IAR Embedded Workbench 소개

EW Compiler Series는 전세계적으로 임베디드 시스템 개발자들이 가장 많이 사용하는 C/C++ Compiler 및 Debugging Tool 입니다. 8/ 16/ 32 bit Microprocessor 와 DSP 등 30가지 이상의 Compiler 시리즈를 지원하며, 세계적인 기업인 컴파일러 개발 전문 회사 IAR 사 (www.iar.com) 제품으로써, 안정된 Tool 과 최적의 개발자 환경은, 이미 미국/유럽/아시아 각지의 Microcontroller Vender (ATMEL , Hitach, TI, SAMSUNG 등)들을 통해서 인정을 받고 있습니다.
국내기업 SAMSUNG 에서 의뢰되어 만들어진 SAM8 compiler (SAM8 : Samsung 8 bit Microprocessor) 또한, IAR 사의 EW Series 중에 하나이며, 현재까지도 계속적인 Upgrade 와 지원으로 삼성을 포함한 국내외 많은 사용자들에 의해서 꾸준히 사용되고 있습니다. (제품명, EWSAM8 Compiler)

EWARM compiler 는 ARM7/9/9E/10/11 시리즈 계열을 모두 지원하며, USB 방식의 H/W Debugger Tool (J-Link) 와 EWAVR Compiler S/W (C-Spy Debugger) 를 이용하여 쉽고, 빠른 Debugging 환경을 제공하고 있습니다.
뿐만 아니라, STR71x 를 위한 Evaluation boards 를 위한 별도의 Example Project Source 와 IAR Compiler 를 활용하여 기본적인 주변 Peripheral 간단하게 제어 할 수 있는 예제 소스를 제공하여, 사용자 편의를 도모한다.

- IAR Compiler Series 와 지원 Vander

EWAVR, EWARM, EWMSP430, EWSAM8, EW8051, EWx96, EWH8300, EWH8, EWSuperH, EWCOP8, EWCR16, EWCR16C, EWZ80, EWeZ80, EW80251, EWPIC16/17, EWPIC18, (New) EWdsPIC, EW7700, EW68HC11, EWM16C, EWM32C, EW68HC11, EW68HC12, EW68HC16, (NEW) EWHCS12, EW78K, EW78K4, EWV850, EWMK5, EWTLCS-900, EWR8C, EW6502, EWMAXQ 등 . (2005. 4월 현재)

Analog Devices, ARM architecture, Atmel, Cirrus Logic, Cypress, Freescale Semiconductor, Infineon, Maxim/Dallas, Microchip , NEC, OKI Semiconductor, Philips Semiconductors, Renesas Technology, Samsung, Sharp, Silicon Laboratories, STMicroelectronics, Texas Instruments, Toshiba, WDC, ZiLOG 등.. (2005. 4월 현재)

C/C++ compiler and debugger tools for ARM 4.30A

- Embedded focus and debugging
- Hardware debugging support
- IAR J-Link for ARM OPTIONAL
- RTOS support
- Graphical integrated development environment
- Language and standards
- User assistance

Highlights in the current version

- ARM11 support now included
- Improved speed optimizations and rewritten floating-point library
- OSEK Run-Time Interface (ORTI) support
- OSE Epsilon RTOS plugin included
- Multiple flash loaders
- Generic flash loader API guide
- Ready-made C/C++ and assembler peripheral register definition files for chips from Analog Devices ARM, Atmel, Cirrus Logic, Freescale Semiconductor, Intel, NetSilicon, OKI, Philips, Samsung, Sharp, STMicroelectronics and Texas Instruments
- Flash loaders for devices from Analog Devices, Atmel, Freescale, OKI, Philips, STMicroelectronics and Texas Instruments
- Sample projects for evaluation boards from IAR Systems, Analog Devices, ARM, Atmel, Cirrus Logic, Freescale, Keil, OKI, Olimex, Pasat, Philips, Phytec, STMicroelectronics and Texas Instruments

Supported ARM cores and devices

ARM7 (ARM7TDMI, ARM7TDMI-S and ARM720T), ARM9 (ARM9TDMI, ARM920T, ARM922T and ARM940T), ARM9E (ARM926EJ-S, ARM946E-S and ARM966E-S), ARM10 (ARM1020E and ARM1022E), ARM11 and Intel XScale.

- Analog Devices : ADuC7020, ADuC7021, ADuC7022, ADuC7024, ADuC7025, ADuC7026, ADuC7027
- Atmel AT91 : AT91FR4042, AT91FR4016, AT91M40400, AT91M40800, AT91R40807, AT91M40807,AT91M42800, AT91M55800, AT91M63200, AT91M40400, AT91RM3400, AT91RM9200, AT91SAM7S32, AT91SAM7S64, AT91SAM7S128, AT91SAM7S256, AT91SAM7A3.
- Cirrus Logic : EP7312, EP9301, EP9302, EP9307, EP9312, EP9315.
- Freescale Semiconductor : Dragonball MC9328MX1, MAC7100, MAC7101, MAC7106, MAC7111, MAC7112, MAC7121, MAC7122, MAC7126, MAC7131, MAC7136, MAC7141, MAC7142, MAC7202, MAC7212, MAC7222, MAC7242, MAC7252.
- Intel : XScale PXA255.
- NetSilicon : NS9360.
- OKI : ML670100, ML671000, ML672300, ML674000, ML674001, ML675001, ML67Q4002, ML67Q4003, ML67Q4050, ML67Q4051, ML67Q4060, ML67Q4061, ML67Q5002, ML67Q5003.
- Philips : LPC2114, LPC2119, LPC2124, LPC2129, LPC2130, LPC2131, LPC2132, LPC2134, LPC2136, LPC2138, LPC2142, LPC2148, LPC2194, LPC2212, LPC2214, LPC2292, LPC2294.
- Samsung : S3C2410, S3C2440A, S3C2440X, S3C44A0A, S3C44B0X, S3C4510B, S3C4610, S3C4640, S3C4650, S3C380D, S3F380D, S3F445HX, SC3400, KS32C6400.
- Sharp : LH7A400, LH7A404, LH75400, LH75401, LH75410, LH75411.
- STMicroelectronics : STR71x, STR715, STR72x, STR730.
- Texas Instruments : TMS470R1A128, TMS470R1A1M, TMS470R1A256, TMS470R1A384

EWARM Demo : 1. Evaluation Version (Time limited for 30 Days)
2. KickStart Version (Code limited for 32KByte)

* 데모 버전을 사용하기 위해서는 반드시 등록과정을 거쳐야 하며, 등록과 동시에 등록된 고객의 e-mail 주소를 통하여 발급된 License Key 와 License Number 가 있어야지만, Install 이 가능하므로 반드시 기재해야만 합니다.
(다운로드 시간 : 약 2~3 분)

EWARM package 종류

1. EWARM-PRO : EWARM C Compiler/Assembler + C-SPY + visualSTATE + SUA
2. EWARM-Full : EWARM C Compiler/Assembler + C-SPY + SUA
3. EWARM-LE : EWARM C Compiler/Assembler + SUA
4. EWARM-BL : EWARM C Compiler/Assembler (256KB Code Limited)
5. EWARM DEMO : EWARM C Compiler/Assembler (32KB or 30Days) + C-SPY

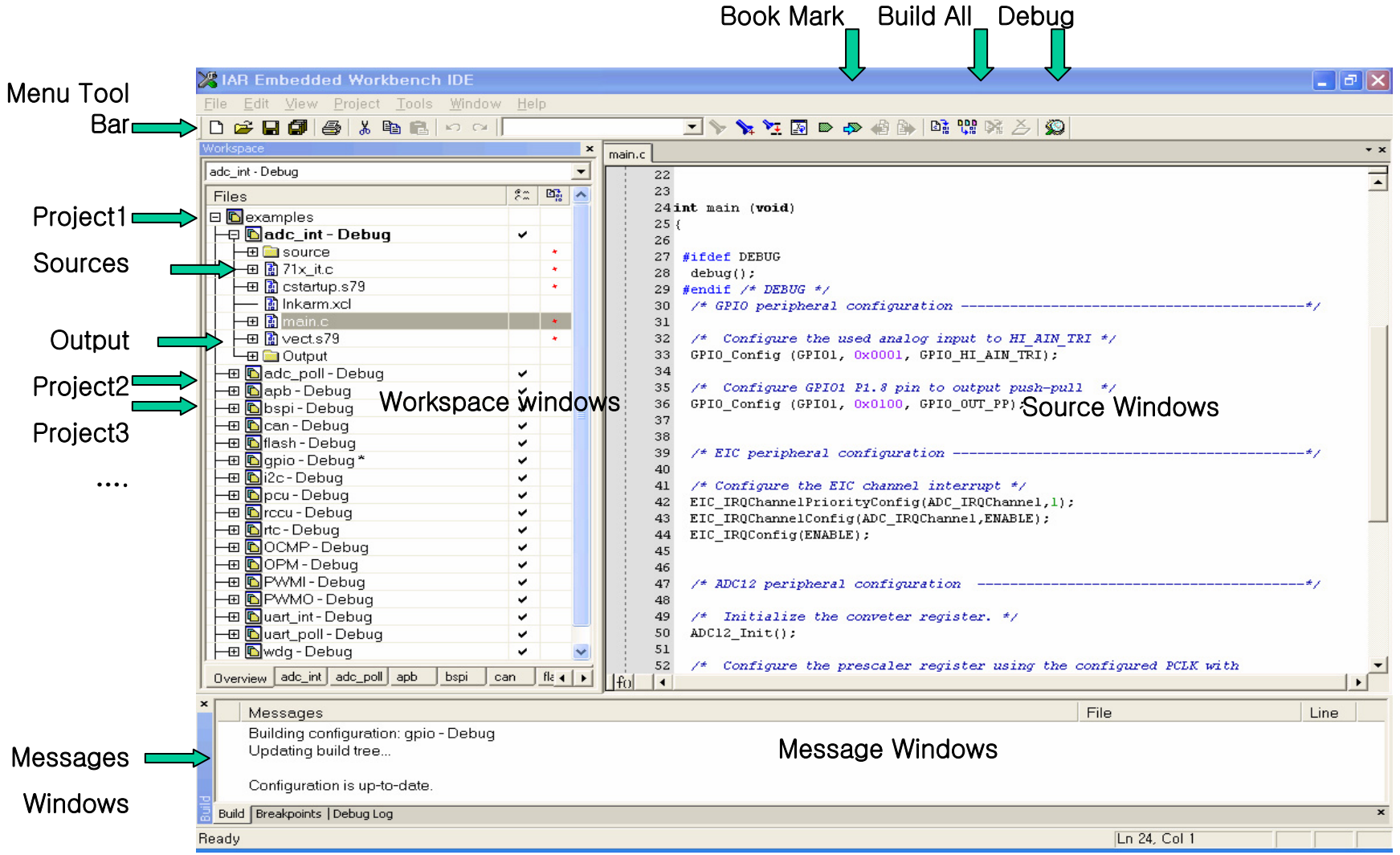
* LE : Limited Edition * SUA : Support and Upgrade Agreement

