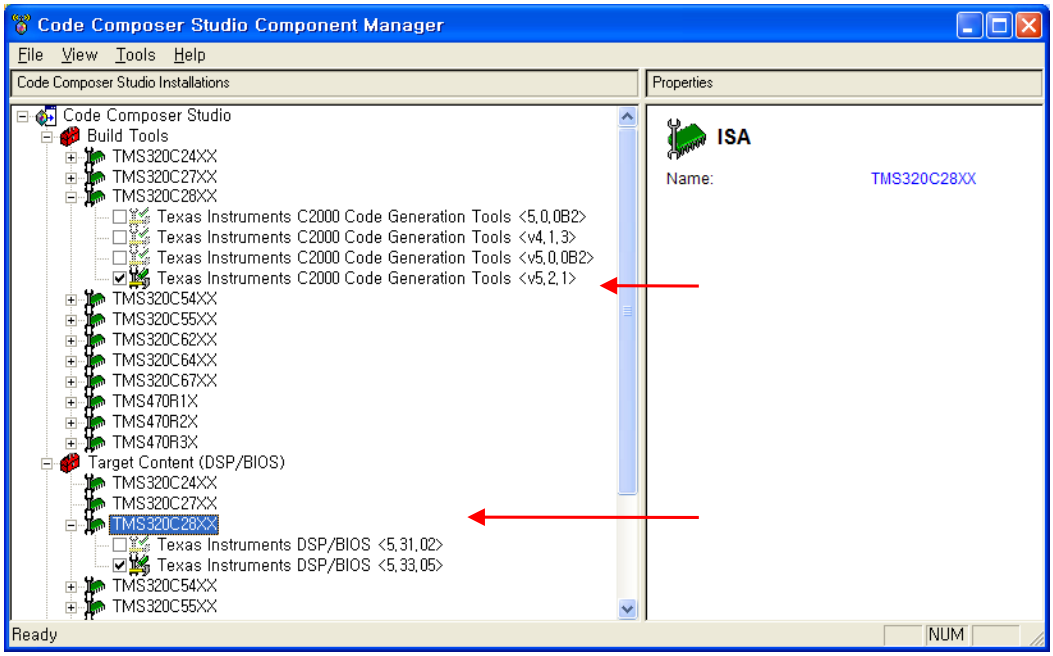
* CCS3.3 DSP/BIOS 환경 만들기

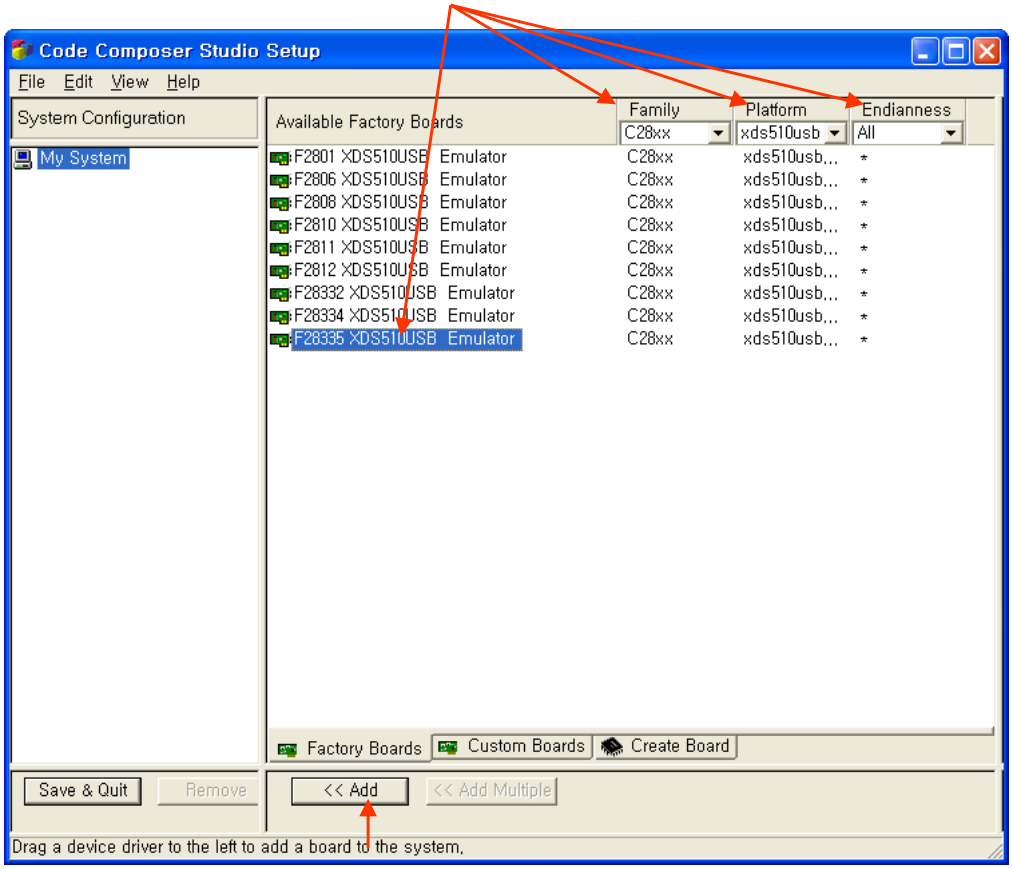
1. TI제공 예제 파일을 다운 받은후 압축을 작업 폴더에 풉니다.
   . spra958h.zip 예제프로그램

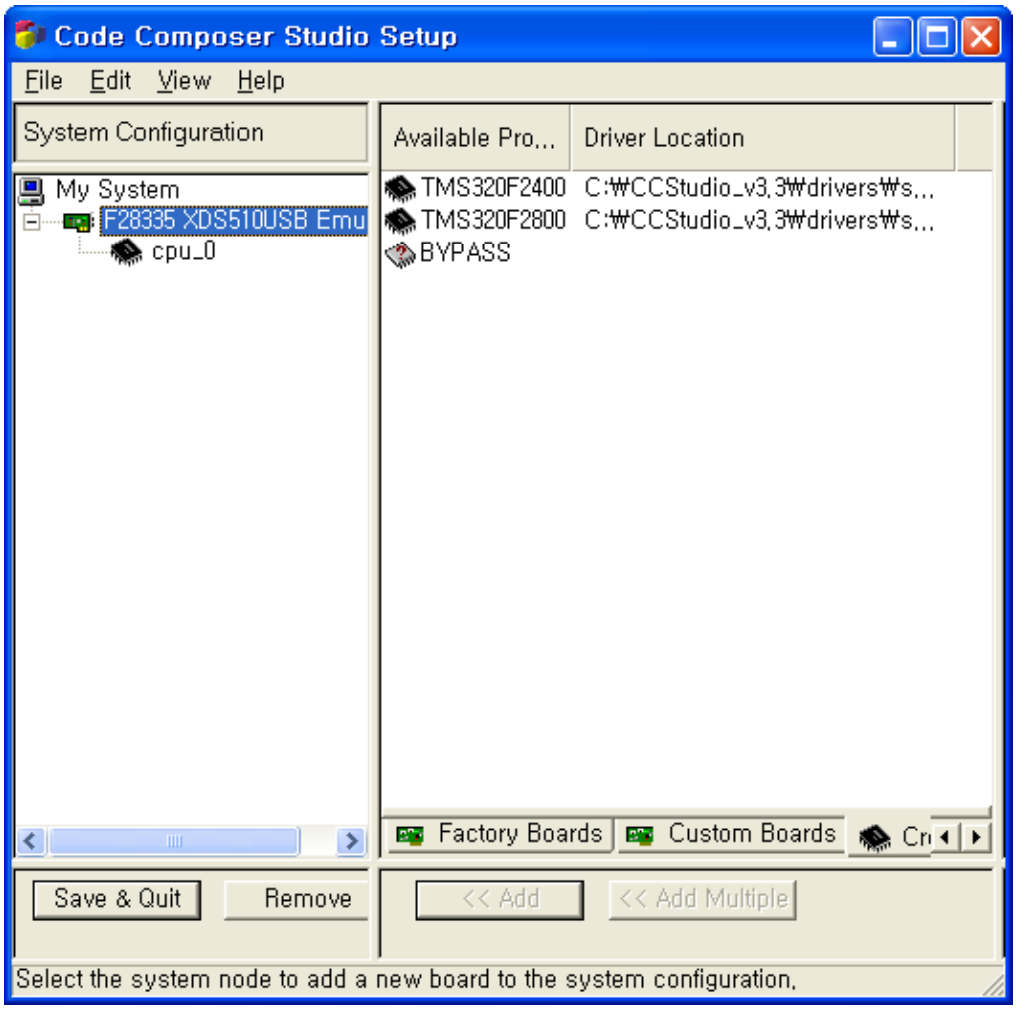
2. CCS3.3 폴더에 설치된 DSP BIOS버전을 확인후 최신버전(5.33.xx 이상)으로 업그레이드 한다.
   업그레이드된 BIOS버전을 Code Composer Manager에 등록 한다.



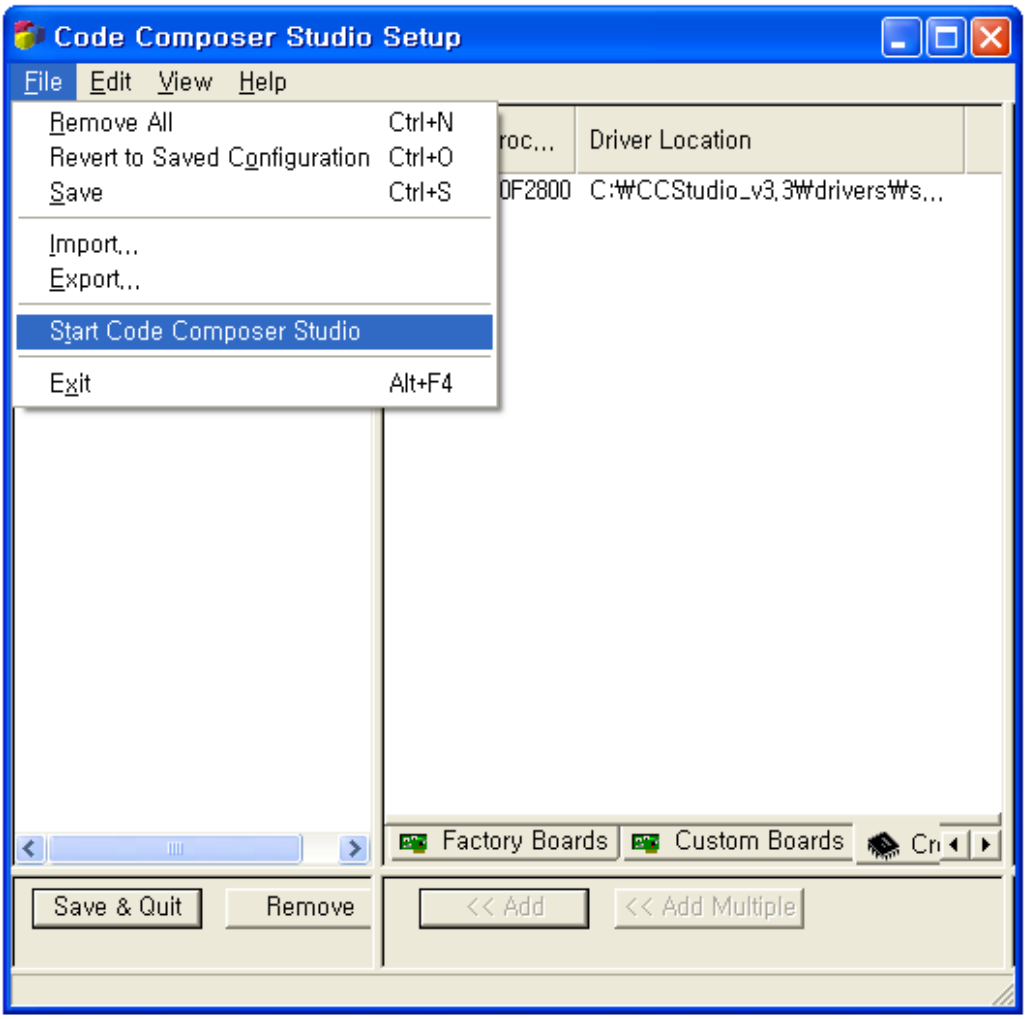
3. Setup CCStudio v3.3을 실행 합니다.

4. 프로그램 및 디버거에 사용할 장치를 선택 한후 Add버튼을 클릭 합니다.





