부록 7. MPLAB IDE v.8.83 의 설치 및 간단한 사용방법

- 마이크로프로세서설계실습에서 사용하게 될 MPLAB IDE v8.83의 간단한 설치 방법에 대해서 알아보자.

1. MPLAB IDE의 다운로드

일단 아래의 주소를 클릭하면 그림 1.1과 같은 창이 열릴 것입니다. 여기서 오른쪽 스크롤 바를 아래로 끝까지 내리시면 그림 1.2와 같은 항목이 나타난다.

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002#P173_5150

[그림 1.1] MPLAB IDE v8.83을 다운받을 수 있는 웹사이트

[그림 1.2] 그림 1.1의 하단 부

(a) (b) (c)

[그림 1.3] 다운로드 받는 단계

그림 1.2의 붉은 색 박스 부분에 있는 MPLAB IDE v8.83을 클릭하면 그림 1.3의 (a)와 같은 창이 뜨고, 저장 버튼을 누르면 그림 1.3의 (b)와 같이 바탕화면을 저장 위치로 선정한 다음 다시 저장 버튼을 누르면 프로그램이 다운로드 된다. 프로그램의 다운이 완료되면 바탕화면에 그림 1.3의 (c)와 같은 아이콘이 나타난다. 다운로드가 끝나면 다운받은 파일의 압축을 풀고 프로그램을 설치하면 된다.

2. MPLAB_IDE 압축 풀기 및 프로그램 설치

바탕화면에 생성된 MPLAB_IDE_8_83.zip 파일을 더블클릭하면 그림 1.4의 (a)와 같은 창이 열리고 이때 압축풀기 항목을 선택하면 그림 1.4의 (b)의 폴더가 바탕화면에 생성됩니다. 생성된 폴더 안에는 그림 1.4의 (c)에서와 같이 5개의 파일이 있습니다. 이중 setup.exe 파일을 실행 시킵니다.

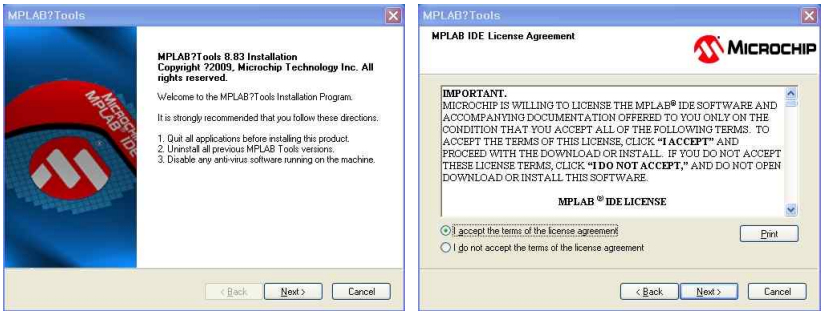
(a) (b) (c)

[그림 1.4] 압축 파일 푸는 단계

그러면, 그림 1.5와 와 같이 MPLAB® CERTIFIED라는 로그와 Install Shield Wizard라는 창이 실행됩니다.

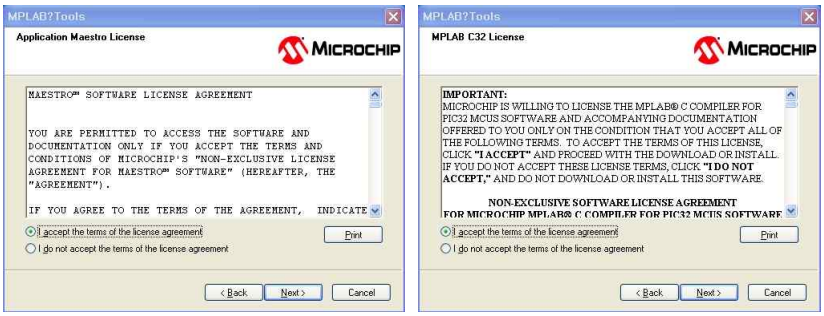
(a) 프로그램 설치 단계-1

Install Shield Wizard가 수행이 끝나면 아래의 그림 1.6과 같은 창이 뜬다. 여기서 부터는 Next와 I accept the terms of the license agreement를 선택한 후 Next를 선택한 후 Setup Type을 Complete로 설정하고 계속해서 Next를 누른다. Choose Destination Location 창에서 프로그램을 설치할 위치를 정한다. (여기서는 그냥 default값을 그대로 사용함.) 다시 Next를 누르면 Application Maestro License 창이 뜨고 역시 I accept the terms of the license agreement을 선택한 다음 Next를 누른다. 다음에 MPLAB C32 License 창에서도 역시 I accept the terms of the license agreement를 선택한 후 Next를 누른다.

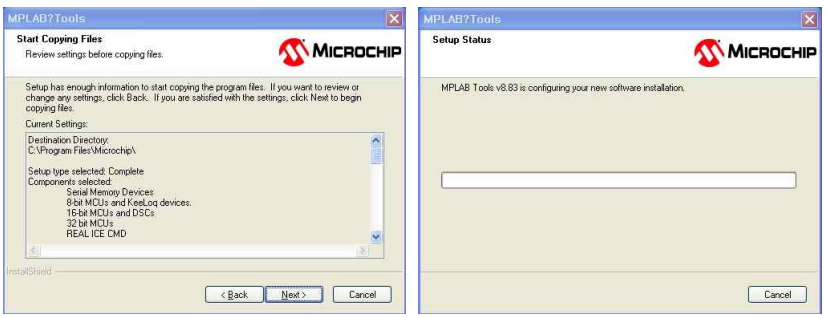


(b) 프로그램 설치 단계-2

(c) 프로그램 설치 단계-3



(d) 프로그램 설치 단계-4

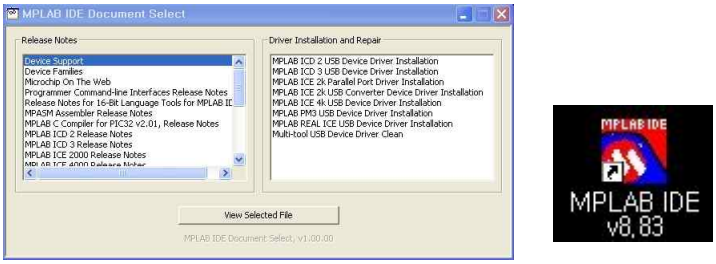


(e) 프로그램 설치 단계-5

다시 Start Copying Files 창이 뜨면 역시 Next를 누르고 나면 Setup Status 창이 뜨고 프로그램이 다 설치되면 HI-TECH C를 설치할 것인지 말 것인지 확인하는 창이 뜬다. 여기서 그냥 아니오(N)을 선택한다. (물론 예를 선택하면 HI-TECH C를 설치함.) 이제 프로그램 설치의 마지막으로 Install Shield Wizard Complete라는 창이 뜨면 Yes, I want to restart my computer now를 선택한 후 Finish를 누른다. 이렇게 하면 컴퓨터가 재부팅되고 바탕화면에 MPLAB IDE v.8.83이라는 아이콘이 생성되고, MPLAB IDE Document

Select 창이 뜨는데 그냥 무시하고 창을 닫는다. 이제 MPLAB IDE v.8.83이 설치가 다 끝났다.(설치 방법그림 1.5의 단계-1부터 7까지 절차 참고)

(f) 프로그램 설치 단계-6

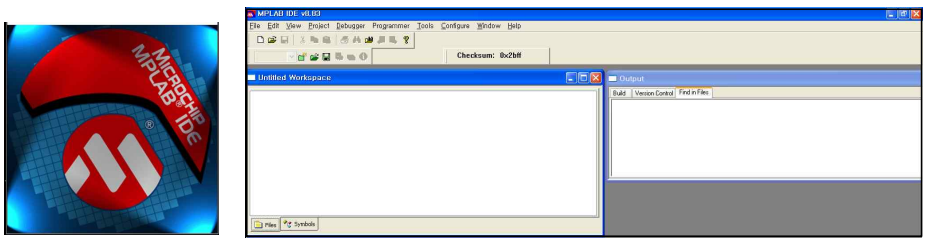


(g) 프로그램 설치 단계-7

[그림1.5] 프로그램 설치 단계

3. 프로그램 실행

바탕화면에 생성된 MPLAB IDE v.8.83 아이콘()을 더블클릭한다. 그림1.6의 (a)와 같은 아이콘이 생성된 후 MPLAB IDE v8.83이라는 창이 뜨고 창에는 Untilted Workspace 창과 Ouput 창이 생성되어 있다.



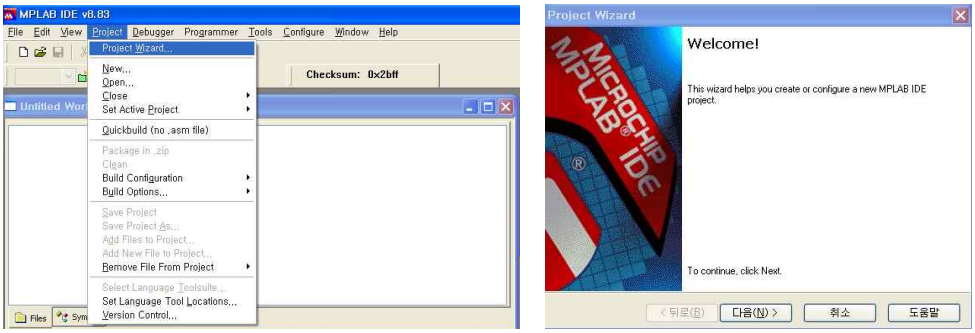
(a) 실행 대기 아이콘 (b) 프로그램 초기 화면

[그림 1.6] 프로그램 실행 초기화면

(1) 프로젝트 생성하기

프로젝트를 생성하기 위해서는 몇 가지 방법이 있지만 여기서는 Project Wizard를 이용해서 프로젝트를 만들어 보자. 아래와 같이 메뉴 바에서 Project>>Project Wizard..를 선택한

다. Project Wizard 창이 뜨면 다음을 클릭한다.



