LemonIDE

User Guide

Ver 1.1

2013. 03 27

Serial Communication Experts
SystemBase
Since 1987

Revision History

Date	Version	Page	Description
2007. 8. 31	1.0	All	
2007.11. 29	1.0a	All	
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
2013.3.27	1.1	All	Updated New Firmware

Table of Contents

Chapter 1. Introduction

This chapter is an introduction to LemonIDE, Integrated Development Environment, by SystemBase.

1.1 About this manual

This manual includes procedures, functions, and usages of LemonIDE, an IDE developed by SystemBase so that programmers can easily write applications and execute them on the target board.

1.2 Reader

This manual is designed for developers who wish to write applications using LemonIDE. It is strongly recommended that anyone trying to apply or configure LemonIDE to read this document. This manual includes various functions, development process, and important notes on using LemonIDE. It is a great starting point for any programmer who wants to execute custom applications on connected devices.

Contents

Chapter 1. Introduction is a preface with general information and introductory notices.

Chapter 2. Getting Started explains about features and structure of LemonIDE.

Chapter 3. Installing in Windows provides information on first-time installation of LemonIDE Windows and host configuration.

Chapter 4. Using IDE describes each menu and its function in LemonIDE.

Chapter 5. LemonIDE Debugger explains how to debug remote applications running on the target using LemonIDE debugger.

Chapter 6. Monitoring Tool explains how to monitor the target status in real-time with LemonIDE.

1.3 LemonIDE related Documents

The following shows the documents related to LemonIDE.

Name	Description
User Guide	LemonIDE configuration and management
LemonIDE Spec Sheet	Technical specification of LemonIDE

To access more information about LemonIDE, please refer to the SystemBase website at http://www.sysbas.com or developer's community at http://www.embeddedmodule.com
From these websites, technical documents or latest updates are available for download.

Note
All documents are updated promptly at the homepage or community website. The contents in these documents are subject to change without a prior notice.

1.4 Technical Support

You can reach our technical support by following methods:

1. Visit developer's community at http://www.embeddedmodule.com and share information and tips about LemonIDE with developers all around the world.
2. Visit us at http://www.solvline.com and go to "Technical Support" menu or "Technical Product Info." where you can submit an inquiry or review FAQs.
3. Email our technical support team at tech@sysbas.com. Any kind of inquiries, requests, and comments are welcomed.
4. Call our customer center at +82-2-855-0501 for immediate support. (Mon. – Fri., 09:00 – 18:00 KST)

Chapter 2. Getting Started

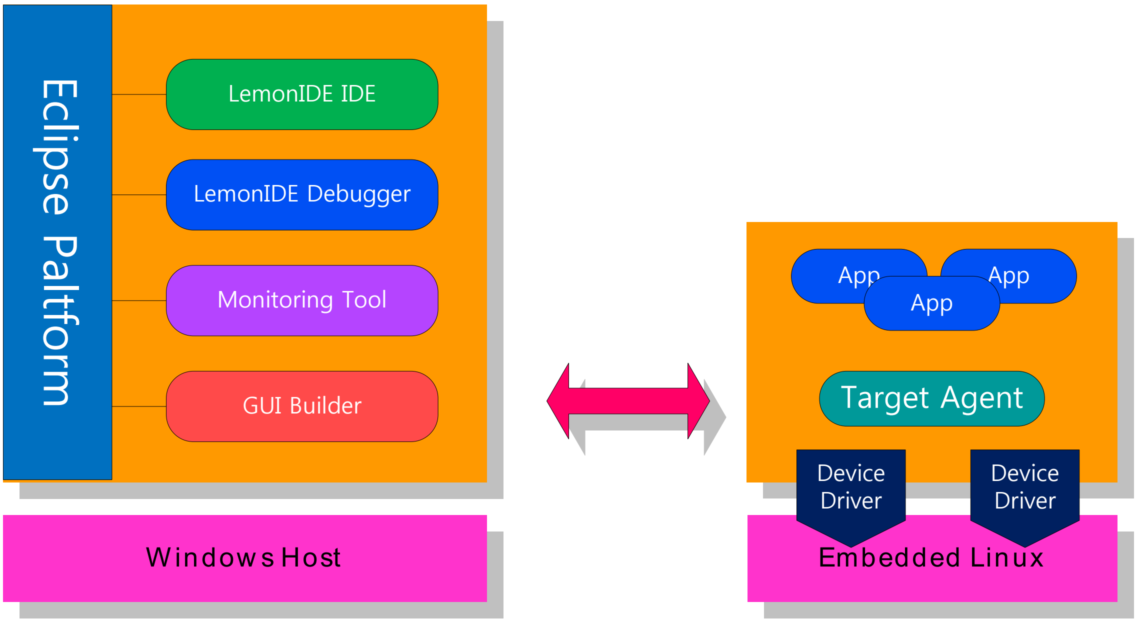
This chapter includes general information, main features, and applications of LemonIDE.

2.1 Overview

LemonIDE is an integrated develoment environment (IDE) for embedded software development, based on a world-famous open source project Eclipse. It includes GUI (Graphical User Interface) environment which helps writing applications or firmware running on an Embedded Linux. It also eliminates inconvenience from TUI (Text User Interface) with integrated structure linking GNU C/C++ compiler, source code editor, and debugger to one interface. This means programmers do not need to acclimate compiler or debugger commands. With a unified GUI, all development process can be handled by mouse clicks. In addition, Makefile auto-creation, source auto-completion, remote debugging, plugin support, and target system monitoring features are provided.

2.2 Structure

From Windows, users can develop embedded software running on the target system based on Eddy platform in a more convenient and accelerated manner. Tools included in LemonIDE are as follows.



LemonIDE IDE: Project-based management with C/C++ source program editing, cross-compile, remote execution, and GUI builder integration.

LemonIDE Debugger: A built-in debugger in LemonIDE supporting both breakpoint and tracepoint debugging.

Monitoring Tool: Memory, process, resource, and battery usage of the target system displayed visually.

Target Agent: Running on the target, the agent is connected to the host to transfer file and accept remote execution command.

2.3 Specification and Features

Main features of LemonIDE are as follows.

Eclipse

Eclipse SDK 3.2.2

CDT core 3.1.2

CVS core 3.2.2

Compiler

GNU C/C++ 4.4.7

ARM-Linux cross compiler

Editor

C/C++ source code editor

C/C++ code auto-completion

Syntax highlighting

Makefile auto-creation

Source browser

Multiple file editing with tab interface

File search and advanced search

External editor link for various file types

Detailed information on error syntax

Auto Build

Remote execution

Debugger

GNU gdb 7.4.1

Visual source code debugger

Breakpoint and tracepoint-based debugging

Remote debugging

Multi-thread debugging

Target Monitor Tool

Targetview Plug-in 1.0.0

Monitor Plug-in 1.3.4

Chapter 3. WInstalling in Windows

This chapter describes how to install LemonIDE on the Windows host system.

3.1 Install Environment

3.1.1 Windows Host System

LemonIDE has been tested on the following Windows versions.

Windows XP SP3 32bit

Windows 2000

Windows 2003

Windows 7 32 bit

Note
Instructions in this manual are based on LemonIDE installed in Windows 7 (32bit) and XP (SP3 32bit).

3.1.2 Target System

In order to download and execute the application created from LemonIDE, a target system is required. SystemBase embedded modules applicable as target systems are as follows.

Eddy Series	(Ver 2.xx)
Eddy-DK	(Ver 2.xx)

Refer to Eddy-DK Programmer's Guide for host-target connections and operation tests

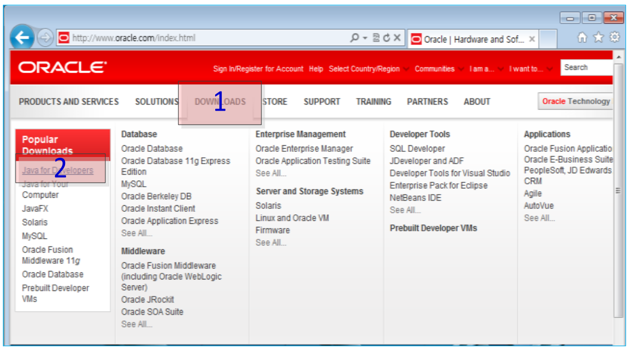
3.2 Installation

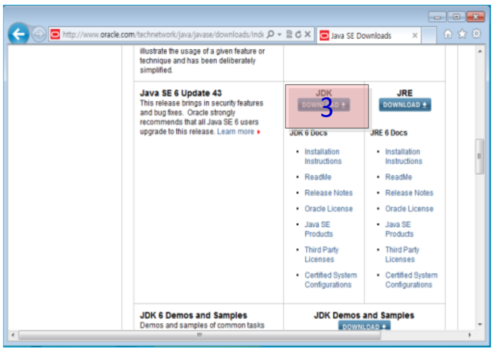
This chapter describes how to install LemonIDE based in Windows 7 (32bit).

When you install Eddy SDK, LemonIDE will be installed automatically, but since it is Eclipse based, installation of JDK is required.

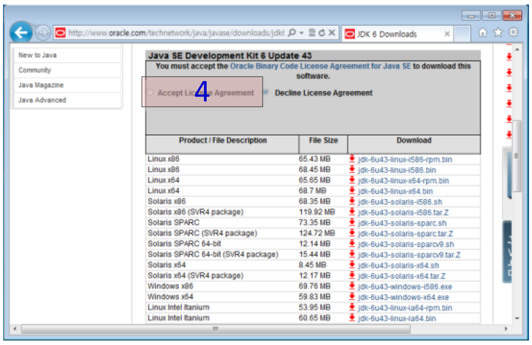
To run LemonIDE, Java Runtime Environment must be pre-installed and it should be Java 5 JRE (Java Runtime Environment) or higher version. JDK license does not allow distribution of JDK so the users must download it from the official website. The following shows how to download JDK 6.0 and install it.

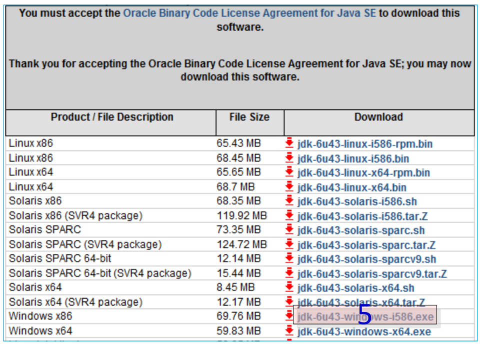
Go to http://www.oracle.com/ and hover your mouse pointer over area 1 shown below to see the extended window. Click on Java for Developers shown as are 2 to see the Java SE Download page. In the middle of the page, you can see the link where you can download the latest Java 6 series. Click the link for Java 6.





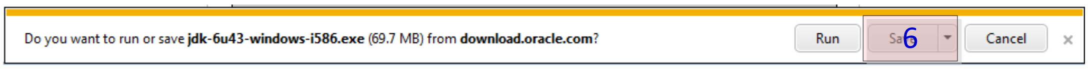
After clicking the link, based on the OS, there are various Java 6 installation files listed.Click on Accpt License Agreement in area 4 and select Windows x86 installation file below. If you are using Internet Explorer 9, you will see a download window. Click are 6 to save the file and start downloading.


JDK 6 Downloads
Java SE Development Kit 6 Update 43
Accept License Agreement
4
Decline License Agreement


You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

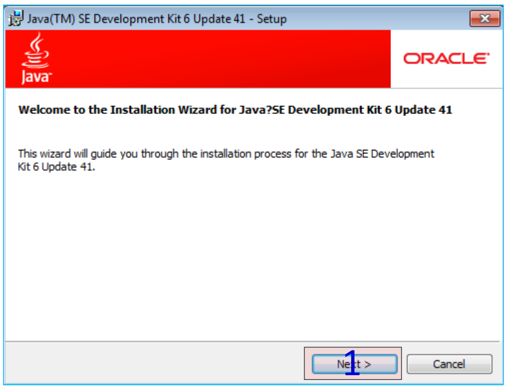
Product / File Description	File Size	Download
Linux x86	65.43 MB	jdk-6u43-linux-i586-rpm.bin
Linux x86	68.45 MB	jdk-6u43-linux-i586.bin
Linux x64	65.65 MB	jdk-6u43-linux-x64-rpm.bin
Linux x64	68.7 MB	jdk-6u43-linux-x64.bin
Solaris x86	68.35 MB	jdk-6u43-solaris-i586.sh
Solaris x86 (SVR4 package)	119.92 MB	jdk-6u43-solaris-i586.tar.Z
Solaris SPARC	73.35 MB	jdk-6u43-solaris-sparc.sh
Solaris SPARC (SVR4 package)	124.72 MB	jdk-6u43-solaris-sparc.tar.Z
Solaris SPARC 64-bit	12.14 MB	jdk-6u43-solaris-sparcv9.sh
Solaris SPARC 64-bit (SVR4 package)	15.44 MB	jdk-6u43-solaris-sparcv9.tar.Z
Solaris x64	8.45 MB	jdk-6u43-solaris-x64.sh
Solaris x64 (SVR4 package)	12.17 MB	jdk-6u43-solaris-x64.tar.Z
Windows x86	69.76 MB	jdk-6u43-windows-i586.exe
Windows x64	59.83 MB	jdk-6u43-windows-x64.exe


Do you want to run or save **jdk-6u43-windows-i586.exe** (69.7 MB) from **download.oracle.com**?

Check for few things below when you are downloading.

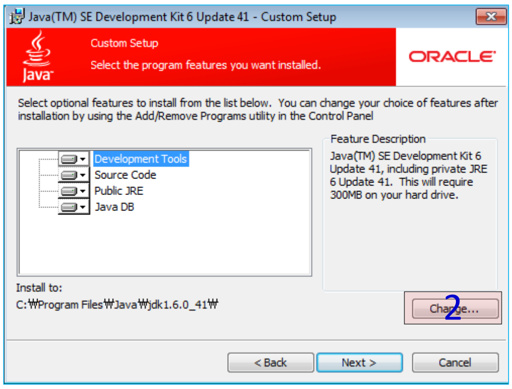
- JDK 7 is not supported so download the latest JDK 6series.
- x64 is not supported yet. Please install x86(i586) version.
- Windows 64 bit series are not supported yet.

Install JDK that you downloaded. Installing order is as shown below.


Java(TM) SE Development Kit 6 Update 41 - Setup

Welcome to the Installation Wizard for Java?SE Development Kit 6 Update 41

This wizard will guide you through the installation process for the Java SE Development Kit 6 Update 41.

