MPLAB® C18
컴파일러 시작하기

목차

서문

이 문서의 개요

이 문서는 마이크로칩사의 MPLAB C18 컴파일러의 개요, 설치 및 활용에 대해서 빠르게 학습할 수 있도록 안내해 준다. MPLAB C18에 대한 더욱 상세한 사항은 MPLAB C18 사용자 매뉴얼을 참고하면 된다. 이 문서에는 MPLAB C18의 특징뿐 아니라, MPLAB C18의 설치/제거, 예제 코드 및 MPLAB IDE 6.xx와 함께 사용하는 방법 등이 들어있다.

이 문서의 구성

1장 : MPLAB C18 컴파일러 개요– 컴파일러의 개요, 특징 및 설치 구성요소 등이 설명되어 있다.

2장 : MPLAB C18 컴파일러의 설치– 정식버전 및 데모버전의 설명, 컴파일러의 설치 및 제거 등이 설명되어 있다.

3장 : 예제 사용하기- MPLAB C18 설치할 때, 제공되는 예제 프로그램을 사용하는 방법을 설명하였다.

4장 : 부록- 한국마이크로칩 및 대리점 연락처, 영문 용어집

이 문서의 서체 사용

서체	내용	예
소스코드(Courier체)		
일반 문자	소스코드, 파일명 또는 경로	`#define START, c:\mcc18`
[]괄호: []	옵션 변수	MPASMWIN[main.asm]
{}괄호 및 |문자:{|}	상호 배타적인 변수의 선택, OR 선택	`errorlevel{0|1}`
이탤릭체 문자	변수 값 또는 커맨드 라인 옵션 값	`char isascii(char ch);`
생략부호: …	더 이상의 추가적인 문자들이 있을 수 있음을 의미	`list[list_optioin…,list_option]`
문서(바탕체)		
밑줄 쳐진 이탤릭 Arial 문자	메뉴바에서 메뉴 선택	
굵은 Arial 문자	윈도 창에서 클릭할 버튼	**OK, Cancel**
<>괄호 안의 Arial 문자	키보드의 키	<Tab>,<Ctrl-C>
이탤릭 문자	마이크로칩의 다른 매뉴얼	*MPLAB IDE User's Guide*

추천 문서 목록

MPLAB C18 컴파일러를 MPLAB IDE와 같이 사용하거나, 라이브러리 활용 및 링커 같이 다른 언어 툴과 사용하기 위해선, 아래에서 추천하는 문서들을 읽을 필요가 있다.

README.C18

README.XXX

MPLAB C18 C Complier User's Guide(DS51288)

MPLAB C18 C Compiler Library(DS51297)

MPLAB IDE User's Guide(DS51025)

MPASM User's Guide with MPLINK and MPLIB(DS33014)

MPLAB ICD2 Quick Start Guide

상기 문서들은 마이크로칩 웹사이트(www.microchip.com)에서 영문판을 다운로딩 받을 수 있다. 또한 이용 가능한 한글 문서는 한국마이크로칩 웹사이트(www.microchipkorea.com)에서 다운로딩 받을 수 있다.

기술문의

본 문서 및 마이크로칩 디바이스에 대한 기술문의는 한국 마이크로칩 및 각 대리점에 문의할 수 있다. 한국마이크로칩 및 각 대리점의 연락처는 본 문서 맨 뒤의 연락처를 참고한다.

1장 MPLAB C18 컴파일러 개요

1.1 개요

MPLAB C18은 마이크로칩의 고성능 8비트 MCU인 PIC18 시리즈용 C 컴파일러이다. C18은 PIC18 MCU의 효율성을 높이기 위해 일부 ANSI 표준과 다르게 설계된 부분을 제외하고 ANSI 표준 X3.159-1989에 따라 개발되었다. C18은 마이크로소프트의 윈도 OS 상에서 동작하며, 마이크로칩사의 MCU 통합개발환경인 MPLAB IDE 상에서 동작할 수 있다.

1.2 특징

MPLAB C18 컴파일러의 특징은 다음과 같다.

- ANSI '89 표준과 호환
- MPLAB IDE 상에서 동작이 가능하여 소스 레벨 디버깅(Source-level debugging) 및 프로젝트 기능을 활용할 수 있음
- 코드 재사용을 위해 재배치가 가능한 오브젝트 코드 생성기능
- PICmicro MCU의 어샘블러인 MPASM과 오브젝트 코드가 호환되어, C 언어와 어셈블리 언어를 섞어서 사용할 수 있음
- 외부 메모리 쓰기/읽기 기능
- 인라인 어셈블리 기능
- 다단계의 코드 최적화 기능
- 다양한 라이브러리 함수지원- 주변기기(peripherals), 배열 및 수학함수 등
- 사용자 정의형 데이터 및 코드 배치 기능

1.3 MPLAB C18 다운로딩 및 설치 환경

MPLAB C18은 유료인 정식버전과 무료로 다운로딩 받을 수 있는 데모버전이 있다. 데모버전은 마이크로칩 홈페이지에서 다운로딩 받을 수 있다.

http://www.microchip.com/1010/pline/tools/picmicro/code/mplab18/index.htm

데모버전(MPLAB-C18-vx.xx-demo-win32.exe)은 30일간의 사용기간 제약이 있다. 데모버전 파일명에서 vx.xx는 버전(version)을 말한다. 현재는 2.20a 버전이 최신이다. MPLAB C18은 유료이므로 정식버전(MPLAB-C18-vx.xx-win32.exe)을 얻기 위해서는 한국마이크로칩이나 대리점에 문의하면 된다. 한국마이크로칩 및 대리점의 연락처는 부록B를 참조한다. MPLAB C18을 사용하기 위해서는 어셈블러와 링커인 MPASM 및 MPLINK가 있어야 한다. MPASM과 MPLINK는 PICmicro 통합개발환경인 MPLAB IDE에 포함되어 있다. 따라서 MPLAB IDE를 다운로딩 받아 PC에 설치하면 MPASM과 MPLink는 자동적으로 설치된다. MPLAB IDE의 다운로딩 및 설치에 대해서는 마이크로칩 홈페이지에서 *MPLAB IDE User's Guide(DS51025)* 매뉴얼을 참고하면 된다. MPLAB C18을 설치하기 위해서는 윈도95 이상

의 OS와 25MB 이상의 하드 디스크 공간이 필요하다.

1.4 설치 구성요소

MPLAB C18을 PC에 설치하면 다음과 같은 디렉터리 구조가 생긴다. 각각의 디렉터리의 역할은 아래의 표1-1과 같다.

표1-1: MPLAB C18 서브디렉터리 구조

디렉토리	개요
bin	컴파일러, 링커 및 여러 실행파일을 가지고 있다
cpp	C18 컴파일러에서 사용되는 프리프로세서들의 소스코드를 가지고 있다. 이 소스코드는 참고용으로 제공된다.
doc	C18 컴파일러의 온라인 영문 매뉴얼을 가지고 있다. 컴파일러를 사용하기 위해서는 여기의 문서를 참고한다.
example	MPLAB IDE에서 MPLAB C18을 사용하여 프로젝트를 활용, 프로그램을 만드는 예제를 가지고 있다.
h	표준 C 라이브러리, MCU 라이브러리의 헤더파일을 가지고 있다.
lib	표준 C 라이브러리(clib.lib), MCU 라이브러리(p18xxxx.lib) 및 스타트업 코드(c018.o,c018i.o,c018iz.o)를 가지고 있다
lkr	MPLINK가 사용하는 링커 스크립트(linker script)를 가지고 있다.
mpasm	어샘블러인 MPASM(코맨드 라인 버전), 어샘블러용 디바이스 헤더파일(p18xxxx.inc) 및 어셈블리 헤더파일을 가지고 있다.
src	C 및 어셈블러로 구성된 각종 라이브러리의 소스코드가 들어있다. 포함된 라이브러리는 표준 C 라이브러리, MCU 라이브러리, 스타트업 코드 등이다.

1.5 컴파일러 실행파일

MPLAB C18은 여러 가지 중요한 실행파일이 있다. 각각의 실행파일을 역할은 아래의 표1-2와 같다.

표1-2:MPLAB C18 각종 실행파일

실행파일	역할
mcc18.exe c18demo.exe	이 실행파일이 c 컴파일러이다. c 소스 파일(예:*.c)을 입력받아 오브젝트 파일(*.o)을 생성한다. mcc18.exe는 정식버전용 실행파일이며 c18demo.exe는 데모버전용 실행파일이다
cpp18.exe	이 실행파일은 프리프로세서(preprocessor)이다. c 소스파일에 포함되어 있는 프리프로세서들을 해석하여 그 결과를 c 컴파일러에 전달한다.
mplink.exe	이 실행파일은 MPLAM C18 링커를 실행시키는 파일이다. 이 파일은 링커 스크립트 파일(*.lkr), 오브젝트 파일(*.o) 및 라이브러리 파일을 취하여 링커에게 전달하는 역할을 한다. 또한 링커가 입력파일 들로부터 결과물인 coff 파일을 생성하면 이것을 mp2cod.exe 및

	mp2hex.exe로 전달한다.
_mplink.exe	이 실행파일은 링커이다. 링커는 링커 스트립트, 오브젝트 파일 및 라이브러리 파일을 입력 받아 coff 파일(예:*.out 또는 *.cof)을 출력한다. 또한 링커는 선택적으로 map 파일을 생성할 수 있다.
mp2cod.exe	이 실행파일은 coff 파일을 cod 파일로 변환한다. cod 파일은 MPLAB IDE에서 사용되는 심볼릭 디버깅 파일포맷 이다. mp2cod.exe는 _mplink.exe의 출력인 coff 파일을 입력받아 cod 파일을 생성한다. 또한 이 실행파일은 기계어 코드와 소스코드를 비교해주는 리스팅 파일도 생성한다.
mp2hex.exe	이 실행파일은 coff 파일을 .hex 파일로 변환한다. hex 파일은 마이크로칩사의 프로그래머인 PICSTART Plus나 PROMATE2를 이용하여 실제 디바이스에 라이팅할 수 있다. mp2hex.exe는 coff 파일을 입력받아 hex파일을 생성한다.
mplib.exe	이 실행파일은 라이브러리(예:*.lib)를 생성하는데 사용된다. 라이브러리 파일은 재사용이 가능한 코드를 만들고 관리하는데 편리하다.
mpsam.exe	이 실행파일은 커맨드 라인 방식의 어샘블러이다. 어샘블리 소스 파일(예:*.asm)을 입력받아 다음 파일들을 생성한다; 오브젝트 파일(*.o), cod 파일, hex 파일, 리스팅 파일(*.lst) 및 에러 파일(*.err)

2장 MPLAB C18의 설치

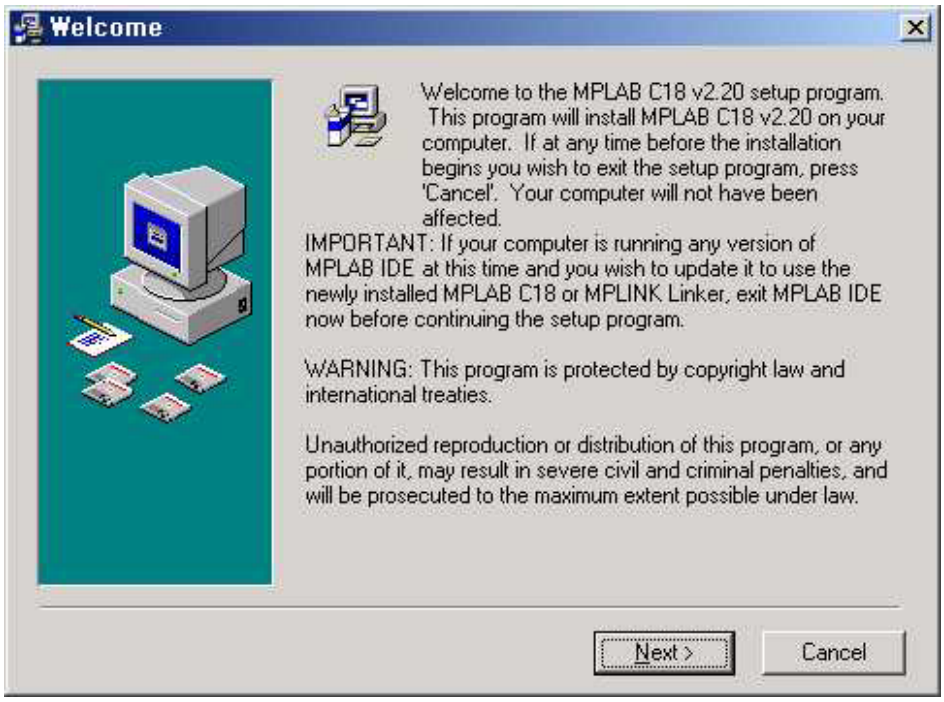
2.1 MPLAB C18 데모버전과 정식버전

MPLAB C18의 데모버전의 압축파일(c18v220ad.zip)을 다운로딩 받은 후, 압축파일을 풀면 MPLAB C18 데모버전 MPLAB-C18-v2.20a-demo-win32.exe이 나온다. 이것을 실행시키면 MPLAB C18 데모버전의 설치가 시작된다.

2.2 MPLAB C18 설치과정

MPLAB C18을 설치하기 위해서 셋업(setup) 프로그램을 실행시켜야 한다. CD에서 인스톨을 하는 경우는 CD상에서 셋업 프로그램을 실행시켜서 인스톨을 시작한다. 마이크로칩 홈페이지에서 다운로딩을 받은 경우에는, 다운로딩 받은 실행파일(MPLAB-C18-v2.20a-win32.exe)을 실행시켜서 인스톨을 시작한다. 셋업 프로그램을 실행시키면 다음과 같은 시작 화면이 뜬다.

