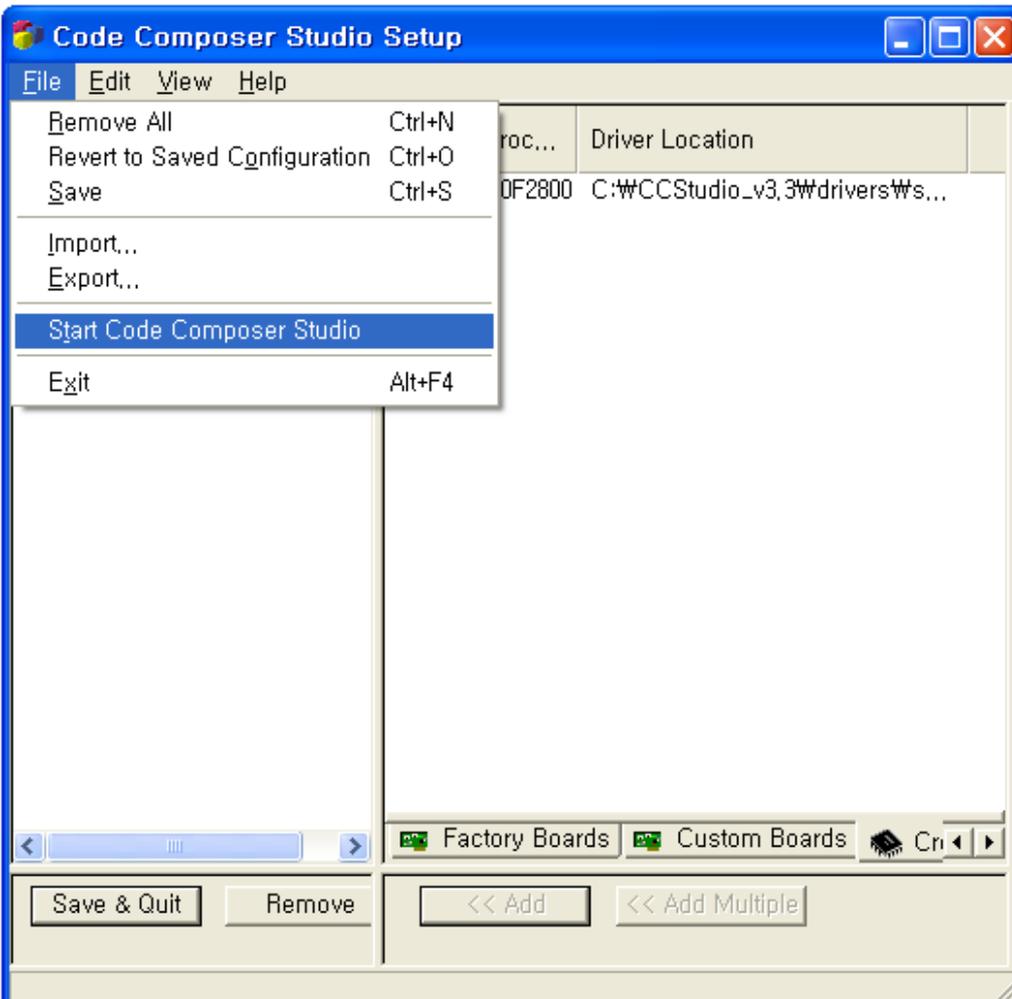


5. My System에서 F283335 XDS510USB Emu를 선택후 Start Code Composer Studio를 실행 합니다.

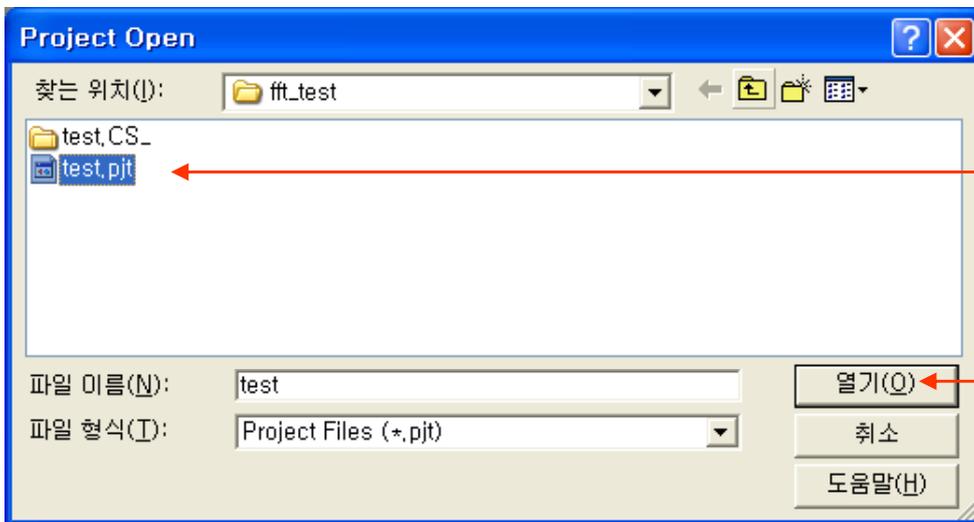
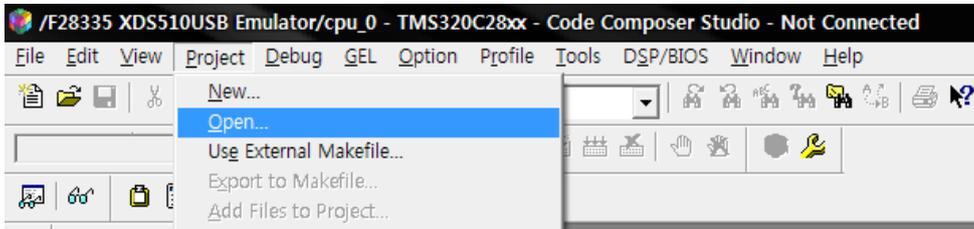


