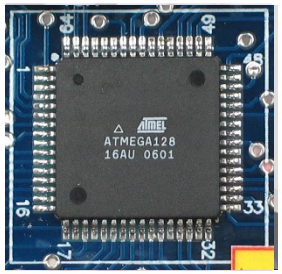
한국공학대학교
TECH UNIVERSITY OF KOREA



27 May 2022

LCD 표시 장치의 제어

목차

- LCD 모듈의 구조
- LCD 모듈의 단자 기능
- LCD 컨트롤러의 기능
- LCD 컨트롤러의 명령어
- LCD 인터페이스와 구동 프로그램
- LCD 제어 실험

학습목표

- 그래픽 및 문자 표시용 LCD 장치의 사용 방법

  ->I/O 포트를 이용한 인터페이스

  ->버스를 이용한 인터페이스 방법
- 문자 표시용 LCD 장치를 사용한 문자 표시 방법
- 문자 표시용 LCD에서의 새로운 문자의 생성
- PC문서 편집시에 나타나는 현상을 바탕으로 문자를 표시 하는 다양한 방법

  -> 커서이 이동, 블링킹 등…

LCD 모듈의 구조

한국공학대학교 TECH UNIVERSITY OF KOREA

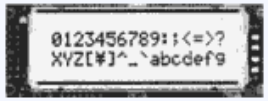
LCD 표시 장치

- 문자 표시용 : 영문이나 숫자(특수 문자 포함)
- 그래픽 표시용 (한글 및 배경 그림의 구현)

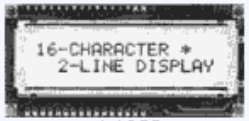
문자 표시용 LCD 모듈

L1671

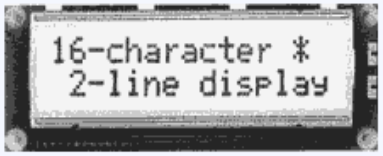
L1681



L1672

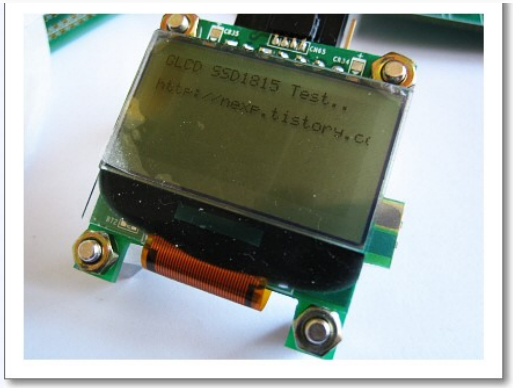


L1682



L1692

L1634



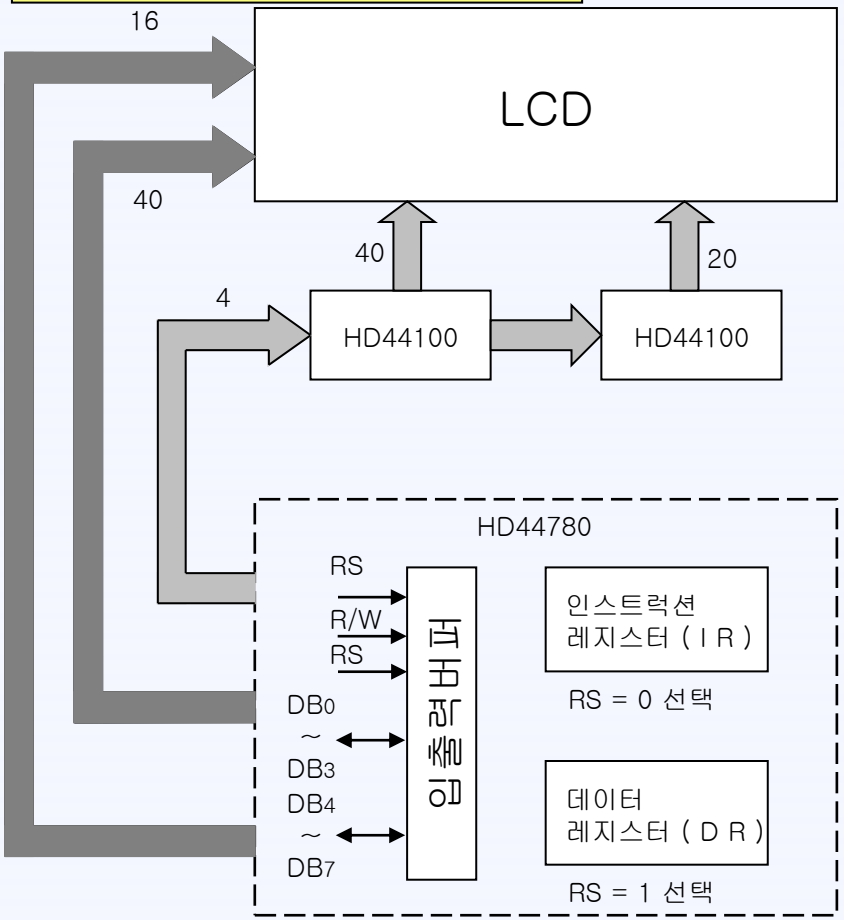
- 20문자/4라인, 16문자/2라인, 14문자/4라인 등이 출시
- 제어기 : HD44780

LCD 모듈의 구조

LCD 모듈(HD44780 내장) 모듈의 특징

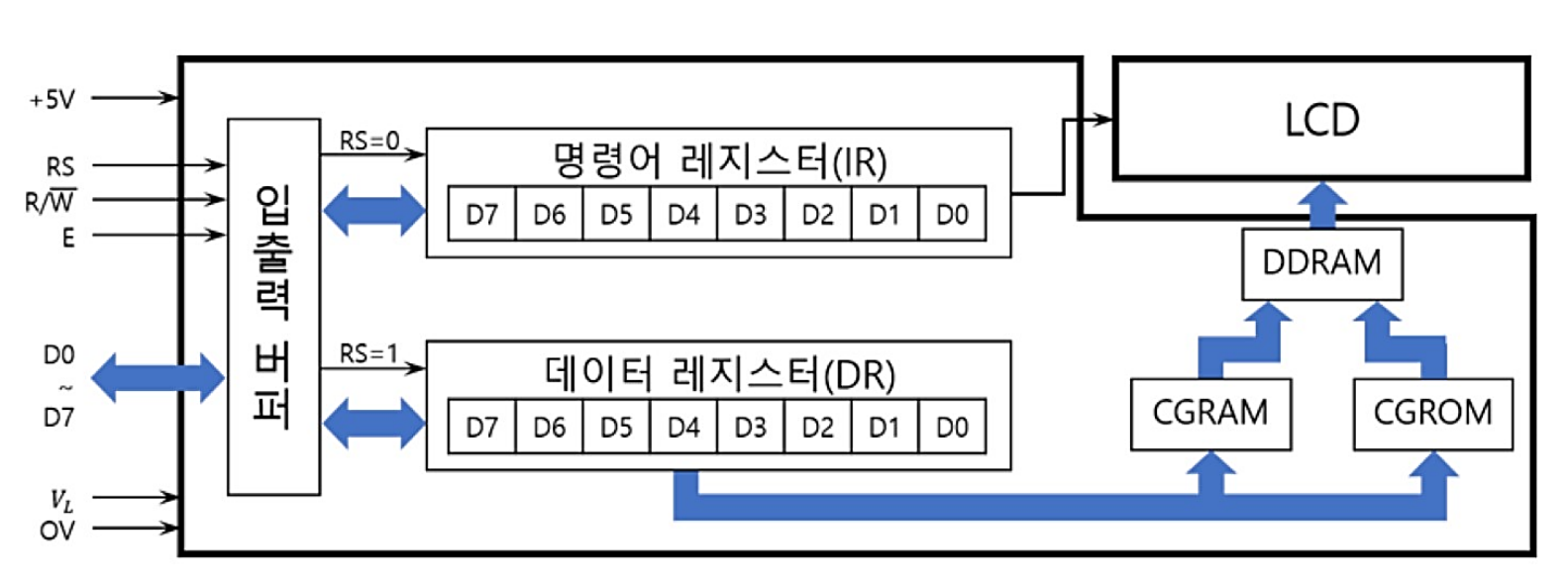
- 4비트, 8비트의 마이크로 컨트롤러와 인터페이스 가능
- 5 x 8 도트, 5 x 10 도트의 디스플레이 가능
- 80 x 8비트의 DDRAM(Display Data RAM : 최대 80글자까지) 내장
- 240 문자폰트를 내장하고 있는 CGROM(Character Generator ROM) 내장
- 64 x 8비트의 문자를 만들 수 있는 CGRAM(Character Generator RAM) 내장
- +5V단일 전원 사용

LCD 모듈의 내부 구조



LCD 모듈의 구조

LCD 제어기(HD44780)의 내부 구조



- 명령레지스터(IR) : LCD모듈 환경을 어떻게 사용할 것인가를 설정
- 데이터레지스터(DR) : LCD모듈에 글자를 나타내기 위한 데이터 값을 기록
- DDRAM, CGROM, CGRAM 메모리 내장

