

# LemonIDE

---

## 사용설명서

Ver 1.0e  
2010. 07 02



## 개정 이력

작성일	매뉴얼 버전	페이지	설명
2007. 8. 31	1.0	All	jhkim
2007.11. 29	1.0a	All	shlee
2008. 4. 10	1.0b	All	LemonIDE for Windows appended
2008.5.10	1.0c	6,4	Nonstop Debugging Appanded
2009.2.2	1.0d	All	Eddy CPU 2.1 Environment Appanded
2010.7.2	1.0e	2	Open Linux Version

## 목차

<b>1 장. 개요</b>	<b>1-1</b>
1.1 이 매뉴얼에 대해	1-1
1.2 독자	1-1
1.3 LemonIDE 관련 문서	1-3
1.4 기술지원	1-3
<b>2 장. 시작하기</b>	<b>2-1</b>
2.1 개요	2-1
2.2 구조	2-2
2.3 사양 및 기능	2-3
<b>3 장. Windows 환경에 설치하기</b>	<b>3-1</b>
3.1 설치환경	3-1
3.1.1 Windows 시스템	3-1
3.1.2 Target 시스템	3-1
3.2 설치하기	3-1
3.2.1 JDK 설치	3-2
3.2.2 Windows 환경변수 설정	3-3
3.2.3 LemonIDE 설치	3-3
3.3 프로젝트 등록	3-4
3.3.1 Workspace 등록하기	3-5
3.3.2 프로젝트 등록하기	3-5
3.3.3 Path 설정하기	3-7
<b>4 장. Linux 환경에 설치하기</b>	<b>4-1</b>
4.1 설치환경	4-1
4.1.1 리눅스 호스트 시스템	4-1
4.1.2 Target 시스템	4-1
4.2 설치하기	4-2
4.2.1 Toolchain 및 LemonIDE 패키지 설치	4-2
4.2.2 JDK 설치	4-2
4.3 환경 설정	4-4
4.4 LemonIDE 실행	4-4
4.5 Workspace 등록하기	4-4
4.6 프로젝트 등록하기	4-5
4.6.1 LemonIDE C/C++ Project 생성	4-5
<b>5 장. IDE 사용하기</b>	<b>5-1</b>
5.1 IDE 화면 구성	5-1

5.2	소스작성 및 편집 .....	5-2
5.3	컴파일 환경설정 및 빌드 .....	5-3
5.3.1	Makefile 수정 및 작성.....	5-3
5.3.2	Make Target 등록.....	5-4
5.3.3	소스 컴파일 .....	5-5
5.4	실행환경 Run Method 등록하기.....	5-5
5.4.1	Method 이름 및 바이너리 파일 등록 .....	5-6
5.4.2	Arguments 설정 .....	5-7
5.4.3	Target Agent 설정 .....	5-7
5.4.4	Debugger 설정.....	5-7
5.4.5	Common 설정 .....	5-8
5.4.6	Method 등록 결과.....	5-9
5.5	Run Method 실행하기 .....	5-9
5.5.1	Eddy 의 Target Agent 설정하기 .....	5-9
5.5.2	Run Method 실행하기 (시리얼포트로 결과확인).....	5-10
5.5.3	Run Method 실행하기 (Telnet 으로 결과확인) .....	5-11
5.6	펌웨어 이미지 만들기.....	5-12
5.6.1	Makefile 수정 .....	5-12
5.6.2	Make Target 등록.....	5-13
5.6.3	펌웨어 이미지 생성 .....	5-14
5.6.4	펌웨어 업데이트 .....	5-14
6 장.	LemonIDE 디버거.....	6-15
6.1	디버깅 준비 .....	6-15
6.1.1	Target Agent 실행 .....	6-15
6.1.2	컴파일 옵션 설정 시 주의 사항.....	6-15
6.2	디버깅 실행 .....	6-16
6.3	정지점디버깅 .....	6-18
6.3.1	컴파일 환경 수정 및 디버깅 실행 .....	6-18
6.3.2	정지점 추가하기 .....	6-18
6.3.3	프로그램의 실행 제어 .....	6-19
6.4	비정지 디버깅 .....	6-19
6.4.1	컴파일 환경 수정 및 디버깅 실행 .....	6-19
6.4.2	추적점 설치 및 action 지정.....	6-20
6.4.3	프로그램의 추적 제어 .....	6-22
7 장.	모니터링 도구 .....	7-1
7.1	Target 브라우저 .....	7-1
7.1.1	Target 브라우저의 유저 인터페이스 .....	7-1
7.1.2	Target 브라우저에 Target 추가 .....	7-3
7.2	Target 상태 모니터 .....	7-4
7.3	Terminal .....	7-5

---

7.4	CPU Memory View .....	7-7
7.5	Registers .....	7-7
7.6	Workspace 변경 .....	7-7

# 1장. 개요

이 장은 시스템베이스의 통합개발환경 LemonIDE 의 개요에 대해 설명한다.

## 1.1 이 매뉴얼에 대해

이 매뉴얼은 시스템베이스의 통합개발환경 LemonIDE 를 이용하여 손쉽게 프로그램을 작성하여 Target 보드에  
서 동작시키기 위한 절차와 기능, 사용법을 다루고 있다.

## 1.2 독자

이 매뉴얼의 대상 독자는 LemonIDE 를 이용하여 프로그램을 개발하려는 개발자로 LemonIDE 를 사용하거나  
설정하기 전에 이 매뉴얼을 읽을 것을 추천한다. 이 매뉴얼에는 LemonIDE 의 다양한 기능과 개발 절차, 사용  
시 주의사항에 대한 내용이 포함되어 있다. 이 매뉴얼의 내용을 잘 숙지하면, LemonIDE를 통해 연결대상 장비  
에서 직접 작성한 프로그램을 작동시키는 작업에 있어서 도움이 될 것이다.

매뉴얼 구성

[1장 개요](#) 는 일반적인 정보와 소개를 담고 있다.

[2장 시작하기](#) 는 LemonIDE 의 기능과 구조에 대한 소개를 담고 있다.

[3장 Windows 환경에 설치하기](#) 는 Windows 환경에 LemonIDE 를 설치하고 환경을 구축하여 실행이 가능한 상태로 호스트를 설정하는 과정을 다루고 있다.

[4장 Linux 환경에 설치하기](#) 는 Linux 환경에 LemonIDE 를 설치하고 환경을 구축하여 실행이 가능한 상태로 호스트를 설정하는 과정을 다루고 있다.

[5장 IDE 사용하기](#) 는 LemonIDE 의 각 메뉴와 기능에 대해 상세하게 설명한다.

[6장 LemonIDE 디버거](#) 은 LemonIDE 를 통해 원격의 Target에서 동작하는 어플리케이션을 디버깅 하는 방법에 대해 설명한다.

[7장 모니터링 도구](#) 는 Target의 상태를 LemonIDE 에서 바로 실시간 모니터링 할 수 있는 기능에 대해 설명한다.

## 1.3 LemonIDE 관련 문서

LemonIDE와 관련된 기술문서는 다음과 같다.

문서명	설명
사용설명서	LemonIDE 의 설정, 관리에 대한 설명
LemonIDE Spec Sheet	LemonIDE 의 상세 기술문서

LemonIDE 와 에 대한 추가정보를 얻으려면, 시스템베이스 홈페이지(<http://www.sysbas.com>) 혹은 개발자 커뮤니티 (<http://www.embeddedmodule.com>) 를 방문하기 바란다. 이들 웹사이트를 통해 LemonIDE 관련 기술문서 나 최신버전을 다운로드 받을 수 있다.

### 참고

모든 문서는 항상 최신 버전으로 업데이트하여 홈페이지 및 커뮤니티에 게재 되며 문서의 내용은 사전 공지 없이 변경될 수 있다.

## 1.4 기술지원

시스템베이스는 다음의 세 가지 방법으로 고객에 대한 기술지원을 제공한다.

1. 개발자 커뮤니티인 <http://www.embeddedmodule.com> 을 방문하면 전세계의 개발자와 LemonIDE 를 사용한 프로그래밍에 대해 의견을 교환할 수 있다.
2. 자사 홈페이지 <http://www.sysbas.com/> 의 기술지원 페이지를 통해 자주 묻는 질문(FAQ)이나 게시판을 통해 기술지원을 받을 수 있다.
3. 시스템베이스의 기술팀([tech@sysbas.com](mailto:tech@sysbas.com)) 이메일을 통해 질문, 요청, 의견도 보내서 기술 지원을 받을 수 있다.
4. 긴급한 상황일 경우 전화 상담을 통해 기술지원을 받을 수 있다. (T. 02-855-0501)



## 2장. 시작하기

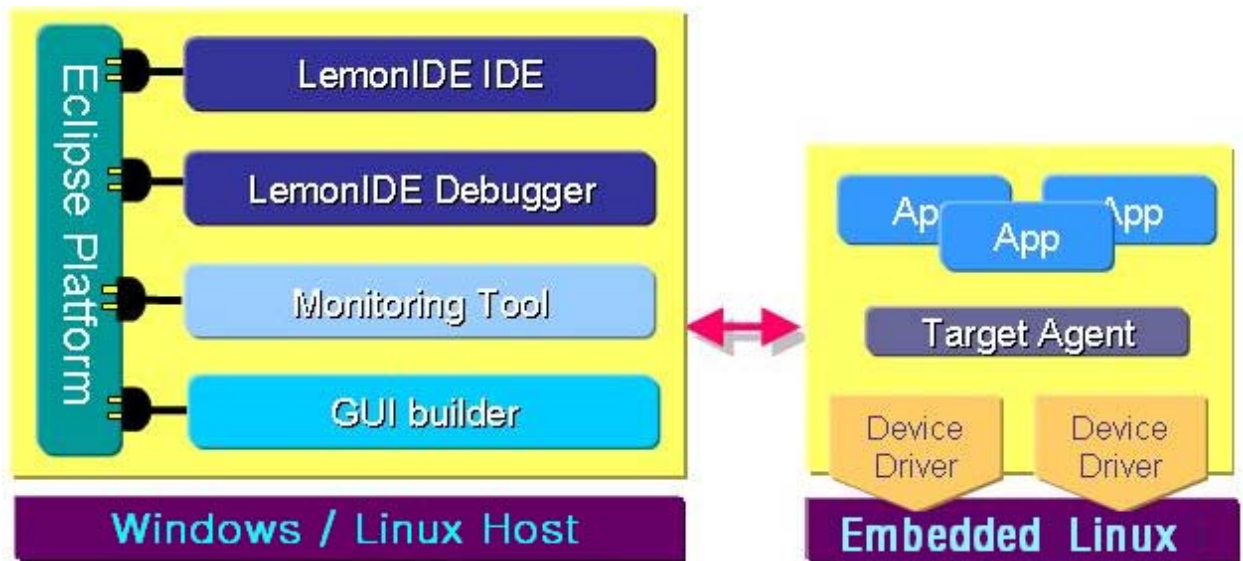
이 장에서는 LemonIDE의 개요와 핵심 기능, 활용 분야에 대해 설명한다.

### 2.1 개요

LemonIDE는 오픈 소스 프로젝트인 Eclipse 기반의 임베디드 소프트웨어 통합 개발 환경이다. Embedded Linux 기반에서 수행되는 어플리케이션, 펌웨어를 쉽게 개발할 수 있는 GUI(Graphical User Interface) 환경을 제공한다. 또한 GNU C/C++ 컴파일러, 소스코드 에디터, 디버거 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리할 수 있는 환경을 제공함으로써 기존의 TUI(Text User Interface) 기반 개발환경의 불편함을 해결해 줄 것이다. 이는 각각의 컴파일러, 디버거 프로그램의 명령어를 익힐 필요가 없다는 것을 뜻한다. LemonIDE는 통일된 GUI를 제공함으로써 컴파일에서부터 디버깅까지의 일련의 작업들을 마우스 클릭만으로 실행 할 수 있는 편의를 제공한다. 또한 Makefile 자동생성, 소스 자동 완성, 원격 디버깅, 플러그인 지원, Target 시스템 모니터링등 프로그램 개발에 필요한 거의 모든 기능을 지원한다.

## 2.2 구조

사용자는 리눅스 호스트 시스템에서 Eddy 플랫폼 기반의 임베디드 시스템(Target 시스템)에서 수행되는 펌웨어, 응용 등의 임베디드 소프트웨어를 보다 쉽고 빠르게 개발할 수 있다. LemonIDE에 포함되는 각 도구들은 다음과 같다.



**LemonIDE IDE** : 프로젝트 기반으로 C/C++ 소스 프로그램 편집, 크로스 컴파일, 원격 실행, GUI 빌더와의 연동 등을 지원하는 도구

**LemonIDE 디버거** : LemonIDE 를 통하여 작성된 프로그램을 디버깅하는 도구로, 정지점 기반의 디버깅과 프로그램을 정지시키지 않고 디버깅하는 추적점 기반의 디버깅을 지원

**모니터링 도구** : Target 시스템의 메모리, 프로세스, 자원, 배터리 소모량 등의 정보를 보여주는 도구

**Target Agent** : Target에서 수행되며 호스트와 연동하여 파일 전송, 원격 실행과 같은 기능을 제공하는 도구

## 2.3 사양 및 기능

LemonIDE의 기본사양 및 기능은 아래와 같다.

### Eclipse

Eclipse SDK 3.2.2

CDT core 3.1.2

CVS core 3.2.2

### Compiler

GNU C/C++ 3.4.3

arm-linux 크로스 컴파일러

### Editor

C/C++ 소스 코드 에디터

C/C++ 코드 자동 완성

구문 하이라이팅

Makefile 자동 생성

소스 브라우저

탭 인터페이스를 통한 다중 파일 편집

파일 검색 및 고급 검색 지원

파일의 종류에 따른 외부 Editor 선택 기능

오류 구문에 대한 명확한 표시

Auto Build 기능

원격 실행 지원

### Debugger

GNU gdb 6.3.90

비주얼 소스 코드 디버거

정지점 및 추적점 기반 디버깅 지원

원격 디버깅 지원

멀티 쓰레드 디버깅 지원

### Target Monitor Tool

Targetview Plug-in 1.0.0

Monitor Plug-in 1.3.4

## 3장. Windows 환경에 설치하기

이 장에서는 LemonIDE 를 Windows 호스트 시스템에 설치하는 방법에 대해 설명한다.

Cygwin 설치, Toolchain 설치, DK Source 설치에 대해서는 “Eddy\_DK\_Programmer\_Guide” 매뉴얼을 참조하기 바란다.

### 3.1 설치환경

#### 3.1.1 Windows 시스템

LemonIDE 는 다음과 같은 Windows 버전에서 성공적으로 동작 테스트를 마쳤다.

Windows XP SP2

Windows 2000

Windows 2003

---

**(참고)** 이 매뉴얼에서는 Windows XP 에 설치된 LemonIDE 를 기준으로 설명한다.

---

#### 3.1.2 Target 시스템

LemonIDE 에서 작성한 어플리케이션을 다운로드 해서 실행하기 위해서는 Target 시스템이 필요하다. Target 시스템으로 사용 가능한 시스템베이스의 제품군은 아래와 같다.

Eddy Series (Ver 2.xx)

Eddy-DK (Ver 2.xx)

Target과 호스트를 연결하고 동작을 확인하는 작업에 대해서는 Eddy-DK Programmer Guide 를 참고하시기 바란다.

### 3.2 설치하기

이 장은 Eddy 통합개발환경인 LemonIDE 의 설치방법을 설명한다.

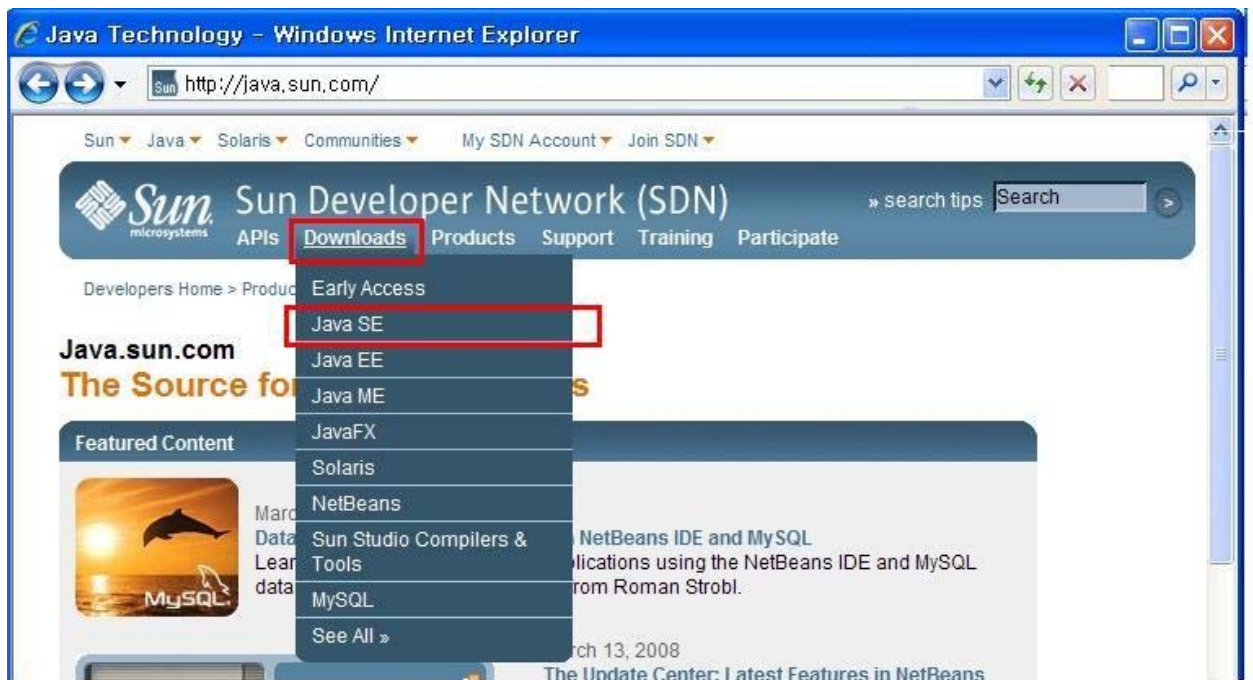
LemonIDE for Windows 설치에 Cygwin 과 Toolchain 이 이미 설치된 상태이어야 한다.

Cygwin 설치, Toolchain 설치, DK Source 설치에 대해서는 “Eddy\_DK\_Programmer\_Guide” 매뉴얼을 참조하기 바란다.

