

W7100A / W7100 Debugger Guide

Version 1.2



© 2012 WIZnet Co.,Ltd. All Rights Reserved.

For more information, visit our website at <http://www.wiznet.co.kr>

Table of Contents

1	Driver Installation for the Debugger	3
2	Connect the Debugger	5
3	Installation of the W7100A / W7100 debugger.....	6
4	KEIL μ Vision Project	7
5	Debugger Menu	12
5.1	Open Project	12
5.2	Initialize the W7100A / W7100 debugger	14
5.3	Run and Stop	16
5.4	Break Point	18
6	Search & Variable Window	23
6.1	Search Window	24
6.2	Local Variable Window	25
6.3	Global Variable Window.....	25
6.4	Symbol Window.....	26
6.5	Register Window.....	26
7	Memory Window	27
7.1	IData Memory Window.....	28
7.2	External Data Memory Window	29
7.3	Code Memory Window	29
7.4	Flash Memory Window.....	30
7.4.1	Code Memory Domain.....	30
7.4.2	Data Memory Domain	30

1 Driver Installation for the Debugger

W7100A / W7100 debugger에 USB 케이블을 연결하고 PC와 연결한다. 그러면 Fig.1.1과 같은 새 하드웨어 검색 마법사 창이 나타난다. 소프트웨어 자동으로 설치(권장)을 선택 후 다음 버튼을 누른다.



<Fig.1.1> The new hardware search window 1

소프트웨어 자동 설치가 되지 않는 경우, '목록 또는 특정 위치에서 설치(고급)'를 선택한 뒤 아래와 같이 드라이버를 설치한다.



<Fig.1.2> The new hardware search window 2

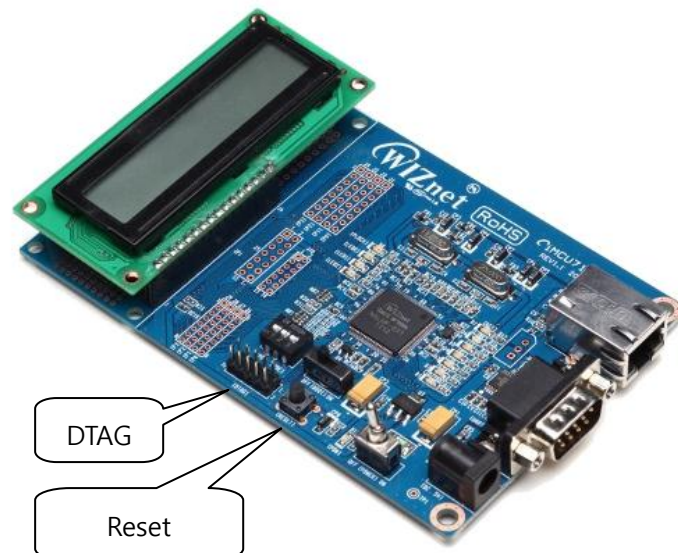
새 하드웨어 검색 마법사가 W7100A / W7100 debugger의 디바이스 드라이브를 찾을 때까지 기다린다.
W7100A / W7100 debugger의 디바이스 드라이브가 설치 끝났다면 마침 버튼을 눌러 종료한다.



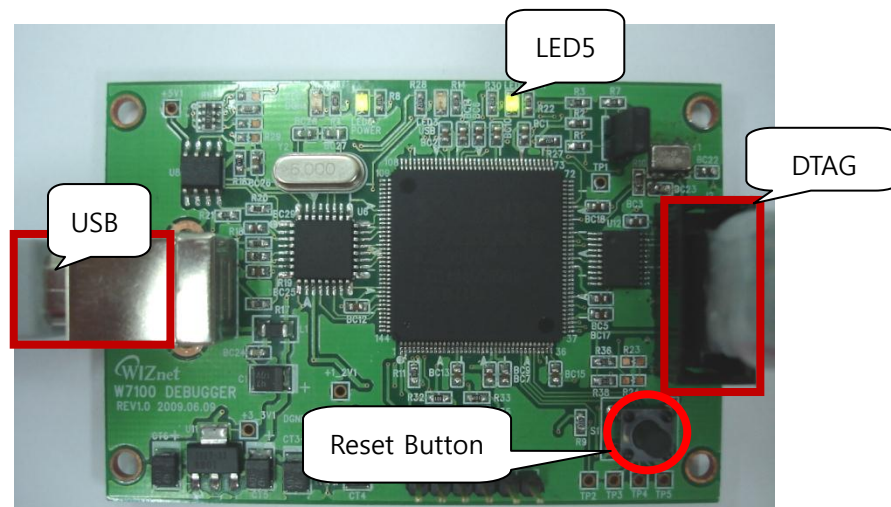
<Fig.1.3> Completing the new hardware searching #2

2 Connect the Debugger

iMCU7100EVb와 Debugger의 연결은 아래의 그림의 iMCU7100EVb의 DTAG 포트와 Debugger를 연결하면 된다.



<Fig.2.1> iMCU7100EVb



<Fig.2.2> W7100A / W7100 debugger

Debugger와 iMCU7100EVb를 DTAG케이블을 통해 연결한다. 또한, Debugger는 USB 케이블을 이용해서 PC와 연결한다. 이 때 DTAG케이블의 빨간 선과 iMCU7100EVb 보드 DTAG의 1번이 일치해야 한다. iMCU7100EVb에 전원 케이블을 연결하고 Fig.2.1의 리셋 버튼을 누른다. W7100A / W7100 debugger 리셋 버튼을 눌렀을 때 LED5가 점등되는지 확인한다. Debugger가 iMCU7100EVb를 정상적으로 인식했다면 LED5가 점등된다. LED5가 점등되지 않았다면 iMCU7100EVb가 정상적으로 인식되지 않은 상황이므로 Fig.2.2의 Debugger 리셋버튼을 눌러서 Debugger를 리셋해 본다. LED5가 점등되어야만 Debugger를 정상적으로 사용할 수 있다.

3 Installation of the W7100A / W7100 debugger

Debugger 설치 프로그램을 실행한다.

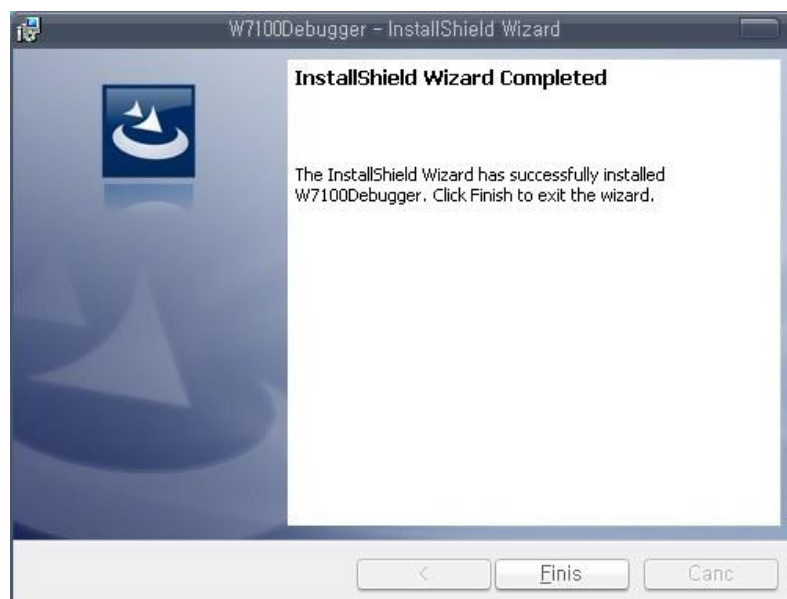


<Fig.3.1> The W7100A / W7100 debugger install file

Next 버튼을 누르면 설치가 시작된다.



<Fig.3.2> Setup the W7100A / W7100 debugger



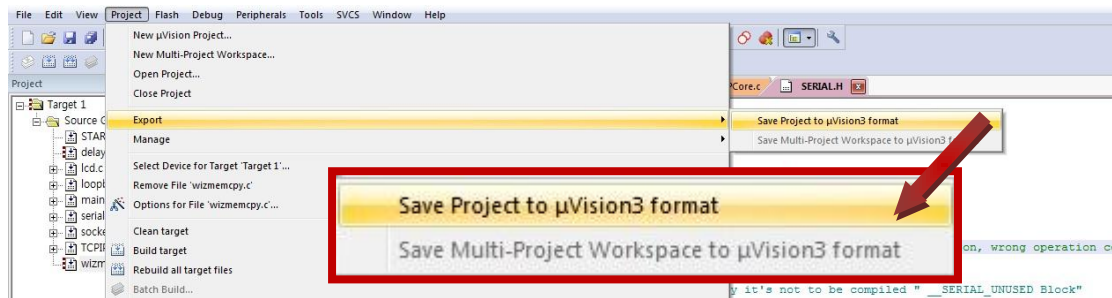
<Fig.3.3> End of debugger installation

4 KEIL μ Vision Project

W7100A / W7100 debugger에서 지원하는 project와 파일

- KEIL μ Vision2 project
- KEIL μ Vision3 project
- Only HEX file (HEX file만 로드 하면 Symbol을 이용한 기능이 제약됨)

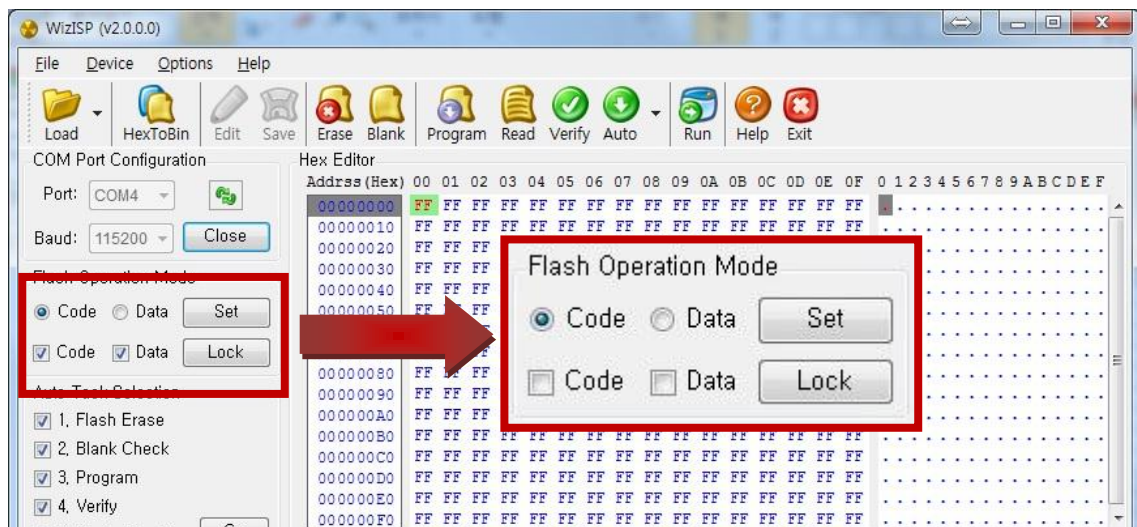
※ 만약 KEIL μ Vision4 이상 버전 사용자라면 debugger에서 project file을 지원하지 않으므로, 다음 Figure 4.1과 같이 μ Vision3 format으로 project file을 저장하여 debugger를 이용할 수 있다.



