make sure E is initially low

lcd\_write\_4(theInstruction); // write the upper 4-bits of the data

lcd\_write\_4(theInstruction << 4); // write the lower 4-bits of the data

}

### lcd\_write\_4()

/\*

함수명: lcd\_write\_4

목적: LCD에 바이트 정보 전달

Entry: 전달하고 싶은 바이트 정보 (바이트)

RS is configured for the desired LCD register

E is low

RW is low

Exit: no parameters

참고: LCD모듈 사용가능 여부를 판단하기 위한 busy플래그 대신, 적정 시간지연을 활용

\*/

void lcd\_write\_4(uint8\_t theByte)

{

lcd\_D7\_port &= ~\_BV(lcd\_D7\_bit); // assume that data is '0'

if (theByte & 1<<7) lcd\_D7\_port |= \_BV(lcd\_D7\_bit); // make data = '1' if necessary

lcd\_D6\_port &= ~\_BV(lcd\_D6\_bit); // repeat for each data bit

if (theByte & 1<<6) lcd\_D6\_port |= \_BV(lcd\_D6\_bit);

lcd\_D5\_port &= ~\_BV(lcd\_D5\_bit);

if (theByte & 1<<5) lcd\_D5\_port |= \_BV(lcd\_D5\_bit);

lcd\_D4\_port &= ~\_BV(lcd\_D4\_bit);

if (theByte & 1<<4) lcd\_D4\_port |= \_BV(lcd\_D4\_bit);

// write the data

// 'Address set-up time' (40 nS)

lcd\_E\_port |= \_BV(lcd\_E\_bit); // Enable pin high

\_delay\_us(1); // implement 'Data set-up time' (80 nS) and 'Enable pulse width' (230 nS)

lcd\_E\_port &= ~\_BV(lcd\_E\_bit); // Enable pin low

\_delay\_us(1); // implement 'Data hold time' (10 nS) and 'Enable cycle time' (500 nS)

}

initINT()

# 프로그램 컴파일 및 업로드

## 컴파일

## 업로드