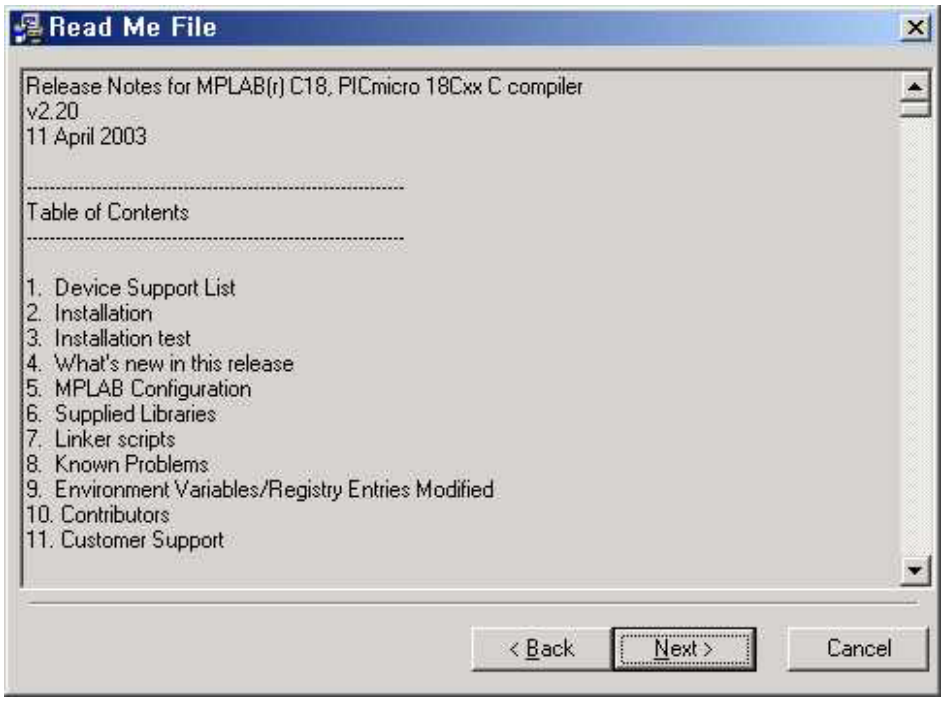
2.2.1 환영 창(Welcome window)



이 화면은 MPLAB C18의 버전 정보와 인스톨을 시작한다는 메시지를 보여준다. 인스톨을 진행하려면 **Next**를 클릭한다.

2.2.2 Readme 파일

다음 화면에는 readme 파일이 보여진다. 이 파일은 알려진 버그나, C18에 대한 중요한 최신 정보를 담고있다. 이것을 다 본 후, 인스톨을 진행하기 위해 **Next**를 클릭한다.



2.2.3 설치 디렉터리 선택

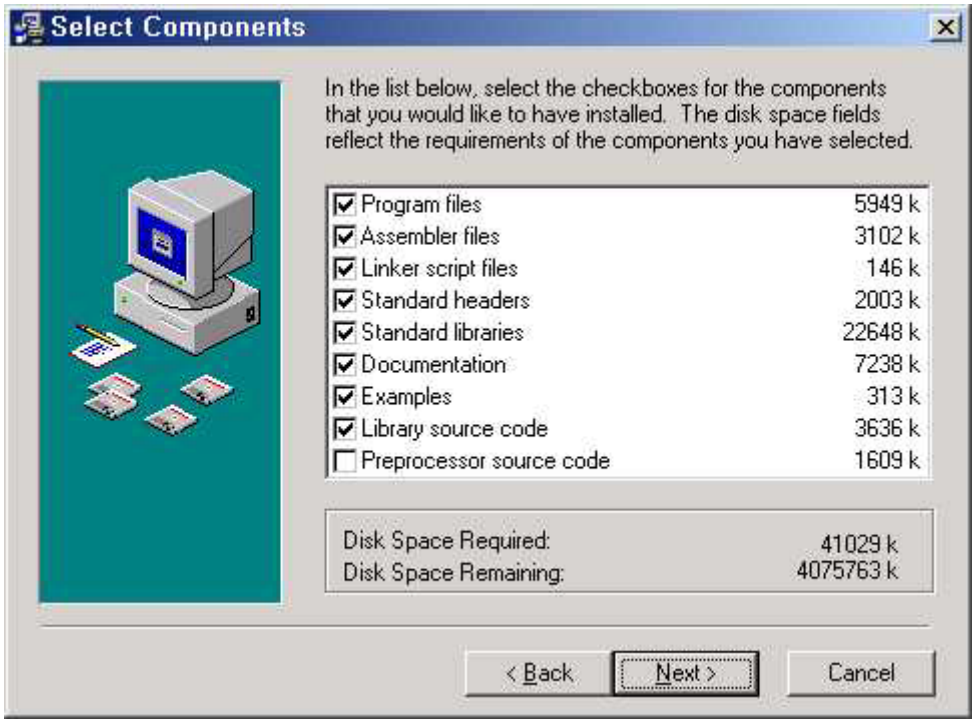
다음 단계는 MPLAB C18의 인스톨 디렉터리를 선택하는 과정이다. MPLAB C18 인스톨 프로그램은 사용자가 C18의 설치 디렉터리를 마음대로 바꿀 수 있도록 지원한다. 디렉터리 선택 화면에서 **Browse**를 클릭함으로 새로운 MPLAB C18 설치 디렉터리를 결정할 수 있다.

디폴트(default) 설치 디렉터리는 `C:\mcc18`이다. 향후 MPLAB IDE에서 MPLAB C18을 등록할 때를 고려하여, 디폴트 디렉터리를 이용하는 것이 바람직하다. MPLAB C18을 업그레

이드 할 때, 셋업 프로그램이 MPLAB C18 설치 디렉터리 및 서브디렉토리를 겹쳐 쓰게 되므로 사용자가 수정한 링크 스크립트나 사용자가 생성한 라이브러리 들은 백업을 받아 놓아야 한다. MPLAB C18이 설치될 디렉터리를 결정했으면, **Next**를 클릭함으로 다음단계로 넘어간다.

2.2.4 컴퍼넌트 선택

그 다음 단계는 결정된 디렉터리에 설치할 C18 소프트웨어 컴퍼넌트를 결정하는 단계이다.



C18은 여러 개의 소프트웨어 컴퍼넌트로 구성되어 있다. 이 화면에서 각 컴퍼넌트를 클릭함으로 설치할 컴퍼넌트를 선택할 수 있다. 각 컴퍼넌트의 설명은 아래의 표와 같다. 이 표 2-1을 참조하여 필요한 컴퍼넌트를 클릭하여 설치될 수 있도록 한다. Preprocessor 컴퍼넌트를 제외하고는 모든 컴퍼넌트를 PC에 설치해주는 것이 좋다.

표2-1: MPLAB C18 소프트웨어 컴퍼넌트

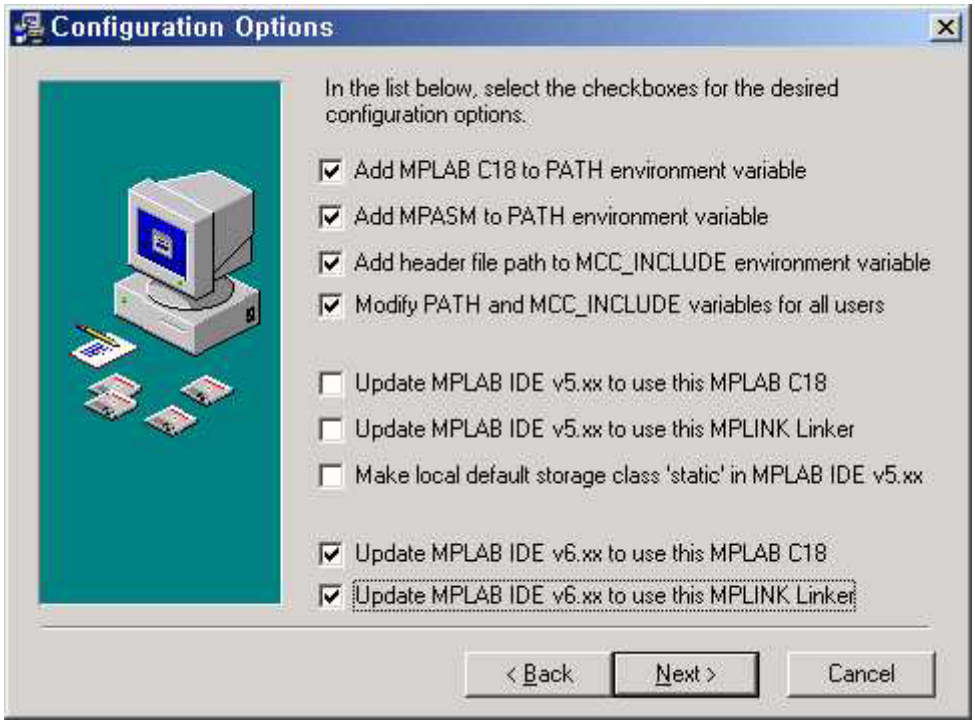
컴포넌트	내용
Program files	이 컴포넌트는 컴파일러와 링커 실행파일 들이다. 사용자는 MPLAB C18 인스톨 할 때 이 파일들을 반드시 설치해야 한다.
Assembler files	이 컴포넌트는 커맨드라인 방식 어셈블러 실행파일(mpasm.exe)과 디바이스용 헤더파일(p18xxx.inc) 들을 가지고 있다.
Linker script files	이 컴포넌트는 MLPINK가 사용하는 링커 스크립트 파일 들을 가지고 있다. 각각의 PICmicro 디바이스 하나마다 하나씩 링커 스크립트가 있다. 링크 스크립트는 메모리 구성, 데이터/코드 배치 에 대

	한 정보를 가지고 있다. 이 링커 스크립트는 MPLAB IDE에서 있는 MPASM 링커 스크립트와는 달리 MPLAB C18에 최적화 되어 있다. MPLINK가 이 파일들을 필요로 하므로 사용자는 반드시 이 컴퍼넌트를 인스톨 해야 한다.
Standard headers	이 컴퍼넌트는 MPLAB C18에서 사용하는 디바이스용 헤더파일 및 라이브러리용 헤더파일을 가지고 있다. 사용자가 standard libraries를 인스톨 할 경우 자동으로 인스톨 된다.
Standard libraries	이 컴퍼넌트는 스타트업(start-up) 모듈, 표준 C 라이브러리 및 디바이스용 C 라이브러리를 가지고 있다. 표준 C 라이브러리는 *MPLAB C18 C Complier Libraries* 매뉴얼을 참고해야 한다. 전형적인 MPLAB C18 프로그램은 스타트업 모듈과 라이브러리를 사용하므로, 이 컴퍼넌트도 반드시 인스톨 되어야 한다.
Documentation	MPLAB C18의 매뉴얼 들이 들어있다.
Examples	이 컴퍼넌트에는 MPLAB C18을 처음 사용하는 사용자에게 유용한 예제가 포함되어 있다.
Library source code	이 컴퍼넌트에는 표준 C 라이브러리와 디바이스용 C 라이브러리의 소스코드가 들어있다. 라이브러리를 수정하고자 하는 사용자는 이 컴퍼넌트를 인스톨 해야 한다.
Preprocessor Source code	이 컴퍼넌트는 C18에 사용되는 각종 프리프로세서의 소스가 들어있다. 이것은 참고용으로 제공된다.

컴퍼넌트의 선택이 끝났으면 **Next**를 클릭하여 다음단계로 넘어간다.

2.2.5 컨피규레이션 잡아주기

컴퍼넌트 선택의 다음 단계는 MPLAB C18의 컨피규레이션을 잡아 주는 것이다. 아래의 컨피규레이션 옵션창에는 여러 가지 MPLAB C18에 관련된 옵션이 나타난다. 여기에서 MPLAB C18 사용에 필요한 옵션을 선택해주면 된다.



Add MPLAB C18 to PATH environment variable

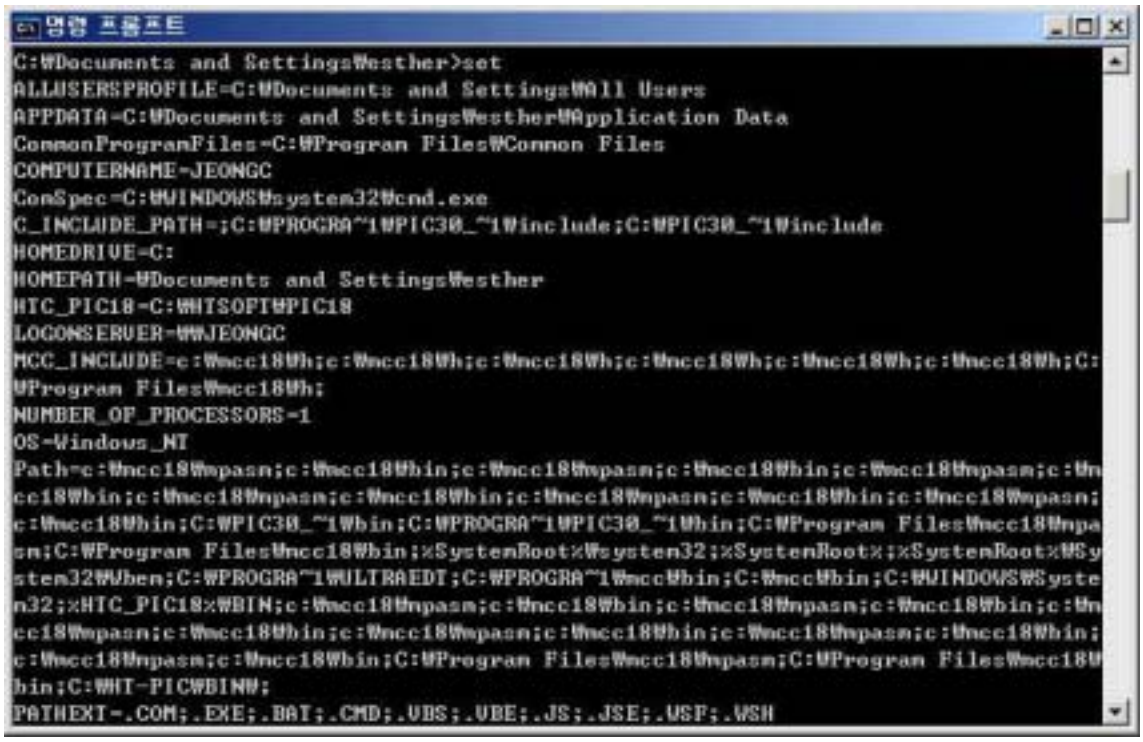
이 항목을 선택하면, MPLAB C18 컴파일러와 링커의 실행파일의 경로(path)를 PC의 PATH 환경변수에 추가한다. 이를 통하여, 사용자는 C18 컴파일러와 링커를 어떤 디렉터리에서든지 실행할 수가 있게 된다.

Add MPASM to PATH environment variable

이 항목을 선택하면, MPASM의 실행파일의 경로(path)를 PC의 환경변수에 추가한다. 이를 통하여 사용자는 MPSAM을 어떤 디렉터리에서든지 실행할 수가 있게 된다.