- EWARM Download -

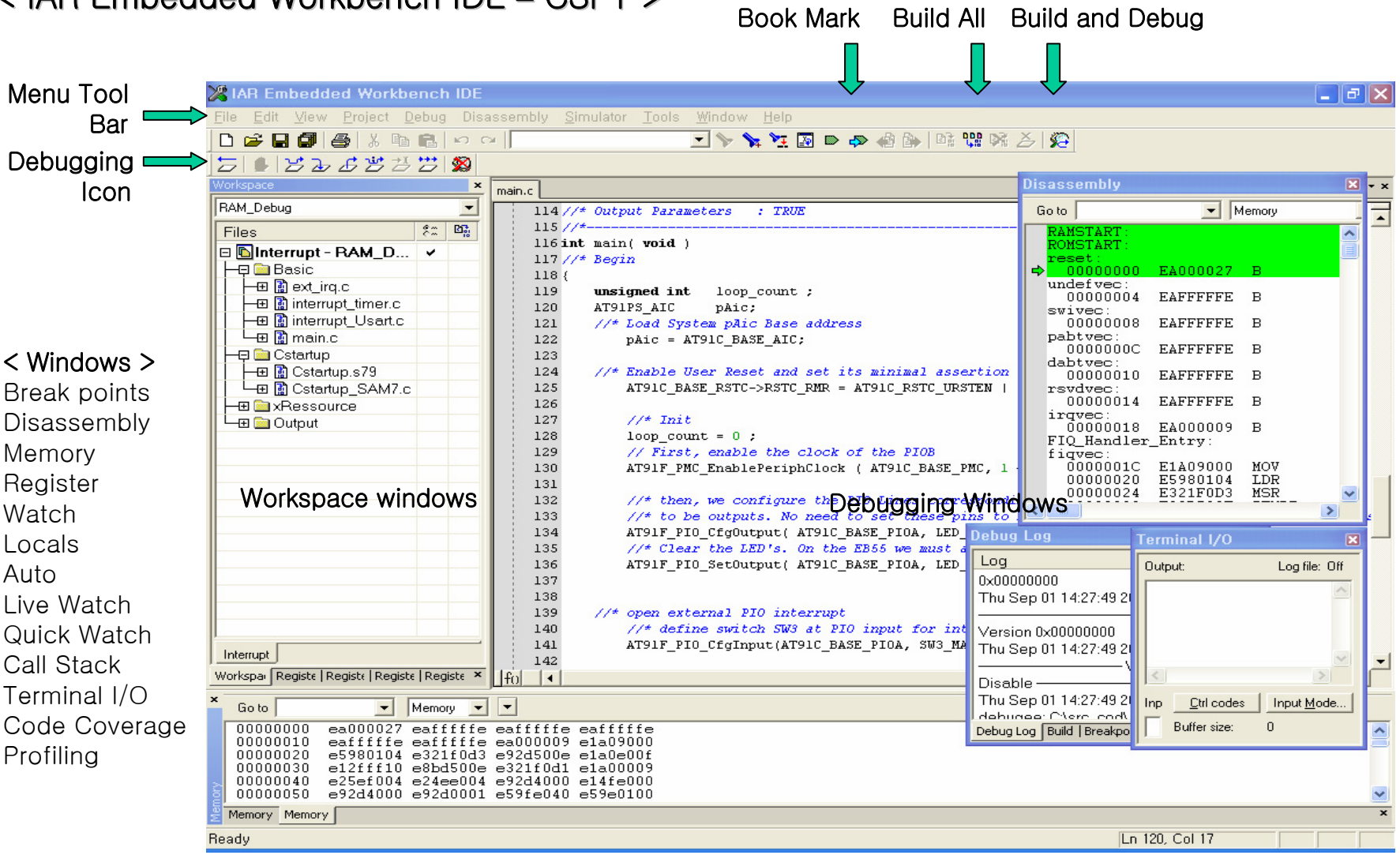
• 마이크로비젼 : http://www.mvtool.co.kr
- 제품소개 Products > 컴파일러/디버거 > EW-Compiler 목록 중 컴파일러 종류 > Atmel – AVR 클릭 (IAR Home page LINK)

• IAR : http://www.iar.com
- Downloads > Evaluation Software > IAR Embedded Workbench – 30 day evaluation edition > AVR
- Downloads > Evaluation Software > IAR Embedded Workbench – KickStart edition > AVR (4KB edition)

< IAR Embedded Workbench IDE >

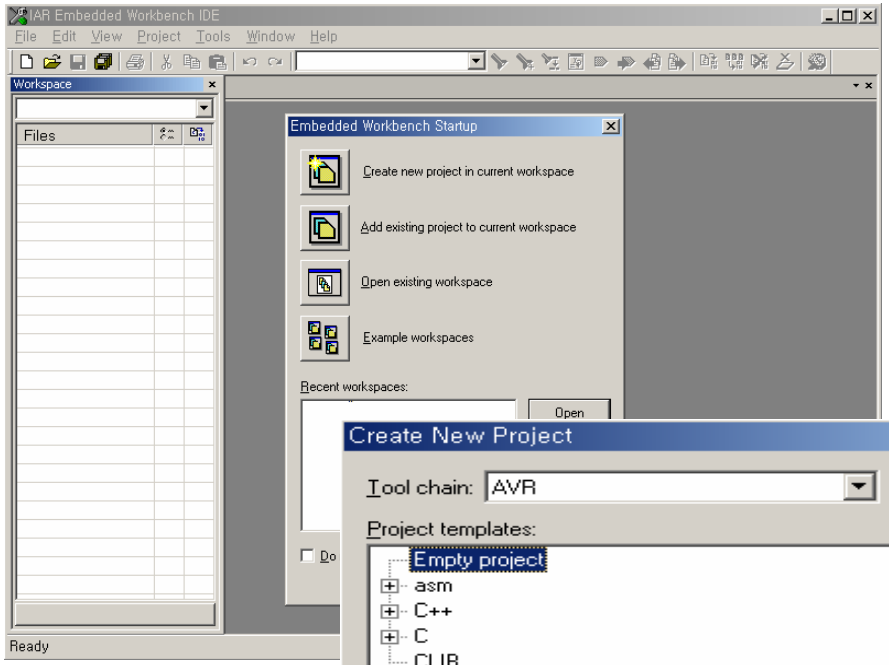


< IAR Embedded Workbench IDE – CSPY >



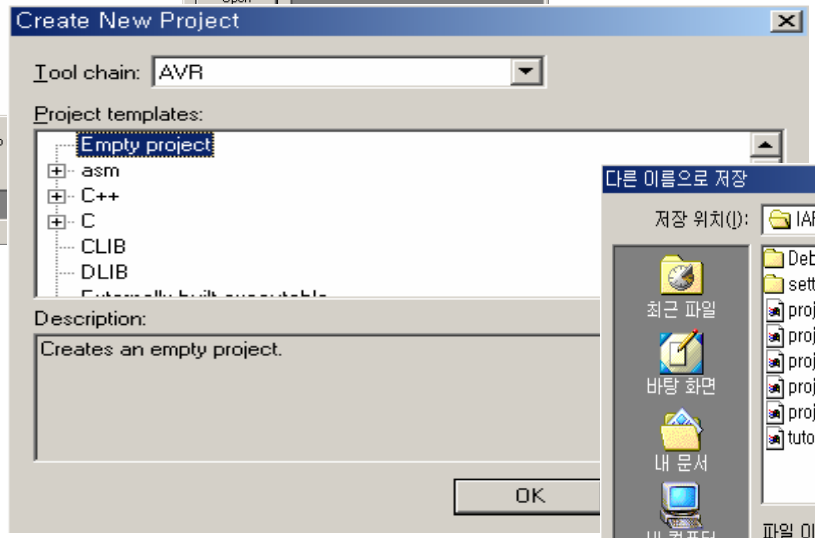
< Windows >
Break points
Disassembly
Memory
Register
Watch
Locals
Auto
Live Watch
Quick Watch
Call Stack
Terminal I/O
Code Coverage
Profiling

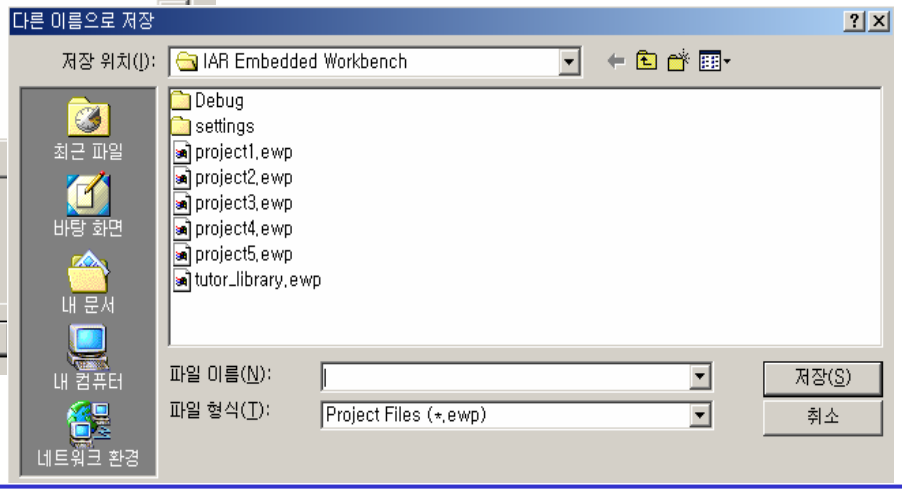
< IAR Embedded Workbench 새 프로젝트 만들기 >



우선, **Create New Project in current Workspace** 실행시킴으로써, 미리 생성된 Workspace 에 Project 를 추가합니다.

Create New Project 창이 뜨면, **Empty Project** 를 선택한 후에, **OK**를 누르면 New project 가 생성되어 Workspace 에 추가됩니다. (" Project name.ewp " 생성)



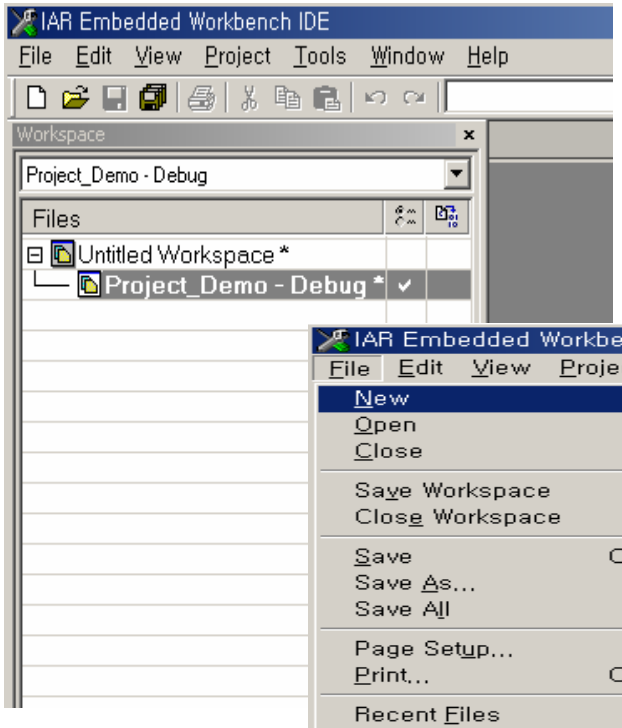


Note) 생성되는 파일들.

