

## 제7장 프로그램 수행시킴과 디버깅하기

### 1. 프로그램 수행시킴

다음으로는 작성된 프로그램을 수행시키면서 디버깅을 해보자. 개발장비 메인 화면에서 작성된 프로그램을 **Debug** 상태(**Release** 상태가 아님)로 설정하고 **Project**에서 **Build All** 하여 컴파일 및 링크를 한다. 그리고 **Output** 화면의 **Build** 창에 에러가 없이 나오면, 메인 화면의 **Debugger** 항목을 선택하여 **Select Tool**에서 6번의 **Real ICE**를 선택한다. 정상적으로 인식이 되면, 현재의 상태가 **Output**에 나타난다. 정상적이 아니면 다시 **Debugger** 항목에 들어가 **Reconnect**를 수행한다. 그래도 안되면 **settings**를 통해서 동작을 확인하고 조치를 취한다. 일반적으로는 개발장비의 프로그램을 구동시키면 자동으로 모든 것이 설정 되도록 되어 있으나, 혹시 변경된 경우나 내가 초기 설정을 한다는 조건에서 관심있게 보기 바란다.

**ICE**가 정상 동작되어 **target board**의 마이컴과 연결이 되면, **connected** 라는 문구가 나오며, 이 경우에는 메인 화면의 **Debugger** 항목에 들어가 보면 **Program** 이라는 항목이 활성화 되어 있다. 따라서 이를 클릭 하면 프로그램이 진행된다. 그리고 에러가 없으면 아래 그림과 같은 화면이 활성화 된다.








Debugger Menu	Run	Halt	Animate	Step Into	Step Over	Step Out	Reset
Toolbar Buttons							

그림 7-1. 디버깅을 위한 메인화면 아이콘 들

그림에 있는 각 아이콘에 대한 기능과 의미를 설명하면 아래와 같다.

- **Run**: 프로그램을 처음부터 무작정 동작시키는 기능이다.
- **Halt**: 프로그램을 현 상태에서 멈추어라이며, 이때까지 수행된 결과가 화면에 나오므로 프로그램의 흐름을 검증할 수 있다.  
(이전 표시와 현 표시 사이에 값이 변한 것은 붉은 색으로 표현됨)
- **Step Into**: 프로그램을 멈춘 상태에서 명령어 단위로 수행시키는 기능이다. 따라서 이 키는 개별 명령어에 의한 영향을 확인하고 싶을 때 사용한다.

- **Step Over:** Step into와 비슷하게 명령어 1줄 단위로 동작한다.  
그러나 step into와는 달리 부프로그램에 해당하는 CALL 명령어는 한꺼번에 수행하고 멈춘다.
- **Reset:** cpu를 reset시키며, 최초의 프로그램이 들어있는 시작 위치를 지정한다. 우리가 사용하는 프로세서는 프로그램 시작위치가 0000번 지 이므로 이 위치로 시작점을 변경한다.
- **Animate:** Step into를 장비 스스로 수행하는 모드이다. 멈추고 싶을 경우에는 Break를 누르면 된다.
- **Step out:** Step in 상태에서 벗어나고 싶을 때 사용한다. 즉 Step into로 Call 프로그램 안에 들어간 경우에 이 버튼을 사용하면 Call 문 안의 나머지 프로그램은 자동적으로 수행하고 Call 문을 벗어나는 기능이다.

긴 프로그램의 경우 위의 버튼 만으로 프로그램을 따라가면서 문제점을 발견하기는 한계가 있으므로, 문제가 생길만한 곳에서만 프로그램을 멈추고 동작 상태를 확인하기 위한 용도가 **Break point** 이다.

Break point는 디버깅 하고 싶은 프로그램 위치에 커서를 올리고 더블 클릭을 하면 붉은 점이 표시가 된다. 즉 이 라인에 있는 프로그램을 수행하고는 마이컴이 스스로 멈추게 된다는 것이다. 아래 화면은 Break point를 설정한 화면이다.