(a) Project Wizard 선택 화면 (b) Project Wizard 창

[그림 1.7] 프로젝트 설정 초기 화면

⊡ 단계 1 : Step One ☞ Select a device : PIC16F873 또는 PIC15F873A 선택 후 다음 클릭

[그림 1.8] 사용하려고 하는 소자를 선택하는 단계

⊡ 단계 2 : Step Two ☞ Select a language toolsuite

그림 1.9의 프로젝트 설정2 단계 (b)의 Active Toolsuite 항목을 클릭해보면 여러 가지 프로그램 언어를 선택할 수 있다. 마이크로프로세서설계 실험에서는 어셈블리어로 프로그램을

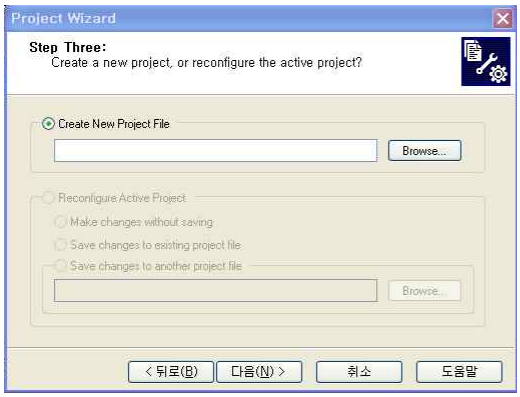
(a) 프로젝트 설정 단계 2 (b) 프로젝트 설정 단계 2

[그림 1.9] 프로젝트 설정 단계 2

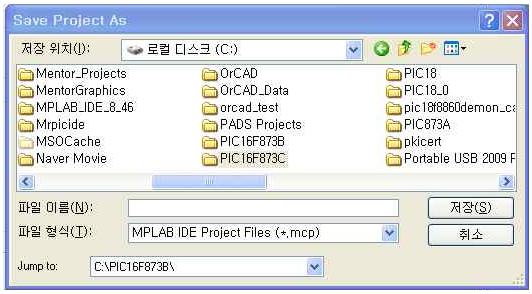
작성함으로 Microchip MPASM Toolsuite를 선택한다. 이때 컴파일러는 MPASM

Assembler(mpasmwin.exe) v5.43을 선택한 후 다음을 클릭 한다.

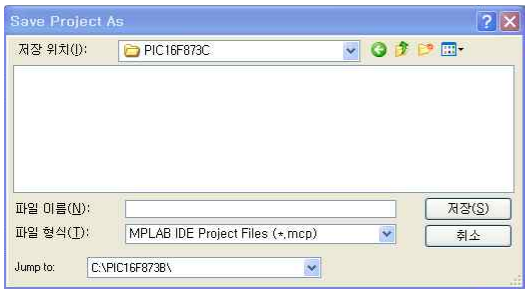
⚄ 단계 3 : Step Three ☞ Create a new Project, or reconfigure the active project?
이 단계에서 새로운 프로젝트를 생성 한다. 이때 Browse..를 클릭하여 새로운 폴더를 만들거나 만들어진 폴더를 선택 한다. 여기서는 미리 만들어 놓은 C:₩PIC16F873C를 더블클릭하여 선택한 다음 프로젝트 이름을 파일 이름(N)에 작성한다. 본 설명에서는 led_test.mcp라고 작성하고 저장 버튼을 누른다. 그러면 그림 1.10 프로젝트 설정 단계 3의 (e)와 같이 프로젝트가 생성된다.



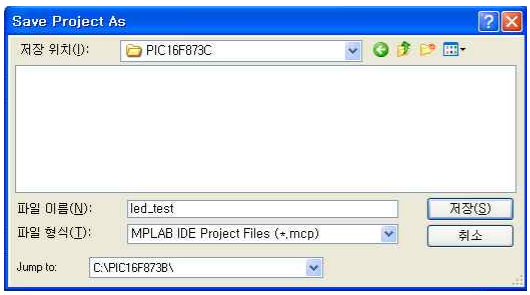
(a) 프로젝트 설정 단계 3



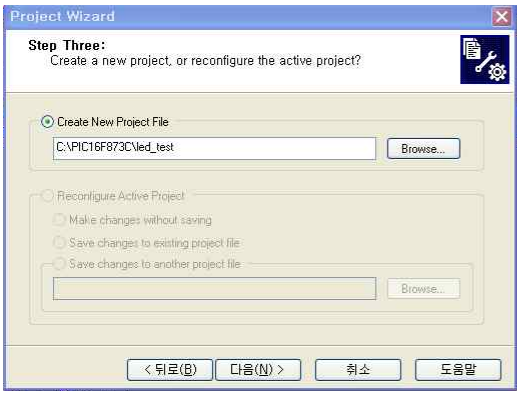
(b) 프로젝트 설정 단계 3



(c) 프로젝트 설정 단계 3



(d) 프로젝트 설정 단계 3



(e) 프로젝트 설정 단계 3

[그림 1.10] 프로젝트 설정 단계 3

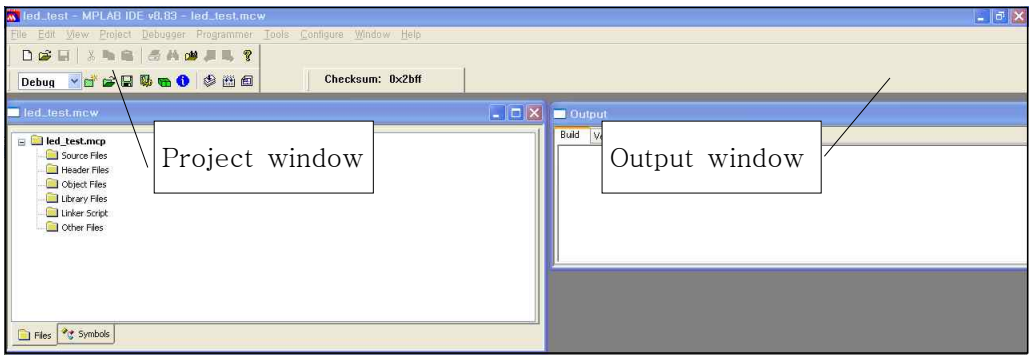
단계 3이 완료되면 다음을 누른다.

⚀ 단계 4 : Step Four ☞ Add existing files to your project.

(a) 프로젝트 설정 단계 4 (b) 프로젝트 설정 단계 4

[그림 1.11] 프로젝트 설정 단계 4

현재 PIC16F873C 폴더에는 어떤 파일도 존재 하지 않으므로 다음을 선택한다.(추가할 project 파일이 없으므로 그냥 다음을 누르고, 추가할 파일 폴더 내에 있다면 선택하면 된다.) 최종적으로 Summary 창이 나타나고 선택한 Device와 선택한 프로그램 언어 및 프로젝트 파일 이름이 완성된 것을 확인할 수 있다. 이제 마침을 누르면 새로운 프로젝트가 생성 완료된다.



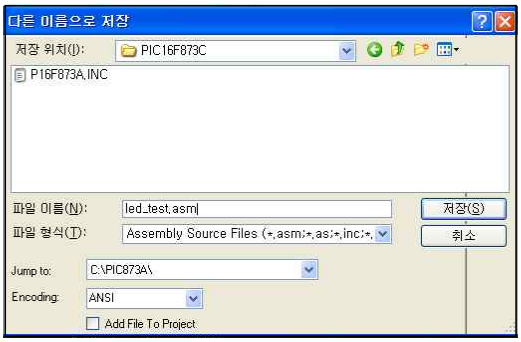
[그림 1.12] 완성된 프로젝트 화면

(2) 소스 코드 작성 및 등록하기

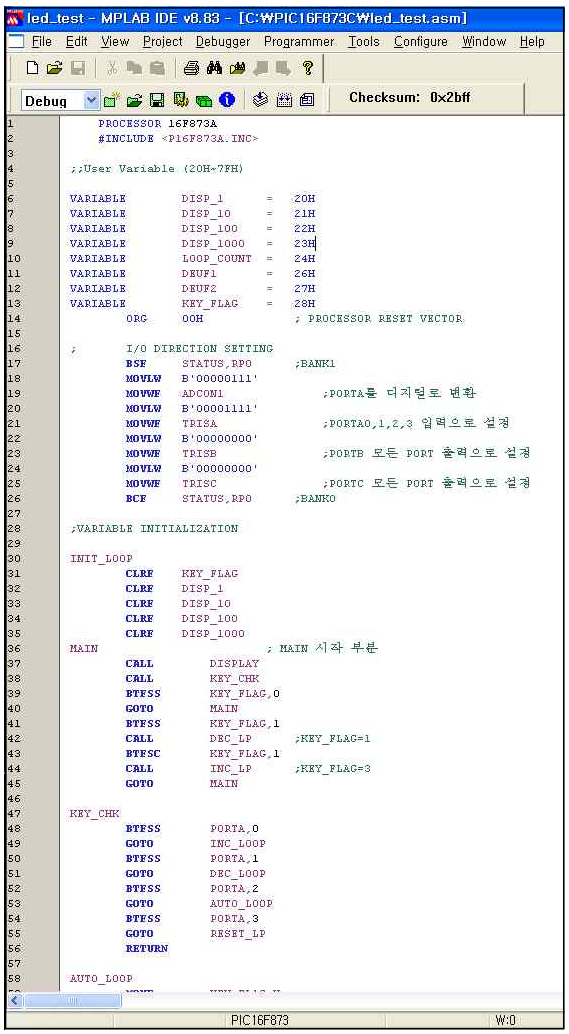
이제 프로젝트를 생성하였으니 프로젝트 내에 포함될 소스 파일을 작성해보자. 소스 파일을 작성하기위해서 메뉴바의 File>>New를 누르거나 (CTRL+N or 툴바에서 New File 아이콘(🗋)을 클릭 한다. 그러면 Untitled 라는 창이 열리게 되며 여기에 프로그램을 작성하면 된다.