<Fig.4.1> Save Project to μ Vision3 format

W7100A 사용 시 주의점

W7100A는 W7100과 달리 chip에 code, data lock 설정이 되어 있어 Read가 불가능하므로 lock 상태로는 debugger를 사용할 수 없다. 따라서 debugger 사용 시 WizISP 프로그램을 사용하여 lock을 해제하여야 한다. W7100A 보드에 UART로 연결한 뒤 Figure 4.2와 같이 체크박스를 해제하고 Lock 버튼을 누르면 unlock된다. WizISP 프로그램은 [WIZnet website](http://www.wiznet.co.kr)에서 다운로드 할 수 있으며, 좀 더 자세한 사용 방법은 동일 website의 WizISP User's Manual을 참조하기 바란다.



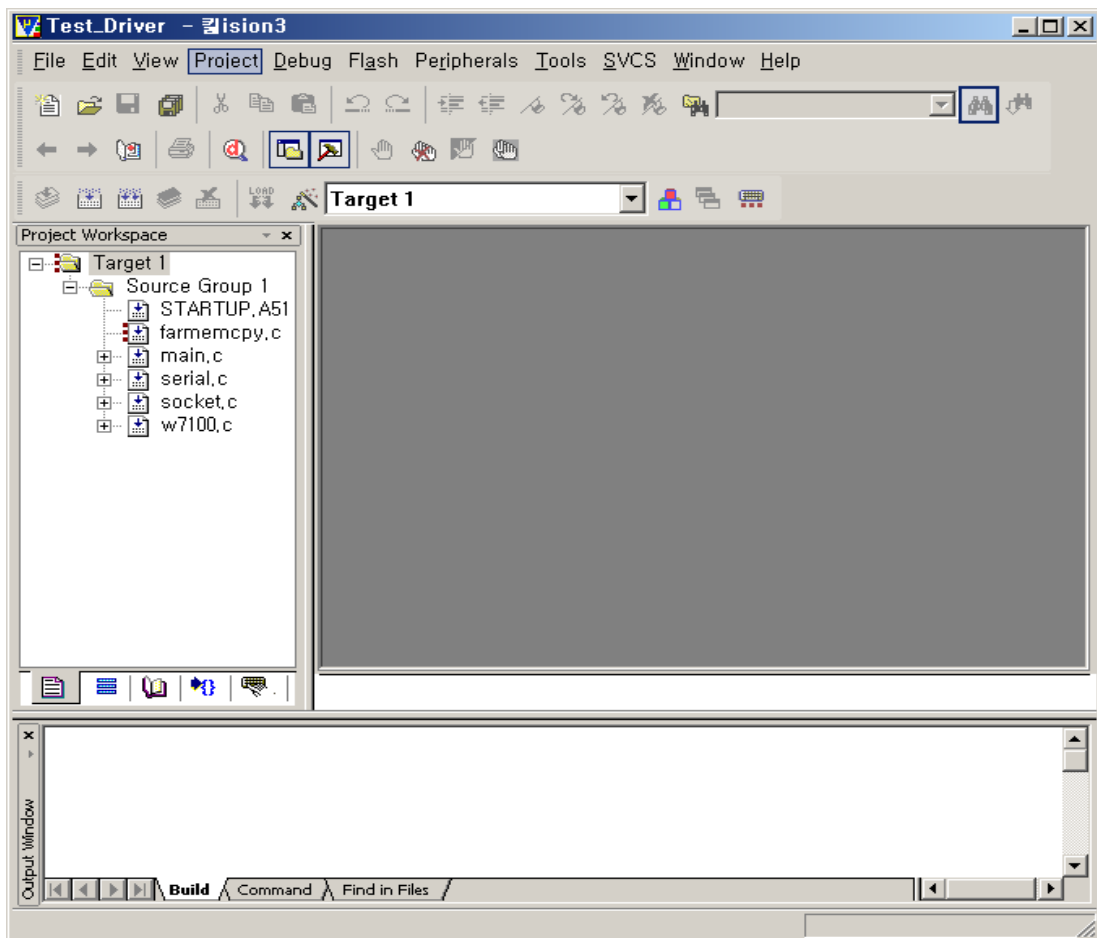
<Fig.4.2> Unlock the W7100A using WizISP program

W7100A / W7100 debugger를 사용하기 위해서는 project 생성 시 project 파일과 소스 파일이 모두 동일한 directory 내에 위치해야 정상적으로 project가 open 된다.

또한 KEIL μ Vision project에서 HEX 파일 생성 옵션과 심볼 생성 옵션을 활성화 시켜주어야 한다. 그렇지 않으면 W7100A / W7100 debugger에서 로드 할 HEX 파일 찾지 못하거나 HEX 파일을 찾았다고 할지라도 open 시 오류를 출력할 것이다.

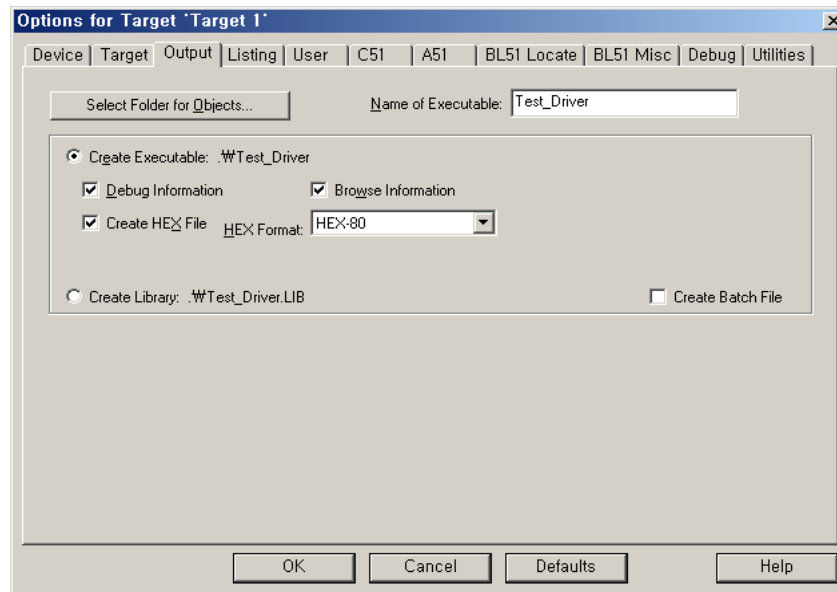
만약 소스파일 내용 중 delay를 사용하는 부분이 있다면(ex: S/W reset) debugger에서 run 수행 시 delay 구간에서 오랜 시간 지체될 수 있다. Debugger를 통해 코드를 수행할 때는 실제 동작 clock보다 훨씬 낮은 clock에서 동작하기 때문이다. 이러한 경우 소스코드 상의 delay 구간을 임시 주석 처리하여 debugging 하는 것을 권장한다.

먼저 기존의 KEIL μ Vision project를 열고, 풀다운 메뉴에서 “Project” -> “Options for Target ‘타겟이름’” 메뉴를 Click한다.



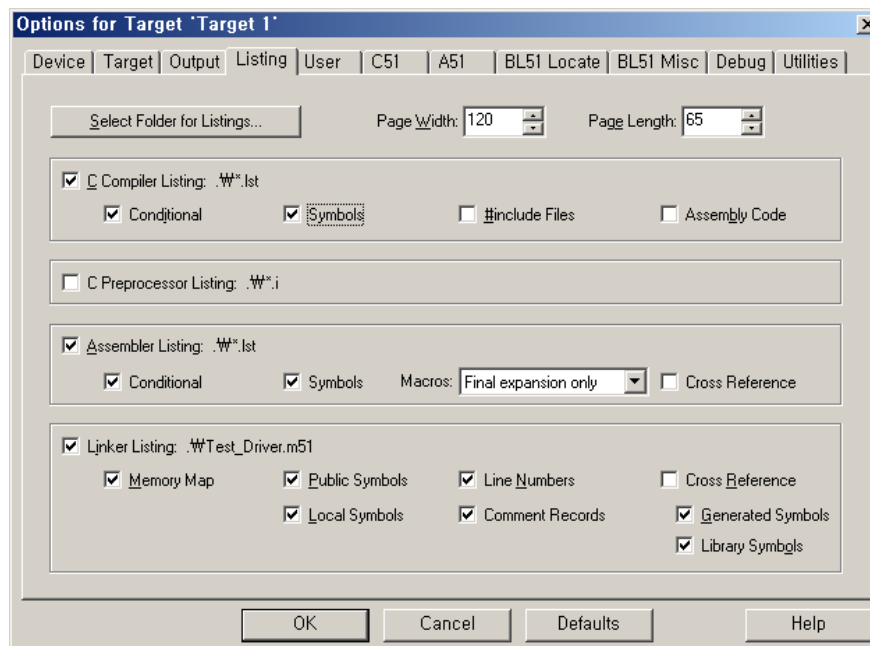
<Fig.4.3> Open the KEIL μ Vision Project

“Output” 탭을 선택한 후 “Create HEX File” 체크 버튼을 위 그림과 같이 체크한다.



<Fig.4.4> Output tap of the ‘Options for Target’ window

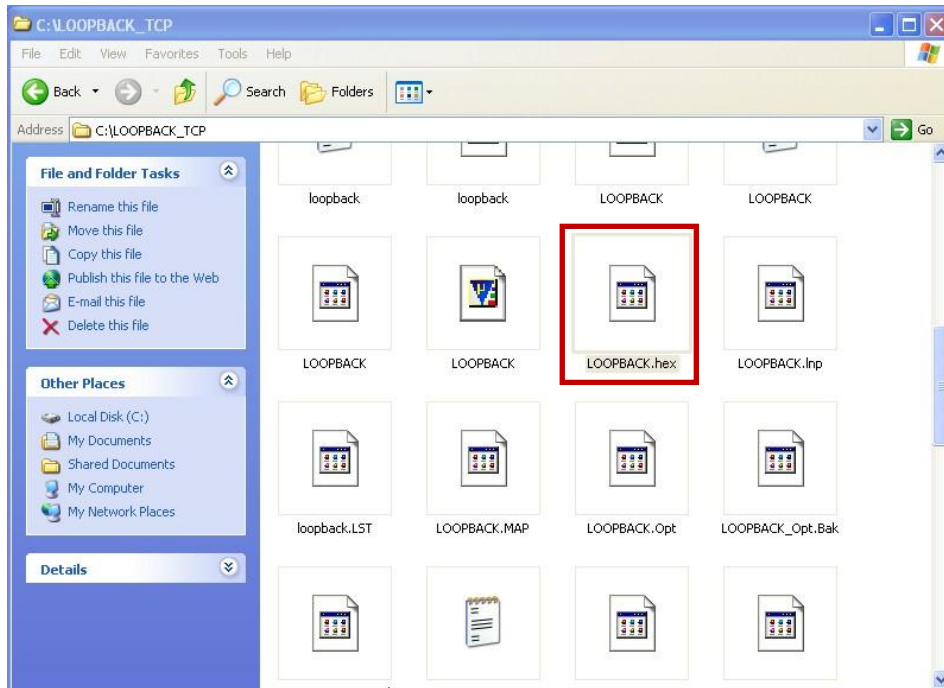
“Listing” 탭을 선택한 후 “C Compiler Listing” 부분의 “Symbols” 체크 버튼을 선택한다.



