Chapter 6

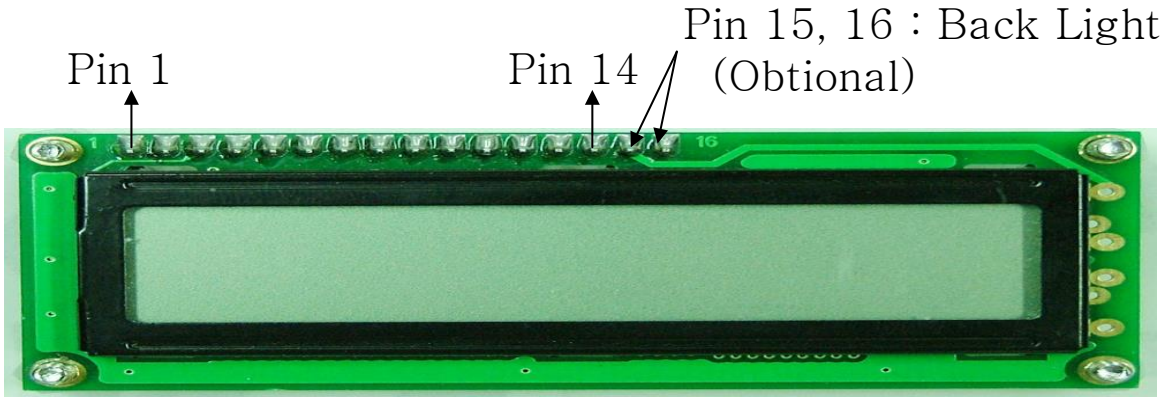
Driving
LCD Character Display Module

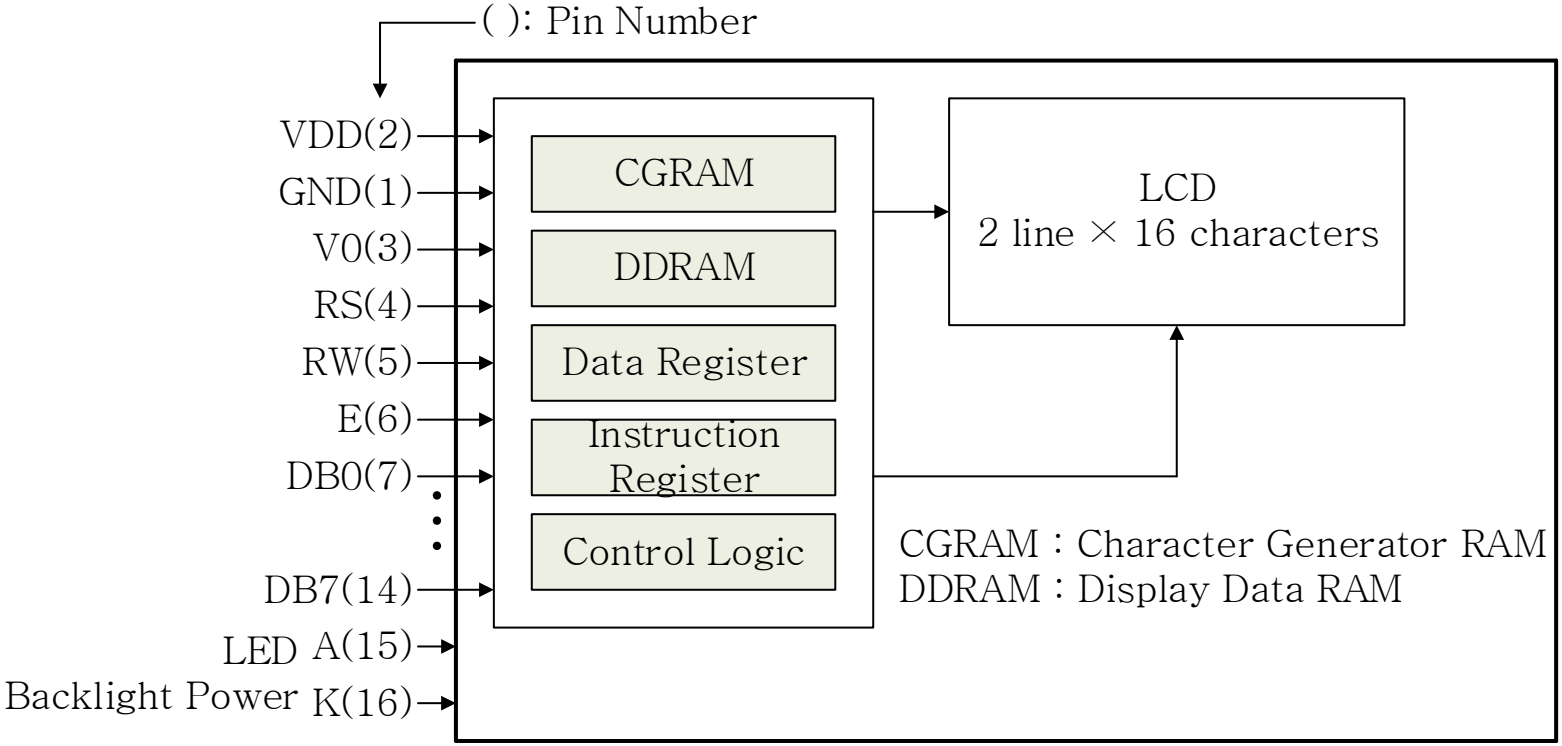
Objectives

- Understanding the structure of LCD Character display module
- Understanding how to interface LCD module with Microcontroller
- Understanding how to configure the functions of LCD module by programming the control registers of the module
- Understanding how to develop sub-functions required for driving the LCD module properly
- Practice with the LCD module