* CCS3.3 DSP FFT Project 시작

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



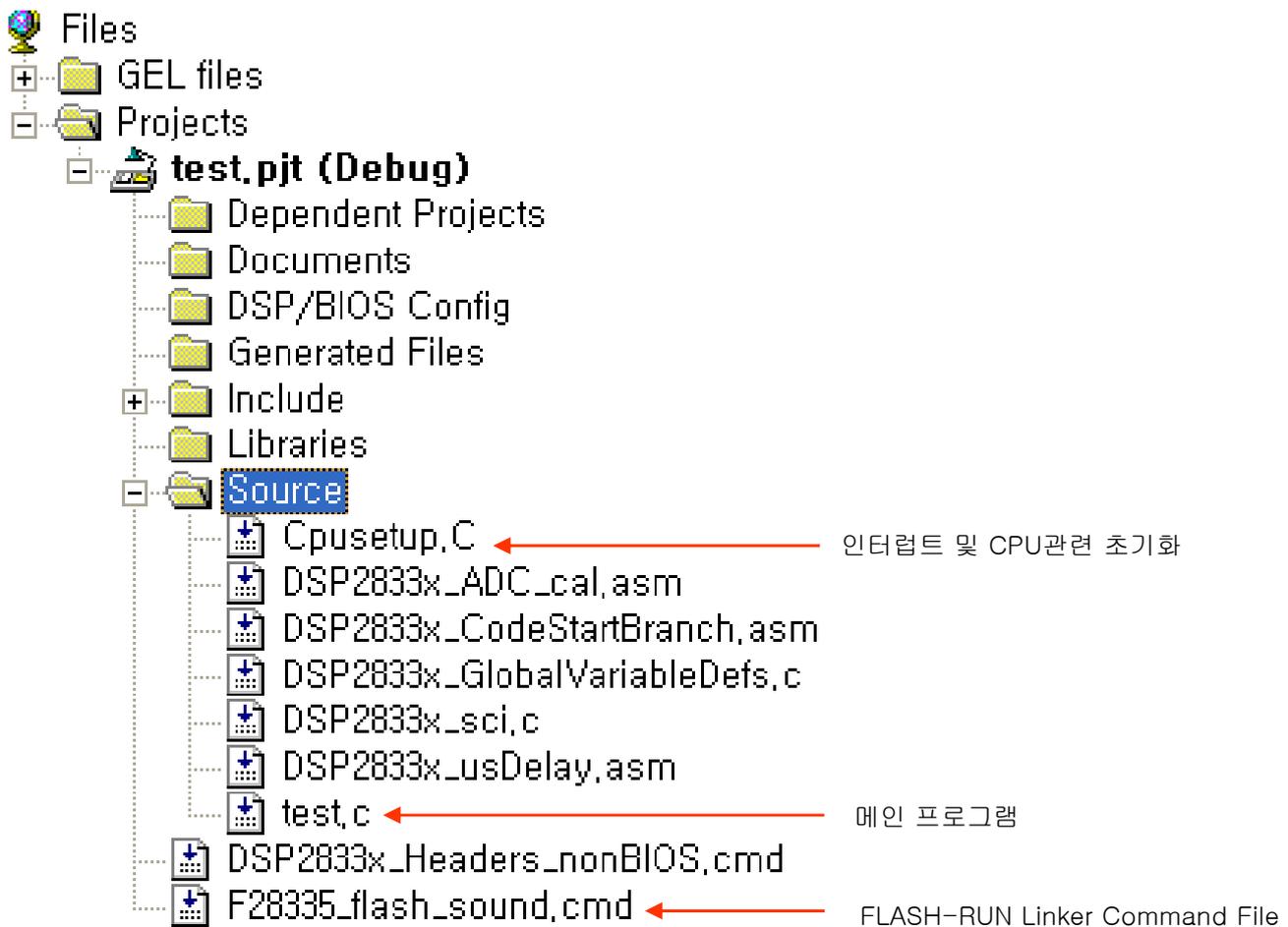
2. 아래와 같이 Project를 오픈 합니다.(Project->Open)