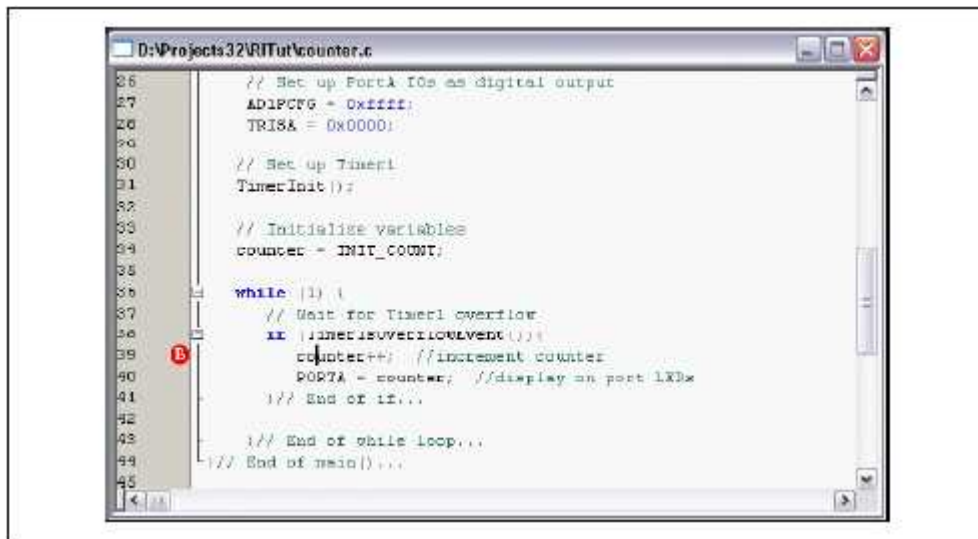


그림 7-2. 디버깅 화면에서 브레이크 포인터가 설정된 상태

다음으로는 현재 수행된 결과가 정상적으로 이루어진 것인가를 확인하는 방법에 대하여 설명한다. 즉 디버깅이란 현재 동작 상태를 확인하여 개발자가 원하는 결과가 들어 있는가를 직접 확인하는 것이며, 이를 나타내는 화면이 메인 화면의 View 항목을 클릭하면 나온다.

