

## Basler Components

```
CLUINT32 portIndex((CLUINT32)-1);
char szPortID[260] = {0};
CLUINT32 portIDLen = sizeof(szPortID);
if (numPorts > 1)
{
    for (CLUINT32 i = 0; i < numPorts; ++i)
    {
        portIDLen = sizeof(szPortID);
        res = pfn_clGetSerialPortIdentifier(i, szPortID, &portIDLen);
        if (res != CL_ERR_NO_ERR)
        {
```

## Building Pylon Applications with Eclipse Under Linux

### APPLICATION NOTES

Document Number: AW001025

Version: 01 Language: 000 (English)

Release Date: 29 July 2011



## **Contacting Basler Support Worldwide**

### **Europe:**

Basler AG  
An der Strusbek 60 - 62  
22926 Ahrensburg  
Germany  
Tel.: +49-4102-463-515  
Fax.: +49-4102-463-599  
bc.support.europe@baslerweb.com

### **Americas:**

Basler, Inc.  
855 Springdale Drive, Suite 203  
Exton, PA 19341  
U.S.A.  
Tel.: +1-610-280-0171  
Fax.: +1-610-280-7608  
bc.support.usa@baslerweb.com

### **Asia:**

Basler Asia Pte. Ltd  
8 Boon Lay Way  
# 03 - 03 Tradehub 21  
Singapore 609964  
Tel.: +65-6425-0472  
Fax.: +65-6425-0473  
bc.support.asia@baslerweb.com

**[www.baslerweb.com](http://www.baslerweb.com)**

**All material in this publication is subject to change without notice and is copyright Basler Vision Technologies.**

# 1 Introduction

This document explains how pylon applications can be built while using the Eclipse IDE under various Linux distributions.

The procedures described in this document assume you are using Ubuntu 9.10 x86, pylon 2.3 x86 and Eclipse IDE for C/C++ Developers, Version: Helios Service Release 2 (part of the Eclipse Helios 3.6.2 packages for Linux).

Be aware that in order to be able to run Eclipse, you must have previously installed the latest Java JRE (Java Runtime Environment). In our case the "openjdk-6-jre" (Open JDK Java Runtime) package and its related packages were already installed from the Ubuntu Synaptic Package Manager.

# 2 Steps

## 1. Pylon Installation

We first recommend downloading and installing the pylon 2.3 SDK package and making sure you are able to run the pylon tools, i.e., the pylon Viewer and the IP Configurator. Also make sure that you can compile the pylon SDK samples as described in the README and INSTALL files included in the SDK package.

The pylon SDK packages can be downloaded from the Basler's website:

<http://www.baslerweb.com>

## 2. System Configuration

For the linker to be able to find the pylon libraries at compile and run time, paths to the pylon library folders must be added to the linker's search path. This can be done by adding the paths to the pylon folders to the */etc/ld.so.conf* file.

To do that open a terminal and execute:

```
sudo gedit /etc/ld.so.conf
```

Enter your root password now and add the following paths to the beginning of the file based on where pylon was actually installed, e.g:

```
/opt/pylon/lib
```

```
/opt/pylon/genicam/bin/Linux32_i86
```

The */sbin/ldconfig* command must be issued after modifying the */etc/ld.so.conf* file, i.e.:

```
sudo /sbin/ldconfig
```

For running pylon based applications, the following environment variables must also be exported based on where pylon was actually installed, e.g.:

```
export PYLON_ROOT=/opt/pylon
```

```
export GENICAM_ROOT_V2_1=${PYLON_ROOT}/genicam
```

To accomplish this, you must add the above two lines to the beginning of your *.profile* file, i.e., open a terminal and execute:

```
gedit .profile
```

In order to improve the performance of your application, you should also add the following line:

```
export GENICAM_CACHE=${PYLON_ROOT}/xml_cache
```