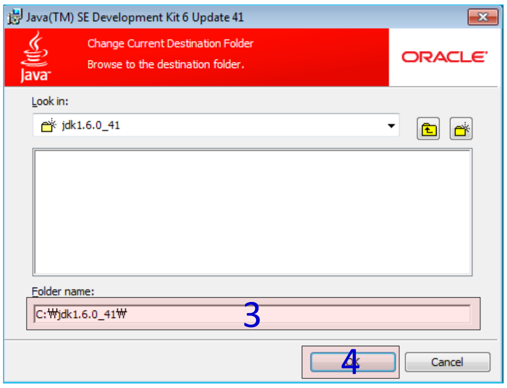

Java(TM) SE Development Kit 6 Update 41 - Custom Setup

Custom Setup
Select the program features you want installed.

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

Feature Description
Java(TM) SE Development Kit 6 Update 41, including private JRE 6 Update 41. This will require 300MB on your hard drive.

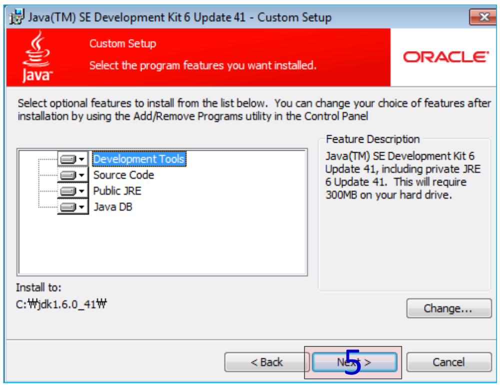
Install to:
C:₩Program Files₩Java₩jdk1.6.0_41₩


Java(TM) SE Development Kit 6 Update 41

Change Current Destination Folder
Browse to the destination folder.

Look in:
jdk1.6.0_41

Folder name:
C:₩jdk1.6.0_41₩


Java(TM) SE Development Kit 6 Update 41 - Custom Setup

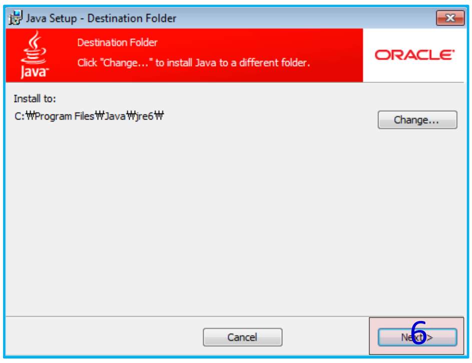
Custom Setup
Select the program features you want installed.

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

Feature Description
Java(TM) SE Development Kit 6 Update 41, including private JRE 6 Update 41. This will require 300MB on your hard drive.

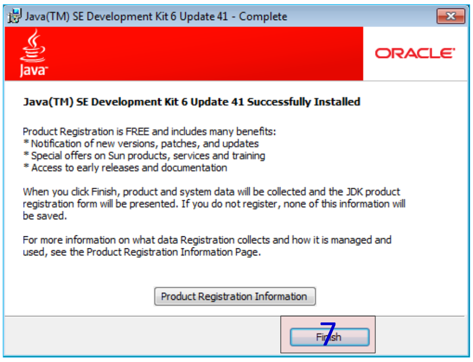
Install to:
C:₩jdk1.6.0_41₩

Follow the numbers in the screenshot. Choose "C:\" when it let's you choose which directory to be installed. Therefore, content of 3 will be "c:\jdk1.6.0_41". After clicking on 5 shown above, files will be installed. Click "Next" from 6 and "Finish" from 7 to complete the installation of SDK.


Java Setup - Destination Folder

Destination Folder
Click "Change..." to install Java to a different folder.

Install to:
C:₩Program Files₩Java₩jre6₩


Java(TM) SE Development Kit 6 Update 41 - Complete

Java(TM) SE Development Kit 6 Update 41 Successfully Installed

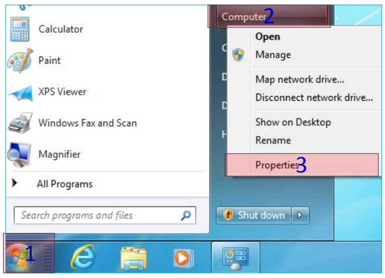
Product Registration is FREE and includes many benefits:
- Notification of new versions, patches, and updates
- Special offers on Sun products, services and training
- Access to early releases and documentation

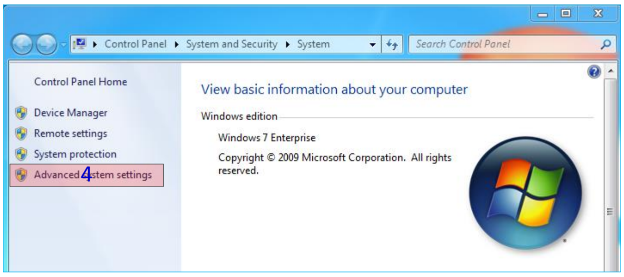
When you click Finish, product and system data will be collected and the JDK product registration form will be presented. If you do not register, none of this information will be saved.

For more information on what data Registration collects and how it is managed and used, see the Product Registration Information Page.

3.2.1 Configure Windows Environment Variables

Path must be set for LemonIDE to reference Java Libraries.



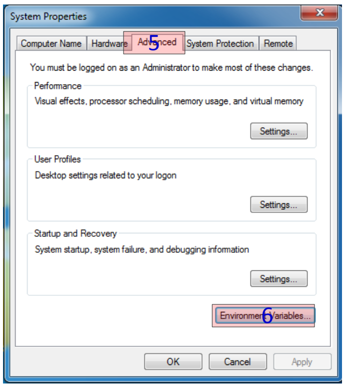


View basic information about your computer

Windows edition

Windows 7 Enterprise

Copyright © 2009 Microsoft Corporation. All rights reserved.



You must be logged on as an Administrator to make most of these changes.

Performance

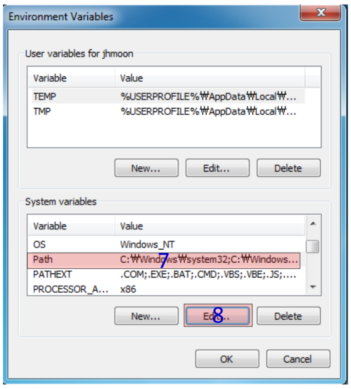
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your logon

Startup and Recovery

System startup, system failure, and debugging information

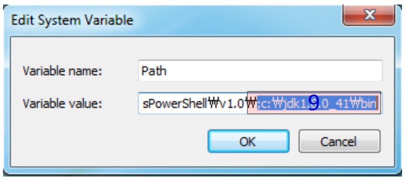


User variables for jhmoon

Variable	Value
TEMP	%USERPROFILE%₩AppData₩Local₩...
TMP	%USERPROFILE%₩AppData₩Local₩...

System variables

Variable	Value
OS	Windows_NT
Path	C:₩Windows₩system32;C:₩Windows...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86



Variable name: Path

Variable value: sPowerShell₩v1.0₩;c:₩jdk1.6.0_41₩bin

For Windows 7, refer to above order. For Windows XP, select Windows Desktop → My Computer → Right mouse click → Property → Advance Tab → Environment Variables and from System variables, select Path and click "Edit" to add the following. For Windows 7, area 9 from above is where the following content should be added.

```
;c:₩jdk1.<Version>₩bin
```

Version is the version of JDK where JDK is installed.

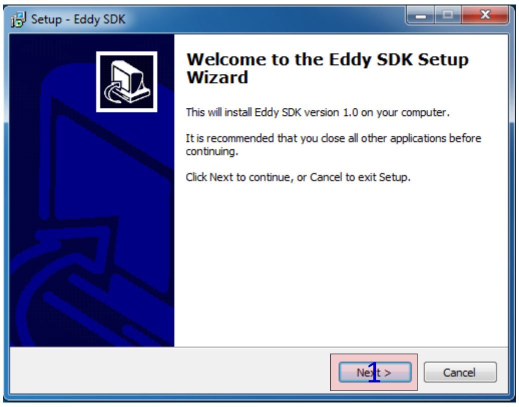
If JDK 6 is installed version under C:\ root, "Version" in the example should be "6.0_41" therefore, for the path, add ";c:\jdk1.6.0_41\bin".

3.2.2 Install LemonIDE

Install LemonIDE package. EddySDK.exe is installation file for LemonIDE package for Windows

(Note) This manual is written under assuming that LemonIDE is installed under C:\ in the hard drive. If you install it other than C:\ driver, the environment variable needs to be changed also so please install it under C:\ if possible.

Run installation file for Eddy SDK and follow the installation order shown below.

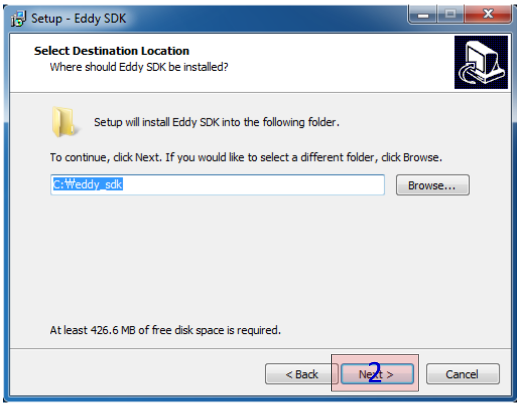

Setup - Eddy SDK

Welcome to the Eddy SDK Setup Wizard

This will install Eddy SDK version 1.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.


Setup - Eddy SDK

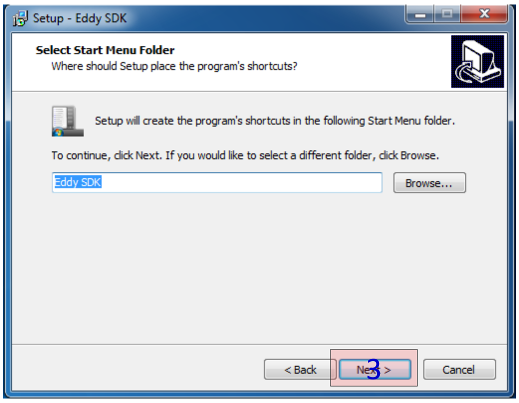
Select Destination Location
Where should Eddy SDK be installed?

Setup will install Eddy SDK into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:₩eddy_sdk

At least 426.6 MB of free disk space is required.

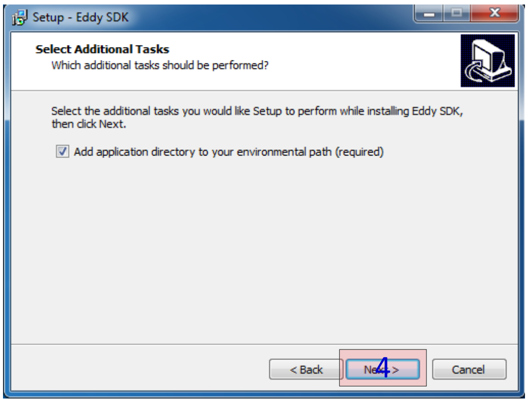

Setup - Eddy SDK

Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

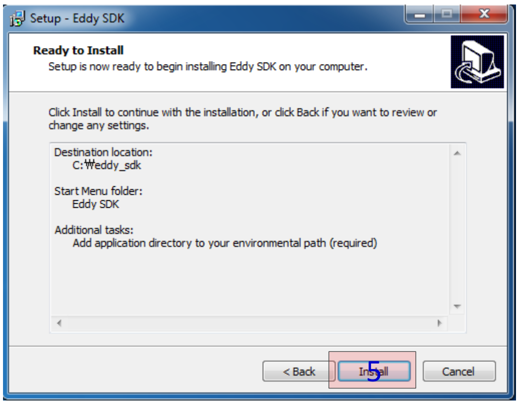
Eddy SDK


Setup - Eddy SDK

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Eddy SDK, then click Next.

☑ Add application directory to your environmental path (required)


Setup - Eddy SDK

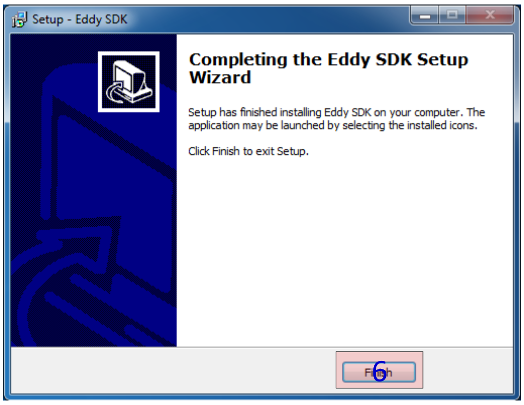
Ready to Install
Setup is now ready to begin installing Eddy SDK on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:₩eddy_sdk

Start Menu folder:
Eddy SDK

Additional tasks:
Add application directory to your environmental path (required)


Setup - Eddy SDK

Completing the Eddy SDK Setup Wizard

Setup has finished installing Eddy SDK on your computer. The application may be launched by selecting the installed icons.

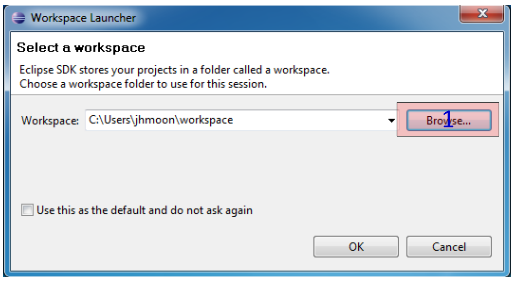
Click Finish to exit Setup.

When installation is complete, clicl “Finish” to exit.

3.3 Creating Project Space

After installation is complete, go to the installed directory and run lemonide.exe. Or you can double click on LemonIDE icon generated in the Desktop after installation is complete. After showing Lemonide logo and few momenots of loading, when "Workspace Launcher" window appears, it is ready to be used.

3.3.1 Create Workspace

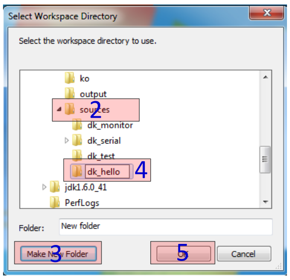

Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\jhmoon\workspace

□ Use this as the default and do not ask again

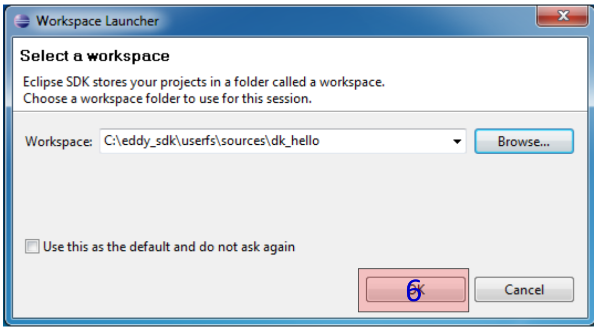

Select Workspace Directory

Select the workspace directory to use.

Folder: New folder

When you run LemonIDE, a window will pop-up where you can create a workspace.