### 3.2.1 JDK 설치

LemonIDE 를 실행하기 위해서는 Java Runtime Environment 를 설치해야 하며, Java5 JRE (Java Runtime Environment) 이상 버전이 설치되어 있어야 한다. JDK는 License 정책상 배포가 금지되어 있어서 사용자는 이를 인터넷을 통해서 다운을 받아서 설치하여야 한다. 다음은 JDK 6.0을 다운 받아 설치하는 과정이다.

<http://java.sun.com/> 에 접속을 한다.



위의 그림에서 처럼 Download 탭을 선택한 후 다시 Java SE 를 선택하면 다운로드 사이트로 접근을 한다.



위 그림의 현재 최신버전인 JDK 6 버전을 다운로드하도록 좌측 다운로드 탭을 클릭한다,

#### Windows Platform - Java SE Development Kit 6 Update 5

↓ Windows Offline Installation, Multi-language	jdk-6u5-windows-i586-p.exe	71.39 MB
↓ Windows Online Installation, Multi-language	jdk-6u5-windows-i586-p-iftw.exe	373.39 KB

#### Linux Platform - Java SE Development Kit 6 Update 5

↓ Linux RPM in self-extracting file	jdk-6u5-linux-i586-rpm.bin	63.49 MB
↓ Linux self-extracting file	jdk-6u5-linux-i586.bin	67.23 MB

#### (참고)

JDK 가 설치되는 디렉토리는 가급적 “C:\” 에 설치되도록 한다.

다운로드 후 자동으로 설치를 위한 위자드가 실행되며, 네트워크 상태에 따라 설치시간이 다소 소요 될 수 있다.

### 3.2.2 Windows 환경변수 설정

LemonIDE 가 실행에 필요한 java 라이브러리를 참조가능하도록 Path 에 등록한다.

Windows 바탕화면 → 내컴퓨터 → 오른쪽 마우스로 클릭 → 속성 → 고급 탭 → 환경 변수를 선택하여 시스템 변수 중 Path 를 선택하여 편집을 클릭한 후 맨 뒤에 다음을 추가한다.

```
;c:\jdk1.<Version>\bin
```

Version 은 설치된 JDK 의 버전으로 JDK 가 설치된 디렉토리를 말한다.

JDK 6 버전으로 C:\ 루트 밑에 설치하였다면 Version 은 “6.0\_05” 로서 설치된 디렉토리는 c:\jdk1.6.0\_05 이므로 Path 등록은 “;c:\jdk1.0\_05\bin” 으로 등록한다.

### 3.2.3 LemonIDE 설치

LemonIDE 패키지를 설치한다. LemonIDE 패키지는 Eddy DK CD 의 SDK/Windows 폴더 밑에 “Lemonide-windows-1xx.tgz” 파일이다. 이 파일을 하드디스크 “C:” 드라이브의 루트 디렉토리에 복사한 다음 Windows 의 명령프롬프트 프로그램인 “CMD” 를 실행하여 아래 그림과 같은 방식으로 압축을 푼다.

#### (대소문자 주의)

LemonIDE 패키지가 설치되는 기본 디렉토리는 “c:\cygwin\opt\lemonix\lemonide” 이다.

(참고) 이 매뉴얼에서는 LemonIDE 가 설치되는 하드디스크 공간을 C:\ 로 가정하여 설명한다. C:\ 이 외의 드라이브에 설치하려면 설치 후 환경설정을 변경해야하는 번거로운 과정이 있으므로 가급적 C:\ 에 설치하도록 권장한다.

```

C:\WINDOWS\system32\cmd.exe
PATH = c:\cygwin\bin
cd W
tar zxvf lemonide-windows*.tgz -C c:\cygwin\
opt/lemonix/bin/arm-win32-estogdb.exe
opt/lemonix/bin/lemonide
opt/lemonix/lemonide
opt/lemonix/lemonide/lemonide.exe
opt/lemonix/lemonide/plugins
opt/lemonix/lemonide/startup.jar
opt/lemonix/lemonide/configuration

```

### 3.3 프로젝트 등록

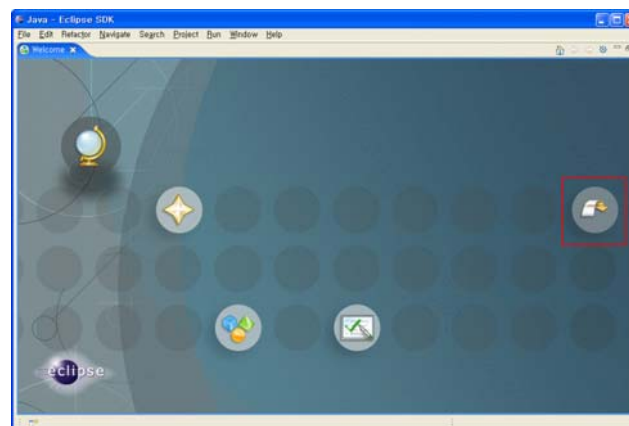
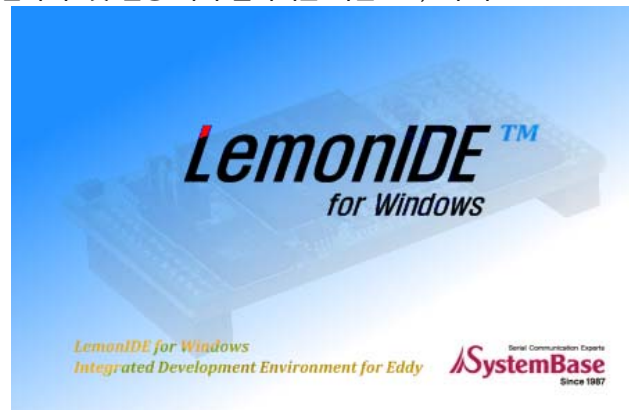
모든 설치와 완료되었으면, 설치 디렉토리인 /cygwin/opt/lemonix/lemonide 디렉토리인 이동하여 lemonide.exe 를 실행한다.

LemonIDE 실행 프로그램은 “c:\cygwin\opt\lemonix\lemonide\lemonide.exe” 에 위치한다.

사용자의 편의를 위해 Windows 바탕화면에 바로가기로 만들어 사용할 수 있다.

Lemonide 로고와 함께 잠시의 로딩이 지난 후 IDE 초기화면이 나타나면 성공이다.

아래 화면은 LemonIDE 설치 후 첫 실행 시에 출력되는 화면으로, 우측 “WorkBench” 아이콘을 클릭한다.



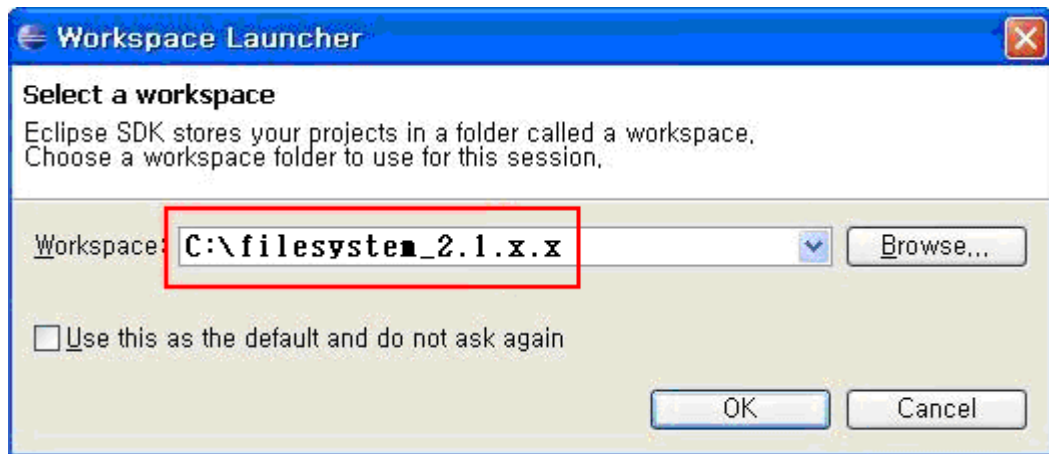


### 3.3.1 Workspace 등록하기

작업하고자 하는 Workspace 디렉토리를 생성한다.

이 매뉴얼에서는 작업영역인 DK Source 를 설치한 경우이므로 DK Source 가 설치된 디렉토리를 선택한다.

Workspace 를 변경하려면 본 매뉴얼의 “7.6 Workspace 변경” 를 참조한다.



(참고) Workspace 는 “C:\documents and settings” 와 같이 공백을 포함해서는 안된다.

### 3.3.2 프로젝트 등록하기

등록한 Workspace 에 작업할 프로젝트를 등록한다.

프로젝트는 LemonIDE 를 통해 소스를 코딩, 빌드, 실행 및 디버깅할 수 있는 공간이다.

이 매뉴얼에서는 Eddy DK Source 에 대한 작업을 설명하므로 Workspace 영역인 DK Source 디렉토리 내의 어플리케이션 작업영역 프로젝트와 펌웨어이미지 작업영역 프로젝트를 등록한다.

LemonIDE 프로젝트는 makefile 생성 여부와 C 또는 C++ 프로그램 여부에 따라 다음과 같은 4 가지 유형으로 구분된다.

Standard Make C++ Project	미리 작성된 Makefile 을 사용하는 C++ 또는 C 프로그램에 대한 프로젝트
Standard Make C Project	
Managed Make C++ Project	Makefile 이 자동으로 생성되는 C++ 또는 C 프로그램에 대한 프로젝트
Managed Make C Project	

(참고) 일반적인 개발환경에서는 특정 라이브러리를 링크하는 경우 등이 많고 대다수의 Open Source 도 Makefile 을 제공하므로 Standard 방식을 추천한다.

제공하는 DK Source 에서도 Makefile 이 정의되어 있으므로 용도에 맞게 수정을 할 수 있는 Standard 방식을 기준으로 설명한다.

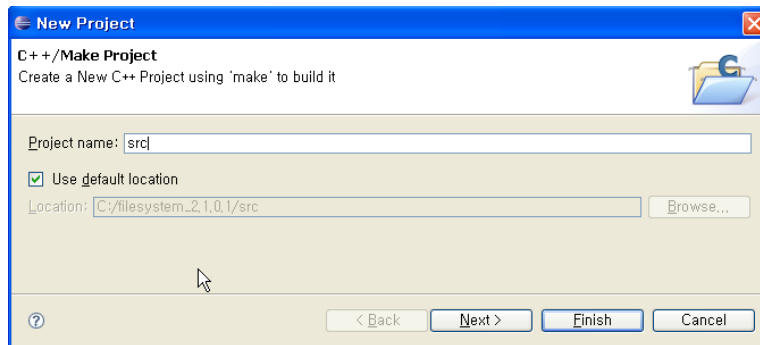
“Standard Make C++ Project” 또는 “Standard C Project” 프로젝트를 생성하려면 LemonIDE 상위메뉴 중



“File” → “New” → “Project” → “LemonIDE Application” → “Standard Make C++ Project” 또는 Standard C Project” 를 선택하거나, “File” → “New” → “Standard C++Project” 또는 “Standard C Project” 를 선택한다.

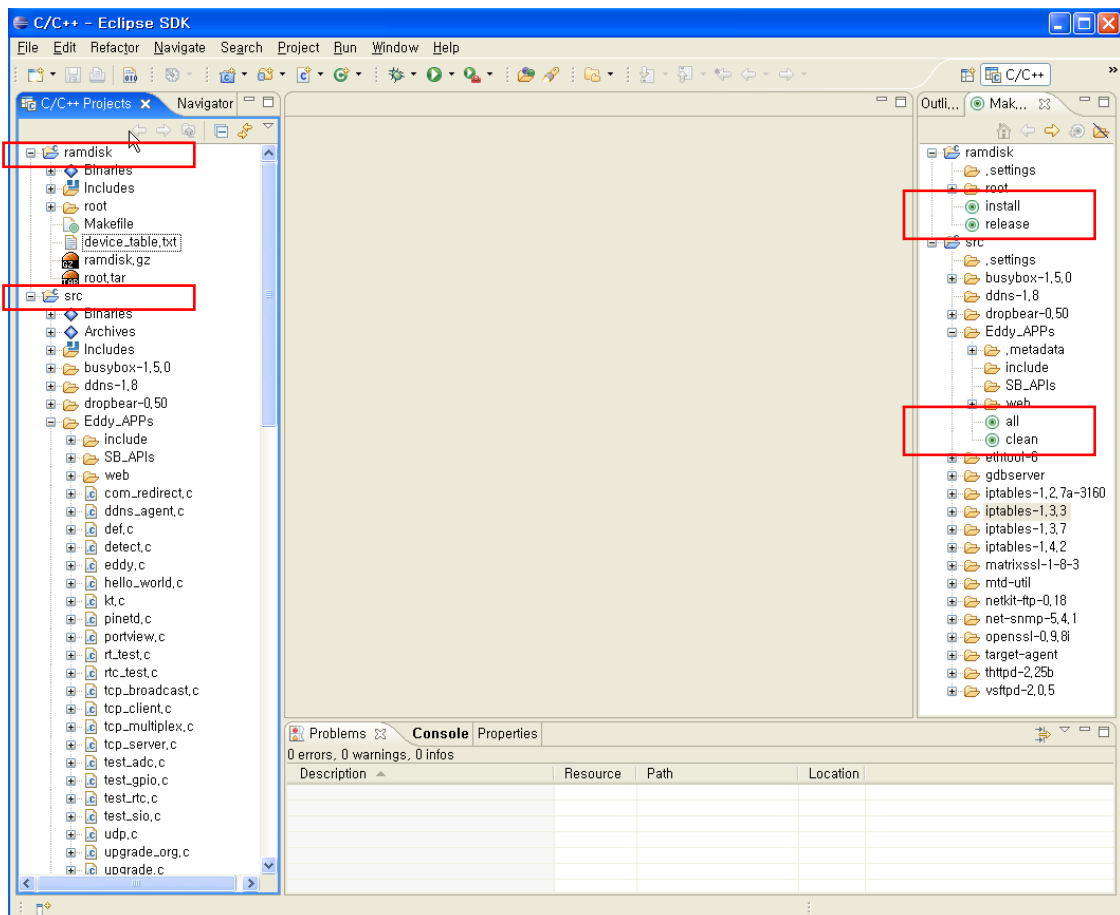
Project Name 에 “SRC” 를 등록하고 “Finish” 버튼을 클릭하면 프로젝트가 등록한다.

“Src” 는 Workspace 로 등록된 Filesystem\_2.1.x.x 의 어플리케이션 디렉토리이다.



같은 방식으로 프로젝트를 하나 더 생성하여 Project name 에 “ramdisk” 를 등록하고 “finish” 버튼을 클릭하여 File System 이미지 작업용 프로젝트를 등록한다.

정상적으로 등록하였다면, Workspace 로 등록한 “filesystem\_2.1.x.x” 에 두개의 프로젝트 “src” , “ramdisk” 가 등록된 화면은 아래와 같을 것이다.



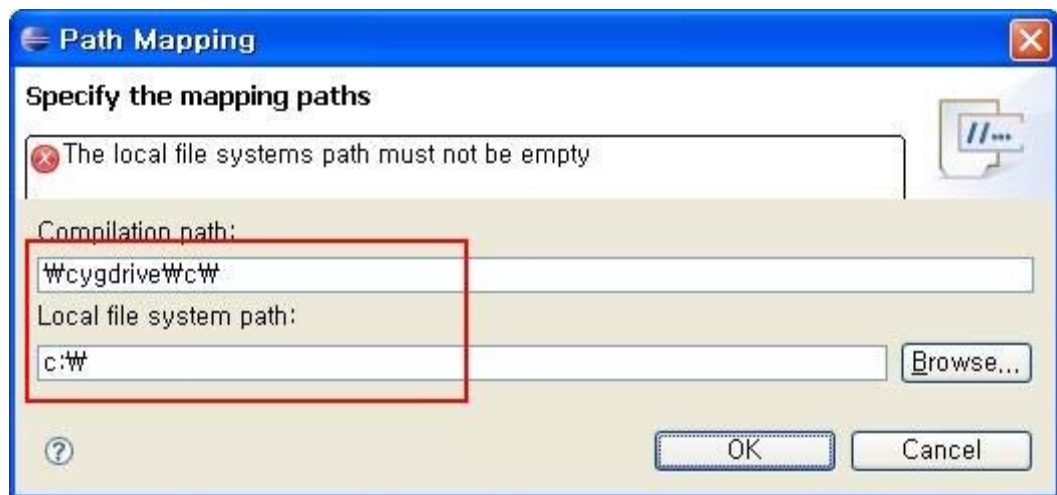
### 3.3.3 Path 설정하기

LemonIDE 는 Cygwin 에서 제공하는 라이브러리를 일부사용한다,  
이러한 이유로 Cygwin 에서 사용하는 Path 의 “/” 는 c:\cygwin\ 이고, Windows 에서 사용하는 기본 Path  
는 C:\ 이므로 이를 호환하도록 설정해 주어야 한다.

LemonIDE 메뉴 창의 Windows → Preferences... → C/C++ → Debug → Common Source Lookup Path → Add  
→ Path Mapping 를 클릭하면, Path Mapping : New Mapping 가 생성된다.

생성된 Path Mapping : New Mapping → Edit 버튼 → Add 를 클릭하면 Mapping 할 Path 를 등록하는 화면이  
다음과 같이 출력되며, 아래 그림과 같이 등록한다.

\cygdrive\c\ → c:\



## 4장. Linux 환경에 설치하기

이 장에서는 LemonIDE 를 리눅스 호스트 시스템에 설치하고 크로스 컴파일 환경을 구축하여 LemonIDE 와 연동하는 방법에 대해 설명한다.

Toolchain 설치와 DK Source 의 설치에 대해서는 “Eddy\_DK\_Programmer\_Guide” 매뉴얼을 참조하기 바란다.

### 4.1 설치환경

#### 4.1.1 리눅스 호스트 시스템

LemonIDE 가 실행되는 환경은 다음의 리눅스 배포버전에서 성공적으로 동작 테스트를 마쳤다.