LCD 모듈의 단자 기능

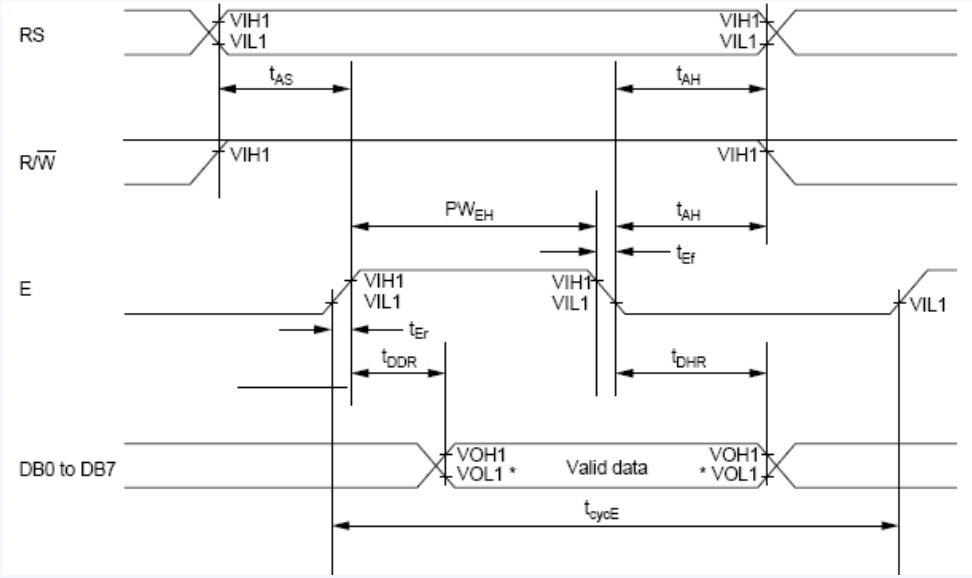
한국공학대학교 TECH UNIVERSITY OF KOREA

LCD 모듈의 단자

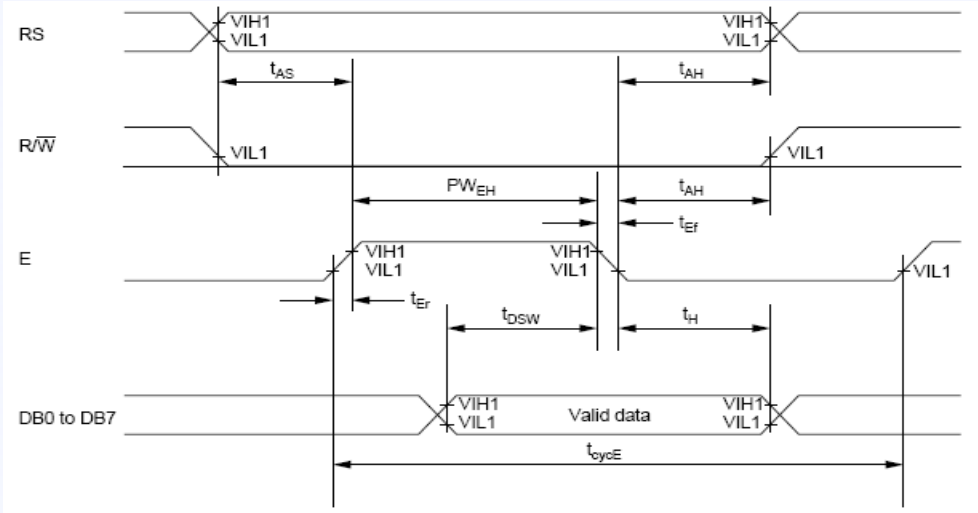
핀 번호	기 호	레 벨	기 능	
1	V_{SS}	-	0V	
2	V_{DD}	-	+5V	
3	Vo		LCD 밝기 조정(가변저항)	
4	RS	H/L	L : 명령 입력 (IR 선택) H : 데이터 입력 (DR 선택)	
5	$R/\overline{W}$	H/L	L : 쓰기(CPU → LCD module) H : 읽기(CPU ← LCD module)	
6	E	H	LCD 모듈의 허가 신호	
7	DB0	H/L	4비트 데이터 버스 이용시 사용 불가	8비트 데이터 버스 이용시 모두 사용
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L	4비트 데이터 버스 이용시 사용 가능	
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

LCD 모듈의 단자 기능

읽기 동작 타이밍도



쓰기 동작 타이밍도



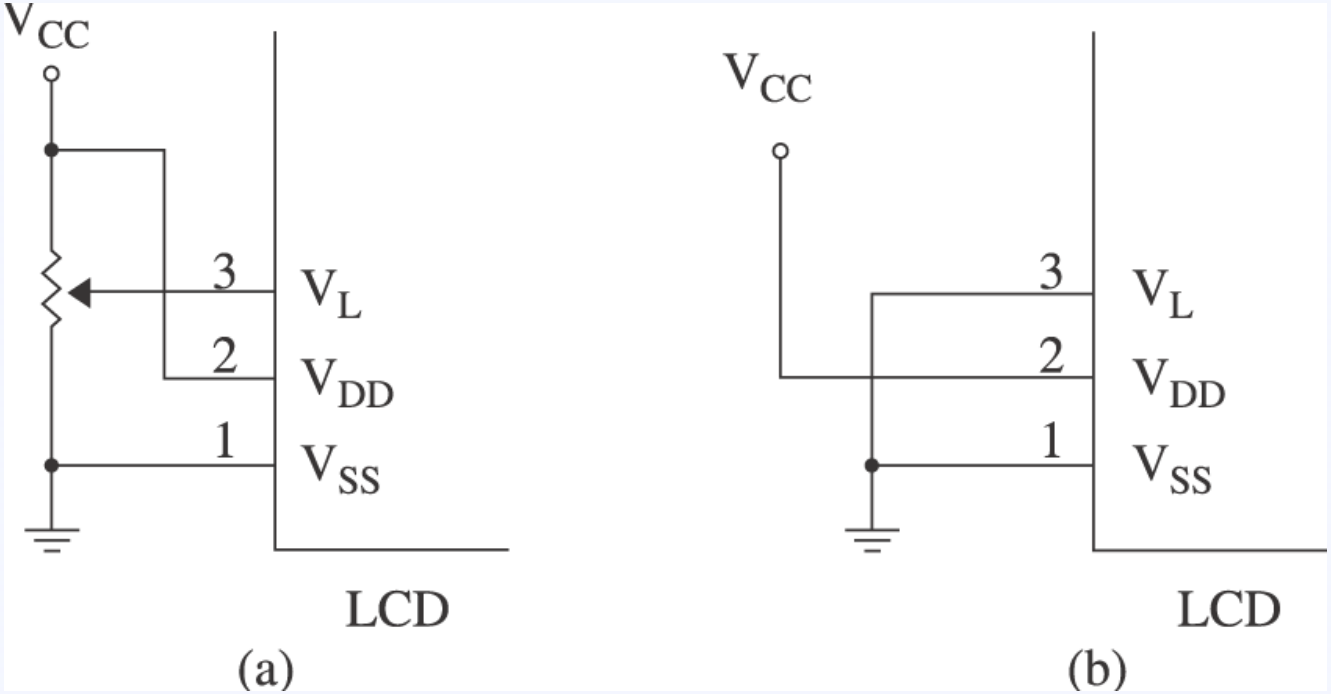
LCD 모듈의 단자 기능

타이밍 특성

Item	Symbol	Limit		Unit	비고
		min	max		
Enable Cycle Time	t_{cycE}	1000	-	ns	write/read 동작
Enable Pulse Width	PW_{EH}	450	-	ns	write/read 동작
Enable Rise/Fall Time	t_{Er},t_{Ef}	-	25	ns	write/read 동작
Address Setup Time	t_{AS}	140	-	ns	write/read 동작
Address Hold Time	t_{AH}	10	-	ns	write/read 동작
Data Setup Time	t_{DSW}	195	-	ns	write 동작
Data Hold Time	t_{H}	10	-	ns	write 동작
Data Delay Time	t_{DDR}	-	320	ns	read 동작
Data Hold Time	t_{DHR}	20	-	ns	read 동작

LCD 모듈의 단자 기능

LCD 모듈의 전원 처리 방법



- 10kΩ의 가변저항 값을 조절하여 밝기를 조절
- 밝기를 조절하고 싶지 않다면 Vo를 GND에 연결

LCD 컨트롤러의 기능

- 두 개의 레지스터(IR, DR)
- 세 개의 메모리 (DDRAM, CGRAM, CGROM)

레지스터

- IR(명령 레지스터)
  - LCD 화면 클리어, 커서시프트, 문자표시 ON/OFF 등 LCD의 제어에 필요한 명령을 저장.
  - 표시데이터 RAM(DDRAM)의 위치 주소와 문자 발생기 RAM(CGRAM)의 위치를 지정하기 위한 주소 정보를 저장.
- DR(데이터 레지스터)
  - DR 레지스터에 데이터를 쓰면 LCD의 내부적인 동작에 의해서 IR에 의해 지정된 DDRAM 또는 CGRAM의 주소로 전달.
  - DR 레지스터의 데이터를 읽으면, IR에 의해 지정된 DDRAM 또는 CGRAM의 주소 데이터가 마이크로컨트롤러로 전달.

