( )

"Micro processor" Summary

ppT 1. 3장 정리(!)

pT 1장. 목표

1. 일반적인 마이크로컴퓨터와 마이크로 컨트롤러 차이점.
2. μ-processor, μ-controller 발전과정
3. MCS-51 장단점
4. 응용분야

★ 마이크로 프로세서 (μ processor)

- 한개의 IC 칩 cpu. (Register. ALU. Multiplexer. decoder)

★ 마이크로 컨트롤러 (μ - controller)

★ cpu + ROM + RAM + I/O port
(데이터메모리) (프로그램메모리) (입출력)

★ 특징. 제품의 소형화. 경량화. 저가. 큰 신뢰성. 기능확장.

용도.
- 연산용 (데이터 처리)
- 제어용 (기계제어) 주목적.

8051 - μ controller.

( )

< 4 bit 8. 16. 32 bit >

AT 89 s 51

- AT: Atmel회사
- 89: Intel core
- s: C: Rom writer 필요. / S: Isp (In system programing) 지원
- 51: 8051 core. (하버드 구조. CISC)

{ Register. ALU. Control unit.
Bus. (Address. Data. control)

40 pin - 4port (×8) = 32 pin

- 8 pin - { Vcc , GND , Isp , CLK(2개) / interrupt 등 }

ROM 종류.

{ MASK, Programmable~
EP~ EEP~
Flash ~

8051 구조인. ACC 누산기 , Program State Word (상태 Reg)

Program Counter , Data Pointer Registor (DPTR)
외부메모리의 주소 지정.

