# 1. CCS3.3  DSP/BIOS SWI(Sofyware Interrupt Manager) 생성

1. 디렉토리 구성

..₩cmd : Linker 컴맨드 파일

..DSP2833x_headers : Chip관련 헤더 파일 및 헤더용 Linker  컴맨드 파일

..₩include : 사용자 인쿠르드 파일

..₩testprj_3 : 사용자 프로젝트 파일 및 실행 파일(.HEX)

..₩testsrc_3 :  사용자 소스 파일

2. 디렉토리 설명

다른 디렉토리는 기존 Task 생성 예제 파일을 사용 하시고 testprj_3, testsrc_3를 복사 후 include 에 있는 *.h 파일을 복사해서 사용 하시면 됩니다.

3. Setup CCStudio v3.3을 실행 합니다.
프로젝트 Open 사용법은 Task  생성 예제를 참조 하시기 바랍니다.

4. 소스코드 설명(Main_Bios.c)

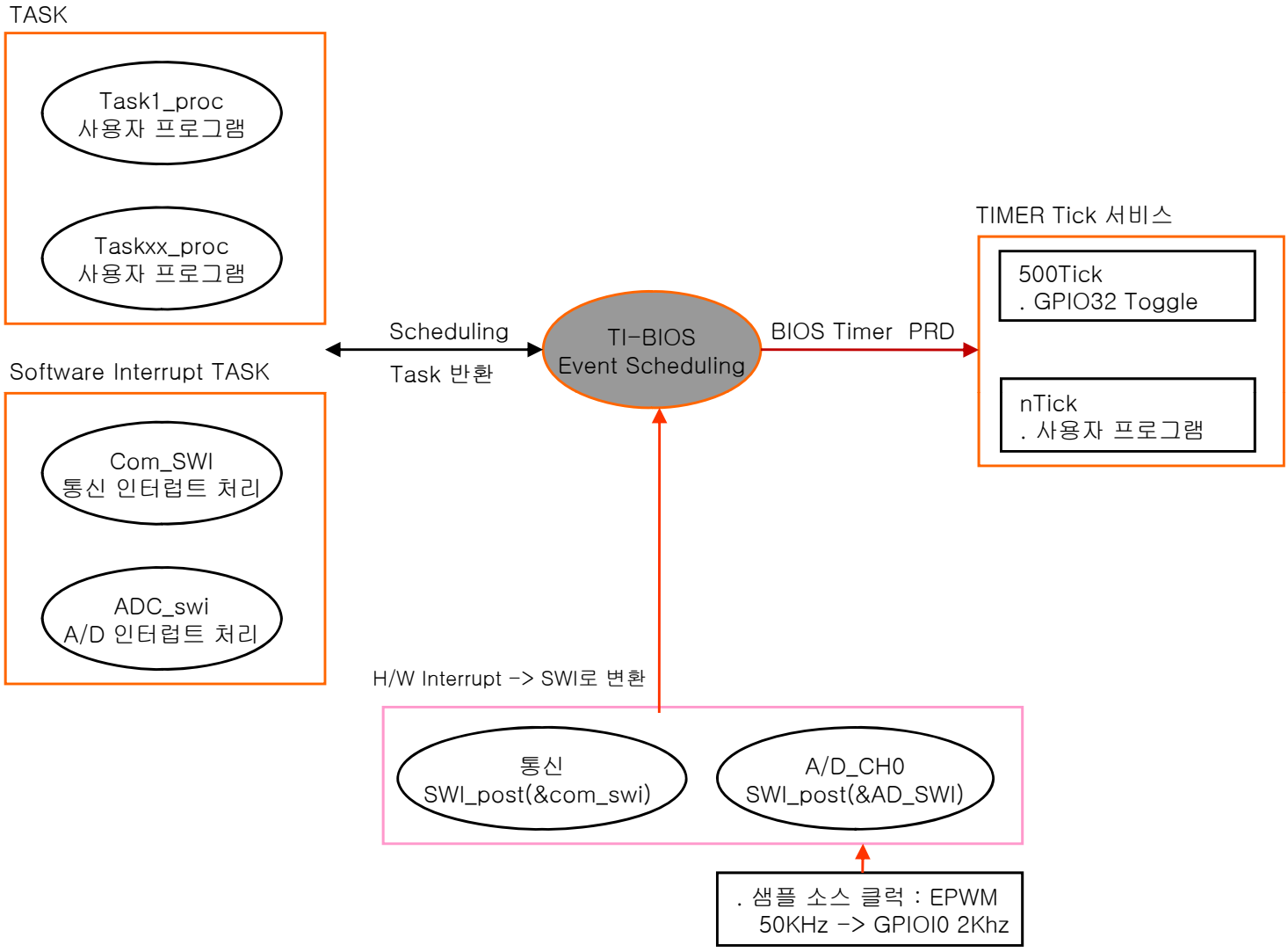
```
#include "DSP2833x_Device.h“   <- DSP 초기화 및 설정 관련
#include "F28335_example.h”    <- 사용자 외부 함수,변수,정의 관리

void main(void)
{
      InitSysCtrl();                <- CPU 클럭 설정((30*10) / 2 = 150M)
      InitPieCtrl();                <- 인터럽트 관련 초기화
      InitWatchdog();               <- watch-dog 설정 및 초기화
      InitGpio();                   <- CPU I/O 설정(IN,OUT,기본기능..) _EX_BUS_ON정의에 따라 외부 버스 ON
      InitXintf();                  <- 내부 주변 디바이스 클럭 설정 및 외부 버스 타이밍 설정

** DSP/BIOS 관련 설정 **
ifdef EXAMPLE_FLASH
      memcpy(&secureRamFuncs_runstart,
            &secureRamFuncs_loadstart,&secureRamFuncs_loadend - &secureRamFuncs_loadstart);
      InitFlash();
#endif

// Peripheral Initialization
      InitAdc();                    <- ADC CH0 초기화및 인터럽트 On
      InitEPwm();                   <- ADC 소스 클럭 공급 PWM 초기화


      asm(" EALLOW");               <-  Enable EALLOW protected register access
      GpioCtrlRegs.GPBMUX1.bit.GPIO32 = 0;<- GPIO032 GPIO
      GpioCtrlRegs.GPBDIR.bit.GPIO32 = 1;   <- GPIO32  output
      GpioDataRegs.GPBSET.bit.GPIO32 = 1; <- GPIO32 pin is set to 1
      asm(" EDIS");                 // Disable EALLOW protected register access

**  DSP/BIOS에서 TINT2,DLOGINT를 사용 하므로 BIOS사용 인터럽트 허가 **
      SetDBGIER(IER | 0x6000);   <- Enable everything in IER, plus TINT2 and DLOGINT
      *(volatile unsigned int *)0x00000C14 |= 0x0C00;<- Set TIMER2 FREE=SOFT=1

**  아래 main()를 종료 하면 DSP/BIOS가 동작.. **

}

void UserInit(void){                <- 이 함수는 리셋시 DSP/BIOS 초기화 부분에서 한번 수행 후
                                       DSP/BIOS관련  및 사용자 초기화 함수 추가

}

void task1_proc(void){              <- 이 함수는 스케줄러에 관리 되는 TASK


}
```

5. 소스코드 설명(DefaultIsr_BIOS.c)

```
void ADCINT_ISR(void){              <-  ADC H/W(P1_6) 인터럽트를 바로 처리하지 않고 SWI로 변환하여 실행.

      PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;   <- Must acknowledge the PIE group

      SWI_post(&ADC_swi);           <-  SWI 로 변환
}
```

