4.1 ADS v1.2를 사용한 C언어 프로그래밍 기초

이 절에서는 퍼스널컴퓨터에서 ARM사의 ADS v1.2 C컴파일러를 사용하여 C언어로 사용자 프로그램을 작성하고 이를 번역하여 생성된 바이너리 파일을 *OK-7S256* 키트에 다운로드하고 실행하는 방법을 익힌다.

관련 내용	(1) 제1.2절 AT91SAM7S256의 기본 구조와 기능 (2) 제3.1절 *OK-7S256* 키트의 구조와 기능 (3) 제3.2절 ARM 개발 툴 및 ISP 다운로드 방법 (4) 제3.3절 ADS v1.2 컴파일러의 설치 및 사용

1. 프로그래밍 실습을 시작하기 전에

이제부터 AT91SAM7S256의 프로그래밍 실습을 시작하기 전에 여러분은 제1장~제3장을 충분히 읽고 그 내용을 최대한 이해하도록 노력하라. 지금으로서는 여러분이 이 부분의 내용들을 모두 이해하는 것은 매우 어렵겠지만, 그렇다고 하더라도 제1장에서 AT91SAM7S256의 기본적인 구조와 기능이나 제3장에서 *OK-7S256* 키트의 개략적인 구조와 이에 필요한 소프트웨어들의 구성 및 설치에 대하여는 특별히 세심한 관심을 가져야만 앞으로의 실습을 진행할 수 있다. 또한, 실제로 *OK-7S256* 키트를 옆에 두고 직접 관찰하면서 눈에 보이는 기본적인 구조를 익혀두는 것이 좋다. 지금 책이나 키트에서 이해가 잘 되지 않는 부분들은 이제부터 점차적으로 한가지씩 익혀나간다는 마음가짐으로 시작하면 된다.

이 분야의 초보자로서 가장 바람직하지 않은 태도중의 하나는 마이크로프로세서의 기본 구조나 프로그래밍 기술의 걸음마도 떼어보지 않고 모터가 무엇인지도 배우기 전에 로봇부터 만들어 보겠다든가 하는 허황된 마음을 갖는 것이다. 목표를 크게 세우고 공부하는 것은 좋으나 기술을 배우는데는 거쳐야할 단계가 있다. 기술의 세계에서 단계를 무시한 발전은 기대하기 어려우며, 과정 없이 결과만을 얻으려 하는 것은 무모하다. 자신의 발로 걸음마도 떼지 못하는 사람은 절대로 올림픽의 100미터 경기에 나갈 수가 없는 것이다. AT91SAM7S256 마이크로콘트롤러를 잘 사용하고 싶으면

적어도 전기/전자회로의 기초에서부터 C언어 프로그래밍의 기초를 먼저 착실하게 익힐 일이다.

이제 여러분은 이 책과 함께 *OK-7S256* 키트를 이용하여 AT91SAM7S256 ARM 마이크로콘트롤러를 익히려는 첫걸음 단계에서 다음과 같은 경귀를 항상 염두에 두기 바란다.

> ① **百聞이 不如一見** : 백번 듣는 것보다 내 눈으로 한번 직접 보는 것이 나으니라.
> ② **百見이 不如一打** : 백번 보는 것보다 내 손으로 한번 직접 키보드를 쳐보는 것이 나으니라.
> ③ **百打가 不如一作** : 백번 키보드를 치는 것보다 내 머리로 한번 직접 설계하거나 프로그램을 작성해 보는 것이 나으니라.

그러나, 처음부터 설계와 프로그래밍이 내 뜻대로 쉽게 되지는 않는다. 남의 것을 보는데서 출발하여 기초를 충분히 닦고 정석(定石)대로 따라하다 보면 스스로 능력과 개성을 발휘할 때가 오는 것이니, **무릇 창조(創造)는 모방(模倣)에서 출발하느니라**. 기초를 닦아야 할 때 행여 재주를 부리려고 나서지 말아야 한다.

마이크로콘트롤러를 배우고는 싶은데 꾸준히 공부를 계속하지 못한다면 그것은 아마도 2가지중의 하나일 것이다. 하나는 공부 방법이 잘못된 경우이다. 재미가 없다면 꾸준히 공부가 될 수 없다. 마이크로콘트롤러가 재미있으려면 책만 보지 말고 **좋은 키트를 가지고 프로그래밍 실습을 병행해야 한다**. 책을 읽어 이해가 안되는 부분은 프로그래밍 실습으로 보충하고 프로그램에서 생기는 의문은 책에서 찾아내야 한다.

또 하나는 마음이 나약하여 쉽게 지치고 포기하는 경우다. 그러나, 만약 여러분의 마음이 독하지 못하여 **작심삼일(作心三日)이면 3일마다 작심(作心)하라**. 프로그램에서 내가 그 의미를 모르고는 단 1개의 명령이라도 사용하지 마라. 남의 프로그램을 읽을 때 단 한 줄이라도 이해가 안되는 부분이 있으면 그 다음 줄로 넘어가지 마라. 오늘 밤 내가 공부하다가 **모르면 잠을 자지 마라**.

