Controller Tester User Guides

3.3 — Last update: Oct 16, 2020

Suresofttech

Table of Contents

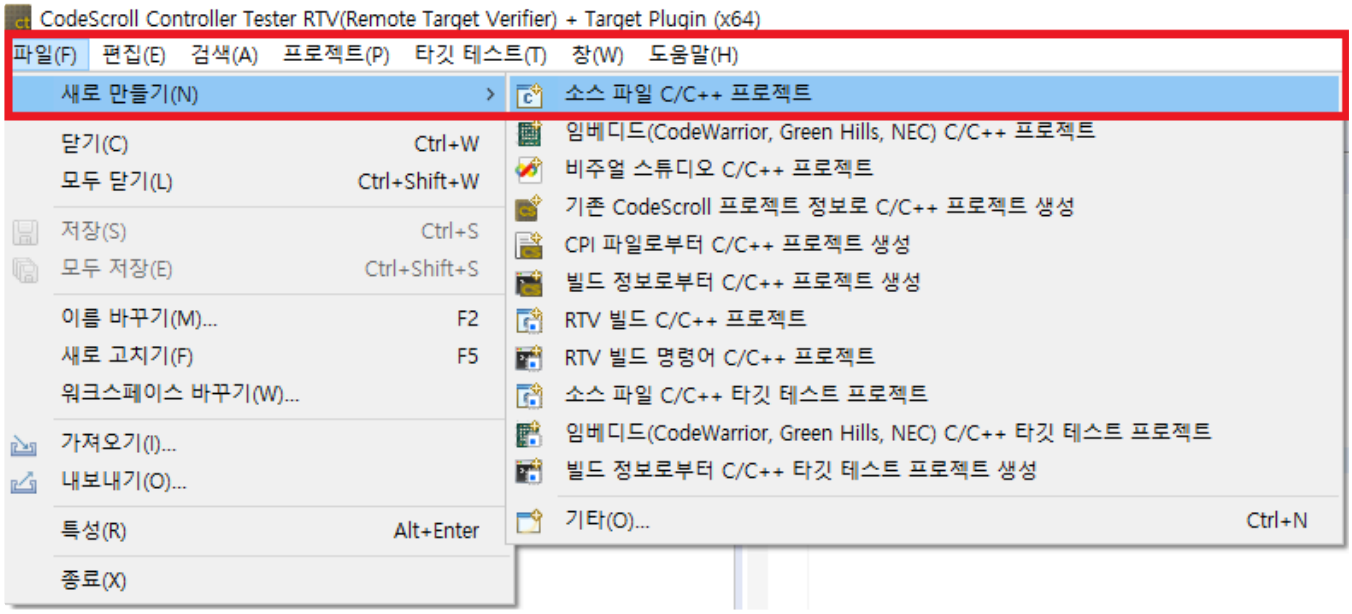
1. Controller Tester 타깃 테스트 가이드

CodeScroll Controller Tester를 사용하여 타깃을 테스트 하는 시나리오를 설명합니다.

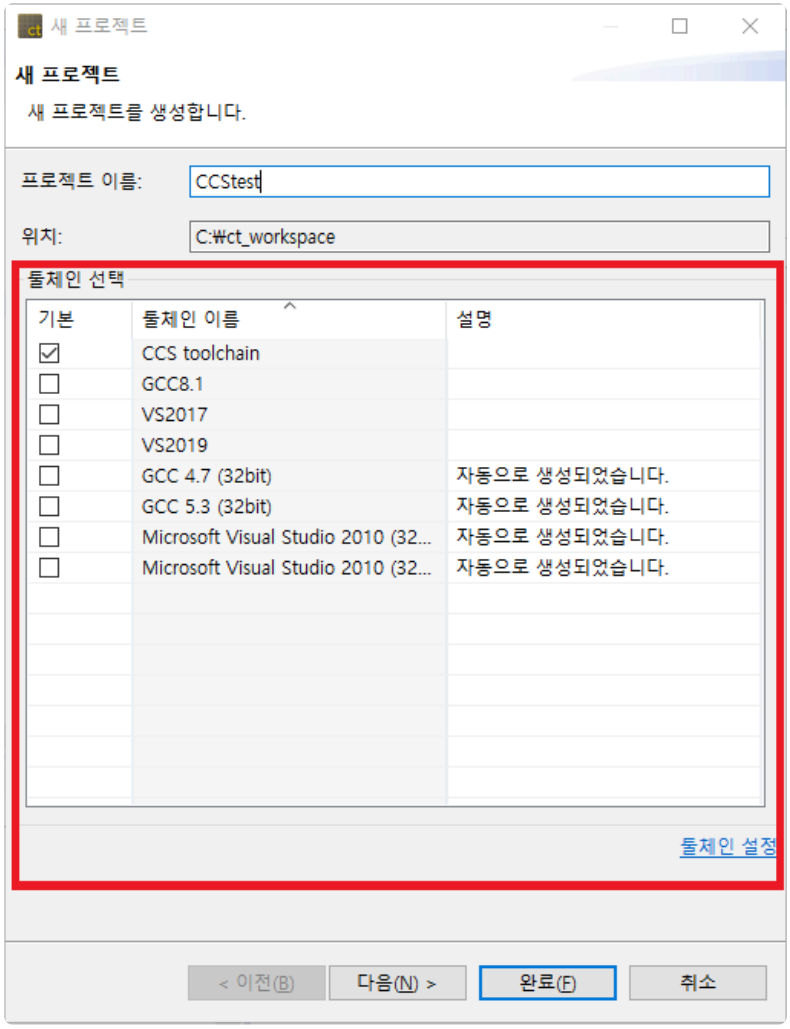
Texas Instruments Code Composer Studio

1.1. Texas Instruments Code Composer Studio

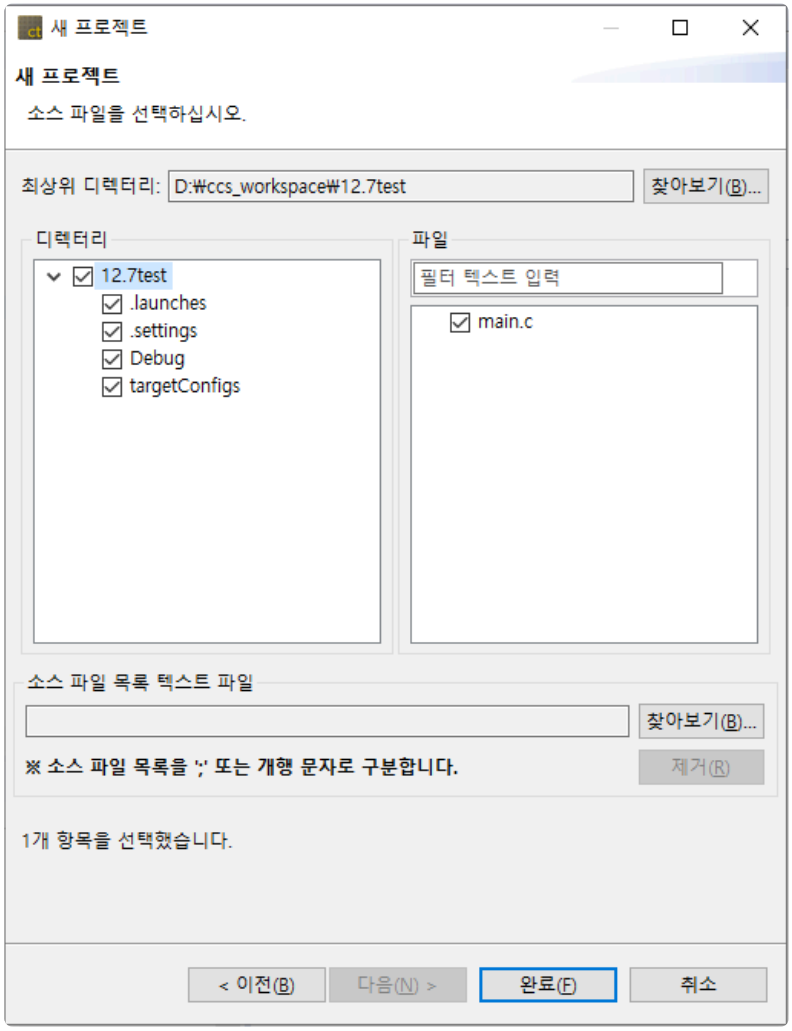
1-1. CodeScroll Controller Tester 프로젝트를 생성합니다.



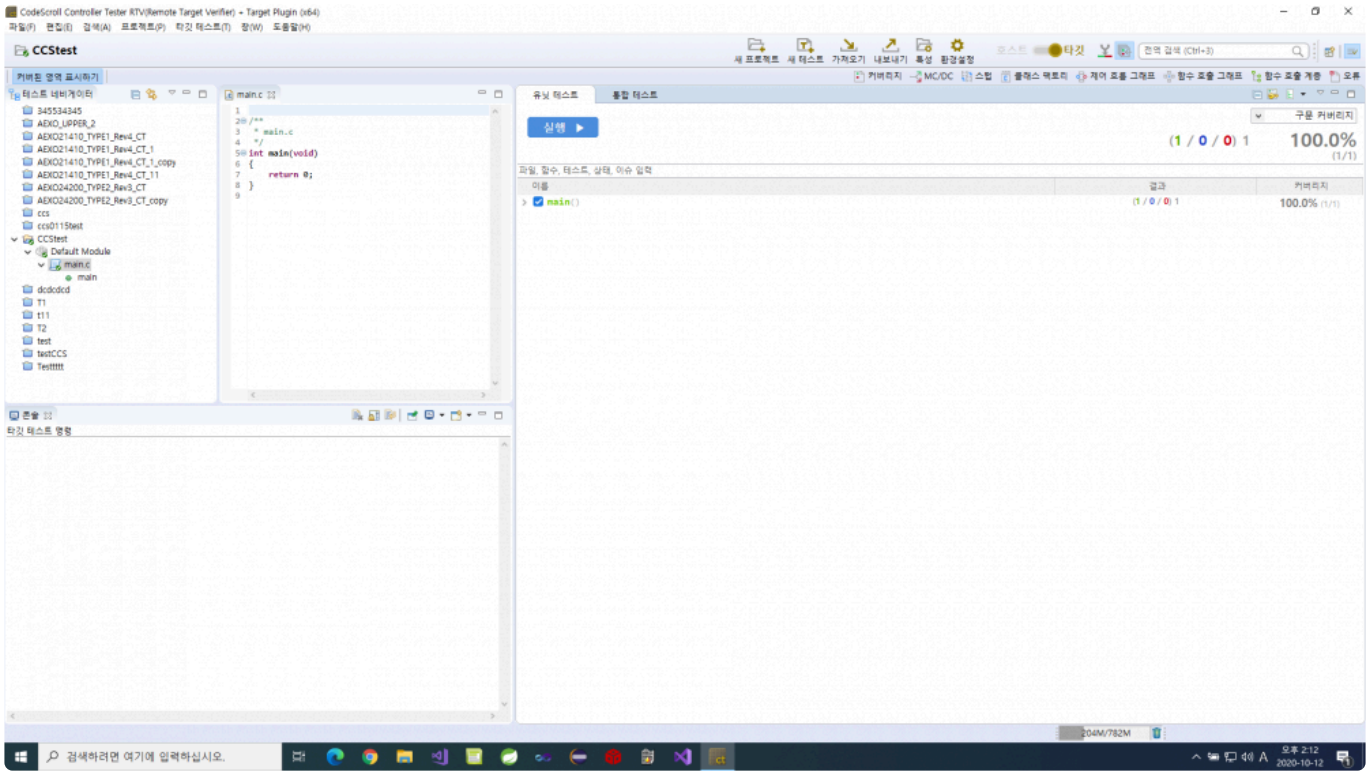
1-2. 생성한 CCS toolchain을 선택합니다.



1-3. 테스트할 소스 파일을 선택합니다.



해당 설정을 모두 마치고 완료를 클릭하면 프로젝트가 생성됩니다.



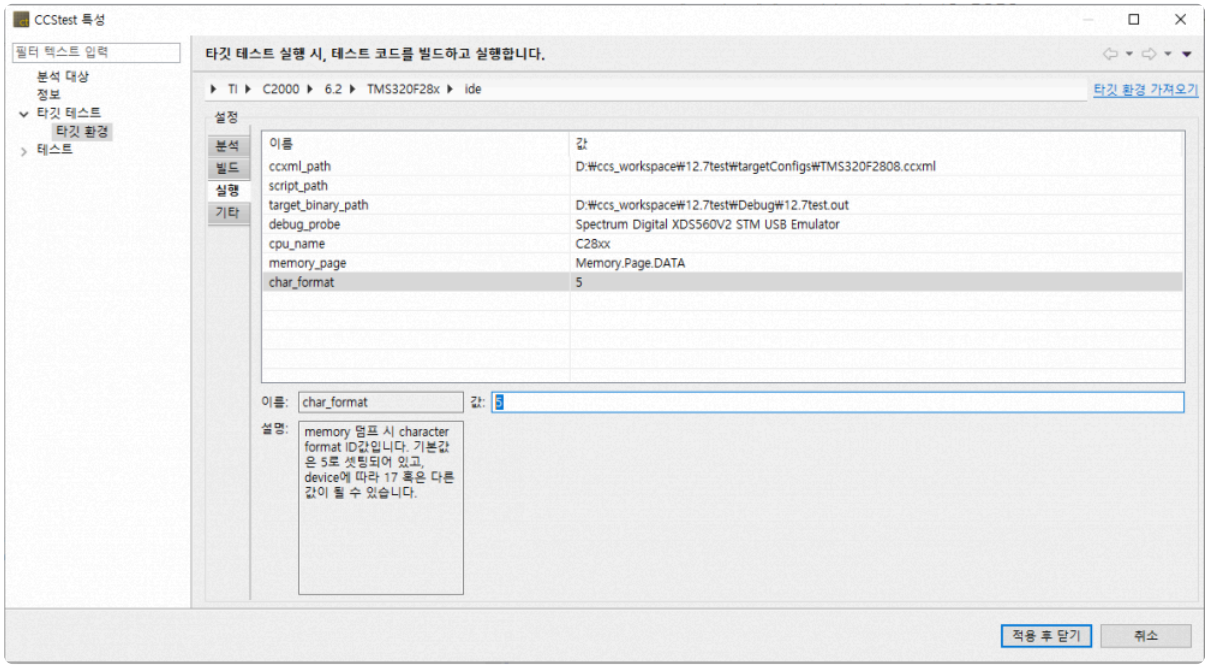
2. 디버거를 사용하기 위해 CCS, CodeScroll Controller Tester 에서 설정을 합니다.
[Texas Instruments Code Composer Studio 디버거 사용 가이드] 참고

3. CodeScroll Controller Tester 의 해당 프로젝트 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 설정을 해야 합니다.

3-1. 빌드 탭

Texas Instruments Code Composer Studio 빌드 가이드 참고

3-2. 실행 탭

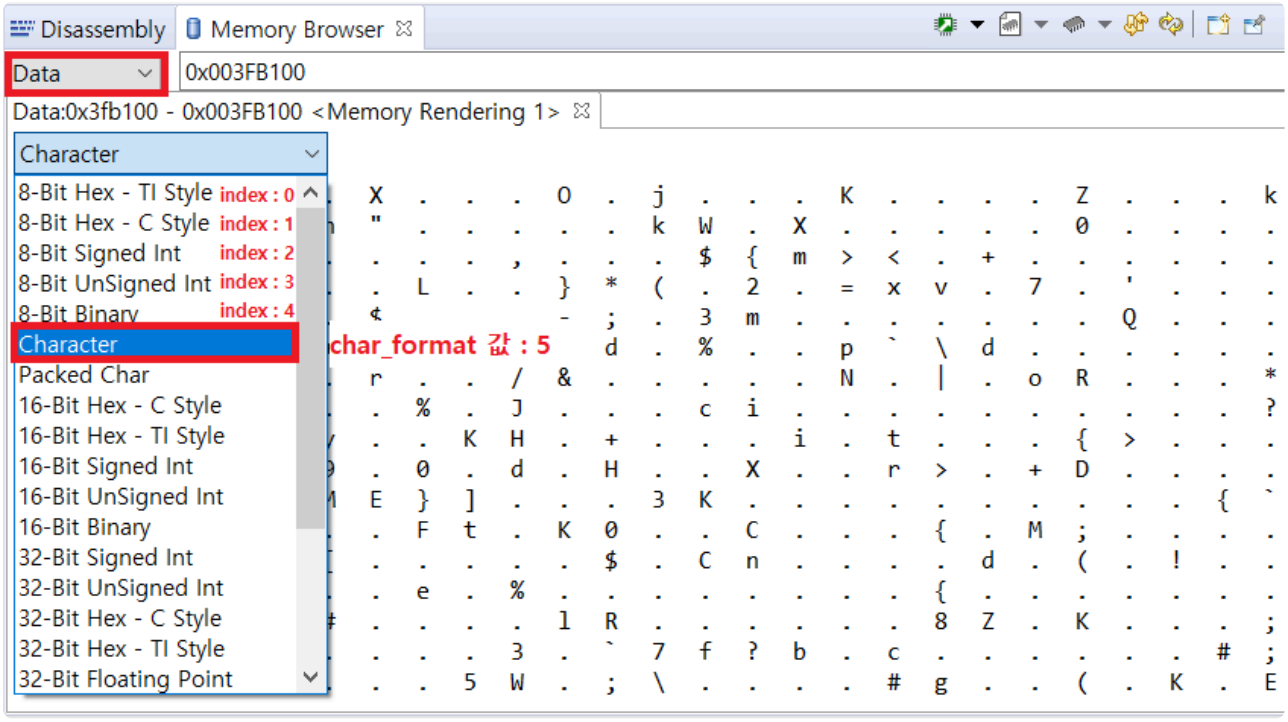


- 실행 탭

ccxml_path	CCS 프로젝트 경로에 있는 프로젝트 설정 파일입니다. 이 파일의 이름은 CCS에서 설정한 타깃의 이름입니다. 프로젝트 경로와 타깃의 이름이 맞는지 확인합니다. 예: {project_path}\targetConfig\{target_name}.ccxml
script_path	실행 스크립트가 따로 작성되어있을 시, 경로를 입력 (없으면 입력하지 않아도 됩니다.)
target_binary_path	CCS 빌드 시 생성되는 .out파일의 경로 예: {project_path}\Debug\{project_name}.out
debug_probe	debug_probe의 이름, CCS properties 사진의 Device 앞부분 (예시의 사진에서는 Spectrum Digital XDS560V2 STM USB Emulator)
cpu_name	cpu의 이름, CCS properties 사진의 뒷부분 (예시의 사진에서는 C28xx)
memory_page	CCS의 memory_page, CCS Memory Browser 사진 참고
char_format	CCS의 데이터 포맷 중 character의 순서. 상단으로부터 1로 시작 예: character가 해당 바의 상단에서부터 5번째에 있으면, char_format 값에 5를 입력하면 됩니다.

- CCS properties

- CCS Memory Browser



4. 테스트 실행

커버리지가 정상적으로 표시될 시 타깃 테스트 성공

2. Controller Tester 디버거 사용 가이드

CodeScroll Controller Tester 타깃 테스트 수행 시 디버거 사용 방법에 대한 가이드 문서입니다.

- Lauterbach TRACE32
- PLS Universal Debug Engine
- iSYSTEM winIDEA Debugger
- IAR Embedded Workbench C-SPY Debugger
- Texas Instruments Code Composer Studio

2.1. Lauterbach TRACE32

Controller Tester는 TRACE32 디버거를 사용하여 타깃 테스트를 할 수 있습니다.
Controller Tester는 TRACE32의 cmm 스크립트를 사용하여 타깃 환경에서 테스트를 실행하고 결과를 가져옵니다.
TRACE32에서 지원하는 타깃 목록은 Lauterbach 홈페이지 에서 확인할 수 있습니다.

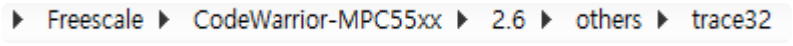
2.1.1. cmm 스크립트 자동 생성 지원 타깃 목록

Controller Tester는 cmm 스크립트 파일을 자동으로 생성하거나 사용자에게 입력받습니다.
cmm 스크립트를 자동으로 생성할 수 있는 경우에는 타깃의 칩 이름만 입력하면 됩니다. cmm 스크립트를 자동으로 생성할 수 없는 경우에는 사용자가 직접 cmm 스크립트 파일 경로를 입력해야 합니다.
현재 cmm 스크립트 자동 생성을 지원하는 타깃은 아래와 같습니다.

PowerPC	mpc5554, mpc5553, mpc5534, mpc556x, mpc551x, mpc560xe, spc560bxx, spc560pxx, spc560sxx, mpc560xb, mpc560xp, mpc560xs, spc563m54, mpc5632m, spc563m60, mpc5633m, spc563m64, mpc5634m, mpc564xs, mpc5668, mpc5674, mpc5644a, spc564a80, mpc5642a, spc564a70, mpc567xk, spc56hk, mpc5643l, spc56el60, spc56el70, mpc5644b, mpc5644c, spc564b64, spc56ec64, mpc5645b, spc564b70, mpc5645c, spc56ec70, mpc5646b, spc564b74, mpc5646c, spc56ec74, mpc5676r, spc56ap, mpc5746m, mpc5744k, spc574k74, mpc5777m, spc57hm90, mpc574xp, mpc574xg, mpc574xr, mpc577xk, mpc5777c, spc570s, mpc5726l, spc572l, spc574s, spc58ne, spc58eg, spc58nn, spc582b, spc58ec, spc58nh, spc584b, s32r274, s32r264, s32r372
ARM	mkw01, mkw20, mkv30, mkv40, mkv10, mkv50, mkm30, mkl0, mkl10, mkl20, mkl30, mkl40, mkl80, mk0, mk10, mk20, mk30, mk40, mk50, mk60, mk70, mk80, mac57d54h, mac71×1, mac71×2, mac71×4, mac71×5, mac71×6, mac72×1, lpc51u68, lpc54xx, lpc8xx, lpc11xx, lpc12xx, lpc13xx, lpc17xx, lpc18xx, lpc21xx, lpc22xx, lpc23xx, lpc24xx, lpc28xx, lpc29xx, lpc40xx, lpc43xx, imxrt1064, xmc1100, xmc1200, xmc1300, xmc1400, xmc4100, xmc4200, xmc4300, xmc4400, xmc4500, xmc4700, xmc4800, tle98, s3fm02g, s32k, s6e1a, s6e1c, s6j3
tricore	tc2dx, tc21x, tc22x, tc23x, tc26x, tc27x, tc29x, tc35x, tc37x, tc38x, tc39x, tc116x, tx1167, tx1197, tc1724, tc1728, tc1736, tc1762, tc1764, tc1766, tc1767, tc1782, tc1784, tc1791, tc1792, tc1793, tc1796, tc1797, tc1798

2.1.2. Step1: Controller Tester에서 타깃 환경 설정하기

Controller Tester의 타깃 환경 설정 페이지에서 디버거를 선택합니다. 프로젝트에 선택된 툴체인에 따라 지원하는 디버거 목록만 표시됩니다.
디버거를 TRACE32로 설정합니다.



