

W7100A / W7100 Debugger Guide

Version 1.1



© 2012 WIZnet Co., Inc. All Rights Reserved. For more information, visit our website at <u>http://www.wiznet.co.kr</u>

1



Table of Contents

1	Drive	er installa	tion for the Debugger	3			
2	Connect the Debugger						
3	Installation of the W7100A / W7100 Debugger 7						
4	KEIL	Project		8			
5	Debu	igger men	าน				
	5.1	Oper	n project				
	5.2	Initia	alize the W7100A / W7100 Debugger				
	5.3	Run	and Stop				
	5.4	Brea	ık Point				
6	Searc	ch & Varia	able Window	24			
	6.1	6.1 Search Window					
	6.2 Local Variable Window						
	6.3 Global Variable Window						
	6.4	Syml	bol window				
	6.5	Regi	ster window	27			
7	Mem	ory windo	DW				
	7.1 IData Memory Window						
	7.2 External Data Memory Window						
	7.3 Code Memory Window						
	7.4	Flash	h Memory Window	30			
		7.4.1	Code Memory Domain				
		7.4.2	Data Memory Domain				



1 Driver installation for the Debugger

Connect the Debugger and PC with USB cable, and then the 'new hardware search window' will open as figure 1.1 below.

Select 'Automatically setup the software (I)' and click the Next (N) button.



[Fig.1.1] The new hardware search window 1

If it is not automatically installed, select the 'Install from a list or specific location (Advanced) and set our attached driver as below.

Found New Hardware V		
Please choose your so	Browse For Folder Select the folder that contains drivers for your hardwar	re.
 Search for the bes Use the check box paths and removat Search removed Include this D:\ Don't search. I will Choose this option the driver you choose 	ancamera CompanyWork CompanyWork CDM 2.04.16 WHQL Certified amd64 386 Second Beeys Second Bee	hat
	To view any subfolders, click a plus sign above.	

[Fig.1.2] The new hardware search window 2

Wait for searching the device driver of the Debugger. When the device driver installation is completed,



click the END button to finish.

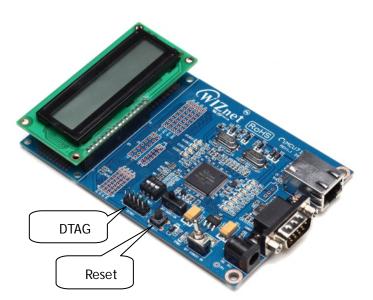
Found New Hardware Wizard							
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: USB Serial Converter						
	Click Finish to close the wizard.						
	< Back Finish Cancel						
Found New Hardware Wiz	ard						
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: USB Serial Port						
	Click Finish to close the wizard.						
	< Back Finish Cancel						

[Fig.1.3] Completing the new hardware searching #2

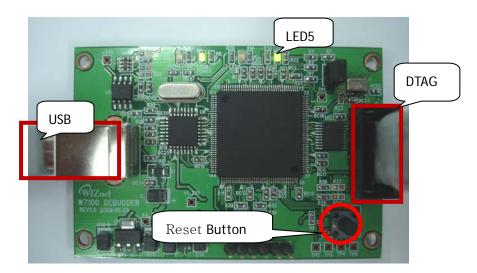


2 Connect the Debugger

This section will explain how the Debugger is connected with iMCU7100EVB. Connect the Debugger to the DTAG of IMCU7100EVB.



[Fig.2.2] iMCU7100EVB



[Fig.2.2] W7100A / W7100 Debugger

Connect the other DTAG cable between the Debugger and iMCU7100EVB. The red line of DTAG cable must be connected to line number 1 of iMCU7100A DTAG socket. Also connect the USB cable between the Debugger and PC. Next, insert the power cable to the IMCU7100EVB, and push the EVB reset button as figure 2.1. Then, push the debugger reset button of the Debugger. Confirm the LED5 (RUN LED) after pushing these reset button.



If the Debugger successfully recognizes the W7100A / W7100, the LED5 (RUN LED) will turn on. But if the LED5 is not turned on, it means that W7100A / W7100 is not recognized. Check the connecting cable or try to reset the IMCU7100EVB and the Debugger. The LED5 should be turned on before using the W7100A / W7100 Debugger.



3 Installation of the W7100A / W7100 Debugger

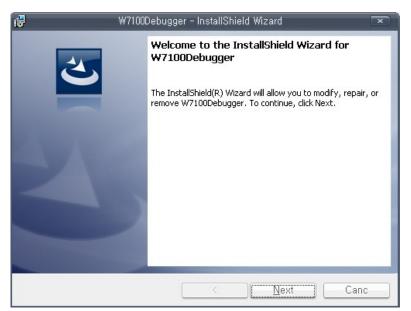
In this section, the installation of W7100A / W7100 Debugger is explained. First, run the W7100A / W7100 Debugger install file.



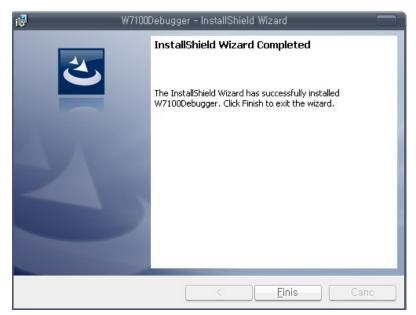
W7100Debugger,exe Setup Launcher Unicode WIZnet

[Fig.3.1] The W7100A / W7100 Debugger install file

Then the installation of iMCU7100EVB debugger is started. Push the Next button to setup.



[Fig.3.2] Setup the W7100A / W7100 Debugger



[Fig.3.3] End of debugger installation

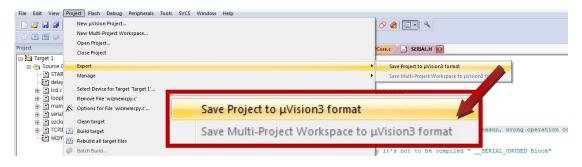


4 KEIL Project

Projects that are supported by the W7100A / W7100 debugger are as below.

- Keil uVision2 project
- Keil uVision3 project
- Only hex file (Restricted symbol function)

% The W7100A / W7100 Debugger don't support the KEIL $\mu Vision4$ or upper version project. But user can easily convert the $\mu Vision4$ to $\mu Vision3$ project as shown in the figure 4.1 to use the W7100A / W7100 debugger.