2. C언어 프로그램의 작성 및 실행 방법

ADS v1.2 C컴파일러를 사용하여 C언어 프로그램을 작성하는 방법은 제3.3절에서 이미 설명하였다. 그러나, 여기서는 이제부터 많은 예제 프로그램을 사용해야 하므로

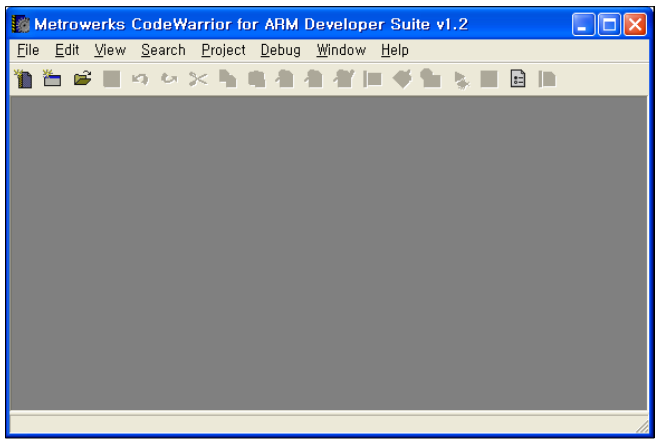
좀더 편리하게 수행할 수 있는 프로그래밍 방법을 요약하여 다시 한번 설명한다.

이제부터 C언어 프로그램을 작성하려면 제3.3절의 설명에 따라서 ADS v1.2 C컴파일러와 이 책에서 함께 제공하는 C언어 예제 프로그램 OK7S256ads.zip이 여러분의 퍼스널컴퓨터에 설치되어 있어야 하며, 올바른 동작이 확인된 *OK-7S256* 키트와 ISP 다운로드 케이블을 가지고 있어야 한다. 다운로드 케이블로는 USB A-B형 케이블 또는 RS-232C 직렬통신 케이블 중에서 어느 것을 사용해도 좋다.

(1) 프로젝트에서 예제 프로그램 소스 파일을 교체 등록

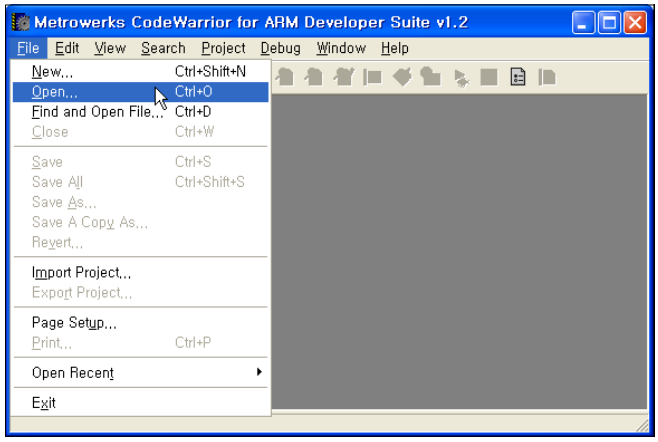
일반적으로 하나의 사용자 프로그램은 하나의 프로젝트로 만들어지며, 따라서 사용자 프로그램이 달라지면 프로젝트로 다른 것을 사용하게 된다. 그러나, 이제부터 이 책의 모든 ADS v1.2용 C언어 예제 프로그램에서는 프로젝트명을 Xads.mcp로 통일하여 사용하기로 한다. 그러므로, 예제 프로그램이 달라지면 이 프로젝트에 등록되어 있는 소스 파일도 해당 예제 프로그램의 소스 파일로 교체하여야 한다.

① 바탕화면에서 ADS v1.2의 바로가기 아이콘을 누르면 <그림 4.1.1>과 같이 IDE 소프트웨어인 CodeWarrior의 화면이 나타난다.

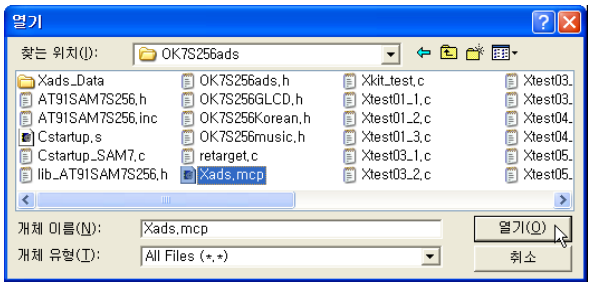


<그림 4.1.1> ADS v1.2를 실행하였을 때의 첫 화면

② 프로젝트 파일 Xads.mcp를 불러들이려면 <그림 4.1.2>처럼 File → Open... 메뉴를 선택하고, <그림 4.1.3>과 같이 파일 선택창이 나타나면 원하는 프로젝트 파일을 지정하고 열기(O) 버튼을 누른다.