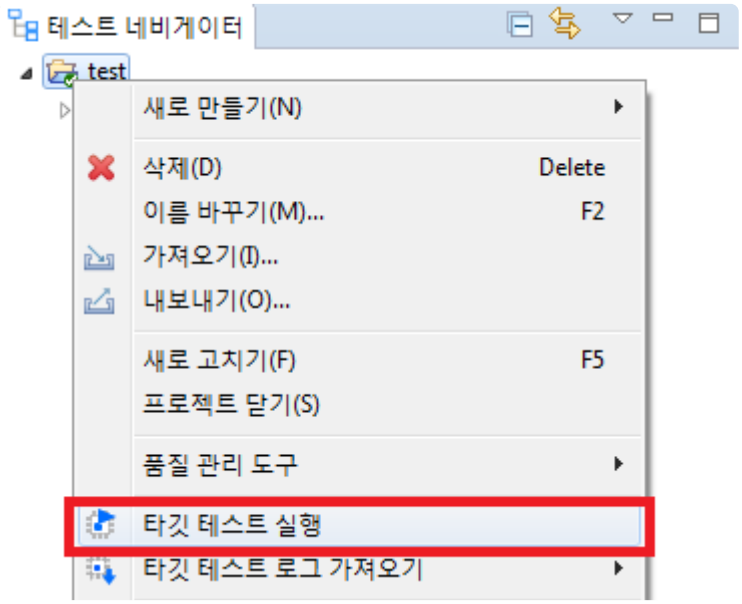
선택한 정보에 따라 설정 항목들이 표시됩니다. TRACE32 디버거를 사용하는 경우에 설정해야하는 항목은 아래 표와 같습니다.
설정 중 일부는 필수 항목입니다.

trace32_exe_file_path	TRACE32 실행파일 경로입니다. 타깃마다 실행파일이 다르기 때문에, 사용하는 타깃의 실행파일이 맞는지 확인해야 합니다. 필수 항목입니다.
target_binary_path	타깃 환경에 로드하기 위한 바이너리 파일 경로입니다. 사용하는 IDE 또는 빌드 스크립트에서 타깃 바이너리 파일이 생성되는 경로를 확인하고 입력합니다. 필수 항목입니다.
chip	사용하는 타깃의 칩 이름을 입력합니다. cmm 스크립트를 자동 생성할 때 사용되기 때문에 정확한 칩 이름을 입력해야 합니다.
user_defined_cmm_script_file_path	사용자 정의 cmm 스크립트 파일 경로입니다. cmm 스크립트 자동 생성을 지원하지 않는 타깃인 경우에는 디버거와 타깃 사용 환경을 설정하는 스크립트를 작성하거나, 사용하고 있는 cmm 스크립트 파일 경로를 입력해야 합니다.

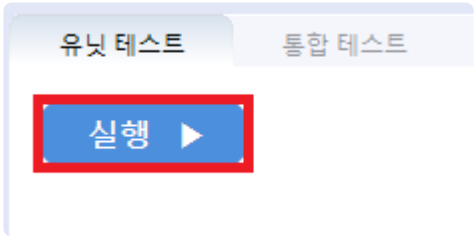
2.1.3. Step2: 타깃 테스트 실행하기

타깃 테스트를 실행하기 전에 실행 중인 TRACE32 프로그램을 종료해야 합니다.
테스트 네비게이터 뷰의 프로젝트 컨텍스트 메뉴에서 [타깃 테스트 실행하기]를 선택하거나 테스트 뷰의 [실행] 버튼을 클릭하여, 타깃 테스트를 실행할 수 있습니다.

- [타깃 테스트 실행하기]



- [실행]



타깃 테스트를 실행할 때 TRACE32 프로그램이 실행됩니다. 정상적으로 테스트가 완료되면 TRACE32 프로그램이 자동으로 종료됩니다.

2.1.4. 디버거를 통해 타깃 테스트 디버깅하기

1. 타깃으로 설정 후 ‘유닛 테스트’ 뷰에서 테스트 케이스 우클릭 후 ‘디버그 정보 확인’ 클릭
2. 사용자 프로젝트를 직접 빌드 혹은 Controller Tester 프로젝트에서 ‘타깃 환경’ 설정에 등록한 빌드 스크립트 수행
3. 빌드 성공하는지 확인
4. Controller Tester에서 프로젝트를 열어 원본 소스 복원
5. Trace32 실행 후 cmm 스크립트 파일(start.cmm)을 열어 ‘debug’를 실행(Controller_Tester_프로젝트_경로/.csdata/target/start.cmm)
6. ‘step’ 버튼을 클릭하면 target.cmm 스크립트 첫 라인으로 이동
7. target.cmm 파일의 Go.HII에 breakpoint 지정
8. Trace32의 Var > Show Function 클릭
9. 테스트 대상 함수 검색 후 더블 클릭
10. 함수 시작부에 breakpoint 지정
11. ‘step’ 버튼을 클릭하면 디버깅 포인트가 10번에서 지정한 위치로 이동하는 것을 확인
12. Var > Show Local…. 클릭하면 지역변수의 값이 변하는 것을 확인 가능
13. 디버깅 포인트까지 실행

2.2. PLS Universal Debug Engine (UDE)

Controller Tester는 UDE 디버거를 사용하여 타깃 테스트를 할 수 있습니다.
Controller Tester는 UDE에서 지원하는 디버깅 스크립트를 사용하여 타깃 환경에서 테스트를 실행하고 결과를 가져옵니다.
UDE에 연결하여 사용할 수 있는 타깃 목록은 PLS 홈페이지 에서 확인할 수 있습니다.

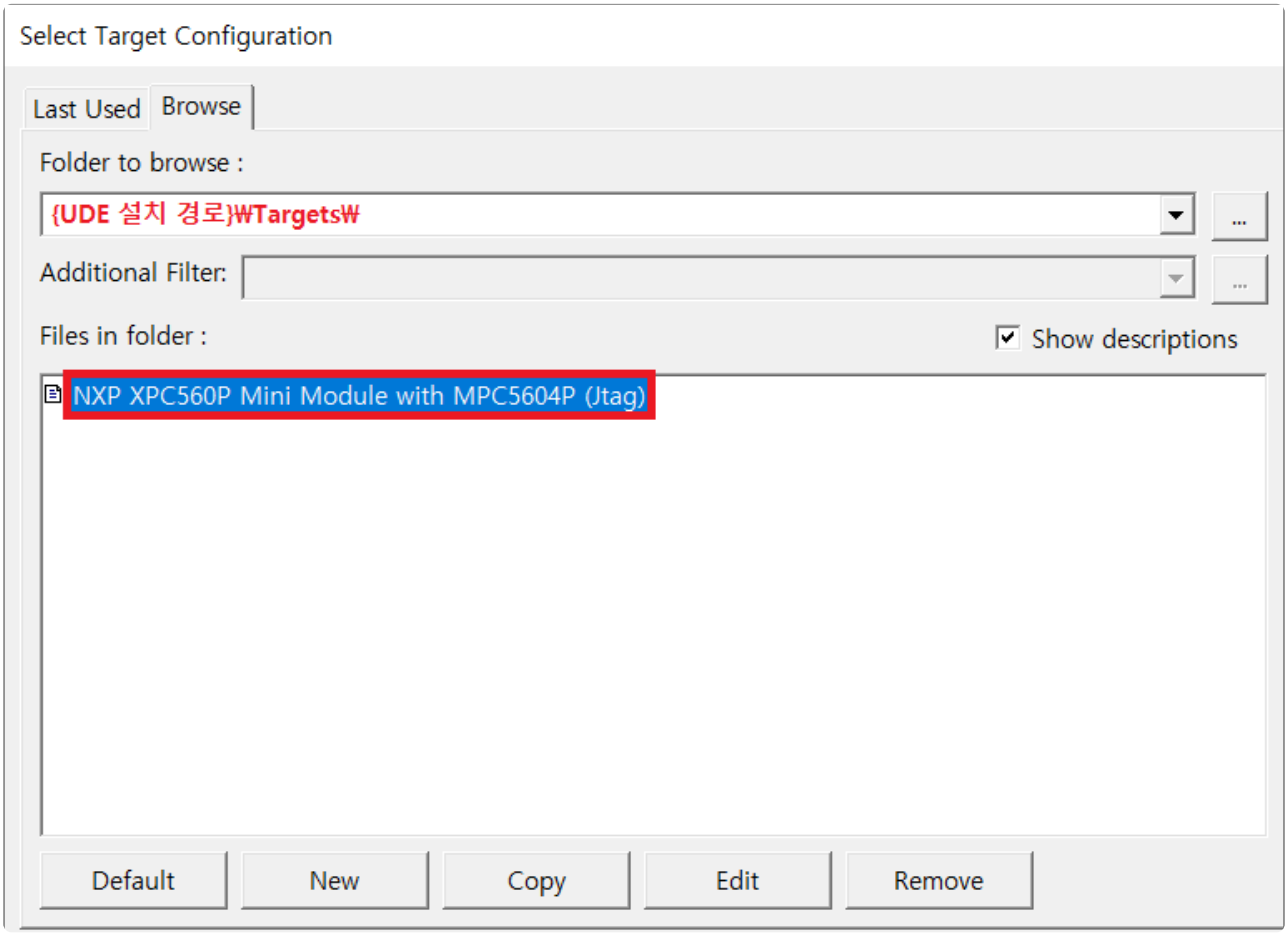
Controller Tester에서는 UDE 워크스페이스 정보를 사용하여 타깃 테스트를 수행합니다. 그렇기 때문에 사용자는 타깃 테스트 수행 전에 먼저 워크스페이스를 생성해야 합니다.

- Step1: UDE IDE에서 워크스페이스 생성하기
- Step2: Controller Tester에서 타깃 환경 설정하기
- Step3: 타깃 테스트 실행하기

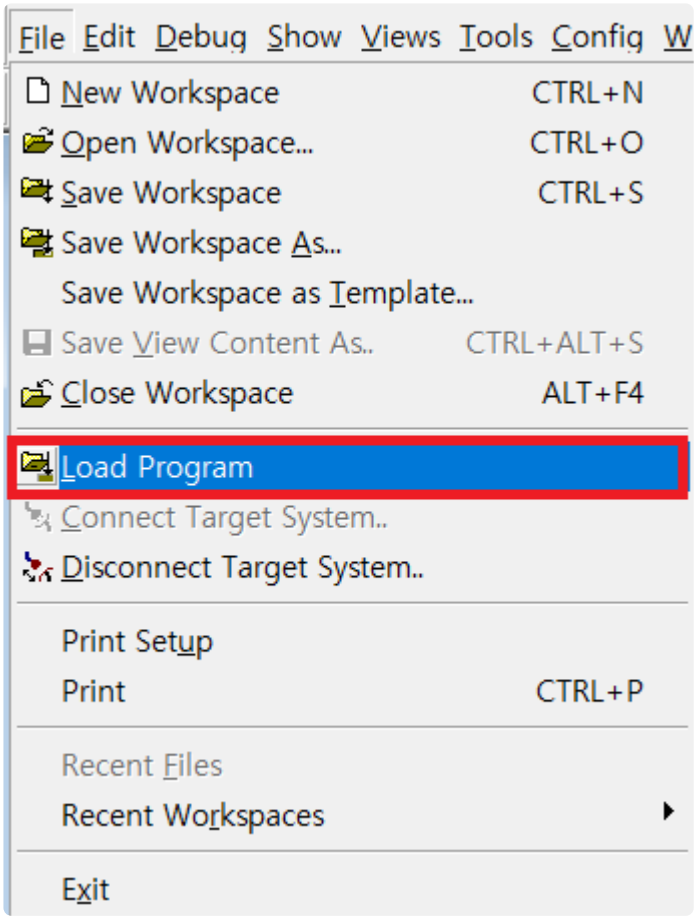
2.2.1. Step1: UDE IDE에서 워크스페이스 생성하기

UDE는 UDE desktop IDE에서 UDE 워크스페이스를 생성할 수 있습니다.

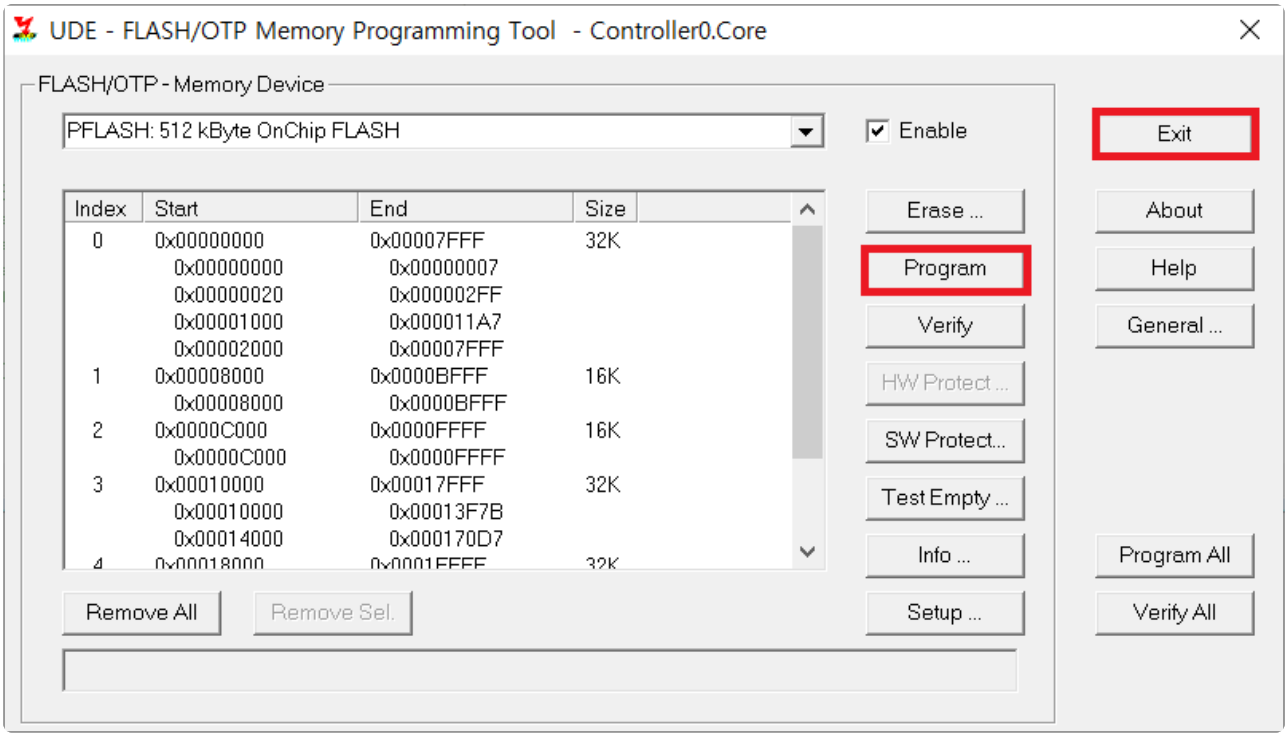
1. 사용하는 타깃에 맞는 설정 파일을 선택하여 워크스페이스를 생성합니다.



2. 상단 메뉴에서 [File] > [Load Program] 버튼을 클릭하면 바이너리 파일을 로드할 수 있습니다. 이때 테스트 코드로부터 빌드된 바이너리 파일을 선택합니다.



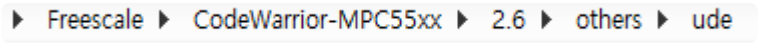
3. 이후 안내에 따라 [program] 버튼을 누르면 타깃 설정에 따라 바이너리 파일이 타깃에 로드됩니다. 로드가 정상적으로 끝나면 워크스페이스 설정이 완료됩니다. [Exit]를 클릭하여 창을 빠져나옵니다.



자세한 사항은 UDE에서 제공하는 매뉴얼을 참조하십시오.

2.2.2. Step2: Controller Tester에서 타깃 환경 설정하기

Controller Tester의 타깃 환경 설정 페이지에서 디버거를 선택합니다. 프로젝트에 선택된 툴체인에 따라 지원하는 디버거 목록만 표시됩니다.
디버거를 UDE 로 설정합니다.



선택한 정보에 따라 설정 항목들이 표시됩니다. UDE 디버거를 사용하는 경우에 설정해야하는 항목은 아래 표와 같습니다.
설정 중 일부는 필수 항목입니다.

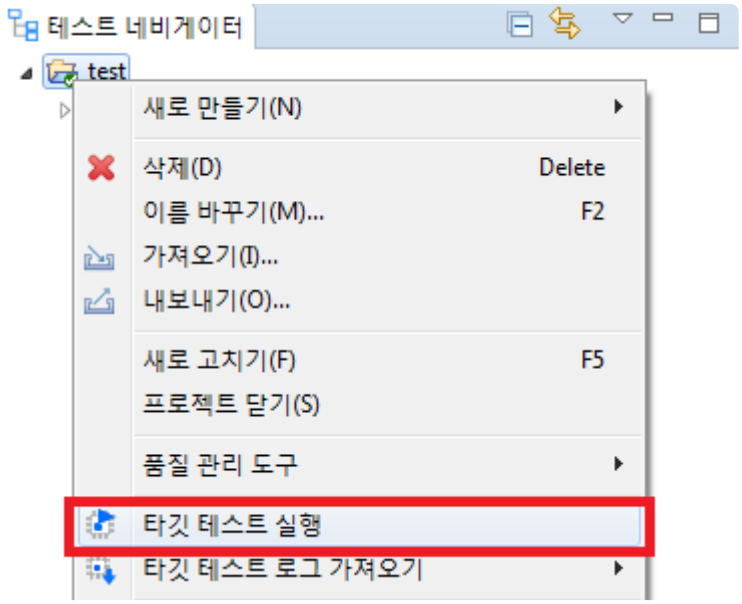
target_binary_path	타깃 환경에 로드하기 위한 바이너리 파일 경로입니다. 사용하는 IDE 또는 빌드 스크립트에서 타깃 바이너리 파일이 생성되는 경로를 확인하고 입력합니다. 필수 항목입니다.
ude_project_file	UDE IDE에서 생성된 워크스페이스 프로젝트 파일(.wsx) 경로입니다. 필수 항목입니다.

Controller Tester에서 사용하는 기본 스크립트 언어는 visual basic script입니다.
타깃 환경 설정이 끝나면 [OK] 또는 [Finish]버튼을 클릭합니다. 타깃 테스트를 수행할 준비가 끝났습니다.

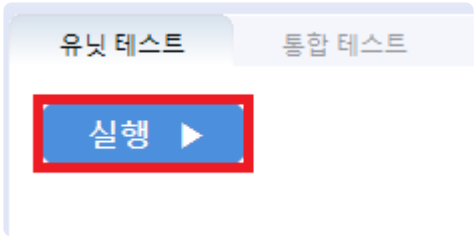
2.2.3. Step3: 타깃 테스트 실행하기

타깃 테스트를 실행하기 위해서는 UDE IDE를 종료한 상태여야 합니다.
테스트 네비게이터 뷰의 프로젝트 컨텍스트 메뉴에서 [타깃 테스트 실행하기]를 선택하거나 테스트 뷰의 [실행] 버튼을 클릭하여, 타깃 테스트를 실행할 수 있습니다.

- [타깃 테스트 실행하기]



- [실행]



UDE 디버깅 스크립트는 C++, .NET, Perl 등의 언어로 작성할 수 있습니다. 스크립트로 작성할 수 있는 다른 지원 언어들에 대해서는 UDE 매뉴얼에 포함되는 UDE Automation Basics 설명서를 확인하십시오.

2.2.4. 디버거를 통해 타깃 테스트 디버깅하기

1. 타깃으로 설정 후 ‘유닛 테스트’ 뷰에서 테스트 케이스 우클릭 후 ‘디버그 정보 확인’ 클릭
2. 사용자 프로젝트를 직접 빌드 혹은 Controller Tester 프로젝트에서 ‘타깃 환경’ 설정에 등록한 빌드 스크립트 수행
3. 빌드 성공하는지 확인
4. Controller Tester에서 프로젝트를 열어 원본 소스 복원
5. Pls Ude 실행 후 프로젝트 선택(.wsx파일)
6. 2번에서 빌드된 output파일을 선택
7. 좌측 네비게이션에 output 파일에 들어 있는 소스 파일과 함수 정보가 표시되는 것을 확인
8. 테스트 대상 함수가 있는 소스파일 선택 후 함수에 디버깅 포인트 지정
9. F5버튼을 누르면 entry point에서부터 실행

2.3. iSYSTEM winIDEA Debugger

Controller Tester는 winIDEA 디버깅 스크립트를 통해 자동으로 타깃 환경에서 테스트를 실행하고 결과를 가져오는 기능을 제공합니다.
winIDEA가 지원하는 타깃 목록은 iSYSTEM 홈페이지 에서 확인할 수 있습니다.

디버깅 스크립트의 실행은 winIDEA 설치 시 함께 설치되는 python SDK를 필요로 합니다. 만일 설치되어 있지 않은 경우 iSYSTEM SDK 설치 페이지 에서 다운로드받을 수 있습니다. 또한, 사용하는 winIDEA 버전이 해당 SDK를 지원하고 있는지 확인해야 합니다. Controller Tester에서 기본으로 제공하는 디버깅 스크립트는 python 3.3 버전을 기준으로 합니다.

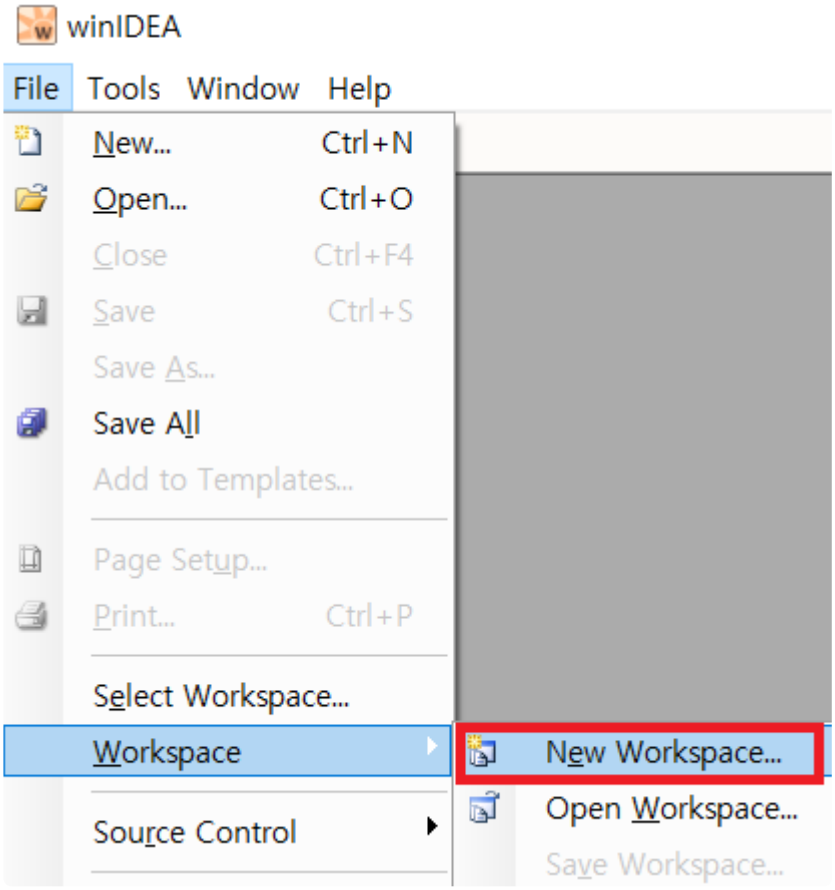
이 문서에서는 winIDEA에서 프로젝트를 생성하는 것부터 Controller Tester에서 타깃 테스트를 실행하기까지의 과정을 예시와 함께 설명합니다.
예시에서는 iSYSTEM BlueBox iC5000 Unit 디버거와 NXP의 MPC56xx 타깃을 사용합니다.