프로그램을 작성하기 시작하면 툴바의 Save File 아이콘(🖫)이 활성화 된다. 프로그램을 다 작성하였으면 작성된 프로그램을 저장하자.(파일 작성한 파일을 Source Files에 추가 하면 모든 것이 끝난다.) 파일 이름을 led_test.asm으로 변경하기 위하여 메뉴바에서 File>>Save As..를 선택 하면 그림 1.13와 같은 창이 열리고 파일 이름(N)란에 led_test.asm을 적고 저

장한다(물론 여기서 폴더의 위치는 자신이 원하는 곳을 지정 하면 되나 여기서는 미리 만들어 둔 PIC16F873C 폴더를 이용함). 그러면 그림 1.14과 같이 작성된 프로그램이 나타난다. 이제 작성된 프로그램을 만들어 놓은 프로젝트(led_test.mcp)에 등록시켜야 한다.



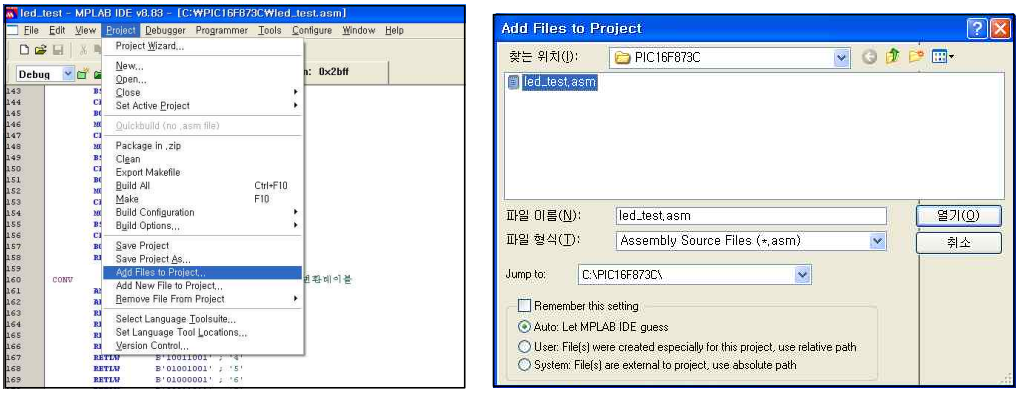
[그림 1.13] 다른 이름으로 저장하기



[그림 1.14] 소스파일 윈도우

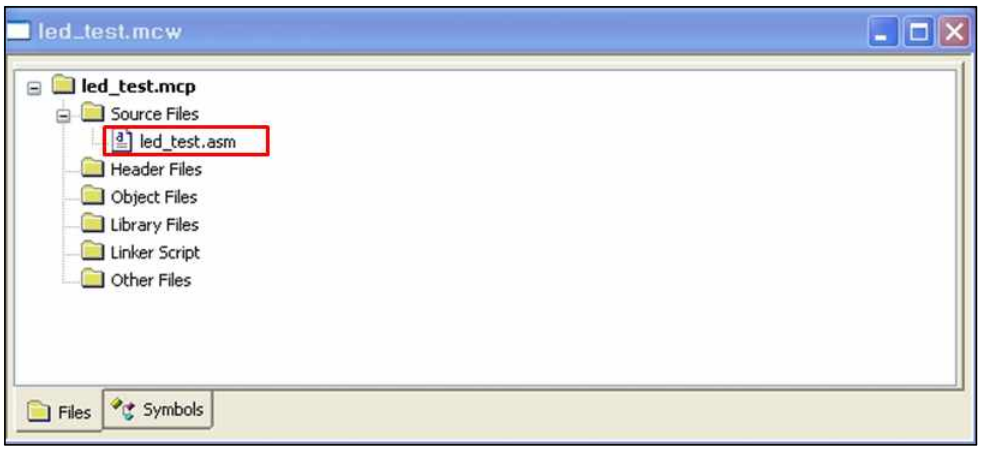
프로젝트에 소스를 등록하기 위해서는 그림 1.15의 (a)와 같이 메뉴 바의 Project>>Add

File to Project..를 선택한다. 그러면 그림 1.15의 (b)와 같은 창이 뜨고 작성된 파일을 클릭 하여 프로젝트에 등록 시킨다. 이제 열기를 클릭한다. 그러면 작성된 파일이 프로젝트에 등록이 끝났다. 그림 1.16의 Project window에 Source Files 폴더 아래에 led_test.asm이 등록된 것을 확인할 수 있다.



(a) (b)

[그림 1.15] 프로젝트에 소스파일 등록하기

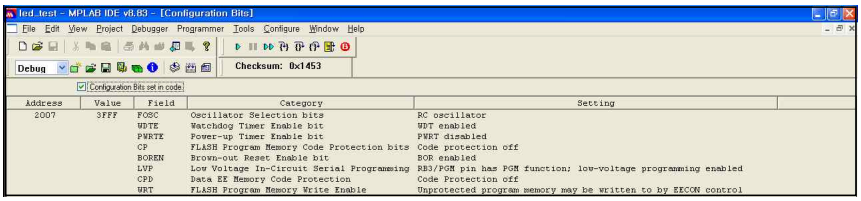


[그림 1.16] 프로젝트에 등록된 소스파일 확인

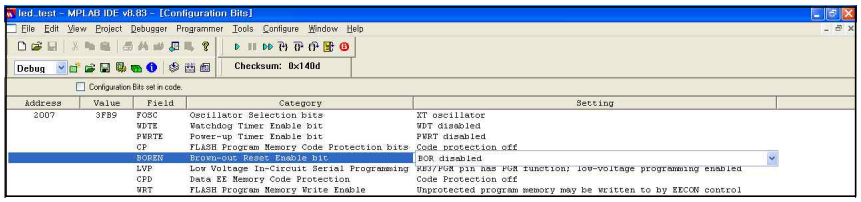
(3) Configuration Bits 설정하기

다음은 Configuration Bits 설정을 위해 그림 1.17과 같이 메뉴 바에서 Configure>>Configuration Bits.. 항목을 선택하면 그림 1.18와 같은 창이 열린다. 여기서 Configuration Bits set in code의 체크를 해제한 후 FOSC 설정을 XT oscillator로, WDTE를 WDT disable로 설정한다. (물론 Watch dog 기능을 사용하려면 enable로 설정해야한다.) Brown-out Reset Enable bit는 disable하고 나머지 Configuration Bit들도 사용목적에 따라 설정하면 된다. 여기서 WDT와 Brown-out Reset은 추후에 설명하기로 한다. (꼭 Brown-out Reset를 여기서 disable 할 필요는 없다.)

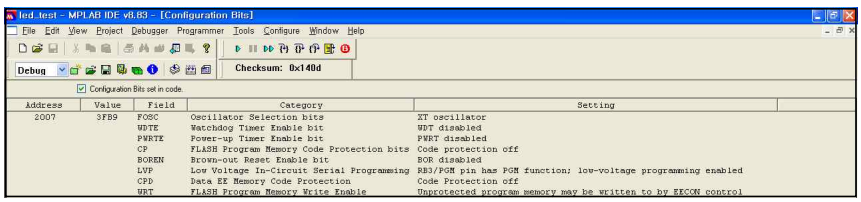
(a) Configuration Bits 설정을 위한 메뉴바 보기



(b) Configuration Bits 설정을 위한 메뉴바-설정 전



(c) Configuration Bits 설정을 위한 메뉴바-설정 중



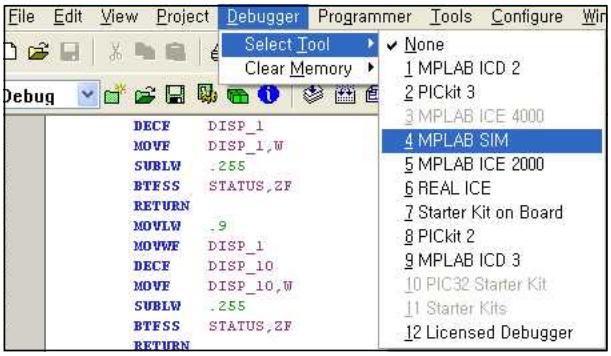
(d) Configuration Bits 설정을 위한 메뉴바-설정 후

[그림 1.17] Configuration Bits 설정을 위한 메뉴바 관련 화면

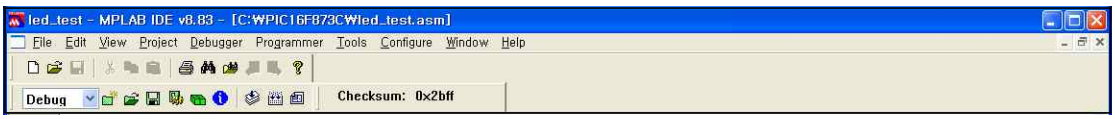
Configuration Bits 설정에 대한 구체적인 의미는 초보자에게는 앞의 내용으로 하면 되나, 완전한 상품으로 프로그램을 구현하기 위해서는 매우 종요한 하드웨어적인 초기 설정 조건이 되며, 앞으로 필요에 따라서 추가 설명을 할 것이다.