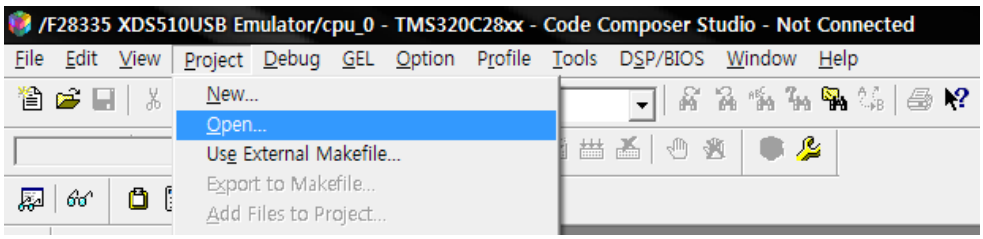
5. My System에서 F283335 XDS510USB Emu를 선택후 Start Code Composer Studio를 실행 합니다.

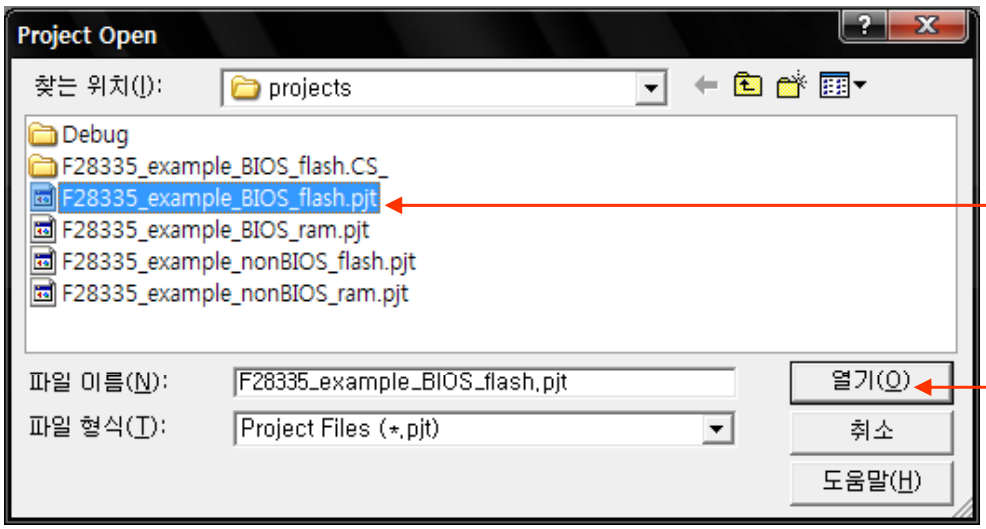


* CCS3.3 DSP/BIOS 예제 실행

1. Setup CCStudio v3.3 이나 CCSStudio3.3을 실행 합니다.

2. 아래와 같이 Project를 오픈 합니다.(Project->Open)

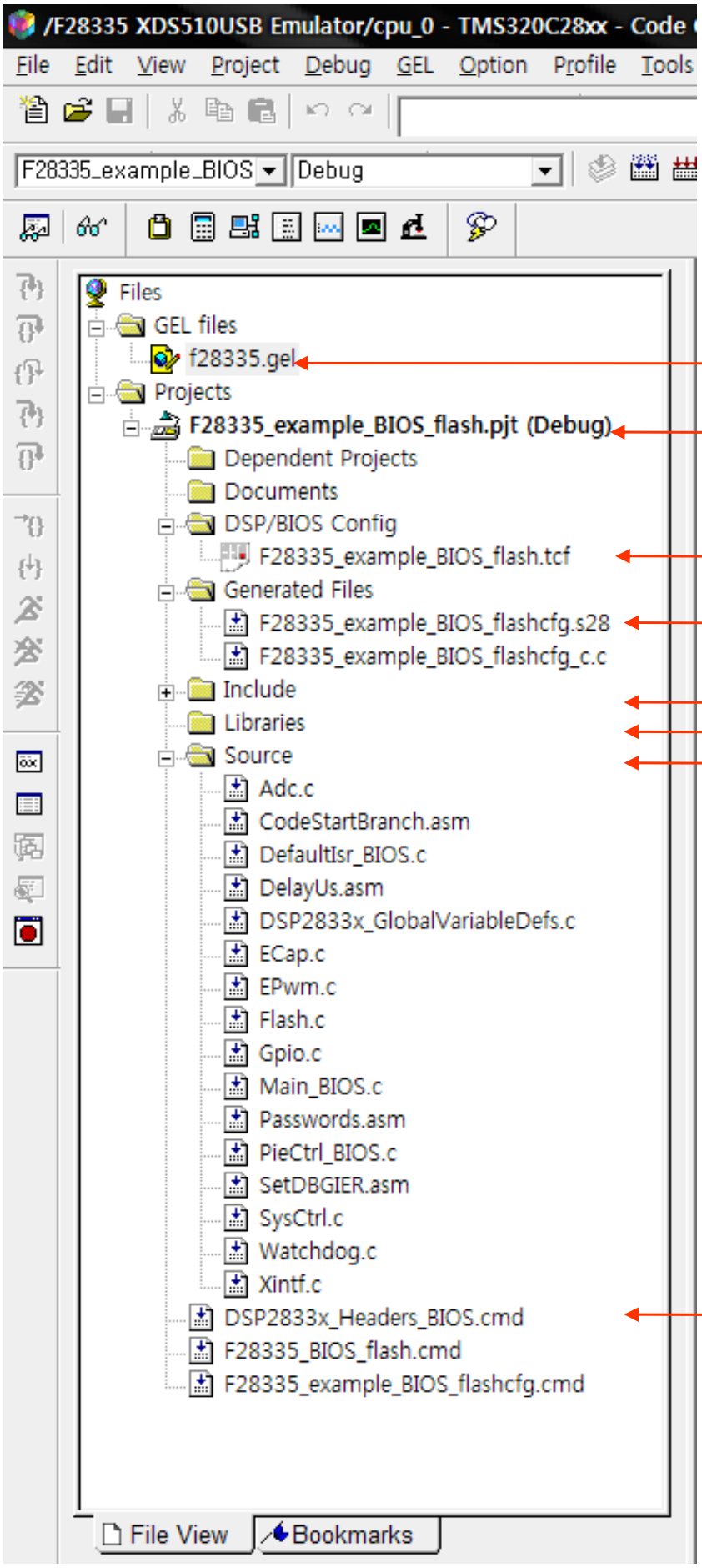




FLASH에서 실행되는 프로젝트

버튼 클릭

* File View 설명
  - 항목 추가시 : 오른쪽 마우스 클릭 후 Add File to Project
  - 항목 삭제시 : 파일명이나 프로젝트 명 선택후 Delete 키



CCS 사용 환경 스크립트

프로젝트명(여러 개의 프로젝트를 오픈한경우 오른쪽 마우스를 클릭후 Set as Active Project 를 이용하여 해당 프록젝트를 활성화 함)

DSP/BIOS 환경 설정 파일(더블클릭/ 오른쪽마우스 DSP/BIOS Config ->Text Edit)

