제13강

C51 프로그래밍 II

C소스와 ASM소스간 인터페이싱

포트 입출력

* DotMatrix & Keypad 모듈제어

CLCD 제어

실습과제

Ref.) Chapter 12, 13, C51_02.zip 다운로드

CnASM 인터페이싱

* C 소스와 Assembly 소스간 인터페이싱

: 인자 순서 및 유형에 따라 사용되는 범용레지스터가 예약됨

■ 인자 전달시

인자수	char, 1byte ptr	int, 2byte ptr	long, float	generic ptr
1	R7	R6, R7	R4,..,R7	R1,..,R3
2	R5	R4, R5	R4,..,R7	R1,..,R3
3	R3	R2, R3		R1,..,R3

■ 반환값의 전달시

반환 유형	레지스터	저장 형식
bit	C flag	
(unsigned) char, 1byte pointer	R7	
(unsigned) int, 2byte pointer	R6, R7	MSB=R6, LSB=R7
(unsigned) long	R4,..,R7	MSB=R4, LSB=R7
float	R4,..,R7	32-Bit IEEE format
Generic pointer	R1,..,R3	Mem.type=R3, MSB=R2, LSB=R1

CnASM 인터페이싱(계속)

- 아래 소스의 char func_a(unsigned char)
  : char 형은 R7을 통하여....(C51_02\c2asm_app 폴더)

C 소스

```
//=======================
//   c2asm.c
//=======================
extern unsigned char func_a(unsigned char);
extern void func_c(void);

void main(void) {
        unsigned char c;

        c = func_a(0x30);     // increment
        func_c();

        while(1);
}

//=======================
//   func_c.c
//=======================
void func_c(void) {
        int  i = 2;

        i++;
}
```

Assembly 소스

```
;=============================
;   func_a.asm
;=============================
?PR?func_a      SEGMENT CODE
        PUBLIC      _func_a
; func_a function with global ?PR?func_a
; Relocatable segment

_func_a:
        USING 0     ; Use register bank 0
        MOV A,R7    ; arg. Passing by R7
        INC A

        MOV R7,A    ; return the value by R7
        RET
```

예제실습

[실습9] C와 어셈블리소스간 인터페이싱

: C51_02\C51nASM(C와 어셈블리소스간 인터페이싱) 폴더

: 데이터 0x1189와 0x80의 가산/감산 프로그램

```
//=====================================
// C51_09.C
//      main c51 file ... interfacing c51 and asm files
//=====================================
extern void gotoMonior(void);    // assembly func
extern int addIntChar(unsigned int, unsigned char);  // assembly func
extern void writeMem8100(unsigned int);              // assembly func
extern int subIntChar(unsigned int val1, unsigned char val2);  // C func

void main(void) {
   unsigned int xdata *mem = 0x8120;
   unsigned int opr1, res;
   unsigned char opr2;

   opr1 = 0x1189;            // int
   opr2 = 0x80;              // char
```

```
    res = addIntChar(opr1, opr2);
          // 인자로 unsigned int와 unsigned char를 전달
          // opr1은 R6, R7로, opr2는 R5로 전달, res는 R6, R7로 반환
    writeMem8100(res);          // res은 R6, R7로 전달

    res = subIntChar(opr1, opr2);
    *mem = res;

    gotoMonitor();
}

;=======================================
// C51_09_A.ASM
//      sub asm file ... interfacing c51 and asm files
;=======================================
addIntChar    SEGMENT CODE
    PUBLIC    _addIntChar
    RSEG      addIntChar

_addIntChar:              ; (R6, R7) <- (R6, R7) + (#0, R5)
    MOV  A,   R7
    ADD  A,   R5
    MOV  R7,  A

    MOV  A,   R6
    ADDC A,   #0
    MOV  R6,  A
    RET
```

```
writeMem8100 SEGMENT CODE
    PUBLIC    _writeMem8100
    RSEG      writeMem8100

_writeMem8100:                  ; (8100H) <- (R6, R7)
    MOV DPTR,#08100H
    MOV A, R6
    MOVX @DPTR, A

    INC DPTR
    MOV A, R7
    MOVX @DPTR, A
    RET

    END


//=====================================
// C51_09_C.C
//      sub c51 file ... interfacing c51 and asm files
//=====================================
int subIntChar(unsigned int val1, unsigned char val2) {
    unsigned int tmp;

    tmp = val1 - val2;
    return tmp;
}
```