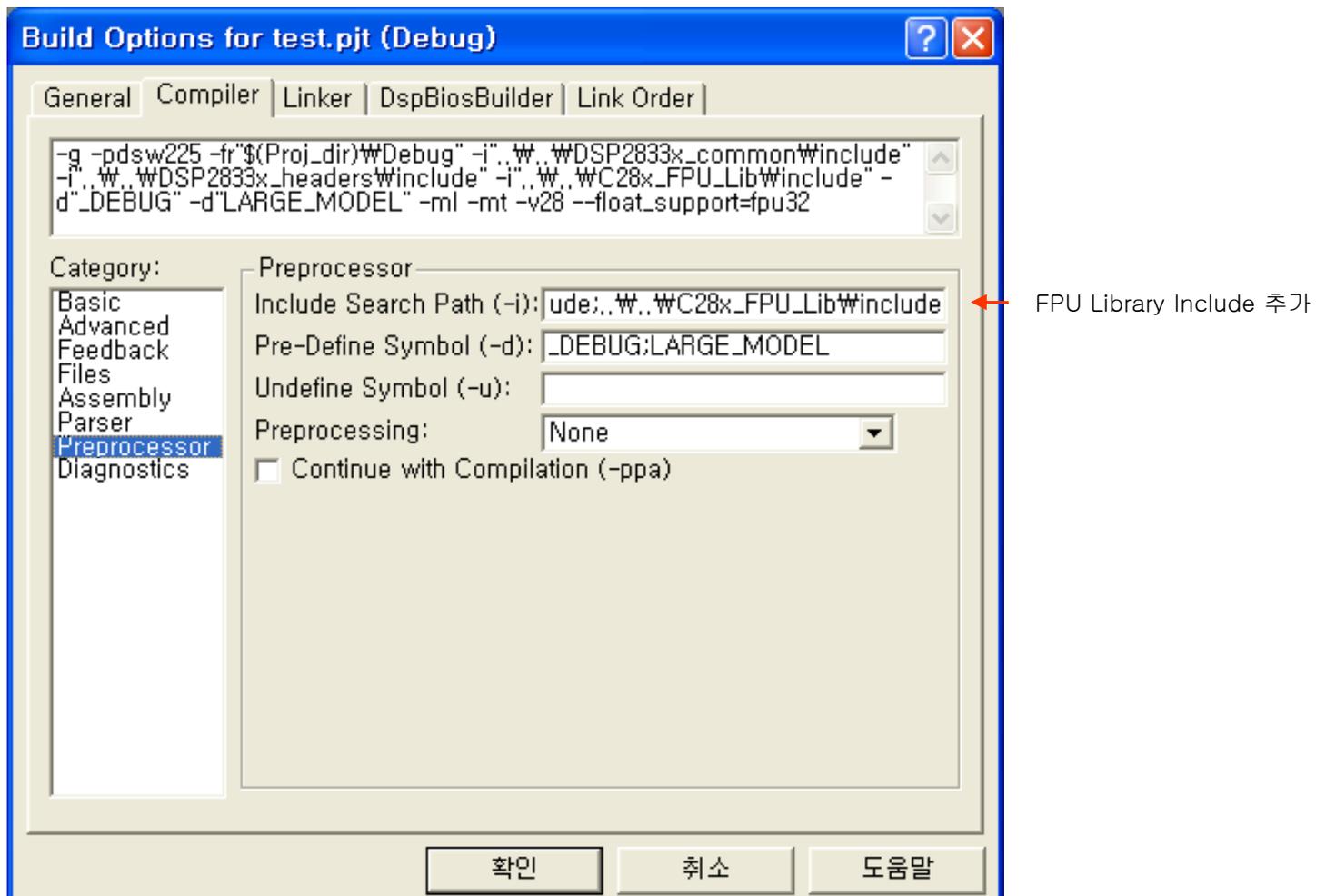
FLASH에서 실행되는
프로젝트

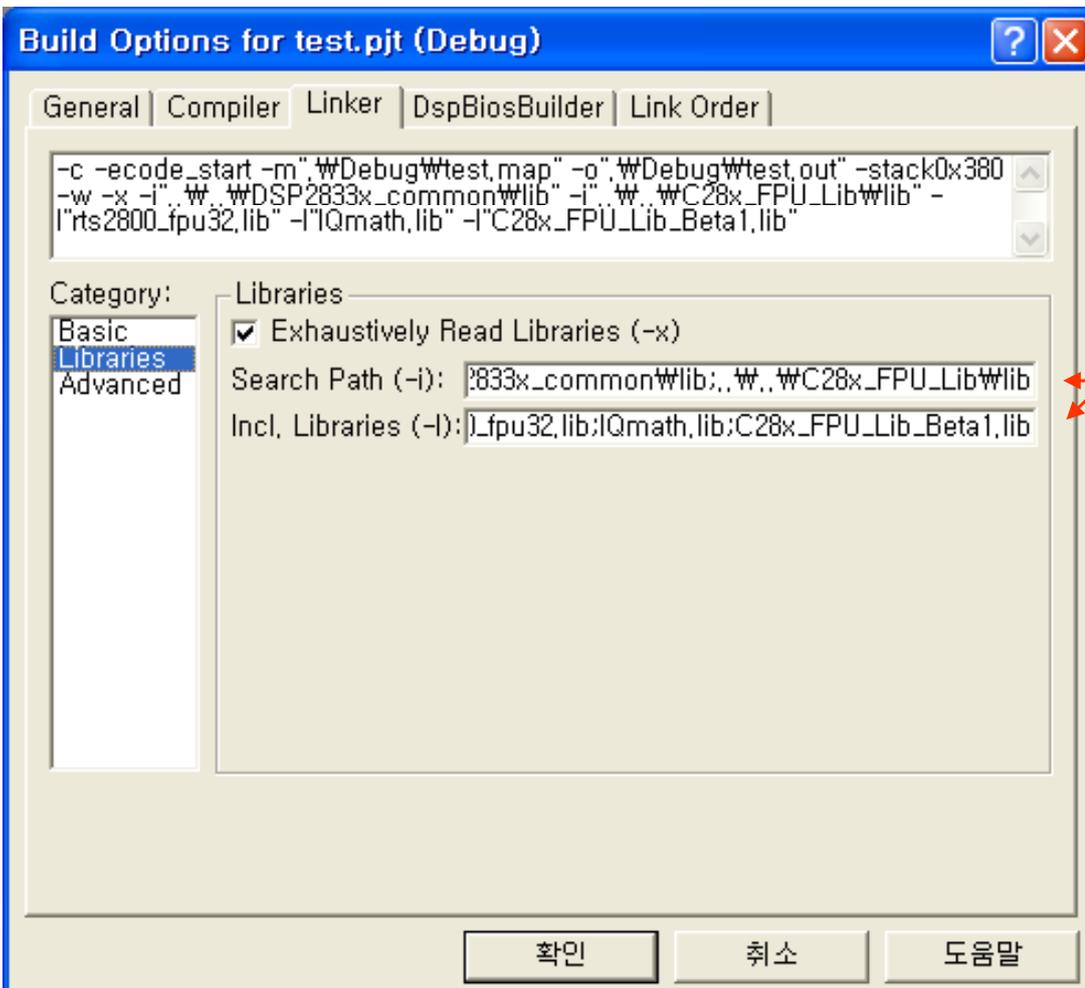
버튼 클릭

3. Projects Source 파일 구성



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

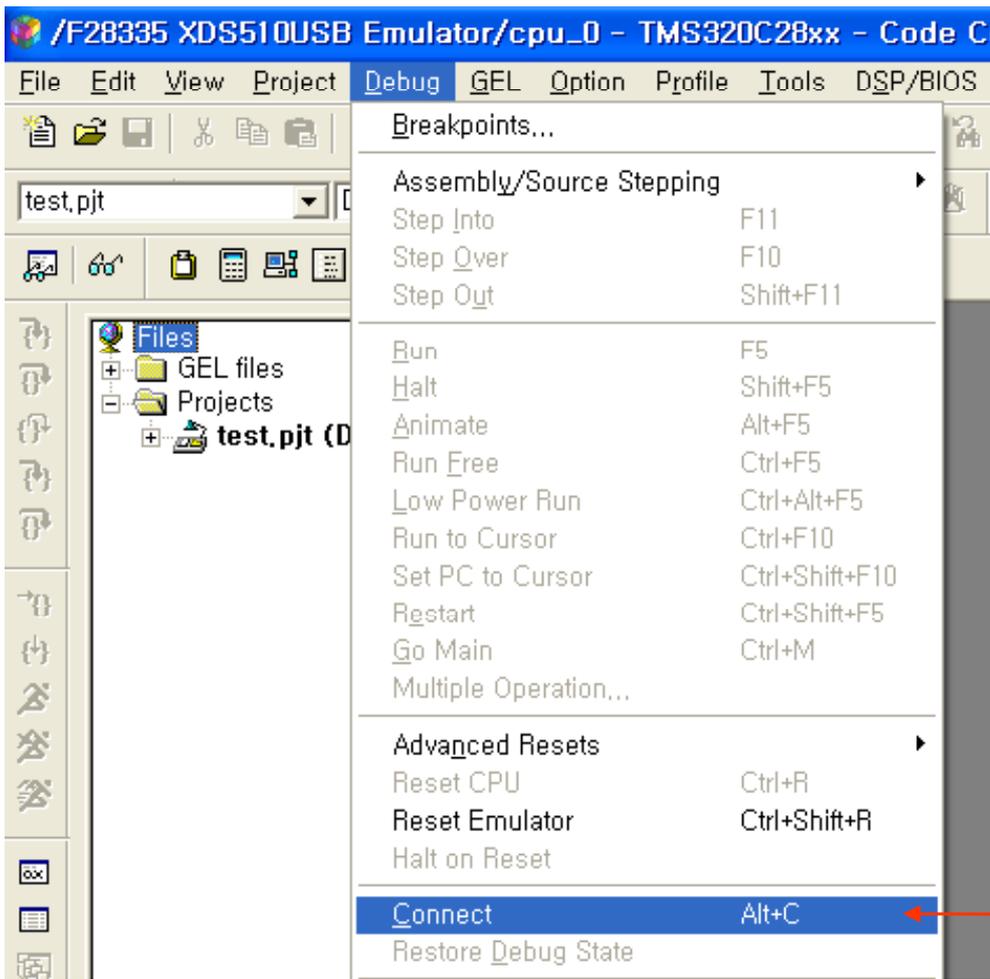