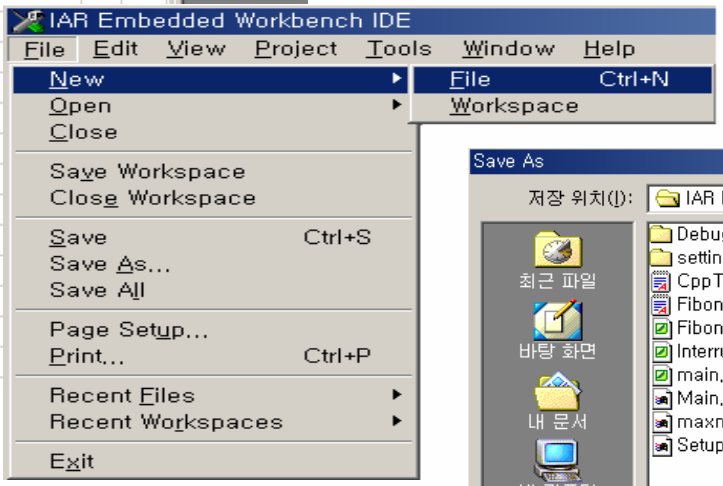
*.EWW *.EWP *.EWD *.ESDT Setting Folder... 등

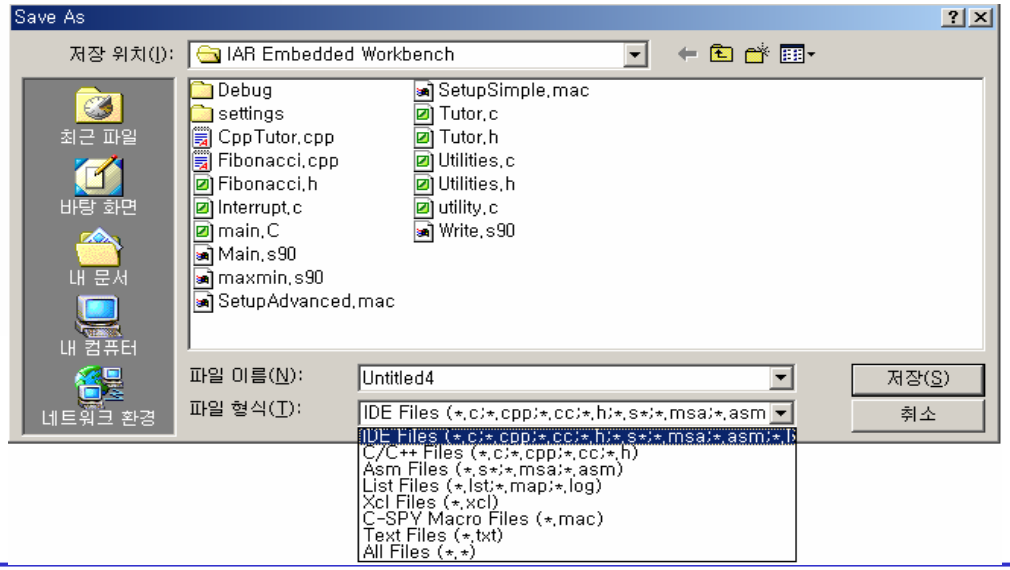
< IAR Embedded Workbench 새 소스 파일 프로젝트에 링크 시키기 >

File -> New -> File 를 선택하면 새로운 텍스트 문서창이 열리며 Untitled1, Untitled2... 등으로 기본 파일이 생성됩니다.



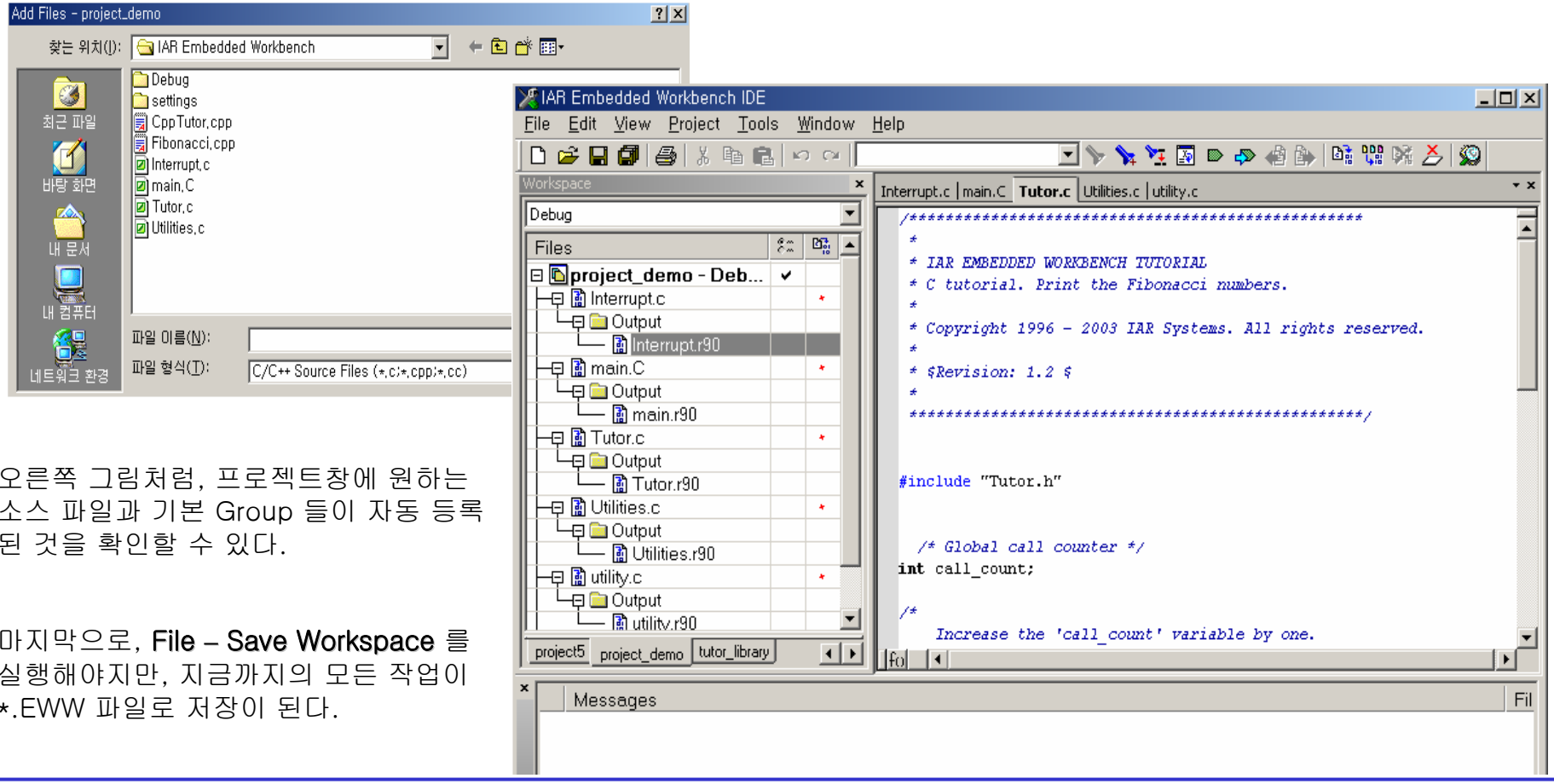
생성된 텍스트 문서를 어떤 파일로 사용할 지 결정한 후에
File -> Save As... 를 눌러 원하는 source-name.C 를 입력하고 저장.
이렇게 생성된 파일은 Project 에 추가 (Add) 가 되야, 비로서 컴파일
하기위한 기본 틀이 완성이 되는 것이다.





Project -> Add Files... 을 누르면 아래의 창이 뜨며, 원하는 소스를 선택하여 열기 버튼을 누르면, 현재 커서가 위치해 있는 프로젝트에 소스가 등록이 된다. (Active Project 가 기준이 아님을 명심하자.)

만약, Group (Output, Document ..)별로 추가하고 싶은 파일이 따로 있다면, 그룹을 만든 뒤 (Project – Add Group), 커서를 그룹에 위치시키고 같은 방식으로 파일을 추가 시키면 된다.

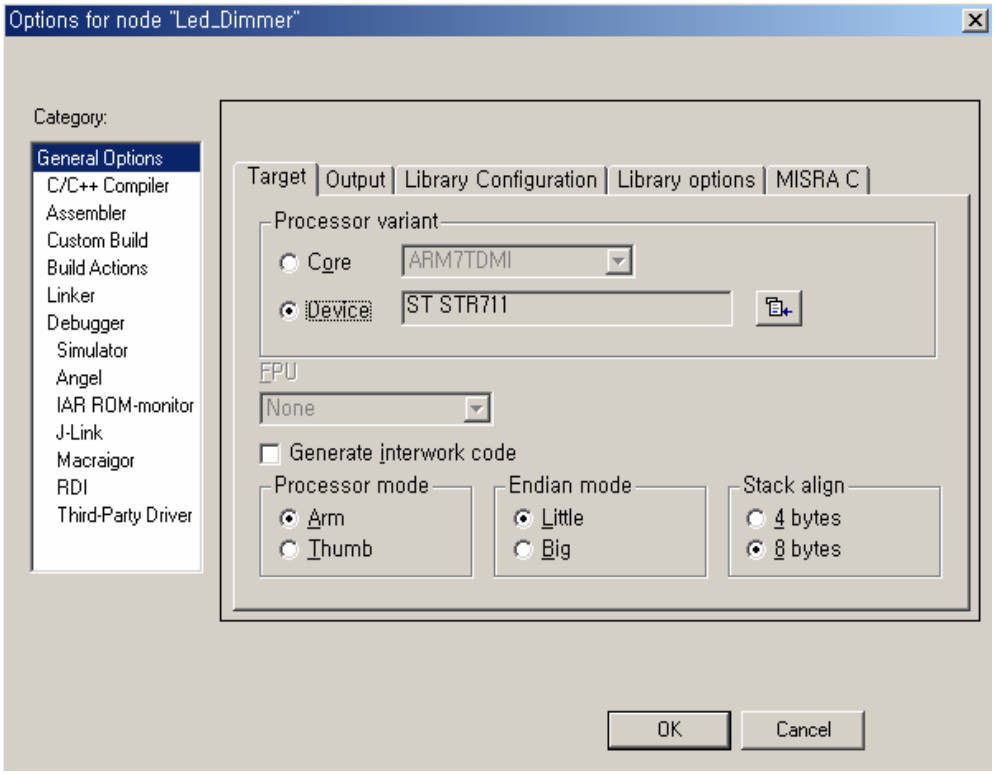


오른쪽 그림처럼, 프로젝트창에 원하는 소스 파일과 기본 Group 들이 자동 등록된 것을 확인할 수 있다.

마지막으로, File – Save Workspace 를 실행해야지만, 지금까지의 모든 작업이 *.EWW 파일로 저장이 된다.

1. Project -> Option -> General option -> Target

Target 은 사용하는 CPU 에 맞게 컴파일러 시스템을 맞추는 가장 기본적이고도 중요한 작업이다.



Processor variant :

사용하게 될 Device 의 Core 를 선택하거나 Device 를 직접 선택하도록 한다. (*.i79)

이 설정은 Core 별 Library 참조와 CSPY 단계에서의 정확한 DDF 파일을 참조하기 위해서 이다.

* DDF : Device Description file

FPU :

Float-pointing unit 용으로써, VFP 코프로세서가 지원 이 될 경우에만 사용할 수 있다.

참고로, ARM9E family 계열은 VFPv2(VFP9-s) 를 지원하고 있다. (단, Core 선택 시에만 가능.)

* VFP : Vector Floating pointing

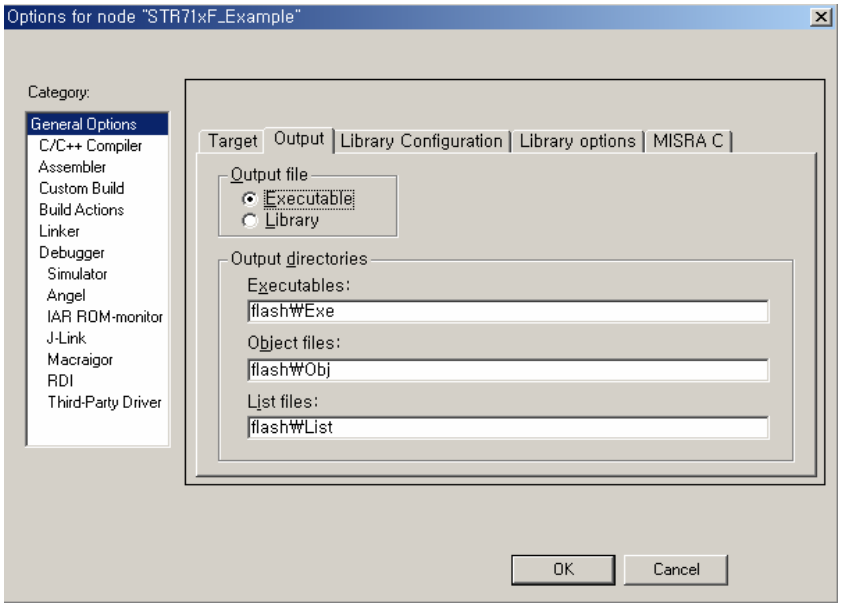
Generate Interwork Code :

ARM mode 와 Thumb mode 를 병행해서 사용할 경우 설정한다.