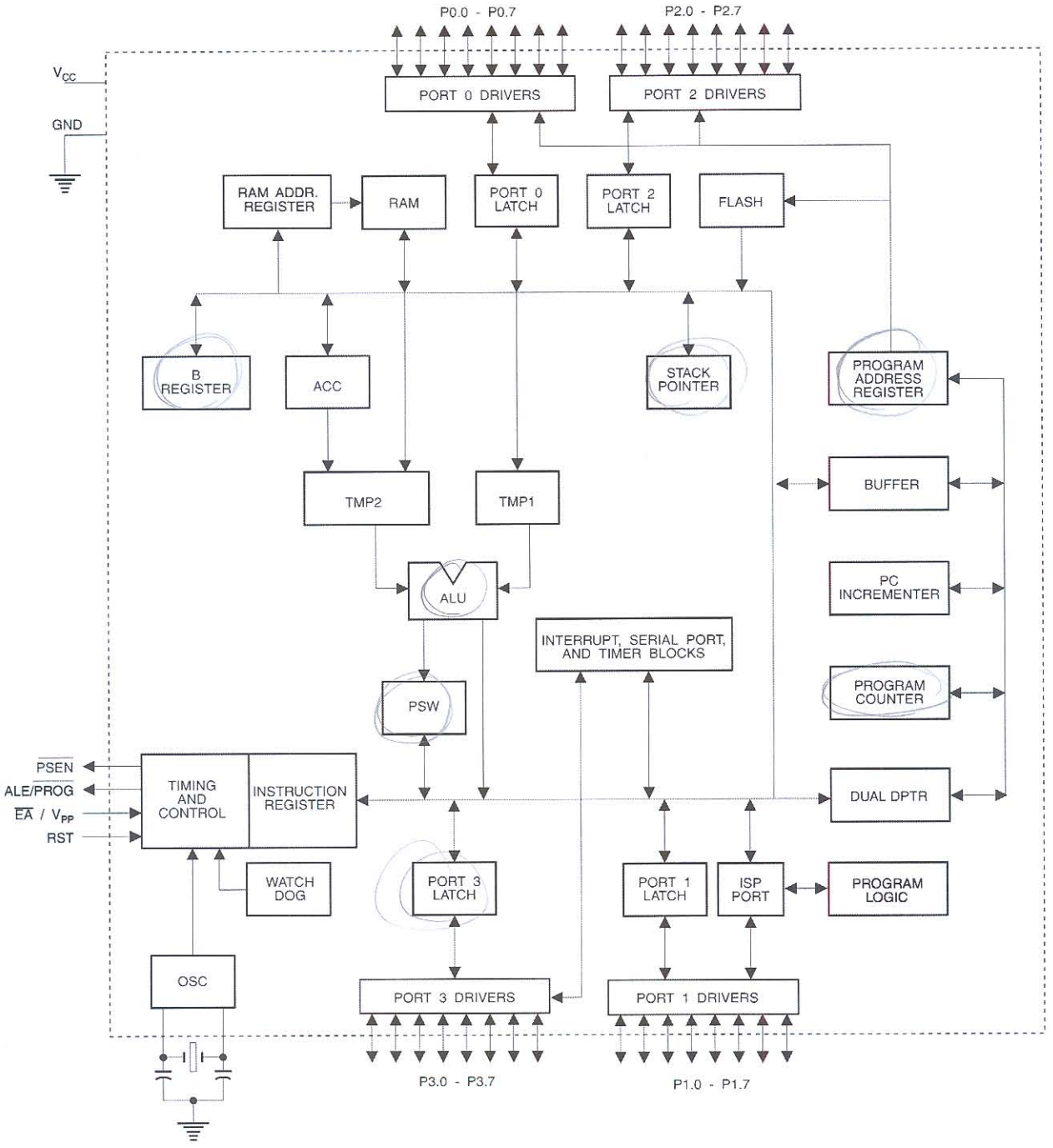
8051 기본특징

{ 8 bit CPU. 5V. (낮은 전력소모) bit 제어.
4Kb ROM (자체내장) . 128byte RAM , 128byte SFR 내장.

Special Function Register.

{ 외부 프로그램 메모리, 데이터 메모리 → 64kbyte 확장가능.
4개의 8 bit 입출력 port. → 외부장치 제어.

3. Block Diagram



( )

port 0. 양방향 In/out port. (pull-up 달아주어야한다)

* 외부메모리 (ROM. RAM) , 하위 Address 와 데이터 버스 사용

port 1. 내부풀업. (양방향 I/O port) , 범용 port.

port 2. 양방향 IO port.

외부 메모리 , 상위 어드레스 (A8 ~ 15) 사용.

port 3. • Timer / counter. (T0.T1 (P3.4,5)) 직렬포트 (RXD. TXD). (외부메모리 데이터. 입출력 제어선. etc.)
P3.0, 1
$\overline{WR}$, $\overline{RD}$

• 외부 인터럽트 $\overline{INT0,1}$
P3. 2, 3

RST = reset. XTAL 1, 2 → CLK.

31 번 pin. $\overline{EA}$ / V_{pp} 외부메모리 사용할 때 setting.

30 pin ALE / $\overline{prog}$ 어드레스. 데이터 구분해주는 핀.

29 pin. ROM에 저장된 Data를 읽을 때 사용. (Active low 상태)

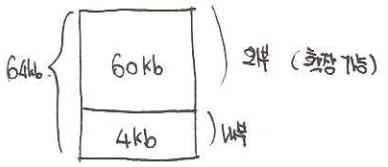
$\overline{PSEN}$ (program strobe Enable)

(XTAL 1 인버팅 증폭
2 ~ 출력

( )

8051 메모리 구조.

- 프로그램 메모리 (ROM)
  - 명령어가 저장되어 실행됨.
  - 읽기만 가능.



0000 ~ 0FFF. 16×3 = 48

1000 ~ FFFF

- 데이터 메모리 (RAM)

내부 데이터 메모리 - "Register. bank. bit address. Stack."

bank: 8byte씩 묶음. < Reg 0~7 > ↕ Bank 0~3.

bit address: 비트연산 연산을 지원.

* SFR - 범용 Reg. 기능 과 각종 칩 내장 장치 기능 수행.