FPU Library 추가

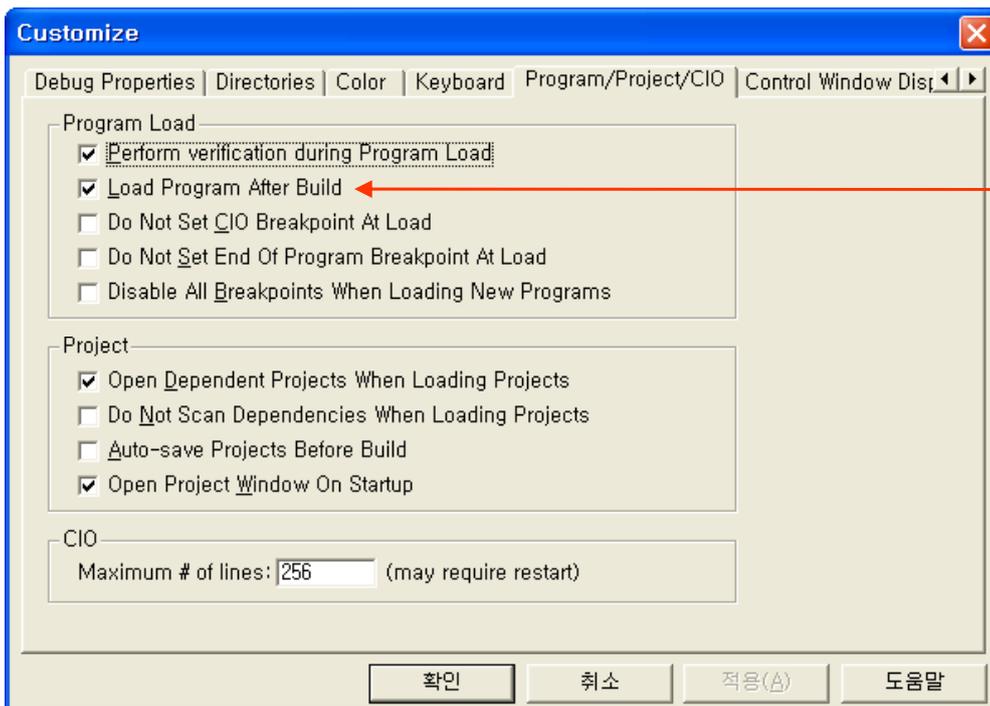
- CCS3.3 DSP Program 실행

1. JTAG 및 에뮬레이터를 연결 합니다.



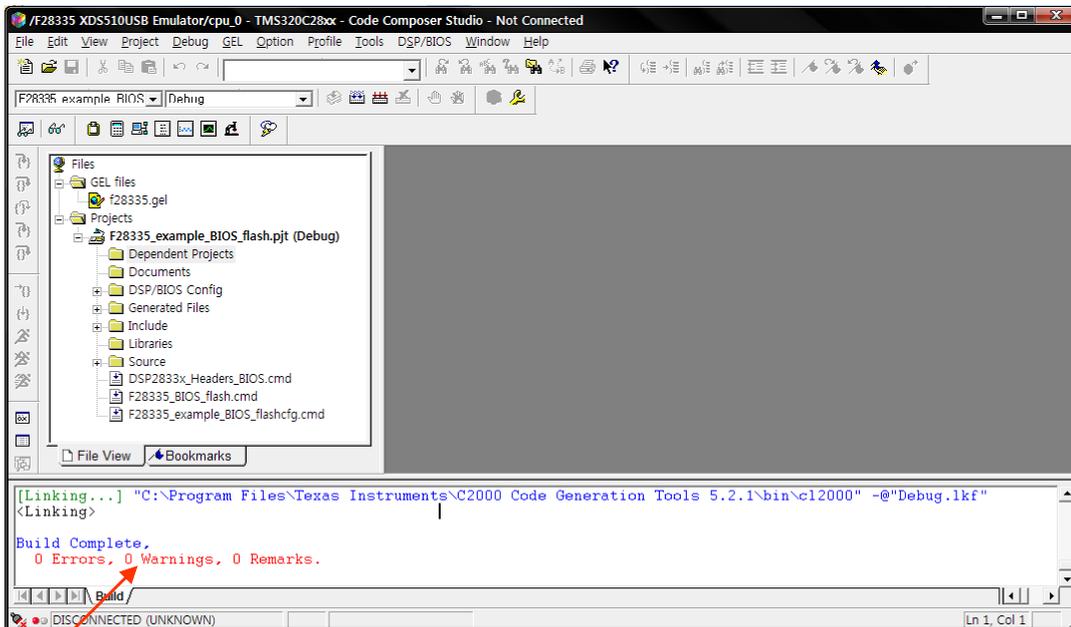
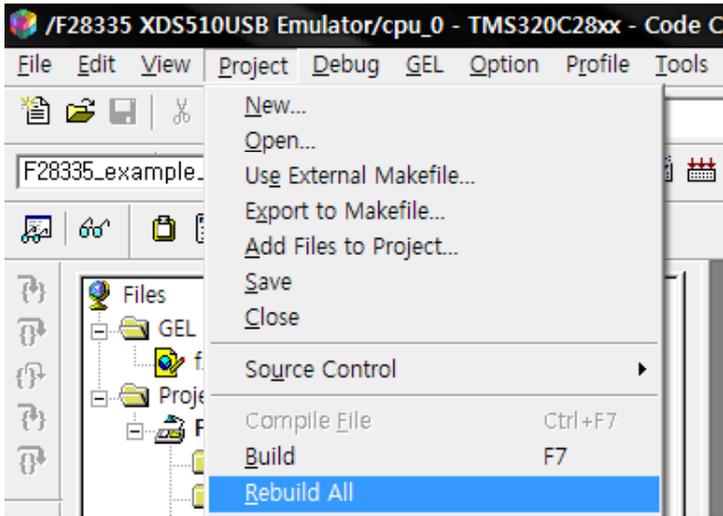
선택후
마우스 왼쪽 버튼 클릭