Red Hat 9.0

Fedora Core 4, 5, 6

SUSE Linux Enterprise Server 10.2

Ubuntu Linux 6.x, 7.x

Debian Linux 4.0

CentOS 4.5

Asianux 3.0

---

**(참고)** 이 매뉴얼에서는 Linux Fedora core 5 에 설치된 LemonIDE 를 기준으로 설명한다.  
다른 리눅스 배포버전에서도 거의 동일한 방식으로 동작한다.

---

#### 4.1.2 Target 시스템

LemonIDE 에서 작성한 어플리케이션을 다운로드 해서 실행하기 위해서는 Target 시스템이 필요하다. Target 시스템으로 사용 가능한 시스템베이스의 제품군은 아래와 같다.

Eddy Series (Ver 2.xx, Ver 2.1.x.x)

Eddy-DK (Ver 2.xx, Ver 2.1.x.x)

Target과 호스트를 연결하고 동작을 확인하는 작업에 대해서는 Eddy-DK Programmer Guide 를 참고하시기 바란다.

## 4.2 설치하기

Linux 호스트에 Eclipse 기반의 통합개발환경을 LemonIDE 설치한다.

### 4.2.1 Toolchain 및 LemonIDE 패키지 설치

LemonIDE 는 tar.gz 의 확장자 형태로 제공된다. 설치파일을 '/' 디렉터리에서 풀면 /opt/lemonix 디렉터리에 LemonIDE 패키지가 설치된다. “Eddy\_DK\_Programmer\_Guide” 매뉴얼에서 이미 설치 하였다면 다시 설치하지 않아도 된다. (대소문자 주의)

(참고) 설치하기의 모든과정은 super user 권한으로 한다.

아래의 예제에서 CDRUM 을 /mnt/cdrom 에 마운트 했다고 가정한다.

```
# cd /
# tar -zxvf /mnt/cdrom/SDK/linux/lemonide*.tar.gz -C /
```

### 4.2.2 JDK 설치

LemonIDE 를 실행하기 위해서는 리눅스 호스트 시스템에 Java Runtime Environment 를 설치해야 한다.

Java5 JRE (Java Runtime Environment) 이상 버전이 설치되어 있어야 한다. Java6 JRE 에서도 동작을 하지만, 세부기능 별로 제대로 동작하지 않는 경우가 있을 수 있기 때문에 Java5 JRE 버전을 권장한다.

JDK는 License 정책상 배포가 금지되어 있어서 사용자는 이를 인터넷을 통해서 다운을 받아서 설치하여야 한다. 다음은 JDK 5.0을 다운 받아 설치하는 과정이다.

<http://java.sun.com/> 에 접속을 한다.

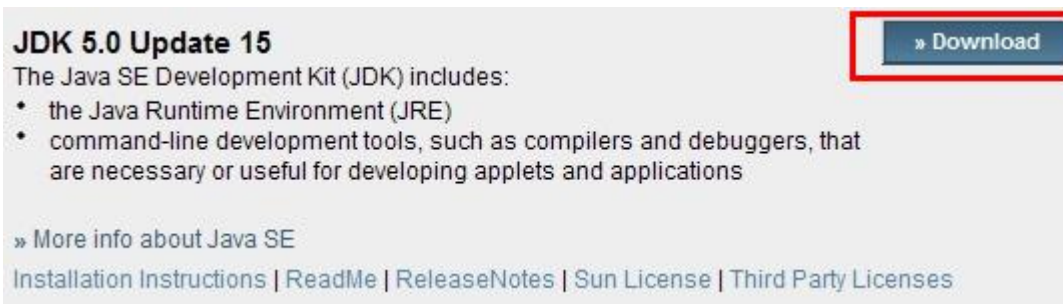


위의 그림에서 화살표가 된 부분을 클릭하면 다운로드 사이트로 접근을 한다. 여기서 최신의 JDK에서부터 이전 버전 JDK를 다운 받을 수 있다.



LemonIDE에서 사용하는 JDK 는 J2SE 5.0 이다. 그러나 최신 버전은 JDK 6 버전이므로 이를 위해서는 이전 버전을 찾는다. 다음은 아래의 순서대로 다운을 받는다.

**(참고)** 현재 매뉴얼에서 설명한 JDK 다운 방법은 JDK공급 업체가 임의로 수정할 수 있으므로 다운 순서나 그림은 차후 수정이 될 수 있으며 실제와 다를 수 있다.



Linux Platform - Java Development Kit 5.0 Update 15		
↓ Linux RPM in self-extracting file	jdk-1_5_0_15-linux-i586-rpm.bin	45.60 MB
↓ Linux self-extracting file	jdk-1_5_0_15-linux-i586.bin	47.35 MB

다운로드가 완료되면 아래와 같이 Linux 호스트에 복사하여 설치한다.

```
# chmod +x jdk-1_5_0_<version>-linux-i586.bin
# ./jdk-1_5_0_<version>-linux-i586.bin
# sudo cp -rf jdk1.5.0_<version> /opt/lemonix/jdk
```

위의 과정 대로 실행을 하면 JDK의 설치가 끝난다.

### 4.3 환경 설정

아래와 같은 환경 변수 설정을 홈 디렉터리의 .bashrc 에 추가하고,

```
export PATH=/opt/lemonix/bin:$PATH
```

source 명령을 실행시켜 변경사항을 반영시킨다.

```
source .bashrc
```

### 4.4 LemonIDE 실행

모든 설치와 환경설정이 완료되었으면, 설치 디렉터리인 /opt/lemonide/bin 디렉터리로 이동하여 lemonide 를 실행시킨다.

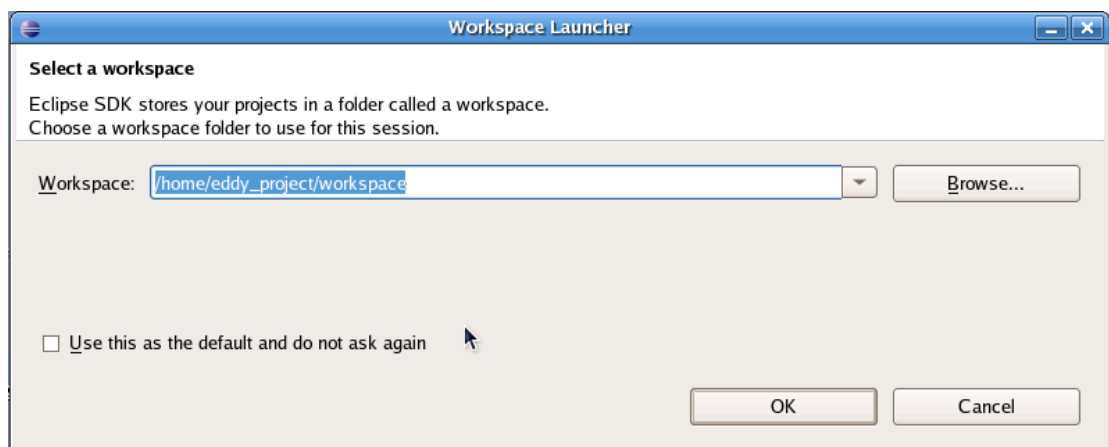
```
cd /opt/lemonix/bin
./lemonide
```

Lemonide 로고와 함께 잠시의 로딩이 지난 후 IDE 초기화면이 나타나면 성공이다.

### 4.5 Workspace 등록하기

사용하고자 하는 Workspace 디렉토리를 선택하고, <OK> 버튼을 누르면 다음과 같은 LemonIDE 초기 화면이 나타난다. (위의 예시는 실제 사용자의 디렉터리 구조와 다를 수 있다.)

수행 결과 다음과 같이 Workspace를 선택하는 창이 나타난다.



## 4.6 프로젝트 등록하기

등록한 Workspace 에 작업할 프로젝트를 등록한다

프로젝트는 프로그램 개발에 필요한 자원 정보의 집합체로 소스 코드 정보, 빌드 정보등으로 구성된다. LemonIDE는 Target 시스템에서 수행되는 프로그램 모듈을 프로젝트 단위로 관리하며, 프로젝트의 빌드 환경은 make 명령 기반으로 수행된다.

Make 명령 수행을 위해서 makefile이 필요하며, LemonIDE의 프로젝트는 makefile을 자동으로 생성하는 경우와 수동으로 생성하는 경우로 구분된다. Makefile을 자동으로 생성하는 경우에는 사용자가 쉽게 UI를 통하여 빌드 정보를 수정할 수 있으며, makefile을 수동으로 생성하는 경우에는 빌드 정보 수정을 위하여 사용자가 직접 makefile 수정을 해야 하는 불편함이 있다. 그러나 기존 makefile 기반의 소스 프로그램을 LemonIDE로 가져와서 개발해야 하는 경우에는 makefile을 수동으로 생성하는 경우가 유리하다. 이미 작성된 makefile의 빌드 정보를 UI를 통하여 직접 입력하는 경우가 더 어려울 수 있기 때문이다.

LemonIDE의 프로젝트는 makefile 생성 여부와 C 또는 C++ 프로그램 여부에 따라 다음과 같은 4가지 유형으로 구분된다.

Standard Make C++ Project Standard Make C Project	미리 작성된 Makefile 을 사용하는 C++ 또는 C 프로그램에 대한 프로젝트
LemonIDE Make C++ Project LemonIDE Make C Project	Makefile 이 자동으로 생성되는 C++ 또는 C 프로그램에 대한 프로젝트

**(참고)** 일반적인 개발환경에서는 특정 라이브러리를 링크하는 경우 등이 많고 대다수의 Open Source 도 Makefile 을 제공하므로 Standard 방식을 추천한다.

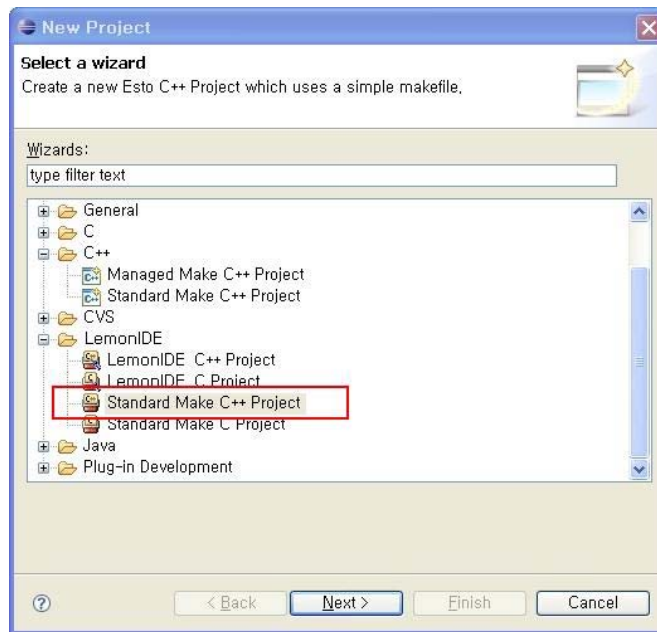
제공하는 DK Source 에서도 Makefile 이 정의되어 있으므로 용도에 맞게 수정을 할 수 있는 Standard 방식을 기준으로 설명한다.

### 4.6.1 LemonIDE C/C++ Project 생성

LemonIDE 프로젝트 생성은 새롭게 C 프로그램을 생성하는 경우에, 자동으로 Makefile 을 생성하므로 유리하지만, 이 매뉴얼에서는 이미 제공하는 DK Source 에 MakeFile 이 포함되어 있으므로 권장하지 않으며, Standard 방식을 권장한다.

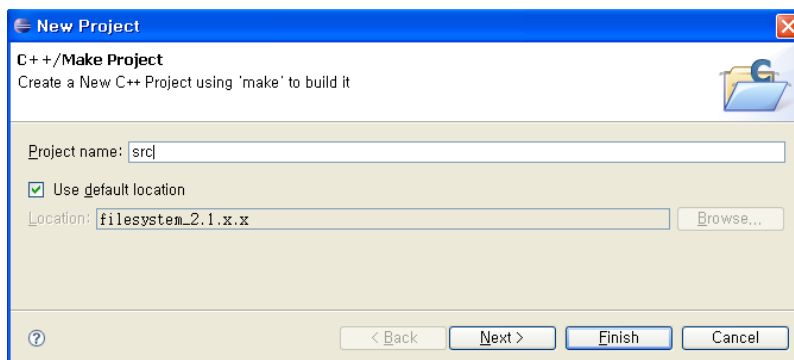
실행된 LemonIDE 화면에서 상위메뉴 중 "File" → "New" → "Project" 를 선택한다.

LemonIDE Application → Standard Make C++ Project 를 선택한다.



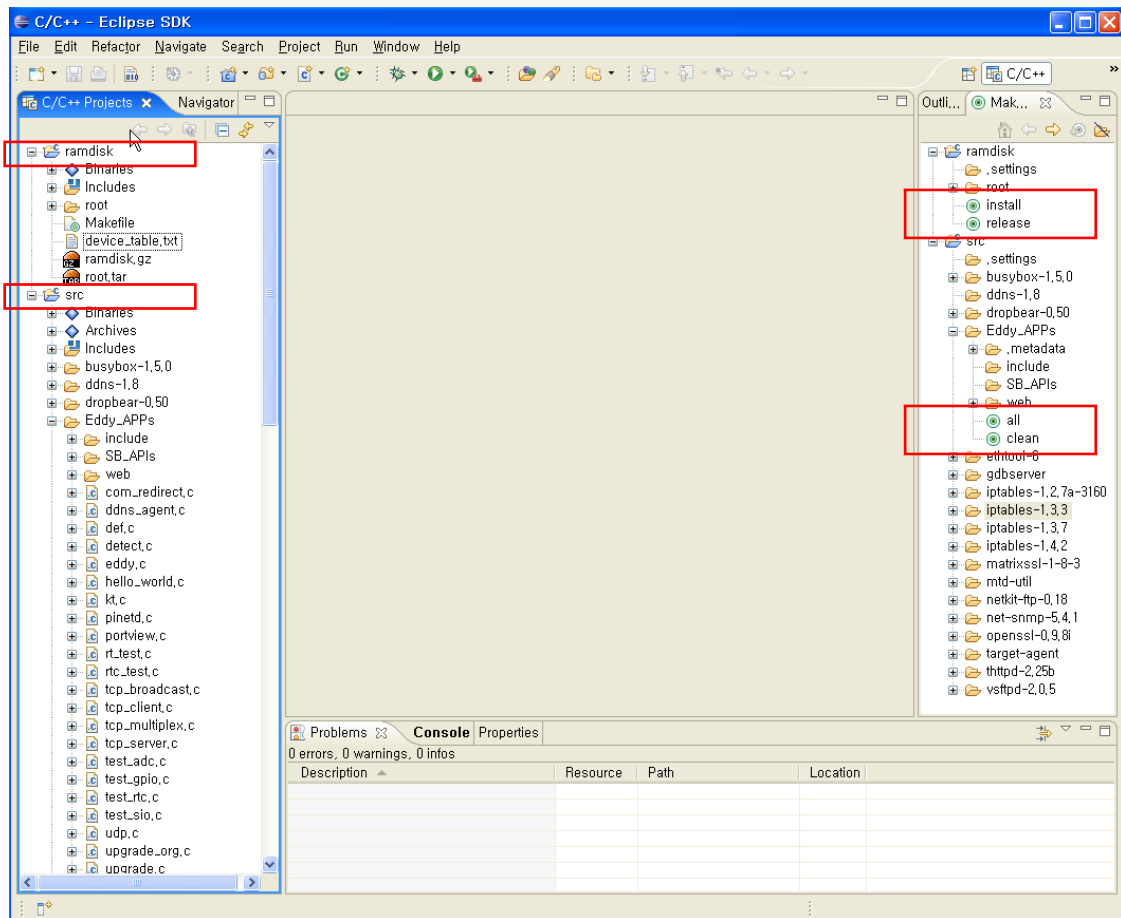
Project Name 에 “SRC” 를 등록하고 “Finish” 버튼을 클릭하여 Eddy Application 프로젝트를 등록한다  
SRC 를 등록한다는 것은 DK Source 가 설치된 디렉토리가 Workspace 로 등록되었으므로 그 하단의 Src 디렉토리를 프로젝트로 등록한다는 뜻이다.

같은 방식으로 Project name 에 “ramdisk” 를 등록하고 “finish” 버튼을 클릭하여 File System 이미지 작업용 프로젝트를 등록한다.



정상적으로 등록하였다면, Workspace 로 등록한 “Filesystem\_2.1.x.x” 에 두개의 프로젝트 “src” , “ramdisk” 가 등록된 화면은 아래와 같을 것이다.





## 5장. IDE 사용하기

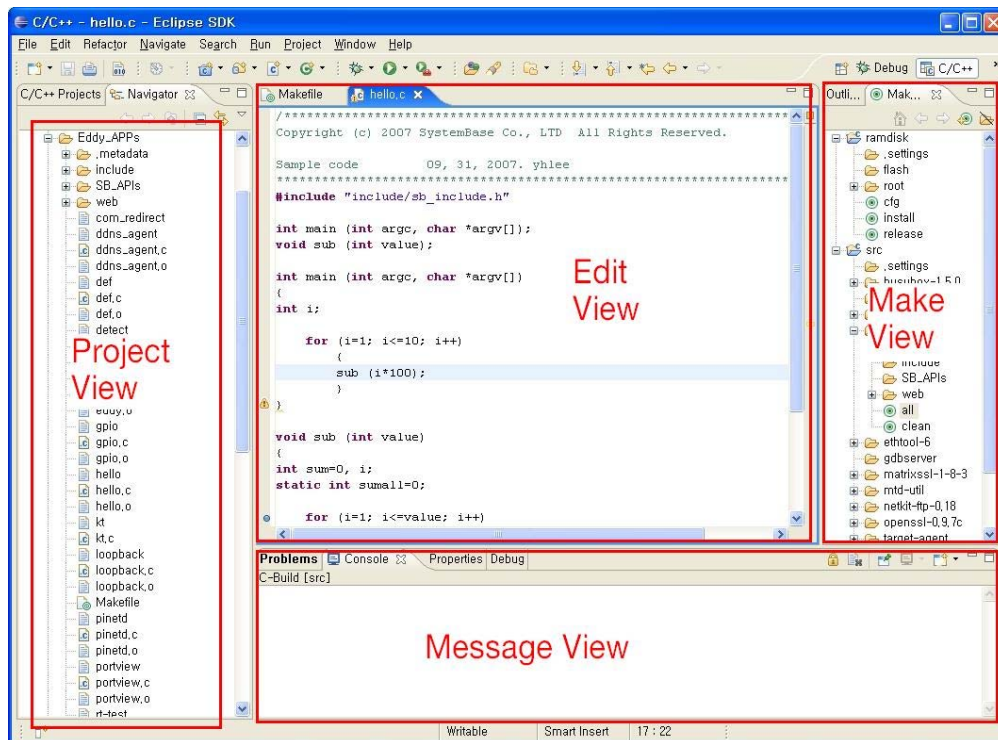
이 장에서는 LemonIDE 의 IDE 기능에 대해 설명한다.