(4) 디버깅 환경 설정하기

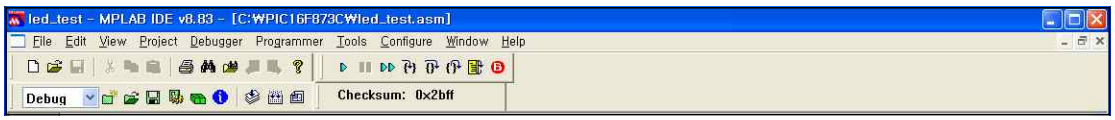
이제 작성된 파일이 에러 없이 작성된 것인지를 확인하기 위해서 Debugger 모드를 선택해야 한다. 현재 디버그 장비가 연결되어있지 않음으로 그림 1.18와 같이 메뉴 바의 Debugger>>Select Tool>>4 MPLAB SIM을 선택한다. 그러면 그림 1.19와 같은 화면이 나타난다. 이 모드를 설정하기 전과 후의 툴바 모양이 달라 진 것을 확인할 수 있다. 즉 디버그 전용 툴바( )가 새롭게 생성되었다. 이 툴바를 프로그램의 디버깅에 사용한다. 툴바의 상세한 설명은 **4. 프로그램 실행시키기**를 참조하자.



[그림 1.18] 디버그 모드 설정



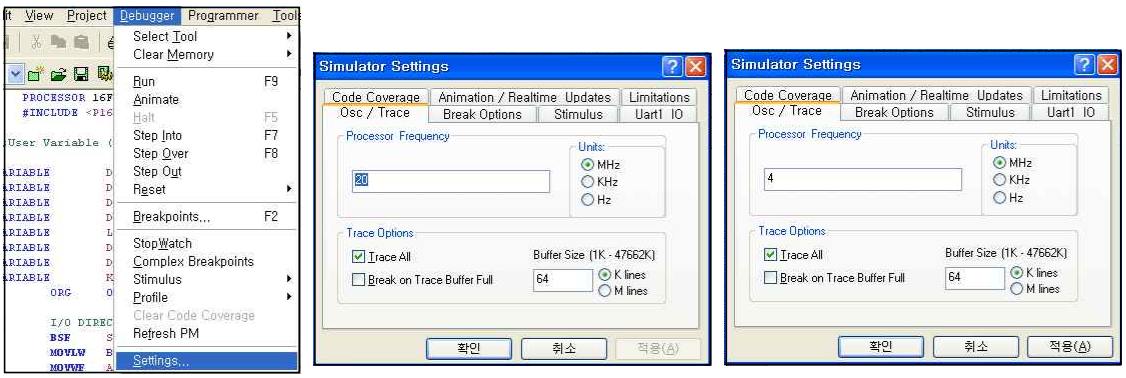
(a) 툴바의 모양 변화 - 설정 전



(b) 툴바의 모양 변화 - 설정 후

[그림 1.19] 툴바의 모양 변화 변화

이제 PIC16F873 마이크로프로세서에 입력될 클럭 주파수를 설정하자. 클럭 주파수를 설정하기위해서는 역시 그림 1.20의 (a)와 같이 Debugger >> Settings..를 선택한다. 그러면 그림 1.20의 (b)와 같은 창이 열리고 이때 클럭을 설정하기 위해서는 Osc/Trace 항목을 선택한다. 기본 값으로 20MHz가 선정 되어 있다. 이것을 4MHz로 수정하면 된다.(그림 1.20의 (c) 참고)

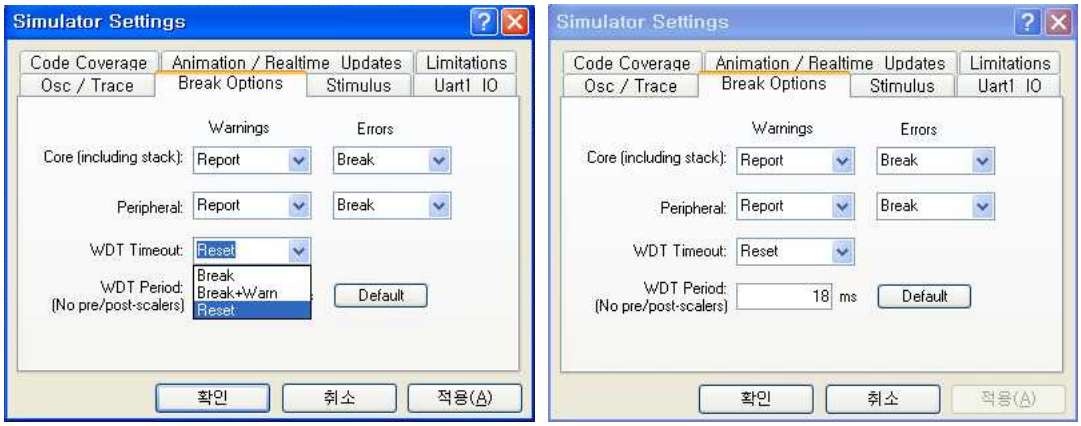


(a) Debugger설정을 위한 메뉴 (b)Simulator Settings 창 (c)

[그림 1.20] 마이크로프로세서 동작을 위한 클럭 주파수 선택 방법

디버그 설정에서 추가적으로 한 가지 더 설정하자. 그림 1.20의 (c)에서 Break Options를 선택한 후 WDT Timeout을 Reset으로 변경한다. WDT의 경우 Configuration 항목에서 설

정 및 해제를 할 수 있다. 당연하게 Configuration에서의 설정이 이곳에서 설정된 것 보다 우선한다. 만약 Configuration 설정에서 WDT가 enable되었다면 18msec 마다 WDT가 마이크로프로세서를 리셋시킬 것이며 따라서 프로그램이 정상적으로 진행하지 못한다.(그림 1.21 참고)

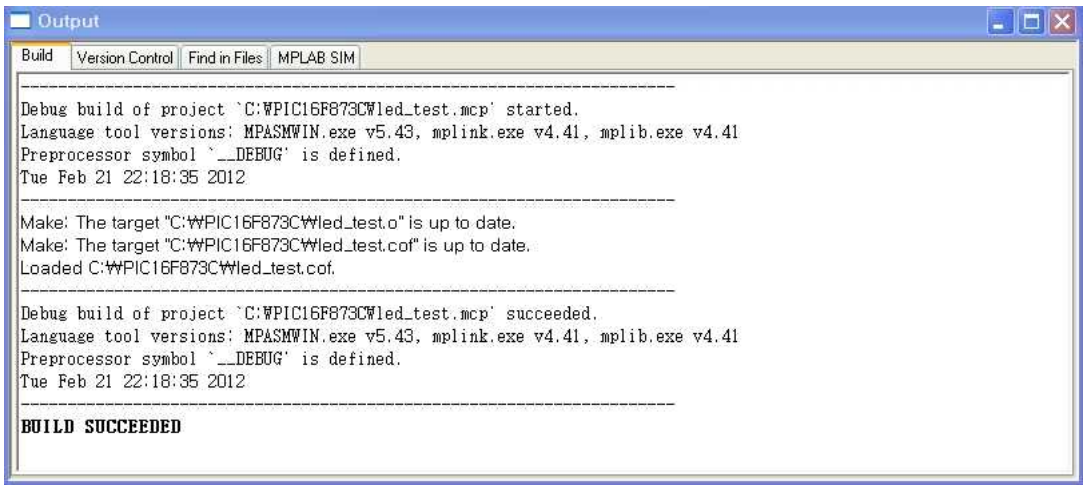


(a) WDT Timeout 설정- Reset (b) WDT Period 확인 - 18msec

[그림 1.21] WDT Timeout 설정

(5) 컴파일 하기

작성된 led_test.asm을 컴파일 하기위해서 툴바에서  (make) 항목을 누르거나 Project>>Make를 누르거나 키보드에서 F10 키를 누르면 컴파일 된다.



[그림 1.22] 컴파일 에러가 없는 경우

만약 에러가 없다면 BUILD SUCCEEDED라는 메시지가 그림 1.22와 같이 Output window에 나타난다. 만약 에러가 발생한다면 BUILD FAILED이라는 메시지가 Output window에 나타난다. Output window는 소스를 컴파일 한 후 그 결과를 사용자에게 알려주며 주로 디버깅시 에러의 위치 및 종류를 표시해주는 역할을 한다.

그림 1.23에 에러가 경우에 대해서 Output window의 출력 결과를 나타내었다. 그림 1.23

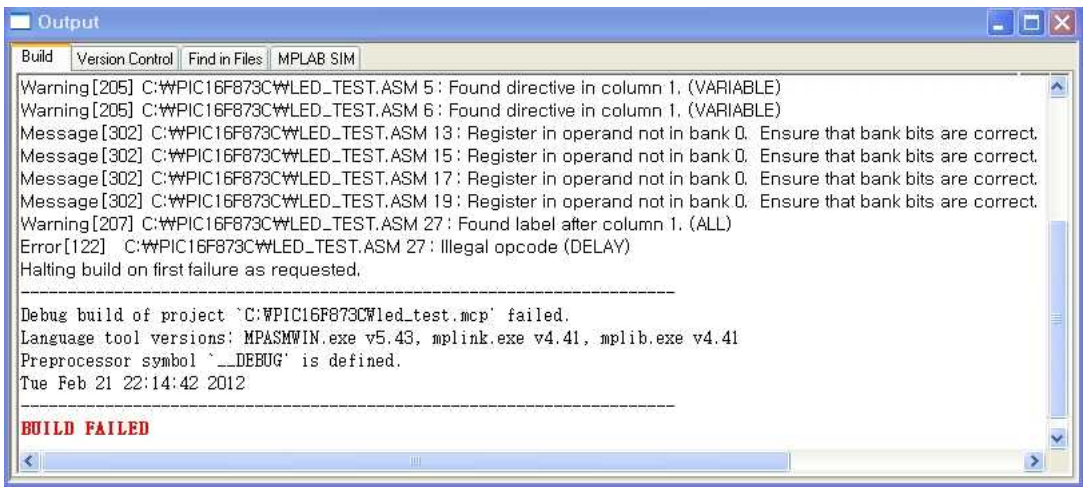
의 에러로 표시된 부분의 내용을 읽어보면,

Error[122] C:₩PIC16F873C₩LED_TEST.ASM 27 : Illegal opcode (DELAY)