실습과제

[과제1] C와 ASM 코드간 인터페이싱 I

: [실습9] 소스 참조, P1에 FND 모듈 연결

: 함수의 호출 결과를 P1으로 출력하여 육안 확인토록

(addIntChar(), subIntChar())

[과제2] C와 ASM 코드간 인터페이싱 II

: [실습9] 소스 참조

: 다음의 함수(인자의 순서 바꿈) 호출하여 바른 결과 얻도록...

```
res = addIntChar(opr2, opr1);
```

포트 입출력(C51_02\INOUT 폴더)

* 입출력 처리

```
#include <reg51.h>

    :
unsigned char tmp;

tmp = P1;                  // P1으로부터 입력
P1 = 0x69;                 // P1으로 출력

tmp = P1 & 0x0F;       // 유효 하위 니블 값만 tmp로 입력
tmp = (~P1) & 0x0F;
                  // 반전후, 유효 하위 니블 값만 tmp로 입력
    :
```

포트 입출력(계속)

* 시간지연함수 : delay()

```
void delay(unsigned int t) {
    while(t--);
}
```

: 대응하는 어셈블리 코드

```
    MOV t,R6              ; 2T x 1
    MOV t+01H,R7          ; 2T x 1
?C0001:
    MOV A,t+01H           ; 1T x (256 x R6 + R7 +1)
    DEC t+01H             ; 1T x (256 x R6 + R7 +1)
    MOV R6,t              ; 2T x (256 x R6 + R7 +1)
    JNZ ?C0104            ; 2T x (256 x R6 + R7 +1)
    DEC t                 ; 1T x (R6 +1)
?C0104:
```

```
MOV R7,A            ; 1T x (256 x R6 + R7 +1)
MOV A,R7            ; 1T x (256 x R6 + R7 +1)
ORL A,R6            ; 1T x (256 x R6 + R7 +1)
JNZ ?C0001          ; 2T x (256 x R6 + R7 +1)
RET                 ; 2T x 1
```

총 소요 머신사이클 수 :

(2817 x R6 + 11 x R7 + 18) T

* 0.5sec지연시 인자값 결정

2817 x R6 + 11 x R7 + 18 = 500000

R6 = 500000/2817 = 177.49... = 177 = B1H

2817 x 177 + 11 x R7 + 18 = 500000

R7 = (500000 – (498609 + 18)) / 11

= 1373/11 = 124.818 = 125 = 7DH

따라서, delay(0xB17D); // 약 0.5sec, t≅45,000

포트 입출력(계속)

* 시간지연함수 : msec_delay()

```
void msec_delay(unsigned int ms) {
      unsigned int i, j;

      for(i=1; i<=ms; i++)
            for(j=1; j<=125; j++) ;            // 약 1msec
}
```

* 시간지연함수 소프트웨어 모듈(라이브러리)

delay.h : 헤더파일

delay.c : 함수정의 소스

예제실습

[실습1] LED 제어

```
//=====================================
// INOUT_01.C
//  LED control
//       P1_L : LED module
//=====================================
#include <reg51.h>
#include "delay.h"

void main(void) {
   unsigned char led;

   led = 0x01;
   while(1) {
        P1 = ~led;
        delay(0xB17D);                  // 0.5sec

        led = (led << 1);
        if(led == 0x10)
             led = 0x01;
   }
}
```