2.3.1. iSYSTEM winIDEA 를 사용하기 위한 사전 준비

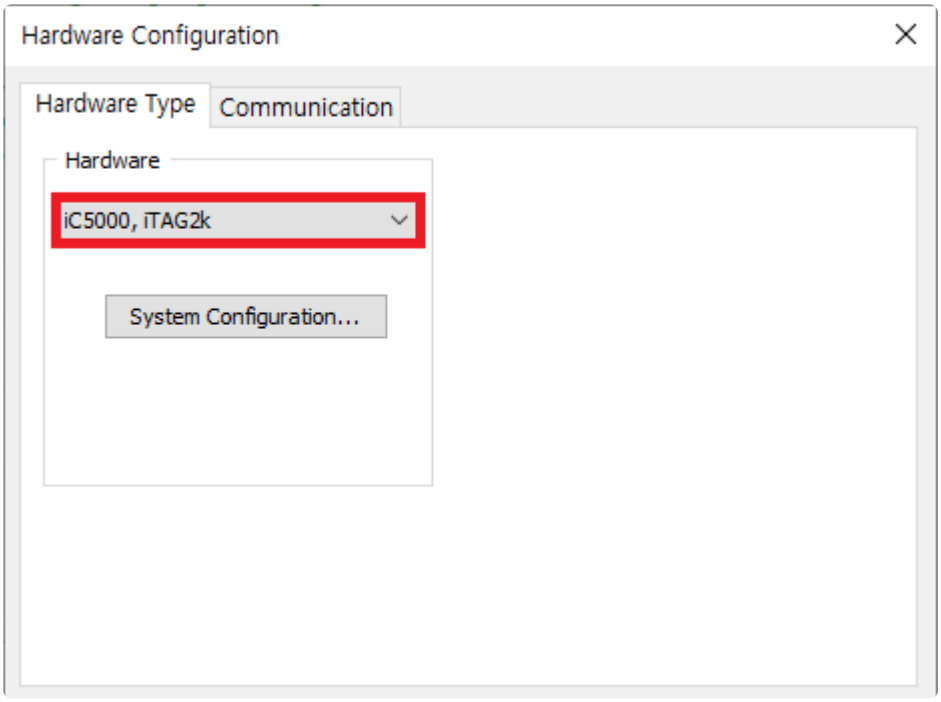
Controller Tester에서 winIDEA를 사용하여 타깃 테스트를 하려면 winIDEA와 연결하여 사용 가능한 디버거가 필요합니다.
사용자는 타깃 테스트 수행 전에 winIDEA 워크스페이스를 생성하고 사용할 디버거를 Controller Tester가 설치된 PC와 연결해야 합니다.

2.3.2. Step1: winIDEA 워크스페이스 생성 및 설정

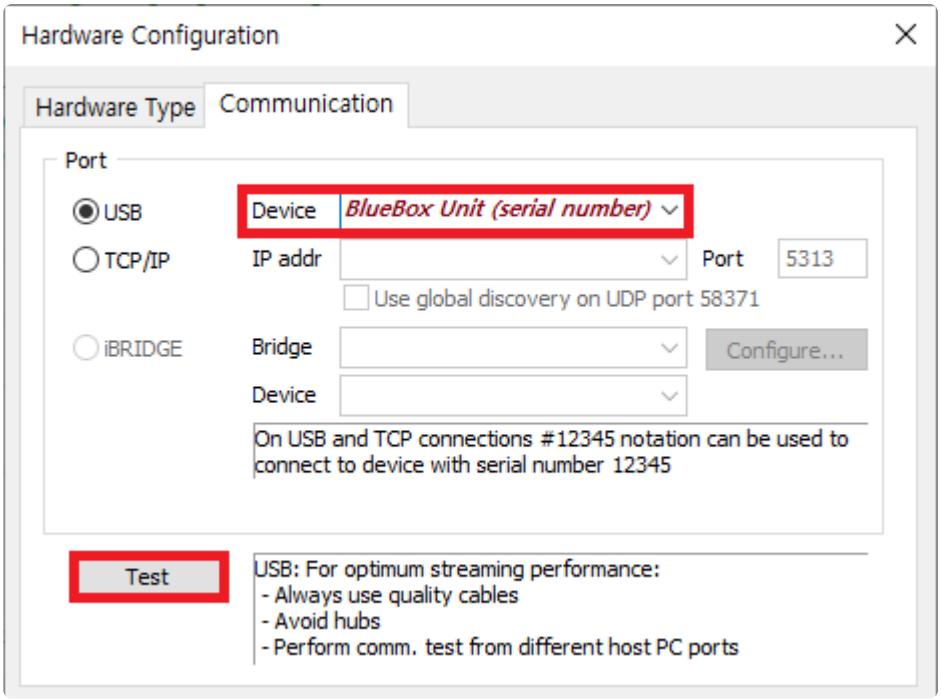
1. winIDEA 실행 후, 상단 메뉴에서 [File] > [Workspace] > [New Workspace…]를 선택하여 새로운 워크스페이스를 생성합니다. 생성한 워크스페이스를 Controller Tester 타깃 테스트에 사용하기 위해서는 추가 환경 설정이 필요합니다.



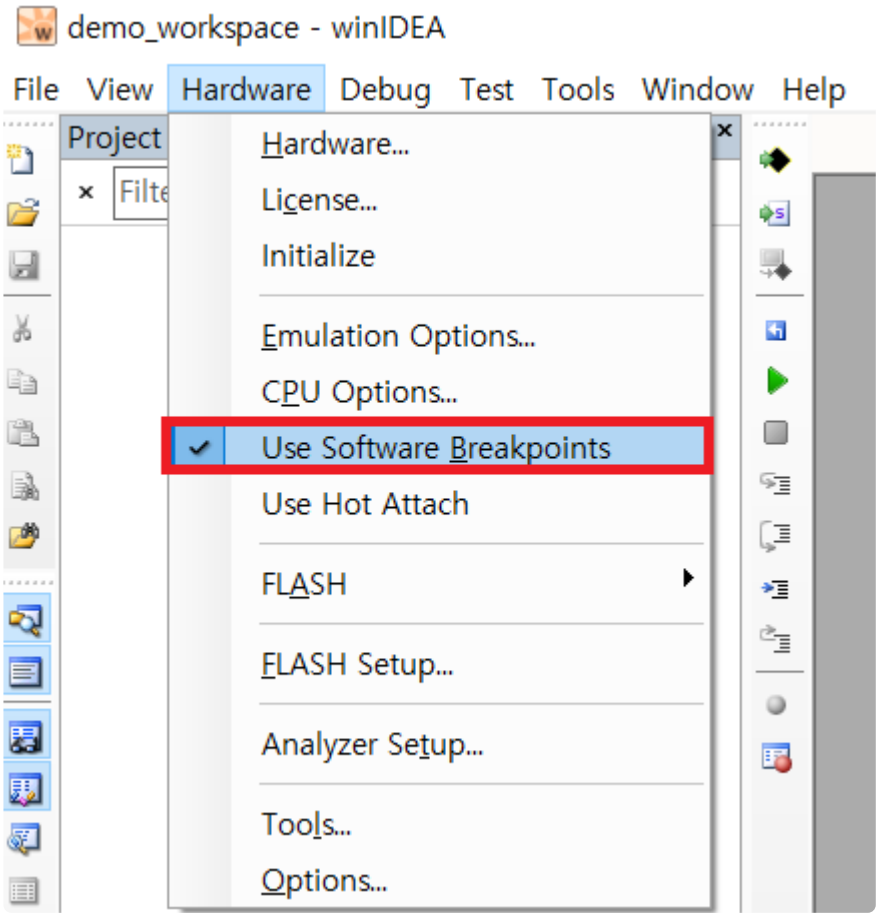
2. 첫 번째로 상단 메뉴에서 [Hardware] > [Hardware…] 를 클릭한 후, [Hardware Type] 탭에서 연결된 BlueBox의 종류를 선택합니다.

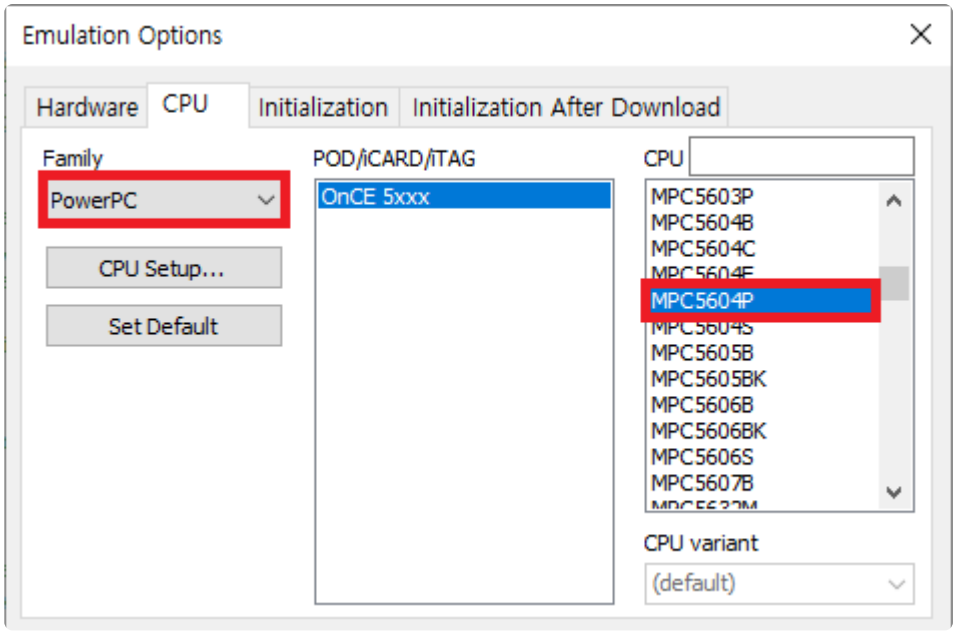


3. 다음으로 [Communication] 탭에서 통신 방식 설정하고 [Test] 버튼을 눌러 기기 연결을 확인합니다. 통신 방식에 따른 디버거 장비 연결 방법에 대한 설명은 iSYSTEM BlueBox 매뉴얼을 참조하십시오.



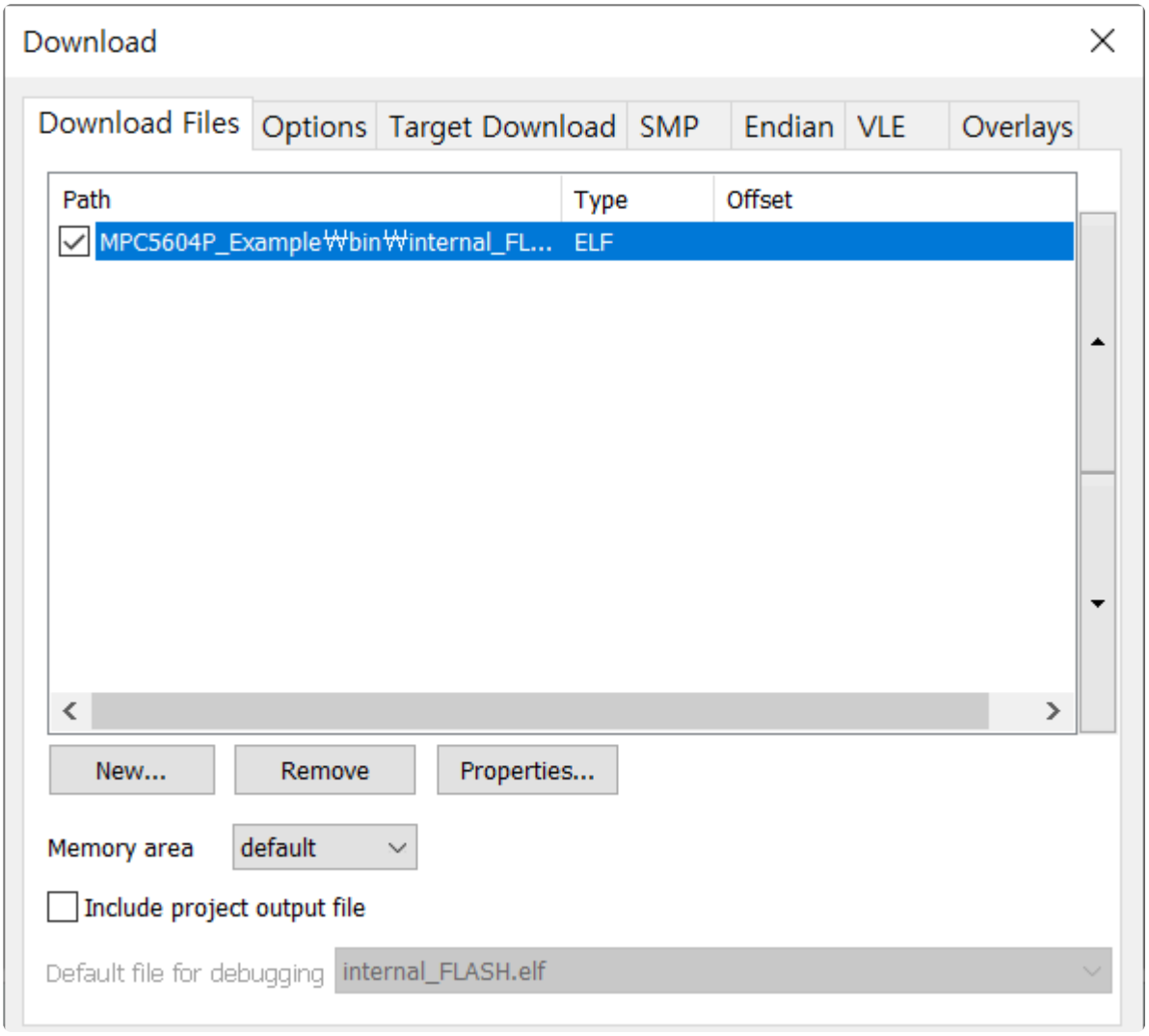
4. 상단 메뉴 중 [Hardware] > [Use Software Breakpoints] 를 클릭해서 활성화시킨 후, [Hardware] > [Emulation options…] 의 [CPU] 항목에서 사용할 타깃 종류를 선택합니다.



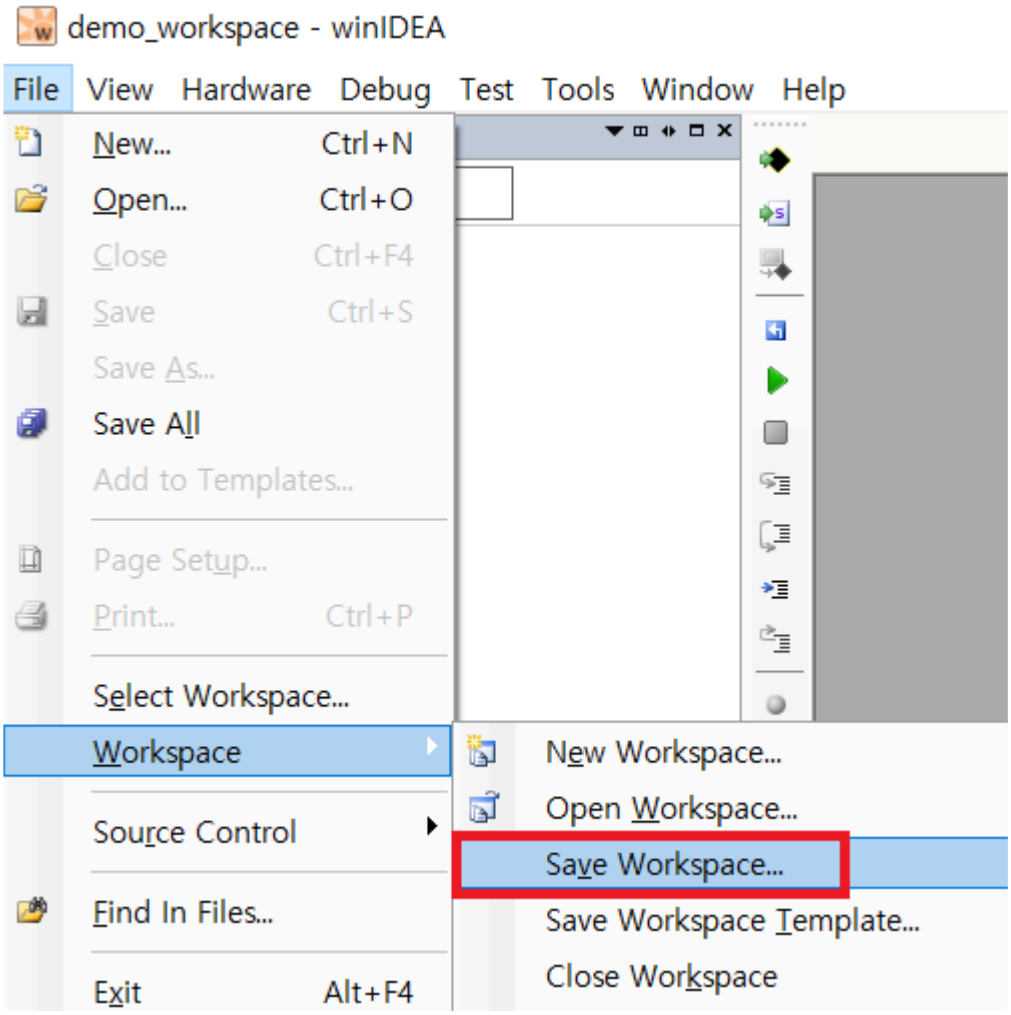


타깃 별로 설정해야 하는 세부 옵션이 다를 수 있습니다.

5. 기기 설정을 끝내고 나면 테스트할 바이너리 경로를 등록해야 합니다. 먼저 테스트 대상 소스 코드를 빌드하여 바이너리를 생성합니다. 그런 다음 winIDEA의 상단 메뉴 [Debug] > [Files for Download…]에서 [New…]를 선택하고 생성한 바이너리 경로를 추가해줍니다.



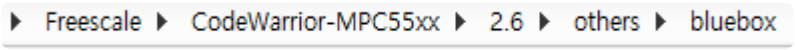
6. 워크스페이스 설정이 끝나고, 워크스페이스를 저장하여 winIDEA 워크스페이스 파일(.xjrf)을 생성합니다. 워크스페이스 파일은 Controller Tester에서 winIDEA를 이용한 타깃 테스트 설정 시 사용됩니다.



이제 타깃 테스트를 하기 위한 winIDEA 워크스페이스 생성이 끝났습니다.

2.3.3. Step2: Controller Tester에서 타깃 환경 설정

Controller Tester의 타깃 테스트 프로젝트 생성 마법사 또는 프로젝트 프로퍼티의 타깃 환경 설정에서 디버거를 선택합니다. 프로젝트 생성 시 선택한 툴체인에 따라 선택 가능한 디버거 목록이 달라집니다.
디버거를 BlueBox 로 설정합니다.



선택한 정보에 따라 설정 항목들이 표시됩니다. BlueBox를 사용하는 경우에 설정해야 하는 항목은 아래 표와 같습니다.
필수로 입력해야 하는 항목은 Controller Tester에서 붉은 색으로 표시됩니다.

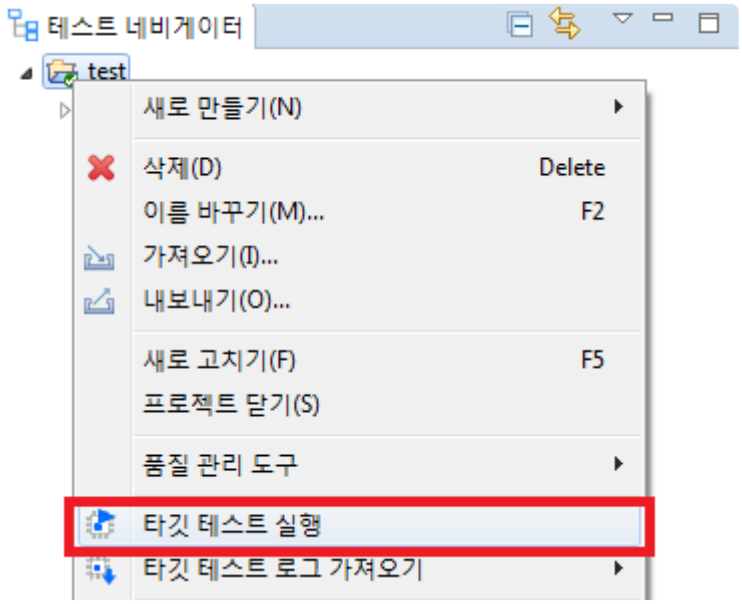
winidea_binary_path	winIDEA 실행 파일(winIDEA.exe) 경로입니다. 필수 항목입니다.
winidea_workspace_file_path	winIDEA에서 생성한 워크스페이스 파일(.xjrf) 경로입니다. 필수 항목입니다.

Controller Tester에서 사용하는 기본 스크립트 언어는 python입니다. 사용자 정의 디버깅 스크립트를 사용하는 경우, python으로 작성해야 원활한 타깃 테스트가 가능합니다. 다른 언어를 사용하여 디버깅 스크립트를 작성하는 경우에는 iSYSTEM 홈페이지 를 참고하여 추가 SDK 설치를 진행해야 합니다.
타깃 환경 설정이 끝나면 [OK] 또는 [Finish]버튼을 클릭합니다. 타깃 테스트를 수행할 준비가 끝났습니다.

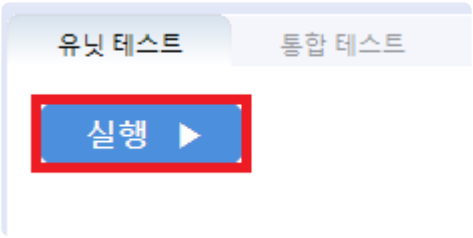
2.3.4. Step3: 타깃 테스트 실행

테스트 네비게이터 뷰의 프로젝트 컨텍스트 메뉴에서 [타깃 테스트 실행하기]를 선택하거나 테스트 뷰의 [실행] 버튼을 클릭하여, 타깃 테스트를 실행할 수 있습니다.

- [타깃 테스트 실행하기]



- [실행]



winIDEA가 실행 중인 상태에서는 타깃 테스트를 수행할 수 없습니다. Controller Tester에서 타깃 테스트를 실행하기 전에 반드시 winIDEA를 종료해야 합니다.

