* CCS3.3 FFT 알고리즘 Project

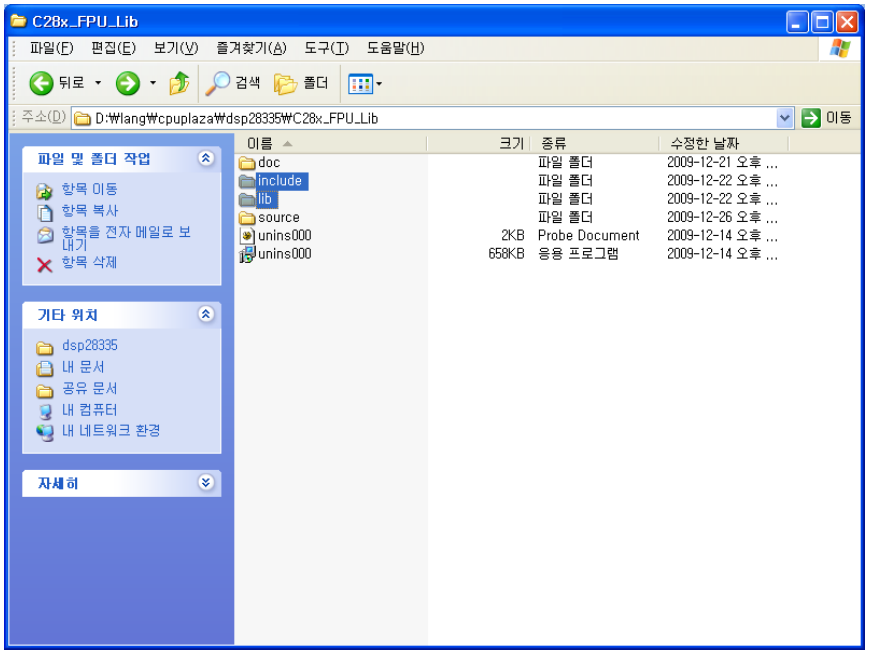
1. TI제공 예제파일을 설치합니다.

   . SPRC624.zip 예제프로그램

2. 본 예제 에서는 당사에서 제공한 아래 라이브러리를 사용한다.

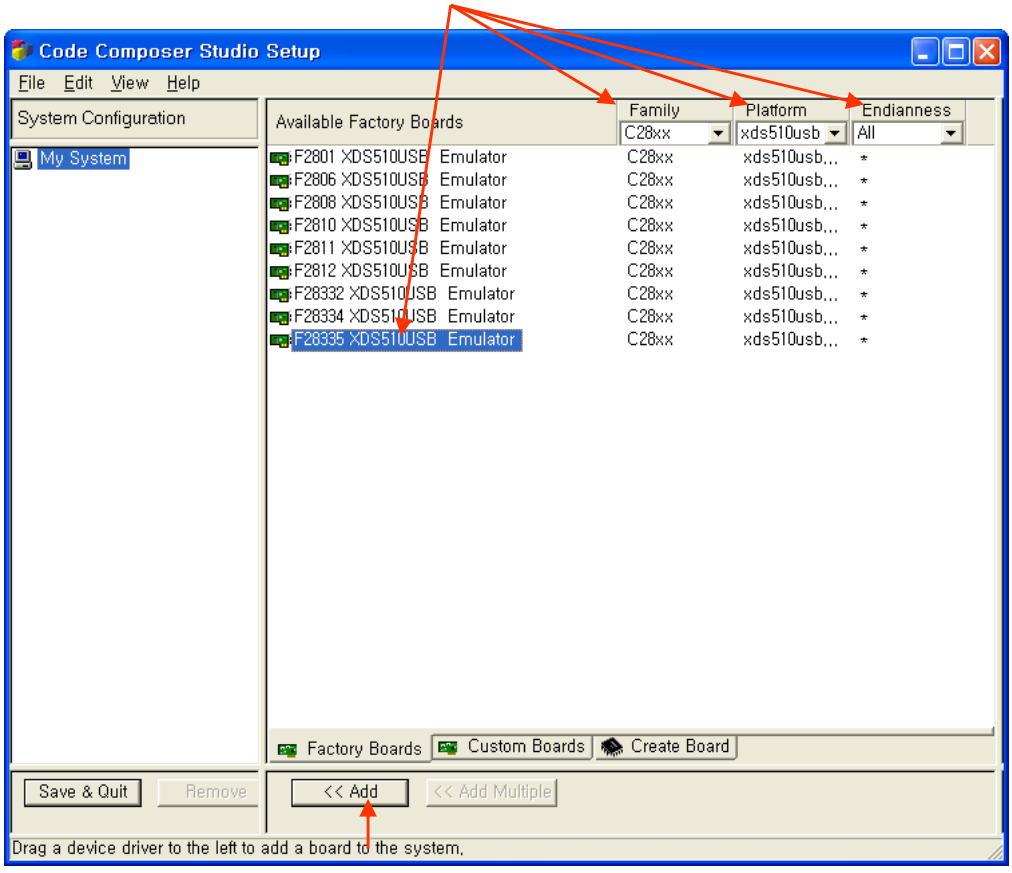
   기존 FFT라이브러리에 함수 추가 후 FLASH 메모리에서 실행 하도록 하였습니다.