The order above shows creating folder in "C:\eddy_sdk\userfs\sources" folder and creating workspace, which is recommended. In the example, "dk_hello" folder is created and used as the path. After setting the path for workspace location is completed, the path is shown as below. Click area 6 to complete it.


Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\eddy_sdk\userfs\sources\dk_hello

□ Use this as the default and do not ask again

You can create workspace at temporary location, but when importing previously used files and checking the content of Makefile, the location needs to be in the Makefile.

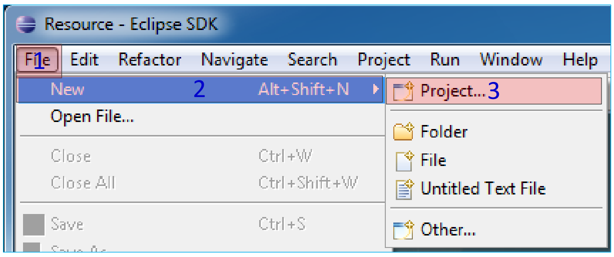
When you use "C:\eddy_sdk\userfs\sources" as recommended from above, no additional task is needed.

(Note) Workspace should not have a path with blank space in it. For example, "C:\documents and settings" is not accepted.

3.3.2 Create Project

When creating a workspace is completed, create a project you wish to work on.

Click in order of File → New → Project

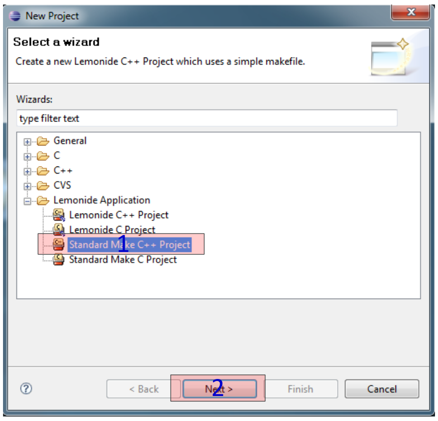


Project is a space where you can code, build, execute and debug with LemonIDE.

Whether Makefile is created or not or use of C++, LemonIDE project fall under 4 categories shown as below.

Standard Make C++ Project Standard Make C Project	Project for C/C++ with pre-written Makefile
Managed Make C++ Project Managed Make C Project	Project for C/C++ with automatically generated Makefile

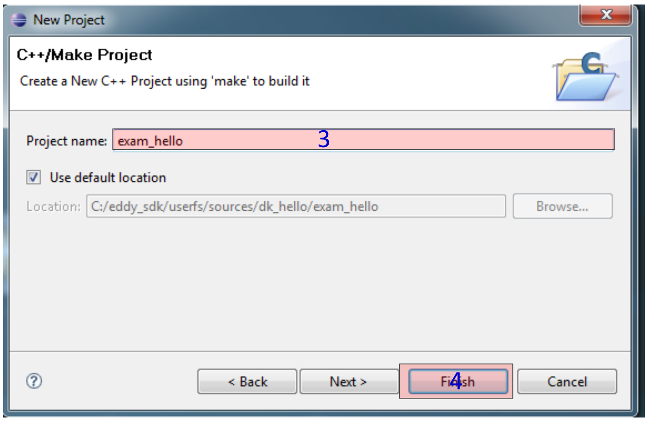
(Note)
Generally, certain libraries are linked many times and most of open source provides Makefile therefore, Standard type is recommended.
DK source that is provided also defines Makefile so Standard type, easier for customizing, will be based when we explain about it in here.


New Project

Select a wizard

Create a new Lemonide C++ Project which uses a simple makefile.

Wizards: type filter text


New Project

C++/Make Project

Create a New C++ Project using 'make' to build it

Project name: exam_hello

☑ Use default location

Location: C:/eddy_sdk/userfs/sources/dk_hello/exam_hello

Select Standard Make C++ Project and click “Next”.

You can name the project in this step. The name “exam_hello” is used in this example.

You can click “Next” to select various options but in this example, we will conclude at this point by clicking “Finish” button.

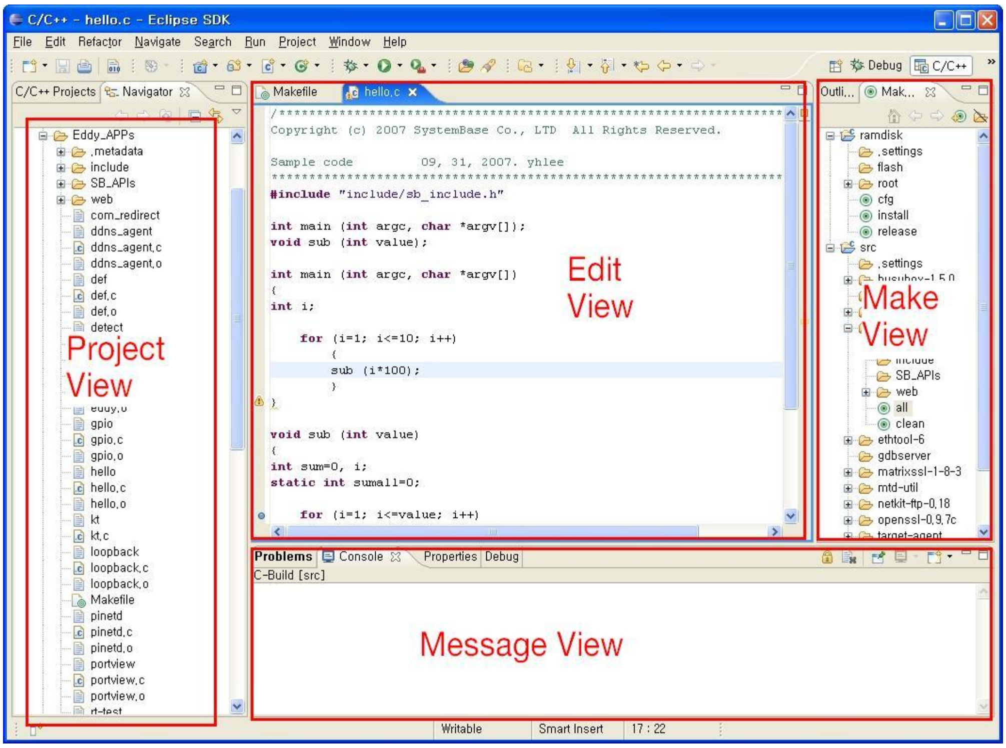
Chapter 4. Using IDE

In this chapter, features in LemonIDE are explained.
We will create, build and execute a source file named hello.c in Eddy. Procedures for making a firmware image will be outlined. Example shown here is done under Windows XP. Configuration for Windows 7 is the same.

4.1 IDE Screnn Configuration

LemonIDE IDE (Integrated Development Environment) is based on Eclipse and supports C/C++ source editing, cross compile, remote execution, and GUI builder integration. LemonIDE is a plugin of Eclipse and uses of Eclipse UI (workbench) as a basic layout. This chapter focuses primarily on LemonIDE-specific characteristics rather than general Eclipse UI context.

LemonIDE consists of following views



```
/*******************************************************************
Copyright (c) 2007 SystemBase Co., LTD  All Rights Reserved.

Sample code         09, 31, 2007. yhlee
********************************************************************
#include "include/sb_include.h"

int main (int argc, char *argv[]);
void sub (int value);

int main (int argc, char *argv[])
{
int i;

    for (i=1; i<=10; i++)
        {
        sub (i*100);
        }
}

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

    for (i=1; i<=value; i++)
```

Project View Displays project lists included in Workspace

Make View	Editing source code, headers, functions, and structures in class building supported.
Edit View	Window for editing source code. Multiple tabs can be opened at the same time.
Message View	Display messages during project buid and Target Brower supported by LemonIDE

4.2 Create and Edit Source Code

Create a new source file or open an existing source file.
There are two cases in adding source files to a project: creating new files or importing existing files.
When editing an existing file, double click on the file to edit on Project View and the file will be opened in Edit View.
To create a new source file, select "File" → "New" → "Source File" and assign proper saving location and name for the new file. The created new file can be seen in the Project View and will be opened in Edit View for editing

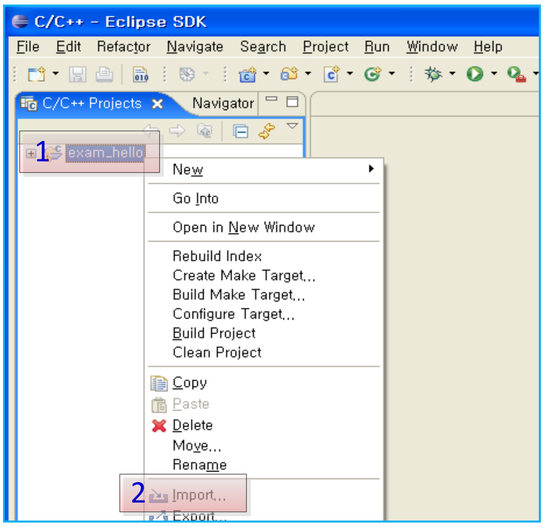
4.2.1 Import Source code

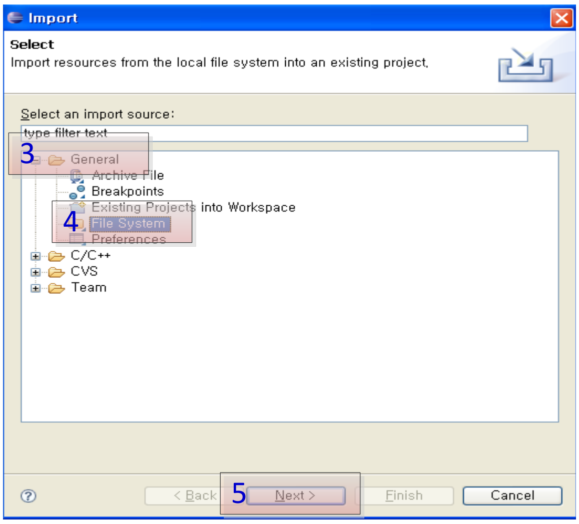
With various source codes and libraries, users can use them to apply into the application program that they wish to write. In this section, it shows how to import examples when writing a application program.

The left example from below shows that project named "exam_hello" is created.

If you click on the right mouse button at the exam_hello project, few menus will be display. Select import menu from them.

The right example from below shows different sources that you can select from. Select General → File System the, click "Next".

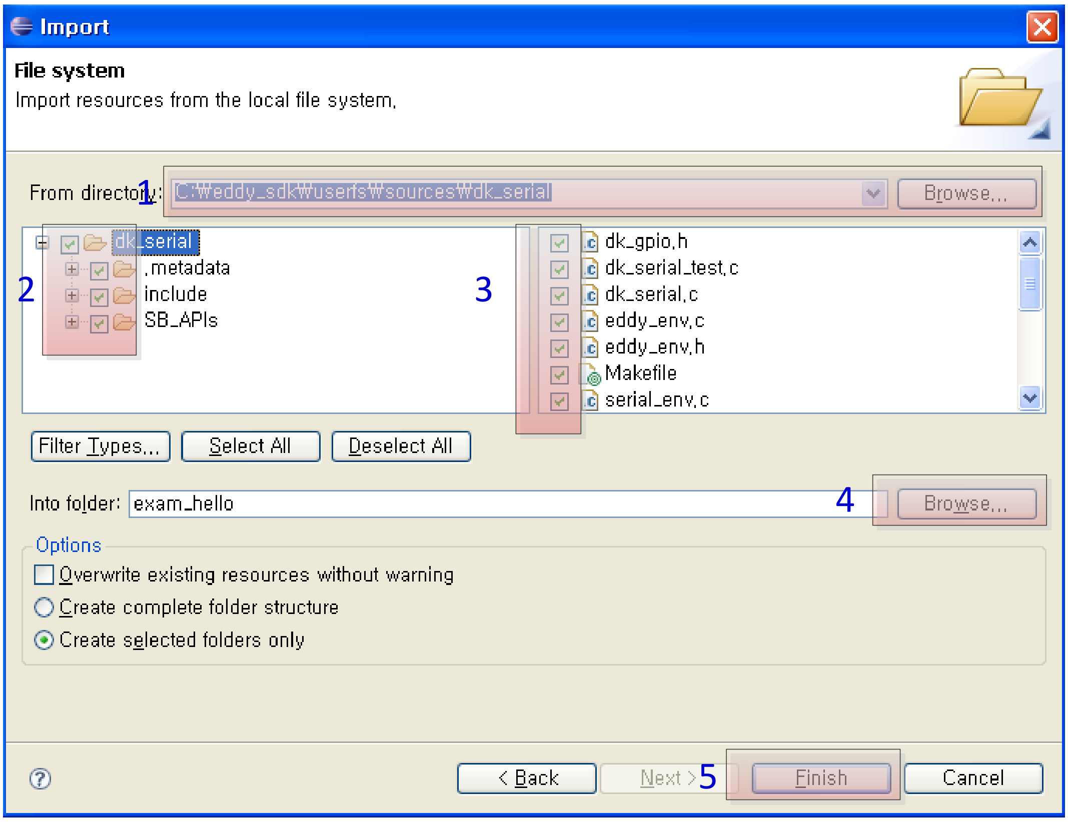




Import resources from the local file system into an existing project.

Next, from below, you can select which directory that you can retrieve files from.

Select "Browser" from the top right button, and select C:\eddy_sdk\userfs\sources\dk_serial folder. Depending on the features that you want to add, you can select dk_test or dk_monitor sources from the directory. You do not have to import the files that you do not need, so please select the ones you need. Click "Finish" when you are done selecting to complete.



Import

File system

Import resources from the local file system.

From directory: C:₩eddy_sdk₩users₩sources₩dk_serial

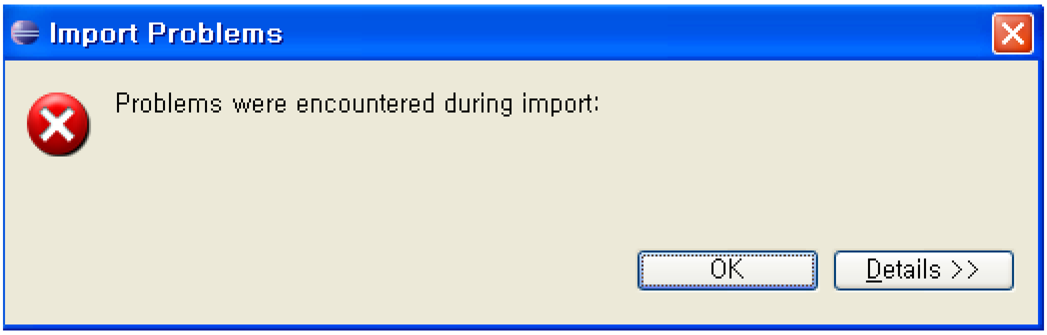
☑ dk_serial
- ☑ .metadata
- ☑ include
- ☑ SB_APIs

☑ dk_gpio.h
☑ dk_serial_test.c
☑ dk_serial.c
☑ eddy_env.c
☑ eddy_env.h
☑ Makefile
☑ serial_env.c

Into folder: exam_hello

Options
- □ Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

When importing is not successful, you will see a message shown as below. Change the workspace when this happens. This message appears when the location of the files are the same as where they will be saved. Careful not to create a workspace in the same directory where files to be imported are located.