DSP/BIOS 환경 설정 파일 에서 생성된 파일

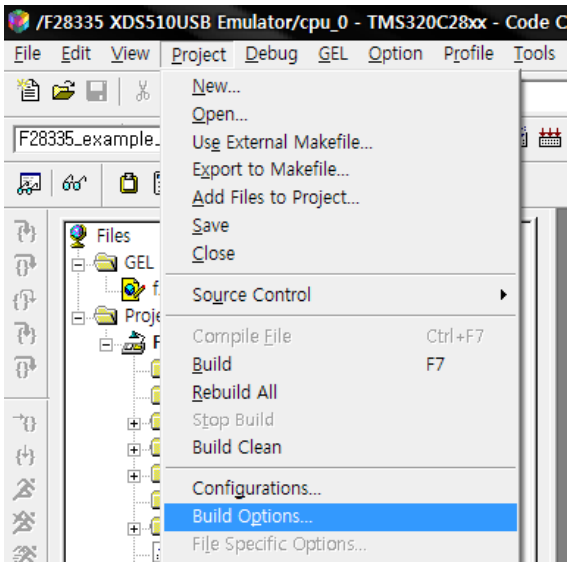
프로젝트 인쿠르드 파일

프로젝트 라이브러리 파일

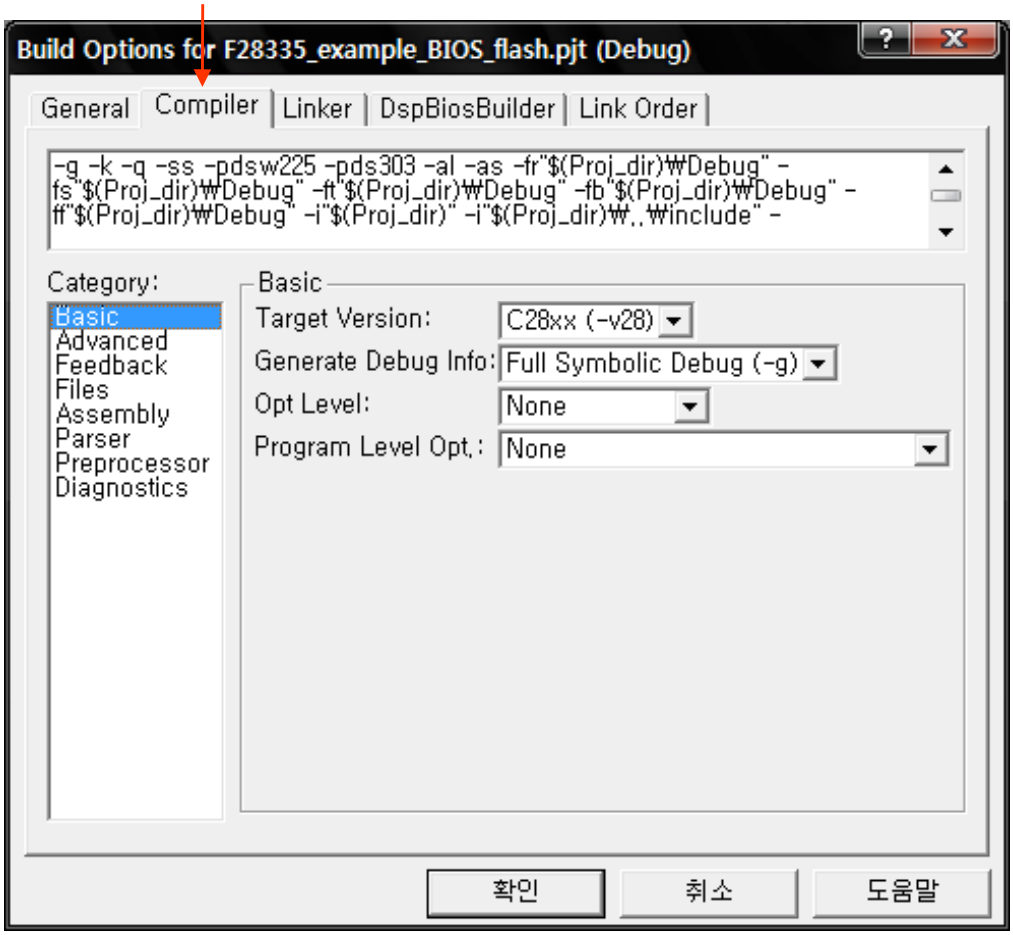
프로젝트 소스 파일

Linker Command File

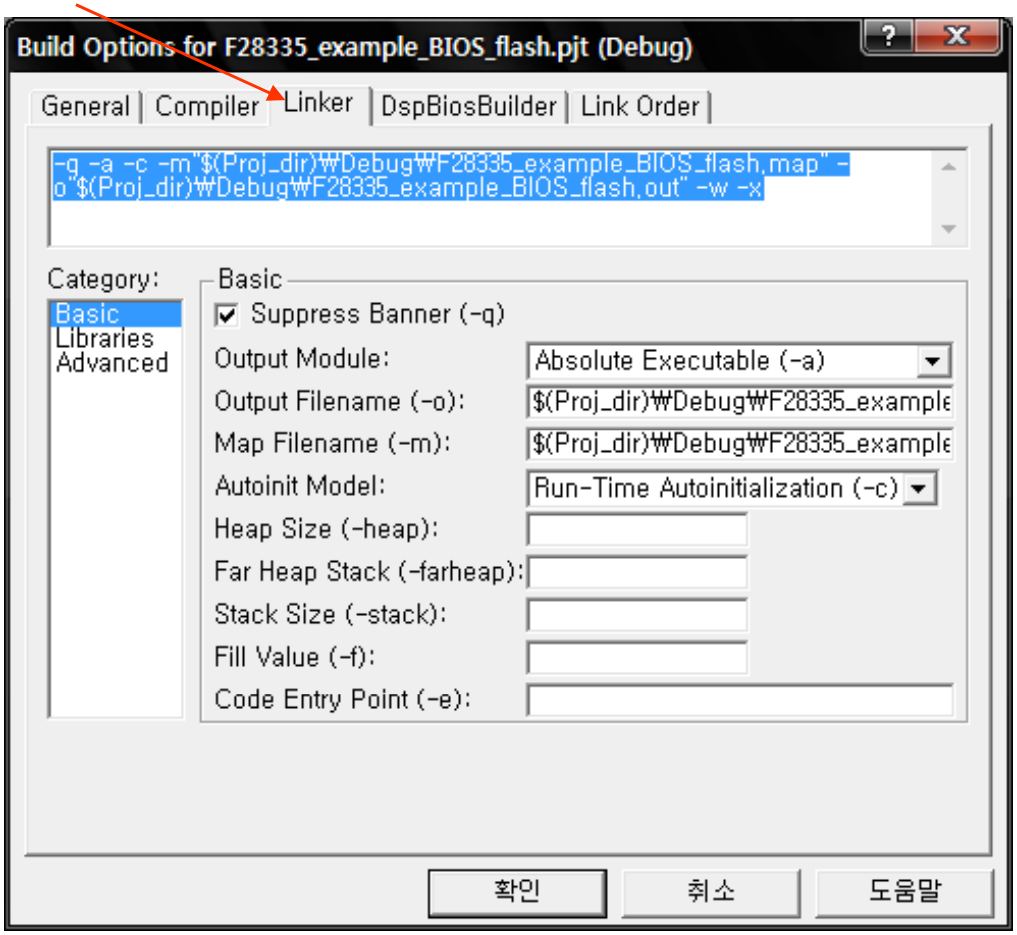
3. 프로젝트 옵션 설정(Project->Build Options)