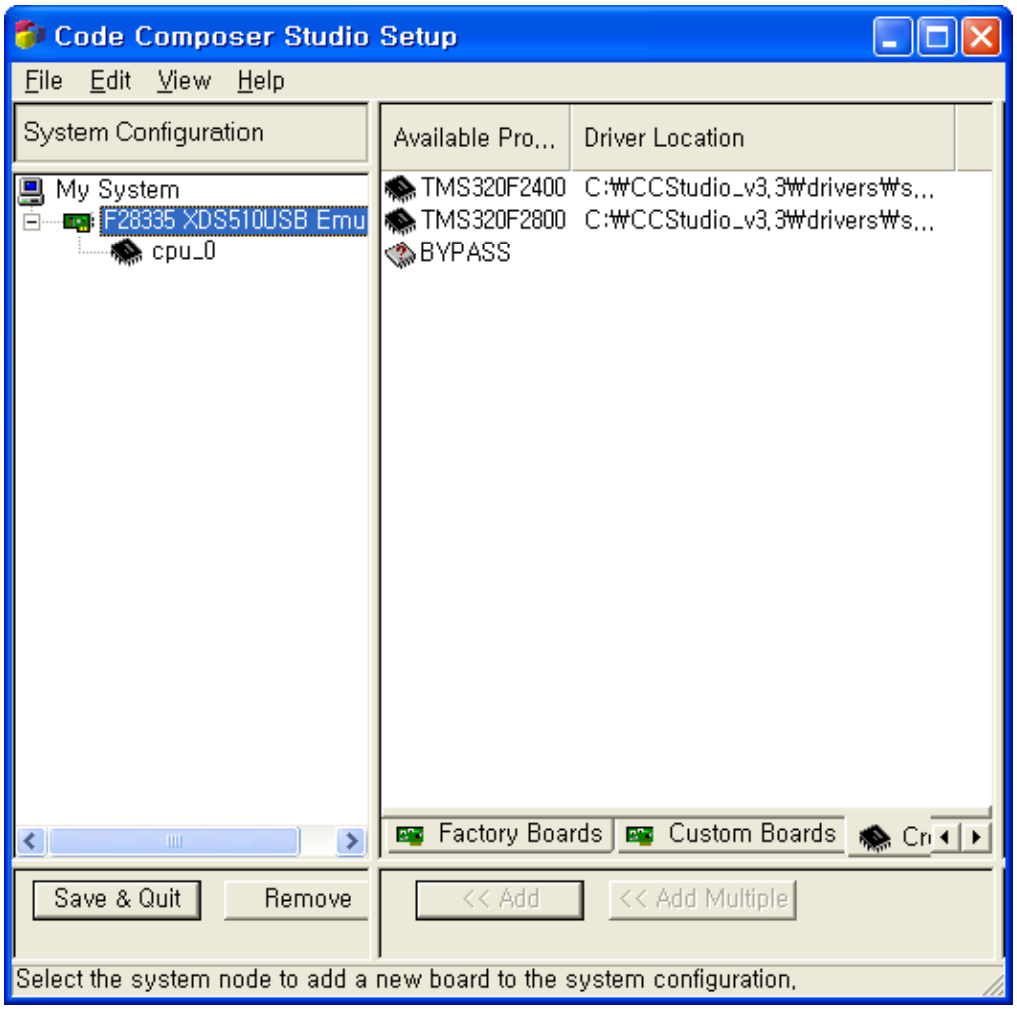
   C:₩tidcs₩c28₩C28x_FPU_Lib



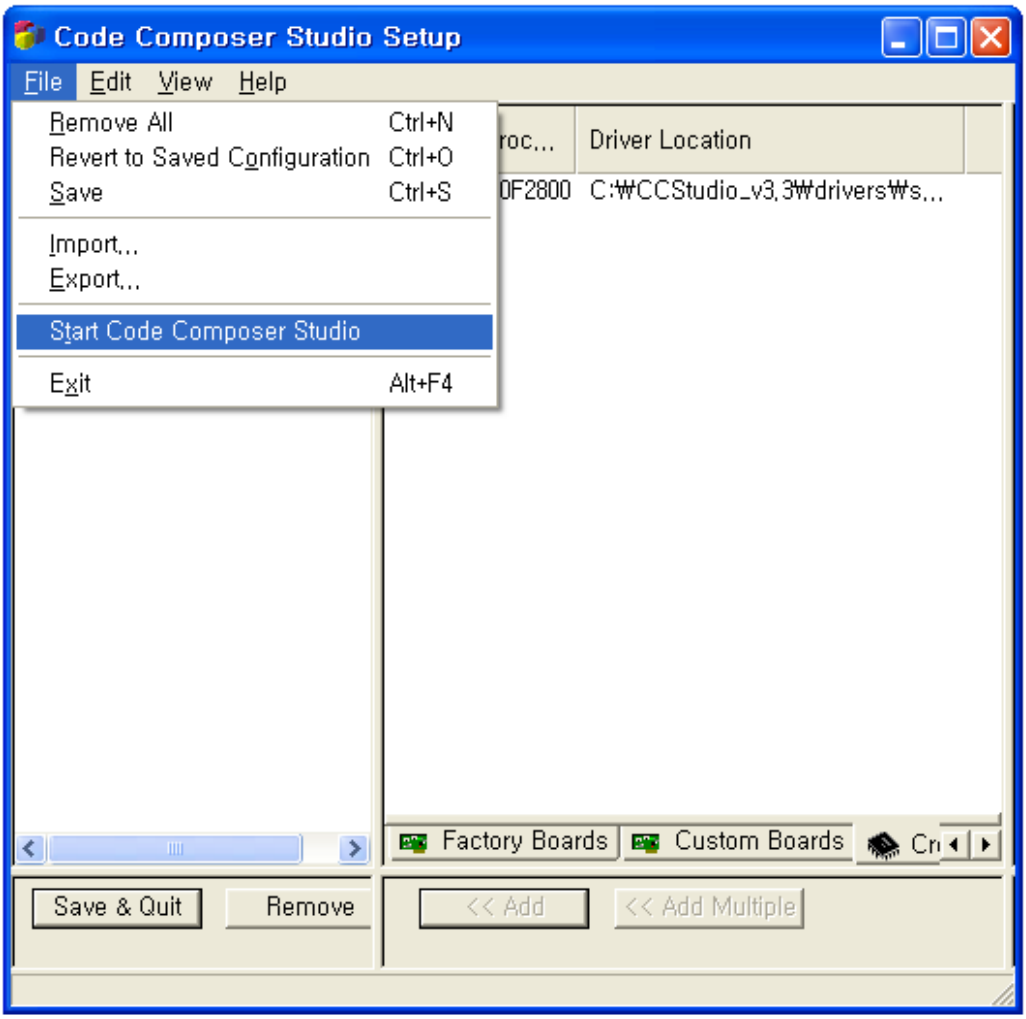
3. Setup CCStudio v3.3을 실행 합니다.

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





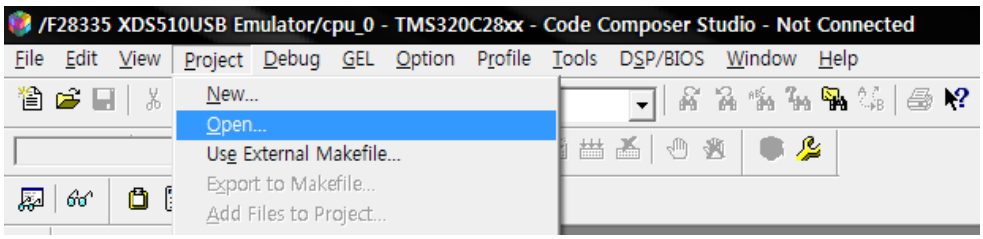
5. My System에서 F283335 XDS510USB Emu를 선택후 Start Code Composer Studio를 실행 합니다.