외부 데이터 메모리

※

Table 5-1. AT89S51 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H								0CFH
0C0H								0C7H
0B8H	IP XX000000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0X000000							0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX	0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0	8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000 87H

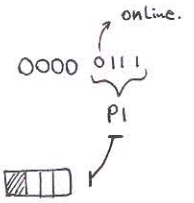
User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

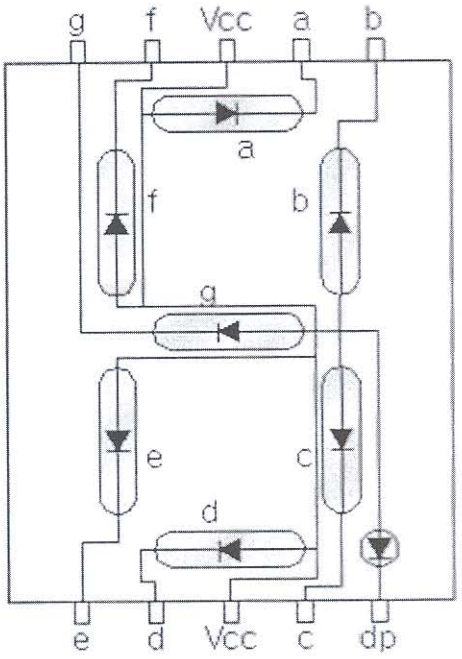
Interrupt Registers: The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the five interrupt sources in the IP register.

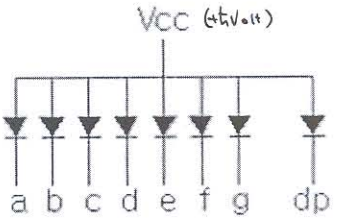
(7-2)

```
#include <io51.h>

void delay (char);

 void delay (char p)
 {
 int i, j ;

   for ( j = 0 ; j < p ; j++)

          for ( i = 0 ; i < 20000 ; i++) ;

 }

 void main(void)
 {

  for( ;; ) {

    P1 = 0x07 ; // 맨 왼쪽의 7-Segmnet 만 키도록 한다.
    P0 = 0xfe ; //a 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/


    P1 = 0x07 ;
    P0 = 0xfd ; //b 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0xfb ; //c 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0xf7 ; //d 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0xef ; //e 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0xdf ; //f 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0xbf ; //g 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/

    P1 = 0x07 ;
    P0 = 0x7f ; //p 만 키도록 한다.
    delay(10); // 육안으로 확인할 수 있도록 약 1초 정도의 시간 딜레이를 준다 .*/


      }
 }
```

< 7-segment 각 LED를 순차적으로 on-off >

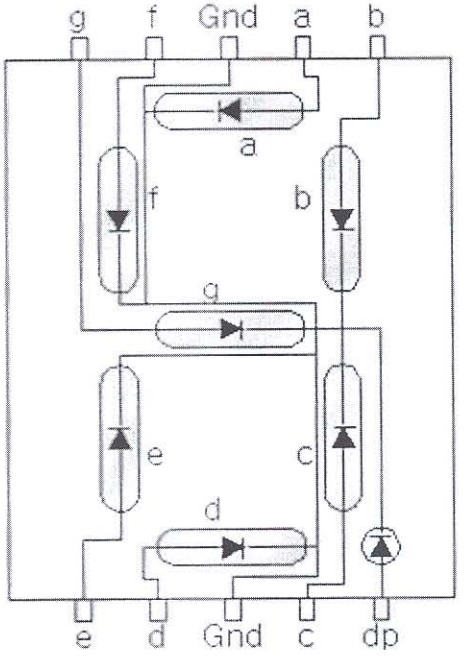


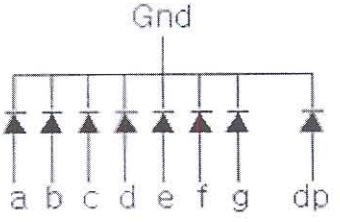




common-anode type

↳ μ-controller 에서 주로 쓰는 방법.