Add header file path to MCC_INCLUDE environment variable

이 항목을 선택하면, MPLAB C18의 헤더파일의 경로(path)가 MCC_INCLUDE 환경변수에 추가된다. 만약 사용자 PC의 환경변수에 MCC_INCLUDE 환경변수가 없으면, 이 환경변수가 생성된다. C18은 –I 옵션에 명기된 디렉터리외에 MCC_INCLUDE에 명시된 경로에서 헤더파일을 찾는다. 이 옵션을 선택함으로, 사용자는 커맨드 라인 옵션인 –I를 사용하지 않고도 표준 헤더파일을 인크루드(include) 할 수 있다. 이 옵션을 선택한 후 프롬프트 창에서 set 명령을 실행하면 환경 변수를 볼 수 있는데, 그 속에서 MCC_INCLUDE를 볼 수 있다.



Modify PATH and MCC_INCLUDE variable for all users

이 옵션을 선택하면, 상기 3가지의 옵션의 변경사항을 PC의 모든 사용자의 환경변수에 적용한다. 이 옵션을 선택하지 않으면, 오직 현재 로그인 한 사용자의 환경 변수에만 적용한다. 이 옵션은 사용자가 윈도200이나 윈도NT를 가진 PC에 주관리자(administrator)로 로그인 할 경우에만 나타난다.

Update MPLAB IDE v5.xx to use this MPLAB C18

이 옵션은 사용자의 PC에 MPLAB IDE v5xx가 설치된 경우에만 나타난다. 이 옵션을 선택하면, MPLAB IDE v5.xx에서 설치되는 MPLAB C18을 사용할 수 있도록 해준다.

Update MPLAB IDE v5.xx to use this MPLINK linker

이 옵션은 사용자의 PC에 MPLAB IDE v5xx가 설치된 경우에만 나타난다. 이 옵션을 선택하면, MPLAB IDE v5.xx에서 설치되는 MPLINK를 사용할 수 있도록 해준다.

Make local default storage class 'static' in MPLAB IDE v5.xx

이 옵션은 사용자의 PC에 MPLAB IDE v5xx가 설치된 경우에만 나타난다. 이 옵션이 선택되면, 프로그램 내에서 선언된 모든 지역 변수와 파라미터 변수들이 정전(static)으로 선언된다. 이 말은 변수들이 소프트웨어 스택에 저장되지 않고 전역(global)변수로 데이터 메모리에 저장된다. 자동(auto)변수를 처리할 때 보다 정적변수 처리할 때 더 적은 어셈블리 명령어가 사용된다.

Update MPLAB IDE v6.xx to use this MPLAB C18

이 옵션은 사용자의 PC에 MPLAB IDE v6.xx가 설치된 경우에만 나타난다. 이 옵션을 선택하면, MPLAB IDE v6.xx에서 설치되는 MPLAB C18을 사용할 수 있도록 해준다. 이 옵션을

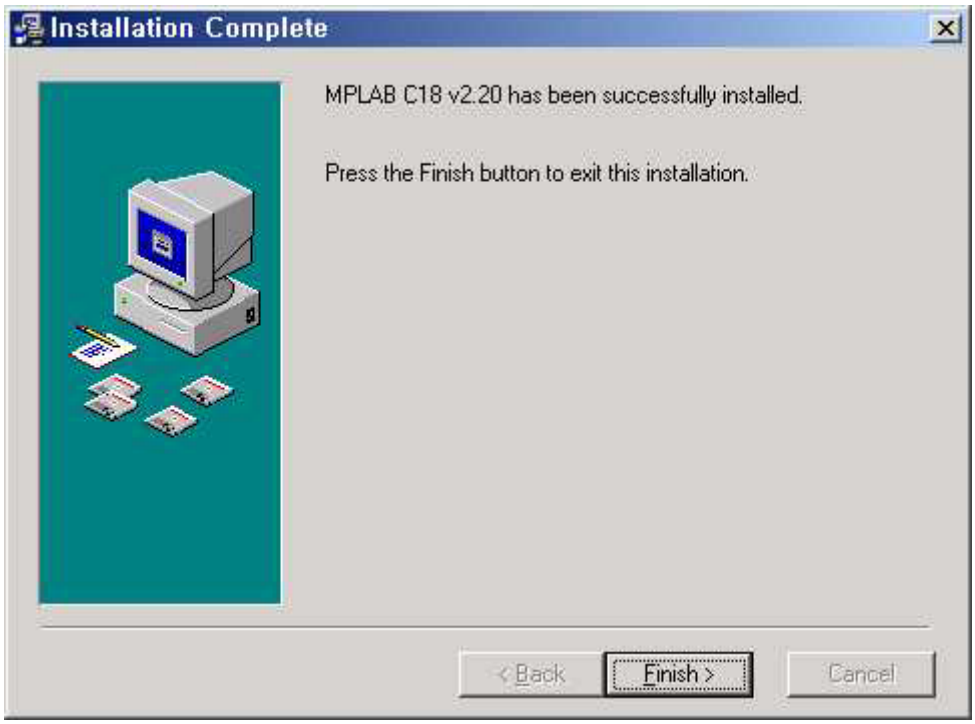
선택하면, C18 라이브러리 디렉터리를 디폴트 라이브러리 디렉터리 경로로 사용하게 된다.

Update MPLAB IDE v6.xx to use this MPLINK linker

이 옵션은 사용자의 PC에 MPLAB IDE v6.xx가 설치된 경우에만 나타난다. 이 옵션을 선택하면, MPLAB IDE v5.xx에서 설치되는 MPLINK를 사용할 수 있도록 해준다. 컨피규레이션이 끝났으면, **Next**를 클릭하여 다음으로 넘어간다. 다음 단계에서 실질적인 MPLAB C18의 인스톨이 시작된다.

2.2.6 MPLAB C18 설치시작

다음 화면이 나오면 MPLAB C18의 인스톨이 끝이 났다.



3장 예제 사용해 보기

3.1 예제 프로그램 개요

MPLAB C18을 MPLAB IDE에 등록을 했으면, C18이 설치된 example 디렉터리에서 예제 프로그램을 작성하여 실행해 볼 수 있다. 이 예제를 실행하려면, MPLAB IDE 6.xx 버전이 설치되어 있어야 한다. 또한 일부 예제 프로그램은 마이크로칩사에서 제공하는 인-서킷 디버거(In-circuit debugger)인 ICD2와 데모보드 인 PICDEM2 plus가 필요하다. 이 두 가지를 구매하기 위해서는 마이크로칩사나 대리점에 접촉하면 된다.

C18 디렉터리에는 모두 4가지의 예제 프로그램이 있다. 각 예제 프로그램의 설명은 다음과 같다.

- **Example 1**: MPLAB IDE에서 프로젝트를 활용하는 예제. 이 예제를 통해 MPLAB IDE의 프로젝트를 생성, 빌딩 및 디버깅 등을 학습한다.
- **Example 2**: C18에 포함되어 있는 PIC18용 각종 라이브러리를 사용해보는 예제; C18은 많은 유용한 라이브러리를 제공하는 데, 이 예제를 통해서 PIC18용 페리퍼럴 라이브러리, 표준 C 라이브러리 및 수학 라이브러리 등의 활용을 학습할 수 있다.
- **Example 3**: PIC18의 데이터 메모리 중 억세스 RAM(Access RAM)에 변수를 할당하는 방법을 학습한다.
- **Example 4**: PIC18용 페리퍼럴 라이브러리를 활용하여 인터럽트 서비스 루틴의 활용을 학습할 수 있다.

3.2 예제1 따라 하기

이 예제는 MPLAB IDE를 이용하여 프로그램을 개발하여 MPLAB SIM 시뮬레이터와 PICDEM2 plus 보드에서 실행해 볼 수 있다. 예제1의 디렉터리는 `c:\mcc18\example\getting_started\example1` 이다. 이 예제를 통해서 MPLAB IDE의 프로젝트를 생성하고 빌드 한 후, MPLAB SIM이나 PICDEM2 plus에서 실행해볼 수 있다. 이 예제에서는 C18이 c:\mcc18 디렉터리에 설치된 것으로 간주하고 학습을 진행한다. 예제1의 소스 코드인 example1.c는 다음과 같다.

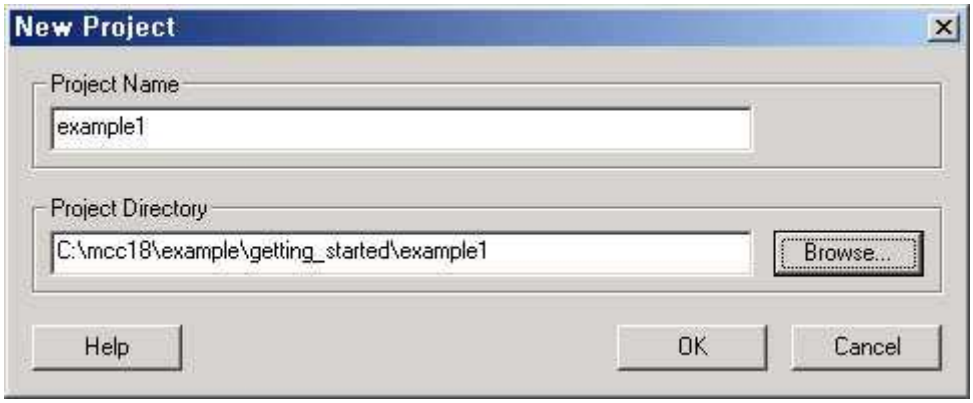
```
/*
 * This is example 1 from "Getting Started with MPLAB C18".
 */
#include <p18cxxx.h>   /* for TRISB and PORTB declarations */
int counter;
void main (void)
{
   counter = 1;
```

```
    TRISB = 0;          /* configure PORTB for output */
    while (counter <= 15)
    {
      PORTB = counter;   /* display value of 'counter' on the LEDs */
      counter++;
    }
}
```

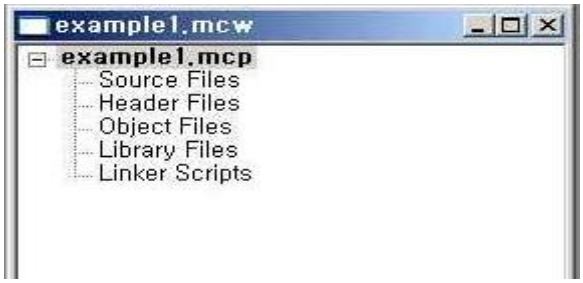
TRISB와 PORTB는 PIC18Fxx 디바이스의 I/O 제어용 레지스터들이다. PICDEM2 plus 보드에서는 PORTB가 LED 연결되어 있다. 따라서 PORTB를 세트(set)하면 LED가 켜지고, 클리어(clear)하면 LED가 꺼지게 된다. TRISB 레지스터는 세트일 때, PORTB 핀이 입력(input)이 되고 클리어일 때 출력(output)이 된다.

3.2.1 프로젝트 생성하기

MPLAB IDE를 실행하고 *Project>New* 메뉴를 실행한다. 이 메뉴를 실행하면 생성되는 프로젝트의 이름과 실행 디렉터리를 정할 수 있다.



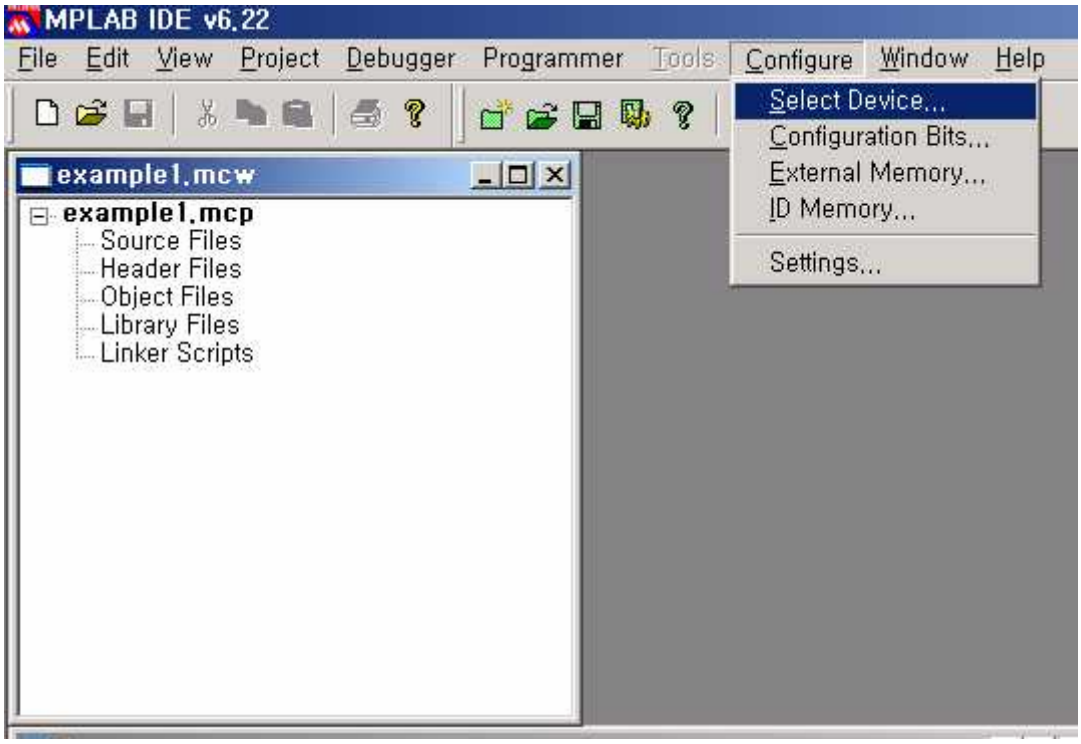
프로젝트의 이름은 example1으로 하고, 현재 소스 코드인 example1.c가 설치된 디렉터리가 `c:\mcc18\example\getting_started\example1` 이므로 프로젝트 디렉터리는 위 화면과 같이 잡아준다. 새로운 프로젝트의 이름과 디렉터리를 정한 후 **OK**를 클릭하면, 아래와 같은 프로젝트 창이 뜬다. MPLAB IDE의 프로젝트 파일은 확장명이 *.mcw 이다.