<Fig.4.5> Listing tap of the ‘Options for Target’ window

그리고 “Assembler Listing” 부분의 “Symbols” 체크 버튼 역시 선택한다. “Assembler Listing” 부분은 선택 사항이라 반드시 체크 할 필요는 없다. “OK” 버튼을 눌러 설정을 종료하고 KEIL μ Vision project 를 다시 컴파일 한다.

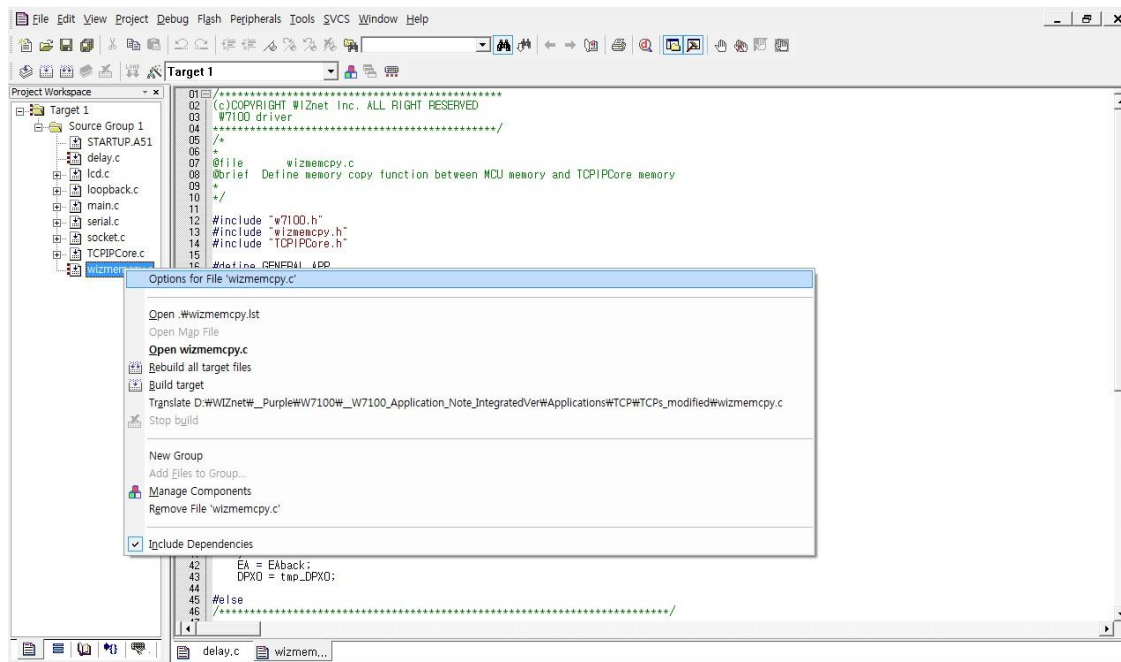
그러면 아래의 그림과 같이 project 디렉터리 내에 HEX 파일이 생성된다.



<Fig.4.6> HEX file of the KEIL μ Vision project

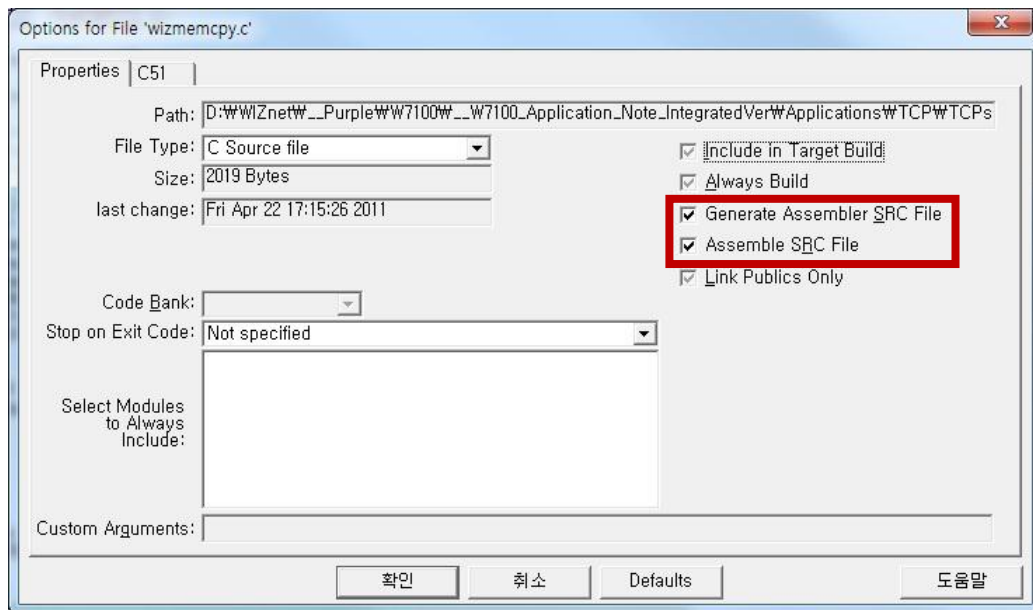
Assemble code를 사용할 경우, assemble code를 포함한 file은 Fig.4.7 - 8과 같이 ‘Generate Assembler SRC File’ 와 ‘Assemble SRC File’ 옵션을 체크해야 한다.

먼저 assemble code를 사용하는 파일의 Options for File ‘xxx.c’을 연다.



<Fig.4.7> Open the Options for File ‘xxx.c’

그런 후, 아래의 그림과 같이 ‘Generate Assembler SRC File’와 ‘Assemble SRC File’ 체크박스를 클릭하여 옵션을 활성화 시킨다.




<Fig.4.8> Setting the Options for File ‘xxx.c’

5 Debugger Menu

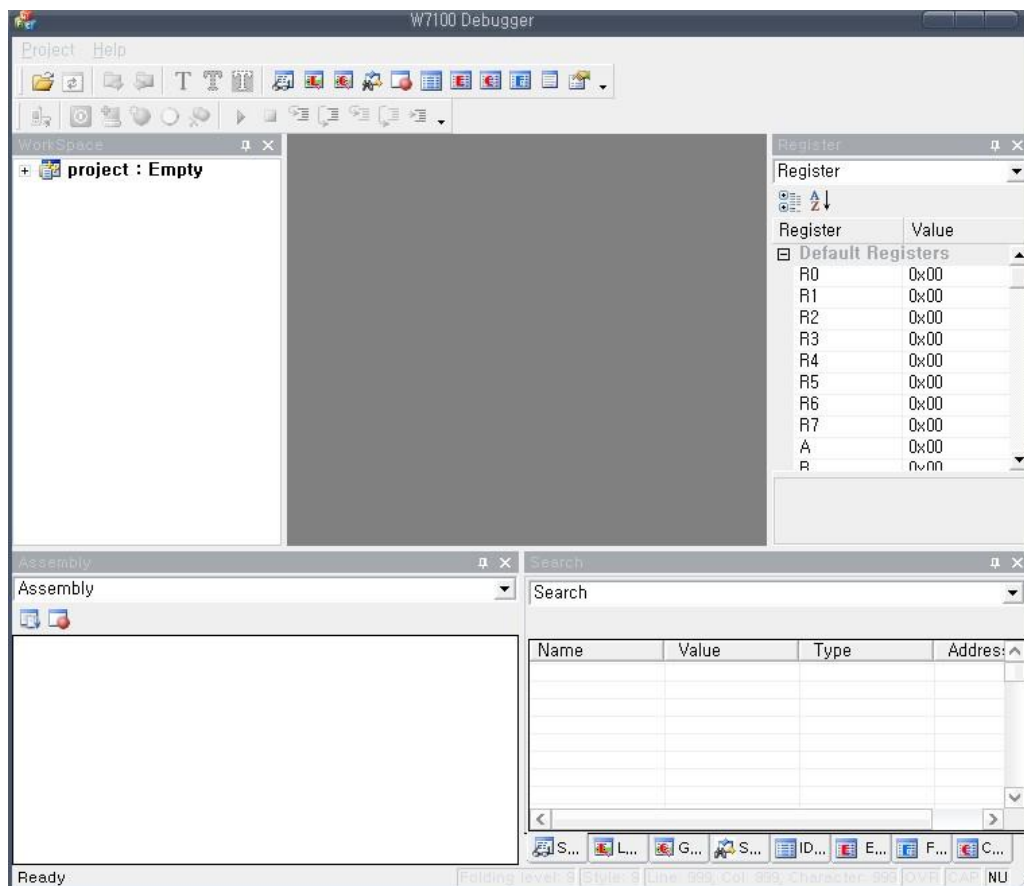
5.1 Open Project

이제 생성한 KEIL μ Vision project HEX 파일을 W7100A / W7100 debugger를 가지고 디버깅 한다. 먼저 W7100A / W7100 debugger를 실행시키고 HEX 파일을 open 한다. 일반적으로 open 하기 위해 the *.LST, *.SRC, *.M51 or *.MAP, *.C, *.Uv2, *.A51 파일들이 필요하다. *.HEX file을 이용하여 open 할 경우에는 ‘assembly window’만 open 되는 것을 상기하기 바란다.