<Fig.4.1> Save Project to µVision3 format

* Caution on W7100A / W7100A Lock function

The W7100A / W7100A has the memory lock function. If the lock was set, user cannot use the W7100A / W7100 debugger because it cannot read anything from W7100A / W7100A memory. So user should unlock to use the W7100A / W7100 debugger properly. It is easy to unlock using the WizISP program. As shown in the figure 4.2, clear the check box and click the "Lock" button. Whole memory will be erased after the unlocking so user should re-write the firmware again.

Please refer to the Wiz ISP user manual for more detailed information about WizISP. WizISP program and user manual are available on the <u>WIZnet website</u>.



😵 WizISP (v2.0.0.0)	
<u>File Device Options H</u> elp	
Load HexToBin	Image: Second
COM Port Configuration	Hex Editor
Port: COM4 👻 🍋	Addrss(Hex) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0 1 2 3 4 5 6 7 8 9 A B C D E F 000000000 FF
Baud: 115200 👻 Close	00000010 FF FF FF 00000020 FF FF FF 00000020 FF FF FF 00000020 FF FF FF 00000020 FF FF FF
Flash Operation Mode Code Data Set	00000030 FF FF FF 00000040 FF FF FF ● Code ○ Data Set
🔽 Code 👽 Data 🛛 Lock	00000070 FF FF FF Code Data Lock
Auto Task Selection	00000090 FF FF FF
📝 1. Flash Erase 📝 2. Blank Check	000000A0 FF
📝 3. Program	00000000 FF F
Verify	000000F0 FF F

<Fig.4.2> Unlock the W7100A / W7100A using WizISP program

The W7100A / W7100 debugger cannot recognize multi directory structure of KEIL project so all project file and source file must be placed in same directory to use W7100A / W7100 debugger. And the 'make hex file' option and 'make symbol' option should be activated in the KEIL project. If they are not activated, the W7100A / W7100 debugger cannot find the hex file to load. Also, an error message could appear even if it finds the hex file.

Since the W7100A / W7100 debugger use slower clock than normal running clock, running speed under the debugging mode is very slow. So we recommend that don't use a delay function or something like that.

First, open the KEIL project. Click 'Project' => 'Options for Target 'Target name' menu.



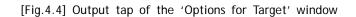
🕎 Test_Driver – Ziision3 📃	1×
<u>File Edit V</u> iew Project Debug Flash Peripherals Tools SVCS Window Help	
🖆 😅 🖬 🕼 🖉 🕰 😇 律 使 み % % % 🐂 📃 🖼 州	
😻 🕮 🥌 👗 🛱 🎢 Target 1 💽 🛃 🖷	
Project Workspace	
Image: Target 1 Image: Source Group 1 Image: Stratup,A51 Image: Stratup,c Image: Stratup,c <td></td>	
Build Command Find in Files	
Build Command Find in Files	

[Fig.4.3] Open the KEIL project

Check the 'Create HEX File' option from the 'Output' tap as figure 4.2.

Options for Target 'Target 1'
Device Target Output Listing User C51 A51 BL51 Locate BL51 Misc Debug Utilities
Select Folder for Objects Name of Executable: Test_Driver
Cr <u>e</u> ate Executable: .₩Test_Driver
Debug Information F Browse Information
Create HEX File HEX Format: HEX-80
◯ Create Library: .\#Test_Driver.LIB
OK Cancel Defaults Help





Select the 'Listing' tap and check the 'Symbols' option of the 'C Compiler Listing' region.

Options for Target Target 1'								
Device Target Output Listing User C51 A51 BL51 Locate BL51 Misc Debug Utilities								
Select Folder for Listings Page Width: 120 - Page Length: 65 -								
☑ Compiler Listing: .₩*.lst								
Conditional	Symbols	#include Files	Assem <u>bly</u> Code					
C Preprocessor Listing: .\#	*.i							
✓ Assembler Listing: .₩*.lst								
Conditional	Symbols Macro	s: Final expansion only	Cross Reference					
☑ Ljnker Listing: .\#Test_Driv	ver.m51							
Memory Map	Public Symbols	Line <u>N</u> umbers	Cross <u>R</u> eference					
	🔽 Local Symbols	Comment Records	Generated Symbols					
			✓ Library Symbols					
	OK Ca	ncel Defaults	Help					

[Fig.4.5] Listing tap of the 'Options for Target' window

Also check the 'Symbols' option of the 'Assembler Listing' region. The 'Assembler Listing' is optional. Click the 'OK' button to finish, and compile the KEIL project.

Confirm the HEX file in the project directory after compiling.



C:\LOOPBACK_TCP					
File Edit View Favorites Tools	Help				27
🌀 Back 🔹 🕥 🕤 🏂 🔎 Sea	arch 😥 Folders				
Address 🛅 C:\LOOPBACK_TCP					💌 🄁 Go
File and Folder Tasks 🔹	-			-	^
🗐 Rename this file	loopback	loopback	LOOPBACK	LOOPBACK	
Copy this file Publish this file to the Web E-mail this file Collete this file		V			
Other Places 🛞	LOOPBACK	LOOPBACK	LOOPBACK.hex	LOOPBACK.inp	
 Local Disk (C:) My Documents Shared Documents My Computer 					
Section 2015 My Network Places	loopback.LST	LOOPBACK.MAP	LOOPBACK.Opt	LOOPBACK_Opt.Bak	
Details					
					~

[Fig.4.6] Hex file of the KEIL project

If the KEIL project includes the assembly code such as xxx.SRC file, the 'Generate Assemble SRC File' and 'Assembler SRC File' options must be checked.

First, open the Options for File 'xxx.c' that uses the Assemble code.