common-cathode type

(7-7 program) keyboard Scan *

{ SW0 → 0000 / SW4 → 4444 / SW8 → 8888
SW12 → CCCC 7-segment 에 출력 program.* }

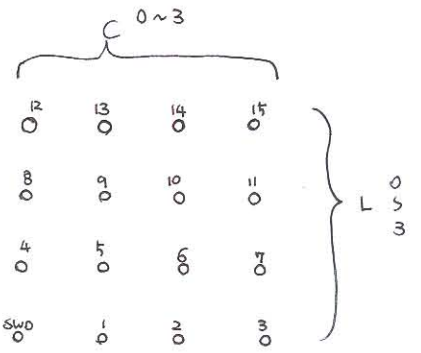
```
#include <io51.h>

bit C0    = 0xA0;
bit C1    = 0xA1;
bit C2    = 0xA2;
bit C3    = 0xA3;

bit L0    = 0xA4;
bit L1    = 0xA5;
bit L2    = 0xA6;
bit L3    = 0xA7;   → Port A의 8번째. ( Port A . 7 )
```

《 key 값 회로도 반드시 숙지할 것! 》

```
#define S7_1   0x07
#define S7_2   0x0b
#define S7_3   0x0d
#define S7_4   0x0e
```

0000 0111 *
0000 1011
" 1101
" 1110

```
const char Font[17]  = { 0xc0, 0xf9, 0xa4, //'0', '1', '2'
                         0xb0, 0x99, 0x92, // '3', '4', '5'
                         0x83, 0xf8, 0x80, // '6', '7', '8'
                         0x98, 0x88, 0x83, // '9', 'A', 'b'
                         0xc6, 0xa1, 0x86, // 'C', 'd', 'E'
                         0x8e, 0x7f } ;    // 'F', '.'

unsigned char KeyValue;

void Control_7Seg(int); char KeyScan(void);

void Control_7Seg(int N)
{
  unsigned int  i, d ;

   for ( i = 0 ; i < 100 ; i ++ ) {
       P1 = S7_1 ;          // 첫번째  7-Segmnet를 켜준다.
       P0 = Font[N] ;
       for(d = 0 ; d < 10; d++ ) ;

       P1 = S7_2 ;          // 두번째  7-Segmnet를 켜준다.
       P0 = Font[N] ;
       for(d = 0 ; d < 10; d++ ) ;

       P1 = S7_3;           // 세번째  7-Segmnet를 켜준다.
       P0 = Font[N];
       for(d = 0 ; d < 10; d++ ) ;

       P1 = S7_4 ;         // 네번째  7-Segmnet를 켜준다.
       P0 = Font[N] ;
       for(d = 0 ; d < 10; d++ ) ;
   }
}
```




```
char KeyScan(void)
{
 unsigned char Inkey = 0, key = 0; ;

   C0 = 0; C1 = 1; C2 = 1; C3= 1;

   Inkey = L3 ; key =Inkey << 3;
   Inkey = L2 ; key = key | (Inkey << 2 ) ;
   Inkey = L1 ; key = key | (Inkey << 1 ) ;
   Inkey = L0 ; key = key | Inkey ;

   if(key==0x0e) return 0;
   else if (key==0x0d) return 4 ;
   else if (key == 0x0b) return 8;
   else if (key == 0x07) return 12 ;
   else return 0xff ;

 }
```

첫번째 열 라인 "SW0. SW4 SW8 SW12" 인식하기 위한 setting.

| → or 연산.

0000 1110 = 0x0e
0000 1101 = 0x0d
0000 1011 = 0x0b
0000 0111 = 0x07

```
 void main(void)
  {

    P2 = 0xff ;

        for( ;; ) {
              KeyValue = KeyScan();
              Control_7Seg(KeyValue);

     }
  }
```

key scan 값을 키value에 넣어준다!

Segment 출력.

( )

제 5장 LCD controller (HD44780) 제어

83p. module pin 번호 및 사항에 대해 check!

92p. HD44780 instructions *

~

97p

98p ~ Initialyze. 및 program 확실히 알아둘것!