2.3.5. 디버거를 통해 타깃 테스트 디버깅하기

1. 타깃으로 설정 후 ‘유닛 테스트’ 뷰에서 테스트 케이스 우클릭 후 ‘디버그 정보 확인’ 클릭
2. 사용자 프로젝트를 직접 빌드 혹은 Controller Tester 프로젝트에서 ‘타깃 환경’ 설정에 등록한 빌드 스크립트 수행
3. 빌드 성공하는지 확인
4. Controller Tester에서 프로젝트를 열어 원본 소스 복원
5. winIDEA 실행 후 빌드한 프로젝트 포함하는 워크스페이스 선택 (.xjrf파일)
6. [Debug] > [Download] 선택하여 바이너리 파일 타깃에 다운로드
7. 상단에 Run 버튼을 누르면 디버깅 모드
8. [project workspace 뷰] > [Functions] 항목 더블 클릭하여 해당 함수 위치로 이동 후, 원하는 곳에 디버깅 포인트 설정
9. F5 눌러 디버깅 진행

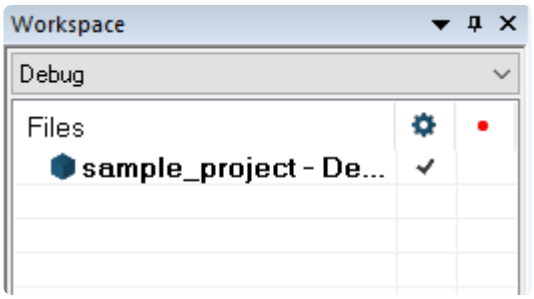
2.4. IAR Embedded Workbench C-SPY Debugger

Controller Tester는 IAR Embedded Workbench C-SPY 디버깅 기능을 통해 자동으로 타깃 환경에서 테스트를 실행하고 결과를 가져오는 기능을 제공합니다.
C-SPY에서 지원하는 타깃 목록은 IAR 홈페이지 에서 확인할 수 있습니다.

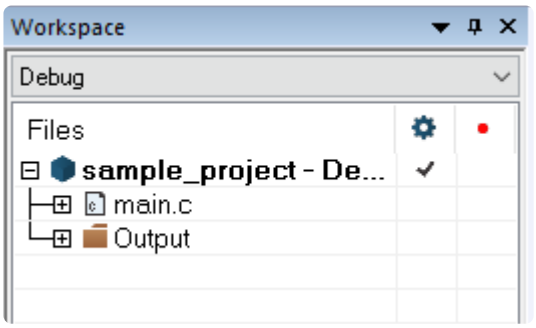
Controller Tester에서 IAR Embedded Workbench C-SPY로 타깃 테스트를 하려면 C-SPY 디버깅 기능을 사용할 수 있는 디버깅 프로브가 필요합니다. 사용자는 타깃 테스트 수행 전에 IAR Embedded Workbench 프로젝트를 생성하고 사용할 디버깅 프로브를 Controller Tester가 설치된 PC와 연결해야 합니다.
IAR에서 제공하는 디버깅 프로브 목록은 IAR 홈페이지 에서 확인할 수 있습니다.

2.4.1. Step1: IAR Embedded Workbench 프로젝트 생성

1. [File] > [New Workspace]를 클릭하여 새 워크스페이스를 생성한 후 [Project] > [Create New Project…]를 클릭하여 프로젝트 파일(.ewp)을 생성합니다. 프로젝트 파일을 생성하고 나면 IAR Embedded Workbench의 [workspace 뷰]에 프로젝트 이름이 표시됩니다.



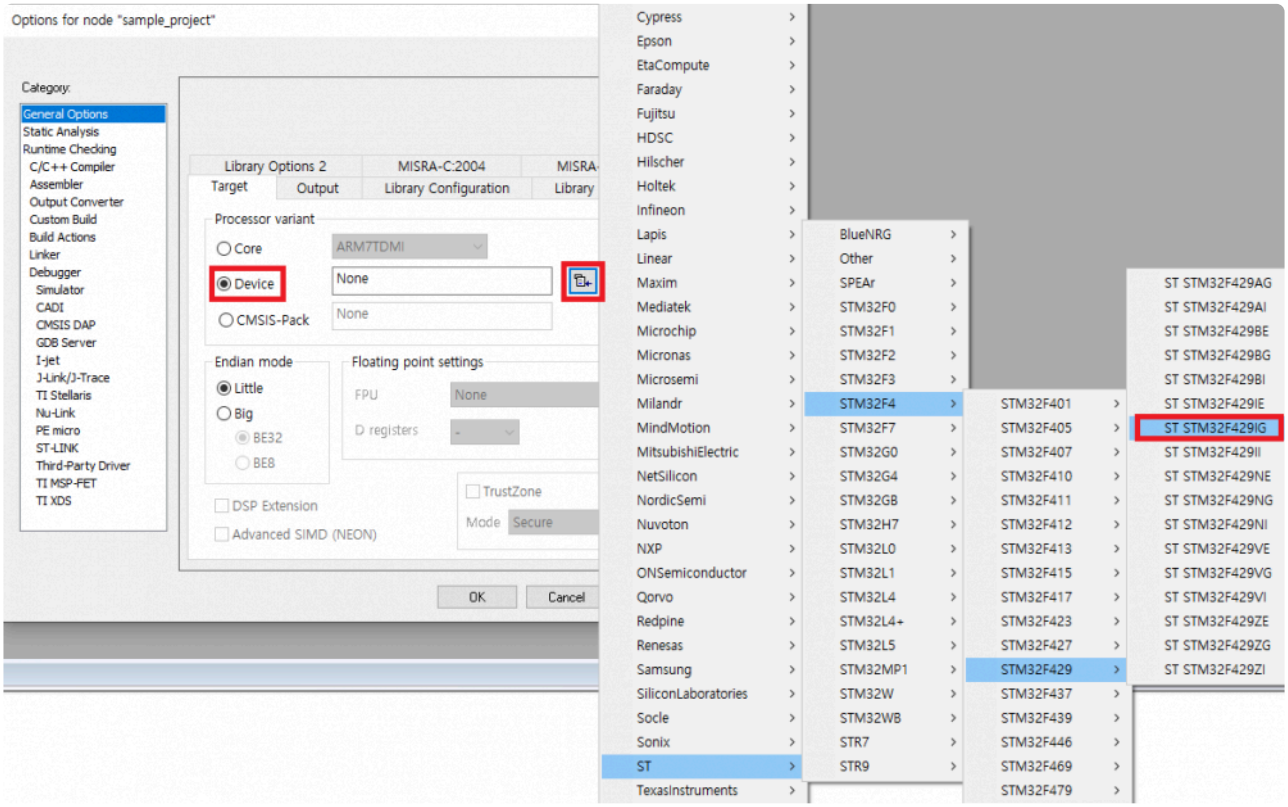
2. 다음으로 프로젝트에 시험 대상 소스 파일을 추가해야 합니다. 프로젝트를 우클릭을 하여 [Add] > [Add Files…]를 선택하고, 시험 대상 소스 파일을 추가합니다. 추가 후 [workspace 뷰]에서 계층 구조로 추가된 소스 파일을 확인할 수 있습니다.



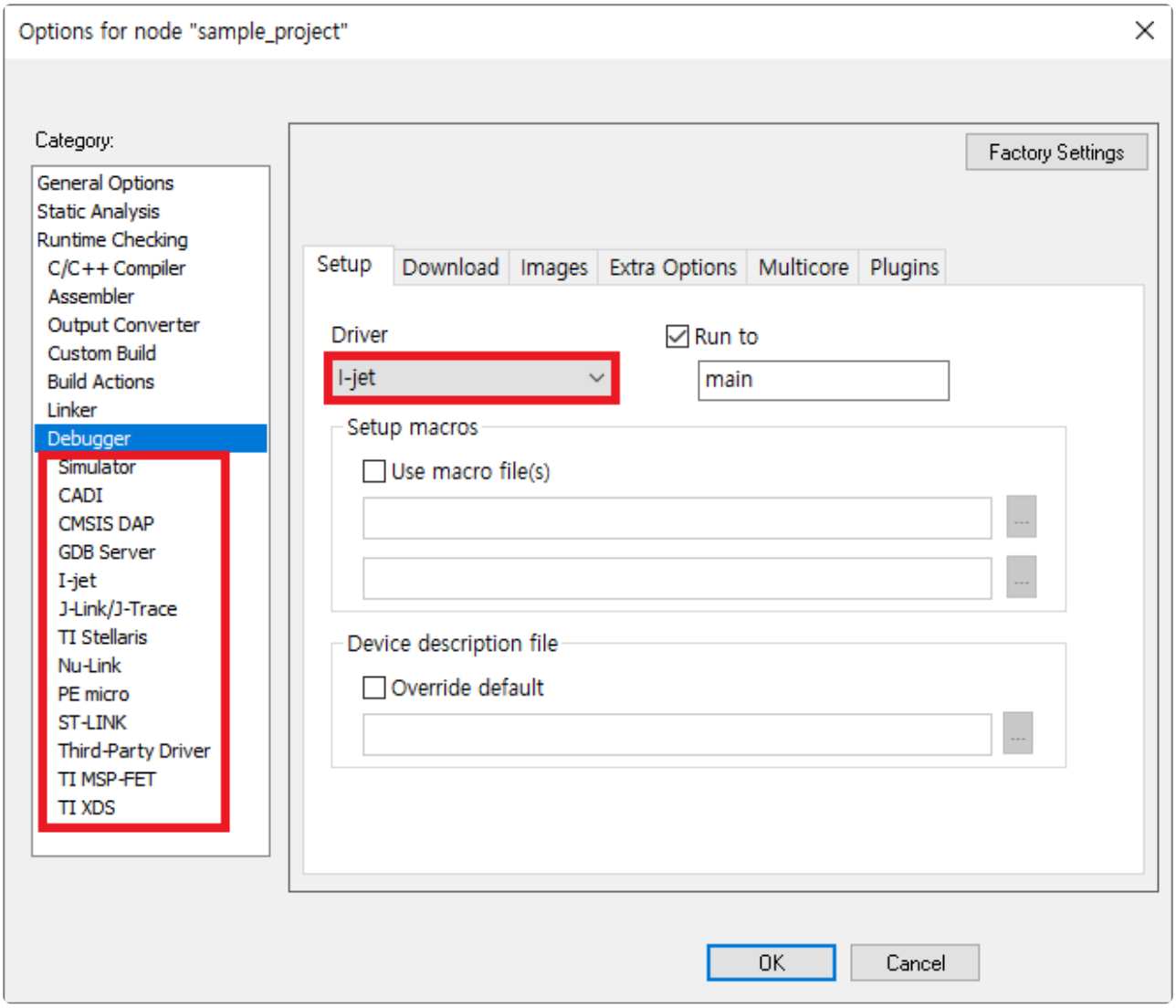
2.4.2. Step2: IAR 프로젝트 설정

프로젝트를 생성했다면 C-SPY 디버깅 기능 사용을 위한 프로젝트 설정을 해야 합니다. 생성한 프로젝트를 우클릭하여 [Options...]를 선택합니다.

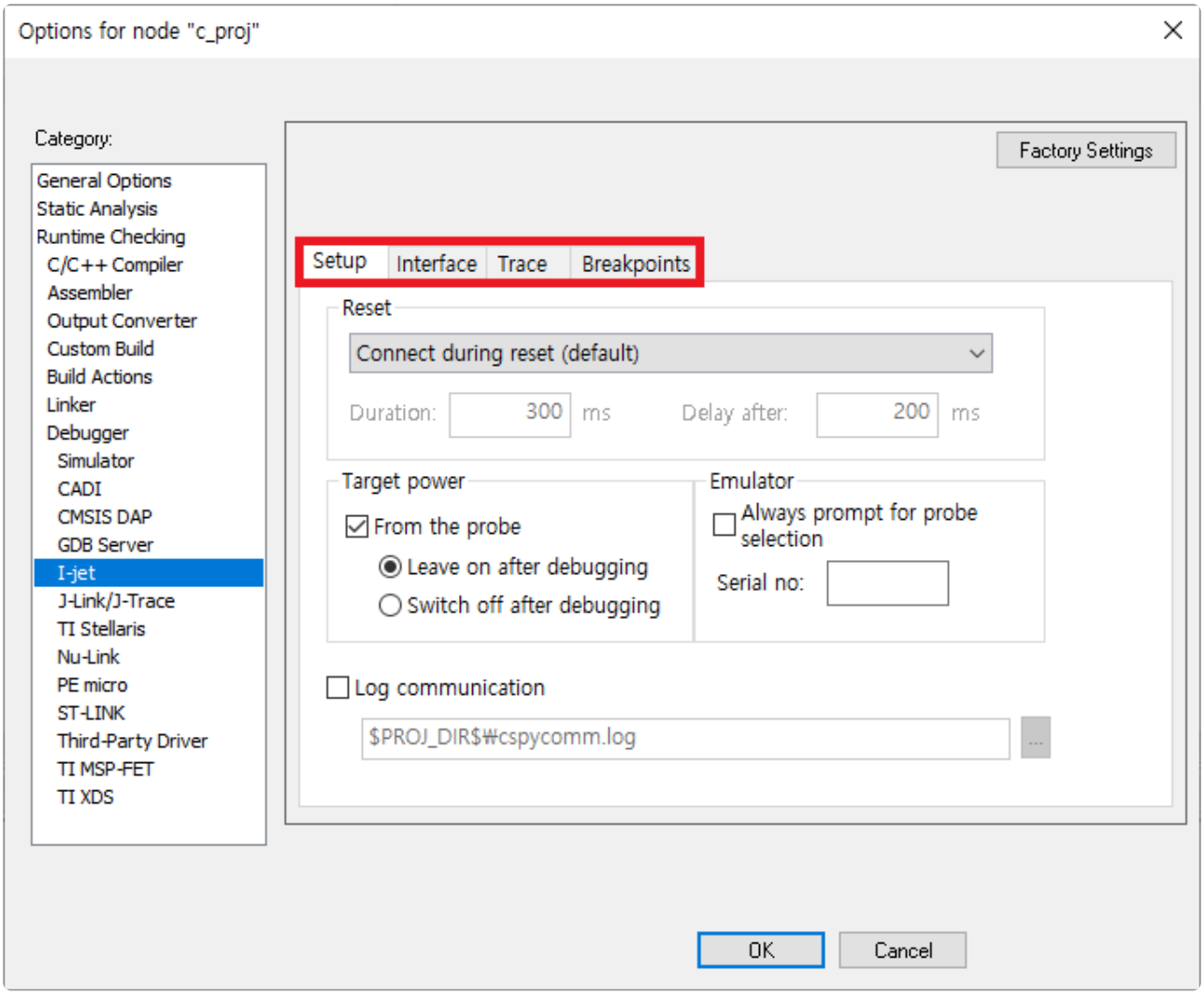
1. 먼저, [General Options]에서 [Processor variant]를 설정합니다. 예를 들어, ARM의 STM32F429IG 타깃의 경우 Device를 선택한 후 우측의 타깃 목록에서 해당 타깃에 맞는 이름을 고릅니다.



2. 두번째로 [Debugger] 카테고리로 이동하여, [Driver] 항목에서 사용하려는 디버깅 프로브를 선택합니다. 선택한 디버깅 프로브와 PC를 연결한 방식에 따라 [Debugger] 카테고리 하단의 디버깅 프로브 항목에서 세부 내용을 설정합니다.



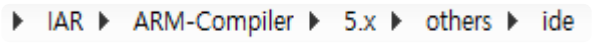
3. I-jet를 선택한 경우, [Debugger] 카테고리 하단의 [I-jet] 항목을 선택하여 세부 내용을 설정합니다. 각 설정 탭에 대한 설명은 사용하고자 하는 IAR 디버거 메뉴얼을 참조하십시오.



이제 타깃 테스트를 하기 위한 IAR 프로젝트 생성 및 설정이 끝났습니다.

2.4.3. Step3: Controller Tester에서 타깃 환경 설정

Controller Tester의 타깃 테스트 프로젝트 생성 마법사 또는 프로젝트 프로퍼티의 타깃 환경 설정에서 디버거를 선택합니다. 프로젝트 생성 시 선택한 툴체인에 따라 선택 가능한 디버거 목록이 달라집니다.
IAR 툴체인을 사용하여 프로젝트 생성 시, IAR C-SPY 디버깅 기능을 사용하기 위해서는 디버거를 ide로 설정해야 합니다.



선택한 정보에 따라 설정 항목들이 표시됩니다. C-SPY를 사용하는 경우에 설정해야 하는 항목은 아래 표와 같습니다.
필수로 입력해야 하는 항목은 Controller Tester에서 붉은 색으로 표시됩니다.

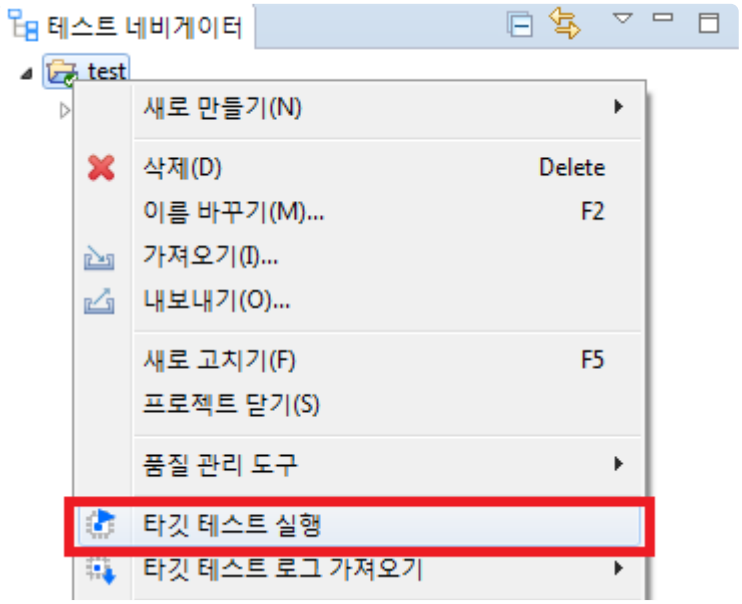
cspy_debug_general_xcl_file_path	IAR Embedded Workbench C-SPY 디버거를 사용 시 필요한 debug.general.xcl 파일 경로입니다. IAR 프로젝트 생성 시 프로젝트 파일(.ewp)이 저장된 위치의 [setting] 폴더에 자동 생성됩니다. 필수 항목입니다.
cspy_debug_driver_xcl_file_path	IAR Embedded Workbench C-SPY 디버거를 사용 시 필요한 debug.driver.xcl 파일 경로입니다. IAR 프로젝트 생성 시 프로젝트 파일(.ewp)이 저장된 위치의 [setting] 폴더에 자동 생성됩니다. 필수 항목입니다.

타깃 환경 설정이 끝나면 [OK] 또는 [Finish]버튼을 클릭합니다. 타깃 테스트를 수행할 준비가 끝났습니다.

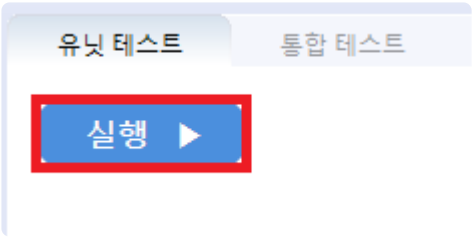
2.4.4. Step4: 타깃 테스트 실행

테스트 네비게이터 뷰의 프로젝트 컨텍스트 메뉴에서 [타깃 테스트 실행하기]를 선택하거나 테스트 뷰의 [실행] 버튼을 클릭하여, 타깃 테스트를 실행할 수 있습니다.

- [타깃 테스트 실행하기]



- [실행]



2.4.5. 디버거를 통해 타깃 테스트 디버깅하기

1. 타깃으로 설정 후 ‘유닛 테스트’ 뷰에서 테스트 케이스 우클릭 후 ‘디버그 정보 확인’ 클릭Output 파일이 떨어진 것을 확인
2. 사용자 프로젝트를 직접 빌드 혹은 Controller Tester 프로젝트에서 ‘타깃 환경’ 설정에 등록한 빌드 스크립트 수행
3. 빌드 성공하는지 확인
4. IAR Workbench 실행 후 빌드한 프로젝트 포함하는 워크스페이스 선택 (.eww파일)
5. workspace 뷰에서 테스트 대상 함수가 있는 소스파일 선택 후, 라인 왼쪽을 클릭하여 디버깅 포인트 지정
6. workspace 뷰에서 프로젝트를 우클릭, Options…을 열어 Debugger 항목에서 Run to 옵션 체크 확인 및 ‘main’으로 지정되어 있는지 확인
7. 상단의 Download and Debug 버튼을 누르면 main부터 실행
8. F5 눌러서 디버깅 포인트까지 진행하여 디버깅

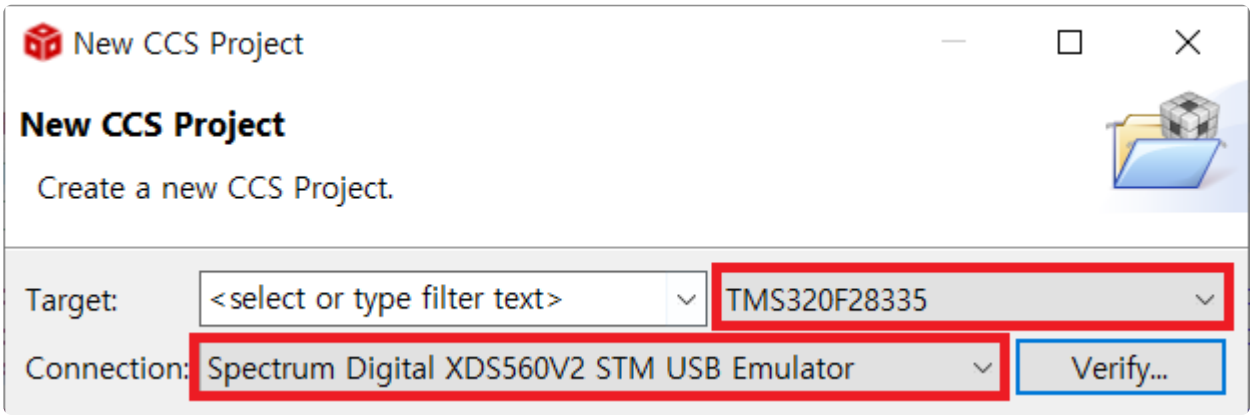
2.5. Texas Instruments Code Composer Studio (CCSv4 and greater)

Controller Tester는 CCS 디버거를 사용하여 타깃 테스트를 할 수 있습니다.
Controller Tester는 CCS가 지원하는 디버깅 스크립트(4.x버전부터)를 사용하여 타깃 환경에서 테스트를 실행하고 결과를 가져옵니다.
CCS에 연결하여 사용할 수 있는 디버깅 장비 목록은 CCS 매뉴얼을 확인하십시오.