Eile Edit View Broject Debug Flash Peripherals Tools SVCS Window Help	_ 8 X
12 12 12 12 12 12 12 12 12 12 12 12 12 1	
🖻 芭 西 参 Ă 🛱 孫 [Target 1 📃 🚽 👌 告 熙	
Project Workspace	[*
Remove File 'wizmemcpy.c'	
Include Dependencies	
42 EA = EAback: 43 DFXO = tap_DFXO; 44 + 45 # 46 + 47 + 48 + 49 + 44 + 45 # 46 + 47 + 48 + 49 + 49 + 40 + 41 + 41 + 42 + 43 + 44 + 45 + 45 + 46 + 47 + 48 + 49 + 49 + 44 + 45 + 46 + 47 + 48 + 49 + 49 + 41 + 41 + 42 + 43 + 44 + 45 + 46 + 47 + 48 +	• •

[Fig.4.7] Open the Options for File 'xxx.c'



Properties C51						
Path:	D:₩WIZnet₩Purp	le₩W7100₩	W7100_Applic	ation_Note_	.IntegratedVer₩Applications₩TC	P₩TCPs
File Type:	C Source file		•		🔽 Include in Target Build	
Size:	2019 Bytes		_		🔽 <u>A</u> lways Build	
last change:	Fri Apr 22 17:15:26 2	:011	_	- r	✓ Generate Assembler SRC I	File
					✓ Assemble S <u>R</u> C File	
					🔽 Link Publics Only	
Code <u>B</u> ank:	v					
Stop on Exit Code:	Not specified			-		
[
Select Modules						
to Always Include:						
Custom Arguments:						

Then, check the 'Generate Assembler SRC File' and 'Assemble SRC File' options.

[Fig.4.8] Setting the Options for File 'xxx.c'



5 Debugger menu

5.1 Open project

The W7100A / W7100 debugger can debug the KEIL project or the hex file. To debug the KEIL project or the hex file, run the W7100A / W7100 debugger and open the hex file.

Open Project.

The Open Project command is used to open a project.

Click the *icon* of the toolbar. It can also be done by clicking Project Menu => Open or pushing the short key Ctrl + O

The file open browser is opened as figure 5.2 'File open browser.'

👫 W7100 Debugg	er				
🖻 🗉 🔍 🖾 🗍 🍸 📆 🗐 🖉 🖉 🖉	🗉 📑 🚽				
WorkSpace # ×			Register		ņΧ
🕀 🚰 project : Empty			Register		-
			Register	Value	
			Default R		
			R0 R1	0×00 0×00	
			R2	0x00	
			R3	0x00	
			R4	0×00	
			R5	0x00	
			R6 87	0×00 0×00	
			A	0x00	
			B	0,00	-
Assembly 4 ×	Search				џ×
Assembly 💌	Search				-
💀 🎜					
	Name	Value	Type	Addr	res: 🔨
					_
					1.1
	<				>
	<	💽 G 🚑 S	ID 📻 E	. 📻 F 🙉	

[Fig.5.3] Open project



🤀 W7100 D	ebugger					
Project Help						
	-	1 🗸 🗉 🖉 🦌	1 📑 🔳 🗉	🗖 🚰 🗸		
 1	Open					? ×
WorkSpace	Look in:	COPBACK_	тср	•	⊨ 🗈 💣 •	
	My Recent Documents Desktop My Documents	DOOPBACK.he	x			
	My Network Places	File name: Files of type:	LOOPBACK HEX File (*.hex)		• •	Open Cancel

[Fig.5.2] File open browser

If the user opens the project that is consisted only with a HEX file, the Debugger will show a caution message. And it shows only 'Assembly window'.

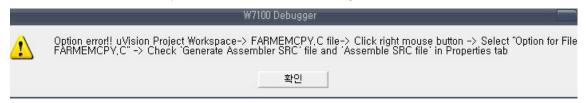
- 			W7100 Deb	ugger - STA	RTUP, A51						
Project Debug	<u>T</u> ool <u>W</u> indow	Environment	Help								
) 🚰 🛃 📮 🖗	🗖 T T 🔟	🗊 🎩 🖲 🕵 l	🧋 📰 📧 🗷	1 🖬 🗖 😭	•						
] 🗈, 🔟 📜 🌂	0 👂 🕨 🗖	9 (J 9 (J	*1.								
WorkSpace	ų ×	MAIN,C	FARMEMCF	Y,C 🖸 ST.	ARTUP, A	51 x			Register		џх
= 📅 project :		117 ?	STACK	3	EGMENT	IDA	አጥል		Register		-
😑 🗁 Source		118	OINON	5	DOMENT	107	11.0	- 10	ê A↓		
	RMEMCPY,C	119		R	SEG	?STAC	ск		Register	Value	
	ARTUP, A51	120		D	ន	1			🖻 Default I		
		121							RO	0x00	
		122					C_START)		B1	0x00	
		123		P	UBLIC	?c_sı	TARTUP		R2 R3	0x00 0x00	
		124 125			a n a		00000H		R4	0x00	
			C_STARTU		SEG JMP	AT STARI			R5	0×00	
		127		с. л	OPIE	DIAN	LOFT		R6	0x00	
		128		I R	SEG	2C C5	51STARTUP		B7	0x00	
		129	1. 1. 3	1 1 7		-			A	0x00	
			FARTUP1:								-
		131		ANL	OF	₽Н, #О)7FH	>			
Assembly			_	л×	Search	_					ąх
Assembly				•	Search						-
🖪 📮											
·	126 :	<pre>?C_STARTUP:</pre>	LJMP	STARTU 🛆	Name		Value	Ty	De	Address	^
¢ 0x0000	02001E		LJMP LOOO1								
0x000b	020126		LJMP LOOOO								
0x001b	02015E		LJMP LOOOO								
	131 :	ANL OFFH	#07FH								-
0x001e	53FF7F	L0001:	ANL OFFh #	7Fh	L						V
	134 :		MOV	RO,#ID∨	Se	💽 Lo	🧕 GI 🗛 S	y 🔲			Co
Ready			Fold	ing level: 0	Style: 0 Li	ne: 126, 0	Coll 1, Character	4592	CAR NU	MISCRE	1

[Fig.5.3] Complete opening the project



*Note: When the user open the project and have the error message like shown below, the user must delete the xxx.SRC file in project directory, and check both the 'Generate Assembler SRC File' and 'Assemble SRC File' options to avoid error on the W7100A / W7100 debugger.

For a more detailed solution, please refer to section4 'KEIL Project'.



[Fig.5.4] Open error message

Reload project

The Reload command is used to reload a project. When the loaded project is changed, click the reload toolbar button at the project. It can also be done by clicking 'Project' => 'Reload' or pushing the short key F4.

Close project

The Close project command is used to close a project. Select 'Project' => 'Close' or push the short key Ctrl + Q.

*Note: While using the W7100A / W7100 debugger, the user must use the 'stop' command in order to close the debugger.