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



체크

3. 컴파일 하기(Project->Rebuild All)

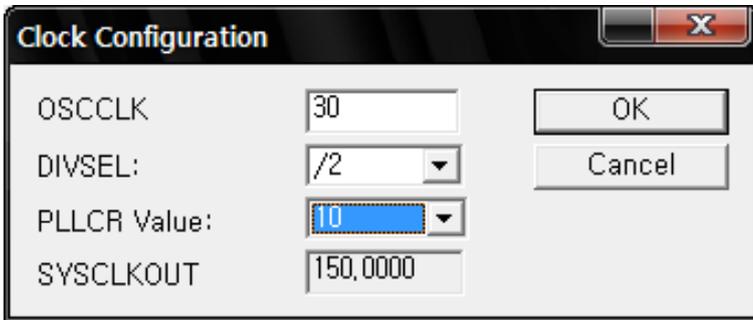


에러 확인

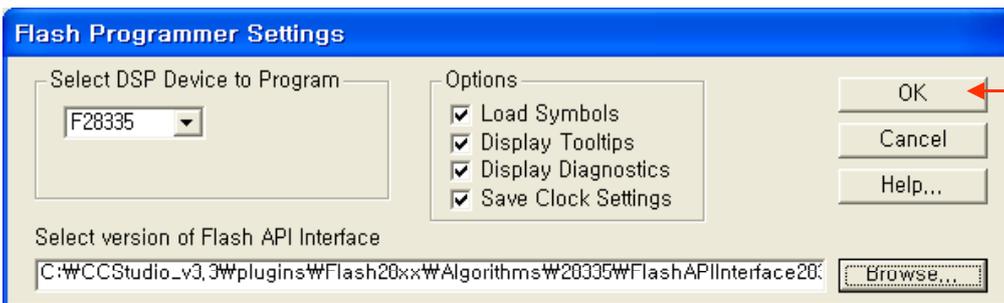
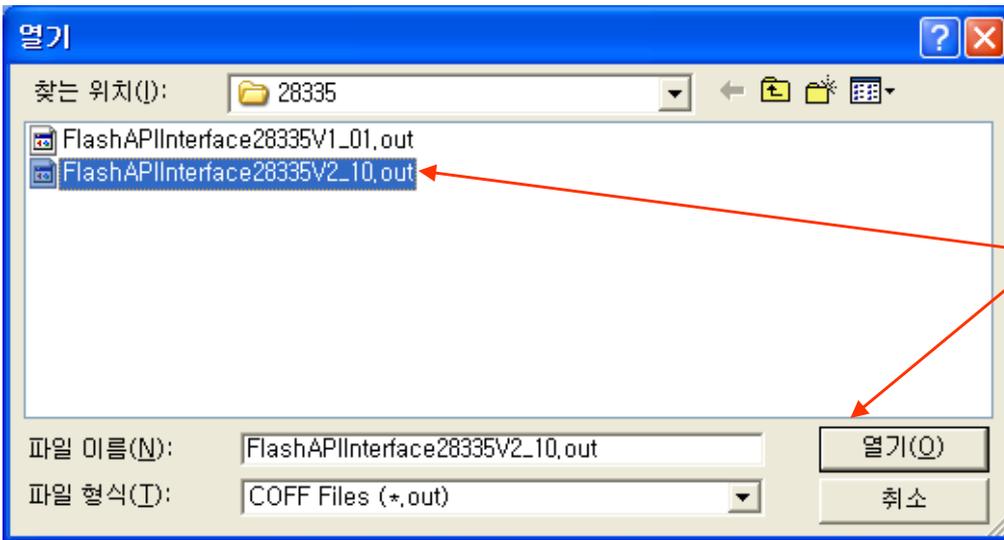
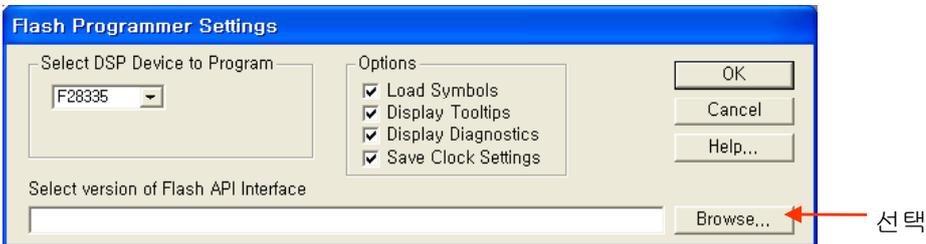
4. FLASH에 프로그램 하기

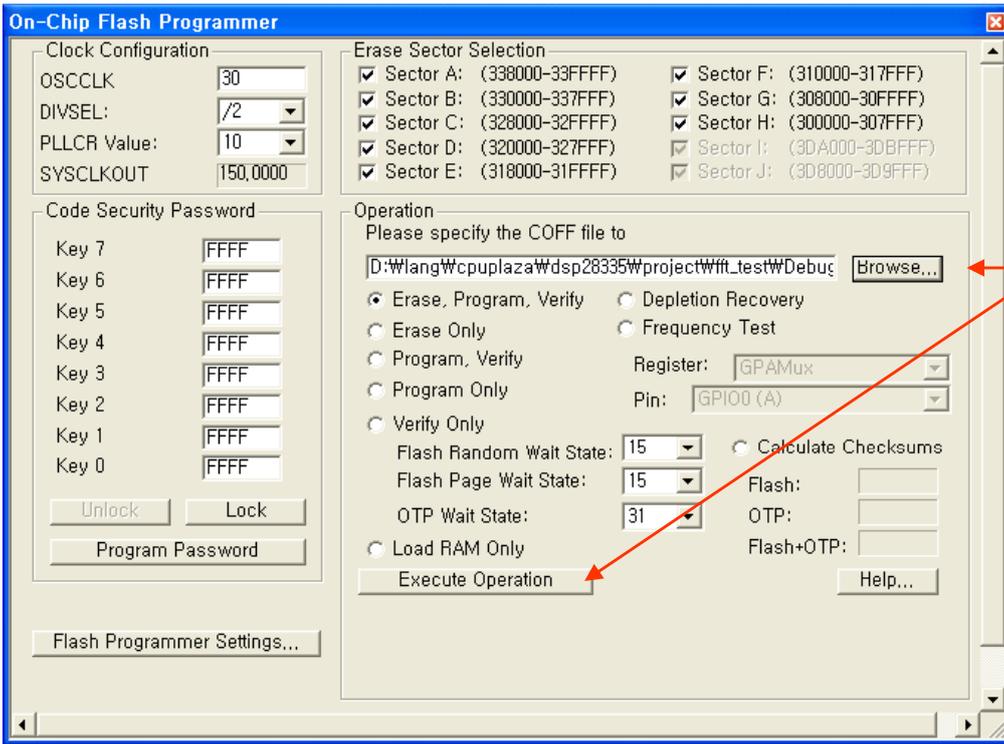


* 아래 CLOCK 설정 메뉴를 사용자에게 맞게 설정 합니다.