LCD 컨트롤러의 기능

한국공학대학교 TECH UNIVERSITY OF KOREA

레지스터의 선택

RS	$R/\overline{W}$	동 작
0	0	IR을 선택하여 제어 명령 쓰기(디스플레이 클리어 등)
0	1	DB7로부터 비지플래그를 읽기/주소 카운터의 내용을 DB0 ~ DB6으로부터 읽기
1	0	DR을 선택하여 데이터 값을 쓰기(DR 에서 DDRAM 또는 CG RAM로)
1	1	DR을 선택하여 데이터 값을 읽기(DDRAM 또는 CGRAM에서 DR로)

비지 플래그(Busy Flag : BF)

- **연속적으로 LCD 모듈에 제어명령이 입력될 때 LCD모듈이 이 명령을 처리 할 수 있는가를 나타내는 상태 표시 플래그**
- **RS=0, $R/\overline{W}=1$일 때 출력되며, 이 때 BF 플래그는 데이터버스의 DB7로 출력.**
  - **BF = 0 : LCD 모듈로 다음 명령을 쓸 수 있음.**
  - **BF = 1 : LCD 컨트롤러(HD44780U)는 현재 IR로 입력된 명령어를 처리하고 있는 상태로서, 다음 제어 명령을 쓸 수 없는 상태.**

LCD 컨트롤러의 기능

주소 카운터

- DDRAM과 CGRAM의 주소를 지정하는데 사용.
- 주소 카운터의 값은 IR레지스터에 기록
- 주소정보는 IR레지스터에서 내부의 주소 카운터로 자동으로 전송되어 해당 주소 카운터의 값으로 세트.
- DDRAM 또는 CGRAM의 선택은 LCD명령에 의해서 결정.
- DDRAM 또는 CGRAM에 데이터를 써 넣으면 주소 카운터는 모드에 따라서 자동으로 1씩 증가하거나 또는 1씩 감소함

표시데이터 RAM

- DDRAM의 주소와 LCD 표시장치와의 관계 (20문자 x 2라인을 사용하는 경우)

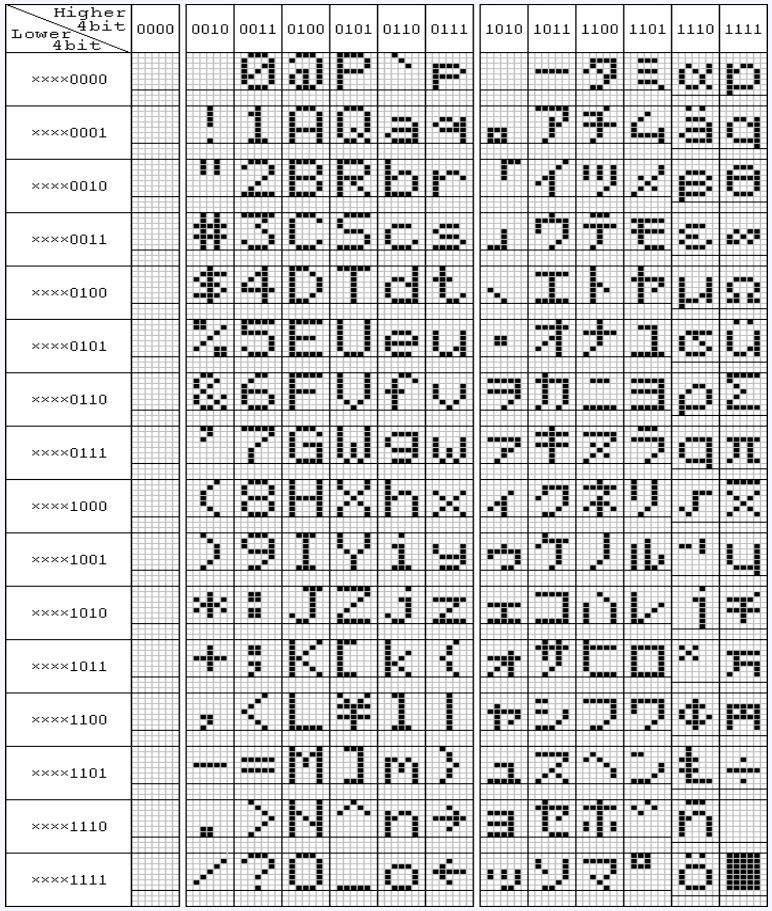
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	← 문자위치
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	←1열 DDRAM 주소
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	←2열 DDRAM 주소

LCD 컨트롤러의 기능

- DDRAM은 화면에 표시할 8비트의 문자를 저장하는 곳으로 용량은 80 x 8 비트 또는 80문자의 용량을 가지고 있음.
- 화면에 표시 되지 않은 영역의 RAM은 일반적인 데이터 저장용의 메모리로 사용 가능.

문자 발생기 ROM(CGROM)

- 8비트 문자 패턴을 저장하고 있는 메모리
- 208개의 5X8도트 혹은 32개의 5X10도트의 문자 패턴을 저장
- 문자코드 0x31에서 0xFF까지는 ASCII 코드와 일치.



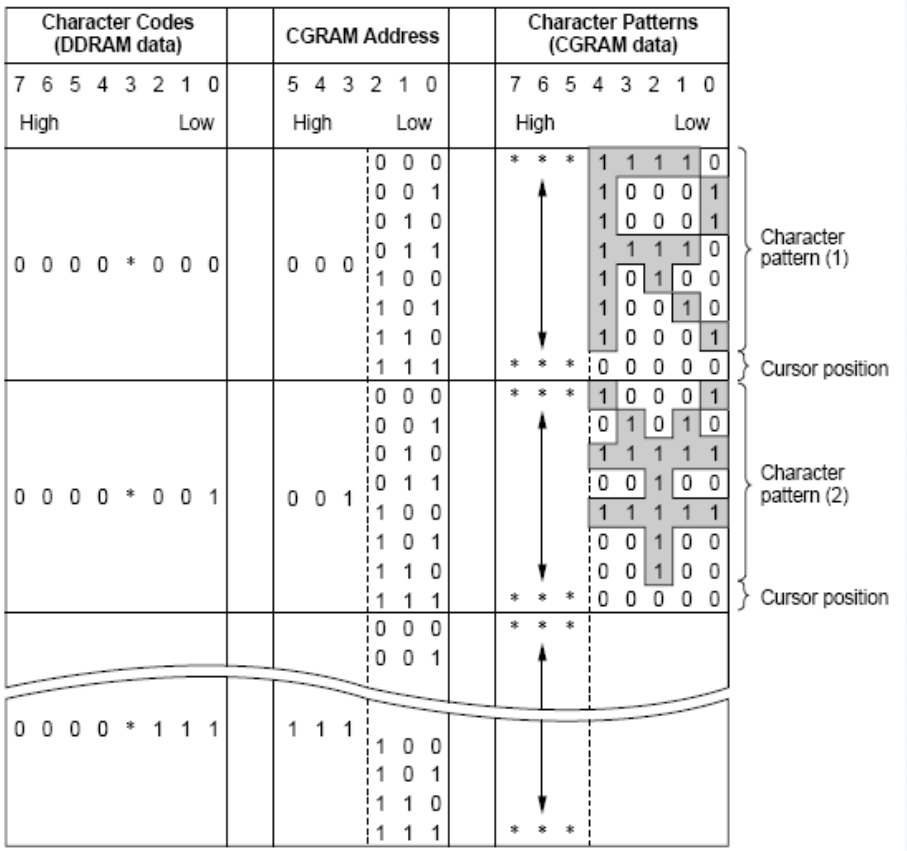
< CG ROM의 내용 >

LCD 컨트롤러의 기능

한국공학대학교 TECH UNIVERSITY OF KOREA

문자 발생기 CGRAM

- 사용자가 프로그램에 의해서 원하는 문자 패턴을 만들고자 할 때 사용하는 RAM영역.
- 5X8도트의 문자 패턴을 만들 경우, 최대 8종류의 패턴 생성 가능
- 5X10도트의 경우, 4종류의 문자 패턴 생성 가능
- 새로운 문자를 만드는 방법
  - 설계된 비트 패턴은 CGRAM 주소로 지정되고, 주소는 6비트를 사용
  - 5X8도트일 경우에 한 문자당 8바이트를 사용되어 문자는 8개로 제한
  - 문자코드의 비트 4~7이 0일대 선택 (3번 비트는 무효비트)
  - ‘R’ 문자의 설계
    - CG 패턴내의 주소 ‘0b000000’ ~'0b000111'에 저장
    - 저장된 문자 코드는 0x00 또는 0x08에 의해 선택
    - CGRAM으로의 데이터 쓰기/읽기 명령과 함께 사용됨.