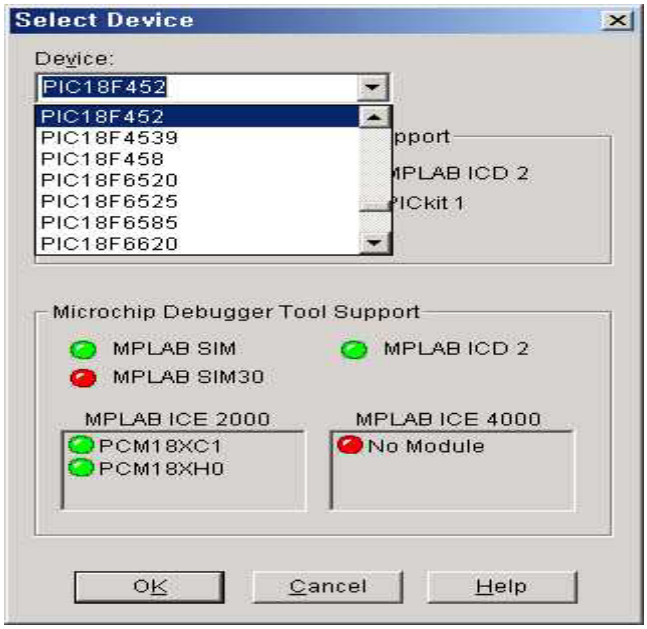
3.2.2 디바이스 결정하기

프로젝트 창이 만들어 졌으면 프로젝트를 만들기 위한 기반작업이 끝났다. 이 후에 해야 할

것은 프로젝트에서 사용될 PIC18 디바이스를 선택하는 것이다. 디바이스의 선택은 아래의 그림처럼 *Configure > Select Device* 메뉴에서 할 수 있다. 디바이스 선택이 잘못되면 컴파일 과정이나 어셈블리 과정에 에러가 발생한다.



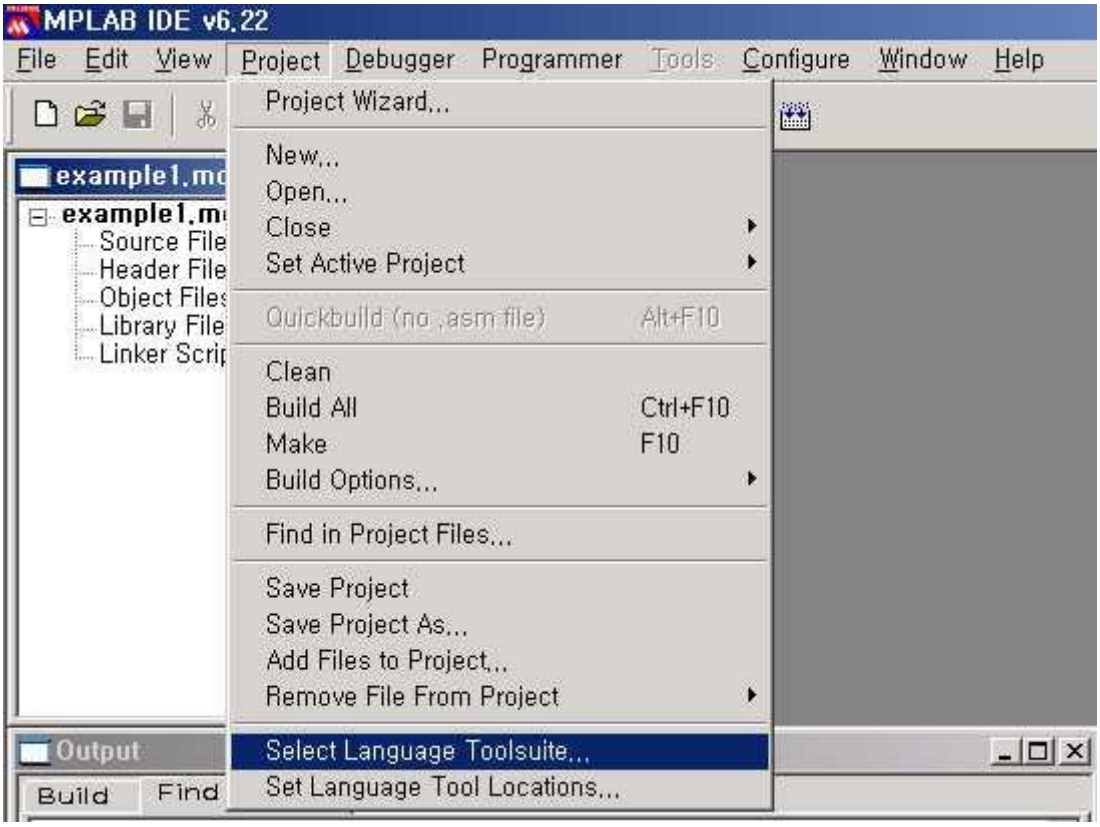
예제에서 사용하는 디바이스는 PIC18F452 이므로 이 디바이스를 선택해 준다.



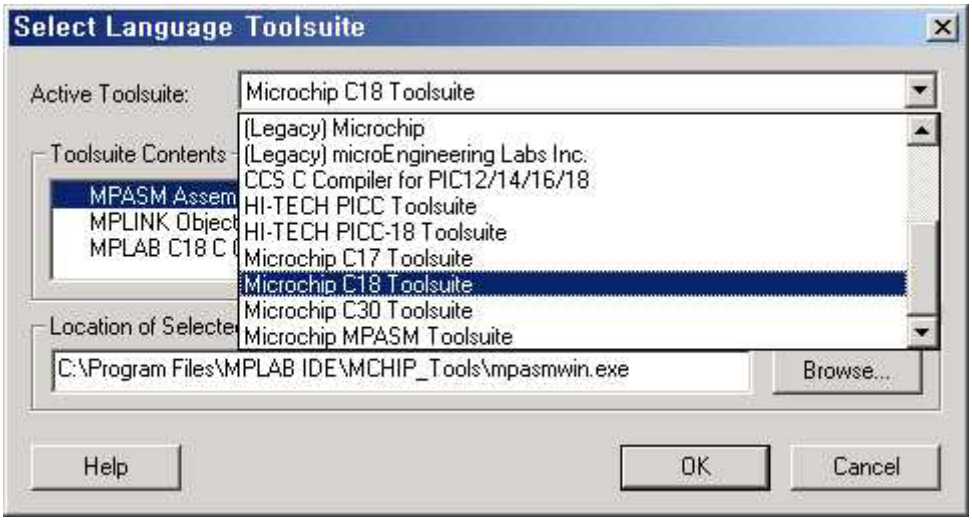
디바이스의 선택이 끝나면 **OK**를 클릭한다. 이것으로 디바이스의 선택이 끝났다.

3.2.3 프로젝트 셋팅하기

디바이스를 선택한 후 할 일은 프로젝트에서 사용할 언어를 선택하는 것이다. 사용할 언어의 선택은 *Project > Select Language Toolsuite* 메뉴에서 할 수 있다.

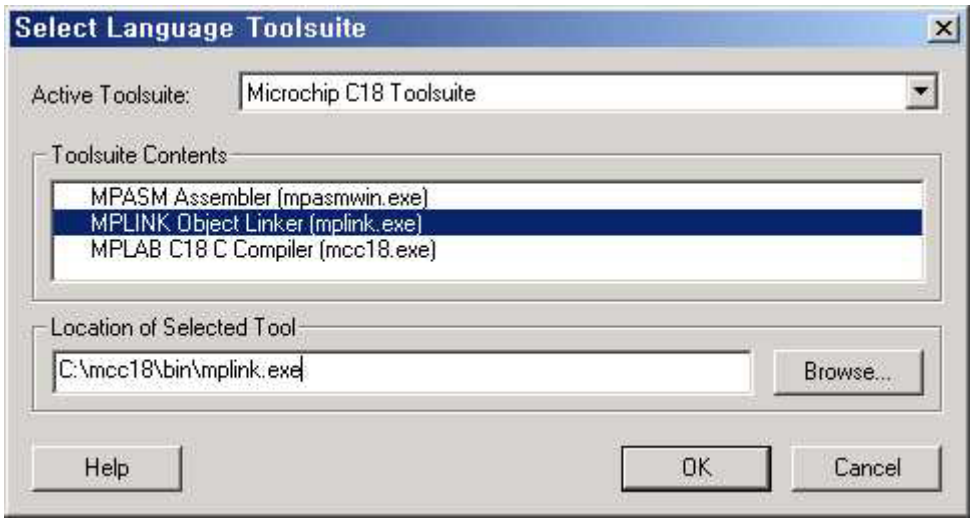


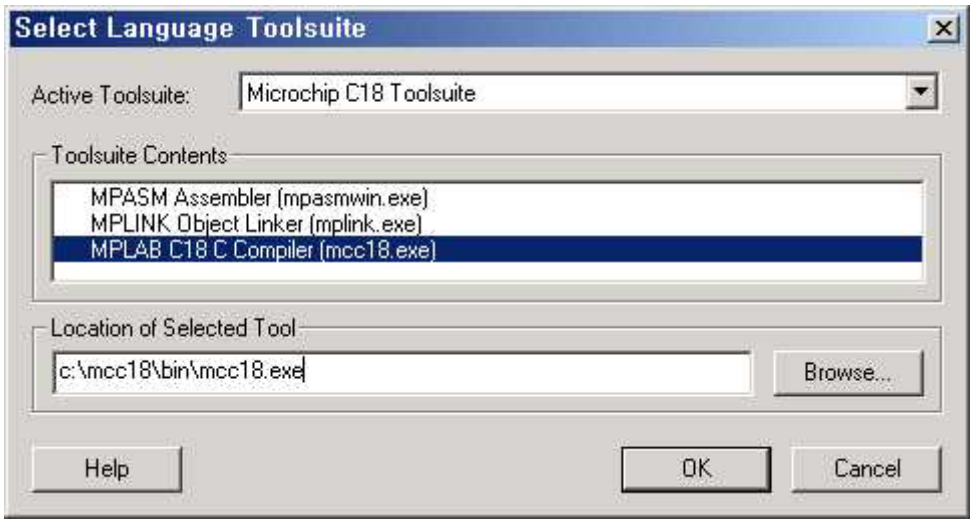
Set Language Toolsuite 메뉴를 선택하면 아래와 같은 창이 뜬다. 예제1에서는 MPLAB C18을 사용할 것이므로 Microchip C18 Toolsuite를 선택해준다. 선택을 해준 후 **OK**를 클릭해준다.



Set Language Toolsuite에서 사용할 언어의 실행파일 및 디렉터리가 정확히 MPLAB IDE에 등록이 되어 있는가를 확인해야 한다. 만약 Select Language Toolsuite 창의 Toolsuite Contents 항목의 각 언어 실행파일 들 앞에 X 표시가 되어 있으면 이것은 MPLAB C18의

등록이 제대로 되어 있지 않음을 말한다. 이런 경우 대게 MPLAB IDE 6.xx를 설치할 때, MPLAB C18을 사용옵션을 선택하지 않을 경우 발생한다. 이때는 MPLAB IDE의 MPLAB C18 사용옵션을 설정해주어야 한다. 자세한 사항은 *MPLAB IDE User's Guide*를 참고한다. 예제1에서 사용할 C18의 경우, Toolsuite Contents에서 MPLINK 및 MPLAB C18의 실행파일 및 디렉터리를 아래의 그림과 같이 설정해준다.

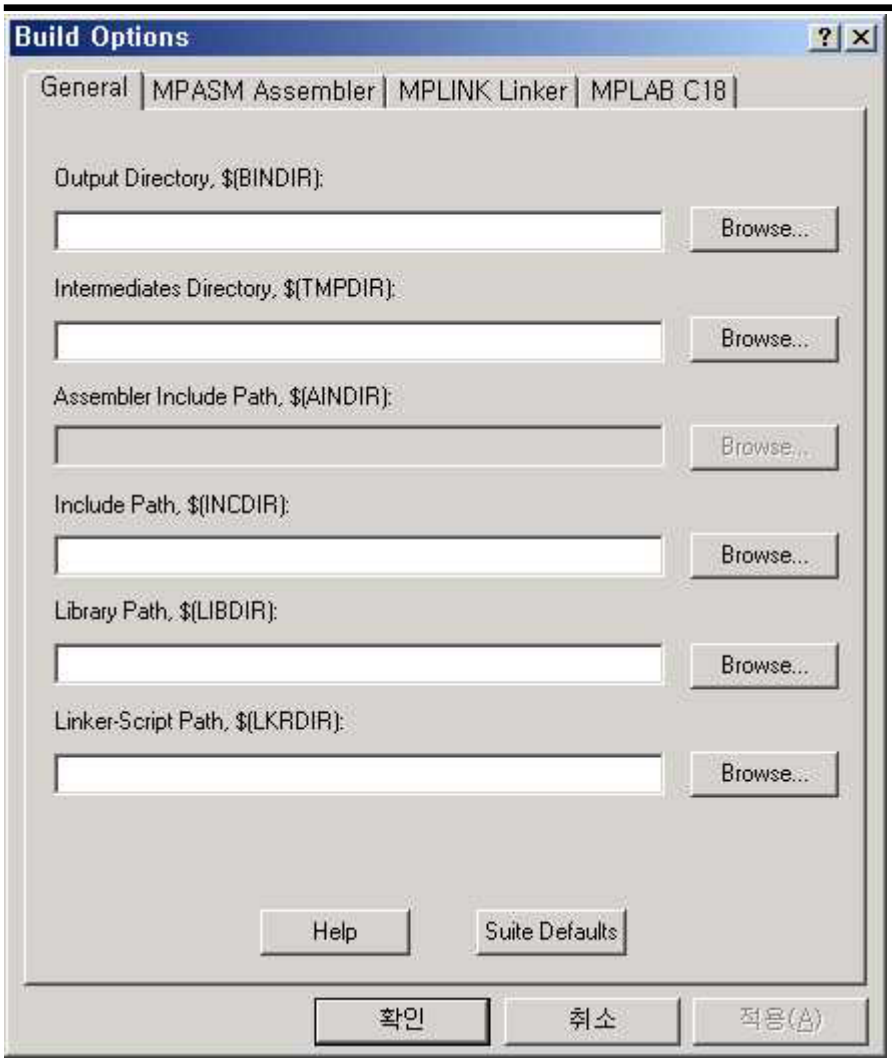




3.2.4 컴파일러 링커 커맨드라인 옵션 설정 하기

프로젝트에서 사용할 언어의 선택이 끝났으면, 다음에는 컴파일러 및 링커의 커맨드 라인 옵션을 선택해주어야 한다. 이것은 *Project > Build Options > Project* 메뉴에서 설정할 수 있다. 이 메뉴에서 설정해야 할 주요 항목은 헤더파일 등 각종 참조파일이 있는 곳의 디렉터리를 설정과 컴파일러/링커/어셈블러의 커맨드라인 옵션을 설정해 주는 것이다.