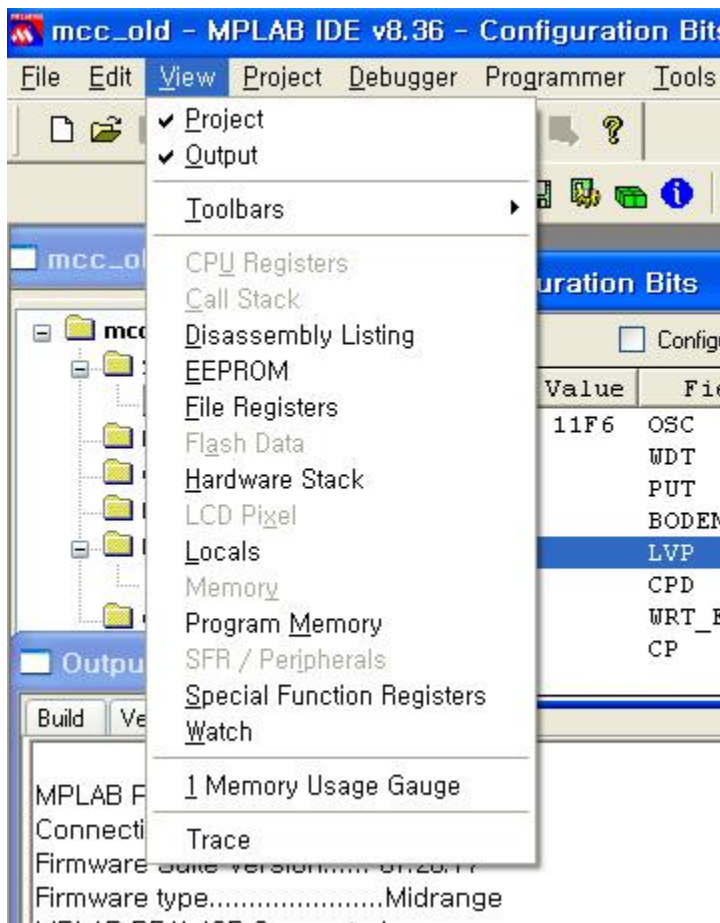


그림 7-3. 디버깅을 하기 위한 화면 만들기

위 기능에서 디버깅과 관계되는 기능은 마이컴 내부의 레지스터나 메모리의 내용을 개발장비 화면에서 볼 수 있는 기능으로 File Registers, Special Function Registers 그리고 Watch 가 있다.

- File Registers: 마이컴 내부에 있는 모든 기억소자를 주소 순으로 전부 보여준다. 따라서 내가 필요로 하는 메모리 주소를 알아야 디버깅이 가능하다. 따라서 별로 권장하는 방법은 아니다.
- Special Function Registers: 마이컴이 가지는 메모리 중에서 마이컴을 제어하는 기능을 가지는 특수 목적 기능의 기억장치 만을 표시해주는 방법이다. 보통의 디버깅에서 가장 많이 사용하는 방식이다.
- Watch: 보고 싶은 기억장치 만을 선택해서 보여주는 방식이다. 따라서 가장 좋은 방식이다. 이 기능은 마이컴이 사용하는 특수목적 기억장치와 개발자가 정의한 변수(데이터 메모리)의 내용을 동시에 선택 가능하며, 따라서 선택 항목이 2 종류로 나누어져 있다.

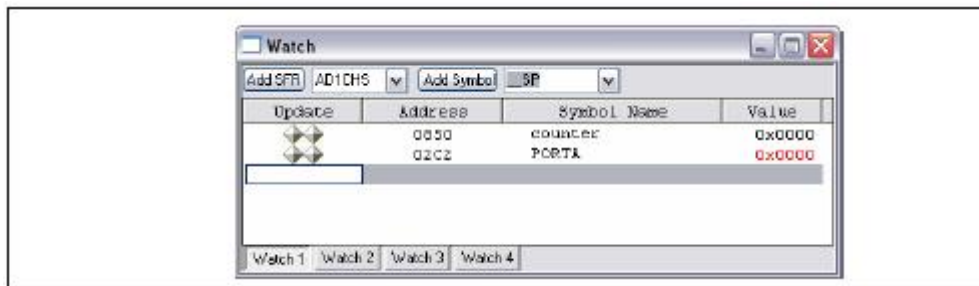


그림 7-4. 디버깅을 위한 Watch 화면을 띄운 상태

그림에서 **counter**는 개발자가 프로그램에서 정의한 변수이고, **PORTA**는 마이컴이 사용하는 고유한 특수목적 기억장치이다. 그리고 그 결과를 다양한 형태로 확인할 수 있다. 이러한 내용은 글로 읽는 것 보다 직접 해보는 것이 더 이해하기가 쉬우며, 다른 기능도 스스로 확인해 보기 바란다.

## 2. 디버깅이란?

MPLAB-IDE은 여러 가지의 디버깅 방법을 제공하고 있습니다. 디버깅이란 말 그대로 버그를 잡는 작업을 말합니다. MPLAB IDE 같은 에뮬레이터를 사용하는 궁극적인 목표도 바로 디버깅을 쉽게 하기 위해서입니다.

흔히 에뮬레이터를 사용하는 사람을 보게 되면 <다운로드>의 기능만을 활용하는 경우가 많은데, 이것은 에뮬레이터를 100% 활용하지 못하는 것입니다. 버그를 잡는 기술도 약간의 테크닉을 필요로 하는데 본 장에서는 이러한 디버깅 방법과 테크닉에 대한 설명을 다루고 있습니다.

### 3. 스텝 실행과 실행 포인터 변경

스텝 실행은 가장 기본적인 디버깅 방법입니다. 실행 중 <Break>버튼을 눌러서 실행을 멈추면, 화면은 실행을 멈춘 위치에서 머물러 있습니다. 이때 <Step>버튼을 누르면 현재 명령 포인터(빨간 박스)가 가르치고 있는 명령 하나를 실행하고 그 결과 변경된 레지스터 등은 화면에 표시됩니다.