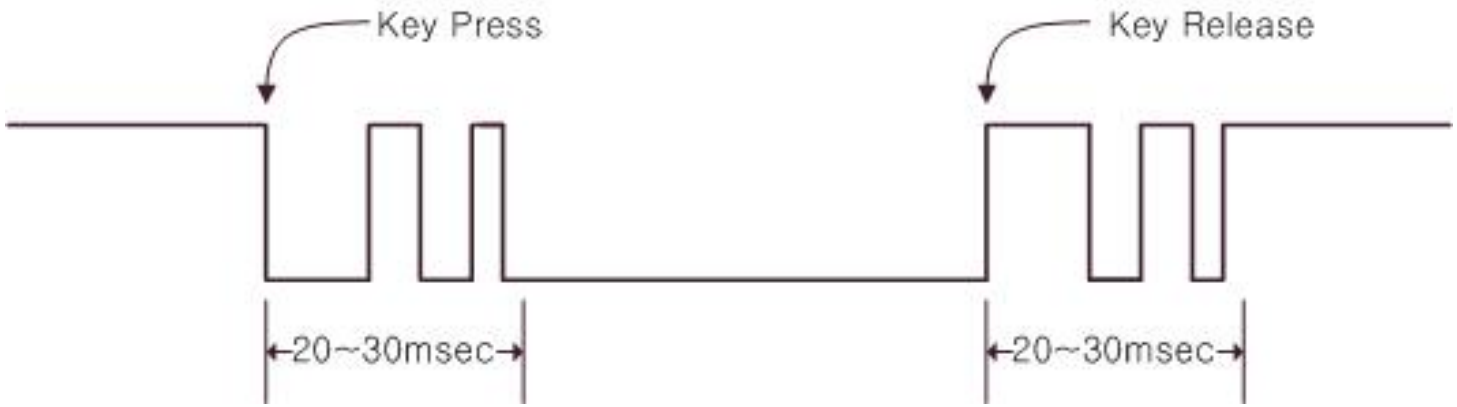
예제실습(계속)

[실습2] 버튼입력에 대응하는 LED 제어

```
//=====================================
// INOUT_02.C
//  Button switch & LED control
//       P1_H : Button module, P1_L : LED module
//=====================================
#include <reg51.h>

void main(void) {
   unsigned char pat;

   pat = 0xFF;                    // initial pattern, all LED OFF
   P1 = pat;

   while(1) {
        pat = P1;                 // input P1
        P1 = (pat >> 4) | 0xF0;
                                  // output, swap high and low nibble
   }
}
```

포트 입출력(계속)

* 버튼 입력시 채터링(바운싱) 현상 및 디바운싱



1) 시간지연 함수를 활용 (p.373)

2) 다수판독에 의한 안정화 (p.373)

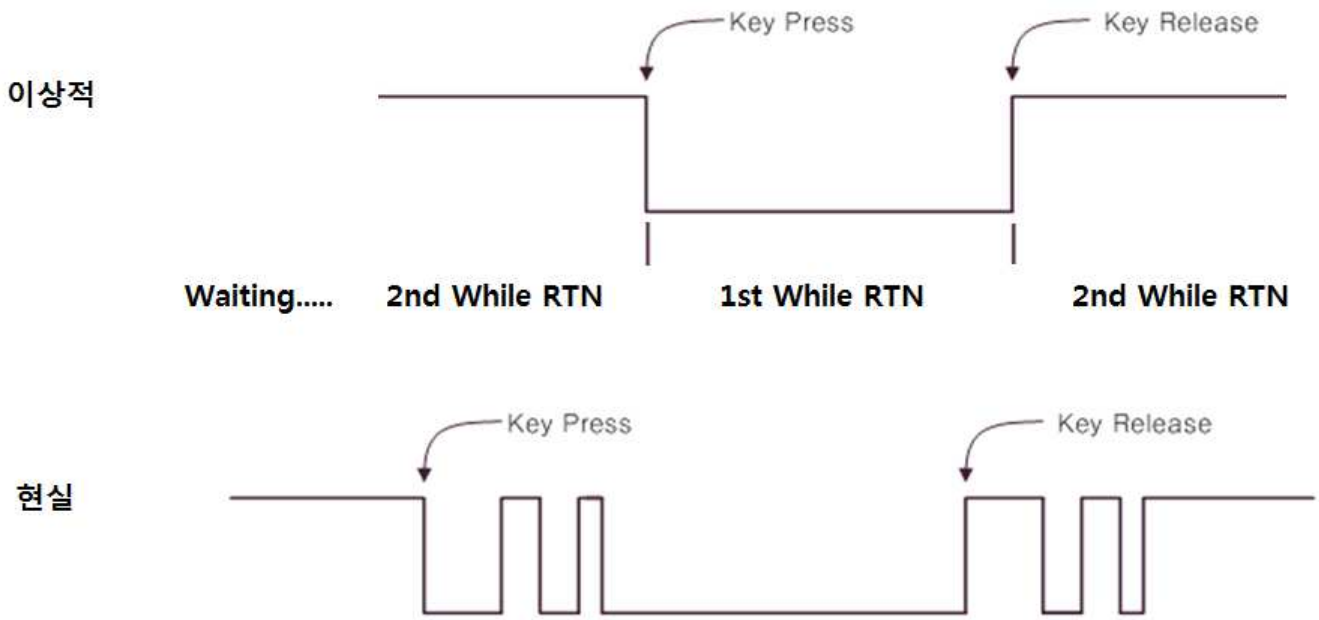
예제실습(계속)