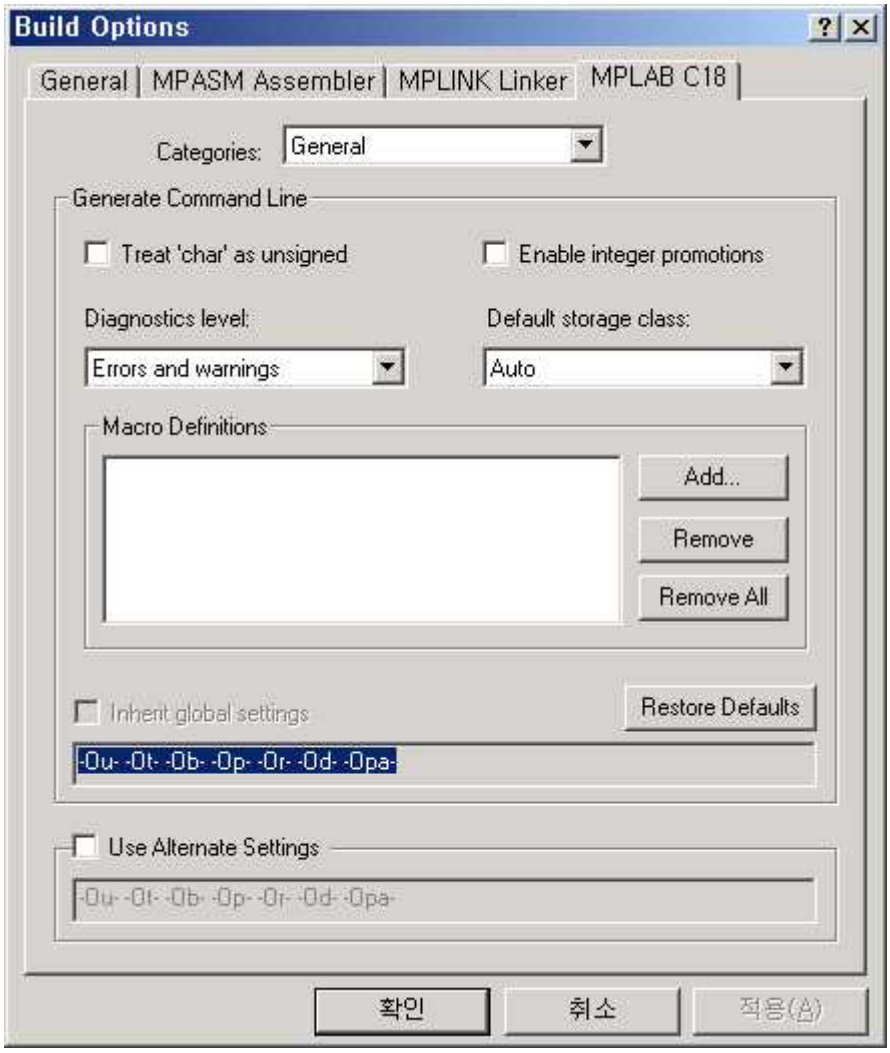
먼저 해주어야 할 것은 각 참조파일의 디렉터리를 설정해 주는 것이다. *Project > Build Options > Project* 메뉴를 선택하면 아래의 창이 뜬다. 여기서 General 탭을 선택하면 각 참조파일의 디렉터리를 설정하는 화면이 나온다.



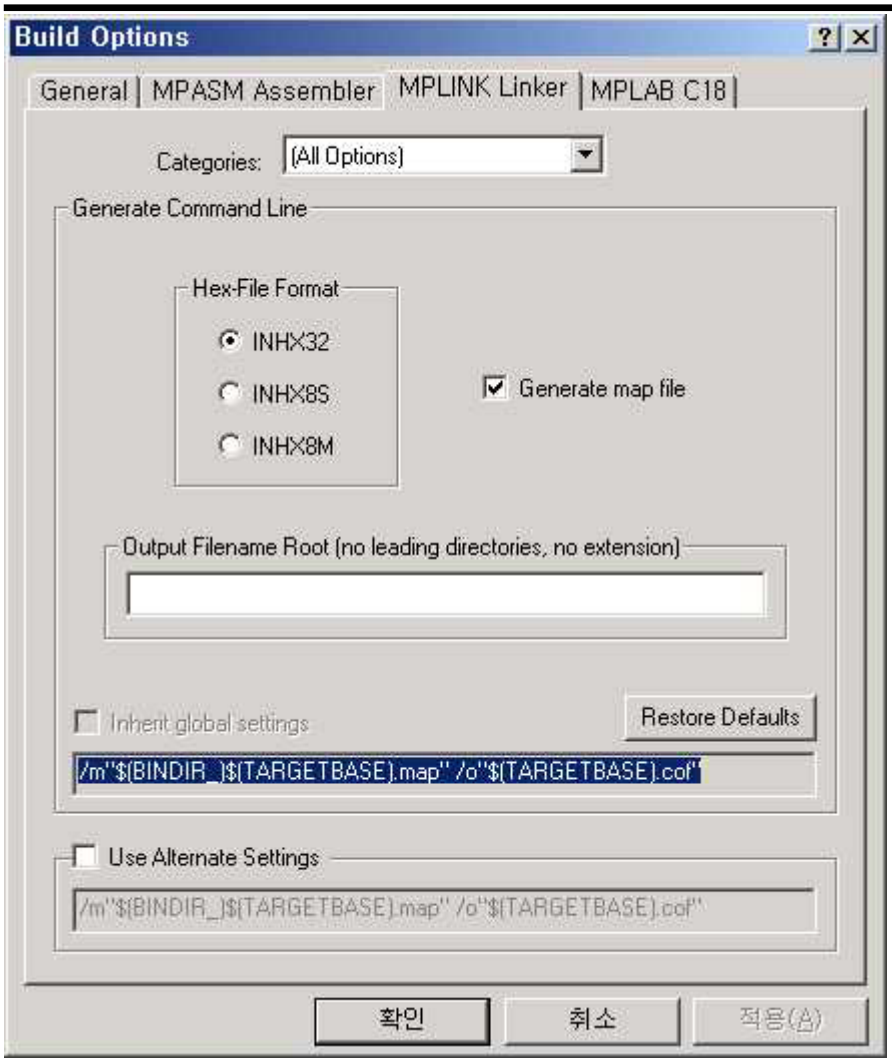
Include Path,$(INCDIR)에 MPLAB C18이 설치된 디렉터리의 헤더파일 서브디렉터리를 써넣는다. C18을 디폴트로 인스톨 했으면, 헤더파일 서브디렉터리는 `c:\mcc18\h`이다. Include Path를 써넣는 것은 컴파일러 커맨드라인 옵션의 –I를 설정해 주는 것과 같다. 컴파일러는 컴파일 할 때, Include Path의 경로에서 헤더파일을 찾는다. 또한 Library Path,$(LIBDIR)와 Linker-Script Path,$(LKRDIR)를 MPLAB C18 설치 디렉터리의 라이브러리 서브디렉터리 및 링커 스크립트 서브 디렉터리에 써넣는다. 역시 C18을 디폴트로 설치했으면, 라이브러리 서브디렉터리는 `c:\mcc18\lib`이고 링커스트립트 서브 디렉터리는 `c:\mcc18\lkr`이 될 것이다. 링커는 프로그램을 링크할 때, Library Path에서 라이브러리 파일을 찾고 Linker-Script Path에서 링커스트립트 파일을 찾을 것이다. Output Directory는 프로그램을 빌드 했을 때, 출력파일 인 cod, cof 및 hex 파일이 놓여지는 위치이다. 이것을 공란으로 두면, 출력파일 들은 프로젝트 디렉터리에 놓여지게 된다. Intermediate Directory는 컴파일 할 때 생기는 오브젝트 파일들이 놓여지는 디렉터리다. 이것을 공란으로 두면 오브젝트 파일은 소스파일 디렉터리에 놓여지게 된다.

General 탭에서 각종 디렉터리 경로를 정했으면, 다음에는 컴파일러와 링커의 커맨드라인 옵션을 정해주어야 한다. Build Option 창의 MPLAB C18 탭에서 컴파일러의 커맨드라인 옵

션을 설정할 수 있다. 이 매뉴얼에 포함된 예제들의 경우 디폴트 커맨드라인 옵션으로 빌드할 수 있다. 자세한 커맨드라인 옵션에 대해서는 *MPLAB C18 C Compiler User Guide*를 참조하면 된다.

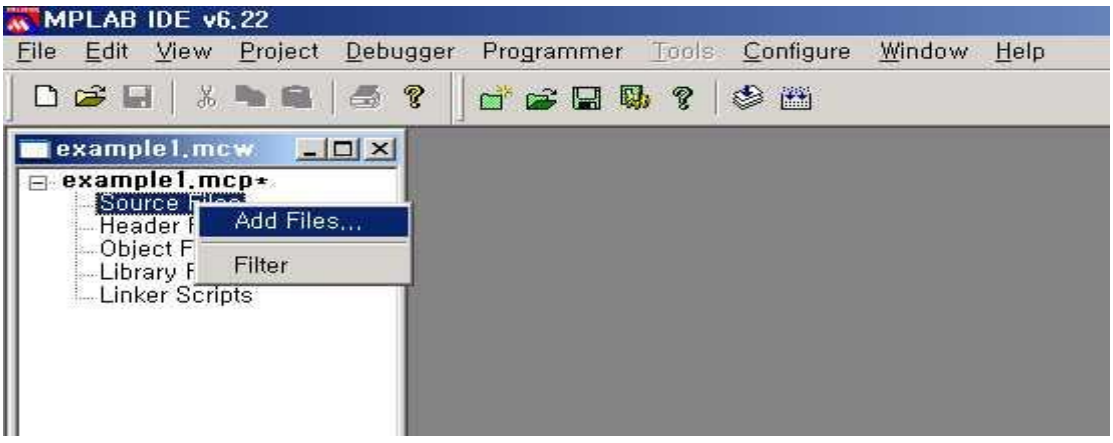


링커의 커맨드라인 옵션은 Build Option 창의 MPLINK Linker 탭에서 설정할 수 있다. 역시 이 매뉴얼의 예제들은 별도의 설정 없이 디폴트 커맨드라인 옵션으로 링크할 수 있다. 디폴트 커맨드라인 옵션을 사용할 때는 .map 파일이 생성되지 않는다. .map 파일을 생성하기 위해서는 MPLINK Linker 탭에서 Generate map file을 클릭하고 **OK**를 클릭하면 된다.



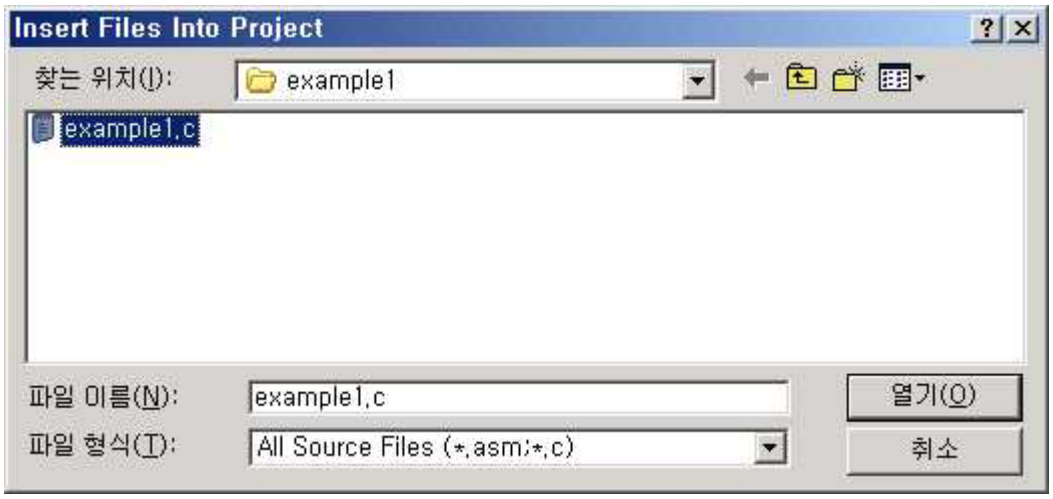
3.2.5 프로젝트에 파일 더하기

컴파일러 및 링커의 커맨드라인 옵션 설정이 끝났으면, 다음에는 프로젝트에 C 소스파일을 더해야 한다. 프로젝트 창에서 Source files을 마우스로 가리키고 마우스 오른쪽 버튼을 클릭한다. Add files란 항목이 나올 것이다. 이것을 클릭한다.



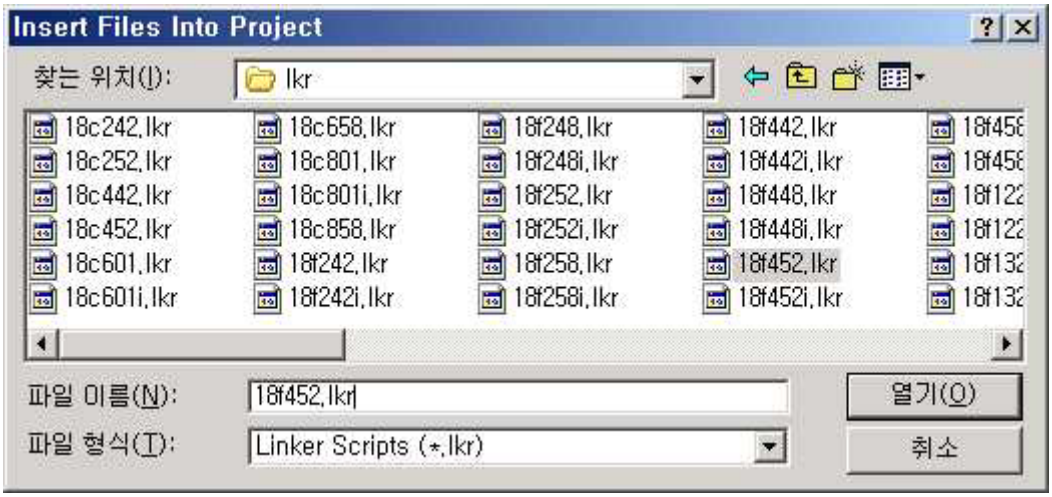
예제 1의 소스파일은 ,MPLAB C18을 디폴트로 설치 했다면,

c:\mcc18\example\getting_started\example1 에 있다. 이 디렉토리에서 example1.c를 선택한다.



그리고 **열기**를 클릭하여, 소스 파일을 프로젝트에 더한다. 프로젝트 창에 소스파일이 첨가된 것이 보일 것이다. 예제의 헤더파일의 경우 C 소스코드에 명시되어 있으므로 프로젝트 창에서 Header Files에 어떤 파일을 추가할 필요는 없다. 컴파일러는 C 소스파일에서 #include<xxx.h>로 명시된 헤더파일은 Build Option 창에서 설정해준 Include Directory에서 찾을 것이다. 또한 컴파일러는 #include"xxx.h"로 명시된 헤더파일은 소스파일 디렉터리에서 찾을 것이다.