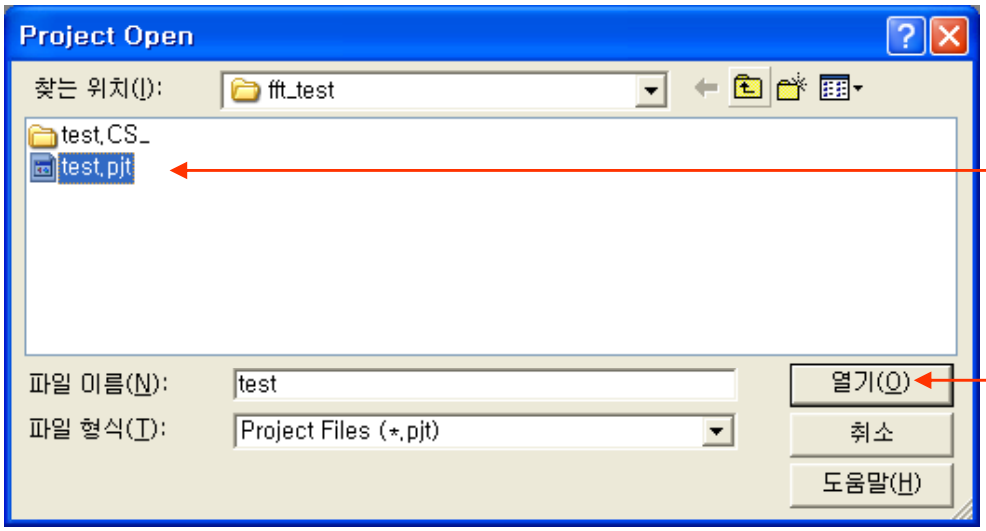


* CCS3.3 DSP FFT Project 시작

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

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



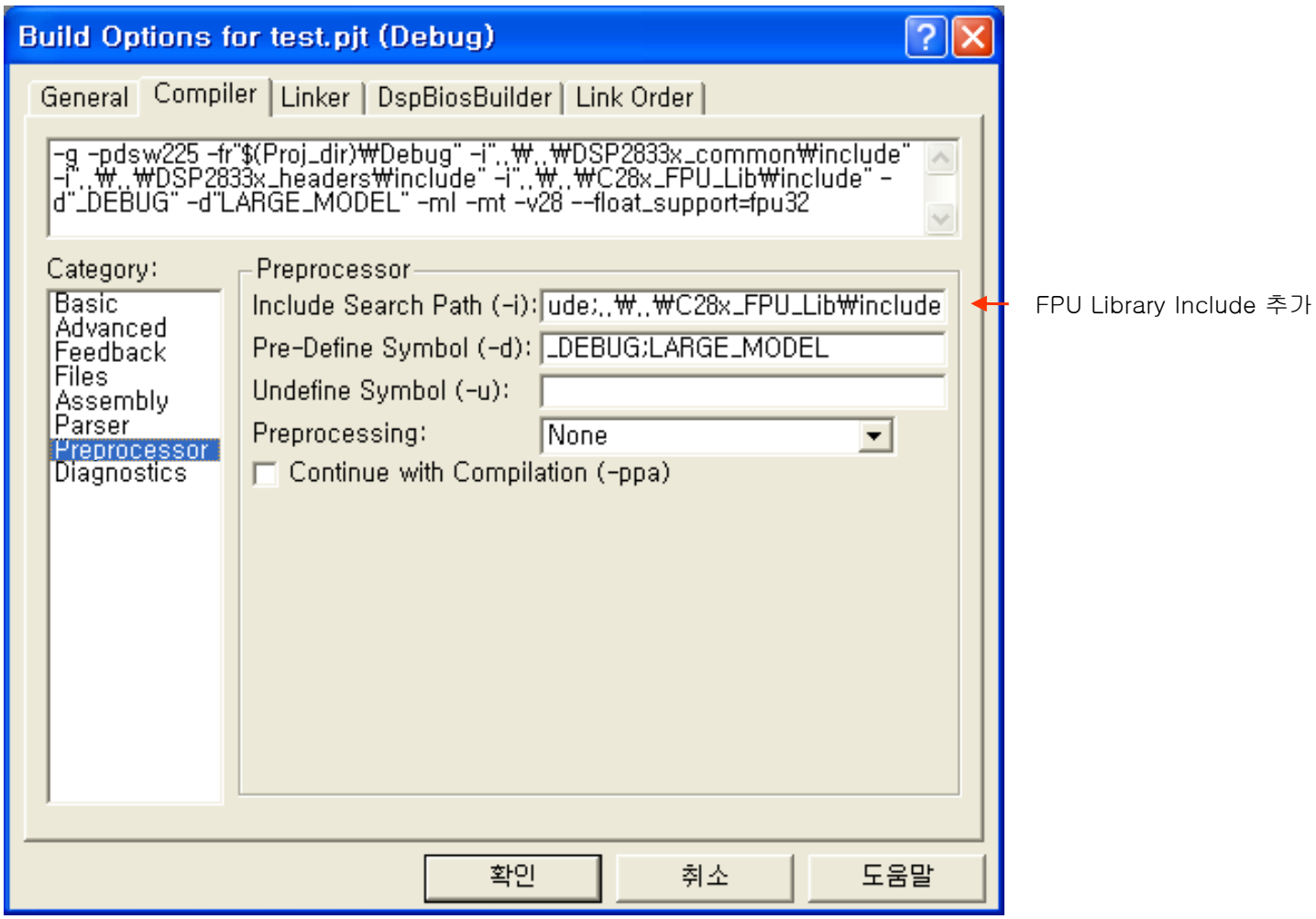


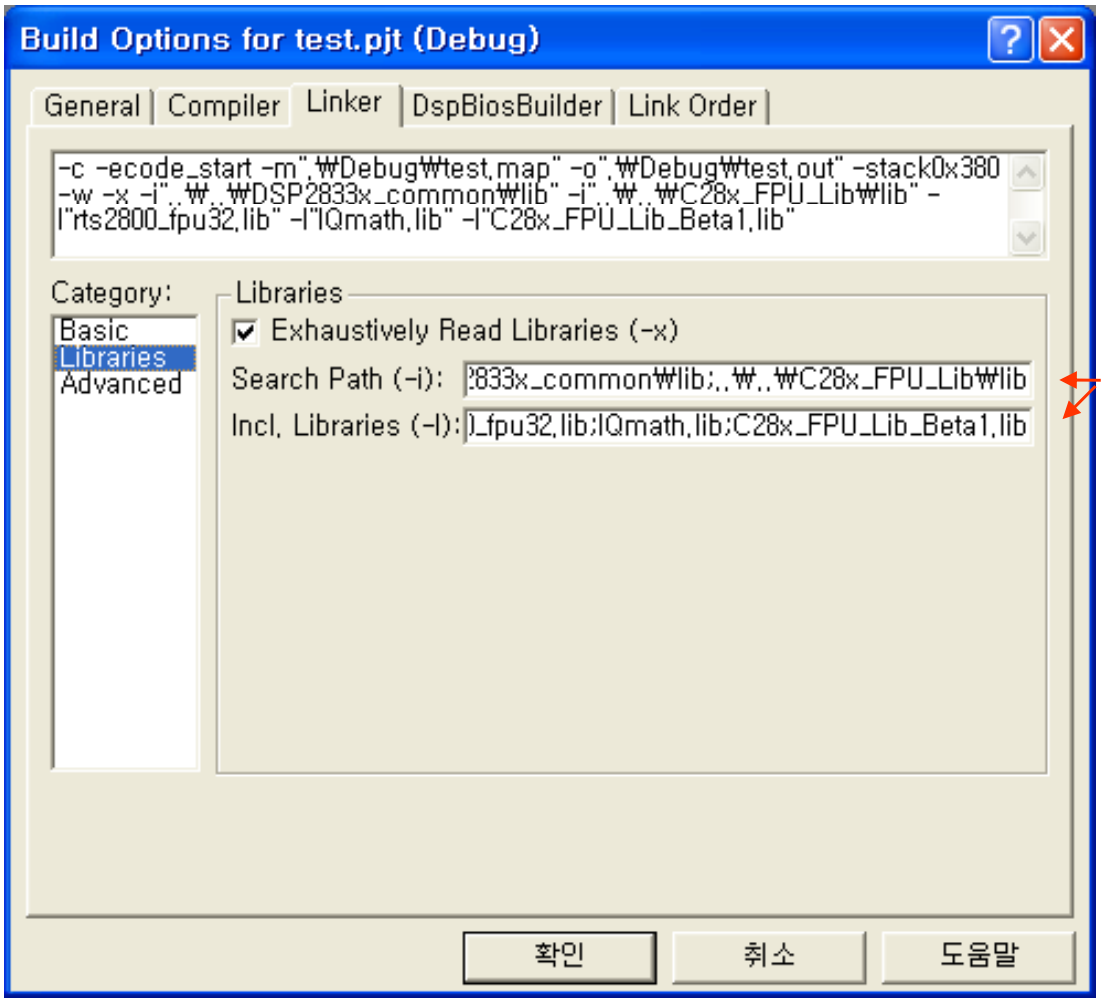
FLASH에서 실행되는 프로젝트

버튼 클릭

3. Projects Source 파일 구성

4. Projects Option 구성(Project->Build Option)