5.2 Initialize the W7100A / W7100 Debugger

This section is for initialization of the debugger and iMCU7100EVB

1			W7100 Debugge	r - STARTUP, AS
<u>P</u> roject	Debug	Tool	Window Environment Help	
1	🕞 📮	Т	T 👖 🖉 🗉 📾 🎝 🖬 🗉 🗉	📑 🚰 🖕
a 6	1 🔶 🐘	0.5	▶ ■ ⁹ ∃ (] ⁹] , I •] ,	

[Fig.5.5] Initialize menu

Debugger and board reset

The Debugger and board reset command is used to reset the debugger and board simultaneously. If breakpoints were set, all initialized breakpoints will be removed.

To reset, click the solution on the toolbar, or select 'Tool menu' => 'Init Debugger and Board reset' menu. Then the Debugger and board will reset.



Board reset only

The Board reset command only is used to reset the board only. If breakpoints were set, they will be maintained. To reset the board, click the obstrained button of the toolbar or select the 'Debug menu' => 'Board reset'. It can also be done by short key F2.

Image load

The Image load command is used to load an image.

Write the HEX image for debugging to the Flash memory.

Click the solution or select Debug menu => Image Load menu to execute 'Image Load.' It can also be done by short key F3.

If the image writing and verifying is successful, 'Write OK' message will show.

Note: If there is a mismatch between the board image file and present HEX file, the error message is shown when execute the 'Debugger commands' such as 'Continue, Step, Next'.



[Fig.5.6] Load image mismatch

Debugging

The W7100A / W7100 Debugger serves 'Debug' and 'Tool' menu and Toolbar button for debugging.

<i>f</i>	W7100 Debugger - STARTUP, A51
Project Debug Tool	<u>Window Environment H</u> elp
🛛 😂 🔁 🔁 🖬	T 👖 🗸 🗉 🏟 🎝 🖬 🗉 🖬 🗉 🖀 🗸
] 🖦 🔟 🔚 🍋 🔾) SI (I SI (I *I ,

[Fig.5.7] Deugging menu

Go to Source Line

The Go to Source Line command is used to move the cursor to the wanted source line. To use this function, select the 'Tool' menu => 'Go to Source Line' or use short key Ctrl + G.

Find

The Find command is used to find some words on the source window. To use this function, select the 'Tool' menu => 'Find' or use short key Ctrl + F.



5.3 Run and Stop

This section will explain how to use commands like step in, out, next, and more

Continue

The Continue command runs the program until the Break point is met. To use this command, click the button on the toolbar or select the Debug menu => Continue. It can also be done by short key F5.

During the Continuing state, user cannot use the debugging menu. In order to use the debugging menu, execute the Stop command.

Stop

The Stop command is used to stop the running program. To use this command, click the button on the toolbar or select the Debug menu => Stop. It can also be done by short key F6.

The debugging commands (Search, memory view, break point, etc.) are executed only in the Stop state.

Step

The step command is used to run only one specific line among all lines. The Step command is executed in units of one line of the C code. If there is a Function call, the Step command will enter into the related Function. After the execution, the debugger is stopped and waits for the next command.

To use this command, click the button on the toolbar or select the Debug menu => Step. It can also be done by short key F11. If the user has only the HEX file to debug, the Step command executes one instruction, just like the Stepi command.

Next

The Next command is basically the same as the Step command. The Next command is executed in units of one line of the C code. But if there is a Function call, the Next command does not enter into the Function. After the execution, the debugger is stopped and waits for the next command.

To use this command, click the button on the toolbar or select the Debug menu => Next. It can also be done by short key F10. If the user has only the HEX file to debug, the Step command executes one instruction, just like the Nexti command.

Stepi

The Stepi command executes only one instruction. If the instruction is a Function call, the command enters into the related Function. After the execution, the debugger is stopped and waits for the next command.

To use this command click the button on the toolbar or select the Debug menu => Stepi. It can also be done by short key F7.



Nexti

The Nexti command executes one instruction. But the command does not enter into the related Function even if the instruction is a Function call. After its execution, the debugger is stopped and waits for next command.

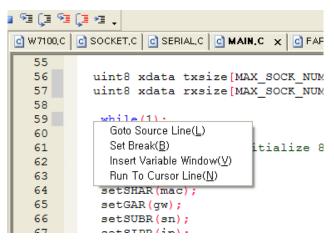
To use this command click the Justice button or select the Debug menu => Nexti.

It can also be done by short key F6.

Run to cursor line

The 'Run to cursor line' command executes the code until the cursor on the debugger source window is met.

To use this command, select the push the mouse right button => select 'Run To Cursor Line(n)' or select the Debug Menu => Run To Cursor Line.



[Fig.5.8] Run To Cursor Line command

Sometimes the 'Run To Cursor Line' command continues when the cursor is not properly set. If that is the case, execute the Stop command to stop it.



5.4 Break Point

This section will explain how to set/clear and find breakpoints. The breakpoint can be set only when the debugger is not being used. If the user was using the debugger, use the Stop command to stop in order to set the breakpoint.

🛛 🗣 📮 🗖 🕇	dow Environment Help	
	■ 91 [3 91 [3 91 .	
	X C W7100,C C SOCKET,C C SERIAL,C C MAIN,C X C FARMEMCPY,C	
월 project : test_drive Source Files Source Files SOCKET,C - 아이 SERIA_C - 아이 SE	<pre>49</pre>	R2 0x0e R3 0xff R4 0xf8 R5 0x00 R6 0x08 R7 0x2e A 0x78 B 0x08 SP 0x61 DPTR0 0x7700 PC\$ 0x00dc
		<u>></u>
embly	A X BreakPo	
embly	BreakPo	
0x0059 79C9	MOV R1 #0C9h	Label
0x005b 7E00		
0x005d 7F08	MOV R7 #8h), Line : 52 main + 19), Line : 53 main + 38
0x005f 12095D	LCALL L0016 MAIN.C	, Line : 54 main + 57 , Line : 56 main + 76
57 :	uint8 xdata rxsize[MAX_SOCK_NUM] = {2,2,2,2,2,2,2,2,2,2} MAIN.C	Line : 57 main + 95 Line : 57 main + 97
0x0062 7884	MAIN.C MOV R0 #84h	Line : 57 main + 99
0x0064 7C00	MOV R4 #0h	, Line : 57 main + 101 , Line : 57 main + 105 , Line : 57 main + 103
- STOODA 1000	MOV R5 #1h	, Line - 57 Main + 105
0,00066 7001	PIOV R.3 #111	
0x0066 7D01	MOL D2 AOFEN	
0x0068 7BFF	MOV R3 #0FFh	
0x0068 7BFF 0x006a 7A0E	MOV R2 #0Eh	
0x0068 7BFF 0x006a 7A0E 0x006c 79D1	MOV R2 #0Eh MOV R1 #0D1h	
0x0068 7BFF 0x006a 7A0E 0x006c 79D1 0x006e 7E00	MOV R2 #0Eh MOV R1 #0D1h MOV R6 #0h	
0x0068 7BFF 0x006a 7A0E 0x006c 79D1	MOV R2 #0Eh MOV R1 #0D1h	