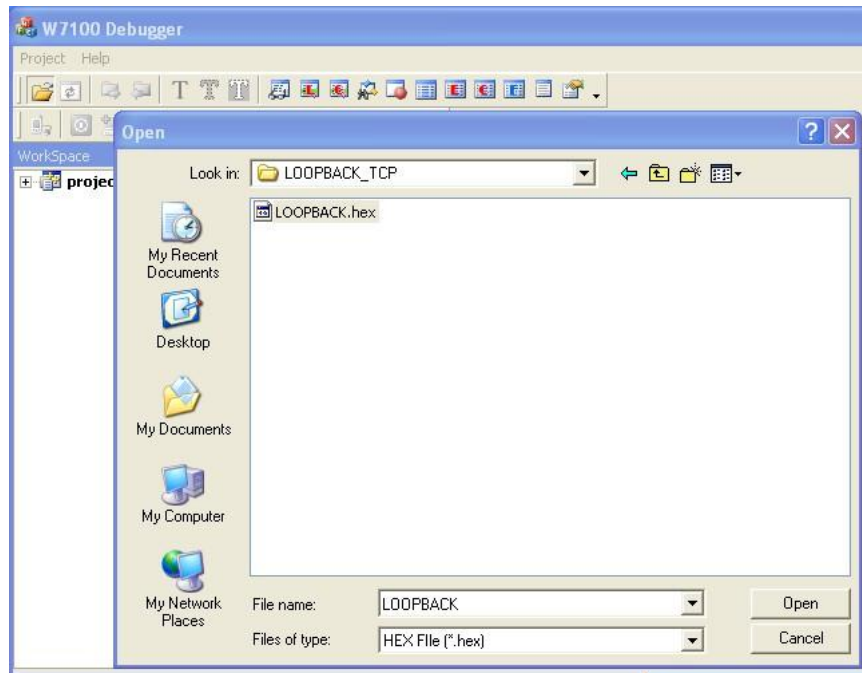
Open Project.

project를  open 하기 위해 Open Project 명령어를 이용한다. ‘Tool’ bar의 아이콘을 Click한다. 또한, Project Menu에 Open이나 short key ‘Ctrl + O’를 이용하여 project의 open이 가능 한다.

위의 방법들 중 하나로 project를 open하면 아래의 그림과 같이 file browser가 실행 된다.

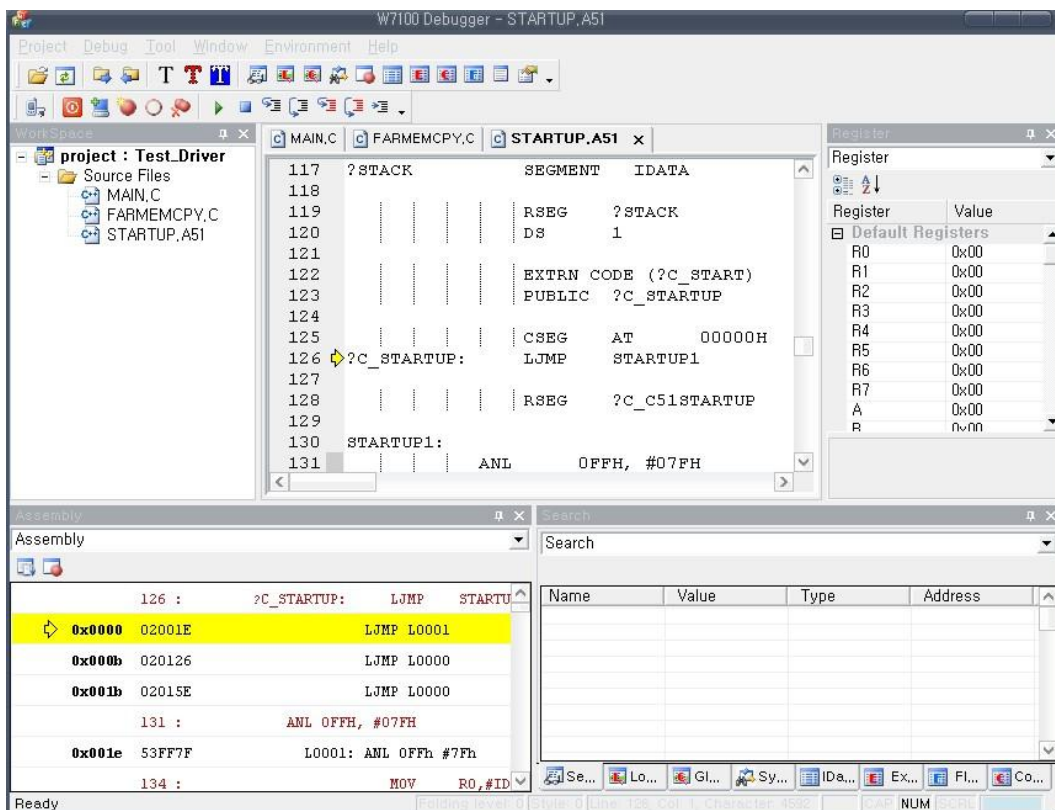


<Fig.5.1> Open project



<Fig.5.2> File open browser

File browser에서 debugging을 위한 project의 HEX file을 선택하여 더블 Click하거나 'open'를 실행하면 Fig5.3과 같이 debugging을 위한 열기가 완료되며 'Assembly window'에 assemble code를 확인 할 수 있다.




<Fig.5.3> Complete opening the project

*주의 : project open시 아래와 같은 error message가 나오면 project 디렉터리 내의 SRC 파일을 지우고 'Generate Assembler SRC File' 와 'Assemble SRC File' 옵션을 확인 하여야 한다. 자세한 내용은 section 4 KEIL Project에 언급되어 있다.



<Fig.5.4> Open error message

Reload project

Load 된 project가 변경이 되었을 때 Project를 reload 하기 위해서는  Reload Tool bar 버튼을 click 하면 된다. 또한, Project Menu에서 Reload를 선택 하여 project를 reload 할 수도 있다. (short key:F4)

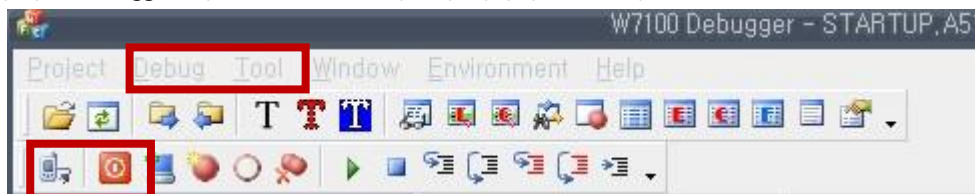
Close project

Project가 열려 있는 상태에서 다른 project를 열기 위해서는 먼저 Project를 close시켜야 하며, close 하기 위해서는 Project Menu에서 close를 선택 하면 된다. (short key : Ctrl + Q)

*주의사항: Debugger의 상태가 continue 일 때는, project를 close 할 수 없으므로, stop 한 후에 close를 해야 한다.


5.2 Initialize the W7100A / W7100 debugger

지금부터는 debugger 와 iMCU7100EVB 기능에 대하여 설명한다.



<Fig.5.5> Initialize menu

Debugger and board reset

W7100A / W7100 debugger와 보드를 함께 초기화 한다. 만약 브레이크 포인터가 설정되어 있었다면 브레이크 포인터 모두 초기화 된다. 위 그림에서 Tool bar에 있는  버튼 또는 Tool menu의 Init Emulator and Board reset 메뉴를 선택하면 W7100A / W7100 debugger 및 보드를 함께 초기화 할 수 있다.

Board reset only

W7100A / W7100 debugger는 리셋하지 않고 보드만 리셋 한다. 만약 브레이크 포인터가 설정되어 있



있다면 브레이크 포인터는 그대로 유지된다. Tool bar에 있는  버튼 또는 Debug menu의 Board reset 메뉴를 선택하여 보드만 초기화 할 수 있다 (short key : F2)

Image load

디버깅에 사용할 HEX 이미지를 보드의 플래시 메모리에 write 한다. Tool bar에서  버튼 또는 Debug menu의 Image Load menu를 선택하며 Image load를 실행할 수 있다. 정상적으로 Write했고 이미지를 다시 보드에서 받아와서 검증까지 완료 되었다면 “Write OK”라는 메시지 창이 뜬다.

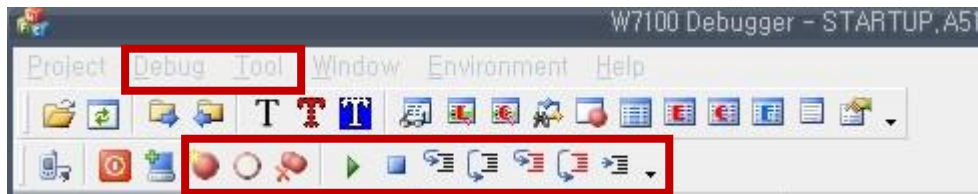
-참고: HEX 파일을 open 후 Continue, Step, Next등과 같은 debugger 명령을 수행했을 때 Fig.5.6.과 같은 ‘Load image mismatch’ 경고 창이 나타난다면 현재 HEX 파일과 보드의 이미지 파일이 서로 일치하지 않는 경우이니 이미지 로드 명령을 수행해준다.



<Fig.5.6> Load image mismatch

Debugging

이번 과정에서는 디버깅을 위한 전반적인 기능들을 소개한다. debugger에서 제공하는 각종 기능들은 Fig.5.7에서 보이는 것과 같이 “Debug” 메뉴를 사용하거나 Tool bar 버튼을 이용하여 각종 디버깅 명령을 내릴 수 있다.



<Fig.5.7> Deugging menu

Go to Source Line

Go to Source Line 명령은 원하는 source code의 line으로 커서를 이동시키기 위해 사용한다. ‘Tool’ menu -> ‘Go to Source Line’ 또는 short key Ctrl + G을 이용하여 Go to Source Line 명령을 실행한다.


Find

Find 명령은 ‘source window’에서 특정 word를 찾기 위해 사용한다. ‘Tool’ menu -> ‘Find’ 또는 short key Ctrl + F을 이용하여 Find 명령을 실행한다.


5.3 Run and Stop

이번 장에서는 stem in, out, next 등의 명령어의 사용법에 관해 설명한다.


Continue

Continue 명령은 브레이크 포인트를 만날 때까지 프로그램을 수행하는 명령이다. Tool bar에서  버튼을 누르거나 Debug menu->Continue 메뉴를 선택하면 Continue 명령을 수행할 수 있다. (short key : F5) Continue 상태에서는 board에 debugging을 위한 명령을 주지 못합니다. 그러므로, break point나 memory load 등의 명령을 내리기 위해서는 항상 Stop 상태로 변경 시켜 주어야 합니다. Continue 상태에서 Stop 상태로 변경시켜주기 위해서는 아래 나오는 Stop 명령을 수행하면 된다.

Stop


Stop 명령은 Continue 상태인 프로그램을 정지 시킨다. Tool bar에서  버튼을 누르거나 Debug menu-> Stop 메뉴를 선택하면 Stop 명령을 내릴 수 있다. Stop 명령이 내려지면 W7100A / W7100 debugger는 Board에서 실행중인 프로그램을 정지 시킨다. (short key : F6) Stop 상태여야만 debugger가 보드를 디버깅 할 수 있는 각종 명령(Ex: Search, memory view, break point , etc)을 내릴 수 있다.

Step


Step 명령은 C code의 1 line을 기준으로 실행을 하고, Function call이 있다면, 해당 function으로 진입하게 된다. C code 1 line을 수행 후에 debugger는 stop 상태가 되고 다음 명령을 기다리게 된다. Step 명령은 Tool bar에서  버튼을 누르거나 Debug menu-> Step을 선택하여 명령을 내릴 수 있다. (short key : F11) 만약 C 소스 없이 HEX 파일만 가지고 디버깅을 한다면 Step 명령은 Step1 명령과 동일하게 한 instruction만을 수행한다.