<5X8 도트문자를 CGRAM 주소와 문자코드, 문자패턴의 관계>

LCD 컨트롤러의 명령

LCD 제어기 명령어 (LCD 모듈 내부 레지스터)

명령	코 드										기 능	실행 시간
	RS	R/$\overline{W}$	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
화면 지움	0	0	0	0	0	0	0	0	0	1	화면을 클리어하고, 커서가 홈 위치인 0번지로 돌아간다.	1.64ms
커서 홈	0	0	0	0	0	0	0	0	1	×	커서를 홈 위치로 돌아가게 한다. 또한 시프트되어 표시된 것도 되돌아가게 된다. DDRAM의 내용은 변하지 않는다.	1.64ms
엔트리 모드세트	0	0	0	0	0	0	0	1	I/D	S	데이터를 쓰거나 읽기를 수행할 때의 동작 모드를 결정한다. 즉, 커서의 진행 방향과 화면을 자동으로 시프트 시킬 것인지를 결정한다.	40㎲
화면 ON/OFF	0	0	0	0	0	0	1	D	C	B	화면 표시 ON/OFF(D), 커서 ON/ OFF(C), 커서 위치에 있는 문자의 블링크 기능(B)을 설정한다.	40㎲
커서/표시 시프트	0	0	0	0	0	1	S/C	R/L	×		화면 표시 내용은 변경시키지 않고, 커서와 화면의 이동과 시프트 동작을 설정한다.	40㎲
기능 설정	0	0	0	0	1	DL	N	F	×	×	LCD의 인터페이스 데이터 길이(DL), 표시 행수(N), 문자 폰트(F) 등을 설정한다.	40㎲
CGRAM 주소 설정	0	0	0	1	A_{CG}						CGRAM의 주소를 설정한다. 이후 전송되는 데이터는 CGRAM의 데이터이다.	40㎲
DDRAM 주소 설정	0	0	1	A_{DD}							DDRAM의 주소를 설정한다. 이후 전송되는 데이터는 DDRAM의 데이터이다.	40㎲
BF/주소 설정	0	1	BF	AC							LCD 모듈의 동작 여부와 현재 설정된 주소의 내용을 알기 위해서 BF 및 AC의 내용을 읽는다. CGRAM, DDRAM 양쪽 모두 사용할 수 있다.	0㎲
CG RAM, DD RAM으로 데이터 써넣기	1	0	써넣을 데이터								DDRAM 또는 CGRAM에 데이터를 써넣는다.	40㎲
CG RAM, DD RAM에서 데이터 읽기	1	1	읽을 데이터								DDRAM 또는 CGRAM에서 데이터를 읽는다.	40㎲

A_{CG} : CGRAM 주소
A_{DD} : DDRAM 주소
AC : 주소 카운터

I/D=1 : 증가(+1)
I/D=0 : 감소(−1)
S=1 : 표시 시프트 ON
S=0 : 표시 시프트 OFF

D=1 : 화면 ON
C=1 : 커서 ON
B=1 : 블링크 ON

D=0 : 화면 OFF
C=0 : 커서 OFF
B=0 : 블링크 OFF

R/L=1 : 우 시프트
S/C=1 : 화면 이동

R/L=0 : 좌 시프트
S/C=0 : 커서 이동

DL=1 : 8비트
N=1 : 2행
F=1 : 4*10도트

DL=0 : 4비트
N=0 : 1행
F=0 : 5*7도트

BF=1 : 내부 동작중 BF=0 : 명령 쓰기 가능

LCD 컨트롤러의 명령

화면 클리어

- LCD의 전체 화면 표시를 클리어한 후 커서는 홈 위치로 돌아가게 함 (DDRAM의 모든 내용을 “20H” 로 쓰고, 주소카운터를 00번지로 함)

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

커서 홈

- DDRAM의 내용은 변경하지 않고 주소 카운터를 00H로 하여 커서만 홈 위치로 보냄

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	*

LCD 컨트롤러의 명령

엔트리 모드 세트

➢ 마이크로컨트롤러가 LCD모듈에 데이터를 쓰기/읽기를 수행할 경우에 DDRAM을 왼쪽 또는 오른쪽으로의 증가와 화면시프트의 여부를 결정

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

- I/D : 문자 코드를 DDRAM에 쓰거나 또는 읽을 때, DDRAM의 주소를 +1 증가하거나 -1로 감소함, CGRAM에 데이터를 쓰거나 읽을 때에도 마찬가임
  - I/D=1일 때, 주소를 +1시키고, 커서 또는 블링크 위치가 우측으로 이동
  - I/D=0일 때, 주소를 -1시키고, 커서 또는 블링크 위치가 좌측으로 이동
- S : 커서의 위치를 자동으로 시프트 하는 기능, S=1이면, DDRAM의 내용을 화면에 표시한 후에 화면 전체를 I/D 비트에 설정된 값에 따라 좌/우로 이동시킨다. 커서 위치는 변하지 않고, 화면만 이동
  - S=1, I/D=1일 때, 좌로 시프트 함
  - S=1, I/D=0일 때, 우로 시프트 함
  - S=0, 화면은 시프트 되지 않음

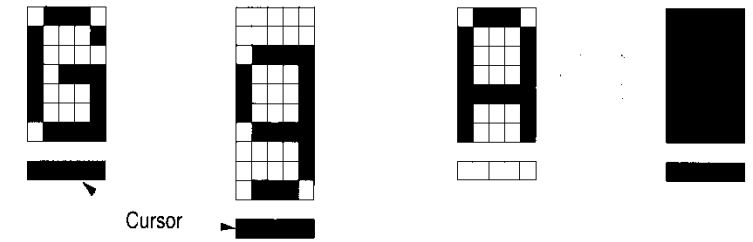
LCD 컨트롤러의 명령

화면 표시 ON/OFF

➢ **화면의 ON/OFF, 커서의 ON/OFF, 커서 위치에 있는 문자의 점멸 등의 기능 설정. (커서의 ON/OFF 및 점멸은 AC로 지정되어있는 DDRAM의 주소가 가리키는 행)**

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

- D : 화면 표시를 할 것인지를 결정
  - D=1이면, 화면 표시를 ON함
  - D=0 이면, 화면 표시를 OFF함
  - D=0으로 해서 표시를 OFF한 경우에는 화면에 표시되는 데이터는 DDRAM에 남아 있 기 때문에, D=1로 변경하면 다시 표시됨
- C : 커서를 화면에 표시할 것인지를 결정함
  - C=1이면, 커서를 표시함
  - C=0이면, 커서를 표시하지 않음
  - 커서는 5×7 도트 매트릭스의 문자 폰트의 경우에 8번째 라인에 표시되고, 5×10 도트 경우에 11번째 라인에 표시됨
- B : 커서의 위치에 있는 문자를 점멸할지를 결정함
  - B=1이면, 커서 위치에 상당하는 문자를 점멸함
  - B=0이면, 점멸하지 않음
  - 커서와 점멸 기능은 동시에 설정될 수도 있음



< 커서의 모양 >

커서 표시 시프트

- **DDRAM의 내용을 변경하지 않은 상태에서 커서를 움직이게 하고, 글자가 표시되는 부분을 시프트하기 위한 명령.**

RS	$R/\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	*	*

< 커서 시프트 명령의 기능 >

S/C	R/L	동 작
0	0	커서 위치를 좌로 이동한다.(AC를 -1 시킨다)
0	1	커서 위치를 우로 이동한다.(AC를 +1 시킨다)
1	0	표시 화면 전체를 좌로 이동한다. 커서는 화면의 움직임에 따라 같이 움직인다.
1	1	표시 화면 전체를 우로 이동한다. 커서는 화면의 움직임에 따라 같이 움직인다.

LCD 컨트롤러의 명령

한국공학대학교 TECH UNIVERSITY OF KOREA

기능 설정

➢ LCD를 사용하기에 앞서 선행되어야 하는 명령으로 LCD를 어떠한 구성으로 사용할 지를 결정하는 기능을 수행.

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	FL	*	*

- DL : 인터페이스 길이를 설정
  - -DL=1이면, 데이터 길이=8비트(DB7-DB0사용)
  - -DL=0이면, 데이터 길이= 4비트(DB7-DB4사용)
    데이터를 2번에 나누어 전송.
    └ 상위 4비트 먼저 전송→하위 4비트 전송
- N과 F는 화면 표시 행수와 문자 폰트를 설정하는 기능을 수행

< 기능 설정 명령의 세부 기능 >

NF	표시 행수	문자 폰트	듀티비	형 명
00	1	5×7 도트	1/8	-
01	1	5×10 도트	1/11	M24111, L4041
1*	2	5×7 도트	1/16	M1641, M1632, L2012, L2432, M4032, L4042, M4024

CGRAM 주소 설정

- 먼저 CGRAM의 주소를 설정하고, 주소가 된 후에 송수신하는 데이터는 CGRAM의 데이터임.
- DB5~DB0의 6비트까지 주소설정.
- CGRAM에 새로운 문자를 생성할 경우 이 명령을 이용하여 번지를 지정하고 문자폰트를 CGRAM에 입력.

RS	R/$\overline{W}$
0	0

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	A	A	A	A	A	A

DDRAM 주소 설정

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

- 먼저 CGRAM의 주소를 설정하고, 주소 설정 후에 송수신하는 데이터는 DDRAM의 데이터임.
- DB6~DB0의 7비트까지 주소설정.
- N의 값에 따라 LCD 라인의 주소가 결정됨.
  - N=0이면 한 개의 라인만 갖는 LCD.
    - AC범위 : 0x00 ~ 0x4F
  - N=1이면 두 개의 라인을 갖는 LCD.
    - 첫 번째 라인의 AC범위 : 0x00 ~ 0x27
    - 두 번째 라인의 AC범위 : 0x40 ~ 0x67

LCD 컨트롤러의 명령

비지 플래그 / 어드레스 읽기

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	BF	A	A	A	A	A	A	A

- 이전에 받은 명령에 의해 모듈이 내부 동작 중인지를 알기 위해서 BF 신호를 읽는 명령.
  - BF=1: BUSY, BF=0: 명령 수행 완료
- DB7은 BF의 상태를 나타내고 DB6~DB0까지는 AC의 값을 나타낸다.
- 이 명령이전에 설정된 RAM의 주소에 따라 CGRAM 또는 DDRAM의 AC를 읽을 수 있음.