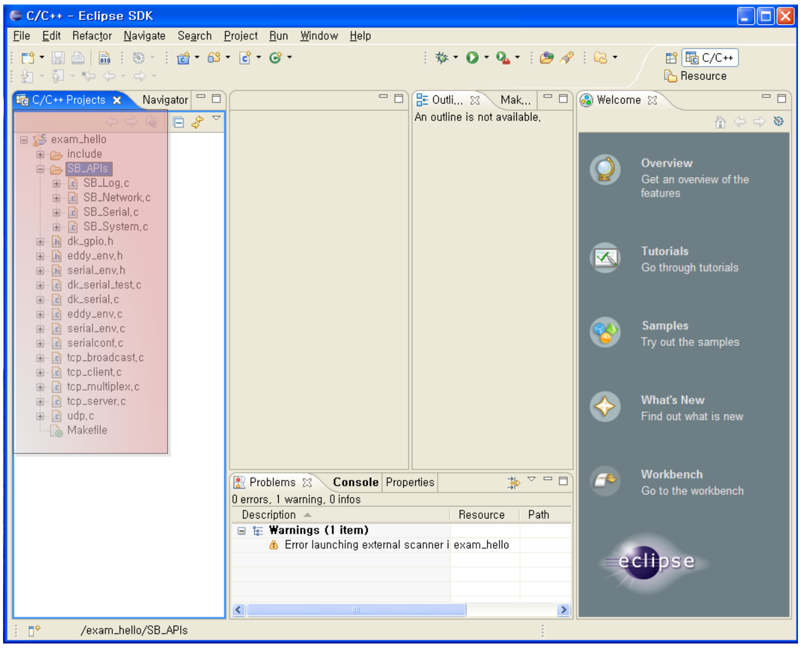


Import Problems

Problems were encountered during import:

When importing is successful, as shown below, you will see the imported files from Project View.

Let's create and edit these source codes.



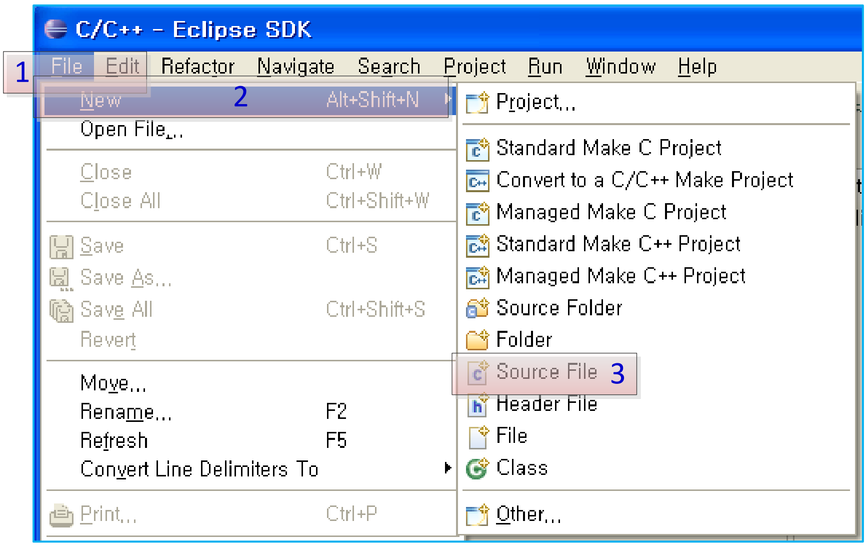
0 errors, 1 warning, 0 infos

Description	Resource	Path
Warnings (1 item)		
Error launching external scanner i	exam_hello	

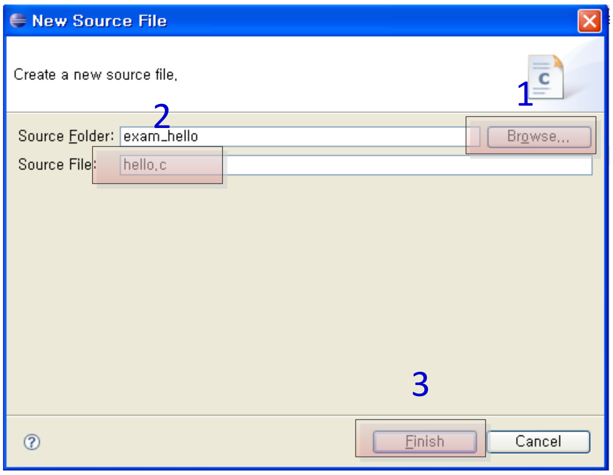
4.2.2 Create Source code

Image below is an example regarding writing hello.c to help you understand about the use of LemonIDE.

Select "File" -> "New" -> "Source File".



As shown in the left image below, when you give a source file name and the source folder and click “Finish”, hello.c is added to the project as shown in the right image below.

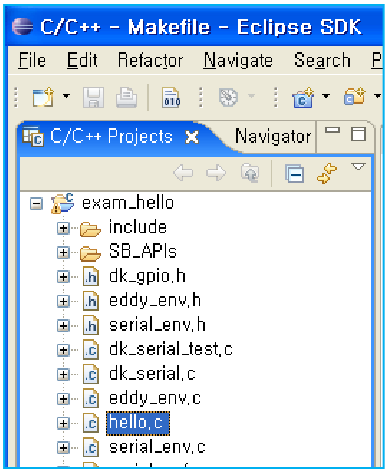


New Source File

Create a new source file.

Source Folder: exam_hello

Source File: hello.c



C/C++ - Makefile - Eclipse SDK
File Edit Refactor Navigate Search P
C/C++ Projects
Navigator

```
exam_hello
    include
    SB_APIs
    dk_gpio.h
    eddy_env.h
    serial_env.h
    dk_serial_test.c
    dk_serial.c
    eddy_env.c
    hello.c
    serial_env.c
```

Following image is an example to help you understand the use of LemonIDE.

```
#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);

}
```

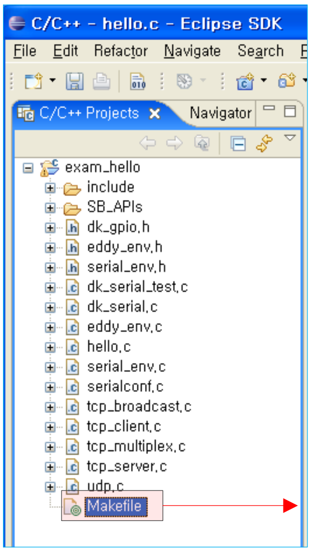
4.3 Compile Environment Settings & Build

Compile is a step where compiling, linking and building a binary image so that it will be able to execute from Eddy.

Compile in LemonIDE projects are basically done by makefile. In this chapter, compiling a source code file utilizing Makefile is introduced.

4.3.1 Modify Makefile

If a new source code file was created or if a previous source code was modified in the Project View, Makefile needs to be written to compile the source code. Following image is an example for adding compile settings for a newly created hello.c file in Makefile included in “exam_hello”.



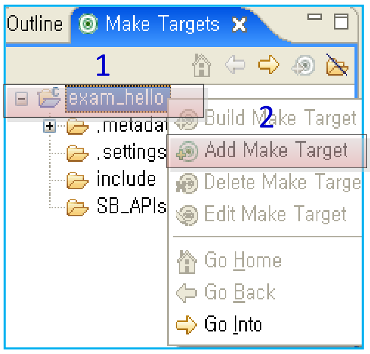
```
CC = arm-linux-gcc
CFLAGS = -Os

DK_SERIAL_OBJS = dk_serial.o
DK_SERIAL_OBJS += eddy_env.o
DK_SERIAL_OBJS += serial_env.o
DK_SERIAL_OBJS += tcp_server.o
DK_SERIAL_OBJS += tcp_client.o
DK_SERIAL_OBJS += tcp_multiplex.o
DK_SERIAL_OBJS += tcp_broadcast.o
DK_SERIAL_OBJS += udp.o
DK_API_OBJS    += SB_APIs/SB_Serial.o
DK_API_OBJS    += SB_APIs/SB_System.o
DK_API_OBJS    += SB_APIs/SB_Network.o
DK_API_OBJS    += SB_APIs/SB_Log.o

TARGETS = dk_serial serialconf dk_serial_test hello

all : $(TARGETS)
        @echo done

clean :
        @rm -f *.o $(DK_API_OBJS)
        @rm -f $(TARGETS)



install : $(TARGETS)
        @cp $(TARGETS) ../../dk_default/usr/bin/

dk_serial : $(DK_SERIAL_OBJS) $(DK_API_OBJS)

serialconf : serialconf.o serial_env.o eddy_env.o

dk_serial_test : dk_serial_test.o $(DK_API_OBJS)

hello : hello.o
```

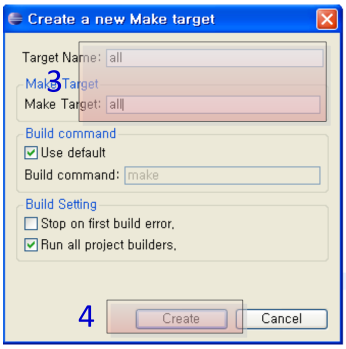
If you double-click Makefile in the Project View from the left image above, you can edit Makefile. Add codes in the red from the right image above and save the progress by clicking the Save button from the Tools.

4.3.2 Add Make Target

When you wrote the Makefile, it should be added to LemonIDE through Make Tearget so that it can be compiled.

Make Target is located in Make View, right side in LemonIDE. To add Make Target, select the location of source code and Makefile, and click right mouse button to select "Add Make Target".





Target Name: all

Make Target

Make Target: all

Build command

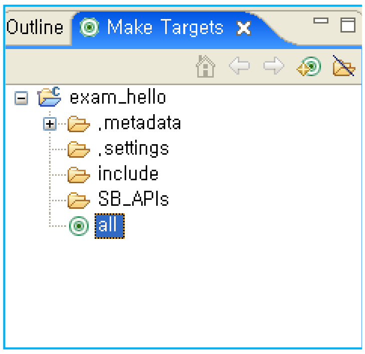
☑ Use default

Build command: make

Build Setting

☐ Stop on first build error.

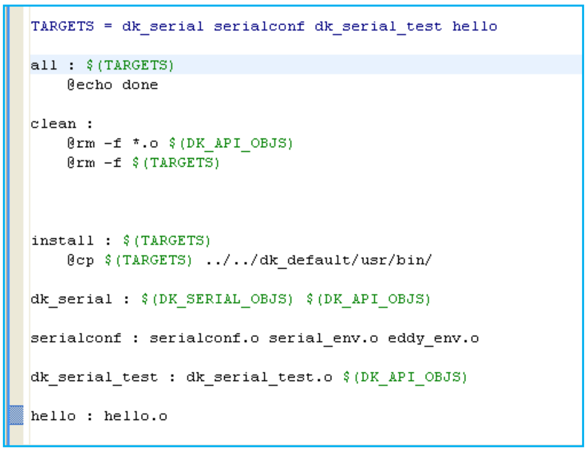
☑ Run all project builders.



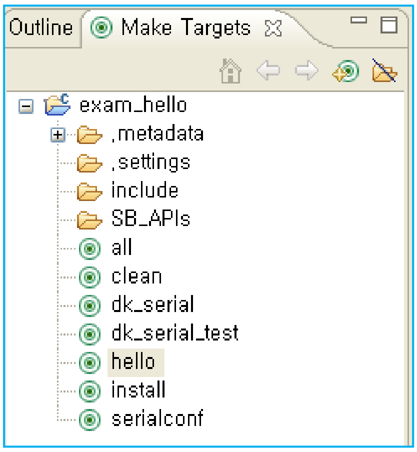
New source code creaded from "4.2 Create & Edit Source Code" is "Hello.c" and from "4.3.1 Modify Makefile", compile settings were added therefore you can add hello.c compile settings as shown above. The right image shows that Hello is added in Make Targets.

Target Name: Specify alias of target source code needs to be compiled

Make Target: argument value to pass when Make is executed (Compile settings added to Makefile)



```
TARGETS = dk_serial serialconf dk_serial_test hello

all : $(TARGETS)
    @echo done

clean :
    @rm -f *.o $(DK_API_OBJS)
    @rm -f $(TARGETS)


install : $(TARGETS)
    @cp $(TARGETS) ../../dk_default/usr/bin/

dk_serial : $(DK_SERIAL_OBJS) $(DK_API_OBJS)

serialconf : serialconf.o serial_env.o eddy_env.o

dk_serial_test : dk_serial_test.o $(DK_API_OBJS)

hello : hello.o
```

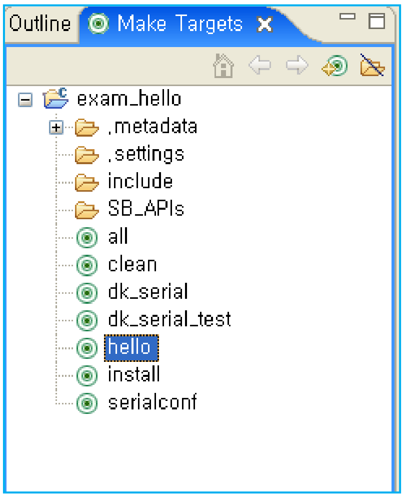


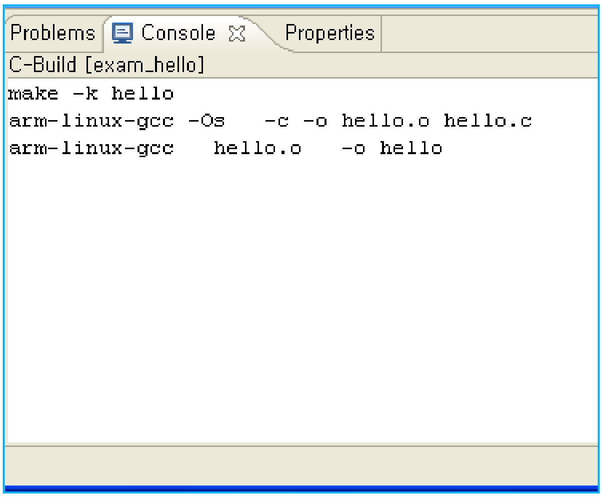
You can add compile settings in Makefile shown above in order as shown in the right image. By clicking added Make Target, you can execute it.