이 장에서는 hello.c 라는 소스화일을 만들고 빌드하여 Eddy 를 통해 실행해보고 최종적으로 Eddy 에 적용할 펌웨어 이미지를 만드는 과정을 순서대로 설명한다.

### 5.1 IDE 화면 구성

LemonIDE IDE(Integrated Development Environment)는 Eclipse 기반의 통합 개발 환경으로 프로젝트 기반의 C/C++ 소스 편집, 크로스 컴파일, 원격 실행, GUI 빌더와의 연동 기능을 지원한다. LemonIDE는 Eclipse의 플러그인 형태로서, LemonIDE의 UI 대부분은 Eclipse UI(workbench)를 이용한다. 따라서, 본 장에서는 Eclipse UI에 대한 사용 설명 보다는 LemonIDE 에 국한되는 내용 중심으로 설명하고자 한다.

LemonIDE 의 화면 구성을 뷰 별로 나누어 보면 아래와 같다.



Project View

Workspace 에 포함된 project 리스트를 보여준다.

Make View

편집중인 소스의 헤더 및 함수, 구조체의 클래스 및 소스의 빌드 지원

Edit View

작업할 소스를 편집할 수 있는 창이며, 다중탭 형식으로 여러 개의 소스를 편집가능하다.

Message View

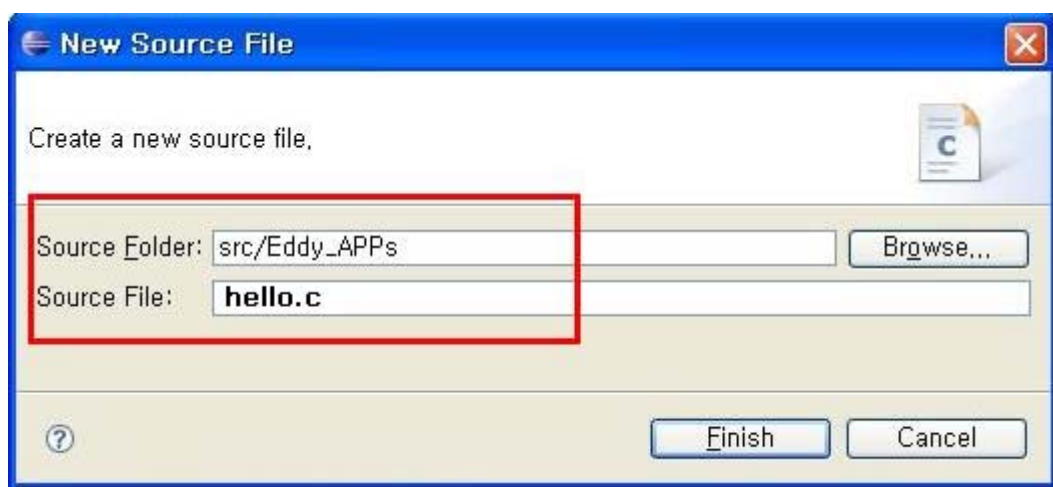
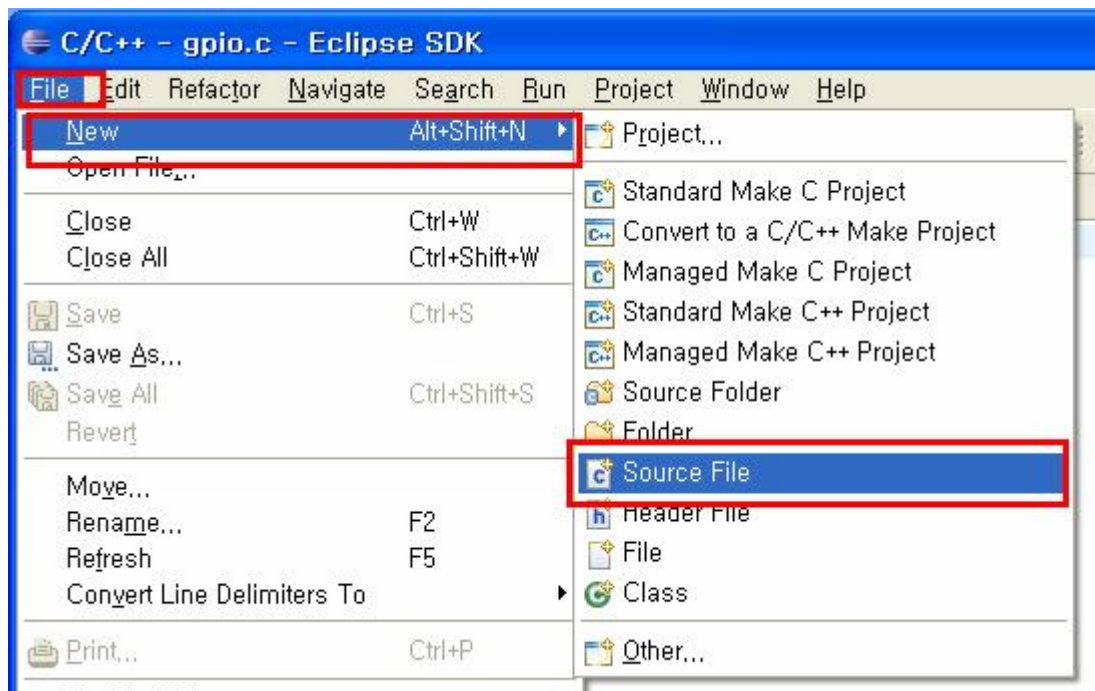
프로젝트 빌드 과정의 메시지 및 기타 LemonIDE 에서 지원하는 Target Browser

## 5.2 소스작성 및 편집

작업할 소스화일을 생성하거나 기존 소스화일을 열어 편집한다.

생성된 프로젝트에 대한 소스 추가는 새 파일을 만드는 경우와 기존 소스를 프로젝트로 가져오는 경우가 있다. 기존 소스를 편집하는 경우에는 Project View 에서 편집하고자하는 파일을 더블클릭하면 Edit View 에 오픈된다. 새로운 소스화일을 생성하려면 “File” → “New” → “Source File” 을 선택한 후 생성할 파일이 저장될 위치와 파일이름을 지정하면 Project View 창에 새로 생성한 파일이 보여지고 Edit View 창에 생성한 소스화일을 편집할 수 있도록 오픈된다.

아래의 그림은 “SRC” 프로젝트 밑에 “Eddy\_APPS” 디렉토리에 “hello.c” 파일을 생성하는 예이다.



아래그림은 LemonIDE 사용방법의 이해를 돕기 위해 작성한 Hello.c 샘플 프로그램이다.

```
#include <stdio.h>

void sub (int value);

int main (int argc, char *argv[])
{
    sub (1000);
    return 0;
}

void sub (int value)
{
    int sum=0, i;

    for (i=1; i<=value; i++) {
        sum += i;
    }

    printf ("SUM (1 ~ %4d) = %7d\n", value, sum);
    return;
}
```

## 5.3 컴파일 환경설정 및 빌드

컴파일은 소스 파일을 컴파일, 링킹하여 Eddy 에서 실행가능한 바이너리 이미지로 만드는 단계를 말한다. LemonIDE 프로젝트에 대한 컴파일은 기본적으로 makefile 기반으로 수행된다. 이 장에서는 DK Source 에서 제공하는 Makefile 을 응용하여 소스화일을 컴파일하는 과정을 소개한다.

### 5.3.1 Makefile 수정 및 작성

새로 소스화일을 만들었거나, Project View 에 있는 기존 소스화일을 수정하였다면 소스화일을 컴파일할 Makefile 을 작성해야한다. DK Source 에서 제공하는 Makefile 을 참조하여 Makefile 을 작성하거나, 새로운 소스화일을 컴파일 하도록 수정해야 한다.

아래의 그림은 “SRC” 프로젝트 밑에 “Eddy\_APPS” 디렉토리에 포함된 Makefile 에 새로 작성한 hello.c 파일의 컴파일환경을 추가하는 예이다.

```

CROSS    = /opt/lemonix/cdt/bin/arm-linux-
LDFLAGS += -L/opt/lemonix/cdt/lib -L/opt/lemonix/cdt/bin
IFLAGS   += -I/opt/lemonix/cdt/include -I./include
.
.
.
TARGET   = eddy      pinetd    def                ddns_agent      \
            Upgrade  portview  upgradetftp      detect          \
            udp       rt_test   test_rtc         test_sio         \
            test_adc  test_gpio hello
LIBS      = -lrt SB_APIs/SB_APIs.a

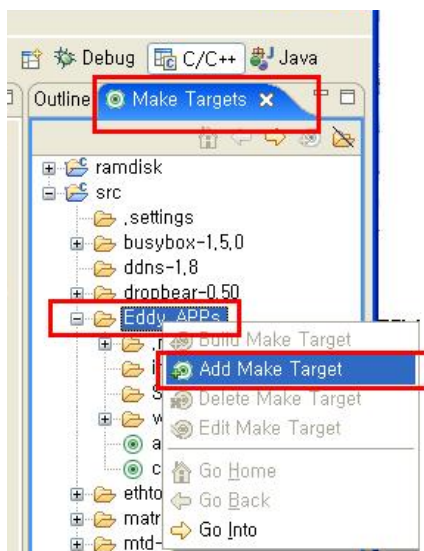
all : $(TARGET)

hello : hello.o
        rm -f $@
        $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o
        $(STRIP) $@

udp : udp.o
        rm -f $@
        $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o $(LIBS)
        $(STRIP) $@
..

```

### 5.3.2 Make Target 등록

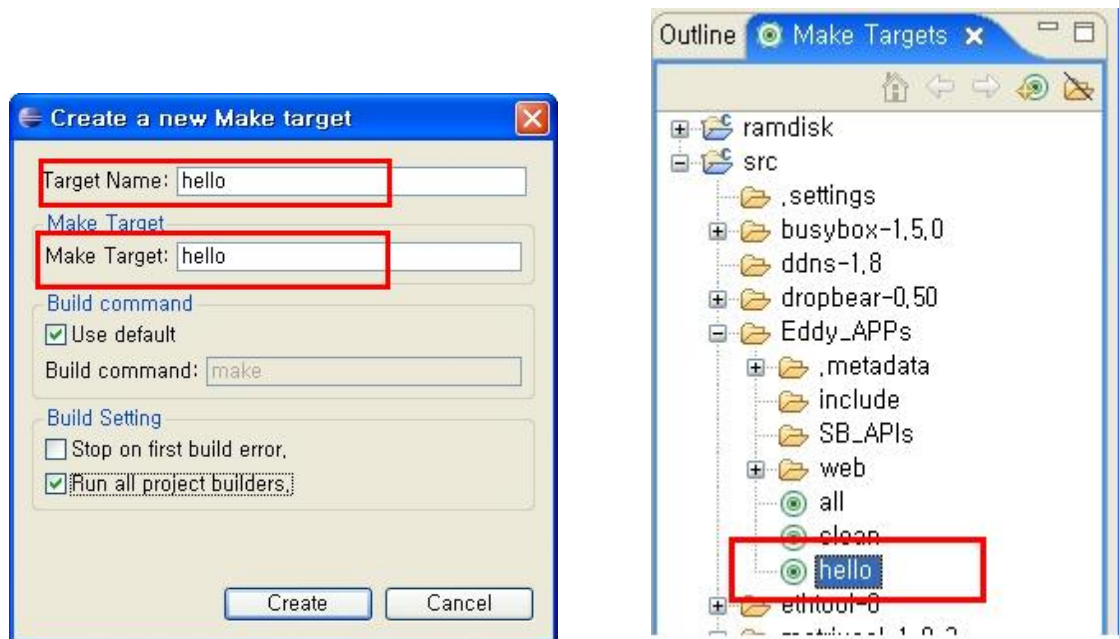


소스파일과 Makefile 을 작성하였다면, LemonIDE 에서 컴파일이 가능하도록 Make Target 에 등록한다.

Make Target 은 LemonIDE 우측에 Make View 화면에 존재하며, Make Target 등록은 소스파일과 Makefile 이 위치한 디렉토리를 선택하고 오른쪽 마우스를 클릭하여 “Add Make Target” 을 선택한다.

“5.2 소스 작성 및 편집” 에서 새로 작성한 소스파일의 “Hello.c” 이고 “5.3.1 Makefile 수정” 에서 Hello.c 에 대한 컴파일환경을 추가하였으므로 아래의 그림처럼 hello.c 에 대한 컴파일환경을 등록한다.

우측그림은 Make Targets 에 Hello 가 등록된 상태를 보여준다.



**Target Name** : 컴파일할 타겟 소스에 대한 Alias 이름 지정

**Make Target** : Make 실행 시 전달할 argument 값 (Makefile 에 등록된 컴파일 환경)

### 5.3.3 소스 컴파일

작성한 소스화일을 Eddy 에서 실행가능한 바이너리 파일로 컴파일한다.

컴파일 실행은 “5.3.2 Make Target 등록” 에서 작성한 Make Target 을 클릭하면 컴파일된다. ( “Make View” 화면의 “Hello” 를 더블 클릭한다)

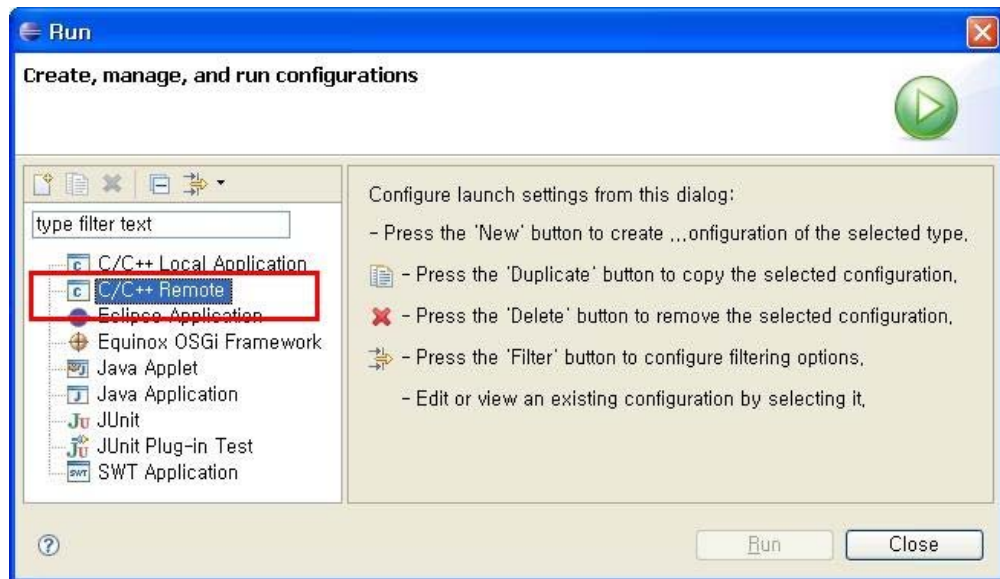
컴파일 되는 과정은 LemonIDE 화면 하단의 “Message View” 의 “Console” 란에서 확인할 수 있다.

DK Source 에서 이미 제공하는 “make Target” 에는 “All” 과 “Clean” 이 있으며, “All” 은 Makefile 에 등록된 모든 어플리케이션 소스를 컴파일하도록 하였고, “Clean” 은 컴파일된 모든 어플리케이션의 초기화하도록 구성된 Make Target 이다. ( “Makefile” 참조)

## 5.4 실행환경 Run Method 등록하기

이 장에서는 빌드된 바이너리화일 즉 Eddy 에서 실행가능한 바이너리 화일을 Eddy 에 업로드하여 실행 또는 디버깅을 하기위해 LemonIDE 에 Run/debug Method 를 작성하는 과정을 설명한다.

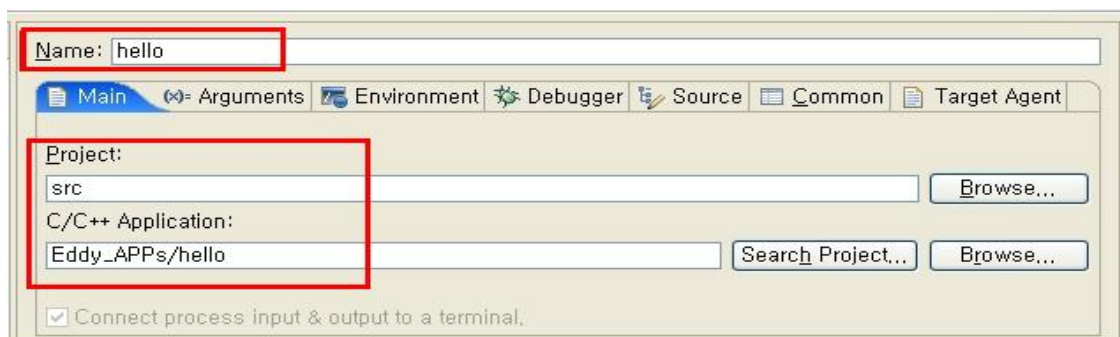
LemonIDE 의 타이틀 메뉴에서 “Run” → “Run...” 을 선택하면 Method 작성화면이 보여진다.



"C/C++ Remote" 를 더블클릭하게 되면 실행환경을 등록할 수 있는 Method 가 생성된다.

#### 5.4.1 Method 이름 및 바이너리 파일 등록

"Main" 탭을 선택하여 실행할 바이너리 파일에 대한 이름과 위치를 등록한다.