❑ Understanding the LCD module

- Structure and Functions of the Pins



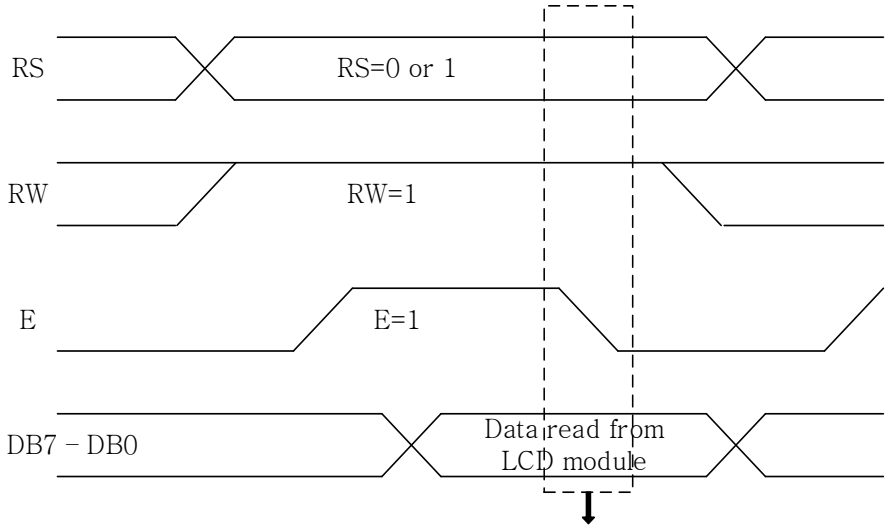


❑ Understanding the LCD module

▪ Operation of LCD module

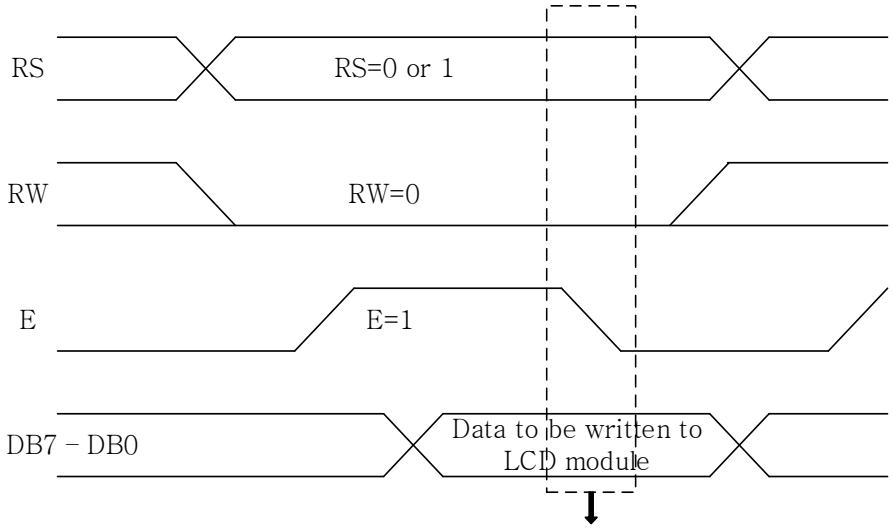
표 6.1 LCD Operations (C; Command, D: Data, W: Write, R: Read)

RS(pin 4)	RW(pin 5	Operation
0(C)	0(W)	Write Command to LCD module
0(C)	1(R)	Read Status from LCD module
1(D)	0(W)	Write ASCII code data to LCD module
1(D)	1(R)	Read ASCII code data from LCD module