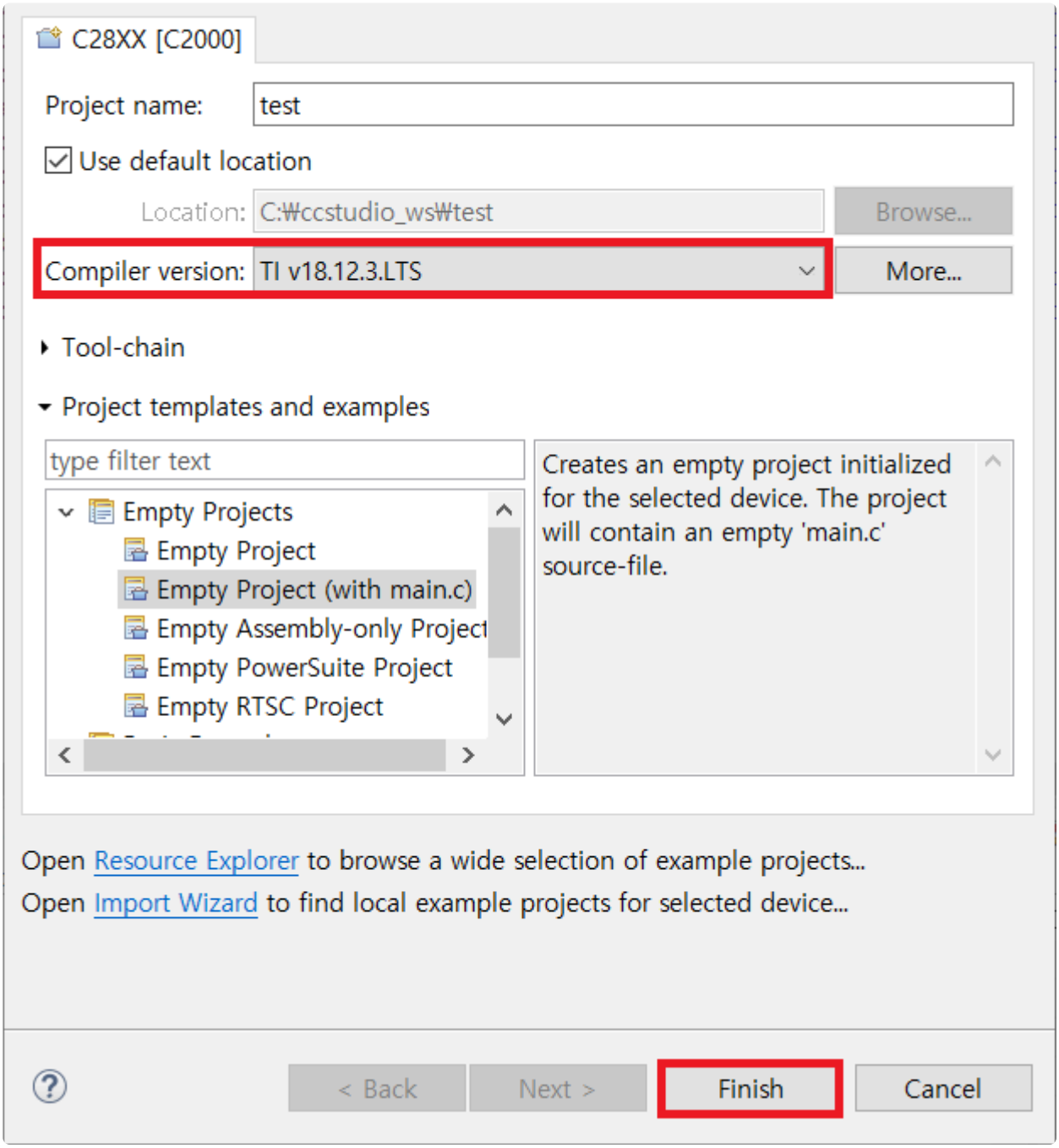
이 문서에서는 CCS에서 프로젝트를 생성하는 것부터 Controller Tester에서 타깃 테스트를 실행하기까지의 과정을 예시와 함께 설명합니다.
예시에서는 Spectrum Digital의 XDS560v2와 Texas Instruments의 TMS320을 사용합니다.

2.5.1. Step1: CCS에서 프로젝트 생성하기

1. CCS를 실행하고 새 프로젝트를 생성합니다. 최상위 메뉴에서 [File]-[New]를 선택한 뒤에 원하는 형태의 프로젝트를 선택합니다. 여기서는 [CCS project]를 클릭하여 프로젝트를 생성합니다. 사용하는 타깃과 디버거 정보를 입력한 후 [verify]를 클릭하면, 정상적으로 연결되었는지 확인할 수 있습니다.



2. 디버거와 타깃 연결을 확인한 후 나머지 설정도 입력합니다. 예시에서는 C2000 Ti 컴파일러를 사용합니다. [Finish]를 클릭하면 CCS 프로젝트가 워크스페이스에 생성됩니다.



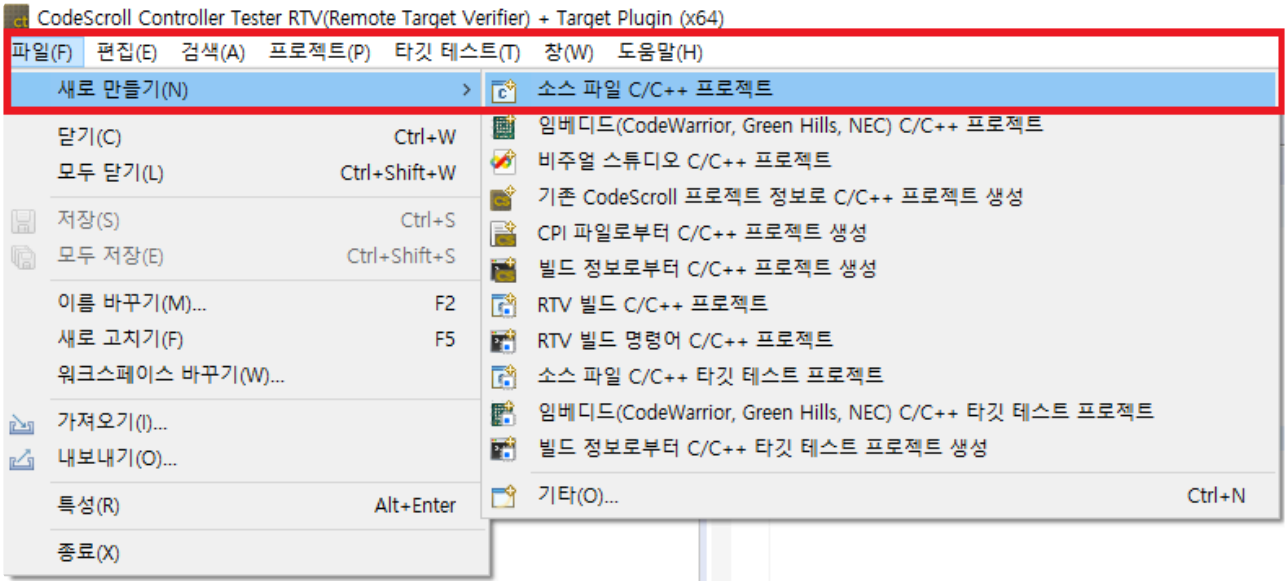
CCS는 Texas Instruments에서 기본으로 제공하는 디버거 외에 몇가지 디버거를 더 지원합니다.

1. TI XDS USB (CCStudio default)
2. BlackHawk JTAG emulator
3. Spectrum digital
4. MSP430 USB
5. MSP432 USB
6. Tiva/Stellaris ICDI

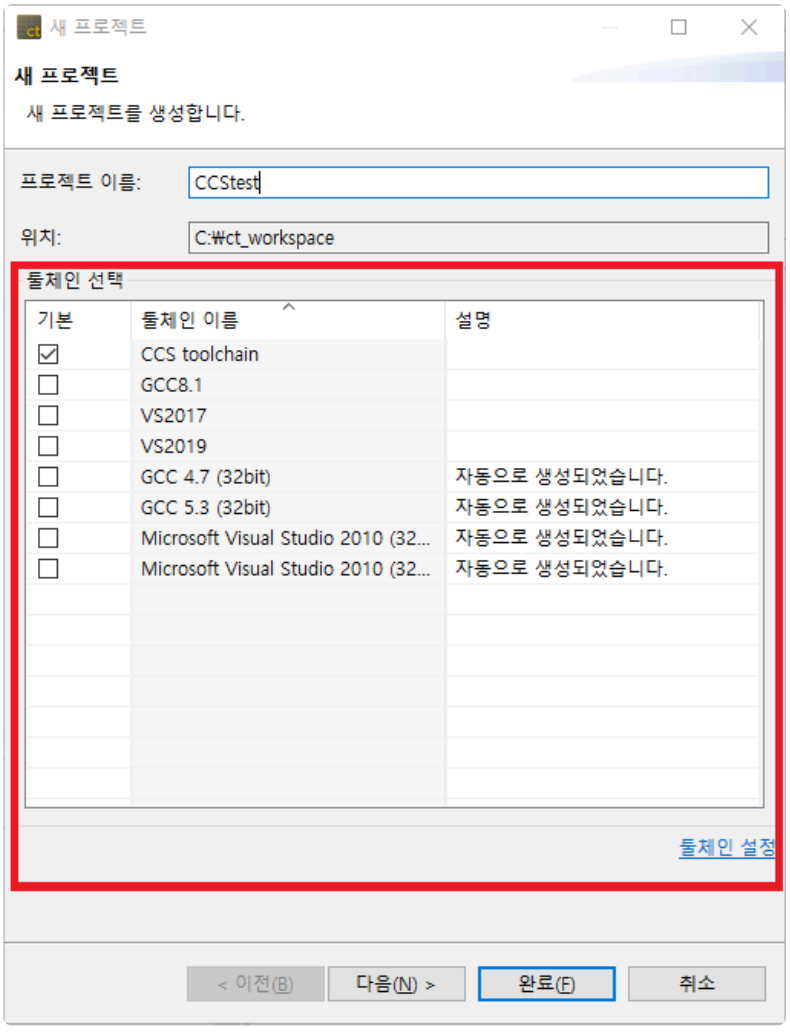
Controller Tester는 CCS에서 지원하는 디버거를 javascript로 제어합니다. CCS의 프로젝트 설정 화면에서 타깃과 디버거 관련 상세 정보를 확인할 수 있습니다.

2.5.2. Step2: Controller Tester에서 타깃 환경 설정하기

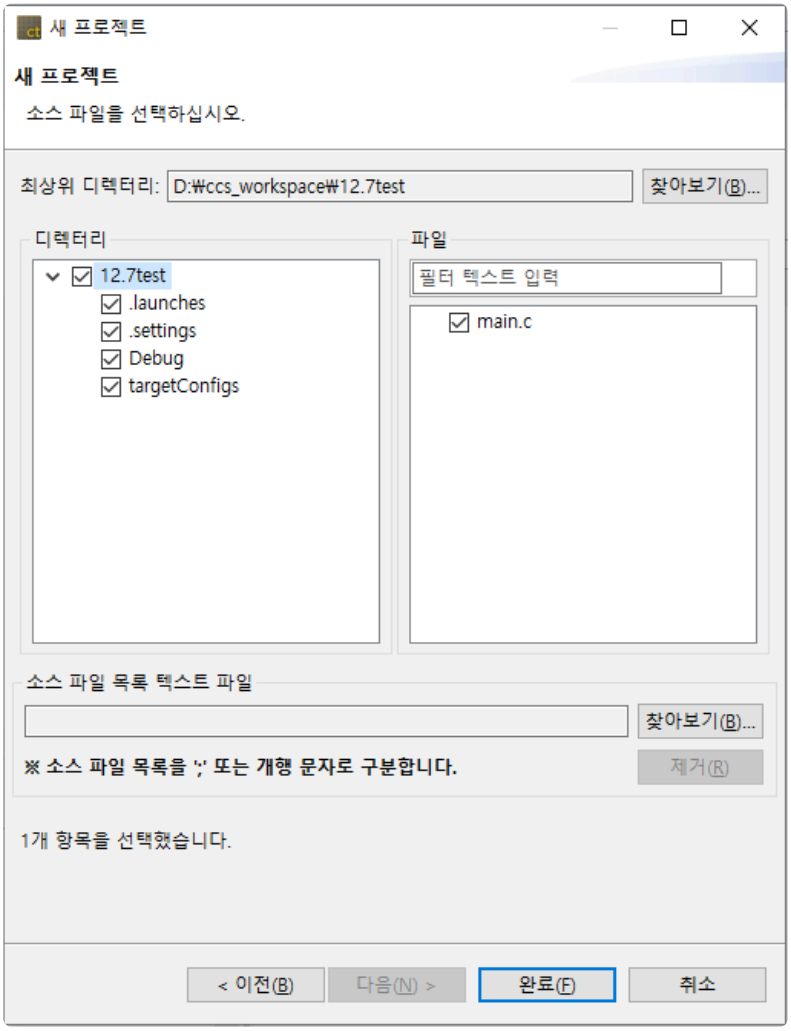
1. CodeScroll Controller Tester 프로젝트를 생성합니다.



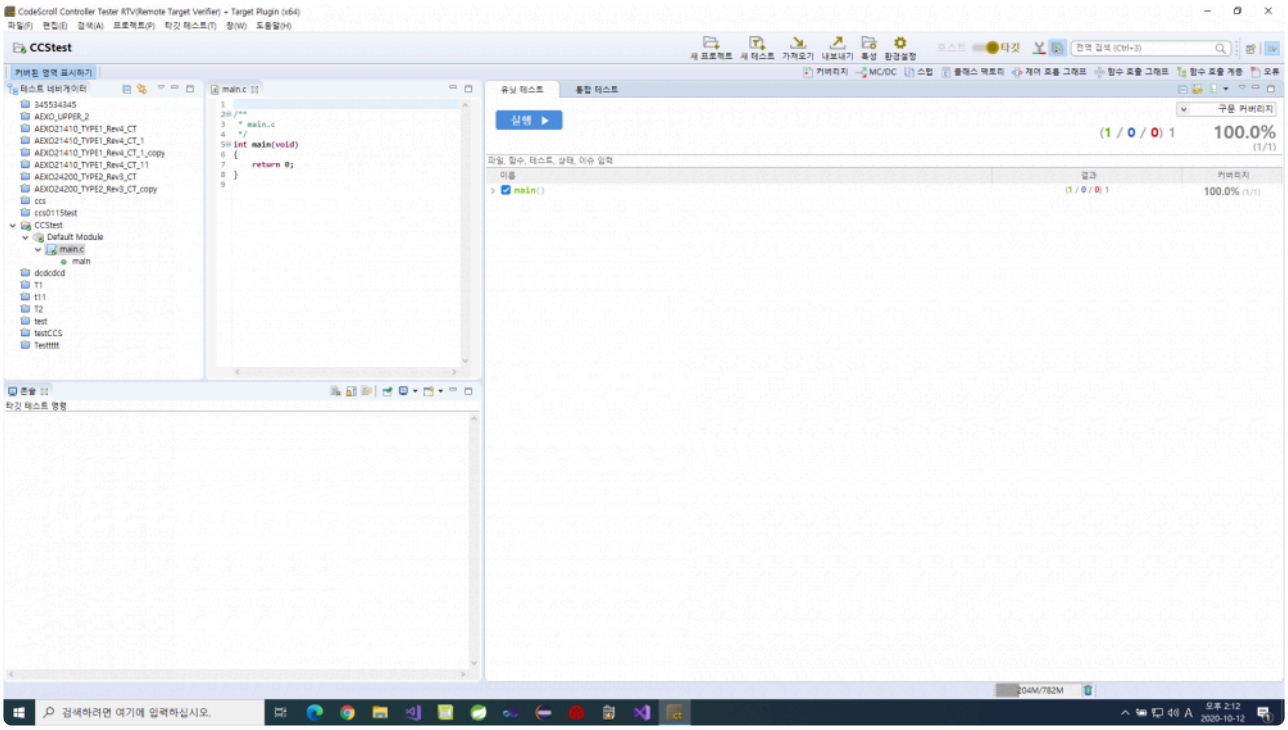
2. 생성한 Code Composer Studio toolchain을 선택합니다.



3. 테스트할 소스 파일을 선택합니다.



4. 완료 후 프로젝트를 생성합니다.

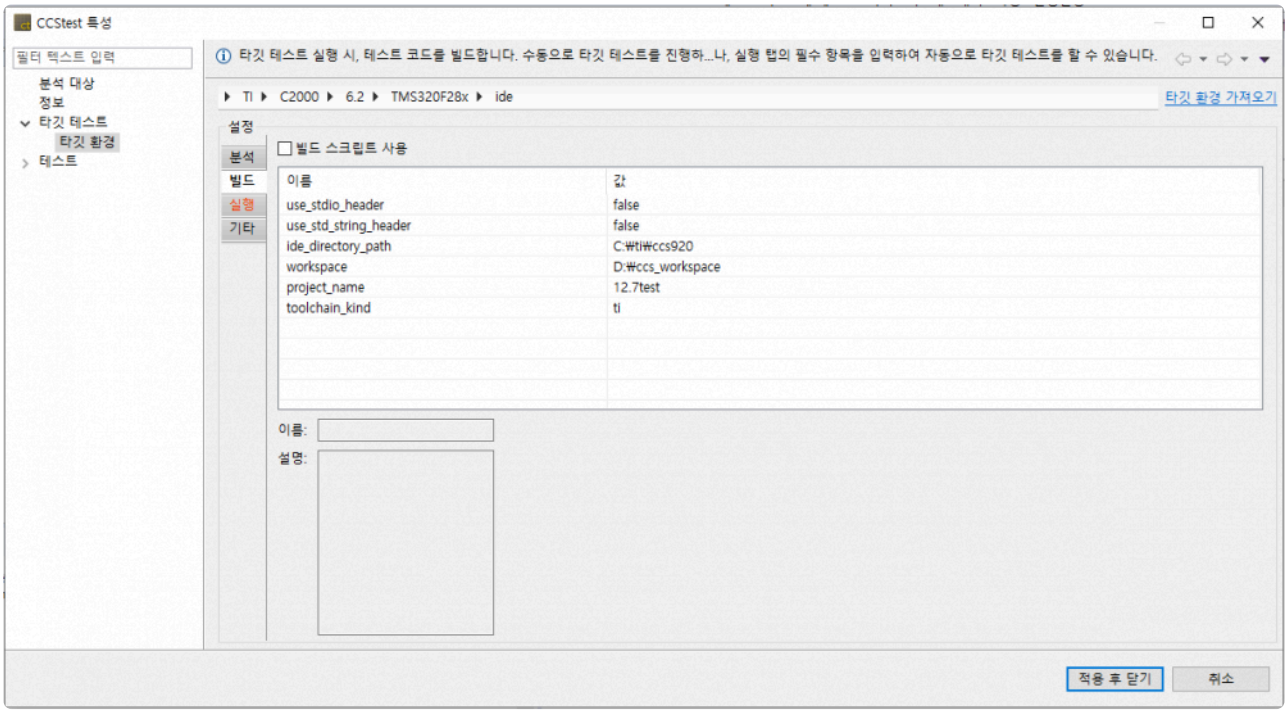


5. CodeScroll Controller Tester 의 해당 프로젝트 우클릭 후 [특성] > [타깃 테스트] > [타깃 환경] 에서 설정

을 해야 합니다.
선택한 정보에 따라 설정 항목들이 표시됩니다. Code Composer Studio 디버거를 사용하여 타깃 테스트를 실행 위해서는 [빌드] 와 [실행] 탭에서 빨간색으로 표시되는 부분에 값을 입력해야 합니다.

6. 디버거를 사용하기 위해 Code Composer Studio, CodeScroll Controller Tester 에서 설정을 합니다.
Controller Tester의 타깃 환경 설정 페이지에서 디버거를 선택합니다. 프로젝트에 선택된 툴체인에 따라 지원하는 디버거 목록만 표시됩니다. 예시에서는 Code Composer Studio 디버거를 사용하기 때문에 IDE 디버거를 선택합니다.
▸ TI ▸ C2000 ▸ 6.2 ▸ TMS320F28x ▸ ide

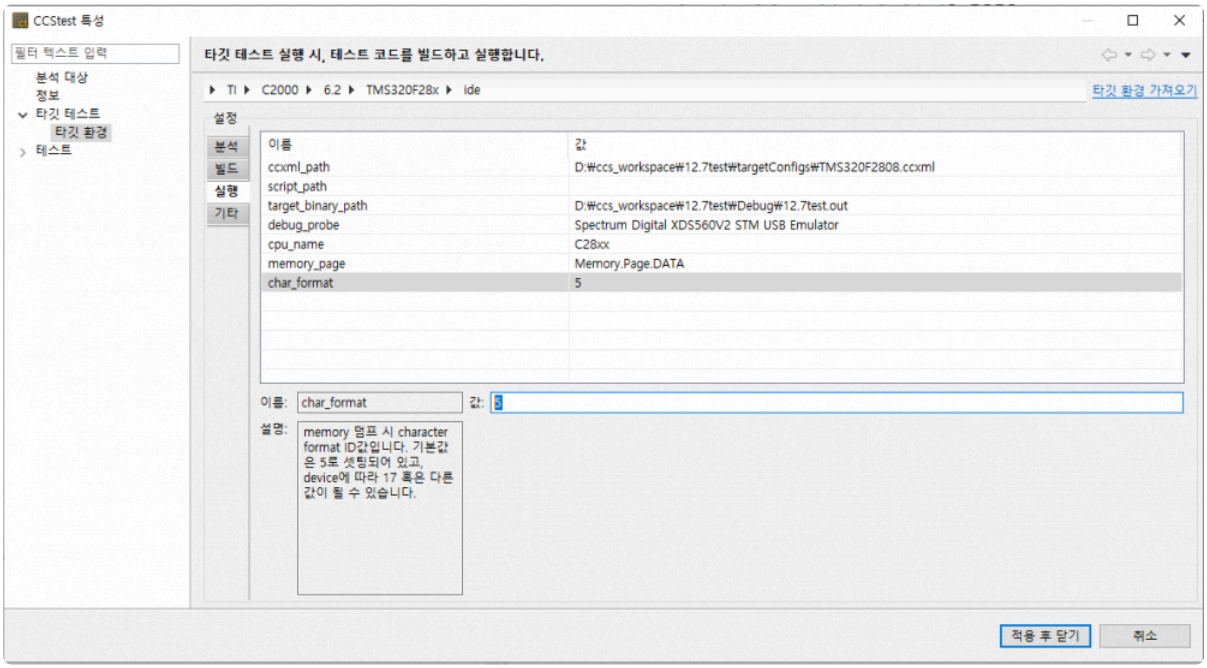
7. 빌드 탭
Code Composer Studio 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.