## - CCS3.3 DSP/BIOS SWI생성

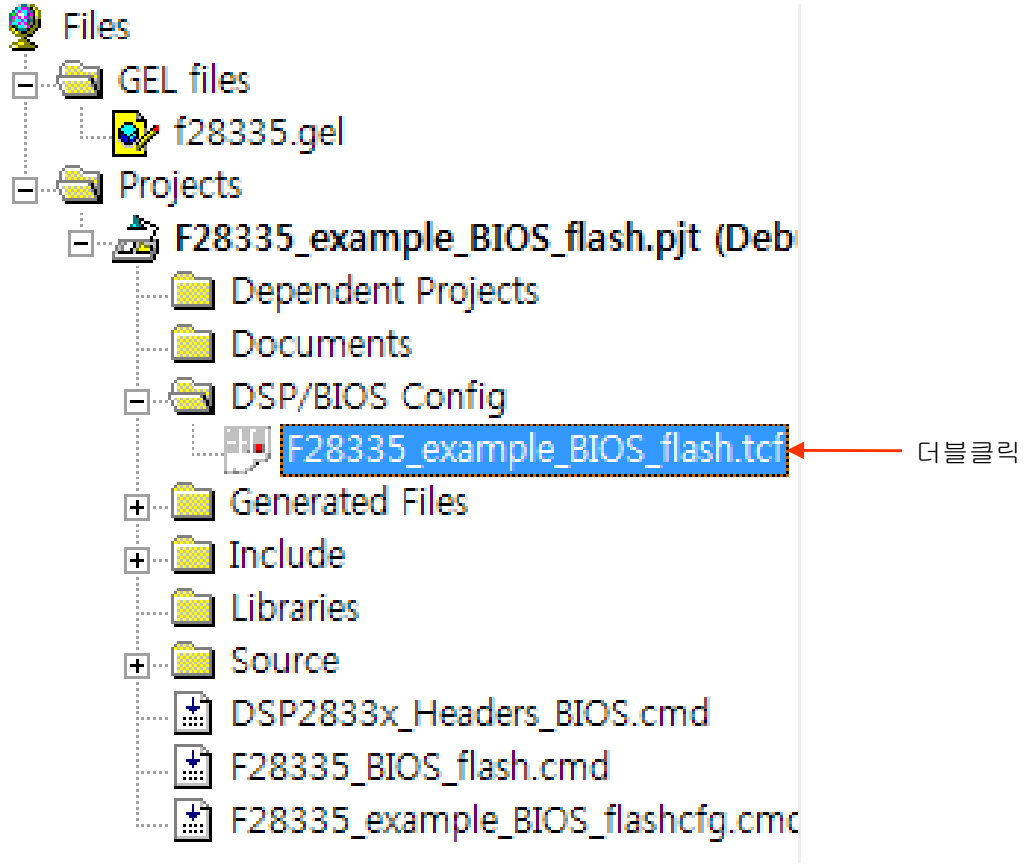
* SWI란 하드웨어 인터럽트에서 발생한 인터럽트를 바로 처리 하지 않고 SWI 인터럽트를 처리하는 함수 (TASK)를 미리 생성해 그곳에서 인터럽트를 처리 하도록 하는 기능이 있다. HWI 루틴에서는 SWI_post( ) 함수를 이용 한다,
일반적으로 인터럽트가 발생하면 바로처리 하는데 반해 SWI가 있는 이유는 BIOS에서 다른 TASK를 처리할 시간이 모자라므로 인터럽트도 TASK화 하여 관리 하기 위함이다.
물론 SWI인터럽트를 사용 하지 않을 수도 있다.



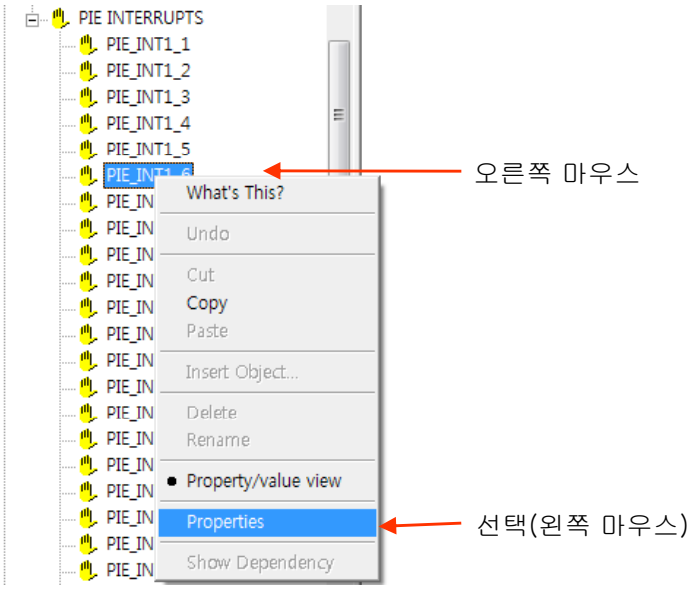

1. Main_Bios.c 를 open후 아래 소스코드를 입력 후 저장한다.

```
void AdcSwi(void)                       <- SWI 스케줄 함수 명 :: 실제 ADC DATA 부분 처리
{
static Uint16 *AdcBufPtr = AdcBuf;      <- ADC 저장 값 버퍼

// Manage the ADC registers
                AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;     <- Reset SEQ1 to CONV00 state
                AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;  <- Clear ADC SEQ1 interrupt flag
// Read the ADC result
                *AdcBufPtr++ = AdcRegs.ADCRESULT0 >> 4; <- ADC 데이터 저장
// Brute-force the circular buffer
                if( AdcBufPtr == (AdcBuf + ADC_BUF_LEN) ) <- 버퍼 초과 검사 Ring Buf 구현
                        AdcBufPtr = AdcBuf;
}
```

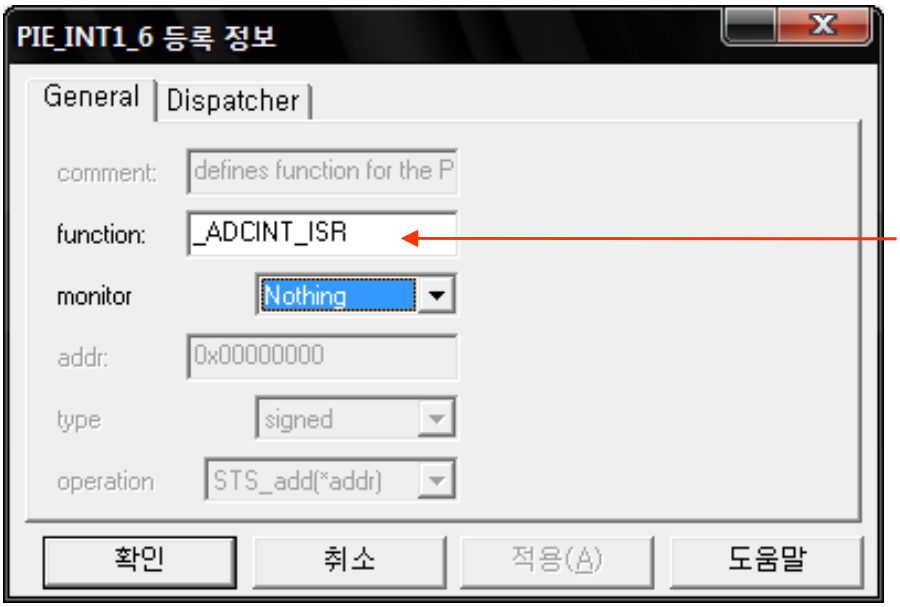
2. DSP/BIOS Config->*.tcf 를 실행 한다.



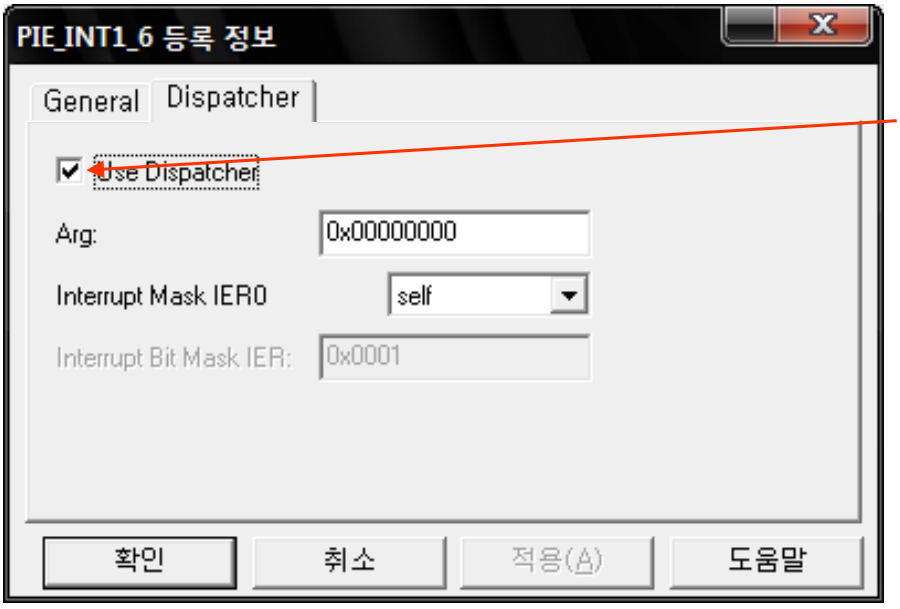
3. 하드웨어 인터럽트를 SWI로 변환(Scheduling ->HWI Hardware Interrupt Service Routine Manager)