[Fig.5.9] Break point of the debugger

Set break point

To set the breakpoint

- 1. Click the shadowed region side of the source window
 - ck the 🛛 🐌 utton on the Toolbar
- 2. Locate the cursor and click the Utton c
- 3. Locate the cursor and click the Debug Menu => 'Set Break'
- 4. Double click the C code line in the source window
- 5. Double click the Assembly code line in the Assembly window
- 6. Locate the cursor in the Assembly window and click the
- 7. Short key F9

Set the breakpoint and execute the Continue command. Then the debugger will run until it meets the breakpoint and will stop at the checked breakpoint. After that, the user can use



loolbar



other Debug commands. If the PC(program counter) can't move to the breakpoint, the debugger will keep running. If that is the case, the user should execute the Stop command to stop it.

Clear Break

The Clear command is used to clear the breakpoint.

- 1. Click the checked breakpoint 🦲 region
- Click the O button on the selected breakpoint line
- 3. Locate the cursor at the C code line and click the Debug Menu => Clear Break
- 4. Double click the breakpoint directly on the C code line
- 5. Double click the breakpoint directly on the Assembly code line
- 6. Select the breakpoint on the Assembly code line and click the 📮 button on the Assembly Toolbar
- 7. Select the breakpoint on the C code line and select the 🗶 button on the Breakpoint Toolbar
- 8. Short key F9

Clear All Breaks

The clear all breaks command will clear all breakpoints, both from the C code and Assembly code. To clear all breakpoints,

- Click the 💭 button on the Breakpoint Toolbar (Breakpoint window)
- Click the 🧔 button on the Toolbar
- Select the 'Debug' menu => 'Clear All Breaks'

Goto Next Break Point

The Goto Next Break Point Command is used to move the cursor to the next break point.

To use this function,

Click the 📴 button on the Toolbar

Select the 'Tool' menu => 'Goto Next Break Point'

Goto Previous Break Point

The Goto Previous Break Point command will move the cursor to the previous break point. To use this function,

Click the 💭 button on the Toolbar

Select the 'Tool' menu => 'Goto Pre Break Point'



Break Point Window

The Break Point Window command shows all checked breakpoints.

The break point Toolbar buttons are as below.

- 🗶 : Clear the selected breakpoint
- Section 2 Clear all breakpoints
- 🛃 : Move to selected breakpoint

Source and Assembly window

In the Source window and Assembly window, user can see the source code and assembly code. The user can also use Debug commands.

In the source window, user can check all variables and add it to the search window.



[Fig.5.10] Window menu

To use the Assembly window, click the button on the Toolbar or select the Window Menu => Assembly menu. The Source window is always shown unless opening only HEX file.

Source Window

🖸 W7100,C	C SOCKET,C C SERIAL,C C MAIN.C × C FARMEMCPY,C C STARTUP,A51
48	// uint8 sn_sr_tmp, sn_mr_tmp;
49	<pre>// uint8 xdata tmp[6] = {0,0,0,0,0,0};</pre>
50	
51	uint8 xdata ip[4] = {192,168,1,86}; // fo:
52 📒	uint8 xdata gw[4] = {192,168,1,1}; // fc
53 🧧	uint8 xdata sn[4] = {255,255,255,0}; //
54 🛑	uint8 xdata mac[6] = {0x00,0x08,0xDC,0x11,0x22,0x86};
55	
56 🛑	uint8 xdata txsize[MAX_SOCK_NUM] = {2,2,2,2,2,2,2,2};
57 🗘	uint8 xdata rxsize[MAX_SOCK_NUM] = {2,2,2,2,2,2,2,2};
58	Goto Source Line(L)
59	
60	Init805: Set Break(B) 8051
61	iinchip_ Insert Variable Window(⊻)
62	Run To Cursor Line(<u>N</u>)
63	setSHAR(mac);
64	setGAR(gw);
65	<pre>setSUBR(sn);</pre>
66 📒	<pre>setSIPR(ip);</pre>
67	sysinit(txsize,rxsize);
68	
69 📒	PutStringLn("====================================
70	PutStringLn(" W7100 Net Config Information");

[Fig.5.11] Source Window menu

The Source window is shown in the figure 5.11.



- t> : indicates the PC (Program Counter) in the source code
- Image: means Breakpoint is set on this line
- : the breakpoint can only be set on this shadowed line

Click the mouse right button in the Source window, user can see the Source window menu as in the figure 5.11.

- Goto source line: Move the cursor to the set line
- Set break: Set the breakpoint
- Insert Variable window: Add the variable to the Search window
- Run To Cursor Line: Run the C code to cursor line

Moreover, the Source window can assist some functions as below.

- Can set the breakpoint by double clicking the C code.
- Show the indicated variable value when the mouse pointer on the variable.

Assembly window

The assembly window shows the currently running assembly code. Also if the user clicks the C code at the source window, it shows the related assembly codes. If the user opens only the HEX file without Keil project files, only the assembly window will be shown without the source window.