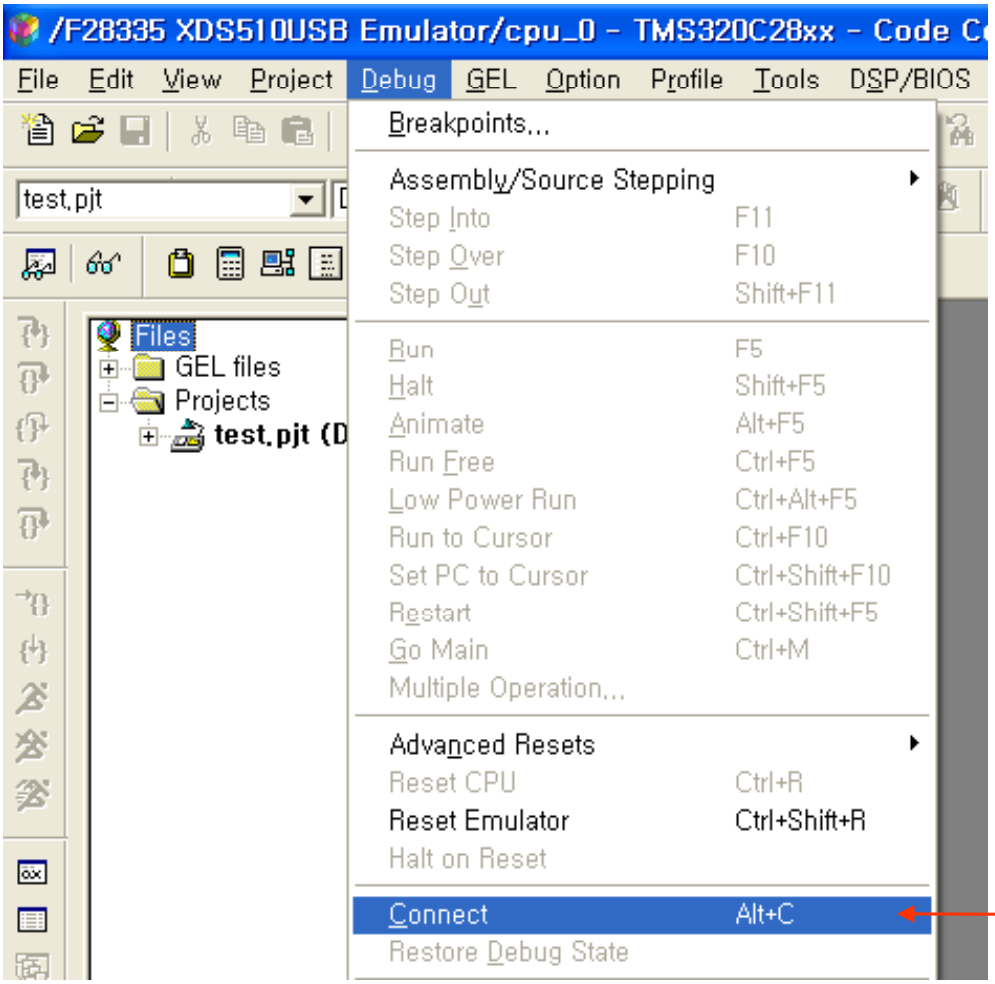




FPU Library 추가

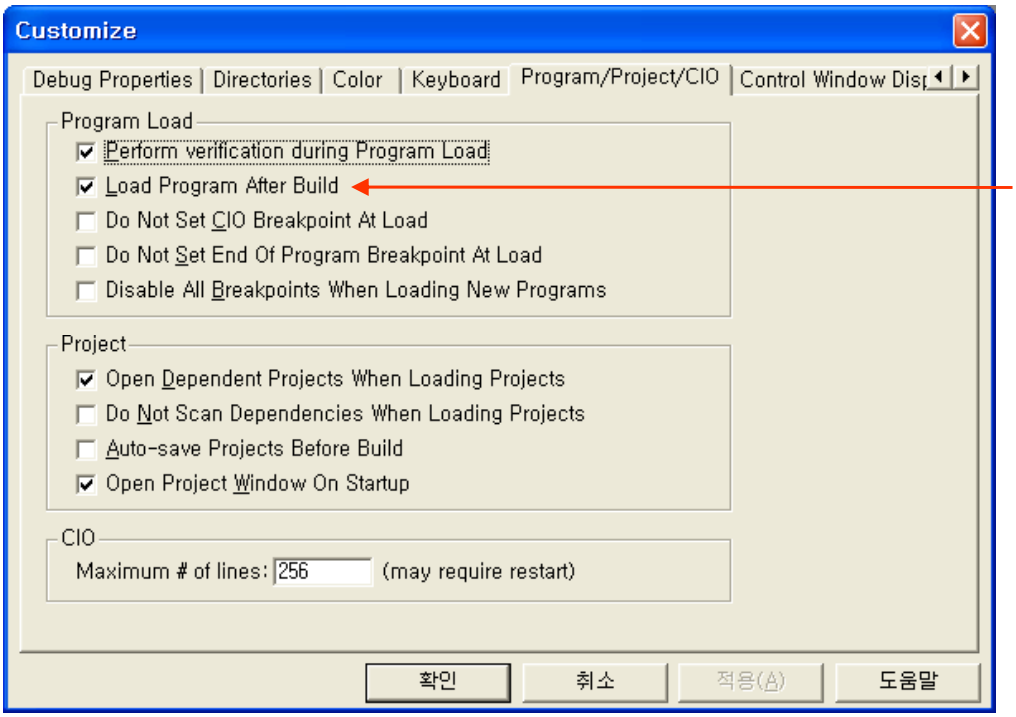
- CCS3.3 DSP Program 실행

1. JTAG 및 에뮬레이터를 연결 합니다.



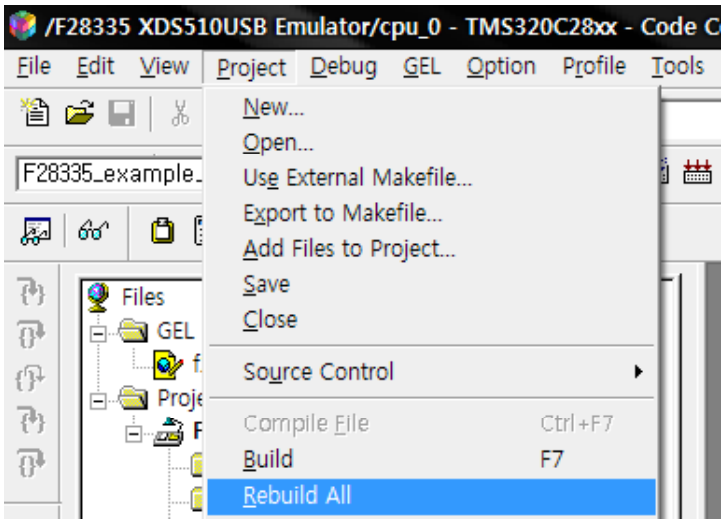
선택후
마우스 왼쪽 버튼 클릭

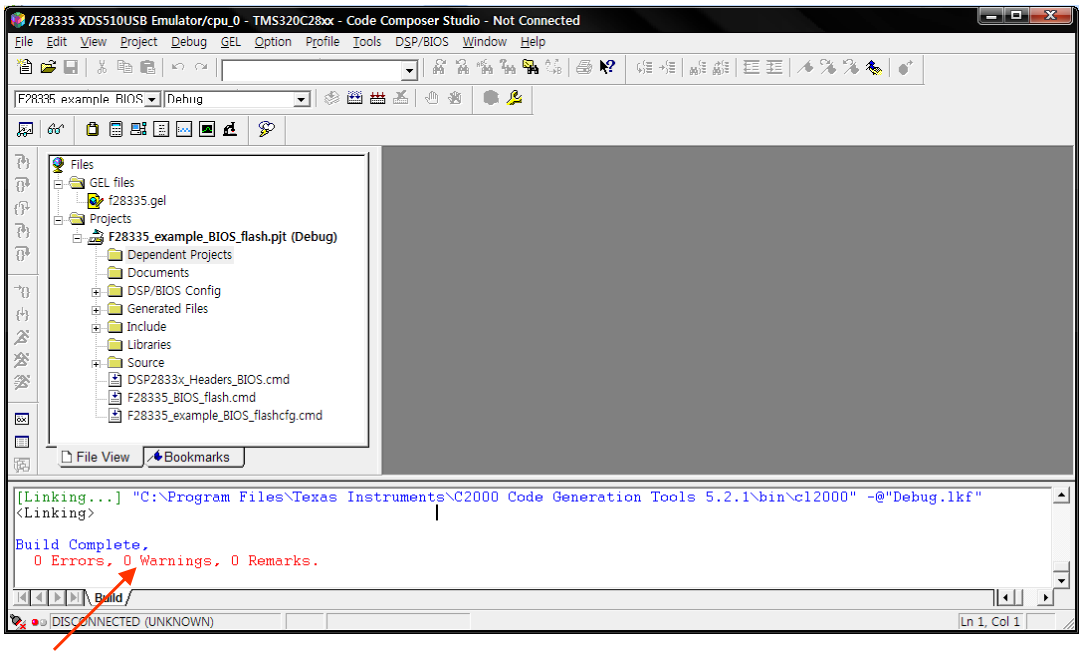
2. 내부램 으로 프로그램을 실행할 경우 아래와 같이 설정 합니다.(Option->Customize)



체크

3. 컴파일 하기(Project->Rebuild All)