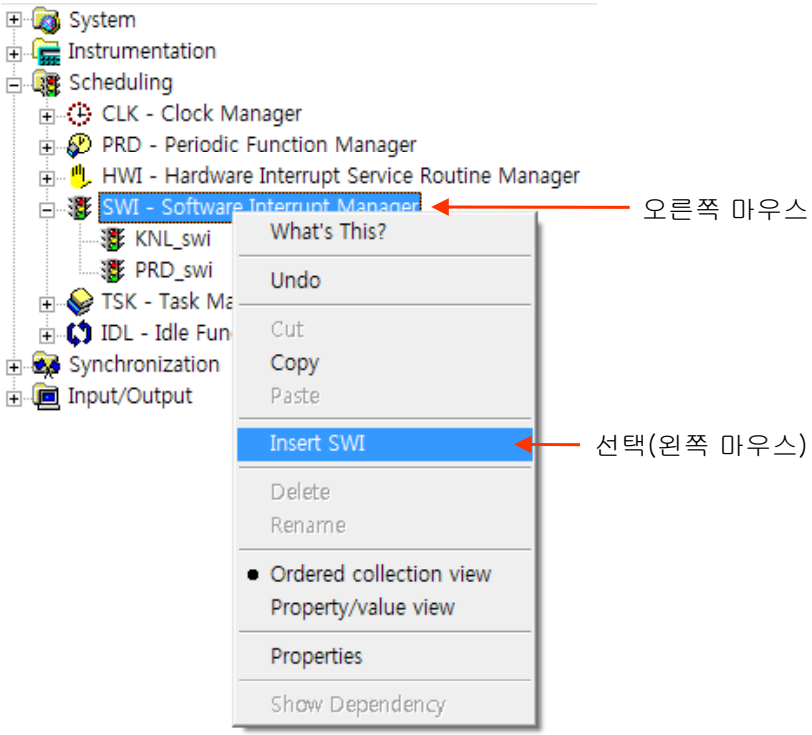

* 하드웨어 인터럽트 등록(HWI)



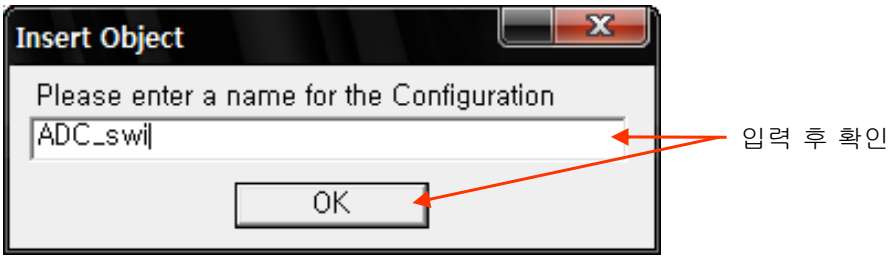

* 소프트웨어 인터럽트 허가(SWI)



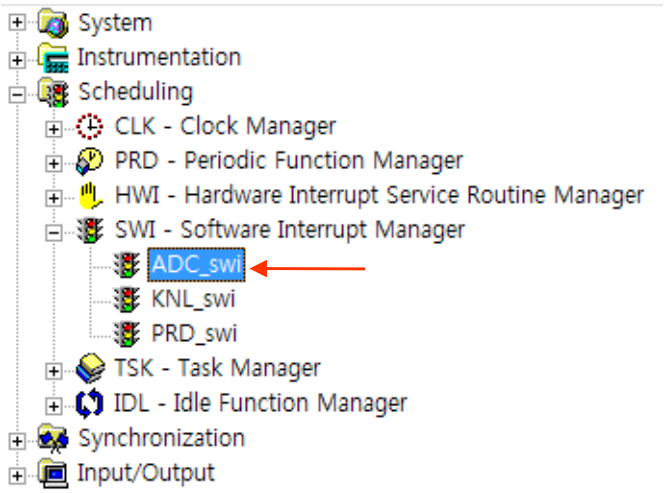
4. Scheduling SWI 등록(. Scheduling ->Software Interrupt Manager)



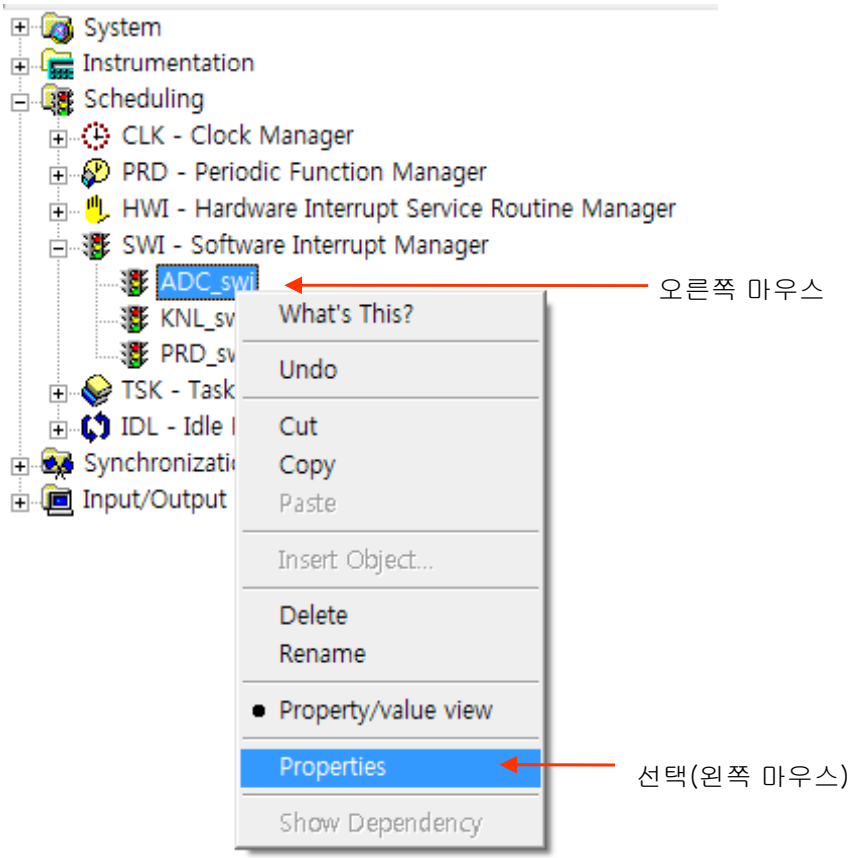

* SWI 관리 명을 입력 한다.(프로그램 에서  호출 주소가 됨.)