Default :

- Generate Interwork Code [**enable**]
- Processor mode [**Thumb**]
- Endian mode [**Little**]
- Stack align [**4 bytes**]

2. Project -> Option -> General Option -> Output / Library Configuration



Output 은 컴파일 후에 실행 될 혹은 참조하게 될 파일들이 생성되는 Output files 이 저장되는 폴더를 명시한다.

Output files :

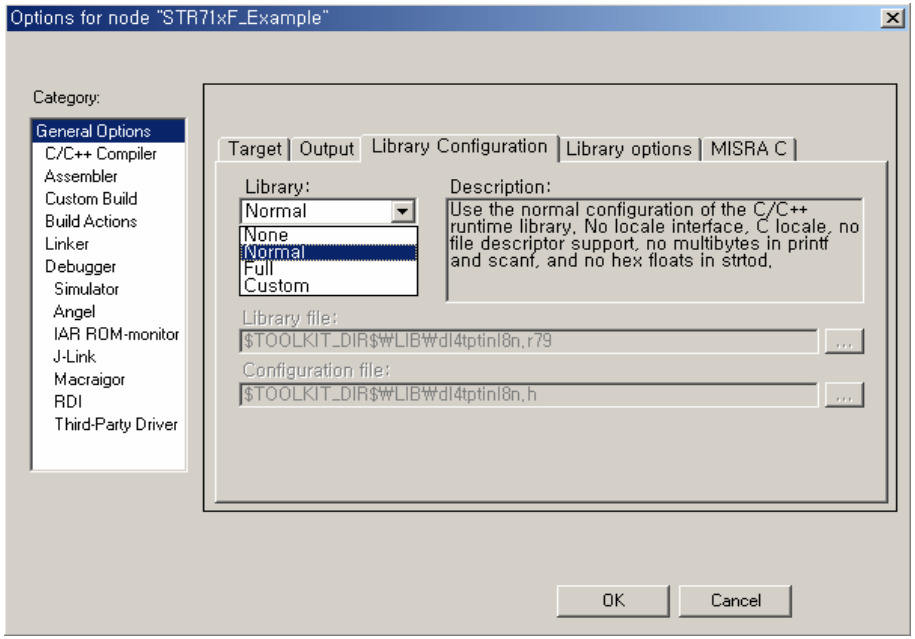
Executable type 과 Library type 중 선택

Output Directories :

Executable(Libraries) /Object /List (MAP) files 생성 디렉토리 설정

Library Configuration 은 사용하게 될 Library 를 결정한다. 이미, 기본적으로 참조하게 될 Library file 은 Target 설정에 의해서 대부분 지정된다.

- Library (Normal or Full)
- Core
- ARM/ Thumb
- Interwork
- Endian
- Stack Align
- VFP



3. Project -> Option -> General Option -> Library Options

Library Options 은 printf /scanf formatter 의 Level 을 설정할 수 있다.

Formatting capabilities	_PrintfFull (default)	_PrintfLarge	_PrintfSmall	_PrintfTiny
Multibyte support	*	*	*	No
Conversion specifiers a, and A	Yes	No	No	No
Conversion specifiers e, E, f, F, g, and G	Yes	Yes	No	No
Conversion specifier n	Yes	Yes	No	No
Format flag space, +, -, #, and 0	Yes	Yes	Yes	No
Length modifiers h, l, L, s, t, and z	Yes	Yes	Yes	No
Field width and precision, including *	Yes	Yes	Yes	No
long long support	Yes	Yes	No	No
Approximate relative size	100%	85%	20%	10%

Table 11: Formatters for printf

Formatting capabilities	_ScanfFull (default)	_ScanfLarge	_ScanfSmall
Multibyte support	*	*	*
Floating-point support	Yes	No	No
Conversion specifier n	Yes	No	No
Scan set [and]	Yes	Yes	No
Assignment suppressing *	Yes	Yes	No
long long support	Yes	No	No
Approximate relative size	100%	35%	30%

Table 12: Formatters for scanf

Printf formatter :

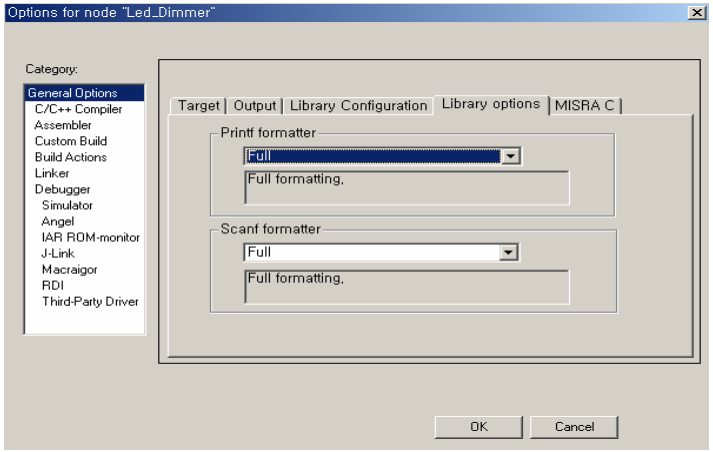
Full (default) / Large/ Small/ Tiny Level

Scanf formatter :

Full (default) / Large/ Small Level

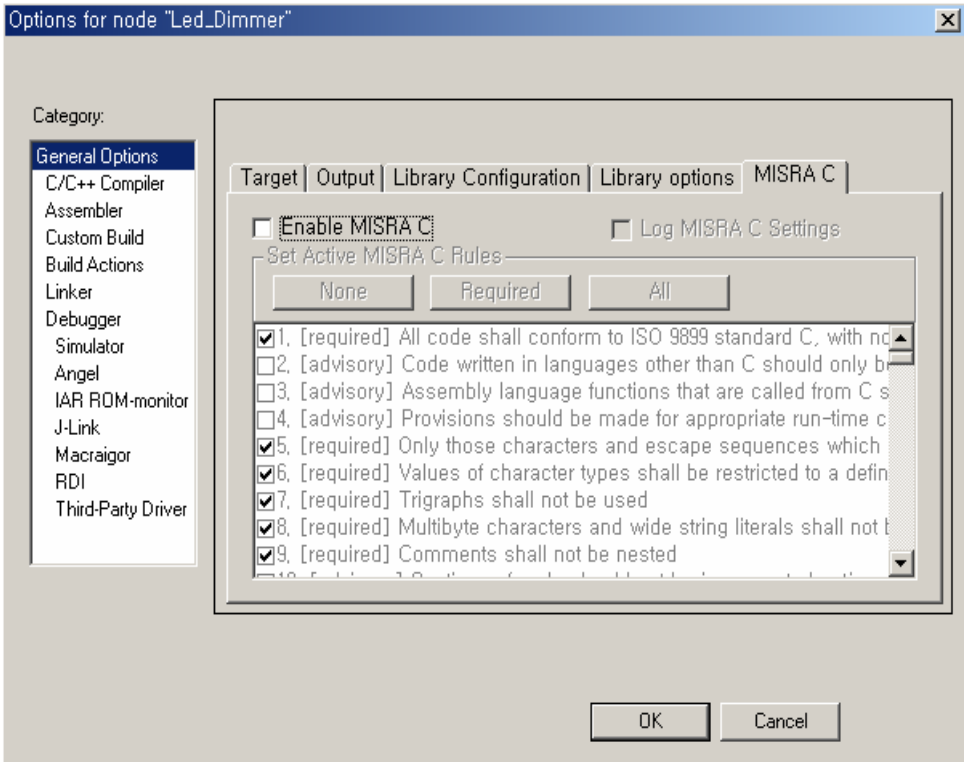
* : Library Configuration 설정에 따라 ON/OFF

(Normal – No/ Full – Yes)



4. Project -> Option -> General option -> MISRA C

MISRA C 는 자동차 산업용으로 개발된 추가 Option 팁 유틸이다.



How does it work?

The IAR MISRA C Checker is completely integrated with the IAR C Compiler. From IAR Embedded Workbench, you can control which MISRA C rules are checked; the settings will be used for both the compiler and the linker.

What is MISRA C?

The Motor Industry Software Reliability Association (MISRA) is an organization in the UK that promotes safety in automotive software.

In 1998, MISRA published its “Guidelines for the Use of the C Language in Vehicle Based Software”. The guidelines address the ambiguities of the C language and establish coding rules for the automotive industry.

MISRA C includes 127 rules. 93 of these are required and the remaining 34 are advisory. All rules apply to the source code and not to the object code generated by the compiler.

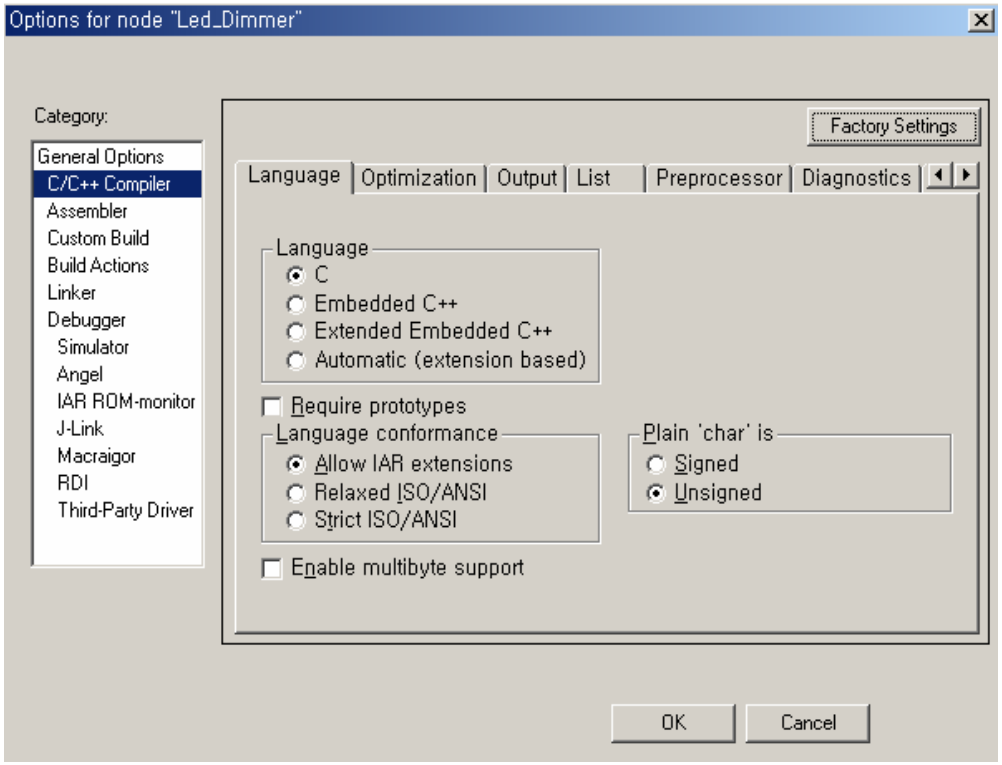
Who should use MISRA C?