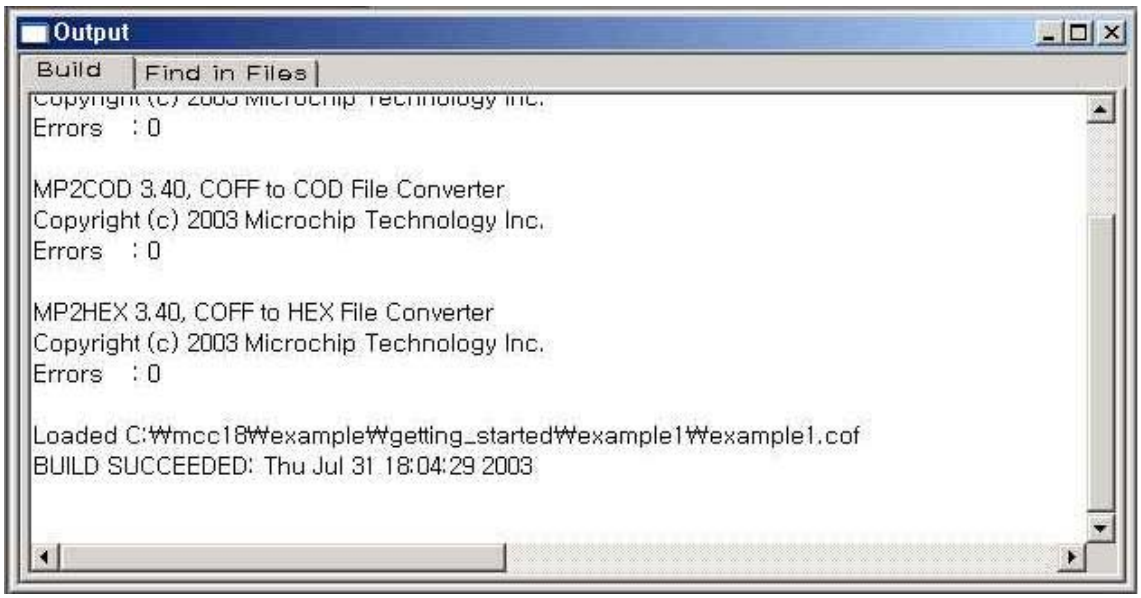
MPLAB C18이 필요로 하는 스타트업 코드, 표준 라이브러리 및 디바이스 라이브러리 들은 링커 스크립트에 명시되어 있으므로 특별히 프로젝트 창에서 더해 줄 필요는 없다. 한편 MPLINK가 필요로 하는 링커 스크립트 파일은 C 소스파일과 같은 방법으로 프로젝트 창에서 더해질 수 있다. 링크 스크립트 파일은 c:\mcc18\lkr 디렉터리에 있다. 예제에서 사용하는 디바이스가 PIC18F452이므로 18f452.lkr 링크 스크립트 파일이 필요하다.



열기를 누름으로 18f452.lkr을 프로젝트 창에 추가한다.

3.2.6 프로젝트 빌드 하기

프로젝트에 여러 가지 필요한 파일을 더했으면, 다음에는 프로젝트 빌드를 해야한다. 프로젝트 빌드는 ***Project>Build*** All 메뉴로 실행할 수 있다. 빌드 과정에서 에러나 경고가 발생하면 Output 창에 해당 메시지가 뜬다. 에러 없이 빌드가 완료되면 아래와 같은 메시지가 Output 창에 뜬다. 이 빌드를 통해 cod, cof 및 hex 파일이 생성되어 소스파일 디렉터리에 놓여진다.

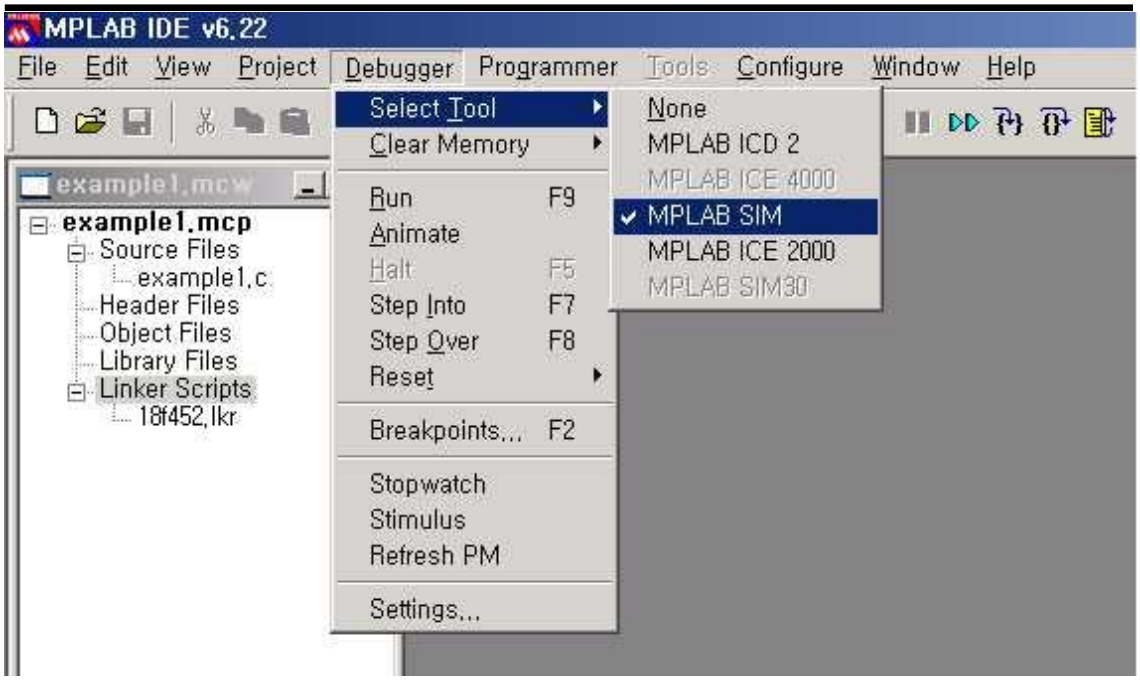


빌드가 끝나면, 빌드 된 코드를 실행시켜 볼 수 있다. 코드를 실행시키는 방법은 두 가지가 있다. 첫번째는 MPLAB IDE에 내장된 PIC 시뮬레이터(MPLAB SIM)를 이용하는 것이다. 두번째 방법은 마이크로칩사의 인-서킷 디버거(In-circuit Debugger)인 ICD2를 이용하는 것이다.

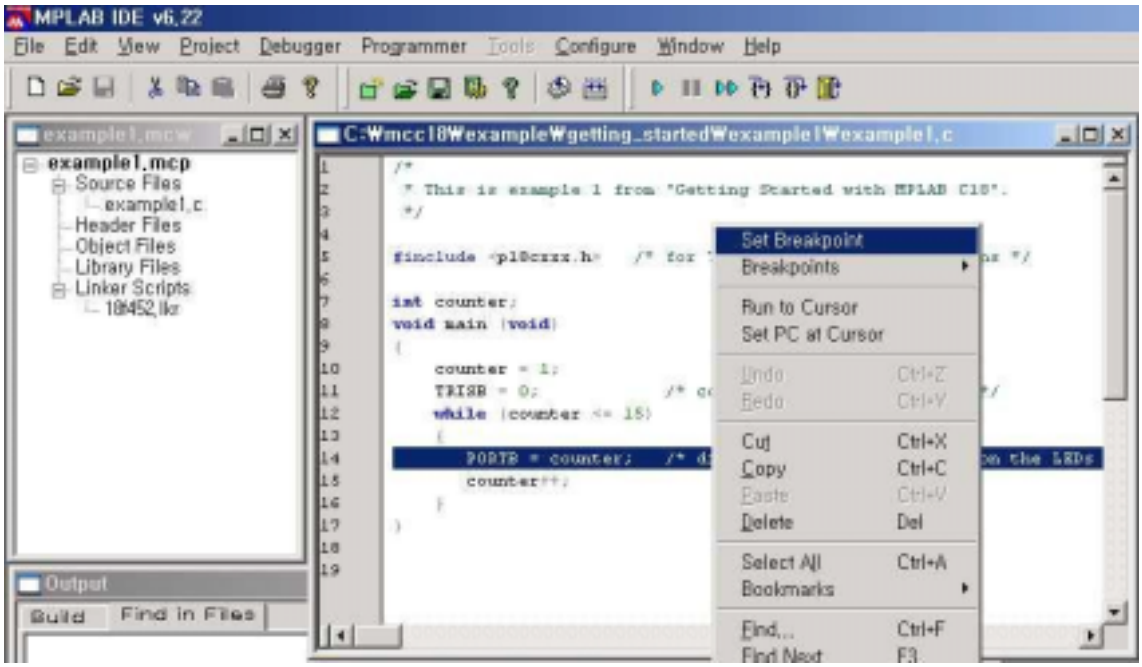
3.2.7 MPLAB SIM

MPLAB SIM을 실행시키기 위해서는 ***Debugger>select Tool>MPLAB SIM*** 메뉴를 사용한다. 프로젝트 빌드 후, 아래의 그림과 같이 MPLAB SIM을 실행시키면 프로젝트에서 빌드 된 출력파일이 자동으로 MPLAB SIM에 로드 되고 MPLAB SIM이 실행된다. MPLAB SIM에서는 실행, 스탭실행, 브레이크포인트 등 디버깅을 위한 대부분의 기능이 제공된다. 시뮬레이터이므로 제약이 있는데, 이에 대한 자세한 사항은 *MPLAB IDE user's Guide*매뉴얼을 참고하면 된다.

MPLAB C18 C 컴파일러 시작하기

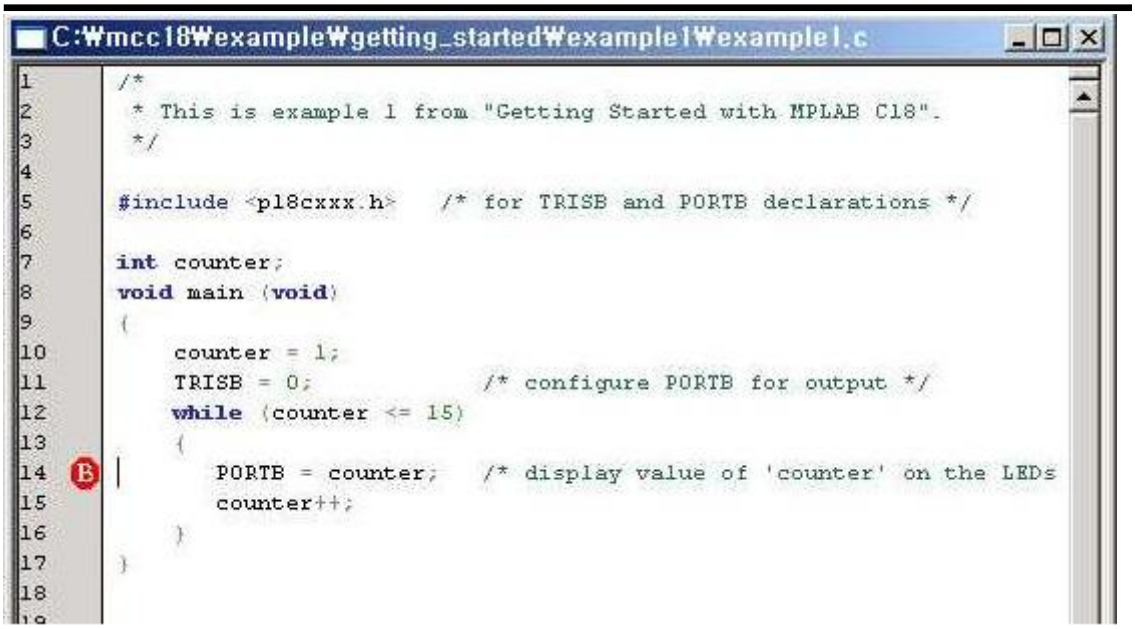


MPLAB SIM을 이용하여 프로그램을 실행하고 디버깅을 하는 방법은 다음과 같다. 먼저 프로젝트 창에서 C 소스파일을 클릭하여 소스창을 연다. 아래의 그림과 같이 소스창을 열고, 소스창에서 브레이크포인트를 잡고 싶은 행을 더블클릭 하거나, 마우스 오른쪽 버튼을 통해 Set Breakpoint를 선택한다.

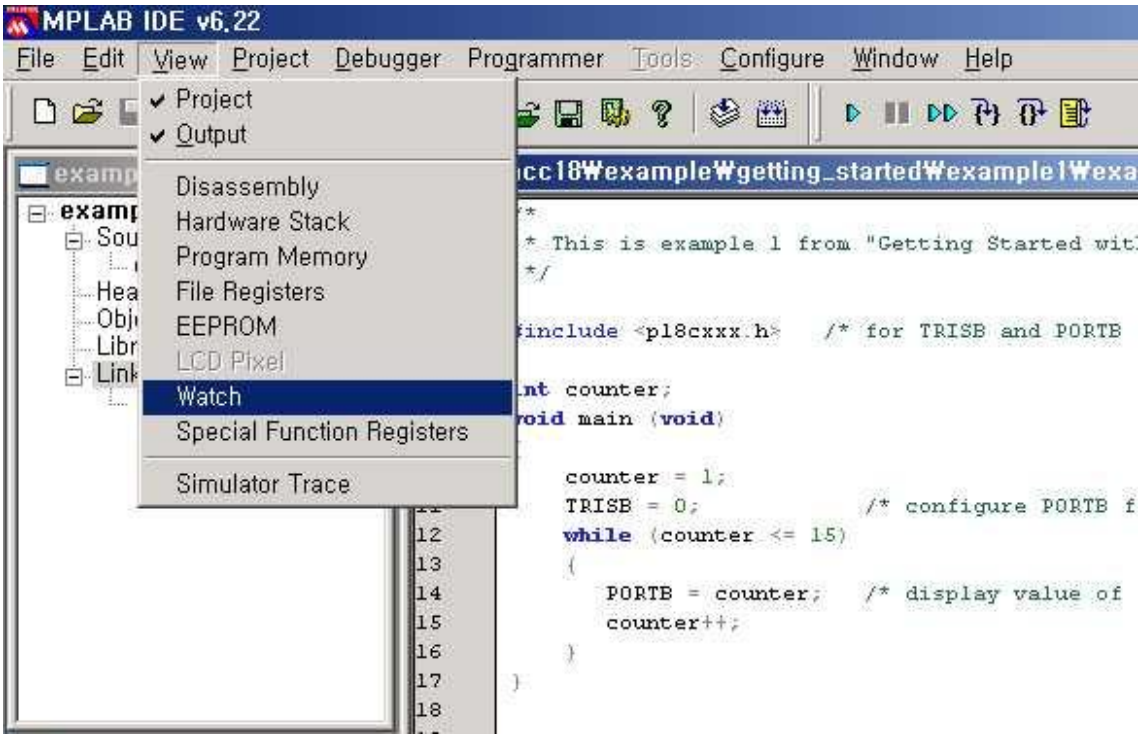


소스창의 왼쪽 가장자리에 표시가 브레이크포인트가 잡힌 것이다.