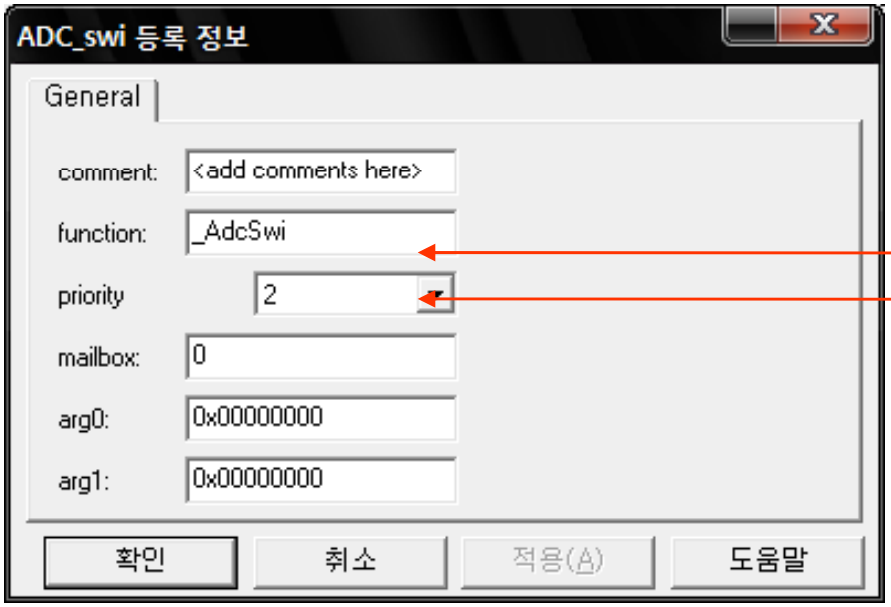
* Task 관리자 생성 확인



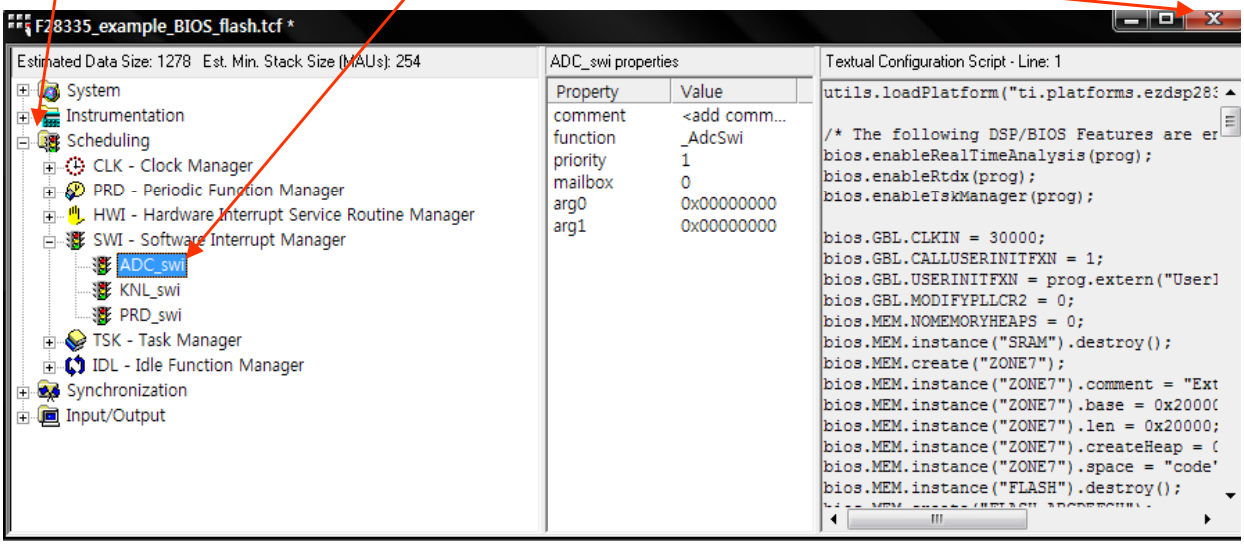

* 생성한  LOG tace에 사용자 환경을 설정 한다.(trace 선택후 오른쪽 버튼)

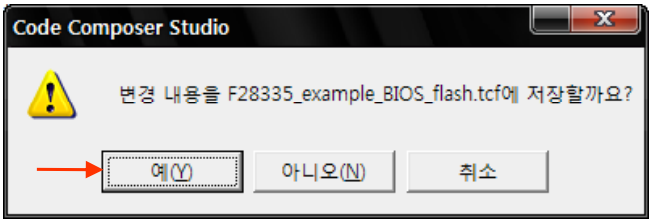


* General 에서 기본 정보를 설정 한다.



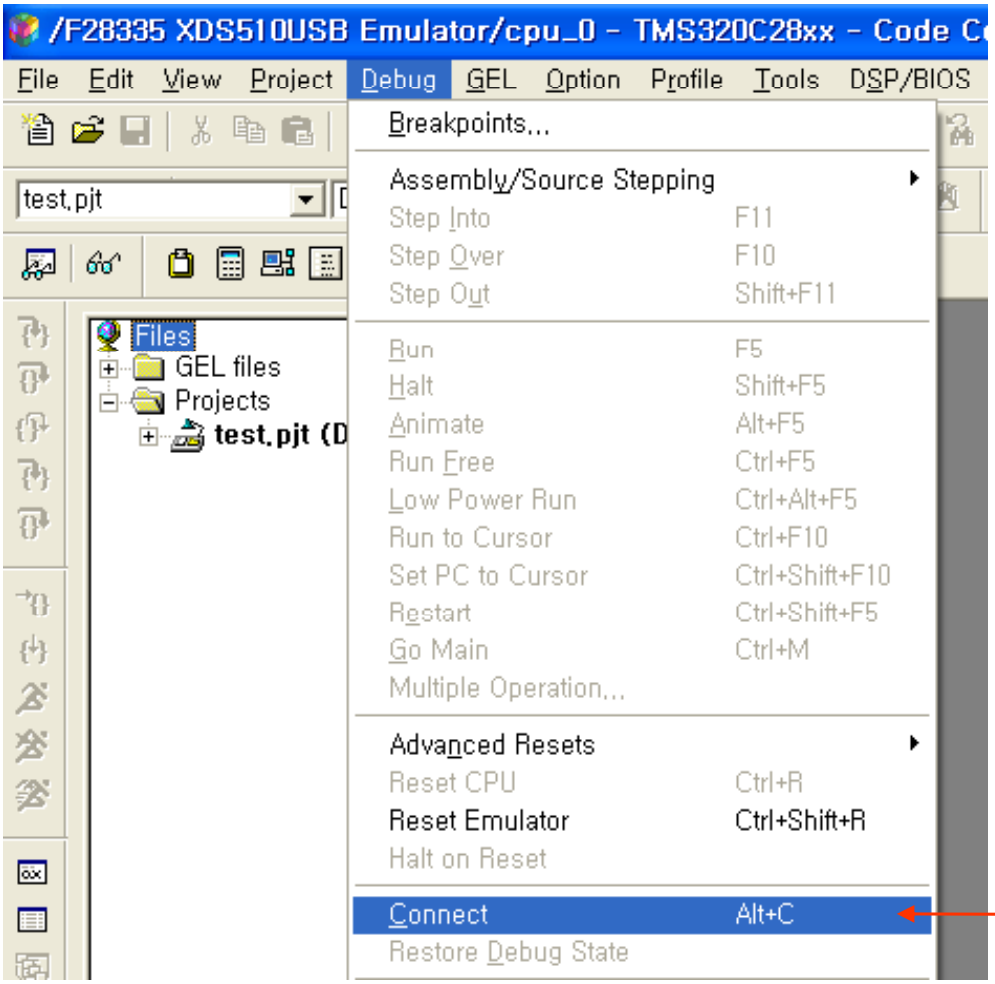

* Scheduling SWI에서 생성 된 ADC_swi를 확인후 *.tcf 파일을 종료 합니다.