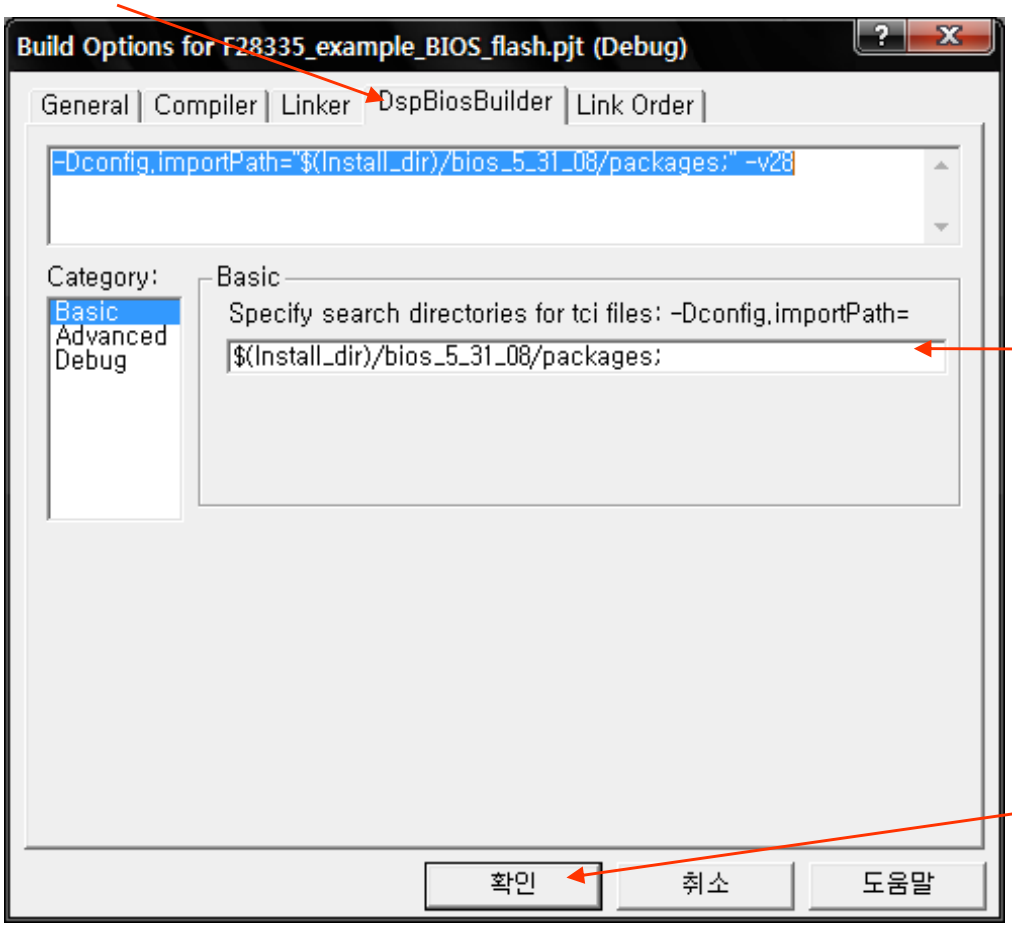
. 컴파일러 설정



. 링커 설정



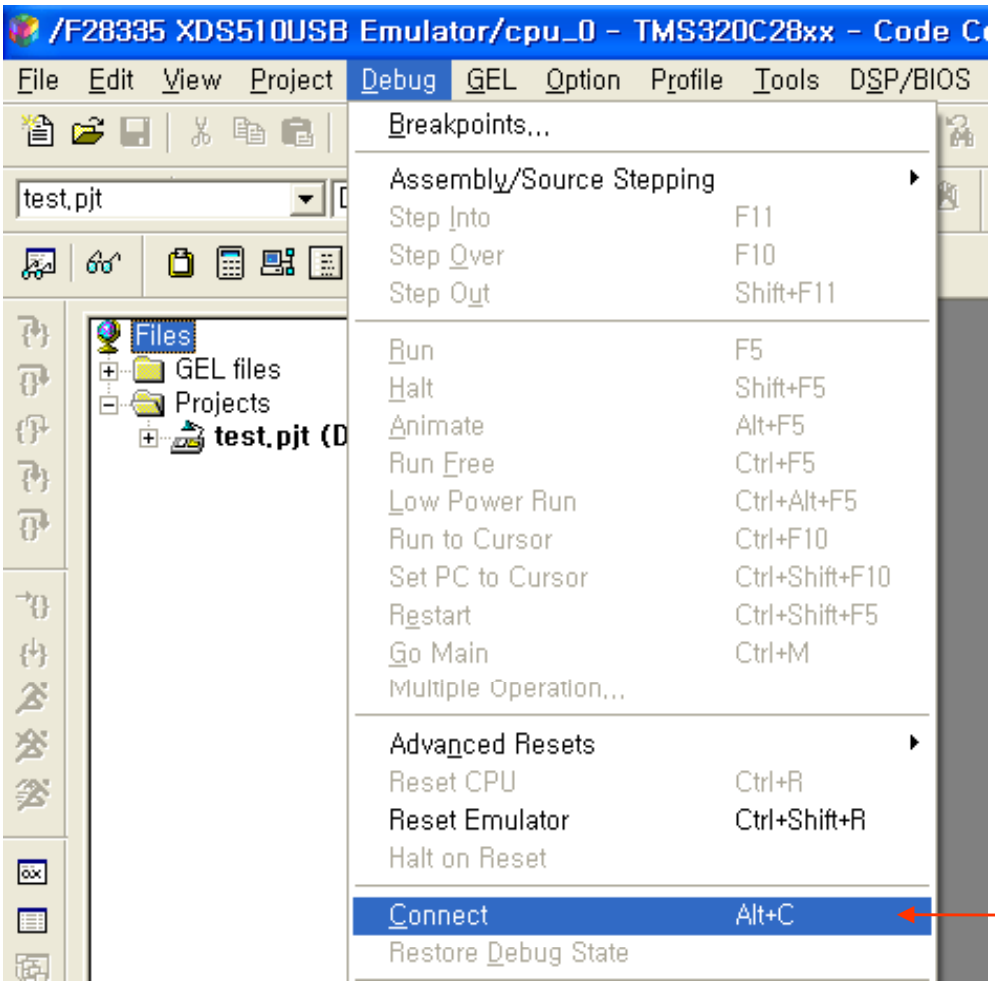
. DSP/BIOS 설정



CCS3.30이 설치된 폴더에서
DSP/BIOS 폴더 위치를 지정

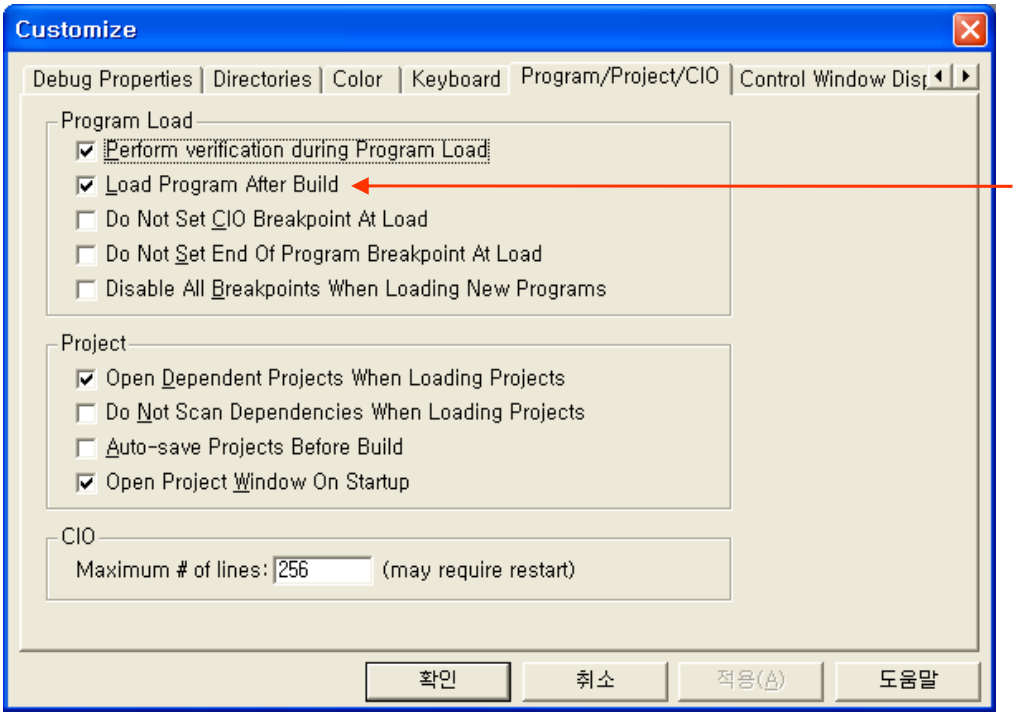
선택

4. JTAG 및 에뮬레이터를 연결 합니다.



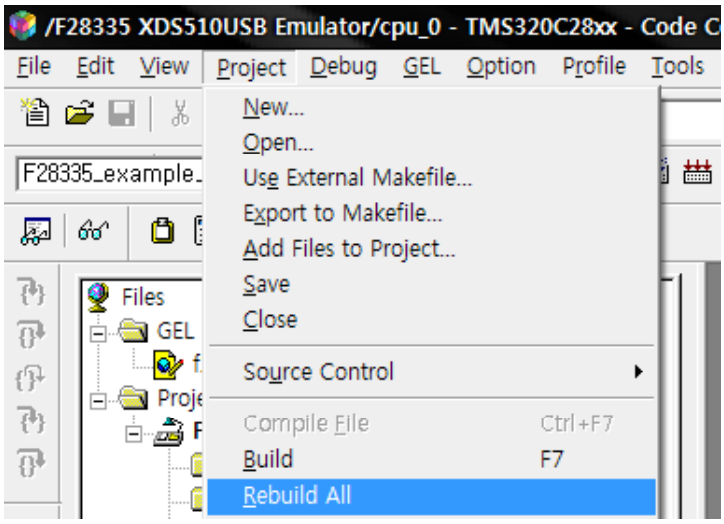
선택후
마우스 왼쪽 버튼 클릭

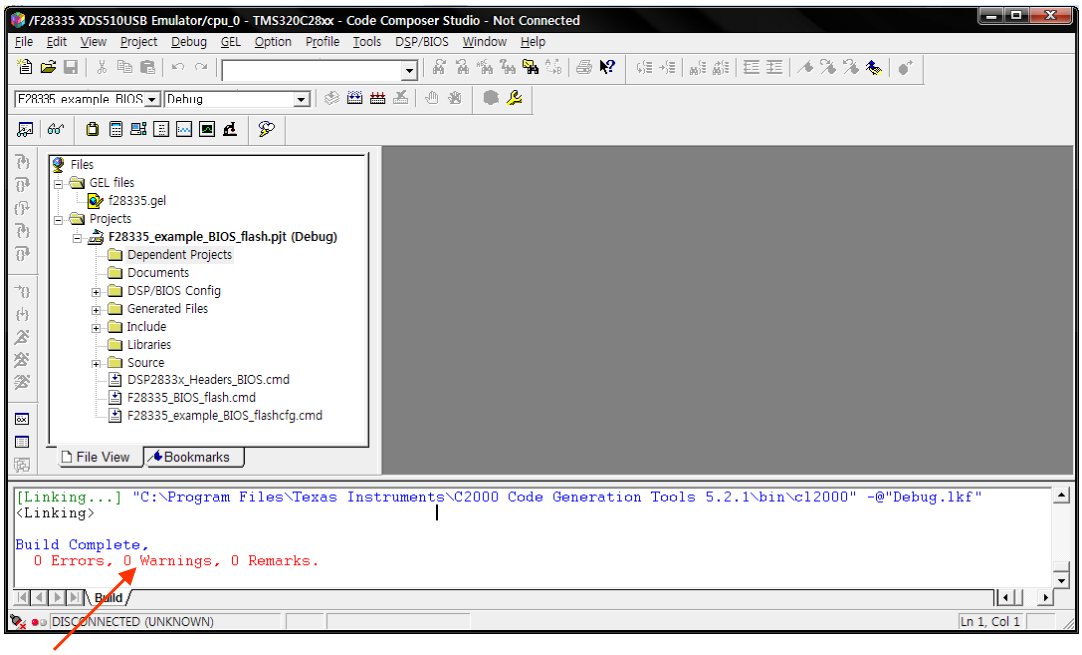
5. 내부램 으로 프로그램을 실행할 경우 아래와 같이 설정 합니다.(Option->Customize)



체크

6. 컴파일 하기(Project->Rebuild All)

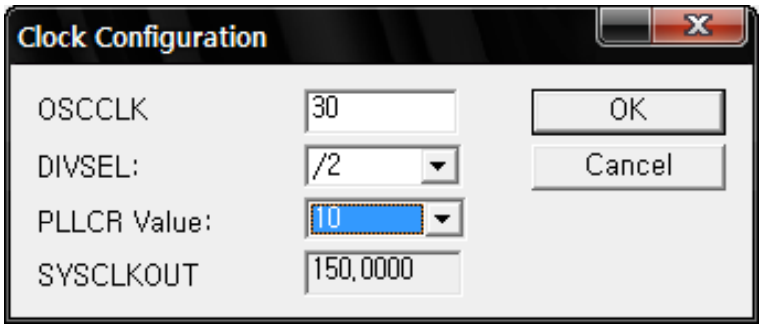




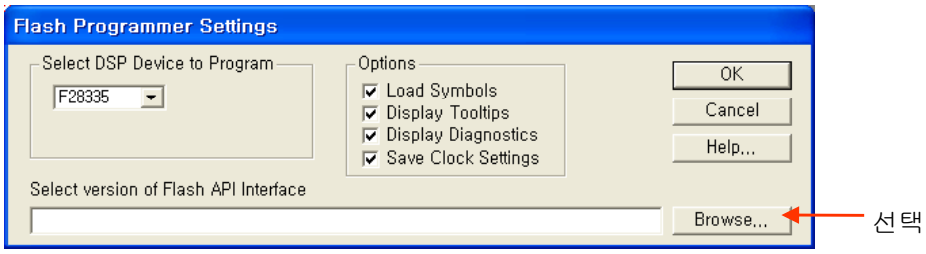
에러 확인

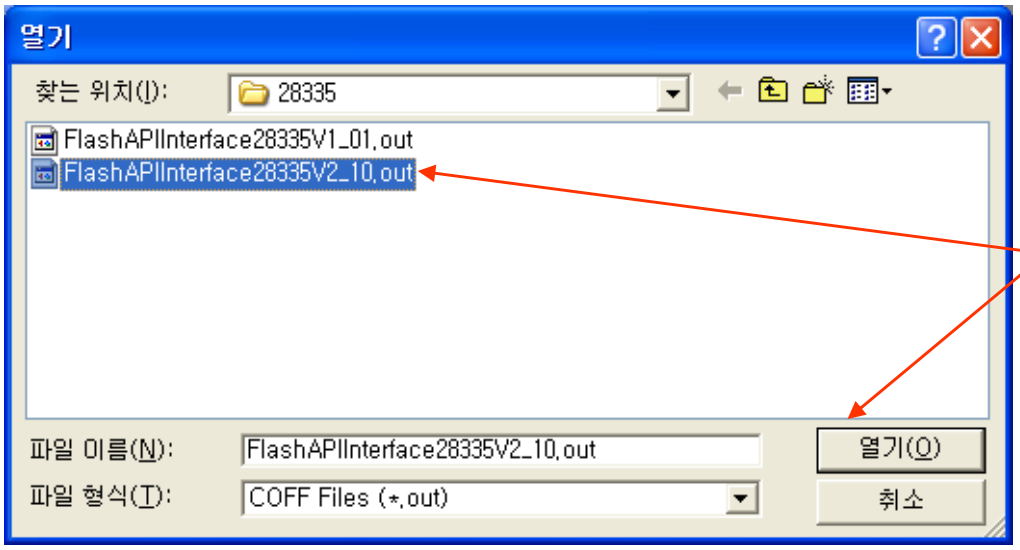
7. FLASH에 프로그램 하기

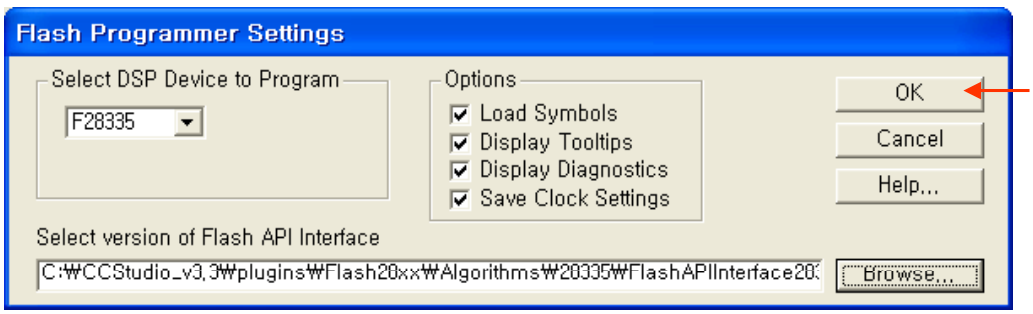
* 아래 CLOCK 설정 메뉴를 사용자에 맞게 설정 합니다.

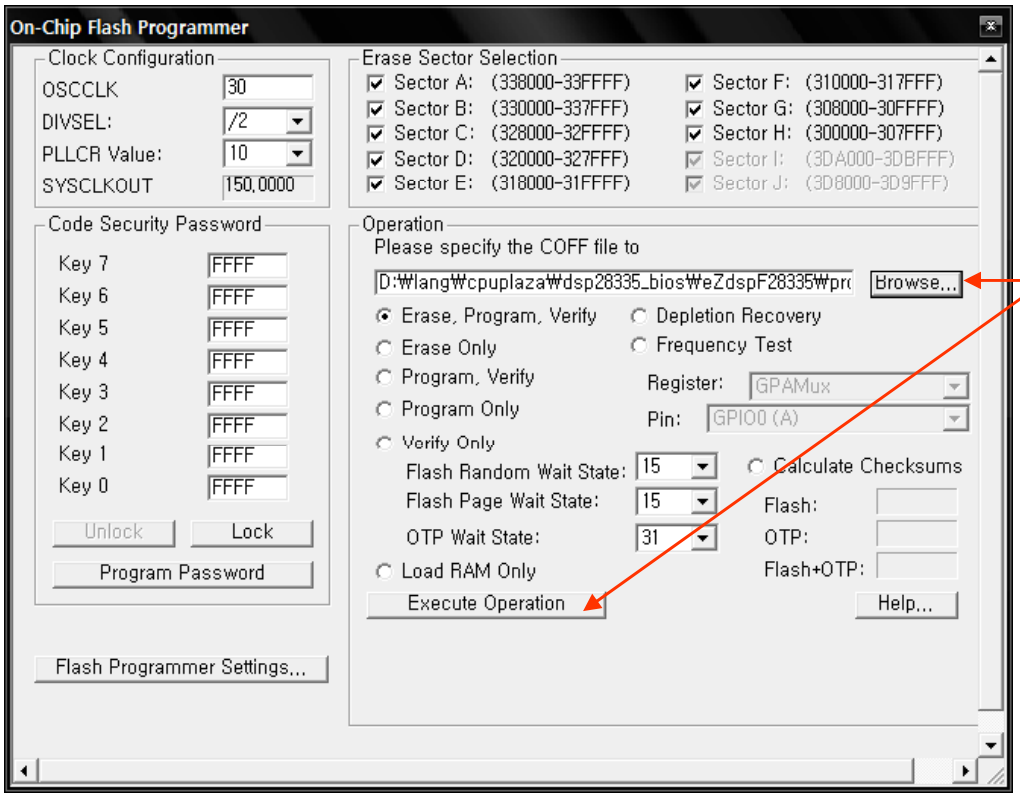


* API Interface 파일을 등록 합니다.



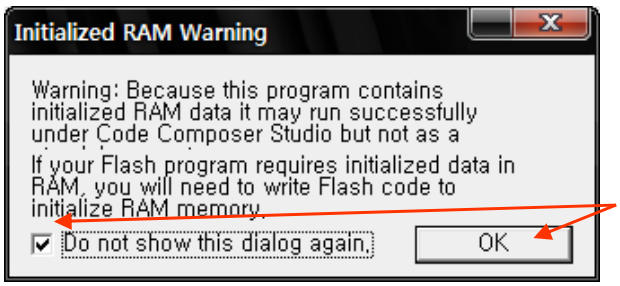




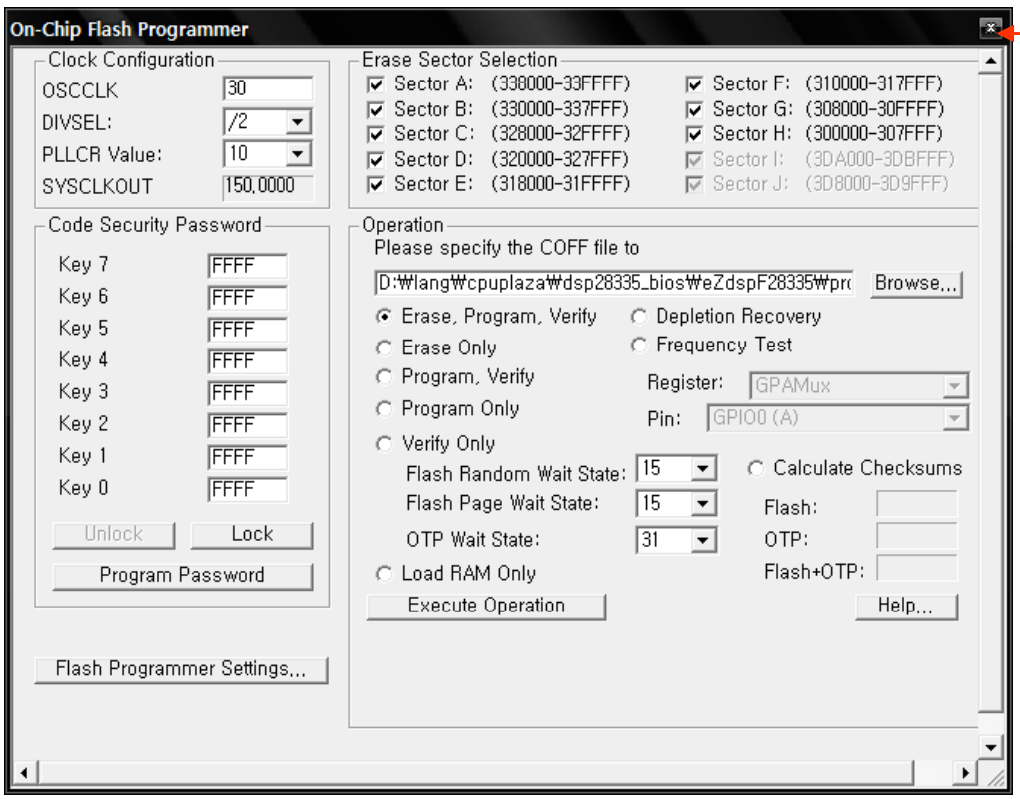


Browe.. 창에서
파일을 선택후 Excute
Operation탭을 실행
합니다.