- 빌드 탭 항목

ide_directory_path	Code Composer Studio의 디렉토리 경로 *ex.C:\ti\ccs930*
workspace	Code Composer Studio의 워크스페이스 디렉토리 경로
project_name	Controller Tester에서 분석할 Code Composer Studio 프로젝트 이름

8. 실행 탭

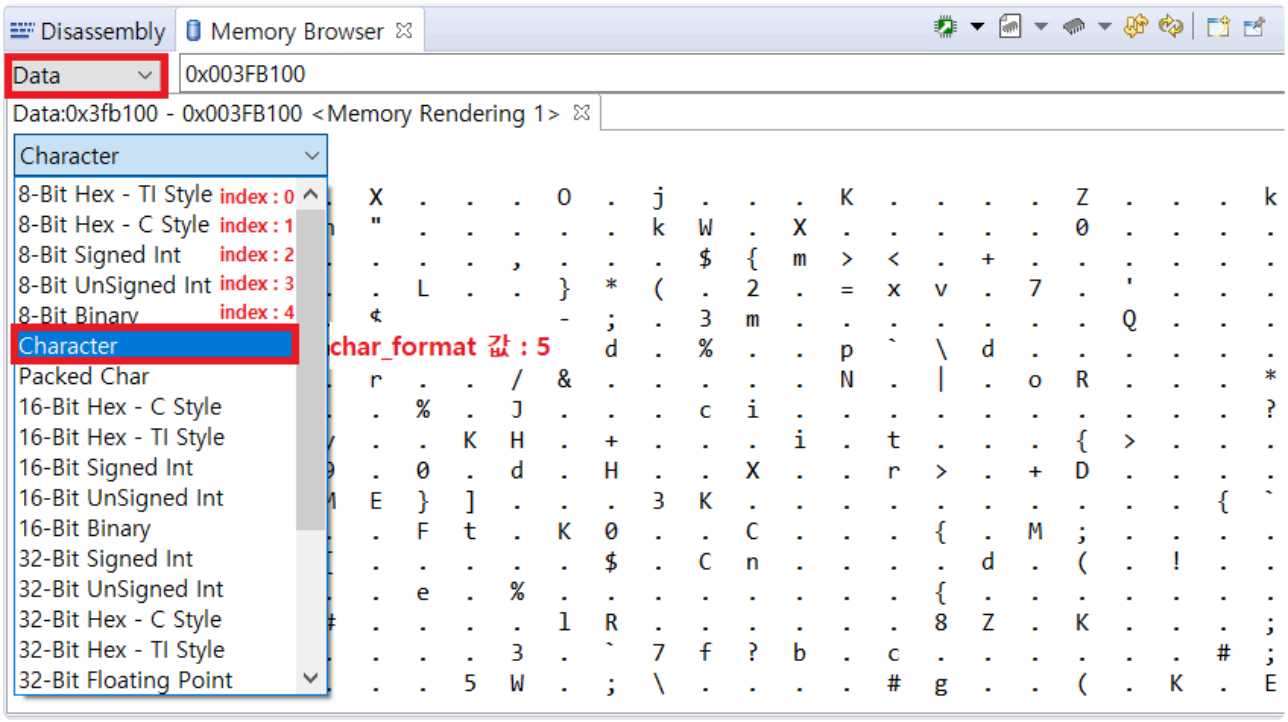


- 실행 탭 항목

ccxml_path	Code Composer Studio 타깃 구성 파일입니다. 이 파일의 이름은 Code Composer Studio에서 설정한 타깃의 이름입니다. 프로젝트 경로와 타깃의 이름이 맞는지 확인합니다. 예: {project_path}\targetConfig\{target_name}.ccxml
script_path	실행 스크립트가 따로 작성되어있을 시, 경로를 입력 (없으면 입력하지 않아도 됩니다.)
target_binary_path	Code Composer Studio 빌드 시 생성되는 바이너리 파일의 경로 예: {project_path}\Debug\{project_name}.out
debug_probe	해당 타겟 디버거의 이름, Code Composer Studio properties의 Device 앞부분 (예시의 사진에서는 Spectrum Digital XDS560V2 STM USB Emulator)
cpu_name	해당 타겟 CPU의 이름, Code Composer Studio properties의 뒷부분 (예시의 사진에서는 C28xx)
memory_page	Code Composer Studio의 디버거가 접근 가능한 memory page , Code Composer Studio Memory Browser 사진 참고
char_format	Code Composer Studio의 데이터 포맷 중 character의 순서. 상단으로부터 1로 시작 예: character가 해당 바의 상단에서부터 5번째에 있으면, char_format 값에 5를 입력하면 됩니다.

- Code Composer Studio properties (Code Composer Studio 프로젝트 우클릭 후 [Properties] > [Debug] > [Device])

- Code Composer Studio Memory Browser



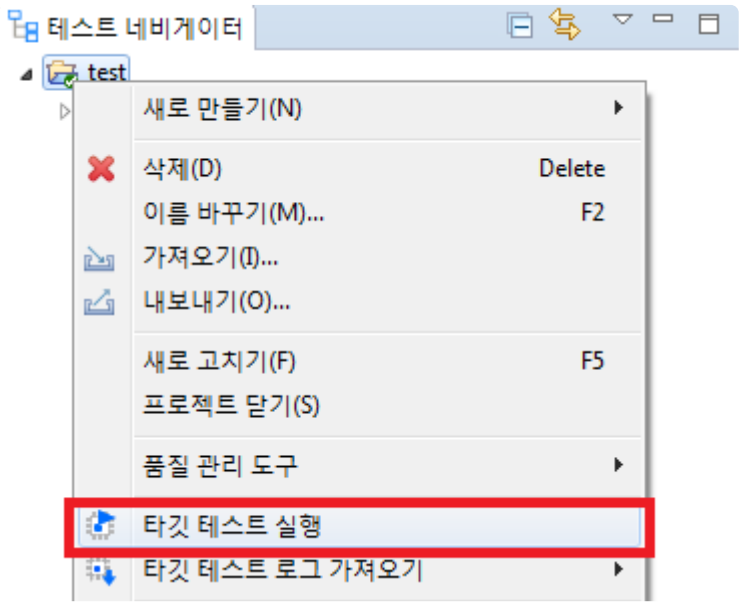
> ! 실행 과정에서 Code Composer Studio가 실행되어 있으면 컴파일 에러가 발생합니다.

타깃 환경 설정이 끝나면 [OK] 또는 [Finish]버튼을 클릭합니다. 타깃 테스트를 수행할 준비가 끝났습니다.

2.5.3. Step3: 타깃 테스트 실행하기

타깃 테스트를 실행하기에 앞서 빌드하고자 하는 프로젝트가 위치한 워크스페이스의 사용을 종료해야 합니다.
IDE에서 워크스페이스를 사용 중이라면 타깃 테스트가 정상적으로 수행되지 않습니다.
테스트 네비게이터 뷰의 프로젝트 컨텍스트 메뉴에서 [타깃 테스트 실행하기]를 선택하거나 테스트 뷰의 [실행] 버튼을 클릭하여, 타깃 테스트를 실행할 수 있습니다.

- [타깃 테스트 실행하기]



- [실행]

CCS의 디버그 스크립팅에 대한 자세한 정보는 Texas Instruments 홈페이지 를 참고하십시오.

2.5.4. 디버거를 통해 타깃 테스트 디버깅하기

1. ‘타깃 테스트 코드 내보내기’ 수행
2. CT_워크스페이스\.metadata\.plugins\com.codescroll.ut.embedded\프로젝트명\TestFixture\cs 경로로 이동
3. notepad로 해당 경로 아래에 있는 모든 파일들의 내용에서 #line 문자를 //#line으로 변경(#line -> //#line)
4. Code Composer Studio에서 내보내기 된 코드 빌드 수행
5. 디버깅 모드로 실행하면 cs_tfx.c 파일의 main 함수에 break point가 걸리는 것을 확인
6. Code Composer Studio 에서 좌측 상단에 ‘File’ > ‘Open File’ 클릭
7. 2번의 경로에서 테스트함수_test숫자.c 파일 열기(ex. zlibVersion_test0.c 파일)
8. ‘Open File’로 테스트 대상 함수의 정의가 있는 소스파일_숫자.c 파일도 열기(ex. inffast_1.c 파일)
9. 7번에 있는 함수에 break point를 지정, 8번에 있는 테스트 대상 함수의 시작부에 break point 지정
10. 7번과 8번에서 열었는 파일에서 break point가 걸리는 것을 확인할 수 있음(8번의 경우 함수 정의 시작부분에 break point를 지정)
11. 디버깅 수행

> ! Code Composer Studio에서는 탐침이 들어간 상태의 코드로만 디버깅을 할 수 있습니다.
> 또한 Code Composer Studio에서 디버깅이 가능한 환경이 구성되어 있지 않으면 디버깅을 할 수 없습니다.

3. Controller Tester 타깃 빌드 가이드

CodeScroll Controller Tester 타깃 프로젝트 정보를 사용하여 타깃 테스트 코드를 빌드하는 방법을 안내합니다.

- IAR Embedded Workbench IDE Build
- Texas Instruments CodeComposer, all versions
- CodeWarrior IDE
- Hightec Development Platform IDE
- Tasking VX IDE
- Renesas CS+ IDE
- MPLAB X IDE

3.1. IAR Embedded Workbench IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

IAR Embedded Workbench 빌드를 하기 위해서는 타깃 환경 설정의 분석 및 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 분석 탭

cpu	Processor variant의 Core에서 선택할 수 있는 타겟의 cpu

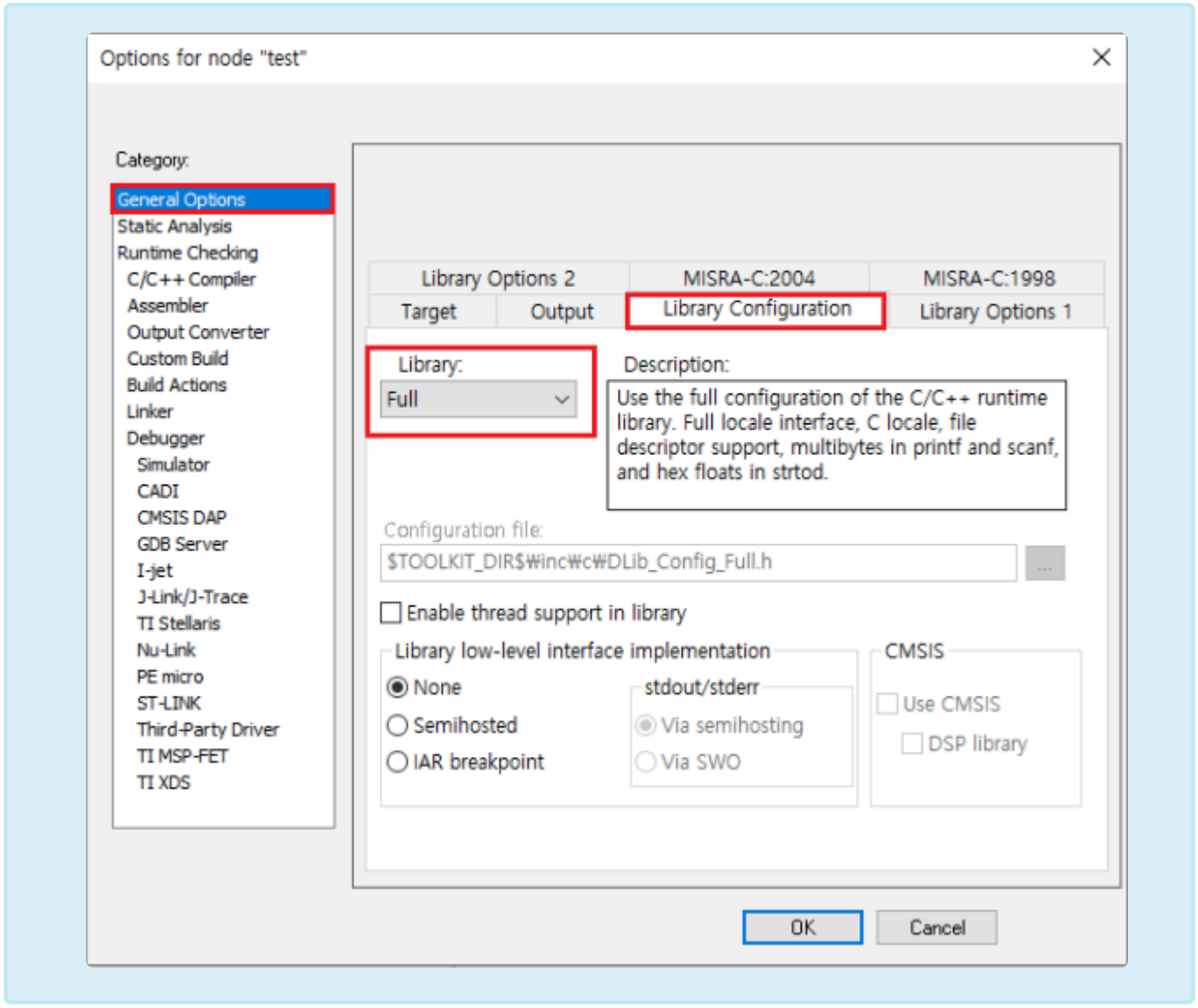
- 빌드 탭

ide_directory_path	IAR Embedded Workbench IDE 설치 경로 *ex. C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.4*
project_file_path	IAR Embedded Workbench 프로젝트 파일(.ewp) 경로

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

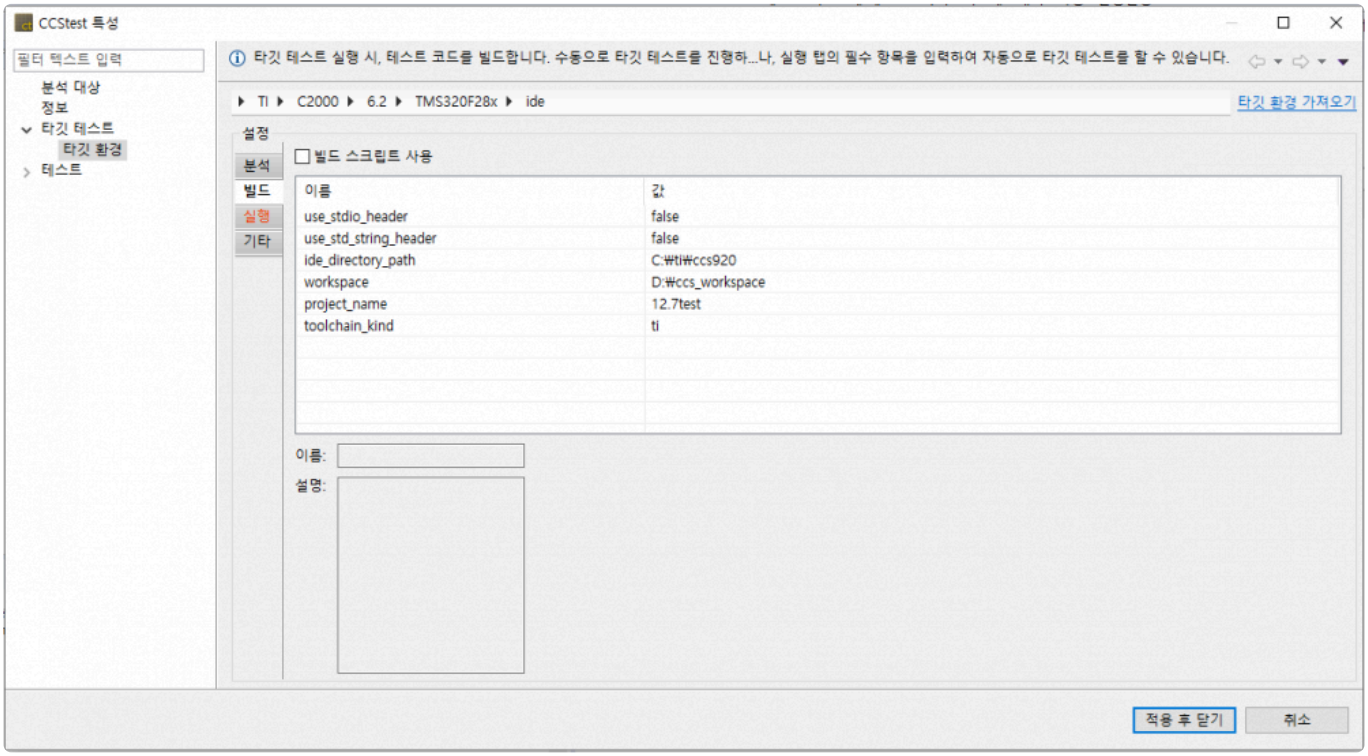
✻ stdio.h 의 IO 함수 사용 시 라이브러리 설정 변경이 필요합니다.
워크스페이스의 프로젝트 우클릭 -> Options -> General Options -> Library Configuration -> Library tab 을 Full로 변경합니다.


Options for node "test"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet
J-Link/J-Trace
TI Stellaris
Nu-Link
PE micro
ST-LINK
Third-Party Driver
TI MSP-FET
TI XDS
Library Options 2
MISRA-C:2004
MISRA-C:1998
Target
Output
Library Configuration
Library Options 1
Library:
Full
Description:
Use the full configuration of the C/C++ runtime library. Full locale interface, C locale, file descriptor support, multibytes in printf and scanf, and hex floats in strtod.
Configuration file:
$TOOLKIT_DIR$¥inc¥c¥DLib_Config_Full.h
Enable thread support in library
Library low-level interface implementation
None
Semihosted
IAR breakpoint
stdout/stderr
Via semihosting
Via SWO
CMSIS
Use CMSIS
DSP library
OK
Cancel

3.2. Texas Instruments CodeComposer, all versions

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

Code Composer Studio 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.



- 빌드 탭

ide_directory_path	Code Composer Studio의 디렉토리 경로 *ex.C:\ti\ccs930*
workspace	Code Composer Studio의 워크스페이스 디렉토리 경로
project_name	Controller Tester에서 분석할 Code Composer Studio 프로젝트 이름

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

> ! 실행 과정에서 Code Composer Studio가 실행되어 있으면 컴파일 에러가 발생합니다.

3.3. CodeWarrior IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

CodeWarrior 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

ide_directory_path	CodeWarrior IDE 설치 경로 *ex. C:\Program Files (x86)\Freescale\CW for MPC55xx and MPC56xx 2.10, C:\Freescale\CW MCU*
ide_version	IDE 버전, Classic 또는 Eclipse
project_file_path	Classic의 경우 프로젝트 생성 시 명명한 .mcp 파일, Eclipse의 경우 프로젝트 생성 시 만들어진 .project 파일

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

3.4. Hightec Development Platform IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

Hightec IDE 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

ide_directory_path	Hightec IDE 설치 경로 *ex. C:\HIGHTEC\toolchains\arm\v4.6.5.0*
project_directory_path	HighTec IDE에서 생성한 프로젝트 디렉터리의 경로

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

3.5. Tasking VX IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

Tasking VX IDE 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

ide_version	설치된 Tasking VX IDE의 버전
makefile_path	Tasking 프로젝트에 생성된 makefile의 경로
ide_directory_path	Tasking VX IDE가 설치된 디렉토리 경로 *ex. C:\Program Files (x86)\TASKING\C166-VX v3.1r2*

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

3.6. Renesas CS+ IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

Renesas CS+ IDE 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

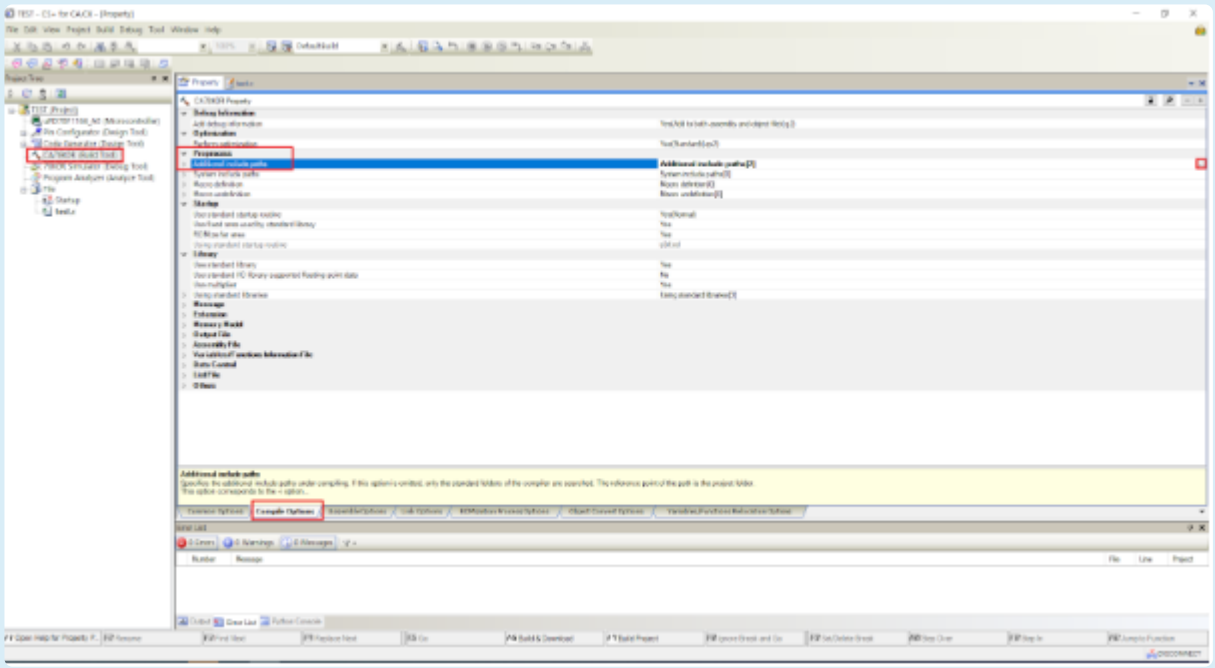
- 빌드 탭

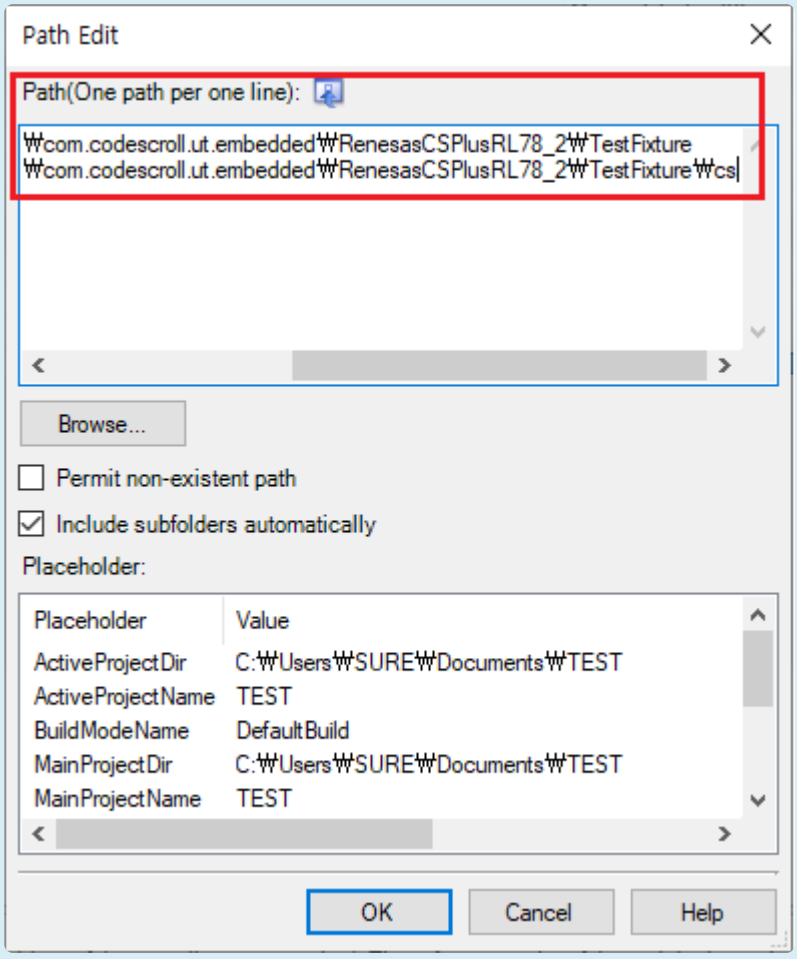
ide_directory_path	Renesas CS+ 설치 디렉토리 경로 *ex. C:\Program Files (x86)\Renesas Electronics*
ide_kind	IDE 종류(CS+)
workspace_path	Renesas HEW IDE의 경우만 필요하므로,CS+에서는 임의의 경로 지정
project_file_path	Renesas CS+에서 생성한 프로젝트 파일 경로(.mtpj)

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

Controller Tester에서 테스트 코드들을 내보낼 때 일부 코드는 상대 경로를 참조합니다.
타깃 테스트 코드를 빌드하기 위해서는 Renesas CS+ 프로젝트에 해당 경로의 절대 경로를 참조하도록 설정해야 합니다.
- Build Tool의 Property -> Compile Options -> Preprocess -> Additional include paths 에서 아래의 경로 추가
(CTWORKSPACE) \.metadata\.plugins\com.codescroll.ut.embedded\ *CT project name* \TestFixture
(CTWORKSPACE) \.metadata\.plugins\com.codescroll.ut.embedded\ *CT project name* \TestFixture\cs




Path Edit
Path(One path per one line):
₩com.codescroll.ut.embedded₩RenesasCSPlusRL78_2₩TestFixture
₩com.codescroll.ut.embedded₩RenesasCSPlusRL78_2₩TestFixture₩cs
Browse...
Permit non-existent path
Include subfolders automatically
Placeholder:
Placeholder
Value
ActiveProjectDir
C:₩Users₩SURE₩Documents₩TEST
ActiveProjectName
TEST
BuildModeName
DefaultBuild
MainProjectDir
C:₩Users₩SURE₩Documents₩TEST
MainProjectName
TEST
OK
Cancel
Help

3.7. MPLAB X IDE

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

MPLAB X IDE 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

ide_directory_path	MPLAB X IDE 설치 경로 *ex. C:\Program Files (x86)\Microchip\MPLABX\v5.35*
project_directory_path	MPLAB X IDE에서 생성한 프로젝트 디렉터리 경로

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

3.8. Microsoft Visual Studio

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

Microsoft Visual Studio 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

ide_directory_path	Microsoft Visual Studio 설치 경로 *ex. C:\Program Files (x86)\Microsoft Visual Studio 10.0*
build_configuration	대상 솔루션의 테스트할 구성 및 플랫폼
sin_path	대상 솔루션의 파일 경로 (.sin 파일)

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

3.9. GNU Compiler

타깃 환경 설정 페이지는 사용자가 선택한 툴체인에 따라 자동으로 정보가 채워집니다. 선택 가능한 디버거의 종류는 툴체인 분석 설정에 따라 달라집니다.