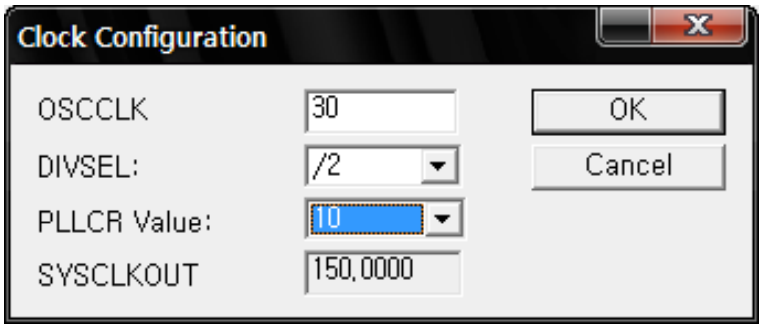




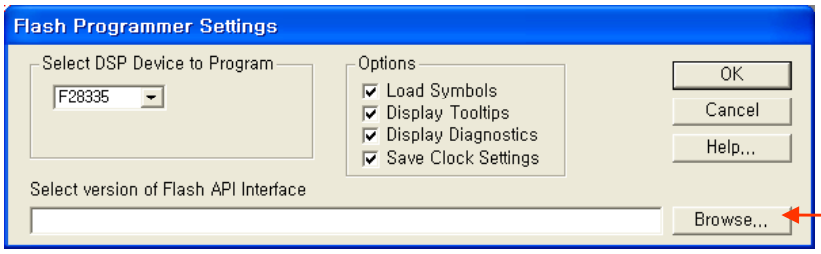
에러 확인

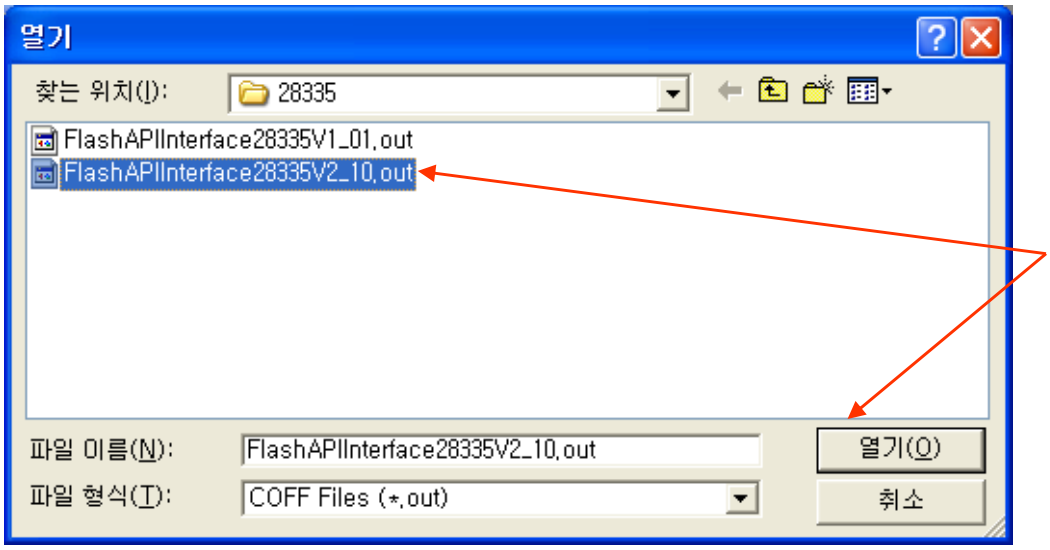
4. FLASH에 프로그램 하기

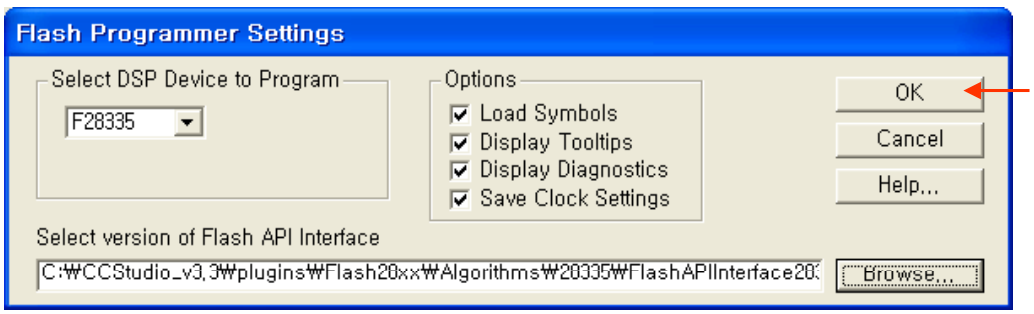
* 아래 CLOCK 설정 메뉴를 사용자에 맞게 설정 합니다.

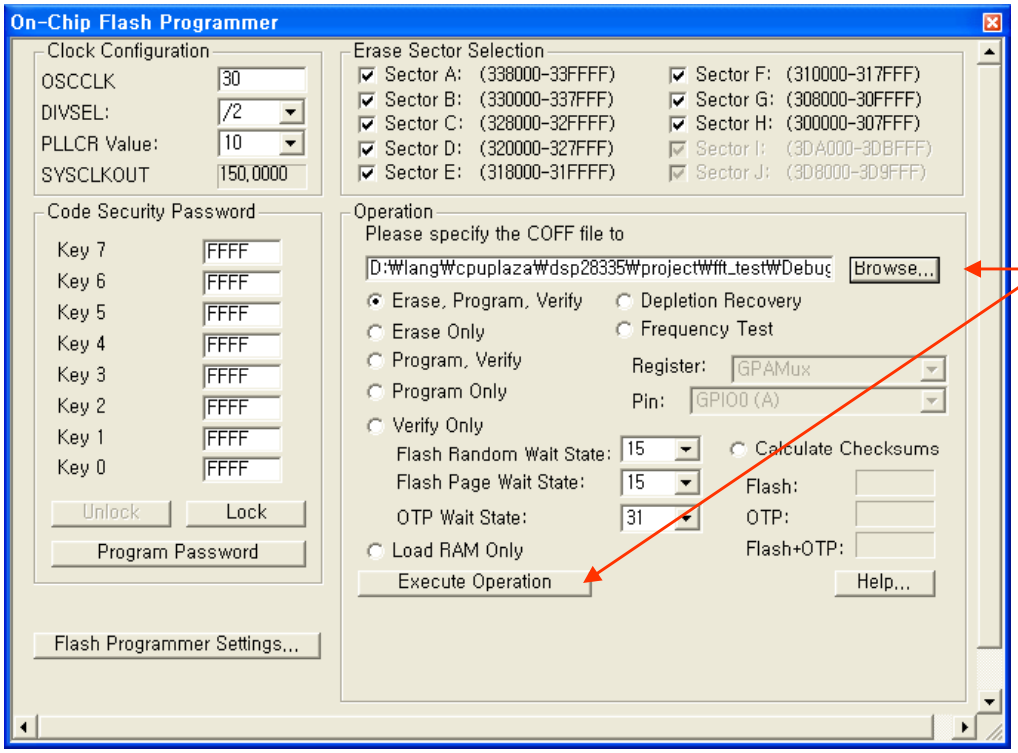


* API Interface 파일을 등록 합니다.



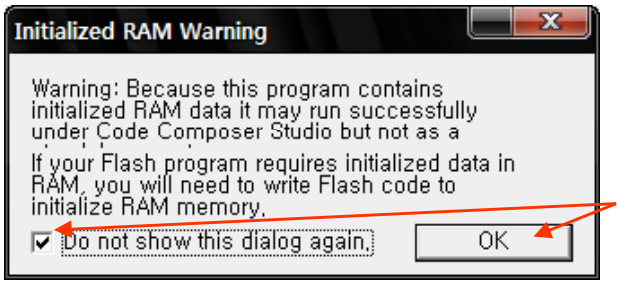




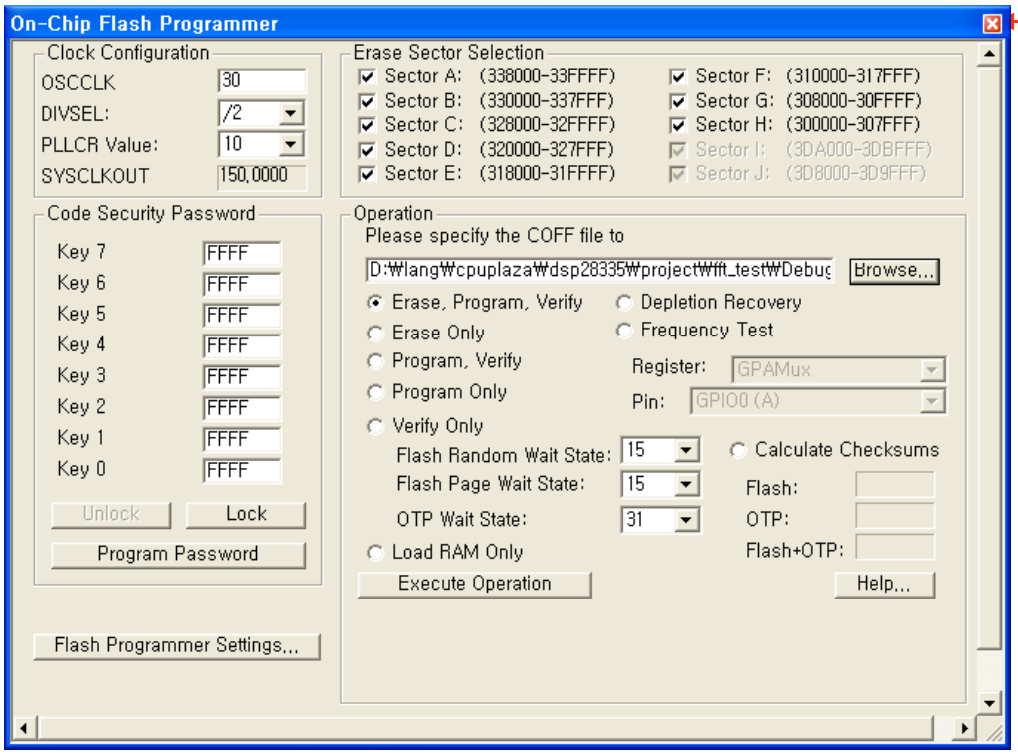


Browe.. 창에서
파일을 선택후 Excute
Operation탭을 실행
합니다.