* TI 실행 파일은 *.OUT로
현재 작업 디렉토리
..₩debug₩ 에 있습니다.

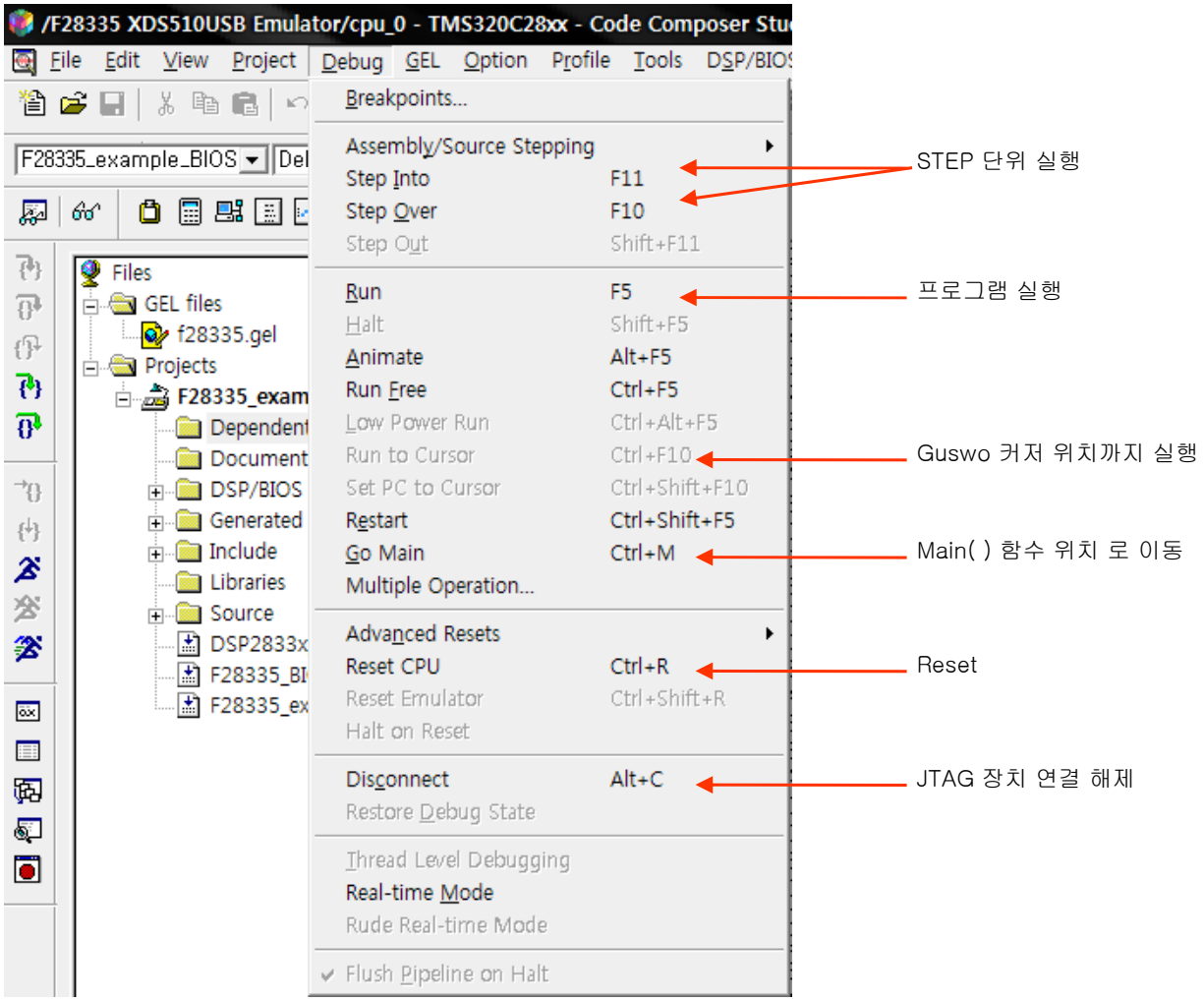


체크후 확인

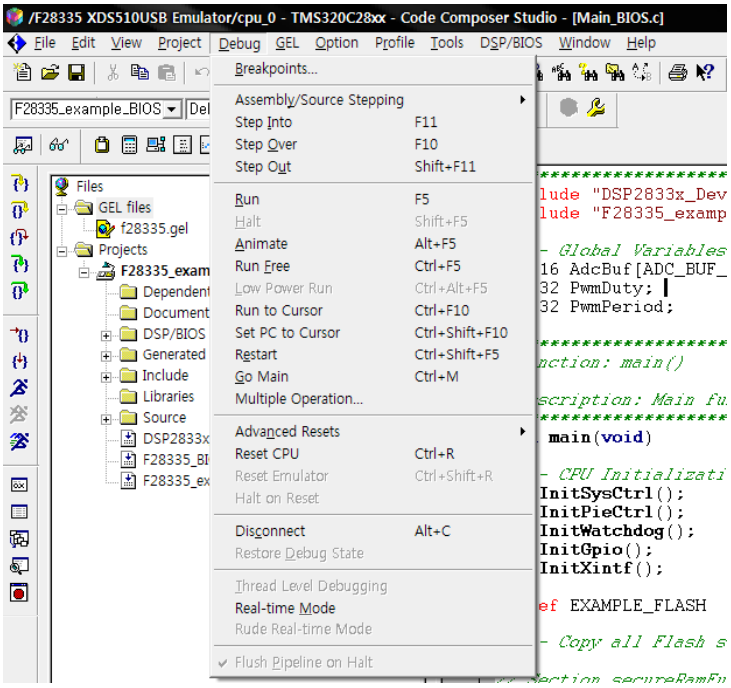


닫음

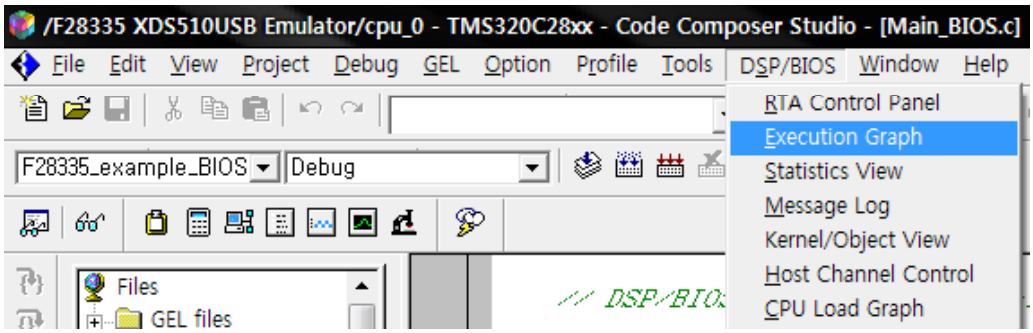
8. 프로그램을 로딩후 Debug 탭에서 Debug기능을 선택 실행 합니다.

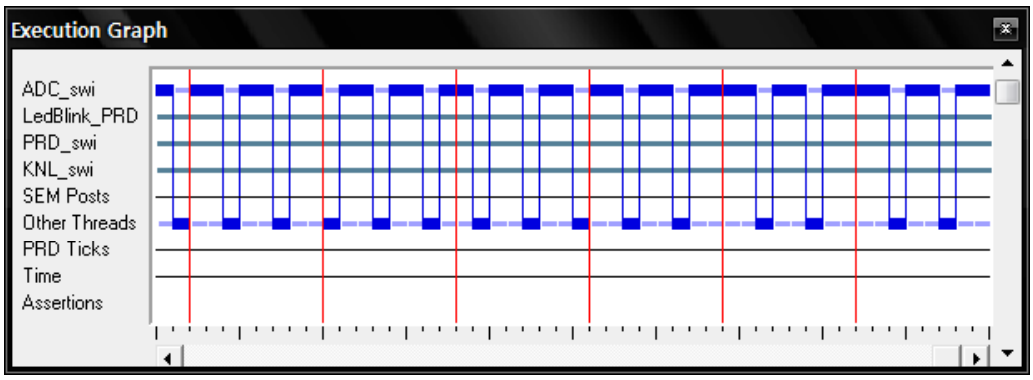


9. 실행
   - Debug 창 에서 Run(F5)을 클릭 하면 바로 실행
   - Debug 창 에서 Go Main후 Run(F5)이나 Debug 메뉴 실행

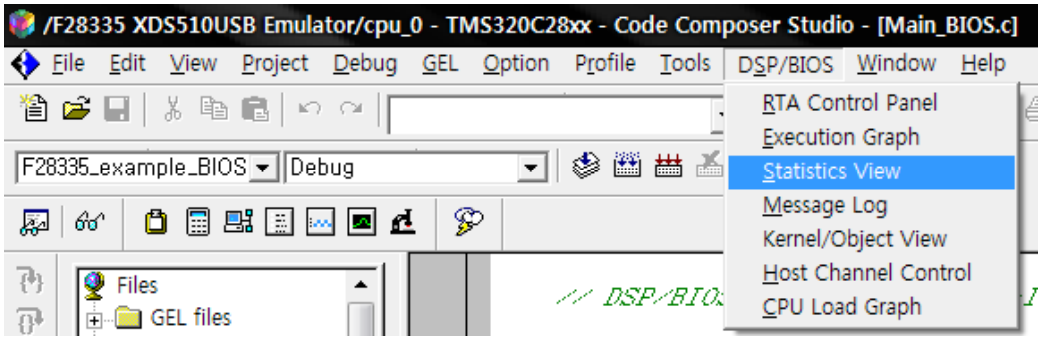


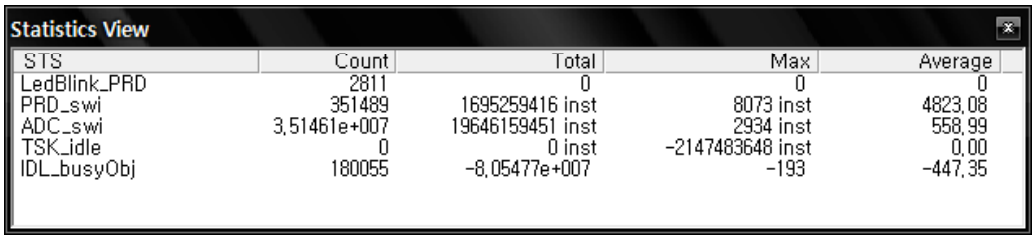
10. DSP BIOS Software Logic 및 Debug Event Timing(DSP/BIOS->Execution)





11. DSP BIOS Profile Routines w/o halting the CPU(DSP/BIOS->Statistics View)

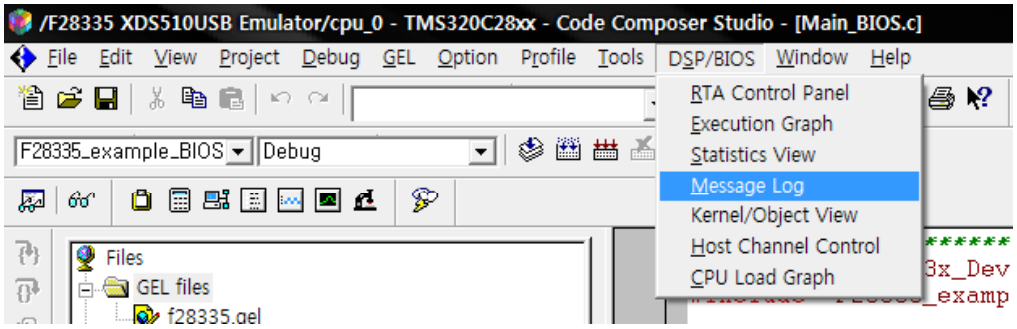


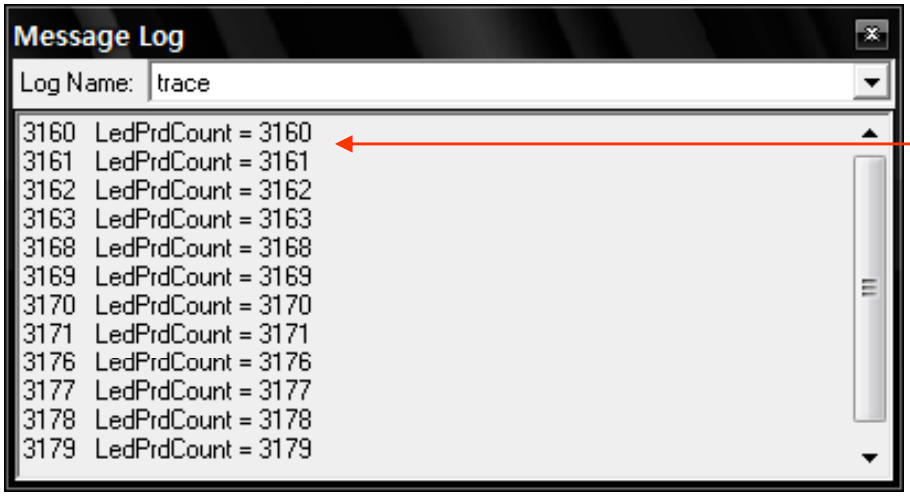


Statistics View

STS	Count	Total	Max	Average
LedBlink_PRD	2811	0	0	0
PRD_swi	351489	1695259416 inst	8073 inst	4823,08
ADC_swi	3,51461e+007	19646159451 inst	2934 inst	558,99
TSK_idle	0	0 inst	-2147483648 inst	0,00
IDL_busyObj	180055	-8,05477e+007	-193	-447,35