[실습3] 채터링현상 (P1^7 bit)

```
//=====================================
// INOUT_03.C
//  bouncing
//       P1_H : Button module, P1_L : LED module
//=====================================
#include <reg51.h>

void main(void) {
   unsigned char pat;
   unsigned char in;

   pat = 0x01;                          // 정논리 패턴
   while(1) {
        P1 = ~pat;

        while((~P1 & 0x80)) ;           // wait until msb release
        while(!(~P1 & 0x80)) ;          // wait until msb press

        pat = (pat << 1);
        if(pat == 0x10)
             pat = 0x01;
   }
```

```
}
```

* 이상과 현실


이상적
Key Press
Key Release
Waiting.....
2nd While RTN
1st While RTN
2nd While RTN
현실
Key Press
Key Release

예제실습(계속)

[실습4] 디바운싱 (지연루틴 활용)

```
//=====================================
// INOUT_04.C
//  debouncing ... using delay
//       P1_H : Button module, P1_L : LED module
//=====================================
#include <reg51.h>
#include "delay.h"

void main(void) {
   unsigned char pat;

   pat = 0x01;
   while(1) {
        P1 = ~pat;

        while(!(~P1 & 0xF0)) ;          // 눌릴 때까지 대기
        msec_delay(30);                 // 눌릴 때 디바운싱

        while(~P1 & 0xF0) ;             // 뗄 때까지 대기
        msec_delay(30);                 // 뗄 때 디바운싱

        pat = (pat << 1);
```

```
            if(pat == 0x10)
                pat = 0x01;
        }
    }
```

실습과제

[과제1] 다수판독에 의한 디바운싱

[과제2] FND 모듈 응용

: 16진 숫자 표시하는 프로그램 작성

: 1초 간격으로

[과제3] 버튼 입력값만큼 LED 점멸

[과제4] 버튼에 따라 각기 다른 동작

: 동작은 각각 함수로 구현

DotMatrix/Keypad 모듈

* C51_02\DOTnKEY 폴더

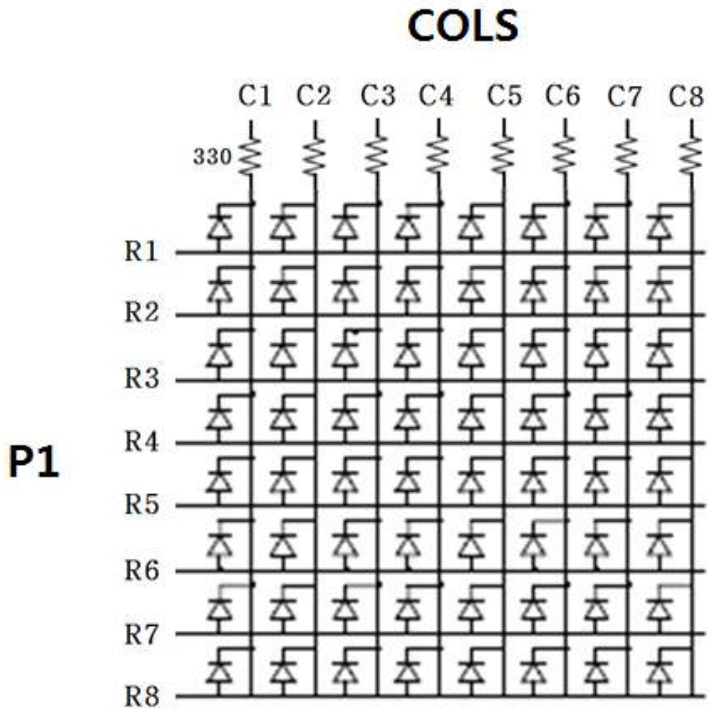
  : 관련 제어 방법만 소개.....