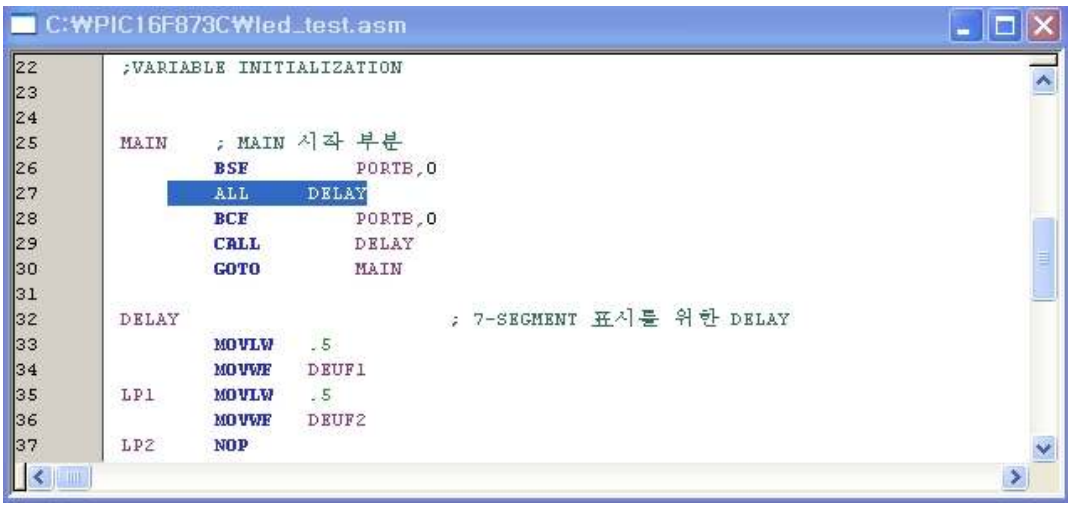
라고 되어 있다. 즉 잘못된 명령을 작성했다는 의미이고 잘못된 프로그램의 위치는 souce program led_test.asm 의 27번째 줄이라는 것을 의미한다. 실제 이 에러는 작성된 소스에서 **27 번째 줄**에서 CALL 대신 ALL로 잘 못 적어 발생된 에러이다. 에러 표시 앞줄에는 또 다른 에러에 대한 힌트를 준다. Warning은 컴파일 에러는 아니다. 그러나 주의할 필요가 있다는 경고이므로 될 수 있는 한 Warning이 없게 프로그램을 작성하는 것이 좋다

Warning[207] C:₩PIC16F873C₩LED_TEST.ASM 27 : Found label after column 1. (ALL)

그림 1.24에 작성된 소스 프로그램 중 잘못 된 부분을 나타내었다. 디버깅시 명령어를 잘못 작성하거나 해서 발생하는 에러들은 이런 방식으로 찾아서 수정하면 된다. 발생되는 에러의 종류는 매우 다양하므로 각자가 프로그램 작성 중 하나하나 알아가길 바란다.

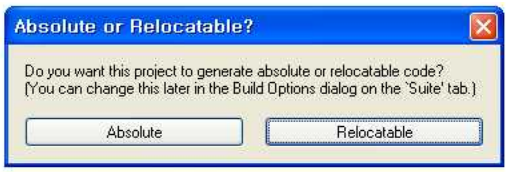


[그림 1.23] 컴파일 에러가 있는 경우



[그림 1.24] 소스에 발생된 에러 확인

참고 1. Absolute or Relocatable? 창
컴파일을 처음 실시 할 경우 아래와 같은 메시지의 창이 나타나는데 여기서는 Relocatable를 선택하면 된다. 메시지의 내용처럼 Absolute를 선택하려면 나중에 Build Options에서 설정을 변경해주면 된다. (메뉴 바에서 Project>>Build Options..>>Project 선택)



[그림 1.25] Absolute or Relocatable? 창

참고 2.  (make)와  (Build All)

☞ Make : 작성한 파일의 에러를 신속하게 확인하는 것.
빌더에서는 1개의 유닛만 compile한 후에 link하여 exe를 만들어 줍니다.
수정한 유닛만 컴파일 다시 하니 전체 과정이 비교적 짧게 걸립니다.

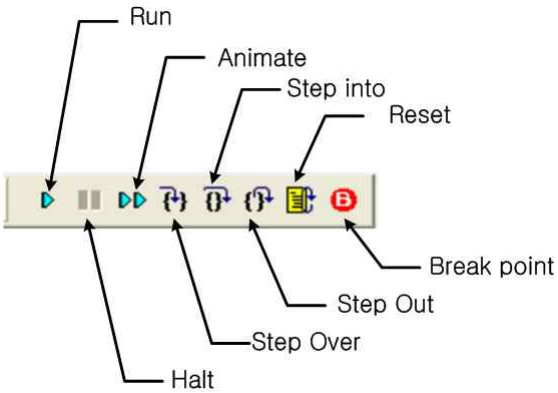
☞ Build : 여러 개의 오브젝트들을 link 시켜주는 것.
이미 만들어져 있는 *.obj 파일을 다 새로 만든 후에 link하여 exe를 만들어 줌. 따라서 전체 다 컴파일을 다시 하니 시간이 많이 걸림

그러나 여러분이 작성하는 프로그램은 길이가 짧은 것들이므로 Build All 하는 것이 바람직함.

4. 프로그램 실행시키기

(1) Debugger icon 사용하여 프로그램 실행 시키기

이제 작성된 프로그램을 실행시켜 보자. 프로그램을 실행시키기 전에 먼저 프로그램을 실행시키기 위한 툴바에 대해서 알아보자. 그림 1.23에 프로그램 실행을 위한 툴바의 그림을 나타 내었다(이하 Debugger를 위한 툴바로 표현).



[그림 1.26] Debugger를 위한 툴바

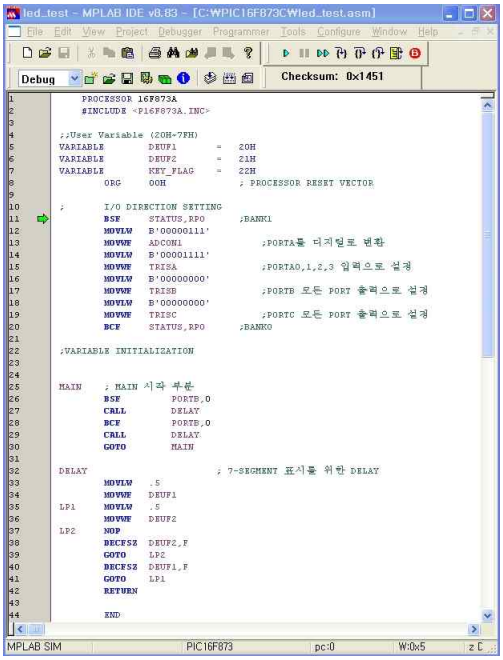
그림 1.26의 Debugger 툴바를 이용하여 프로그램을 실행 시킨다. 또는 메뉴 바의 Debugger 항목을 사용해서 실행 시켜도 되며, 각 아이콘의 의미는 다음과 같다.

표 1.1 프로그램 실행 툴바의 기능 요약

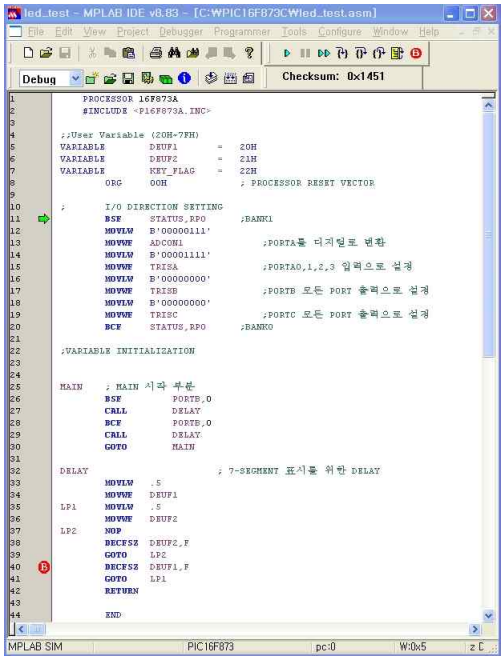
√ Run	프로그램을 수행(F9)
√ Halt	프로그램 수행 중지(F5)
√ Animate	일정한 시간 간격으로 step를 수행. 수행 속도의 조정은 Debugger>>Settings..>>Animation/Realtime Updates에서 조정함.
√ Step into	개별 명령어 한 줄 수행(F7)
√ Step Over	개별 명령어 한 줄 수행(F8) 그러나 CALL명령어는 한꺼번에 수행하고 멈춤 (따라서 sub-routine 내부를 skip할 경우에 사용함)
√ Step Out	CALL 명령어로 작성된 sub-routine에서 빠져 나올 때 사용
√ Reset	CPU를 reset 시킴 프로그램 카운터를 00H로 만듦, 따라서 프로그램을 처음부터 다시 수행 시키려고 할 경우에 사용함.
√ B	Break point. Break point 전 까지 명령어 수행 (break point의 설정은 원하는 소스의 위치에서 더블클릭으로 설정함)

간단하게 위의 아이콘을 이용해 프로그램을 동작시켜 보자 먼저 Reset 버튼()을 눌러 프로그램을 초기 위치에 오게 한다. 이때 좌측에 표시된 연두색 화살표( :이하 커서라 표현함)는 앞으로 수행될 프로그램 위치를 표시 한다. 다음 멈추고 싶은 위치에 Break point ()를 활성화시킨다(원하는 위치에 더블 클릭하면 설정됨). 그러면 프로그램 소스 좌측에 표시가 나타나게 된다.
프로그램을 수행시키는 방법 중 가장 일반적인 것이 Run이다. 즉 이 방법은 여러분이 사용하는 모든 프로그램의 수행방법이다. 그러나 Run는 수행 결과만 볼 수 있으므로 원하는 결과를 얻기위한, 아니 기능을 만들어야하는 개발자의 입장에서는 별로 의미가 없는 수행방법이다. 따라서 프로그램을 개발하는 개발자를 위해서는 프로그램 단계별로 동작 상태를 확인할 필요가 있으며, 따라서 위의 다양한 수행 방법이 필요한 것이다. 따라서 앞으로 여러분이 이 Debugger 사용법을 정확히 그리고 효율적으로 사용할 수 있다면 프로그램 개발 기간을 최대한 단축시킬 수 있을 것이다.