Assembly			×
Assembly			•
🖪 📮			
0x0044	7A0E	MOV R2 #0Eh	^
0x0046	79C3	MOV R1 #0C3h	
0x0048	7E00	MOV R6 #0h	
0x004a	7F06	MOV R7 #6h	
0x004c	12095D	LCALL L0016	
	56 : uint8 xdata	<pre>a txsize[MAX_SOCK_NUM] = {2,2,2,2,2,2,2,2,2};</pre>	
🗘 0x004f	787C	MOV R0 #7Ch	
0 x0051	7C00	MOV R4 #0h	
0x0053	7D01	MOV R5 #1h	
0x0055	7BFF	MOV R3 #0FFh	
0x0057	7A0E	MOV R2 #0Eh	
0x0059	79C9	MOV R1 #0C9h	
0x005b	7E00	MOV R6 #0h	
0x005d	7F08	MOV R7 #8h	
0x005f	12095D	LCALL L0016	

[Fig.5.12] Assembly window

In figure5.12,

- Indicates the PC's location of the instruction
- 🛑 : Indicates the break point
- 📊 : Shows the assemble code with related C code
- Shows the break point
- double click: Sets or unsets the break point



6 Search & Variable Window

The W7100A / W7100 debugger provides various windows to see the local variable, global variable, and the register value.

- Global variable window: Shows the value of global variable in the project
- Local variable window: Shows the value of local variable at the PC(program counter) now
- Search window: Puts the variable name, then it shows the value at the PC now
- Register window: Shows the value of registers at the PC now
- Symbol window: Shows the all symbols and its values of the project

To activate each window, select the related window on the Window menu or click the window button on the Toolbar.

*					W710	0 Debuggei	r - STARTUP, A	151
Project	Debug]	lool	<u>W</u> indow	Environment	Help		1993	
) 🚰 🛃	📮 📮	Т	T 🔟 🍃	j 🗉 🖲 🖗	🎜 🔳 (E 🚰 🗸	
	1 🐮 🐌 🤇	⊃ 👳		91 (I 9 1 ()	. ⊵•			



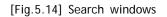
- Activate Search window: Click the Menu
- Activate Local variable window: Click the III or select the 'Local Variable Window' on the Window Menu
- Activate Symbol window: Click the Activate Symbol window: Click the Activate Symbol Window: On the Window Menu
- Activate Register window: Click the I or select the 'Register Window' on the Window Menu

6.1 Search Window

The search window shows the global variable, local variable, or the memory value, depending on which ones the user registered. The global/local variable windows also show the values of variables. But usually users want to see one or two variable values and sometimes want to see both global and local variables simultaneously. In this case, just register the variables on the search window.



Search				>
Search				-
Name	Value	Туре	Address	· · · · · · · · · · · · · · · · · · ·
D:0x00:4 C:0x00:4 X:0x6a:4	249, 42, 14, 255 2, 18, 106, 120 192, 168, 1, 86	ARBAY ARBAY ARBAY		
s len	4	U_CHAR INT	D:0x000008 D:0x000009	
				1



To register the global/local variables, write down the register name on the name field of the search window.

On the search window, user can not only confirm the variable but also can modify its value. To modify the value of the variable, click the value field on the search window and write down the value.

To see the memory value on the search window, write in the form, 'memory type : address : size.' The memory types are C, D, and X. (code memory, internal data memory and external data memory).

Ex) Code memory => C:0x1000:5 Internal memory => D:0x30:2 External memory => E:0x000300:4

Local Variable	9			_				‡×
Search								•
Name	Value	Туре		Address				^
S	0	U_CHAR		D:0x000008				
len i j gw sn mac txsize rxsize	0 0 192,168,1,86 192,168,1,1 255,255,255,0 0,8,220,17,34, 2,2,2,2,2,2,2,2 2,2,2,2,2,2,2,2	IN U_CHAR ARRAY ARRAY ARRAY ARRAY ARRAY ARRAY ARRAY ARRAY	OC.	ARY (B) TAVO (O) CIMAL (D) < (X) A.0x000070 X:0x00007C X:0x000084				
								×
🔲 IData Mem	o 💽 Global Vari	🗛 Symbol	Co 🔝	de Memo 📧	External	M 💽 Lo	ocal Varia,	🗿 Search

[Fig.6.1] Display format

User can select the display format by clicking the mouse right button on the search, local and global window such as binary, octal, decimal and hexadecimal.



6.2 Local Variable Window

The local variable window shows the value of the local variable in the PC located function.

Local Varia	able			×
Search				•
Name	Value	Туре	Address	~
s	4	U_CHAR	D:0x000008	
len	0	INT	D:0x000009	
i	0	ULCHAR	D:0x00000B	
ip	192,168,1,86	ABBAY	X:0x00006A	
gw	192,168,1,1	ABBAY	X:0x00006E	
sn	255,255,255,0	ARRAY	X:0x000072	
mac	0,8,220,17,34,	ABBAY	X:0x000076	
txsize	2,2,2,2,2,2,2,2,2	ABBAY	X:0x00007C	
rxsize	2,2,2,2,2,2,2,2,2	ARRAY	X:0x000084	
				×

[Fig.6.2] Local variable window

The local variable window has almost the same interface as the search window. But it is not allowed to register a new variable or a memory address.

6.3 Global Variable Window

The global variable window shows all the global variables in the project. The global variable window has the same interface as the local variable window.

Global Variable	•			×
Global Variable				•
			1	
Name	Value	Туре	Address	
data_buf local_port read_addr_h read_addr_l	28672 33823 54 132	PTR U_INT U_CHAR U_CHAR	D:0x00005F X:0x000099 X:0x000068 X:0x000069	

[Fig.6.3]	Global	variable	window
[1 19.0.0]	olobul	variable	window

6.4 Symbol window

The symbol window shows the information of all symbols in the project. It shows the class type and the address of the related symbol.



Symbol						
Symbol						
X 📯 🗃						
Name	Class Type	Address		 		
- 🔄 ?C_STARTUP						
🗍 👰 ?C_STARTUP	PUBLIC	C:0x000000				
- 🗿 acc	SFRSYM	D:0x0000E0				
- 🧿 В	SFRSYM	D:0x0000F0				
- 👰 DPH	SFRSYM	D:0x000083				
-@ DPL	SFRSYM	D:0x000082				
- 🗿 IDATALOOP	SYMBOL	C:0x00126D				
- 🗿 PPAGE_SFR	SFRSYM	D:0x0000A0				
- 🗿 SP	SFRSYM	D:0x000081				
- 🧿 STARTUP1	SYMBOL	C:0x00126A				
🕂 🧿 FARMEMCPY						
- 🤕 MAIN						
- 🙆 _ICE_DUMMY_	SYMBOL	C:0x000000				
- 🧿 P0	SFRSYM	D:0x000080				
- 🧿 P1	SFRSYM	D:0x000090				
- 🧿 P2	SFRSYM	D:0x0000A0				
- 🤕 P3	SFRSYM	D:0x0000B0				
-@ IE	SFRSYM	D:0x0000A8				
-@ IP	SFRSYM	D:0x0000B8				
🖓 main	PIIRT TC	C+04000003	_			

[Fig.6.4] Symbol window

6.5 Register window

In the register window, user can check or modify the value of the register. It shows the default registers, PSW, and special function register (SFR). To modify the value of the register, click the field of register value and write the new value. But if the register is read only, then the value is not changed.