DotMatrix 모듈

* DotMatrix 모듈 회로 (8x8)

: 64개의 LED 2차원배열

: 8비트 포트 2개 필요



DotMatrix 모듈(계속)

* DotMatrix 모듈의 동적제어

: 인간 시각의 잔상효과를 활용

: 칼럼선택을 위한 8비트, 패턴데이터를 위한 8비트 제어신호

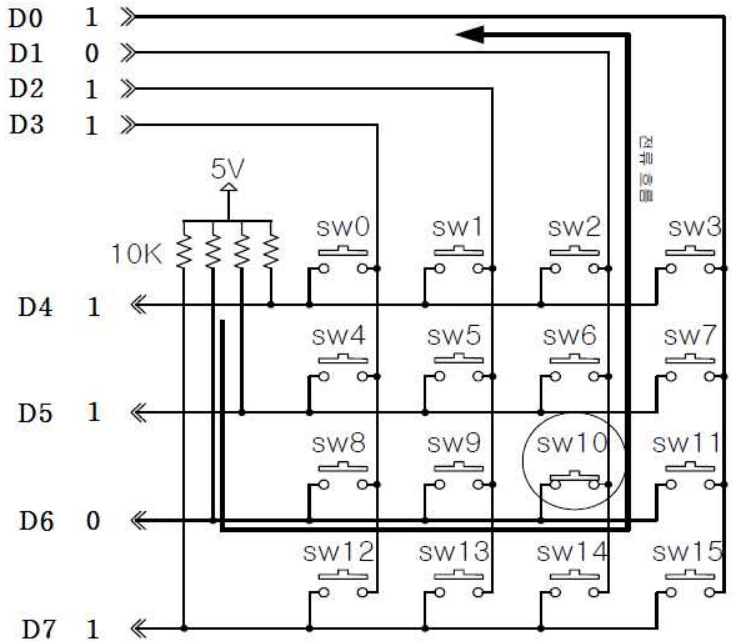
: 한 시점에서 한 칼럼만 선택후, 패턴데이터 출력, 고속으로

```
unsigned char colData = 0x01;
unsighed char i;

COLS = colData;
i = 0;
while(1) {
      delay(10);                        // short delay
      if(++i == 8)
            i = 0;

      COLS = ~(colData << i);           // low signal to only a col
      P1 = 0xFF;                        // all bits ON pattern
}
```

Keypad 모듈

* Keypad 모듈 회로 (4x4)

: 16개의 버튼스위치



Keypad 모듈(계속)

* Keypad 모듈의 동적제어

: 행과 열을 위한 각각 4비트씩, 총 8비트로 제어

: 특정 열을 선택하고, 행의 정보 읽음

: 열선택 신호(4비트)와 행 판독정보(4비트)로 코드화

```
out = 0x01;                    // 특정 열 선택
for(i=0; i<4; i++) {
     P1 = ~out;                // 특정 열에 Low 신호 인가

     in = (~P1) & 0xF0;        // 상위 니블 판독
     if(in) {
          in |= out;           // 코드 생성
          break;
     }
     out <<= 1;                // 다음 열
}
```

CLCD 제어(C51_02\CLCD 폴더)

* 8비트 제어모드용 라이브러리

```
//=====================================
// clcd8.h
//  LCD 8bit interface
//=====================================
#ifndef __CLCD8_H__
#define __CLCD8_H__

#define LCD_I_WR    (*(unsigned char xdata *)0xF800)
#define LCD_D_WR    (*(unsigned char xdata *)0xF801)
#define LCD_I_RD    (*(unsigned char xdata *)0xF802)
#define LCD_D_RD    (*(unsigned char xdata *)0xF803)

void initLCD_8(void);
void writeCommand_8(unsigned char cmd);
void putChar_8(unsigned char ch);
void putStr_8(unsigned char *str);
void checkBF(void);
void delayBF(void);

#endif
```

```
//=====================================
// clcd8.c
//  LCD 8bit interface
//=====================================
#include "clcd8.h"
#include "delay.h"

// LCD init.
void initLCD_8(void) {
    delay(400);
    writeCommand_8(0x38);     // Function Set, 8bit interface mode
    writeCommand_8(0x38);
    writeCommand_8(0x38);
    writeCommand_8(0x08);     // Display on, cursor off, blink off
    writeCommand_8(0x0E);     // Display on cursor on, blink off
    writeCommand_8(0x06);     // cursor increment, no shift
}

// LCD command write
void writeCommand_8(unsigned char cmd) {
    checkBF();                // check Busy Flag

    LCD_I_WR=cmd;
}

// LCD data write(write a char to LCD)
void putChar_8(unsigned char ch) {
    checkBF();
```

```
    LCD_D_WR = ch;
}

// write a string to LCD
void putStr_8(unsigned char *str) {
    unsigned char i;

    i = 0;
    do {
        putChar_8(str[i]);
    } while(str[++i] != 0);              // check end of string
}

// busy flag check
void checkBF(void) {
    while(LCD_I_RD & 0x80) ;             // wait until BF==0
}

// delay instead of busy flag check
void delayBF(void) {
    msec_delay(2);                       // 2msec
}
```