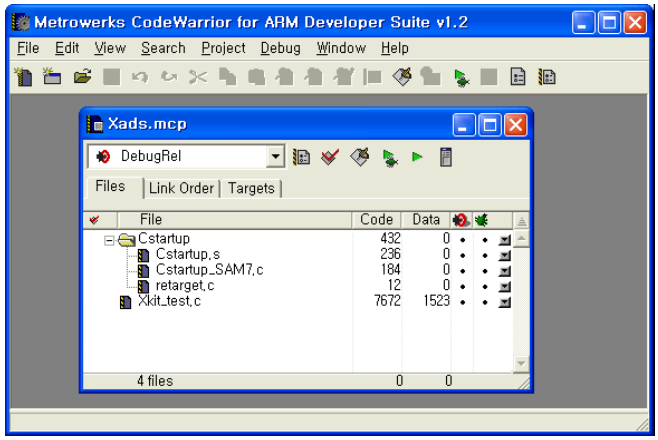


<그림 4.1.2> 기존의 프로젝트를 불러오기 위한 화면



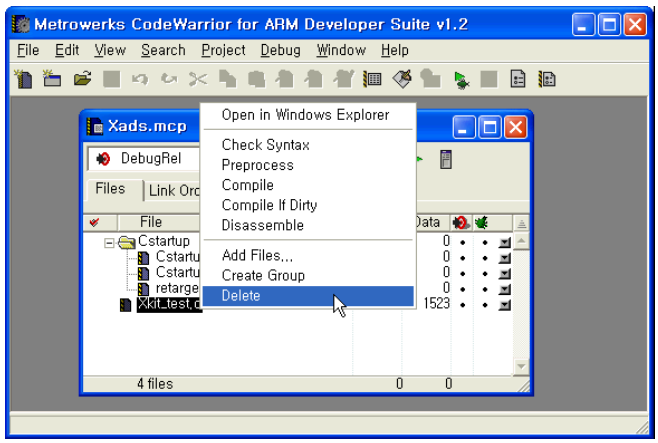
<그림 4.1.3> 기존의 프로젝트 파일 Xads.mcp를 불러오기

③ 이와 같이 프로젝트 Xads.mcp를 읽어들이면 화면에 <그림 4.1.4>와 같은 프로젝트창이 표시된다.

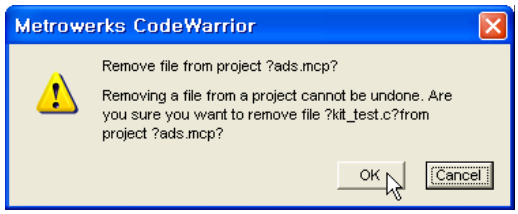


<그림 4.1.4> 프로젝트 파일 Xads.mcp를 불러온 화면

④ 이제 이 프로젝트명 Xads.mcp를 그대로 유지하면서 사용자 예제 프로그램을 Xkit_test.c에서 Xtest01_1.c로 교체하여 등록해 보기로 한다. 먼저 현재의 사용자 예제 프로그램 소스 파일 Xkit_test.c를 프로젝트에서 제거(등록 해제)하려면 <그림 4.1.5>처럼 이 파일명 위에 마우스를 놓고 오른쪽 버튼을 눌러서 Delete 메뉴를 선택하면 되는데, 이때 <그림 4.1.6>처럼 제거할 것인지를 확인하는 화면이 나타나면 OK 버튼을 누른다.

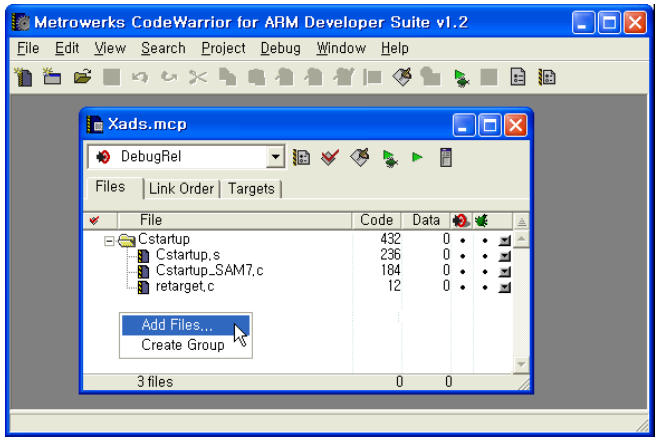


<그림 4.1.5> 현재의 프로젝트에서 예제 프로그램 소스 파일을 제거하기 위한 화면

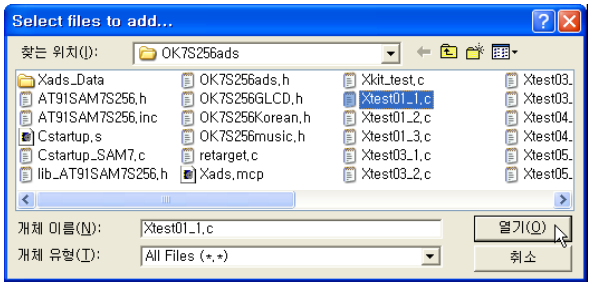


<그림 4.1.6> 현재의 프로젝트에서 예제 프로그램 소스 파일을 제거하는 것을 확인

⑤ 나음에는 이 프로젝트에 사용자 예제 프로그램의 소스 파일을 새로 등록하여야 한다. 이를 위해서 <그림 4.1.7>처럼 프로젝트 창에서 마우스의 오른쪽 버튼을 누르면 팝업창이 뜨는데 여기서 Add Files... 메뉴를 누른다. 그러면 <그림 4.1.8>처럼 파일을 선택하는 화면이 나오는데 여기서 원하는 C언어 소스파일 Xtest01_1.c를 선택하고 열기(O) 버튼을 누른다.

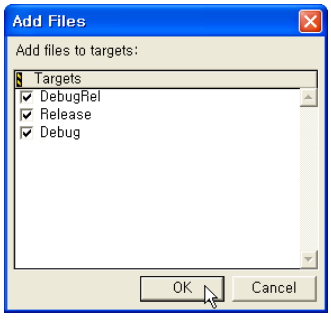


<그림 4.1.7> 현재의 프로젝트에 예제 프로그램의 소스 파일을 등록하기 위한 화면

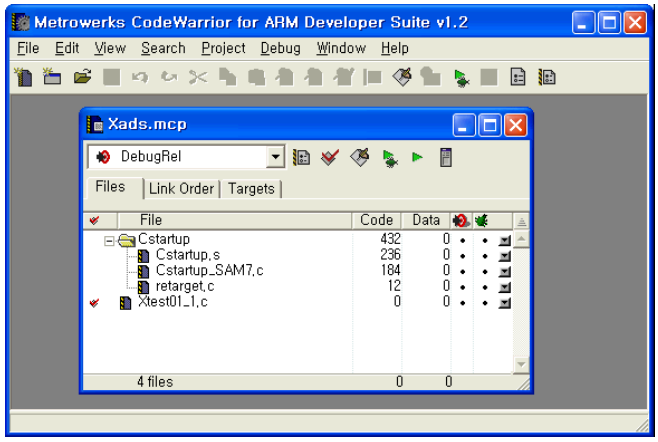


<그림 4.1.8> 현재의 프로젝트에 등록할 예제 프로그램의 소스 파일을 선택

⑥ 그러면 <그림 4.1.9>처럼 다시 타겟 폴더를 지정하는 팝업창을 거쳐서 <그림 4.1.10>처럼 새로운 사용자 예제 파일이 현재의 프로젝트에 등록 완료된 화면이 나타난다.