Compliance with the MISRA C guidelines is a require-ment in many automotive companies but could be beneficial in any development organization.

The guidelines enforce sound coding practices and address the ambiguities of C; they help developers write code in a consistent manner and avoid confusing constructions.

5. Project -> Option -> C/C++ Compiler -> Language

C/C++ 컴파일러에 관련된 기본적인 옵션을 설정한다.



Language : [C]

사용하게 될 컴파일러의 종류에 대해서 결정한다.

Default 로 C language 로 설정이 되어있다.

Require prototypes : [Disable]

함수에 대한 프로토타입을 설정하여 작업.

* EWARM_User Guide " Require prototypes " 참조

Language conformance : [Allow IAR extensions]

현재 설정된 language 의 상태를 가장 기본적인 Strict ISO/ANSI Type 으로 할 지, 혹은 Core 와 컴파일러에 의해서 확장된 language 로 할지 설정한다.

* Compiler Reference Guide " IAR language extension " 참조

Enable Multibyte Support : [Disable]

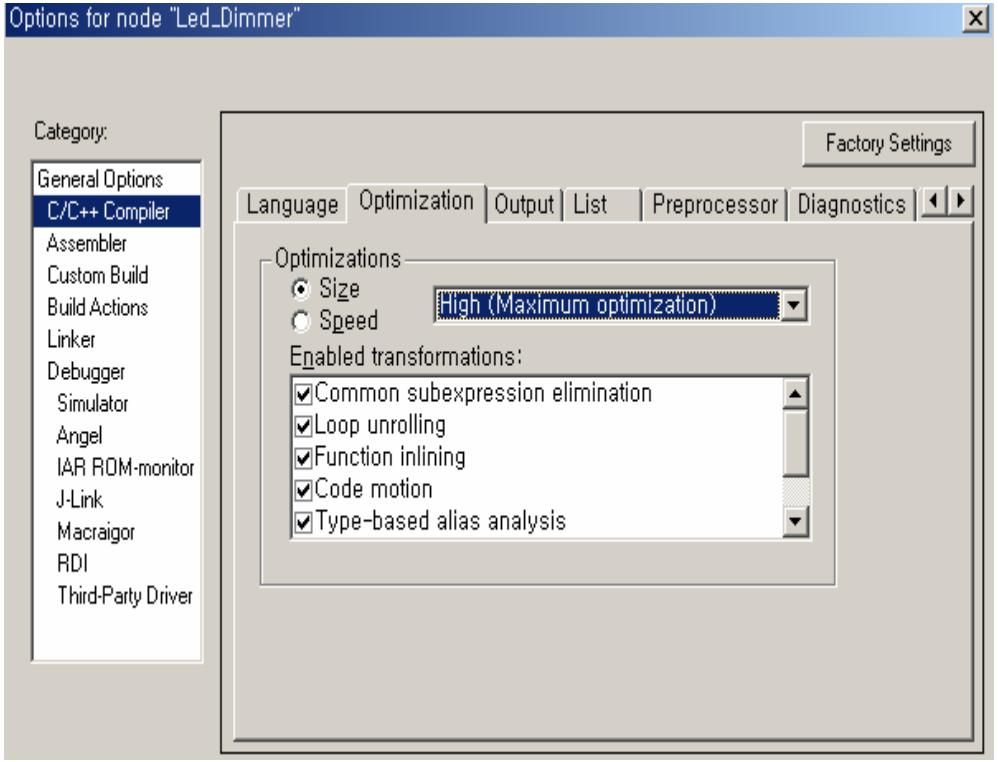
Multibyte의 사용 유무를 결정 할 수 있다.

Plain 'char' is : [Unsigned]

(Unsigned) " Char " 변수 타입을 변경할 수 있다.

6. Project -> Option -> C/C++ Compiler -> Optimization

C/C++ 컴파일러에 관련된 기본적인 옵션을 설정한다.



이 설정은 최초 Transformation 설정이 Debug 인지 Release 인지에 따라서 틀려지며, 임의적으로 변경이 가능하다.

Optimization : [Size]

ARM IAR C/C++ 컴파일러에서는 기본적으로 2가지의 Optimization model(**Size** or **Speed**) 에 따라서, 각각의 4가지 Optimization Level 을 선택할 수 있다.

Optimization Level :

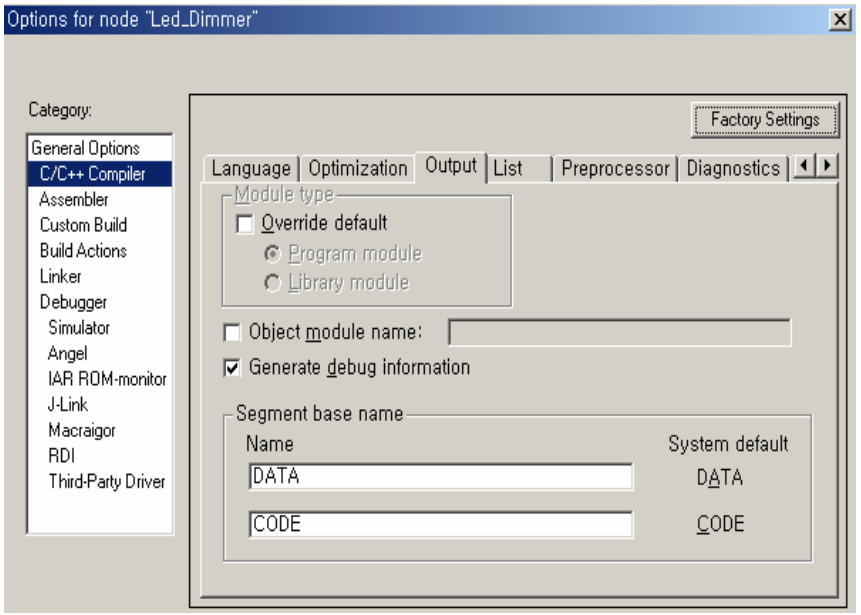
- None [Best debug support]
- Low
- Medium
- High [Maximum optimization]

Enabled transformation :

- Common Sub-expression elimination
- Loop unrolling
- Function in-lining
- Code motion
- Type-based alias analysis
- Static variable clustering
- Instruction scheduling

* EWARM compiler Guide " Efficient coding for ... " 참조

7. Project -> Option -> C/C++ Compiler -> Output / List

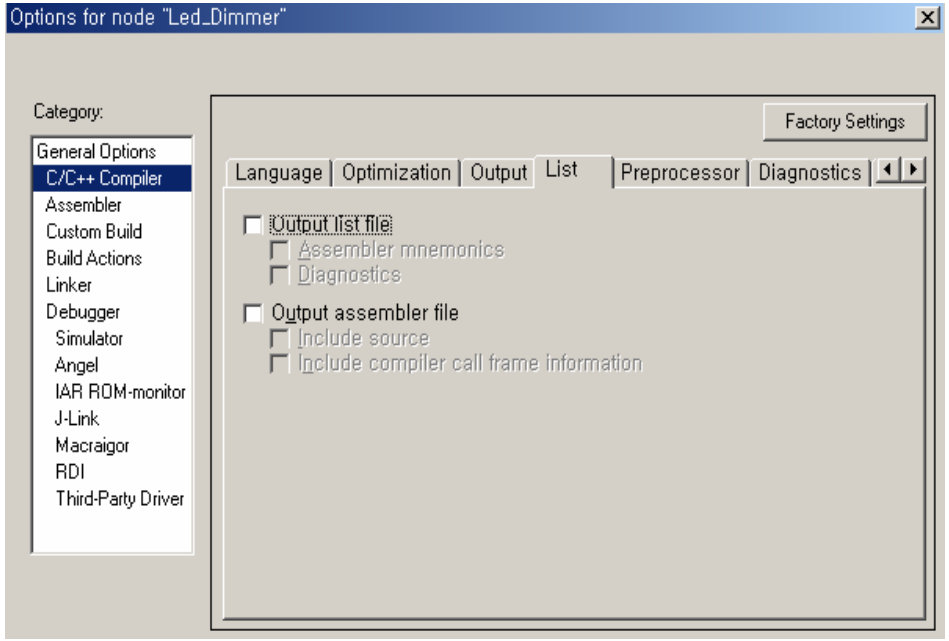


Output 은 컴파일된 파일을 program module[default] 과 Library module 로 구분하여 컴파일 설정한다.
생성된 module은 임의적으로 이름설정이 가능하며, 디버깅용 소스 정보를 포함 유무에 대해서도 설정할 수 있다.

* EWARM compiler Guide " segment reference " 참조

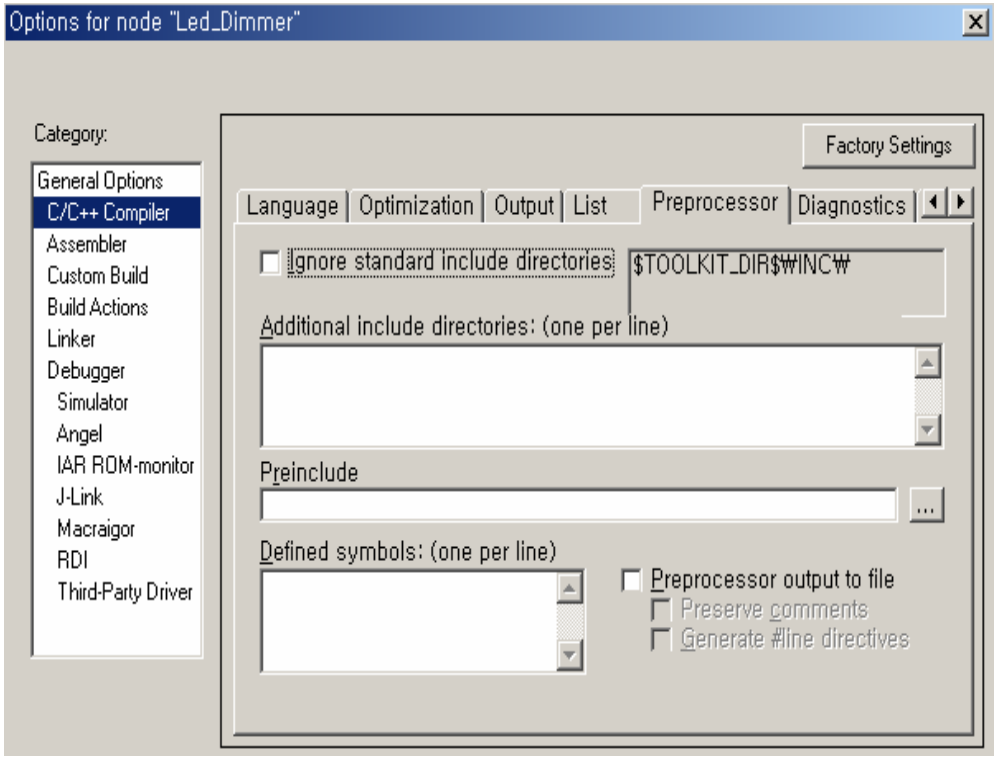
List 는 list 파일(*.lst)과 ASM source (*.s79) 을 생성할 수 있도록 만들어 주는 설정이다.
각각의 file 생성에 따른 부가적으로 Assembler mnemonics 와 diagnostics 가 포함이 된 파일 생성 및 Source include 와 Compiler call frame information 포함이 가능해진다. (Demo 제한)

* EWARM User guide " compiler – list " 참조



8. Project -> Option -> C/C++ Compiler -> Preprocessor

Preprocessor 는 컴파일러의 고유 기능 중 하나인 전처리 기능이다.



