



Product:	WagoIPCLib
Product version:	V1.0
Document ID:	PM-WagoIPCLib
Doc revision:	A
Written/Appr.:	RE / SL
Date:	10/03/2008

## Industrial Control Design AS



# WagoIPCLib V1.0

## Programmers Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, [www.icd.no](http://www.icd.no), [support@icd.no](mailto:support@icd.no), [forum.icd.no](http://forum.icd.no)

# Contents

---

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1. About.....	3
<hr/>	
<b>2. INSTALLATION.....</b>	<b>4</b>
2.1. Prerequisites.....	4
2.2. Quick setup of a project utilizing the WagoIPCLib library.....	4
2.3. Install on a Wago IPC.....	8

# 1. Introduction

This document describes how to set up the WagoIPCLib for programming in a CDP system. The WagoIPCLib has the following component:

- WagoIPCIO – an I/O Server component for communicating with the WAGO K-BUS.

## 1.1. About

- Works under On-Time RTOS32.
- Interrupt-driven PCI communication
- Converts CDP Signals to physical signals
- Converts Physical signals into CDP Signals
- Possibility to Auto-Detect module configuration and update the XML file to reflect the new configuration
- Has Online/Offline State for checking communication-problems.

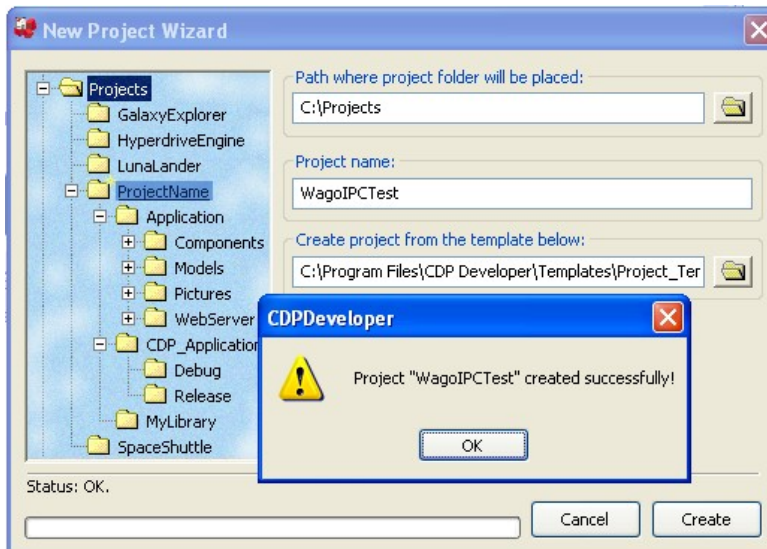
## 2. Installation

### 2.1. Prerequisites

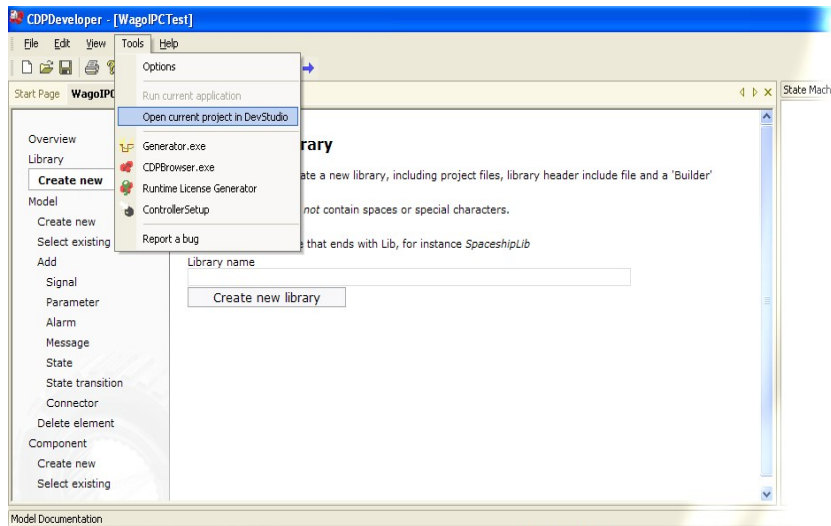
- A valid CDP license
- You are familiar with CDP
- CDP version 2.3.1.0 for OnTime RTOS32, and OnTime RTOS32 version 5.09 is installed
- XMLParser add-on is installed
- Wago IPC (Wago 758-870)

### 2.2. Quick setup of a project utilizing the WagoIPCLib library

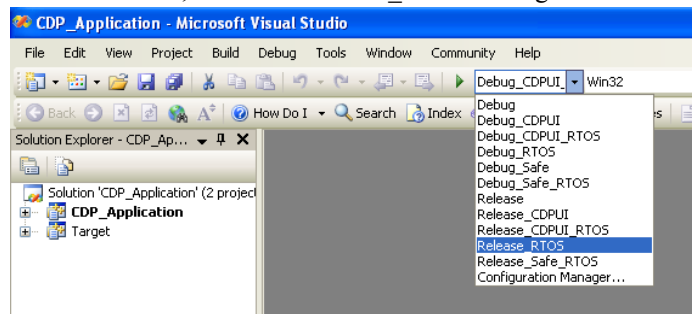
1. Start CDPDeveloper
2. Make a new project by Selecting File->New, type in project name and click Create (or skip to step 4 and open an already existing project in Visual Studio)