<그림 4.1.9> 프로젝트에 등록할 예제 프로그램 소스 파일의 타겟 폴더를 지정

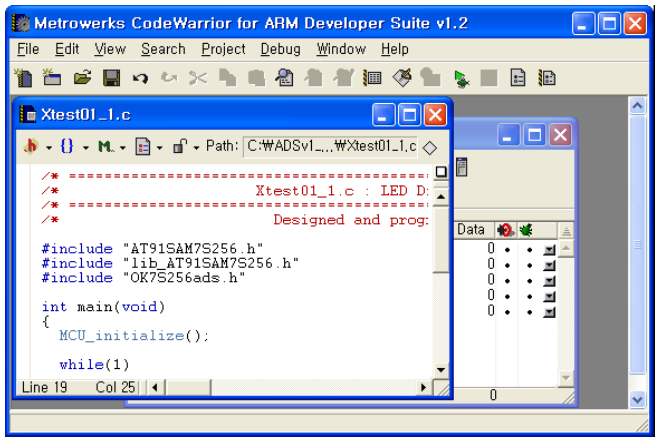


<그림 4.1.10> 현재의 프로젝트에 새로운 예제 프로그램 소스 파일이 등록된 화면

(2) 예제 프로그램을 편집하거나 컴파일

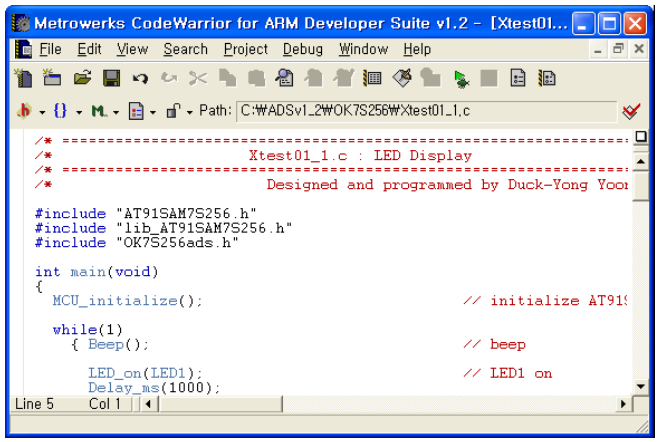
이렇게 하여 현재의 프로젝트명 Xads.mcp로 새로운 사용자 예제 프로그램 소스 파일 Xtest01_1.c를 불러들여서 새로 등록하였으면 필요할 경우 이를 편집하거나 컴파일하게 된다.

① 사용자 프로그램의 소스 파일의 내용을 편집하려면 프로젝트창에 있는 사용자 프로그램 소스 파일명 Xtest01_1.c를 클릭하면 되는데, 이렇게 하면 소스 파일의 내용이 편집창에 <그림 4.1.11>과 같이 나타난다.



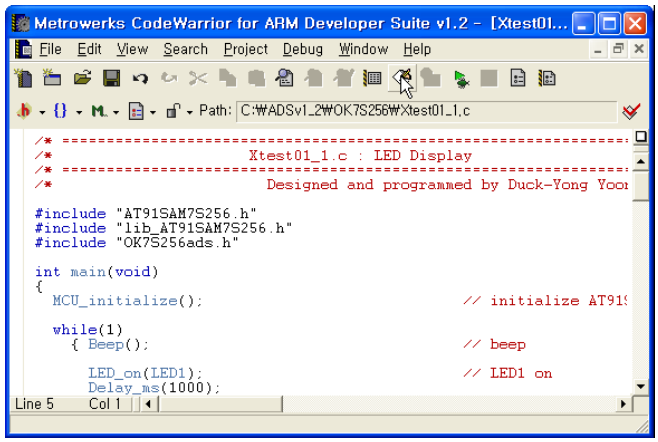
<그림 4.1.11> 소스 파일을 편집하기 위한 화면

② 여기서 만약 필요하다면 편집창의 화면 확대 버튼을 눌러서 <그림 4.1.12>처럼 편집창을 크게 할 수 있다.

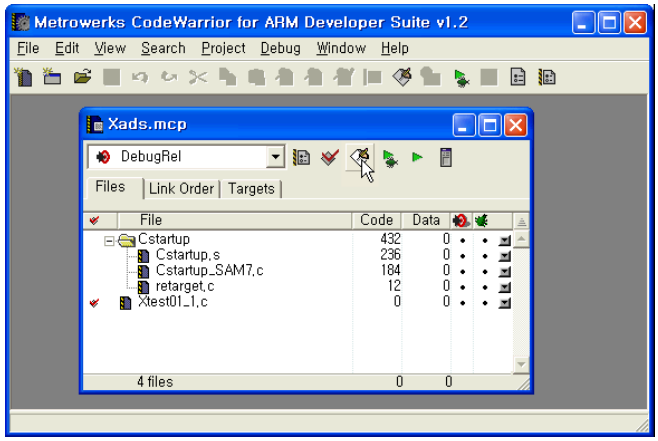


<그림 4.1.12> 편집창을 최대화한 화면

③ 이제 소스 파일의 편집이 끝나고 컴파일 및 링크를 수행하려면 <그림 4.1.13>과 같이 메인 화면에 있는 Make 아이콘을 누르거나 <그림 4.1.14>와 같이 프로젝트 창에서 Make 아이콘을 누른다. 또는 메뉴에서 Project → Make 메뉴를 선택하거나 단축키 F7을 눌러도 된다.