4.3.3 Source Code Compile

Compile the source code to executable binary for Eddy.

To compile, click Make Target that you created from "4.3.2 Add Make Target". (Double-click "Hello" from "Make View" screen.) You can check the progress at the bottom of LemonIDE screen under "Console" in "Message View". DK source provides make target which contains "all" and "clean". In "all", all application source code are registered to Makefile to be compiled. In "clean", make target with all the compiled application to be initialized. (See "Makefile".)

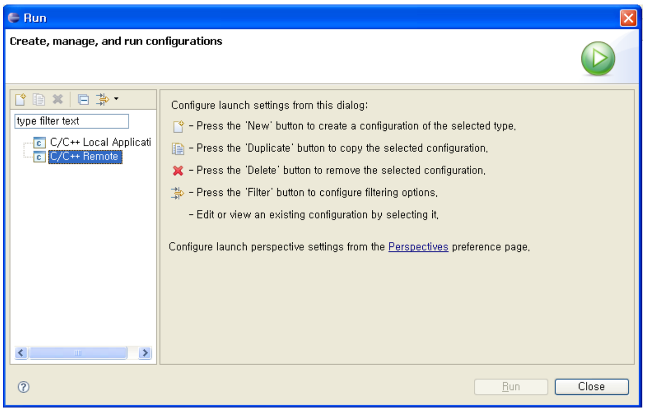




```
C-Build [exam_hello]
make -k hello
arm-linux-gcc -Os   -c -o hello.o hello.c
arm-linux-gcc   hello.o   -o hello
```

4.4 Register Run Method

In this section, you will learn how to make a Run/debug Method so that you can upload to Eddy and execute or debug a binary that you built.



Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

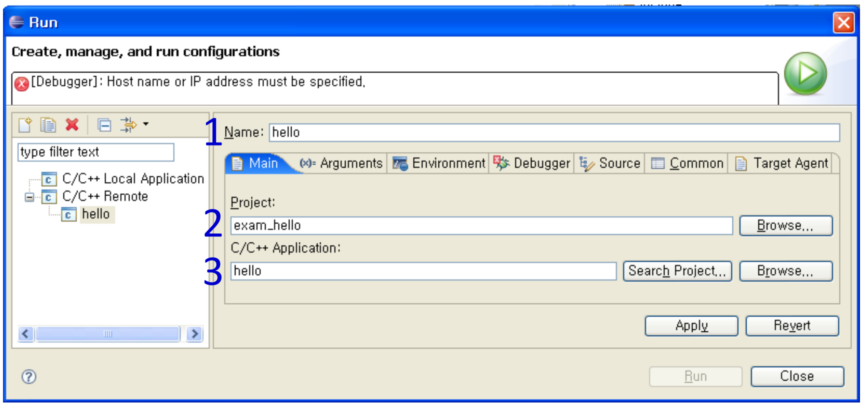
Configure launch perspective settings from the Perspectives preference page.

From the title menu in LemonIDE, when you select "Run" → "Run…" it will display a Create Method window as shown above.

Double click "C/C++ Remote" to create a Method where you can add execute settings.

4.4.1 Register Method Name and Binary File

Select "Main" tab to register names and location of executable binary.



[Debugger]: Host name or IP address must be specified.

Name: hello

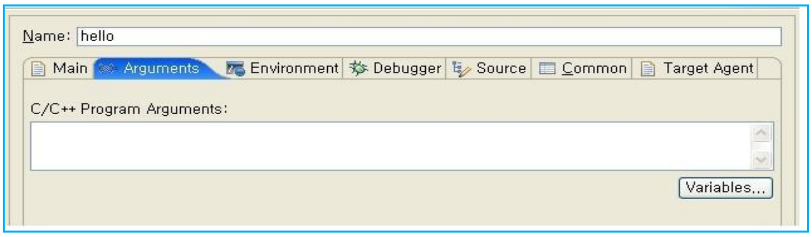
Project: exam_hello

C/C++ Application: hello

1. Name : Register alias of Method to be executed
2. Project : Select which project from the workspace the binary belongs to.
3. C/C++ Application : Select binary file that you wish to execute.

4.4.2 Configure Arguments

Select "Arguments" tab and add arguments if needed.



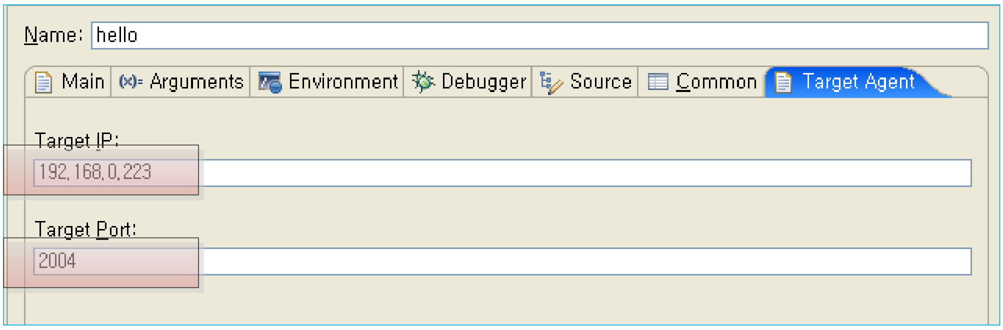
Name: hello

C/C++ Program Arguments:

4.4.3 Configure Target Agent

Select "Target Agent" tab and add IP address and port number for the device that you will run.

From the example, 192.168.0.223, the default target IP, is used. If necessary, the target IP can be changed with different address, but the port number should be fixed to 2004.



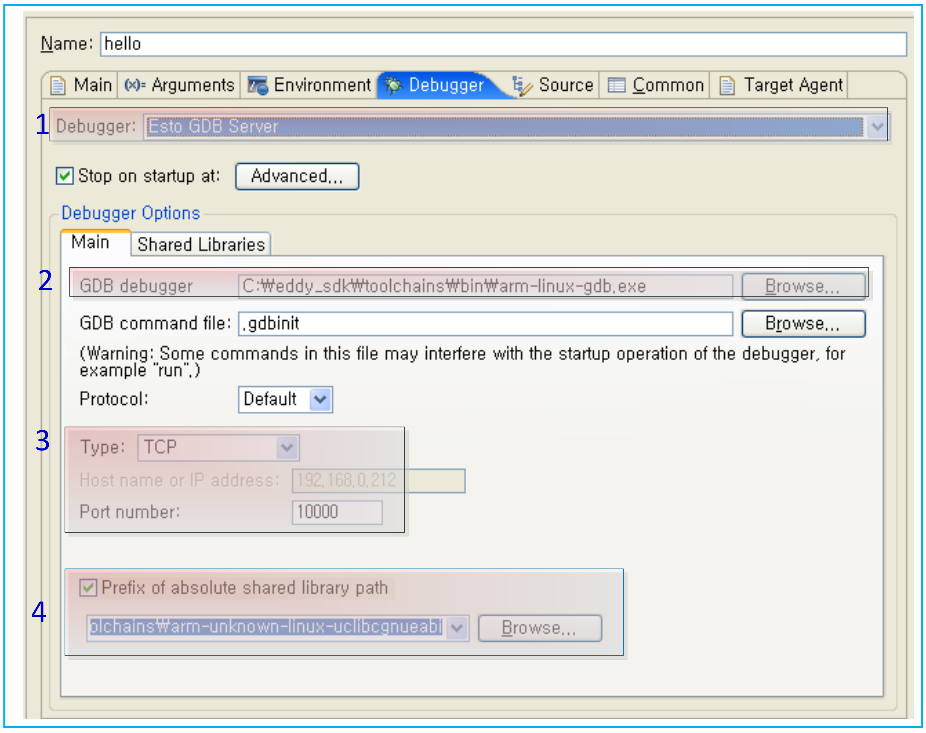
Name: hello

Target IP: 192.168.0.223

Target Port: 2004

4.4.4 Configure Debugger

Select "Debugger" tab and register debugger you wish to use in LemonIDE.



Name: hello

Debugger: Esto GDB Server

☑ Stop on startup at:

GDB debugger C:₩eddy_sdk₩toolchains₩bin₩arm-linux-gdb.exe

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

Protocol: Default

Type: TCP

Host name or IP address: 192.168.0.212

Port number: 10000

☑ Prefix of absolute shared library path

olchains₩arm-unknown-linux-uclibcgnueabi

1. Debugger :

Select the type of debugger.

To use the debugger for Eddy, select "**Esto GDB Server**" for Windows and "**LemonIDE GDB Server**" for Linux in LemonIDE.

2. GDB Debugger :

In the file location, add the file location where the debugger file is placed.

For Windows select [C:\eddy_sdk\toolchains\bin\arm-linux-gdb.exe] for LemonIDE.

3. Type :

Choose debugging information transmission style of LemonIDE and Eddy. This version only supports TCP. Select "TCP" and enter any value for the port number except "2004".

4. Prefix of absolute shared library path :

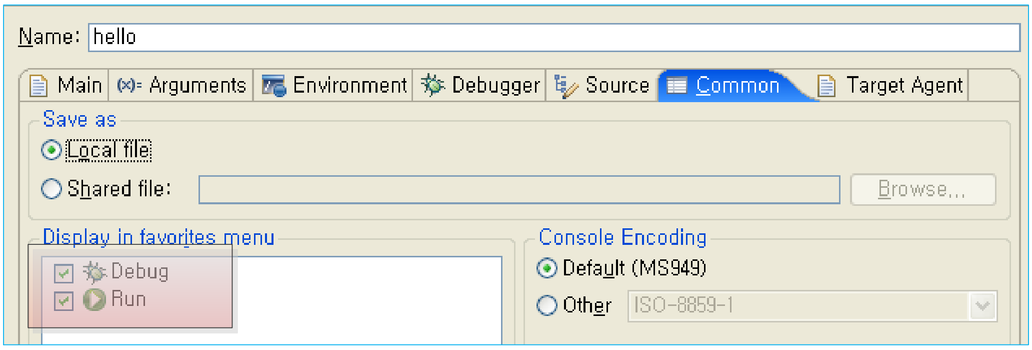
To debug, target root file system must be installed in the host system. /lib and /usr/lib libraries in root file system are required to debug. If target library is not presented at /lib in the host system, there can be problem while debugging.

If the library is present for Windows:

"C:\eddy_sdk\toolchains\arm-unknown-linux-uclibcgnueabi"

4.4.5 Configure Common

Select "Common" tab and choose if you would like to see debug, run, both results and none from "Display in favorites menu". Usually, both options are checked.



Name: hello

Save as
● Local file
○ Shared file:

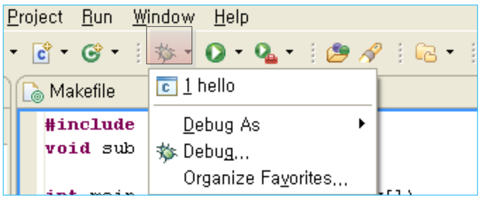
Display in favorites menu
☑ Debug
☑ Run

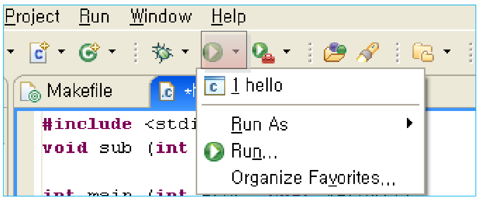
Console Encoding
● Default (MS949)
○ Other ISO-8859-1

When registering options for "Hello" method is all completed, click "Close" icon to finish the process.

4.4.6 Method Register Result

When registering all the process for the binary file that you wish to execute, select debug icon and run icon in the top menu from the LemonIDE to check the registered method.





If “hello” method is not registered as shown above, select “Run” → “External Tools” → “Organize Favoites” → “Add” to show the “hello” method.

4.5 Execute Run Method

In this section, procedures for uploading binary files which is built to Eddy will be introducted. There are two ways to upload and execute binary file to Eddy. First, upload through GDB Server in LemonIDE and execute or debug it easily. Second, use FTP to send it and check with telnet. For using FTP, please refer to “Eddy_DK_Programmer_Guide”. This manual explains about how to do it with GDB Server. If run method for execution environment for Hello.c was created in previous section, run method can be executed by clicking the Run icon under the title menu bar in LemonIDE.

4.5.1 Execute Target Agent in Eddy

The target system of LemonIDE is Eddy. A server program is required for LemonIDE so that it can respond to requests regarding Run or Debug.This is called the target agent, it should be configured to connect to “LemonIDE Target Agent” in Eddy.

The example below shows how to login to Eddy using telnet or SSH and operating the target agent.
/usr/bin/tae command is used to execute the target agent with “&” (ampersand) to make it run in background. If ampersand is not used, you cannot execute other commands until the target agent ends.

```
# /usr/bin/tae &
TargetAgent 1.1.9 started.
#
```

(Note) The default console port for Eddy is serial port. Therefore, all results run through LemonIDE is displayed from the console port in Eddy DK. If the target board is not an Eddy DK board, using serial port to check the result is not possible. In this case, use telnet as a console.

4.5.2 Execute Run Method (Check the result in serial port)

By default, target agent in Eddy prints out its results through serial console port in Eddy. Therefore, in order to verify the results printed out to serial console port in Eddy, an emulator program for a serial communication is required. Steps on “5.5.1 Configuring Eddy’s Target Agent” must be followed first for this method.

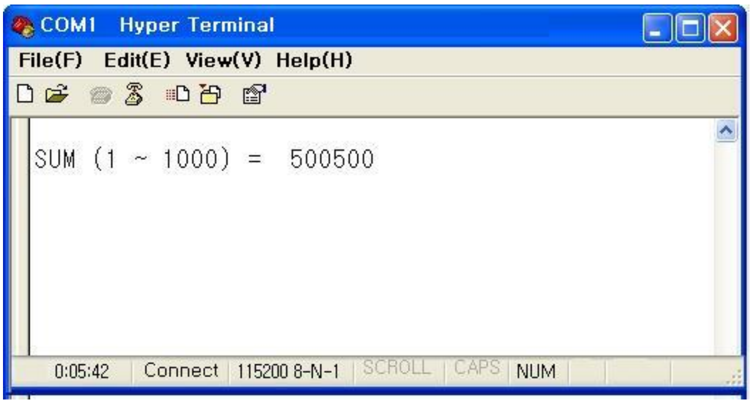
Eddy’s console port is marked as ‘DEBUG PORT” on Eddy DK board, and its communication settings are set to 115200 bps, none parity, 8 data bits with 1 stop bit. In Windows environment, connect console port in Eddy and serial port in PC running a Windows with a serial crossover cable (Only 3 lines of Tx, Rx, and GND are required). Run HyperTerminal and set communication environment to 115200 bps, None Parity, 8 Data Bits, 1 stop bit. In order to use serial communication emulator in LemonIDE, please refer to “7.3 Terminal”. If all configurations are set, click Run Method to upload the binary file to the target system, in this case to Eddy, and check its result of execution.