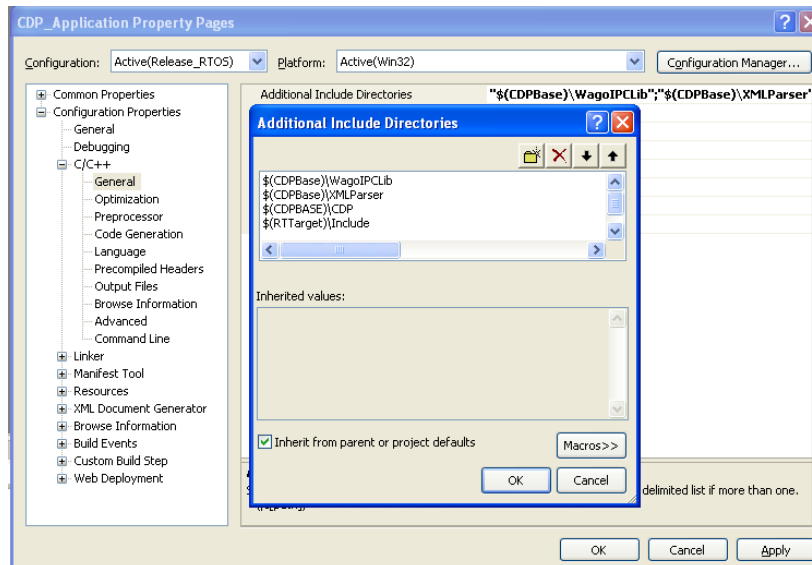
3. Choose 'Open current project in Devstudio' from the Tools menu:



4. If Visual Studio asks to convert the project, accept this by selecting 'Next'/'Finish'/'Close' until done. Close the conversion report.
5. In Visual Studio, select the 'Release\_RTOS' configuration.

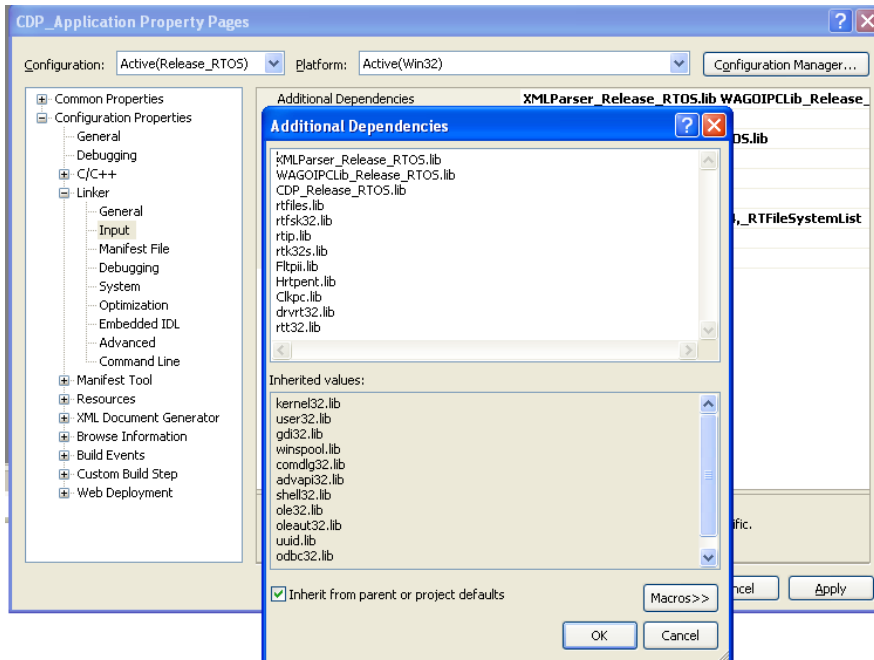


6. Select 'CDP\_Application' from the 'Solution Explorer', right-click and select Properties.
7. In C++/Additional include Directories, make sure it says:  
`"$(CdpBase)\XMLParser";$(CdpBase)\WagoIPCLib";$(CdpBase)\CDP";$(RTTarget)\Include"`



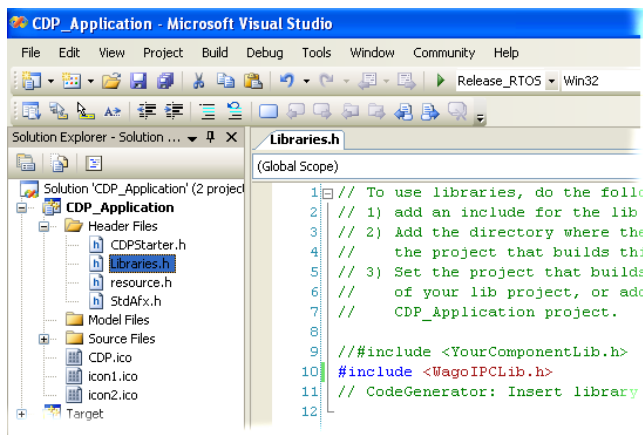
The compiler will look in the directories specified in 'Additional Include Directories' for files that you #include in your .cpp and .h files. If you get an error 'Can not open include file ...' then it is most likely caused by a missing include directory, or that the file you #include does not exist.