<그림 4.1.13> 메인 화면에서 Make 아이콘을 눌러 프로그램을 컴파일하는 방법



<그림 4.1.14> 프로젝트 창에서 Make 아이콘을 눌러 프로그램을 컴파일하는 방법

④ 이렇게 컴파일을 수행하면 아무 에러가 없을 경우 그 결과가 팝업창으로 표시되면서 C:\ADSv1_2\OK7S256ads\Xads_Data\DebugRel\ 폴더에 사용자 프로그램의 실행 파일이 Xads.bin으로 생성된다. 그러나, 에러가 있을 경우에는 팝업창에 에러의 내용이 표시되므로 해당 부분을 찾아서 수정하고 다시 컴파일해야 한다.

◀참고▶ 소스를 입력할 때는 모든 내용을 일일이 키보드에서 입력하지 말고 필요하다면 기존의 파일에서 필요한 부분을 복사해 오는 것도 좋은 방법이다. 이를 위해서는 현재의 프로젝트에 여러개의 소스 파일을 등록해 놓고 그 파일들을 찾아다니면서 필요한 부분을 블록으로 지정하여 복사해 오는 것이 좋다. 이러한 편집작업이 완료되고 나면 이제 필요 없는 파일들은 프로젝트에서 등록을 해제하여 제거하면 된다.

또한 기존의 파일과 상당히 유사한 파일을 신규로 작성할 때는 이를 새로 만들기 보다 Windows 운영체제에서 그 파일을 원하는 새로운 이름으로 C:\ADSv1_2\OK7S256ads\ 폴더에 복사하여 놓은 다음에 이를 현재의 프로젝트에 교체 등록하고 편집하여 수정하면 편리하다.

(3) 사용자 프로그램을 *OK-7S256* 키트에 다운로드하여 실행

이와 같이 Xads.mcp 프로젝트에서 사용자 프로그램을 작성하고 컴파일하였으면 이제는 이를 *OK-7S256* 키트에 다운로드하여 실행한다. 이렇게 다운로드할 바이너리 실행 파일은 항상 C:\ADSv1_2\OK7S256ads\Xads_Data\DebugRel\Xads.bin 파일이다.

OK-7S256 키트에서 다운로드 케이블로서 만약 USB A-B형 케이블을 사용한다면 이는 키트의 콘넥터 CN4에 접속하며, RS-232C 직렬통신 케이블을 사용한다면 이는 키트의 콘넥터 CN5에 접속한다. *OK-7S256* 키트에 바이너리 실행 파일을 다운로드하는 방법에 대하여는 제3.2절을 참조하라.

3. *OK-7S256* 키트를 위한 사용자 헤더 파일

▣ LED 점등 예제

앞에서 소개한 예제 프로그램 Xtest01_1.c는 LED1과 LED2를 교대로 점등하는 프로그램으로서 그 내용은 다음과 같다.

<프로그램 Xtest01_1.c> LED를 점등하는 예제(1)

```
/* ====================================================================== */
/*                        Xtest01_1.c : LED Display(1)                    */
/* ====================================================================== */
/*                          Designed and programmed by Duck-Yong Yoon in 2007.  */

#include "AT91SAM7S256.h"
#include "lib_AT91SAM7S256.h"
#include "OK7S256ads.h"

int main(void)
{
  MCU_initialize();                          // initialize AT91SAM7S256 & kit

  while(1)
    { Beep();                                // beep

      LED_on(LED1);                          // LED1 on
      Delay_ms(1000);
      LED_off(LED1);                         // LED1 off
      Delay_ms(1000);

      LED_on(LED2);                          // LED2 on
      Delay_ms(1000);
      LED_off(LED2);                         // LED2 off
      Delay_ms(1000);

      LED_on(LED2|LED1);                     // LED2 and LED1 on
      Delay_ms(1000);
      LED_off(LED2|LED1);                    // LED2 and LED1 off
      Delay_ms(1000);
    }
}
```

☞ 이 프로그램에 LED1이 켜지거나 꺼질 때 스피커에서 잡음이 발생하는 것은 병렬 I/O 포트 PA0의 출력을 LED1과 스피커가 공유하기 때문이다. 이로 인하여 LED1이 켜지는 순간에 스피커에 인가되는 전류 필스에 의하여 스피커에서 잡음이 발생하게 되는데, 대부분 이를 무시하면 되지만 귀에 거슬린다면 콘넥터 CN7을 연결하지 말고 뽑아두면 된다.

이를 컴파일하여 *OK-7S256* 키트에 다운로드하고 실행하여 보라. 프로그램의 흐름을 보면서 LED의 동작이 올바른 것을 확인할 수 있겠는가?

이제 예제 프로그램 Xtest01_1.c를 편집창에서 아래와 같이 수정하여 컴파일하고, 이를 *OK-7S256* 키트에 다운로드하여 실행시켜 보라. 그리고, LED1과 LED2의 점등 상태가 어떻게 달라지는지 Xtest01_1.c 프로그램과 비교하여 보라.