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

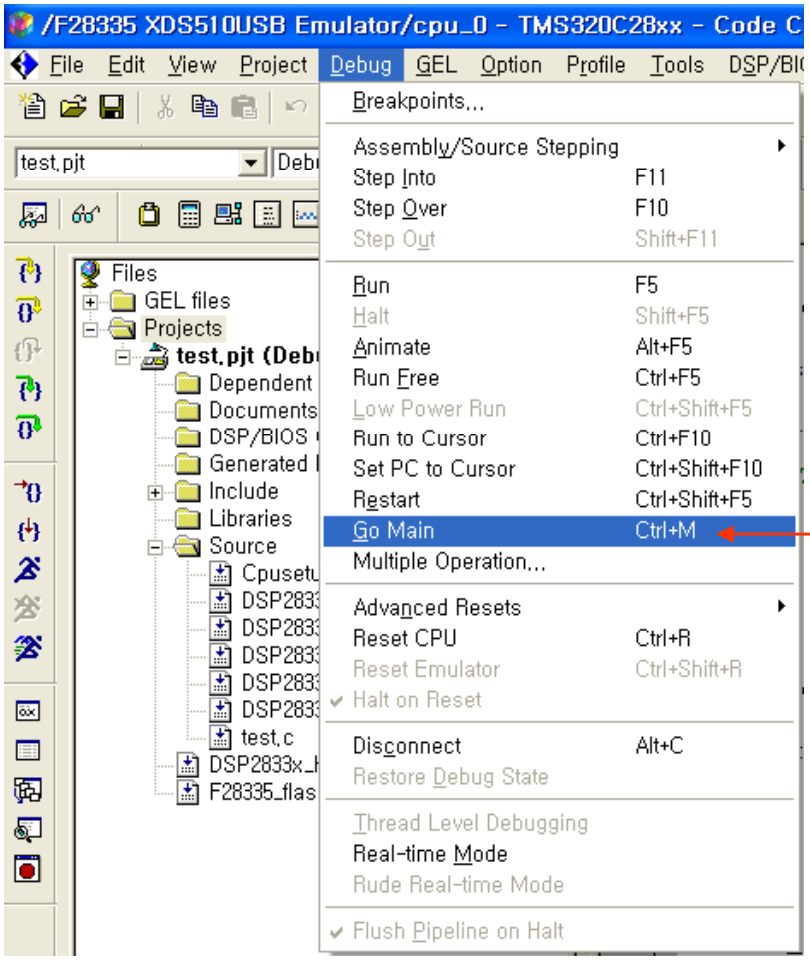


체크후 확인



닫음

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

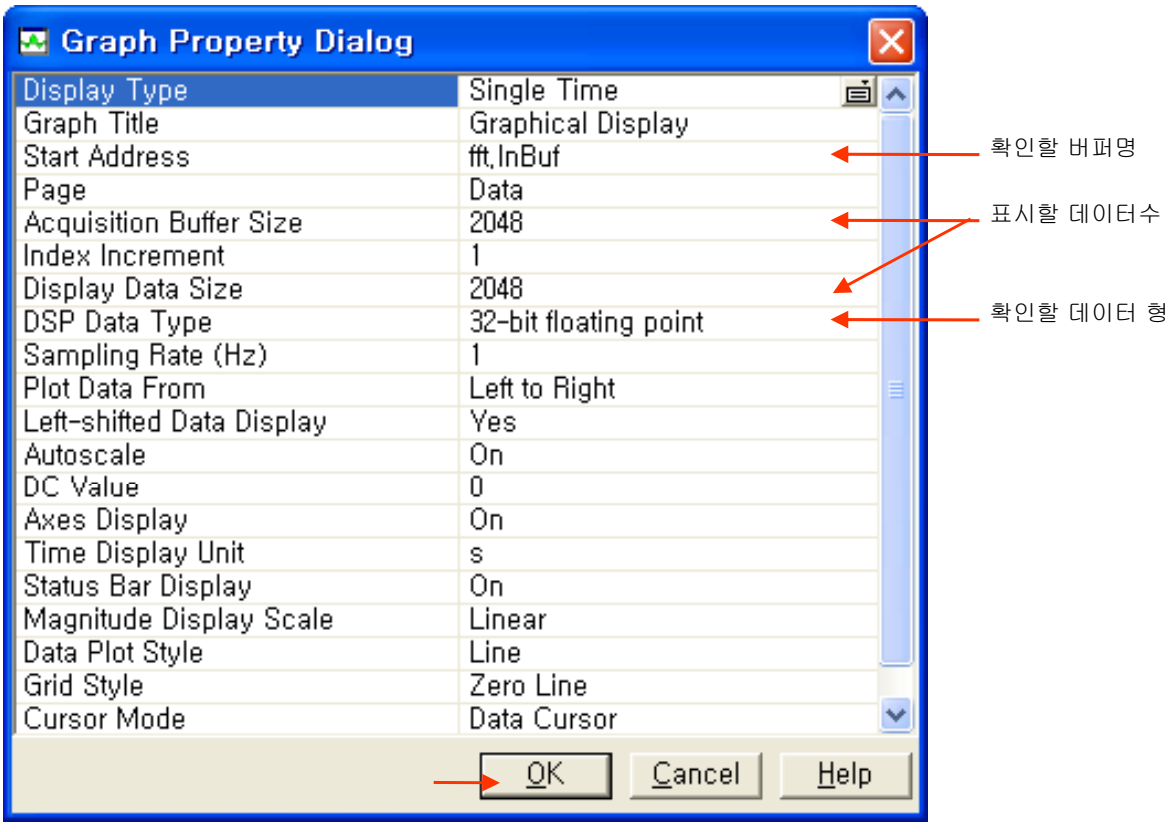


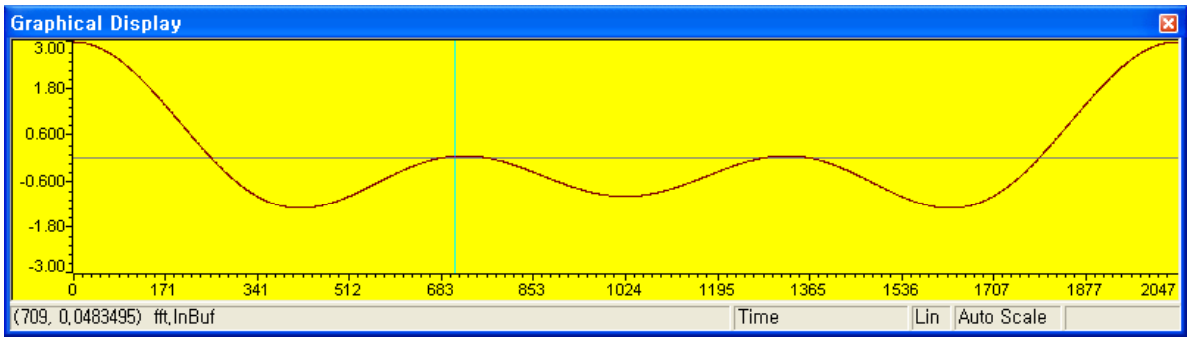

```
// 함수 선언

// 외부 변수 선언
extern unsigned char com1_recv_data[];              // COM1 수신 인터럽트 관련 fifo
extern int com1_rx_usi,com1_rx_udi;

// 변수 선언
unsigned int *EX_SRAM_ADDR = (unsigned int*)0x200000;       // SRAM 주소(ZCS7)

// FFT 관련
float32 InBuffer[FFT_SIZE];                 // TIME 파형 입력 버퍼
float32 OutBuffer[FFT_HALF_SIZE];             // FFT 결과 버퍼
float32 TwiddleBuffer[FFT_SIZE];              // FFT함수 참고용 테이블
float32 MagBuffer[FFT_ #define FFT_SIZE 2048          // FFT지원 라이브러리에서 FFT결과를 연산하여 저장 하는 버퍼
float32 Fft_ResultBuffer[FFT_HALF_SIZE];          // FFT 최종 연산 결과 저장
RFFT_F32_STRUCT fft;                        // FFT 사용 구조체

// ============================== 메인 프로그램 ==============================
// [인수]: void
// ---------------------------------------------------------------------------
void main(void)
{
unsigned char ch;
unsigned int i,k;
float theta,dataf,step_f1 = 1.0 / (float)FFT_SIZE;

    cpu_setup();                            // H/W 초기화 및 관리

// FFT 초기화
    fft.InBuf = InBuffer;                   // Input data buffer
    fft.OutBuf = OutBuffer;                 // FFT output buffer
    fft.CosSinBuf = TwiddleBuffer;            // Twiddle factor buffer
    fft.FFTSize = FFT_SIZE;                 // FFT length
    fft.FFTStages = FFT_STAGES;             // FFT Stages
    fft.MagBuf = MagBuffer;                 // Magnitude buffer

    theta = 2.0 * M_PI / (float)FFT_SIZE;           // signal creat(TIME 파형)
    for(i = k = 0; i <= FFT_SIZE; i++,k++){
        dataf = 1*cos(3*k*theta)+1*cos(2*k*theta)+1*cos(1*k*theta);
        fft.InBuf[i] = dataf;
     }
    RFFT_f32_sincostable(&fft);             // Initialize twiddle buffer
```

6. TIME 파형 생성 부분에 Break(F9 KEY)를 설정후 RUN(F5 키)시켜 파형을 확인 합니다.