		•
8≣ 2↓		
Register	Value	
Default Registers		
RO	0×f9	
R1	0x2a	
R2	0x0e	
R3	0×ff	
B4	0×14	
R5	0×04	
R6	0×44	
87	0×14	
A	0×f8	
В	0×07	
SP	0×61	
DPTRO	0×0344	
PC \$	0×0211	
□ PS₩		
P	1	
F1	0	
ov	Ō	
RS	0	
FO	0	
AC	0	
CY	Ō	
Speicial Registers		
PO	Oxff	
DPL	0x03	
DPH	0×44	
DPTR1	0x0000	
DPS	0×00	
PCON	0x80	
TCON	0xc0	
TMOD	0x20	
TLO	0x00	
TL1	Oxfe	
THO	0×00	



7 Memory window

Notice: To read a memory value of iMCU7100EVB, a Keil project or HEX file should be opened first. To read a memory value of iMCU7100EVB, any Keil project or HEX file should be opened first. But there are no differences between Keil project or HEX file and image file which loaded to iMCU7100EVB.

The iMCU7100API debugger supports below memory windows.

- IData memory window: Shows the value of internal data memory. Update the value when stop state after the debugger command
- External memory window: Shows the value of external data memory
- Code memory window: Shows the value of code memory
- Flash memory window: Shows the value of flash memory

To activate each window, use the 'Window' menu and Toolbar as in the figure 7.1.

R.	W7100 Debugger - STARTUP, A5
Project Debug Tool	Window Environment Help
🛛 😂 📮 🔽 T	T T 🖉 🗸 🔍 🎝 🐨 🗸 🖉 🖬 🖬 🖬 🖬 🖓 🗸
) I ~I (I ~I (I ~I (

- Activate IData memory window: Click the is button or select the 'Window' menu => 'IData Memory Window'
- Activate External memory window: Click the solution or select the 'Window' menu => 'External Memory Window'
- Activate Code memory window: Click the solution or select the 'Window' menu => 'Code Memory Window'
- Activate Flash memory window: Click the fill button or select the 'Window' menu => 'Flash Memory Window'

[[]Fig.7.1] Memory window



7.1 IData Memory Window

The IData memory window can show the value of internal data memory (D:0x00 ~ D:0xFF) or modify the value.

IData Memory		_		_	×
Start Address: []	×000000	End Address:	0x0000FF		
248					
Address	0123	4567	89AB	CDEF	<u>^</u>
0x000000 :	F92A0EFF	14044414	04000000	0701138F	
0x000010 :	00070008	40004000	00000000	00000000	
0x000020 :	00000000	00000000	00000000	00000000	
0x000030 :	00000000	00000000	00000000	00000000	
0x000040 :	00000000	00000000	00000000	00000000	=
0x000050 :	00000000	00000000	00000000	00000070	
0x000060 :	00000202	C4193516	00000000	00000000	
0x000070 :	00000000	00000000	00000000	00000000	
0x000080 :	FF610344	00000080	C02000FE	00FC0100	
0x000090 :	FF000600	00000000	50000000	00000000	
0x0000a0 :	FF000000	00000000	00000000	00000000	
0х0000Ъ0 :	FF000000	00000000	00000000	00000000	×

[Fig.7.2] IData Memory window

To modify the value of internal data memory, click the value field of internal data memory and write the new value. All memory windows can change into the display format, such as 2/4/8 byte display unit or CHAR/INT/LONG display format. It is shown in figure7.3.

IData Memory					άx
Start Address:	0×000000	End Addre	ss: Ox0000FF		
248		_			
Address	0123	4567	89AB	CDEF	<u>^</u>
0x000000 :	62000000	00000100	00000000	00000000	
0x000010 :	00000000	00000000	00000000	00000000	
0x000020 :	00000000	00000000	00000000	00000000	
0x000030 :	00000000	CHAR(C	, ^^ pooooo _	00000000	
0x000040 :	00000000		000000	00000000	
0x000050 :	00000000	o INT()	000000	00000000	
0x000060 :	70000000	이 LONG(L)) 000000	00000000	
0x000070 :	00000000	0		00000000	
0x000080 :	FF62FB1A	00000000	00000000	00000700	
0x000090 :	FF000700	00000000	00000000	00000000	
0x0000a0 :	FF000000	00000000	00000000	00000000	~
Lo					
🔲 IData Memo	퉪 Global Vari,,	. 🙀 Symbol	💽 Code Memo,	🔳 External N	4 💽 Local Varia 👰 Search

[Fig.7.3] Change the display format

If the internal data memory value updates and the debugger command is in stop state, it updates the internal data memory value to the iMCU7100API board.

7.2 External Data Memory Window

The external data memory window shows the value of external data memory (X:0x000000 ~ X:0xFFFFF).



External Memo	ry				×
Start Address:	0×000000	End Addres:	s: 0x000100		
248					
Address	0123	4567	89AB	CDEF	<u> </u>
0x000000 :	07FF07FF	07FF07FF	07FF07FF	07FF07FF	
0x000010 :	07FF07FF	07FF07FF	07FF07FF	07FF07FF	
0x000020 :	08000800	08000800	08000800	08000800	
0x000030 :	08000800	08000800	08000800	08000800	
0x000040 :	5F03217F	3C2665E5	C000C800	D000D800	
0x000050 :	E000E800	F000F800	80008800	90009800	
0x000060 :	A000A800	B000B800	3684C0A8	0156C0A8	
0x000070 :	0101FFFF	FF000008	DC112286	02020202	
0x000080 :	02020202	02020202	02020202	01423DAC	
0x000090 :	20008320	16270C5B	C4841FFC	714B82B4	
0x0000a0 :	9600F98F	4E076809	4D28A880	2124BF55	
	[[:~ 7	11 External	Data momory		

[Fig.7.4] External Data memory window

The value of external data memory is not automatically updated. If the value of external data memory updates on each debugging command, the speed of the debugger may be slower due to the size of external data memory being larger than the internal data memory. Therefore, the debugger has a button **(2)** for updating the value of external data memory.