<프로그램 Xtest01_2.c> LED를 점등하는 예제(2)

```
/* ====================================================================== */
/*                        Xtest01_2.c : LED Display(2)                    */
/* ====================================================================== */
/*                          Designed and programmed by Duck-Yong Yoon in 2007.  */

#include "AT91SAM7S256.h"
#include "lib_AT91SAM7S256.h"
#include "OK7S256ads.h"

int main(void)
{
  MCU_initialize();                          // initialize AT91SAM7S256 & kit

  while(1)
    { LED_on(LED1);                          // LED1 on
      Delay_ms(100);
      LED_off(LED1);                         // LED1 off
      Delay_ms(100);

      LED_on(LED2);                          // LED2 on
      Delay_ms(100);
      LED_off(LED2);                         // LED2 off
      Delay_ms(100);

      LED_on(LED2|LED1);                     // LED2 and LED1 on
      Delay_ms(100);
      LED_off(LED2|LED1);                    // LED2 and LED1 off
      Delay_ms(100);
    }
}
```

▣ 사용자 헤더파일 OK7S256ads.h

앞으로 ADS v1.2를 사용하여 C언어 예제 프로그램을 작성할 때는 이와 같이 항상 AT91SAM7S256의 I/O 레지스터 정의용 헤더파일 AT91SAM7S256.h와 이 소자를 편리하게 사용할 수 있도록 만든 라이브러리 함수용 헤더파일 lib_AT91SAM7S256.h를 인클루드한다. 이것들은 Atmel사에서 제공하는 것인데 제3.3절에서 자세하게 설명한 바 있다.

그러나, *OK-7S256* 키드의 하드웨어에 맞도록 AT91SAM7S256을 초기화하거나 또는 C언어 프로그램을 작성할 때 자주 사용하는 사용자 함수들은 독립적인 헤더파일로 만들어 두는 것이 좋은데, 이 책에서는 이를 OK7S256ads.h로 제공하며 그 내용은

다음과 같다.

나중에 보듯이 *OK-7S256* 키트용의 사용자 헤더파일은 이밖에도 그래픽 LCD 모듈과 관련된 OK7S256GLCD.h, 그래픽 LCD 모듈에 한글 문자를 출력하는 함수를 제공하는 OK7S256Korean.h, 그리고 스피커를 통하여 음악 멜로디를 연주하는데 관련된 OK7S256music.h 등이 있다.