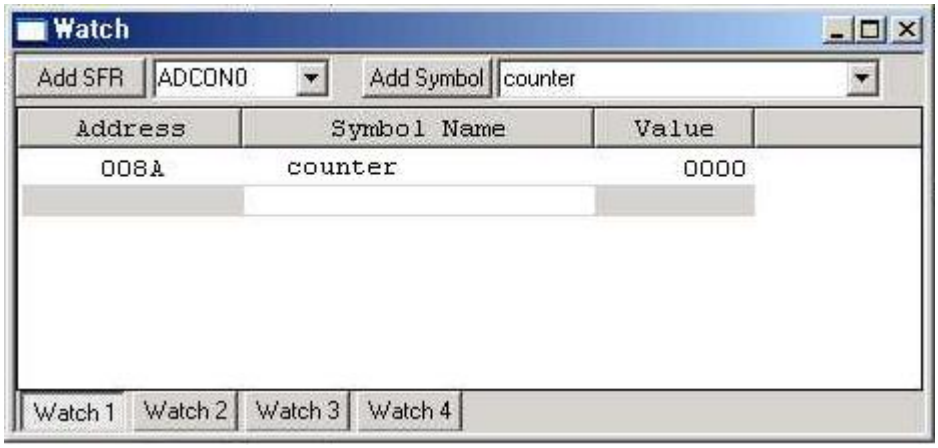
MPLAB C18 C 컴파일러 시작하기



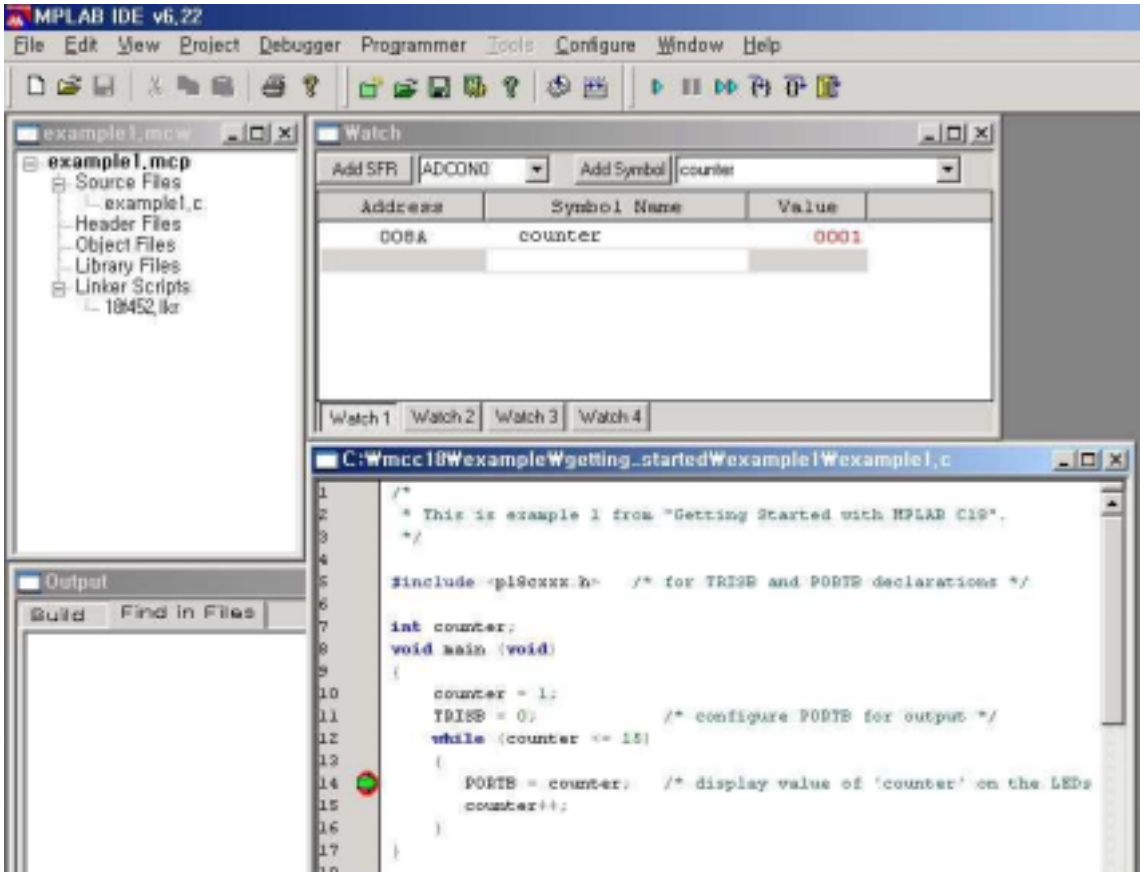
예제1이 MPLAB SIM에서 동작하는 것을 관찰하기 위해서는 MPLAB IDE의 Watch 기능을 이용한다. Watch 기능은 *View>Watch* 메뉴로 실행시킨다.



Watch 창에서 Counter를 선택하고 **Add Symbol** 버튼을 클릭한다.

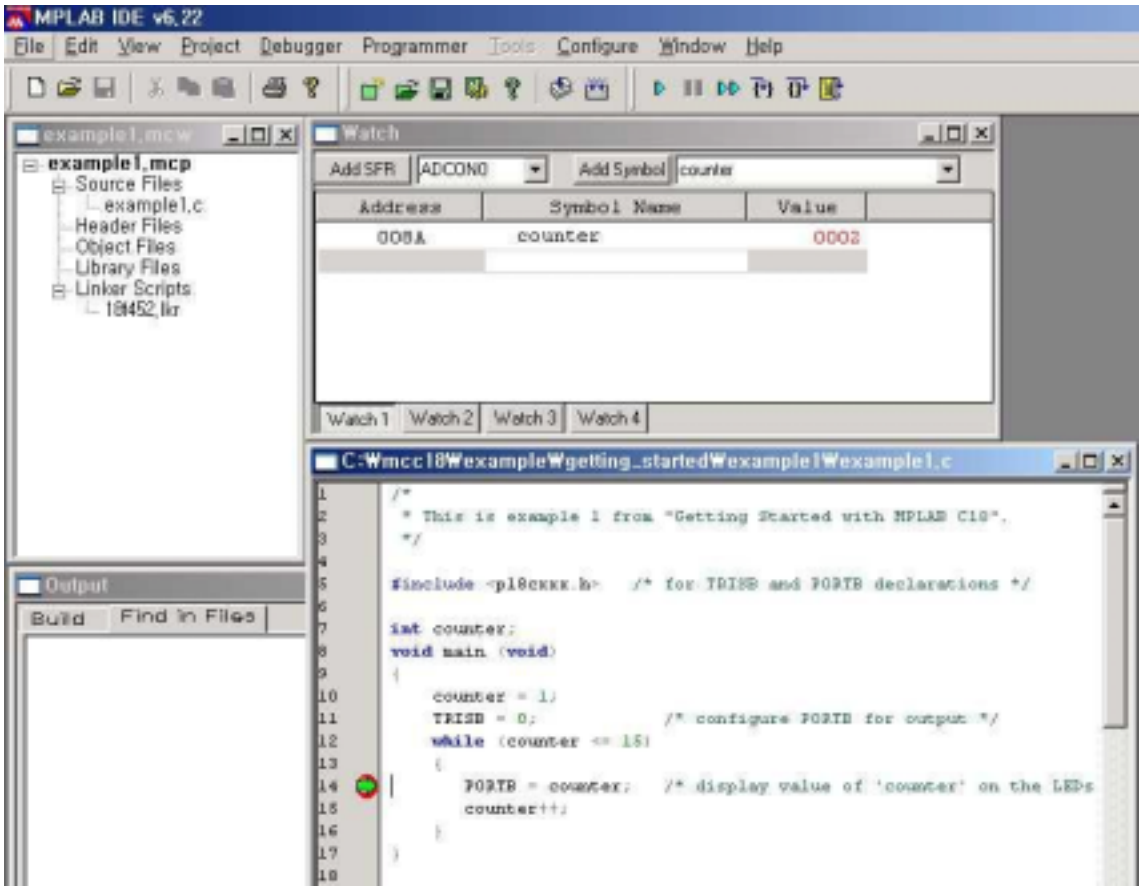


그리고 프로그램을 실행키 위해서 툴바(Tool Bar)에서 Run 아이콘인  를 클릭한다. 프로그램이 실행되고, 브레이크포인트에서 제대로 멈추어야 한다. 소스창의 녹색 화살표는 다음에 실행되어야 할 코드를 가리킨다. 프로그램이 실행된 후, 브레이크포인트에서 멈추었으므로 Watch 창의 counter값은 1이어야 한다.



소스코드에서 코드 한 줄씩 만 실행시키고자 할 때는 Step Into 메뉴를 실행한다. 툴바의 아이콘  를 실행시키면 된다. 한 줄의 코드가 실행될 때마다, 녹색 화살표는 다음 실행될

코드로 이동한다. 위의 상태에서 프로그램을 Run 메뉴를 통해 계속 실행해볼 수 있다. 계속 실행 시킬 경우, 매번 브레이크포인트에서 프로그램의 실행이 멈추면서 counter는 1씩 증가한다. 이렇게 Watch창을 통해, 예제1이 제대로 빌드 되었고, MPLAB SIM에서 제대로 동작하는 것을 확인할 수 있다.



3.2.8 맵(Map) 파일과 리스팅(Listing) 파일

맵 파일(example1.map)과 리스팅 파일(xample1.lst)들은 프로젝트 빌드 후, 프로젝트 디렉터리에 위치한다. 이 두 파일은 *File> Open* 메뉴로 열어 볼 수 있다. 이 두 파일은 디버깅에 유리한 추가적인 정보를 제공한다. 예를 들어 변수가 할당된 위치를 보여주거나, C코드에 상응하는 어셈블리 코드를 보여준다. 예제의 맵 파일은 변수 counter가 메모리 주소 0x8a에 저장된 것을 보여준다.

예3-1: 맵 파일

```
Symbols - Sorted by Name
    Name    Address     Location     Storage File
---------  ---------  ---------  --------- ---------
  counter    0x00008a    data         extern
:c\mcc18\example\getting_started\example1\example1.c
```

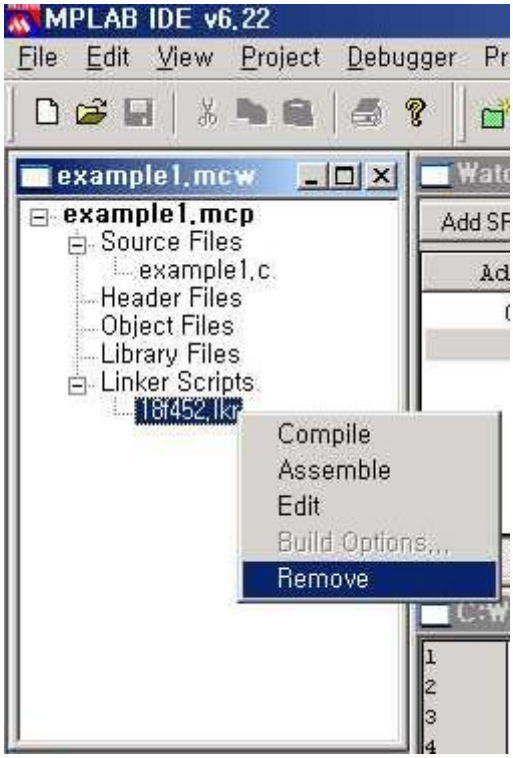
리스팅 파일은 C 소스코드와 이에 상응하는 기계어 코드의 리스트를 보여준다. 각 코드에 따른 주소, 기계어 값 그리고 디스어샘블리 코드(disassembly code)를 볼 수 있다. 아래의 리스팅 파일은 보기 편하도록 일부 수정되었다.

예 3-2: 리스팅 파일

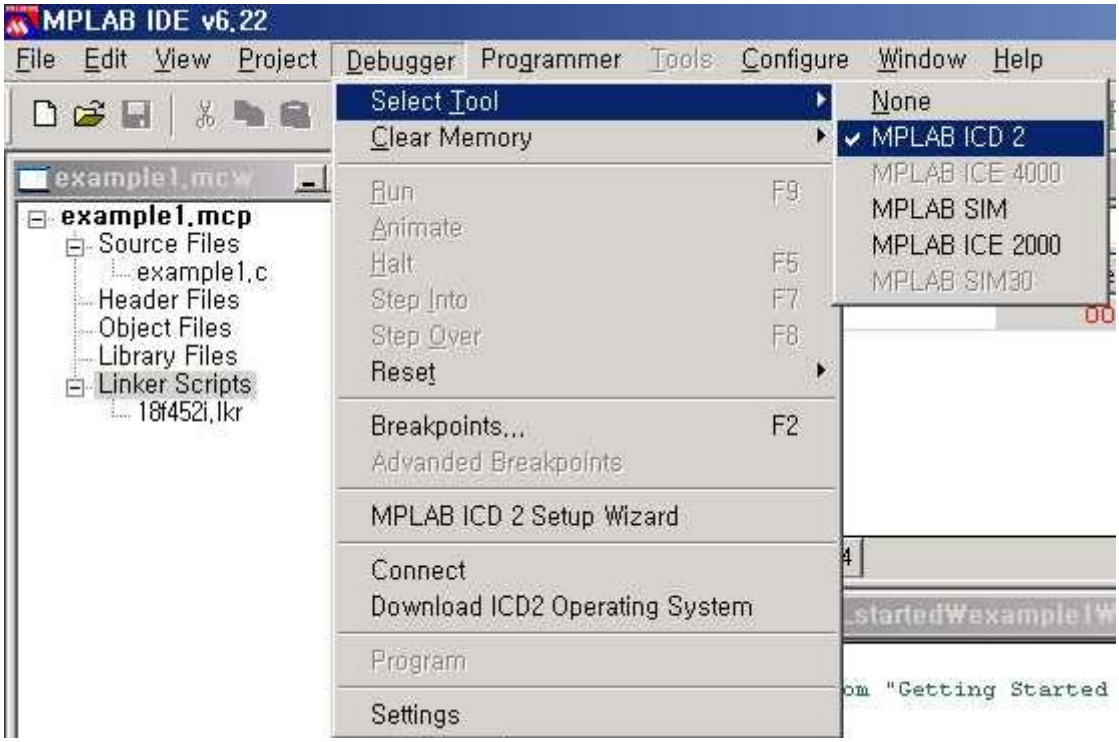
```
Address    Value     Disassembly               Source
-------- --------  -----------------  -----------------------------
#include <p18cxxx.h>  /* for TRISB and PORTB declarations */
                                          int counter;
                                          void main (void)
                                        {
0000e2   0e01    MOVLW    0x1              counter = 1;
0000e4   0100    MOVLB    0x0
0000e6   6f8a    MOVWF    0x8a,0x1
0000e8   6b8b    CLRF     0x8b,0x1
0000ea   6a93    CLRF     0x93,0x0          TRISB = 0;          /*
configure PORTB for output */
C:\mcc18\example\getting_started\example1\example1.c
0000ec   518b    MOVF     0x8b,0x0,0x1      while (counter <= 15)
0000ee   0a00    XORLW    0x0
0000f0   aee8    BTFSS    0xe8,0x7,0x0
0000f2   d002    BRA      0xf8
0000f4   358b    RLCF     0x8b,0x0,0x1
0000f6   d005    BRA      0x102
0000f8   0e0f    MOVLW    0xf
0000fa   80d8    BSF      0xd8,0x0,0x0
0000fc   558a    SUBFWB    0x8a,0x0,0x1
0000fe   0e00    MOVLW    0x0
000100   558b    SUBFWB    0x8b,0x0,0x1
000102   e306    BNC      0x110
00010e   d7ee    BRA      0xec
                                        {
000104   c08a    MOVFF    0x8a,0xf81          PORTB = counter;   /*
display value of 'counter' on the LEDs */
000106   ff81
000108   2b8a    INCF     0x8a,0x1,0x1        counter++;
00010a   0e00    MOVLW    0x0
00010c   238b    ADDWFC    0x8b,0x1,0x1
                                        }
000110   0012    RETURN    0x0          }
```