GNU Compiler 빌드를 하기 위해서는 타깃 환경 설정의 빌드 탭에서 필요한 정보들을 입력하고 완료를 누릅니다. 작성해야 하는 항목은 아래 표와 같으며, 이는 필수 항목입니다.

- 빌드 탭

makefile_path	사용자가 작성한 makefile의 경로

위 항목을 작성하지 못하고 타깃 환경 설정 페이지의 완료 버튼을 누른 경우나 경로가 변경된 경우에는 테스트 네비게이터의 프로젝트 마우스 우클릭 -> 특성 -> 타깃 테스트 -> 타깃 환경에서 다시 설정할 수 있습니다.

타겟 환경 설정을 마치고 유닛 테스트 뷰의 실행 버튼을 누르면 Controller Tester에서 타겟 테스트 코드를 빌드합니다.

4. 다른 사용자와 프로젝트 공유하기

Controller Tester에서 사용하는 프로젝트를 다른 사람과 공유할 수 있습니다.
Controller Tester 3.3 이상에서는 [프로젝트 내보내기], [프로젝트 가져오기] 기능을 사용합니다.

- Controller Tester 3.3 이후/ 프로젝트 공유 시 가이드
- Controller Tester 3.2 이전/ RTV 프로젝트 공유 시 가이드

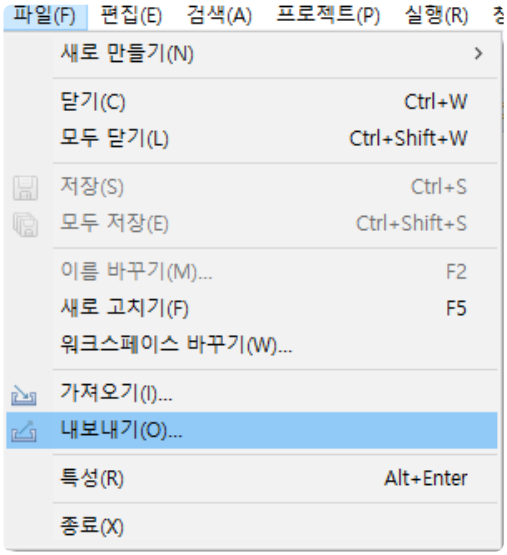
4.1. (Controller Tester 3.3 이후) 프로젝트 공유 시 가이드

Controller Tester 3.3 이후부터는 [프로젝트 내보내기], [프로젝트 가져오기] 기능으로 손쉽게 프로젝트를 공유할 수 있습니다.

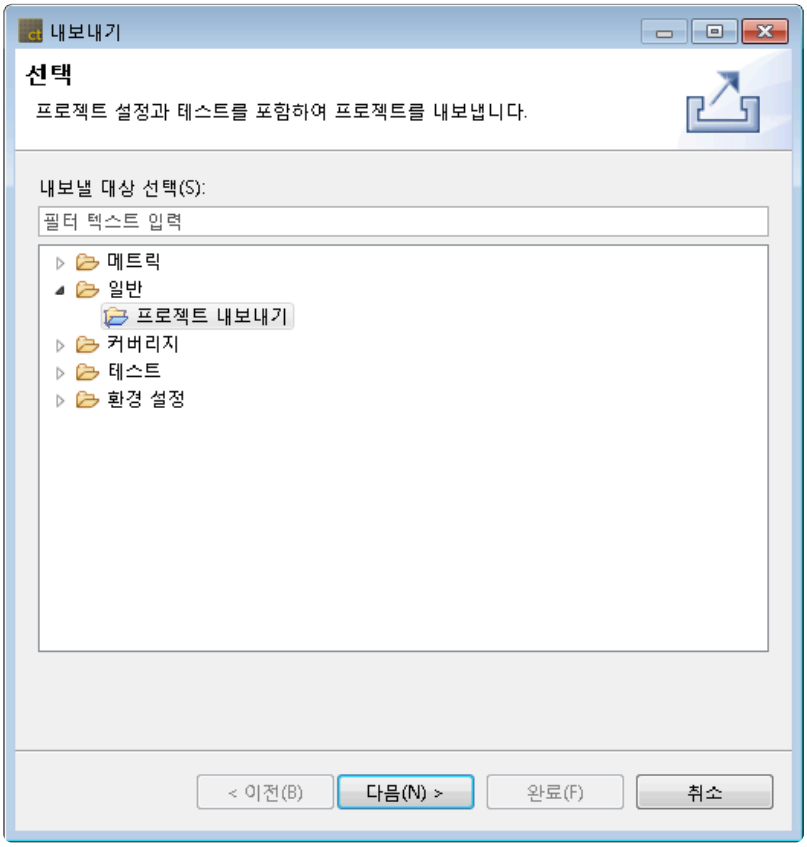
4.1.1. 프로젝트 내보내기

프로젝트 설정과 테스트를 포함하여 프로젝트를 내보낼 수 있습니다.

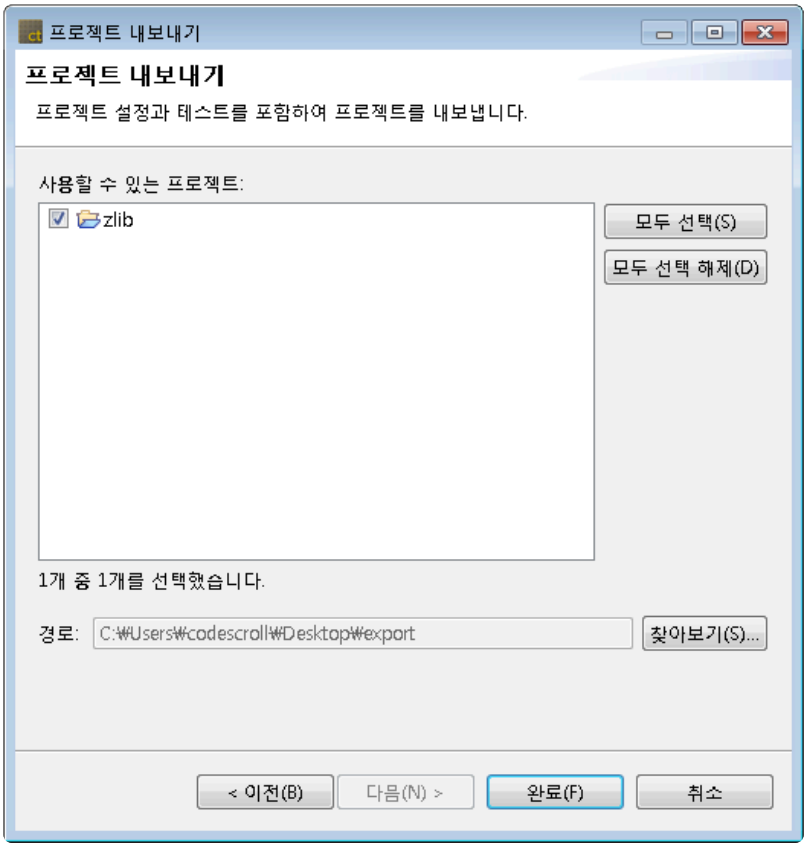
1. 메인 메뉴에서 [파일] > [내보내기]를 클릭합니다. 내보내기 마법사가 열립니다.



2. [일반] > [프로젝트 내보내기]를 클릭합니다.



3. 내보내기 대상 프로젝트와 내보낼 경로를 선택한 후, [완료] 버튼을 클릭합니다.



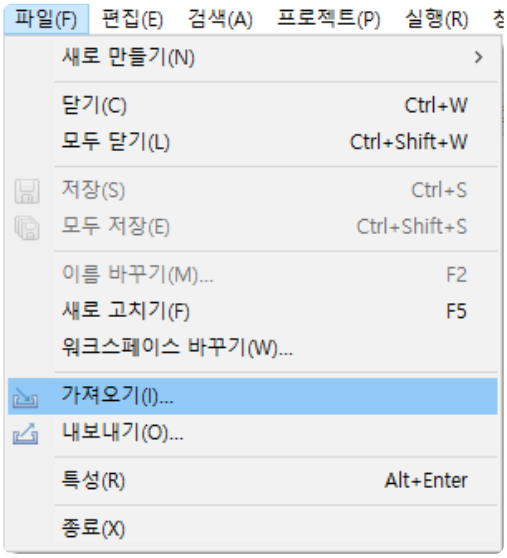
4. 내보낸 경로에 내보내기한 프로젝트 이름이 포함된 폴더가 생긴 것을 확인할 수 있습니다. 폴더를 압축하여 공유하고자 하는 사용자의 컴퓨터로 옮깁니다.

4.1.2. 프로젝트 가져오기

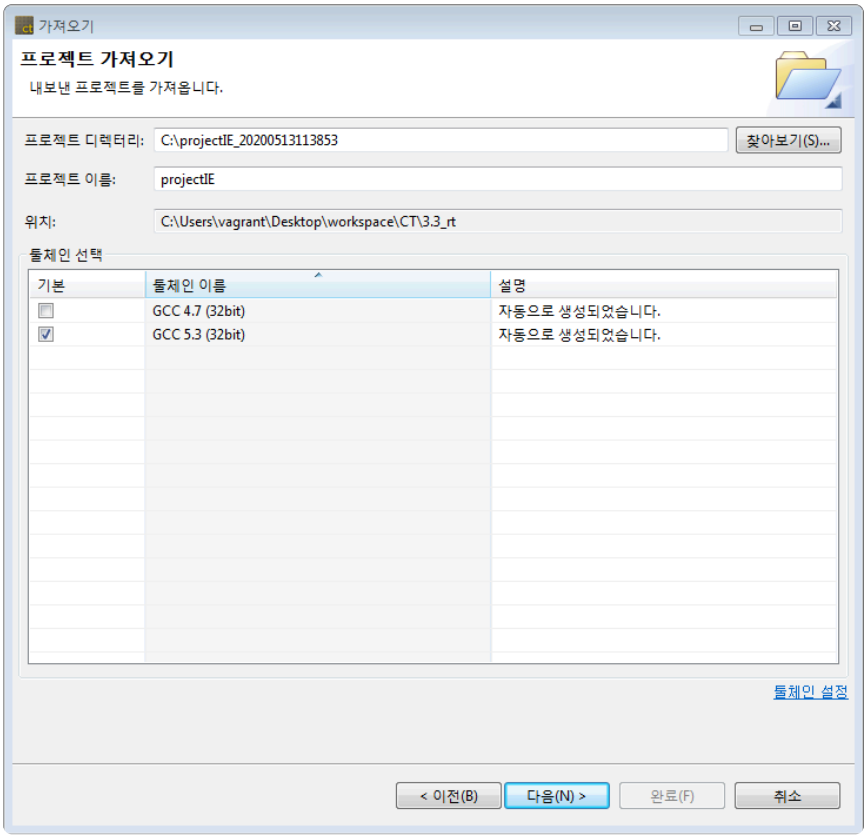
프로젝트 가져오기 기능을 이용하여, 다른 PC에서 내보낸 프로젝트를 워크스페이스로 가져올 수 있습니다.

일반 프로젝트 가져오기

1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.

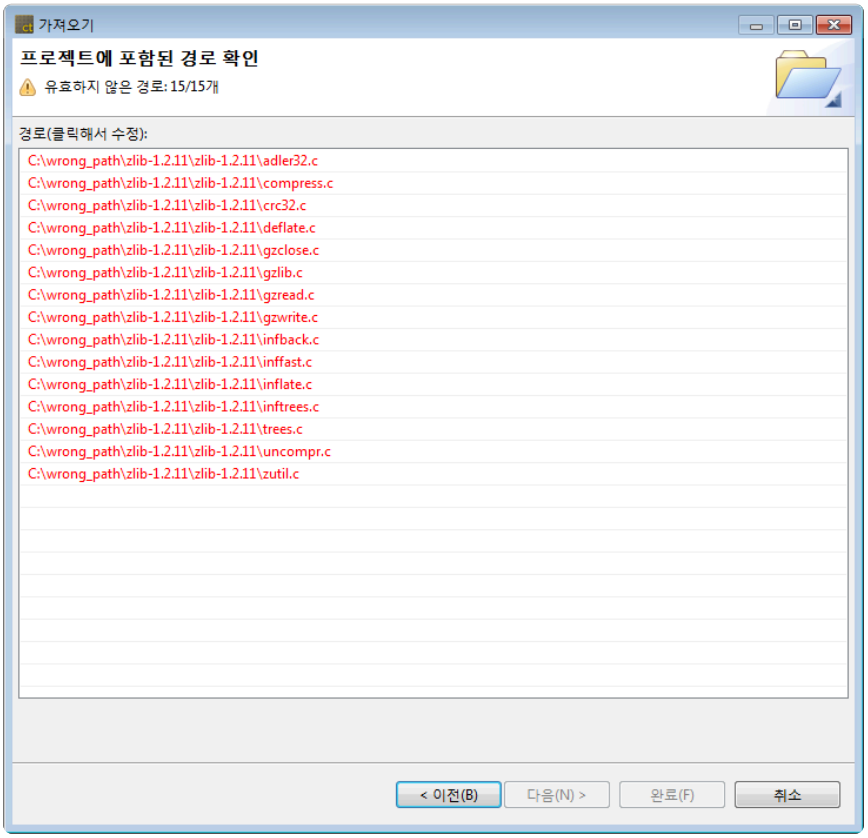


2. [일반] -> [프로젝트 가져오기]를 클릭한 후 [다음] 버튼을 클릭합니다.
3. [찾아보기] 버튼을 클릭하여 내보낸 프로젝트에 해당하는 디렉터리를 찾습니다.
4. 디렉터리를 선택하면 가져오기할 프로젝트 정보로부터 툴체인이 자동으로 선택됩니다. 동일한 이름의 프로젝트가 이미 워크스페이스에 존재하는 경우, 프로젝트 이름을 수정해야 합니다.

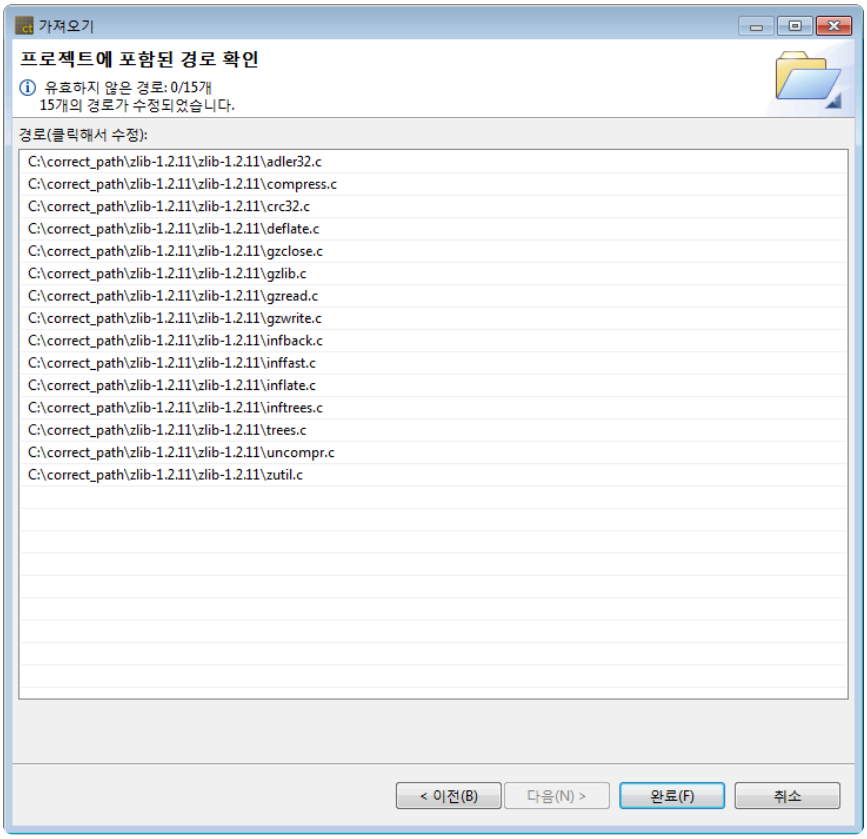


> 가져올 프로젝트의 툴체인과 동일한 이름의 툴체인이 없는 경우, 먼저 가져올 프로젝트의 툴체인을 내보낸 후 가져와야 합니다. 자세한 내용은 CodeScroll Controller Tester 문서의 [툴체인 가져오기] 및 [툴체인 내보내기] 항목을 참조하십시오.

5. [다음]버튼을 클릭합니다.
6. 가져오기할 프로젝트에 포함된 소스 경로를 확인할 수 있습니다. 유효하지 않은 경로는 빨간색으로 표시되며, 경로 창을 클릭하여 수정할 수 있습니다.



7. 유효하지 않은 경로가 있을 경우, 하나의 파일 경로를 수정하면 연관된 파일 경로가 자동으로 수정됩니다. 이 때 상단에서 수정된 경로의 개수를 확인할 수 있습니다.



✻ 윈도우 절대 경로 형식이 아닌 경우에는 경로가 유효한지 여부를 검사하지 않습니다.

8. [완료]버튼을 클릭합니다.

RTV 프로젝트 가져오기

RTV C/C++ 프로젝트는 일반 C/C++ 프로젝트 가져오기와 동일한 방법으로 가져올 수 있습니다.

1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사에서 [일반] -> [프로젝트 가져오기]를 선택하고 [다음]을 누릅니다.
2. [찾아보기] 버튼을 클릭하여 가져올 프로젝트의 디렉터리를 선택합니다. 디렉터리를 선택하면 가져오기할 프로젝트 정보로부터 툴체인이 자동으로 선택됩니다. [다음] 버튼을 클릭합니다.

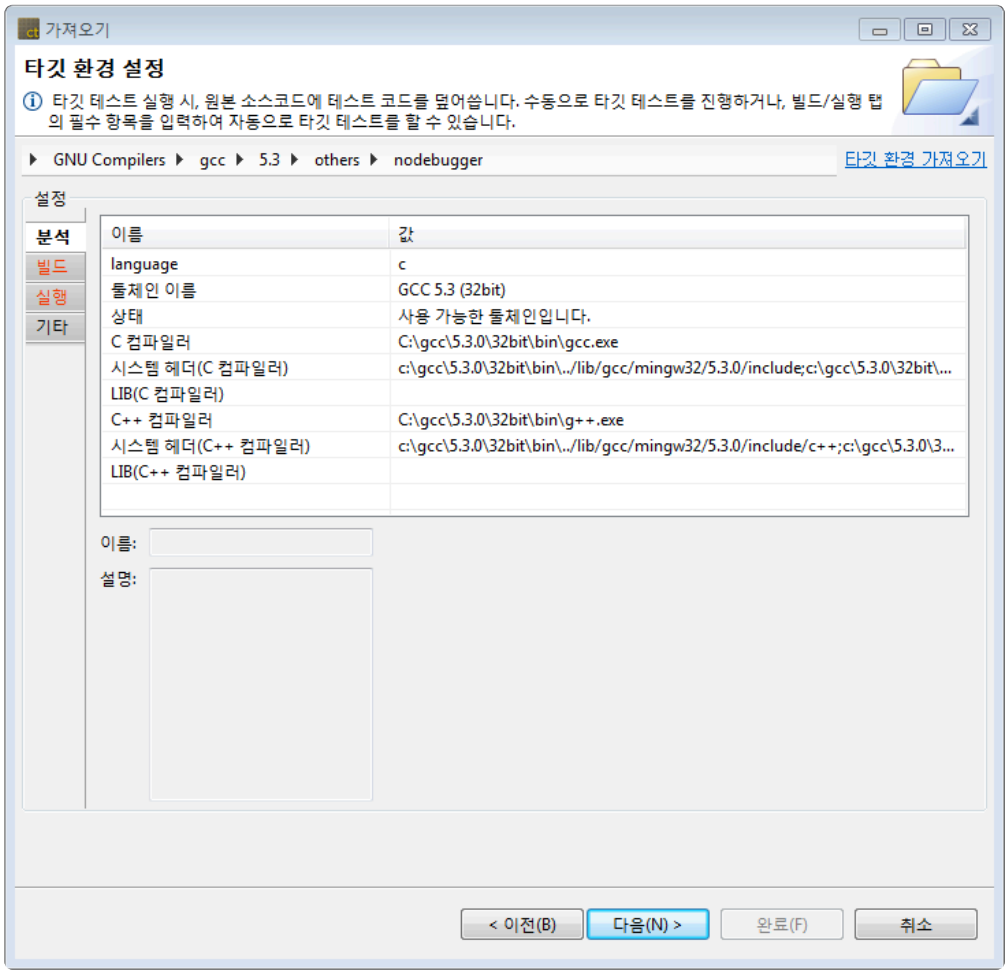
✻ 가져올 프로젝트와 동일한 RTV 서버 및 툴체인 정보가 없으면, 가져올 프로젝트로부터 자동으로 RTV 서버 및 툴체인 정보를 생성합니다.

3. 가져오기할 프로젝트에 포함된 소스 경로를 확인할 수 있습니다. 유효하지 않은 경로는 빨간색으로 표시되며, 경로 창을 클릭하여 수정할 수 있습니다.
4. [완료]버튼을 클릭합니다.