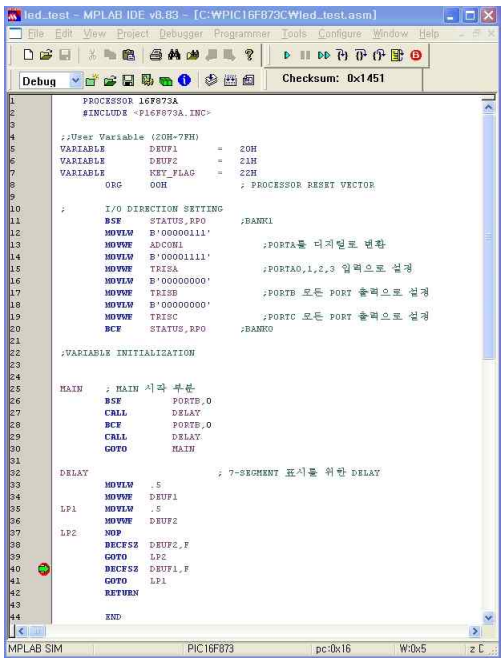
이제 가장 첫 단계로 Run()을 수행 시켜 보자 그러면 Break point가 표시된 위치에서 프로그램이 정지되고 커서가 Break point위치에 머물러 있는 것을 확인 할 수 있다. 다시 Reset를 누르면 프로그램은 다시 처음으로 돌아간다. 이에 대한 프로그램의 실행 상황은 그림 1.27과 같다.



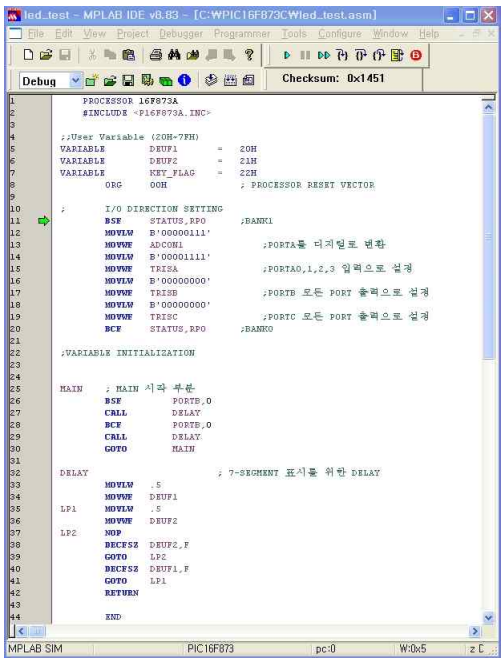
(a) Reset 수행 후 커서 위치



(b) Break point 설정 위치



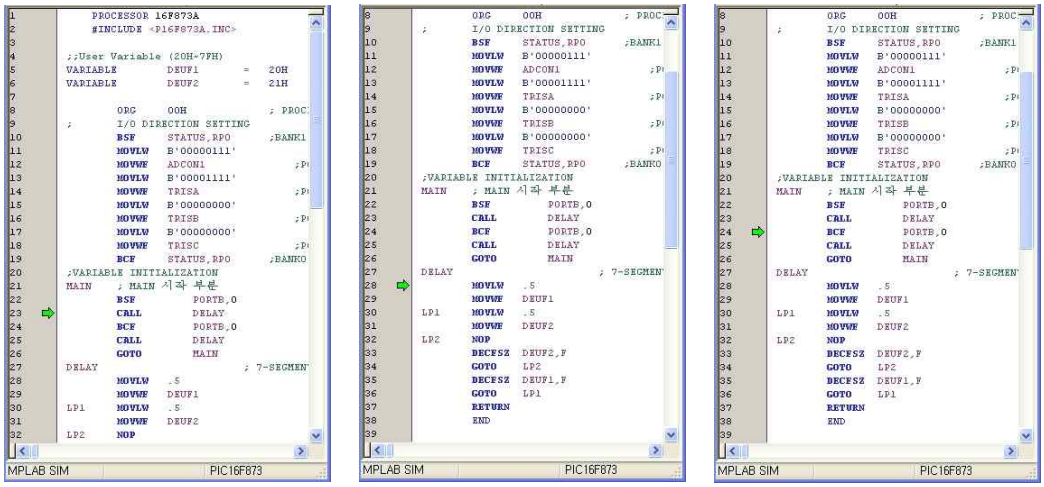
(c) Run 수행 후 커서의 위치



(d) Reset 수행 후 커서의 위치

[그림 1.27] 프로그램의 실행 예 1

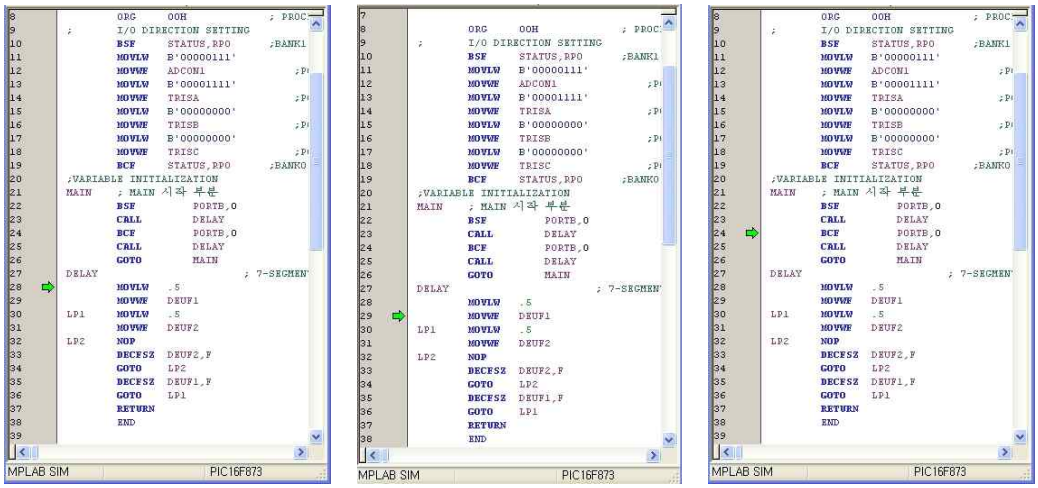
다음은 프로그램을 일괄 수행시키는 방법이 아닌 단계별 수행시키는 Step into( ) 와 Step Over( ) 그리고 Step out( )의 차이를 확인하여 보자.



(a) 현재 프로그램 위치 (b) Step into 결과 (c) Step Over 결과

[그림 1.28] 프로그램의 실행 예 2

그림 1.28의 (a)에서 연두색 커서는 CALL DELAY 라는 프로그램을 가르키고 있습니다. 따라서 다음 단계는 이 명령어(프로그램)를 수행할 것입니다. CALL 명령어는 C 언어의 부 프로그램(subroutine)과 같은 의미의 assembly 명령어입니다. 따라서 프로그램의 수행은 DELAY 라는 부프로그램으로 이동해서 수행을 완료하면(RETURN을 만나면) 다시 되돌아오는 형식으로 진행합니다. 이 상황에서 Debugger 기능 중 Step into를 수행시키면 프로그램이 DELAY 함수로 수행이 이동하는 것을 볼 수 있다. 이것은 Step into 명령이 source program를 한번에 한줄씩 수행시킴을 의미한다. 반면 Step Over의 경우는 그림 1.29의 (c)에서와 같이 CALL DELAY 프로그램 다음 줄로 프로그램이 이동되는 것을 알 수 있다. 이것은 Step Over는 호출할 함수 전체를 한 번에 수행하고 다음 줄로 이동하는 명령이라는 것을 알 수 있다. 만약 Step Over 명령이 함수를 호출하지 않을 경우는 Step into와 동일한 명령이 된다. Step Out의 동작은 그림 1.29을 통해서 알아보자.



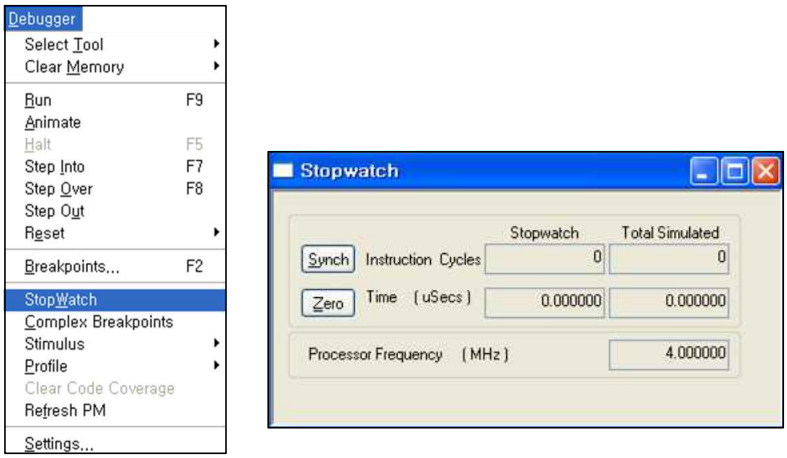
(a) 현재 프로그램의 위치 (b) Step into/ Over 결과 (c) Step Out 결과

[그림 1.29] 프로그램의 실행 예 3

DELAY 함수 안에서 Step into와 Step Over가 동일한 동작을 하며, Step Out의 경우는 DELAY 함수를 한 번에 모두 수행한 후 CALL DELAY 프로그램 다음 줄 즉, BCF PORTB .0으로 이동하는 것을 알 수 있다.
이 설명으로 Step into와 Step over를 충분히 구분할 수가 있는가? 그러나 이 둘의 차이점을 이해하고 잘 활용하는 것이 능력있는 프로그램 개발자가 될 수 있음을 인식했으면 한다. 그리고 일반적인 프로그램 개발 장비에는 Step out 기능은 제공되지 않는 것들이 많이 있다.