스텝 실행은 다음과 같은 3가지 종류가 있습니다.

STEP	한 행씩 실행하는 스텝기능입니다.
STEP OVER	실행하려는 명령이 CALL명령이면 서브루틴을 수행한 후 실행을 멈춥니다. CALL명령이 아니면 평상시대로 한 줄만 수행합니다. (주로 간단한 서브루틴을 스킵하고자 할 때 사용합니다.)
ANIMATE	일정한 시간 간격으로 스텝을 수행하는 기능입니다. MR.PIC-IDE을 쓰다 보면 STEP버튼을 계속 누르는 경우가 많이 발생하는데, 이때 ANIMATE를 실행하면 여러 번 누르는 수고를 줄일 수 있습니다.

ANIMATE를 수행하는 시간간격은 SETUP메뉴에 있는 Animate Speed Setting으로 조정할 수 있습니다. 조정가능한 시간은 100 ms에서 2초까지입니다. (시스템 성능에 따라 다소 차이가 있을 수 있습니다.)

스텝실행을 하다보면 여러 번 반복하는 루프 안으로 들어가는 경우가 종종 있습니다. 다음과 같은 경우입니다.

	MOVLW 200
	MOVWF LOOP_CNT1
LOOP1	CLRWDI
	DECFSZ LOOP_CNT1
	GOTO LOOP1
	NOP

이런 경우는 루프 안의 세 줄을 200번 반복해서 <Step>버튼을 눌러야만 그 밑의 줄을 실행해 볼 수 있을 것입니다. MPLAB-IDE에서는 이런 루프를 간단히 빠져나갈 수 있는 방법을 제공하고 있는데 이것이 바로 <실행 포인터

의 변경>입니다.

소스 리스트 윈도우의 **NOP**명령(실행 포인터를 변경하고 싶은 위치)에 마우스 커서를 위치하고 마우스의 오른쪽 버튼을 누르면 간단히 실행 포인터는 변경되고 **NOP**명령에 빨간 박스가 놓여집니다.(왼쪽이 아닌 오른쪽 버튼이라는 것에 주의하십시오)

```

                                MOVLW    200
                                MOVWF    LOOP_CNT1
LOOP1                          CLRWDT
                                DECFSZ   LOOP_CNT1
                                GOTO      LOOP1
                                NOP

```

#### 4. 레지스터 값의 변경

레지스터값을 변경하려면 변수 윈도우나 와치 윈도우에 변경을 원하는 윈도우에 마우스 커서를 위치시키고 왼쪽 버튼을 더블클릭하면 됩니다.

그리고 value부분에 변경을 원하는 값을 써넣고 Enter key 를 누르면 해당 값으로 간단히 변경됩니다. 레지스터 변경 윈도우 상의 16진, 10진 표현은 C 언어에서의 표현방식을 따릅니다. 즉 16진의 경우에는 0x를 앞에 붙여서 입력해야 합니다. (예를들면 0xAB, 0x10)

#### 5. 와치 윈도우의 사용

디버깅을 하다보면 관심이 집중되는 몇 개의 레지스터가 생기게 됩니다. MPLAB-IDE의 변수윈도우나 특수 레지스터 윈도우에는 모든 레지스터의 내용을 표시하고 있기 때문에 별 문제가 없을 것 같지만 작업을 하다보면 스크롤 바를 눌러가면서 찾아 다녀야 하는 불편함이 있다는 사실을 알게 될 것입니다.

이러한 불편함을 제거하고 그리고 관심있는 레지스터를 서로 비교해 볼 수도 있는 와치 윈도우는 디버깅 작업에서 빼놓을 수 없는 역할을 하고 있습니다.