★ 6-3 (main) <LCD prgum>

```
#include <io51.h>

bit P1_0    = 0x90;
bit P1_1    = 0x91;
bit P1_2    = 0x92;

#define     LCD_RS      P1_0
#define     LCD_RW      P1_1
#define     LCD_E       P1_2

#define     LCD_DATA    P0

#define     ON          1
#define     OFF         2
#define     NO          0
#define     RIGHT       1
#define     LEFT        2


void E_Pulse(void);
void Func_set(void);
void Init_Lcd(void);
void delay (char);
void write_char(char);
void clrscr (void);
void lcd_disp ( char, char );
void write_lcd(char, char, char *);
void move_display (char);
void Entry_shift (char);
void move_cursor(char);
void cursor_home(void);
void display_ON_OFF(char, char, char);


void delay (char p)
{
  int i, j ;
    for ( j = 0 ; j < p ; j++)
          for ( i = 0 ; i < 20000 ; i++) ;
}

/* 앞장에서 알 수 있듯이 LCD로 데이터가 입력되려면 반드시 데이터를 래치시킨 상태에서 LED Enable신호에 펄스 하나를 내주어야 된다. */

void E_Pulse(void)
{
          char i;
          LCD_E = 1;            /* LCD Enable 신호 'HIGH' */
          for (i=0; i< 20 ; i++); /* LCD Enable 펄스를 어느 시간 이상 지연해 준다.*/
          LCD_E = 0 ;           /* LCD Enable 신호 'LOW' */
}


/*  LCD컨트롤러 HD44780의 기능을 설정하는 함수로 데이터 라인을 8비트 인터페이스, 2라인 디스플레이, 5*7글자 폰트를 설정한다.  */
void Func_set(void)
{
          LCD_RW = 0;           /* LCD R/W */
          LCD_RS = 0;           /* LCD RS   */

          LCD_DATA = 0x38 ;     /* LCD DB0~DB7까지의 값이다. */

          E_Pulse();            /* 모든 명령(데이터 입력)후 반드시 LCD Enable 펄스를 하나 내주어야 한다 */
}


void Init_Lcd(void)       /* LCD를 초기화 함수이다.  */
{
  int i ;
  LCD_E = 0 ;

  /*  전원이 4.5V  이상으로 올라간 후에 15ms 이상을 기다려야 한다.  */
  for ( i=0 ; i<2000 ; i++); /*  i값 1000은 컴파일러마다 틀릴 수 있다. */
  Func_set() ;

  /*  4.1ms 이상 기다려야 한다.  */
  for ( i=0 ; i<1000 ; i++);  /*  i값 500은 컴파일러마다 틀릴 수 있다. */
  Func_set() ;

  /*  100us 이상 기다려야 한다.  */
  for ( i=0 ; i<200 ; i++) ;  /*  i값 100은 컴파일러마다 틀릴 수 있다. */
  Func_set() ;

  /*  디스플레이 ON/OFF 제어 : 디스플레이와 커서를 ON하고, 커서는 깜빡이게 된다. 초기화 직후이기 때문에 글자는 하나도 나타나지 않는다.  */
   LCD_DATA = 0x0f;
   E_Pulse() ;

   /*  엔트리 모드 세트 : 어드레스는 +1, D.D. /C.G.RAM 쓰기 후에 커서는 오른쪽으로 시프트하도록 한다.이때 디스플레이는 시프트하지 않는다 */
  LCD_DATA = 0x06 ;
  E_Pulse() ;
}

void write_char(char s)
{
```

```
 LCD_RS = 1;
 LCD_DATA = s;
 E_Pulse();
}

void clrscr(void)
{
            int i;

            LCD_RS = 0 ;
            LCD_RW = 0 ;
            LCD_DATA = 0x01 ;
            E_Pulse() ;


            for (i = 0 ; i < 600 ; i++ ); /* 최소한 1.64ms의 시간 지연을 주어야 한다. */
}

void lcd_disp ( char x, char y )