CLCD 제어(계속)

* 4비트 제어모드용 라이브러리

```
//=====================================
// file : clcd4.h
//  LCD 4bit interface
//=====================================
#ifndef __CLCD4_H__
#define __CLCD4_H__

#define FUNCSET     0x28                /* Function Set */
#define ENTMODE     0x06                /* Entry Mode Set */
#define ALLCLR      0x01                /* All Clear */
#define DISPON      0x0C                /* Display On */
#define DISPOFF     0x08                /* Display Off */
#define LINE2       0xC0                /* 2nd Line Move */
#define HOME        0x02                /* Cursor Home */

#define LCD_I_WR    (*(unsigned char xdata *)0xF800)
#define LCD_D_WR    (*(unsigned char xdata *)0xF801)
#define LCD_I_RD    (*(unsigned char xdata *)0xF802)
#define LCD_D_RD    (*(unsigned char xdata *)0xF803)

void initLCD_4(void);
void writeCommand_4(unsigned char cmd);
```

```
void putChar_4(unsigned char ch);
void putStr_4(unsigned char *str);
void checkBF(void);
void delayBF(void);

#endif


//=======================================
// file : clcd4.c
//  LCD 4bit interface
//=======================================
#include "clcd4.h"
#include "delay.h"

// LCD init.
void initLCD_4(void) {
    delay(1500);
    writeCommand_4(0x20);
    delay(500);
    writeCommand_4(0x20);
    delay(100);
    writeCommand_4(0x20);

    writeCommand_4(0x28);
    writeCommand_4(0x28);
    writeCommand_4(FUNCSET);       // Function Set, 4bit mode
    writeCommand_4(DISPOFF);       // Display on, cursor off, blink off
```

```
    writeCommand_4(DISPON);         // Display on, cursor off, blink off
    writeCommand_4(ENTMODE);        // cursor increment, no shift
}

// LCD command write
void writeCommand_4(unsigned char cmd) {
    unsigned char tmp;

    checkBF();                      // check Busy Flag
    tmp = cmd & 0xF0;               // get higher nibble
    LCD_I_WR = tmp;

    //checkBF();                    // check Busy Flag
    tmp = cmd & 0x0F;               // get lower nibble
    tmp <<= 4;                      // shift to higher nibble
    LCD_I_WR = tmp;
}


// LCD data write(write a char to LCD)
void putChar_4(unsigned char ch) {
    unsigned char tmp;

    checkBF();
    tmp = ch & 0xF0;                // get higher nibble
    LCD_D_WR = tmp;

    //checkBF();
```

```
    tmp = ch & 0x0F;            // get lower nibble
    tmp <<= 4;                  // shift to higher nibble
    LCD_D_WR = tmp;
}

// write a string to LCD
void putStr_4(unsigned char *str) {
    unsigned char i;

    i = 0;
    do {
        putChar_4(str[i]);
    } while(str[++i] != 0);     // check end of string
}

// busy flag check
void checkBF(void) {
    while(LCD_I_RD & 0x80) ;    // wait until BF==0
}

// delay instead of busy flag check
void delayBF(void) {
    msec_delay(2);
}
```