다. 와치 윈도우에 원하는 레지스터를 등록하는 방법은 매우 간단합니다. 변수 윈도우나 특수 레지스터 윈도우 중에서 원하는 레지스터에 마우스 커서를 위치하고 오른쪽 버튼을 클릭하면 선택한 레지스터가 와치 윈도우에도 표시됩니다. 와치 윈도우에는 16진, 10진, 2진이 모두 표시됩니다. 와치 윈도우의 내용을 모두 지우고 싶을 때에는 Debug 메뉴의 Clear Watch Window 메뉴를 선택하면 됩니다. 와치 윈도우 내용 중 한 줄만 지우고 싶다면 해당 위치에 가서 마우스 오른쪽 버튼을 클릭하면 됩니다.

## 6. 브레이크 포인트

브레이크 포인트는 디버깅 작업에 있어서 없어서는 안 될 가장 중요한 기능입니다. 단어가 의미하는 대로 브레이크를 걸 수 있는 포인트를 설정하는 기능입니다. 다시 말해 프로그램 실행 도중 원하는 부분에서 실행을 멈추고 그 당시의 상황을 볼 수 있도록 하는 기능입니다.

브레이크 포인트를 설정하는 방법 역시 매우 간단합니다. 소스 리스트 윈도우 중 브레이크 포인트를 설정하고 싶은 위치에 마우스 커서를 놓고 마우스 왼쪽 버튼을 더블클릭하면 됩니다. 그러면 해당 위치가 검은색 바로 반전되어 나타나고 다시 해제하고 싶으면 같은 방법으로 더블클릭하면 됩니다. 이런 식으로 프로그램 메모리 전 영역에 걸쳐서 브레이크 포인트를 설정/해제할 수 있습니다.

브레이크 포인트가 다수 설정되어 있을 경우에는 찾아다니면서 해제시키는데 어려움이 있으므로 **Debug** 메뉴 밑에 **All Clear Break Point**라는 메뉴를 사용해서 모든 브레이크 포인트를 한번에 해제할 수도 있습니다.(다른 방법으로는 다운로드를 한번 더 하는 방법도 있습니다. **다운로드시 모든 브레이크 포인트는 해제**됩니다.)

브레이크 포인트는 주로 프로그램 중 특정부분이 제대로 수행되고 있는지를 확인할 때 사용합니다. 만약 브레이크 포인트를 설정해놓았는데 불구하고 실행이 멈추지 않았다면 그 부분은 실행되지 않았다는 것을 의미합니다.

반대로 실행되면 안 되는 부분에 브레이크 포인트를 설정해놓고 실행하였을 때, 브레이크가 걸리면 잘못 되었다는 것을 의미하기도 합니다.

또 하나의 방법으로는 어떤 액션을 취했을 때 브레이크가 걸리게 하는 방

법이 있습니다. 예를 들어 키펀력을 처리하는 루틴의 시작 부분에 브레이크 포인트를 설정해놓고 실행한 다음, 키펀력을 하면 브레이크가 걸리게 됩니다.

브레이크 포인트를 활용하는 경우는 이와 같은 3가지 경우가 대부분입니다. 흔히 초보자는 브레이크 포인트의 활용방법을 잘 몰라서 못쓰는 경우가 많은데, 위의 3가지 경우가 생겼다면 서슴없이 브레이크 포인트 기능을 활용해보시기 바랍니다.

참고: Debug메뉴에 있는 Find Break Point라는 명령은 브레이크 포인트를 여러 개 설정해 놓았을 때 매우 유용한 기능입니다. 설정해 놓은 브레이크 포인트가 있는 곳을 찾아주는 기능으로 Ctrl-B키를 사용해서 쉽게 사용할 수도 있습니다. (MR.PIC-IDE을 사용하다 보면 브레이크 포인트를 설정해놓은 곳을 몰라 소스 리스트 윈도우를 한참 동안 뒤지는 경우가 종종 발생하는데 이때 Ctrl-B를 누르면 간단히 브레이크 포인트를 찾을 수 있습니다.)

자세한 사용 방법은 부록에 나와 있는 것을 참고하시기 바랍니다.

---