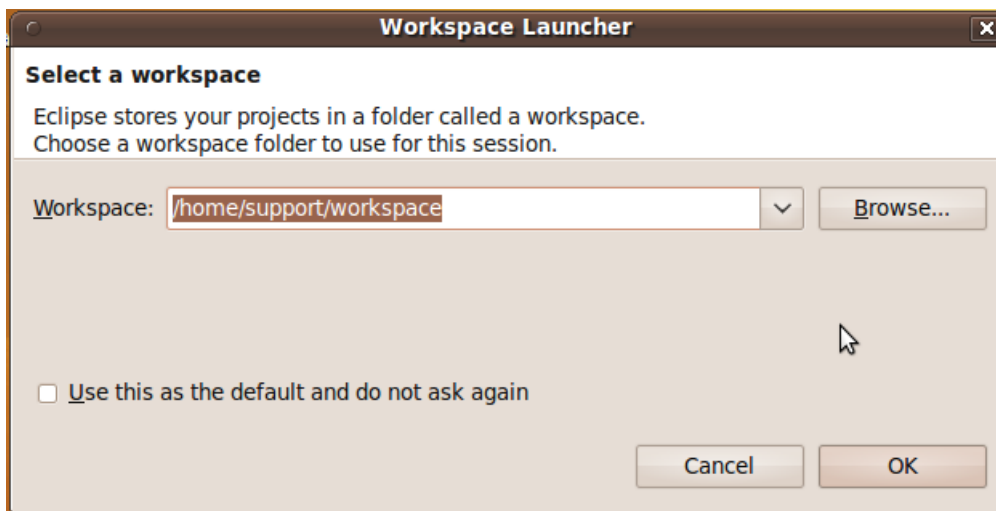
Note that the *xml\_cache* folder must be created first.

After you made these changes to the *.profile* file, you must log out or reboot for the changes to take effect.

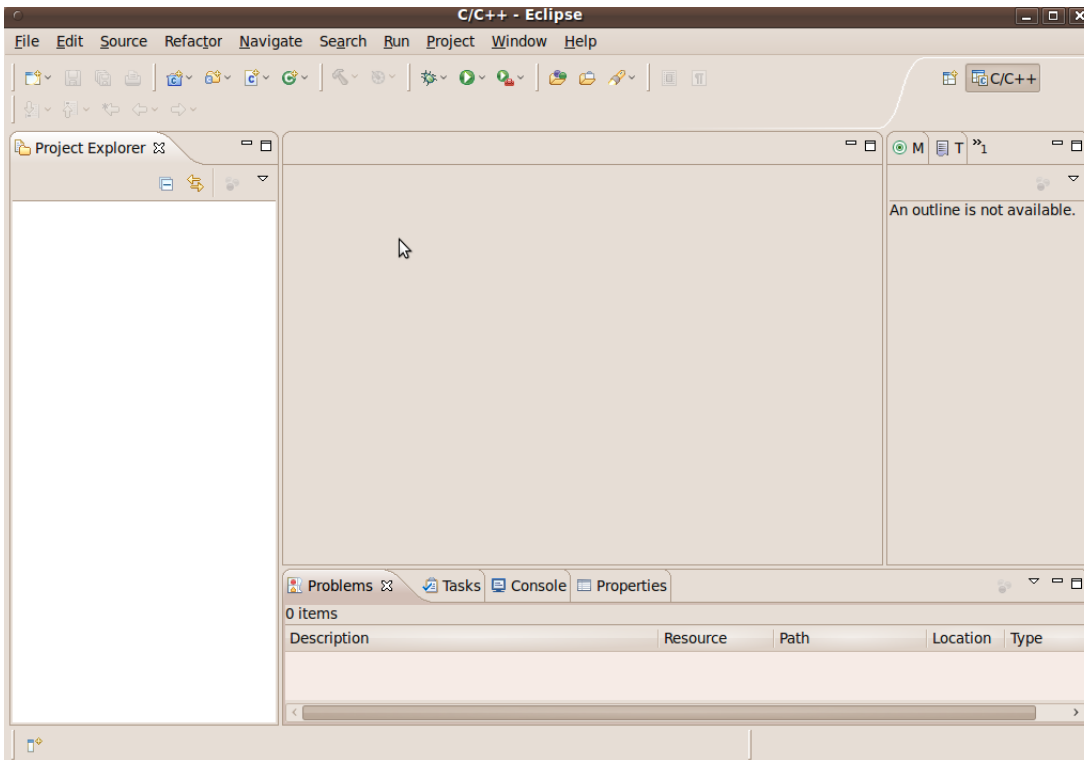
### 3. Project Setup

Now that you have logged in again, you can run Eclipse by double clicking the "eclipse" binary or from a terminal.

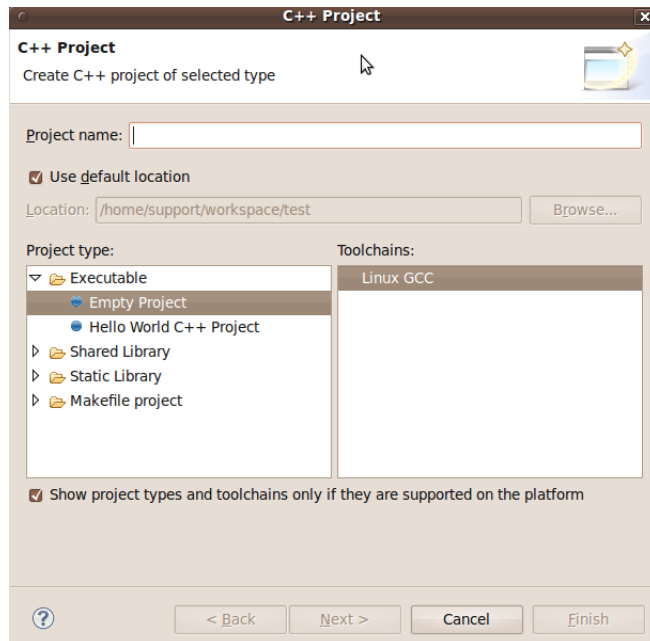
You will be asked to select a workspace where your projects will be saved now:



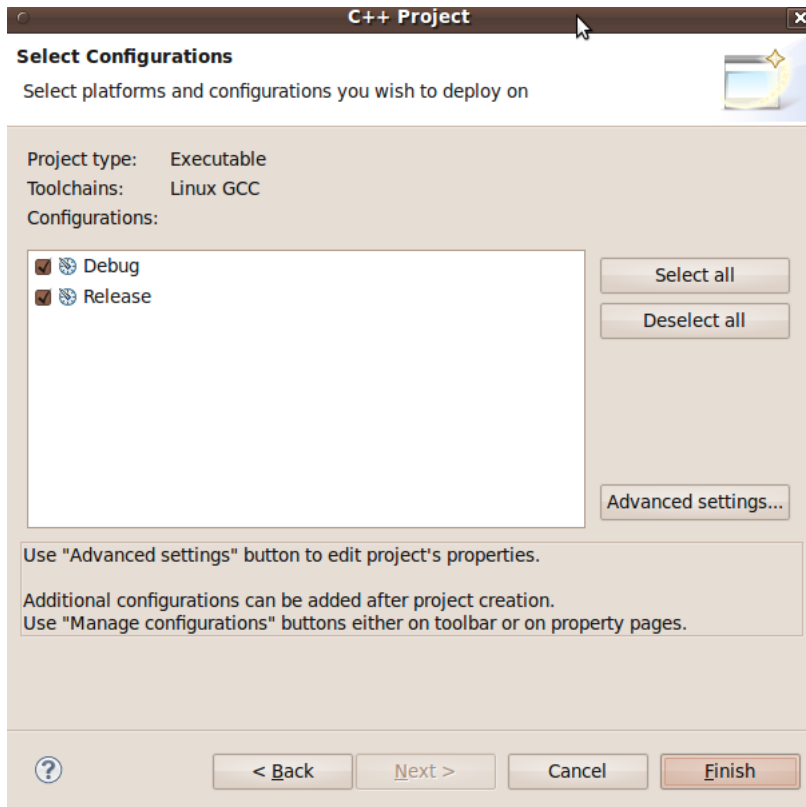
After selecting a workspace, you will see the main window:



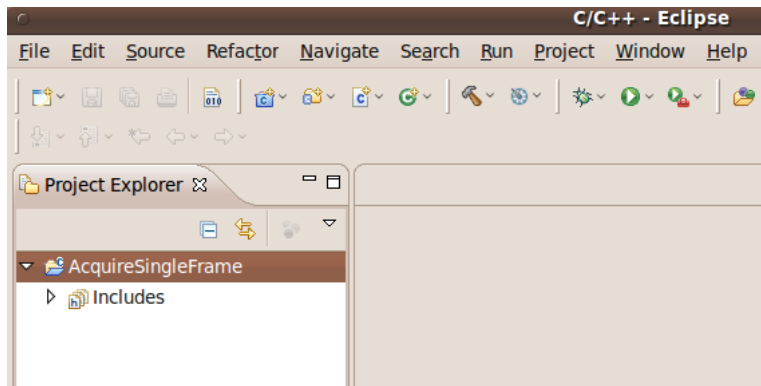
To create your own project, go to **File** → **New** → **C++ Project** and type in the project name, e.g., *AcquireSingleFrame*, and click **Next**.