LCD 컨트롤러의 명령

CGRAM과 DDRAM으로 데이터 쓰기

- DB7~DB0의 문자를 AC가 지정하고 있는 주소에 기록.
- IR 레지스터에 지정된 명령에 따라 CGRAM, DDRAM이 선택됨.
- 주소는 엔트리 모드의 설정에 따라 자동으로 1씩 증가 또는 감소함

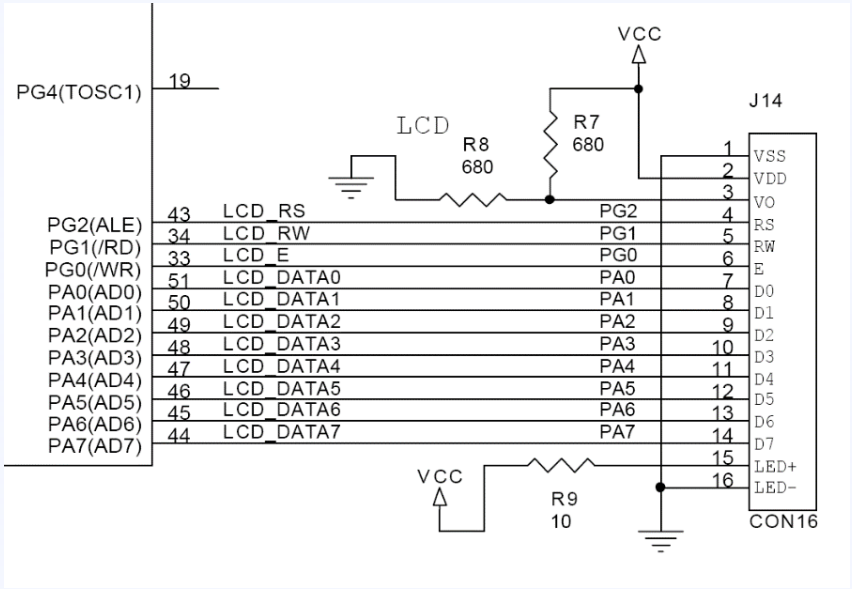
RS	$R/\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D	D	D	D	D	D	D	D

CGRAM과 DDRAM으로부터 데이터 읽기

- AC가 지정하고 있는 주소의 내용을 DB7~DB0에 기록.
- IR 레지스터에 지정된 명령에 따라 CGRAM, DDRAM이 선택됨
- 데이터를 읽기 전에 CGRAM 또는 DDRAM이 선택되지 않았을 때는 읽기동작이 무효임.

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

LCD 인터페이스 회로



- 신호선(D0~D7) :
  ATmega128의 Port A0 ~ A7의 I/O에 LCD의 D0~D7 핀을 Pin to Pin 연결.
- LCD모듈의 RS(4번)단자 :
  명령 레지스터인지 데이터 레지스터인지 구분하는 단자로서 ATmega128 Port G2핀과 연결.
- R/$\overline{W}$단자 :
  Port G1핀에 연결되어 0인 경우는 쓰기 동작이 되고 1인 경우는 읽기동작.
- LCD의 6번 단자(E) :
  LCD모듈의 사용 허가하는 단자로서 Port G0핀에 연결.

LCD 구동을 위한 기본 구동 함수

- LCD 모듈을 제어하는데 가장 기본적으로 수행되어야 하는 기능
  - 명령레지스터(IR)에 명령 쓰기.
  - 데이터 레지스터(DR)에 데이터 쓰기.
  - 명령 처리기시간을 고려한 시간 지연

- LCD에 읽기/쓰기 동작은 IR 레지스터와 DR 레지스터를 통해 이루어짐

신호 제어선의 정의

```
#define LCD_WDATA  PORTA        // LCD 데이터 버스 정의 (데이터 쓰기)
#define LCD_WINST  PORTA        // LCD 데이터 버스 정의 (명령어 쓰기)
#define LCD_RDATA  PINA         // LCD 데이터 버스 정의 (데이터 읽기)
#define LCD_CTRL   PORTG        // LCD 제어 신호 정의
#define LCD_EN     0            // Enable 신호
#define LCD_RW     1            // 읽기(1)/쓰기(0)
#define LCD_RS     2            // 데이터(1)/명령어(0)
```

명령 레지스터에 명령어 쓰기 함수

➢ 그림 10.4의 쓰기 타이밍 참조

신호 / 순서	RS(PORTG.2)	R/$\overline{W}$ (PORTG.1)	E(PORTG.0)	데이터 버스 (PORTA)
①	0	0	0	Select IR
②	0	0	1	명령어 쓰기
③	0	0	0	

```
void LCD_Comm(Byte ch)
{
    LCD_CTRL &= ~(1 << LCD_RS);     // RS==0으로 명령어 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);      // LCD Enable
    _delay_us(50);                   // 시간지연
    LCD_WINST = ch;                 // 명령어 쓰기
    _delay_us(50);                   // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);     // LCD Disable
}
```

데이터 레지스터에 데이터 쓰기 함수 ➢ 그림 10.4의 쓰기 타이밍 참조

신호 / 순서	RS (PORTG.2)	$\overline{R/W}$ (PORTG.1)	E (PORTG.0)	데이터 버스 (PORTA)
①	1	0	0	Select DR
②	1	0	1	명령어 쓰기
③	1	0	0	

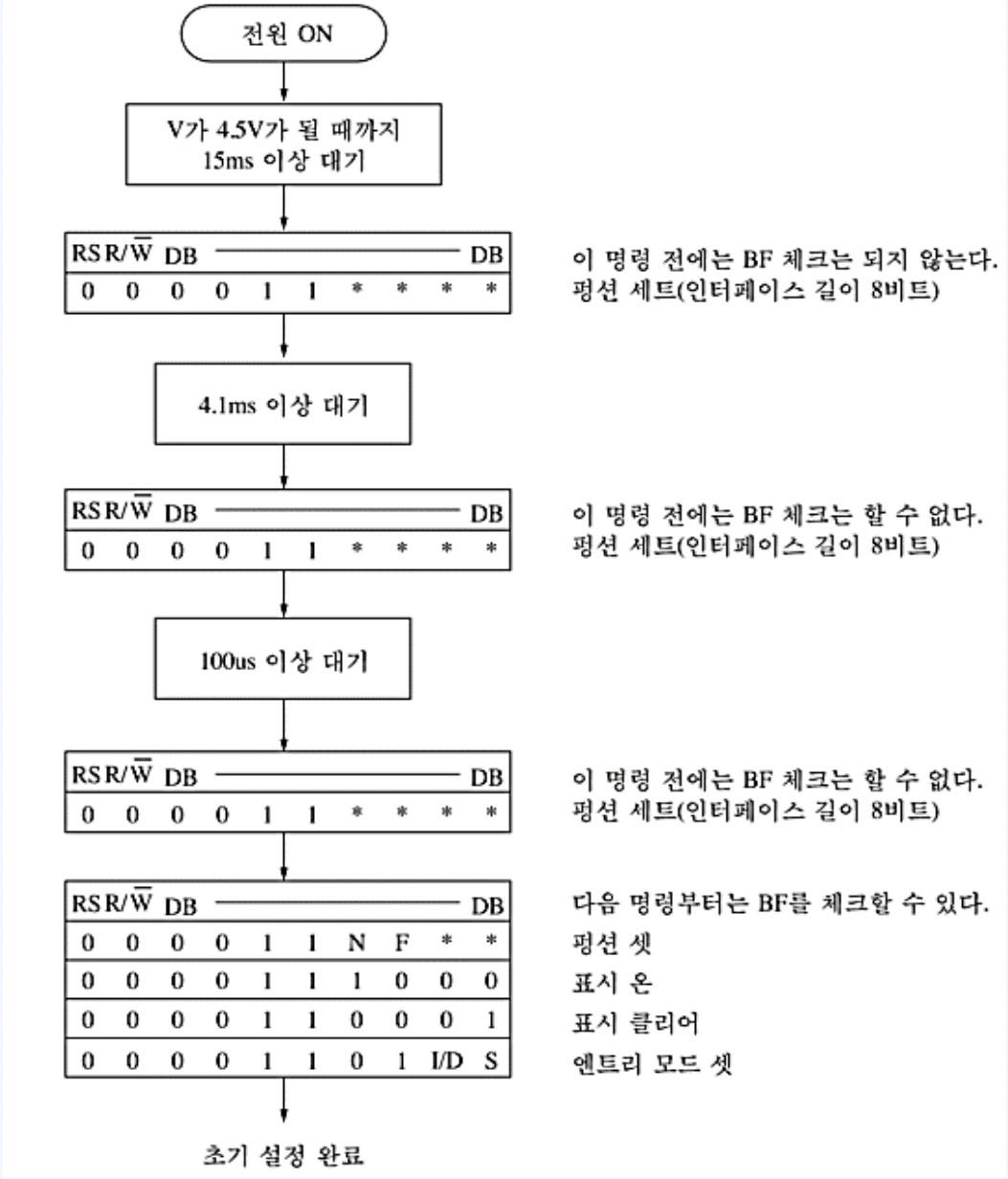
```
void LCD_Data(Byte ch)
{
    LCD_CTRL |= (1 << LCD_RS);      // RS=1, =0으로 데이터 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);      // LCD Enable
    _delay_us(50);                   // 시간지연
    LCD_WDATA = ch;                 // 데이터 출력
    _delay_us(50);                   // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);     // LCD Disable
}
```