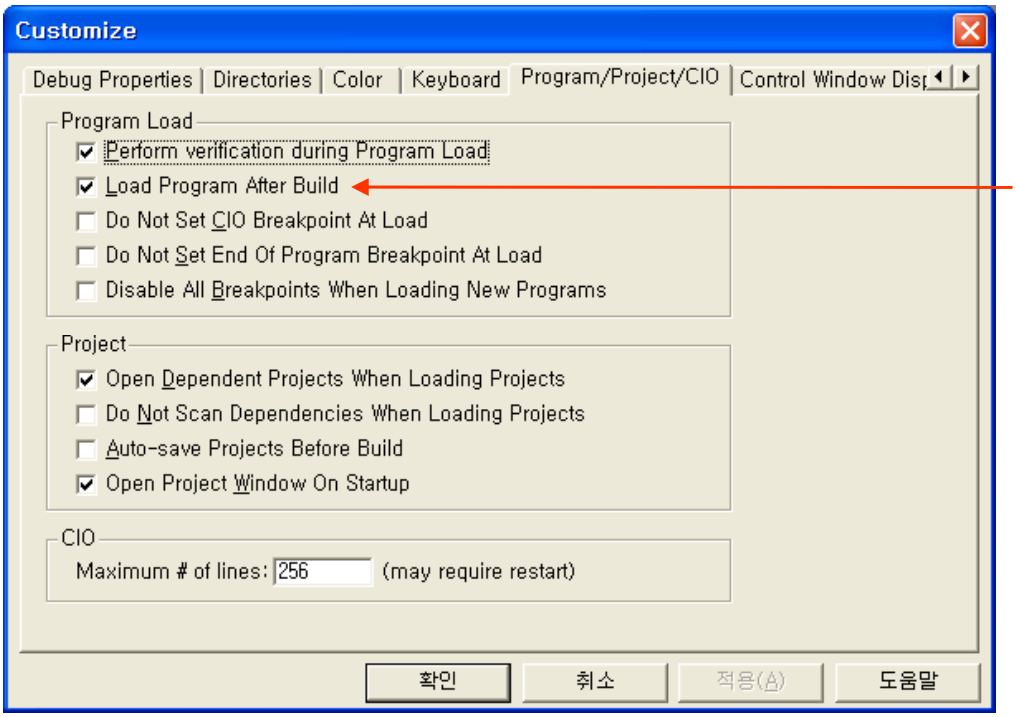
## - CCS3.3 DSP/BIOS PRD TASK 실행

1. JTAG 및 에뮬레이터를 연결 합니다.



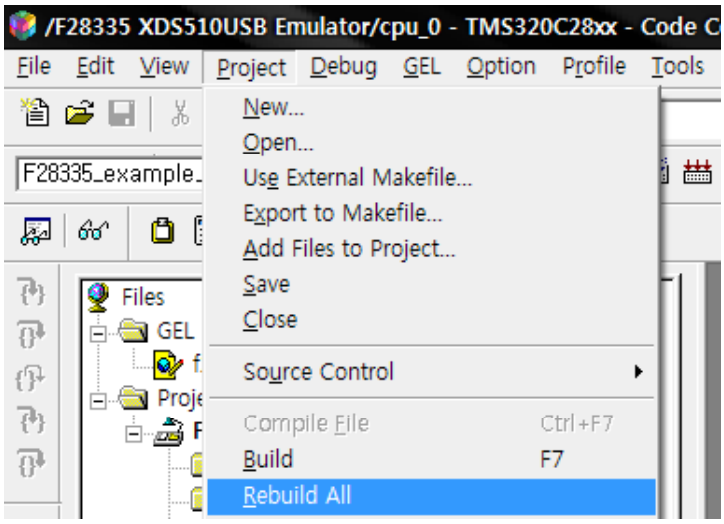

선택후
마우스 왼쪽 버튼 클릭

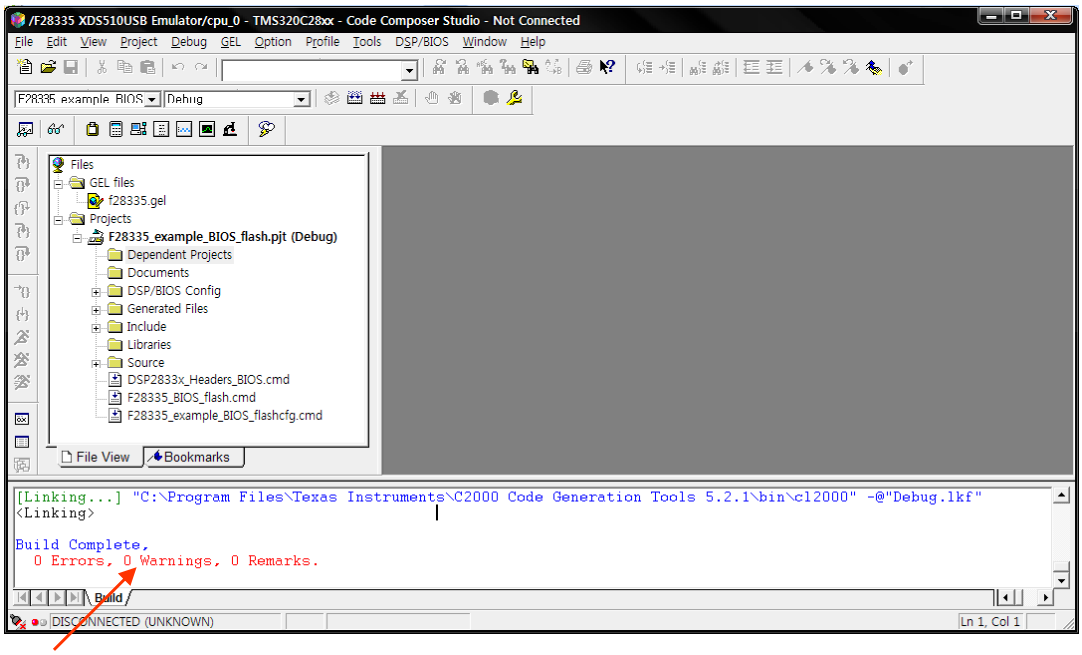
2. 내부램 으로 프로그램을 실행할 경우 아래와 같이 설정 합니다.(Option->Customize)




체크

3. 컴파일 하기(Project->Rebuild All)

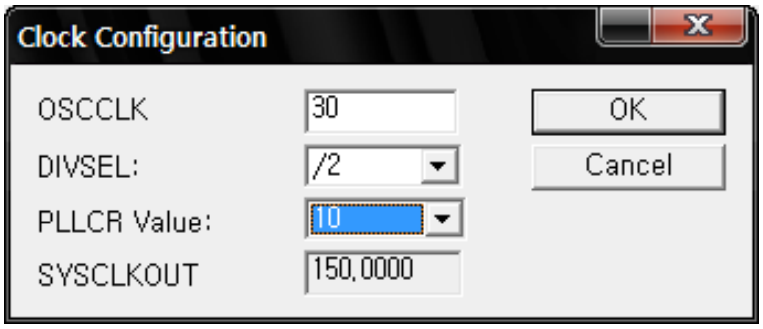


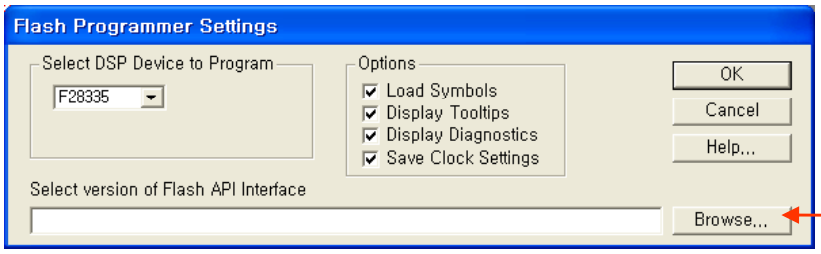

에러 확인

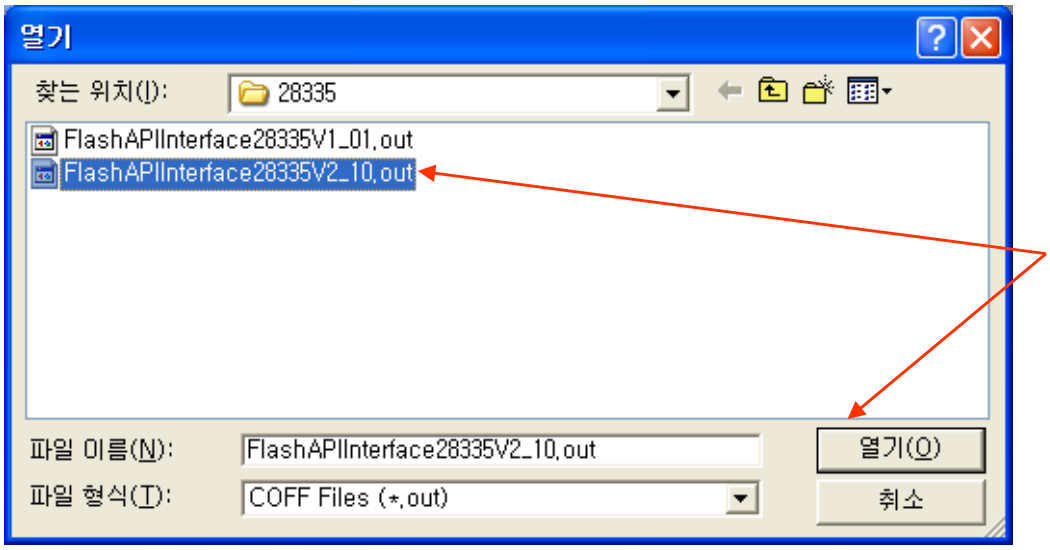
4. FLASH에 프로그램 하기

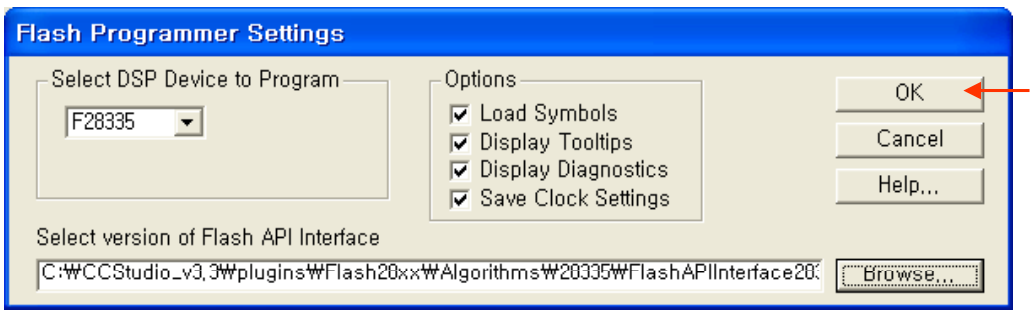

* 아래 CLOCK 설정 메뉴를 사용자에 맞게 설정 합니다.