Next


Next명령은 C code의 1 line을 기준으로 실행을 하고, Function call이 있다면 해당 function을 수행하지만 해당 function으로 진입하지는 않는다. C code 1 line을 수행 후에는 debugger는 stop 상태가 되고 다음 명령을 기다리게 된다.

Tool bar에서  버튼을 누르거나 Debug menu-> Next 메뉴를 선택하여 Next 명령을 수행할 수 있다. (short key : F10) Next 명령은 Step 명령과 마찬가지로 C 소스 파일 없이 HEX 파일만 가지고 디버깅 할 시에는 Next1과 같이 한 instruction만 수행한다.

Step1

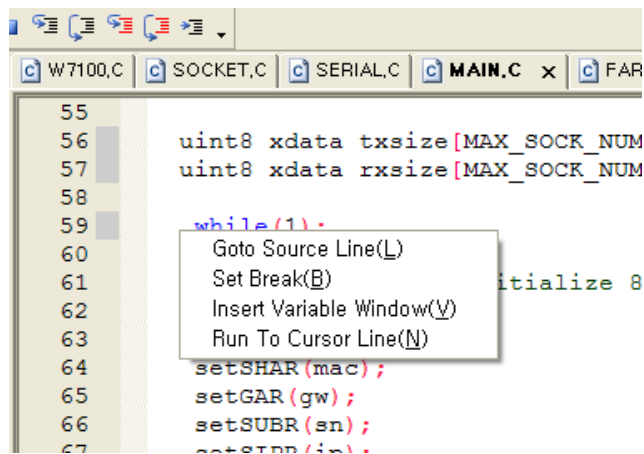
Step1 명령은 하나의instruction 을 실행을 하고 instruction이 function call이라면 해당 function으로 진입하게 된다. Instruction 1 line을 수행 후에는 debugger는 stop 상태가 되고 다음 명령을 기다리게 된다. Tool bar에서  버튼을 누르거나 Debug menu-> Step1 메뉴를 선택하면 Step1 명령을 수행 할 수 있다.

Nexti

Nexti 명령은 하나의 instruction을 실행을 하고 instruction이 function call이라도 해당 function을 수행하긴 하지만 해당 function 안으로 진입하지는 않는다. Instruction 1 line을 수행 후에는 debugger는 stop 상태가 되고 다음 명령을 기다리게 된다. Tool bar에서  버튼을 누르거나 Debug menu->Nexti를 선택하면 Nexti 명령을 수행 할 수 있다. (short key : F6)

Run to cursor line

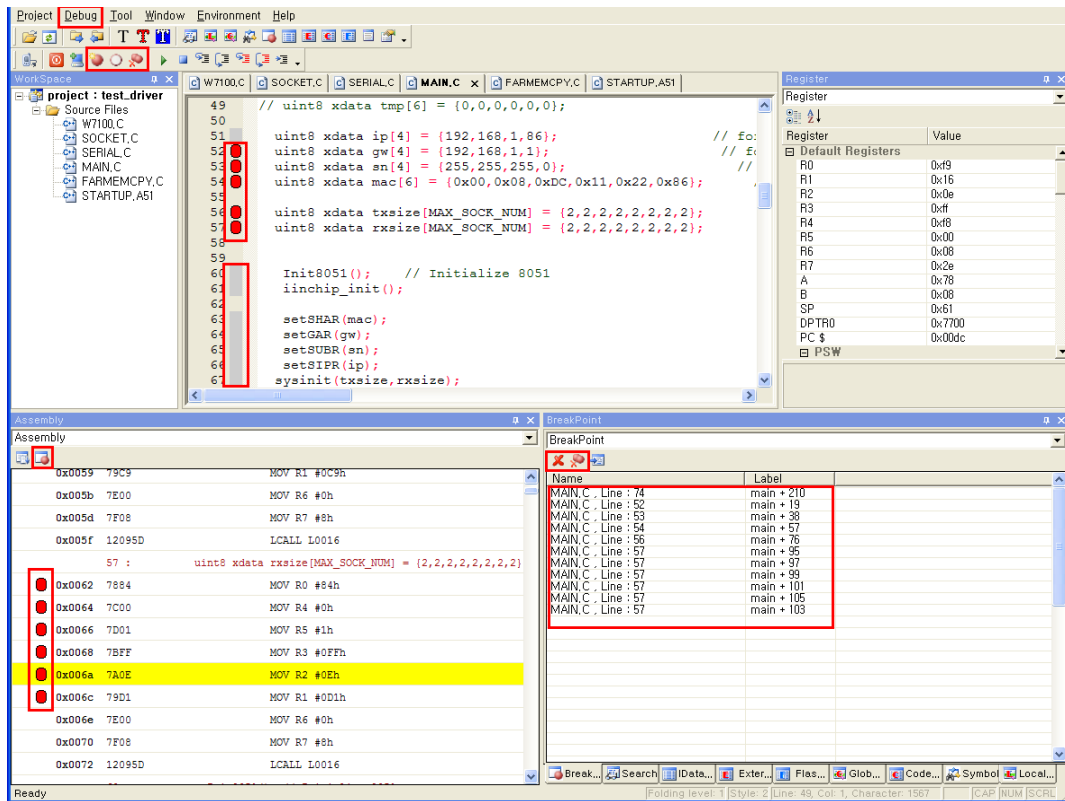
Run to cursor line 명령은 W7100A / W7100 debugger 소스 창에 표시되어 있는 cursor 위치까지 프로그램 수행한 후 멈춘다. W7100A / W7100 debugger의 'Source window' 에서 오른쪽 버튼을 click시 나오는 메뉴에서 Run To Cursor Line메뉴(Fig.5.8)를 선택 하거나 Debug Menu->Run To Cursor Line을 선택하여 실행 할 수 있다. 만약 해당 라인으로 프로그램이 오지 않는다면 Run To Cursor Line 명령은 계속 Continue 상태일 수 있다. 이럴 때는 Stop 명령을 내려 멈추면 된다.



<Fig.5.8> Run To Cursor Line command

5.4 Break Point




다음은 브레이크 포인트와 관련된 내용이다. 브레이크 포인트를 debugger에서 설정/해제 하는 방법과 설정된 브레이크 포인트를 확인하는 방법 브레이크 포인트 사용시 주의 사항 등을 설명한다. 먼저 브레이크 포인트를 설정하기 위해서는 Stop 상태에서만 가능하다. 때문에 현재 Continue 상태라면 Stop 명령을 내려 Stop 상태로 전환 후 브레이크 포인트를 설정한다.



<Fig.5.9> Break point of the debugger

Set break point

브레이크 포인트를 설정하는 방식에는 다음과 같은 방식이 있다.





1. 소스 창의 좌측 라인번호 옆에 있는  부분을 click.
2. Toolbar에서  버튼을 click
3. 소스 창에서 Break point를 설정하고자 하는 line에 cursor를 위치시키고Debug Menu->Set break메뉴를 click
4. 소스 창에서 break point를 설정하고자 하는 line을 double click
5. 'Assembly window'에서 break point를 설정하고자 하는 line을 double click.
6. 'Assembly window'에서 break point를 설정하고자 하는 line을 select 한 후 'Assembly window'의 Tool bar에서  버튼을 click
7. Short key F9

브레이크 포인터를 설정하고 continue 명령을 내리면 프로그램 수행 중 브레이크 포인터에 해당하는 주소로 프로그램 카운터가 이동하면 브레이크가 걸려 debugger가 Stop 상태로 된다. 이후 Stop 상태에서 각종 debugger 명령을 내리며 디버깅 하는 것이 가능하다.

만약 브레이크 포인터로 설정된 주소로 프로그램 카운터가 이동하지 않는다면 계속 Continue 상태일 수가 있다. 이때는 Stop 명령을 내려 Stop 상태로 전환하는 것이 가능하다.

Clear Break



브레이크 포인터를 해제하는 방식에는 다음과 같은 것들이 있다.

1. 소스 창의 좌측 브레이크 포인터 설정 라인에서  부분을 click
2. Toolbar에서  버튼을 click
3. 소스 창에서 Break point를 해제하고자 하는 line에 cursor를 위치시키고 Debug Menu->Clear break 메뉴를 click.
4. 소스 창에서 break point를 해제하고자 하는 line을 double click
5. Assembly window에서 break point를 해제하고자 하는 line을 double click.
6. Assembly window에서 break point를 해제하고자 하는 line을 select 한 후 Assembly window의 toolbar에서  버튼을 click
7. Break point window에서 해제하고자 하는 line을 select한 다음 Break point window의 tool bar에서  버튼을 click.
8. Short key F9

위 방법 중 한 방법을 선택하면 브레이크 포인터를 없애는 것이 가능하다.

Clear All Breaks

모든 브레이크 포인터를 제거하는 명령이다. Clear break all 명령을 내리는 방법에는 다음과 같은 것들이 있다.

- Toolbar에서  버튼을 Click
- Break point toolbar에서  버튼을 Click
- 'Debug' menu 에서 'Clear All Breaks'를 click

Goto Next Break Point


현재의 브레이크 포인터에서 다음번에 존재하는 브레이크 포인터로 이동하기 위한 명령어이다.

Tool bar에서  버튼을 click

'Tool' menu에서 'Goto Next Break Point' 선택

Goto Previous Break Point

현재의 브레이크 포인터에서 이전에 존재하는 브레이크 포인터로 이동하기 위한 명령어 이다.

Tool bar에서  버튼을 click

'Tool' menu에서 'Goto Previous Break Point' 선택

Break Point Window

브레이크 포인터 창에서는 현재 설정된 모든 break point의 목록을 나타낸다. 브레이크 포인터 창에 있는 Tool bar 버튼들은 다음과 같은 기능을 가지고 있다.

- : 선택된 break point의 삭제
- : 모든 break point의 제거
- : 선택된 break point로 소스코드 이동.