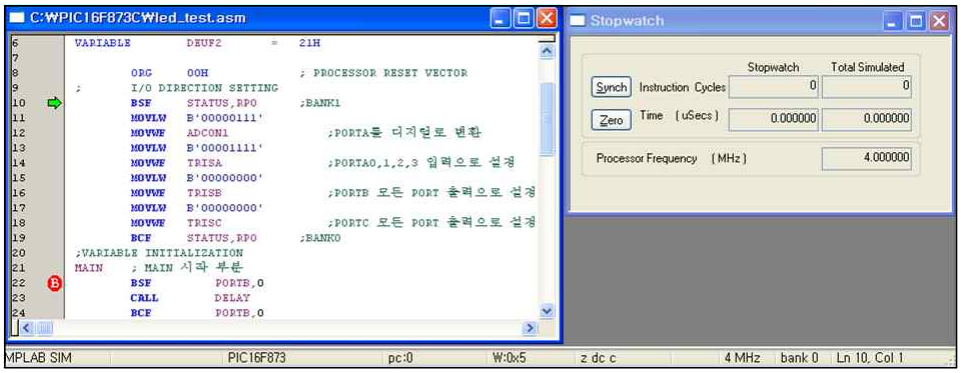
(2) STOP Watch 사용하기 : 프로그램의 수행 시간 계산

마이크로프로세서에서 프로그램을 작성하다 보면 정확한 시간을 알아야 할 경우가 있다. 우리가 사용하는 PIC16F873의 경우 명령어 처리시간이 분기 명령어의 경우 2[usec] 나머지 명령어는 1[usec]로 동작 합니다(단, 입력 클럭 Fosc가 4MHz 일 경우). 따라서 직접 손으로 계산해도 정확한 동작 시간을 알 수 있다. 그러나 계산 자체가 귀찮으니 STOP Watch 기능을 이용해 프로그램 수행 시간을 확인해 보자. 그림 1. 30의 (a)에서와 같이 메뉴바의 Debugger>>StopWatch를 선택하면 아래 그림 1.30의 (b)와 같은 Stop Watch창이 뜬다. Synch 버튼은 Stopwatch의 값을 Total Simulated 값과 일치시키는 기능이고, Zero 버턴은 Stopwatch 값을 0으로 설정하는 기능을 한다.

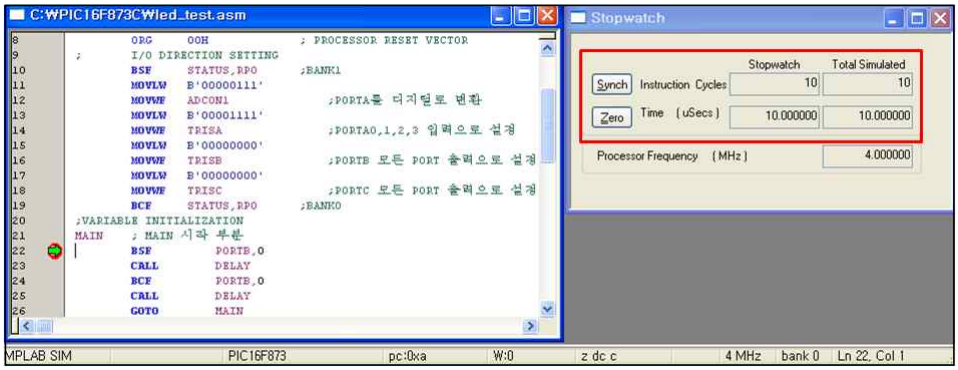


[그림 1.30] Stopwatch window 띄우기

프로그램을 Reset() 시킨 후 Break point()가 설정된 위치까지의 프로그램 수행 시간을 확인하여 보자. 그림 1. 31에서 보듯 Break point 가 설정된 곳 까지는 10줄의 명령어가 있고 이들은 모두 1[usec]로 동작을 한다. 따라서 Break point가 있는 위치까지의 시간은 10[usec]가 됨을 알 수 있다. 이러한 프로그램 수행 시간 계산은 초보자에게는 너무 어려운 이야기 이므로 여기서는 이런 기능이 있다는 것만 알고 지나가도 충분하다. 그러나 많은 제어 프로그램은 시간을 다루는 일이 기본 기능이므로 수행 시간의 계산과 확인은 앞으로 매우 중요한 관심 사항이다.



(a) Break point(Ⓑ)와 Reset(🔄) 수행 결과



(b) Run(▶) 수행 후

[그림 1.31] Stopwatch 기능 확인

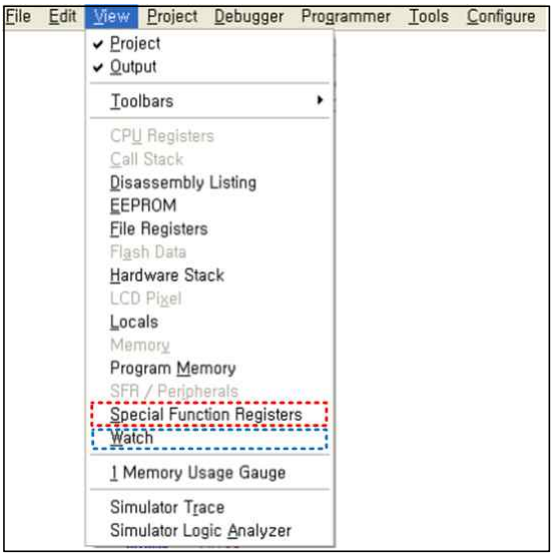
5. Register 정보 보기

Debugging을 하기는 위에서 설명한 프로그램 실행시키기와 함께 연동해서 사용되어 지는 것으로 작성한 프로그램의 오류를 빨리 찾아내기 위해 자주 사용하는 기능들을 설명한다.

참고로 작성한 프로그램이 원하는 기능으로 동작을 하지 않는 것은 단순히 프로그램을 작성하는데 있어서 언어를 올바르게 사용하지 않았다는 문법적 오류가 아니라 여러 명령어가 어우러져서 하나의 기능을 만들어 낼 때 나타나는 동작 오류가 더 문제이다. 앞으로 설명에서도 알 수 있었겠지만, 문법적인 오류는 컴파일러가 오류가 발생한 줄까지 찾아주기 때문에 바로 수정이 가능하지만, 동작오류 또는 프로그램의 오류는 자동으로 찾는 방법이 없으며, 오직 개발자가 담당해야할 몫이다. 이 강의의 주 목적의 어떤 기능을 어떻게 프로그램하여 구현 하는가도 배우지만, 또 이런 동작 오류를 어떻게 찾아내는가를 배우는 것이 주 목적이다.

이를 위해서 준비된 것이 컴퓨터의 내부 상태를 들여다 보는 방법이다. 즉 컴퓨터는 기억된 정보를 다루는 장치이므로 기억된 내용을 확인하면 컴퓨터의 내부를 들여다 볼 수 있고 이를 Register 정보 보기라고 한다. 그리고 우리는 이를 Debugging 이라고 부른다. 하지만 컴퓨터에 내장된 기억장치는 매우 많이 있으며, 이를 무작정 들여다 볼 수도 없고 들여다 보아야 아무런 의미가 없을 수 있다. 따라서 컴퓨터의 내부 동작에 크게 영향을 주는 특별한 기억장치만 보는 것이 더 가치가 있으며, 이런 기억 장치를 우리는 Register라고 부른다. 하지만, Register의 개수도 적은 것이 아니므로 이를 효율적으로 관리 하면서 들여다

볼 필요가 있다. 따라서 먼저 MPLAB IDE의 창을 각자가 선호하는 방식으로 꾸며 보자. MPLAB IDE 창을 꾸미기 위해서는 메뉴바에서 View를 이용해야 한다. 그림 1. 32는 View 메뉴의 구성이다. View 메뉴의 각 항목들은 기능 정의는 표 1.2과 같다.