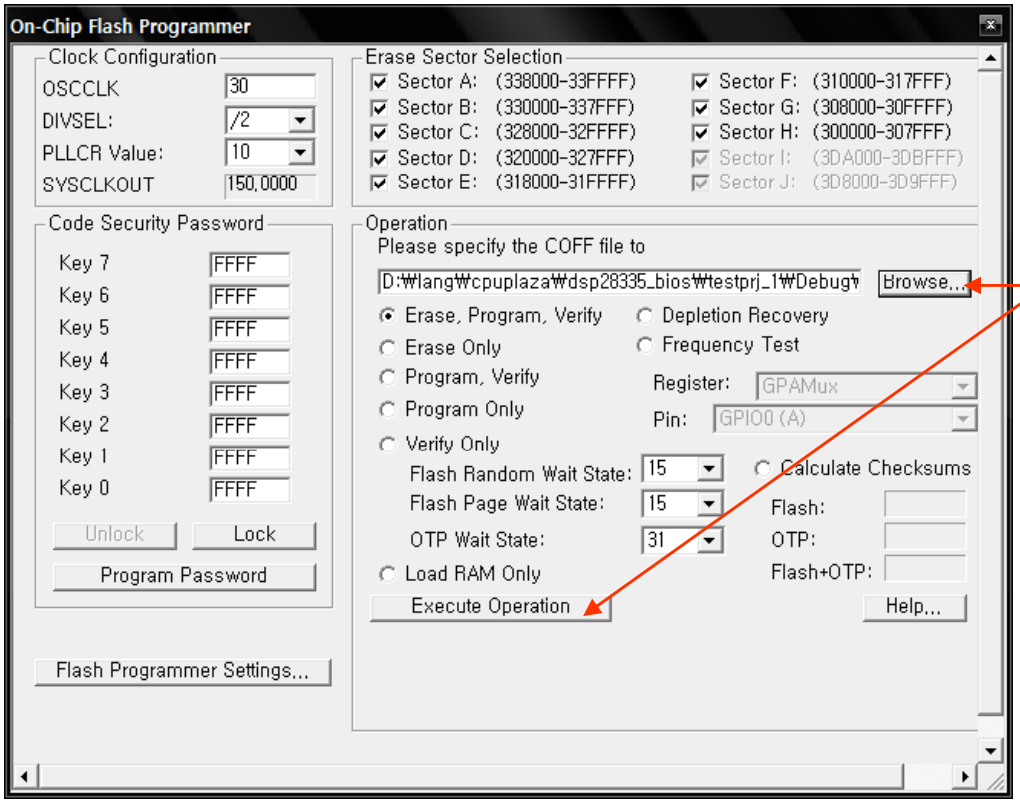



* API Interface 파일을 등록 합니다.



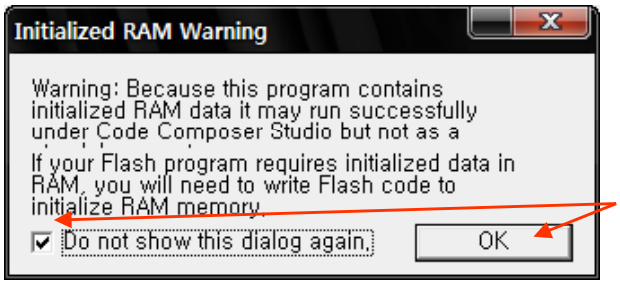



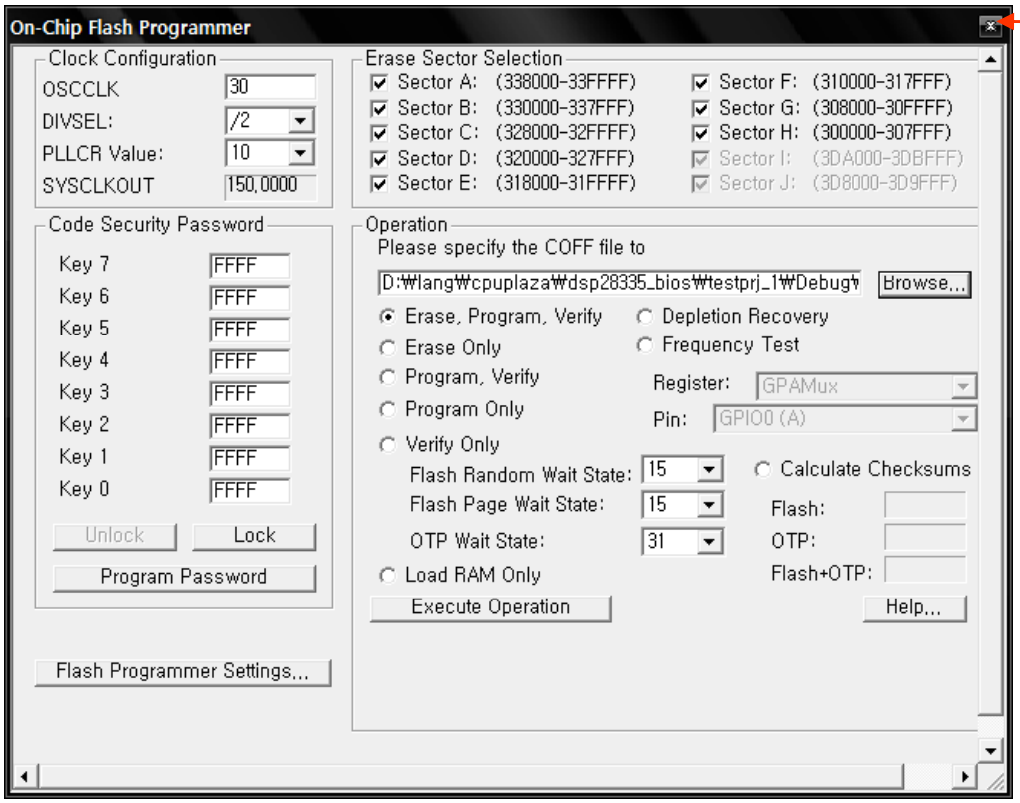



Browe.. 창에서
파일을 선택후 Excute
Operation탭을 실행
합니다.