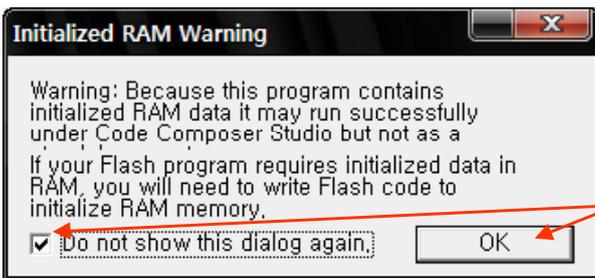
* API Interface 파일을 등록 합니다.



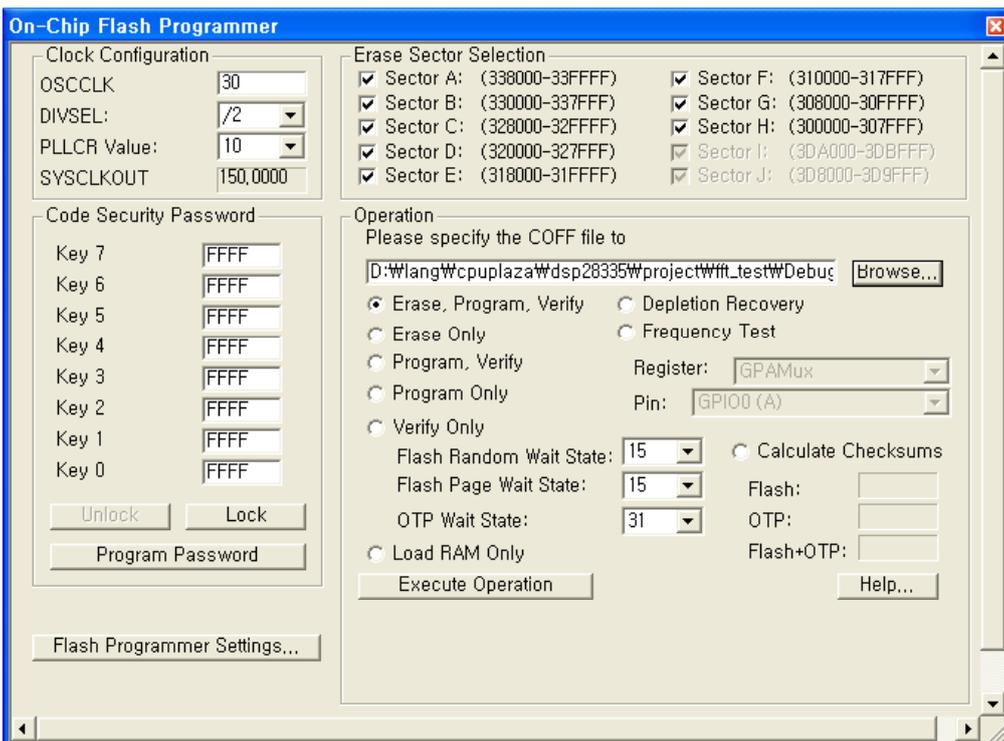


Browse.. 창에서 파일을 선택후 Execute Operation 탭을 실행합니다.

* TI 실행 파일은 *.OUT로 현재 작업 디렉토리 ..WdebugW 에 있습니다.