{
            LCD_RS = 0 ;
            LCD_RW = 0 ;

            if (y==0) LCD_DATA = x +0x80 ;
            else if (y==1) LCD_DATA = x + 0xc0 ;
            E_Pulse() ;
}

void write_lcd(char x, char y, char *str )
{
            lcd_disp(x, y) ;
            while(*str)
            write_char(*str++) ;
}


  void display_ON_OFF(char d, char c, char b)
  {

  unsigned char display = 0x08 ;

    LCD_RS = 0 ;
    LCD_RW = 0 ;

    if(d==ON) d = 0x04 ;
    else d = 0x00 ;

    if(c==ON) c = 0x02 ;
    else c = 0x00 ;

    if(b==ON) b = 0x01 ;
    else b = 0x00 ;

    LCD_DATA = display |d  |c |b ;

    E_Pulse() ;
  }

  void Entry_shift (char p)

  {
      LCD_RS = 0 ;
    LCD_RW = 0 ;

   if(p==RIGHT)  LCD_DATA = 0x05 ;
   else if (p==LEFT)  LCD_DATA = 0x07 ;
   else if (p==NO)   LCD_DATA = 0x06 ;
   E_Pulse() ;

  }


 void move_cursor(char p)
  {

    LCD_RS = 0 ;
    LCD_RW = 0 ;

    if(p==RIGHT)   LCD_DATA = 0x14 ;
    else if(p==LEFT)   LCD_DATA = 0x10 ;
    E_Pulse() ;
  }


  void move_display (char p)
  {

   LCD_RS = 0 ;
   LCD_RW = 0 ;

   if (p == LEFT)   LCD_DATA = 0x18 ;
   else if (p == RIGHT)   LCD_DATA = 0x1c ;
```

```
  E_Pulse() ;
 }

void cursor_home(void)
 {
int i ;
   LCD_RS = 0 ;
   LCD_RW = 0 ;
   LCD_DATA = 0x02 ;

   /*최소 1.64 ms 지연 */
   for( i = 0 ; i < 600 ; i++) ;
   E_Pulse() ;
 }


void main(void)
 {
 char t ='A';
 int i ;
 Init_Lcd() ;
 clrscr();

write_lcd (0, 0, "MCS- 51") ;
write_lcd (0, 1, "Welcom MCS-51 ") ;

delay(5) ;

//커서를 초기 위치로
 lcd_disp(0,0);

 /* 디스플레이를 OFF시킨다. */
 display_ON_OFF(OFF, ON, ON) ;
 delay(10) ;

 /* 디스플레이를 ON시킨다. */
 display_ON_OFF(ON, ON, ON) ;
 delay(10) ;

 /*커서를 오른쪽, 왼쪽으로 시프트 시킨다. */
 for (i = 0 ; i< 10 ; i++) {

           if(i<5) move_cursor(RIGHT) ;
           else move_cursor(LEFT) ;
      delay(2) ;
 }

 /*디스플레이 전체를 오른쪽, 왼쪽으로 시프트시킨다. */
 for( i = 0 ; i <10 ; i++) {

           if(i<5) move_display(RIGHT) ;
           else move_display(LEFT) ;
           delay(2) ;
 }


 /* 커서 깜빡임을 OFF시킨다. */
 display_ON_OFF(ON, ON, OFF) ;
 delay(10) ;
 /*커서 깜빡임을 ON시킨다. */
 display_ON_OFF(ON, ON, ON) ;
 delay(10) ;

 clrscr() ;


 /*
   엔트리 모드를 사용하여 글자를 쓰면서
   디스플레이를 왼쪽으로 시프트시킨다.
 */
 lcd_disp(10,0);
 for(i = 0 ; i < 10 ; i++) {
    Entry_shift(LEFT) ;
           write_char(t++) ;
           if(t>'Z') t = 'A' ;
           delay(2) ;
 }

 /*
 엔트리 시프트를 본래대로 환원하고 디스플레이를 클리어한다.
 */
 Entry_shift(NO) ;
 clrscr() ;
 write_lcd(0, 1, "End Of Demo.") ;
 cursor_home();

 }
```