- In Linker->Input, make sure you list XMLParser\_Release\_RTOS.lib WAGOIPCLib\_Release\_RTOS.lib CDP\_Release\_RTOS.lib in addition to all the on-time rtos32 libraries you need. This will ensure that the linker finds all the functions that is referenced in the code.



- In the 'Solution Explorer', inside the 'CDP\_Application' project, locate the Header file 'Libraries.h', and add the line

`#include <WagoIPCLib.h>`:



10. In the CDP\_Application project, locate the file CDPStarter.cpp, and in that file, locate the function `bool Handle_PCI_Device(struct CDP_PCIDevice* pDevice)`. Modify it so that it looks like this:

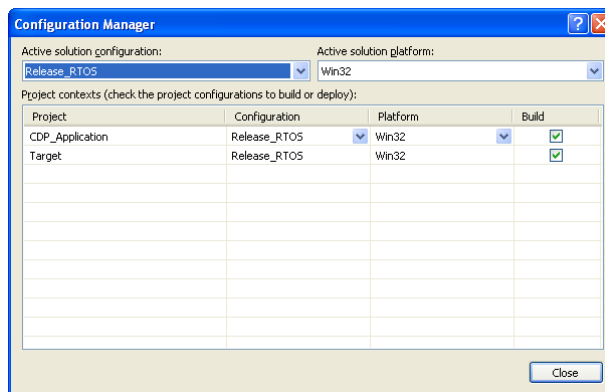
```

/**
 * PCI enumeration callback.
 * This gets called for every PCI Device in the system.
 * Return true if you handle a device, false if not.
 *
 * If you don't handle the device (i.e. return false) ,
 * the cdp system will handle it (Network devices only).
 *
 * Note that you can modify the contents of the pDevice
 * structure (return false to have CDP use your modified values.)
 */
PCIEnumCallback Handle_PCI9030_Device = NULL;

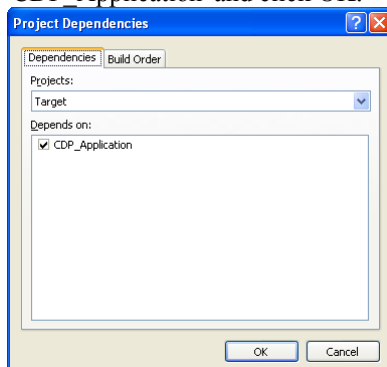
bool Handle_PCI_Device(struct CDP_PCIDevice* pDevice)
{
    // TODO: handle Special PCI Devices:
    if(Handle_PCI9030_Device!=NULL && Handle_PCI9030_Device(pDevice)==true)
        return true;
    return false;
}
    
```

*Note: Take care to add the declaration of the PCIEnumCallback `Handle_PCI9030_Device = NULL;` above. The above code will be called for each PCI device detected by CDP. The `Handle_PCI_9030_Device` code will enable the WagoIPCIO component to communicate correctly with the K-BUS on the Wago IPC.*

11. If there is no 'Target' project in your 'Solution Explorer' Select 'Solution' -> 'Add' ->'Existing Project'.
12. Choose the 'Target.vcproj' from the same folder in which your 'CDP\_Application.vcproj' is at.
13. Right-Click the Target project, select Properties. Under General, make sure 'Configuration Type' says 'Utility'. Click OK.
13. Right-Click Solution, select 'Configuration manager...'. Make sure that all configurations have 'Release\_RTOS' selected in 'Configuration' :



14. Choose 'Solution' -> 'Project Dependencies', choose 'Target' from the pull-down list, check 'CDP\_Application' and click OK.



15. Rebuild the entire solution to generate the CDP.rtb file in the CDP\_Application\Release\_RTOS\ folder.

## 2.3. Install on a Wago IPC

Assuming you have a Wago IPC with a CompactFlash disk, you can use ControllerSetup.exe to download a bare-bone CDP application to the controller. See the document 'ControllerSetup.pdf' in the Doc folder where you installed CDP for more information on setting up a controller.

When the Controller has a CDP Application installed, you can use CDPFileManager to upload files to the controller. Upload these files to the root of the controller disk.

- CDP\_Application\Release\_RTOS\CDP.rtb
- Everything inside the Application folder.

Make sure that you also upload rttboot.com from your \$(RTTarget)\bin folder to the root folder of the controller. This ensures that the RTOS32 loader is the most recent version.