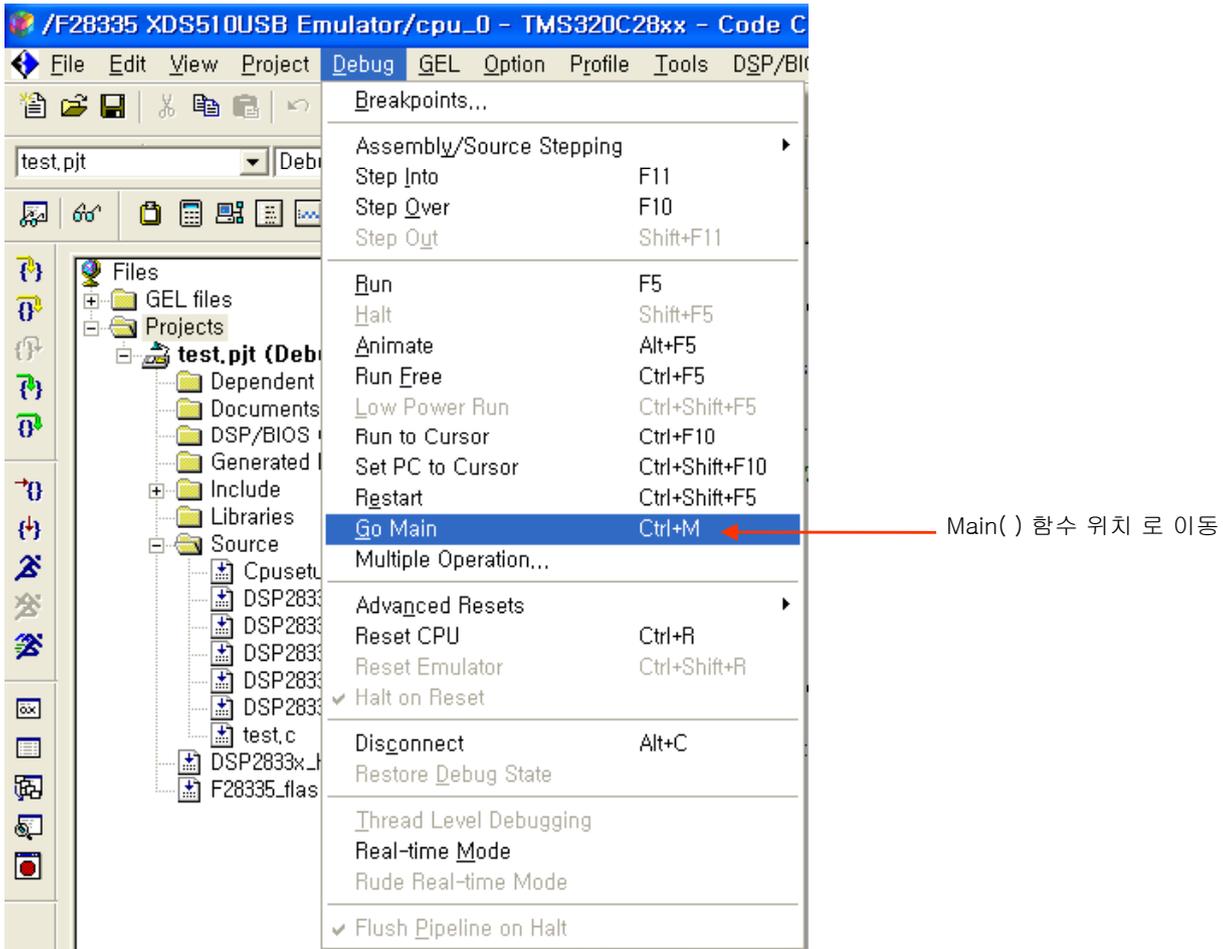


체크후 확인



다음

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



```

// 함수 선언
// 외부 변수 선언
extern unsigned char com1_rcv_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_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
}

```

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]; // FFT지인 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

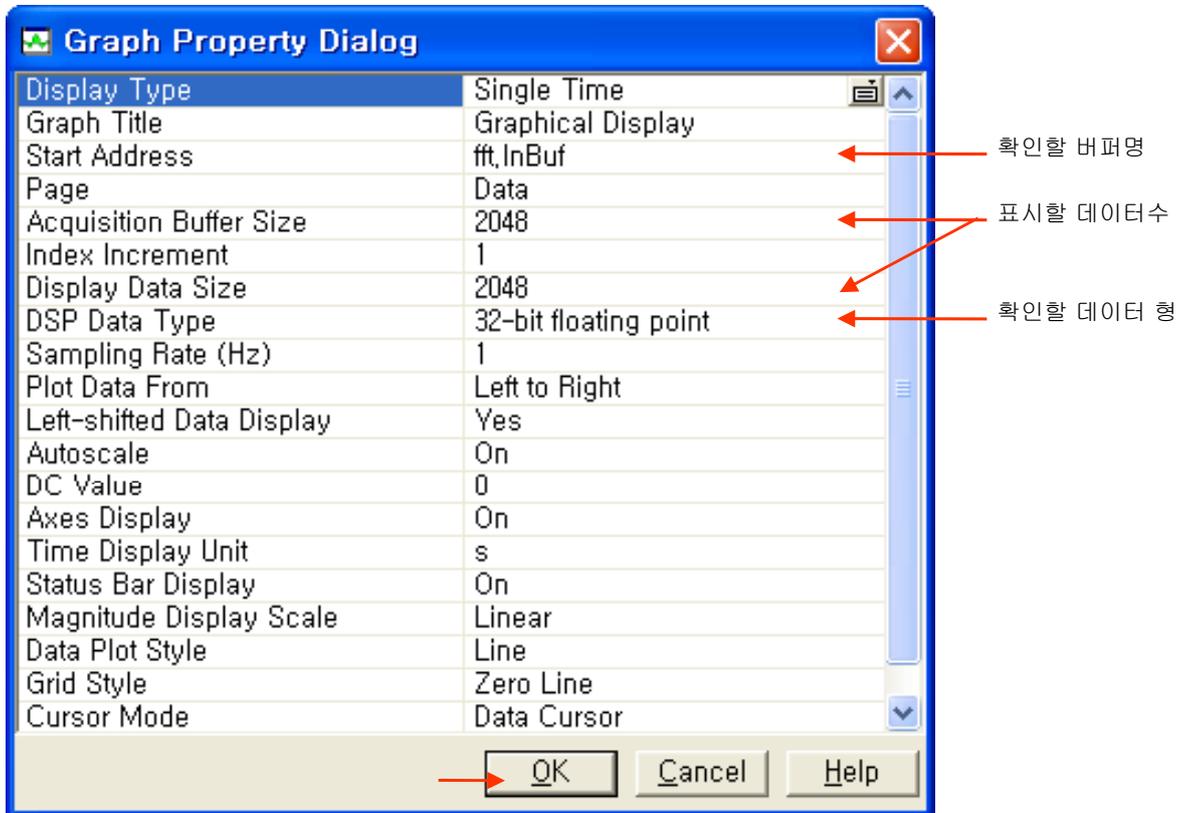
    // FFT 초기화
    fft.InBuf = InBuffer; // Input data buffer
    fft.OutBuf = OutBuffer; // FFT output buffer
    fft.CosSinBuf = TwiddleBuffer; // Twiddle factor buffer
    fft.FFTSize = FFT_SIZE; // FFT length
    fft.FFTStages = FFT_STAGES; // FFT Stages
    fft.MagBuf = MagBuffer; // Magnitude buffer

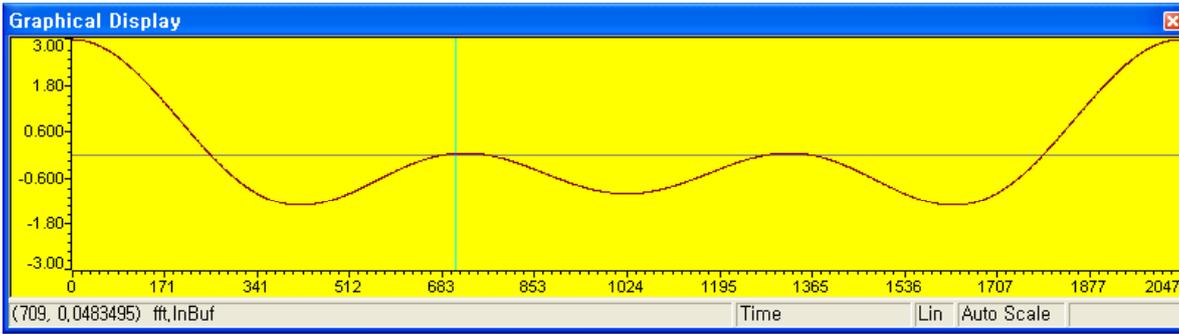
    theta = 2.0 * M_PI / (float)FFT_SIZE; // signal creat(TIME 파형)
    for(i = k = 0; i <= FFT_SIZE; i++,k++){
        dataf = 1*cos(3*k*theta)+1*cos(2*k*theta)+1*cos(1*k*theta);
        fft.InBuf[i] = dataf;
    }

    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->Frequency..)