Here LCD data is loaded to Microcontroller to complete the read operation

<Read Timing Diagram>

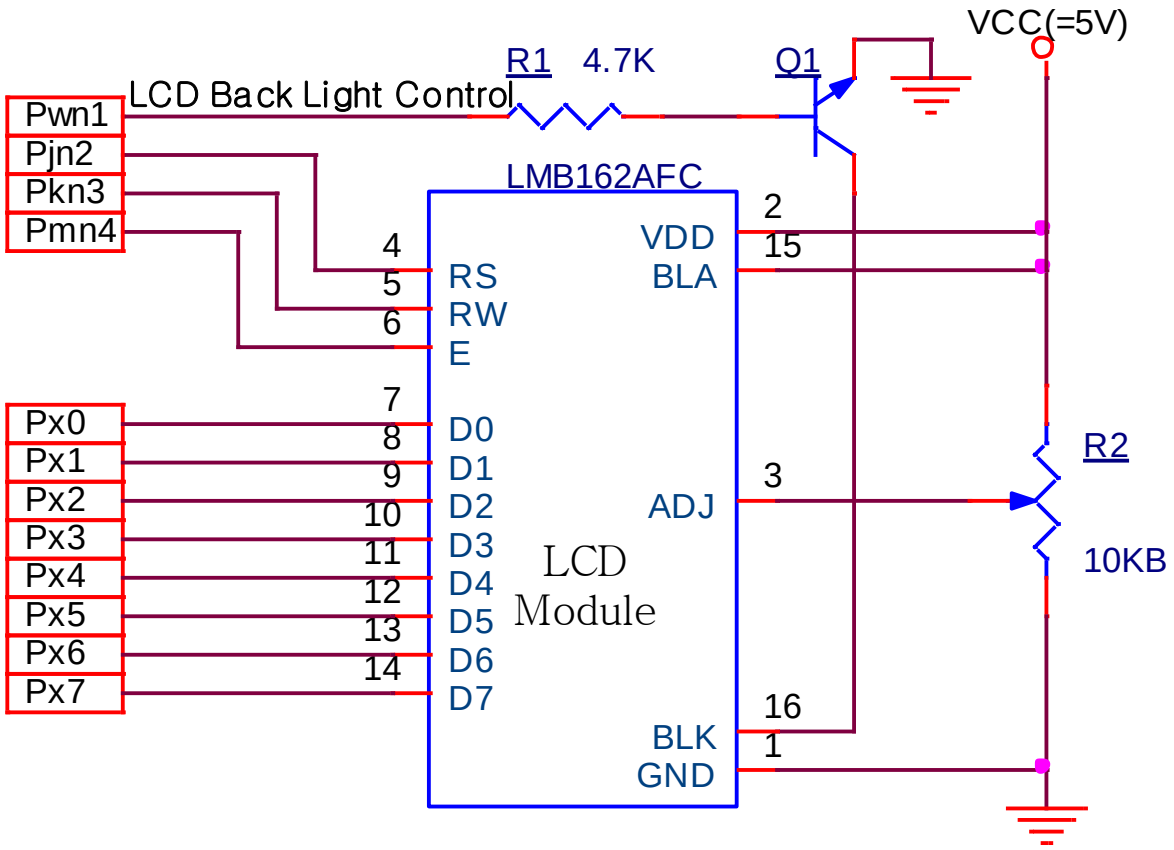


Here Data is loaded into LCD module to complete write operation

<Write Timing Diagram>

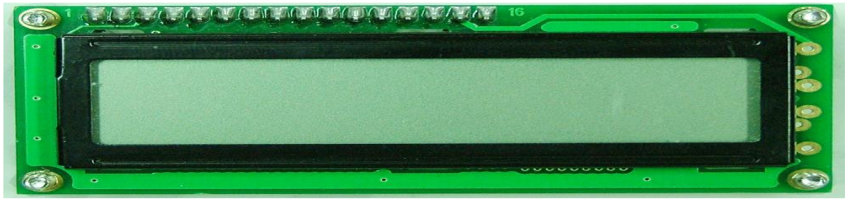
❑ Circuit for driving LCD module

- Interfacing LCD module to Microcontroller
  - 8 bit Interface or 4 bit Interface (we use 8 bit Interface)
  - Backlight can be controlled by PWM pulse so that the Brightness can be changed smoothly

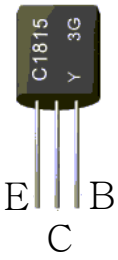


❑ Circuit for driving LCD module

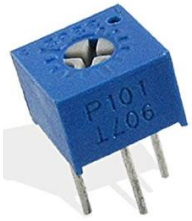
- Circuit construction on the Breadboard