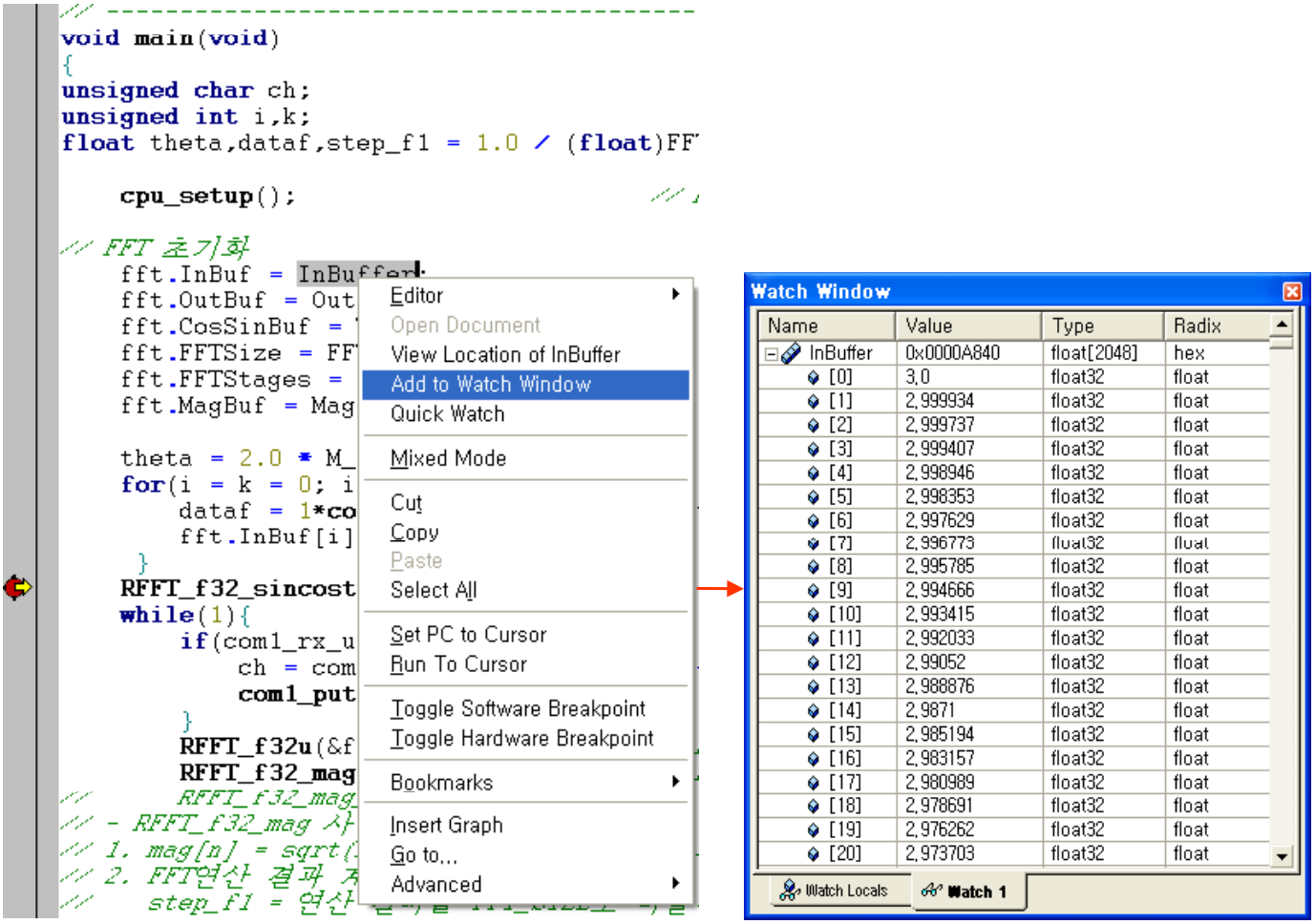
```
// FFT 관련
float32 InBuffer[FFT_SIZE];                 // TIME 파형 입력 버퍼
float32 OutBuffer[FFT_HALF_SIZE];               // FFT 결과 버퍼
float32 TwiddleBuffer[FFT_SIZE];                // FFT함수 참고용 테이블
float32 MagBuffer[FFT_HALF_SIZE];               // FFT지원 라이브러리에서 FFT결과를 연산하여 저장 하는 버퍼
float32 Fft_ResultBuffer[FFT_HALF_SIZE];            // FFT 최종 연산 결과 저장
RFFT_F32_STRUCT fft;                        // FFT 사용 구조체

// ============================== 메인 프로그램 ============================
// [인수]: void
// -------------------------------------------------------------------------
void main(void)
{
unsigned char ch;
unsigned int i,k;
float theta,dataf,step_f1 = 1.0 / (float)FFT_SIZE;

    cpu_setup();                        // H/W 초기화 및 관리

// FFT 초기화
    fft.InBuf = InBuffer;                   // Input data buffer
    fft.OutBuf = OutBuffer;                 // FFT output buffer
    fft.CosSinBuf = TwiddleBuffer;              // Twiddle factor buffer
    fft.FFTSize = FFT_SIZE;                 // FFT length
    fft.FFTStages = FFT_STAGES;             // FFT Stages
    fft.MagBuf = MagBuffer;                 // Magnitude buffer

    theta = 2.0 * M_PI / (float)FFT_SIZE;           // signal creat(TIME 파형)
    for(i = k = 0; i <= FFT_SIZE; i++,k++){
        dataf = 1*cos(3*k*theta)+1*cos(2*k*theta)+1*cos(1*k*theta);
        fft.InBuf[i] = dataf;
     }
    RFFT_f32_sincostable(&fft);             // Initialize twiddle buffer
    while(1){
        if(com1_rx_usi != com1_rx_udi){         // 수신 검사
            ch = com1_recv_data[com1_rx_udi++]; // ring 버퍼 데이터 로드
            com1_putch(ch);             // 데이터 송신
        }
        RFFT_f32u(&fft);                // FFT 함수
        RFFT_f32_mag(&fft);             // FFT 결과 연산 fft.MagBuf[n] = sqrt(real^2+image^2)
//      RFFT_f32_mag_non_sqrt(&fft);            // FFT 결과 연산 fft.MagBuf[n] = (r[n]^2+I^2)
// - RFFT_f32_mag 사용시
```