Source and Assembly window

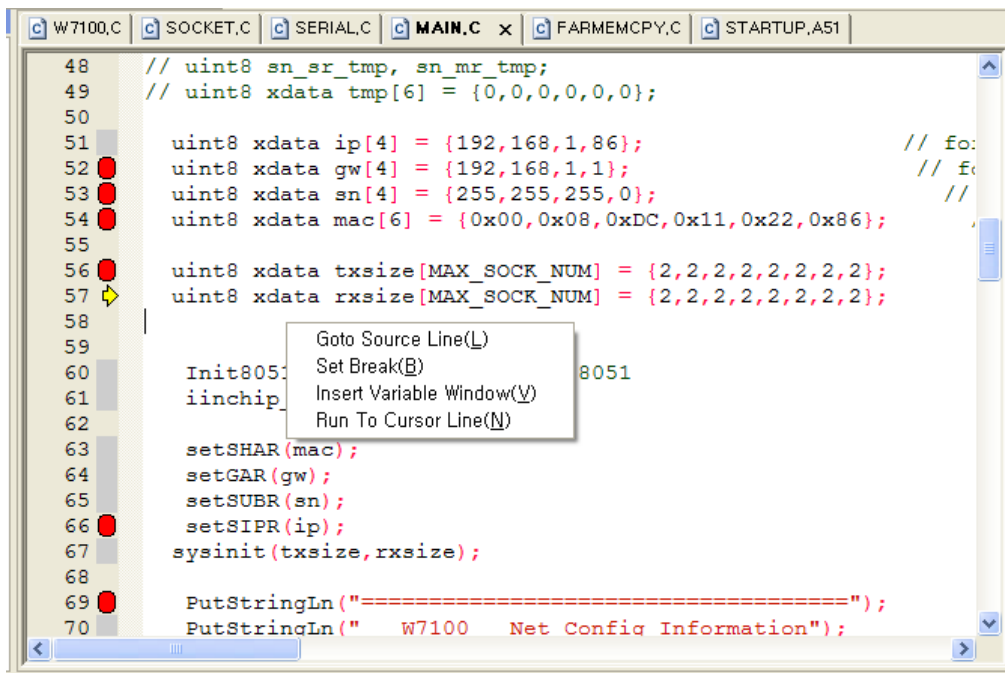
소스 창과 'assembly window'에서는 실행중인 프로그램의 PC가 해당하는 소스와 assemble code 를 볼 수 있고, 브레이크 포인터를 설정/해제 하기 위한 명령을 내릴 수 있다. 그리고 소스 창에서는 각종 변수의 값을 확인할 수 있고, 해당 변수를 'search window'에 등록할 수도 있다.



<Fig.5.10> Window menu




'assembly window'을 보기 위해서는 debugger Tool bar에서 버튼을 누르거나 Window Menu의 Assembly menu 선택하면 되고 소스 창은 소스 없이 HEX 파일만 open 하지 않는 한 기본적으로 나타난다.

Source Window



<Fig.5.11> Source Window menu

Fig.5.11은 소스 창 의 모습을 보여준다. 위 그림에서 각각의 아이콘의 의미는 다음과 같다.

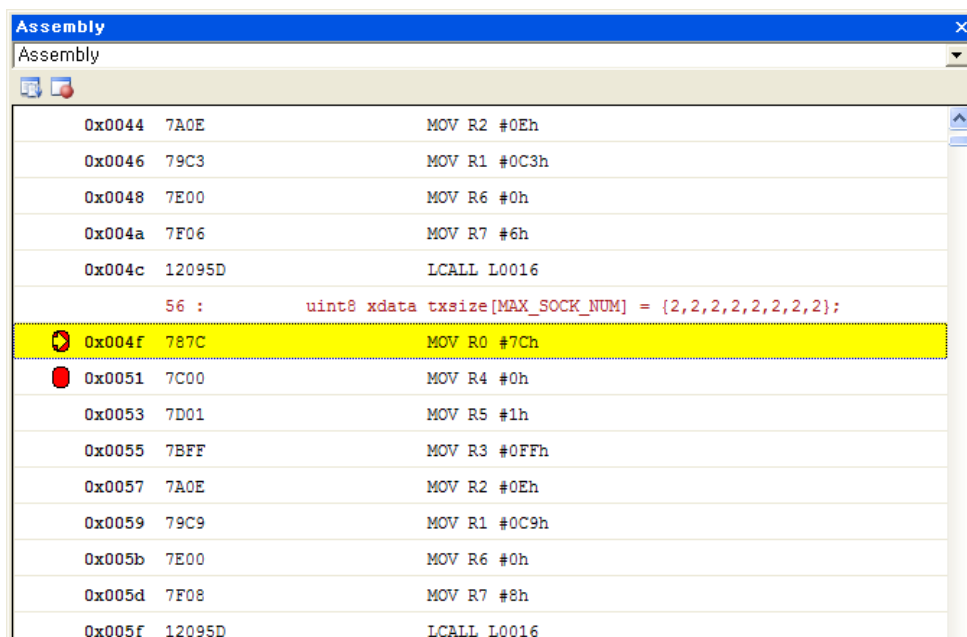
- : source code에서 PC의 위치를 표시합니다.
- : break point가 설정되었음을 나타낸다.
- : break point를 설정할 수 있는 위치를 표시한다.

그리고 소스 창에서 마우스 오른쪽 버튼을 Click하면 Fig.5.11과 같이 소스 창 메뉴가 나타나는데 그 각각의 메뉴가 의미하는 바는 다음과 같다.

- Goto source line: 해당 라인으로 cursor 이동
- Set break: 현재의 cursor가 위치한 line에 break point 설정
- Insert Variable window: cursor가 위치한 변수를 Search window에 등록
- Run To Cursor Line: cursor가 위치한 곳까지 수행 기타 소스 창이 지원하는 기능은 다음과 같다.
 - Code를 double click시에 break point를 설정할 수 있다.
 - 변수 위에 마우스 포인터를 올려 두면 tooltip으로 변수의 값을 보여 준다.




Assembly window

‘assembly window’은 현재 수행중인 어셈블리 코드를 보여주는 창이다. 더불어서 소스 창에서 특정 소스 라인을 Click하면 Click한 라인에 해당하는 어셈블리 코드를 보여주기도 한다. C 소스 없이 HEX 파일만 로딩하게 되면 소스 창이 사라지고 ‘assembly window’만이 보이게 된다. ‘assembly window’은 현재 수행중인 어셈블리 코드를 보여주는 창이다. 더불어서 소스 창에서 특정 소스 라인을 Click하면 Click한 라인에 해당하는 어셈블리 코드를 보여주기도 한다. C 소스 없이 HEX 파일만 로딩하게 되면 소스 창이 사라지고 ‘assembly window’만이 보이게 된다.



<Fig.5.12> Assembly window

Fig.5.12는 ‘assembly window’에서 각 아이콘들의 의미와 ‘assembly window’의 기능은 다음과 같다.

- : Instruction 에서 PC의 위치를 표시
- : break point를 표시
- : Assembly window에서 assembly만 출력할지, c code와 함께 출력할지를 선택
- Instruction double click: 선택된 line의 break point를 설정 또는 해제

6 Search & Variable Window






이번 장에서는 지역 변수 , 전역 변수, memory의 value를 확인할 수 있는 기능에 대해 설명한다.
W7100A / W7100 debugger에는 각종 변수의 값을 확인하기 위해 다음과 같은 창을 제공한다.

- Global variable window : 현재 project의 전역 변수와 값을 보여줌
- Local variable window : 현재 PC에서 지역 변수와 값을 보여줌
- Search window : 현재 PC에서 사용자가 등록한 전역 변수,
또는 지역 변수 또는 메모리 값이 변하는 것을 보여줌
- Register window : 현재 PC에서 레지스터 값을 보여줌
- Symbol window : 현재 project의 각종 심볼 정보를 보여줌

각각의 창을 활성화 시키기 위해서는 그림 29에서 보이는 바와 같이 Window 메뉴에서 해당하는 메뉴를 선택하거나 Tool bar에서 각종 창 버튼을 Click하면 된다.

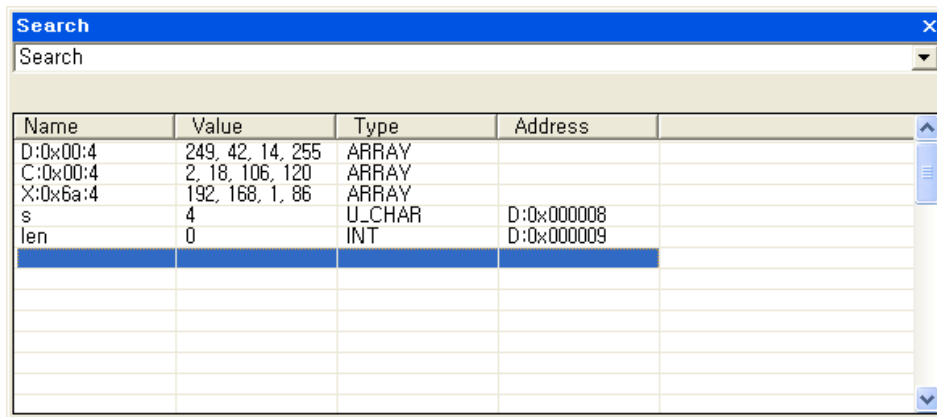


<Fig.5.13> Search window

- Search window 활성화:  또는, Window Menu의 Search Window선택.
- Local variable window 활성화:  또는, Window Menu의 Local Variable Window선택.
- Global variable window 활성화:  또는, Window Menu의 Global Variable Window선택.
- Symbol window 활성화:  또는, Window Menu의 Global Variable Window선택.
- Register window 활성화:  또는, Window Menu의 Register Window 선택.

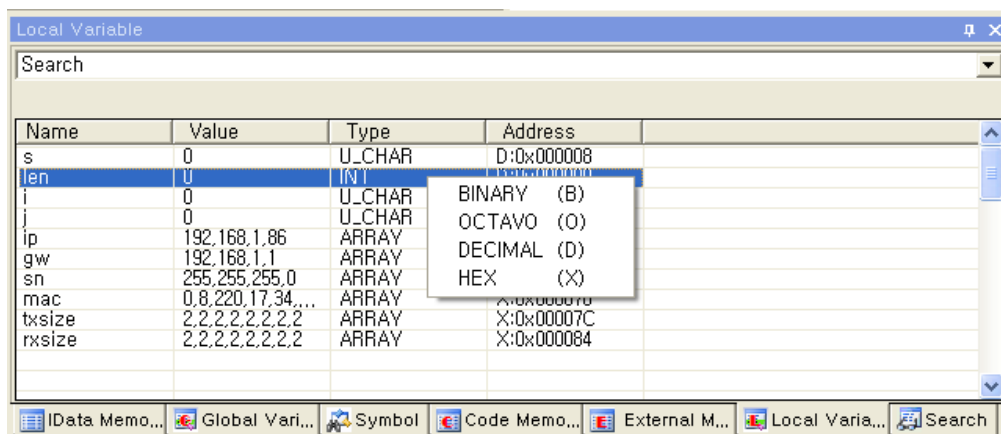
6.1 Search Window

‘search window’ 은 사용자가 등록한 전역 및 지역 변수 혹은 메모리 값을 보여주는 창이다. 전역/지역 변수 창을 보면서 디버깅해도 되지만 보통 디버깅 할 때는 모든 변수의 값을 알고 싶은 것이 아니라 원하는 한 개 또는 두 개의 변수 값만 확인하면 되는 경우가 많다.이러한 경우 확인을 원하는 변수를 ‘search window’ 창에 등록해놓으면 해당 변수가 변하는 것을 손쉽게 확인할 수 있다.