Ignore standard include directories : [Disable]

기본 설치 디렉토리인 C:₩...arm₩inc 폴더를 ($TOOLKIT_DIR&₩INC₩) 를 기본 참조 include directory 로 설정한다.

Additional include directories : (one per line)

설정된 기본 디렉토리 이 외의 include 디렉토리에 대해서 추가 입력할 수 있다.

Ex) $PROJ_DIR$₩srciar₩

$PROJ_DIR$₩..₩..₩

Pre-include :

include 파일을 전처리기처럼 컴파일러 옵션상에서 지정할 수 있다.

Ex) test_source.h

test_asm.s79

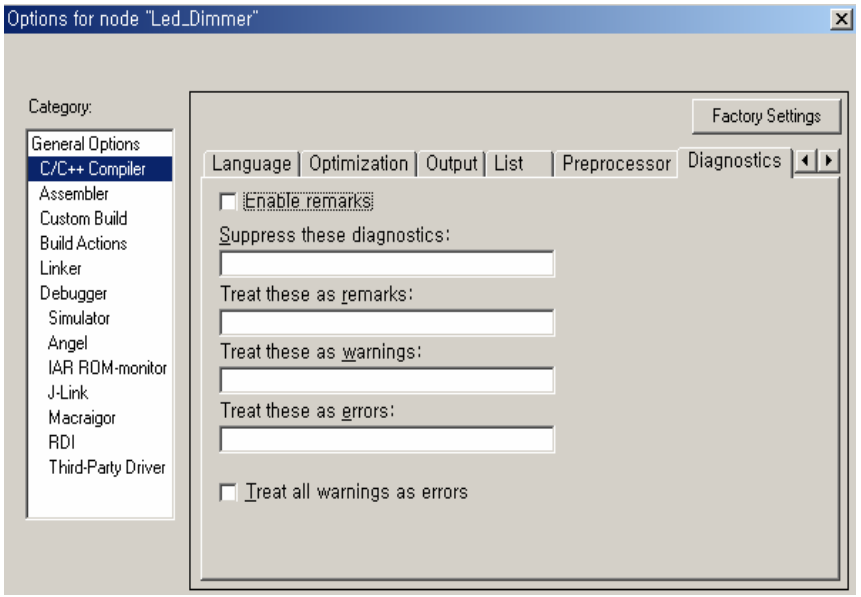
Defined symbols : (one per line)

Source 내에서 사용이 될 Define symbol 을 입력해 넣는다.

Preprocessor output to file 을 통하여 파일로 저장도 가능하다.

Ex) TESTVER = 1

7. Project -> Option -> C/C++ Compiler -> Diagnostics/ Extra Options

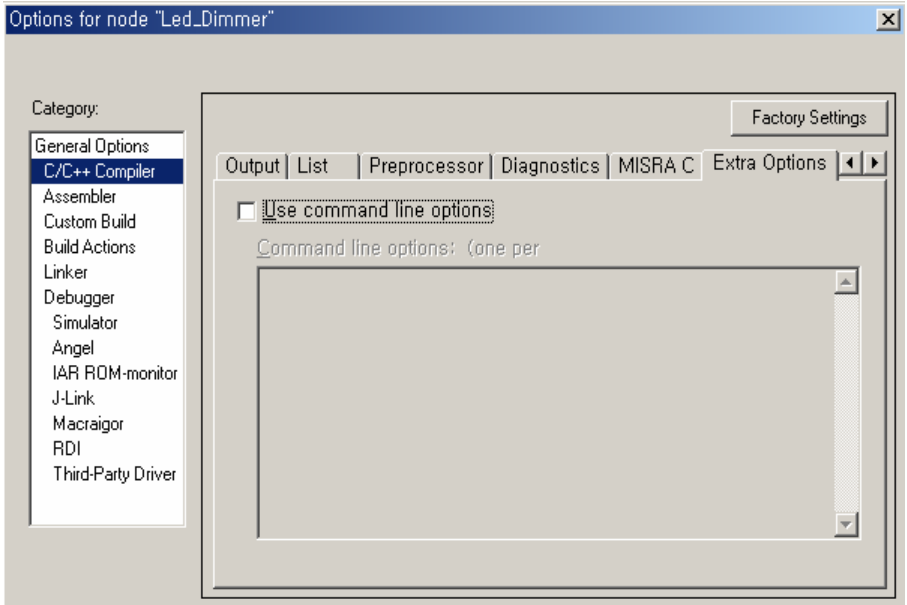


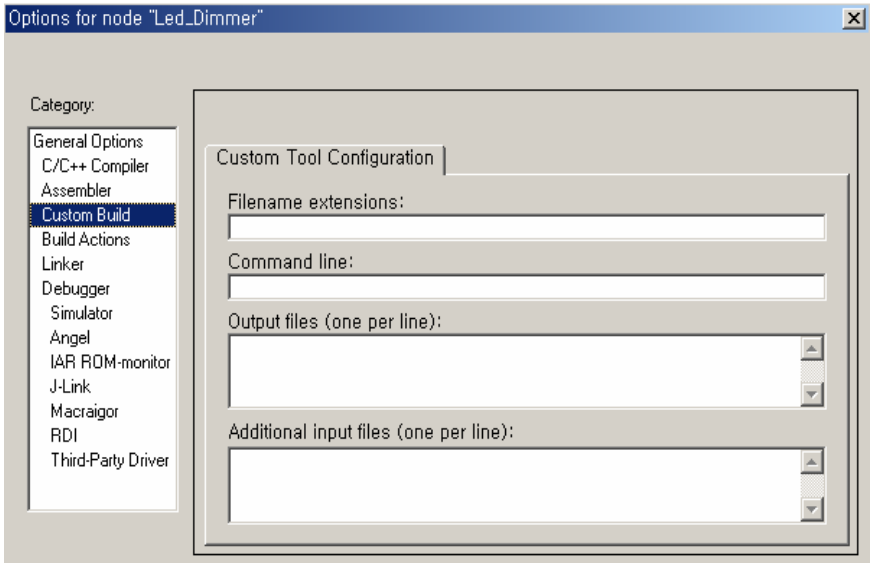
Diagnostics 는 컴파일 과정에서 발생 되는 Remark/ Warning/ Error Message 에 대한 처리를 돕는다.

* EWARM User guide " diagnostics " 참조
* EWARM Compiler guide " diagnostics " 참조

Extra Options 는 GUI 환경에서 구현 할 수 없는 컴파일러의 여러 가지 다른 Option 들에 대해서 사용자에 의해서 임의적으로 설정할 수 있도록 마련해둔 Option 설정 창이다.

* GUI : Graphic User Interface





8. Project -> Option -> Customer Build

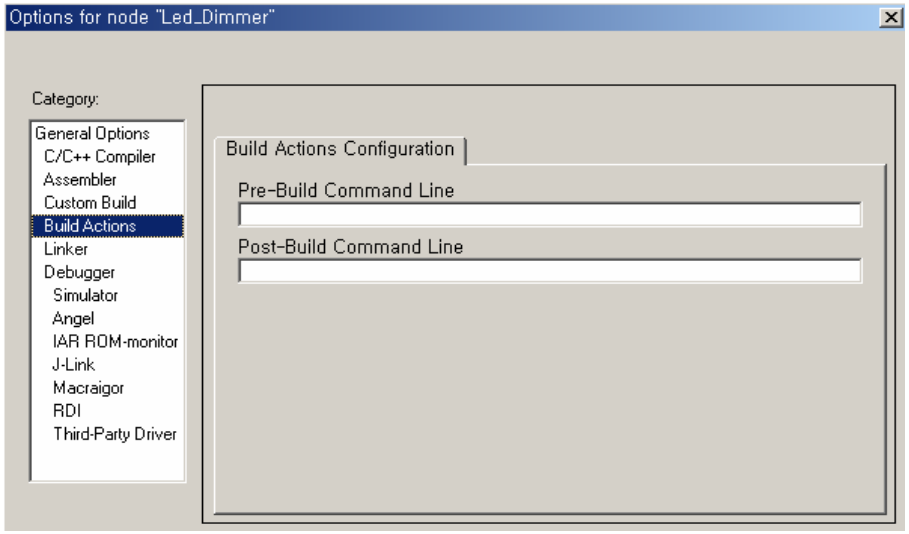
Customer Build 기능은 최근에 추가된 Option 으로써,
IAR 사에서 제공되는 S/W 이외의 다른 S/W Tool 에 대해서,
Tool chain 을 사용해 컴파일 환경을 제공하는 기능이다.
Ex) Lex , YACC

* EWARM User guide " Extending the tool chain " 참조
* EWARM Compiler guide " diagnostics " 참조

9. Project -> Option -> Build Actions

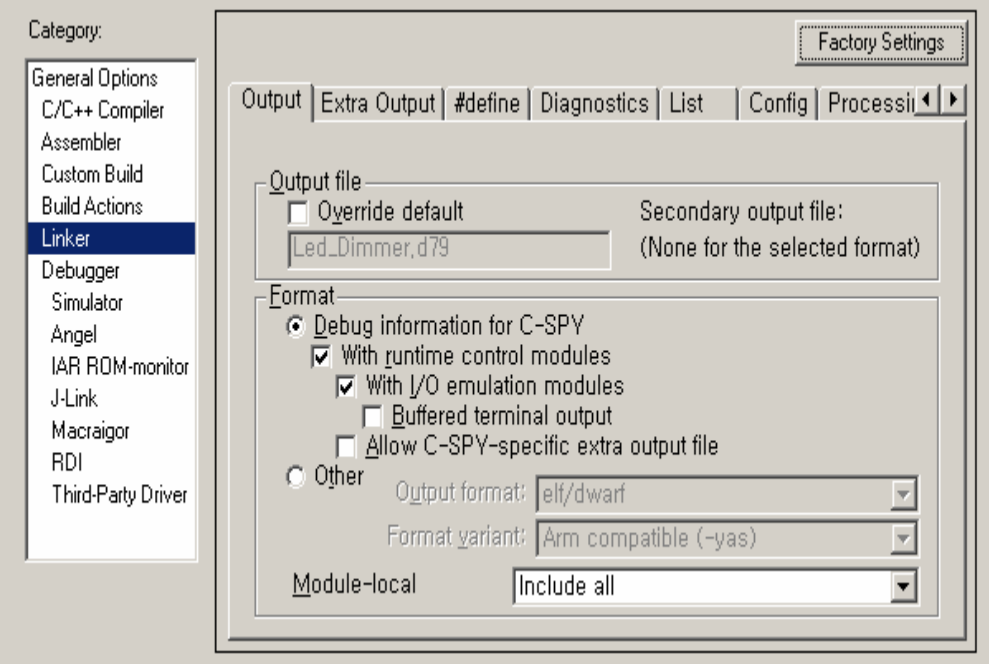
Build Actions 는 필요에 의해서 build 전후에 pre-build / post-build 하기 위해서 설정하는 옵션이다.

* EWARM User guide " building " 참조



10. Project -> Option -> Linker -> Output

Output 은 실행파일의 Format 형태와 Output file 타입을 설정할 수 있다.



Output : [Disable]
C-SPY 용 Format 외에 다른 Format 이 필요할 경우 사용한다.

Output format :
- mpds-code : Binary Image 가 필요한 경우
- elf/dwart : 다른 H/W tool Debugger 를 사용 할 경우
- intel-stadard : HEX format 의 file 이 필요할 경우

Module-local : Symbol 호출 시 제약 Option 이다.

Output :
임의로 설정된 Format 에 따라서 실행파일의 형태가 선택되며, 기본적으로 실행파일명은 최초 프로젝트 명을 따르도록 되어 있다.