LCD 시간 지연 함수

- 명령을 IR레지스터에 쓰고 나면 40㎲ 또는 1.64㎳를 기다린 후에 다음 명령을 쓸 수 있음
- CGRAM 또는 DDRAM에 데이터를 쓰고 나면 40㎲를 기다린 후에 다음명령을 쓸 수 있음
- Delay. h에 있는 시간 지연 함수 사용
  - delay_us(unsigned int value), delay_ms(unsigned int value) 함수
- 명령 데이터를 쓰기 위한 지연 시간은 40㎲ 또는 1.64㎳ 이지만, 본 강의내용 내의 예제 프로그램에서는 편의상 2㎳의 시간지연으로 작성함.

```
void LCD_delay(Byte us)
{
    _delay_us(us);
}
```

LCD 초기화 과정 및 기본 구동 함수



기능 설정

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	
기능 설정	0	0	0	0	1	DL	N	F	x	x	LCD의 인터페이스 데이터 길이(DL), 표시 행수(N), 문자 폰트(F) 등을 설정한다.

- **LCD모듈의 초기화 설정에 관련된 기능으로 사용하려는 LCD의 인터페이스 방식과 도트의 구성을 설정하기 위하여 사용됨.**
  - **8비트 데이터 라인**
  - **2라인 디스플레이**
  - **5X7 도트 글자폰트 사용**

// 설정된 값(0x3*)을 IR 레지스터에 연속하여 3회 전송 후 LCD 표기를 위한 세팅, 명령을 쓰는 과정에서 명령어가 처리 되는 시간을 시간지연함수를 사용하여 기다림.

```
LCD_Comm(0x30);          // 초기화 Set,
LCD_delay(4100);           // 4.1ms 지연
LCD_Comm(0x30);          // 초기화 Set,
LCD_delay(100);            // 100us 지연
LCD_Comm(0x30);          // 초기화 Set,
LCD_delay(100);            // 100us 지연
LCD_Comm(0x38);          // 초기화 Set, 데이터 길이 8Bit, 표시라인 2행 사용을 위한 기능 설정
LCD_delay(1000);           // 명령을 처리 하는데 최소 40us 지연이 발생하기에 여유를 고려하여 설정
```

만약, 4Bit Control을 희망한다면 0x38 (X) -> 0x28(O) 수정

화면 ON/OFF 제어

- 문자 표시를 ON/OFF하는 기능(D).
- 커서를 ON/OFF하는 기능(C).
- 문자표시를 깜박거리게 하는 기능(B).

```
LCD_Comm(0x0e); // 모든 기능을 ON 한다.
LCD_delay(1700);// 1.64ms 이상을 기다림
```

화면 클리어

- 화면을 클리어 하고 커서는 홈 위치로 돌아감.

```
LCD_Comm(0x01); // LCD Clear
LCD_delay(1700);// 1.64ms 이상을 기다림
```

엔트리 모드 세트

- I/D 비트에 의해 커서 진행방향을 설정
- S비트에 의해 글자를 시프트 할 것 인지를 설정

```
LCD_Comm(0x06); // Entry Mode Set
LCD_delay(1700);// 1.64ms 이상을 기다림
```

LCD초기화 함수

➢ **Port A와 Port G의 4비트(실제 사용은 3bit)는 출력포트로 지정**

```
void Init_Lcd(void)
{
    DDRA = 0xFF;                    // PORTA를 출력으로 지정
    DDRG = 0x0F;                    // PORTC의 하위 4비트를 출력으로 지정, 또는 0x07(3bit)
    //LCD_delay(15);                // 필요시…

    LCD_Comm(0x30);                 // 초기화 Set,
    LCD_delay(4100);                  // 4.1ms 지연
    LCD_Comm(0x30);                 // 초기화 Set,
    LCD_delay(100);                   // 100us 지연
    LCD_Comm(0x30);                 // 초기화 Set,
    LCD_delay(100);                   // 100us 지연
    LCD_Comm(0x38);                 // 초기화 Set, 데이터 길이 8Bit, 표시라인 2행 사용을 위한 기능 설정
    LCD_delay(1000);                  // 명령을 처리 하는데 최소 40us 지연이 발생하기에 여유를 고려하여 설정
    LCD_Comm(0x0e);                 // Display ON, Cursor On, Blink Off
    LCD_delay(1000);                  // 40㎲ 이상을 기다림
    LCD_Comm(0x01);                 // LCD Clear
    LCD_delay(2000);                  // 1.64㎳ 이상을 기다림
    LCD_Comm(0x06);                 // Cursor Entry Mode Set, 표시 위치 +1씩 증가
    LCD_delay(1000);                  // 40㎲ 이상을 기다림
}
```

이외의 LCD 구동 함수

- 커서 홈(Cursor Home)
- 커서 디스플레이 시프트(Cursor Display Shift)
- 표시 위치 설정(Display Position Set)

커서 홈

➢ 커서의 위치를 화면의 처음으로 설정

```
void cursor_home(void)
{
    LCD_Comm(0x02);              // Cursor Home
    LCD_delay(2000);             // 2ms 시간지연
}
```

표시 위치 설정

➢ 좌표의 개념으로 원하는 위치에 커서와 표시 위치를 설정
➢ 표시 위치와 DDRAM의 어드레스의 관계에 의해 결정됨.

```
void LCD_pos(unsigned char row, unsigned char col)      // LCD 포지션 설정
{
        LCD_Comm(0x80 | (col + row * 0x40));        // row = 문자행(0,1),  col = 문자열(0 … 15)
}
```

디스플레이 시프트

- **LCD에 표시된 글자에 대해서 왼쪽으로 1 또는 오른쪽으로 1을 시프트 하기 위한 함수**
  - **display_shift(RIGHT), display_shift(LEFT)**
  - **cursor_shift(char p)**

```
#define RIGHT 1
#define LEFT 0
void Display_Shift(char p)      // 디스플레이 시프트 (5)
{
                                // 표시 화면 전체를 오른쪽으로 이동
    if(p == RIGHT) {
        LCD_Comm(0x1C);         // * A에서 C로 바꿈
        LCD_delay(1000);        // 시간 지연
    }
                                // 표시 화면 전체를 왼쪽으로 이동
    else if(p == LEFT) {
        LCD_Comm(0x18);
        LCD_delay(1000);
    }
}
```

```
void Cursor_shift(Byte p)
{
    if(p == RIGHT) {
        LCD_Comm(0x14);
        LCD_delay(1000);
    }
    else if(p == LEFT) {
        LCD_Comm(0x10);
        LCD_delay(1000);
    }
}
```

LCD구동 프로그램

- LCD의 정해진 위치에 문자를 표시하기 위한 과정
  - 명령레지스터에 DDRAM주소 설정 명령을 이용하여 위치를 설정
  - 표시하고자 하는 문자를 문자 발생기 ROM상에 있는 문자를 호출하여 데이터 레지스터에 기록.
- 문자(열)을 이용하여 LCD에 표시 하는 함수

- void LCD_CHAR(Byte c) //하나의 문자를 LCD에 표시하는 함수
- void LCD_STR(Byte *str) // 문자열을 LCD에 표시하는 함수

Void LCD_CHAR(Byte C)

- 데이터 레지스터에 데이터를 쓰기 위한 함수

```
void LCD_CHAR(Byte c)              // 한 문자 출력
{
    LCD_delay(1);                  // 명령어를 쓰기전에 일정 시간 지연(busy 플래그 확인 대용)
    Putc_LCD(c);                   // DDRAM으로 데이터 전달
}
```

- 이 함수를 이용하여 ASCII코드표의 내용으로 그대로 출력 가능
  예) LCD_CHAR('L');
  LCD_CHAR(0x4C);

Void LCD_STR(Byte *str)

- 문자열을 str 포인터를 통해 받고, 이 포인터가 가리키는 문자열을 문자열이 끝날 때까지 출력하는 함수
- C언어에서 문자열은 아래 표와 같이 배열로 처리되고, 마지막 요소는 NULL문자('\0')으로 끝나므로, 이를 이용하여 문자열의 끝을 확인함.