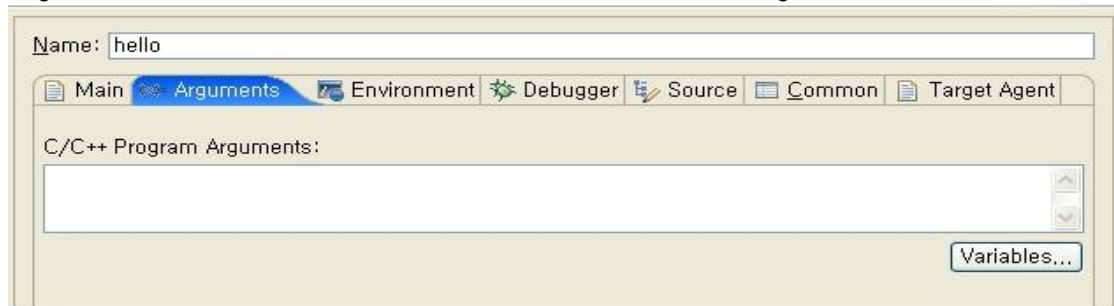


- Name : 실행할 Method 의 Alias 이름을 등록한다.
- Project : Workspace 의 어느 프로젝트에 속한 바이너리 파일인지 선택한다.
- C/C++ Application : 프로젝트 내의 실행할 바이너리 파일을 선택한다.  
그림의 예처럼 "Eddy\_APPS/hello" 같이 프로젝트 하단의 디렉토리와 바이너리파일 이름을 등록하거나, "Browser" 로  
"C:\filesystem\_2.1.x.x\src\Eddy\_APPS\hello"  
와 같이 절대 경로를 등록해도 무방하다.



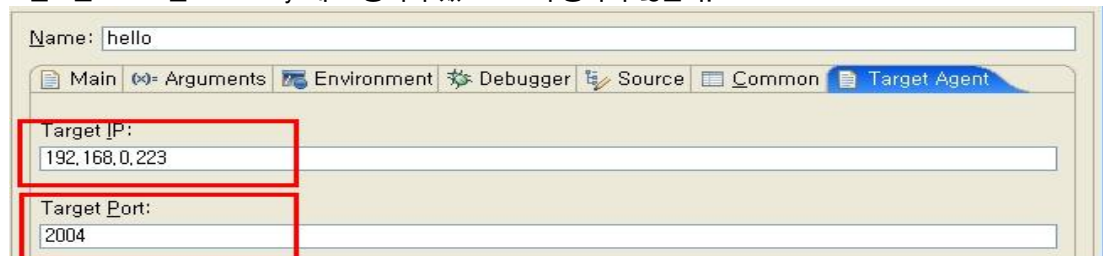
### 5.4.2 Arguments 설정

“Arguments” 탭을 선택하여 바이너리화일이 실행 시에 필요할 수 있는 argument 를 등록한다.



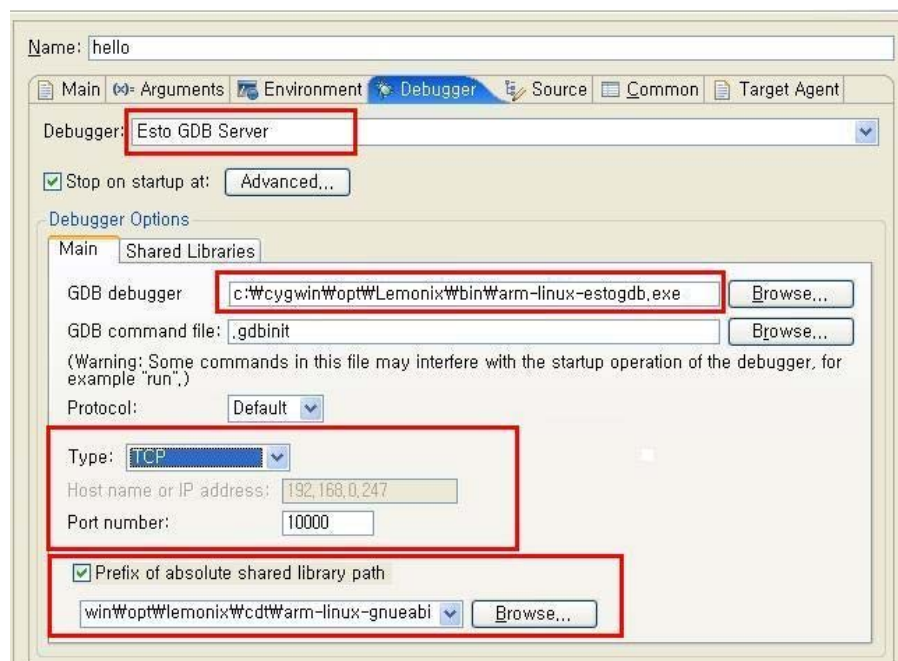
### 5.4.3 Target Agent 설정

“Target Agent” 탭을 선택하여 실행될 Target 장비인 Eddy 의 IP 주소와 포트를 등록한다.  
포트번호는 2004 번으로 Eddy 에 고정되어 있으므로 수정하지 않는다.



### 5.4.4 Debugger 설정

“Debugger” 탭을 선택하여 LemonIDE 에서 사용할 디버거를 등록한다.

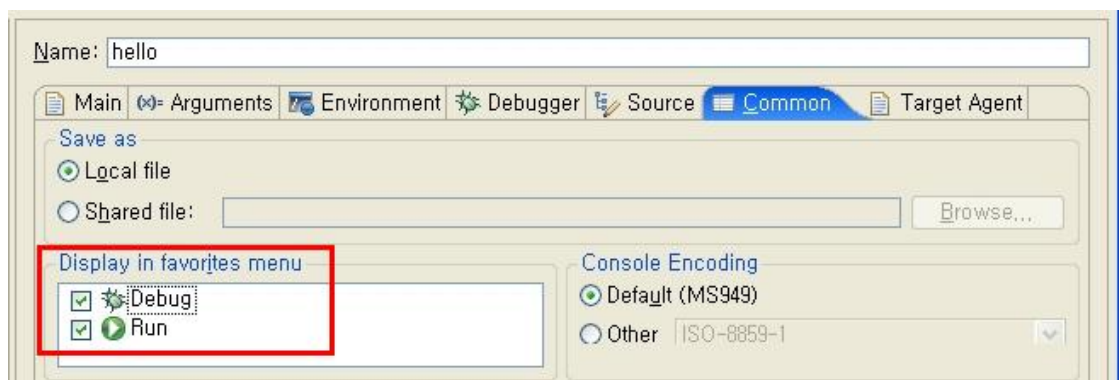




- Debugger : 디버거의 종류를 선택한다.  
Eddy 를 위한 디버거는 Windows 환경에서의 LemonIDE에서는 “**Esto GDB Server**” 를 선택하고 Linux 버전인 경우에는 “**LemonIDE GDB Server**” 를 선택한다.
- GDB Debugger : 위란의 Debugger 에서 선택한 디버거 실행화일의 실제 위치를 등록한다.  
Windows 환경에서의 LemonIDE 에서는 c:\cygwin\opt\lemoix\lib\arm-linux-estogdb.exe를 선택하고 Linux 버전인 경우에는 /opt/lemonix/arm-linux-lemonidegdb” 를 선택한다.
- Type : LemonIDE 와 타겟인 Eddy 와의 디버깅 정보 전송방식을 선택한다.  
이 버전에서는 TCP 만을 지원하며, Serial 은 지원하지 않으므로 “TCP” 를 선택하고 Port Number 는 “2004” 을 제외한 번호로 등록한다.
- Prefix of absolute shared library path : 디버깅을 사용하려면 Target의 루트 파일 시스템이 호스트에 도 설치 되어 있어야 한다. 특히, 루트 파일 시스템 중에서 /lib과 /usr/lib 은 디버깅하는데 있어서 꼭 필요한 라이브러리이다.호스트 컴퓨터의 /lib 디렉터리에 Target의 library가 없는 경우 디버깅에 문제가 발생할 수도 있기 때문에 타겟 디버깅을 위해서는 반드시 설정해주어야 한다.  
라이브러리가 있는 위치는 Windows 환경의 경우 “c:\cygwin\opt\lemonix\cdt\arm-linux-gnueabi” 이며, Linux 환경의 경우 “/opt/lemonix/cdt/arm-linux-gnueabi” 를 선택한다.

### 5.4.5 Common 설정

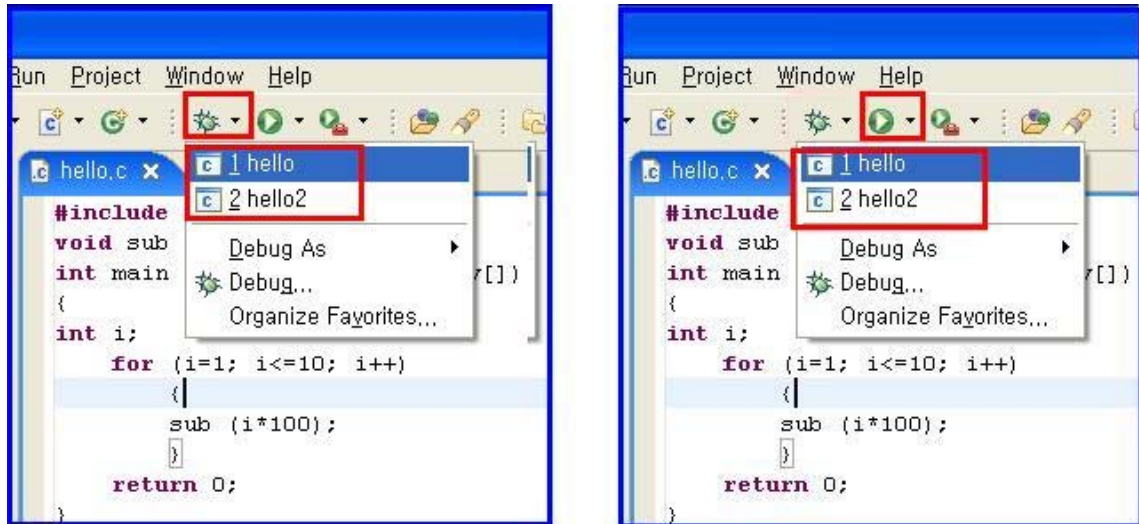
“Common” 탭을 선택하여 실행결과만 확인할 것인지 디버깅 정보까지 확인할 것인지를 설정한 다. 일반적으로 2가지 옵션을 모두 선택한다.



“Hello” 메소드에 대한 옵션을 모두 등록 하였으면 화면 하단의 “Close” 아이콘을 클릭하여 등록을 마친다.

### 5.4.6 Method 등록 결과

실행해 보고 싶은 바이너리 파일에 대해 위 과정을 모두 등록하게 되면 LemonIDE 화면 상단의 디버거아이콘과 실행아이콘 바로 옆에 아래를 향하는 방향 아이콘을 선택하면 다음과 같이 등록된 Method 가 보여진다.



만일 위 그림과 같이 “Hello” Method 가 등록 되지 않았다면, “Run” → “External Tools” → “Organize Favorites” → “Add” 를 선택하면 등록된 “Hello” 메소드가 표시되므로 이를 선택한다.

## 5.5 Run Method 실행하기

이 장에서는 빌드된 바이너리화일 즉 Eddy 에서 실행가능한 바이너리 화일을 Eddy 에 업로드하여 실행해 보는 과정을 소개한다. Eddy 에 업로드하여 실행해 보는 방법은 두가지가 있는데 첫번째는 LemonIDE 의 GDB Server 를 통해 업로드하고 실행 또는 디버깅을 간편하게 할 수 있는 방법과 FTP 로 전송하여 Telnet 에서 확인해보는 수동적인 방법이 있다. FTP 에 의한 방법은 “Eddy\_DK\_Programmer\_Guide” 를 참조하기 바라며 이 매뉴얼에서는 GDB Server 에 의한 방법을 설명한다.

위 과정에서 작성한 Hello.c 실행환경을 위한 Run Method 를 작성하였다면, LemonIDE 타이틀 메뉴의 Run 아이콘을 통해 실행할 수 있다.

### 5.5.1 Eddy 의 Target Agent 설정하기

LemonIDE 의 Target 시스템은 Eddy 이므로, Eddy 에서 LemonIDE 의 Run 또는 Debugging 요청에 대한 대응 Server 프로그램이 필요하다. 이것을 Target Agent 라고 하는데 Eddy 의 환경설정 부분에서 “LemonIDE Target Agent” 를 실행하도록 설정해 주어야 한다.

아래의 화면은 WEB 브라우저로 Eddy 에 접속하여 Target Agent 를 설정한 화면으로, Target Agent 를 Enable 로 설정 후에 저장하고 리셋해야한다.

(참고) Eddy 의 기본 콘솔포트는 시리얼포트이다. 그러므로 LemonIDE 를 통해 실행되는 결과는 모

두 Eddy DK 보드의 콘솔포트로 출력된다. 타겟보드가 Eddy DK 보드가 아닌 모델인 경우에는 콘솔 포트가 없으므로 시리얼포트를 통한 결과확인이 불가능하다. 이런 경우에는 Telnet 을 콘솔로 사용하도록 해야하므로 “LemonIDE Target Agent” 설정을 “Disable” 로 설정해야하며 Eddy 에 Telnet 으로 접속하여 Target Agent 를 수동으로 실행해야 한다.

**Eddy™** means real-time

[Network Settings] Device Name: Eddy Logged in as eddy

Setup Menu

- Summary
- Network Settings**
- Serial Settings
- GPIO Settings
- Change Password
- Update Firmware
- Factory Default
- Save & Reboot

Copyright 2007 SystemBase Co., Ltd. All rights reserved.

General Configuration	
Device Name	Eddy <a href="#">Help</a>
Line Type	Static IP <a href="#">Help</a>
IP Address	192.168.0.223 <a href="#">Help</a>
Subnet Mask	255.255.255.0 <a href="#">Help</a>
Gateway	192.168.0.254 <a href="#">Help</a>
DNS	168.126.63.1 <a href="#">Help</a>

Network Service Configuration	
PortView IP / Port	0.0.0.0 / 4000 <a href="#">Help</a>
SNMP Agent	Disable <a href="#">Help</a>
Telnet Service	Enable <a href="#">Help</a>
FTP Service	Enable <a href="#">Help</a>
WEB Service	Enable <a href="#">Help</a>
LemonIDE Target Agent	Enable <a href="#">Help</a>

### 5.5.2 Run Method 실행하기 (시리얼포트로 결과확인)

Eddy 의 Target Agent 실행은 기본적으로 Eddy의 시리얼 콘솔포트를 통해 출력된다. 그러므로 실행결과를 확인하려면 시리얼 통신 에뮬레이터 프로그램을 통해 확인해야 한다.

이 방법을 사용하려면 “5.5.1 Eddy 의 Target Agent 설정하기” 를 수행해야한다.

Eddy 의 콘솔포트는 DK 보드상에 “DEBUG PORT” 로 표시되어 있으며, 통신스펙은 115200 bps, None Parity, 8 Data Bits, 1 Stop Bit 이다. Windows 환경인 경우 Eddy 의 콘솔포트와 Windows PC 의 시리얼을 Cross 케이블 (Tx, Rx, GND 3선만 필요)로 연결한 다음 Windows 에 내장된 Hyper Terminal 을 실행하여 115200 bps, None Parity, 8 Data Bits, 1 Stop bit 로 설정한다.

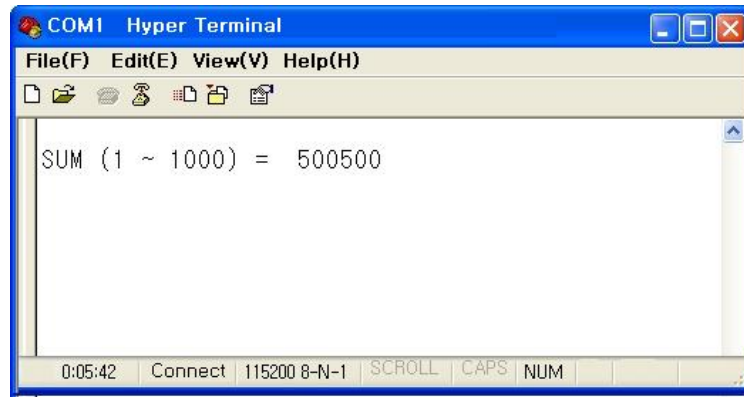
LemonIDE 가 지원하는 시리얼통신 에뮬레이터를 사용하려면 “7.3 Terminal” 을 참조한다.

실행준비가 완료됐으면 Run Method 를 클릭하여 바이너리를 Target System 인 Eddy 로 전송하여 실행결과를 확인한다.



아이콘을 클릭하면 마지막으로 실행했던 Run Method 가 처리되고, 아이콘 옆에 리스트 박스는 다수로 등록된 Run Method 중 하나를 선택할 수 있다.

아래 그림은 5.2 소스작성 및 편집에서 작성된 hello.c 의 Hyper Terminal 에서의 실행결과이다.



### 5.5.3 Run Method 실행하기 (Telnet 으로 결과확인)

Eddy DK 보드 이외의 모델에서는 시리얼 콘솔포트가 없으므로, 시리얼포트를 통한 결과확인是不가하다. 그러므로 콘솔포트를 Telnet 으로 사용할 수 있도록 해야한다.

이 방법을 사용하려면 “5.5.1 Eddy 의 Target Agent 설정하기” 를 수행해서는 안된다.

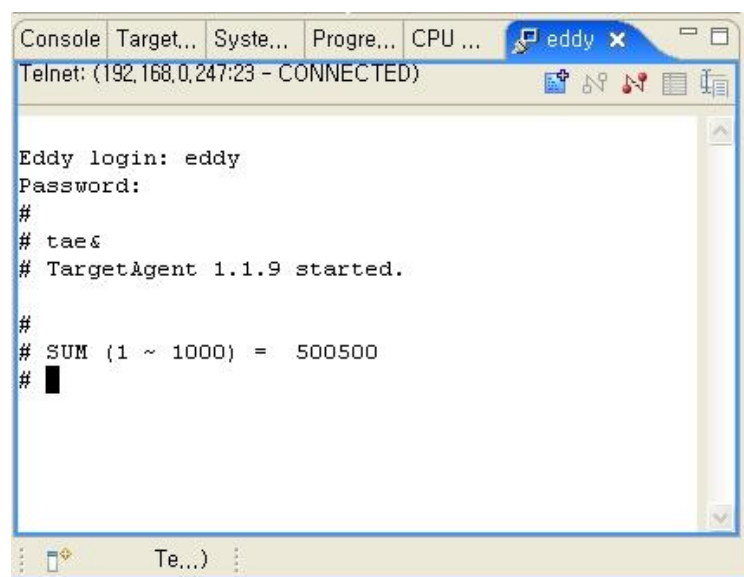
콘솔포트를 시리얼 포트에서 Telnet 으로 전환하려면 아래 그림과 같이 Terminal 프로그램으로 Eddy 에 로그인하여 수동으로 Eddy Target Agent 프로그램인 “tae” 를 백그라운드로 실행하여 LemonIDE 를 통한 바이너리 파일을 실행결과를 확인할 수 있다. Telnet 터미널 연결은 “7.3 Terminal” 을 참조바란다.