<Fig.6.1> Search windows

전역/지역 변수를 ‘search window’에 등록하기 위해서는 Name field에 확인하고 싶은 전역/지역 변수명을 입력 하면 됩니다. 입력할 때는 마우스로 해당 필드를 Click하면 입력 가능하게 된다. ‘Search window’에서는 전역/지역 변수와 메모리 값을 볼 수 있는 것 뿐만 아니라 값을 변경하는 것도 가능하다. 등록 할 때와 마찬가지로 해당 값 필드에 마우스 Click 후 값을 입력하면 된다.



<Fig.6.2> Display format

‘Search window’에 메모리 값을 보기 위해서는 Name field에 Memory type:Address:Size 형식으로 입력 하면 된다. Memory type은 세 종류로 C: code memory, D: internal memory, X: external memory 가 있다.

Ex) Code memory => C:0x1000:5

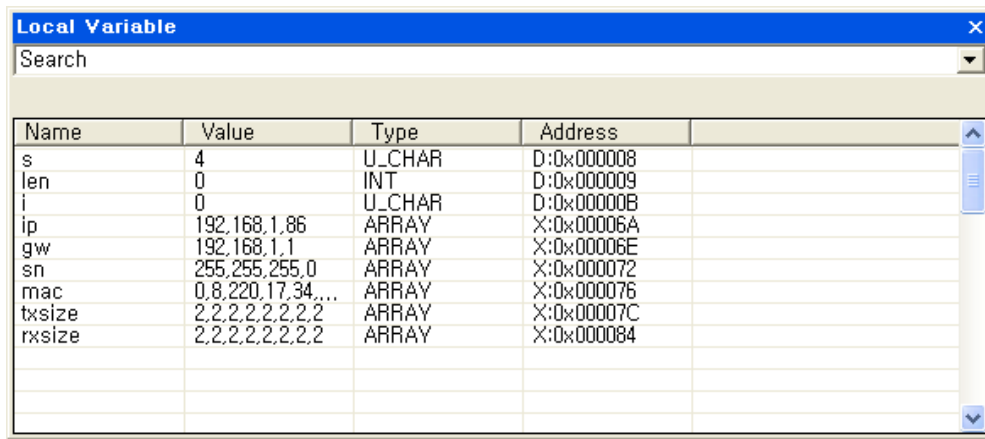
Internal memory => D:0x30:2

External memory => E:0x000300:4

그리고 'search window', 지역/전역 변수 창에서 마우스 오른쪽 Click을 하면 위 그림과 같이 출력되는 형식을 2진, 8진, 10진, 16진수로 변경하는 것이 가능하다.

6.2 Local Variable Window

지역 변수 창은 PC가 해당하는 현재 함수에서 지역 변수들과 그 값을 보여준다. 지역 변수창의 사용법은 'search window'의 사용법과 거의 동일하다. 단, 지역 변수 창에는 새로운 변수나 메모리 주소를 등록 할 수 없다.

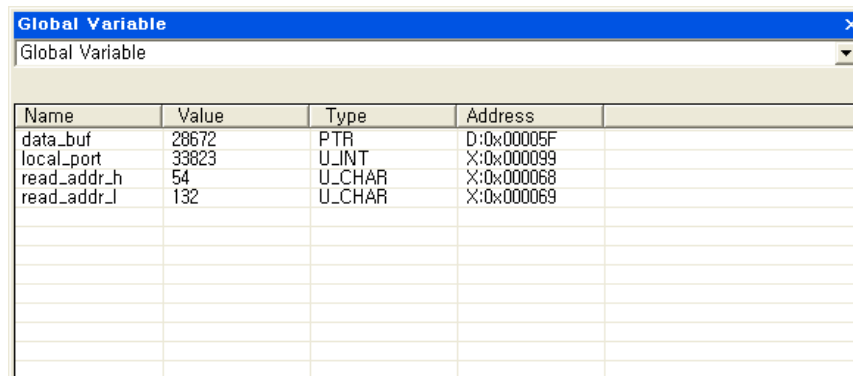


Name	Value	Type	Address
s	4	U_CHAR	D:0x000008
len	0	INT	D:0x000009
i	0	U_CHAR	D:0x00000B
ip	192.168.1.86	ARRAY	X:0x00006A
gw	192.168.1.1	ARRAY	X:0x00006E
sn	255,255,255,0	ARRAY	X:0x000072
mac	0.8.220.17.34,...	ARRAY	X:0x000076
txsize	2,2,2,2,2,2,2,2	ARRAY	X:0x00007C
rxsize	2,2,2,2,2,2,2,2	ARRAY	X:0x000084

<Fig.6.3> Local variable window

6.3 Global Variable Window

전역 변수 창은 현재 project의 모든 전역 변수들과 그 값을 보여준다.

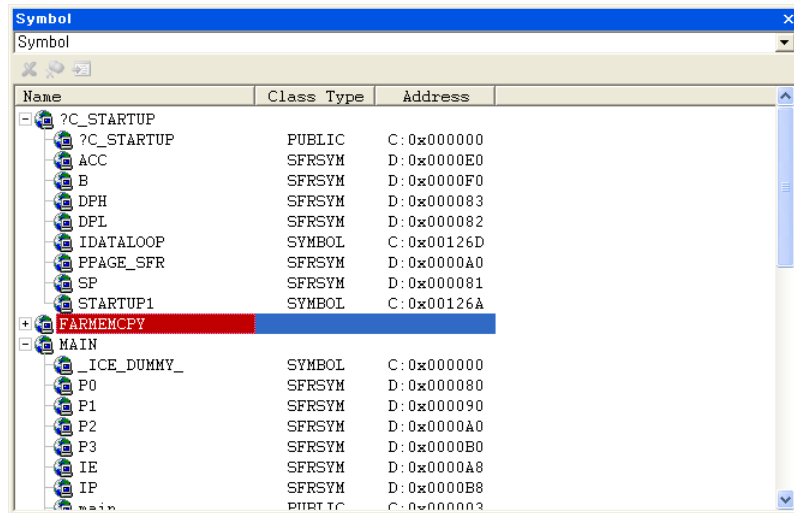


Name	Value	Type	Address
data_buf	28672	PTR	D:0x00005F
local_port	33823	U_INT	X:0x000099
read_addr_h	54	U_CHAR	X:0x000068
read_addr_l	132	U_CHAR	X:0x000069

<Fig.6.4> Global variable window

6.4 Symbol window

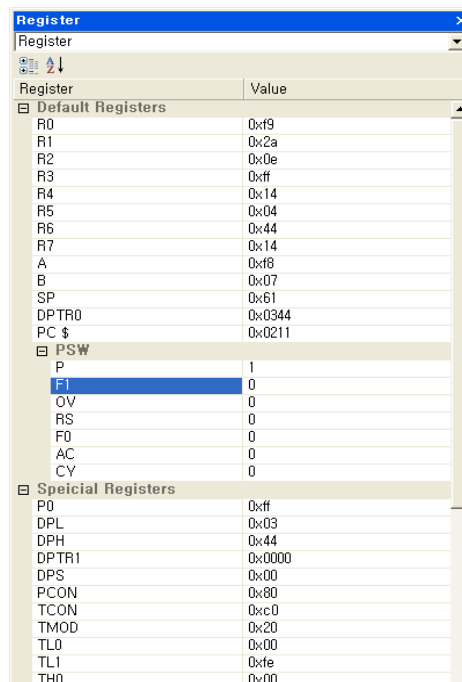
심볼 창에는 현재 project에 있는 각종 심볼들의 정보를 보여준다. 각종 심볼들의 class type과 주소를 확인하려면 심볼 창을 참조하면 된다.



<Fig.6.5> Symbol window

6.5 Register window

레지스터 창에서는 현재 레지스터 값을 확인하거나 변경할 수 있다. 레지스터 창에는 범용 레지스터인 R0-R7 외에도 SFR 레지스터들도 함께 보여준다. 레지스터 창의 레지스터 값을 변경하려면 변경하고자 하는 레지스터의 값 필드를 마우스 Click하고 입력 상태일 때 값을 입력하면 된다. 이때 해당 레지스터가 쓰기 가능한 레지스터라면 값이 변경된다.



<Fig.6.6> Register window

7 Memory Window

Notice: W7100A / W7100 debugger에서 iMCU7100EVB의 메모리값을 읽어오려면 먼저 KEIL μ Vision project 혹은 HEX파일을 open한 상태여야 한다. 이 때, open할 KEIL μ Vision project 혹은 HEX파일은 iMCU7100EVB에 load되어있는 이미지와 달라도 상관없다.





The iMCU7100EVB debugger supports below memory windows. 다음은 iMCU7100EVB debugger 에서 제공하는 각종 메모리 창에 대한 설명이다. iMCU7100EVB debugger 는 아래와 같은 메모리 창들을 제공한다.

- IData memory window : 내부 데이터 메모리 값을 보여줌.
debugger 명령을 내린 후 stop 상태일 때마다 매번 update 됨
- External memory window : 외부 데이터 메모리 값을 보여줌
- Code memory window : 코드 메모리 값을 보여줌
- Flash memory window : flash 데이터 메모리 값을 보여줌

각각의 창들을 활성화하기 위해서는 Fig.7.1 과 같이 debugger 메뉴 중 “Window” 메뉴에서 활성화 여부를 선택할 수 있다.

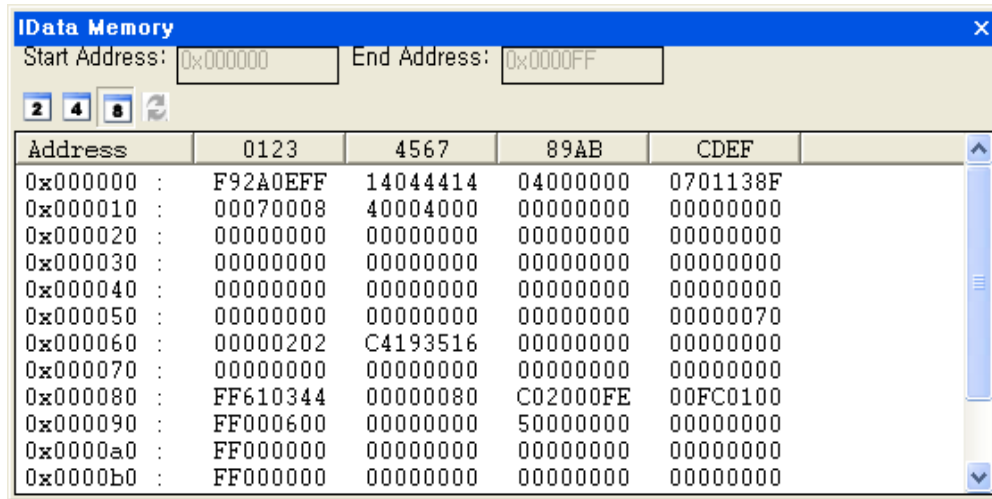


<Fig.7.1> Memory window

- IData memory window 활성화: 
또는, Window Menu->IData Memory Window 선택
- External memory window 활성화: 
또는, Window Menu->External Memory Window 선택
- Code memory window 활성화: 
또는, Window Menu->Code Memory Window 선택.
- Flash memory window 활성화: 
또는, Window Menu->Flash Memory Window 선택.