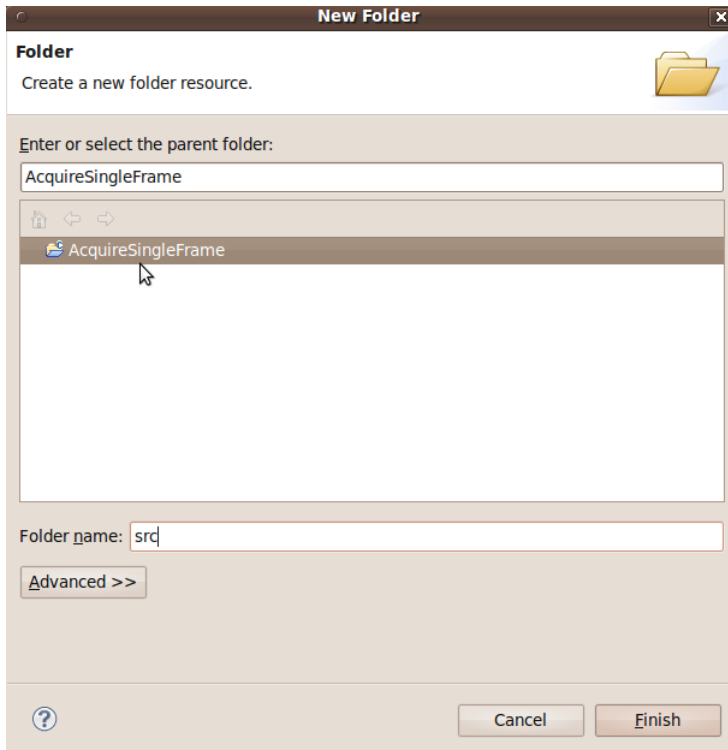
Click **Finish** in order to set up your project:



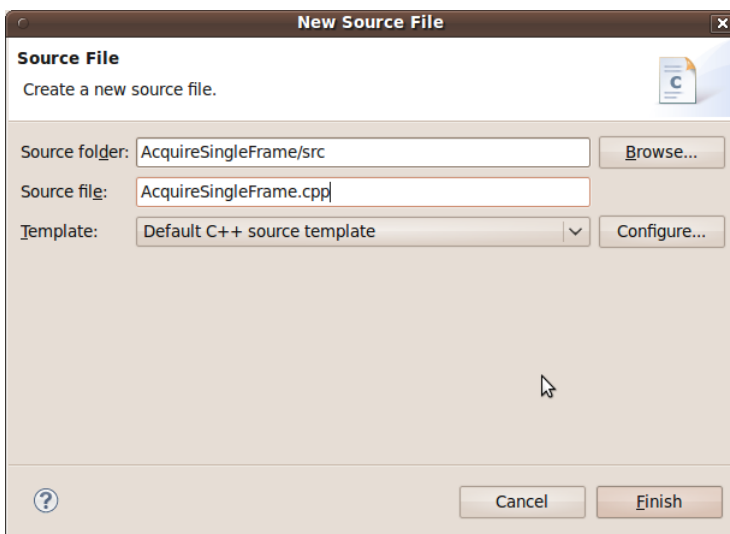
Your project will now be listed in the **Project Explorer** window:



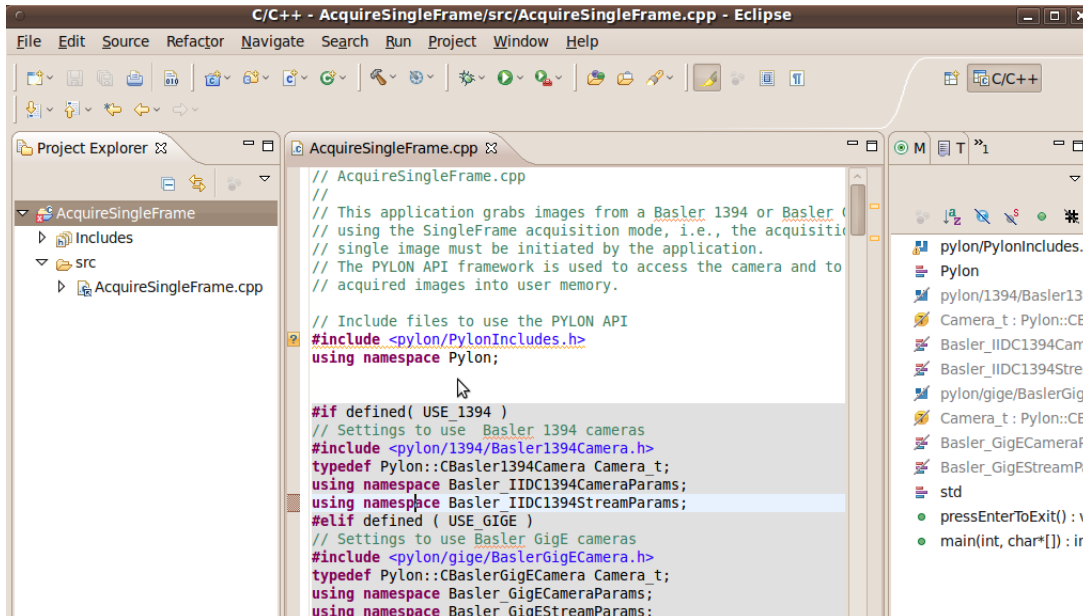
We will now create an additional folder called "src", into which our source code will be stored. To do this, right click on the project, i.e., *AcquireSingleFrame*, go to **New** and select **Folder**. Finally, type in "src" in the **Folder name** field and click **Finish**:



We will now create a new source file to contain our source code. To do that, right click on *src* and select **New** → **Source File**. Now type in the name of the source file, e.g., *AcquireSingleFrame.cpp* and click **Finish**:



Now that we have created our source file, we will copy the source code out of the pylon SDK sample *AcquireSingleFrame* and paste it into our source file:

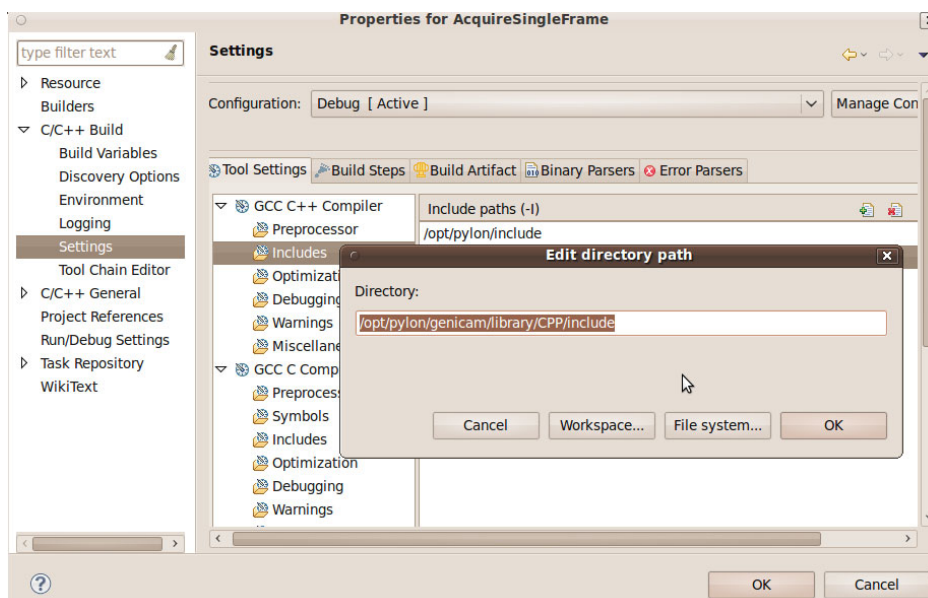


To configure our project settings appropriately, right click on the project and select **Properties**.

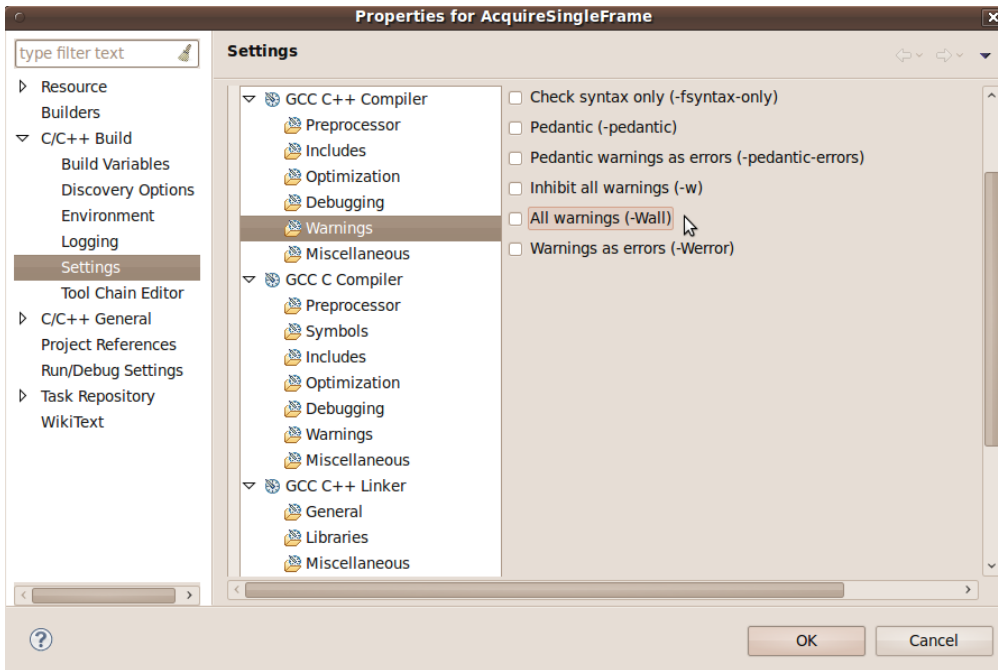
Now go to **C/C++ Build** → **Settings** → **GCC C++ Compiler** → **Includes** and add the following two include paths corresponding to where pylon was installed. For example, if you installed pylon under */opt/pylon*, your include paths would look like this:

*/opt/pylon/include*

*/opt/pylon/genicam/library/CPP/include*



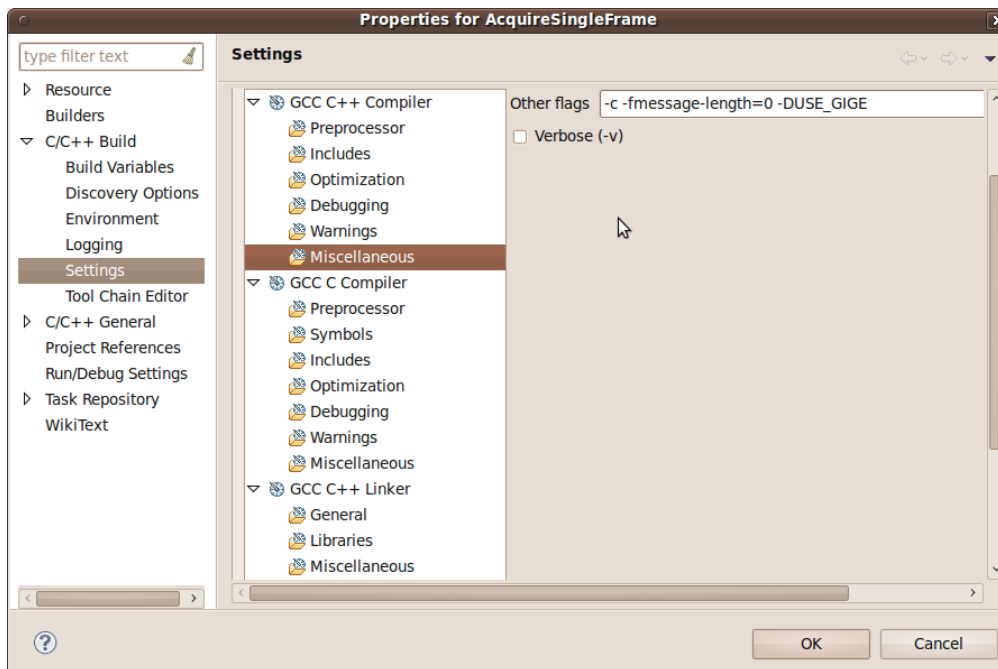
Next go to **Warnings** and deactivate **All warnings (-Wall)**:



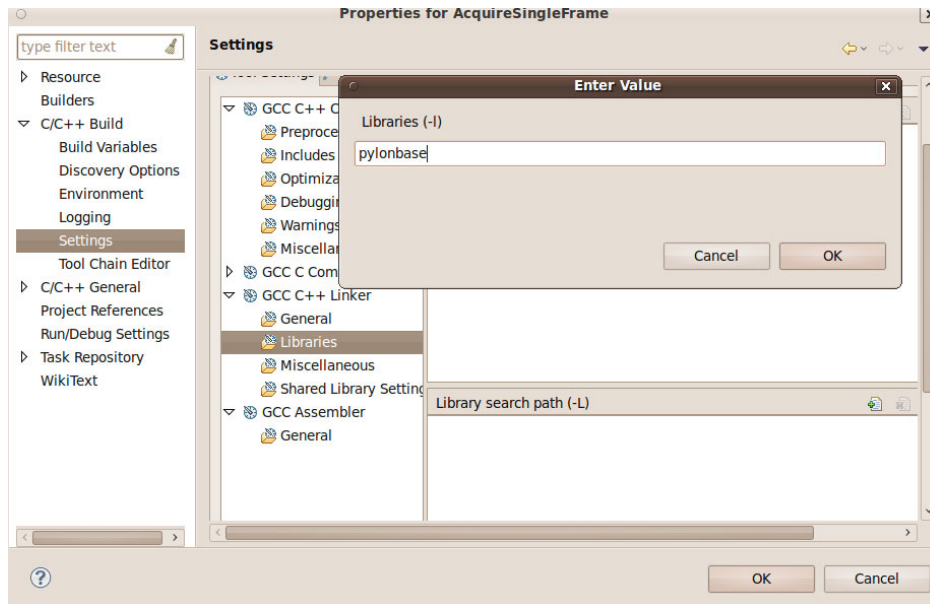
Next go to **Miscellaneous** and add the following flag to **Other flags**:

`-DUSE_GIGE`

Note that this flag is only needed for building the pylon SDK samples.