When you click the icon, last executed “Run Method” will be processed. List box next to icon list is where you can select a “Run Method” that is registered multiply.

Image shown below is a result of executing binary file made from hello.c in HyperTerminal. (4.2 Create and Edit Source Code)

For Windows Vista/7/8 users please refer to following link for HyperTerminal replacement.

http://windows.microsoft.com/en-us/windows-vista/what-happened-to-hyperterminal



```
SUM (1 ~ 1000) =  500500
```

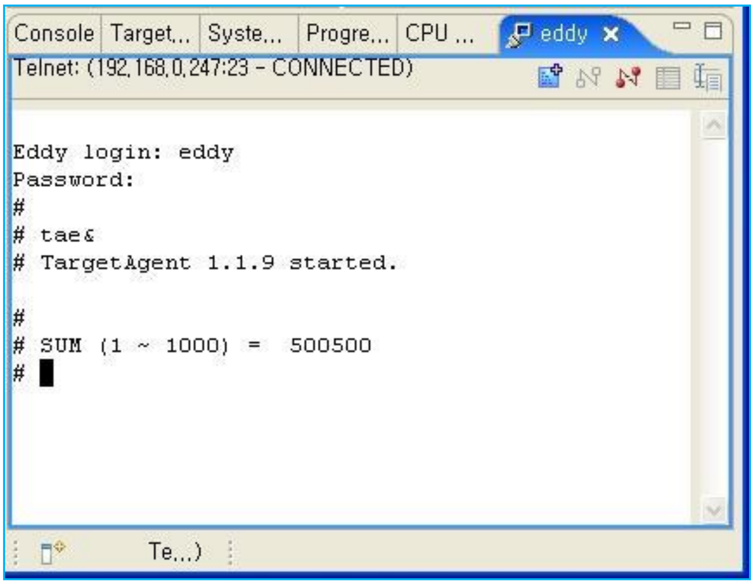
4.5.3 Execute Run Method (Check the result in Telnet)

There are no models in Eddy series except Eddy DK that has serial console port, thus checking result through serial port is not possible. In this case, telnet can be used as a console port.

To change the console port from serial port to telnet, login to Eddy using a terminal program and execute Eddy target agent, aka. “tae”, manually as a background process as shown below. Execution results of uploads from LemonIDE can now be checked through telnet. Refer to “7.3 Terminal” for instructions on telnet terminal connections. When you are ready to execute, click Run Method. When the binary file is sent to the target system, in this case the Eddy, you can check the result of execution from it.

When you click the icon, last executed “Run Method” will be processed. List box next to icon listis where you can select a “Run Method” that is registered multiply.

Image shown below is a result of executing binary file made from hello.c in telnet from section 4.2 Create and Edit Source Code.



Telnet: (192.168.0.247:23 - CONNECTED)

```
Eddy login: eddy
Password:
#
# tae&
# TargetAgent 1.1.9 started.

#
# SUM (1 ~ 1000) =  500500
# 
```

4.6 Creating a Firmware Image

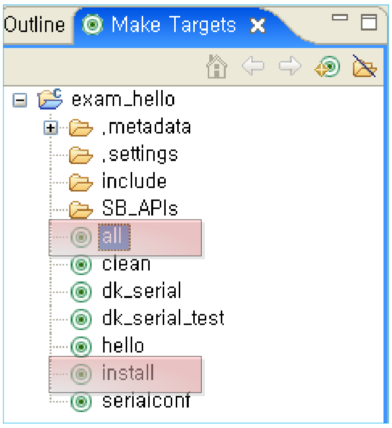
In this chapter, the process of creating a firmware image, compile it and load it to its target is shown.

4.6.1 Modify Makefile

To create an image, modify the Makefile added in “ exam_hello” project. Please refer to the example from section 4.3.1, 4.3.2, and 4.3.3 when modifying the make target.

4.6.2 Add Make Target

When you are done with modifying the Makefile, add it to make target. Make target is located in Make View left section of the screen in LemonIDE. Use the right-mouse click from the related project to select “Add Make Target”. You can find more information about make target from “4.3.2 Add Make Target”. Following image is an example from make target created from section 4.3.2.



all :

Create a binary file added to Target in Makefile.

install :

Copy the binary file added to Target in Makefile to /usr/bin directory, location of Ramdisk in Eddy. Copy the binary file registered in target from Makefile to /usr/bin directory in Eddy Ramdisk folder (dk_default).

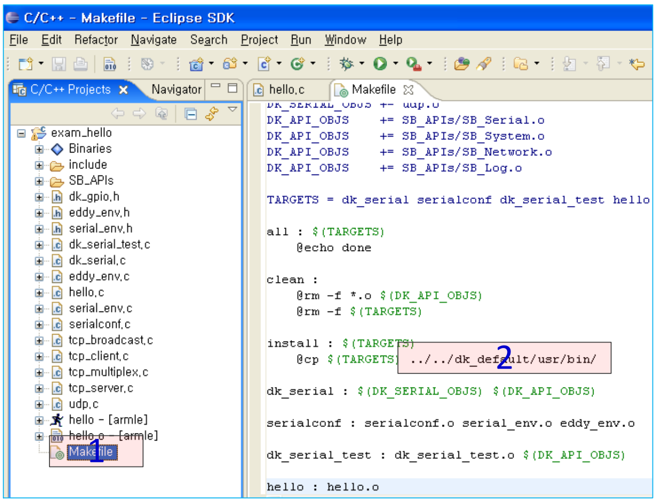
4.6.3 Create Firmware Image

When you are trying to copy the binary file, which is going to be made for firmware image, you created to Eddy Ramdisk (Linux file system), double-click "install" in exam_hello project from "Make View" screen. It will copy all the registered binary files to Eddy ramdisk.

If the following message appeared in the console window when you double-clicked "install", or if the user added to some arbitrary folder, Makefile needs to be modified.

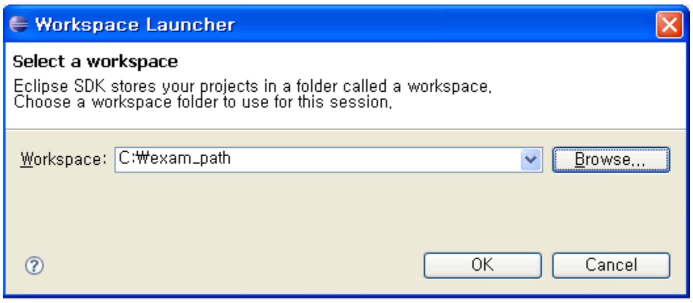
```
cp: target `../../dk_default/usr/bin/' is not a directory
Makefile:29: recipe for target `install' failed
make: *** [install] Error 1
```

Box 1 shown below is a Makefile from the Project View. When you double-click it you can modify it. If you would like to use a dedicated editor, you can go to its location and access from there. After opening Makefile, content from 2 will appear in code related to install. This is where you should modify.



```
DK_SERIAL_OBJS  += udp.o
DK_API_OBJS     += SB_APIs/SB_Serial.o
DK_API_OBJS     += SB_APIs/SB_System.o
DK_API_OBJS     += SB_APIs/SB_Network.o
DK_API_OBJS     += SB_APIs/SB_Log.o

TARGETS = dk_serial serialconf dk_serial_test hello

all : $(TARGETS)
    @echo done

clean :
    @rm -f *.o $(DK_API_OBJS)
    @rm -f $(TARGETS)

install : $(TARGETS)
    @cp $(TARGETS) ../../dk_default/usr/bin/

dk_serial : $(DK_SERIAL_OBJS) $(DK_API_OBJS)

serialconf : serialconf.o serial_env.o eddy_env.o

dk_serial_test : dk_serial_test.o $(DK_API_OBJS)

hello : hello.o
```

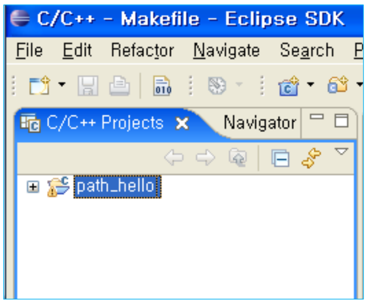
In this example, C:\eddy_sdk\userfs\dk_default\usr\bin location is /usr/bin location in Eddy ramdisk. Register the target addrees in 2 so that it can be accessed by Makefile. The below is an example of how to register workspace in hello_path folder at C: drive and when path_hello project is created, how to modify Makefile.


Workspace Launcher

Select a workspace

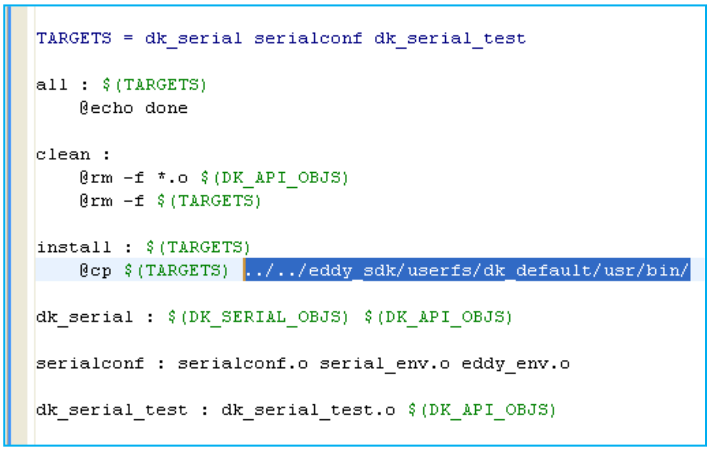
Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:₩exam_path


C/C++ - Makefile - Eclipse SDK
File Edit Refactor Navigate Search
C/C++ Projects
Navigator
path_hello

Please refer to examples in chapter 4 to import files and create Make Target. The location of Makefile in this case is at "C:\exam_path\path_hello\Makefile".

Therefore, the path for "C:\eddy_sdk\userfs\dk_default\usr\bin" in Makefile will be saved as "../../eddy_sdk/userfs/dk_default/usr/bin/" as shown below. Save the progress once the modification is completed.

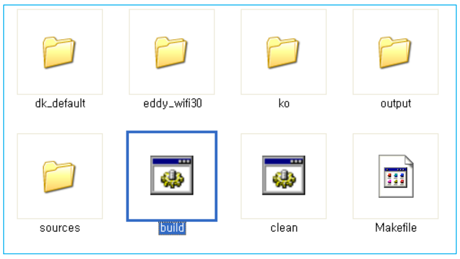

```
TARGETS = dk_serial serialconf dk_serial_test

all : $(TARGETS)
    @echo done

clean :
    @rm -f *.o $(DK_API_OBJS)
    @rm -f $(TARGETS)

install : $(TARGETS)
    @cp $(TARGETS) ../../eddy_sdk/userfs/dk_default/usr/bin/

dk_serial : $(DK_SERIAL_OBJS) $(DK_API_OBJS)

serialconf : serialconf.o serial_env.o eddy_env.o

dk_serial_test : dk_serial_test.o $(DK_API_OBJS)
```

After running install and copying user written binary file to ramdisk in Eddy, let's zip the image file and apply it to Eddy. C:\eddy_sdk\userfs\dk_default folder is the ramdisk image folder in Eddy DK. The parent directory of this is userfs folder where Makefile, with which you can zip ramdisk folder in Eddy, exists. You can simply run "make" to zip these.

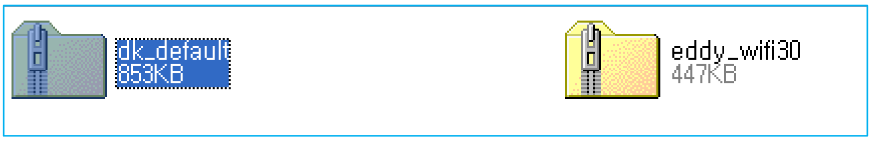
The following is an example for using command prompt to run "make".

```
C:\eddy_sdk\userfs>make
   adding: etc/ (stored 0%)
   adding: etc/init.d/ (stored 0%)
   adding: etc/init.d/U01i2c (deflated 30%)
   …
   adding: usr/sbin/wpa_passphrase (deflated 43%)
   adding: usr/sbin/wpa_supplicant (deflated 51%)

C:\eddy_sdk\userfs>
```

Eddy SDK provides an executable file that the user can use in GUI environment. Click build file at "C:\eddy_sdk\userfs" which is same as running the "make".



After zipping process is complete, you can check the result in "C:\eddy_sdk\userfs\output".



4.6.4 Firmware update

Upload and save the created firmware file in the flash memory in Eddy.

There are three ways to upgrade the firmware by FTP, Web browser, and bootloader. For upgrading by web browser, please refer to "Eddy-Serial-User_Guide". For upgrading by FTP server or bootloader please refer to "Eddy_DK_Programmer_Guide". Refer to "5.2 Firmware update using FTP server" and "9.1 System recovery" each from those documents.

Chapter 5. LemonIDE Debugger

LemonIDE supports breakpoint and tracepoint debugging. This chapter will demonstrate procedures of breakpoint debugging with the hello.c project introduced in section "4.2 Creating & Editing Source Codes"

Adding a breakpoint
Remote debugging configuration
Starting remote debugging
Program run control
Moving into a function
Confirm and modify the values of variables, registers, and equations.
Leaving a function

5.1 Debugging Preparation

Unlike debugging on a host, remote target debugging requires some configurations to be set.

5.1.1 Execute Target Agent

Execute
Refer to "4.5.2 Execute Run Method (Check the result in serial port)" or "4.5.3 Execute Run Method (Check the result in Telnet)" then run the target agent in Eddy.

5.1.2 What to watch out when setting for compiler option

The following debugging options must be set with care.
Makefile in "4.3.1. Creating and Updating Makefile" is set with optimization compile option and breakpoint debugging and tracepoint debugging is not possible.

Optimization option Debugging not possible	-O, -O1, -O2, -O3, -Os	Optimization option to minimize the size or the speed of binary file. Debugging is not possible when binary file is compiled with this option.
	strip	Zip program uses this option to minimize the size while it

		compiles. Debugging information is stripped out when compiled with this option and debugging is not possible.
Debugging option	-g1 -g -g3	Minimal Debug Level breakpoint debugging option Default Debug Level breakpoint debugging option Maximum Debug Level breakpoint debugging option

Following is an example where –Os is removed from the previous configuration for debugging Makefile in “4.3.1 Modify Makefile”.