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

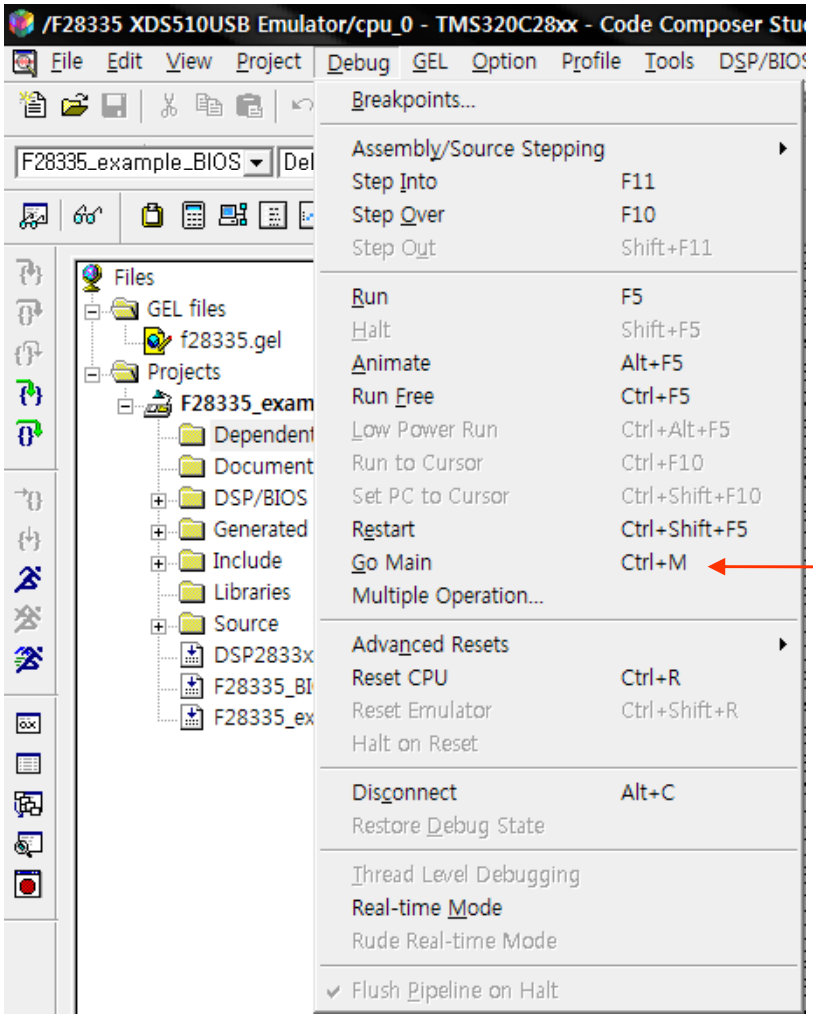



체크후 확인




닫음

5. 프로그램을 로딩후 Debug 탭에서 Go Main 기능을 실행 합니다.



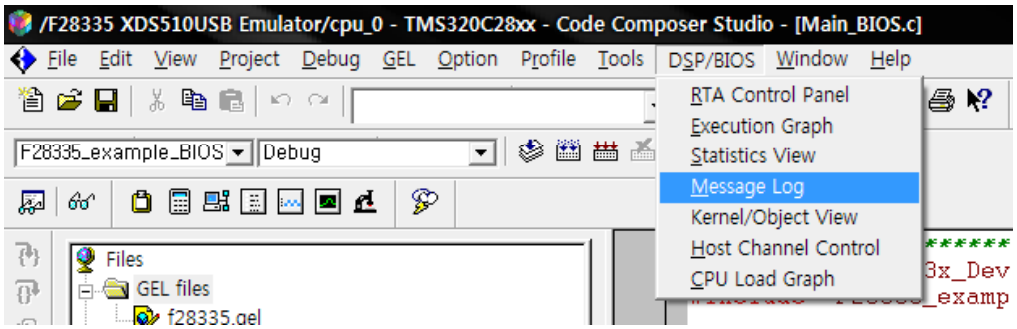

Main( ) 함수 위치 로 이동

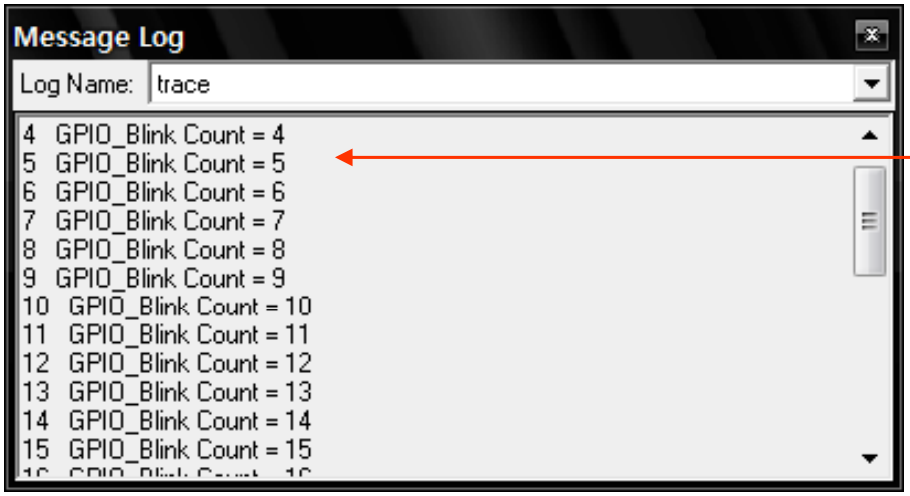
6. Main_Bios.c의 AdcSWI() 에 break mode를 설정 합니다.

```
// ======================= Adc 소프트웨어 인터럽트 ====================
// [인수] void
// [참고] DSP/BIOS 500ms 스케줄 타임
// ---------------------------------------------------------------------
void AdcSwi(void)
{
static Uint16 *AdcBufPtr = AdcBuf;                  // Pointer to buffer

// Manage the ADC registers
    AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;               // Reset SEQ1 to CONV00 state
    AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;             // Clear ADC SEQ1 interrupt flag

// Read the ADC result
    *AdcBufPtr++ = AdcRegs.ADCRESULT0 >> 4;          // Read the result
// Brute-force the circular buffer
    if( AdcBufPtr == (AdcBuf + ADC_BUF_LEN) )
        AdcBufPtr = AdcBuf;                 // Rewind the pointer to beginning

}
```

- F5 : Debuf->Run
- F9 : Debuf->Break Point Toggle

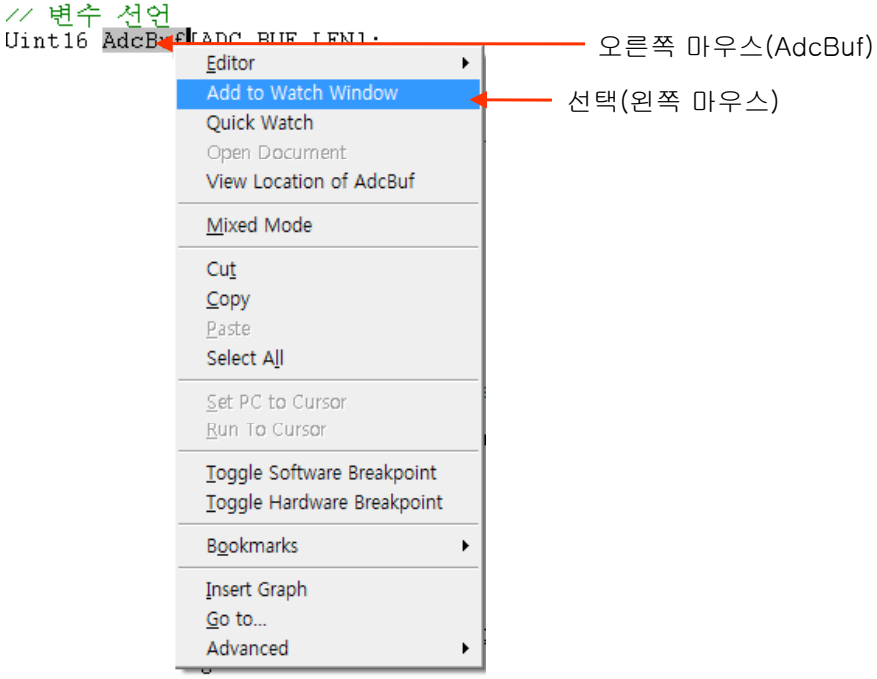
7. DSP BIOS Message Log(DSP/BIOS->Message Log)
   예제 프로그램에서 LOG_printf()로 출력되는 멧세지를 확인 할수 있다.