타깃 프로젝트 가져오기

타깃 C/C++ 프로젝트를 가져올 때는 타깃 환경 설정을 추가로 작성해야 합니다.

1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사에서 [일반] -> [프로젝트 가져오기]를 선택하고 [다음]을 누릅니다.
2. [찾아보기] 버튼을 클릭하여 가져올 프로젝트의 디렉터리를 선택합니다. 디렉터리를 선택하면 가져오기할 프로젝트 정보로부터 툴체인이 자동으로 선택됩니다. [다음] 버튼을 클릭합니다.
3. 타깃 프로젝트의 경우 [타깃 환경 설정] 창이 뜹니다. 가져올 프로젝트 정보로부터 타깃 환경 설정을 불러오며, 유효하지 않은 경로를 가진 항목은 빨간색으로 표시됩니다.



✻ 타깃 C/C++ 프로젝트가 아니더라도 타깃 환경 설정을 포함하고 있는 프로젝트라면, [프로젝트 가져오기]를 수행할 때 타깃 환경 설정 창이 뜹니다.

! 유효하지 않은 경로를 포함하고 있어도 타깃 환경 설정을 완료하고 다음으로 넘어갈 수 있으나, 원클릭 타깃 테스트 실행이 안될 수 있습니다.

4. 타깃 환경 설정을 완료하고 [다음] 버튼을 클릭합니다.
5. 가져오기할 프로젝트에 포함된 소스 경로를 확인할 수 있습니다. 유효하지 않은 경로는 빨간색으로 표시되며, 경로 창을 클릭하여 수정할 수 있습니다.
6. [완료]버튼을 클릭합니다.

4.2. (Controller Tester 3.2 이전) RTV 프로젝트 공유 시 가이드

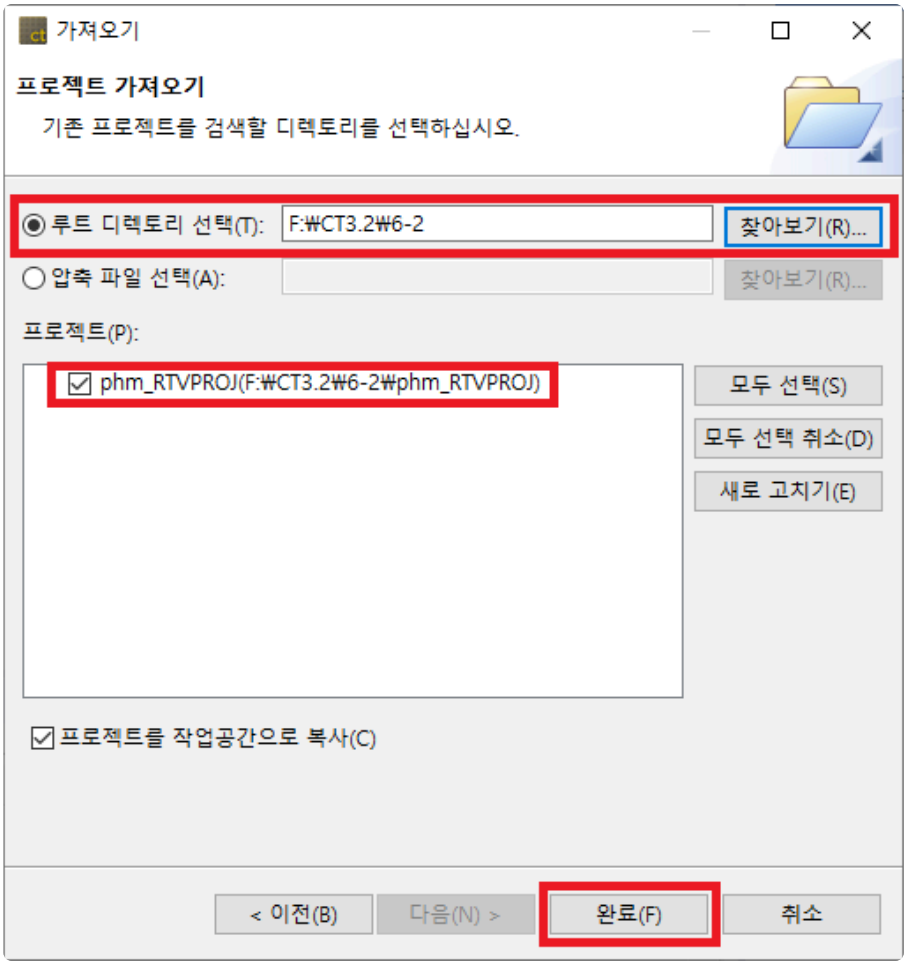
RTV 프로젝트는 툴체인과 소스 파일 정보를 RTV 서버에서 가져올 수 있기 때문에 쉽게 프로젝트를 공유할 수 있습니다.
사용 환경에 따른 단계별 시나리오는 다음과 같으며, 해당 시나리오대로 수행했을 때 RTV 프로젝트를 공유할 수 있습니다.

- 프로젝트 공유 시나리오
- RTV 서버 사용 가이드

4.2.1. 프로젝트 공유 시나리오

[기존 프로젝트를 워크스페이스로 가져오기] 기능을 사용할 경우

1. RTV 프로젝트를 생성하면 Controller Tester 워크스페이스 아래에 RTV 프로젝트 디렉토리(이하 RTV_A 프로젝트)가 생성됩니다.
2. 프로젝트를 공유받고자 하는 사용자가 위 단계에서 생성된 RTV_A 프로젝트 디렉토리를 전달받아, 자신이 사용하는 Controller Tester 워크스페이스 경로에 RTV_A 프로젝트 디렉토리를 복사 후 붙여넣습니다.
3. [가져오기] > [일반] > [기존 프로젝트를 워크스페이스로] 기능을 사용하여 해당 프로젝트 디렉토리 최상위 경로를 선택해 가져옵니다.



4. 프로젝트에 필요한 정보들을 RTV 서버에서 받아오게 되며, [프로젝트의 툴체인 또는 리소스 설정이 올바르지 않습니다. 자동으로 재설정하시겠습니까?] 라는 메세지 창이 뜨는 경우 '예'를 클릭하면 자동으로 RTV 설정(RTV 서버와 프로젝트 생성 시 사용한 툴체인 등록)이 완료됩니다.
5. Controller Tester 테스트 네비게이터 뷰에 RTV_A와 동일한 이름의 RTV 프로젝트(이하 RTV_A' 프로젝트)가 생성된걸 확인할 수 있습니다.
6. [테스트 네비게이터 뷰]의 RTV_A' 프로젝트를 우클릭하고 [재분석]을 수행합니다.
7. 동일한 RTV 서버와 연결되어 있을 때 수행해야 합니다.

[RTV 빌드 C/C++ 프로젝트 생성] 기능을 사용할 경우

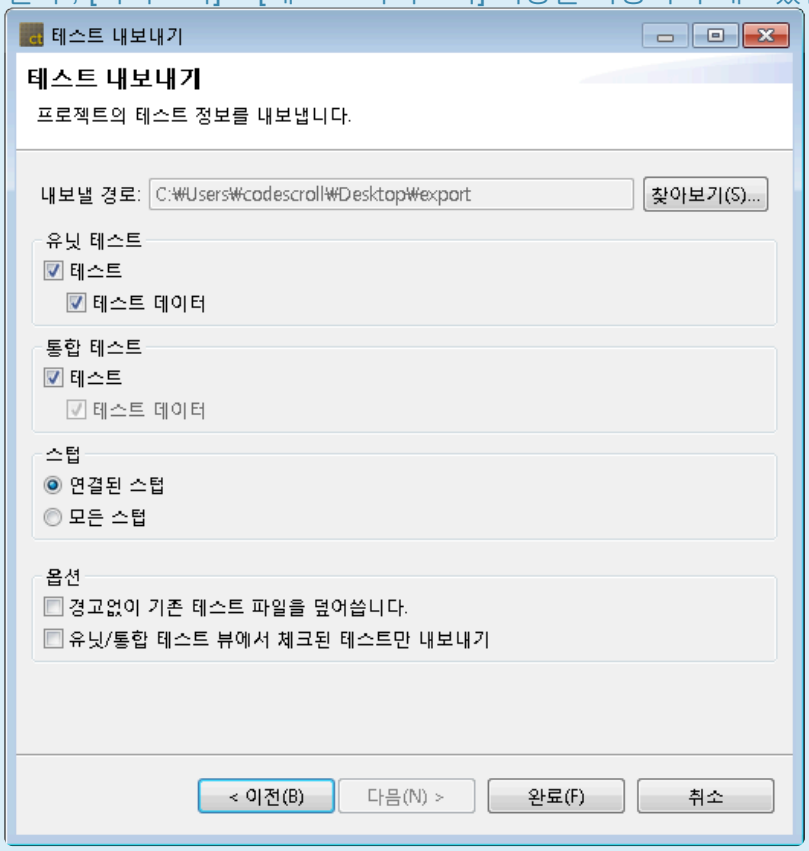
1. RTV 프로젝트를 생성하면 Controller Tester 워크스페이스 아래에 RTV 프로젝트 디렉토리(이하 RTV_A 프로젝트)가 생성됩니다.

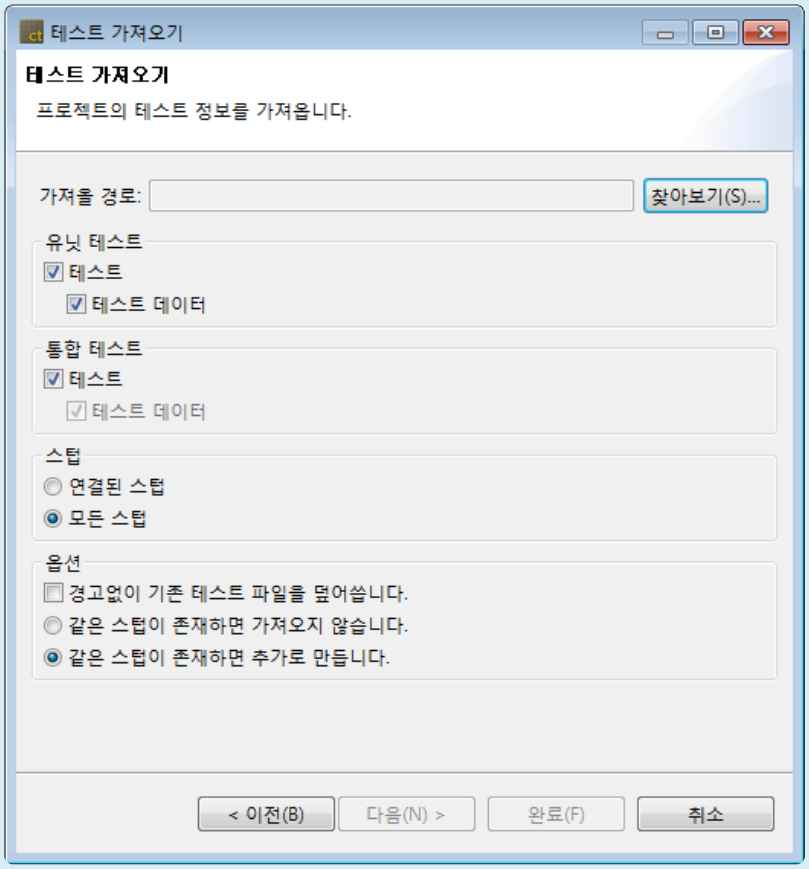
2. 프로젝트를 공유받고자 하는 사용자가 자신이 사용하는 Controller Tester에서 위 프로젝트를 생성한 곳과 동일한 RTV 서버를 연결하고, 동일한 RTV 툴체인을 등록합니다.
3. 프로젝트 생성 마법사에서 [RTV 빌드 C/C++ 프로젝트] 를 선택하여 RTV 프로젝트(이하 RTV_A' 프로젝트)를 생성합니다.
4. Controller Tester 워크스페이스의 RTV_A 프로젝트 폴더로부터 *$(프로젝트 폴더)/.csdata/link.mk* 파일을 가져와 RTV_A' 프로젝트 폴더의 link.mk 파일을 덮어씁니다.
5. 동일한 테스트 데이터를 공유하고자 할 때는 하단의 [참고2]를 확인하십시오.

✻ 소스 파일이 있는 경로가 길면 전체 소스 파일을 제대로 못 가져오는 경우가 있을 수 있습니다. 소스 파일이 있는 경로가 너무 긴 경우, CT의 글로벌 경로를 드라이브 바로 아래에 지정하도록 합니다. (ex. *C:\temp*)
CT 글로벌 경로를 수정하는 방법은 CT 패키지가 설치된 위치에 있는 CodeScroll.ini 파일을 열고 -g 옵션 아래의 default를 새롭게 설정할 글로벌 경로로 바꾸면 됩니다.

```
-startup
plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.500.v20170531-1133
-data
@noDefault
-g
C:\temp
-vmargs
```

✻ 동일한 프로젝트를 공유한 경우에도, 유닛 테스트를 각각 생성하면 커버리지 결과가 다를 수 있습니다. 테스트를 공유할 때에는 [내보내기] > [테스트 내보내기] 기능을 사용하여 테스트를 내보낸 후, [가져오기] > [테스트 가져오기] 기능을 사용하여 내보냈던 테스트를 가져와야 합니다.




테스트 가져오기
테스트 가져오기
프로젝트의 테스트 정보를 가져옵니다.
가져올 경로:
찾아보기(S)...
유닛 테스트
테스트
테스트 데이터
통합 테스트
테스트
테스트 데이터
스텁
연결된 스텁
모든 스텁
옵션
경고없이 기존 테스트 파일을 덮어씁니다.
같은 스텁이 존재하면 가져오지 않습니다.
같은 스텁이 존재하면 추가로 만듭니다.
< 이전(B)
다음(N) >
완료(F)
취소

4.2.2. RTV 서버 사용 가이드

하나의 RTV 서버를 사용하는 경우

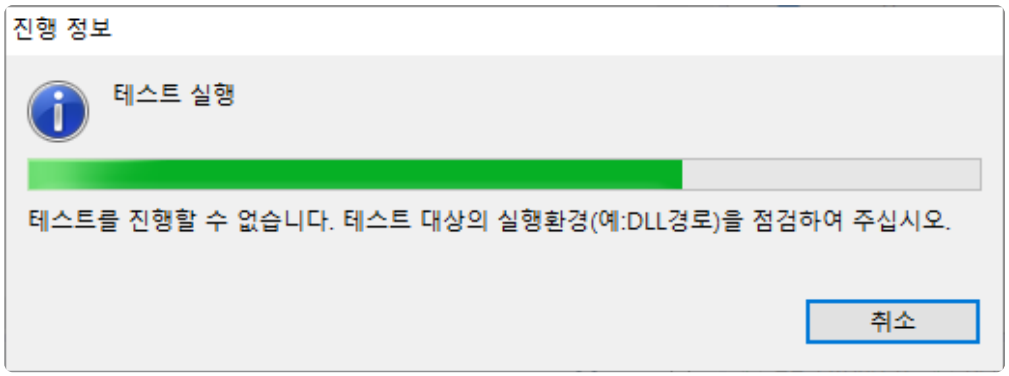
1. RTV 서버에 csbuild capture 기능을 사용하여 빌드한 프로젝트가 있는 경우
    a. 별도의 설정이 필요 없이, 위의 프로젝트 공유 시나리오에 따라 프로젝트를 가져올 수 있습니다.
2. RTV 서버 연결은 되어 있으나, 서버(아이피/포트) 정보가 다른 경우
    a. 프로젝트를 생성할 당시의 서버(아이피/포트) 정보를 가져오기 때문에, 기존 서버 정보를 가져오며 툴체인 정보를 가져오지 않습니다.
    b. 서버 정보를 동일 서버에 접속할 수 있도록 수정한 후, 툴체인 가져오기를 통해 동일한 이름의 툴체인을 가져옵니다. 이 때, 프로젝트에서 사용한 툴체인의 경로는 같아야 합니다.

> ! 둘 이상의 RTV 서버를 사용하는 경우(동일한 소스 파일, 툴체인을 구성하여 사용하려는 경우나 RTV 서버가 설치된 가상머신 파일을 전달받아 사용하려는 경우)에는 RTV 프로젝트 공유가 어려울 수 있습니다.

5. 테스트 오류 발생 시 원인 파악하기

Controller Tester에서 테스트를 수행할 때 오류가 발생하는 경우가 있습니다. 이 때 사용자는 Controller Tester의 디버그 정보 확인 기능으로 테스트 오류 발생 원인을 찾아볼 수 있습니다.

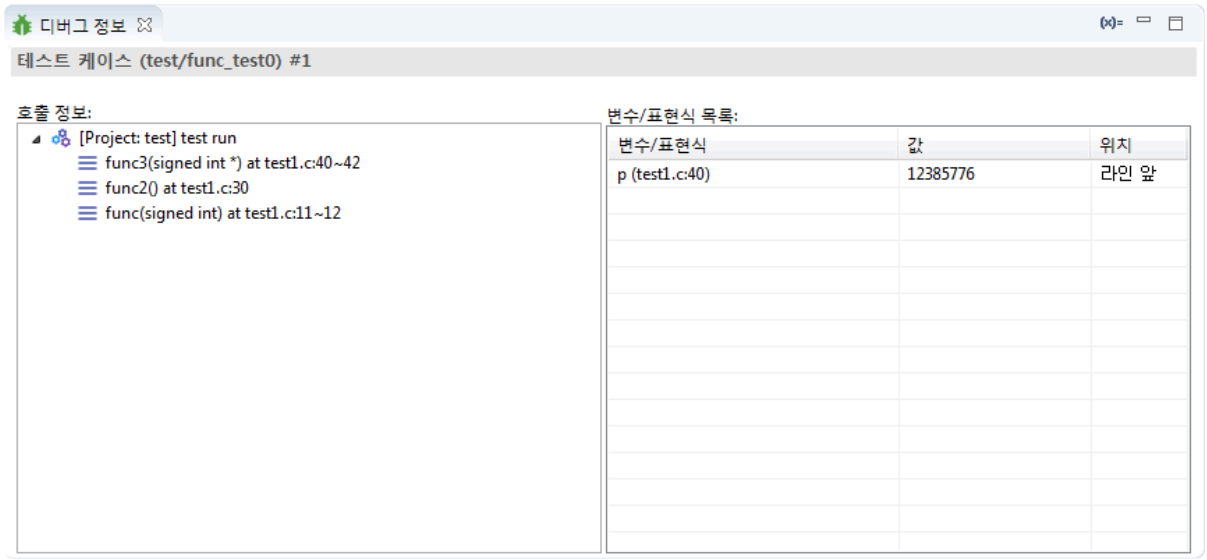
테스트 실행이 실패하는 경우



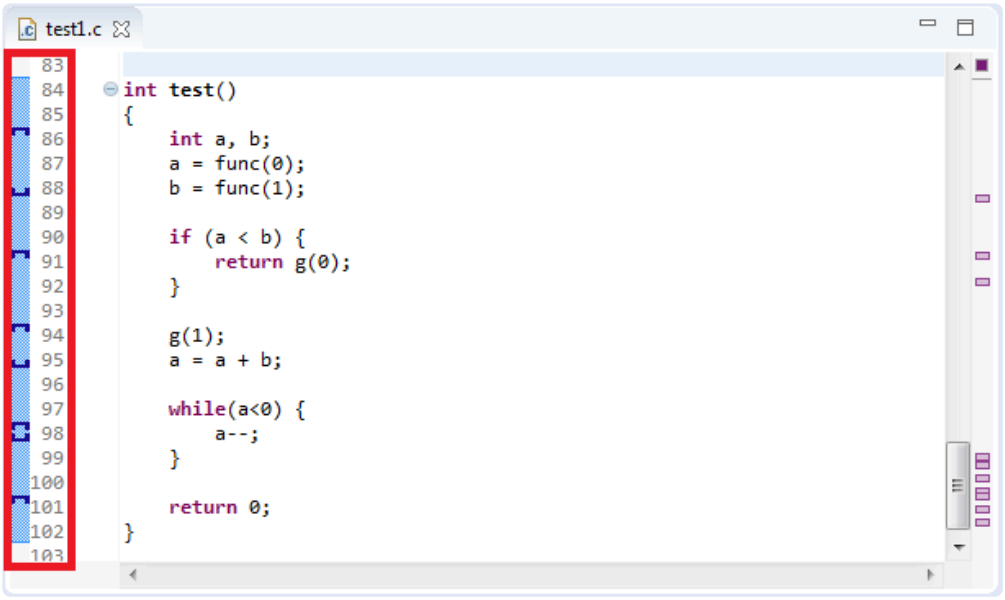
테스트가 실패하는 경우에도 디버그 정보 확인을 수행할 수 있습니다. 생성된 테스트 케이스의 [디버그 정보 확인]을 수행하면 [디버그 정보 뷰]에 호출 정보가 표시되며, 테스트가 실패한 위치를 알 수 있습니다.

테스트 실행 후 결과가 오류를 포함하는 경우

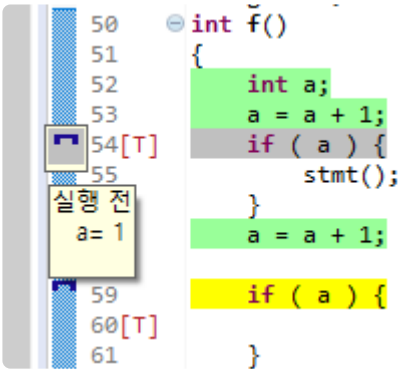
Controller Tester에서 테스트를 수행한 후 Signaled, Abnormal Exit과 같은 오류 결과를 표시하는 경우가 있습니다. 오류가 발생한 테스트 케이스의 [디버그 정보 확인]을 수행하면 [디버그 정보 뷰]에 함수 호출 정보가 표시됩니다. 만일 디버그할 변수/표현식을 추가했다면, 실행된 변수/표현식 목록이 함께 표시됩니다.



호출 정보는 함수 호출 순서를 나타냅니다. 함수가 호출된 위치가 기록되며, 호출 정보 최상단에 마지막 실행 위치가 기록됩니다.
변수/표현식 목록은 테스트 케이스와 함께 실행된 변수/표현식 값을 나타냅니다. 소스 코드 전체에 추가된 변수/표현식 목록은 [디버그 정보 뷰]의 툴바 메뉴의 [변수/표현식 목록]에서 확인할 수 있습니다.



소스 코드 편집기의 마커에서도 디버그할 변수/표현식 정보를 확인할 수 있습니다. 디버그할 변수/표현식을 추가하게 되면 소스 코드 편집기에 추가 위치가 마커로 표현되며, 각 마커 위로 마우스를 올리면 해당 위치에 추가된 변수/표현식 목록을 확인할 수 있습니다.
디버그 정보를 포함하는 테스트 케이스를 선택한 경우, 각 마커에 테스트 케이스가 실행한 변수/표현식 결과가 함께 표시됩니다.



함수 호출 정보와 실행된 변수/표현식 값을 통해 [디버그 정보 확인]을 수행한 테스트 케이스의 오류 발생 원인을 찾아볼 수 있습니다.

> 디버그할 변수/표현식을 추가하는 방법은 CodeScroll Controller Tester 문서의 [디버그할 변수/표현식 추가하기] 항목을 참조하십시오.