LCD Display Module

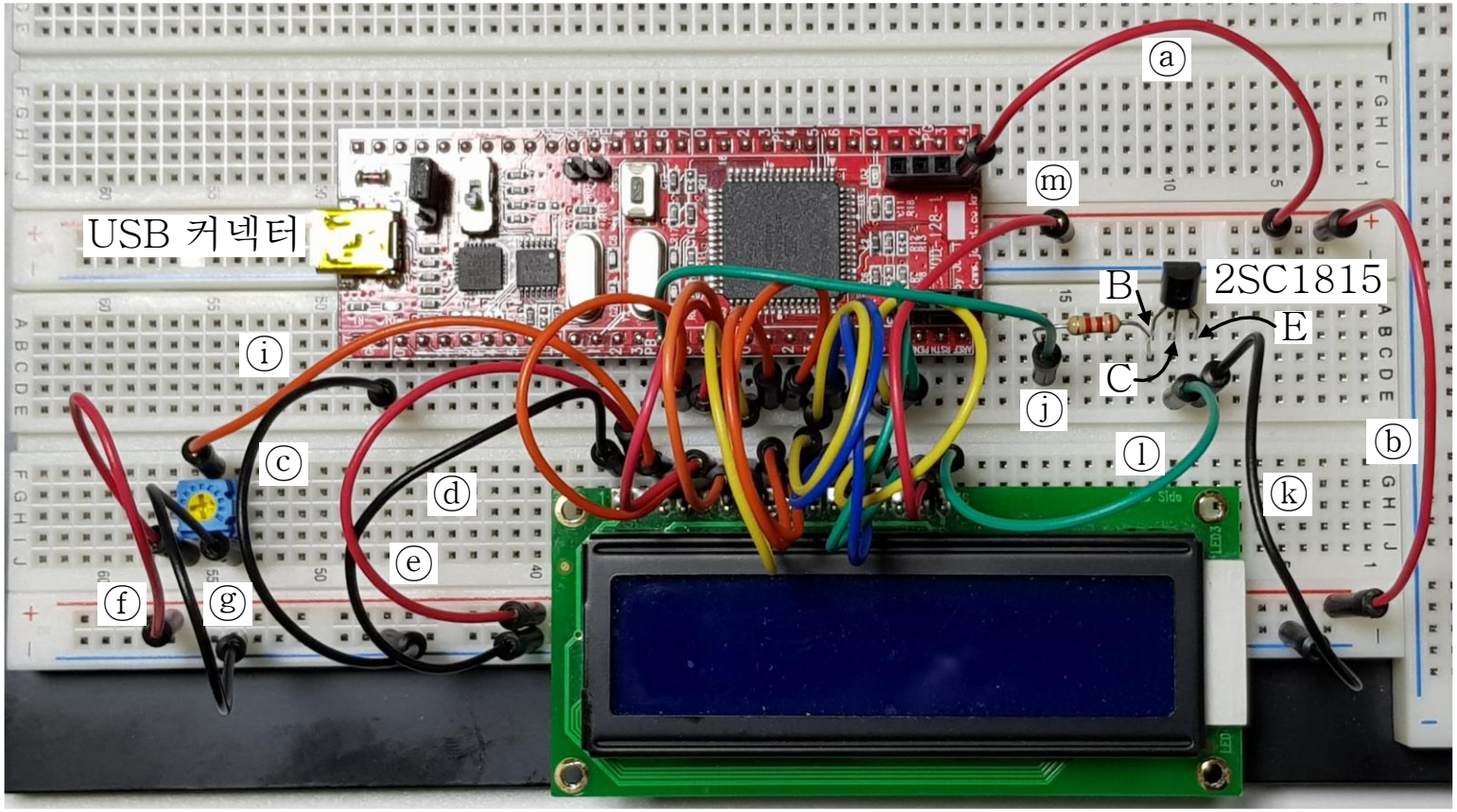


Transistor



Potentiometer

Resistor



❑ Developing sub-functions for driving LCD module

- **EnablePulse()**
  - Apply a single positive pulse to the 'E' pin of LCD module
  - A positive pulse should be applied to 'E' pin of the LCD module to complete the operation of every command to the LCD module

```
void EnablePulse(void) {
    LCD_CONTROL_PORT &= ~(1<<LCD_E);  // Apply '0' to E Pin
    LCD_CONTROL_PORT |= 1<<LCD_E;     // Apply '1' to E Pin
    _delay_us(40);                    // delay for 40uS
    LCD_CONTROL_PORT &= ~(1<<LCD_E); // Apply '0' to E Pin
}
```

- **FunctionSet()**
  - Set Interface BUS width, number of Display Lines, Size of Character Font

Function Set Command Code

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	DL	N	F	X	X

DL: '0' – 4 bit, '1', – 8 bit
N: '0' – 1 lines, '1' – 2 line
F: '0' – 5x8, '1' – 5x11 font

❑ Developing sub-functions for driving LCD module

```
void FunctionSet(void) {
    // Set to command write mode(set RW = RS = '0')
    LCD_CONTROL_PORT &= ~((1<<LCD_RW) | (1<<LCD_RS));
    LCD_DATA_PORT = 0x38; // use 8 bit BUS, 2 Lines, 5 x 8 font
    EnablePulse();
}
```