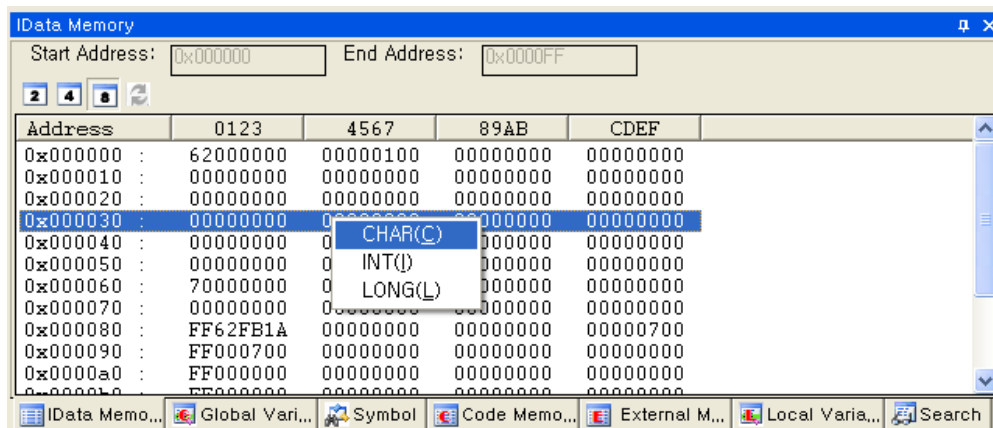
7.1 IData Memory Window

내부 데이터 메모리 창은 내부 데이터 메모리(D:0x00 ~ D:0xFF)의 값을 보여주거나 변경하는 것이 가능하다.



<Fig.7.2> IData Memory window

내부 데이터 메모리의 값을 변경하기 위해서는 변경하고자 하는 내부 메모리 값 필드를 마우스 Click 후 입력 상태일 때 바꾸고자 하는 값을 적어 넣으면 된다. 내부 메모리뿐만 아니라 모든 메모리 창들은 마우스 오른쪽 Click으로 아래 그림과 같이 출력 단위를 1/2/4 바이트로 변환할 수 있다. 출력 단위 변환은 각 메모리 창의 Tool bar에서도 가능하다.

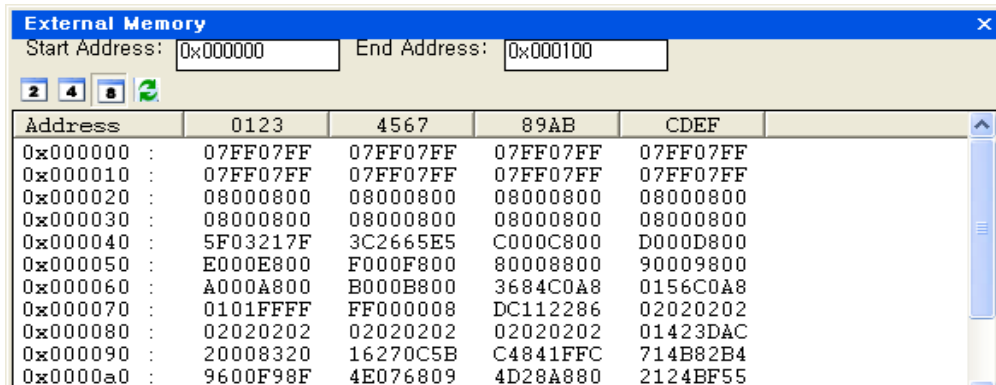


<Fig.7.3> Change the display format


내부 데이터 메모리의 값은 매 debugger 명령이 발생시 stop 상태가 되면 실제 보드의 값으로 매번 update 한다.

7.2 External Data Memory Window


외부 데이터 메모리 창은 X:0x000000 부터 X:0xFFFFFFFF 범위에 해당하는 외부데이터 메모리 값을 보여준다.

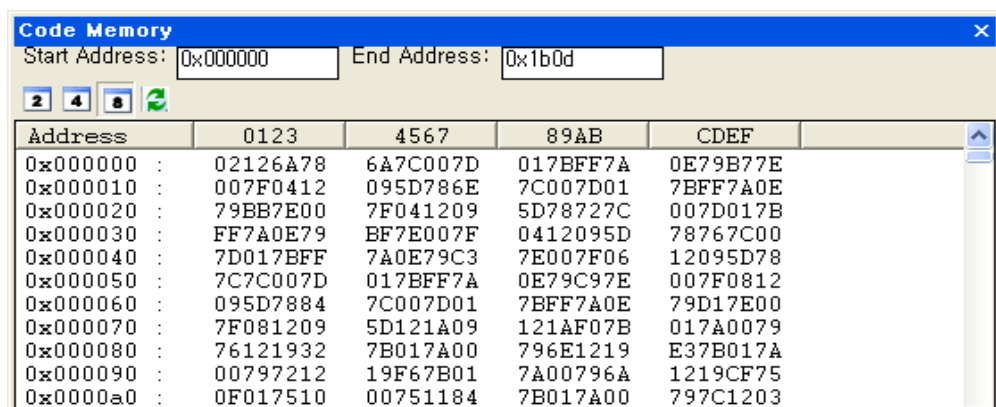


<Fig.7.4> External Data memory window

외부 데이터 메모리는 내부 데이터 메모리에 비해 사이즈가 크기 때문에 매 디버깅 명령시 마다 값을 받아오게 하면 디버깅 명령 속도가 떨어지게 된다. 때문에 외부 데이터 메모리의 값을 읽어오기 위해서는 Tool bar에서  버튼을 Click하여야 한다. 외부 데이터 메모리 역시 내부 데이터 메모리와 마찬가지로 값을 변경하게 되면 실제 보드의 값이 바로 바뀌게 된다. 그리고 외부 데이터 메모리는 가져올 데이터의 범위를 X:0x0 ~ X:0xFFFFFFFF 범위 내에서 자유롭게 정해줄 수 있다.

7.3 Code Memory Window



코드 메모리 창은 코드 메모리의 값을 보여준다. 코드 메모리는 읽기 전용이기 때문에 값을 변경할 수는 없다. 코드 메모리 창 역시 실제 보드의 값을 받아오려면 Tool bar에 있는  버튼을 눌러 코드 메모리 값을 받아와야 한다.

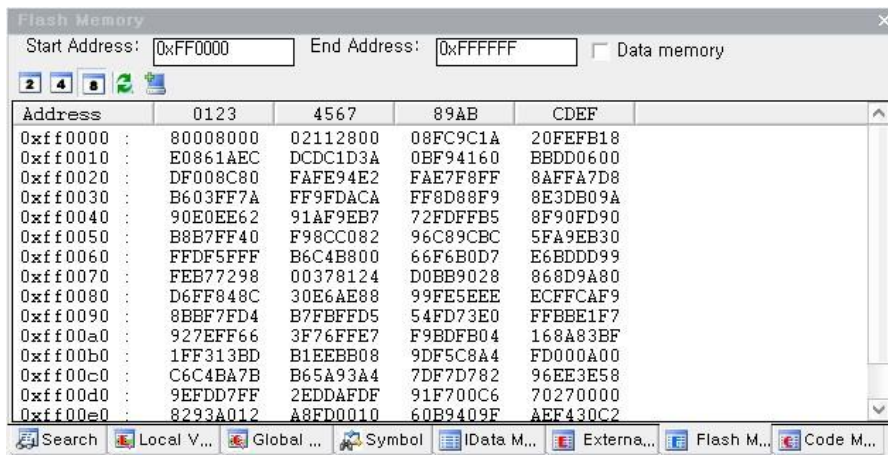


<Fig.7.5> Code Memory window

7.4 Flash Memory Window

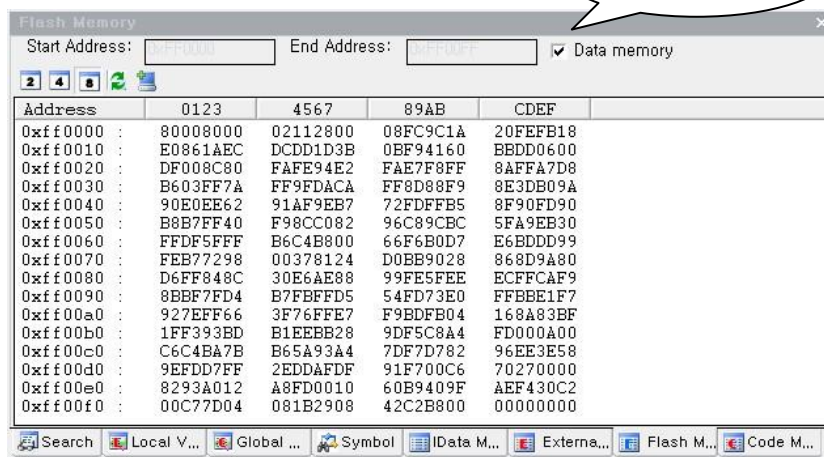
7.4.1 Code Memory Domain

플래시 메모리 창은 X:0xFF0000 ~ X:0xFFFFF 영역의 플래시 메모리 영역의 값을 보여준다. 플래시 메모리 창이 다른 메모리 창과 다른 점은 플래시 메모리 창에서 값을 변경하고 Tool bar의  버튼을 누르면 플래시 메모리에 변경된 값을 write 한다는 점이다. 이때는 변경된 값만 write 하지 않고 전체 값을 다 write 한다. 플래시 메모리 역시 현재 플래시 메모리 값을 받아오기 위해서는 Tool bar에 있는  버튼을 눌러 플래시 메모리 값을 새롭게 받아와야 한다.





<Fig.7.6> Flash memory window for code memory

7.4.2 Data Memory Domain



<Fig.7.7> Flash memory window for data memory

‘Data memory’ 옵션이 체크 된 경우, 플래시 데이터 메모리 (0x00-0xFF) 의 값을 확인 할 수 있다. ‘Flash memory window’의 인터페이스는 다른 ‘memory windows’와 기본적으로 같다. 하지만, ‘Flash memory window’는 모든 flash memory의 값을  버튼을 이용하여 write 할 수 있다. 또한, flash memory의 값을  버튼을 이용하여 update 할 수 있다.

Document History Information

Version	Date	Descriptions
Ver. 0.9Beta	2009. 10	Release with W7100 launching
Ver. 0.91	2009. 12	Modify section2 "Connect the debugger". Add about debugger reset button.
Ver. 1.0	2011. 6	Release with W7100A launching
Ver. 1.1	Jun, 2012	Fixed some awkward expressions of English documents

Copyright Notice

Copyright 2012 WIZnet Co.,Ltd. All Rights Reserved.

Technical Support: support@wiznet.co.kr

Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at <http://www.wiznet.co.kr>