L	C	D		T	e	s	t	.	.	'\0'			

```
void LCD_STR(Byte *str)                    // 문자열 출력
{
    while(*str != 0) {
        LCD_CHAR(*str);
        str++;
    }
}
```

- 이 함수를 이용하여 문자열을 출력하는 예

  LCD_STR("LCD Test..");

10진수	16진수	ASCII	10진수	16진수	ASCII	10진수	16진수	ASCII	10진수	16진수	ASCII
0	0×00	NULL	32	0×20	SP	64	0×40	@	96	0×60	.
1	0×01	SOH	33	0×21	!	65	0×41	A	97	0×61	a
2	0×02	STX	34	0×22	"	66	0×42	B	98	0×62	b
3	0×03	ETX	35	0×23	#	67	0×43	C	99	0×63	c
4	0×04	EOT	36	0×24	$	68	0×44	D	100	0×64	d
5	0×05	ENQ	37	0×25	%	**69**	**0×45**	**E**	101	0×65	e
6	0×06	ACK	38	0×26	&	70	0×46	F	102	0×66	f
7	0×07	BEL	39	0×27	'	71	0×47	G	103	0×67	g
8	0×08	BS	40	0×28	(	72	0×48	H	104	0×68	h
9	0×09	HT	41	0×29	)	73	0×49	I	105	0×69	i
10	0×0A	LF	42	0×2A	*	74	0×4A	J	106	0×6A	j
11	0×0B	VT	43	0×2B	+	75	0×4B	K	107	0×6B	k
12	0×0C	FF	44	0×2C	'	76	0×4C	L	108	0×6C	l
13	0×0D	CR	45	0×2D	-	77	0×4D	M	109	0×6D	m
14	0×0E	SO	46	0×2E	.	78	0×4E	N	110	0×6E	n
15	0×0F	SI	47	0×2F	/	79	0×4F	O	111	0×6F	o
16	0×10	DLE	**48**	**0×30**	**0**	80	0×50	P	112	0×70	p
17	0×11	DC1	**49**	**0×31**	**1**	81	0×51	Q	113	0×71	q
18	0×12	SC2	**50**	**0×32**	**2**	82	0×52	R	114	0×72	r
19	0×13	SC3	**51**	**0×33**	**3**	**83**	**0×53**	**S**	115	0×73	s
20	0×14	SC4	**52**	**0×34**	**4**	84	0×54	T	116	0×74	t
21	0×15	NAK	**53**	**0×35**	**5**	85	0×55	U	117	0×75	u
22	0×16	SYN	**54**	**0×36**	**6**	86	0×56	V	118	0×76	v
23	0×17	ETB	**55**	**0×37**	**7**	87	0×57	W	119	0×77	w
24	0×18	CAN	**56**	**0×38**	**8**	88	0×58	X	120	0×78	x
25	0×19	EM	**57**	**0×39**	**9**	89	0×59	Y	121	0×79	y
26	0×1A	SUB	58	0×3A	:	90	0×5A	Z	122	0×7A	z
27	0×1B	ESC	59	0×3B	;	91	0×5B	[	123	0×7B	{
28	0×1C	FS	60	0×3C	<	92	0×5C	₩	124	0×7C	|
29	0×1D	GS	61	0×3D	=	93	0×5D	]	125	0×7D	}
30	0×1E	RS	62	0×3E	>	94	0×5E	^	126	0×7E	~
31	0×1F	US	63	0×3F	?	95	0×5F	_	127	0×7F	DEL

문자열을 LCD에 표시하는 프로그램의 작성

```
#define F_CPU 14.7456E6
#include <avr/io.h>
#include <util/delay.h>

#define LCD_WDATA   PORTA     // LCD 데이터 포트 정의
#define LCD_WINST   PORTA
#define LCD_CTRL    PORTG     // LCD 제어포트 정의
#define LCD_EN       0
#define LCD_RW       1
#define LCD_RS       2

typedef unsigned char Byte;

void PortInit(void)
{
    DDRA = 0xFF;   // PORTA를 출력으로
    DDRG = 0x0F;   // PORTC의 하위 4비트를 출력으로
}
```

```
void LCD_Data(Byte ch)                  // LCD_DR에 데이터 출력
{
    LCD_CTRL |= (1 << LCD_RS);          // RS=1, =0으로 데이터 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);          // LCD Enable
    _delay_ms(50);                        // 시간지연
    LCD_WDATA = ch; // 데이터 출력
    _delay_ms(50);                        // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);         // LCD Disable
}

void LCD_Comm(Byte ch)                  // LCD IR에 명령어 쓰기
{
    LCD_CTRL &= ~(1 << LCD_RS);         // RS==0으로 명령어 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);          // LCD Enable
    _delay_ms(50);                        // 시간지연
    LCD_WINST = ch;                     // 명령어 쓰기
    _delay_ms(50);                        // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);         // LCD Disable
}

void LCD_delay(Byte ms)
{
     _delay_ms(ms);
}
```

```
void LCD_CHAR(Byte c)       // 한 문자 출력
{
    LCD_Data(c);
    _delay_ms(1);
}

void LCD_STR(Byte *str)     // 문자열 출력
{
     while(*str != 0) {
     LCD_CHAR(*str);
     str++;
     }
}

void LCD_pos(unsigned char row, unsigned char col) // LCD 포지션 설정
{
    LCD_Comm(0x80|(col+row*0x40));          // row = 문자행, col = 문자열
}

void LCD_Clear(void)        // 화면 클리어 (1)
{
     LCD_Comm(0x01);
     _delay_ms(2);
}
```

```
void LCD_Init(void)             // LCD 초기화
{
     LCD_Comm(0x38);            // DDRAM, 데이터 8비트사용, LCD 2열로 사용 (6)
     LCD_delay(2);              // 2ms 지연
     LCD_Comm(0x38);            // DDRAM, 데이터 8비트사용, LCD 2열로 사용 (6)
     LCD_delay(2);              // 2ms 지연
     LCD_Comm(0x38);            // DDRAM, 데이터 8비트사용, LCD 2열로 사용 (6)
     LCD_delay(2);              // 2ms 지연
     LCD_Comm(0x0e);            // Display ON/OFF
     LCD_delay(2);              // 2ms 지연
     LCD_Comm(0x06);            // 주소+1 , 커서를 우측 이동 (3)
     LCD_delay(2);              // 2ms 지연
     LCD_Clear();               // LCD 화면 클리어
}

int main(void)
{
    Byte str[] = "LCD Test..";// Byte 자료형은 없음 “ unsigned char” 로 생각하면 됨.
    PortInit();                 // LCD 출력 포트 설정
    LCD_Init();                 // LCD 초기화
    LCD_pos(0,0);               // LCD 포지션 0행 0열 지정, 만약 두번째 행 이라면 LCD_pos(0,1);
    LCD_STR(str);               // 문자열 str을 LCD 출력
    while(1);
}
```

2번째 라인에 "Micom World" 를 표시하는 프로그램의 작성

```
int main(void)
{
    unsigned char str1[] = "LCD Test..";
    unsigned char str2[] = "Micom World";

    PortInit();                 // LCD 출력 포트 설정
    LCD_Init();                 // LCD 초기화

    LCD_pos(0,1);               // 문자열 위치 0행 1열 지정
    LCD_STR(str1);              // 문자열 str을 LCD 첫번째 라인에 출력
    LCD_pos(1,1);               // 문자열 위치 1행 1열 지정
    LCD_STR(str2);              // 문자열 str을 LCD 두번째 라인에 출력

    while(1);
}
```

L	C	D		T	e	s	t	.	.					
M	i	c	o	m		W	o	r	l	d				

LCD 제어 실험

한국공학대학교 TECH UNIVERSITY OF KOREA

예제 1: 문자의 이동

- **영문자 “ATMega128", "AVR LCD Test"라는 문자열을 표시한 후 오른쪽으로 다시 처음 자리에 글자를 출력하는 프로그램 작성**

LCD 제어 실험

- 참고 사항에 있는 내용을 참조하여 lcd.h와 lcd.c를 이용하여 프로그램 작성함.

```
#include  “lcd.h”   // lcd.h 파일이 있는 위치 지정

int main(void)
{
   unsigned char str1[]= “ATmega 128” ;
   unsigned char str2[]= “AVR LCD Test” ;
   unsigned char loop;
   PortInit();                    // LCD 출력 포트 설정
   LCD_Init();                    // LCD 초기화

   LCD_pos(0,0);                  // LCD 표시 위치를 0행 0열로 지정
   LCD_str(str1);                 // 문자열 str1을 LCD에 출력
   LCD_pos(1,0);                  // LCD 표시 위치를 1행 0열로 지정
   LCD_str(str2);                 // 문자열 str2을 LCD에 출력
   while(1)
   {
        for(loop =0; loop<5; loop++)
        {
            LCD_Shift(RIGHT);           // LCD 디스플레이 오른쪽으로 시프트
            _delay_ms(500);             // 이동 지연
        }
        Cursor_Home();          // 커서 홈 리턴
    }
}
```

예제 2: 시간 경과에 따른 문자 표시

- **"Current Time"을 1번째 라인에 DISPLAY한 후 두 번째 라인에는 AM/PM 시, 분, 초(초기 시작 12:00:00)를 출력하는 프로그램 작성**
  - **1초 단위의 카운터 필요 → 8bit 타이머 0,2 이용(50us), 비교 일치 인터럽트 이용**