- ClearLCD()
  - Clears LCD screen and move the cursor to the left-uppermost position

```
void ClearLCD(void) {
    // Set to command write mode(set RW = RS = '0')
    LCD_CONTROL_PORT &= ~((1<<LCD_RW) | (1<<LCD_RS));
    LCD_DATA_PORT = 0x01;          // Clear LCD command(0x01)
    EnablePulse();
    _delay_ms(1.5)
}
```

❑ Developing sub-functions for driving LCD module

- Cursor()
  - Controls Cursor operations such as “Invisible”, “Normal”, “Blinking”

Display Control Command Code

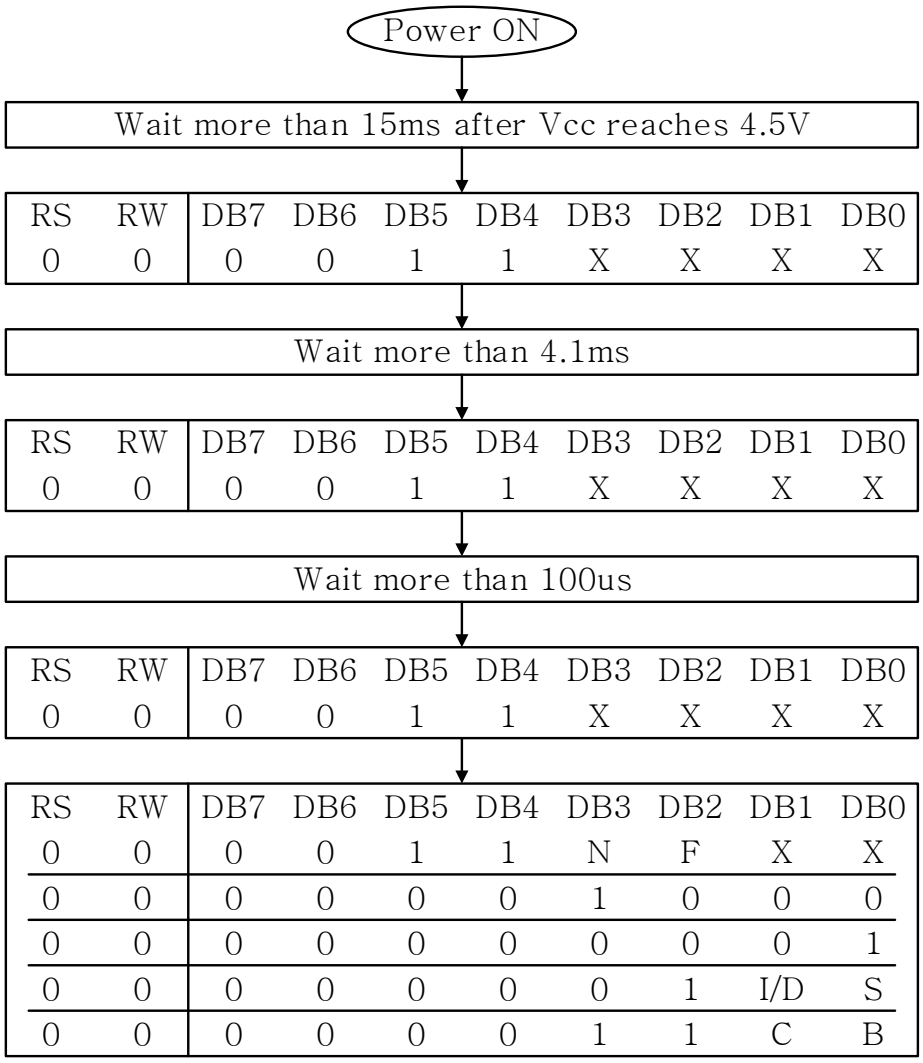
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	D	C	B

```
void Cursor(char CursorMode, char BlinkingMode) {
    unsigned char Tmp;

    // Set to command write mode(set RW = RS = '0')
    LCD_CONTROL_PORT &= ~((1<<LCD_RW) | (1<<LCD_RS));
    // Combine Cursor Mode Argument and Blinking Mode Argument
    Tmp = (CursorMode<<1) | BlinkingMode;
    LCD_DATA_PORT = Tmp | 0x0C;
    EnablePulse();
    _delay_ms(1.5)
}
```

❑ Developing sub-functions for driving LCD module

- InitLCD()
  - Initializes LCD module using functions made in the previous pages