3.2.9 MPLAB ICD2

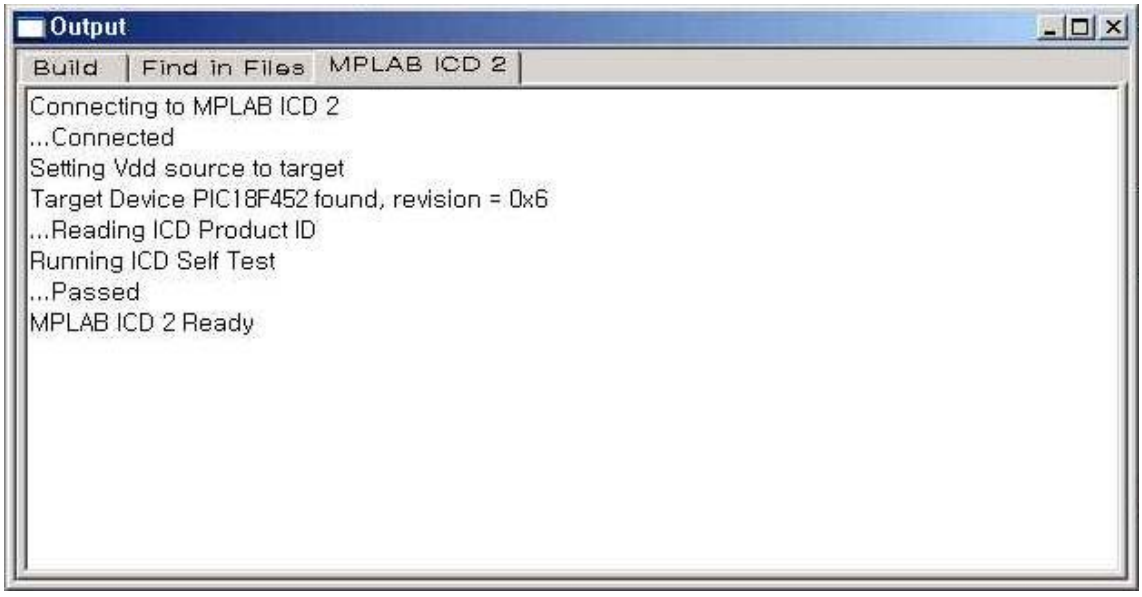
예제의 프로그램은 MPLAB ICD2 인-서킷 디버거에서도 실행시킬 수 있다. ICD2를 이용하기 위해서는 실제 디바이스를 가진 데모보드가 필요하다. 여기서는 마이크로칩사에서 나온 PICDEM2 Plus 데모보드를 이용하여 예제 프로그램을 실행하는 것에 대해서 설명한다. ICD2는 실제 디바이스에서 프로그램을 실행, 프로그래밍 및 디버깅을 할 수 있도록 해준다. ICD2를 사용하기 전, *MPLAB ICD2 Quick Start Guide*를 참조하여 데모보드, ICD2 및 PC를 연결한다. MPLAB SIM에서 실행했던 예제의 프로그램을 ICD2에서 실행하기 위해서는 다시 빌드를 해주어야 한다. 기존 링크 스크립트인 18f452.lkr을 제거하고 ICD2용 링크 스크립트인 18f453i.lkr을 프로젝트에 추가해야 한다. 이를 위해서 아래와 같이, 프로젝트창에서 마우스 우측버튼을 이용 링커 스크립트를 제거한다.



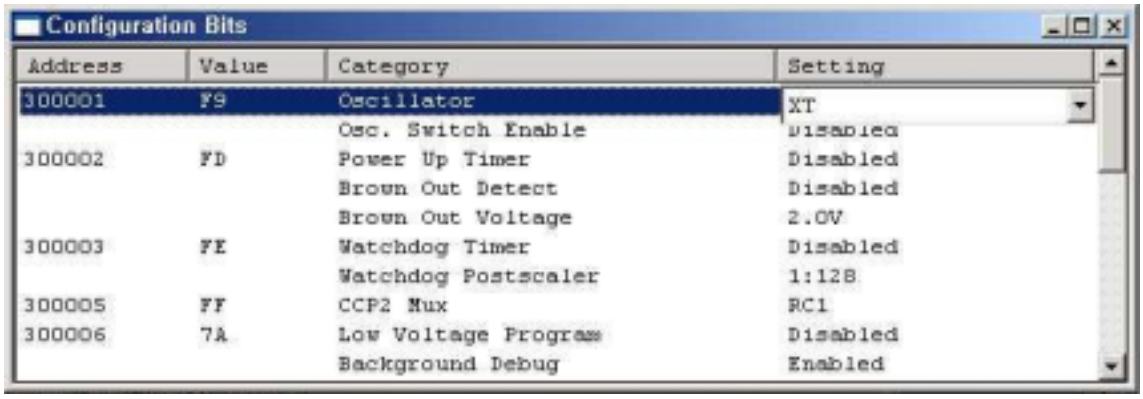
ICD2용 링커 스크립트인 18f452i.lkr을 lkr 디렉터리로부터 프로젝트 창의 Linker Scripts에 더한다. 18f453i.lkr은 링커에게 ICD2에 필요한 메모리를 할당토록 한다. 링커 스크립트 교체가 끝났으면 ***Project>BuildAll*** 메뉴로 프로젝트를 다시 빌드 한다. 빌드 후 ***Debugger>Select Tool>MPLAB ICD2*** 메뉴를 이용하여 MPLAB ICD2를 툴로 선택한다.



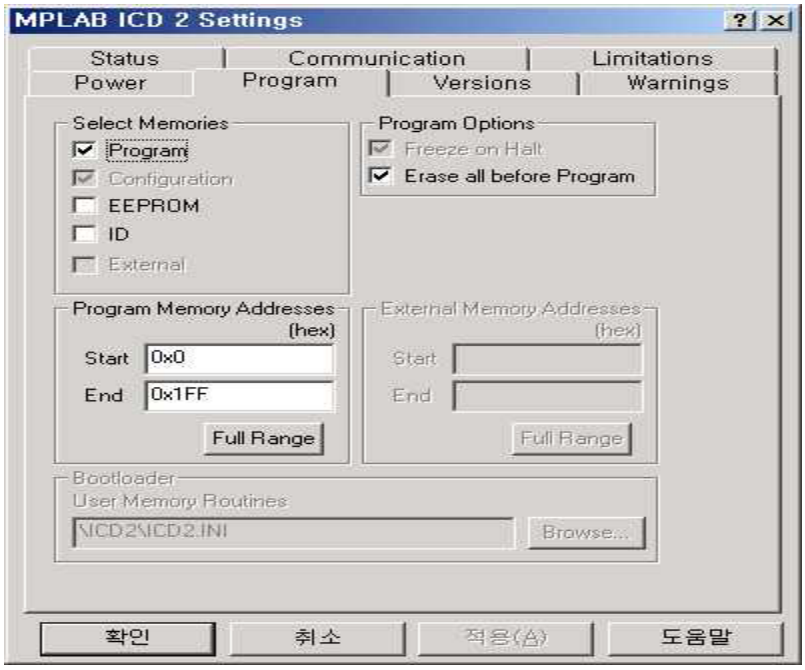
툴을 선택한 후, ICD2를 MPLAB IDE와 연결하기 위해 Debugger>Connect 메뉴를 실행한다. ICD2가 제대로 연결이 되어 있으면 다음과 같은 메시지가 Output 창에 뜰 것이다.



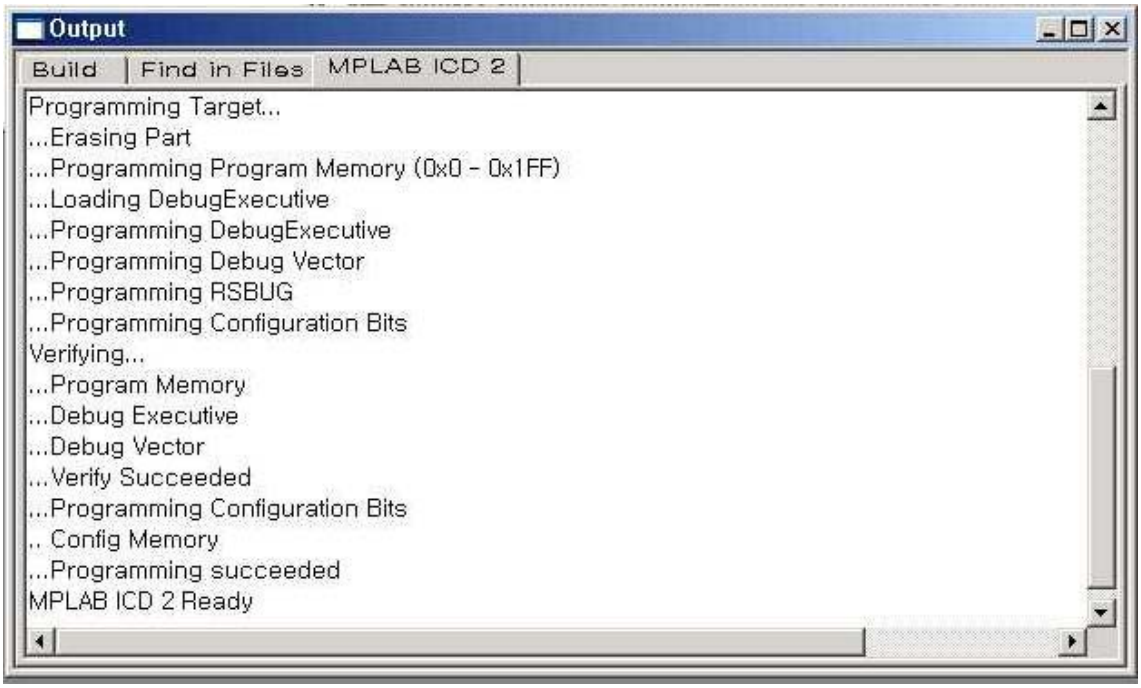
연결이 제대로 되지 않으면 ICD2 매뉴얼을 참조하여, ***Debugger>Connect*** 메뉴를 통해 재연결을 해본다. ICD2의 연결이 성공적으로 되었으면 ***Configure>Configuration Bits*** 메뉴를 실행하여 PIC18F452의 컨피규레이션 비트를 설정한다. ICD2를 디버깅 용도로 사용할 것이면, 컨피규레이션 비트에서 Watchdog timer와 Low Voltage Programming을 disable로 만든다. 또한 Oscillator는 XT, Background Debugging은 enable로 설정한다. 아래의 화면과 같이 설정하면 된다.



디바이스 컨피규레이션 및 ICD2에 대해서는 각각의 매뉴얼을 참고하면 된다. 컨피규레이션 설정이 끝났으면, 프로그래밍을 하기 전, 프로그래밍할 영역을 설정한다. 그냥 프로그래밍을 진행하면 MPLAB IDE는 PIC18F452의 경우 0번지에서 0x7DBF번지를 프로그래밍 한다. 이렇게 되면 프로그래밍 시간이 많이 걸린다. 예제1의 경우 빌드 된 프로그램의 크기가 0x1FF 이하이므로, ***Debugger>Setting*** 메뉴의 Program 탭에서 프로그래밍하는 영역을 0~0x7DBF에서 0~0x1FF로 변경한다. 이렇게 함으로서 프로그래밍 시간이 대폭 줄어든다. 주의할 점은 프로그래밍 영역을 실제 빌드 된 프로그램 크기보다 크게 잡아야 한다는 것이다.



다음에는 실행할 프로그램을 디바이스에 프로그래밍해야 한다. 디바이스를 프로그래밍 하기위해서는 ***Debugger>Program*** 메뉴를 실행한다 이 메뉴를 실행하면 디바이스에 프로그래밍을 한다. 프로그래밍이 성공적으로 끝나면 다음과 같은 메시지가 Output창에 뜬다.



프로그래밍이 끝났으면 *Debugger>Run*메뉴나 Run 아이콘으로 프로그램을 실행시킨다. 그러면 예제1 프로그램이 디버깅 모드에서 실행이 된다. MPLAB SIM에서 소스파일 중에 브레이크포인트를 잡았으므로 프로그램 실행은 이 브레이크포인트에서 정지하게 된다. 이미 PICDEM2 Plus의 LED를 키는 코드는 실행이 되었으므로, 데모보드의 LED가 1개 켜질 것이다. 계속 Run을 실행시키면, LED가 바이너리로 하나씩 증가한 값을 표시할 것이다.