12. DSP BIOS Message Log(DSP/BIOS->Message Log)
예제 프로그램에서 LOG_printf()로 출력되는 멧세지를 확인 할수 있다.

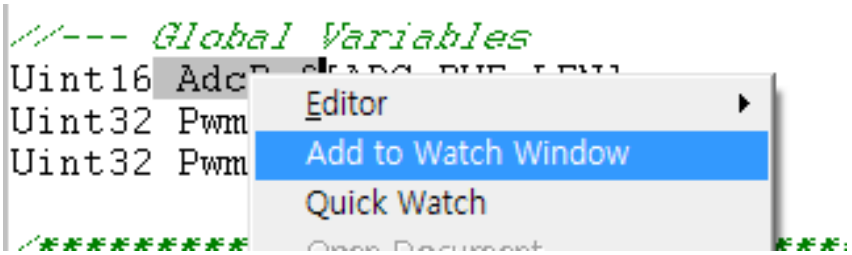


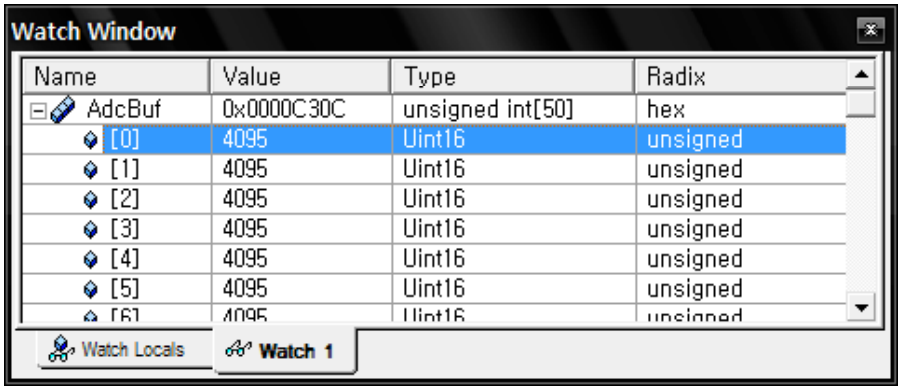


Led_Blink() 함수에서 표시

13. 변수 확인

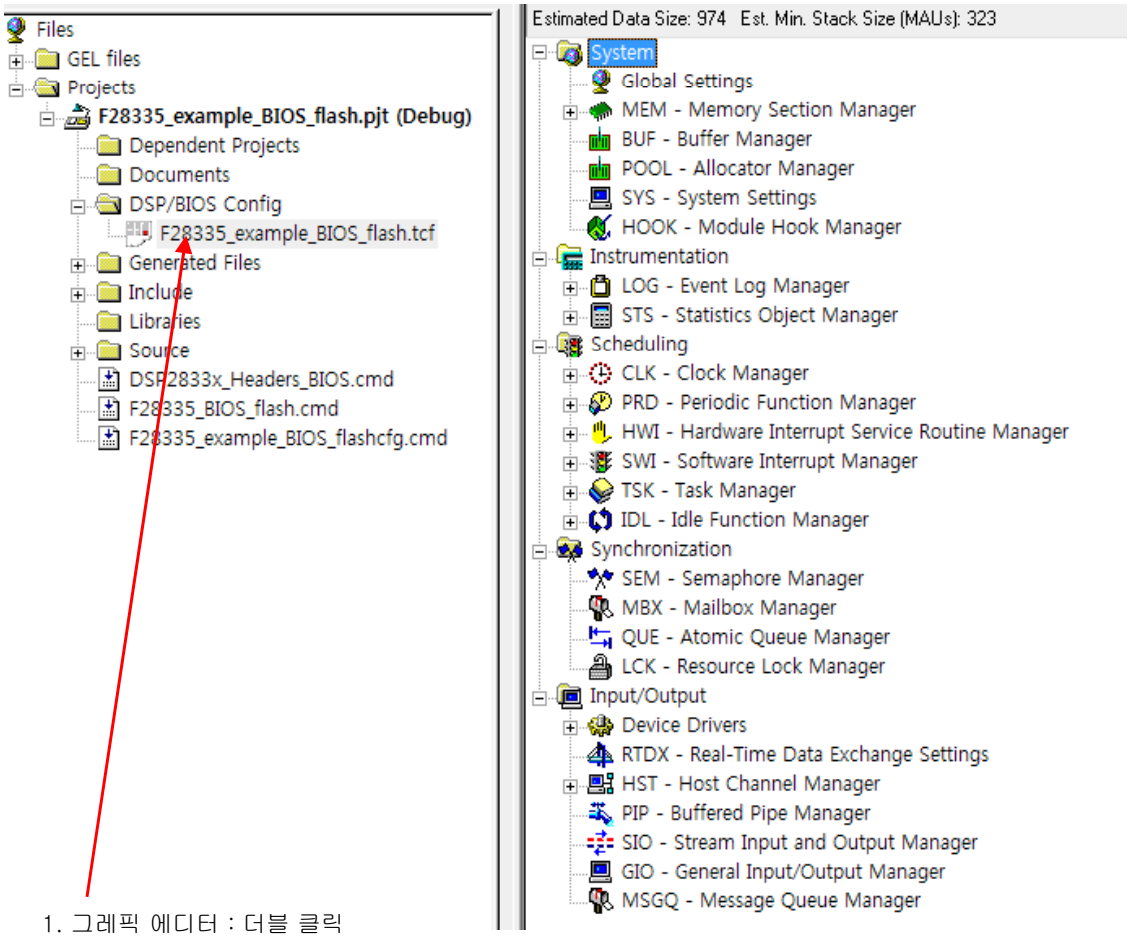
확인하고자 하는 변수를 지정 후 오른쪽 마우스를 클릭 Add Watch Window 창을 연다.





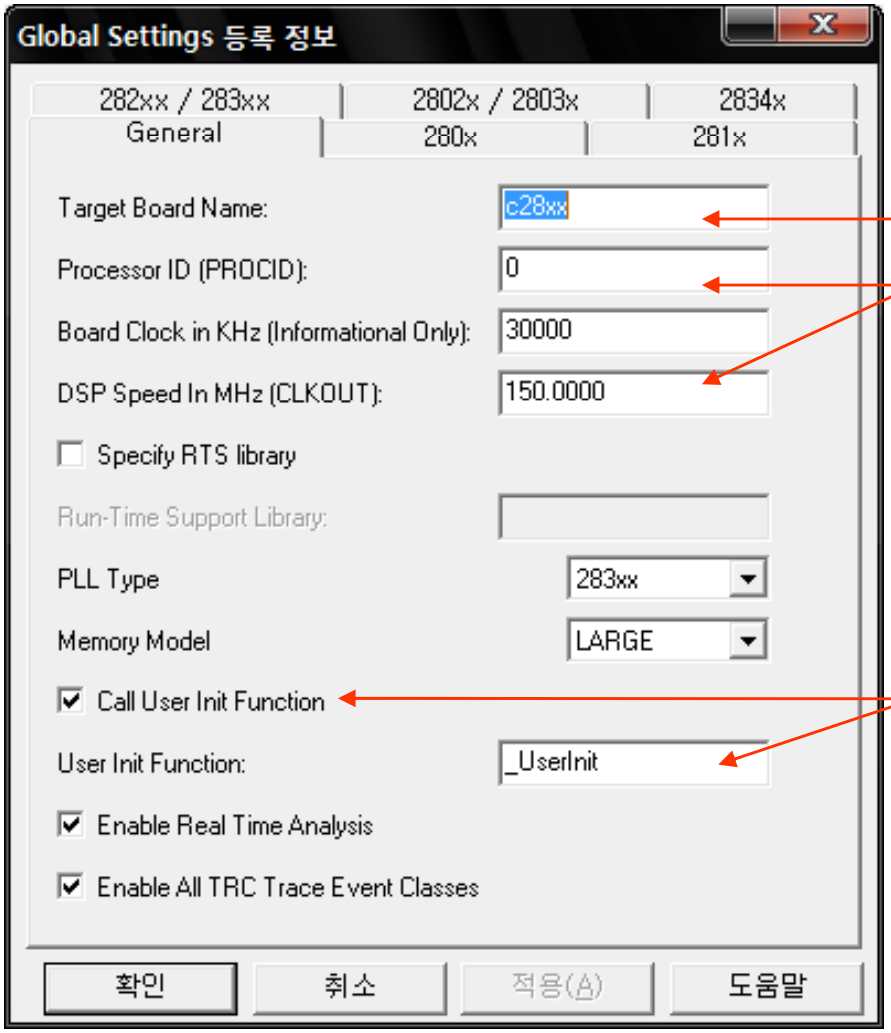
* DSP/BIOS 기본

1. DSP/BIOS 설정 파일(*.tcf) 설명
   DSP/BIOS는 일반 C 프로그램 과 다른 점은 BIOS Configuration File(*.tcf)이 추가 된다.
   이 파일은 CCS에서 그레픽/TEXT 에디터 로 변경 할수 있으며 주 내용은 아래와 같다.
   - System Setup Tools
   - Real –Time Analysis Tools
   - Real –Time Scheduler
   - Synchronization
   - Real-Time I/O
   - *.tcf 파일은 컴파일시 *cfg.cmd, *cfg.sxx, *cfg.hxx, *cfg_c.c, *cfg.h, *.cdb 파일이 생성된다.



1. 그레픽 에디터 : 더블 클릭
2. 오른쪽 마우스 ->DSP/BIOS Config ->Text Edit

* System->Global Setting(오른쪽 마우스 클릭 후 Properties선택)
  - 사용할 CPU 클럭 및 기본 정보를 설정 한다.