실행준비가 완료됐으면 Run Method 를 클릭하여 바이너리를 Target system 인 Eddy 로 전송하여 실행결과를 확인한다



아이콘을 클릭하면 마지막으로 실행했던 Run Method 가 처리되고, 아이콘 옆에 리스트 박스는 다수로 등록된 Run Method 중 하나를 선택할 수 있다.

아래 그림은 5.2 소스작성 및 편집에서 작성된 hello.c 의 Telnet 에서의 실행결과이다.



## 5.6 펌웨어 이미지 만들기

이 장에서는 응용 프로그램을 만들고 컴파일하여 타겟에 실행해 본 예제 프로그램을 Firmware 이미지 파일로 만들고 타겟에 적용하는 방법에 대해 소개한다.

### 5.6.1 Makefile 수정

펌웨어 이미지를 만들기 위해서는 Project 로 등록한 “Ramdisk” (DK 소스의 “Ramdisk” 디렉토리) 내의 MakeFile 을 수정한다. Makefile 은 펌웨어 이미지를 만들 때 필요한 버전정보, 사용할 Ramdisk 의 크기, 복사할 Application 등의 정보를 설정할 수 있다.

아래의 그림은 “Ramdisk” 프로젝트 밑에 “ramdisk” 디렉토리에 포함된 Makefile 에 새로 작성한 hello 바이너리 파일을 펌웨어 이미지에 추가되도록 하는 예이다.

#### (참고)

DK Source 는 기본적으로 Linux 환경으로 배포된다.

그러므로 Windows 환경에서 Makefile 내의 일부 명령을 인식하지 못하는 경우가 있다, 이런 경우를 대비하여 Makefile 내에 아래의 그림에서 처럼 일부 유틸리티의 이름 뒤에 .exe 를 추가 하여야 한다.

```
../tool/genext2fs → ../tool/genext2fs.exe
../tool/mkimage → ../tool/mkimage.exe
```

```
IMAGE=ramdisk
FW_NAME      =      eddy-fs-2.1.x.x.bin      → Firmware Name & Viersion Info

FIRMWARE_DIR =      ../firmware      → firmware is stored in a folder

## Check environments
#include ../Make.check

install:

    #@echo "Making ramdisk image..."
    #$(TOOL) -b 8192 -d root -D device_table.txt ramdisk
    #../tool/genext2fs -U -b 5110 -d root -D device_table.txt ramdisk
    #../tool/genext2fs -U -b 7158 -d root -D device_table.txt ramdisk
    #../tool/mkcramfs -q -D device_table.txt root ramdisk
    ../tool/genext2fs.exe -U -b 10240 -d - N 1024 root -D device_table.txt ramdisk → Ramdisk
    크기를 10,240 K 로 설정하며, Device_table.txt 를 참조하여 Eddy /dev 의 디바이스를 등록한다.
    gzip -vf9 ramdisk
    test -f ramdisk.gz
    ../tool/mkimage.exe -A arm -O linux -T ramdisk -C gzip -a 0 -e 0 -n $(FW_NAME) -d ../ramdisk.gz
    $(FW_NAME)
```

```
test -f $(FW_NAME)
mv $(FW_NAME) $(FIRMWARE_DIR)/
```

release: → Eddy 에 복사될 Application 을 해당 디렉토리에 복사되도록 등록

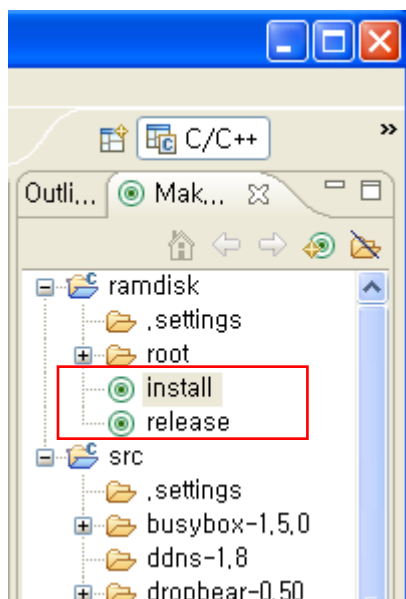
**cp -f ./src/Eddy\_Apps/hello root/sbin → Hello 바이너리파일을 Eddy Ramdisk 로 복사**

```
cp -f ./src/busybox-1.5.0/busybox root/bin
cp -f ./src/dropbear-0.50/dropbear root/usr/local/sbin
cp -f ./src/dropbear-0.50/dropbearkey root/usr/local/sbin
cp -f ./src/ethtool-6/ethtool root/usr/local/sbin
cp -f ./src/tftp-2.25b/tftp root/usr/local/sbin
cp -f ./src/ucd-snmp-3.6.2/agent/snmpd root/usr/local/sbin
cp -f ./src/ucd-snmp-3.6.2/agent/snmpd.conf root/etc
cp -f ./src/netkit-ftp-0.18/ftp/ftp root/usr/local/bin
```

### 5.6.2 Make Target 등록

펌웨어 이미지 작성용 Makefile 을 수정하였다면 Make Target 에 등록한다.

Make Target 은 LemonIDE 우측에 Make View 화면에 존재하며, 등록은 Ramdisk 프로젝트를 선택하고 오른쪽 마우스를 클릭하여 “Add Make Target” 을 선택한다. DK 소스에서는 미리 cfg, release, install 이 등록되어 있으므로 재 등록은 필요없으며, Make Target 등록에 대한 자세한 사항은 “5.3.2 Make Target 등록” 을 참조하기 바란다.



install ; Eddy 에서 사용할 Filesystem 을 펌웨어 이미지 파일로 생성한다.

release ; Makefile 의 release 에 등록된 바이너리파일을 Eddy ramdisk 디렉토리로 복사한다.

### 5.6.3 펌웨어 이미지 생성

작성한 바이너리 화일을 펌웨어 이미지로 만들어질 Eddy Ramdisk (Linux file system) 에 복사하기 위해 “Make View” 화면에 Ramdisk 프로젝트의 “Release” 를 더블클릭하면, Makefile 에 등록된 바이너리 파일들을 모두 Eddy Ramdisk 에 복사된다. 만일 작성된 바이너리 파일을 이미 “Ramdisk/root/sbin” 에 복사하였다면 “Release” 는 수행할 필요없다.

펌웨어 이미지로 만들 Eddy Ramdisk 가 준비되었으면, “Make View” 화면에 Ramdisk 프로젝트의 “install” 를 더블클릭하면, ramdisk 디렉토리를 펌웨어 이미지로 생성한다.

생성된 펌웨어 이미지는 ramdisk 의 firmware 디렉토리에 생성된다.

### 5.6.4 펌웨어 업데이트

생성된 펌웨어 파일을 Eddy 에 업로드하여 Flash Memory 에 저장한다.

펌웨어 업그레이드 방법은 FTP 에 의한 방법과, WEB 브라우저를 이용한 방법, 그리고 부트로드를 이용한 방법 3가지를 제공한다. Web 브라우저를 이용한 방법은 “Eddy-Serial-User\_Guide” 매뉴얼을 참고하기 바라며, FTP 서버를 이용한 업그레이드와 부트로드를 이용한 방법은 “Eddy\_DK\_Programmer\_Guide” 매뉴얼의 각각 “5.2 FTP 서버를 이용한 펌웨어 업데이트” 와 “9.1 시스템 복구” 를 참조하기 바란다.

## 6장. LemonIDE 디버거

LemonIDE 에서 지원하는 디버깅은 정지점디버깅과 비정지디버깅의 2가지 디버깅 방법을 제공한다.  
이 장에서는 “5.2 소스 작성 및 편집” 에서 작성한 hello.c 를 이용하여 디버깅하는 방법을 순서대로 설명한다.

정지점 추가  
원격 디버깅 환경 설정  
원격 디버깅 시작  
프로그램의 실행 제어  
함수 안으로 이동  
변수의 값, 레지스터의 값, 수식의 값을 확인하고 변경하기  
함수에서 빠져나가기

### 6.1 디버깅 준비

원격 디버깅을 위해서는 호스트에서 프로그램을 작성하고 디버깅하는 것과 다르게 몇 가지 환경을 설정 해주어야 한다.

#### 6.1.1 Target Agent 실행

“5.5.2 Run Method 실행하기 (시리얼포트로 결과확인)” 또는 “5.5.3 Run Method 실행하기 (Telnet 으로 결과 확인)” 을 참조하여 타겟 시스템인 Eddy 에 Target agent 를 실행한다.

#### 6.1.2 컴파일 옵션 설정 시 주의 사항

디버깅을 하기 위해서는 다음과 같은 옵션 설정에 주의를 해야 한다.

“5.3.1 Makefile 수정 및 작성” 에서의 Makefile 에는 최적화 옵션으로 컴파일되도록 설정되어 있어 정지점 및 비정지점 디버깅을 바로 실행할 수 없다. 디버깅을 위해서는 아래 표와 같이 Makefile 의 CFLAGS 옵션을 수정 하여야만 한다.

디버깅을 배제한 최적화 옵션	-O, -O1, -O2, -O3	옵티마이즈 옵션으로 사이즈 우선이나, 속도우선으로 바이너리 화일을 만들기 위해 사용된다. 이 옵션을 사용하는 경우 디버깅이 불가하다
	strip	컴파일 시 사이즈를 줄이기 위해 사용되는 압축프로그램



		으로 이 옵션을 사용하는 경우 디버깅에 필요한 정보가 제거 되어 디버깅이 불가하다.
정지점디버깅 옵션	-g1 -g -g3	Minimal Debug Level의 정지점 디버깅 Default Debug Level 의 정지점 디버깅 Maximum Debug Level 의 정지점 디버깅
비정지점디버깅 옵션	-gstabs+	Nonstop Debug Level 의 비 정지점 디버깅

다음은 “5.3.1 Makefile 수정 및 작성” 에서의 Makefile 을 정지점디버깅을 위해 수정한 예로, 기존 설정에서 -O2 옵션과 \$(STRIP) \$@을 제거하였다. 비정지점디버깅을 위해선 -g 옵션을 -gstabs+ 로 수정하면 된다.

```

.
.
CFLAGS = -g -Wall -Wno-nonnull
.
.
hello : hello.o
    rm -f $@
    $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o
.
.

```

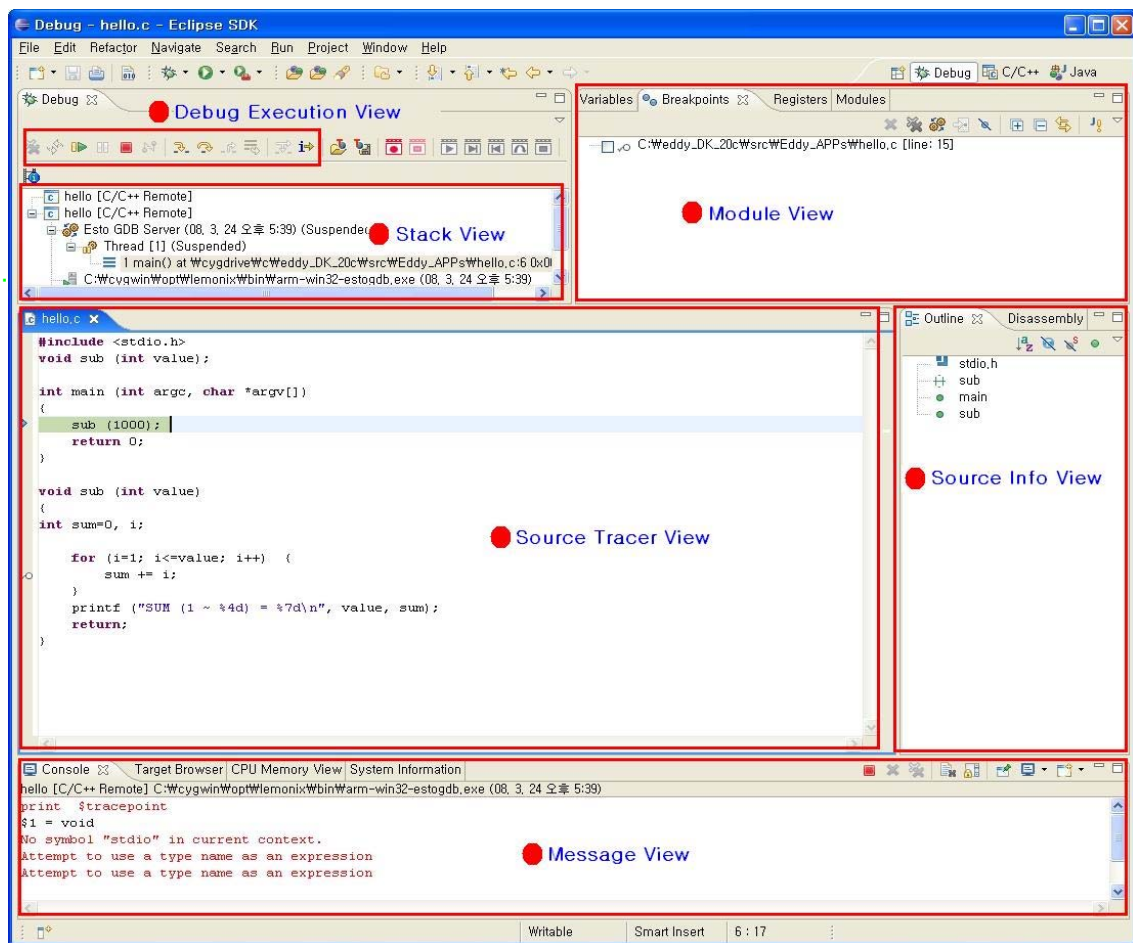
## 6.2 디버깅 실행

이 장에서는 “5.2 소스 작성 및 편집” 에서 작성한 hello.c 를 이용하여 디버깅 방법을 설명한다.



타겟인 Eddy 에 업로드하여 디버깅(정지, 비정지)하기위한 아이콘으로, 이 아이콘을 클릭하면 마지막으로 실행했던 Debug Method 가 처리된다. 이이콘 옆에 리스트 박스는 다수로 등록된 Debug Method 중 하나를 선택할 수 있다.

디버깅 시작 시에는 소스를 코딩하고 컴파일하여 실행했던 “C/C++” 화면에서 “Debug” 화면으로 변경된다. 아래 그림은 디버깅 시의 변경된 화면이다.



Stack View	디버깅 중 프로그램의 쓰레드와 스택정보를 보여준다.
Module View	디버깅 수행 시의 변수, 정지점, 수식, 모듈에 대한 정보를 보여준다.
Source Tracer view	라인단위로 소스코드의 디버깅 위치를 보여준다.
Source Info View	소스화일에 대한 관련 함수 및 역어셈블리 코드를 보여준다.
Message View	타겟의 기타 상태정보를 보여준다.
Debug Execution View	디버깅 시작, 정지, 브레이크 포인트 단위의 실행을 제어한다.

## 6.3 정지점디버깅

이 장에서는 정지점디버깅 시에 사용방법을 설명한다.

디버깅이 완료된 후 완성된 프로그램의 경우 디버깅을 위해 수정한 CFLAGS 옵션과 STRIP 을 복원하여 실행모듈을 최적화하는 것을 권장한다.

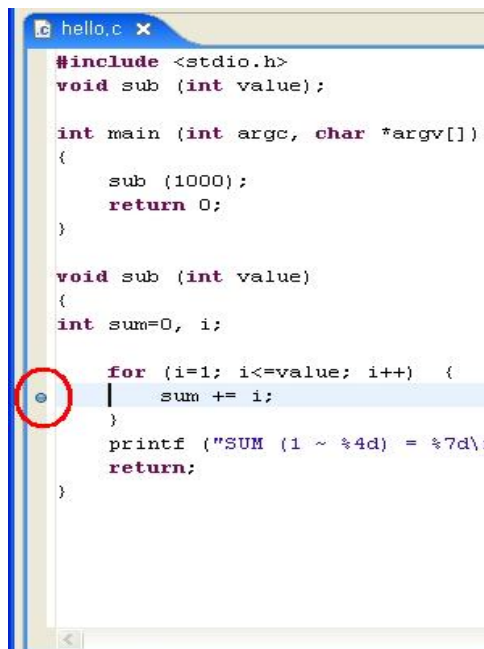
### 6.3.1 컴파일 환경 수정 및 디버깅 실행

“6.1.2 컴파일 옵션설정시 주의사항” 에서 언급했듯이, 디버깅을 위해서는 MakeFile 을 수정하여 디버깅이 가능하도록 수정 해야한다. 아래는 최적화 옵션을 제거하고 정지점 디버깅이 가능한 옵션으로 수정하였고, STRIP 을 제거하여 실행모듈을 압축 하지 않도록 하였다.

```
CFLAGS = -g -Wall -Wno-nonnull

hello : hello.o
    rm -f $@
    $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o
#    $(STRIP) $@
```

Makefile 을 수정한 후 저장하고 디버깅이 가능하도록 hello.c 를 다시 컴파일 하고, 를 통해 hello 를 디버깅 상태로 실행한다.



### 6.3.2 정지점 추가하기

정지점은 C/C++ 퍼스펙티브나 디버그 퍼스펙티브에서 추가할 수 있다. 정지점을 추가하고자 하는 소스코드 라인에서 에디터뷰의 왼쪽 부분을 클릭하면 정지점이 추가되고, 다시 더블 클릭하면 정지점이 제거된다. 소스코드

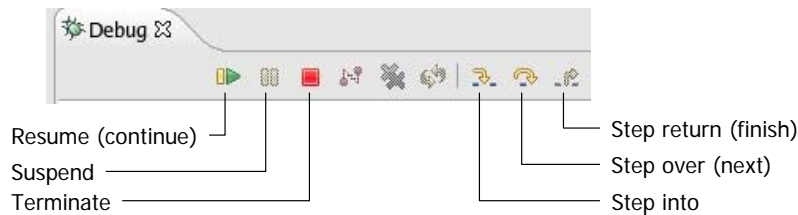
라인에서 마우스 오른쪽 버튼을 누르면 나오는 단축메뉴에서 정지점을 추가할 수도 있다.

추가한 정지점은 “Module View” 화면의 “BreakPoints” 에 추가된다

“Windows” → “Show View” → “Other..” → “Debug” 를 선택하면 각종 디버깅 정보를 선택하여 확인할 수 있다.