초기 시작

C	u	r	r	e	n	t		T	i	m	e				
A	M		1	2	:	0	0	:	0	0					

1초 경과

C	u	r	r	e	n	t		T	i	m	e				
A	M		1	2	:	0	0	:	0	1					

2초 경과

C	u	r	r	e	n	t		T	i	m	e				
A	M		1	2	:	0	0	:	0	2					

3초 경과

C	u	r	r	e	n	t		T	i	m	e				
A	M		1	2	:	0	1	:	0	0					

LCD 제어 실험


```
#include "lcd.h “                    // lcd.h 파일이 있는 위치 지정
unsigned int cnt=0;
unsigned char Temp=0;
unsigned char sec=0, min=0, hour=0;

// 50us 간격으로 인터럽트 발생
void Init_Timer0(void)
{
    TCCR0 |= (1 << WGM01);       //TCCR0 레지스터 CTC 모드 설정
    OCR0 = 50;                   //50us마다 출력 비교를 한다
    TIMSK = (1<<OCIE0);          //출력비교 인터럽트 허가
}

// 50us마다 인터럽트 비교 일치 인터럽트가 발생
interrupt [TIM0_COMP] void timer0_out_comp(void)
{
     cnt++;                      //인터럽트 횟수 증가
                                 //50us의 인터럽트를 20000번 계수하여 1초의 시간을 만든다.
     if(cnt == 20000) // 50us * 20000 = 1sec
          {
             cnt = 0;
             sec++;
             if(sec >= 60) {
                     min++; sec = 0;
             }
             if(min >= 60) {
                     hour++; min = 0;
                 }
              if(hour>=24) hour = 0;
          }
}
```

LCD 제어 실험


```
int main(void)
{
    unsigned char str1[] = "Current Time";
    unsigned char str2[] = "AM 12:00:00";
    unsigned char AM[] = "AM";
    unsigned char PM[] = "PM";

    Temp = cnt = sec = min = 0;
    hour = 12;

    Init_Timer0();       // Timer 초기화

   // LCD 초기화
   PortInit();           // LCD 출력 포트 설정
   LCD_Init();           // LCD 초기화
   LCD_pos(0,0);         // LCD 포지션 0행 0열 지정
   LCD_STR(str1);        // 문자열 str1을 LCD 출력
   LCD_pos(1,0);         // LCD 포지션 1행 0열 지정
   LCD_STR(str2);        // 문자열 str2을 LCD 출력

   sei();                // 전역 인트럽트 허가
   TCCR0 |= 1 << CS01; //타이머0 클럭 공급
```

```
while(1)
{
      if(hour > 12)
      {
         LCD_pos(1,0);
         LCD_STR(PM);
         LCD_CHAR(((hour-12)/10)+'0');
         LCD_CHAR(((hour-12)%10)+'0');
       }
       else
       {
         LCD_pos(1,0);
         LCD_STR(AM);
         LCD_CHAR((hour/10)+'0');
         LCD_CHAR((hour%10)+'0');
        }

         LCD_pos(1,6);
         LCD_CHAR((min/10)+'0');
         LCD_CHAR((min%10)+'0'  );

         LCD_pos(1,9);
         LCD_CHAR((sec/10)+'0');
         LCD_CHAR((sec%10)+'0');
      }
}
```

예제 3: 스위치 상태에 따른 문자 표시

- **PORTB에 연결되어 있는 4개의 스위치의 상태에 따라 다음의 4가지 형태로 LCD에 출력하는 프로그램 작성**

PORTD.0 ON

	P	U	S	H		A	r	r	o	w		K	e	y	
	S	t	a	t	e		:		L	E	F	T			

PORTD.1 ON

	P	U	S	H		A	r	r	o	w		K	e	y	
	S	t	a	t	e		:		R	I	G	H	T		

PORTD.2 ON

	P	U	S	H		A	r	r	o	w		K	e	y	
	S	t	a	t	e		:		U	P					

PORTD.3 ON

	P	U	S	H		A	r	r	o	w		K	e	y	
	S	t	a	t	e		:		D	O	W	N			

LCD 제어 실험

※ byte 자료형은 기본 자료형이 아님
학습자가 unsigned char 로 재 정의 하여 사용 할 것.

```
#include "lcd.h”
typedef unsigned char Byte;

void Switch_Verify(void)
{
    Byte Left[] = "LEFT ";
    Byte Right[] = "RIGHT";
    Byte Up[] = "UP ";
    Byte Down[] = "DOWN ";
    Byte Emt[] = " ";
    Byte sw;

    sw = sw = (0x0f & PIND);

    switch(sw){
       case 0x0e : {LCD_STR(Left); break;}
       case 0x0d : {LCD_STR(Right); break;}
       case 0x0b : {LCD_STR(Up); break;}
       case 0x07 : {LCD_STR(Down); break;}
       default : LCD_STR(Emt); break;
     }
}
```

```
int main(void)
{
    Byte str1[] = "PUSH Arrow Key";
    Byte str2[] = "State : Plz key";
    DDRD = 0xF0;       // DIP Switch 입력 설정

    PortInit();        // LCD 출력 포트 설정
    LCD_Init();        // LCD 초기화
    LCD_pos(0,0);      // LCD 포지션 0행 1열 지정
    LCD_STR(str1);     // 문자열 str을 LCD 출력
    LCD_pos(1,0);      // LCD 포지션 0행 1열 지정
    LCD_STR(str2);     // 문자열 str을 LCD 출력

    while(1)
    {
        LCD_pos(1,8);
        Switch_Verify();
    }
}
```

LCD 제어 실험

한국공학대학교 TECH UNIVERSITY OF KOREA

예제 4: 한글 문자 표시

- "김치" 라는 한글을 LCD에 표시하는 프로그램 작성
- CGROM에 없는 문자를 만들어 LCD에 표시하는 문제로서, 내부 CGROM에 없는 문자를 만들어 표시하기 위해서는 CGRAM을 이용하여야 함

문자코드	CGRAM 어드레스	문자패턴 데이터	
7 6 5 4 3 2 1 0	5 4 3 2 1 0	7 6 5 4 3 2 1 0	
0 0 0 0 * 0 0 0	0 0 0	* * * 1 1 1 1	문자패턴의 예 "김"
		* * * 1 1	
		* * * 1 1	
		* * * 1 1	
		* * * 1 1	
		* * * 1 1 1 1 1	
		* * * 1 1	
		* * * 1 1 1 1 1	커서위치
0 0 0 0 * 0 0 1	0 0 1	* * * 1	문자패턴의 예 "치"
		* * * 1	
		* * * 1 1 1 1	
		* * * 1 1	
		* * * 1 1 1 1	
		* * * 1 1 1 1	
		* * * 1 1 1 1	
		* * * 1 1 1 1 1	커서위치

-설계된 문자 패턴과 CGRAM 사용법-

LCD 제어 실험

- CGRAM의 주소 비트 0-2 비트는 문자 패턴의 라인 위치를 나타낸다. 문자는 5 x 7을 사용하므로 8번째 라인은 커서 위치를 나타낸다. 본 예제에서는 커서를 사용하지 않을 것이므로 5 x 8 비트 패턴을 사용

- 문자 패턴의 열 위치는 CGRAM 데이터 비트 0-4 비트에 해당한다. CG RAM 데이터 비트 5-7 비트는 표시에 사용되지 않기 때문에 일반 데이터 RAM으로 사용될 수 있음

- CG RAM 문자 패턴은 문자 코드 비트 4-7이 0일 때 선택된다. '김'이라는 문자는 문자 코드 0에 해당되고, '치'라는 문자는 문자코드 1에 해당한다.

- 그럼 예제에서 주어진 문자를 CGRAM에 설계하여 써 넣는 과정을 살펴보자. 먼저 문자를 설계하여 배열에 저장한다. CGRAM 데이터 비트 0-4를 사용하여 이루어진다. 이 과정이 끝나면 이 문자를 CGRAM 주소에 세트하고, 첫 번째 '김'은 주소 0에 '치'은 주소 1에 써 넣음

LCD 제어 실험

※ byte 자료형은 기본 자료형이 아님
학습자가 unsigned char 로 재 정의 하여 사용 할 것.

```
#include "lcd.h “
//typedef unsigned char Byte;

void CGRAM_Set()
{
    int i;
    Byte kim[] = {0x1d, 0x05, 0x05, 0x00, 0x1f, 0x11, 0x11, 0x1f};
    Byte chi[] = {0x01, 0x09, 0x1d, 0x09, 0x15, 0x15, 0x15, 0x15};
    // CGRAM 사용(DB6 = set) 주소 설정 : CGRAM 0번지(0bx1000xxx)

    LCD_delay(1);
    LCD_Comm(0x40);              // CGRAM address set
    LCD_delay(1);

   for(i=0; i<8; i++)
      {
         Putc_LCD(kim[i]);       // 한글 김, data set 8 byte
         LCD_delay(1);
      }
   // CGRAM 사용(DB6 = set) 주소 설정 : CGRAM 1번지(0bx1001xxx)

    LCD_Comm(0x48); // CCGRAM address set
    LCD_delay(1);

    for(i=0; i<8; i++)
       {
          Putc_LCD(chi[i]);      //  ‘치’  문자를 CGRAM에 쓴다
          LCD_delay(1);
       }
}
```

```
int main(void)
{
    PortInit();             // LCD 출력 포트 설정
    LCD_Init();             // LCD 초기화

    CGRAM_Set();

    LCD_pos(0,0);           // LCD 표시 위치 1행 1열 지정
    LCD_delay(1);

    Putc_LCD(0x00);     //CGRAM에 작성된 0~7번지 메모리에 기록된 패턴 출력
    Putc_LCD(0x01);     //CGRAM에 작성된 8~15번지 메모리에 기록된 패턴 출력

    while(1);
}
```

Q & A