Now that we are ready with the compiler settings, go to **GCC C++ Linker** → **Libraries** and add *pylonbase* to **Libraries**:

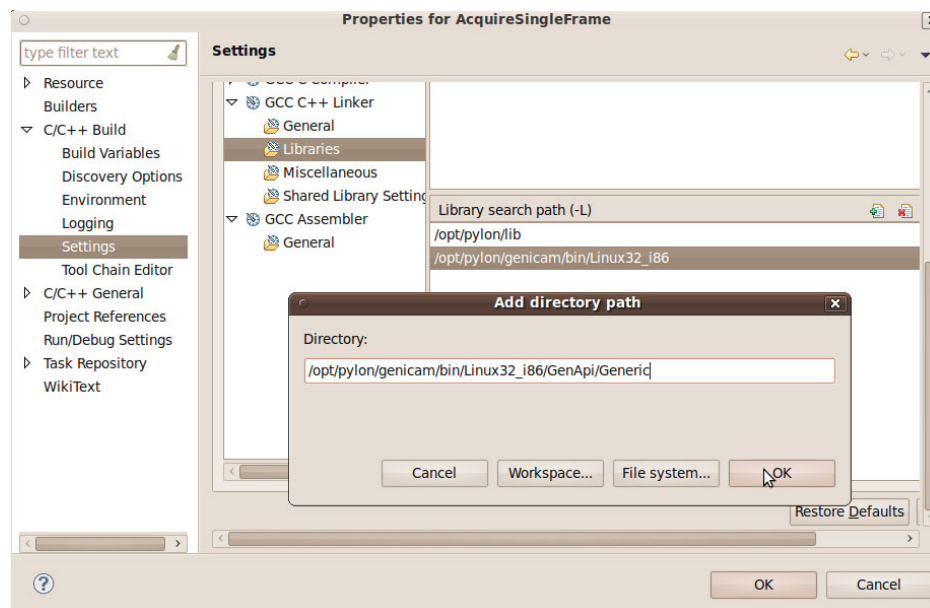


In addition, add the following three paths to **Library search path** based on where pylon was installed:

`/opt/pylon/lib`

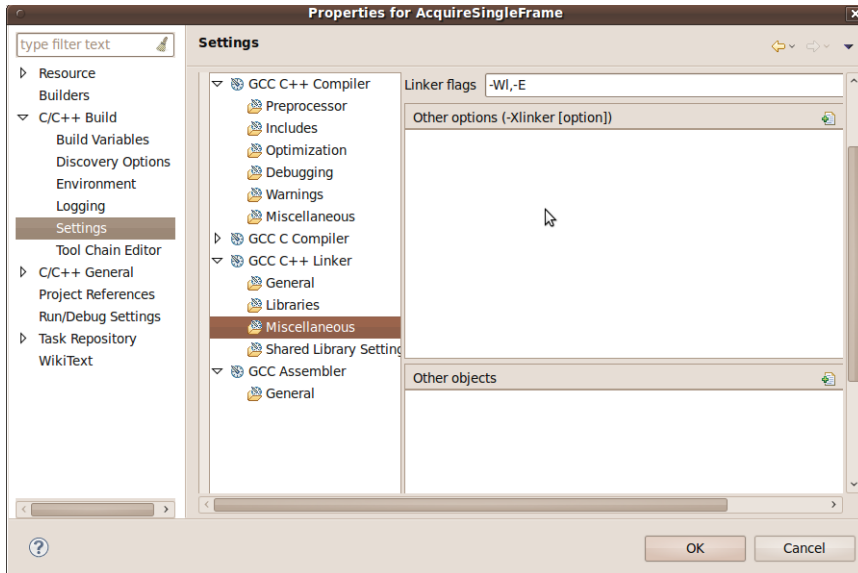
`/opt/pylon/genicam/bin/Linux32_i86`

`/opt/pylon/genicam/bin/Linux32_i86/GenApi/Generic`



Now go to **Miscellaneous** and add the following flags to **Linker flags**:

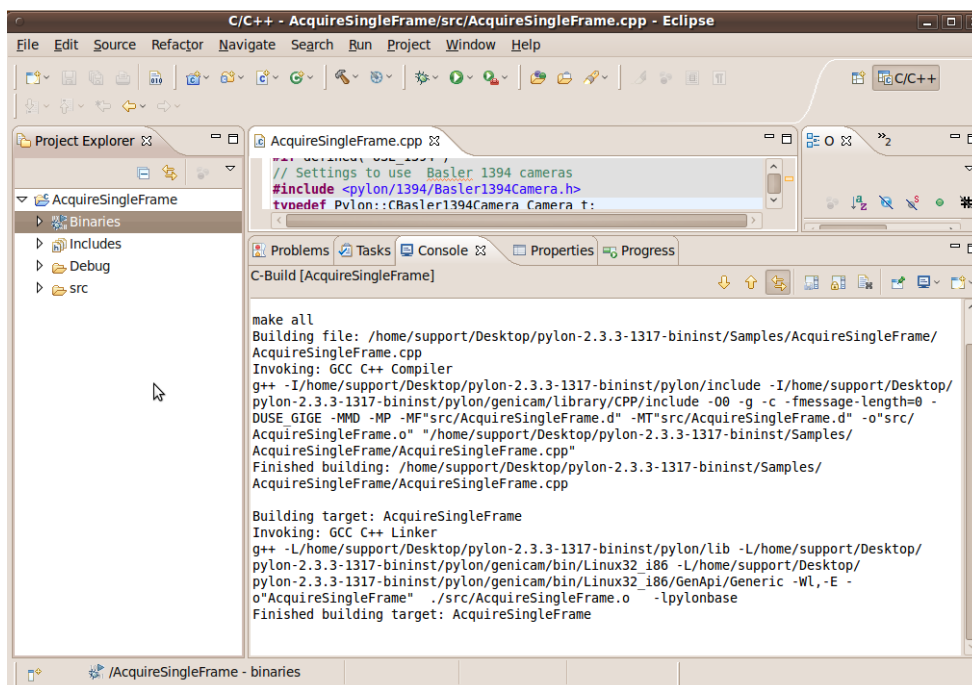
`-WL,-E`



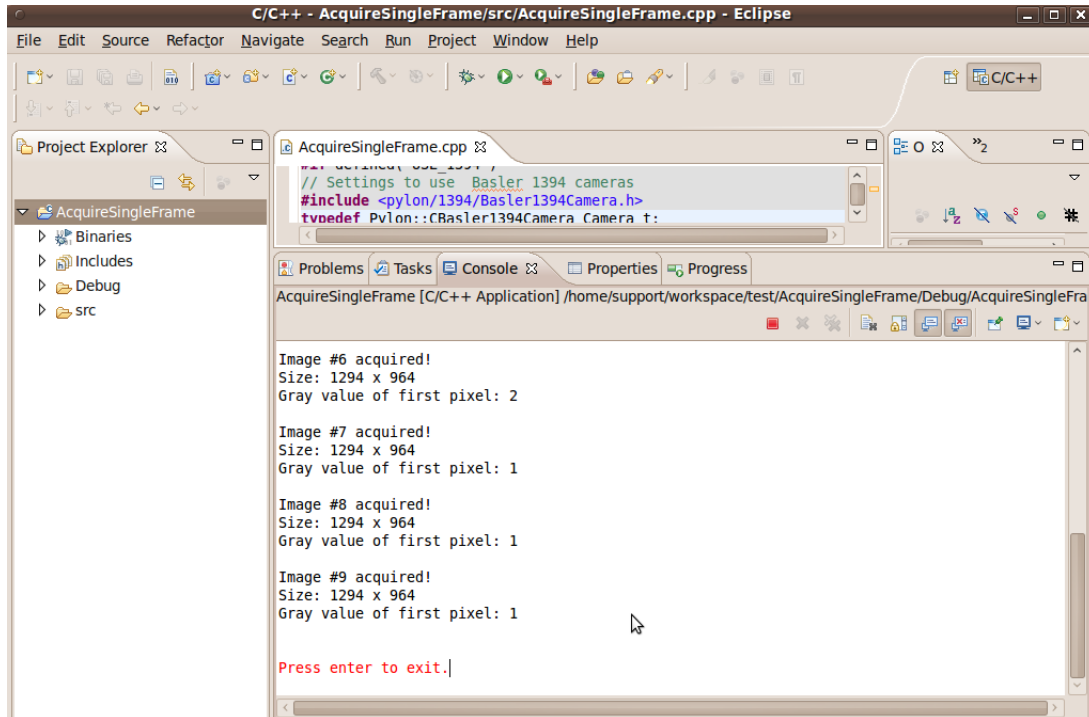
Now we have our project configured and we can build it.

To do that, right click on the project and select **Build Project**.

If you correctly followed the instructions in this document, no errors will be reported and two additional folders will be generated in your project, i.e., *Binaries* and *Debug*:



Assuming you have already connected and correctly configured your GigE camera, you can run your project now, i.e., either simply press *Ctrl + F11* or go to the **Run** menu bar and select **Run**:





---

## Revision History

Doc. ID Number	Date	Changes
AW00102501000	29 Jul 2011	Initial release of this document.