예제실습

[실습1] 8비트 모드

```
//=====================================
// CLCD_01.C
//  8bit interface, BF check
//  software module : clcd8.h, clcd8.c
//=====================================
#include "clcd8.h"

void main(void) {

    initLCD_8();                        // initialize LCD

    writeCommand_8(0x01);               // 1st line
    putStr_8("KUT 8051 Board");

    writeCommand_8(0xC0);               // 2nd line
    putStr_8("LCD Test OK!");

    while(1);                           // loop
}
```

예제실습(계속)

[실습2] 4비트 모드

```
//=====================================
// CLCD_02.C
//  4bit interface, BF check
//  software module : clcd4.h, clcd4.c
//=====================================
#include "clcd4.h"

void main(void) {

    initLCD_4();                        // initialize LCD

    writeCommand_4(0x01);               // 1st line
    putStr_4("KUT 8051 Board");

    writeCommand_4(0xC0);               // 2nd line
    putStr_4("LCD Test OK!");

    while(1) ;                          // loop
}
```

예제실습(계속)

[실습3] 문자출력

```
//=====================================
// CLCD_03.C
//  display char patters in CG RAM/ROM
//   software module : clcd8.h, clcd8.c
//=====================================
#include "clcd8.h"
#include "delay.h"

static unsigned char lcdTitle[] = "CG RAM/ROM Pats ";

void main(void) {
    unsigned char ch;
    unsigned char cnt;

    initLCD_8();                        // initialize LCD

    writeCommand_8(0x01);               // 1st line
    putStr_8(lcdTitle);                 // put title

    writeCommand_8(0xC0);               // 2nd line
    putStr_8("....... ");
```

```
    cnt = 0;
    while(1) {
        writeCommand_8(0xCA);          // 2nd line, 10Col
        ch = (unsigned char)cnt;
        putChar_8(ch);

        delay(50000);
        delay(50000);
        cnt++;
    }
}
```

예제실습(계속)

[실습4] 사용자패턴 등록 및 출력

```
//=====================================
// CLCD_04.C
//  8bit interface, user pattern
//  software module : clcd8.h, clcd8.c
//=====================================
#include "clcd8.h"

void regUserPattern(void);
void putUserPattern(void);

static unsigned char lcdTitle[] = "KOREATECH DICE";
unsigned char userPattern[64] = {                  // 5x8 font
    0x1E, 0x12, 0x1E, 0x00, 0x04, 0x07, 0x04, 0x04,   //마
    0x09, 0x15, 0x09, 0x01, 0x01, 0x01, 0x01, 0x01,   //이
    0x1F, 0x01, 0x07, 0x01, 0x00, 0x1F, 0x00, 0x00,   //크
    0x1F, 0x01, 0x1F, 0x10, 0x1F, 0x00, 0x04, 0x1F,   //로
    0x1F, 0x0A, 0x0A, 0x1F, 0x00, 0x1F, 0x00, 0x00,   //프
    0x1F, 0x01, 0x1F, 0x10, 0x1F, 0x00, 0x04, 0x1F,   //로
    0x04, 0x0A, 0x11, 0x00, 0x05, 0x1D, 0x05, 0x05,   //세
    0x04, 0x0A, 0x11, 0x00, 0x01, 0x1F, 0x01, 0x01    //서
};
```

```
int main(void) {

    initLCD_8();                        // initialize LCD

    regUserPattern();                   // register userPattern

    writeCommand_8(0x01);               // 1st line
    putStr_8(lcdTitle);                 // put title

    writeCommand_8(0xC0);               // 2nd line
    putUserPattern();                   // put user patterns

    while(1) ;                          // loop

}

void regUserPattern(void) {
    unsigned char cmd;
    unsigned char i;

    cmd = 0x40;                 // 0100_0000, CG_RAM 주소 설정
    writeCommand_8(cmd);

    for(i=0; i<64; i++)
        putChar_8(userPattern[i]);
}

void putUserPattern(void) {
```

```
    unsigned char i;

    for(i=0; i<8; i++)
        putChar_8(i);              // i...pattern address
}
```

실습과제

[과제1] CLCD 주소접근방법

: [실습1]의 소스를 활용하여,

: CLCD 주소 접근 방법을 달리하여

(매크로, _at_ 키워드 등)

[과제2] 사용자패턴 등록 및 출력

: [실습4]의 소스를 활용하여,

: 사용자패턴을 랜덤한 순으로 등록 후, 정상출력