### 6.3.3 프로그램의 실행 제어

디버깅 시 프로그램의 실행을 제어하기 위해서 “Debug Execution View” 의 명령에 의해 실행된다.



사용하는 주요 명령어는 다음과 같다.

Resume: 다음 정지점을 만날 때까지 계속 실행

Suspend: 실행 중인 프로그램을 일시 정지

Terminate: 디버깅 세션을 완전히 종료

Step into: 다음 문장으로 이동하되 함수인 경우 함수 안으로 이동

Step over: 다음 문장으로 이동하되 함수인 경우 함수 안으로 들어가지 않음

Step return: 함수의 끝까지 실행함

## 6.4 비정지 디버깅

이 장에서는 비정지점 디버깅의 사용방법을 설명한다.

정지점 디버깅은 BreakPoint 로 지정한 라인까지 단계별로 디버깅이 가능한 디버깅 방법이지만,

비정지 디버깅은 프로그램을 계속 수행하는 동안 Tracepoint 로 설정한 특정 구간을 감시하고 그 결과를 저장하여 디버깅을 마친 다음 저장된 디버깅 정보로 실행상태를 되돌려 볼 수 있는 디버깅 방법이다.

### 6.4.1 컴파일 환경 수정 및 디버깅 실행

“6.1.2 컴파일 옵션설정시 주의사항” 에서 언급했듯이, 디버깅을 위해서는 MakeFile 을 수정하여 디버깅이 가능하도록 수정 해야한다. 아래는 최적화 옵션을 제거하고 정지점 디버깅이 가능한 옵션으로 수정하였고, STRIP 을 제거하여 실행모듈을 압축 하지 않도록 하였다.

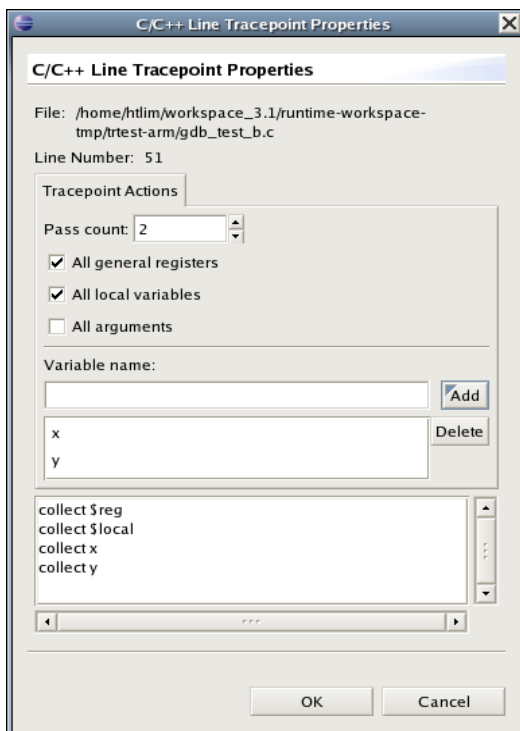
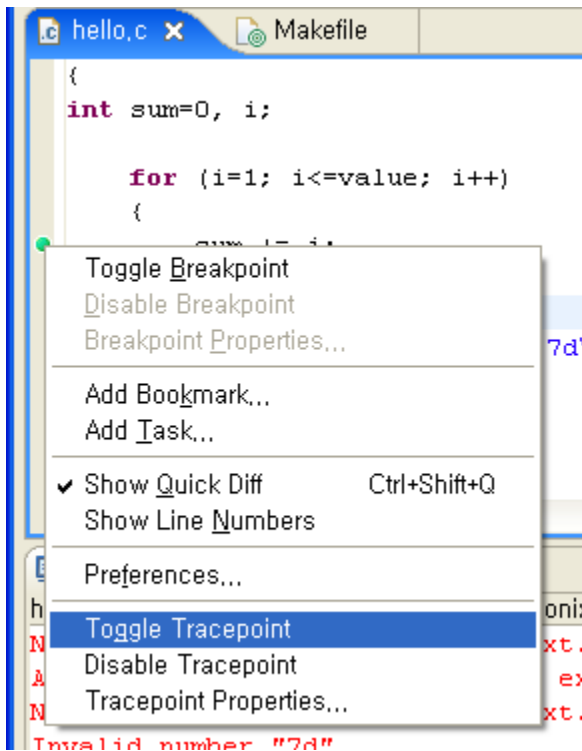
```
CFLAGS = -gstabs+ -Wall -Wno-nonnull

hello : hello.o
    rm -f $@
    $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o
# $(STRIP) $@
```

Makefile 을 수정한 후 저장하고 디버깅이 가능하도록 hello.c 를 다시 컴파일 하고, 를 통해 hello 를 디버깅 상태로 실행한다.



#### 6.4.2 추적점 설치 및 action 지정



새로운 Tracepoint 를 등록한다.

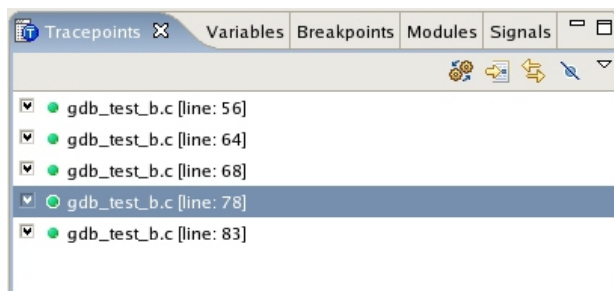
Tracepoint 로 등록할 소스 코드 (17 라인) 왼쪽에서 마우스 오른쪽 버튼을 선택하면, 다음 그림처럼 단축메뉴에서 “Toggle Tracepoint” 를 선택하면 추적점에 대한 액션을 지정할 수 있는 Tracepoint Properties 윈도우가 나타난다.

“Tracepoint Properties”에서는 Pass count, 레지스터, 지역 변수, 인자를 선택할 수 있고, 원하는 변수만 지정할 수도 있다.

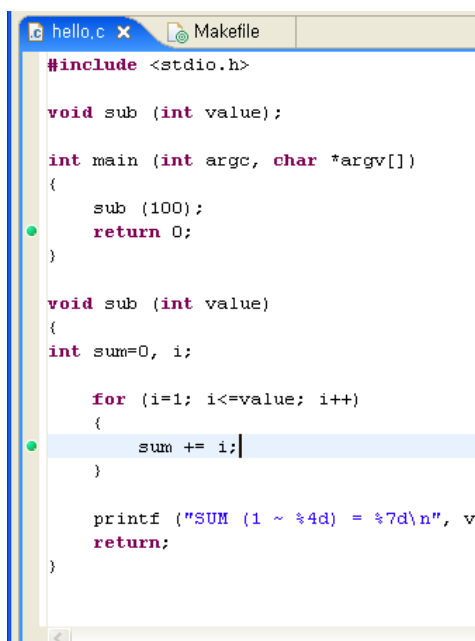
Tracepoint Properties 윈도우의 OK 버튼을 누른다.

그러면 초록색의 추적점이 생긴다.

Pass Count	이 라인이 실행되는 횟수가 만족할 때 비정지 디버깅을 종료한다. 비정지 디버깅은 추적을 종료할 위치를 반드시 명시되어야 추적이 끝난 후 상태를 되돌려 확인할 수 있다. 0 으로 설정 시 추적을 계속하며, 0 보다 큰 수일 때 이 라인이 실행되는 횟수에서 디버깅을 멈춘다. 실제 디버깅 시에는 디버깅 구간을 여러단계로 설정하는 경우가 많으므로 중간 라인의 경우 0 으로 설정하고, 마지막 종료라인에 0 보다 큰수를 등록한다.
All General registers	Register 정보 내용을 확인한다.
All local Variable	실행 모듈내의 전체 변수의 값을 확인한다.
All arguments	실행 모듈에 전달된 argument 값을 확인한다.
Variable name	실행모듈내의 특정한 변수의 값만을 확인하고자 하는경우 변수이름을 등록한다.



“Windows” → “Show View” → “TracePoint” 메뉴를 선택하면 현재 워크스페이스에 추가된 추적점들의 리스트를 볼 수 있다.



그림의 예에서는 8 번째 라인과 17 번째 라인을 Tracepoint 로 설정하였다.

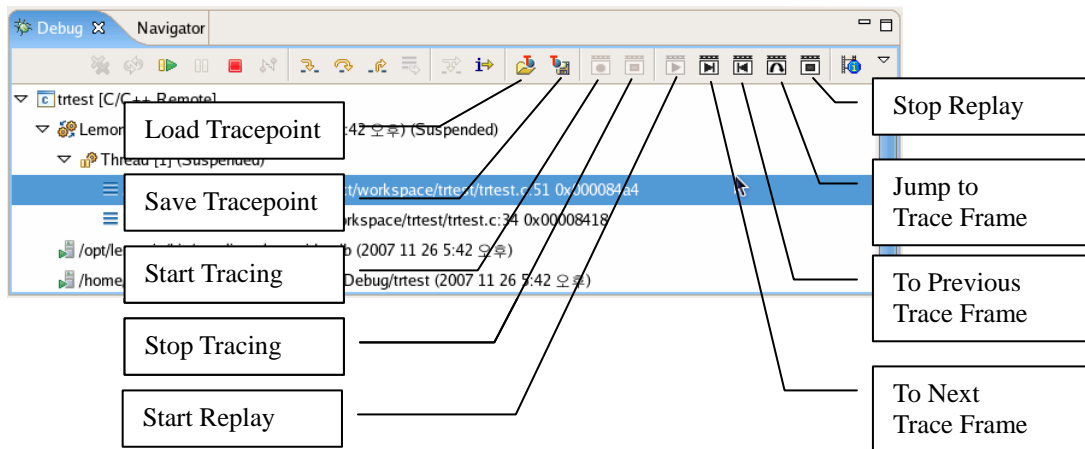
17 라인에는 Pass Count 는 0 으로 설정하여

추적 시작위치를 설정하였고, 8번째 라인에는


Pass Count 를 1로 하여 이 라인이 1첫째 실행될때가 프로그램 이 종료되는 위치이므로 이 라인에서 추적을 종료하게 하였다.


### 6.4.3 프로그램의 추적 제어


비정지 디버깅에서 추적과 재연에 사용되는 버튼은 다음과 같다.

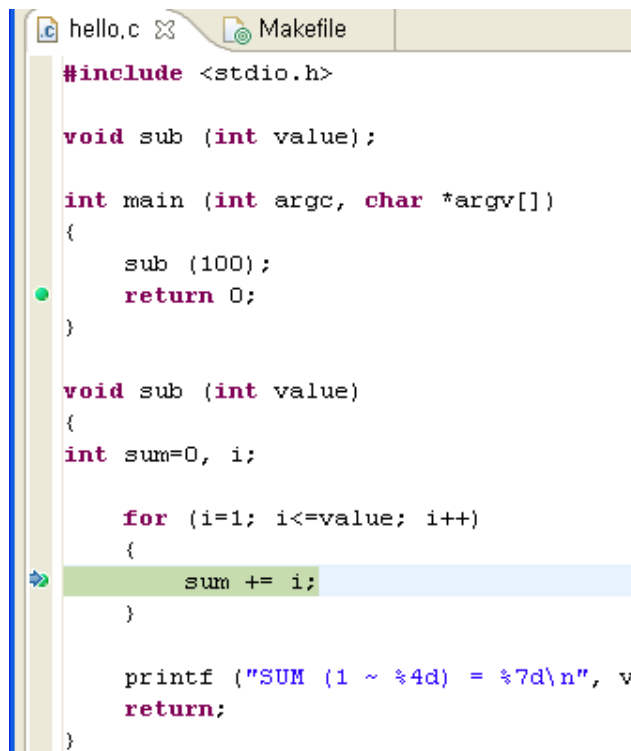



#### 추적 시작 및 종료



추적 시작 버튼(  )을 누르면 Target에 있는 hello 프로그램이 실행되면서 추적이 진행된다.

Target의 화면을 보면서 추적이 종료되었는지 확인한다. (추적이 종료되면 Target 화면 (시리얼 포트 또는 Telnet)에 “Trace Finished.” 라고 출력된다.) 추적이 종료되었으면 추적 종료 버튼(  )을 누른다.

정상적으로 추적이 종료된 경우 재연 시작 버튼(  )이 활성화되고 디버그 뷰, 변수 뷰, 레지스터 뷰 등은 추적이 끝난 지점에 대한 정보를 보여준다.



재연 시작 버튼(  )을 누르면 다음과 같이 첫 번째 추적 프레임에서 멈춘다.



 버튼을 누르면 다음 추적 프레임으로 이동된다.  버튼을 누르면 이전 추적 프레임으로 이동된다.

각 추적점에서 Variables view와 Registers View의 내용을 확인해본다.

Variables View에서 유의할 점은 추적하지 않은 지역 변수도 나타난다는 점이다. 추적한 변수와 추적하지 않은 변수가 구분되지 않기 때문에 사용자가 유의해서 봐야 한다.

각 추적점에서 TraceDump 뷰와 LemonIDE 하단의 Console 화면에서 추적 단계를 확인할 수 있다.

Variables Breakpoints TraceDump Modules Signals							
Trace Dump							
Trace	Frame	Time	PC	FP	SP	file	Line
5	0		0x85b0	0xbeffffe0c	0xbeffffce0	../gdb_test_b.c	51
6	1		0x85c0	0xbeffffe0c	0xbeffffce0	../gdb_test_b.c	54
3	2		0x8684	0xbeffffcdc	0xbeffffcc0	../gdb_test_b.c	78
4	3		0x86a8	0xbeffffcdc	0xbeffffcc0	../gdb_test_b.c	83
1	10		0x8610	0xbeffffe0c	0xbeffffce0	../gdb_test_b.c	64

디버깅을 종료하려면 Resume (  ) 버튼을 눌러서 끝까지 실행시키거나 Terminate (  ) 버튼을 눌러서 디버깅 세션을 강제로 종료시킨다.

“Windows” → “Show View” → “Other..” → “Debug” 를 선택하면 각종 디버깅 정보를 선택하여 확인할 수 있다.



## 7장. 모니터링 도구

이 장에서는 LemonIDE 를 통해 Target 시스템의 정보를 모니터링 할 수 있는 Target 브라우저와 모니터링 도구에 대해 설명한다.

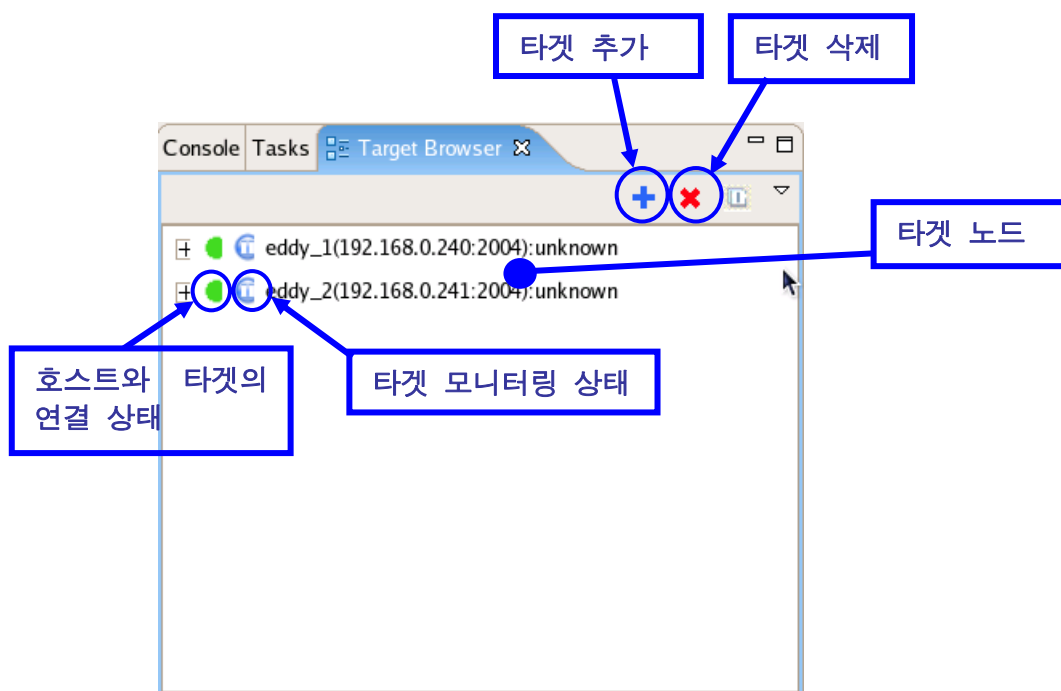
모니터링 도구는 LemonIDE 타이틀 메뉴상에 “Windows” → “Show View” 를 선택하면 LemonIDE 에서 지원하는 모든 모니터링 도구를 선택할 수 있어 LemonIDE 유저 인터페이스를 사용자 환경에 맞도록 재 구성할 수 있다.

### 7.1 Target 브라우저

먼저 Target 브라우저의 유저 인터페이스를 알아본 후에 사용하는 방법을 설명한다.

#### 7.1.1 Target 브라우저의 유저 인터페이스

Target 브라우저의 모양은 아래의 그림과 같다.



Target 브라우저에는 여러 개의 Target 노드가 존재할 수 있다. Target 노드는 다음과 같은 정보를 보여준다.

호스트와 Target의 연결 상태

원격 모니터링 상태

Target 이름

IP 주소

포트 번호

Target의 아키텍처

호스트와 Target의 연결 상태를 나타내는 아이콘은 연결 상태에 따라서 색상이 다르다.