7. TIME 파형 확인(View->Graph->Freqency..)





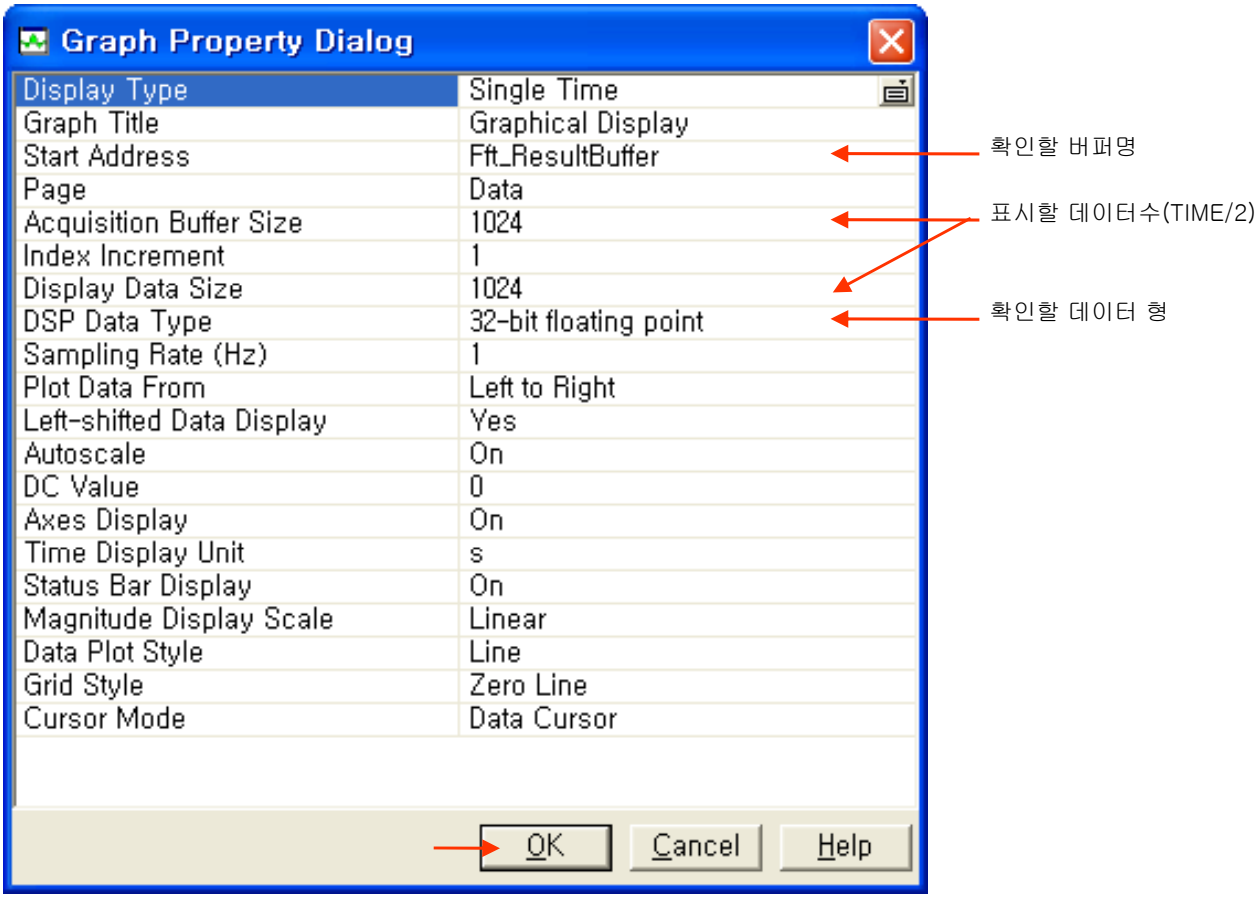
- 실제값을 확인(변수위에 오른쪽 마우스 누른후 -> Add to Watch Window)

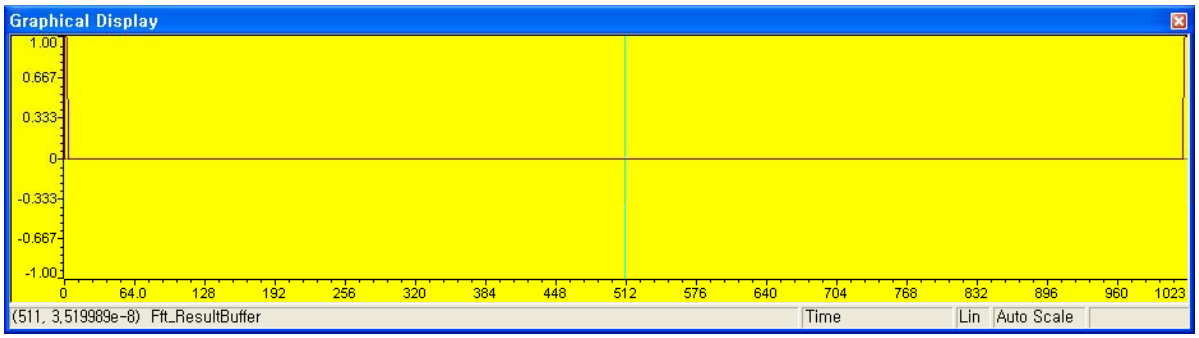


8. FFT VOLT 생성 부분에 Break(F9 KEY)를 설정후 RUN(F5 키)시켜 파형을 확인 합니다.

```
    while(1){
        if(com1_rx_usi != com1_rx_udi){          // 수신 검사
            ch = com1_recv_data[com1_rx_udi++]; // ring 버퍼 데이터 로드
            com1_putch(ch);                     // 데이터 송신
        }
        RFFT_f32u(&fft);                        // FFT 함수
        RFFT_f32_mag(&fft);                     // FFT 결과 연산 fft.MagBuf[n] = sqrt(real^2+image^2)
//      RFFT_f32_mag_non_sqrt(&fft);            // FFT 결과 연산 fft.MagBuf[n] = (r[n]^2+I^2)
// - RFFT_f32_mag 사용시
// 1. mag[n] = sqrt(real^2+image^2)기 계산 되어 나오므로
// 2. FFT연산 결과 계산된 값에 FFT_SIZE만큼 값이 더해지고, 한쪽면값만 나오므로  (연산결과 / FFT_SIZE) * 2.0을 해야함, 단 DC는
//    step_f1 = 연산 결과를 FFT_SIZE로 나눌때 연산 시간이 걸리므로 역수를 취해 곱함

// - 주파수별 VOLT값
// Fft_ResultBuffer[0] : DC값
// Fft_ResultBuffer[1] ~ [FFT_HALF_SIZE] : 각 주파수별 VOLT값
// - 주파수 구하는법
// Fft_ResultBuffer[0] : 0HZ
// Fft_ResultBuffer[1] ~ [FFT_HALF_SIZE] : (A/D SAMPLE 주파수 / FFT_SIZE) * 각 배열 포인터
        for(i = 0; i < FFT_HALF_SIZE; i++){     // FFT 결과 VOLT 계산
// RFFT_f32_mag() 사용시
            if(i == 0) Fft_ResultBuffer[i] = (fft.MagBuf[i] * step_f1);// result[DC] = (mag[n] / FFT_SIZE)
            else Fft_ResultBuffer[i] = (fft.MagBuf[i] * step_f1) * 2.0;// result[V] = (mag[n] / FFT_SIZE) * 2.0
// RFFT_f32_mag_non_sqrt() 사용시
//          if(i == 0) Fft_ResultBuffer[i] = (sqrt(fft.MagBuf[i]) * step_f1);// result[DC] = sqrt(mag[n] / FFT_SIZE)
//          else Fft_ResultBuffer[i] = (sqrt(fft.MagBuf[i]) * step_f1) * 2.0;// result[V] = sqrt(mag[n] / FFT_SIZE) * 2.0
        }
        run_cnt++;                  // RUN-Count
    }
}
```

9. FFT VOLT 파형 확인(View->Graph->Freqency..)





- 실제값을 확인(변수위에 오른쪽 마우스 누른후 -> Add to Watch Window)