During this time, there should be no data transfer to LCD module

Function Set command(8bit interface)
BF Flag cannot be checked yet

During this time, there should be no data transfer to LCD module

Function Set command(8bit interface)
BF Flag cannot be checked yet

During this time, there should be no data transfer to LCD module

Function Set command(8bit interface)
BF Flag can be checked afterwards

Function Set : 8bit, Determine Display Lines, Size of Font

Display OFF

Clear Display

Entry Mode Set

Display ON : Cursor/Blinking
C, B can be set if needed

❑ Developing sub-functions for driving LCD module

```
void InitLCD(void) {

    // 초기 RS = RW = '0'으로 설정 LCD 모듈에 대한 명령어 쓰기 모드
    LCD_CONTROL_PORT &= ~((1<<LCD_RW) | (1<<LCD_RS));

    _delay_ms(15);        // 15 mS 이상 대기

    LCD_DATA_PORT = 0x30; // 기능설정(8 비트 인터페이스)
    EnablePulse();

    _delay_ms(5);             // 5.0 mS(>4.1 mS) 대기

    LCD_DATA_PORT = 0x30; // 기능설정(8 비트 인터페이스)
    EnablePulse();

    _delay_ms(0.1);       // 100 uS 대기
```

❑ Developing sub-functions for driving LCD module

```
    LCD_DATA_PORT = 0x30;      // 기능설정(8 비트 인터페이스)
    EnablePulse();

    _delay_ms(1.5);

    LCD_DATA_PORT = 0x38; // 8 비트, 2 라인 5 x 8 폰트
    EnablePulse();

    LCD_DATA_PORT = 0x08; // Display OFF
    EnablePulse();

    LCD_DATA_PORT = 0x06; // Entry mode 설정, 커서 오른쪽으로 이동
    EnablePulse();

    LCD_DATA_PORT = 0x0C; // Display ON, 커서 안 보임, 깜박임 없음
    EnablePulse();

}
```

❑ Developing sub-functions for driving LCD module

- GotoXY()
  - Moves cursor to (X,Y) position in the LCD Character Screen
    - X : horizontal coordinate, 0≤X≤15
    - Y : vertical coordinate, 0≤Y≤1
  - Mapping between LCD data memory address and (X,Y) positon

Horizontal Axis X 0≤X≤15

Vertical Axis Y 0≤Y≤1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

00 ~ 0F, 40 ~ 4F : Data memory address expressed in Hexadecimal

Display Data Memory Address for (X, Y) = 0x40*Y + X

Command Code for setting Address Counter

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

❑ Developing sub-functions for driving LCD module

```
void GotoXY(unsigned char X, unsigned char Y) {
    // Set to command write mode(set RW = RS = '0')
    LCD_CONTROL_PORT &= ~((1<<LCD_RW) | (1<<LCD_RS));
    // Calculate Memory Address and convert it to LCD command format
    // and output to the LCD module
    LCD_DATA_PORT = 0x40*Y + X + 0x80;
    EnablePulse();
}
```

- PutCharacter()
  - Display a character at current cursor position on the screen

```
void PutCharacter(char Character) {
    // Set to data write mode(set RW = '0', RS = '1')
    LCD_CONTROL_PORT |= 1<<LCD_RS;
    LCD_CONTROL_PORT &= ~(1<<LCD_RW);
    // Output ASCII Code to LCD module
    LCD_DATA_PORT = Character;
    EnablePulse();
}
```

❑ Developing sub-functions for driving LCD module

- PutString()
  - Display a character string at current cursor position on the screen
  - The String should be terminated by the NULL('₩0') character
    - In case when the String is stored in the RAM

```
void PutString(char *StrPtr) {
    while(*StrPtr) PutCharacter(*StrPtr++);
}
```

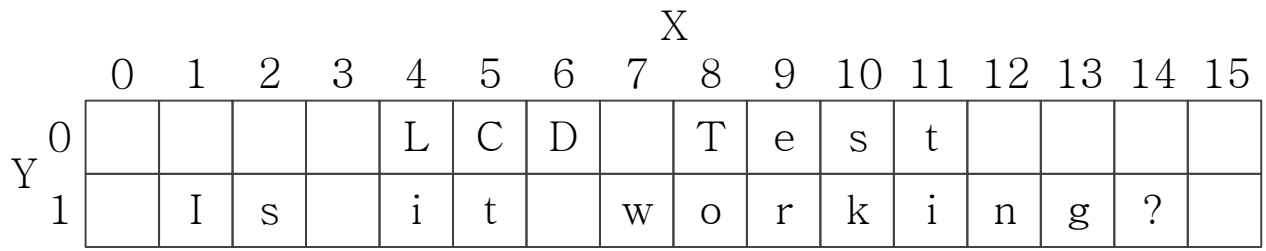
    - In case when the String is stored in the ROM

```
void PutString_ROM(const char *StrPtr) {
    while(*StrPtr) PutCharacter(*StrPtr++);
}
```

  - ➢ Store all the sub-functions in a file named "lcd.c" and store the file in the "inc" folder because they will be used by several practices in later chapters

❑ P6-1: Display Strings on the LCD screen(ex6-1)

- Goal of the Practice
  - Display following Message on the LCD screen



- Necessary Techniques and Programming Algorithms
  - Necessary Techniques
    - We already have all sub-functions needed!
  - Programming Algorithm
    - If you have very good sub-functions then main function could be made with only series of callings to sub-functions