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

5.2 Execute Debugging

The hello.c introduced in “4.2 Create and Edit Source Code” will be used to demonstrated the use of debugger

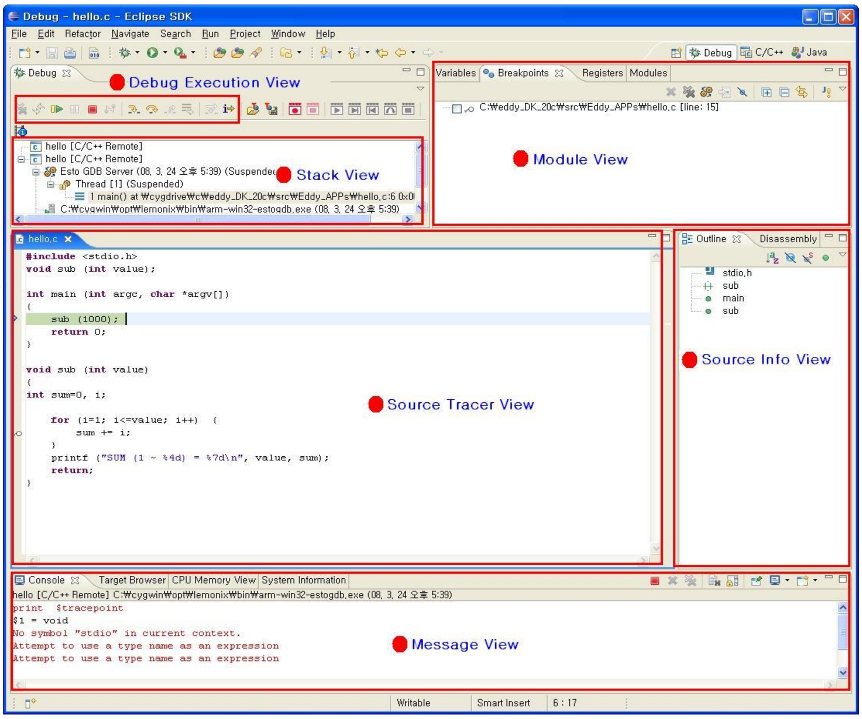
Debug Icon (Breakpoint & Tracepoint)

Click on the left icon to execute the last Debug Method.

List box next to the icon contains multiple Debug Method which you can select.

When debugging is started, the screen where “C/C++” is shown will be changed to “Debug” screen.

The following screenshot shows the changed screen.



```
C:₩eddy_DK_20c₩src₩Eddy_APPs₩hello.c [line: 15]
```

```
hello [C/C++ Remote]
hello [C/C++ Remote]
  Esto GDB Server (08, 3, 24 오후 5:39) (Suspended
    Thread [1] (Suspended)
      1 main() at ₩cygdrive₩c₩eddy_DK_20c₩src₩Eddy_APPs₩hello.c:6 0x0
  C:₩cygwin₩opt₩lemonix₩bin₩arm-win32-estogdb.exe (08, 3, 24 오후 5:39)
```

```
#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;
}
```

```
hello [C/C++ Remote] C:₩cygwin₩opt₩lemonix₩bin₩arm-win32-estogdb.exe (08, 3, 24 오후 5:39)
print  $tracepoint
$1 = void
No symbol "stdio" in current context.
Attempt to use a type name as an expression
Attempt to use a type name as an expression
```

Stack View	Displays information related to threads and stacks during debugging.
Module View	Displays information on variables, breakpoint, equations and modules during debugging.
Source Tracer view	Displays breakpoint locations in source codes.
Source Info View	Displays functions and reverse assembly codes related to source file.
Message View	Displays target's status.
Debug Execution View	Controls debugging start, stop and breakpoints.

5.3 Breakpoint Debugging

This chapter will introduce breakpoint debugging.

After debugging, it is recommended that the completed program be restored with original CFLAGS option and STRIP values to optimize execution module.

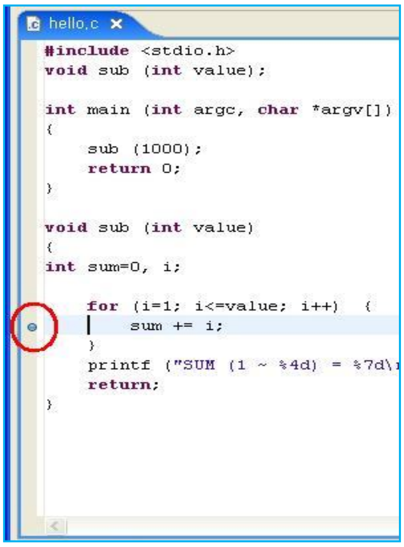
5.3.1 Modifying Compiling Environment and Execute Debugging

As mentioned in "5.1.2 What to watch out when setting for compiler option", Makefile must be modified for debugging. The below shows removed optimized option, but added options to enable breakpoint debugging where STRIP is removed so that execution module will not be compressed.

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

After modifying and saving Makefile, recompile hello.c so that it can be debugged.

5.3.2 Add Breakpoint



```
#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\
    return;
}
```

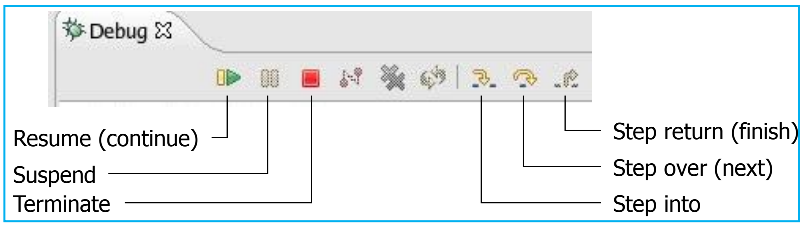
Breakpoints can be added in C/C++ perspective or debug perspective. From the source line that you would like to add a breakpoint, select the left margin in the edit view to add a breakpoint. Double-click to remove the breakpoint. You can also add a breakpoint from the popup menu by right-clicking on the source code line.

Added breakpoint is added to "BreakPoints" of "Module View".

Various debugging information can be checked on "Windows" → "Show View" → "Other.." → "Debug".

5.3.3 Control Execution in Program

" Debug Execution View"provides commands to control program execution during debugging.



Main commands used to control program execution are as follows:

Resume: Continue until the next breakpoint.

Suspend: Pause the program.

Terminate: Terminate the debugging session completely.

Step into: Continue to the next line, and move into a function when there is one.

Step over: Continue to the next line, but do not enter a function.

Step return: Execute until the end of a function

Chapter 6. Monitoring Tool

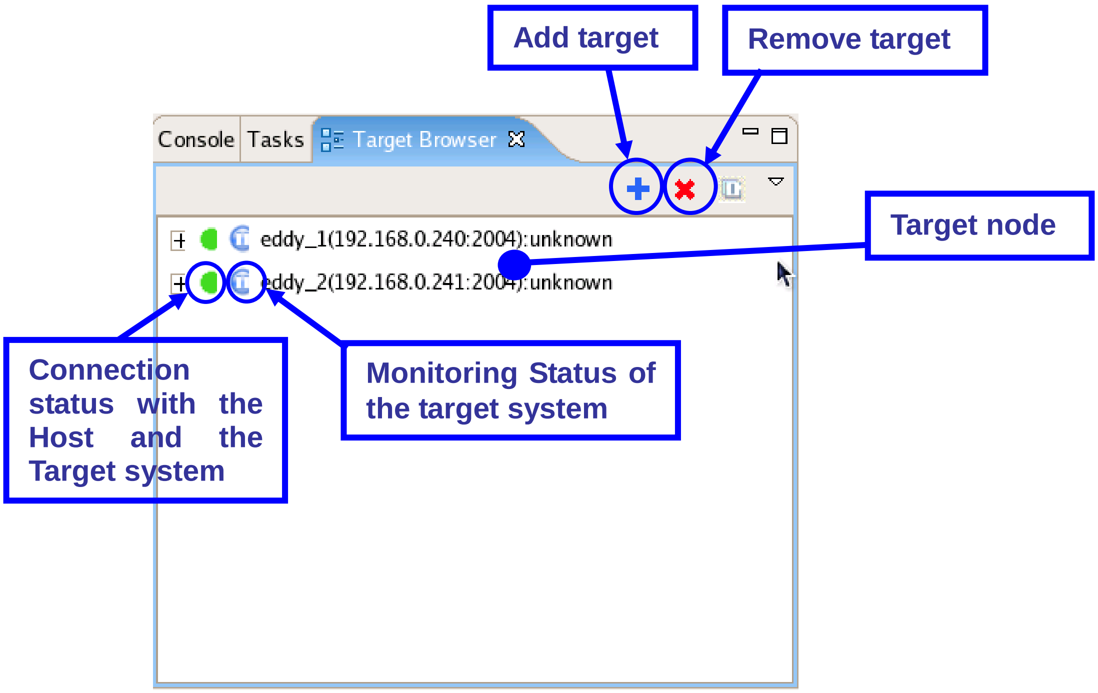
This chapter focuses on the target browser and the monitoring tool used to monitor target system information from LemonIDE. Select "Windows" → "Show View" from LemonIDE title menu to select a monitoring tool supported in LemonIDE.

6.1 Target Browser

Target browser's user interface & its usage will be discussed in this section.

6.1.1 Target Browser User Interface

Target browser user interface is shown below.


Add target
Remove target
Console
Tasks
Target Browser
eddy_1(192.168.0.240:2004):unknown
eddy_2(192.168.0.241:2004):unknown
Target node
Connection status with the Host and the Target system
Monitoring Status of the target system

Target browser may contain multiple target nodes. Each target node includes the following information.

Connection status between the host and the target

Remote monitoring status

Target name

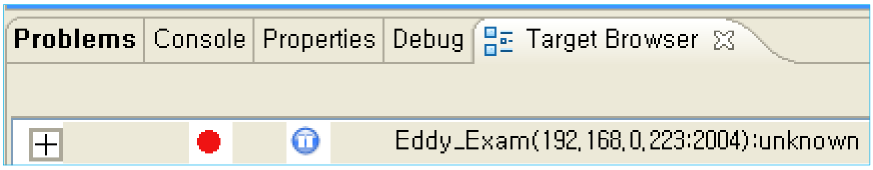
IP address

Port number

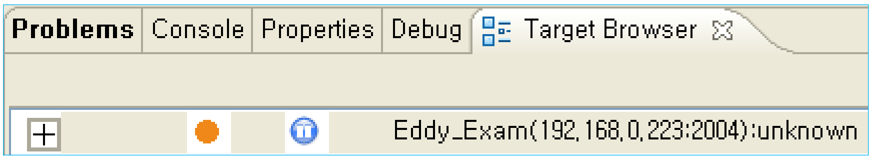
Target architecture

The colors of icons showing the connection status of the host and the target are different depending on the connection status.

Red: Not connected with the target or the target does not exist. (The target does not respond to a ping)

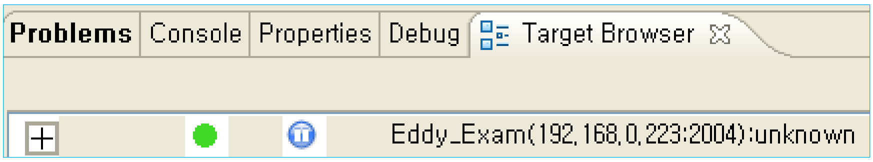


Orange: Target is alive and connect to the network, but not with the target agent.

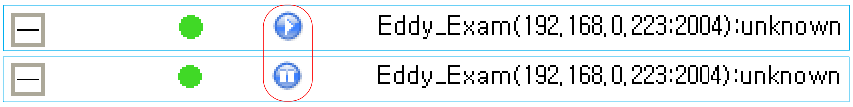


Green: Connected to the target agent and can start developing remotely.

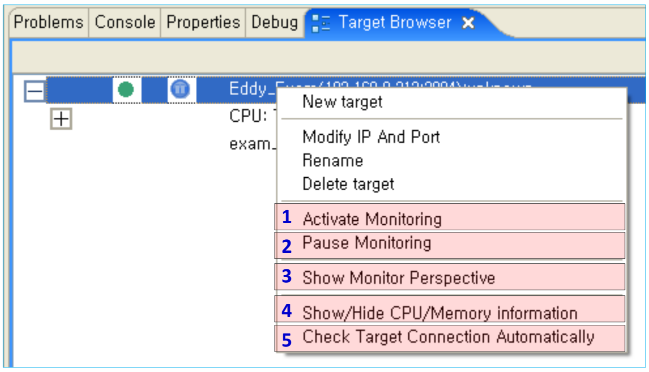
(Transfer compiled binary file from the host to the target board and execute and checking results from the host.)



When remote monitoring is taking place, the target monitoring status icon will show a "play" icon, but when it is not, a "pause" icon will be displayed as shown below.


Eddy_Exam(192.168.0.223:2004):unknown
Eddy_Exam(192.168.0.223:2004):unknown

Each target in the target browser can register various execution related settings. Execution related settings are referring to objects which include information regarding execution of the program and the debugging. It can be created, removed or modified from "Run / Debug Configuration" windows in Eclipse LemonIDE.



1.Activate Monitoring: enables views in target status monitor. CPU/Memory View and Process List View will periodically start monitoring.

2. Pause Monitoring: Stops remote monitoring for the target. Periodically monitoring CPU/Memory View and Process List View will be stopped.

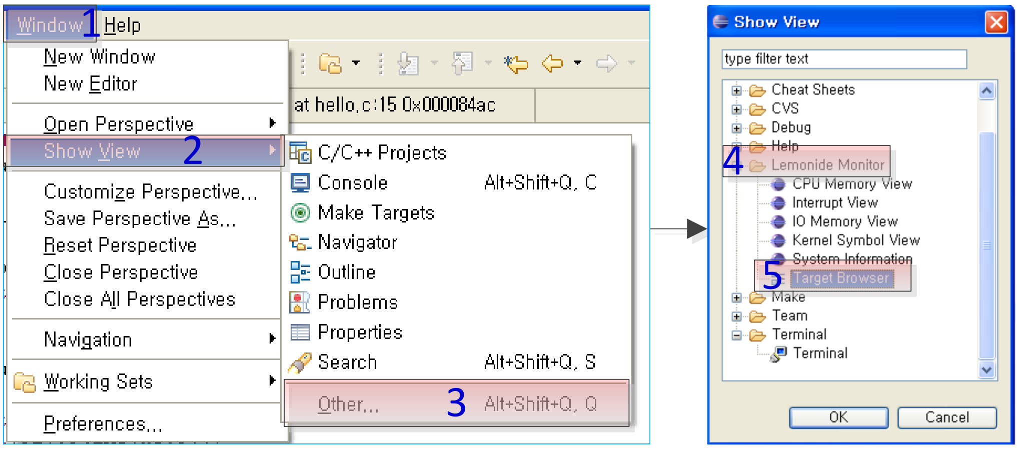
3. Show Monitor Perspectives: Show LemonIDE Monitor perspective screen.