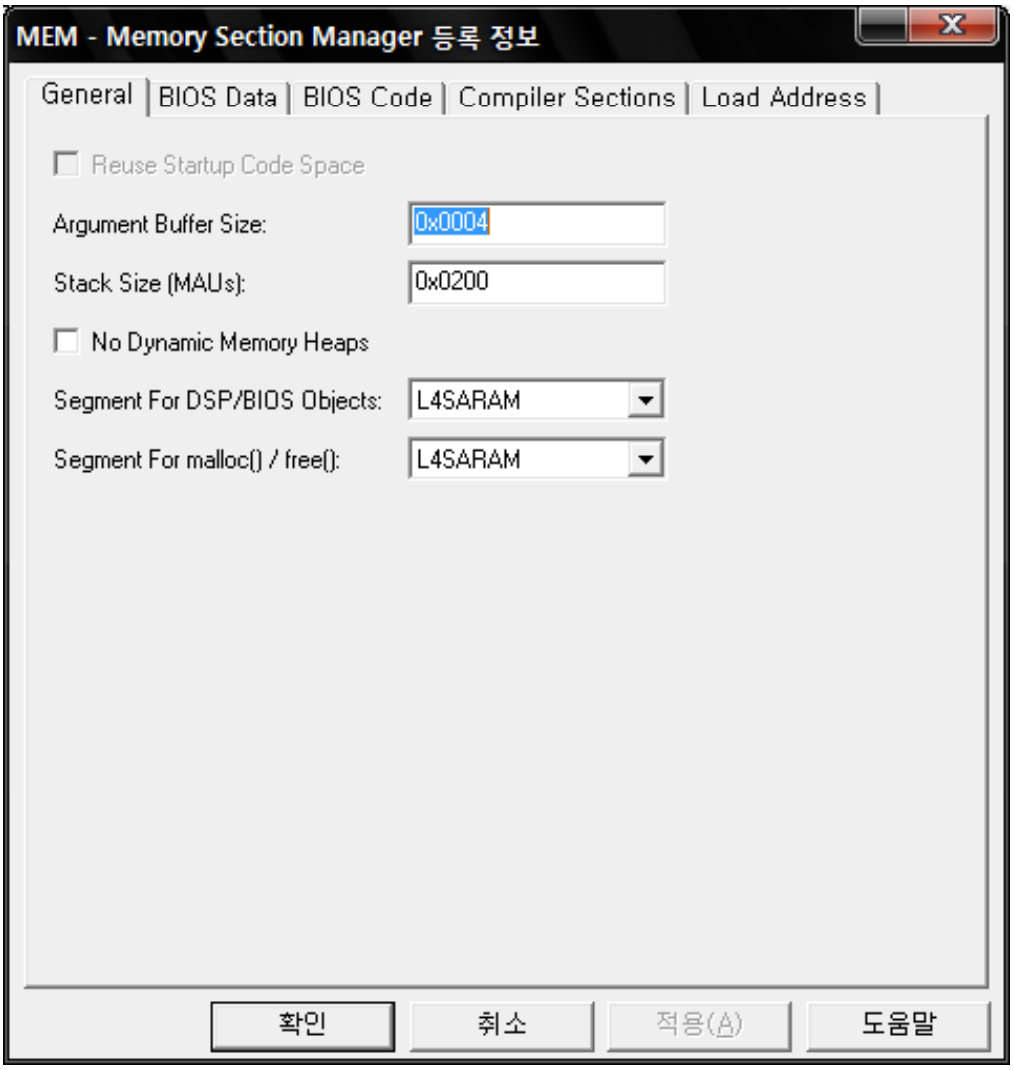
CPU Type

CPU 클럭 정의

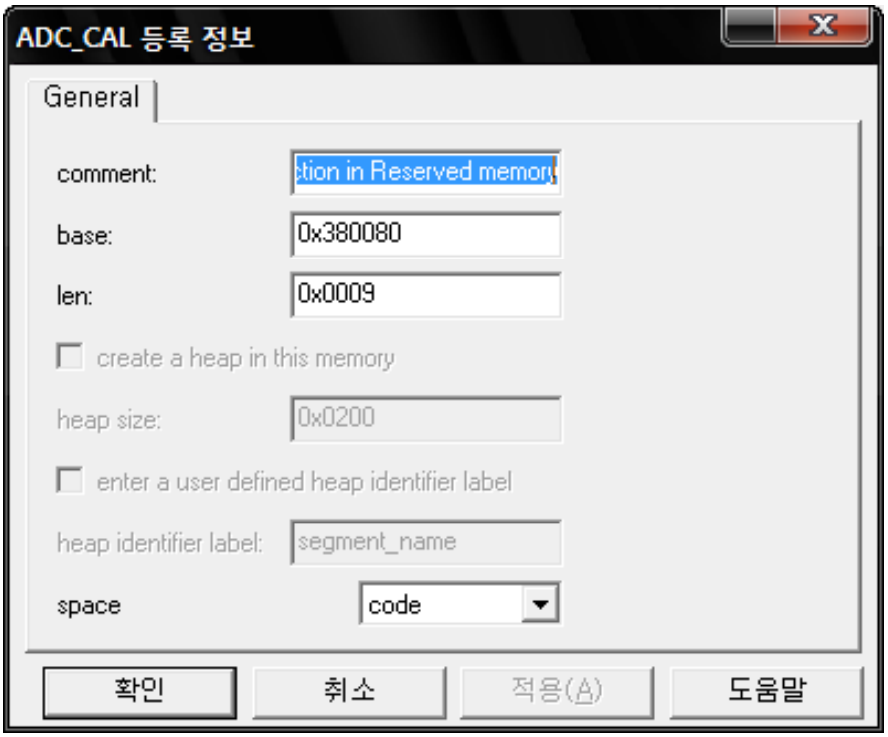
체크후 수행할 함수명 앞에 _를 붙여 정의 한다.

* 시스템 리셋후 호출 되는 사용자가 작성한 함수 c_int_00함수 초기부분에서 호출 된다.
  보통 BIOS 초기화 함수 나 사용자 초기 필요 함수를 추가 할수 있다

* System->Memory Section Manager
(오른쪽 마우스 클릭 후 Properties선택, 또는 메모리 항목 추가시 Inser MEM 선택)
- 프로그램 에서 사용 되는 메모리를 정의 한다.

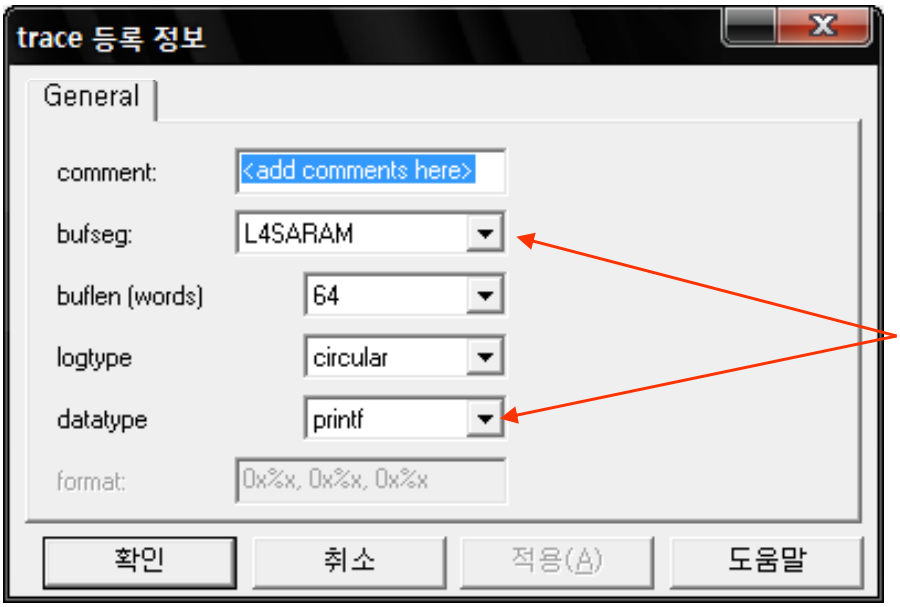


* System->Memory Section Manager->ADC_CAL(메모리명) (오른쪽 마우스 클릭 후 Properties선택)
- 각 메모리별 주소와 사이즈 Location 위치를 지정 한다.



* Instrumentation->LOG-Event Log Manager (오른쪽 마우스 클릭후 Inser LOG를 선택 항목을 추가)

* Instrumentation- >LOG-Event Log Manager >trace(오른쪽 마우스 클릭 후 Properties선택)
  - LOG_printf()를 통해 CCS3.3 에서 메시지를 확인 할수 있다.



LOG_printf(&trace,) 호출시
printf() 연결

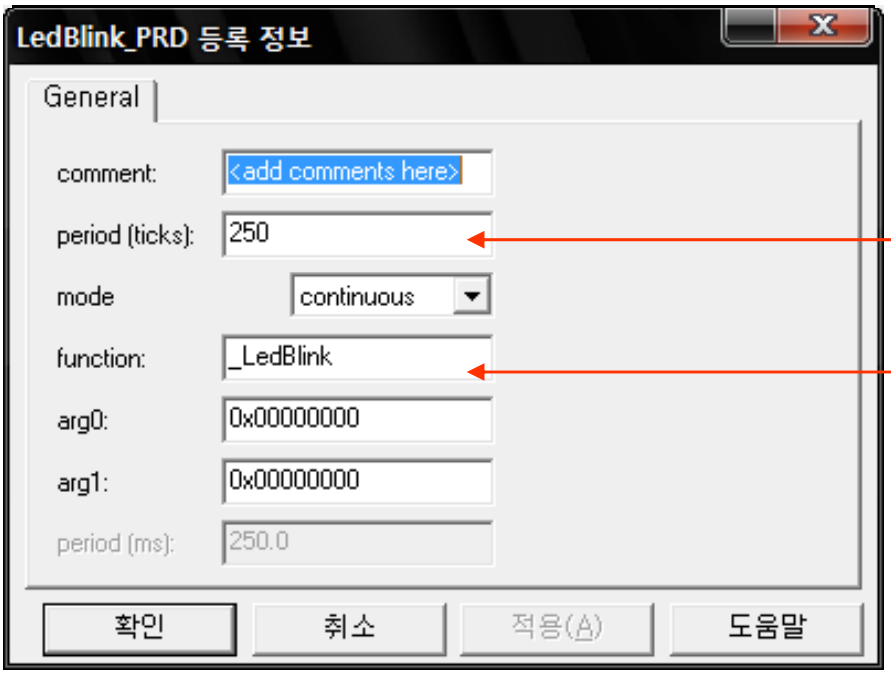
* Scheduling->CLK-Clock Manager(오른쪽 마우스 클릭후 Inser CLK를 선택 항목을 추가)

* Scheduling->CLK-Clock Manager(오른쪽 마우스 클릭 후 Properties선택)
  - 스케쥴링에 사용 되는 기본 클럭을 설정 한다.

1000us->1ms

* Scheduling->PRD-Peroiodic Function Manager (오른쪽 마우스 클릭후 Inser PRD를 선택 항목을 추가)

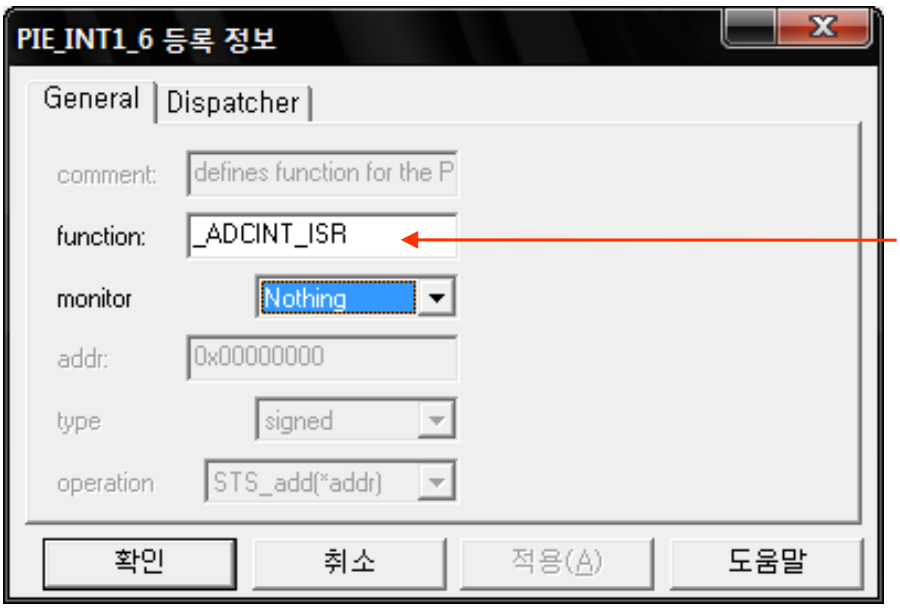
* Scheduling->PRD-Peroiodic Function Manager->LedBlink_PRD(오른쪽 마우스 클릭후 Properties선택)
  - LedBlink함수를 호출하는 시간을 정의 하고 있다.
    (작성시 이 함수 안에서 오래머무르는 기능은 제외 시키길 권장 합니다.
    :: 일반 C의 Timer 인터럽트라 생각 하시면 됨)