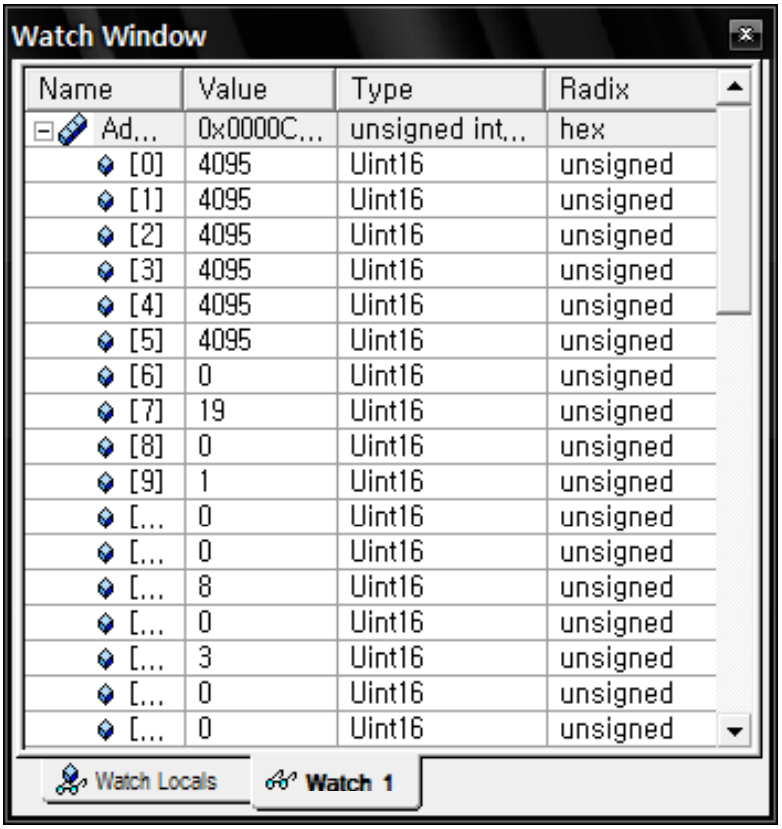


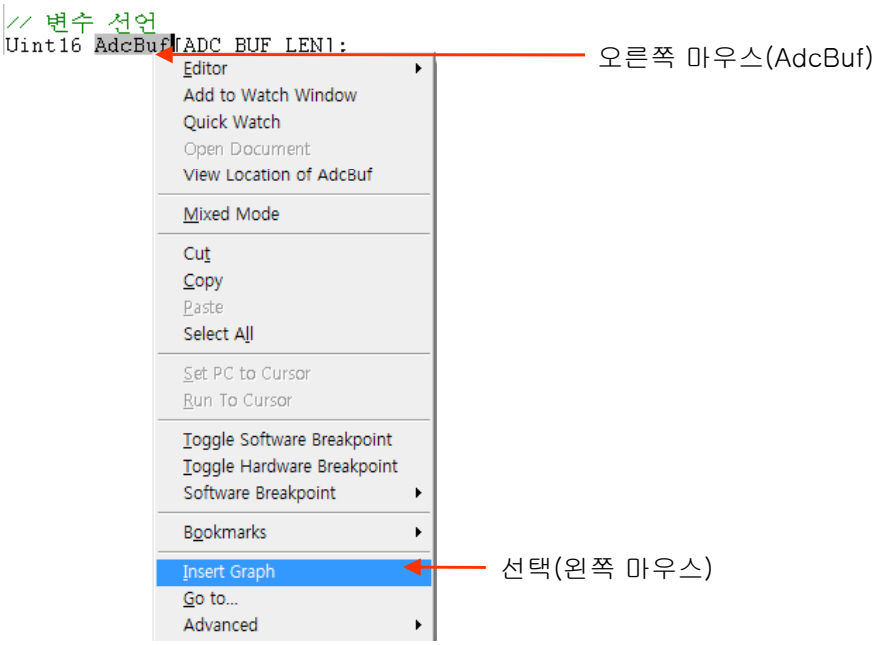

Io_Blink() 함수에서 표시

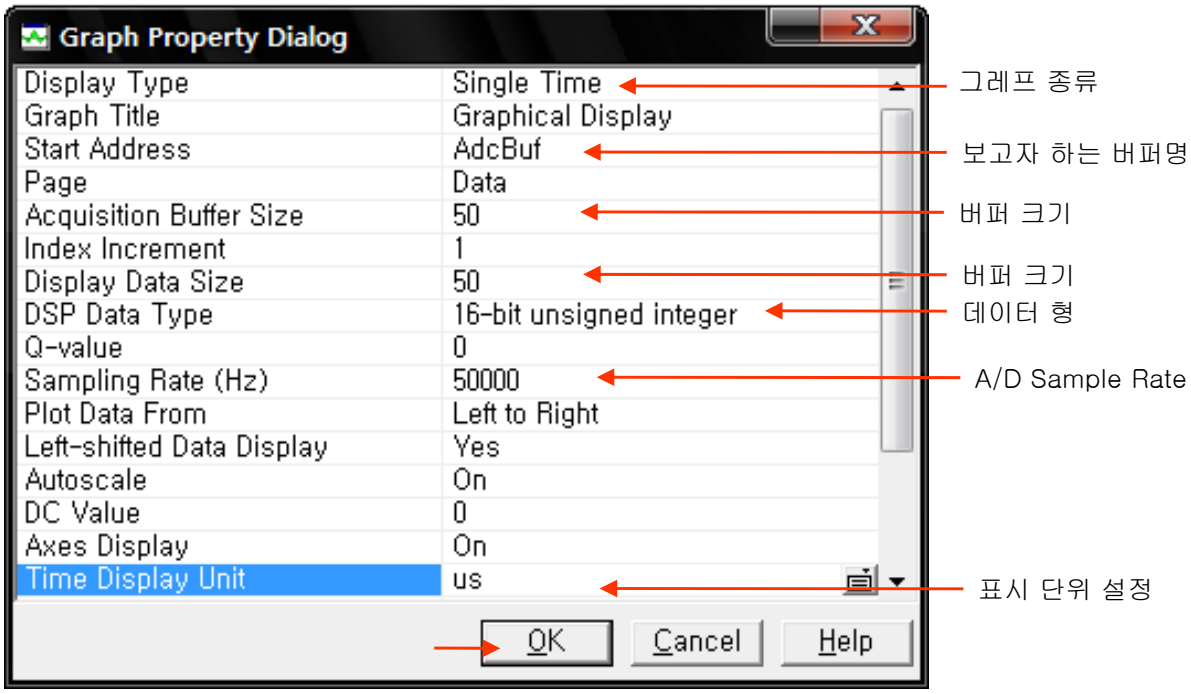
8. A/D 데이터 확인(데이터 버퍼)

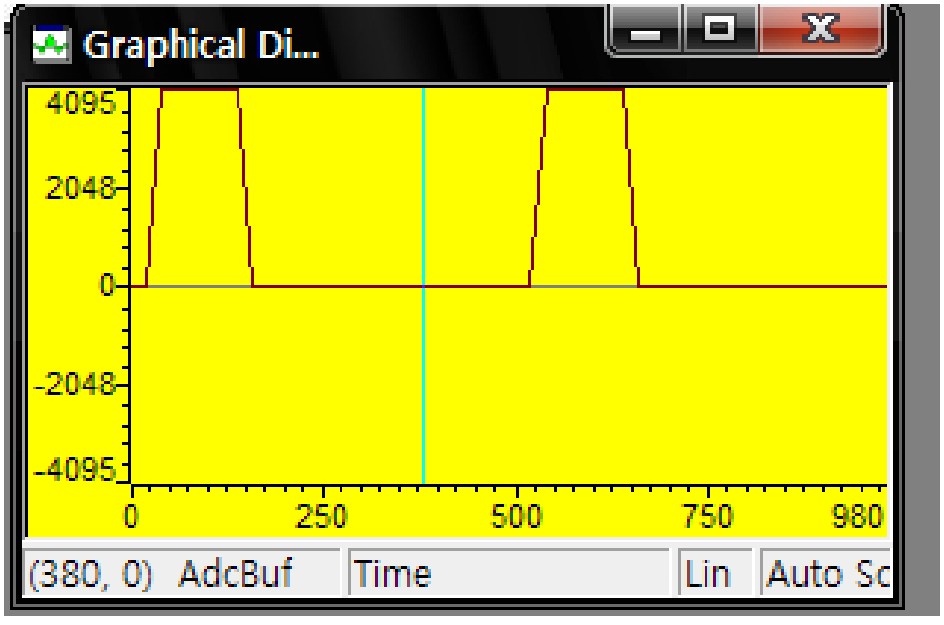


9. A/D 데이터 그레프 확인



10. A/D 데이터 리얼 타임 표시
- 항목8,9를 실시간으로 모니터링 하기 위한 방법이다.
** 주의 아래 순서를 지키지 않으면 장비에 심각한 문제가 발생 할수 있다 **

* START
1. 프로그램을 로딩 한다.
2. ccs3.3 Debug->Halt 상태나, STOP 상태로 만든다.
3. Debug->Real-Time_Mode를 선택하여 체크 한다.
4. View->Real Time Options->Global Continuous를 선택하여 체크 후 OK 버튼 클릭.
5. 8번 이나, 9번 항목 실행 :: 데이터 보기

* STOP

1. Debug->Halt 를 실행 한다.
2. Debug->Real-Time_Mode를 해제 한다.