Ex) project.d79 (For Debugging format),
project.elf (For Executable and Linkable format)
project.tsk (For binary image format)

Format :

Debug information for C-SPY : [Disable]
디버깅 정보를 C-SPY 용 Format 으로 변경

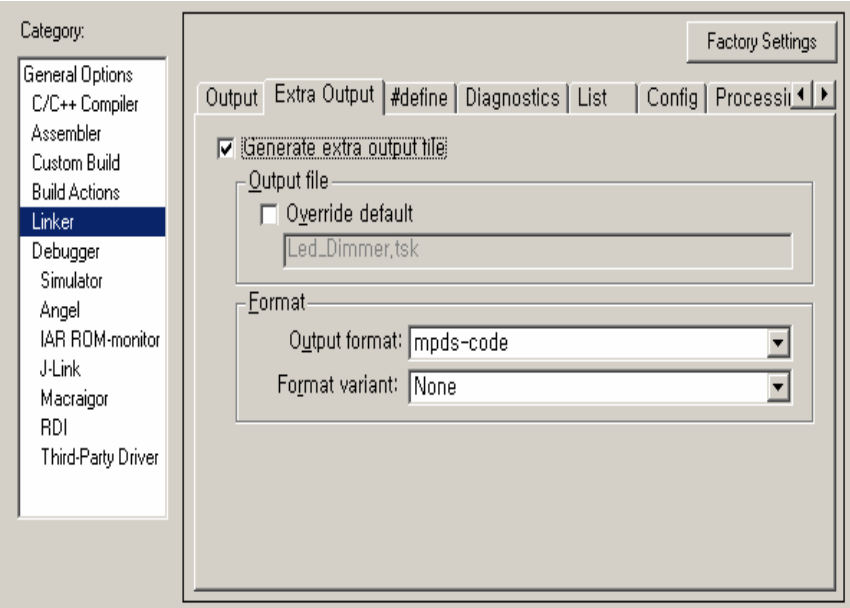
With runtime control modules : [Enable]
Program abort, exit, assertions etc.

With I/O emulation modules : [Enable]
Terminal I/O handling 설정 (stdin stdout)

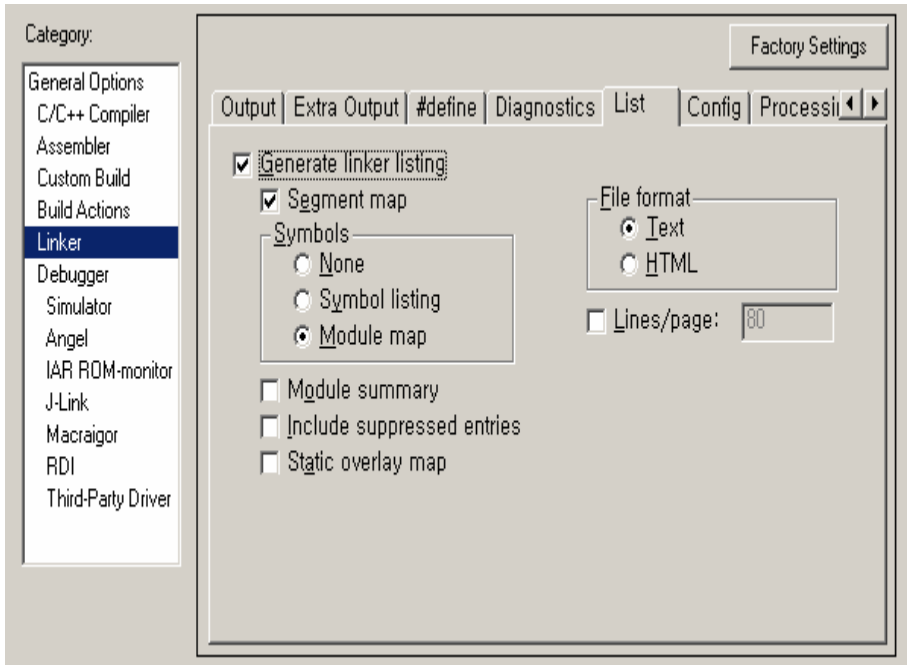
Buffered terminal output : [Disable]
Terminal 이용 시 Buffer 역할을 해준다.

Allow C-SPY-specific extra output file:[Enable]
Extra Output page enable Option

11. Project -> Option -> Linker -> Extra output / List



Extra output 은 **Output** 에서 설정된 Format 의 실행파일 이 외에 다른 실행 파일을 출력하고자 할 경우 사용한다. 일단, " **Options** – **Linker** – **Output** – (Format) **Allow C-SPY-specific extra output file** option 을 **Enable** 시킨 뒤에 사용할 수 있다.



List 는 MAP 파일(*.map) 수 있도록 만들어 주는 설정이다.

C/C++ Compiler 에 List 파일 생성 옵션과는 틀리다.

Linking 이 끝난 상태에서만 생성되며, **Symbol** Option 설정에 따라서 global / local symbol 포함여부를 결정할 수 있다.

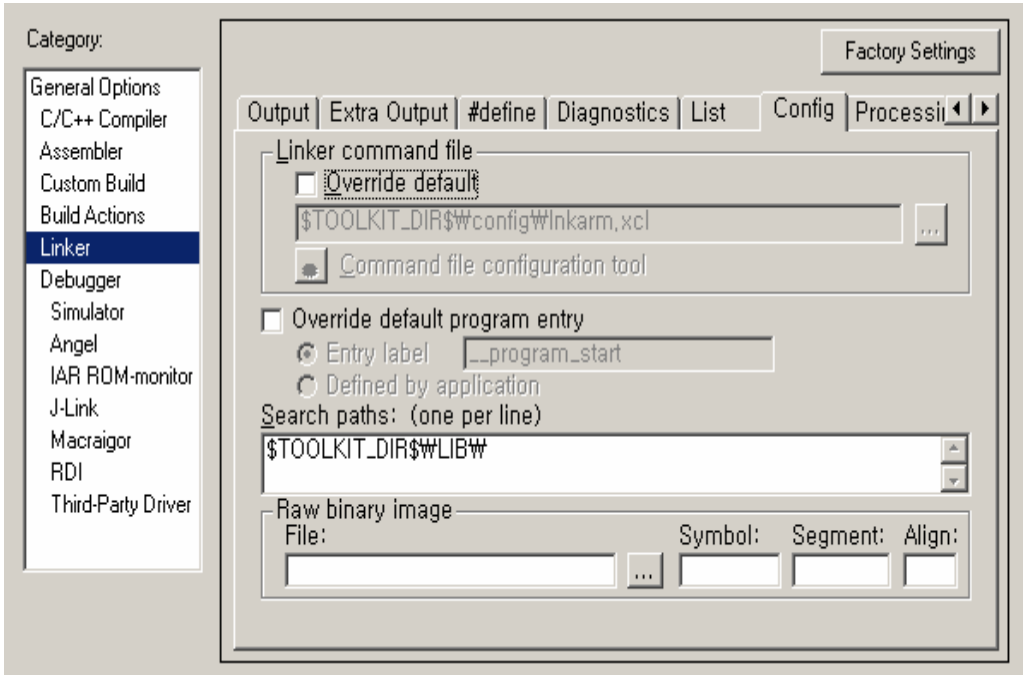
Module summary : module 별 메모리 사용량 표시

Include suppressed entries : 일반 List 파일에서 생략된 segment 에 관련된 모든 정보를 확인해 볼 수 있다.

Static overlay map : Static overlay system 정보 추가

12. Project -> Option -> Linker -> Config

Config 는 XCL 파일과 Startup code 의 entry point 등의 설정을 도와준다.

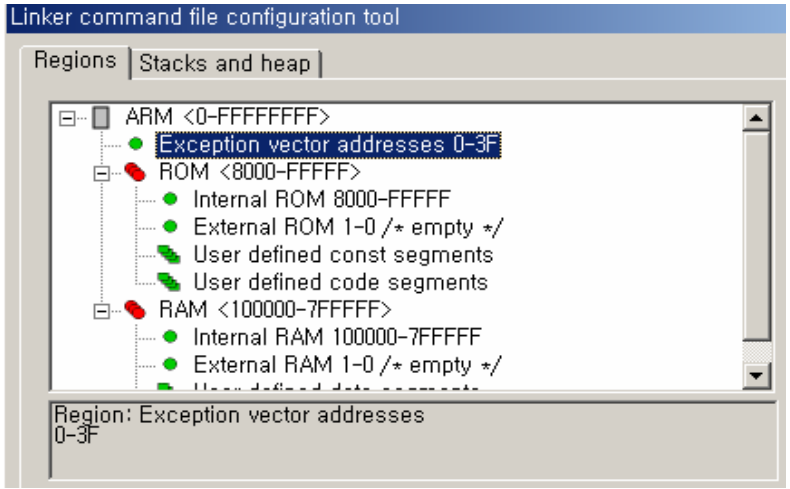


Linker command file :

기본적으로 iar_arm.xcl 파일을 선택이 되며, XCL 파일에는 기본적으로, 사용하는 CPU 설정을 비롯하여, Interrupt Vertor 사용 영역 설정과 ROM과 RAM의 세그먼트가 링크 된 어드레스 번지 설정 등의 내용을 포함하고 있다.

Command file configuration tool :

수정이 필요하다면, 직접 XCL 파일을 열어 보는 불필요함이 없이 UI 환경을 이용해서 편리하게 수정할 수 있다.



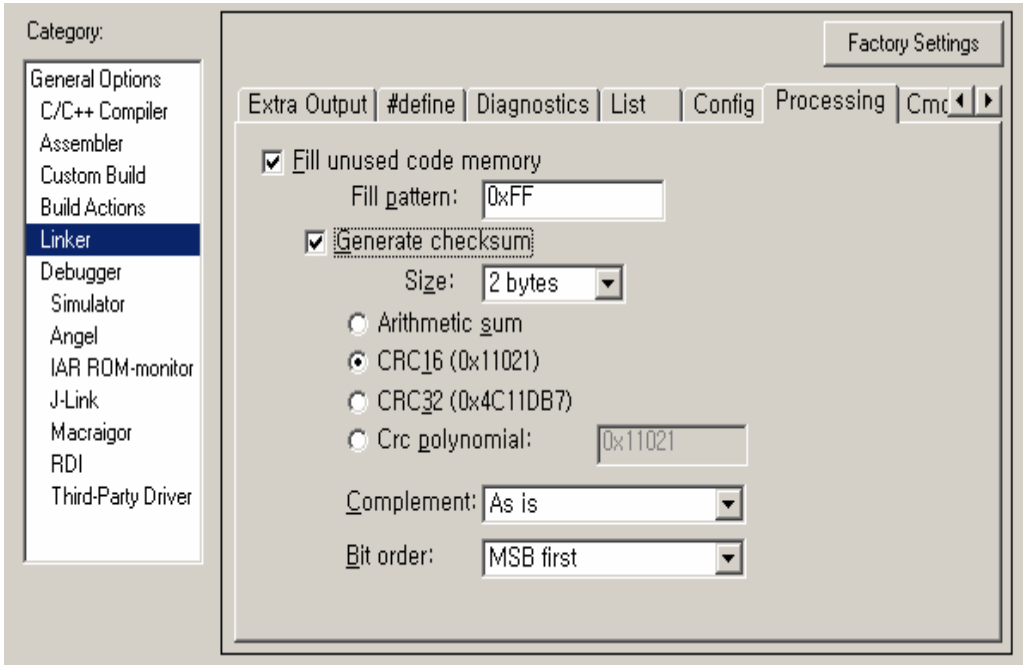
Override default program entry :

Cstartup code 에서 정의 되어 있는 entry point 의 위치를 재설정 해준다. __program_start 라면 문제가 되지 않지만, 그렇지 않을 경우 반드시 변경해 주어야만 한다.