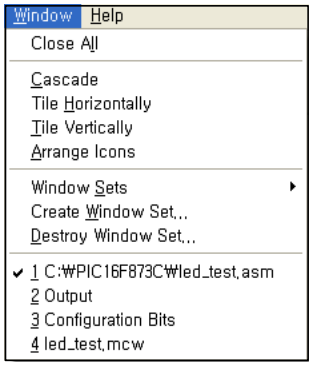


[그림 1.32] View 메뉴의 내용

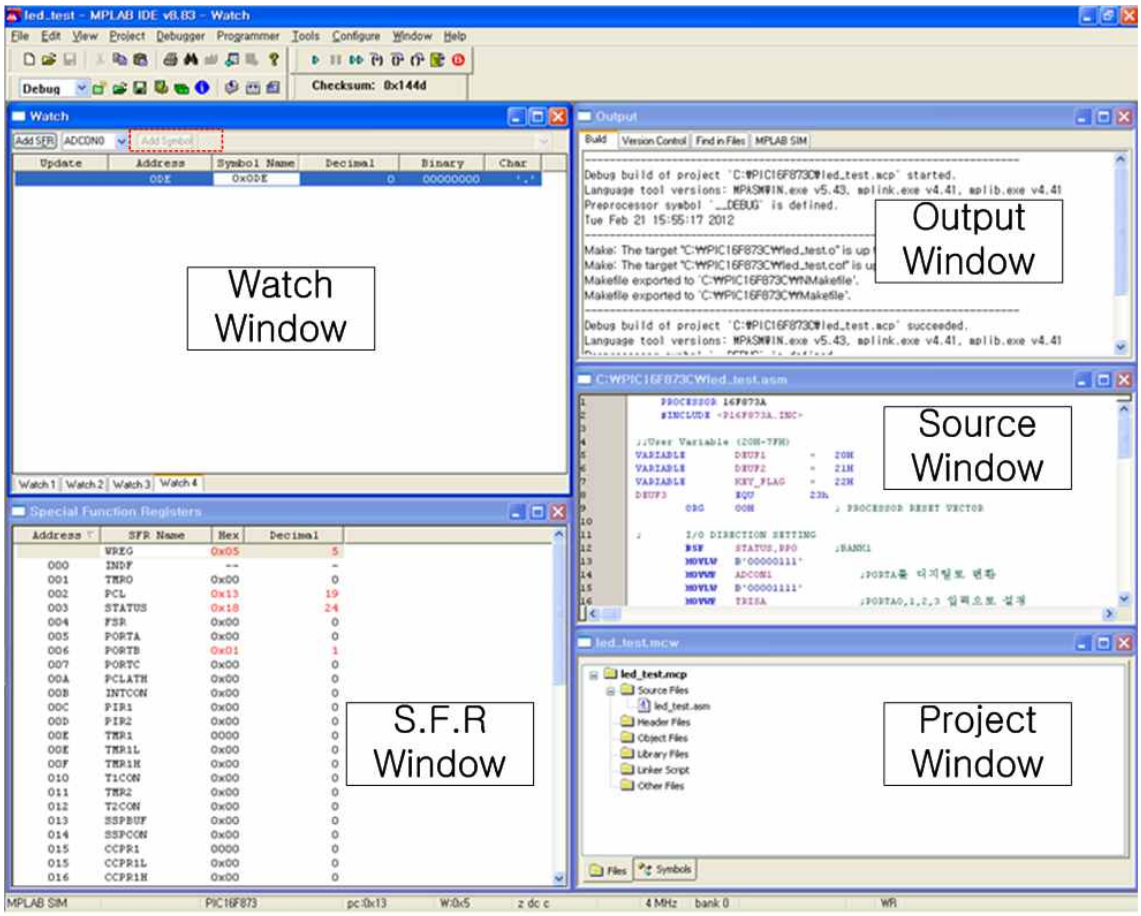
표 1.2 View 메뉴의 기능 정의

√ Disassembly Listing	작성된 프로그램을 기계어 코드로 확인
√ EEPROM	EEPROM 내용을 볼 수 있음
√ File Registers	컴퓨터의 내부 기억장치 중 Register를 보여줌 DATA memory 내용도 볼 수 있음
√ Hardware Stack	Steack의 내용을 볼 수 있음
√ Special Function Registers	S.F.R의 내용을 볼 수 있음 S.F.R의 Regiser 내용 수정 가능
√ Watch 창	사용자 Register의 내용 사용자 Regiser 내용 수정 가능을 볼 수 있음

편의상 모든 창을 띄우고 확인하디에는 번잡하므로, 본 교재의 경우 Special Function Registers 창과 Watch창을 선택하기로 한다. 다음 그림 1. 33의 메뉴바의 Window>>Title vertically를 선택한다. 그러면 Watch, Special Function Registers, Output, Source, Project window 인 총 5개의 창이 그림 1.34과 같이 생성된다.



[그림 1.33] Window 메뉴의 내용



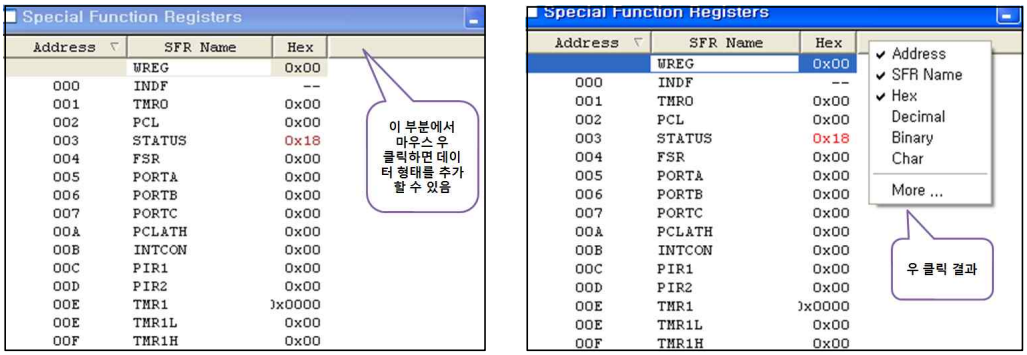
[그림 1.34] Window 메뉴의 내용

각 창의 크기는 사용자의 필요에 따라서 다르므로 각자 창모양은 조정하여 사용 하자. 다만 일반적으로는 source window가 가장 넓게, project window는 가장 작게 잡는 것이 프로그램 개발 하기가 편리하다. 이제 Debugging을 위해 두 가지 중요한 창에 대한 설명한다.

(1) Special Function Registers Window

Special Function Registers(SFR)의 경우 이미 마이크로프로세서 즉 PIC16F873의 레지스터 중 기능이 정의된(주소가 결정된-목적이 결정된)것을 의미하며 이창을 통해서 S.F.R의 값을 볼 수 있고 또 개발자가 강제로 바꿀 수도 있다. F.S.R 값을 바꾸려면 해당 레지스터의 변수 값에 해당되는 부분을 더블 클릭한 다음 원하는 값을 써 넣고 ENTER 키를 치면 된다.

Special Function Registers window에서는 레지스터 값들의 데이터 형태를 여러 가지로 설정할 수 있다. 지원하는 데이터 형태는 Hex(16진수), Decimal(10진수), Binary(2진수), Char(8bit 문자형)이며 현재 설정은 Hex만 설정된 상태이다. 다른 데이터 형태의 추가는 그림 1.35에서처럼 마우스 우 클릭을 통해 한다. 그 결과를 그림 1.36에 나타내었다.



[그림 1.35] SFR 윈도에서의 데이터 형태 추가 방법

Address	SFR Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	--	-	-	-
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x00	0	00000000	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'

[그림 1.36] 모든 데이터 형태 표시 후 SFR 윈도우

다음은 레지스터 값을 변경하기 위한 방법을 알아보자. PORTB라는 레지스터의 값을 03으로 변경하고자 한다. 그림 1.37에 (a)를 보면 PORTB값이 0x01이라는 값을 가진다. 마우스를 PORTB 레지스터의 데이터 형태를 나타내는 곳으로 가져가 클릭한다. 그림 1.37의 (b)의 경우는 Decimal 이라고 표시된 부분을 클릭한 결과이며 이때 클릭한 위치에 파랑색 박스가 나타나며 활성화 되는 것을 볼 수 있다. 이제 그림 1.37의 (c)에서처럼 3을 입력하면 PORTB 레지스터의 값이 3이 되는 것을 알 수 있다. 이렇게 SFR 값을 임으로 변경 하면 프로그램의 흐름이나 결과가 당연히 다른 값을 나타내게 된다. 구체적인 것은 추후 실험과정에서 다시 확인하기로 하자.

Address	SFR Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	--	-	-	-
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x0B	11	00001011	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x01	1	00000001	'.'
007	PORTC	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'
00B	INTCON	0x00	0	00000000	'.'

(a) PORTB 레지스터 값 변경 전 PORTB=0x01

Address	SFR Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	--	-	-	-
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x0B	11	00001011	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x01	1	00000001	'.'
007	PORTC	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'
00B	INTCON	0x00	0	00000000	'.'

(b) PORTB 레지스터 값 변경을 위해 Decimal 부분 클릭

Address	SFR Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	--	-	-	-
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x0B	11	00001011	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x03	3	00000011	'.'
007	PORTC	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'
00B	INTCON	0x00	0	00000000	'.'

(c) 변경하고자하는 값인 3을 입력

[그림 1.37] SFR 레지스터 값 변경하기

(2) Watch Window

Watch window의 경우 MPLAB IDE v8.46 이하 버전에서는 Add Symbol이 활성화 되어 있어 사용자가 정의해놓은 레지스터를 Add Symbol 콤보 박스에서 등록만 하면 되었으나 v8.83의 경우 Add Symbol이 활성화 되어 있지 않다. 따라서 약간 불편하나 사용자 변수의 주소를 이용해서 등록할 수 있으므로 주소를 등록해서 사용하자. 만약 사용자 변수나 레지스터 값을 바꾸려면 해당 레지스터나 변수 위치(Value 항목)에서 변수 값에 해당되는 부분을 더블클릭한 후 원하는 값을 써 넣으면 된다. 구체적인 방법은 그림 1.37 SFR 레지스터 값 변경하기와 같다. 추가한 레지스터의 항목 삭제는 **Watch window**에서 **마우스를 우 클릭**을 하여 Delete 항목을 선택하면 된다. 이는 당연히 Keyboard에서 Delete 키를 눌러도 된다.

Watch window 역시 사용자 레지스터의 변수 값을 다양한 형태로 지원하며 설정 방법은 SFR 설정 방법과 동일하며(그림 1.35, 그림 1.36 참고), Comment 항목이 추가되어 있다. 따라서 변수이름을 여기다 명시해놓으면 디버깅할 때 유용하게 사용할 수 있다. 그림 1.38에는 Watch window의 데이터 형태와 Comment 값에 20H번지에 선언된 DEUF1이라는 변수이름을 써 놓은 예이다.

[그림 1.38] Watch window 설정

이상으로 무료통합 환경 MPLAB IDE의 설명을 간략하게 정리하며, 중요한 것은 앞으로 프로그램 작성 및 개발 과정에서 수시로 참고 하기 바란다.