빨간색: Target과 연결되지 않았거나 Target이 존재하지 않음

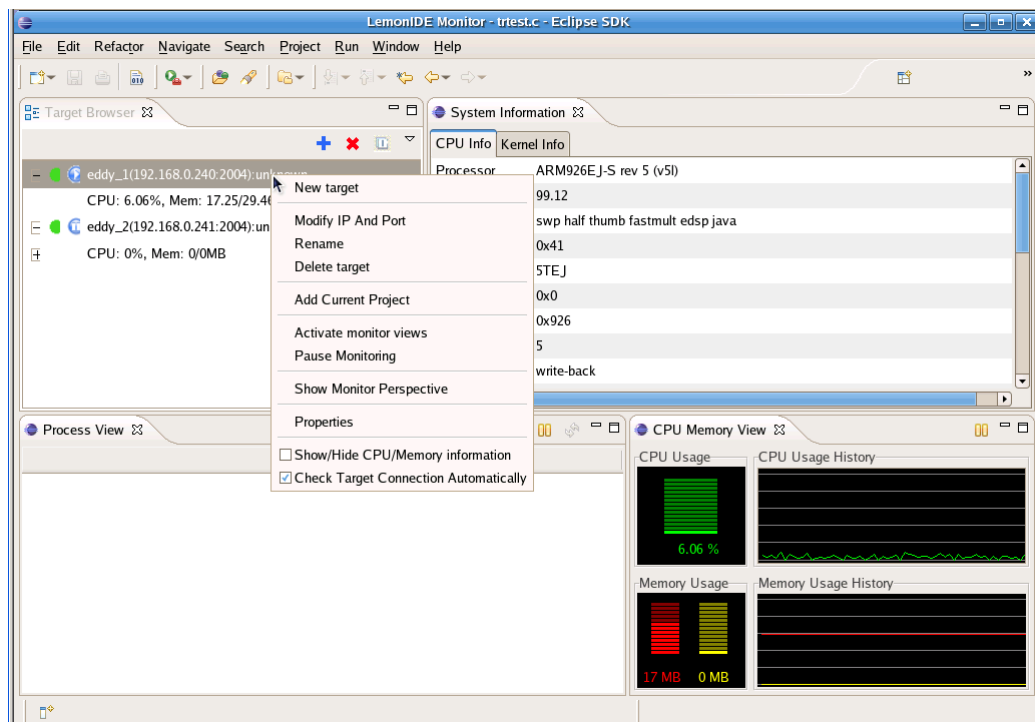
(ping에 응답이 없는 경우)

주황색: Target은 살아있고 네트워크로 연결되어 있지만 Target Agent와 연결되지 않은 경우

초록색: Target Agent와 연결되어서 원격 개발을 할 수 있는 경우

Target에 대해 원격 모니터링 중이면, Target 모니터링 상태 아이콘이 Play 모양이 되고, 원격 모니터링을 하고 있지 않으면 Pause 모양으로 표시된다.

Target 브라우저의 각 Target은 여러 개의 실행 설정이 등록될 수 있다. 실행 설정이란 프로젝트의 실행과 디버깅에 대한 정보를 갖고 있는 개체로서 Eclipse LemonIDE의 Run /Debug Configuration 창에서 생성, 삭제, 수정이 가능하다.



Hide CPU/Memory Information: 체크하지 않으면 Target 노드의 첫번째 서브 노드로 CPU와 메모리 사용량 정보가 표시된다. 단, 원격 모니터링 중이어야 실제 값이 표시된다. 이것은 Target 노드별로는 설정할 수 없고, 전체 Target 노드에 대해서 숨기거나 보여주게 할 수 있다.

Check Target Connection Automatically: Target 브라우저에 등록된 Target들과 호스트 간의 연결 상태를 자동으로 확인하여 보여줄 것인지 여부를 선택한다.

Add Current Project: C/C++ Project 뷰에서 선택된 프로젝트가 “Current Project” 이다. 이 프로젝트에 대한 실행 설정을 현재 Target 노드 밑에 생성해준다. 단, C/C++ Project 뷰에서 프로젝트 이름이 아닌 소스 코드와 같

은 것을 선택한 상태에서는 프로젝트의 실행 설정이 추가되지 않는다.

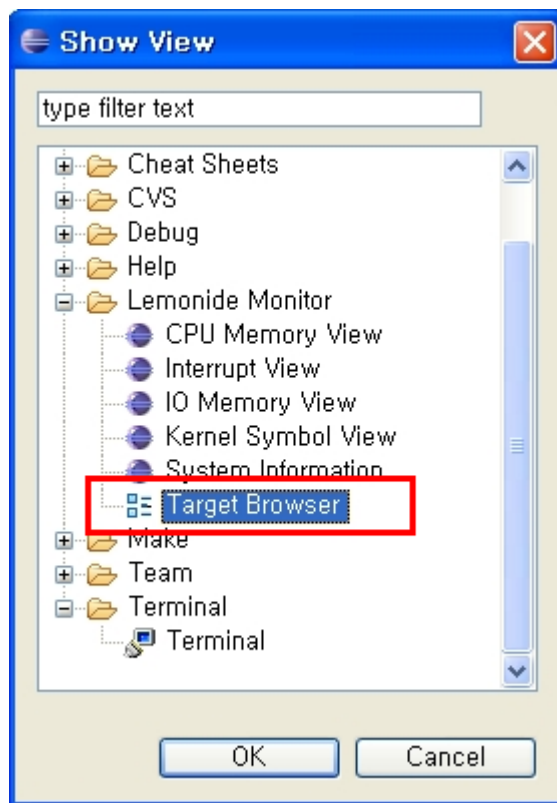
Activate Monitor Views: Target 상태 모니터의 뷰들을 활성화시킨다. CPU/메모리 뷰와 프로세스 리스트 뷰는 주기적인 모니터링을 시작한다.

Pause Monitoring: Target에 대한 원격 모니터링을 중지한다. CPU/메모리 뷰와 프로세스 리스트 뷰는 주기적인 모니터링이 중지된다.

Show Monitor Perspectives: LemonIDE Monitor 퍼스펙티브로 이동한다.

### 7.1.2 Target 브라우저에 Target 추가

LemonIDE 타이틀 메뉴상의 “Windows” → “Show View” → “Other...” 를 선택하여 “Show View” → “LemonIDE Monitor” → Target Browser” 를 선택하면 LemonIDE 하단의 “Message View” 에 Target Browser 가 나타난다.



Target 브라우저에 원하는 Target이 없을 경우 Target을 추가해야 한다. Target 브라우저 우측에 파란색의 + 마크를 클릭하면 아래와 같은 Target 입력 윈도우가 나타난다. 이 곳에 Target의 이름, IP 주소, Target에 실행 중인 Target Agent의 포트 번호를 입력한다.

그러면, 새로운 Target 노드가 Target 브라우저에 생성된다.

다음으로 Target 브라우저의 Target 노드에 실행 설정 추가한다. 예제인 face 프로젝트를 드래그하여 Target 브라우저의 eddy\_1 Target 노드에서 드랍한다.

프로젝트를 Target 노드에 드랍함과 동시에 Debugger Select 윈도우가 나타난다. 처음에는 “gdb” 가 선택되어

있을 것이다. Target에 맞게 디버거를 선택한다. 디버거를 선택하면 그에 맞는 기본적인 “Absolute Shared Library Path” 가 다음과 같이 설정된다.

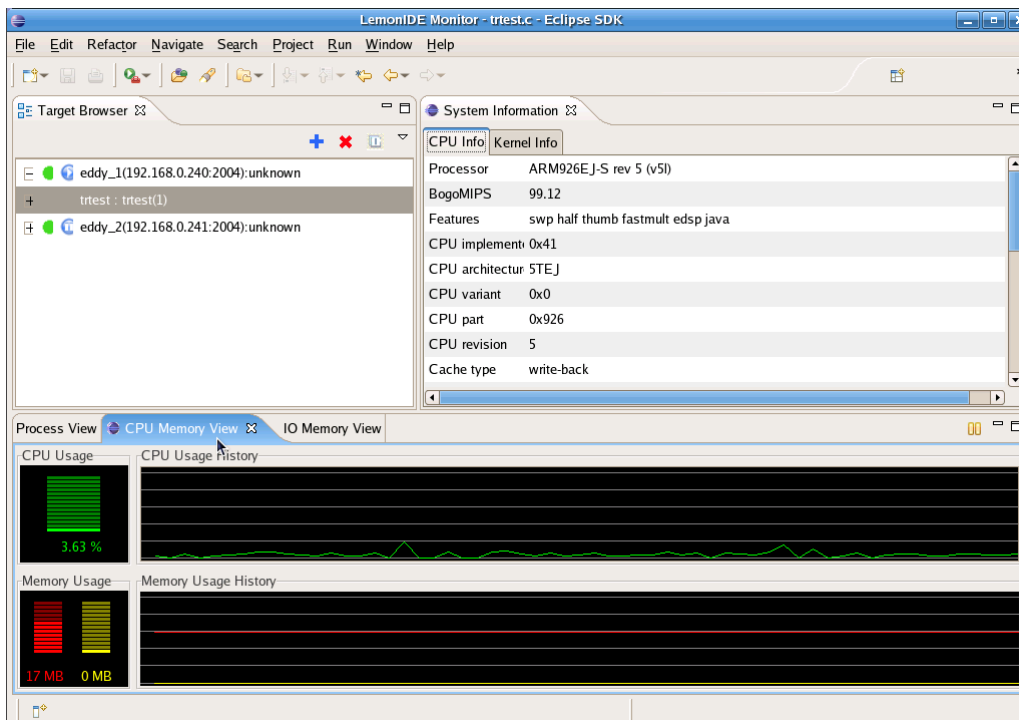
“Absolute Shared Library Path” 의 경로를 기본값이 아닌 곳으로 변경해야 하는 경우 Browse 버튼을 눌러서 다른 경로로 설정하면 된다.

“arm-linux-LemonIDEgdb” → “/opt/LemonIDE/cdt/arm-linux”

Debugger Selection 윈도우에서 디버거 설정을 마치고 OK 버튼을 누르면 노드에 trtest 프로젝트에 대한 Run/Debug Configuration (본 문서에서는 실행 설정이라고도 함)이 생성된다.

## 7.2 Target 상태 모니터

Target 브라우저에서 Target 노드의 “Show Monitor Perspectives” 메뉴를 선택한다. 그러면, 그림과 같은 LemonIDE Monitor 퍼스펙티브로 이동한다. 아직 원격 모니터링을 하지 않았기 때문에 뷰들에 아무것도 표시되지 않았다.

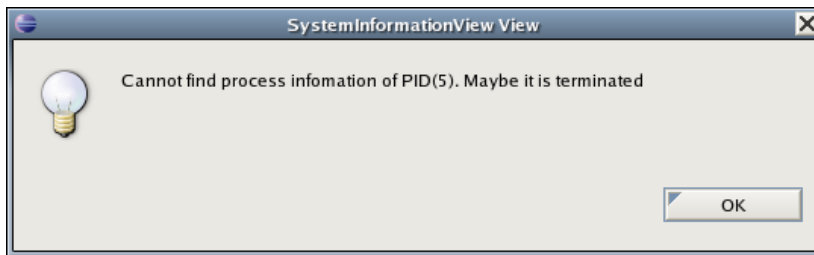


eddy\_1 Target 노드의 단축 메뉴에서 “Activate Monitor Views” 메뉴를 선택한다

그러면, 위의 그림에서처럼 원격 모니터링이 시작되어서 CPU/메모리 뷰에 그래프가 그려지고 프로세스 리스트 뷰는 10초마다 내용이 갱신된다.

eddy\_1 Target 노드의 단축 메뉴에서 “Pause Monitoring” 메뉴를 선택한다. 원격 모니터링이 멈추는 것을 확인한 후에 다시 “Activate Monitor Views” 메뉴를 선택하여 원격 모니터링을 계속 실행시킨다.

프로세스 리스트 뷰에 있는 프로세스를 kill 시켰지만, 이미 해당 프로세스는 저절로 종료되어서 존재하지 않을 수 있다. 이런 경우 아래와 같은 메시지가 출력된다.



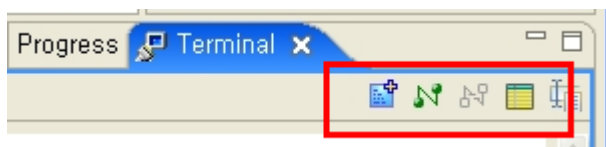
“Windows Show View” 메뉴를 선택하면 LemonIDE Monitor 에 있는 뷰들을 볼 수 있다.

## 7.3 Terminal

타겟을 통해 원격실행과 원격디버깅을 수행하기 위해서는 터미널 프로그램이 필요하다.

터미널 프로그램은 시리얼 에뮬레이터와 Telnet 에뮬레이터로 구분할 수 있는데, LemonIDE 는 이 두가지 터미널을 모두 지원하여 LemonIDE 내에서 타겟에 대한 원격실행과 디버깅을 확인 할 수 있다.

이 기능을 사용하려면 LemonIDE 타이틀 메뉴의 “Windows” → “Show View” → “Other...” → “Terminal” → “Terminal” 을 선택한다. Terminal 의 생성은 LemonIDE 하단의 “Message View” 에 생성 된다.



	New Terminal	: 새로운 터미널을 생성한다.
	Connect	: 등록된 Target 으로 접속한다.
	Disconnect	: 연결된 접속을 종료한다.
	Settings	: 접속할 Target 과 연결할 정보를 설정한다.

터미널을 LemonIDE Message View 안에서가 아닌 외부 독립창으로 실행하려면 Terminal 이 실행된 타이틀 이름에서 오른쪽 마우스를 눌러 “Detached” 메뉴를 선택한다.

LemonIDE 하단의 “Message View” 에 새로 생성된 터미널을 선택하고 으로 접속환경을 설정한다.

View Title : 터미널의 이름을 설정한다.

Connection Type : 사용하고자하는 터미널을 선택한다. (Serial, Telnet)

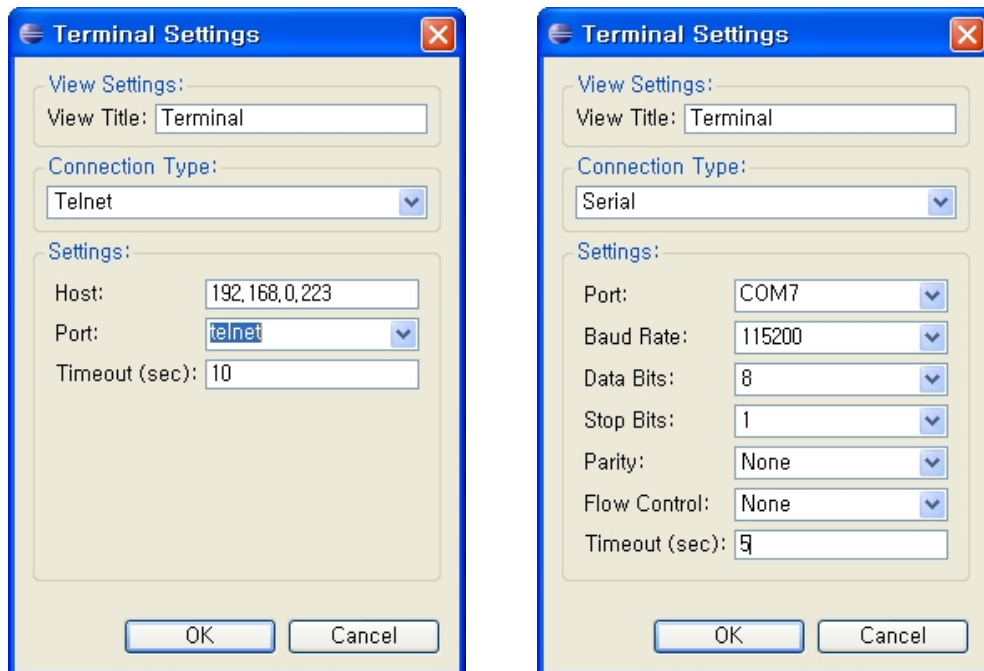
아래 그림의 경우 좌측은 Telnet 터미널의 설정화면이고, 우측은 시리얼 터미널의 설정화면이다.

#### (참고)

시리얼 터미널을 사용하고자 하는 경우에는 다음 작업을 선행하여야 한다.

/opt/lemonix/install\_files/ 디렉토리의 install 파일을 참조하여 LemonIDE 가 설치된 OS 환경에 맞추어 jar 파일과 dll 파일을 시스템에 복사하면 된다.

Windows 버전의 경우에는 Windows/i386-mingw32 디렉토리 내의 2개의 DLL 파일을 PC 의 Windows/system32 디렉토리에 복사하고, RXTXcomm.jar 파일은 “ 3.2.1 JDK 설치하기 ” 에서 설치한 JDK 디렉토리의 jre/lib/ext 디렉토리에 복사한 후 LemonIDE 를 재 기동한다.



#### Telnet 터미널

Host : 접속할 타겟의 IP 주소

Port : 접속할 포트번호 (Telnet 선택)

Timeout : 접속 후 지정시간 이내에 입력이 없으면 접속종료

#### Serial 터미널

Port : 타겟과 연결된 LemonIDE 가 설치된 PC 의 시리얼 포트번호

Baud Rate : 타겟과 통신할 통신속도 (115200 bps 로 설정함)

Data Bits : 데이터 비트수

## 7.4 CPU Memory View

LemonIDE 로 개발하는 타겟 시스템의 CPU 트래픽과 메모리 사용량을 확인할 수 있다.

이 기능을 사용하려면 LemonIDE 타이틀 메뉴의 “Windows” → “Show View” → “Other...” → “Esto Monitor” → “CPU Memory View” 을 선택한다. CPU View 는 LemonIDE 하단의 “Message View” 에 생성된다.



아이콘은 CPU 및 메모리의 상태를 확인을 시작하는 아이콘이고  아이콘은 중지 아이콘이다.

“CPU Memory View” 을 LemonIDE Message View 안에서가 아닌 외부 독립창으로 실행하려면 “CPU Memory View” 이 실행된 타이틀 이름에서 오른쪽 마우스를 눌러 “Detached” 메뉴를 선택한다.

## 7.5 Registers

디버깅 시 타겟 시스템의 레지스터의 정보를 확인할 수 있다.

이 기능을 사용하려면 LemonIDE 타이틀 메뉴의 “Windows” → “Show View” → “Registers” 을 선택한다. Registers 창의 생성은 LemonIDE 우측위 Make View 에 추가된다.

Register 창을 LemonIDE Make View 안에서가 아닌 외부 독립창으로 실행하려면 “Registers” 가 실행된 타이틀 이름에서 오른쪽 마우스를 눌러 “Detached” 메뉴를 선택한다.

## 7.6 Workspace 변경

Workspace 는 타겟을 구동하기 위한 펌웨어를 제작할 목적으로 만든 최상위 디렉토리를 말하며, 다수의 프로젝트를 포함하고 있다. 프로젝트는 Target 에서 구동되는 개별 어플리케이션을 위한 디렉토리라고 정의할 수 있다. Workspace 변경은 LemonIDE 의 타이틀 메뉴상에 “file” → “Switch Workspace” 를 선택하여, 다른 Workspace 로 변경할 수 있다.