<헤더파일 OK7S256ads.h> *OK-7S256* 키트를 위한 기본적인 정의 및 사용자 함수

```
/* ------------------------------------------------------------------------------ */
/*    OK7S256ads.h : Definition & User Function for OK-7S256 V1.0 Training Kit    */
/* ------------------------------------------------------------------------------ */
#define true            -1                      // true  = 0xFFFFFFFF
#define false           0                       // false = 0x00000000

#define EXT_OC          18432000                // external oscilator MAINCK
#define MCK             48000000                // master clock MCK
#define MCKKHz          (MCK/1000)              // MCK/1000

#define PA0             (1 << 0)                // define bit number of PIO
#define PA1             (1 << 1)
#define PA2             (1 << 2)
#define PA7             (1 << 7)
#define PA8             (1 << 8)
#define PA15            (1 << 15)
#define PA17            (1 << 17)
#define PA18            (1 << 18)
#define PA19            (1 << 19)
#define PA20            (1 << 20)
#define PA21            (1 << 21)
#define PA22            (1 << 22)
#define PA23            (1 << 23)

#define LED1            PA0                     // LED1,2 output signal
#define LED2            PA1

#define SPEAKER         PA0                     // speaker
#define BUZZER          PA2                     // buzzer output signal

#define KEY1            PA7                     // KEY1,2 input signal
#define KEY2            PA8

#define CS1             PA15                    // SPI CS1 output signal

#define J3              PA17                    // J3 input signal
#define PCK1            PA17                    // PCK1 output signal

#define LCD_RS          PA18                    // LCD output signal
#define LCD_E           PA19
#define LCD_ALL         (0xFF000000|LCD_E|LCD_RS)

#define GLCD_DI         PA20                    // GLCD output signal
#define GLCD_E          PA21
#define GLCD_CS1        PA22
#define GLCD_CS2        PA23
#define GLCD_ALL        (0xFF000000|GLCD_CS2|GLCD_CS1|GLCD_E|GLCD_DI)

void Delay_1us(void)                            /* 1 us delay for -O2 -Otime option */
{ volatile unsigned int count;

  for(count = 0; count < 4; count++);
}
```

```
void Delay_us(unsigned int us)                  /* us delay for -O2 -Otime option */
{ volatile unsigned int count, countmax = (MCK / 9600000) * us;

  for(count = 0; count < countmax; count++);
}

void Delay_ms(unsigned int ms)                  /* ms delay for -O2 -Otime option */
{ volatile unsigned int count, countmax = (MCK / 10000) * ms;

  for(count = 0; count < countmax; count++);
}

void LED_on(unsigned int led)                   /* LED on */
{
  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LED2|LED1); // PA1-PA0 = output
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, led);
}

void LED_off(unsigned int led)                  /* LED off */
{
  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LED2|LED1); // PA1-PA0 = output
  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, led);
}

void Buzzer_on(void)                            /* buzzer on */
{
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, BUZZER);
}

void Buzzer_off(void)                           /* buzzer off */
{
  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, BUZZER);
}

void Beep(void)                                 /* beep for 50 ms */
{
  Buzzer_on();
  Delay_ms(50);
  Buzzer_off();
}

void Error(void)                                /* beep 3 times */
{
  Beep();
  Delay_ms(50);
  Beep();
  Delay_ms(50);
  Beep();
}

__inline void AT91F_PIO_Opendrain(              /* configure open drain */
        AT91PS_PIO pPio, unsigned int multiDrvEnable)
{
//      pPio->PIO_MDDR = ~multiDrvEnable;
        pPio->PIO_MDER = multiDrvEnable;
}

__inline void AT91F_PIO_Pullup(                 /* configure pull up */
        AT91PS_PIO pPio, unsigned int pullupEnable)
{
//      pPio->PIO_PPUDR = ~pullupEnable;
        pPio->PIO_PPUER = pullupEnable;
}

void MCU_initialize(void)                       /* initialize AT91SAM7S256 & OK-7S256 kit */
{
  AT91F_RSTSetMode(AT91C_BASE_RSTC, AT91C_RSTC_URSTEN); // enable User Reset by NRST pin input

  AT91F_PIOA_CfgPMC();                                  // enable clock of PIO

  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LED2|LED1);      // PA1-PA0 = output
```

```
  LED_off(LED2|LED1);                                   // LED2-LED1 off

  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, BUZZER);         // PA2 = output
  Buzzer_off();                                         // buzzer off

  AT91F_PIO_CfgInput(AT91C_BASE_PIOA, J3|KEY2|KEY1);    // PA17,PA8,PA7 = input with pullup
  AT91F_PIO_Pullup(AT91C_BASE_PIOA, J3|KEY2|KEY1);

  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LCD_ALL);        // open drain output for LCD
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LCD_ALL);
  AT91F_PIO_Opendrain(AT91C_BASE_PIOA, LCD_ALL);

  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, GLCD_ALL);       // open drain output for GLCD
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, GLCD_ALL);
  AT91F_PIO_Opendrain(AT91C_BASE_PIOA, GLCD_ALL);
}

void LCD_command(unsigned int command)          /* write a command(instruction) on text LCD */
{
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LCD_RS);       // RS = 0
  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, command << 24);  // output command
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, (~command) << 24);

  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, LCD_E);          // E = 1
  Delay_1us();
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LCD_E);        // E = 0

  if(command & 0xFC)                                    // wait for operation
    Delay_us(50);
  else
    Delay_ms(2);
}

void LCD_data(unsigned int data)                /* display a character on text LCD */
{
  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, LCD_RS);         // RS = 1
  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, data << 24);     // output data
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, (~data) << 24);

  AT91F_PIO_SetOutput(AT91C_BASE_PIOA, LCD_E);          // E = 1
  Delay_1us();
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LCD_E);        // E = 0

  Delay_us(50);                                         // wait for operation
}

void LCD_initialize(void)                       /* initialize text LCD module */
{
  AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LCD_ALL); // open drain output
  AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LCD_ALL);
  AT91F_PIO_Opendrain(AT91C_BASE_PIOA, LCD_ALL);
  Delay_ms(2);

  LCD_command(0x38);                            // function set(8 bit, 2 line, 5x7 dot)
  LCD_command(0x0C);                            // display control(display on, cursor off)
  LCD_command(0x06);                            // entry mode set(increment, not shift)
  LCD_command(0x01);                            // clear display
}

void LCD_string(unsigned int command, char *string) /* display a string on text LCD */
{
  LCD_command(command);                         // cursor position

  while(*string != '\0')                        // display string
    { LCD_data(*string);
      string++;
    }
}

unsigned char key_flag = 0;
```

```
unsigned int Key_in(void)                          /* input key KEY2 - KEY1 without debouncing */
{ unsigned int keycode;                            //   (no input = 0, KEY1 = 1, KEY2 = 2)

  AT91F_PIO_CfgInput(AT91C_BASE_PIOA, KEY2|KEY1); // PA8,PA7 = input with pullup
  AT91F_PIO_Pullup(AT91C_BASE_PIOA, KEY2|KEY1);

  keycode = (~(AT91F_PIO_GetInput(AT91C_BASE_PIOA) >> 7)) & 0x00000003;

  return keycode;
}

unsigned int Key_input(void)                       /* input key KEY2 - KEY1 with debouncing */
{ unsigned int key;                                //   (no input = 0, KEY1 = 1, KEY2 = 2)

  key = Key_in();                                  // any key pressed ?
  if(key == 0)                                     // if no key, check key off
    { if(key_flag == 0)
        return key;
      else
        { Delay_ms(20);
          key_flag = 0;
          return key;
        }
    }
  else                                             // if key input, check continuous key
    { if(key_flag != 0)                            // if continuous key, treat as no key input
        return 0;
      else                                         // if new key, beep and delay for debounce
        { Beep();
          key_flag = 1;
          return key;
        }
    }
}
```

☞ ADS v1.2용의 이 사용자 헤더파일 OK7S256ads.h는 뒤의 제4.2절에서 설명하는 WinARM용의 사용자 헤더파일 OK7S256gcc.h와 시간지연 함수 Delay_1us(), Delay_us() 및 Delay_ms()를 제외하면 내용이 서로 완전히 같다. ADS v1.2와 WinARM에서 프로그램을 작성할 때 소프트웨어에 의한 시간지연 루틴은 서로 상당히 다르기 때문에 이와 같이 서로 다른 헤더파일을 사용하게 되었다.

여기서, AT91F_PIO_Opendrain()과 AT91F_PIO_Pullup() 함수는 Atmel사의 lib_AT91SAM7S256.h에 제공되는 함수 AT91F_PIO_CfgOpendrain()과 AT91F_PIO_CfgPullup()을 각각 수정한 것이다. lib_AT91SAM7S256.h에 제공되는 원래의 함수들은 기존에 설정되어 있는 다른 비트들을 해제시켜 버리는 기능을 포함하고 있어서 매우 불편하므로 이를 삭제하는 것으로 수정하였다.