4. Hide CPU/Memory Information: When not checked, CPU and memory usage will be shown in the sub-node of the first node. However, the actual value will be displayed when it is currently monitoring. This cannot be set individually by node, but the whole target node can be shown or hidden.

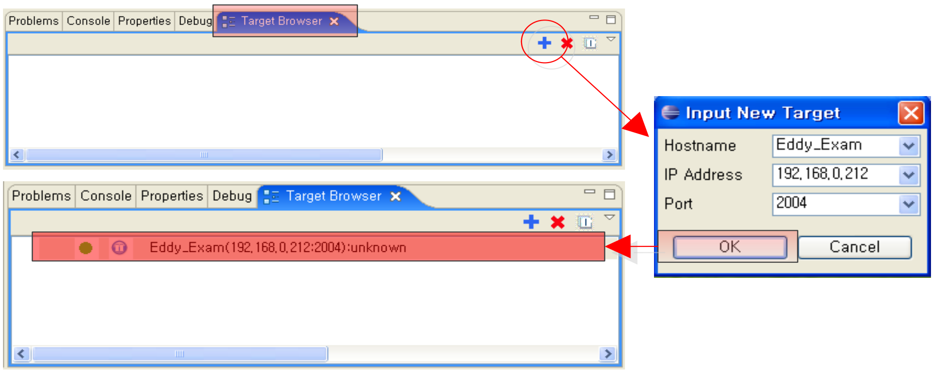
5. Check Target Connection Automatically: Select whether the target and the host added in the target broser to connect automatically or not.

6.1.2 Add Target to Target Browser

From LemonIDE title menu, select "Windows" → "Show View" → "Other.." then "Show View"→ "LemonIDE Monitor"→ Target Browser" to see the Target Browser in the bottom of "Message View" from LemonIDE.



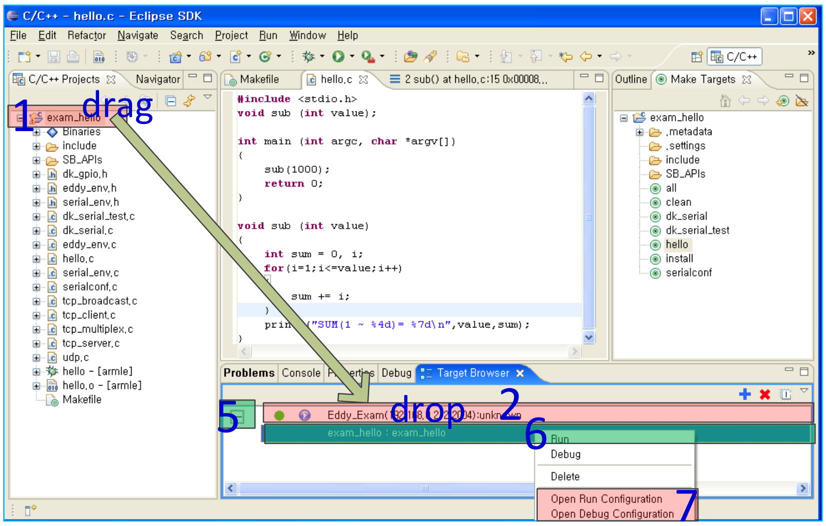
When the desired target is not in the list, it should be added to the Target Browser. From the screen shot below, when click on the blue + icon, a window will appear where you can add the information for the target. Fill the blank for the target IP address, port number of running target agent. In this example, let us assume that the target IP is 192.168.0.212.



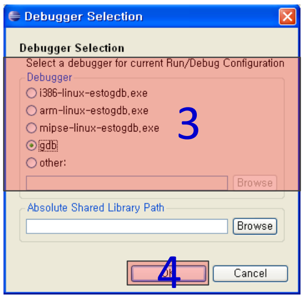
Hostname Eddy_Exam
IP Address 192.168.0.212
Port 2004

Eddy_Exam(192.168.0.212:2004):unknown

Then, new target node will be created in the target browser. Drag the project named “exam_hello” from 1 below to 2 in the target browser. When the project is dropped, a debugger select window will appear. When first appeared, “gdb” will be selected as shown in 3 below. Select a debugger appropriate for the target. When the debugger is selected, “Absolute Shared Library Path” will be automatically set for the corresponding selection.



```
#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);
}
```



Debugger Selection

Select a debugger for current Run/Debug Configuration

Debugger
○ i386-linux-estogdb.exe
○ arm-linux-estogdb.exe
○ mipse-linux-estogdb.exe
● gdb
○ other:

Browse

Absolute Shared Library Path

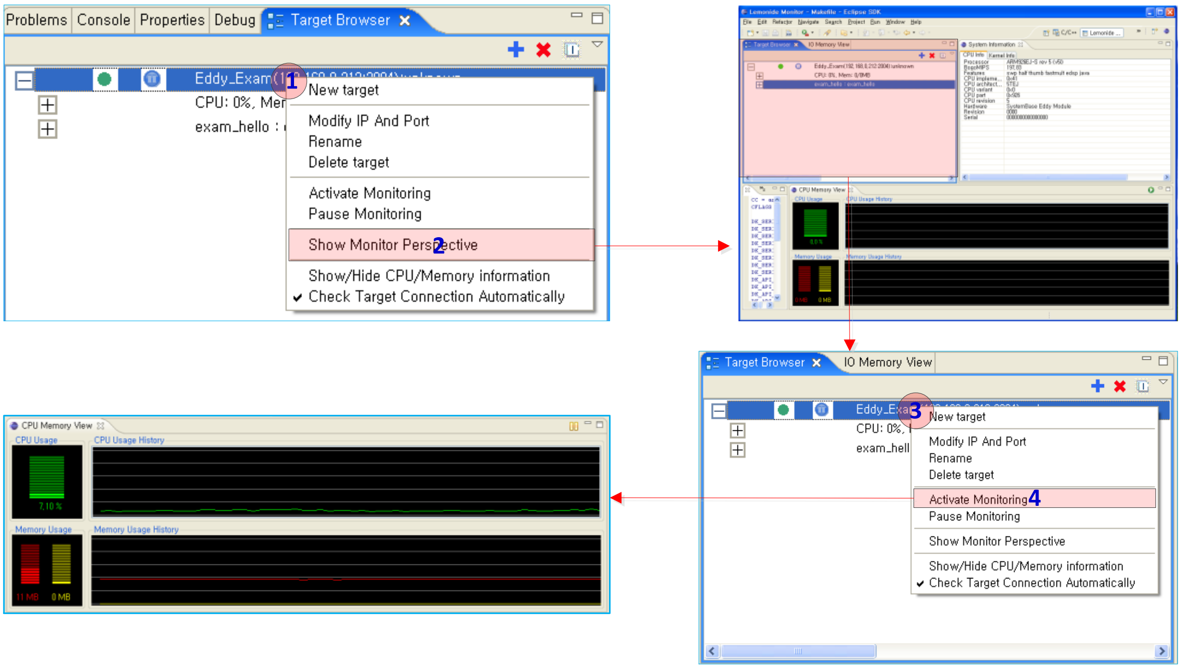
Browse

When “Absolute Shared Library Path” needs to be changed, use the Browse button to change the path.

After setting for the debugger in the Debugger Selection window, click OK button to see the Run / Debug Configuration created in 5 above. Click run from 6 to set settings in 7.

6.2 Monitoring Target Status

Select “Show Monitor Perspectives” menu from the target node in the target browser. Then, LemonIDE Monitor perspective will open, as shown below. No real data is shown yet, since remote monitoring has not been started.



As shown in 3 above, when you click Eddy_Exam node and click "Activeate Monitor Views" from the short-cut menu, remote monitoring will be started where CPU/Memory View will update the graph and the process list view every 10 seconds.

From Eddy_Exam node use right-mouse-click to call the short-cut menu and select "Pause Monitoring" menu. After checking if the remote monitoring is stopped, select "Activate Montior Views" menu to execute to continue remote monitoring.

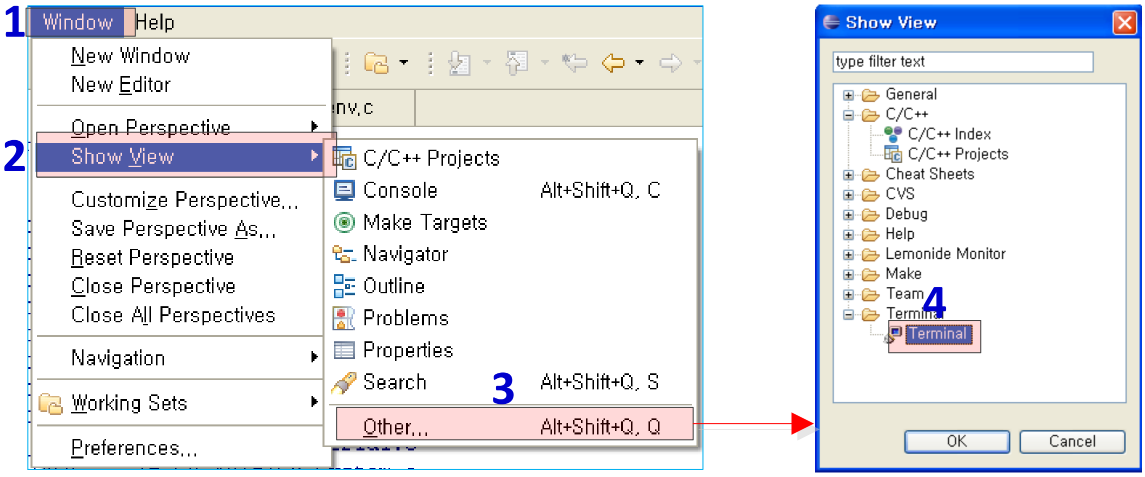
From LemonIDE Monitor, you can check the views by selecting "Windows – Show View" menu.

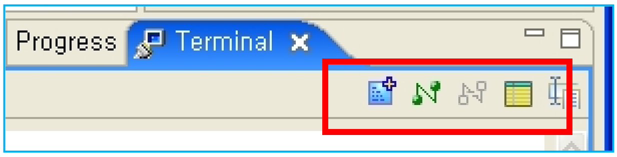
6.3 Terminal

Remote execution and remote debugging on target system requires a terminal program.

A terminal program refers to a serial emulator and a telnet emulator in this case and LemonIDE supports both emulators enabling remote execution & debugging of target system all in LemonIDE's IDE.

Select "Windows" → "Show View" → "Other…" → "Terminal" → "Terminal"from LemonIDE's title menu to enable a terminal. Terminal will be created in "Message View" located on the lower part of LemonIDE.



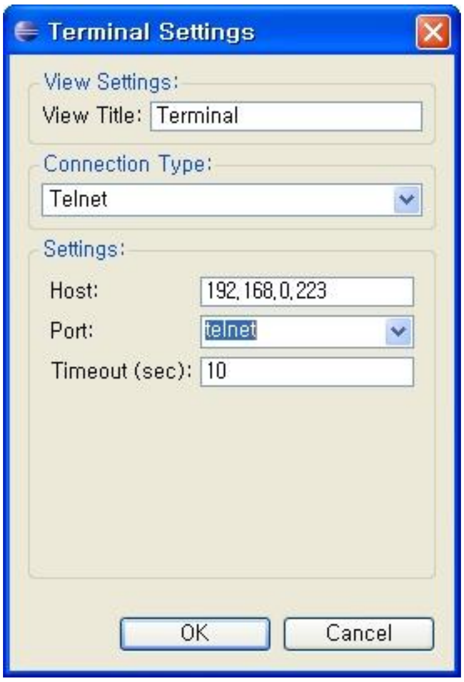


New Terminal	: Create a new terminal.
Connect	: Connect to registered target.
Disconnect	: Terminate the connection.
Settings	: Set target connection information

In order to execute a terminal as an independent window outside of Message View, right click on terminal window's title menu bar and select 'Detached".

In "Message View", on lower part of LemonIDE, select the newly created terminal, and click ▤ to set the connection environment.

View Title : Set terminal's name.



Terminal Settings

View Settings:

View Title: Terminal

Connection Type:

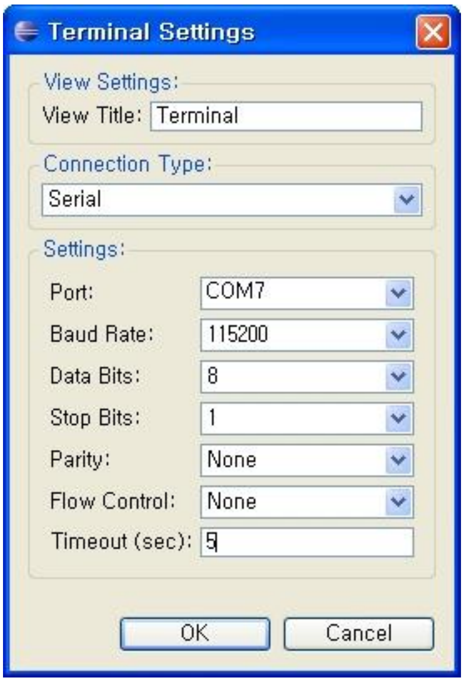
Telnet

Settings:

Host: 192.168.0.223

Port: telnet

Timeout (sec): 10



Terminal Settings

View Settings:

View Title: Terminal

Connection Type:

Serial

Settings:

Port: COM7

Baud Rate: 115200

Data Bits: 8

Stop Bits: 1

Parity: None

Flow Control: None

Timeout (sec): 5

Telnet Terminal

Host : Target's IP to connect

Port : Target's port to connect (Select Telnet)

Timeout : Timeout duration (A specified period of time that will be allowed to elapse in a system before the connection terminates when no activity)

6.4 CPU Memory View

LemonIDE target system's CPU Traffic & Memory usage can be viewed.

Select "Windows" → "Show View" → "Other…" → "Esto Monitor" → "CPU Memory View" from LemonIDE's title menu. CPU View can be found in "Message View" in lower part of LemonIDE.

▶ starts status check of CPU and Memory, and ⏸ stops the process.

In order to execute "CPU Memory View" as an independent window outside of Message View, right click on "CPU Memory View" window's title menu bar and select 'Detached".

6.5 Registers

LemonIDE target system's register status can be viewed.

Select "Windows" → "Show View" → "Registers"from LemonIDE's title menu to activate Registers view window. This window can be found in Make View on right side of LemonIDE.

In order to execute “Registers View” as an independent window outside of Message View, right click on “Registers View” window's title menu bar and select ‘Detached”.

6.6 Modify Workspace

Workspace is top directory intended for creating firmware to be used in target systems and includes numerous projects. A project can be defined as a directory for application to be executed in target systems. To change Workspace, select “file” → “Switch Workspace”from LemonIDE title menu.