Search paths : (one per line) 라이브러리 파일 path 설정

13. Project -> Option -> Linker -> Processing

Processing 은 컴파일 후 사용하지 않는 Code Memory 영역의 처리를 위해 사용된다.



Fill unused code Memory : [Disable]

Fill pattern :

사용하지 않는 공간에 채워넣을 패턴을 넣는다.

Ex) 0xFF

Generate checksum : [Disable]

Size : [1/2/4 bytes]

Arithmetic sum
정해진 사이즈에 따라 연속적으로 더한 결과 값 저장 (자지 올림 없음)

CRC16 (0x11021)
CRC 다항식을 사용한 확인

CRC32 (0x4C11DB7)
CRC 다항식을 사용한 확인

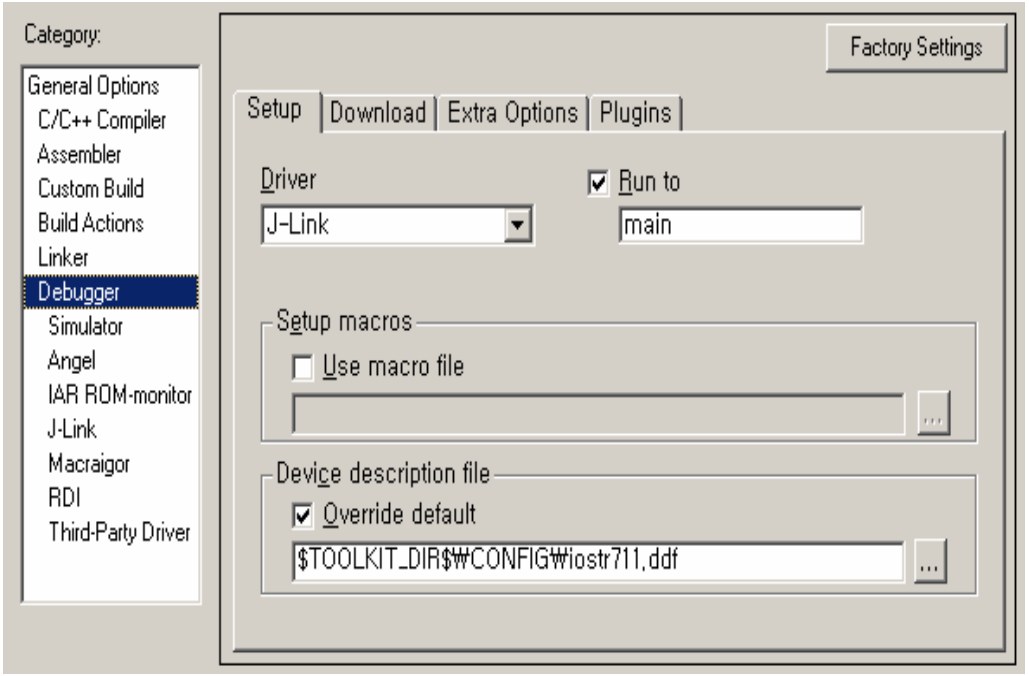
Crc polynomial :
CRC 다항식(직접 입력)을 사용한 확인

Complement :
1과 2의 보수 선택 사용

Bit Order :
상위(MSB)/하위(LSB) 비트의 순서 설정.

14. Project -> Option -> Debugger -> Setup

Debugger Setup 은 디버깅하기 위한 H/W tool을 설정하기 위한 옵션이다.



Driver : [Simulator]

총 7개의 Driver 가 제공되며, 목록상에 나열된 Driver 들은 기본적으로 각각의 *.dll 파일을 갖는다.
사용하게 될 드라이버로 설정하도록 한다.

Run to : [Enable]

임으로 시작 지점을 설정할 수 있도록 한다.
Ex) " Main " 혹은 PC = 0x00000000

Setup macros :

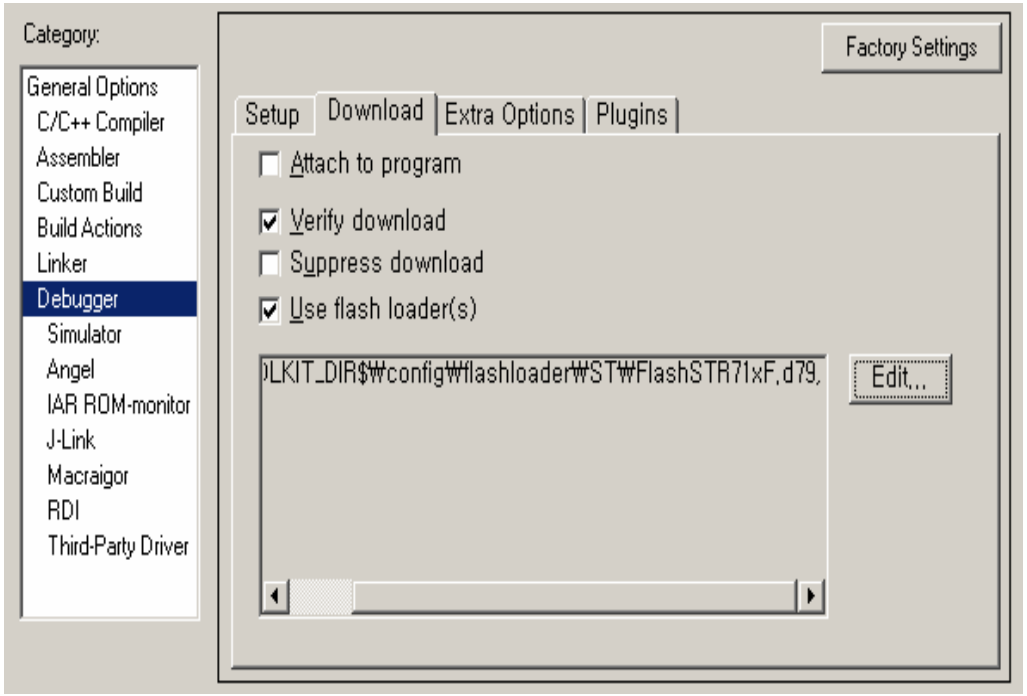
C-SPY 를 초기화 하기 위한 작업으로, Debugger 선택으로 기본적인 Macro file 은 제공된다.

Device description file :

Target 으로 설정된 Device나 core 의 DDF 파일을 지정해준다.

14. Project -> Option -> Debugger -> Download

Download 는 디버거를 구현하기 위한 Flash 다운로드 설정을 한다.



Attach to program : [Disable]

Target board 재부팅 없이 프로그램 다운로딩을 할 수 있다.

Verifiy to download : [Disable]

Flash download 뒤에 확인 작업 실행 유무를 설정한다.

Suppress download : [Disable]

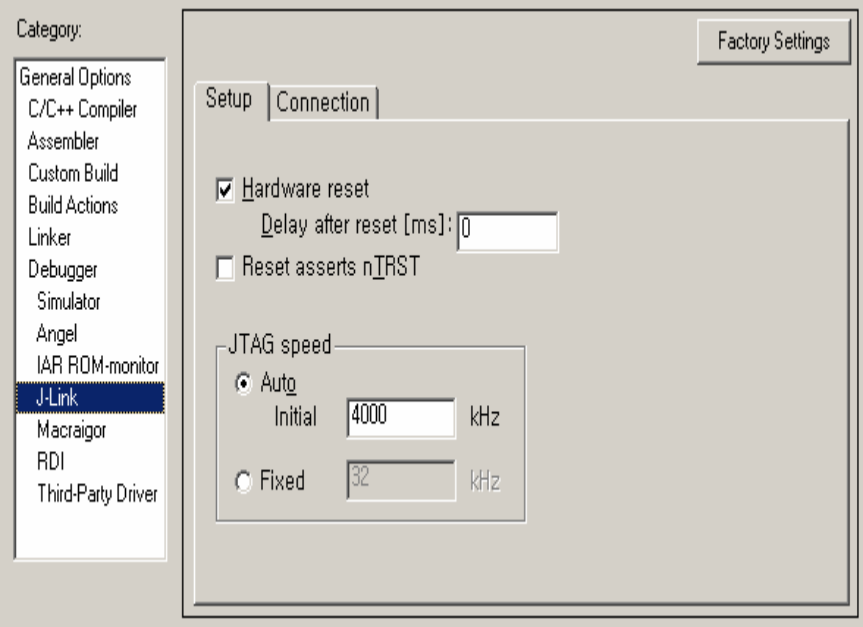
Debugging 용 file 을 다운로드 한다.

* Application Code Download 제한.

Use flash loader(s) : [Disable]

Flash Downloading 을 위한 Flash Loader 의 사용 유무를 확인한다.

15. Project -> Option -> Debugger -> J-Link



Setup 설정은 Debugger 를 J-Link 를 설정 후에 사용하는 J-Link 설정 옵션이다. H/W Reset 제어 부분과 JTAG speed 제어 부분에 대한 설정부분이다.

Hardware Reset : [Disabel]
H/W Reset 이 필요하다면, C-spy를 실행하기 전에 설정해야 하며, flash download 시에만 사용할 수 있다.

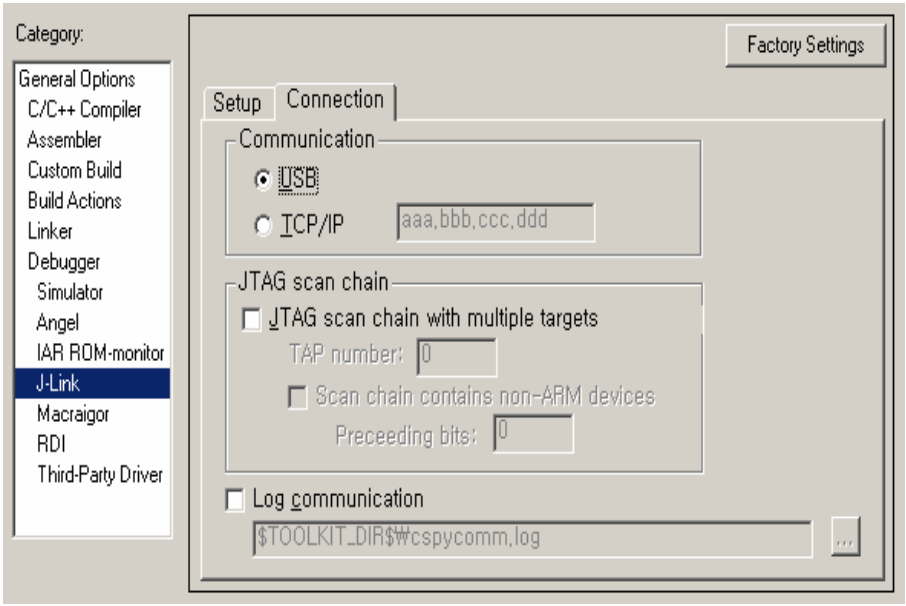
JTAG speed : [32KHz]
30 ~ 8000 KHz

Connection 은 J-Link 의 인터페이스 방식 설정과 JTAG SCAN chain 설정을 할 수 있다.

Communication : [USB]
USB mode 와 Ethernet 을 통한 인터페이스 가능.

JTAG scan chain :
하나 이상의 Device 를 사용할 경우, 사용하고자 하는 TAP 를 설정하여 사용한다.

* TAP : Test Access Point



IAR C 구입 문의

- 마이크로비젼

TEL : 02-3283-0101

FAX : 02-3283-0160

Web : http://www.mvtool.co.kr

※ 기타 질문 사항은 저희 홈페이지 Q/A 를 통하여 질문 주시거나, E-mail 부탁드립니다.

tech@mvtool.co.kr