- Program Explanation
  - How to get the Start Addresses of Strings

❑ P6-1: Display Strings on the LCD screen(ex6-1)

- Program Source List

```
#include          "../../../inc/ex.h"
#include          "../../../inc/lcd.c"

#define           BACK_LIGHT        4

// Global Variables
const char Message[2][17] = {"LCD TEST", "Is it working?" };

int main(void) {
    // initialize LCD
    InitLCD();
    LCD_CONTROL_PORT_DDR |= 1<<BACK_LIGHT;
    LCD_CONTROL_PORT |= 1<<BACK_LIGHT;
    GotoXY(4,0); // Move Cursor to (4, 0)
    PutString_ROM(Message[0]); //
    GotoXY(1,1); //
    PutString_ROM(Message[1]); // Display “Can you see?”
    while(1);
}
```

❑ P6-2: Digital Clock on the LCD screen(ex6-2)

- Goal of the Practice
  - Display following Message on the LCD screen

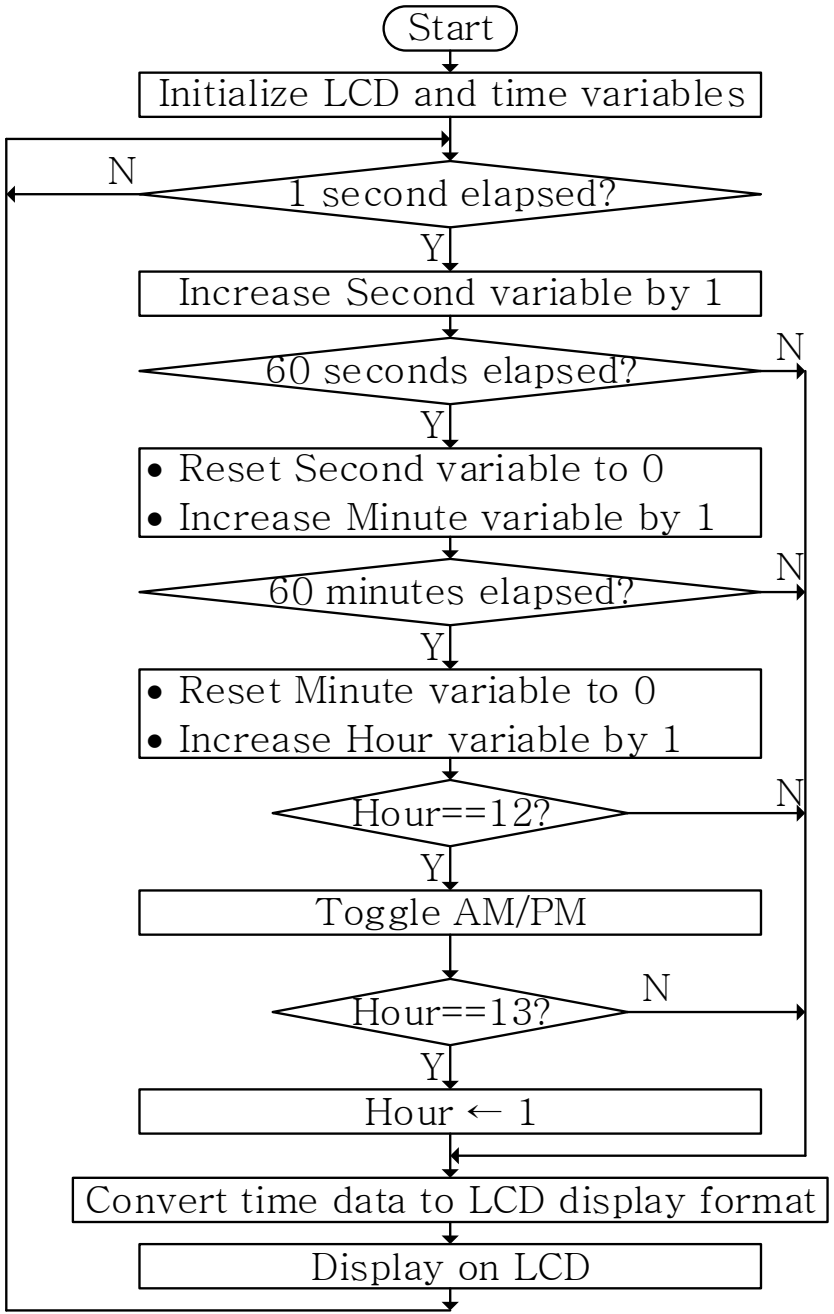
Y \ X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				C	L	O	C	K		V	1	.	0			
1			1	2	:	3	4	:	5	6			P	M		