As in section7.1 'IData Memory Window,' if the user changes the value of external data memory in the debugger, the real value of iMCU7100API board will also change.

7.3 Code Memory Window

The code memory window shows the value of code memory ($0xFF0000 \sim 0xFFFFF$). This window is for reading only, it cannot be modified.

Code Memory Start Address: []	×000000	End Address:	0x1b0d		×
2432			[
Address	0123	4567	89AB	CDEF	▲
0x000000 :	02126A78	6A7C007D	017BFF7A	0E79B77E	
0x000010 :	007F0412	095D786E	7C007D01	7BFF7A0E	
0x000020 :	79BB7E00	7F041209	5D78727C	007D017B	
0x000030 :	FF7A0E79	BF7E007F	0412095D	78767C00	
0x000040 :	7D017BFF	7A0E79C3	7E007F06	12095D78	
0x000050 :	7C7C007D	017BFF7A	0E79C97E	007F0812	
0x000060 :	095D7884	7C007D01	7BFF7A0E	79D17E00	
0x000070 :	7F081209	5D121A09	121AF07B	017A0079	
0x000080 :	76121932	7B017A00	796E1219	E37B017A	
0x000090 :	00797212	19F67B01	7A00796A	1219CF75	
0x0000a0 :	0F017510	00751184	7B017A00	797C1203	

[Fig.7.5] Code Memory window

As in the section 7.2 External Data Memory Window, the code memory window has a button for updating the value of code memory.

7.4 Flash Memory Window

7.4.1 Code Memory Domain

In the flash memory window, if the 'Data memory' option is unchecked, it will shows the



value of code memory (0xFF0000 ~ 0xFFFFF). Basically it is same the 'Code Memory Window' but it can modify the value with flash write button.

Start Address:	0×FF0000	End Addre	ss: OxFFFFFF	📃 🗖 Data mer	nory
2482	1				
Address	0123	4567	89AB	CDEF	^
0xff0000 :	80008000	02112800	08FC9C1A	20FEFB18	
0xff0010 :	E0861AEC	DCDC1D3A	0BF94160	BBDD0600	
0xff0020 :	DF008C80	FAFE94E2	FAE7F8FF	8AFFA7D8	
0xff0030 :	B603FF7A	FF9FDACA	FF8D88F9	8E3DB09A	
0xff0040 :	90E0EE62	91AF9EB7	72FDFFB5	8F90FD90	
0xff0050 :	B8B7FF40	F98CC082	96C89CBC	5FA9EB30	
Oxff0060 :	FFDF5FFF	B6C4B800	66F6B0D7	E6BDDD99	
0xff0070 :	FEB77298	00378124	D0BB9028	868D9A80	
0xff0080 :	D6FF848C	30E6AE88	99FE5EEE	ECFFCAF9	
0xff0090 :	8BBF7FD4	B7FBFFD5	54FD73E0	FFBBE1F7	
Oxff00a0 :	927EFF66	3F76FFE7	F9BDFB04	168A83BF	
Oxff00b0 :	1FF313BD	B1EEBB08	9DF5C8A4	FD000A00	
Oxff00c0 :	C6C4BA7B	B65A93A4	7DF7D782	96EE3E58	
Oxff00d0 :	9EFDD7FF	2EDDAFDF	91F700C6	70270000	
Oxff00e0 :	8293A012	A8FD0010	60B9409F	AEF430C2	~
🖉 Search 🛛 👪 I	Local V 💽 Glo	obal 🗛 Syn	nbol 🔲 IData I	M 📧 Externa	Flash M 💽 Code M

[Fig.7.6] Flash memory window for code memory

The basic interface of the flash memory window is the same as other memory windows. But it has the flash write button shift which writes the whole value of flash memory window to the flash memory of iMCU7100API. Also it has a button for updating the value of flash memory.

7.4.2	2 Dat	ta Me	mory Do	\langle	Checked!!	
					6	×
	Start Addres:	s: OxFF000	End Add	ress: OxFFOOF	🔽 Da	ata memory
	2 4 8	2 📜				
	Address	012	3 4567	89AB	CDEF	
		: 80008	000 02112800		20FEFB18	
	0xff0010	: E0861			BBDD0600	
	0xff0020	: DF008			8AFFA7D8	
	0xff0030	: B603E			8E3DB09A	
	0xff0040	: 90E0E			8F90FD90	
	0xff0050	: B8B7F			5FA9EB30	
	0xff0060	: FFDF5			E6BDDD99	
	0xff0070	: FEB77			868D9A80	
	0xff0080	: D6FF8			ECFFCAF9	
	0xff0090	: 8BBF7			FFBBE1F7	
	0xff00a0	: 927EF			168A83BF	
	0xff00b0	: 1FF39			FD000A00	
	0xff00c0	: C6C4E 9EFDI			96EE3E58 70270000	
	0xff00d0					
	0xff00e0 0xff00f0	: 8293A : 00C77			AEF430C2 00000000	
	UXIIOUIU	: 00077	D04 081B2908	42C2B800	00000000	
	👼 Search 🔳	🕻 Local V	💽 Global 🛛 🚑 S	ymbol 🔝 IData	M Extern	a 📊 Flash M 💽 Code M

[Fig. 7.6] Flash memory window for data memory

If the 'Data memory' option is checked, it will shows the value of flash data memory (0x00 ~ 0xFF). It can modify the value with flash write button. The basic interface of the flash memory window is the same as other memory windows. But it has the flash write button which writes the whole value of flash memory window to the flash memory of iMCU7100API. Also it has a button for updating the value of flash memory.

Document History Information

Version	Date	Descriptions
Ver. 0.9 Beta	2009. 8. 13	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 lunching
Ver. 1.1	Jun, 2012	Fixed some awkward expressions of English documents

Copyright Notice

Copyright 2012 WIZnet, Inc. All Rights Reserved.

Technical Support: <u>support@wiznet.co.kr</u> Sales & Distribution: <u>sales@wiznet.co.kr</u>

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