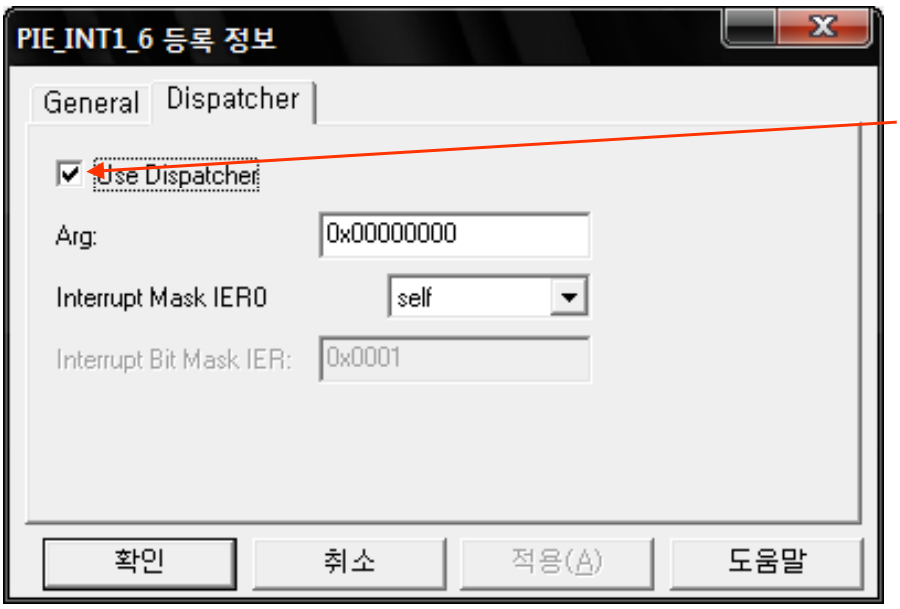
CLK->Clock Manager에서 설정한 시간(1ms) * 250 = 250ms 250Tick이라고 함

호출 되는 함수 명 앞에 _를 붙임

* Scheduling->HWI->PIE Interrupts->PIE_INT1_6(오른쪽 마우스 클릭 후 Properties)
  - 시스템에서 사용되는 인터럽트 함수를 등록 한다.
    특이한 사항은 Disspatecher항목이 있어 하드웨어 인터럽트에서 발생한 인터럽트를 바로 처리 하지 않고 SWI(소프트웨어 인터럽트) 인터럽트를 처리하는 함수(TASK)를 미리 생성해 그곳에서 인터럽트를 처리 하도록 하는 기능이 있다. 이때 HWI루틴에서는 SWI_post( ) 를 이용 한다.
    일반적으로 인터럽트가 발생하면 바로처리 하는데 반해 SWI가 있는 이유는 BIOS에서 다른 TASK를 처리할 시간이 모자라므로 인터럽트도 TASK화 하여 관리 하기 위함이다.
    물론 SWI인터럽트를 사용 하지 않을 수도 있다.



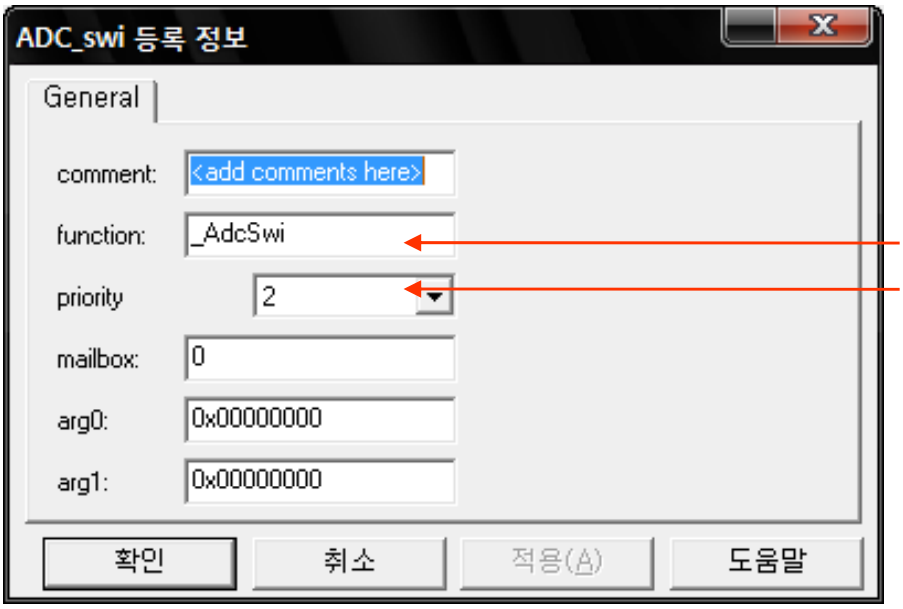
호출 되는 함수 명 앞에 _를 붙임



체크 하지 않으면 HWI, 체크하면 SWI

* Scheduling->SWI
  (오른쪽 마우스 클릭 후 Properties선택 또는 오른쪽 마우스 클릭후 Inser SWI를 선택 항목을 추가 할수 있다)

* Scheduling->SWI ->ADC_SWI(오른쪽 마우스 클릭 후 Properties선택)



호출 되는 함수 명 앞에 _를 붙임

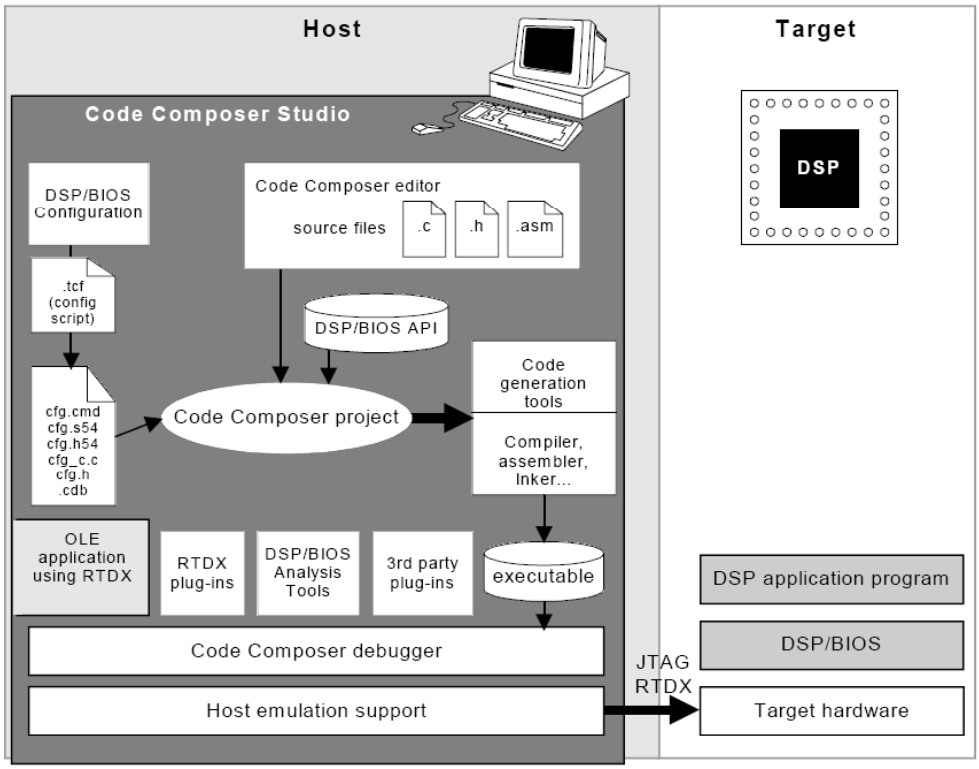
호출 되는 함수 우선순위 레벨

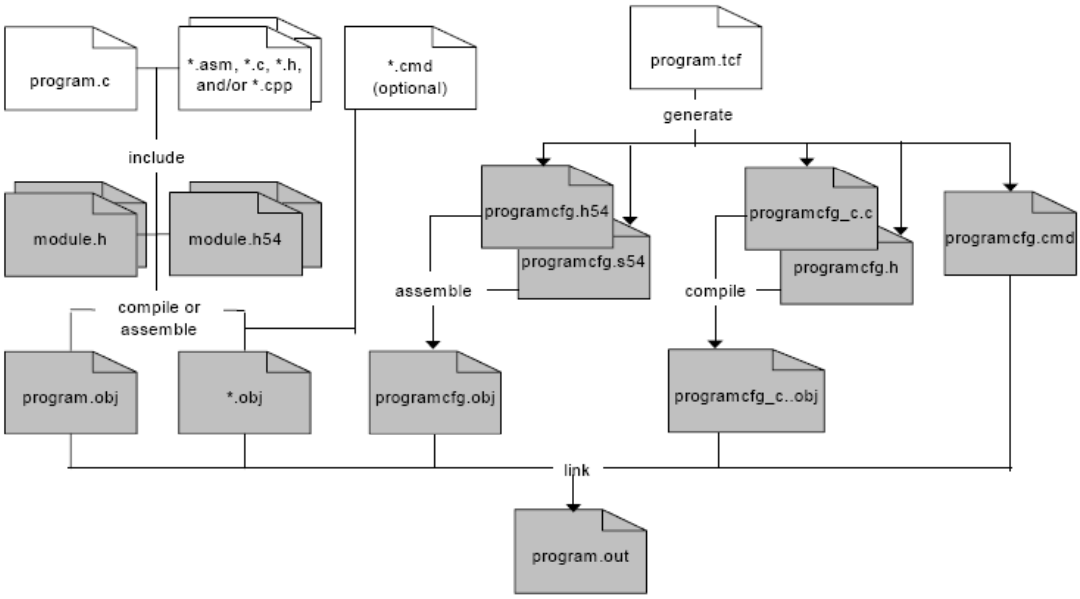
* Scheduling->TASK(오른쪽 마우스 클릭후 Inser TASK를 선택 항목을 추가)
  - 하나의 별도 프로그램 이며, BIOS의 스케쥴러에 의해 관리되며 실행 된다.

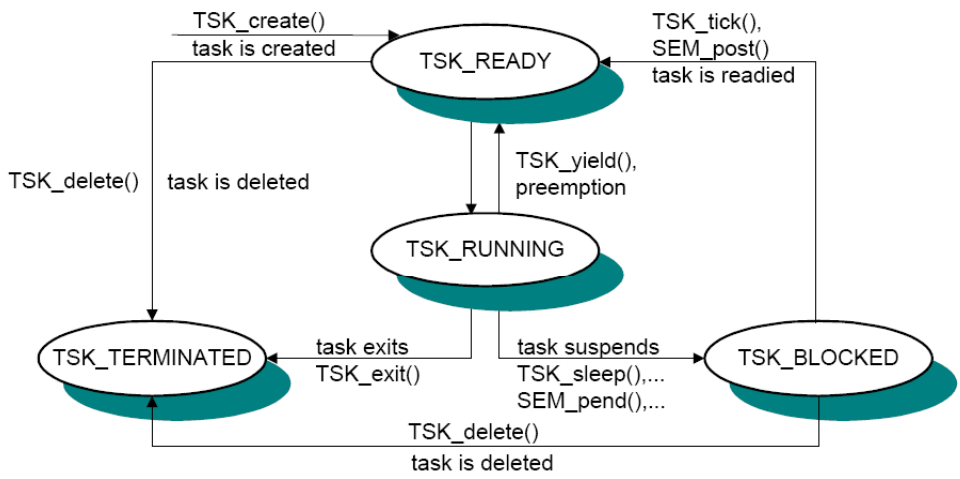
* Synchronization-SEM(오른쪽 마우스 클릭후 Inser SEM을 선택 항목을 추가)
  - TASK 와 TASK간 동기나 자료를 전달하기 위해 사용.

* Synchronization-MBX(오른쪽 마우스 클릭후 Inser MBX을 선택 항목을 추가)
  - TASK 와 TASK간 자료를 전달하기 위해 사용.

* Synchronization-QUE(오른쪽 마우스 클릭후 Inser QUE을 선택 항목을 추가)
  - TASK 와 TASK간 자료를 전달하기 위해 사용.


Host
Target
Code Composer Studio
DSP/BIOS Configuration
Code Composer editor
source files
.c
.h
.asm
.tcf (config script)
DSP/BIOS API
cfg.cmd cfg.s54 cfg.h54 cfg_c.c cfg.h .cdb
Code Composer project
Code generation tools
Compiler, assembler, linker...
OLE application using RTDX
RTDX plug-ins
DSP/BIOS Analysis Tools
3rd party plug-ins
executable
Code Composer debugger
Host emulation support
JTAG RTDX
DSP
DSP application program
DSP/BIOS
Target hardware


program.c
*.asm, *.c, *.h, and/or *.cpp
*.cmd (optional)
program.tcf
generate
include
module.h
module.h54
programcfg.h54
programcfg.s54
programcfg_c.c
programcfg.h
programcfg.cmd
assemble
compile
compile or assemble
program.obj
*.obj
programcfg.obj
programcfg_c..obj
link
program.out


TSK_create()
task is created
TSK_READY
TSK_tick(),
SEM_post()
task is readied
TSK_delete()
task is deleted
TSK_yield(),
preemption
TSK_RUNNING
TSK_TERMINATED
task exits
TSK_exit()
task suspends
TSK_sleep(),...
SEM_pend(),...
TSK_BLOCKED
TSK_delete()
task is deleted