이 헤더파일 OK7S256ads.h에 포함되어 있는 사용자 정의 함수들을 모두 요약 정리하면 <표 4.1.1>과 같다. 이 사용자 정의 함수들은 앞으로 여러분이 *OK-7S256* 키트에서 C언어로 프로그램을 작성할 때 유용하게 사용할 것이므로 잘 알아두어야 하며, 가급적이면 이 함수들의 내용까지 정확히 이해하도록 노력할 것을 권한다. 특히 여기서 LCD_string(), Key_input()이나 시간지연 함수 등은 *OK-7S256* 키트의 하드웨어의 동작과 연계하면서 그 동작원리를 정확히 이해하는 것이 좋다. 현재는 비록 그 내용을 이해하기 어렵겠지만 앞으로 AT91SAM7S256의 공부가 진행될수록 이를 충분히

알 수 있게 되므로 염려하지 않아도 된다.

C언어에서는 대문자와 소문자를 구분하여 사용한다. C언어 고유의 모든 지시어나 내장함수들은 항상 소문자로 시작한다. 그러나, 이 책의 C언어 예제에서는 모든 사용자 정의 함수를 대문자로 시작하는데 유의하라. 그리고, C언어 프로그램을 작성하는 동안 컴파일러의 기능이나 문법에 관한 문제를 만나면 항상 제3.3절의 설명을 참조하거나 ADS v1.2의 사용자 가이드를 찾아보기 바란다. 그리고, 당연히 **표준적인 C언어의 문법에 대한 사항은 C언어에 관한 책을 보고 별도로 익혀야 한다**. 이 책은 C언어를 사용하여 AT91SAM7S256 마이크로콘트롤러의 프로그래밍 기술을 공부하고자 하는 책이지 C언어를 익히자는 책이 아니다.

<표 4.1.1> 헤더파일 OK7S256ads.h에 포함된 사용자 정의 함수들의 요약

사용자 정의 함수	기 능	입력 파라미터	리턴 값
MCU_initialize()	AT91SAM7S256을 *OK-7S256* 키트의 사양에 맞도록 초기화한다.	-	-
LCD_initialize()	텍스트형 LCD 모듈을 초기화한다.	-	-
LCD_command()	텍스트형 LCD 모듈에 1개의 제어명령을 출력한다.	unsigned int command	-
LCD_data()	텍스트형 LCD 모듈에서 현재의 커서 위치에 1문자를 표시한다.	unsigned int data	-
LCD_string()	텍스트형 LCD 모듈에서 커서 위치를 지정하고 그 위치부터 문자열을 출력한다.	unsigned int command char *string	-
Delay_1us() Delay_us() Delay_ms()	1㎲의 시간 지연을 수행한다. 1~65535㎲ 범위의 시간 지연을 수행한다. 1~65535ms 범위의 시간 지연을 수행한다.	- unsigned int us unsigned int ms	-
Key_input()	KEY1~KEY2를 통한 키입력을 읽어온다. 키를 입력할 때 디바운싱 기능을 수행한다.	-	unsigned int
Beep() Error() Buzzer_on() Buzzer_off()	버저를 50ms 동안 1차례 울린다. 버저를 50ms 시간 간격으로 3번 울린다. 버저를 켠다. 버저를 끈다.	-	-
LED_on() LED_off()	LED1~LED2를 지정하여 점등한다. LED1~LED2를 지정하여 소등한다.	unsigned int led unsigned int led	-
AT91F_PIO_OpenDrain() AT91F_PIO_Pullup()	PA0~PA31을 지정하여 오픈 드레인형 출력으로 지정한다. PA0~PA31을 지정하여 내부 풀업저항을 사용하도록 지정한다.	AT91PS_PIO pPio unsigned int	-

▣ 텍스트 LCD 모듈 디스플레이 예제

이번에는 텍스트 LCD 모듈에 메시지를 디스플레이하는 예제 프로그램을 실행시켜 보기로 한다. 아래의 예제 프로그램을 현재의 프로젝트 Xads.mcp에 교체 등록하고 컴파일하여 *OK-7S256* 키트에 다운로드하고 실행하여 보라.

<프로그램 Xtest01_3.c> 텍스트 LCD 모듈 디스플레이 예제

```
/* ========================================================================== */
/*                    Xtest01_3.c : Text LCD Display                          */
/* ========================================================================== */
/*                          Designed and programmed by Duck-Yong Yoon in 2007.  */

#include "AT91SAM7S256.h"
#include "lib_AT91SAM7S256.h"
#include "OK7S256ads.h"

int main(void)
{
  MCU_initialize();                             // initialize AT91SAM7S256 & kit
  Delay_ms(50);                                 // wait for system stabilization
  LCD_initialize();                             // initialize text LCD

  while(1)
    { Beep();                                   // beep

      LCD_string(0x80, " OK-7S256  V1.0 ");     // logo title 1
      LCD_string(0xC0, "AT91SAM7S256 ARM");
      Delay_ms(2000);

      LCD_string(0xC0, "   2007/06/01   ");     // logo title 2
      Delay_ms(2000);

      LCD_string(0xC0, " Duck-Yong Yoon ");     // logo title 3
      Delay_ms(2000);
    }
}
```

☞ 아직은 여러분이 이 프로그램의 내용을 이해하려고 애쓸 필요가 없다. 여기서는 프로그램의 작성, 컴파일, 다운로드, 실행의 절차를 확실히 익히는 것으로 충분하다.

4. 실습 과제

(1) C언어에서 #include < >와 #include " " 지시어는 어떻게 다른가?

(2) C언어에서 /* ~ */ 와 // 형태의 주석문은 어떻게 다른가?

(3) C언어에서 while 문 및 do ~ while 문의 기능과 사용 방법을 설명하라.