- Necessary Techniques and Programming Algorithms
  - Necessary Techniques
    - Simply use delay function to get 1 second time base tick
    - Design functions to make a String which can be displayed by calling PutString() function from current time data : MakeData(), DecimalToASCII()
    - The Time String is stored in a buffer called LCD_DisplayBuffer[] as follows;

0	1	2	3	4	5	6	7	8	9	10	11	12
0x31	0x32	0x3A	0x33	0x34	0x3A	0x35	0x36	0x20	0x20	0x50	0x4D	0x00
Hour		:	Minute		:	Second		Two Spaces		PM/AM		NULL

❑ P6-2: Digital Clock on the LCD screen(ex6-2)

- Programming Algorithm



❑ P6-2: Digital Clock on the LCD screen(ex6-2)

- Program Source List

```
#include        "../../../inc/ex.h"
#include        "../../../inc/lcd.c“

#define         ONE_SECOND      1000 // ms
#define         BACK_LIGHT      4
#define         AM              0
#define         PM              1
#define         SECOND_INIT     55
#define         MINUTE_INIT     59
#define         HOUR_INIT       11
#define         AMPM_INIT       AM

unsigned char LCD_DisplayBuffer[13] = {"00:00:00 AM"};
unsigned char Title[11] = {"Clock V1.0"};

//
void DecimalToASCII(unsigned char Decimal, unsigned char Index) {
    //
    LCD_DisplayBuffer[Index++] = Decimal/10 + 0x30;
```

❑ P6-2: Digital Clock on the LCD screen(ex6-2)

```
    //
    LCD_DisplayBuffer[Index] = Decimal%10 + 0x30;
}

//
void MakeData(char Hour, char Min, char Sec, char AmPm) {
    //
    DecimalToASCII(Hour, 0);
    //
    DecimalToASCII(Min, 3);
    // LCD_DisplayBuffer[6], [7]
    DecimalToASCII(Sec, 6);
    // Update AM, PM
    if(AmPm==AM) LCD_DisplayBuffer[10] = 'A';
    else LCD_DisplayBuffer[10] = 'P';
}

int main(void) {
    char Sec, Min, Hour, AMPM;
```

❑ P6-2: Digital Clock on the LCD screen(ex6-2)

```
// LCD Backlight ON
LCD_CONTROL_PORT_DDR |= 1<<BACK_LIGHT;
LCD_CONTROL_PORT |= 1<<BACK_LIGHT;
// Initialize LCD
InitLCD();

// Initialize Time Variables
Sec = SECOND_INIT; //
Min = MINUTE_INIT; //
Hour = HOUR_INIT; //
AMPM = AMPM_INIT; //

// Construct Display String from Initial Time data
MakeData(Hour, Min, Sec, AMPM); // Convert to ASCII
// Display the Title of Digital Clock on the 1st Line of LCD
GotoXY(3,0);     //
PutString(Title);
GotoXY(2,1);     //
PutString(LCD_DisplayBuffer);        // Display Initial Time
```

❑ P6-2: Digital Clock on the LCD screen(ex6-2)

```
    while(1) {
        Delay_ms(ONE_SECOND);          // Wait for 1 second
        Sec++; // Increment second
        if(Sec==60) {
            Sec = 0; // If 60 seconds elapsed, clear second to 0
            Min++; // Increment minute
            if(Min==60) {
                Min = 0; // If 60 minutes elapsed, clear minute to 0
                Hour++; // Increment hour
                if(Hour==12) {
                    if(AMPM==AM) AMPM = PM; // if 12 O'clock
                    else AMPM = AM; // AM, PM Toggle
                }
                if(Hour==13) Hour = 1; // if 13 O'clock modify to 1 O'clock
            }
        }
        // Display Current Time on the LCD
        MakeData(Hour,Min, Sec, AMPM);
        GotoXY(2,1); //
        PutString(LCD_DisplayBuffer);
    }
}
```

❑ Project 6-1: Stop Watch

- Goal of the Project
  - 3장에서 사용했던 푸시 버튼을 이용하여 스톱워치(Stop watch)를 구현하고자 한다. 시간은 분:초:1/100초 의 식으로 표시되며 프로그램이 시작되면 LCD 화면상에 00:00:00 이 디스플레이 되고 SW2 스위치가 눌려지면 1/100 초 간격으로 디스플레이가 변하도록 한다. 시간이 변하는 중 다시 SW2 스위치가 눌려지면 디스플레이가 정지하여 스위치가 눌려진 순간의 경과 시간이 표시되도록 한다.
  - 다시 SW2 스위치가 눌려지면 시간 표시가 00:00:00 하는 식으로 리셋 되며 다시 SW2 스위치가 눌려 지면 시간 측정이 개시된다.
  - 즉, SW2 스위치 하나로 Stop watch의 Start/Stop 기능은 물론 시계 리셋 기능까지 모두를 순서적으로 제어하도록 시스템을 개발하여 보라.