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

```

void main(void)
{
    unsigned char ch;
    unsigned int i,k;
    float theta,dataf,step_f1 = 1.0 / (float)FFT

    cpu_setup();

    // FFT 초기화
    fft.InBuf = InBuffer;
    fft.OutBuf = OutBuf;
    fft.CosSinBuf = FFTCosSinBuf;
    fft.FFTSize = FFTSize;
    fft.FFTStages = FFTStages;
    fft.MagBuf = MagBuf;

    theta = 2.0 * M_PI;
    for(i = k = 0; i < FFTSize; i++)
        dataf = 1*cos(theta*i);
        fft.InBuf[i] = dataf;

    RFFT_f32_sincost(fft);
    while(1){
        if(com1_rx_uart){
            ch = com1_rx_uart;
            com1_put_uart(ch);
        }
        RFFT_f32u(&fft);
        RFFT_f32_mag(fft);
        // RFFT_f32_mag 사
        // 1. mag[n] = sqrt(fft.MagBuf[n]*fft.MagBuf[n]);
        // 2. FFT연산 결과 차
        // step_f1 = 연산 결과 * step_f1;
    }
}
    
```

Name	Value	Type	Radix
InBuffer	0x0000A840	float[2048]	hex
[0]	3,0	float32	float
[1]	2,999934	float32	float
[2]	2,999737	float32	float
[3]	2,999407	float32	float
[4]	2,998946	float32	float
[5]	2,998353	float32	float
[6]	2,997629	float32	float
[7]	2,996773	float32	float
[8]	2,995785	float32	float
[9]	2,994666	float32	float
[10]	2,993415	float32	float
[11]	2,992033	float32	float
[12]	2,99052	float32	float
[13]	2,988876	float32	float
[14]	2,9871	float32	float
[15]	2,985194	float32	float
[16]	2,983157	float32	float
[17]	2,980989	float32	float
[18]	2,978691	float32	float
[19]	2,976262	float32	float
[20]	2,973703	float32	float

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

```

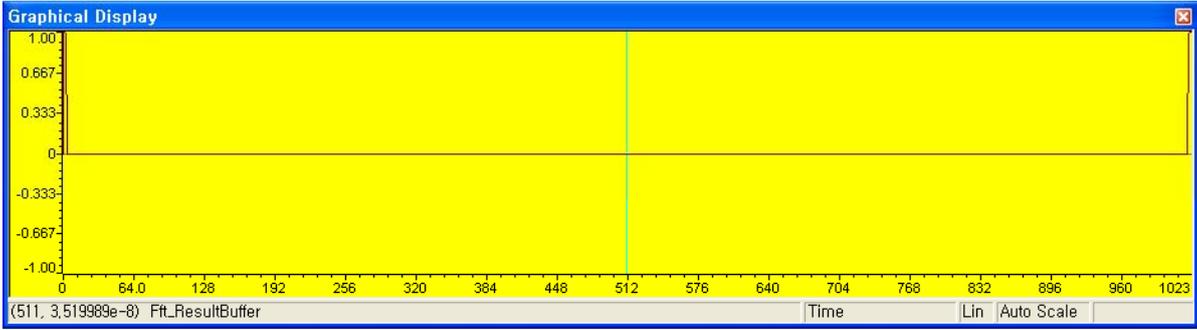
while(1){
  if(com1_rx_usi != com1_rx_udi){ // 수신 검사
    ch = com1_recv_data[com1_rx_udi++]; // ring 버퍼 데이터 로드
    com1_putch(ch); // 데이터 송신
  }
  RFFT_F32u(&fft); // FFT 함수
  RFFT_F32_mag(&fft); // FFT 결과 연산 fft.MagBuf[n] = sqrt(real^2+image^2)
  RFFT_F32_mag_non_sqrt(&fft); // FFT 결과 연산 fft.MagBuf[n] = (r[n]^2+I^2)
  // - RFFT_F32_mag 사용자
  // 1. mag[n] = sqrt(real^2+image^2)가 계산 되어 나오므로
  // 2. FFT연산 결과 계산된 값이 FFT_SIZE만큼 값이 더해지고, 한쪽면값만 나오므로 (연산결과 / FFT_SIZE) * 2.0을 해야함, 단 DC
  // 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] : OHZ
  // 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() 사용자
    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() 사용자
  // 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->Frequency..)

Property	Value
Display Type	Single Time
Graph Title	Graphical Display
Start Address	Fft_ResultBuffer
Page	Data
Acquisition Buffer Size	1024
Index Increment	1
Display Data Size	1024
DSP Data Type	32-bit floating point
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On
Magnitude Display Scale	Linear
Data Plot Style	Line
Grid Style	Zero Line
Cursor Mode	Data Cursor



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

```

// - 주파수별 VOLT값
// Fft_ResultBuffer[0] : DC값
// Fft_ResultBuffer[1] - [FFT_HALF_SIZE] : 각 주파수별 VOL
// - 주파수 구하는법
// Fft_ResultBuffer[0] : OHZ
// Fft_ResultBuffer[1] - [FFT_HALF_SIZE] : (A/D SAMPLE 주파
for(i = 0; i < FFT_HALF_SIZE; i++){ // FFT 결과
// RFFT_F32_mag() 사용시
if(i == 0) Fft_Result
else Fft_ResultBuffer
// RFFT_F32_mag_non_sqrt() 사용시
if(i == 0) Fft_Result
else Fft_ResultBuffer
}
run_cnt++;
}

```



Name	Value	Type	Radix
Fft_ResultBuffer	0x00009040	float[1024]	hex
[0]	7.962808e-08	float32	float
[1]	1.0	float32	float
[2]	1.0	float32	float
[3]	1.0	float32	float
[4]	9.133088e-08	float32	float
[5]	3.522608e-08	float32	float
[6]	4.626166e-08	float32	float
[7]	2.470452e-08	float32	float
[8]	1.053591e-08	float32	float
[9]	2.107226e-08	float32	float
[10]	8.119166e-09	float32	float
[11]	2.039932e-08	float32	float
[12]	3.58206e-09	float32	float
[13]	1.30983e-08	float32	float
[14]	8.633223e-09	float32	float
[15]	1.73171e-08	float32	float
[16]	8.678688e-09	float32	float
[17]	1.461737e-08	float32	float
[18]	1.84083e-08	float32	float
[19]	1.254632e-08	float32	float
[20]	3.208877e-09	float32	float
[21]	4.246763e-09	float32	float
[22]	2.992204e-09	float32	float